UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Implementação de Consultas para um Modelo de Dados Temporal Orientado a Objetos

por

TANISI PEREIRA DE CARVALHO

Dissertação submetida à avaliação, como requisito
parcial para a obtenção do grau de
Mestre em Ciência da Computação

Profa. Dra. Nina Edelweiss
Orientadora

Porto Alegre, abril de 1997

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Carvalho, Tanisi Pereira de

Implementação de Consultas para um Modelo de Dados Temporal Orientado a Objetos / por Tanisi Pereira de Carvalho. — Porto Alegre: CPGCC da UFRGS, 1997.
127 f.: il.

Dissertação (mestrado) — Universidade Federal do Rio Grande do Sul. Curso de Pós-Graduação em Ciência da Computação, Porto Alegre, BR-RS, 1997. Orientador: Edelweiss, Nina.

1. Banco de dados. 2. Banco de dados Temporal. I.Edelweiss, Nina. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitora: Profa. Wrana Panizzi
Pró-Reitor de Pós-Graduação: Prof. José Carlos Ferraz Hennemann
Diretor do Instituto de Informática: Prof. Roberto Tom Price
Coordenador do CPGCC: Prof. Flávio Rech Wagner
Bibliotecária-Chefe do Instituto de Informática: Zita Prates de Oliveira

Agradecimentos

Ao Prof. Dr. José Palazzo Moreira de Oliveira, pela confiança, incentivo e por não ter deixado que eu desistisse desta etapa. Muito Obrigada!

À Profa. Dra. Nina Edelweiss, por ter aceitado ser minha orientadora, por estar sempre disponível para esclarecer minhas dúvidas e pela excelente orientação.

Ao CNPq pelo financiamento deste trabalho.

Aos funcionários do Instituto de Informática que estão sempre prontos para esclarecer nossas dúvidas e ajudar no que for preciso. Muito Obrigada!

Aos meus amigos que dividiram comigo as alegrias e incertezas desta etapa e que contribuíram muito para a conclusão deste trabalho: Rejane, Sílvia, Simone, Kika, Fabiane, Júnior, Tiarajú e Elton.

A meus pais, Tânia e Gildo, pela dedicação e amor. Por serem tão especiais e estarem sempre presentes em todos os momentos da minha vida.

As minhas irmãs, Gisele e Alessandra, e ao meu cunhado Cláudio, pelos momentos felizes, pela amizade e por saber que sempre posso contar com vocês.

A minha querida vó Olinda, por ser esta pessoa maravilhosa e pela lição de vida que nos dá a cada dia.

Um agradecimento especial ao meu namorado Flávio, pelo carinho, amizade e incentivo.

Sumário

Lista de Abreviaturas **7**
Lista de Figuras **8**
Lista de Tabelas **10**
Resumo **11**
Abstract **11**
1 Introdução **13**
1.1 Motivação **13**
1.2 Apresentação do trabalho **15**
1.3 Organização do texto **16**
2 Linguagens de Consulta **17**
2.1 Linguagem Textual de Consulta **17**
2.2 Linguagem Visual de Consulta **19**
2.2.1 Linguagem Baseada em Formulários 20
2.2.1.1 Linguagem QBE 20
2.2.1.2 Linguagem TableTalk 22
2.2.1.3 Linguagem GOODIES 22
2.2.2 Linguagem Gráfica 23
2.2.3 Linguagem de Ícones 25
2.2.4 Linguagem Híbrida ou Visual 26
2.2.5 Comparação entre Linguagens de Consulta 27
2.3 Recuperação de Informações em Banco de Dados Temporais **28**
2.3.1 VQS para Modelos Temporais 29
2.3.1.2 Linguagem de Consulta GL_TEER 30
2.3.1.2 Linguagem de Consulta GTL 31
2.3.2. Requisitos dos Sistemas de Consulta Visual para Modelos Temporais 32
3 O Modelo TF-ORM **34**
3.1 O Conceito de Papéis **34**
3.2 Características Gerais **34**
3.3 A Linguagem de Consulta do Modelo TF-ORM **36**
3.3.1 Cláusula de Especificação 37
3.3.2 Cláusula de Identificação 38
3.3.3 Cláusula de Busca 39
3.3.4 Cláusula de Instante Temporal 41
3.3.5 Predicados, Funções e Operadores Temporais 41

3.3.6 Extensões da Linguagem de Consulta 43
4 Estudo de Caso 45
5 Descrição Geral do Ambiente 49
5.1 Modelos Internos e Externos para Dados e Consulta 50
5.2 Estrutura do Ambiente 51
6 O Sistema Visual de Consulta do TF-ORM 53
6.1 Esquema Conceitual Gráfico 53
6.2 Esquema de Trabalho 55
6.3 Consulta Gráfica 56
6.3.1 Seleção do Esquema Gráfico 58
6.3.2 Janela de Acessórios 59
6.3.3 Definição de Condições 60
6.3.3.1 Janela de Definição de Condições 61
6.3.3.2 Definição de Condições com Propriedades Pré-Definidas 62
6.3.4 Janela de Funções e Predicados 63
6.3.5 Definição de Restrições Temporais 64
6.3.5.1 Janela de Tempo da Consulta 65
6.3.5.2 Janela de Operadores Temporais 66
6.3.5.3 Representação Visual para as Restrições Temporais 69
6.3.6 Exemplos de Consulta e a Representação Gráfica 71
6.3.6.1 Consulta com Restrições Simples e Saída de Dados 72
6.3.6.2 Consulta com Operadores Temporais 73
6.3.6.3 Consulta com Saída Temporal 74
6.3.6.4 Consulta com Predicados 75
6.4 Consulta por Formulários 75
6.6 Representação Textual e Gráfica da Consulta 78
6.6 Apresentação dos Resultados 80
7 Mapeamento entre Modelos 83
7.1 Mapeamento entre Modelos de Dados 83
7.1.1 Mapeamento do Modelo TF-ORM para o Banco de Dados Relacional 83
7.1.2 Esquema Conceitual Relacional e ECGr 87
7.2 Mapeamento entre Modelos de Consulta 88
7.2.1 Mapeamento da Consulta Gráfica para a Consulta Textual do TF-ORM 88
7.2.2 Mapeamento da Consulta Textual para a Consulta Gráfica 91
7.2.3 Mapeamento da Consulta Textual do TF-ORM para SQL 92
7.2.3.1 Definição das Tabelas, Relacionamentos e Restrições Sobre Dados 92
7.2.3.2 Definição das Restrições de Tempo 94
7.2.3.3 Exemplos de Consultas 99
8 Conclusões e Trabalhos Futuros 104
8.1 Revisão do Trabalho 104
8.2 Principais Resultados 105
8.3 Extensões e Trabalhos Futuros 106

Anexo 1 Sintaxe da Linguagem de Consulta do Modelo TF-ORM 108
Anexo 2 Esquema Conceitual 112
Anexo 3 Mapeamento de Predicados e Funções 117
Bibliografia 121

Lista de Abreviaturas

BD	Banco de Dados
ECGr	Esquema Conceitual Gráfico
ECGrT	Esquema Conceitual Gráfico de Trabalho
SGBD	Sistema Gerenciador de Banco de Dados
SQL	Structured Query Language
TF-ORM	Temporal Functionality in Objects with Roles Model
VQS	Visual Query Systems

Lista de Figuras

FIGURA 1 - Formas de Apresentação do Resultado 15
FIGURA 2.1 - Consulta Textual 19
FIGURA 2.2 - Consulta QBE Envolvendo Condições Simples 21
FIGURA 2.3 - Consulta QBE Envolvendo Condições Complexas 21
FIGURA 2.4 - Exemplo de Consulta na Linguagem TableTalk 22
FIGURA 2.5 - Consulta Gráfica 23
FIGURA 2.6 - Exemplo de Consulta no Ambiente SUPER 24
FIGURA 2.7 - Consulta Baseada em Ícones 25
FIGURA 2.8 - Exemplo de Consulta na Linguagem IQL 26
FIGURA 2.9 - Esquema Gráfico do Sistema VILD 30
FIGURA 2.10 - Exemplo de Consulta na Linguagem GL_TEER 31
FIGURA 2.11 - Exemplo de Consulta na Linguagem GTL 32
FIGURA 3.1 - Tipos de Saída para Consulta 38
FIGURA 5.1 - Modelo de Dados e Linguagem de Consulta do Ambiente 50
FIGURA 5.2 - Modelo de Dados e Modelo de Consulta 51
FIGURA 5.3 - Estrutura do Ambiente para Recuperação de Informações 52
FIGURA 6.1 - Interface do VQS TF-ORM 55
FIGURA 6.2 - Definição do Esquema de Trabalho 56
FIGURA 6.3 - Janela para Salvar o Esquema de Trabalho 56
FIGURA 6.4 - Janelas para Consulta Gráfica 57
FIGURA 6.5 - Janela para Abrir Esquema de Trabalho 58
FIGURA 6.6 - Interface para Consulta Gráfica 59
FIGURA 6.7 - Definição de Condição sobre o Esquema Gráfico 60
FIGURA 6.8 - Janela de Definição de Condições 61
FIGURA 6.9 - Definição de Restrições sobre Prop. Pré-Definidas 63
FIGURA 6.10 - Janela de Predicados 64
FIGURA 6.11 - Janela de Funções 64
FIGURA 6.12 - Janela de Tempo da Consulta 65
FIGURA 6.13 - Janela de Operadores Temporais 67
FIGURA 6.14 - Exemplo de Condição Associada a um Operador Temporal 67
FIGURA 6.15 - Operador Temporal sem Argumento 68
FIGURA 6.16 - Janela de Mensagem 69
FIGURA 6.17 - Linha do Tempo 69
FIGURA 6.18 - Representação dos Operadores Temporais do Passado 70
FIGURA 6.19 - Representação dos Operadores Temporais do Futuro 70
FIGURA 6.20 - Representação dos Operadores com dois Argumentos 71
FIGURA 6.21 - Esquema da Consulta com Restrições Simples 73
FIGURA 6.22 - Esquema da Consulta com Operadores Temporais 74
FIGURA 6.23 - Esquema da Consulta com Saída Temporal 75
FIGURA 6.24 - Esquema da Consulta com Predicados e Funções 75
FIGURA 6.25 - Formulário de Consulta para Funcionário 76

FIGURA 6.26 - Apresentação dos Estados de um Objeto 77
FIGURA 6.27 - Formulário de Consulta para Departamento 77
FIGURA 6.28 - Definição de Restrição Temporal 78
FIGURA 6.29 - Janela de Consulta Textual 79
FIGURA 6.30 - Janela de Esquema da Consulta 80
FIGURA 6.31 - Apresentação do Resultado na Forma de Formulários 81
FIGURA 6.32 - Apresentação do Resultado na forma de Tabela 81
FIGURA 6.33 - Apresentação do Resultado na Forma Gráfica 82
FIGURA 6.34 - Apresentação do Resultado na Forma Visual 82
FIGURA 7.1 - Mapeamento da cláusula de Especificação 89
FIGURA 7.2 - Mapeamento da Cláusula de Identificação 90
FIGURA 7.3 - Mapeamento da Cláusula de Busca 91
FIGURA 7.4 - Recuperação dos Dados Atuais 95
FIGURA 7.5 - Recuperação de Estados Válidos 97
FIGURA 7.6 - Validade dos Atributos e o Operador Temporal Sometime Past 97
FIGURA 7.7 - Validade dos Atributos e o Operador Temporal Since 98
FIGURA 7.8 - Atributos Válidos para os Operadores Before e After 99
FIGURA 7.9 - Recuperação de Dados Instantâneos Atuais 100
FIGURA 7.10 - Valor dos Atributos e Tempo de Validade 100
FIGURA 7.11 - Recuperação de Valores em uma Determinada Data 101
FIGURA 7.12 - Recuperação de Dados Históricos - Saída Temporal 102
FIGURA 7.13 - Recuperação de Dados Históricos - Seleção Mista 102
FIGURA 7.14 - Recuperação de Dados Históricos de uma História Passada 103

Lista de Tabelas

TABELA 2.1 - Comparação das Operações Básicas em Linguagens de Consulta 27
TABELA 3.1 - Valores de Propriedades e sua Relação Temporal 39
TABELA 3.2 - Operadores Temporais 41
TABELA 6.1 - Primitivas Gráficas do ECGr 54

Resumo

O modelo TF-ORM (*Temporal Functionality in Objects With Roles Model*) é um modelo de dados temporal orientado a objetos que utiliza o conceito de papéis para representar os diferentes comportamentos dos objetos. O modelo permite a modelagem dos aspectos estáticos e dinâmicos da aplicação pois considera todos os estados dos objetos ao longo de sua evolução. Sua linguagem de consulta é baseada na linguagem SQL e possibilita a recuperação de diferentes histórias do banco de dados.

Este trabalho apresenta um sistema visual de consulta para o modelo TF-ORM. O VQS TF-ORM (Visual Query System TF-ORM) é um ambiente para recuperação de informações temporais. O sistema permite que as consultas sejam elaboradas de três formas alternativas: textual, gráfica ou por formulários. A linguagem gráfica possui o mesmo poder de expressão da linguagem textual, permitindo que a consulta seja elaborada diretamente sobre o esquema conceitual gráfico do modelo com o auxílio de um conjunto de janelas e elementos visuais. A recuperação de informações utilizando-se formulários não possui o mesmo poder de expressão da linguagem textual, mas possibilita a recuperação dos valores das propriedades de um determinado objeto através de uma hierarquia de janelas.

A recuperação de informações através do sistema visual de consulta do modelo apresenta algumas facilidades tais como: representação visual dos operadores temporais do modelo, definição de níveis de detalhe e navegação sobre o esquema gráfico, armazenamento das consultas para posterior utilização, possibilidade de representar uma consulta textual na forma visual e vice-versa, entre outras. Além da preocupação com a definição de restrições temporais, o ambiente considera ainda as diferentes formas de apresentação do resultado da consulta que podem ser selecionadas pelo usuário.

No sistema apresentado neste trabalho, o modelo TF-ORM é implementado em um banco de dados relacional que utiliza a linguagem SQL para recuperação de informações. Para a implementação do modelo em um banco de dados relacional foi feito um mapeamento, que determina como os conceitos de orientação a objetos, papel e tempo devem ser mapeados para tabelas e atributos no modelo relacional. As consultas realizadas na linguagem TF-ORM são então traduzidas para a linguagem de consulta do banco de dados relacional.

O ambiente foi implementado utilizando a ferramenta para desenvolvimento de aplicações Delphi e o banco de dados Watcom, um banco de dados relacional que permite a recuperação de informações no padrão SQL/ANSI.

Palavras-chave: Banco de dados, Recuperação de Informações, Linguagem Visual de Consulta, Modelo Temporal.

Title: "Implementation of Queries for a Temporal Object Data Model "

Abstract

TF-ORM model (Temporal Functionality in Objects with Roles Model) is an object-oriented temporal data model which uses the role concept to represent different behaviors of objects. The model allows modelling of the static and the dynamic aspects of an application representing all the states of its evolution. The TF-ORM query language is based on the SQL language and enables the recovery of different database histories.

This work represents a visual query system for the TF-ORM model. The VQS TF-ORM (Visual Query System TF-ORM) is an environment for recovery of temporal information. The system allows queries to be elaborated in three alternatives way: textual, graphic or by forms. The graphic language has the same functionality of the textual language permitting the query to be elaborated directly on the graphic conceptual schema of the model this operation is supported by a set of windows and visual elements. The information recovery using forms doesn't have the same functionality of the textual language, but enables recovery of property values of an object through window hierarchies.

Information recovery using the visual query system of the model presents some facilities: the visual representation of temporal operators, different levels of details for the navigation on the graphic schema, query storage for later use, possibility of representing a textual query in a visual way and vice-versa. The environment supports the definition of temporal constraints and the selection by the user of different representations forms for the results of a query.

In the presented system, the TF-ORM model is implemented in a relational database which uses SQL language for information recovery. In order to implement the model in a relational database, a mapping was done - the concepts of the object orientation, roles and time were mapped in to tables and attributes to the relational model. The queries performed in the TF-ORM language are translated into the query language of relational database.

The environment was implemented using Delphi and the Watcom database, a relational database which allows information recovery in SQL/ANSI standard.

Keywords: Database, Information Recovery, Visual Query Language, Temporal Model.

1 Introdução

1.1 Motivação

Um modelo de dados temporal pode ser utilizado para especificar tanto os aspectos estáticos como os aspectos dinâmicos de uma aplicação. Bancos de dados temporais permitem armazenar todos os estados de um objeto, registrando sua evolução com o tempo [CLI 95, TAN 93]. Para tal, informações temporais são associadas aos dados (tempo de transação e/ou tempo de validade) [JEN 92]. Dependendo do tipo de informação temporal considerada os bancos de dados podem ser classificados em: (i) banco de dados instantâneo - cada modificação provoca uma transição no banco de dados, sendo que o valor anteriormente armazenado é perdido; (ii) banco de dados de tempo de transação - um rótulo temporal (*timestamp*), representando o tempo de transação, é associado a cada informação armazenada; (iii) banco de dados de tempo de validade - utiliza o tempo de validade para representar o período em que uma informação é valida no mundo real; (iv) banco de dados bitemporal - associa a cada informação tanto o tempo de transação como o tempo de validade. A linguagem de consulta de um banco de dados (BD) temporal deve possibilitar a recuperação de diferentes estados de um objeto de acordo com o instante de tempo desejado.

A utilização de um modelo de dados temporal para especificação de uma aplicação não implica, necessariamente, na utilização de um SGBD específico para o modelo. Bancos de dados comerciais podem ser utilizados se existir um mapeamento adequado entre o modelo temporal e o BD utilizado. Este enfoque está sendo adotado para o modelo de dados temporal orientado a objetos TF-ORM (*Temporal Functionality in Objects With Roles Model*) [EDE 93,94a]. Em [OLI 95] são apresentadas algumas alternativas de mapeamento do modelo TF-ORM para os bancos de dados comerciais O_2, Postgres e Ingres. Todos os valores assumidos pelos objetos durante sua existência são armazenados no BD como propriedades associadas a informações de tempo. Estes ambientes possuem dois tipos de modelos de dados: um modelo de dados externo e um modelo de dados interno. O modelo de dados externo corresponde ao modelo utilizado para expressar a semântica dos dados e o modelo de dados interno corresponde à forma de implementação.

A recuperação de informações temporais utilizando a linguagem de consulta do BD exigiria do usuário um grande conhecimento sobre a forma de mapeamento do modelo para o banco de dados e do relacionamento dos atributos temporais com os valores das propriedades. A utilização de um banco de dados diferente para armazenar as informações do modelo não deve eliminar a possibilidade de utilização da linguagem de consulta do modelo. Estas linguagens permitem a definição de consultas temporais em um alto nível sem que o usuário tenha que se preocupar com

os aspectos de implementação. Além da possibilidade de definir restrições sobre os dados envolvidos na consulta, o usuário pode definir faixas de tempo sobre as quais os dados devem ser considerados e estabelecer comparações entre os valores dos objetos em um determinado instante no tempo.

Ambientes que possuem dois modelos de dados diferentes, geralmente utilizam dois modelos de consultas (um externo e outro interno). O modelo de consultas externo corresponde à linguagem de consulta utilizada pelo usuário para recuperação de informações. A consulta elaborada pelo usuário deverá ser mapeada para o modelo de consulta interno (linguagem de consulta do banco de dados) para que possa ser executada no banco de dados utilizado. O modelo de consulta externo corresponde à linguagem de consulta que será utilizada pelo usuário.

Existem duas categorias de linguagens de consulta: linguagem textual de consulta e linguagem visual de consulta. A recuperação de informações através de uma linguagem textual de consulta requer que o usuário tenha um grande conhecimento da sintaxe da linguagem e do esquema do banco de dados. Este conhecimento inclui também, o nome e o domínio das informações armazenadas no banco dados e a forma pela qual estão relacionadas. A utilização de uma linguagem visual de consulta elimina os problemas das linguagens textuais oferecendo um ambiente mais amigável para o usuário [CAT 95]. O termo linguagem visual de consulta denota um caráter mais abrangente de interação, sendo a comunicação com o usuário feita através de símbolos que carregam a semântica das operações [MEL 94]. O sucesso desta categoria de linguagens de consulta pode ser definido por três características [YEN 93]:

- a linguagem de consulta é de fácil aprendizado - linguagens de consulta visuais apresentam um conjunto de símbolos visuais ou gráficos para representar as informações armazenadas no banco de dados e os principais elementos da linguagem de consulta textual. Através da interação direta com estes elementos visuais o usuário pode realizar a consulta;
- a linguagem de consulta é fácil de entender - o símbolo visual carrega a semântica da operação ou do elemento que representa, facilitando o entendimento por usuários casuais;
- a linguagem de consulta é amigável - as características apresentadas no primeiro e no segundo itens tornam a linguagem de consulta bastante amigável.

Outra aspecto importante no processo de recuperação de informações é a forma de apresentação do resultado da consulta. O resultado da consulta pode ser apresentado de diferentes maneiras. Em algumas situações, por exemplo, a apresentação de um gráfico é muito mais significativo do que apresentar uma tabela. Um exemplo pode ser encontrado em [CRU 92] (fig. 1) onde os horários de partida e chegada de dois vôos são apresentados de forma diferente. A figura 1b apresenta um significado maior do que a figura 1a, pois na segunda forma de representação o usuário pode obter a informação de relacionamento que não aparece de forma tão explícita no primeiro caso. Na segunda forma de representação fica fácil identificar que o primeiro vôo sai antes que o segundo, mas chega depois. Esta comparação entre os horários de partida e chegada não aparece de forma explícita no primeiro caso. As duas formas de representação apresentam a mesma informação, mas a segunda

apresenta o resultado de forma mais clara a medida que considera a característica temporal do resultado.

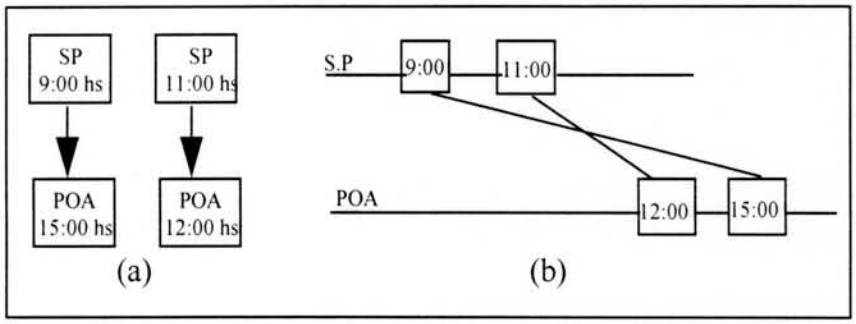

FIGURA 1 - Formas de Apresentação do Resultado

1.2 Apresentação do trabalho

Este trabalho apresenta um ambiente para recuperação de informações do modelo TF-ORM (*Temporal Functionality in Objects With Roles Model*) [EDE 93, 94a]. O modelo TF-ORM foi escolhido por ser um modelo temporal semanticamente completo e por apresentar uma linguagem de consulta bastante poderosa e baseada na linguagem SQL. Sua linguagem de consulta permite a recuperação de diferentes histórias dos objetos, pois o TF-ORM é um modelo de dados bitemporal (associa tempo de transação e tempo de validade as propriedades que mudam de valor com o tempo).

O ambiente trabalha com dois modelos de dados e dois modelos de consulta. O modelo de dados externo é representado pelo modelo TF-ORM e o modelo de dados interno é representado pelo modelo relacional. Para utilizar o modelo TF-ORM com um banco de dados relacional, foi utilizado o mapeamento apresentado em [CAV 95]. Devido às vantagens apresentadas pelas linguagens visuais de consulta, foi desenvolvida neste trabalho uma linguagem visual de consulta específica para o modelo TF-ORM. A interação pode ser feita através da linguagem visual de consulta ou, opcionalmente, através da linguagem textual de consulta do modelo.

A consulta na linguagem textual ou visual é posteriormente traduzida para a linguagem SQL para que possa ser executada no banco de dados relacional. A tradução da consulta é feita em dois níveis. Primeiro, a consulta na linguagem visual é traduzida para a linguagem textual. Nesta etapa o sistema utiliza um conjunto de regras que determinam como as ações realizadas pelo usuário sobre os elementos da linguagem visual podem ser traduzidos para as cláusulas da linguagem textual. Em uma segunda etapa, a consulta na linguagem textual TF-ORM é traduzida para a linguagem SQL. O mapeamento desta fase traduz as classes, subclasses, papéis e propriedades do esquema conceitual do modelo TF-ORM para as tabelas do banco de dados relacional. Além disso, utiliza um conjunto de regras para determinar como as restrições temporais, definidas com as cláusulas especiais da linguagem TF-ORM, devem ser mapeadas para restrições sobre os atributos do banco de dados relacional.

Para a implementação do ambiente foi escolhida a ferramenta para desenvolvimento de aplicações Delphi [BOR 95]. Esta ferramenta apresenta um ambiente bastante amigável para o desenvolvimento de interfaces e possibilita a comunicação via ODBC (Open DataBase Connectivity) com muitos bancos de dados. O banco de dados relacional escolhido foi o banco de dados Watcom. A escolha se deve a dois motivos: (i) as consultas são realizadas no padrão SQL/ANSI [AME 92], permitindo que o sistema possa ser utilizado com qualquer outro banco de dados que utilize a linguagem de consulta SQL; e (ii) o ODBC é bastante flexível - não possui limitação quanto ao tamanho da consulta e possibilita consulta a visões. Estas características não são encontradas em todos os ODBCs para banco de dados.

1.3 Organização do texto

Esta dissertação está organizada em oito capítulos. No segundo capítulo são apresentados os sistemas de consulta textual e visual, suas características, vantagens e desvantagens. O capítulo três descreve o modelo TF-ORM através de suas características gerais, dando ênfase à linguagem de consulta que é bastante importante para este trabalho. No capítulo quatro é apresentado o estudo de caso que será utilizado com exemplo. No capítulo cinco é realizada uma descrição geral do ambiente para recuperação de informações do modelo TF-ORM. No capítulo seis é apresentado o sistema visual de consulta do modelo TF-ORM. No sétimo capítulo são apresentados os mapeamentos entre os modelos de dados e os modelos de consulta e no oitavo capítulo são apresentadas as conclusões e os trabalhos futuros.

2 Linguagens de Consulta

O sucesso de um sistema de banco de dados está bastante relacionado com a facilidade de recuperação das informações armazenadas. Através da linguagem de consulta do banco de dados o usuário deve ser capaz de expressar um conjunto de restrições e selecionar as informações que desejar. Esta linguagem de consulta deve possuir um grande poder expressão sem ser muito complexa.

A linguagem de consulta pode ser entendida como um conjunto formalmente bem definido de operadores que podem ser combinados para expressar consultas em banco de dados. O processo de formulação da consulta é realizado em três etapas [CAT 93]:

- localização - o usuário seleciona o conjunto de informações de seu interesse, ou seja, os objetos que farão parte do seu conjunto de resposta ou que serão utilizados para restringir o conjunto de valores da resposta;
- definição de requisitos - sobre os objetos selecionados na fase anterior, são aplicadas algumas restrições que determinam as condições a serem satisfeitas para que os mesmos possam fazer parte do resultado da consulta;
- visualização do resultado - o resultado da consulta é apresentado ao usuário. Um mesmo conjunto de informações pode ser apresentado de diferentes formas, mas a escolha da representação adequada possibilita uma melhor interpretação dos resultados por parte do usuário.

A consulta pode ser realizada através de uma linguagem textual ou de uma linguagem visual. A linguagem textual de consulta exige que o usuário tenha um grande conhecimento da sintaxe e da forma como as informações estão organizadas no banco de dados. A linguagem visual de consulta apresenta um ambiente mais amigável para recuperação de informações permitindo que usuários não familiarizados com a sintaxe da linguagem de consulta textual formulem suas consultas de forma simples e intuitiva.

2.1 Linguagem Textual de Consulta

Uma linguagem textual de consulta permite a formulação da consulta através de uma expressão textual que possui uma sintaxe e um conjunto de regras bem definidas. Uma das linguagens textuais de consulta para banco de dados relacionais mais conhecida é a linguagem SQL (*Structured Query Language*) [DAT 87]. Uma consulta na linguagem SQL é expressa através de um conjunto de palavras chaves e referências a elementos do esquema do banco de dados. A estes elementos podem ser aplicadas restrições de acordo com a sintaxe da linguagem de consulta e os operadores fornecidos pela mesma. Como o relacionamento entre os objetos no

modelo relacional é feito através da replicação de chaves, o usuário precisa comparar explicitamente estas chaves para relacionar os elementos envolvidos na consulta. A estrutura básica da linguagem SQL é mostrada a seguir:

SELECT <cláusula de especificação>
FROM <cláusula de identificação>
WHERE<cláusula de busca>

A cláusula de especificação determina os objetos de saída da consulta, ou seja, os elementos do esquema que devem ser apresentados no resultado da consulta. A cláusula de identificação contém o conjunto de tabelas utilizadas na consulta e a cláusula de busca apresenta os relacionamentos entre as tabelas e as restrições aplicadas aos atributos. Uma consulta na linguagem SQL pode construída de diferentes maneiras, pois o usuário determina quais elementos devem participar da consulta e as restrições sobre os mesmos, sem definir a forma como a consulta será executada.

Para exemplificar a utilização da linguagem textual será utilizado o esquema conceitual simplificado para uma agência bancária com as seguintes tabelas: (i) tabela *cliente* - armazena as informações relacionadas aos clientes com os atributos *cod_cliente* (código do cliente), *tipo* (tipo do cliente que pode ser uma pessoa ou empresa, por exemplo), *nome* (nome do cliente) e *endereço*; (ii) tabela *conta* - contém os atributos *num_conta*, *cod_cliente* e *saldo*; (iii) tabela *transação* - contém as transações realizadas pelos clientes do banco e apresenta os seguintes atributos: *cod_transação* (código da transação realizada saque ou depósito, por exemplo), *num_conta* (número da conta a qual a transação esta associada) e *quantia* (valor em reais da transação).

Considere a seguinte consulta: "Obter o nome de todas as pessoas que fizeram uma transação bancária em 23/04/96". Uma construção equivalente para esta consulta na linguagem SQL, seria colocar todos as tabelas envolvidas na cláusula **FROM** (*cliente*, *conta* e *transação*), o atributo nome na cláusula **SELECT** e as restrições e associações entre tabelas na cláusula **WHERE**. A representação desta consulta na linguagem SQL é apresentada na figura 2.1a.

Uma consulta equivalente poderia ser construída utilizando-se subconsultas. O operador **IN**, é um dos operadores da linguagem SQL que pode ser utilizado para relacionar um atributo a uma subconsulta, indicando que o valor do atributo deve estar presente no resultado da subconsulta. Por exemplo, se o cliente realizou uma transação, o número de sua conta deve estar presente na tabela que armazena as transações executadas (tabela *transação*). A figura 2.1b apresenta a consulta anterior utilizando-se subconsultas e o operador **IN**.

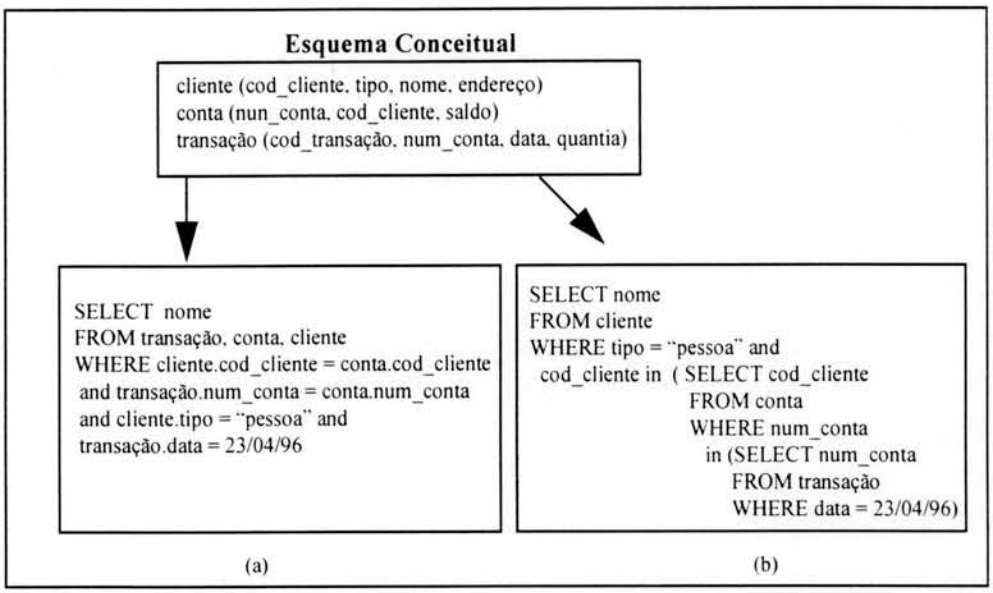

FIGURA 2.1 - Consulta Textual

2.2 Linguagem Visual de Consulta

Os sistemas visuais de consulta (*Visual Query Systems* - VQS) utilizam uma linguagem para expressar as consultas em um formato visual e uma variedade de funcionalidades para facilitar a interação do usuário com o sistema [CAT 94]. Os VQS podem ser vistos como uma extensão das linguagens de consulta para SGBD (Sistema Gerenciadores de Bancos de Dados), permitindo que as pesquisas ao banco de dados sejam feitas por usuários menos experientes.

Um VQS podem ser dividido em duas partes: ambiente para interação com o usuário e ambiente de implementação. A primeira define a forma pela qual o usuário irá visualizar e manipular as informações do esquema e a segunda determina como estas informações serão armazenadas e manipuladas internamente. O ambiente para interação com o usuário deve apresentar um modelo de dados com um grande poder de expressão e uma linguagem de consulta bastante amigável. Já o ambiente de implementação depende do banco de dados utilizado e da linguagem de consulta do mesmo. A possibilidade de utilização de modelos diferentes permite que o ambiente para interação com o usuário seja definido sem considerar os aspectos de implementação.

VQSs apresentam portanto dois modelos de dados para representar dados e consultas [CAT 95]: modelo externo (utilizado pelo ambiente de interação com o usuário) e modelo interno (utilizado pelo ambiente de implementação). O modelo de dados externo é formado por um conjunto de abstrações para expressar a semântica dos dados, e o modelo de dados interno é determinado pelo ambiente de implementação. Por exemplo, o modelo E-R (Entidade-Relacionamento) pode ser utilizado como modelo externo e o modelo relacional como modelo interno. As informações do esquema conceitual são apresentadas ao usuário através do esquema gráfico do modelo E-R, sendo que as informações armazenadas no banco de dados

relacional devem ser mapeadas para o mesmo. De forma similar, o modelo de consulta externo é representado pela linguagem de consulta utilizada pelo usuário e o modelo de consulta interno é manipulado através da linguagem de consulta definida pelo ambiente de implementação. Considerando o exemplo anterior, o modelo de consulta interno utiliza a linguagem de consulta SQL e modelo de consulta externo, a linguagem gráfica definida para o ambiente. O usuário elabora sua consulta utilizando a linguagem gráfica e esta consulta é mapeada para a linguagem SQL para que possa ser executada no banco de dados. Para permitir a tradução entre os modelos interno e externo deve existir um módulo de mapeamento que faça o relacionamento entre as operações e representações de cada um dos modelos.

Os VQSs podem ser classificados em mais de uma categoria de acordo com os elementos visuais definidos para elaboração da consulta e a forma de interação com os mesmos [BAT 96, CAT 96].

2.2.1 Linguagem Baseada em Formulários

Um formulário pode ser visto como uma grade retangular cujos componentes podem ser combinações de células e/ou grupos de células (subformulário). O formulário é uma generalização da tabela, pois os componentes das tabelas são geralmente células elementares não sendo permitido o agrupamento das mesmas. No formulário, a célula é a menor unidade de dado que pode ser referenciada. A representação dos relacionamentos entre formulários pode ser feita através de uma célula ou de um conjunto de células. A seguir serão apresentadas alguns exemplos de linguagens de consulta baseadas em formulários.

2.2.1.1 Linguagem QBE

A linguagem QBE (*Query-by-Example*) [ZLO 77] foi o primeiro sistema a adotar o paradigma tabular, explorando o espaço bidimencional e proporcionando ao usuário uma interface de fácil manipulação. O processo de elaboração da consulta começa com a seleção de uma ou mais tabelas em branco. A seguir é inserido o nome da tabela na primeira coluna e o sistema apresenta todos os campos da mesma. As restrições sobre os atributos das tabelas devem ser definidas na própria grade de acordo com um conjunto de regras da linguagem. Para exemplificar a linguagem QBE considere as seguintes consultas baseadas no esquema conceitual apresentado na seção 2.1. Sobre este esquema podem ser realizadas as seguintes consultas:

- recuperação simples com restrições - considere a consulta: "Selecionar o código do cliente de conta número 1234 e que possui um saldo superior a 3000 reais". Esta consulta é apresentada na figura 2.2a. A utilização da letra P seguida de um ponto (P.) em qualquer coluna da tabela indica que os valores da mesma devem ser apresentados no resultado da consulta, qualquer constante colocada em uma das colunas da tabela indica uma restrição aplicada a consulta (por exemplo, num_conta = 1234). A condição salário superior a 3000 é especificada através do operador > (maior que) seguido do valor na coluna apropriada. A constante 100 sublinhada indica que este valor pode ser considerado na consulta mas não é obrigatório;

- associação entre elementos da mesma tabela - consulta: "Encontrar o código de todos os clientes que têm saldo maior que o cliente de código 100". Na figura 2.2b, o saldo do cliente de código 100 é representado pela constante S1. A restrição de que os clientes devem ter saldo maior que o cliente de código 100, pode ser expressa por saldo maior que S1, já que a constante S1 representa o saldo do cliente de código 100. A ordem das linhas na tabela não é importante, sendo assim o usuário não é obrigado a estruturar a consulta de uma forma específica;

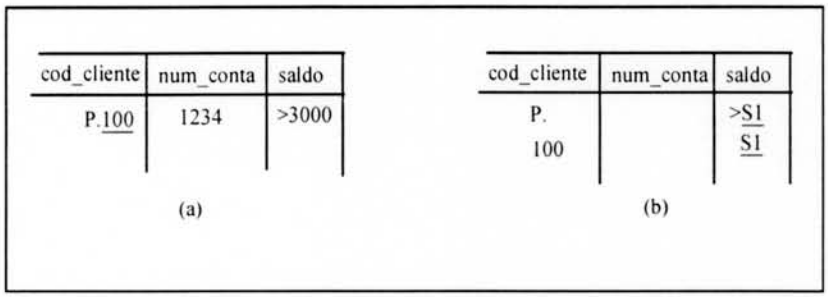

FIGURA 2.2 - Consulta QBE Envolvendo Condições Simples

- recuperação usando negação - consulta: "Encontrar o código dos clientes que não efetuaram nenhuma transação de código 201". A consulta expressa na linguagem QBE é apresentada na figura 2.3a. O operador *not* (¬) é aplicado a entrada da consulta na tabela transação indicando que não deve existir uma ocorrência deste tipo associada ao cliente da tabela de contas;
- recuperação utilizando os operadores **AND** e **OR** - a definição de restrições aos elementos da consulta através dos operadores **AND** e **OR** pode ser feita de duas formas. Na primeira os operadores são utilizados de forma implícita diretamente na grade de consulta, na segunda a expressão que define o predicado da consulta é colocada no *condition box*. Considere a seguinte consulta: "Encontrar o número da conta dos clientes que tem saldo maior que 1000 reais e menor que 1500 ou maior que 3000 reais" (figura 2.3(b)). Os operadores **AND** e **OR** são representados respectivamente pelos símbolos & e |.

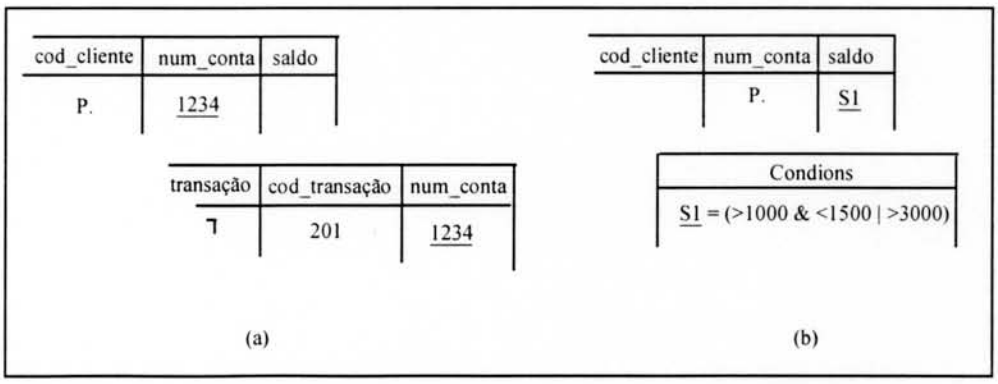

FIGURA 2.3 - Consulta QBE Envolvendo Condições Complexas

Além das consultas apresentadas anteriormente, a linguagem permite ainda a utilização de funções de agregação, operador **ALL**, operações de inserção, atualização, deleção, entre outras.

2.2.1.2 Linguagem TableTalk

Outra linguagem de consulta baseada em formulários é a linguagem TableTalk [EPS 90,91]. Esta linguagem utiliza a metáfora de tabela para expressar a semântica de sua linguagem de consulta e um conjunto de elementos visuais para definir funções e operações. O produto cartesiano, por exemplo pode ser definido através da justaposição horizontal de linhas do formulário contendo o nome das tabelas. As subconsultas podem ser expressadas através de formulários embutidos dentro de outros formulários. TableTalk é uma linguagem de processamento seqüencial. Toda consulta formulada nesta linguagem pode ser vista através de uma seqüência de linhas. Cada linha é a transformação de uma seqüência de entrada em uma seqüência de saída. Considere a seguinte consulta: "Encontrar o código, nome e o saldo do cliente do tipo pessoa que possui o maior saldo dentre todos os clientes". A consulta expressa na linguagem TableTalk é apresentada na figura 2.4. As linhas do formulários que estão justapostas através da conecção vertical **AND** indicam que as duas condições devem ser satisfeitas. A subconsulta é apresentada através de um subformulário associado ao elemento saldo. Os elementos que fazem parte da saída da consulta são apresentados na última linha do formulário.

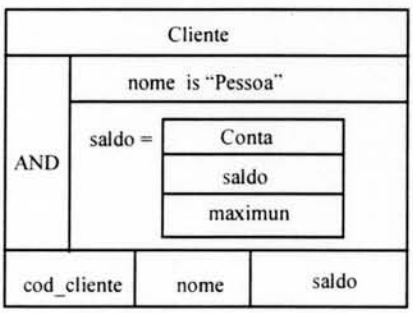

FIGURA 2.4 - Exemplo de Consulta na Linguagem TableTalk

2.2.1.3 Linguagem GOODIES

GOODIES [OLI 93, 93a] é uma interface de consulta para banco de dados orientados a objetos. Pode ser classificada nesta categoria de linguagens de consulta pois apresenta o esquema do banco de dados e as informações relacionadas aos objetos através de um conjunto de janelas. A janela de diretórios, por exemplo, pode ser utilizada para navegação a nível do banco de dados e a janela do banco de dados para verificar quais as classes que estão definidas dentro de um determinado esquema do banco de dados. Na janela de objetos o usuário pode recuperar os valores das instâncias dos objetos, através de botões de navegação.

A elaboração de consultas no sistema GOODIES é bastante simples e consiste na definição de restrições sobre os valores dos atributos dos objetos. A

especificação das restrições irá alterar a semântica dos comandos na janela de objetos. Por exemplo: o botão de *next*, irá buscar o próximo objeto da lista que satisfaz as condições especificadas, e não apenas o próximo elemento. Como as consultas são definidas apenas aplicando-se restrições sobre os atributos dos objetos, não é possível realizar consultas mais complexas.

2.2.2 Linguagem Gráfica

A linguagem gráfica é o formalismo visual mais utilizado pelos VQSs. Esta linguagem utiliza o paradigma diagramático com um conjunto limitado de símbolos, geralmente correspondendo a figuras geométricas (quadrados, círculos, retângulos, etc..), cada um associado a um tipo conceitual. Além dos símbolos geométricos podem existir um conjunto de ligações para expressar os relacionamentos entre os elementos conceituais. Estes elementos gráficos definidos para a linguagem de consulta constituem o conjunto de primitivas gráficas da linguagem [MOH 93, CAT 93]. Os símbolos e suas ligações são geralmente utilizados para visualização do esquema do banco de dados, sendo as consultas realizadas navegando-se pelo esquema e aplicando-se restrições aos elementos do banco de dados. A figura 2.5 apresenta a consulta da seção 2.1 formulada através de uma linguagem gráfica. Não é necessário especificar o relacionamento entre os elementos da consulta (cliente possui conta e realiza transação), pois o mesmo já é apresentado graficamente.

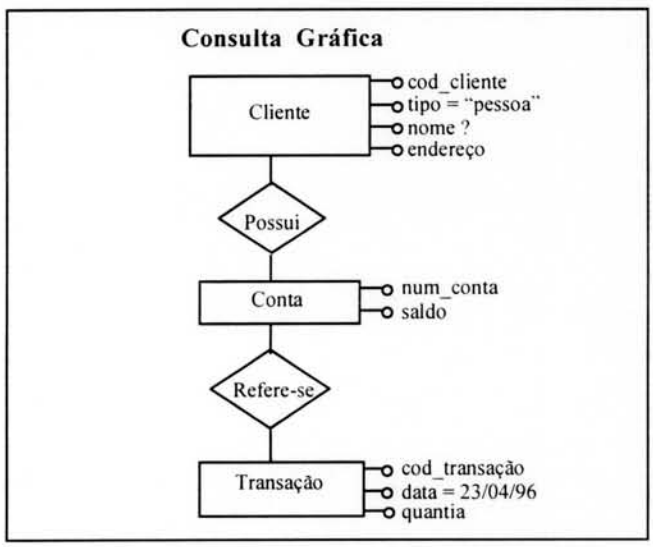

FIGURA 2.5 - Consulta Gráfica

Existem muitos sistemas que possibilitam a formulação de consultas através de uma linguagem gráfica. O sistema VILD [LEO 89], por exemplo, possui uma interface visual de consulta para SGBD multimídia, que apresenta a possibilidade de definição e manipulação de objetos, *browser* e interface de consulta. O processo de navegação sobre o esquema de dados permite a visualização de todos os seus detalhes (objetos, instâncias, relacionamentos e atributos). A interface de consulta permite que os objetos sejam selecionados no esquema de dados e que seja estabelecido um conjunto de restrições sobre os atributos e relacionamentos. Atributos complexos e

versionados são mostrados como uma série de cartões sobrepostos que podem ser investigados separadamente [MEL 94].

SUPER [DEN 95] permite a visualização do esquema utilizando uma extensão do diagrama E-R (Entidade-Relacionamento). Retângulos são utilizados para representar entidades, losângulos para representar relacionamentos, setas para relacionamentos especiais do tipo IS-A, e linhas para ligar entidades, relacionamentos e atributos. A etapa de localização dos elementos da consulta é realizada diretamente sobre o esquema gráfico. A definição do predicado é feita utilizando-se o esquema gráfico e uma caixa de texto que mostra a expressão da consulta com seus operadores e valores sobre os quais os elementos da consulta devem ser considerados. Considere a seguinte consulta: "Recuperar o código dos clientes que possuem contas com saldo superior a 1.000 reais". As entidades e relacionamentos que serão utilizados na consulta são selecionados no esquema gráfico e copiados para uma outra janela onde as restrições sobre os atributos podem ser aplicadas. A figura 2.6 mostra a consulta equivalente na linguagem SUPER. As restrições da consulta são colocadas na caixa de texto que é associada a entidade Cliente através de uma seta. A representação gráfica do esquema é modificada, os losângulos desaparecem e apenas a entidade principal aparece dentro de um retângulo (entidade *Cliente*). A expressão *∃ Possui*, indica que deve existir um relacionamento entre a entidade *Cliente* e a entidade *Conta*. A restrição da consulta (*saldo* > 1.000) e o atributo do resultado (*nome*) são ligados pelo operador lógico **AND** e prefixados pelo nome do relacionamento (se a entidade estiver a direita do relacionamento) e do nome da entidade a qual pertencem.

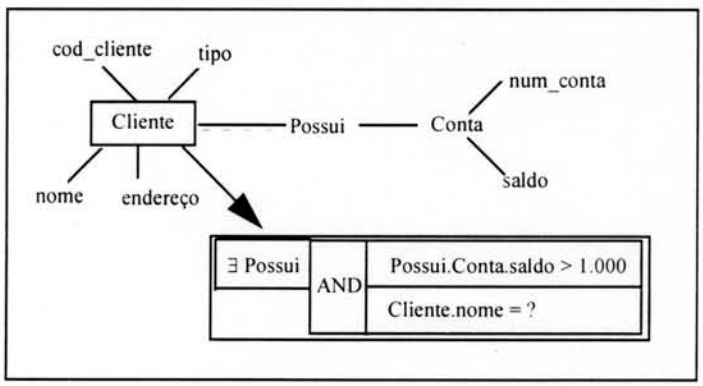

FIGURA 2.6 - Exemplo de Consulta no Ambiente SUPER

A linguagem QBD* [ANG 90] também utiliza o modelo E-R, mas acrescenta ao mesmo um conjunto de generalizações e abstrações. Este sistema permite a simplificação do subesquema de consulta utilizando uma linguagem de transformação: cada passo transforma o subesquema de consulta em outro subesquema. A idéia básica é decompor a consulta em passos elementares, que podem ser expressos por um conjunto de operadores gráficos.

A linguagem gráfica PICASSO [KIM 88] também foi desenvolvida para o modelo de dados relacional mas utiliza uma representação gráfica diferente da apresentada pelo sistema SUPER. Seu poder de expressão é equivalente ao da

linguagem textual, permitindo a elaboração de consultas que relacionem vários elementos do esquema e consultas aninhadas.

2.2.3 Linguagem de Ícones

VQSs não precisam apresentar apenas imagens de objetos reais, podem também apresentar abstrações de conceitos, ações e processos [CAT 96]. A linguagem de ícones utiliza um conjunto de ícones para representar as entidades do banco de dados e as operações que podem ser executadas sobre os mesmos [BAT 96]. Os VQSs que possuem uma linguagem de consulta baseada na utilização de ícones geralmente não apresentam o esquema do banco de dados pois prevêem a utilização do sistema por usuários não familiarizados com os conceitos de banco de dados, e que podem encontrar dificuldades em entender o modelo E-R, por exemplo. Um exemplo deste tipo de sistema é apresentado em [CAT 96] Apud [TSU 90], onde um conjunto de ícones representando os objetos do banco de dados são apresentados ao usuário. Quando um dos ícones é selecionado, o sistema apresenta um menu com um conjunto de ícones que correspondem às operações que podem ser executadas sobre os objetos.

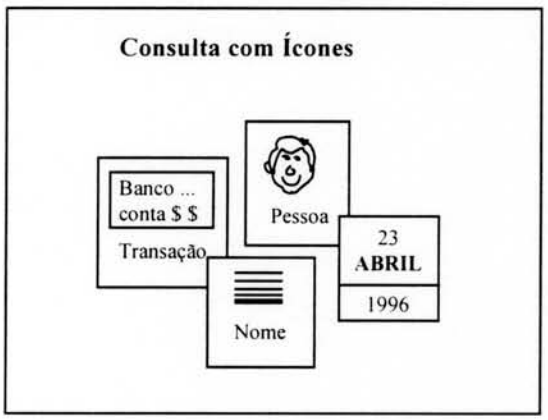

FIGURA 2.7 - Consulta Baseada em Ícones

Construir uma linguagem de consulta baseada em ícones não é muito simples. O problema está em encontrar a representação adequada para objetos e operações, pois uma mesma imagem pode apresentar significados diferentes para diferentes pessoas. Devido a este problema, muitos ícones apresentam um rótulo associado à imagem com o objetivo de facilitar a compreensão e evitar ambigüidades. A consulta da seção 2.1 é apresentada na figura 2.7. Neste exemplo, as restrições aplicadas aos elementos da consulta aparecem na forma de ícones. A restrição *tipo = "pessoa"* é representada por um ícone contendo um retrato e a inscrição *Pessoa*, a restrição *data = 23 de abril de 1996* é representada por um outro ícone contendo esta informação. O ícone com um conjunto de linhas como imagem e a inscrição *Nome* representa a saída da consulta e o ícone com a inscrição *Transação* a necessidade de existir uma transação bancária associada ao conjunto de elementos da consulta. Uma das formas de implementação de uma linguagem baseada em ícones para o modelo relacional, seria associar a cada entidade e atributo do modelo uma ou mais

representações visuais e colocar as restrições definidas sobre estes atributos como inscrições das imagens.

2.2.4 Linguagem Híbrida ou Visual

Os sistemas que apresentam uma linguagem híbrida combinam qualquer umas das linguagens de consulta apresentadas anteriormente para formulação de consultas. A linguagem de consulta híbrida também pode ser chamada de linguagem de consulta visual. O termo linguagem visual de consulta denota um caráter mais abrangente de interação onde a comunicação com o usuário é feita através de símbolos que carregam a semântica das operações [MEL 94]. A linguagem de consulta híbrida apresentada em [KIN 84], por exemplo, utiliza a representação gráfica para o esquema do banco de dados e formulários para apresentar mais detalhes sobre as entidades. Já a linguagem IQL [RAM 92] combina linguagem textual com linguagem visual para formulação de consultas SQL. As operações da linguagem SQL são apresentadas em um menu (*select*, *join*, *group by*, ...) e podem ser selecionadas pelo usuário para formulação da consulta juntamente com as representações visuais na área de trabalho da interface de consulta. A figura 2.8 apresenta a consulta da seção 2.1 na linguagem IQL. As linhas que conectam atributos de mesmo nome estabelecem o relacionamento entre as tabelas. As restrições sobre os atributos aparecem conectadas por linhas e os atributos que fazem parte do resultado da consulta possuem um fundo mais escuro (atributo nome).

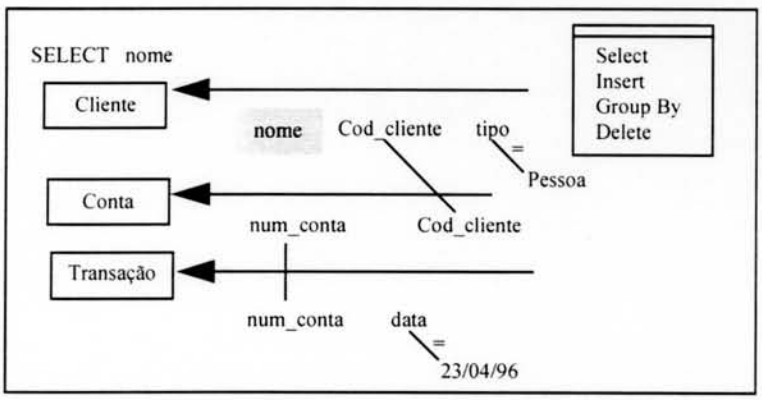

FIGURA 2.8 - Exemplo de Consulta na Linguagem IQL

A linguagem visual de consulta apresentada em [MEL 94] permite consulta gráfica, incremental, navegação e possui um conjunto de janelas que possibilitam a visualização de mais detalhes sobre cada um dos objetos. As consultas podem ser formuladas de forma gráfica ou textual e armazenadas para serem utilizadas posteriormente. A linguagem apresenta ainda a possibilidade de visualização do resultado da consulta na forma gráfica ou tabular.

2.2.5 Comparação entre Linguagens de Consulta

Baseado nos estudos realizados sobre linguagens de consulta, apresentamos aqui uma comparação entre estas linguagens, baseada nas operações realizadas no processo de elaboração da consulta. As operações básicas para elaboração da consulta (seleção, definição de relacionamentos e definição de requisitos) são realizadas por todas as linguagens de consulta visual e textual (SQL), cada uma com um conjunto de ações específicas. A tabela 2.1 apresenta as ações executadas em cada uma das linguagens para cada um dos tipos de operações. Na linguagem baseada em formulários foi utilizada a linguagem de consulta QBE por ser a mais conhecida e mais utilizada linguagem de consulta nesta categoria, assim como para a linguagem de consulta textual foi utilizada a linguagem SQL.

TABELA 2.1 - Comparação das Operações Básicas em Linguagens de Consulta

Linguagens	Seleção	Def. de Relacionamentos	Def. de Requisitos
SQL	conjunto de atributos da cláusula **SELECT** que determina os elementos participantes do resultado da consulta e conjunto de atributos da cláusula **WHERE** que determina os atributos que serão utilizados para restringir a consulta.	conjunto de tabelas da cláusula **FROM** e atributos da cláusula **WHERE** utilizados para comparar os atributos comuns em cada uma das tabelas	conjunto de restrições aplicadas aos atributos da cláusula **WHERE**
QBE	definição de variáveis ou restrições sobre dados colocados nas células da tabela	a mesma variável é colocada em colunas de mesmo nome	restrições colocadas nas células de cada um dos atributos.
Gráfica	definição de alguma restrição sobre os elementos do esquema	não é necessário especificar o relacionamento entre as entidades pois o relacionamento já aparece explicitamente no esquema	definição de restrições diretamente sobre os elementos do esquema
Ícones	seleção dos ícones de interesse para a consulta	sobreposição de ícones utilizados na consulta	seleção de ícones com restrições específicas.

Cada um destes paradigmas apresenta vantagens e desvantagens, sendo direcionada para um conjunto particular de usuários. A linguagem QBE (baseada em formulários) apresenta algumas vantagens como a facilidade de visualização da organização das informações no banco de dados e a facilidade de definição dos requisitos da consulta. [YEN 93] apresenta uma comparação entre a linguagem SQL e a linguagem QBE baseada em testes realizados com a utilização das duas linguagens. Alguns dos resultados são:

- as tabelas apresentadas pela QBE apresentam o conjunto de atributos de cada tabela, livrando o usuário da necessidade de memorizá-los, como é o caso da linguagem SQL;
- usuários que aprenderam inicialmente a linguagem QBE têm mais facilidade para executar consultas na linguagem SQL do que aqueles que não tinham utilizado a linguagem QBE anteriormente. Este fato mostra que a utilização de linguagens visuais deixa os usuários mais familiarizados com o processo de recuperação de informações;
- usuários que utilizaram a linguagem QBE entenderam mais facilmente os conceitos do modelo relacional. No entanto esta linguagem é bastante específica para modelos relacionais, onde as informações são organizadas em tabelas e sua estrutura tabular fica mais intuitiva.

A linguagem de ícones é bastante interessante para usuários não familiarizados com o conceito de banco de dados, mas apresentam algumas desvantagens como: dificuldade de elaboração de consultas mais complexas e necessidade de um estudo visual para elaboração dos ícones (a semântica da representação visual é fundamental).

A linguagem gráfica é certamente a mais utilizada pois associa elementos visuais a cada uma das informações do banco de dados. As linguagens gráficas que apresentam o esquema do banco de dados através do paradigma diagramático permitem que o usuário navegue pelo esquema selecionando os objetos de interesse e aplicando restrições as informações que farão parte do universo de consulta. Outras linguagens de consulta gráfica definem alguns símbolos gráficos e permitem que eles sejam combinados para elaboração da consulta. A utilização de uma linguagem gráfica não elimina a necessidade de uma representação equivalente em linguagem textual, pois os usuários mostram uma curiosidade em visualizar a consulta textual equivalente à consulta visual [DOA 95].

2.3 Recuperação de Informações em Banco de Dados Temporais

A utilização de um banco de dados temporal permite a recuperação dos dados armazenados e das informações de tempo associadas aos mesmos. A maioria das linguagens de consulta textuais para SGBDs temporais baseia-se na linguagem de consulta SQL e TQUEL [SNO 87], apresentando um conjunto de extensões para suportar os aspectos temporais. O modelo TSQL2 [SNO 95] foi definido por um grupo de pesquisadores da área de banco de dados temporais para servir como referência de linguagens de consulta. A linguagem de consulta deste modelo apresenta as cláusulas **SELECT**, **FROM** e **WHERE** da linguagem SQL, e um conjunto de cláusulas especiais, que permitem ao usuário estabelecer restrições temporais e obter informações válidas em determinadas datas e períodos.

As consultas realizadas sobre banco de dados bitemporais (possuem tempo de transação e tempo de validade associados às informações) podem ser classificadas

de acordo com os elementos envolvidos na seleção e na saída da consulta [EDE 94a]. Dependendo do componente de seleção, as consultas podem ser classificadas em:

- consulta de seleção sobre dados - as restrições estabelecidas na consulta são realizadas apenas sobre valores de dados. A consulta não possui nenhuma restrição sobre estes dados, associada a alguma informação temporal como data e período;
- consulta de seleção temporal - as restrições estabelecidas na consulta estão apenas associadas ao tempo no qual as informações devem ser consideradas. Por exemplo, recuperação de um conjunto de informações em uma data ou período;
- consulta de seleção mista - as condições de seleção estabelecidas na consulta atuam sobre os dados e sobre as informações temporais associadas a estes dados.

Banco de dados temporais permitem que toda história dos dados fique armazenada. Os bancos de dados bitemporias permitem a recuperação de cinco diferentes tipos de histórias, possibilitando diferentes interpretações dos dados armazenados:

- *dados instantâneos atuais* - representado por todas as informações válidas no momento presente;
- *dados instantâneos passados* - representados pelos dados válidos em um determinado instante do passado, de acordo com a atual percepção;
- *dados instantâneos de história passada* - considerando todas as informações de um determinado momento no passado, de acordo com a história válida naquele momento;
- *dados históricos* - considera todas as informações armazenadas (presente, passado e futuro) de acordo com a presente história de dados válidos;
- *dados históricos de história passada* - análogo ao anterior porem considerando uma história anterior à atual, definida por um determinado tempo de transação.

2.3.1 VQS para Modelos Temporais

Nos últimos anos muitos estudos têm sido realizados na área de banco de dados temporal e na definição de linguagens de consulta textual, mas pouca atenção tem sido dada à definição de uma linguagem visual de consulta para estes bancos de dados.

Um exemplo de linguagem visual de consulta que considera a evolução dos objetos ao longo do tempo é apresentada em [LEO 89]. A linguagem de consulta do ambiente possui uma representação gráfica para o esquema do banco de dados e uma representação gráfica diferente para os atributos que mudam de valor com o tempo (figura 2.9). Estes atributos são apresentados no esquema conceitual através de um conjunto de quadrados sobrepostos (endereço e saldo), indicando que o atributo pode apresentar diferentes valores. Embora o sistema permita a recuperação de informações através da definição de restrições temporais, a linguagem de consulta é bastante limitada.

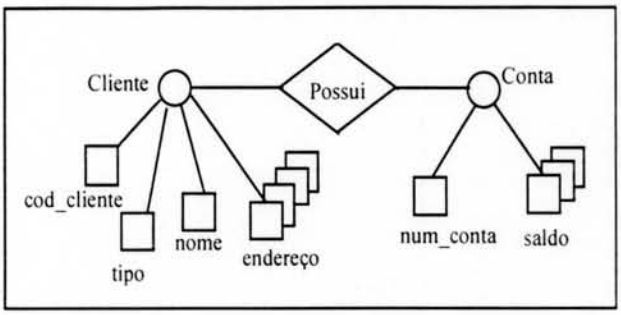

FIGURA 2.9 - Esquema Gráfico do Sistema VILD

Em [FER 94] é apresentado um ambiente gráfico de consultas para um banco de dados temporal orientado a objetos. Entretanto o ambiente preocupa-se apenas com a representação gráfica do esquema conceitual e com a definição de consultas sobre modelos de dados orientados a objetos. A consulta gráfica para definição de restrições temporais e recuperação dos estados dos objetos em determinados instantes do tempo não é definida.

Na literatura são encontradas outras duas linguagens gráficas para recuperação de informações temporais: GL_TEER [KOU 95, 95a] e GTL [OBE 94], que serão apresentadas a seguir.

2.3.1.2 Linguagem de Consulta GL_TEER

GL_TEER[KOU 95, 95a] é a linguagem de consulta gráfica para o modelo de dados TEER (Temporal Enhanced Entity-Relationship) [ELM 93]. O modelo TEER associa informações de tempo a entidades e relacionamentos. As entidades podem ser organizadas em classes e subclasses, estabelecendo-se uma hierarquia de especialização/generalização. O principal elemento temporal definido neste modelo é o *lifespan* que representa o tempo de duração de vida de uma entidade ou relacionamento. A linguagem de consulta do modelo apresenta basicamente duas cláusulas **GET** e **WHERE**:

GET <informações a serem recuperadas> : <projeção temporal>
WHERE <condições para seleção de entidades> : <seleção temporal>

A linguagem de consulta do modelo TEER é bastante simples e eficiente para definição de restrições temporais sobre os objetos envolvidos na consulta. A projeção temporal é especificada no final da cláusula **GET** e pode ser uma data ou um intervalo. A projeção temporal aplicada a uma entidade, restringe todos os valores temporais das entidades ao valor especificado na projeção temporal. A condição de seleção temporal compara dois valores temporais, o valor temporal dos atributos da cláusula **WHERE** e o valor temporal da cláusula de seleção temporal, utilizando para isto os operadores de conjuntos $=$, $\neq$, $\subseteq$, $\supseteq$. Este modelo de dados não apresenta o tempo de transação associado às propriedades que variam com o tempo, por isso não é possível recuperar estados instantâneos passados.

A linguagem de consulta gráfica GL_TEER utiliza o modelo diagramático EER (Extended Entity-Relationship) para apresentação do esquema do banco de dados. Durante o processo de formulação da consulta o usuário pode navegar sobre o

esquema selecionando e removendo relacionamentos, classes e subclasses. Depois de selecionar os elementos de interesse no esquema do banco de dados, o usuário pode utilizar operadores de restrição para definir critérios a serem satisfeitos pelos objetos selecionados. Além dos elementos diagramáticos do modelo E-R, a linguagem apresenta mais alguns elementos visuais: (i) retângulo de bordas pontilhadas - utilizado para agrupar uma parte do esquema gráfico que irá sofrer alguma restrição temporal; e (ii) seta - que associa uma restrição temporal a um conjunto de elementos do esquema. A figura 2.10 mostra a representação na linguagem GL_TEER da seguinte consulta: "Recuperar para o período de 1987 a 1990, o nome e a categoria atual de todos os instrutores do departamento de ciência da computação".

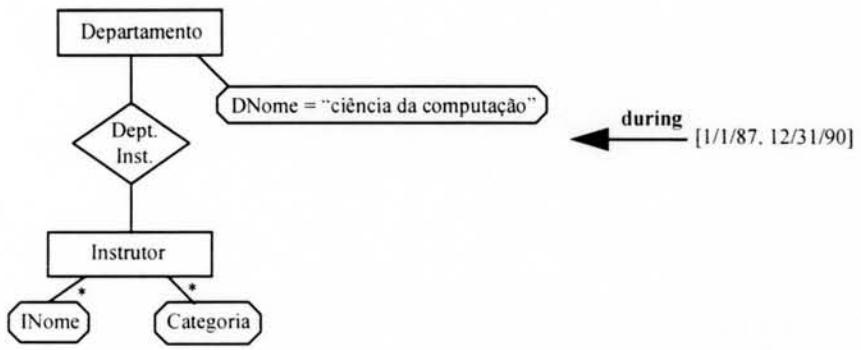

FIGURA 2.10 - Exemplo de Consulta na Linguagem GL_TEER

2.3.1.2 Linguagem de Consulta GTL

A linguagem GTL[OBE 94] é uma linguagem gráfica para dados relacionais temporais. A consulta pode estar relacionada a um único estado do banco de dados ("Qual o estado do banco de dados quando a condição c1 é verdadeira?"), ou a uma seqüência de estados ("Qual a seqüência de estados quando a condição c1 acontece, depois c2 e finalmente c3?"). A representação gráfica da linguagem possui os seguintes símbolos: (i) círculos - representando as relações do esquema; (ii) quadrado - chamado *checkpoint*, que investiga um estado ou uma seqüência de estados do banco de dados. Cada *checkpoint* representa uma condição complexa do banco de dados que deve ser satisfeita no banco de dados temporal. Círculos e quadrados são conectados por arcos para expressar o relacionamento que existe entre o estado do banco de dados e as tuplas da relação. Cada arco possui uma inscrição contendo o conjunto de tuplas da relação. A figura 2.11 apresenta a representação na linguagem GLT da consulta: "Quais as datas em que o professor João trabalhou no departamento de matemática quando era professor adjunto?".

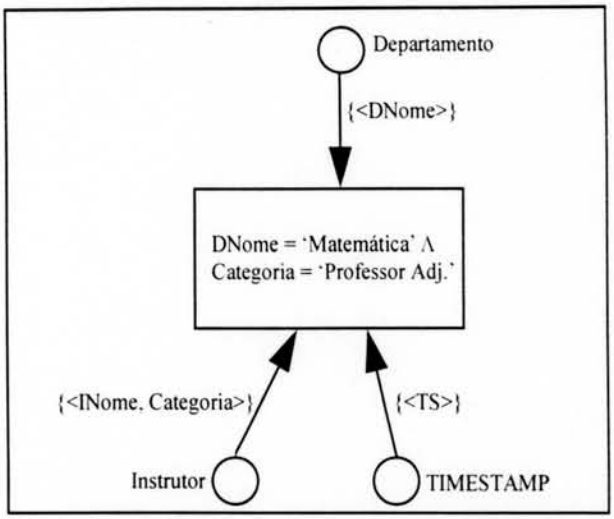

FIGURA 2.11 - Exemplo de Consulta na Linguagem GTL

2.3.2. Requisitos dos Sistemas de Consulta Visual para Modelos Temporais

Alguns autores como [WU 89, MEL 94] que analisaram sistemas de consulta para banco de dados, identificaram algumas características que devem estar presentes em linguagens visuais de consulta. Além destas características, este trabalho apresenta mais alguns requisitos adicionais. As características que devem ser apresentados pelas linguagens visuais de consulta são as seguintes:

- possibilidade de execução de uma operação de mais de uma maneira;
- capacidade de prover mais informações quando questionada;
- facilidade de desfazer uma operação realizada.

Além destes critérios podem ser acrescentados os seguintes:

- *representação do esquema do banco de dados* - a representação visual do esquema conceitual do banco de dados facilita o processo de formulação da consulta. Esta abordagem permite que o usuário tenha uma visão geral sobre a forma de organização das informações e possa definir sua consulta considerando este critério;
- *facilidade para definição de restrições aos objetos da consulta* - os conetivos e elementos relacionados utilizados pela linguagem de consulta textual devem ser apresentados ao usuário através de ícones ou listas. Assim, o usuário não precisa conhecer a priori, a sintaxe e os principais elementos da linguagem de consulta;
- *possibilidade de visualização da consulta na linguagem textual correspondente* - a possibilidade de visualização da consulta na linguagem textual permite que o usuário aprenda a sintaxe da linguagem textual de uma forma mais rápida e intuitiva, ou seja, associando ações e elementos gráficos a comandos da linguagem textual.

A linguagem visual de consulta para modelos de dados temporais deve possuir todas as características apresentadas anteriormente e mais algumas características adicionais. Um modelo de dados temporal permite que se armazene

toda a história dos objetos. Isto significa que um elemento possui vários estados armazenados. Estes diferentes estados podem ser recuperados através de restrições temporais aplicadas aos objetos envolvidos na consulta. Assim, além das restrições sobre os dados, podem ser aplicadas e recuperadas informações temporais. Para que a linguagem visual de consulta definida para um modelo de dados temporal seja eficiente, as seguintes características devem acrescentadas:

- *o editor da consulta temporal* - editores de consulta temporais são mais difíceis de construir e projetar do que editores de consulta não temporais. Em bancos de dados não temporais os usuários podem apenas fazer consultas sobre o estado corrente. Já em bancos de dados temporais, os usuários podem consultar múltiplos estados através de restrições temporais colocadas na consulta. O editor de consulta precisa suportar estas restrições temporais e a possibilidade de recuperar informações de tempo (data, período), além das informações não temporais;
- *representação visual das palavras que indicam restrições temporais* - uma linguagem de consulta temporal apresenta palavras para definição de restrições de tempo. Estas restrições temporais devem possuir uma representação visual adequada na linguagem de consulta. A representação visual para as restrições temporais pode ser feita sobre o eixo do tempo e deve facilitar o entendimento das mesmas;
- *apresentação das datas que correspondem aos estados presente, passado e futuro* - esta informação auxilia o usuário no momento da definição da restrição temporal. Assim é possível saber se a restrição de tempo definida por um período, por exemplo, está recuperando dados do banco de dados passado, atual ou futuro ou de ambos;
- *apresentação do resultado* - o resultado de consultas temporais pode envolver vários estados dos objetos armazenados e conseqüentemente um grande volume de informações. A apresentação dos resultados da consulta deve considerar estas características e apresentar o resultado da melhor maneira possível.

3 O Modelo TF-ORM

TF-ORM (*Temporal Functionality in Objects With Roles Model*) [EDE 93, 94a] é um modelo de dados orientado a objetos que utiliza o conceito de papéis para representar os diferentes comportamentos dos objetos. O modelo permite a modelagem dos aspectos estáticos e dinâmicos da aplicação pois considera todos os estados do objeto, associando informações temporais às propriedades que podem mudar de valor ao longo do tempo.

3.1 O Conceito de Papéis

O conceito de papéis no paradigma da orientação a objetos foi introduzido no modelo ORM (Object with Roles Model) [PER 90]. Nos modelos orientados a objetos tradicionais um objeto é criado como uma instância de uma classe e apresenta um identificador único que o distingue dos outros objetos. Durante todo o ciclo de vida o objeto pertence somente a esta classe, não sendo permitida a migração entre classes. Além disso, uma instância de um objeto não pode pertencer a duas subclasses simultaneamente.

O conceito de papéis tem por objetivo separar os aspectos estáticos dos aspectos dinâmicos. Um objeto continua a ser uma instância de uma única classe, mas poderá assumir ao longo de sua existência um ou mais papéis (comportamentos). O conceito de papéis permite também que um objeto apresente, simultaneamente, diversas instâncias de um mesmo papel. Este conceito não pode ser confundido com o conceito de subclasse. A existência de uma instância de uma subclasse está condicionada à instanciação desta subclasse. Um objeto existe como instância de uma classe independente do fato de estar ou não desempenhando o papel.

3.2 Características Gerais

O modelo TF-ORM considera o tempo linearmente ordenado, variando de forma discreta. As informações temporais são associadas aos objetos e seus atributos, permitindo a representação de informações temporais referentes à vida do objeto (criação e eliminação) e também com respeito à variação dos valores de um atributo (ou propriedade). O modelo apresenta a definição de dois tipos de propriedades: (i) propriedades estáticas - não mudam seu valor ao longo do tempo; e (ii) propriedades dinâmicas - pode ocorrer mudança no valor da propriedade com o passar do tempo, sendo que todos estes valores devem ficar armazenados, permitindo que todo o histórico de uma instância possa ser recuperado.

As propriedades dinâmicas possuem um tempo de transação e um tempo de validade associados. Uma informação é considerada válida quando o tempo de validade é atingido e continuará neste estado até o início da validade de outro valor. A validade de uma informação é definida pela última transação realizada. O tempo de validade pode apresentar valor nulo em alguns períodos de tempo. As propriedades dinâmicas são definidas por triplas dadas por: valor, tempo de transação e tempo de validade.

Para definição do domínio temporal utilizado pelas propriedades, o modelo apresenta alguns tipos de dados temporais: (i) pontos no tempo - utiliza os tipos de dados (data, mês, dia ...) para representar um instante no tempo; (ii) intervalos - utilizados para definir os instantes entre dois pontos no tempo; (iii) duração - representada por um número inteiro seguido de uma unidade de tempo; (iv) informações temporais incompletas - necessárias para representar informações que não possuem valor temporal bem definido.

No modelo TF-ORM cada classe é definida por um nome cn e um conjunto de papéis que uma instância da classe pode assumir. O modelo apresenta três tipos diferentes de classes: (i) classes de recursos - que modelam as informações e documentos; (ii) classes de processos - que representam as atividades que podem ser feitas com as informações e os documentos; (iii) classes de agentes - que representam as pessoas que desempenham as atividades. Cada objeto de classe possui um identificador único que é definido no instante de sua criação, armazenado em uma propriedade estática pré-definada *oId*. Uma classe é identificada por um nome cn e um conjunto de papéis $R_0,R_1,...,R_n$.

$$class = (cn, R_0,R_1,...,R_n)$$

A utilização do conceito de papel possibilita a representação de variações de comportamento de um objeto. Um objeto pode desempenhar diversos papéis durante o seu tempo de vida, simultaneamente ou não, podendo o mesmo papel possuir mais de uma instância. Todo objeto apresenta um papel básico (R_0) que descreve as características iniciais de uma instância e as propriedades globais que controlam sua evolução. A criação de um objeto de uma classe instancia automaticamente o seu papel básico que, ao contrário dos demais papéis, só pode ser instanciado uma vez. O papel básico também define as condições iniciais para os objetos criados, determinando quais os papéis que devem ser instanciados para um novo objeto.

Cada papel Ri é representado por: (i) um nome Rni; (ii) um conjunto de propriedades Pi; (iii) um conjunto de estados Si, que representa os possíveis estados para as instâncias deste papel; (iv) um conjunto de mensagens Mi, que podem ser enviadas ou recebidas; (v) um conjunto de regras de transição de estado e restrições de integridade Rui, que controlam o comportamento estático e dinâmico das instâncias através de restrições. Cada papel também possui um identificador único armazenado na propriedade estática pré-definida *rId*.

$$Ri = <R_{ni}, P_i, S_i, M_i, R_{ui}>$$

Além das propriedades estáticas pré-definidas *oId* e *rId*, o modelo TF-ORM apresenta as seguintes propriedades dinâmicas pré-definidas:

- *object_instance* - esta propriedade está armazenada no papel básico de todas as classes e seus valores representam os intervalos de tempo em que a instância está ativa ou não. Pode apresentar somente dois valores *null* e *nonull*. Se o valor da propriedade for *nonull* significa que a instância está ativa, se o valor for *null* a instância está inativa. O início da vida de um objeto é representado na propriedade *object_instance* cujo valor passa a ser *nonull*;
- *end_object* - esta propriedade também está definida no papel básico de todas as classes e armazena o instante de tempo em que a instância foi descontinuada;
- *role_instance* - esta propriedade armazena as eventuais suspensões e reativações da instância do papel;
- *end_role* - armazena os tempos de transação e validade do final da vida da instância de um papel.

As regras de transição de estado definem as possíveis transições de estado que podem ocorrer, na seguinte forma geral:

rn: state(s_1), msg(m_1) $\Rightarrow$ msg(m_2)...msg(m_n),state(s_2); (<cond. de transição>)

Para que a transição de estado possa ocorrer, o papel precisa estar no estado s1 quando receber a mensagem m1 e a condição de transição deve ser válida. Se esta condição for satisfeita, o papel passa para o estado s2 e envia uma ou mais mensagens (m_2...m_n).

As regras de integridade devem ser sempre satisfeitas e são apresentadas no seguinte formato:

constraint (<pré-condição> $\Rightarrow$ <pós-condição>)

As regras de integridade devem ser avaliadas após a execução de uma regra de transição. Se uma instância do papel satisfaz a pré-condição e não satisfaz a pós-condição, as modificações realizadas pela regra de transição de estados devem ser desfeitas.

3.3 A Linguagem de Consulta do Modelo TF-ORM

A linguagem de consulta do modelo TF-ORM baseia-se na linguagem SQL e apresenta extensões para suportar os aspectos temporais [EDE 94, 94b]. A estrutura geral do comando SQL é mantida: cláusula de especificação (determina os elementos que farão parte do resultado), cláusula de identificação (identifica os objetos sobre os quais a consulta será efetuada) e cláusula de busca (apresenta as condições a serem satisfeitas pelos objetos da consulta). Para suportar os aspectos temporais, a linguagem de consulta do modelo possui mais uma cláusula especial, que determina a história do banco de dados que deve ser considerada, e um conjunto de operadores temporais e predicados que podem ser utilizados para recuperar informações em determinados pontos no tempo.

A linguagem TF-ORM apresenta duas formas alternativas para a consulta:

SELECT <cláusula de especificação >
FROM <cláusula de identificação>
WHERE<cláusula de busca>
[**AS ON** <cláusula de instante temporal>]

SELECT <cláusula de especificação>
FROM <cláusula de identificação>
WHEN <cláusula de busca>
[**AS ON** <cláusula de instante temporal>]

A BNF completa da linguagem de consulta do modelo TF-ORM encontra-se no Anexo 1.

3.3.1 Cláusula de Especificação

A cláusula de especificação define as saídas da consulta, que podem ser de três tipos: saída de dados, saída temporal e mista. As saídas temporal e mista são conseguidas apenas com a cláusula **WHEN**, pois a cláusula **WHERE** recupera apenas os dados do banco de dados atual (hoje).

Para os exemplos a seguir vamos supor que o esquema conceitual apresente uma classe denominada *pessoa* com a propriedade dinâmica *nome* e o papel *funcionário* com a propriedade estática código do funcionário (*codf*) e a propriedade dinâmica *salário*.

A saída de dados é obtida quando são especificadas as partes do objeto que devem ser mostradas pela consulta. Para obter uma saída de dados, a cláusula de especificação pode ser composta pelo nome de uma propriedade e pelo símbolo especial "*", quando forem solicitados os valores de todas as propriedades do(s) objeto(s) identificado(s). Quando uma propriedade é definida como um conjunto ou uma lista de elementos, todos os possíveis elementos deste conjunto ou lista devem ser recuperados. Os exemplo a seguir apresentam saída de dados.

Exemplo 1: "Obter todos os salários do funcionário de código 200".

SELECT salário
FROM pessoa.funcionário
WHEN codf = 200

Exemplo 2: "Obter todas as informações do funcionário de código 200".

SELECT *
FROM pessoa.funcionário
WHEN codf = 200

Quando a cláusula de especificação contiver uma data de transação, uma data de validade ou período, a saída será temporal. Estas informações temporais estão associadas a cada uma das propriedades dinâmicas do modelo. As propriedades estáticas são válidas enquanto a instância estiver ativa, por isso não possuem data associada. Para obter a saída temporal, as seguintes palavras especiais podem ser

utilizadas na cláusula de especificação: **DATE, PERIOD, TRANSACTION_DATE, TRANSACTION_PERIOD**. O exemplo 3 apresenta saída temporal com a utilização da palavra **PERIOD** na cláusula **SELECT**.

Exemplo 3: "Obter todos os períodos em que o funcionário de código 200 teve salário superior a 1.000 reais ".

SELECT PERIOD
FROM pessoa.funcionário
WHEN codf = 200 and salário > 1.000

A saída mista é obtida quando os elementos da saída de dados e da saída temporal forem combinados na cláusula de especificação. O exemplo anterior poderia ser modificado para produzir uma saída mista, acrescentando-se a propriedade salário na cláusula **SELECT**. A consulta é apresentada a seguir:

Exemplo 4: "Obter os valores dos salários e o respectivos períodos de tempo em que o funcionário de código 200 teve salário superior a 1.000 reais ".

SELECT PERIOD, salário
FROM pessoa.funcionário
WHEN codf = 200 and salário > 1.000

A relação entre os elementos da cláusula de especificação e o tipo de saída é mostrada na figura 3.1.

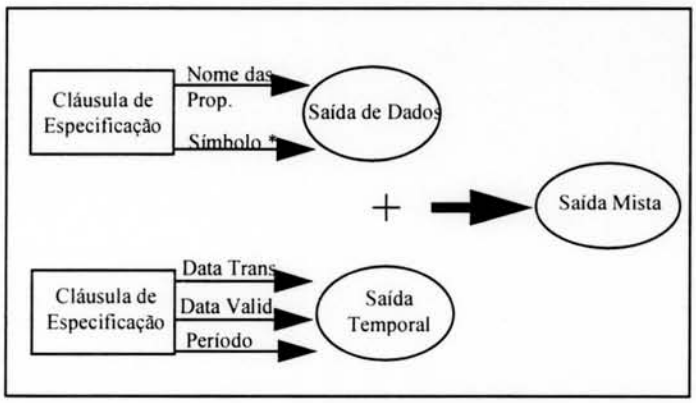

FIGURA 3.1 - Tipos de Saída para Consulta

3.3.2 Cláusula de Identificação

Na cláusula de identificação são especificados as classes e os papéis sobre os quais se deseja recuperar informações. Se o esquema sobre o qual a consulta é definida apresenta todos os papéis com nomes únicos, não é necessário utilizar o nome do papel associado à classe a qual o mesmo pertence. O exemplo a seguir recupera valores da classe pessoa e do papel funcionário, por isso a cláusula **FROM** contém o nome da classe precedido pelo nome do papel. Como o esquema apresenta todos os papéis com nomes únicos o nome da classe poderia ser omitido.

Exemplo 5: "Obter o nome e o salário de todos os funcionários".

SELECT nome, salário
FROM pessoa.funcionário

3.3.3 Cláusula de Busca

A cláusula de busca pode ser identificada pela cláusula **WHERE** ou pela cláusula **WHEN**. As condições temporais são acrescentadas à cláusula de busca **WHEN** para recuperação de informações em uma faixa do tempo e para definição de restrições temporais aos objetos envolvidos na consulta. A cláusula **WHERE** é utilizada para recuperar informações correspondentes ao banco de dados atual, ou seja, as informações válidas hoje. A utilização da cláusula **WHEN** aumenta o universo de busca para todas as informações da história considerada, incluindo dados passados, presentes e futuros.

A cláusula de especificação define o componente de saída da consulta, que será obtido sobre o domínio de valores a serem analisados (cláusula de busca). Na cláusula de busca são especificadas as condições que devem ser satisfeitas pelos objetos recuperados. A tabela 3.1 apresenta os possíveis estados dos funcionários de código 100 e 101

TABELA 3.1 - Valores de Propriedades e sua Relação Temporal

codf	salário	*t_timei*	*v_timei*
100	900	20/03/96	01/04/96
100	1.000	15/07/96	01/08/96
101	1.100	02/06/96	01/07/96
101	1.300	15/07/96	01/08/96
100	1.200	11/09/96	01/10/96
100	1.500	05/11/96	01/01/97
101	1.600	01/02/97	05/04/97

Considere a seguinte consulta: "Obter o salário atual do funcionário de código 100".

SELECT salário
FROM pessoa.funcionário
WHERE codf = 100

Considerando que a data atual é 10/11/96, então o valor recuperado será de 1.200 reais. As propriedades *t_timei* e *v_timei* correspondem, respectivamente, a tempo de transação inicial e tempo de validade inicial. A inserção de um novo valor para a propriedade *v_timei* determina o fim da validade do valor anterior. No exemplo, o valor de 900 reais para a propriedade salário inicia sua validade em 01/04/96 e vai até 01/08/96, quando o valor da propriedade salário passa a ser de 1.000 reais. O banco de dados passado possui três valores para a propriedade salário

pois estes valores estão associados a tempos de validade inferiores à data atual. No banco de dados atual, o valor da propriedade salário é de 1.200 reais: este valor iniciou sua validade em um tempo passado e ainda é válido no instante atual. O banco de dados futuro possui existe apenas um valor definido para a propriedade salário (1.500 reais) com uma data de validade superior a 10/11/96 (data atual).

A cláusula **WHEN** permite a recuperação de toda a história do banco de dados (presente, passado e futuro). Duas situações podem ocorrer quando utilizada a cláusula **WHEN**:

- a cláusula de busca não possui nenhuma restrição temporal - serão analisadas todas as informações do banco de dados (presente, passado e futuro).
- a cláusula de busca contém as palavras **DATE** (data de validade), **PERIOD** (intervalo limitado por duas datas de validade), **TRANSACTION_DATE** (data de transação) ou **TRANSACTION_PERIOD** (intervalo limitado por duas datas de transação) - serão analisadas apenas as informações que satisfazem a restrição temporal da cláusula de busca e as propriedades da cláusula de especificação serão consideradas neste período ou data (e não em toda história do banco de dados).

Considere a seguinte consulta: "Obter todos os salários do funcionário de código 100". A consulta equivalente na linguagem TF-ORM seria:

SELECT salário
FROM pessoa.funcionário
WHEN codf = 100

Com a utilização da cláusula WHEN serão recuperados todos os valores da propriedade salário para o funcionário de código 100. A saída da consulta seriam os valores (900, 1.000, 1.200 e 1.500). Para obter o salário do funcionário de código 100 em um determinado período, por exemplo, de 01/05/96 à 01/09/96, a consulta equivalente seria:

SELECT salário
FROM pessoa.funcionário
WHEN codf=100 and **PERIOD** in [01/05/96, 01/09/96]

Os valores recuperados para esta consulta seriam 900 e 1.000 reais.

A palavra **DATE** pode ser utilizada na cláusula de busca para restringir o universo de consulta a uma determinada data. Considere a seguinte consulta: "Obter o salário do funcionário de código 101 em 15/03/97".

SELECT salário
FROM pessoa.funcionário
WHEN codf=101 and **DATE** = 15/03/97

O valor recuperado para a consulta seria de 1.300 reais pois o valor de 1.600, inserido no banco de dados em 01/02/97, só inicia sua validade em 05/04/97.

3.3.4 Cláusula de Instante Temporal

A cláusula de instante temporal (cláusula **AS ON**) define a data correspondente a uma história passada, mudando o referencial de tempo da consulta para o momento definido e utilizando como base as informações conhecidas naquele instante. A consulta anterior pode ser modificada para: "Obter o salário do funcionário de código 101 de acordo com que se acreditava verdadeiro em 02/07/96".

SELECT salário
FROM pessoa.funcionário
WHERE codf = 101
AS ON 02/07/96

Todas as informações inseridas no banco de dados após a data especificada na cláusula **AS ON** são desconsideradas. O valor da propriedade salário recuperado nesta consulta será de 1.100 reais pois é o último valor definido para esta propriedade antes de 02/07/96.

3.3.5 Predicados, Funções e Operadores Temporais

O modelo TF-ORM possui também um conjunto de operadores temporais que podem ser utilizados na cláusula de busca e permitem a avaliação de condições em determinados momentos. Estes operadores determinam o momento do tempo em que a expressão sobre a qual está agindo deve ser avaliada. A tabela 3.2 apresenta os operadores e seu significado.

TABELA 3.2 - Operadores Temporais

Operadores	Significado
A *always past*	verifica se **A** é verdadeiro em todos os momentos no passado;
A *always future*	verifica se **A** é verdadeiro em todos os momentos no futuro;
immediately past **A**	verifica se **A** é verdadeiro é um momento imediatamente anterior ao momento atual;
immediately future **A**	verifica se **A** é verdadeiro é um momento imediatamente posterior ao momento atual;
sometime past **A**	verifica se **A** é verdadeiro em algum momento no passado;
sometime future **A**	verifica se **A** é verdadeiro em algum momento no futuro;
A *since* **B**	verifica se **A** é válido em todos os momentos desde que **B** se tornou verdadeiro;
A *until* **B**	verifica se **A** é verdadeiro em todos os momentos desde a criação das instâncias envolvidas até o momento em que **B** verdadeiro;
A *before* **B**	verifica se **A** se tornou verdadeiro pela primeira vez em um momento anterior àquele em que B se tornou verdadeiro
A *after* **B**	verifica se **A** é verdadeiro em algum momento posterior àquele em que **B** é verdadeiro;

Para exemplificar a utilização dos operadores temporais, considere a seguinte consulta: "Obter o código e o salário de todos os funcionários que receberam mais de 900 reais em todos os momentos do passado".

SELECT codf, salário
FROM pessoa.funcionário
WHEN always past (salário > 900)

Se a operador *always past* não estivesse associado à condição na cláusula de busca, bastava que a condição fosse verdadeira apenas um vez. A utilização do operador *always past* garante que a condição seja avaliada em todos os momentos do passado.

Se nenhum destes operadores for utilizado, a avaliação das expressões irá depender somente da utilização das cláusulas **WHERE** ou **WHEN** e das palavras **DATE** , **PERIOD**, **TRANSACTION_DATE** e **TRANSACTION_PERIOD**.

Além dos operadores temporais, existe um conjunto de predicados e funções temporais definidos para o modelo que podem ser utilizados na cláusula de busca.

Considere a seguinte consulta: "Obter código dos funcionários que possuíam salário de 800 reais na data de 21/09/95". A linguagem de consulta apresenta o predicado *is_valid_at* que pode ser utilizado para verificar se a propriedade *salário* possui o valor especificado na data de 21/09/95.

SELECT codf
FROM pessoa.funcionário
WHEN is_valid_at (salário, 800, 21/09/95)

Estes predicados permitem verificar se uma instância está ativa ou não, se uma determinada propriedade possui um valor associado em um certo momento, entre outros. A seguir serão apresentados estes predicados:

- *has_class_instance* (NomeClasse, Id) - verifica se existe pelo menos uma instância da classe NomeClasse, retornando o identificador da instância na variável Id;
- *has_role_instance* (IdClasse, NomePapel, IdPapel) e *has_role_instance* (NomePapel, IdPapel) - retorna verdadeiro se existe pelo menos uma instância do papel NomePapel para a instância da classe identificada por IdClasse, o identificador deste papel é retornado em IdPapel. A segunda forma é utilizada quando a instância da classe é identificada pelo contexto;
- *active_class* (Id) e *active_role* (Id) - estes predicados podem ser utilizados para verificar se uma instância está ativa ou não. Id corresponde ao identificador da instância da classe ou do papel considerado. O valor das propriedades class_instance e role_instance determinam se instância está ativa ou não, se o valor for *nonull* a instância está ativa (caso contrário, valor *null*, a instância está inativa).
- *active_class_at* (Id, Tempo) e *active_role_at*(Id, Tempo) - retorna verdadeiro se o papel ou classe identificado por Id estava ativo no instante de tempo determinado por Tempo;
- is_valid (Id, NomePropriedade, Valor) e is_valid (NomePropriedade, Valor) - retorna verdadeiro se o valor da propriedade NomePropriedade for aquele definido por Valor;

- *is_valid_at* (Id, NomePropriedade, Valor, Tempo) e *is_valid_at* (NomePropriedade, Valor, Tempo) - retorna verdadeiro se a propriedade NomePropriedade tiver o valor definido em Valor no tempo determinado;
- *out_role* (Id, NomePapel) - retorna verdadeiro se a instância da classe Id não possui nenhuma instância no papel NomePapel;
- *role* (IdClasse, IdPapel) - retorna verdadeiro se o papel identificado é uma instância da classe;
- *before* (Instante, Instante) - retorna verdadeiro quando o primeiro instante de tempo é anterior ao segundo;

As funções temporais do modelo TF-ORM são as seguintes:

- *value* (Id, NomePropriedade) e *value* (NomePropriedade) - retorna o valor atualmente válido para a propriedade NomePropriedade, sendo a instância identificada por Id;
- *value_at* (Id, NomePropriedade, Tempo) e *value_at* (NomePropriedade, Tempo) - retorna o valor válida da propriedade no instante de tempo definido por Tempo;
- *valid_time* (Id, NomePropriedade) e *valid_time* (NomePropriedade) - retorna o tempo de validade associado ao último valor definido para a propriedade;
- *transaction_time* (Id, NomePropriedade) e *transaction_time* (NomePropriedade) - retorna o instante de tempo em que foi feita a última transação referente à propriedade definida;
- *class_criation_time* (Id) e *role_creation_time* (Id) - Esta função retorna o instante de tempo de criação da instância da classe ou do papel considerado;
- *class_end_time* (Id) e *role_end_time* (Id) - retorna o instante de tempo correspondente à destruição da instância da classe ou do papel.
- *state* (Id) - retorna o estado atual do papel_base da classe (quando utilizado o identificador de classe) ou do papel (no caso de utilizar identificador de papel);
- *state_at* (Id, Tempo) - retorna o estado em que se encontrava o papel base da instância da classe ou o papel da instância do papel, no instante de tempo definido por tempo.

3.3.6 Extensões da Linguagem de Consulta

Foram definidas duas extensões para a linguagem de consulta, visando aumentar seu poder de expressão. A primeira acrescenta à linguagem TF-ORM as funções de agregação da linguagem SQL, que operam sobre um conjunto de valores. As funções de agregação podem ser utilizadas com propriedades estáticas ou dinâmicas e são as seguintes:

- *count* - contador de ocorrências;
- *sum* - acumulador de valores de propriedades estáticas ou dinâmicas com domínio do tipo numérico;
- *min* - menor valor de uma propriedade;

- *max* - maior valor de uma propriedade;
- *avg* - média de valores das propriedades do tipo numérico.

Usando as funções de agregação, é possível obter os valores máximos e mínimos em um determinado período de tempo, ou a média, soma ou número de valores para o período considerado. Considere a seguinte consulta: "Obter o menor e o maior salários pagos aos funcionário no período de 23/07/93 até 01/03/96". A consulta equivalente na linguagem TF-ORM é mostrada a seguir:

SELECT min(salário), max(salário)
FROM pessoa.funcionário
WHEN PERIOD in [23/07/93, 01/03/96]

Para apresentar a outra extensão definida para a linguagem de consulta do TF-ORM considere que o exemplo apresentado da seção 3.3.3 possui mais uma propriedade dinâmica no papel *funcionário*, propriedade *trab_depto* (trabalha departamento). Esta propriedade tem como domínio a classe departamento, que por sua vez possui as propriedades *nomed* (nome do departamento). Considere a seguinte consulta: "Obter no nome do atual departamento em que trabalha o funcionário de código 101". A consulta equivalente na linguagem TF-ORM deve indicar na cláusula de busca que deve existir uma instância da classe departamento associada à propriedade trab_depto no papel funcionário. Para expressar o relacionamento entre classes e papéis são utilizadas as propriedades e funções definidas no modelo. A consulta na linguagem TF-ORM, para este exemplo é mostrada a seguir:

SELECT nomed
FROM pessoa.funcionário
WHERE codf = 101 and
has_class_instance(pessoa, oId1) and
has_role_instance(oId1, funcionário, rId1) and
value (rId1, trab_depto) = Ox and
has_class_instance (departamento,oId2) and
value(oId2, nome) = Oy and Ox = Oy

Para evitar que tenham que todos estes predicados precisem ser colocados para expressar consultas deste tipo, foi introduzido mais um predicado a linguagem de consulta do TF-ORM. O predicado é *has_property_instance* e possui dois argumentos: (i) NomePropriedade - nome da propriedade que tem como do mínio uma classe ou papel; e (ii) NomeClasse ou NomePapel - nome da classe ou do papel que são domínio da propriedade definida em NomePropriedade. O predicado retorna verdadeiro se existe uma instância da classe ou papel associada a propriedade. Utilizando-se este predicado, é possível reduzir o número de condições necessárias na cláusula de busca. A consulta anterior poderia ser escrita da seguinte forma:

SELECT nomed
FROM pessoa.funcionário
WHERE codf = 101 and has_property_instance(trab_depto, departamento)

4 Estudo de Caso

Com o objetivo de exemplificar as consultas no modelo TF-ORM foi escolhido como estudo de caso uma empresa de planos de saúde. O objetivo da empresa é a prestação de serviços de consultas médicas por médicos associados da empresa, exames necessários ao diagnóstico e internação. Este estudo de caso foi escolhido por apresentar um grande número de informações que mudam de valor com o passar do tempo, onde as consultas temporais são bastante importantes. No processo de análise foram identificados os agentes envolvidos, os recursos manipulados e os procedimentos executados.

A empresa possui um conjunto de médicos, funcionários e clientes. O funcionário é identificado por um código e possui um conjunto de informações adicionais como: cargo, salário o departamento em que trabalha. Todas estas informações podem mudar de valor e por isso serão modeladas como propriedades dinâmicas no modelo TF-ORM. Um médico é identificado pelo seu CREMERS e sua especialidade. O cliente da empresa, pessoa que adquire um plano de saúde, é identificado por um código e um conjunto de propriedades dinâmicas: número de consultas que realizou, o plano de saúde que possui e os médicos que consultou. Tanto os funcionários como os médicos e os clientes possuem um conjunto de propriedades comuns: nome, sexo, data de nascimento, endereço. No modelo TF-ORM estas propriedades comuns fazem parte da classe PESSOA e os papéis desta classe são: funcionário, médico e cliente. Isto porque uma pessoa pode ser ao mesmo tempo médico e cliente, por exemplo.

O plano de saúde é modelado como uma classe e possui as seguintes propriedades: código do plano, nome do plano, valor, carência de consulta, carência de internação, carência de urgência e carência de exame. Com exceção do código do plano, todas as outras informações podem variar com o tempo.

A empresa possui ainda alguns hospitais e laboratórios credenciados para atender seus clientes. Os laboratórios contém os exames que realiza, o tempo de resposta de cada exame e o valor. Os hospitais possuem dois tipos de informações: o número de leitos que o hospital possui e se tem ou não atendimento de emergência. Além destas informações, os hospitais e laboratórios possuem um nome e um endereço. As informações comuns são propriedades dinâmicas da classe instituição que possui duas subclasses: hospital e laboratório.

A seguir este estudo de caso é modelo utilizando o modelo TF-ORM. A modelagem não apresenta as mensagens, regras e decisões pois estas informações não são utilizadas na linguagem de consulta.

agent class (
PESSOA,
< **Base_role**,

static properties = {
(sexo, {F,M}),
(data_nasc, date)}
dynamic properties = {
(nome, string),
(endereço, string)}
rules = {
r1: state(active), msg(create_object) ⇒ msg(allow_role(cliente)),
r2: state(active), msg(create_object) ⇒ msg(allow_role(funcionário)),
r3: state(active), msg(create_object) ⇒ msg(allow_role(médico)),
>,

< **Cliente**,
static properties = { (codc, string) }
dynamic properties = {
(num_cons, integer),
(consulta,MÉDICO),
(utiliza, INSTITUIÇÃO),
(plano, PLANO_SAÚDE)}
messages = { *conjunto de mensagens* }
states = { carência, vigência, em_atraso }
decisions = { *conjunto de decisões* }
rules = { *conjunto de regras* }
>,

< **Médico**,
static properties = { (CREMERS, string) }
dynamic properties = {
(especialidade, string)}
messages = { *conjunto de mensagens* }
states = { ativo, jubilado }
decisions = { *conjunto de decisões* }
rules = { *conjunto de regras* }
>

< **Funcionário**,
static properties = { (codf, string) }
dynamic properties = {
(cargo, string),
(salário, real),
(trab_depto, DEPARTAMENTO)}
messages = { *conjunto de mensagens* }
states = { empregado, não_empregado, em_férias }
decisions = { *conjunto de decisões* }
rules = { *conjunto de regras* }
>)

resource class (
DEPARTAMENTO,

< **Base_role**,
static properties = { }
dynamic properties = {
(nomed,string),
(num_func, integer)}
messages = { *conjunto de mensagens* }
decisions = { *conjunto de decisões* }
rules = { *conjunto de regras* }
>)

resource class (
PLANO_DE_SAÚDE,

< **Base_role**,
static properties = { (codp, string) }
dynamic properties = {
(nomep, string),
(valor, real),
(tipo, {familiar, empresarial}),
(carência_internação, integer),
(carência_urgência, integer),
(carência_consulta, integer) }
messages = { *conjunto de mensagens* }
rules = { *conjunto de regras* } >)

resource class (
INSTITUIÇÃO,
< **Base_role**,
static properties = { }
dynamic properties = {
(nome, string),
(endereço,string)}
messages = { *conjunto de mensagens* }
decisions = { *conjunto de decisões* }
rules = { *conjunto de regras* }
>)

resource class (
HOSPITAL is_a **INSTITUIÇÃO**,
< **Base_role**,
static properties = { }
dynamic properties = {
(num_leitos, integer),
(emergência, {Sim,Não})}

messages = { *conjunto de mensagens* }
decisions = { *conjunto de decisões* }
rules = { *conjunto de regras* }
>)

resource class (
LABORATÓRIO is_a **INSTITUIÇÃO**,
< **Base_role**,
static properties = { }
dynamic properties = {
(realiza_exames, EXAME)}
messages = { *conjunto de mensagens* }
decisions = { *conjunto de decisões* }
rules = { *conjunto de regras* }
>)

resource class (
EXAME,
< **Base_role**,
static properties = { }
dynamic properties = {
(nome, string),
(tempoR,integer),
(valor,real)}
messages = { *conjunto de mensagens* }
decisions = { *conjunto de decisões* }
rules = { *conjunto de regras* }
>)

5 Descrição Geral do Ambiente

O ambiente descrito a seguir permite a recuperação de informações armazenadas em um banco de dados relacional, através do modelo TF-ORM. Este modelo apresenta uma linguagem textual de consulta que possibilita a recuperação de diferentes histórias do banco de dados [EDE 94a,b]. Devido às vantagens apresentadas pelas linguagens visuais, foi definido, para o modelo, um sistema visual de consulta, que permite que a recuperação de informações também possa ser realizada de forma visual.

O sistema visual de consulta do modelo TF-ORM (VQS TF-ORM - Visual Query System TF-ORM) tem por objetivo proporcionar um ambiente mais amigável para recuperação de informações. O VQS TF-ORM permite que a consulta seja realizada através de três formas alternativas: textual, gráfica ou por formulários. O ambiente apresenta ainda a possibilidade de navegação sobre o esquema conceitual gráfico, definição de níveis de detalhe, armazenamento de consultas, definição de esquemas de trabalho, representação de uma consulta na forma gráfica ou textual e vice-versa.

Como o modelo TF-ORM não possui um banco de dados específico para sua implementação, foi utilizado um banco de dados relacional. As informações do modelo foram armazenadas no banco de dados de acordo com o mapeamento definido em [CAV 95], que será apresentado posteriormente. Este mapeamento determina como as características de orientação a objetos, papéis e tempo devem ser mapeadas para o banco de dados relacional para que os dados no modelo TF-ORM possam ser armazenados corretamente em um banco de dados relacional. A interface do sistema foi implementada utilizando-se a ferramenta para construção de aplicações cliente/servidor Delphi [BOR 95, BOR 95a, BOR 95b, RUB 95], que além de apresentar um bom ambiente para o desenvolvimento de aplicações, possibilita a comunicação via ODBC (Open DataBase Connectivity).

O banco de dados relacional escolhido foi o banco de dados Watcom, por possibilitar consultas no padrão SQL/ANSI [AME 92] e possuir um ODBC bastante flexível. O ODBC definido para o banco de dados Watcom não possui limitação para o tamanho da consulta enviada via ODBC e permite a consulta a visões. Esta característica não é encontrada em todos os ODBC como, por exemplo, o ODBC do banco de dados Access, que possui uma limitação de 255 caracteres para a consulta.

O VQS TF-ORM pode ser dividido em duas partes: ambiente de interação com o usuário e ambiente de implementação. O ambiente de interação com o usuário utiliza o modelo de dados TF-ORM e as linguagens visuais de consulta (gráfica ou formulários) e textual do modelo. O ambiente de implementação utiliza o modelo relacional e a linguagem de consulta SQL (fig. 5.1). As consultas realizadas na linguagem TF-ORM são traduzidas para a linguagem SQL para que possam ser executadas no banco de dados Watcom. Como as consultas são traduzidas para o

padrão SQL/ANSI o ambiente definido poderia ser implementado sobre qualquer banco de dados relacional. A única restrição é de que o banco de dados permita consultas na linguagem SQL e apresente um ODBC com as mesmas características do ODBC do banco de dados Watcom.

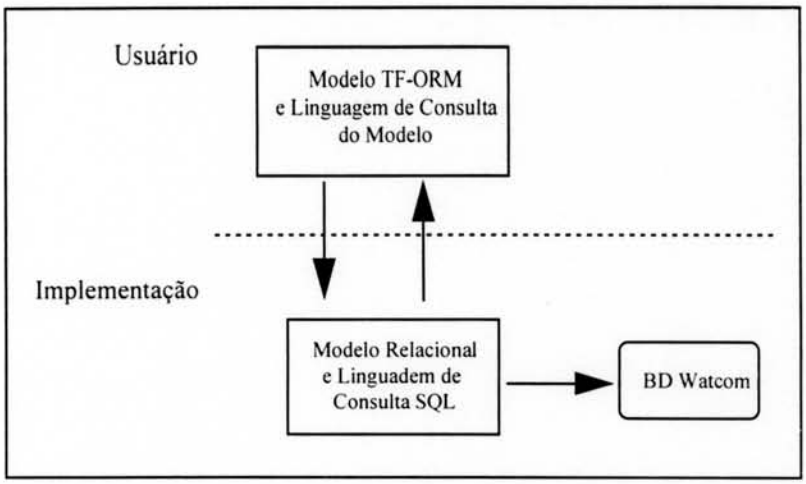

FIGURA 5.1 - Modelo de Dados e Linguagem de Consulta do Ambiente

5.1 Modelos Internos e Externos para Dados e Consulta

Com a utilização de um banco de dados relacional para a implementação do modelo, o ambiente conta com dois modelos de dados diferentes e suas respectivas linguagens de consulta. O modelo de dados e o modelo de consulta externo caracterizam a parte do ambiente sobre a qual o usuário irá atuar e são representados respectivamente pelo modelo TF-ORM e pela linguagem de consulta do modelo. O modelo de dados e o modelo de consulta interno estão relacionados à forma de implementação do ambiente e são representados respectivamente, pelo modelo relacional e pela linguagem de consulta SQL.

A figura 5.2 apresenta os dois níveis de mapeamento utilizados no ambiente:

- mapeamento do modelo de dados - faz o mapeamento do modelo de dados externo para o interno e vice-versa. As regras de mapeamento do modelo TF-ORM para o banco de dados relacional são aplicadas quando as informações do modelo são armazenadas no banco de dados. No processo de formulação da consulta o usuário trabalha sobre o esquema conceitual do modelo TF-ORM. Para que as informações armazenadas no modelo relacional possam ser apresentadas no modelo TF-ORM é aplicado um outro mapeamento que permite a conversão do modelo de dados interno para o modelo de dados externo; e
- mapeamento entre as linguagens de consulta - faz o mapeamento entre a linguagem de consulta do TF-ORM e a linguagem de consulta SQL. A tradução entre as linguagens de consulta é realizada com as informações de mapeamento entre os modelos de dados e com um conjunto de regras de mapeamento das restrições de tempo. Estas regras determinam como as cláusulas especiais na

linguagem TF-ORM serão mapeadas para restrições aplicadas sobre os atributos do banco de dados.

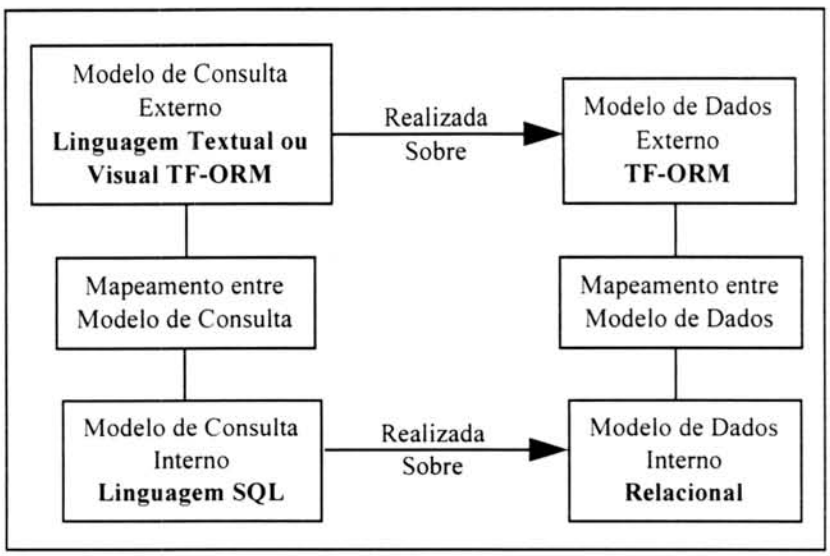

FIGURA 5.2 - Modelo de Dados e Modelo de Consulta

Além do mapeamento entre as linguagens de consulta do modelo interno e externo, existe o mapeamento entre as duas linguagens de consulta do TF-ORM (textual e visual). As consultas elaboradas através da linguagem de consulta visual são mapeadas para a linguagem de consulta textual para que possam ser traduzidas para a linguagem SQL. Existe também um mapeamento para traduzir uma consulta na linguagem textual do modelo para a linguagem visual.

5.2 Estrutura do Ambiente

A figura 5.3 mostra a estrutura do ambiente que permite a recuperação de informações através da linguagem de consulta do modelo TF-ORM sobre um banco de dados relacional, utilizando modelos externos e internos diferentes para dados e consulta. O ambiente possui vários níveis de mapeamento para permitir a utilização de modelos diferentes no mesmo ambiente.

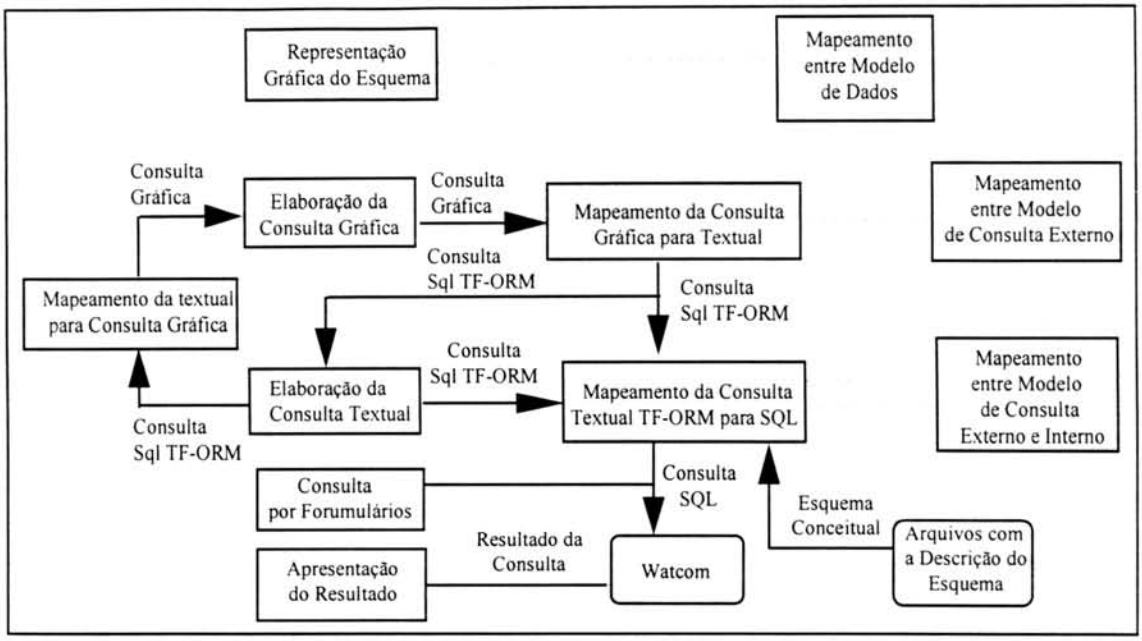

FIGURA 5.3 - Estrutura do Ambiente para Recuperação de Informações

O usuário seleciona a forma de elaboração da consulta, que pode ser textual, gráfico ou por formulários. A linguagem gráfica de consulta permite a recuperação de informações do modelo sem que o usuário conheça a sintaxe da linguagem de consulta do TF-ORM. A consulta pode ser realizada de uma forma amigável, através da utilização de símbolos visuais e de um conjunto de regras para interação com os mesmos. A linguagem de consulta gráfica conta com uma representação gráfica do modelo que permite ao usuário navegar sobre o esquema conceitual e selecionar os elementos de interesse para elaboração da consulta. Como a consulta textual é equivalente à consulta gráfica, o usuário pode realizar a mesma consulta nas duas formas. Por isso o sistema possui o mapeamento da consulta textual para a consulta gráfica e vice-versa.

A medida que o usuário vai elaborando a consulta na linguagem gráfica o sistema vai realizando o mapeamento da consulta para a linguagem textual do TF-ORM. A consulta na linguagem textual do TF-ORM pode ser utilizada pelo usuário se o mesmo optar por realizar a mesma consulta na linguagem textual. Antes que a consulta textual ou gráfica seja enviada para o banco de dados para a execução, o sistema faz a tradução desta consulta para a linguagem SQL. A consulta na linguagem SQL é enviada via ODBC para o banco de dados Watcom para que possa ser executada.

A consulta por formulários permite que o usuário visualize os valores das instâncias dos objetos através de um conjunto de janelas. A consulta por formulários não é equivalente às consultas textual e gráfica, o sistema apresenta todos os valores das instâncias de uma determinada classe sem que o usuário possa definir restrições aos valores de seus atributos. Porém é possível definir restrições de tempo sobre as quais os valores devem ser recuperados.

6 O Sistema Visual de Consulta do TF-ORM

O sistema visual de consulta do modelo TF-ORM (VQS TF-ORM - Visual Query System TF-ORM) permite a recuperação de informações através de um conjunto de símbolos gráficos, representações visuais e janelas.

As consultas às informações armazenadas podem ser realizadas de três diferentes formas:

- consulta gráfica - a consulta pode ser elaborada com a utilização de uma representação gráfica do esquema conceitual e de um conjunto de janelas e acessórios para definição de restrições sobre dados e tempo. O seu poder de expressão é equivalente ao da linguagem textual de consulta do modelo;
- consulta por formulários - os valores das propriedades dos objetos podem ser visualizadas através de uma hierarquia de janelas. Além disso, é possível determinar restrições de tempo para recuperação das informações;
- consulta textual - a consulta é elaborada utilizando-se a linguagem textual de consulta do modelo.

A seguir serão apresentados os principais módulos e elementos do sistema visual de consulta do modelo TF-ORM.

6.1 Esquema Conceitual Gráfico

A modelagem de uma aplicação utilizando o modelo TF-ORM é feita através de um formalismo textual. Porém, para a construção do sistema visual de consulta do modelo, surgiu a necessidade de se definir uma representação gráfica para um esquema do TF-ORM. A utilização de um formalismo gráfico para representar o esquema conceitual facilita a identificação dos elementos e das informações contidas no mesmo. A representação gráfica do esquema é chamada de **Esquema Conceitual Gráfico (ECGr)** e possui um conjunto de primitivas gráficas (símbolos gráficos) para representar classes, subclasses, papéis, propriedades estáticas, propriedades dinâmicas e relacionamentos. A tabela 6.1 apresenta os símbolos gráficos disponíveis e seus significados.

Os diferentes tipos de classes do modelo TF-ORM são diferenciados na representação gráfica. O retângulo de bordas arredondadas representa uma classe ou subclasse e possui uma letra em negrito no canto superior esquerdo que identifica os diferentes tipos de classes: **A** (agente), **R** (recurso) e **P** (processo). As subclasses são, obrigatoriamente, do mesmo tipo da superclasse e por isso não existe necessidade de repetir a letra que determina o tipo da subclasse. Na representação gráfica, a subclasse está relacionada à classe por uma ligação do tipo *é_um*. O rótulo *é_um* aparece em itálico para se destacar dos nomes das propriedades das classes ou papéis. Esta

ligação é representada por uma seta. As ligações entre classes e papéis são identificadas pelo rótulo que também aparece em itálico. Estas ligações são representadas por uma linha com uma terminação na forma de um retângulo na conecção com a primitiva gráfica que representa o papel.

TABELA 6.1 - Primitivas Gráficas do ECGr

Símbolo Gráfico	Semântica
A nome	representa classe ou subclasse agente
R nome	representa classe ou subclasse recurso
P nome	representa classe ou subclasse processo
nome	representa um papel
é_um	liga uma classe a uma subclasse
papel	liga um papel à sua classe ou subclasse
nome_prop.	liga uma propriedade estática a um papel, uma classe ou subclasse
d nome_prop.	liga uma propriedade dinâmica a um papel, uma classe ou subclasse
d nome_prop. nome_prop.	liga um conjunto de propriedades dinâmicas a um papel, uma classe ou subclasse

O TF-ORM possui dois tipos de propriedades: estáticas e dinâmicas. Como foi apresentado no capítulo 3, as propriedades dinâmicas podem mudar de valor ao longo do tempo e as propriedades estáticas não. Por este motivo, estas propriedades devem possuir representações gráficas diferentes. As propriedades dinâmicas têm o rótulo *d* na ligação entre a propriedade e a classe ou papel e as propriedades estáticas não possuem nenhuma identificação na ligação. Caso a classe ou papel tenha mais de uma propriedade dinâmica, o rótulo *d* pode aparecer apenas uma vez com uma linha vertical agrupando todas as propriedades dinâmicas.

O ECGr permite que se tenha idéia de como as informações estão organizadas e a qual classe de elementos do modelo estas informações pertencem (propriedades, papéis, classes entre outras). A figura 6.1 mostra a interface do ambiente para recuperação de informações do modelo TF-ORM. Quando a opção de

visualizar o esquema for selecionada, o sistema irá apresentar o esquema conceitual gráfico do estudo de caso do capítulo 4.

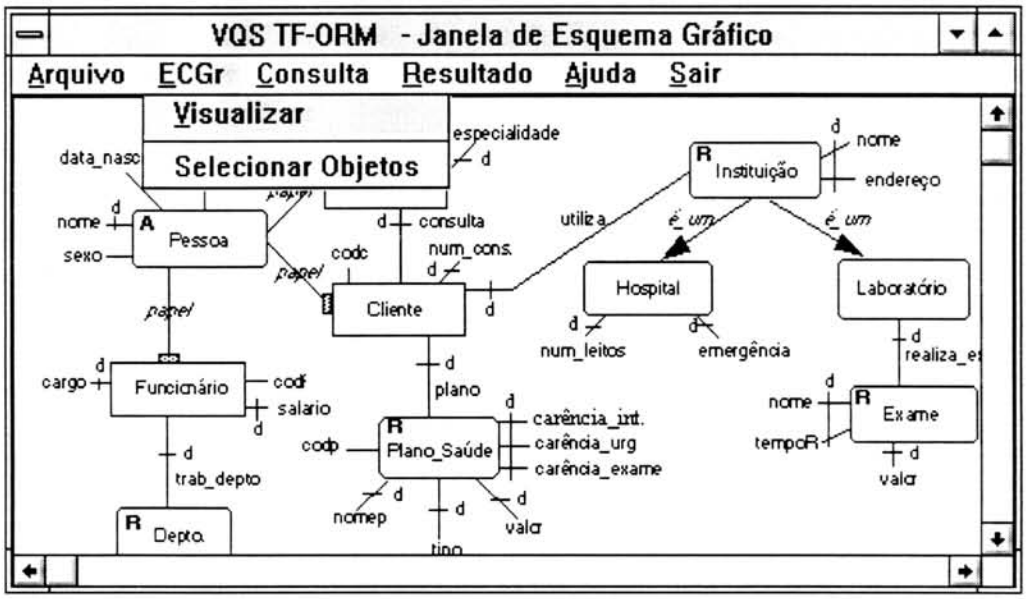

FIGURA 6.1 - Interface do VQS TF-ORM

6.2 Esquema de Trabalho

A definição de níveis de visualização do ECGr torna a linguagem visual ainda mais eficiente. Um conceito bastante interessante quando se trabalha com representação gráfica do modelo de dados é a possibilidade de definição de ECGrT (Esquema Conceitual Gráfico de Trabalho). O esquema de trabalho representa uma parte do esquema geral sobre o qual se deseja trabalhar, ou seja, elaborar consultas. Desta forma, a partir de um esquema conceitual geral podem ser definidos vários esquemas de trabalho, sendo cada um deles utilizado para uma consulta específica.

Um esquema de trabalho é criado através da eliminação de elementos do esquema geral. É possível controlar a visualização ou não dos elementos do ECGr diretamente sobre o mesmo. Esta característica facilita bastante a definição do ECGrT - o usuário pode navegar sobre o esquema e eliminar os elementos que não devem fazer parte do esquema de trabalho.

Os elementos do ECGr podem ser retirados ou acrescentados no esquema através de menus associados às classes, subclasses, papéis ou propriedades. Estes menus possuem itens diferentes para cada um dos elementos do modelo. No exemplo da figura 6.1, por exemplo, a classe *Instituição* possui duas propriedades (*nome* e *endereço*) e duas subclasses (*Hospital* e *Laboratório*). O menu associado a esta classe permite que as propriedades e subclasses estejam visíveis ou não, e que a classe seja excluída. A figura 6.2 mostra o menu de opções da classe *Instituição*, onde as opções *subclasses* e *propriedades* possuem uma marca indicando que estão presentes no esquema gráfico.

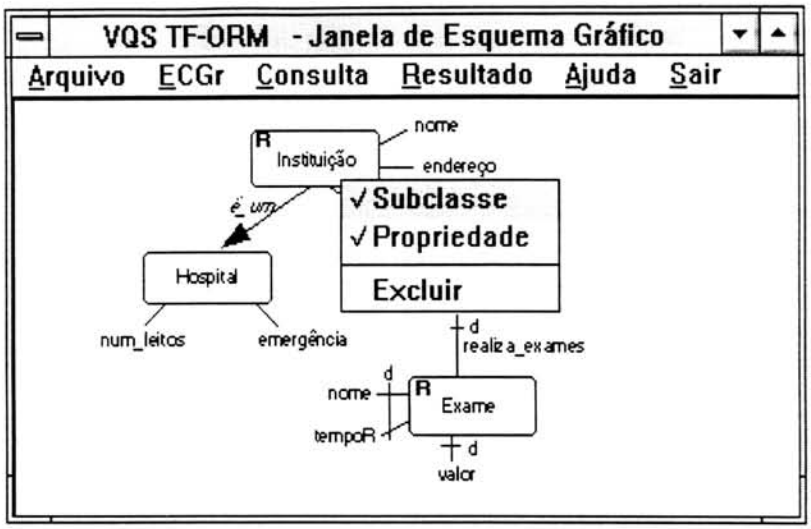

FIGURA 6.2 - Definição do Esquema de Trabalho

Esta forma de definição do esquema trabalho torna o processo de visualização de níveis de interesse mais fácil e atrativo já que é possível determinar quais os elementos que devem aparecer ou não diretamente sobre o esquema. Os ECGrTs podem ser definidos em qualquer momento e armazenados para que sejam utilizados posteriormente no processo de elaboração da consulta. A figura 6.3 mostra a janela utilizada para salvar o ECGrT da figura 6.2. Os esquemas de trabalho podem ser criados a partir de ECGrT já definidos e não apenas a partir do esquema conceitual global. Esta característica facilita ainda mais o processo de definição de níveis de detalhe e a seleção de partes de interesse do esquema gráfico.

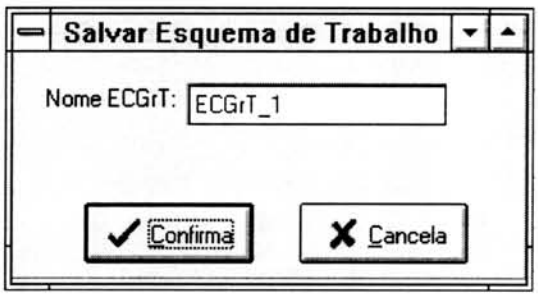

FIGURA 6.3 - Janela para Salvar o Esquema de Trabalho

6.3 Consulta Gráfica

A consulta gráfica permite a recuperação de informações através da definição de um conjunto de predicados sobre o esquema conceitual gráfico. Existem basicamente duas formas de implementação para linguagens visuais de consulta que utilizam o esquema conceitual gráfico para elaboração da consulta.

Na primeira forma de implementação as restrições aplicadas sobre os elementos do esquema gráfico podem ser colocadas diretamente sobre o mesmo. Neste caso o esquema gráfico não representa apenas o esquema conceitual, mas também a consulta. O problema desta implementação é que o esquema pode ficar muito confuso. Esta situação se agrava ainda mais à medida que as consultas vão se tornado complexas e possuem um número muito grande de restrições. Se o esquema conceitual é de um modelo temporal existe ainda mais um agravante - a inclusão de restrições temporais que não estão presentes no esquema conceitual.

A segunda forma de implementação cria uma representação particular para expressar a consulta. A consulta gráfica no VQS TF-ORM utiliza o segundo tipo de abordagem visando eliminar os problemas decorrentes da primeira forma de implementação.

A seleção de elementos do ECGr (esquema conceitual gráfico completo ou esquema de trabalho) e a definição de restrições sobre os mesmos irá gerar o **esquema de consulta**. O esquema de consulta possui as seguintes características:

- é apresentado em outra janela para que o esquema gráfico não fique confuso com as restrições aplicadas aos elementos do esquema;
- contém apenas a porção do esquema gráfico onde foram estabelecidas seleções ou restrições de valores e as restrições temporais da consulta;
- é a representação gráfica da consulta.

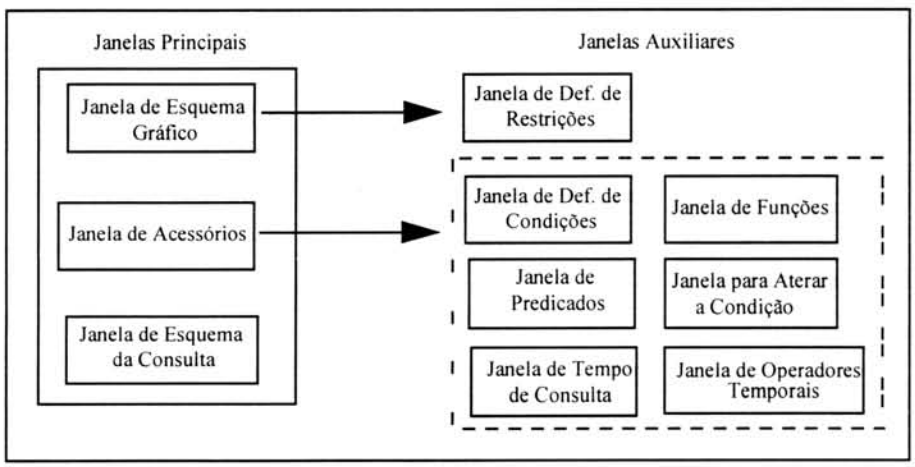

FIGURA 6.4 - Janelas para Consulta Gráfica

A figura 6.4 mostra a estrutura geral do módulo de consulta do sistema. As janelas que são apresentadas na figura em um quadrado com linha contínua (Janela de Esquema Gráfico, Janela de Acessórios e Janela da Esquema da Consulta) aparecem na tela no momento em que o usuário opta por elaborar a consulta de forma gráfica. As outras janelas são ativadas a partir destas. O ponto de partida para elaboração da consulta é o ECGr apresentado na **Janela de Esquema Gráfico**, onde o usuário seleciona os elementos que farão parte da consulta e define as restrições que devem ser aplicadas aos mesmos.

A **Janela de Acessórios** apresenta acessórios para elaboração de consultas e possui opções para ativar diversas janelas que auxiliam a definição de condições. Esta janela foi criada para que o usuário tivesse mais flexibilidade para elaborar a consulta, posicionando a janela em qualquer posição da tela.

A **Janela de Esquema da Consulta** contém a representação gráfica da consulta e é atualizada à medida em que a consulta vai sendo definida.

6.3.1 Seleção do Esquema Gráfico

O processo de recuperação de informações utilizando a consulta gráfica começa com a seleção do esquema gráfico. O usuário pode optar por elaborar a consulta a partir do esquema global ou escolher um esquema de trabalho. A figura 6.5 mostra a janela para seleção do esquema de trabalho, onde o usuário recebe uma lista com os esquemas de trabalho existentes e pode visualizar o conteúdo de cada um deles. No exemplo desta figura, o esquema de trabalho selecionado (ECGrT_1) contém a classe *Pessoa*, a classe *PlanoSaúde* e o papel *Cliente*.

FIGURA 6.5 - Janela para Abrir Esquema de Trabalho

Ao selecionar a opção *Confirma*, a representação gráfica do esquema de trabalho aparece na Janela de Esquema Gráfico. A figura 6.6 mostra a interface do sistema depois da seleção do esquema de trabalho. Além da janela que contém o esquema, são apresentas a Janela de Esquema da Consulta e a Janela de Acessórios.

O esquema gráfico mostra as classes *Pessoa* e *Plano_Saúde*, o papel *Cliente* e suas respectivas propriedades. Com a apresentação do esquema gráfico e da Janela de Acessórios, a consulta já pode ser definida. Todas as ações realizadas sobre estas janelas produzem o esquema da consulta que vai sendo apresentado na Janela de Esquema da Consulta.

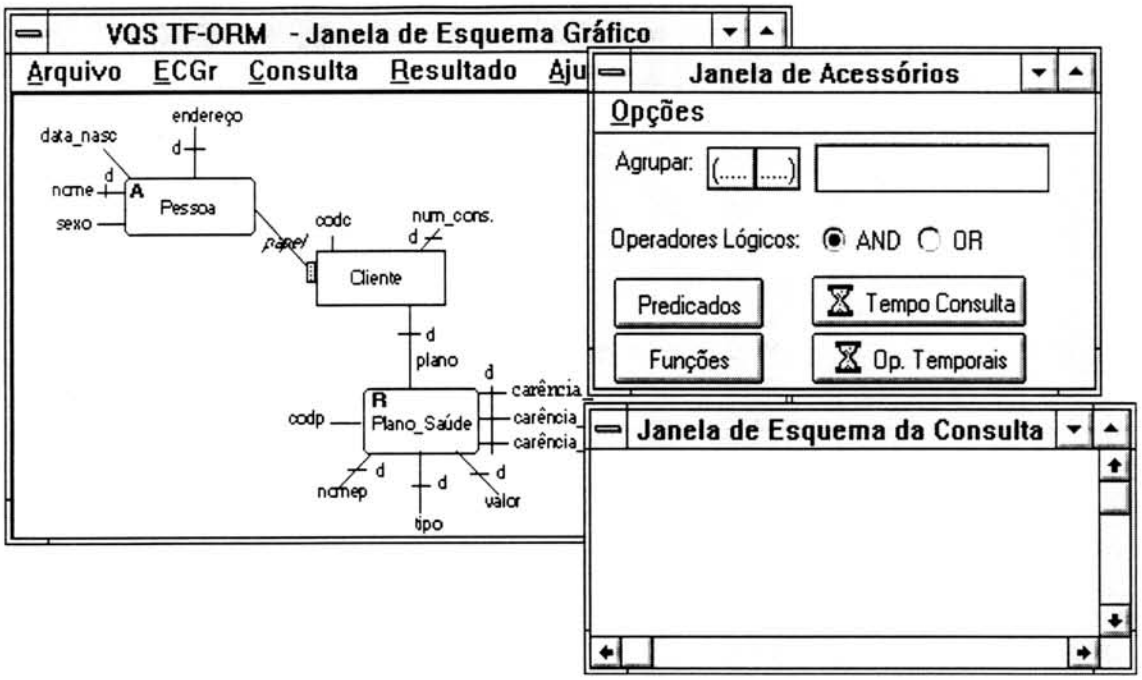

FIGURA 6.6 - Interface para Consulta Gráfica

6.3.2 Janela de Acessórios

A **Janela de Acessórios** contém um conjunto de elementos que podem ser utilizados pelo usuário na formulação da consulta. A janela é apresentada na figura 6.6 e contém os seguintes elementos:

- menu de opções - o menu principal apresenta as seguintes opções, definir condições e saída para consulta, desfazer a última operação realizada e alterar a consulta. A seleção destas opções ativa algumas janelas auxiliares como a **Janela de Definição de Condições**, por exemplo.
- botões para seleção do operador lógico - o usuário pode escolher entre os operadores **AND** e **OR**. O operador *default* é o operador **AND**, mas o operador **OR** pode ser selecionado. As restrições sobre os valores das propriedades vão sendo definidas e ligadas pelo operador lógico selecionada.
- botões com os símbolos de *abre e fecha parênteses* - são utilizados para agrupar um conjunto de condições. Quando o botão com o símbolo de *abre parênteses* é selecionado, todas as condições definidas a partir deste momento serão agrupadas, até que o botão com o símbolo de fecha parênteses seja selecionado;
- caixa de texto para visualização de condições agrupadas - todas as condições definidas na consulta depois que o botão com o símbolo *abre parênteses* é selecionado são colocadas nesta caixa de texto. Desta forma, o usuário pode visualizar as restrições que estão sendo agrupadas. A cada vez que uma condição é determinada para uma propriedade no esquema gráfico, a caixa que contém o predicado da consulta é atualizada se o botão abre parênteses estiver selecionado. Quando o *fecha parênteses* for selecionado, as condições são retiradas da caixa de texto;

- botão *Predicados* - abre a **Janela de Predicados**, onde o usuário pode selecionar um dos predicados da linguagem TF-ORM para incluir na condição da consulta;
- botão *Funções* - abre a **Janela de Funções**, onde o usuário pode selecionar um dos predicados da linguagem TF-ORM para incluir na condição da consulta;
- botão *Tempo Consulta* - abre a **Janela de Tempo da Consulta**, onde o usuário pode definir o tempo em que as informações serão consideradas;
- botão *Operadores Temporais* - abre a **Janela de Operadores Temporais**, onde o usuário pode selecionar um dos operadores temporais para incluir na condição da consulta;

A seguir será apresentada detalhadamente cada uma das janelas utilizadas na consulta gráfica.

6.3.3 Definição de Condições

A definição das condições da consulta envolvendo as propriedades do esquema pode ser feita diretamente sobre o esquema gráfico e, opcionalmente através da **Janela de Definição de Condições**. Para definir alguma restrição sobre uma propriedade no esquema gráfico, o usuário deve selecionar a propriedade desejada. A figura 6.7 apresenta a propriedade *tipo* em negrito no esquema e uma caixa de edição colocada ao lado da propriedade. A caixa de edição já apresenta a propriedade selecionada e o usuário deve completar a condição colocando alguns dos símbolos condicionais (>, <=, =, entre outros) e a constante ou expressão. Por exemplo, considere a seguinte condição: " tipo do plano de saúde deve ser familiar". Na caixa de edição é colocada a seguinte condição: *tipo = familiar*. Para que a propriedade seja apresentada no resultado o símbolo de interrogação deve ser colocado depois do nome da propriedade, por exemplo, *tipo?*.

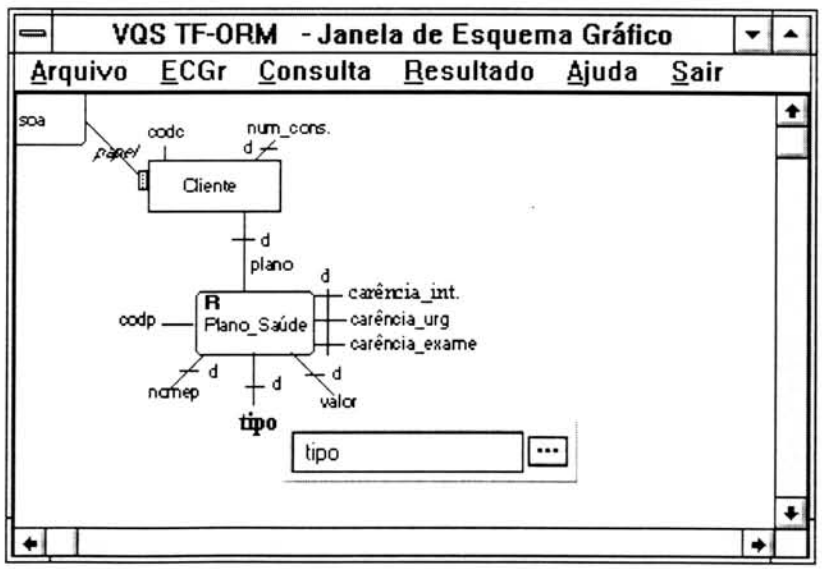

FIGURA 6.7 - Definição de Condição sobre o Esquema Gráfico

6.3.3.1 Janela de Definição de Condições

Ao lado da caixa de edição existe um botão que ativa a **Janela de Definição de Condições**. Esta janela apresenta uma lista com todos os símbolos condições, funções de agregação e permite definir se a propriedade irá aparecer no resultado ou não. A figura 6.8 mostra a Janela de Definição de Condições para a propriedade *tipo*. A propriedade selecionada no esquema gráfico é colocada automaticamente na caixa de texto com o rótulo *propriedade*. A caixa de texto não está habilitada para edição pois o objetivo é definir a condição para a propriedade selecionada no esquema e não escolher outra propriedade. Ao lado da propriedade aparece uma lista de todos os símbolos condicionais oferecidos pela linguagem (<, >=, <=, ≠, =). Depois do símbolo condicional encontra-se uma caixa de texto para definição da constante ou expressão que será associada à propriedade. Se a propriedade possuir alguma restrição para o tipo de valores que pode assumir, estes serão colocados na lista de constantes para que o usuário selecione o valor que desejar. Por exemplo, a propriedade tipo da classe PlanoSaude (*tipo*) foi definida no esquema com a restrição de que pode assumir apenas os valores *familiar* e *empresarial*. Portanto a tripla (propriedade, símbolo condicional e constante/expressão) determina a condição estabelecida sobre a propriedade.

Ainda nesta janela, além de definir uma das condições para execução da consulta, é possível determinar se a propriedade irá aparecer ou não no resultado da consulta e ainda utilizar funções de agregação sobre a mesma. Digamos que além de definir a condição para a propriedade *tipo* o usuário deseja que a propriedade apareça no resultado e que o resultado da consulta apresente o número de ocorrências da propriedade.

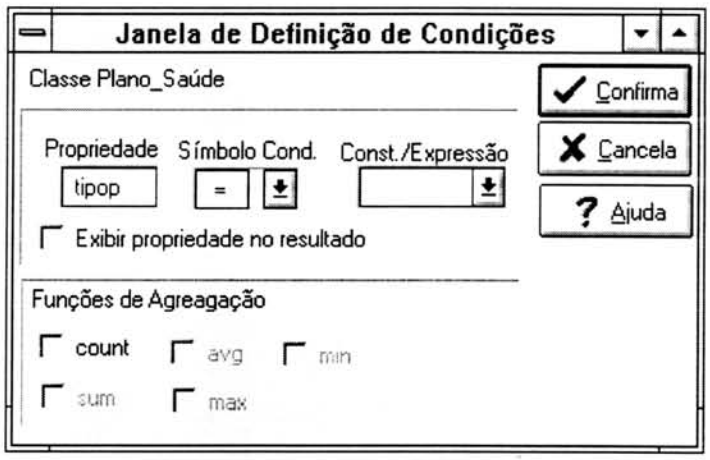

FIGURA 6.8 - Janela de Definição de Condições

Para determinar que a propriedade deve aparecer no resultado e que o mesmo apresente o número de ocorrências da propriedade, as opções *exibir propriedade no resultado* e *count* devem ser selecionadas.

Depois que as restrições foram confirmadas e a Janela de Definição de Condições foi fechada, a seguinte sentença é apresentada na caixa de texto junto a propriedade no esquema gráfico: *tipo?, count(tipo), tipo=familiar*. Depois de

confirmadas as restrições, a propriedade fica em negrito no esquema gráfico e estas restrições irão constituir o esquema de consulta que será apresentado mais tarde.

No exemplo acima, as outras funções de agregação aparecem desabilitadas na janela. Isto acontece porque as funções de agregação (*sum*, *avg*, *max*, *min*) só podem ser utilizadas com propriedades do tipo numérico, e a propriedade *tipo* não possui domínio numérico. Por exemplo, todas as funções de agregação estariam habilitadas se a propriedade *valor* (da classe *PlanoSaúde*), que tem como domínio valores numéricos, estivesse sendo utilizada.

Para uma determinada propriedade não é necessário que a definição de restrições, a associação com funções de agregação e a sua inclusão no resultado da consulta sejam feitas de uma única vez. Estes três passos podem ser executados separadamente, e em qualquer momento. Por exemplo, considere as seguintes ações sendo executadas separadamente:

- propriedade no resultado - o usuário seleciona a propriedade e coloca um ponto de interrogação depois do nome da propriedade na caixa de texto, ou abre a Janela de Definição de Condições e seleciona a opção *exibir propriedade no resultado*;
- condição - o usuário seleciona a propriedade e coloca a restrição na caixa de texto. Se a janela de condições for ativada depois que o primeiro passo foi executado, a opção *exibir propriedade no resultado* estará marcada. Caso a marca seja retirada, a propriedade não fará mais parte do resultado. Isto significa que o primeiro passo será desconsiderado;
- função de agregação - a propriedade é selecionada no esquema gráfico e a função de agregação pode ser associada à propriedade na caixa de texto ou selecionada na janela de condições, como apresentado anteriormente. Se a janela de condições for aberta neste momento, a condição definida anteriormente para a propriedade não irá aparecer na janela. As únicas informações que aparecerão sempre para uma determinada propriedade na Janela de Definição de Condições são: se a propriedade será apresentada no resultado e se existe alguma função de agregação associada a propriedade.

A classe *Pessoa*, por exemplo, armazena as propriedades comuns dos papéis *Funcionário*, *Médico* e *Cliente*. Caso seja selecionada uma propriedade de uma classe que apresenta subclasse ou papéis associados, o sistema irá escolher uma subclasse ou papel como default para associar a propriedade selecionada. Considere que para a classe *Pessoa*, o papel escolhido seja o papel funcionário. Se a propriedade *nome* for selecionada, o nome do papel irá prefixar o nome da propriedade (funcionário.nome). Neste caso a Janela de Definição de Condições irá possibilitar que o usuário troque o nome da subclasse ou papel ao qual a propriedade deve corresponder.

6.3.3.2 Definição de Condições com Propriedades Pré-Definidas

As propriedades pré-definidas da linguagem (*oId*, *rId*, *end_object*, entre outras) e as variáveis criadas pela utilização de uma função na condição da consulta não aparecem no esquema conceitual gráfico. Por isso, para utilizar estas propriedades e variáveis na consulta, a **Janela de Definição de Condições** deve ser ativada através de uma opção na Janela de Acessórios. A janela apresenta uma lista para que o

usuário selecione se deseja aplicar uma condição sobre uma propriedade do esquema, propriedade pré-definida ou variável. Se a opção selecionada for a opção propriedades do esquema, a lista irá conter apenas as propriedades que aparecem no esquema gráfico utilizado na consulta, e não toda as propriedades do esquema conceitual. As funções de agregação não aparecem na janela quando forem selecionadas as propriedades pré-definidas ou as variáveis, pois estas funções foram definidas apenas para as propriedades do modelo. A definição da condição é semelhante à apresentada na seção anterior. A figura 6.9 apresenta a Janela de Definição de Condições para definição de restrições a propriedades pré-definidas.

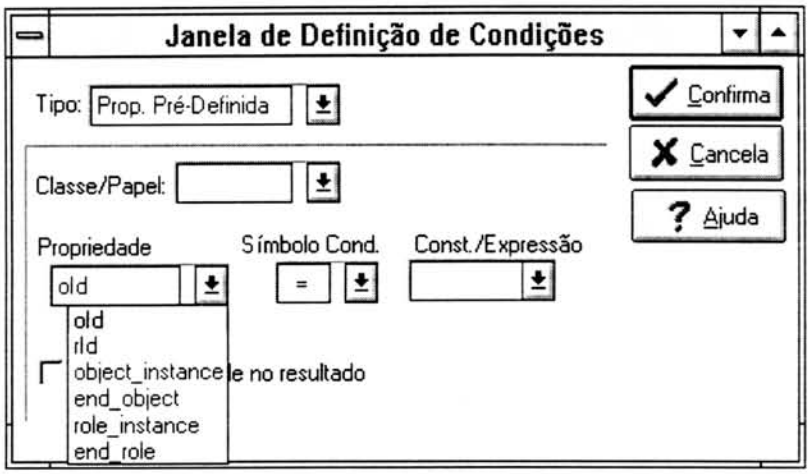

FIGURA 6.9 - Definição de Restrições sobre Prop. Pré-Definidas

6.3.4 Janela de Funções e Predicados

Os predicados e funções definidos no modelo TF-ORM podem ser incluídos na consulta. Na Janela de Acessórios, os botões *Predicados* e *Funções* podem ser utilizados para ativar a **Janela de Predicados** e a **Janela de Funções**, respectivamente. Considere a seguinte condição: "Saber se a propriedade *nome* tinha o valor João no dia 21/08/96". Para incluir esta condição na consulta, o usuário pode utilizar o predicado *is_valid_at* que se encontra na lista de predicados da **Janela de Predicados.** Para cada um dos predicados, existe um conjunto de parâmetros que são apresentados na janela para que o usuário associe os valores necessários para definição do predicado. A figura 6.10 apresenta a **Janela de Predicados** com as restrições definidas sobre os parâmetros do predicado *is_valid_at* para expressar a condição apresentada anteriormente. O parâmetro *Id* aparece com a cor azul, pois seu preenchimento não é obrigatório. Alguns dos parâmetros das propriedades podem possuir uma lista com as variáveis criadas a partir de funções e que podem ser utilizadas nos parâmetros dos predicados.

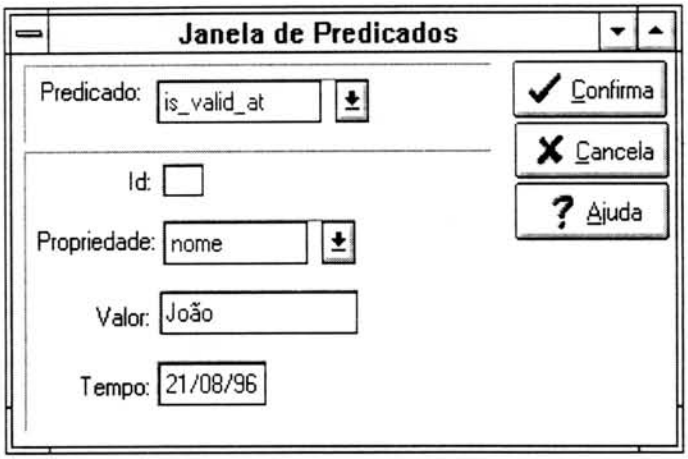

FIGURA 6.10 - Janela de Predicados

A Janela de Funções é semelhante à janela para definição de predicados, a única diferença é que a função retorna um valor e o usuário precisa definir o nome da variável onde o resultado será armazenado ou a expressão associada à função. A figura 6.11 apresenta a janela para definição da função value_at (retorna o valor válido para a propriedade em um instante do tempo). O parâmetro Id também não é obrigatório e por isso, aparece com uma cor diferente.

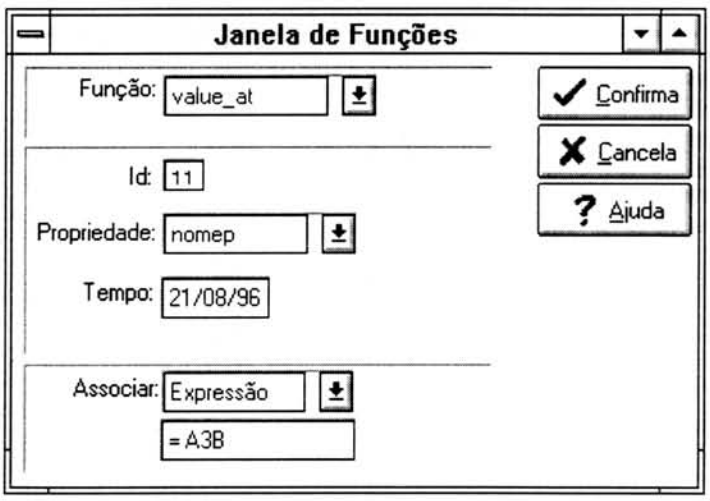

FIGURA 6.11 - Janela de Funções

6.3.5 Definição de Restrições Temporais

Existem dois tipos de restrições temporais: a restrição temporal aplicada à consulta e as restrições temporais aplicadas às condições da consulta. O tempo da consulta determina a faixa de tempo em que as informações serão analisadas. Por exemplo, a consulta pode se referir apenas às informações de uma determinada data ou período. Além do tempo definido para a consulta, as condições da consulta podem sofrer algumas restrições temporais adicionais. Por exemplo, considere que o tempo da consulta são os estados passado, presente e futuro (ou seja, todas as informações armazenadas) e que uma das restrições da consulta seja *nome = João*. Se apenas a

restrição da consulta for estabelecida, basta que esta propriedade tenha assumido este valor em algum momento na história para que a condição seja verdadeira. Para que esta condição seja verdadeira em todo o passado, deve existir uma restrição temporal explícita aplicada à condição *nome = João*.

As janelas utilizadas para definir o tempo da consulta (Janela de Tempo da Consulta) e as restrições temporais sobre condições (Janela de Operadores Temporais) podem ser ativadas através dos botões *Tempo Consulta* e *Operadores Temporais*, respectivamente, na Janela de Acessórios.

6.3.5.1 Janela de Tempo da Consulta

A Janela de Tempo da Consulta é apresentada na figura 6.12 e possui os seguintes elementos:

- tempo da consulta - apresenta as possíveis opções para o tempo da consulta: (i) *Past/Present/Future* - tempo *default* para a consulta que corresponde a todas as informações armazenadas no banco de dados; (ii) *Present* - considera apenas as informações do banco de dados atual; (iii) *Period* - recupera as informações em um determinado período que deve ser definido pelo usuário; (iv) *Date* - recupera as informações na data especificada pelo usuário;
- saída temporal - apresenta uma lista com todos os tipos de saída temporal: (i) *Date* - recupera uma ou mais datas de validade; (ii) *Period* - recupera um ou mais períodos. Cada um dos períodos é definido por uma data de validade inicial e uma data de validade final; (iii) *Transaction_Date* - recupera uma ou mais datas de transação; (iv) *Transaction_Period* - recupera um ou mais períodos. Cada um dos períodos é definido por uma data de transação inicial e uma data de transação final;

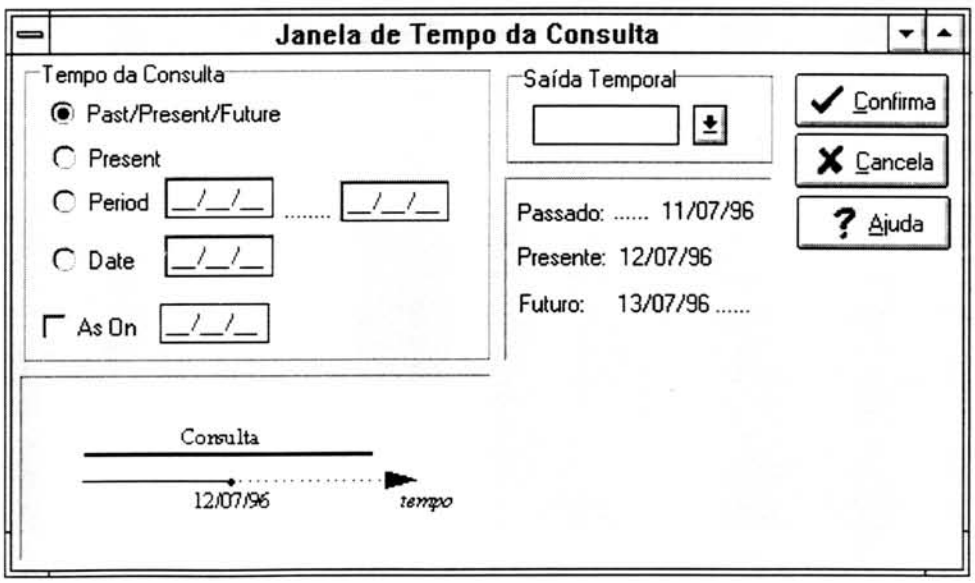

FIGURA 6.12 - Janela de Tempo da Consulta

- datas que correspondem aos estados presente, passado e futuro - algumas restrições de tempo da consulta se referem a estados do tempo, por isso são apresentadas as datas que correspondem a cada um destes estados. Considere que a data de hoje é 12/07/96. Portanto, o presente corresponde a esta data, o passado corresponde a todas as datas anteriores a 11/07/96 inclusive e o futuro a todas as datas a partir de 13/07/12 inclusive;
- representação visual da restrição temporal - a restrição temporal definida para a consulta e a utilização de operadores temporais nas condições da consulta, geram uma representação da restrição temporal no eixo do tempo;

O tempo *default* para a consulta é o banco de dados histórico, ou seja, a opção *Past/Present/Future*. Para mudar o tempo da consulta para o período de 21/02/92 até 12/08/94, por exemplo, o usuário deve executar as seguintes ações: (i) ativar a janela para definição de restrições; (ii) selecionar a opção *Period* na **Janela de Tempo**; e (iii) colocar as datas inicial e final do período de tempo nas caixas de texto que aparecem ao lado da opção *Period*.

6.3.5.2 Janela de Operadores Temporais

O modelo TF-ORM possui um conjunto de operadores temporais apresentados no capítulo 3. Para utilizar um dos operadores temporais na consulta, basta selecionar o botão correspondente na Janela de Acessórios. A figura 6.13 mostra a **Janela de Operadores Temporais**. Além da representação visual da restrição temporal e das datas que correspondem aos estados presente, passado e futuro (que já apareciam na Janela de Tempo da Consulta), encontram-se os seguintes:

- lista com os operadores temporais - lista que contém todos os operadores temporais da linguagem;
- lista de argumentos - existem dois tipos de operadores temporais, os que possuem um e os que possuem dois argumentos. A lista de argumentos, corresponde a todas as condições definidas para a consulta até o momento. O operador temporal *Sometime Past* por exemplo, possui um argumento. Já o operador *Since* possui dois argumentos. Dependendo do operador temporal selecionado serão apresentadas uma ou duas listas de argumentos;

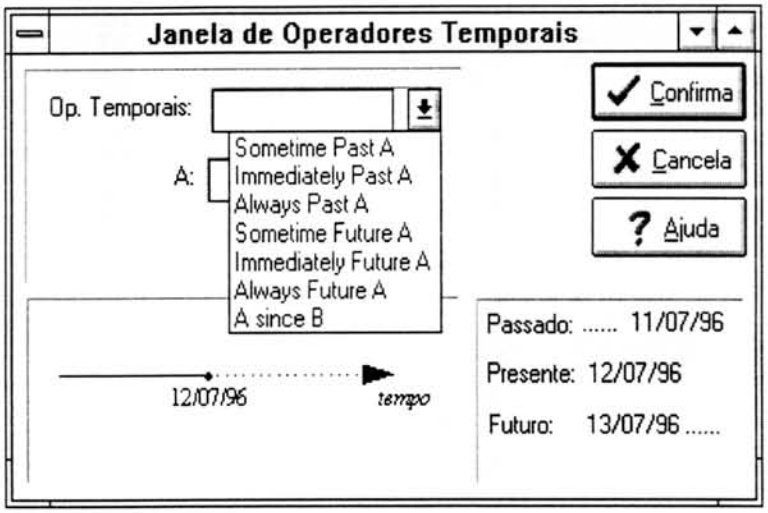

FIGURA 6.13 - Janela de Operadores Temporais

Considere que o usuário até o momento definiu as seguintes condições sobre as propriedades *codp* (código do plano de saúde) e *valor* (valor do plano de saúde): *codp = A3B* e *valor = 70,23*. E deseja que a condição *valor = 70,23* seja válida em todo o futuro. Esta restrição pode ser definida com a utilização do operador temporal *Always Future*. Na Janela de Operadores Temporais, o operador temporal Always Future é selecionado na lista de operadores temporais e a condição *valor = 70,23* na lista de argumento. A figura 6.14 mostra a Janela de Operadores Temporais depois da definição da restrição temporal.

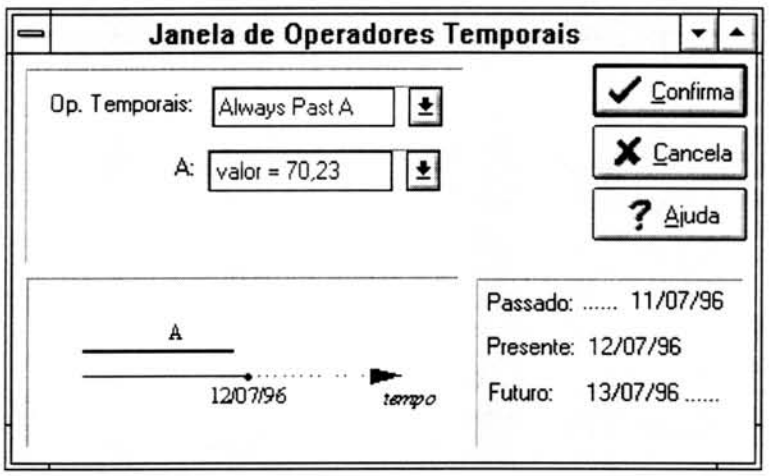

FIGURA 6.14 - Exemplo de Condição Associada a um Operador Temporal

A definição de restrições temporais não precisa ser realizada depois de estabelecida a condição sobre a qual ela irá atuar. Considere que até o momento apenas a condição *codp = A3B* tenha sido definida e que o usuário ative a **Janela de Operadores Temporais** para selecionar a opção *Always Future*. O usuário deseja que este operador temporal seja aplicado a uma ou mais condições que irá definir, ou seja, as condições não se encontram na lista de argumentos. Neste caso, o usuário seleciona apenas o operador temporal e confirma a operação. Na Janela de Acessórios, o

símbolo de abre parênteses aparece selecionado e a caixa de texto ao lado contém o operador temporal e o símbolo de abre parênteses. Isto significa que todas as condições definidas a partir deste momento estarão sofrendo a restrição temporal do operador *Always Future*. O operador temporal deve possuir no mínimo uma condição associada. Depois de definir todas as condições que desejar para o operador temporal, o usuário pode utilizar o símbolo de fecha parênteses para encerrar o argumento. A figura 6.15 mostra a Janela de Acessórios, para a seleção do operador temporal sem a definição do argumento.

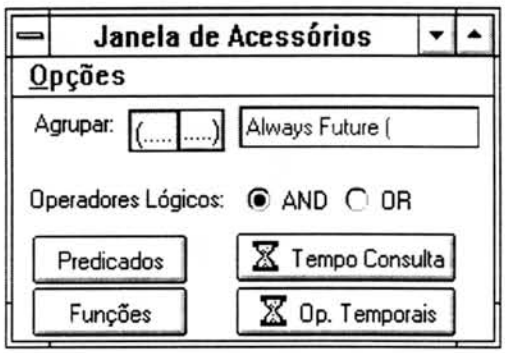

FIGURA 6.15 - Operador Temporal sem Argumento

Estes dois exemplos mostram que a consulta pode ser realizada em uma única etapa ou em duas etapas:

- consulta em duas etapas - na primeira etapa são definidas as restrições sobre as propriedades e inclusão de predicados e funções na consulta. Na segunda etapa é definido o tempo da consulta e as restrições temporais sobre uma ou mais condições já estabelecidas;
- consulta em uma única etapa - as restrições sobre os valores das propriedades e as restrições temporais são definidas em paralelo.

Existe uma restrição relacionada à utilização de operadores temporais na consulta e o tempo definido para a consulta. Os operadores temporais não podem ser utilizados nas consultas que obtém informações do banco de dados atual (opção *present* da Janela de Tempo da Consulta). Se esta opção estiver selecionada e o usuário tentar incluir algum operador temporal na consulta, o sistema irá apresentar a janela de mensagem da figura 6.16. O usuário pode confirmar a inclusão do operador temporal ou cancelar. Se a operação for confirmada, o tempo da consulta será alterado para *Past/Present/Future*, caso contrário o tempo da consulta continua o mesmo e a janela com os operadores temporais será fechada.

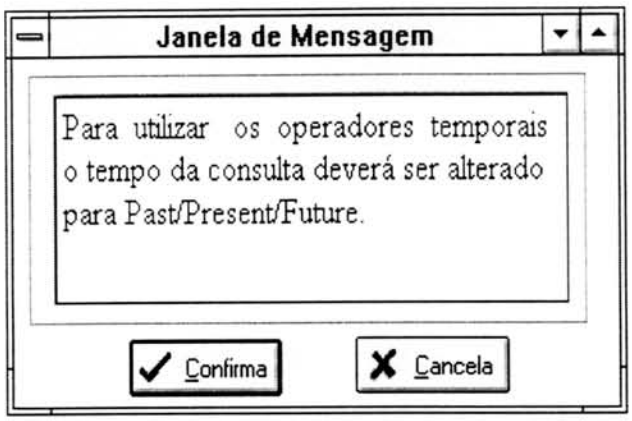

FIGURA 6.16 - Janela de Mensagem

6.3.5.3 Representação Visual para as Restrições Temporais

Cada uma das palavras especiais do modelo TF-ORM utilizadas para restringir informações a intervalos de tempo possuem uma representação visual no ambiente para recuperação de informações. Considere a data atual 12/07/96, assim o presente corresponde a esta data, o passado corresponde a todos as datas anteriores a data atual e o futuro a todas as datas posteriores a data atual. A representação visual destes estados (presente, passado e futuro) aparece no eixo do tempo. O passado é representado por uma linha cheia, o presente por um ponto sobre o eixo do tempo e a data correspondente e o futuro por uma linha pontilhada (fig. 6.17).

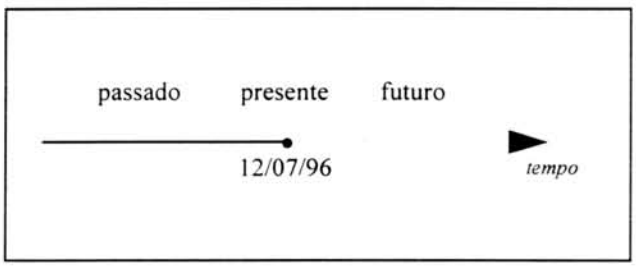

FIGURA 6.17 - Linha do Tempo

A representação visual das restrições temporais é feita sobre este eixo do tempo. Cada restrição de tempo possui uma representação particular:

- *sometime past* **A** - representada por um traço vertical e a letra **A** na região do eixo do tempo que corresponde ao passado (fig. 6.18a). Isso indica que em algum momento do tempo passado **A** deve ser verdadeiro;
- *always past* **A** - representado por uma linha cheia na parte superior do eixo do tempo que corresponde a todo o tempo passado e a letra **A** (fig. 6.18b). Isso indica que **A** deve ser verdadeiro em todos os momentos do passado e não só em algum momento do passado;
- *immediately past* **A** - representado por uma barra vertical no lado esquerdo (passado) do ponto que indica o presente (data atual) no eixo do tempo (fig. 6.18c). A utilização deste símbolo permite diferenciar o presente do passado

imediato (ontem) e indica que **A** deve ser verdadeiro em uma data anterior ao presente;

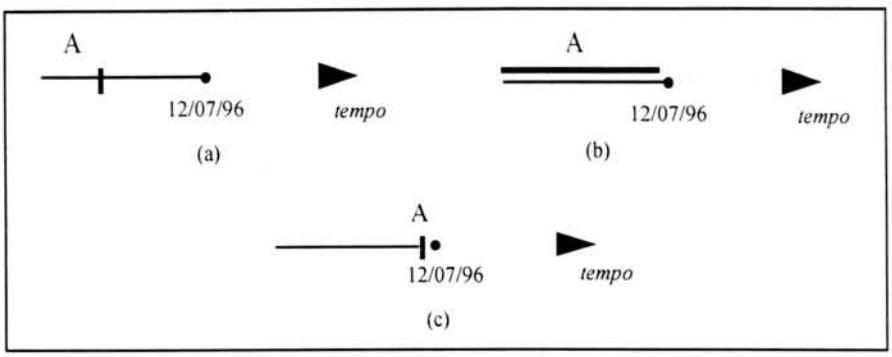

FIGURA 6.18 - Representação dos Operadores Temporais do Passado

- *sometime future* **A** - representado por uma barra vertical e a letra **A** no eixo do tempo que corresponde ao futuro (linha pontilhada) (fig. 6.19a). **A** deve ser verdadeiro em qualquer momento no futuro;
- *always future* **A** - representado por uma linha cheia na parte superior do eixo do tempo que corresponde ao futuro (eixo pontilhado). **A** deve ser verdadeiro em todos os momentos do futuro (fig. 6.19b);
- *immediately future* **A** - representado por uma barra vertical no lado direito (futuro) do ponto que indica o presente (data atual) no eixo do tempo (fig. 6.19c). A utilização deste símbolo permite diferenciar o presente do futuro imediato (amanhã) e indica que **A** deve ser verdadeiro em uma data posterior ao presente.

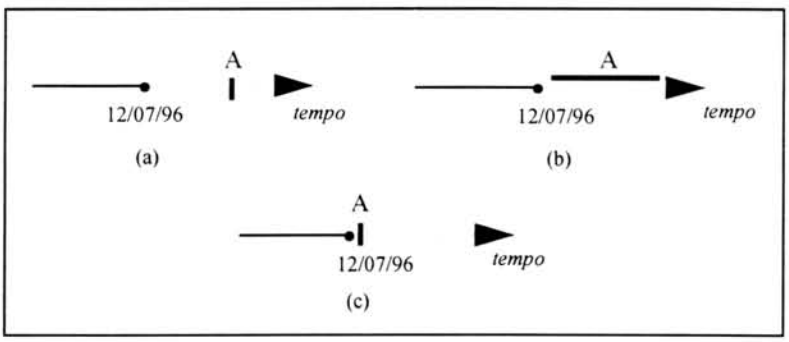

FIGURA 6.19 - Representação dos Operadores Temporais do Futuro

As restrições de tempo que associam dois conjuntos de informações também possuem uma representação gráfica:

• **A** *since* **B** - o argumento **A** é representado através de uma linha cheia e pontilhada sobre o eixo do tempo (fig. 6.20a). A linha é pontilhada antes do argumento **B** existir e depois que o argumento **B** existiu. O importante é que **A** seja válido em todos os momentos desde que **B** existiu. O valor do argumento **A** não importa nos outros momentos no tempo, por isso a linha é pontilhada;

• **A** *until* **B** - o argumento **A** é representado por uma linha cheia até o momento em que o segundo argumento se tornou verdadeiro e depois por uma linha pontilhada. O valor de **A** só interessa antes do argumento **B** se tornar verdadeiro (linha cheia antes de **B**). O valor do argumento **B** é representado por uma linha cheia sobre o eixo do tempo (fig. 6.20b);

• **A** *before* **B** - o argumento **A** é representado por uma linha cheia e pontilhada antes de **B** e por uma linha pontilhada depois de **B**. Isso indica que o argumento **A** deve ser verdadeiro em algum momento anterior a **B**, mas não precisa ser verdadeiro em todos os momentos anteriores a **B** (fig. 6.20c);

• **A** *after* **B** - o argumento **A** é representado por uma linha cheia e pontilhada sobre a linha cheia que representa o argumento **B** (fig. 6.20d). O argumento **A** deve ser verdadeiro em algum momento posterior aquele em que **B** se tornou verdadeiro.

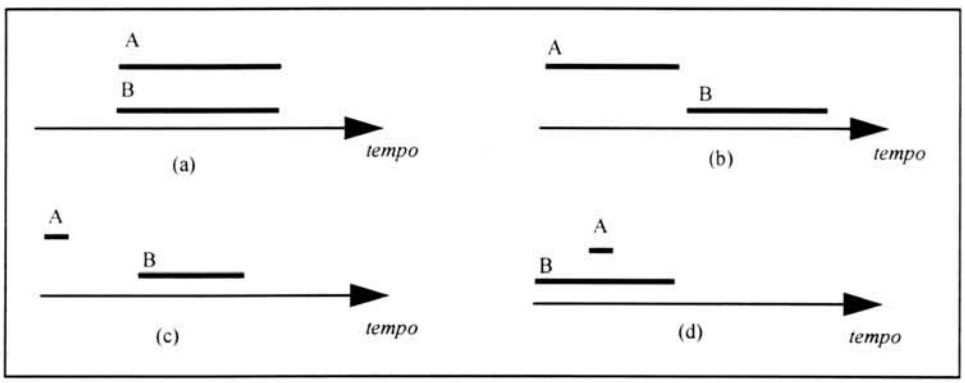

FIGURA 6.20 - Representação dos Operadores com dois Argumentos

6.3.6 Exemplos de Consulta e a Representação Gráfica

À medida que o usuário vai definindo a consulta através de condições sobre as propriedades, restrições temporais, predicados, funções e saída da consulta, o sistema vai apresentando o esquema de consulta na Janela de Esquema da Consulta. Os elementos do esquema de consulta são os seguintes:

- parte do esquema gráfico envolvido na consulta - classes e papéis do esquema gráfico que estão envolvidos na consulta são colocados no esquema de consulta;
- elipses - contém as condições definidas para a consulta, ou as propriedades que fazem parte do resultado. A elipse é ligada à classe ou papel ao qual propriedade pertence. Por exemplo, considere que foi feita a seguinte restrição sobre a propriedade *nome* da classe *Pessoa*: *nome* = *João*. Esta condição aparecerá dentro de uma elipse ligada à classe *Pessoa* no esquema de consulta. Se a propriedade *data_nasc*, também da classe *Pessoa*, fosse selecionada para aparecer no resultado, a propriedade iria parecer dentro de uma elipse com fundo cinza ligada a classe. O fundo cinza e o ponto de interrogação colocado na ligação entre a propriedade e a classe ou papel, indicam que esta propriedade fará parte do resultado;

- restrições temporais - as restrições temporais aparecem escritas em itálico no esquema de consulta e associadas às propriedades através de linhas;
- moldura da consulta - um quadrado envolve todo o gráfico de consulta e uma restrição temporal é associada ao mesmo para indicar o tempo da consulta.

6.3.6.1 Consulta com Restrições Simples e Saída de Dados

Considere a seguinte consulta: "Obter o código de todos os clientes que em algum momento tiveram o plano de saúde de código 2NB ou o plano de saúde 3AB do tipo empresarial". A seguir serão apresentados os passos para elaboração da consulta e a construção do esquema de consulta a medida que o usuário vai definindo a consulta:

- saída da consulta - o usuário seleciona a propriedade *codc* do papel *Funcionário*, ativa a janela para definição de condições e marca a opção exibir no resultado. Neste momento, o papel *Funcionário* é colocado no esquema de consulta com a propriedade *codc* dentro de uma elipse com o fundo cinza. O símbolo de interrogação aparece na linha que liga a propriedade ao papel. Estas restrições no esquema de consulta ficam dentro de um quadrado com a restrição temporal da consulta em itálico neste caso *Past/Present/Future*;
- condição - o usuário seleciona a propriedade codp da classe Plano_Saúde e defina a restrição codp = 2NB na caixa de texto. A classe Plano Saúde é incluída no esquema da consulta com a condição definida dentro de uma elipse ligada à classe. A elipse não possui fundo cinza pois esta propriedade não faz parte do resultado. Antes de definir a segunda e terceira condições, o usuário seleciona o símbolo de abre parênteses para definir o agrupamento entre as outras duas condições. As condições *cod = 2AB* e *tipo = empresarial* são definidas da mesma forma que a primeira condição, a única diferença é que depois de definir a segunda condição o operador lógico deve ser alterado para **OR**. A figura 6.21 mostra o esquema de consulta depois da definição da condição.

Cada uma das condições é colocada em uma elipse separada, de modo que as propriedades podem aparecer repetidas no esquema de consulta. As letras colocadas ao lado das elipses seguem o alfabeto e determinam a ordem em que as condições devem ser avaliadas. As condições que estão agrupadas recebem a mesma letra. A utilização destas letras é bastante importante para definir a seqüência em que as informações devem ser analisadas. Se não existir nenhum operador lógico associado à letra que determina a seqüência de avaliação das condições, o operador é **AND**. Se o operador for **OR**, ele será indicado.

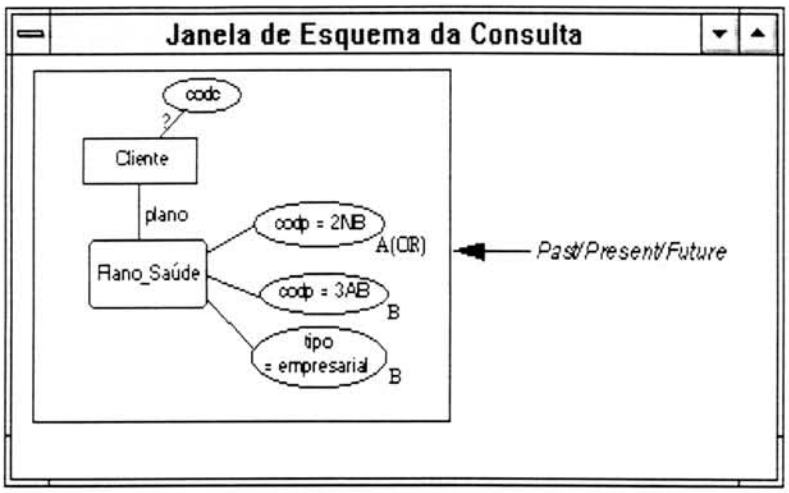

FIGURA 6.21 - Esquema da Consulta com Restrições Simples

6.3.6.2 Consulta com Operadores Temporais

Considere a seguinte consulta: "Obter o nome e o endereço de todos os laboratórios que realizaram o exame de hemograma em algum momento no passado sendo que o período da consulta é de 23/01/90 até 02/09/94". A seguir serão apresentados os passos do usuário para elaboração da consulta e a construção do esquema de consulta a medida que a consulta é definida:

- saída da consulta - o usuário seleciona a propriedade *nome* e marca a opção *exibir propriedade no resultado* na Janela de Definição de Condições. Neste momento a classe *Instituição* é colocado no esquema de consulta com a propriedade *nome* dentro de uma elipse com o fundo cinza. A propriedade endereço também é selecionada e acrescentada ao esquema de consulta. Como no exemplo anterior, as restrições no esquema de consulta ficam dentro de um quadrado com a restrição temporal da consulta em itálico, neste caso, *Period 23/04/90 .. 02/09/94*;

- condição - o usuário seleciona a propriedade *nome* da classe *Exame* e define a restrição *nome = hemograma* na caixa de texto. A classe Exame é incluída no esquema da consulta com a condição definida dentro de uma elipse ligada a classe. A figura 6.22 mostra o esquema de consulta depois da definição da condição.

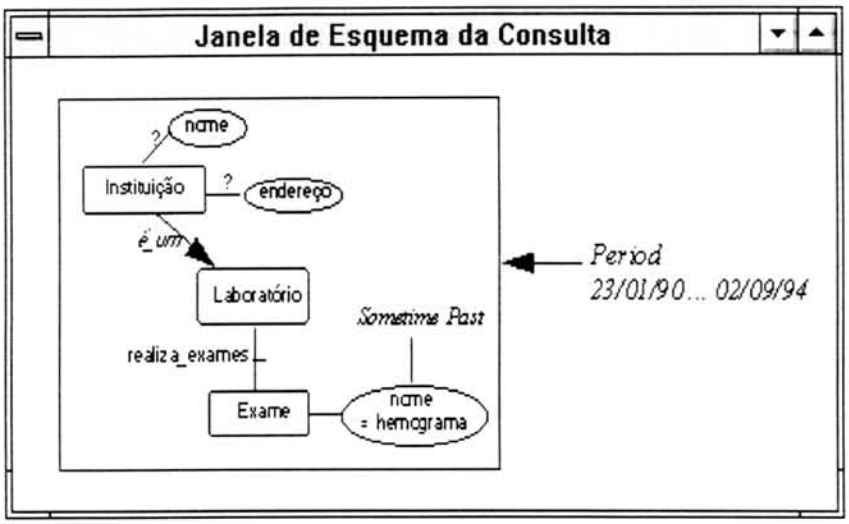

FIGURA 6.22 - Esquema da Consulta com Operadores Temporais

6.3.6.3 Consulta com Saída Temporal

Considere a seguinte consulta: "Obter os períodos em que o cliente de código 101 realizou consultas com o médico de nome João que tem especialidade em traumatologia". Os passos para a consulta são os seguintes:

- a propriedade código do cliente (*codc*) é selecionada e a condição *cod=101* é estabelecida. A propriedade e o papel *Cliente* são colocados no esquema da consulta;

- a propriedade nome é seleciona e associada ao papel *Médico.* Depois que a condição é estabelecida, a propriedade é associada ao papel Médico no esquema de consulta para que não existam ambigüidades sobre a que papel (*Cliente* ou *Médico*) a propriedade *nome* está associada no momento. A moldura do papel fica pontilhada indicando que uma das propriedades não é armazenada diretamente no papel;

- A opção saída temporal (*Period*) é seleciona na Janela de Tempo da Consulta para indicar que o período irá aparecer no resultado. A palavra *Period* é ligada ao quadrado que envolve o esquema da consulta, através de uma seta com o rótulo interrogação (?). A figura 6.23 mostra o esquema de consulta para este exemplo.

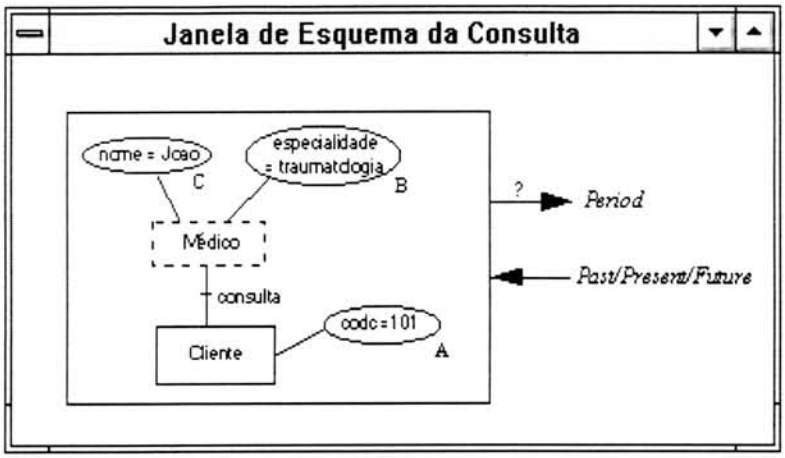

FIGURA 6.23 - Esquema da Consulta com Saída Temporal

6.3.6.4 Consulta com Predicados

Considere a seguinte consulta: "Obter o cargo do funcionário de código 101, quando seu salário era de 900 reais na data de 23/09/96". Para definição da consulta o usuário irá utilizar a janela de Predicados. Na definição do predicado a propriedade salário será selecionada e o papel Funcionário será incluído no esquema de consulta com a restrição salário = 900 associada ao mesmo. A elipse com a restrição sobre o atributo salário possui o nome do predicado (is_valid_at) e o tempo (23/09/96) ligados a propriedade por uma linha. A figura 6.24 mostra o esquema de consulta para este exemplo.

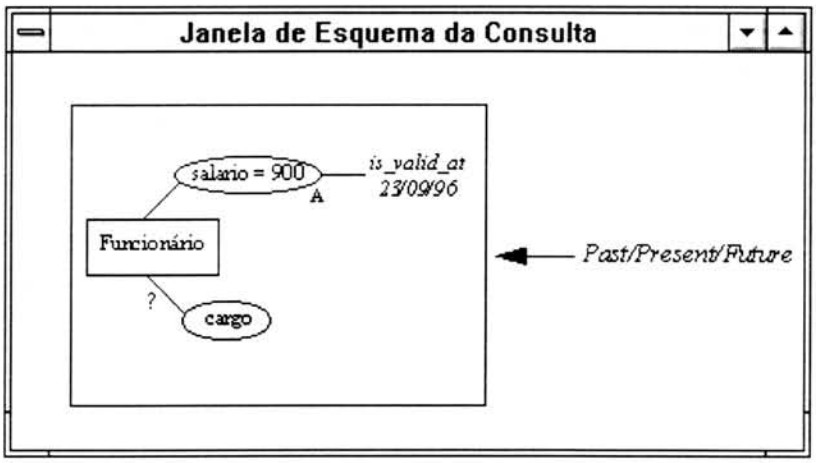

FIGURA 6.24 - Esquema da Consulta com Predicados e Funções

6.4 Consulta por Formulários

A consulta por formulários apresenta o valor das propriedades dos objetos através de um conjunto de janelas. O objetivo desta forma de elaboração da consulta é permitir a recuperação dos valores das instâncias de uma classe e/ou de um papel.

Este tipo de consulta não possibilita a definição de restrições aos valores das propriedades dos objetos, mas é possível restringir o período de tempo ou data na qual a consulta será considerada.

Quando o usuário seleciona a consulta por formulários, o sistema apresenta o esquema gráfico. É no esquema gráfico que o usuário irá selecionar a classe ou papel sobre o qual deseja realizar a consulta. Considere a seguinte consulta: "Visualizar os valores de todos os objetos que desempenham o papel funcionário". Para realizar esta consulta, o usuário deve selecionar o papel *Funcionário* no esquema gráfico. A figura 6.25 mostra a janela que será apresentada quando este papel for selecionado.

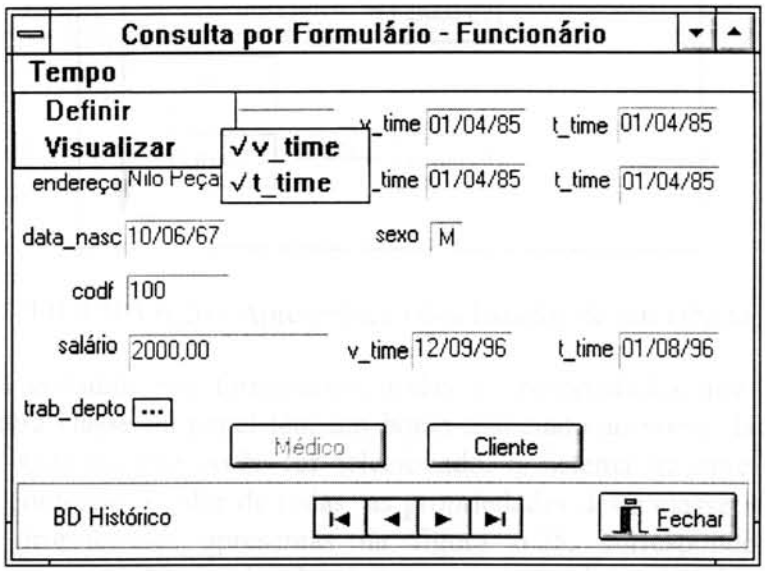

FIGURA 6.25 - Formulário de Consulta para Funcionário

A janela apresenta opções para definição do tempo da consulta e possibilidade de visualizar os valores de tempo de transação e tempo de validade associadas as propriedades dinâmicas. No exemplo apresentado na figura 6.25, aparecem os valores das propriedades da classe *Pessoa* e do papel de *Funcionário* para uma determinada instância. Na região inferior da janela são colocados o tempo no qual a consulta está sendo considerada e um conjunto de botões para navegação. Os botões de navegação permitem que o usuário navegue através das diferentes instâncias dos objetos.

No formulário aparecem ainda dois botões com os rótulos *Médico* e *Cliente*, sendo que o primeiro está desabilitado e o segundo não. À medida que o usuário vai navegando pelos diferentes estados de um mesmo objeto, estes botões podem estar habilitados ou desabilitados. Por exemplo, a figura 6.26 mostra os possíveis estados de um objeto da classe *Pessoa* para os papéis de *Funcionário*, *Médico* ou *Cliente* em relação ao eixo vertical de tempo. De acordo com a figura, ao desempenhar o papel de *Funcionário* no estado F1 o objeto possui simultaneamente, dois estados possíveis com o papel *Cliente*. Já para o estado F2 não existe nenhuma ocorrência deste objeto no papel *Cliente*. Desempenhando o papel de *Funcionário* no estado F3 o objeto apresenta simultaneamente uma ocorrência no papel Cliente. Ao longo do tempo não existe nenhuma ocorrência deste objeto no papel de *Médico*. Considere que os valores

das propriedades no estado F1 aparecem no formulário da figura 6.25. O botão com o rótulo *Cliente* está habilitado pois existem dois estados para este funcionário no papel *Cliente*. Quando o botão com o rótulo *Cliente* for selecionado, o sistema irá apresentar uma janela com todas os valores das propriedades para o papel *Cliente* no estado C1.

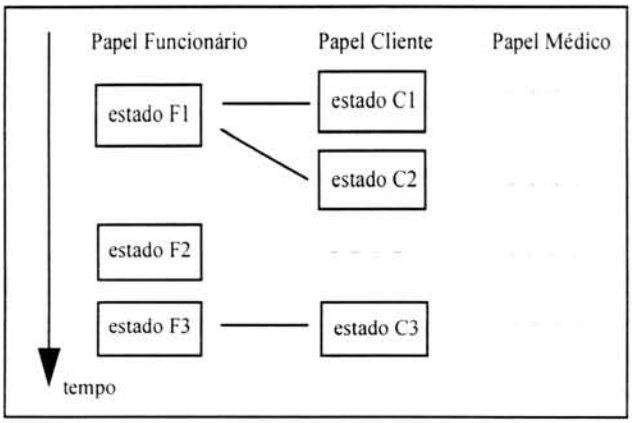

FIGURA 6.26 - Apresentação dos Estados de um Objeto

Na consulta por formulários, todas as propriedades que possuem como domínio outra classe ou papel têm um botão associado ao nome da propriedade no formulário. Quando este botão for selecionado, o sistema irá apresentar um outro formulário contendo o valor de todas as propriedades desta classe. Considere que o valor das propriedades apresentas na figura 6.25, correspondem aos valores assumidos por um determinado objeto em um estado chamado F1. Quando o botão associado à propriedade trab_depto for selecionado o sistema irá apresentar o formulário da figura 6.27. Este formulário apresenta os valores das propriedades da classe *Departamento* para o objeto no estado F1. Este formulário também possui botões para navegação que permitem ao usuário navegar por todos os departamentos em que o funcionário trabalhou enquanto desempenhava o papel *Funcionário* no estado F1.

FIGURA 6.27 - Formulário de Consulta para Departamento

A consulta por formulários não permite que se estabeleça restrições aos valores das propriedades, mas é possível definir o tempo em que a consulta será considerada. A figura 6.28 mostra a Janela de Tempo da Consulta quando ativada

neste módulo do sistema. A única diferença desta janela para aquela apresentada na consulta gráfica, é que a Janela de Tempo da Consulta para a consulta por formulários não possui a seleção da saída temporal. No exemplo da figura 6.28, foi selecionada a opção PERIOD e definidas as datas do intervalo inferior (23/04/92) e superior (10/05/97). A consulta passa a considerar os estados dos objetos no período de 23/04/92 a 10/05/97. Considere que a data atual seja 12/07/96, a apresentação deste intervalo no eixo do tempo corresponde a uma linha colocada sobre a parte do eixo que corresponde ao passado (linha contínua) e ao futuro (linha pontilhada). Pois a data inferior do período é menor que a data atual (12/07/96) e a data superior é maior que (12/07/96).

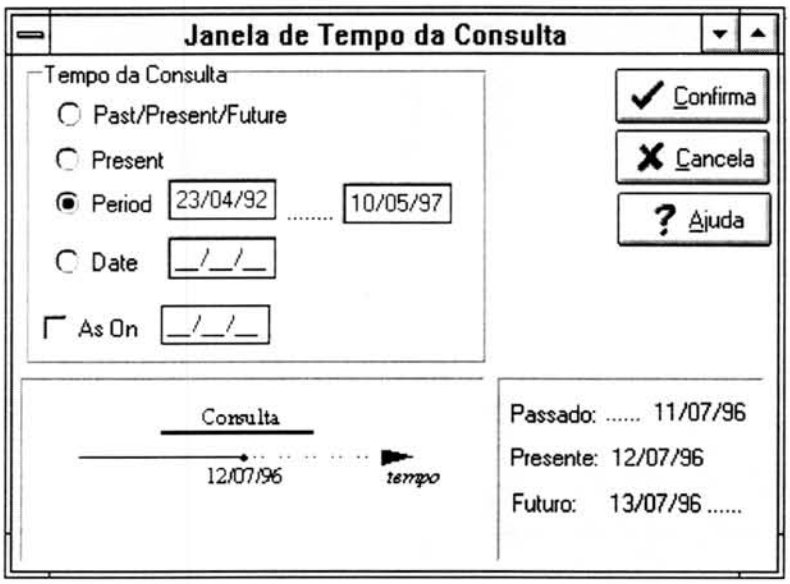

FIGURA 6.28 - Definição de Restrição Temporal

6.6 Representação Textual e Gráfica da Consulta

A consulta também pode ser elaborada textualmente através da linguagem textual de consulta do modelo TF-ORM A linguagem textual é apresentada no capítulo 3 e pode ser utilizada por usuários mais familiarizados com a sintaxe e semântica desta linguagem. Considere a seguinte consulta: "Obter o código dos clientes que possuíam o plano de saúde de código EN5 na data de 21/08/96". A figura 6.29 apresenta a **Janela de Consulta Textual** com a consulta anterior na linguagem textual de consulta do TF-ORM. No menu da janela principal o usuário pode executar a consulta, selecionar a forma de apresentação do resultado, e ainda selecionar as operações mais comuns para edição do texto (copiar, desfazer, localizar, entre outras).

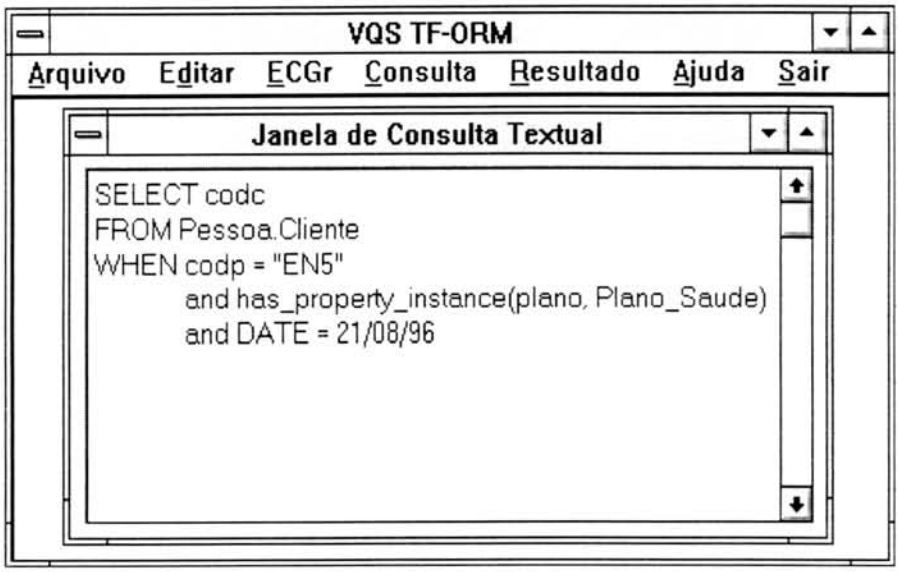

FIGURA 6.29 - Janela de Consulta Textual

Como a consulta gráfica é equivalente à consulta textual, o sistema permite que o usuário visualize a mesma consulta na forma gráfica ou textual. Mas existem algumas regras para que o usuário possa alternar entre as formas de representação:

- a consulta deve possuir no mínimo uma saída - o usuário deve ter definido uma saída para a consulta que pode ser temporal ou uma saída sobre dados;
- as condições da cláusula de busca devem estar completas - por exemplo, o usuário selecionou um operador temporal, mas não definiu um argumento a ser associado com a restrição temporal. Neste caso a conversão para a consulta textual não pode ser feita. Esta regra também é válida para o caso do usuário estar elaborando a consulta textualmente e desejar passar para a consulta gráfica;

Considere que o usuário esteja realizando a consulta textual da figura 6.29. Quando a opção consulta gráfica for selecionada no menu principal, a Janela de Consulta Textual irá desaparecer e o sistema apresenta as janelas de esquema gráfico, acessórios e esquema da consulta. Se nenhum esquema de trabalho foi definido, o sistema apresenta o esquema completo. Neste exemplo, o esquema gráfico irá conter as propriedades *codc*, *cop* e *plano* em negrito. Se a Janela de Definição de Condições for ativada para a propriedade *codc* a opção *exibir propriedade no resultado* estará marcada, pois a propriedade faz parte do resultado (cláusula SELECT da consulta textual). A Janela de Tempo da Consulta possui a opção DATE selecionada e data de 21/08/96 associada a esta opção. A figura 6.30 apresenta a Janela de Esquema da Consulta para a consulta textual da figura 6.29.

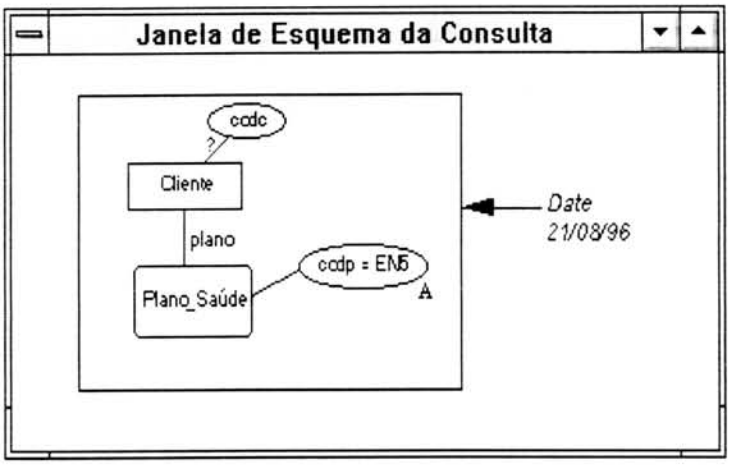

FIGURA 6.30 - Janela de Esquema da Consulta

6.6 Apresentação dos Resultados

A recuperação de informações temporais possui três particularidades:

- o grande número de informações do resultado - como o banco de dados armazena todo o histórico dos dados, o volume de informações que pode ser recuperado é maior que em consultas a bancos de dados não temporais;
- os diferentes estados que um mesmo objeto pode apresentar - o valor das propriedades pode mudar de valor com o tempo, o que produz diferentes estados para os objetos;
- a saída do tipo temporal ou mista - o resultado da consulta pode conter datas e/ou períodos.

O sistema apresenta quatro formas para apresentação do resultado que podem ser selecionadas pelo usuário: formulário, gráfico, tabela e visual.

A saída por formulário e a saída por gráfico devem ser utilizadas em consultas que apresentam muitas propriedades envolvidas no resultado, ou que contenham no resultado propriedades de diferentes classes ou papéis. Considere a seguinte consulta: "Obter o nome e o endereço de todos os laboratórios e as informações sobre os exames que estes laboratórios realizam". O menu principal apresenta as três possibilidades de apresentação do resultado. Quando a consulta for executada, o sistema irá apresentar o resultado da consulta na forma de apresentação que estiver selecionada no menu principal. Para a consulta anterior, por exemplo, o sistema irá selecionar a opção apresentação do resultado por formulário, mas o usuário pode escolher uma outra forma de apresentação se desejar (tabela ou gráfico). A figura 6.31 apresenta o resultado da consulta se a opção formulário estivesse selecionada no momento da execução.

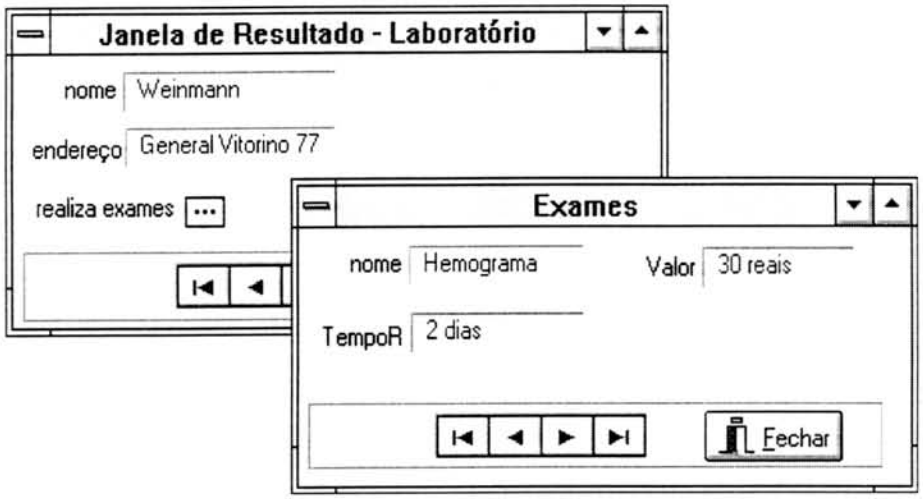

FIGURA 6.31 - Apresentação do Resultado na Forma de Formulários

A apresentação por formulário não pode ser utilizada apenas em um caso, quando a saída for temporal. Se nenhuma restrição sobre valores de dados for definida na consulta, as opções de apresentação do resultado por formulário e gráfico estarão desabilitadas.

Qualquer tipo de resultado pode ser apresentado na forma de tabela. A consulta anterior apresentada através de uma tabela é mostrada na figura 6.32. Nesta forma de apresentação as propriedades do resultado são colocadas lado a lado como uma tabela do modelo relacional e os valores possíveis para cada uma das propriedades é apresentado dentro da coluna indicada.

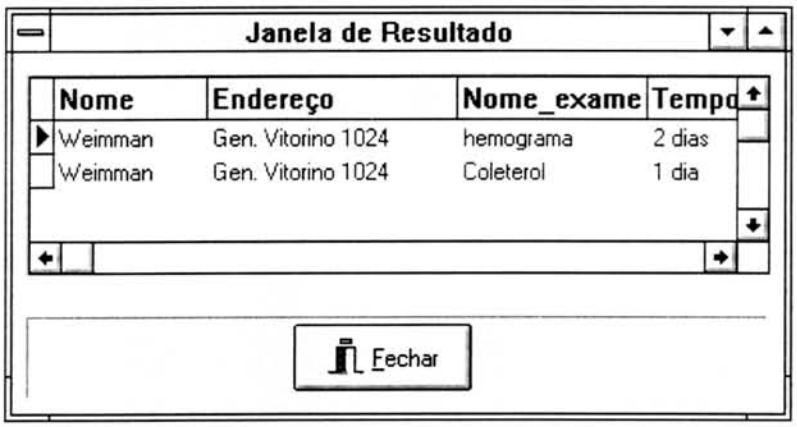

FIGURA 6.32 - Apresentação do Resultado na forma de Tabela

A forma de apresentação gráfica para o resultado da consulta, mostra as informações do resultado diretamente no esquema gráfico. Considere a seguinte consulta: "Obter o código, o salário e o cargo de todos os funcionário e os dados referentes aos departamentos em que estes funcionários trabalharam". A figura 6.33 mostra a saída gráfica para esta consulta. O botão de navegação presente na janela permite que o usuário navegue através do esquema a visualize todos os valores do resultado.

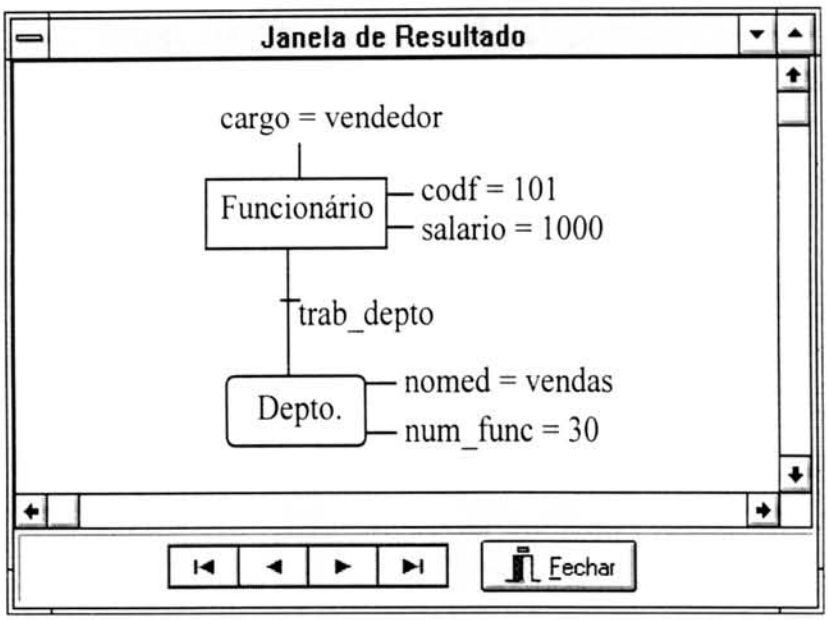

FIGURA 6.33 - Apresentação do Resultado na Forma Gráfica

As consultas que com saída temporal ou mista podem ter seus resultados apresentados na forma visual. A representação visual consiste na apresentação do resultado da consulta sobre o eixo do tempo. Considere a seguinte consulta: "Obter os períodos em que o funcionário João trabalhou no departamento de vendas". A figura 6.34 mostra a apresentação dos resultados no eixo do tempo.

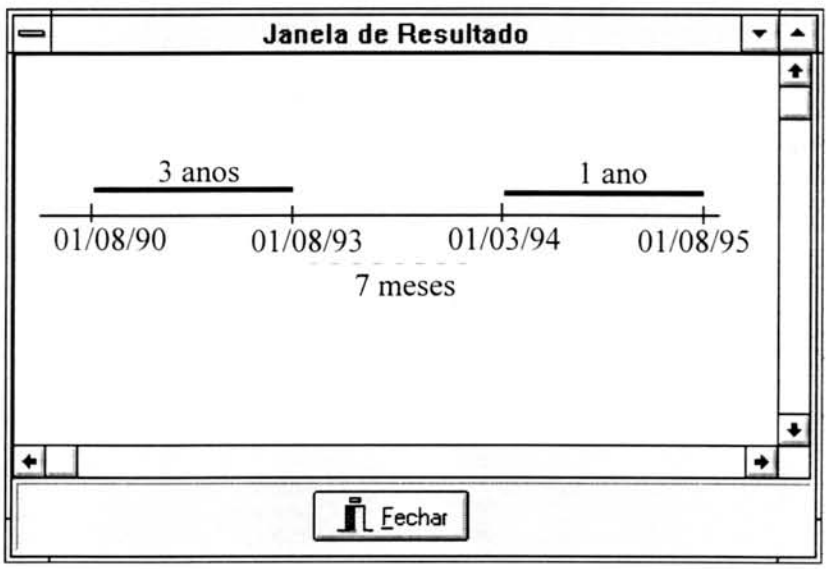

FIGURA 6.34 - Apresentação do Resultado na Forma Visual

7 Mapeamento entre Modelos

A utilização de mais de um modelo no mesmo ambiente torna necessária a definição de mapeamentos adequados entre estes modelos. Este capítulo apresenta os mapeamentos utilizados para os modelos de dados e para os modelos de consulta. O mapeamento entre modelos de dados determina como o modelo TF-ORM será armazenado em um banco relacional e como as informações armazenados neste banco de dados serão mapeadas novamente para o modelo para que possam ser utilizadas pelos usuários. O mapeamento entre os modelos de consulta determina como a consulta realizada na linguagem gráfica de consulta do modelo TF-ORM será traduzida para a linguagem textual e posteriormente para a linguagem de consulta SQL. Outro mapeamento definido para o modelo de consulta faz a tradução da consulta textual do TF-ORM para a consulta gráfica.

7.1 Mapeamento entre Modelos de Dados

Como o ambiente de recuperação de informações temporais utiliza dois modelos diferentes para dados, existe a necessidade de se estabelecer um mapeamento para que se possa passar do modelo de dados interno para o modelo de dados externo e vice-versa. O primeiro mapeamento determina como o modelo TF-ORM será armazenado em um banco de dados relacional. O segundo mapeamento determina como é possível reconstruir o ECGr do modelo TF-ORM a partir das informações armazenadas no banco de dados.

7.1.1 Mapeamento do Modelo TF-ORM para o Banco de Dados Relacional

Este ambiente pode utilizar qualquer banco de dados relacional que utilize o padrão SQL/ANSI para recuperação de informações. Nesta implementação foi utilizado o banco de dados Watcom. Para armazenar as informações do modelo TF-ORM neste banco de dados relacional, foi utilizado um mapeamento baseado em [CAV 95]. O mapeamento determina como os conceitos de orientação a objetos, papéis, mensagens, regras e tempo serão implementados em um banco de dados relacional. Este mapeamento é necessário pois o ambiente possui dois modelos de dados (um interno e outro externo). O modelo de dados externo será utilizado pelo usuário e corresponde ao modelo TF-ORM. O modelo de dados interno está relacionado aos aspectos de implementação e corresponde ao modelo relacional. O resultado deste mapeamento irá definir como as informações do modelo de dados externo serão organizadas no modelo de dados interno.

O mapeamento de um modelo orientado a objetos para um banco de dados relacional pode ser realizado em três diferentes níveis, de acordo com os recursos apresentados pelo sistema de banco de dados [CAV 95]:

- nível 1 - o mapeamento permite a implementação do modelo em SGBDs que não possuem um sistema de regras e mensagens (que devem ser implementadas pelo programa de aplicação). A única exigência é que o banco de dados apresente tipos de dados temporais, como, por exemplo, data;
- nível 2 - permite a implementação das regras de transição de estado e integridade do modelo, devendo o SGBD possuir um sistema de regras;
- nível 3 - permite a implementação completa do modelo considerando também o sistema de mensagens.

Como o banco de dados utilizado não possui um sistema de regras e mensagens, a implementação foi realizada apenas no nível 1. Este mapeamento poderia ser utilizado com outros bancos de dados relacionais, como, por exemplo, Access e Oracle. O banco de dados Watcom foi escolhido pelas características apresentadas no capítulo 5, como, por exemplo, possibilitar consultas no padrão SQL/ANSI e apresentar um ODBC que suporta uma consulta com muitos caracteres.

O mapeamento de um esquema TF-ORM para o Watcom é realizado mapeando cada classe, subclasse, papel, propriedade dinâmica e estado para uma tabela no BD relacional. Uma característica importante deste mapeamento é a utilização de intervalos no tempo para propriedades dinâmicas. O modelo TF-ORM utiliza pontos no tempo, associando ao valor da propriedade apenas o tempo de validade inicial e o tempo em que a transação de definição deste valor foi realizada. A utilização de intervalos no tempo ao invés de pontos no tempo possibilita uma otimização no processo de mapeamento da consulta.

Com a utilização de pontos no tempo, para descobrir se uma tupla da tabela representa uma informação válida no momento considerado, seria necessário pesquisar a data de validade inicial da próxima tupla, para saber se esta tupla é valida ou não. Se cada atributo dinâmico possuir a data de validade inicial e final associada, a pesquisa de validade da tupla fica restrita à verificação das datas nos atributos *v_timei* (tempo de validade inicial) e *v_timef* (tempo de validade final) da mesma tupla. Cada propriedade dinâmica possui portanto, quatro informações temporais associadas:

- tempo de validade inicial (*v_timei*) - indica o instante de validade da informação;
- tempo de validade final (*v_timef*) - indica o instante em que o dado deixou de ser válido;
- tempo de transação inicial (*t_timei*) - corresponde ao instante de tempo em que a informação foi inserida no banco de dados;
- tempo de transação final (*t_timef*) - corresponde ao instante de tempo da próxima transação realizada sobre o dado;

As propriedades estáticas do papel básico são mapeadas para a tabela base de classe (o nome da tabela será igual ao nome da classe), que contém o identificador de classe *oId* e as propriedades estáticas, pe_i:

NomeClasse (*oId*, pe_1, ...pe_n);

No exemplo apresentado no capítulo 4, a classe PlanoSaude possui a propriedade estática *codp* (código do plano de saúde) que será mapeada para a tabela PlanoSaude no esquema relacional, com o atributo *oId* como chave primária:

PlanoSaude (*oId* , *codp*);

As propriedades dinâmicas pré-definidas são mapeadas para outra tabela que está relacionada à tabela base de classe através do atributo *oId* e possui os atributos *object_instance* (armazena o tempo de início de vida de um objeto), *end_object* (armazena o instante de tempo em que uma instância deixou de existir) e os valores de tempo de transação e tempo de validade associados:

NomeClasse_dynamic(*oId, object_instance, v_timei, v_timef, t_timei, t_timef, end_object*);

A tabela do banco de dados que armazena as propriedades dinâmicas pré-definidas da classe PlanoSaude é:

PlanoSaude_dynamic(*oId, object_instance, v_timei, v_timef, t_timei, t_timef, end_object*);

Esta tabela deverá ser relacionada a tabela base de classe (neste caso, a tabela **PlanoSaude**) pelo atributo *oId*, definido como chave estrangeira.

As propriedades dinâmicas, que variam com o tempo, também são mapeadas para tabelas separadas. Cada propriedade dinâmica possui uma tabela com a seguinte estrutura:

NomeClasse_Prop (oId, valor_pd, v_timei, v_timef, t_timei, t_timef),

onde, *oId* é o atributo que relaciona a tabela de propriedades dinâmicas de classe a tabela base de classe (tabela PlanoSaude), *valor_pd* é o valor da propriedade dinâmica, *v_timei* é o tempo de validade inicial, *v_timef* é o tempo de validade final, *t_timei* é o tempo de transação inicial e *t_timef* é o tempo de transação final.

No exemplo, a propriedade dinâmica *valor* da classe *PlanoSaude* será mapeada para a tabela PlanoSaude_valor:

PlanoSaude_valor(*oId, valor, v_timei, v_timef, t_timei, t_timef*);

Os estados são armazenados em tabelas separadas pois variam ao longo do tempo. A tabela de estado de classe possui a seguinte estrutura:

NomeClasse_state(*oId, estado, v_timei, v_timef, t_timei, t_timef*);

O mapeamento dos papéis é análogo ao mapeamento das classes do modelo TF-ORM, a única diferença é que além do atributo *rId* que identifica o papel, a tabela base do papel possui também o atributo *oId* que irá relacioná-la a tabela de classe correspondente. As tabelas resultantes do mapeamento dos papéis do modelo TF-ORM são apresentadas a seguir:

NomePapel (*oId*, *rId*, pe_1,...pe_n);

NomePapel_dynamic (*rId, role_instance ,v_timei ,v_timef, t_timei, t_timef, end_role*);

NomePapel_Prop (*rId, valor_pd, v_timei, v_timef, t_timei, t_timef*);

NomePapel_state (*rId, estado,v_timei, v_timef, t_timei, t_timef*);

No modelo TF-ORM uma propriedade pode apresentar como domínio uma classe, o que representa na realidade uma associação ou relacionamento entre as classes envolvidas. Para representar esta situação no modelo relacional deve-se considerar a cardinalidade do relacionamento e a natureza da propriedade em questão:

- relacionamento 1:1 e propriedade estática - neste caso o atributo que representa a classe relacionada será adicionado à tabela base da classe juntamente com as demais propriedades estáticas. O atributo deve ser definido como chave estrangeira, referenciando a tabela que representa a classe associada;
- relacionamento 1:N e propriedade estática - a representação deste caso é semelhante à anterior, porém o atributo deve ser adicionado à tabela que representa a classe que se relaciona com no máximo uma instância da outra classe;
- relacionamento N:N e propriedade estática - esta situação exige a definição de uma nova tabela para representar o relacionamento. A tabela deve conter os identificadores das classes envolvidas, os quais constituirão a chave primária da tabela. Os outros atributos devem ser definidos como chave estrangeira fazendo referência às tabelas que representam estas classes.
- relacionamento 1:1 e propriedade dinâmica - esta situação exige a criação de uma nova tabela, assim como os demais casos em que a propriedade é dinâmica. Esta tabela deve conter os identificadores das classes e os tempos de validade e transação. Os atributos que representam as classes relacionadas devem ser definidos como chaves estrangeiras referenciando as tabelas correspondentes.
- relacionamento 1:N e propriedade dinâmica - o mapeamento é bastante semelhante ao anterior, a diferença está na definição da chave primária que deve ser composta pelo atributo que representa o identificador da classe que se relaciona a, no máximo, uma instância da outra classe e pelo atributo que representa o instante inicial do período de validade do relacionamento.
- relacionamento N:N e propriedade dinâmica - a nova tabela será composta pelos atributos que representam as classes relacionadas mais os tempos de validade e transação. A chave primária desta tabela deve ser composta pelos atributos que identificam as classes envolvidas no relacionamento e pelo atributo que representa o instante inicial do período de validade do relacionamento. Os atributos que identificam as classes relacionadas também devem ser definidos como chave estrangeira fazendo referência às tabelas base correspondentes.

Na criação do esquema do banco de dados, o sistema gerenciador de banco de dados gera um conjunto de arquivos que contém a descrição das tabelas. A presença destes arquivos é bastante importante no processo de tradução da consulta, pois nos permite descobrir em que tabelas do banco de dados as classes, papéis e propriedades do modelo estão armazenadas.

O mapeamento completo do estudo de caso no modelo TF-ORM para o banco de dados relacional Watcom encontra-se no Anexo 2.

7.1.2 Esquema Conceitual Relacional e ECGr

A seção anterior apresentou o mapeamento do modelo TF-ORM para o modelo relacional. O mapeamento inverso também é necessário neste ambiente pois as consultas são realizadas sobre o esquema do modelo TF-ORM.

Para o mapeamento do esquema relacional para o esquema TF-ORM, o sistema deve obter a descrição do esquema no banco de dados. O banco de dados Watcom, por exemplo, mantém um conjunto de arquivos que armazenam o esquema conceitual: SYS.SYSTABLE (armazena informações sobre as tabelas do banco de dados), SYS.SYSCOLUMN (armazena o nome de todos os atributos de cada uma das tabelas), SYS.SYSFOREIGNKEYS (armazena o nome da tabela principal e da tabela estrangeira, nome da chave estrangeira entre outras). A partir da análise das informações armazenadas nestes arquivos e de um conjunto de regras definidas para o mapeamento do modelo relacional para o modelo TF-ORM, o sistema vai criando o esquema no modelo TF-ORM.

O esquema conceitual no modelo TF-ORM vai sendo armazenado em uma lista onde cada nodo contém informações sobre classes, subclasses ou papéis. Além das informações sobre o esquema conceitual, são armazenadas informações sobre a localização física destes elementos no esquema gráfico (ECGr). As informações armazenadas em um nodo da lista são as seguintes:

- tipo do elemento - os valores possíveis são: classe, subclasse ou papel;
- coordenadas - posição que o elemento ocupa no esquema gráfico;
- visível ou não no ECGr - se o elemento está visível ou não no esquema gráfico;
- lista com as informações sobre as propriedades: nome da propriedade, tipo da propriedade (estática ou dinâmica), domínio (este campo será preenchido apenas se a propriedade possuir como domínio uma classe ou papel), coordenadas e situação (se o atributo foi selecionado ou não no esquema gráfico).

Para criar esta lista a partir das informações do esquema relacional são utilizadas as seguintes regras:

- regra 1 - se a tabela possui o atributo *oId* como atributo chave e este atributo não é chave estrangeira de outra tabela, o nome da tabela representa o nome de uma classe no modelo TF-ORM. Neste caso a letra C é armazenada no campo da lista que armazena o tipo do elemento;
- regra 2 - se a segunda condição da regra 1 não é verdadeira, significa que a tabela é uma subclasse no TF-ORM. Neste caso a letra S é armazenada no campo da lista que armazena o tipo do elemento e o nome da classe ao qual ela pertence. O nome da classe é o nome da tabela que contém a chave primária;
- regra 3 - se a tabela possui o atributo *rId* como atributo chave, o nome da tabela representa o nome de um papel no modelo TF-ORM. Neste caso a letra P é armazenada no campo da lista que armazena o tipo do elemento. Este campo armazena também o nome do classe ao qual o papel está associado. Esta informação é obtida verificando-se o nome da tabela que contém o atributo *oId* que é chave estrangeira nesta tabela. O nome da tabela é armazenada juntamente com a letra P;

- regra 4 - para cada um dos nomes de classes, subclasses ou papéis obtidos no item anterior, o sistema encontra o nome dos outros atributos que estão armazenados nesta tabela e coloca estas informações na lista de propriedades. Se o atributo faz parte das tabelas selecionadas pelas regras 1,2 ou 3, isto significa que o atributo corresponde a uma propriedade estática.
- regra 5 - para cada um dos nomes de tabelas selecionadas pelas regras 1,2 ou 3 o sistema verifica se existe uma outra tabela com este mesma cadeia de caracteres. Por exemplo considere que a tabela *Pessoa* tenha sido selecionada pela regra 1, a tabela *Pessoa_nome* será selecionada pela regra 5, pois possui a cadeia de caracteres *Pessoa*. Isto significa que *nome* é uma propriedade dinâmica da classe *Pessoa*, pois as propriedades dinâmicas são armazenadas em tabelas separadas que contém o nome da classe, subclasse, ou papel a que pertencem como prefixo e o nome da propriedade como sufixo. Se o atributo nome, por exemplo, for chave estrangeira de outra tabela que tem o atributo *oId* ou *rId* como chave, o nome da tabela será incluído como domínio. Isto significa que esta propriedade possui como domínio uma outra classe, subclasse ou papel.

As informações referentes a localização dos elementos do esquema gráfico são obtidas de um arquivo texto que armazenada a localização de todos os objetos.

7.2 Mapeamento entre Modelos de Consulta

A consulta sobre os dados armazenados no modelo TF-ORM pode ser realizada através da linguagem textual ou gráfica do modelo e mapeada para a linguagem SQL. O modelo de consulta externo possui duas linguagens (gráfica e textual) e o modelo de consulta interno a linguagem SQL. Neste caso o mapeamento entre os modelos de consulta não existe apenas entre o modelo externo e interno, mas também entre as duas linguagens de consulta do modelo externo. É possível realizar a mesma consulta na linguagem textual ou gráfica por isso, o mapeamento entre estas linguagens deve ser definido nas duas direções.

7.2.1 Mapeamento da Consulta Gráfica para a Consulta Textual do TF-ORM

Para que a consulta realizada através da linguagem gráfica possa ser executada no banco de dados, ela deve ser traduzida para a linguagem SQL. A consulta visual é traduzida, primeiramente, para a linguagem textual de consulta do TF-ORM. Em um segundo passo, a consulta textual é traduzida para a linguagem SQL. A consulta visual não é mapeada diretamente para a linguagem SQL por dois motivos:

- como uma mesma consulta pode ser elaborada através da linguagem gráfica e textual, já existe a necessidade de um mapeamento entre as duas linguagens de consulta; e
- a consulta elaborada através da linguagem textual do TF-ORM deve ser traduzida para a linguagem SQL. Assim, já existe a necessidade do mapeamento entre as duas linguagens textuais de consulta.

A tradução entre as consultas gráfica e textual, é feita através de um conjunto de regras de mapeamento que possibilitam a tradução das ações realizadas sobre os símbolos gráficos em argumentos para as cláusulas da linguagem textual de consulta.

A consulta pode ser elaborada em qualquer ordem, ou seja, o usuário não precisa necessariamente definir os elementos que irão participar da cláusula de especificação, a seguir da cláusula de identificação, depois da cláusula de busca e opcionalmente da cláusula de instante temporal.

A cláusula de especificação determina o conjunto de informações que farão parte do resultado da consulta. Esta cláusula pode conter nomes de propriedades e palavras especiais utilizadas para recuperar informações de tempo, como data e período e funções de agregação aplicadas sobre as propriedades. A figura 7.1 mostra os módulos da interface gráfica envolvidos nesta etapa do mapeamento. A cláusula de especificação será composta por:

- elementos selecionados no ECGr que tiverem o campo *exibir propriedade no resultado* marcado com um 'x' no módulo de definição do predicado, ou um sinal de interrogação colocado na caixa de texto no ECGr;
- propriedades e uma ou mais funções de agregação associadas na Janela de Definição de Condições ou diretamente no ECGr;
- saída temporal, se houver - na Janela de Tempo da Consulta uma das seguintes palavras: PERIOD, DATE, TRANSACTION DATE ou TRANSACTION PERIOD podem ser escolhidas para indicar a saída temporal.

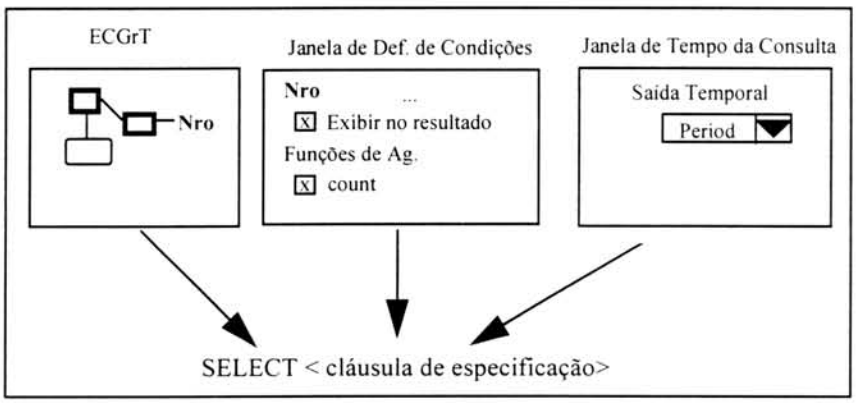

FIGURA 7.1 - Mapeamento da cláusula de Especificação

A cláusula de identificação determina as classes, subclasses e papéis envolvidos na consulta. À medida que o usuário seleciona uma determinada propriedade, o sistema verifica a que classe ou papel esta propriedade pertence. Esta informação é obtida através de uma pesquisa à lista que contém o esquema conceitual no modelo TF-ORM. A classe ou papel a que a propriedade pertence é destacada no esquema gráfico e incluída no lista de elementos da cláusula FROM. A figura 7.2 mostra o módulo da interface envolvido nesta etapa do mapeamento.

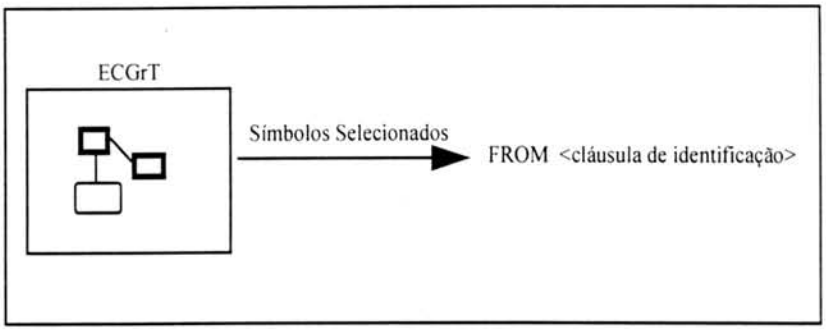

FIGURA 7.2 - Mapeamento da Cláusula de Identificação

A cláusula de busca determina as restrições a serem aplicadas às propriedades do esquema. Na linguagem textual de consulta do modelo as palavras **WHERE** e **WHEN** são utilizadas para determinar os elementos da cláusula de busca. A cadeia de caracteres da cláusula de busca é formada pelas propriedades selecionadas no ECGr, pelas restrições sobre os valores das mesmas (Janela de Definição de Condições), tempo em que estes valores devem ser considerados (Janela de Operadores Temporais), predicados (Janela de Predicados) e Funções (Janela de Funções). A consulta equivalente na linguagem textual recebe a palavra **WHERE** se nenhuma restrição temporal for estabelecida para as propriedades do esquema e a opção *Present* for selecionada na Janela de Tempo da Consulta. Caso a opção para tempo da consulta seja diferente de *Present* a cláusula de busca será composta pela palavra **WHEN**. Cada uma das diferentes opções para o tempo da consulta existentes na Janela de Tempo da Consulta leva a uma construção da cláusula de busca:

- se a opção *Present* for selecionada, a cláusula de busca contém a palavra **WHERE** e nenhuma restrição temporal;
- se a opção *Past/Present/Future* for selecionada, a cláusula de busca contém a palavra **WHEN** e opcionalmente operadores temporais;
- se a opção *Period* for selecionada, a cláusula de busca contém a palavra **WHEN**, a restrição estabelecida para o período da consulta e opcionalmente operadores temporais;
- se a opção *Date* for selecionada, a cláusula de busca contém a palavra **WHEN**, a restrição estabelecida para a data da consulta e opcionalmente operadores temporais;

A figura 7.3 mostra as janelas e elementos da interface envolvidas no mapeamento da cláusula de busca. A cláusula de busca também pode conter operadores temporais que determinam o tempo em que uma ou mais condições devem ser verificadas. Os operadores temporais só podem ser utilizados com propriedades dinâmicas. Para saber o tipo da propriedade, o sistema consulta a lista que contém o esquema conceitual do TF-ORM. Se a propriedade selecionada no esquema gráfico possuir como domínio uma outra classe ou papel, o sistema inclui o predicado has_property_instance na cláusula de busca. A inclusão deste predicado na cláusula de busca indica que deve existir uma instância da classe ou papel associada à propriedade.

A opção AS ON da Janela de Tempo da Consulta pode ser utilizada com qualquer uma das opções (*Present*, *Past/Present/Future*, *Date*, *Period*). Esta cláusula é incluída da linguagem textual juntamente com a data definida.

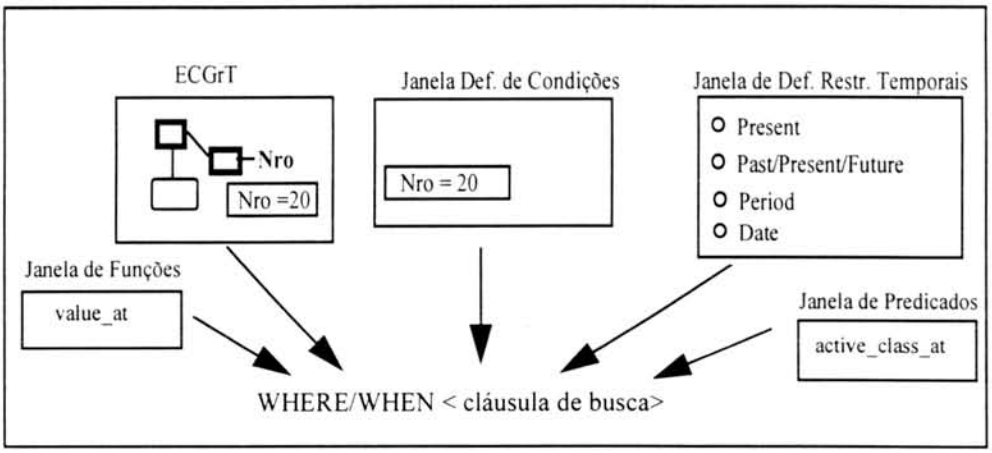

FIGURA 7.3 - Mapeamento da Cláusula de Busca

7.2.2 Mapeamento da Consulta Textual para a Consulta Gráfica

O mapeamento da consulta textual para a consulta gráfica é exatamente o inverso do apresentado na seção anterior. A seguir serão apresentadas os passos para mapear a consulta textual para consulta gráfica:

- cláusula SELECT - as propriedades encontradas na cláusula SELECT serão colocadas em negrito no esquema gráfico. Se a Janela de Definição de Condições for ativada para uma destas propriedades, a *opção exibir propriedade no resultado* aparecerá marcada. As funções de agregação associadas a uma determinada propriedade na cláusula SELECT, também estarão selecionadas na Janela de Definição de Condições. Caso a cláusula SELECT contenha alguma das palavras temporais (PERIOD, DATE, TRNASACTION_DATE e TRANSACTION_PERIOD) a opção correspondente será selecionada na Janela de Tempo da Consulta. As propriedades da cláusula SELECT são colocadas na Janela de Esquema da Consulta dentro de uma elipse com o fundo cinza e associadas a classe ou papel a qual pertencem;
- cláusula FROM - as classes, subclasses e papéis da cláusula FROM serão colocadas em negrito no esquema de gráfico e incluídas do esquema de consulta.
- cláusula WHERE/WHEN - as restrições aplicadas às propriedades são colocadas no esquema de consulta. O sistema coloca a propriedade e a restrição associada em uma elipse e associa este elemento a classe ou papel ao qual a propriedade pertence. Os operadores temporais encontrados nesta etapa são colocadas em itálico no esquema de consulta e ligados por linhas às elipses que contêm as condições a que o operador está associado na consulta textual. As condições que vão sendo analisadas na cláusula de busca recebem uma letra próximo a elipse no esquema de consulta. A primeira condição, por exemplo, recebe a letra **A**, a segunda a letra **B** e assim por diante. As condições que estiverem agrupadas por parênteses receberão a mesma letra. Se a cláusula de busca tiver a palavra

WHERE a opção *Present* será selecionada na Janela de Tempo da Consulta. Se a cláusula de busca tiver apenas a palavra WHEN a opção selecionada será a opção *Present/Past/Future*. Caso a cláusula de busca tenha também as palavras PERIOD e DATE a opção correspondente será selecionada na Janela de Tempo da Consulta. O sistema coloca no esquema de consulta a restrição temporal em itálico associada ao quadrado que envolve toda a consulta;

- cláusula AS ON - se a consulta tiver a cláusula de instante temporal AS ON, o sistema seleciona a opção correspondente na Janela de Tempo da Consulta.

7.2.3 Mapeamento da Consulta Textual do TF-ORM para SQL

O usuário pode utilizar diretamente a linguagem de consulta do modelo TF-ORM para a formulação da consulta. Nos dois casos (gráfica e textual), o sistema faz a tradução desta consulta para a linguagem SQL, para que a consulta seja executada no banco de dados Watcom. O mapeamento da linguagem de consulta TF-ORM para a linguagem SQL é realizado em duas etapas: (a) definição das tabelas, relacionamentos e restrições sobre dados; e (b) definição das restrições temporais.

7.2.3.1 Definição das Tabelas, Relacionamentos e Restrições Sobre Dados

O objetivo desta etapa do mapeamento é identificar em que tabelas do banco de dados as propriedades estão armazenadas, como estas tabelas se relacionam e quais as restrições sobre dados que podem ser colocadas diretamente na consulta SQL.

O sistema recebe uma consulta na linguagem TF-ORM e inicia a análise pela cláusula **SELECT**. Nesta primeira etapa são considerados apenas as propriedades, sendo as informações temporais tratadas posteriormente. O sistema deve descobrir em que tabela do banco de dados está armazenado a propriedade da cláusula **SELECT** da consulta TF-ORM. Esta informação pode ser obtida através de uma consulta com a descrição do esquema, mantidos pelo banco de dados Watcom. Estes arquivos armazenam a descrição do esquema do BD. Os arquivos utilizados são: SYS.SYSTABLE (armazena informações sobre as tabelas do banco de dados), SYS.SYSCOLUMN (armazena o nome de todos os atributos de cada uma das tabelas), SYS.SYSFOREIGNHKEYS (armazena o nome da tabela principal e da tabela estrangeira e o nome da chave estrangeira).

Como apresentado anteriormente, as propriedades do modelo TF-ORM são mapeadas com o mesmo nome para o banco de dados relacional, portanto todas as propriedades que fazem parte da consulta TF-ORM estarão na consulta SQL. As propriedades dinâmicas são mapeadas para tabelas do banco de dados relacional que devem ser incluídas na cláusula **FROM** da consulta SQL.

Considere a seguinte consulta na linguagem TF-ORM:

SELECT salario
FROM Pessoa.Funcionario
WHERE nome = "Pedro" and has_property_instance(trab_depto, Depto)
and active_role(rId3) and is_valid(nomed, "Vendas")

A tradução para a consulta SQL começa com a cláusula **SELECT** da consulta SQL, recebendo a propriedade que está na cláusula **SELECT** da consulta TF-ORM e a cláusula **FROM** SQL, recebendo o nome da tabela onde o atributo está armazenado. Este passo é necessário porque as propriedades dinâmicas do modelo TF-ORM são armazenadas em tabelas separadas e a cláusula **FROM** da linguagem de consulta do modelo possui apenas os nomes das classes e papéis. Cada classe e papel possui uma tabela no banco de dados com o mesmo nome utilizado no modelo TF-ORM.

No banco de dados o atributo *salario* foi armazenado na tabela *Funcionario_salario* que deve ser incluída na cláusula **FROM** do SQL. Esta informação é obtida fazendo-se uma consulta ao arquivo SYS.SYSCOLUMN, que indica a que tabela um determinado atributo pertence. Além desta tabela devem ser incluídas a tabela *Pessoa* e a tabela *Funcionario* que armazenam, respectivamente, o identificador da classe e as propriedades estáticas e o identificador do papel com suas propriedades estáticas. Na consulta SQL os atributos da cláusula **SELECT** são escritos na forma: nome_tabela.atributo. Neste ponto a consulta SQL teria a seguinte estrutura:

```
SELECT t3.salario
FROM Pessoa t1, Funcionario t2, Funcionario_salario t3
```

O próximo passo é fazer a análise da cláusula **WHERE** da linguagem TF-ORM. Para cada propriedade da cláusula **WHERE** é feita uma consulta ao arquivo SYS.SYSCOLUMN para verificar em que tabela esta propriedade está armazenada. Se o nome da tabela não estiver na cláusula **FROM** da consulta SQL, o sistema inclui o nome da tabela.

Todos os predicados e funções encontrados na cláusula **WHERE** da consulta TF-ORM serão tratados posteriormente, exceto o predicado *has_property_instance* (verifica se existe uma instância do papel ou classe para a propriedade associada). Nesta caso, o sistema inclui o segundo parâmetro (nome da classe ou papel) na cláusula **FROM**. O sistema também inclui na cláusula **FROM** as tabelas que serão utilizadas por alguns predicados e funções. O predicado *active_class* e *active_role*, por exemplo, utilizam um atributo armazenado na tabela com o sufixo, dynamic. Se estes predicados forem encontrados na consulta, o sistema inclui a tabela correspondente na cláusula **FROM**.

Depois que todas as propriedades da cláusula **WHERE** (TF-ORM) já tiverem sido analisadas e a cláusula **FROM** está completa, o sistema começa a montar a cláusula **WHERE** da consulta SQL. Inicialmente são definidos todos os relacionamentos existentes entre as tabelas da cláusula **FROM** da consulta SQL consultando os arquivos do banco de dados que armazenam a descrição do esquema. Depois que todos os relacionamentos foram definidos, o sistema acrescenta a cláusula **WHERE** (SQL) as restrições especificadas pela consulta TF-ORM. A estrutura da consulta anterior até este momento seria a seguinte:

```
SELECT t3.salario
FROM Pessoa t1,Funcionario t2, Funcionario_salario t3,
        Pessoa_nome t4, Funcionario_trabdepto t5, Depto t6,
        Funcionario_dynamic t7, Depto_nomed t8
```

WHERE t1.oId= t2.oId and t4.oId = t1.oId
and t3.rId = t2.rId and t5.trabdepto = t6.oId
and t7.rId = t2.rId and t6.oId = t8.oId and t4.nome = 'Pedro'

O passo seguinte é fazer o mapeamento das funções e predicados encontrados na cláusula **WHERE**. A consulta TF-ORM apresentada como exemplo, possui três predicados: *has_property_instance*, *active_role*, *is_valid*. O predicado *has_property_instance* já foi tratado na etapa anterior. O predicado *active_role* retorna o valor verdadeiro se a instância colocada como parâmetro esta ativa. No modelo TF-ORM as propriedades *object_instance* e *role_instance* armazenam esta informação, se seu valor for igual a *null* a instância estará inativa. No mapeamento do modelo TF-ORM para o modelo relacional, estas propriedades foram armazenadas na tabela que contém o nome da classe ou do papel seguido pela palavra dynamic. No exemplo, a consulta está sendo realizada sobre o papel funcionário, por isso a pesquisa será feita no atributo *role_instance* da tabela *Funcionario_dynamic*. O valor deste atributo deve ser diferente de null. O *rId* colocado como parâmetro deve ser verificado na tabela base do papel, ou seja, o atributo *rId* da tabela *Funcionario* deve ter o valor *rId3*.

O predicado *is_valid (NomePropriedade, Valor)* verifica se o valor válido para a propriedade de nome *NomePropriedade* é o valor definido por *Valor*. No exemplo, a propriedade *nomed* deve possuir o valor *Vendas*. Portanto, o atributo *nomed* da tabela *Depto_nomed* deve possuir o valor *Vendas*.

A consulta na linguagem SQL depois do mapeamento dos predicados possui a seguinte estrutura:

SELECT t3.salario
FROM Pessoa t1,Funcionario t2, Funcionario_salario t3,
Pessoa_nome t4, Funcionario_trabdepto t5, Depto t6,
Funcionario_dynamic t7, Depto_nomed t8
WHERE t1.oId= t2.oId and t4.oId = t1.oId
and t3.rId = t2.rId and t5.trabdepto = t6.oId
and t7.rId = t2.rId and t8.oId = t6.oId and t4.nome = 'Pedro'
and t2.rId = 'rId3' and t7.role_instance <> null
and t8.nomed = 'Vendas'

O mapeamento completo dos predicados e funções do TF-ORM para instruções SQL encontra-se no Anexo 3.

7.2.3.2 Definição das Restrições de Tempo

A consulta anterior ainda não é equivalente à consulta TF-ORM, pois não existe nenhuma restrição temporal aplicada aos atributos da consulta. A definição de restrições temporais é a última etapa do mapeamento e é realizada através de um conjunto de regras. A linguagem de consulta do modelo TF-ORM apresenta algumas palavras especiais que permitem a indicação de informações temporais: DATE, PERIOD, TRANSACTION_DATE e TRANSACTION_PERIOD. Estes operadores não são encontrados na linguagem SQL/ANSI e por isso, devem ser mapeados para os atributos temporais do esquema do banco de dados (*v_timei*, *v_timef*, *t_timei*, *t_timef*).

Somente as consultas que utilizam a cláusula **WHEN** podem conter as palavras especiais para definição de restrições temporais, pois com a utilização da cláusula WHERE a informação é buscada no banco de dados instantâneo (*snapshot*).

As consultas que utilizam a cláusula **WHERE** e não a cláusula **WHEN**, desejam recuperar informações do banco de dados instantâneo, correspondendo ao estado atual do banco de dados ou de uma história passada (cláusula **AS ON**). Nas consultas que possuem a cláusula **WHERE**, os atributos dinâmicos envolvidos na consulta recebem restrições sobre os atributos tempo de validade inicial (*v_timei*) e tempo de validade final (*v_timef*):

- *v_timei* - dever ser menor ou igual à data atual para que o valor da propriedade dinâmica represente uma informação que já iniciou sua validade;
- *v_timef* - deve ser maior que a data atual para que o valor da propriedade seja válida na data atual, ou igual a *null* caso a data de validade final ainda não tenha sido estabelecida. Na figura 7.4 as letra a e b apresentam valores válidos para o banco de dados atual e as letras c e d apresentam valores não válidos no BD atual.

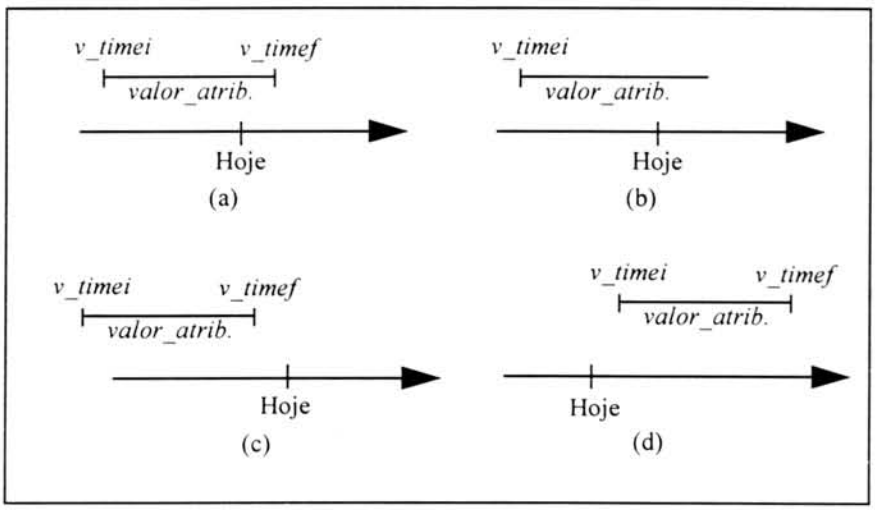

FIGURA 7.4 - Recuperação dos Dados Atuais

Se a cláusula **AS ON** for utilizada na consulta, deve ser feita uma restrição temporal ao atributo tempo de transação inicial (*t_timei*) das propriedades dinâmicas. O atributo *t_timei* deve ser menor ou igual à data especificada pela cláusula **AS ON**. Isso garante que sejam recuperadas as informações que foram inseridas no banco de dados até a data especificada na cláusula de instante temporal, informações inseridas no banco de dados após esta data serão desconsideradas.

Para as consultas que utilizam a cláusula **WHEN** existe mais de uma possibilidade de mapeamento, dependendo das restrições temporais da consulta TF-ORM. A seguir são definidas algumas regras de mapeamento que devem ser aplicadas para produzir a consulta na linguagem SQL:

- **apenas a cláusula WHEN** - as consultas com saída sobre dados e que não possuem nenhuma outra restrição temporal na cláusula de busca, desejam recuperar toda história dos dados. Para que os valores especificados na consulta

representem um estado válido em um instante do tempo, deve existir um ponto de intersecção entre os intervalos de tempo dos atributos da consulta. Considere que *atrib1* e *atrib2* são atributos envolvidos na consulta, então: *v_timei(atrib1)* < *v_timef(atrib2)* e *v_timei(atrib2)* < *v_timef(atrib1)*. Na figura 7.5, as letras (a), (b) e (c) representam estados que obedecem a esta restrição e por isso serão consideradas no resultado da consulta, já a letra (d) não apresenta um ponto de intersecção para os valores dos atributos.

- **PERIOD na cláusula WHEN** - os valores das propriedades dinâmicas deverão satisfazer o intervalo de tempo determinado por PERIOD. O período é aplicado a todas as propriedades dinâmicas da cláusula de busca. O atributo *v_timei* de cada propriedade dinâmica deverá ser menor ou igual a data inicial do período e o atributo *v_timef* deverá ser maior do que à data final do período ou ainda não estar definido. Além desta comparação, deve ser verificado se o conjunto de restrições da cláusula de busca representa um estado válido como apresentado no item anterior.
- **DATE na cláusula WHEN** - os valores das propriedades dinâmicas deverão satisfazer a data determinada pela cláusula DATE. O atributo *v_timei* de cada propriedade dinâmica deve ser menor ou igual a data da cláusula DATE e o atributo *v_timef* deve ser maior que esta data, ou ainda possuir o valor *null*. Neste caso, não é necessário aplicar a regra para obter estados válidos, pois se a restrição anterior for satisfeita, certamente os valores dos atributos representam um estado válido na data especificada;
- **TRANSACTION_DATE ou TRANSACTION_PERIOD na cláusula WHEN** - o procedimento é semelhante ao descrito anteriormente para as palavras date e period, respectivamente, a única diferença é que ao invés do atributo tempo de validade será utilizado o atributo tempo de transação;
- **PERIOD na cláusula SELECT** - a consulta deseja obter todos os períodos que satisfazem as restrições da cláusula de busca. Para obter o período, é necessário recuperar o tempo de validade inicial e final das propriedades dinâmicas. Cada tupla do resultado irá conter tantos atributos *v_timei* e *v_timef* quantas forem as propriedades dinâmicas da cláusula de busca. Antes de apresentar as tuplas do resultado para o usuário, o sistema gera o período correspondente de cada tupla. O período para cada tupla do resultado é obtido como segue:

  Período =[max(pd_1.v_timei,..,pd_n.v_timei),min(pd_1.v_timef,..,pd_n.v_timef)];

  O limite inferior do período é obtido através do maior tempo de validade inicial dentre as propriedades dinâmicas consideradas (pd_1..pd_n) e o limite superior é obtido através do menor tempo de validade final das propriedades dinâmicas;
- **DATE na cláusula SELECT** - a consulta deseja obter a data que satisfaz o conjunto de predicados da cláusula de busca. DATE será mapeado para o atributo tempo de validade inicial (*v_timei*). Como a cláusula de busca pode conter mais de um atributo, a data em que o conjunto de informações da cláusula de busca é verdadeiro corresponde ao maior *v_timei* dos estados válidos. A cláusula **SELECT** da consulta SQL irá conter o tempo de validade inicial (*v_timei*) associado a cada uma das propriedades dinâmicas envolvidas na cláusula de

busca. Para cada uma das tuplas do resultado, o maior *v_timei* corresponde à data em que as informações da cláusula de busca são verdadeiras.

- **TRANSACTION_DATE ou TRANSACTION_PERIOD na cláusula SELECT** - o procedimento é o mesmo descrito anteriormente, só que, ao invés do atributo tempo de validade, será utilizado o atributo tempo de transação;
- **cláusula AS ON** - com a utilização da cláusula **AS ON** serão consideradas apenas as informações que foram inseridas no banco de dados até a data especificada por esta cláusula. Para cada propriedade dinâmica da consulta, o tempo de transação inicial (*t_timei*) deve ser menor ou igual a data da cláusula **AS ON**.

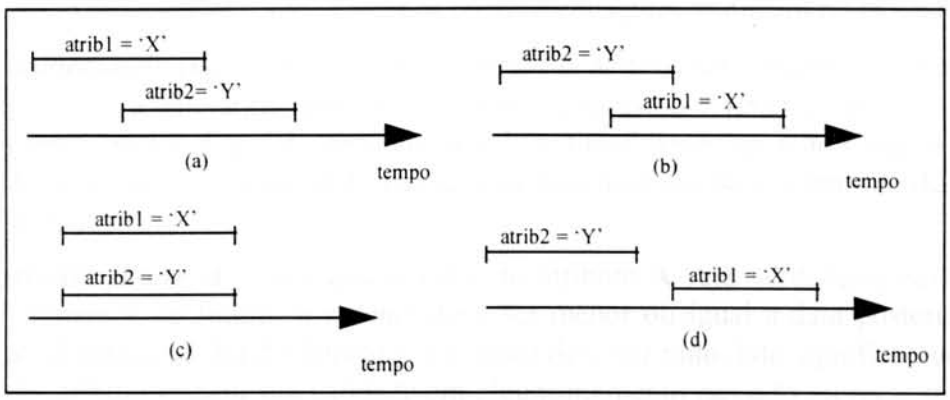

FIGURA 7.5 - Recuperação de Estados Válidos

- ***sometime past* A** - para que o valor da propriedade **A** seja verdadeiro em algum momento do passado o tempo de validade inicial deve ser menor que a data atual, *v_timei* < data_atual. O tempo de validade final não interessa nesta cláusula da consulta pois o valor do atributo precisa ser verdadeiro em apenas um instante do passado. A figura 7.6 apresenta alguns estados possíveis, sendo que apenas as letras a e b apresentam um resultado válido.

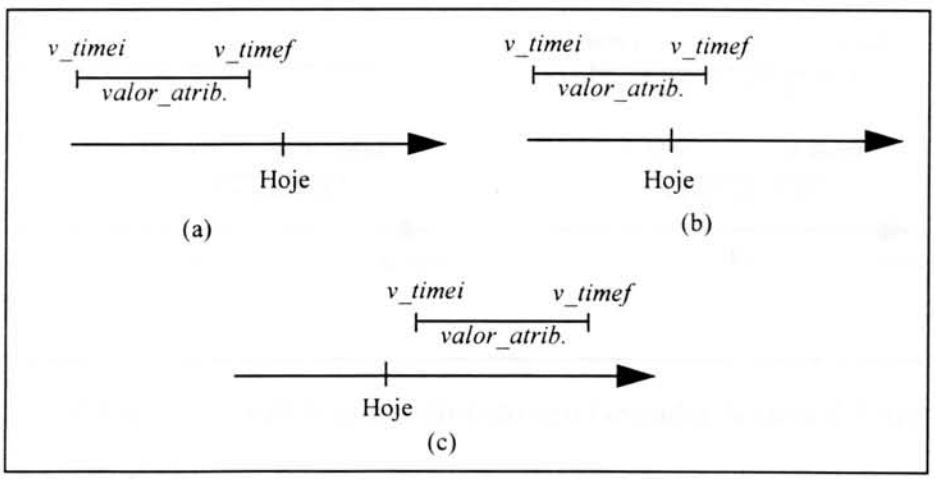

FIGURA 7.6 - Validade dos Atributos e o Operador Temporal Sometime Past

- ***immediately past A*** - O valor do atributo **A** deve ser verdadeiro na data anterior a data atual (data-de-ontem). O sistema obtém esta data e a utiliza para comparar com o tempo de validade inicial do atributo. O *v_timei* do atributo deve ser menor ou igual a data_anterior e o *v_timef* deve ser maior que a data-de-ontem ou nulo.
- ***always past A*** - para que o valor do atributo **A** seja verdadeiro em todos os momentos do passado, o *v_timei* do atributo **A** deve ser igual o menor *v_timei* definido para o atributo A e o *v_timef* deve ser maior que a data atual.
- ***sometime future A*** - para que o valor do atributo **A** seja verdadeiro em algum momento no futuro, o tempo de validade final deve ser maior que a data atual. Neste caso não é necessário testar o valor do atributo *v_timei* pois não interessa o início da validade e sim se o valor aparece em algum momento no passado.
- ***immediately future A*** - o valor do atributo **A** deve ser verdadeiro em uma data posterior a data atual (data-de-amanhã). Para isso o *v_timei* do atributo deve ser menor ou igual a data-de-amanhã e o v_timef deve ser maior que a data-de-amanhã ou igual a *null* (não existe uma data definida para o término de validade do valor).
- ***always future A*** - para que o valor do atributo **A** seja verdadeiro em todos os momentos do futuro, o *v_timei* deve ser menor ou igual a data posterior a data atual (primeiro dia do futuro) e o *v_timef* deve ser nulo. Isto significa que o valor do atributo iniciou sua validade em algum momento passado ou presente e que é verdadeiro em todo o futuro pois não existe nenhum outro valor definido para a propriedade.
- ***A since B*** - para que o primeiro argumento seja verdadeiro em todos os momentos desde que o segundo argumento se tornou verdadeiro, o *v_timei* de **A** deve ser menor ou igual ao *v_timei* de **B** e o v_timef de A deve ser maior ou igual ao *v_timef* de **B** ou ainda pode ser igual a *null* . A figura 7.7 apresenta duas situações em que a os valores dos atributos **A** e **B** são verdadeiros para o operador *since*.

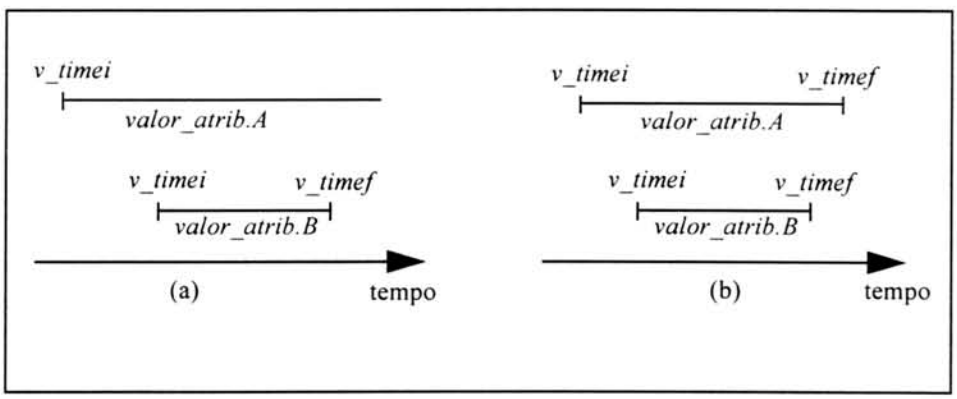

FIGURA 7.7 - Validade dos Atributos e o Operador Temporal Since

- ***A until B*** - o primeiro argumento deve ser verdadeiro em todos os momentos desde a criação das instâncias envolvidas até o momento em que o segundo argumento se tornou verdadeiro. O valor do atributo *v_timei* de **A** deve ser igual a

menor data de validade definida para a instância e o *v_timef* de **A** deve ser maior que o *v_timei* de **B**.

- ***A before B*** - o primeiro argumento deve ser verdadeiro em algum momento anteior àquele em que o segundo argumento se tornou verdadeiro. O *v_timei* de **A** deve ser menor que o *v_timei* de **B**. A figura 7.8a apresenta uma situação em que o valor dos atributos **A** e **B** são verdadeiros para o operador *before*.
- ***A after B*** - o tempo de validade inicial de **A** deve ser maior que o *v_timei* de **B** para que o argumento **A** seja verdadeiro em algum momento posterior a **B**. A figura 7.8b apresenta esta situação;

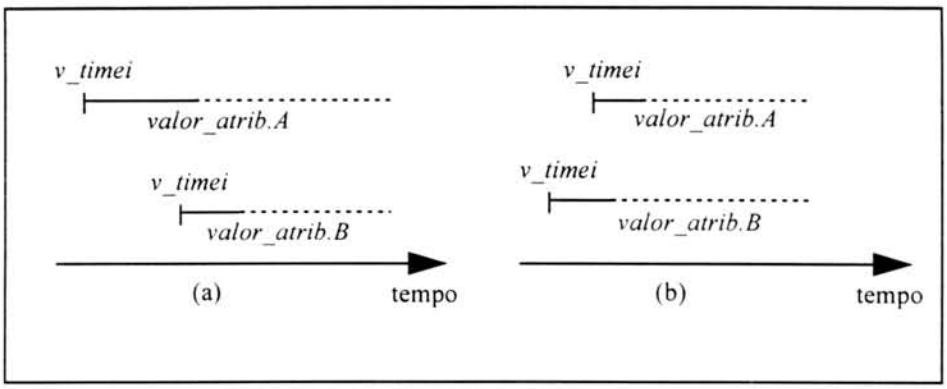

FIGURA 7.8 - Atributos Válidos para os Operadores Before e After

7.2.3.3 Exemplos de Consultas

A linguagem de consulta do modelo permite a recuperação de cinco diferentes tipos de histórias do banco de dados, apresentada no capítulo 2. A seguir serão apresentados alguns exemplos destes tipos de histórias e seu mapeamento para a linguagem SQL.

Um exemplo de consulta que recupera *informações válidas no momento presente* é a seguinte: "Selecione o atual departamento em que trabalha um funcionário denominado João e que ganha mais de R$ 1000,00". A figura 7.9 mostra a consulta na linguagem TF-ORM.

O mapeamento desta consulta para a linguagem SQL modificará totalmente as cláusulas **FROM** e **WHERE** pois serão incluídas novas tabelas e novas restrições sobre os atributos. Na tradução da consulta o sistema faz uma pesquisa nos arquivos com a descrição do esquema mantidos pelo Watcom para verificar para que tabelas as propriedades foram mapeadas. As propriedades *nome*, *salario* e *nomed* envolvidas na consulta, são propriedades que podem mudar de valor ao longo do tempo e por isso foram mapeados para as seguintes tabelas do banco de dados relacional: *Pessoa_nome*, *Funcionario_salario*, *Depto_nomed*, respectivamente. Os relacionamentos entre as tabelas são especificados na cláusula **WHERE** de acordo com o que foi estabelecido na criação do esquema.

Como a consulta deseja obter o departamento atual e as informações da cláusula **WHERE** referem-se ao presente, é necessário fazer restrições temporais ao

atributo tempo de validade (*v_time*) das propriedades dinâmicas. Para que as igualdades estabelecidas na cláusula **WHERE** façam referência a valores atuais, o tempo de validade inicial precisa ser menor ou igual a data atual (por exemplo, 23/04/96) e o tempo de validade final deve ser maior que a data atual ou ainda não estar indefinido (*v_timef* = null). A consulta na linguagem SQL é apresentada na figura 7.9.

TF-ORM	SQL
SELECT nomed FROM Pessoa.Funcionario WHERE nome="Joao" and salario>1000 and has_property_instance(trabdepto, Depto)	SELECT t7.nomed FROM Pessoa t1, Funcionario t2, Pessoa_nome t3, Funcionario_salario t4, Funcionario_trabdepto t5, Depto t6, Depto_nomed t7 WHERE t1.oId= t2.oId and t3.oId = t1.oId and t4.rId = t2.rId and and t5.rId = t2.rId and t5.trabdepto = t6.oId and t6.oId = t7.oId and t3.nome = 'joao' and t4.salario > 1000 and t3.v_timei <= '1996-04-23' and t4.v_timei <= '1996-04-23' and t5.v_timei <= '1996-04-23' and (t3.v_timef > '1996-04-23' or t3.v_timef = null) and (t4.v_timef>'1996-04-23' or t4.v_timef = null) and (t5.v_timef > '1996-04-23' or t5.v_timef = null);

FIGURA 7.9 - Recuperação de Dados Instantâneos Atuais

A figura 7.10 mostra um conjunto de valores possíveis para as propriedades dinâmicas no banco de dados. Para a consulta anterior o resultado obtido seria vendas, pois o atributo *nomed* possuía o valor *compra* no passado, mas no presente o valor do atributo *nomed* é vendas. Isto significa que a data de validade do valor *vendas* para o atributo *depto* iniciou antes da data atual e ainda não terminou.

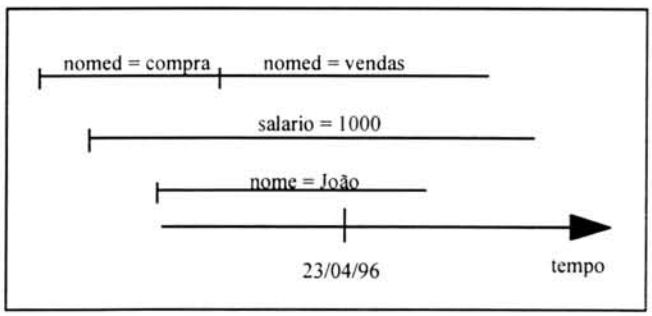

FIGURA 7.10 - Valor dos Atributos e Tempo de Validade

Uma consulta que recupera *dados instantâneos passados* utiliza a cláusula DATE para especificar um instante de tempo passado. A consulta anterior, por

exemplo, poderia representar uma história deste tipo se fosse alterada para: "Selecione o departamento em que trabalhava um funcionário denominado João e que ganhava mais do que R$ 1000,00 em 11/11/95". A única diferença da consulta SQL anterior seria a data que agora é 11/11/95 e não mais a data atual. A figura 7.11 mostra a consulta na linguagem TF-ORM e na linguagem SQL.

TF-ORM	SQL
SELECT nomed FROM Pessoa.Funcionario WHERE nome="Joao" and salario>1000 and has_property_instance (trabdepto, Depto) and DATE = 11/11/95	SELECT t7.nomed FROM Pessoa t1, Funcionario t2, Pessoa_nome t3, Depto_nomed t7 Funcionario_salario t4, Funcionario_trabdepto t5, Depto t6, WHERE t1.oId= t2.oId and t3.oId = t1.oId and t4.rId = t2.rId and t5.rId = t2.rId and t5.trabdepto = t6.oId and t6.oId = t7.oId and t3.nome = 'Joao' and t4.salario > 1000 and t3.v_timei <= '1995-11-11' and t4.v_timei <= '1995-11-11' and t5.v_timei <= '1995-11-11' and (t3.v_timef > '1995-11-11' or t3.v_timef = null) and (t4.v_timef > '1995-11-11' or t4.v_timef = null) and (t5.v_timef > '1995-11-11' or t5.v_timef = null);

FIGURA 7.11 - Recuperação de Valores em uma Determinada Data

Uma consulta que recupera *dados históricos* e que apresenta seleção sobre dados e saída temporal é: "Selecione todos os períodos em que um funcionário denominado João ocupava o cargo de vendedor". As propriedades envolvidas são: *nome* (da classe Pessoa) e *cargo* (do papel *Funcionario* da classe *Pessoa*). A única restrição temporal feita sobre os atributos da consulta é a de que o funcionário tenha o nome igual a João no mesmo período de tempo em que ocupa o cargo de vendedor. Os tempos de validade destes atributos devem ser comparados entre si para verificar se os valores dos atributos *nome* e *cargo* representam um estado do funcionário em um instante do tempo. A cláusula **WHEN** determina que esta situação seja avaliada não apenas no presente, mas também no passado e no futuro. Para obter o período, a cláusula **SELECT** da consulta SQL deve conter os atributos *v_timei* e *v_timef* associados aos atributos (*nome* e *cargo*). O sistema recebe o resultado da consulta e monta o período como apresentado na seção anterior. A figura 7.12 mostra as consultas na linguagem TF-ORM e SQL.

TF-ORM	SQL
SELECT PERIOD FROM Pessoa.Funcionario WHEN nome ="Joao" and cargo = "vendedor"	SELECT t3.v_timei, t4.v_timei, t3.v_timef, t4.v_timef FROM Pessoa t1, Funcionario t2, Pessoa_nome t3, Funcionario_cargo t4 WHERE t1.oId = t2.oId and t3.oId = t1.oId and t4.rId = t2.rId and t3.nome='Joao' and t4.cargo = 'vendedor' and (t3.v_timef > t4.v_timei or t3.v_timef = null) and (t4.v_timef > t3.v_timei or t4.v_timef = null);

FIGURA 7.12 - Recuperação de Dados Históricos - Saída Temporal

TF-ORM	SQL
SELECT PERIOD FROM Pessoa.Funcionario WHEN nome ="Joao" and cargo = "vendedor" and PERIOD in (08/08/95, 09/09/96)	SELECT t3.v_timei, t4.v_timei, t3.v_timef, t4.v_timef FROM Pessoa t1, Funcionario t2, Pessoa_nome t3, Funcionario_cargo t4 WHERE t1.oId = t2.oId and t3.oId = t1.oId and t4.rId = t2.rId and t3.nome='Joao' and t4.cargo = 'vendedor' and (t3.v_timef > t4.v_timei or t3.v_timef = null) and (t4.v_timef > t3.v_timei or t4.v_timef = null); and t3.v_timei < = '1995-08-08' and t4.v_timei <= '1995-08-08' and (t3.v_timef > '1996-09-09' or t3.v_timef = null) and (t4.v_timef > '1996-09-09' or t4.v_timef = null)

FIGURA 7.13 - Recuperação de Dados Históricos - Seleção Mista

Uma consulta bastante parecida com a anterior mas apresentando seleção mista e saída temporal é: "Selecione todos os períodos que um funcionário denominado João exercia o cargo de vendedor durante o período de 08/08/95 a 09/09/96" (fig. 7.13).

Para que os valores dos atributos envolvidos na consulta representem um estado válido é necessário que eles satisfaçam a restrição temporal imposta pela palavra PERIOD e que apresentem os mesmos valores especificados na consulta no mesmo intervalo de tempo. Isto significa que os tempos de validade dos atributos da consulta devem estar dentro do período especificado. Para obter o período como resultado, basta obter o maior tempo de validade inicial de cada tupla e o menor tempo de validade final de cada tupla.

TF-ORM	SQL
SELECT PERIOD FROM Pessoa.Funcionario WHEN nome="Joao" and cargo = "vendedor" and PERIOD in (08/08/95, 09/09/96) AS ON 01/01/95	SELECT t3.v_timei, t4.v_timei, t3.v_timef, t4.v_timef FROM Pessoa t1, Funcionario t2, Pessoa_nome t3, Funcionario_cargo t4 WHERE t1.oId = t2.oId and t3.oId = t1.oId and t4.rId = t2.rId and t3.nome='Joao' and t4.cargo = 'vendedor' and (t3.v_timef > t4.v_timei or t3.v_timef = null) and (t4.v_timef > t3.v_timei or t4.v_timef = null); and t3.v_timei < = '1995-08-08' and t4.v_timei <= '1995-08-08' and (t3.v_timef > '1996-09-09' or t3.v_timef = null) and (t4.v_timef > '1996-09-09' or t4.v_timef = null) and t3.t_timei <= '1995-01-01' and t4.t_timei <= '1995-01-01'

FIGURA 7.14 - Recuperação de Dados Históricos de uma História Passada

Considere a seguinte consulta: "Selecione todos os períodos nos quais o funcionário denominado João exercia o cargo de vendedor, durante o período de 08/08/95 a 09/09/96, de acordo com que se acreditava em 01/01/95". Esta consulta permite a recuperação de *dados históricos de uma história passada* através da utilização da cláusula AS ON. As considerações sobre a restrição do período aplicada aos atributos e a saída da consulta, são as mesmas da consulta anterior. A diferença está na cláusula AS ON que determina a utilização do tempo de transação no mapeamento da consulta para a linguagem SQL. Para que a informação tenha sido inserida no banco de dados antes da data especificada na cláusula AS ON, o tempo de transação de todos os atributos envolvidos na consulta deve ser menor que a data especificada na cláusula AS ON. Esta consulta é apresentada na figura 7.14.

8 Conclusões e Trabalhos Futuros

8.1 Revisão do Trabalho

Este trabalho apresentou um sistema de consulta visual para recuperação de informações temporais utilizando o modelo TF-ORM (VQS TF-ORM). Para construir o sistema visual de consulta foi feito um estudo detalhado das potencialidades oferecidas pela linguagem textual de consulta do modelo, apresentado no capítulo 3. Além disso foram definidas neste capítulo, algumas extensões para aumentar o poder de expressão da linguagem de consulta.

O projeto do sistema de consulta visual procurou levar em consideração as características dos principais sistemas e linguagens visuais de consulta apresentadas no capítulo 2. Além disso, foram definidos mais alguns requisitos necessários em linguagens visuais e de forma específica para linguagens de consulta a modelos temporais. O VQS TF-ORM utiliza a facilidade e simplicidade oferecida pela linguagem de consulta baseada em formulários e a flexibilidade e o poder de expressão da linguagem de consulta gráfica.

O sistema proposto permite que a consulta seja realizada de três formas: gráfica, textual e por formulários. Para tanto, foi criada uma representação gráfica para o esquema do modelo TF-ORM. A representação gráfica do esquema consiste na definição de um conjunto de primitivas gráficas para representar os elementos do modelo (classes, subclasses, papéis, propriedades estáticas e propriedades dinâmicas). Sobre o esquema gráfico é possível definir subesquemas, chamados de esquemas de trabalho (ECGrT). Os ECGrT podem ser armazenados e utilizados no processo de recuperação de informações.

A consulta gráfica tem o mesmo poder de expressão da consulta textual e permite que o usuário utilize o esquema global ou um esquema de trabalho para seleção dos elementos de interesse para a consulta. Além do esquema gráfico, a consulta gráfica conta com um conjunto de janelas que permitem ao usuário incluir predicados e funções a consulta, estabelecer restrições temporais entre outras. A definição da restrição de tempo a consulta é feita em uma janela particular que apresenta todas as possibilidades de restrições temporais oferecidas pela linguagem textual e conta com a representação visual de cada uma das palavras temporais no eixo do tempo.

Como a consulta gráfica é equivalente à consulta textual, é possível elaborar a consulta textual e depois obter a mesma consulta na forma gráfica e vice-versa. Esta característica é bastante importante pois facilita o aprendizado da linguagem de consulta, tornando cada vez mais simples para o usuário o processo de recuperação de informações.

Além da consulta gráfica e textual, o ambiente possibilita a realização da consulta por formulários. A consulta por formulários, não tem o mesmo poder de expressão da consulta textual mas possibilita a definição de restrições temporais. Através de um conjunto de janelas e de botões de navegação, o usuário pode verificar todas as instâncias dos objetos de uma determina classe ou papel. Neste módulo é possível definir o tempo em que as informações devem ser recuperadas.

Como não existia um banco de dados específico para o modelo, utilizou-se um banco de dados relacional. As informações do modelo foram armazenadas no banco de dados segundo o mapeamento definido em [CAV 95].

8.2 Principais Resultados

Este trabalho apresenta quatro principais contribuições:

- o sistema visual de consulta;
- implementação de um ambiente que utiliza dois modelos de dados e dois modelos de consulta;
- mapeamento da linguagem de consulta do TF-ORM para a linguagem SQL;
- extensão da linguagem de consulta do modelo TF-ORM.

O sistema visual de consulta apresenta um conjunto de funcionalidades que torna eficiente o ambiente para recuperação de informações do modelo:

- representação gráfica do esquema conceitual - o esquema gráfico facilita a elaboração da consulta já que o usuário pode visualizar o conjunto de informações do esquema e a forma como estas informações estão relacionadas;
- consulta gráfica, textual e consulta por formulários - a linguagem de consulta textual e gráfica possuem o mesmo poder de expressão, sendo assim , o usuário pode escolher a que melhor lhe convier. A consulta por formulários pode ser utilizada quando o usuário deseja obter o valor de todas as propriedades de uma determinada instância, desejando apenas realizar alguma restrição temporal a consulta como um todo;
- definição de esquemas de trabalho - com a definição de esquemas de trabalho, o usuário visualiza apenas a porção do esquema de interesse para uma determinada consulta;
- representação da mesma consulta na forma textual e gráfica - a possibilidade de alternar a forma de elaboração da consulta, permite que o usuário fique cada vez mais familiarizado com a linguagem textual de consulta;
- armazenamento de esquemas de trabalho e consultas - os esquemas de trabalho podem ser armazenados para serem utilizados para realização de consultas. As consultas também podem ser armazenadas e utilizadas posteriormente;
- apresentação do resultado - de acordo com as características identificadas no resultado de consultas temporais, o sistema apresenta três formas alternativas para

apresentação do resultado que podem ser selecionadas pelo usuário (gráfico, tabela, formulário e visual).

Como o sistema é implementado sobre um banco de dados relacional, existem dois modelos de dados (TF-ORM e relacional). Foram consideradas as formas de armazenamento do esquema conceitual do TF-ORM no banco de dados relacional e sua recuperação. Este segundo mapeamento (relacional para TF-ORM) permite que o usuário trabalhe sobre o esquema no TF-ORM mesmo que o sistema utilize um banco de dados relacional para armazenar as informações. O mapeamento da linguagem de consulta foi definido para tradução da consulta gráfica para a consulta textual do TF-ORM e da consulta textual do TF-ORM para a consulta SQL.

O conjunto de regras definidas para o mapeamento da linguagem TF-ORM para a linguagem SQL, apresenta uma extensão ao trabalho realizado por [CAV 95] que apresenta apenas o mapeamento do modelo. Com o interpretador da linguagem textual de consulta é possível utilizar o modelo TF-ORM em um banco de dados relacional sem que o usuário precise conhecer todos os detalhes de mapeamento para o modelo relacional. Os dados são armazenados em um banco de dados relacional e a consulta pode ser realizada em uma linguagem de alto nível e específica para recuperação de informações temporais. Este mapeamento possibilita que o usuário realize suas consultas sem conhecer os detalhes de implementação. Como a consulta é traduzida para o padrão SQL/ANSI, este mapeamento poderia ser utilizado em qualquer outro banco de dados relacional e possibilitasse a definição de consultas neste formato.

A introdução de funções de agregação à linguagem de consulta do modelo, possibilita a realização de operações sobre um conjunto de valores em um determinado espaço no tempo. Além disso, a inclusão de mais um predicado a linguagem de consulta facilita a definição de relacionamentos entre classes e papéis.

8.3 Extensões e Trabalhos Futuros

Este trabalho apresentou uma linguagem gráfica e uma linguagem baseada em formulários para recuperação de informações temporais. Além disso, foram apresentadas algumas sugestões para representação visual de restrições temporais. Devido as vantagens encontradas na utilização de símbolos visuais, um estudo mais aprofundado poderia ser desenvolvido para criar uma linguagem de ícones para recuperação de informações temporais.

Este trabalho apresentou também algumas alternativas para apresentação dos resultados de consultas temporais. Baseado nas características dos resultados deste tipo de consultas (grande volume de informação, diferentes estados para o mesmo objeto e presença de informações temporais como data e período, associados a valores de dado), poderiam ser estudas outras formas de apresentação do resultado da consulta com o objetivo de facilitar ainda mais interpretação dos resultados.

Além das extensões da linguagem de consulta do modelo TF-ORM apresentadas neste trabalho, as seguintes características poderiam ser incorporadas à linguagem de consulta:

- possibilidade de definição da projeção temporal diferente da seleção temporal - na linguaguem de consulta do modelo, a projeção e a seleção temporal são realizadas sobre a mesma restrição de tempo. A linguagem poderia ser estendida possibilitando ao usuário definir restrições de tempo diferentes para os elementos da cláusula de busca e da cláusula de especificação; e
- possibilidade de utilização de mais de uma restrição de tempo na cláusula de busca - as condições da cláusula de busca poderiam ser associadas a diferentes restrições de tempo como data e período. Assim estas palavras temporais seriam associadas a uma ou mais condições e não apenas à consulta como um todo. Desta forma, a seleção temporal poderia ter diferentes restrições de data e período para os elementos envolvidos na consulta.

Anexo 1 Sintaxe da Linguagem de Consulta do Modelo TF-ORM

<query> ::= <where query> | <when query>
<where query> ::=
SELECT <where target clause>
FROM <object identification>
WHERE<where search clause>|
[**AS ON** <temporal instant clause>]
<when query> ::=
SELECT <when target clause>
FROM <object identification>
WHEN<where search clause>|
[**AS ON** <temporal instant clause>]
<where target clause> ::= <data output>
<when target clause> ::=
<data output>
|<temporal output>
|<data output> **and** <temporal output>
<data output> ::= <property name list>
| **count** "("<property name list>")"
| **sum** "("<property name list>")"
| **avg** "("<property name list>")"
| **max** "("<property name list>")"
| **min** "("<property name list>")"
|"*"
<temporal output> ::=
DATE | TRANSACTION_DATE | PERIOD | TRANSACTION_PERIOD
<property name list> ::= <property name> [, <property name list>]
<property name> ::= <identifier>
<object identification clause> ::=
[<oid variable> ","]<class name>
|[<oid variable>",'[<role variable>","]]<role name>
|[<oid variable>","]<class name>"."[<role variable>","]<role name>
|<class name>"."<role name>
<class name> ::= <identifier>
<role name> ::= <identifier>
<rule name> ::= <identifier>
<where search clause> ::= ::= <logical expression>
|<logical expression>
<when search clause> ::= <query logical expression>
<query logical expression> ::= <logical expression>
|<logical expression> and <predefined query predicate>

<logical expression> ::= <logical term>
|<logical expression><or operator><logical term>
<logical term> ::= <logical factor>
|<logical term><and operator><logical factor>
<logical factor> ::= <logical element> | **not** <logical element>
<logical element> ::= <predicate> | (<logical expression>)
|<logical element><temporal logical operator><predicate>
<or operator> ::= **or** |";"
<and operator> ::= **and** |","
<temporal logical operator> ::= **since** | **until** | **before** | **after**
<predicate> ::=
|<predefined predicate>
|<predefined temporal predicate>
|<state predicate>
|<predefined state predicate>
|[<temporal operator>]<quantifier><variable>(<predicate>)
|<arit expression><comp operator><arit expression>
|<temporal operator><logical expression>
| **false**
| **true**
<predefined predicate> ::=
has_class_instance "(" <class name> "," <oid variable>")"
| **has_role_instance** "("[<oid variable>]","<role name>"," <role variable>")"
| **active_class** "(" <oid variable> ")"
| **active_role** "(" <role variable> ")"
| **active_class_at** "("<oid variable> "," <temporal instant>")"
| **active_role_at** "("<role variable> "," <temporal instant>")"
| **is_valid** "("[<id variable>","]<property name>"," <property value>")"
| **is_valid_at** "(" [<id variable>","]
<property name> , <property value> , <temporal instant>)
| **out_role** "("<oid variable> "," role name")"
| **role** "("<oid variable>","<role variable>")"
<predefined temporal predicate> ::=
belongs "("<function argument>","<function name argument>")"
| **contains** "("<function name argument>","<function name argument>")"
| **before** "("<function argument>","<function argument>")"
| **equal** "("<function argument>","<function argument>")"
<temporal operator> ::= **sometime past** | **immediately past** | **always past**
sometime future | **immediately future** | **always future**
<quantifier> ::= **exists** | **forall**
<comp operator> ::= "<" | ">" | "=" | "≥" | "≤" | "≠"
<arit expression> ::= <term> | - <arit expression>
|<arit expression> "+" <term> | <arit expression> "-" <term>
<term> ::= <factor> | <term> "*" <factor>|<term> "/" <factor>
<factor> ::= <element> | <factor> "**" <element>
|<element> **union** <element>

|<element> **intersection** <element>
<element> ::=
[<id variable>","] <property name>
|[<id variable>","]<predefined property name>
|<function>|<value>|<variable>| "("<arit expression>")"
<predefined property name> ::= **oId** | **object_instance** | **end_object**
| **rId** | **role_instance** | **end_role**
<function> ::=
year "("<function argument>")"
| **month** "("<function argument>")"
| **day** "("<function argument>")"
| **hour** "("<function argument>")"
| **minute** "("<function argument>")"
| **weekday** "("<function argument>")"
| **lower_bound** "("<function name argument>")"
| **upper_bound** "("<function name argument>")"
| **duration** "("<function name argument>")"
| **to_minutes** "("<function argument>")"
| **to_months** "("<function argument>")"
| **to_days** "("<function argument>")"
| **value** "("[<id variable>","]<property name>")"
| **past_value** "("[<id variable>","]<property name><temporal instant>")"
| **valid_time** "("[<id variable>","]<property name>")"
| **transaction_time** "("[<id_variable>","]<property name>")"
| **class_creation_time** "("<oid variable>")"
| **role_creation_time** "("<role variable>")"
| **class_end_time** "("<oid variable>")"
| **role_end_time** "("<role variable>")"
| **state** "("<id variable>")"
| **state_at** "("<id variable><temporal instant>")"
<value> ::= <integer number> | <string> | <temporal value> | **null**
<integer number> ::= <digit> { <digit> }*
<digit> ::= 0|1|2|3|4|5|6|7|8|9
<string> ::= ""{ <any character including blanck> } +""
<temporal value> ::= <temporal instant>|<date value>|<hour value>
<date value> ::= <number> "/" <number> "/" <number>
<hour value> ::= <number> ":" <number>
<number> ::= <digit><digit>
<temporal instant> ::= <number> "/" <number> "/" <number> ","
<number> ":" <number>
<id variable> ::= <oid variable> | <role variable>
<oid variable> ::= <variable>
<role variable> ::= <variable>
<variable> ::= <identifier>
<function argument> ::= [<id variable>","]<property name>
|<temporal instant>|<variable>
<function name argument> ::=

[<id variable>","]<property name>|<variable>

<predefined query predicate> ::=

PERIOD in "["<date value>","<date value>"]"

| **DATE** "=" <date value>

<temporal instant clause> ::= <date value><predefined state predicate> ::= **state** "(" [<oid variable>","]<predefined state>")"

<predefined state> ::= **active** | **suspended**

<state predicate> ::= <defined state predicate> | <predefined state predicate>

<defined state predicate> ::= **state** "("[<id variable> ","]<state name>")"

<state name> ::= <identifier>

<property value> ::= <variable> | ""<string>""

<string> ::= ""{<any character including blanck>}+ ""

Anexo 2 Esquema Conceitual

CREATE TABLE Pessoa (oId CHAR(5), sexo CHAR(1), datanasc DATE, PRIMARY KEY(oId));

CREATE TABLE Pessoa_state (oId CHAR(5), estado CHAR(10), v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(estado, v_timei), FOREIGN KEY (oId) REFERENCES Pessoa (oId));

CREATE TABLE Pessoa_dynamic(oId CHAR (5), object_instance DATE, v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, end_object DATE, PRIMARY KEY(oId), FOREIGN KEY (oId) REFERENCES Pessoa (oId));

CREATE TABLE Pessoa_nome(oId CHAR (5) NOT NULL, nome CHAR(30), v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(nome, v_timei), FOREIGN KEY (oId) REFERENCES Pessoa (oId));

CREATE TABLE Pessoa_endereco(oId CHAR (5), endereco CHAR(30), v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, end_object DATE, PRIMARY KEY(endereco, v_timei), FOREIGN KEY (oId) REFERENCES pessoa (oId));

CREATE TABLE Medico (oId CHAR (5), rId CHAR(5), CREMERS CHAR(20), PRIMARY KEY (rId), FOREIGN KEY (oId) REFERENCES Pessoa (oId));

CREATE TABLE Medico_state (rId CHAR(5), estado CHAR(10), v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(estado, v_timei), FOREIGN KEY (rId) REFERENCES Medico (rId));

CREATE TABLE Medico_dynamic(rId CHAR(5), role_instance DATE, v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, end_role DATE, PRIMARY KEY(rId), FOREIGN KEY (rId) REFERENCES Medico (rId));

CREATE TABLE Medico_especialidade(rId CHAR(5), especialidade CHAR (30), v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(especialidade, v_timei), FOREIGN KEY (rId) REFERENCES Medico (rId));

CREATE TABLE Cliente (oId CHAR(5), rId CHAR(5), codc CHAR(10), PRIMARY KEY (rId), FOREIGN KEY (oId) REFERENCES Pessoa (oId));

CREATE TABLE Cliente_state (rId CHAR(5), estado CHAR(10), v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(estado, v_timei), FOREIGN KEY (rId) REFERENCES Cliente (rId));

CREATE TABLE Cliente_dynamic(rId CHAR(5), role_instance DATE, v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, end_role DATE, PRIMARY KEY(rId), FOREIGN KEY (rId) REFERENCES Cliente (rId));

CREATE TABLE Cliente_numcons(rId CHAR(5), numcons INTEGER, v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(numcons, v_timei), FOREIGN KEY (rId) REFERENCES Cliente (rId));

CREATE TABLE Cliente_plano(rId CHAR(5), plano CHAR(5), v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(plano, v_timei), FOREIGN KEY (plano) REFERENCES PlanoSaude (oId), FOREIGN KEY (rId) REFERENCES Cliente (rId));

CREATE TABLE Cliente_consulta(rId CHAR(5), consulta CHAR(5), v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(consulta, v_timei), FOREIGN KEY (consulta) REFERENCES Medico (rId), FOREIGN KEY (rId) REFERENCES Cliente (rId));

CREATE TABLE PlanoSaude (oId CHAR(5), codp CHAR(10), PRIMARY KEY(oId));

CREATE TABLE PlanoSaude_state (oId CHAR(5), estado CHAR(10), v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(estado, v_timei), FOREIGN KEY (oId) REFERENCES PlanoSaude (oId));

CREATE TABLE PlanoSaude_dynamic(oId CHAR(5), object_instance DATE, v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, end_object DATE, PRIMARY KEY(oId), FOREIGN KEY (oId) REFERENCES PlanoSaude (oId));

CREATE TABLE PlanoSaude_nomep(oId CHAR(5), nomep CHAR(20), v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(nomep, v_timei), FOREIGN KEY (oId) REFERENCES PlanoSaude (oId));

CREATE TABLE PlanoSaude_tipo(oId CHAR(5), tipo CHAR(20), v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(tipo, v_timei), FOREIGN KEY (oId) REFERENCES PlanoSaude (oId), CHECK (tipo = 'familiar' or tipo ='empresarial'));

CREATE TABLE PlanoSaude_carenciaint(oId CHAR(5), carenciaint INTEGER, v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(carenciaint, v_timei), FOREIGN KEY (oId) REFERENCES PlanoSaude (oId));

CREATE TABLE PlanoSaude_carenciaurg(oId CHAR(5), carenciaurg INTEGER, v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(carenciaurg, v_timei), FOREIGN KEY (oId) REFERENCES PlanoSaude (oId));

CREATE TABLE PlanoSaude_carenciaexame(oId CHAR(5), carenciaexame INTEGER, v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(carenciaexame, v_timei), FOREIGN KEY (oId) REFERENCES PlanoSaude (oId));

CREATE TABLE PlanoSaude_valor(oId CHAR(5), valor REAL, v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(valor, v_timei), FOREIGN KEY (oId) REFERENCES PlanoSaude (oId));

CREATE TABLE Depto (oId CHAR(5), PRIMARY KEY(oId));

CREATE TABLE Depto_state (oId CHAR(5), estado CHAR(10), v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(estado, v_timei), FOREIGN KEY (oId) REFERENCES Depto (oId));

CREATE TABLE Depto_dynamic(oId CHAR(5), object_instance DATE, v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, end_object DATE, PRIMARY KEY(oId), FOREIGN KEY (oId) REFERENCES Depto (oId));

CREATE TABLE Depto_nome(oId CHAR(5), nomed CHAR(30), v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, end_object DATE, PRIMARY KEY(nomed, v_timei), FOREIGN KEY (oId) REFERENCES Depto (oId));

CREATE TABLE Depto_numfunc(oId CHAR(5), numfunc INTEGER, v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(numfunc, v_timei), FOREIGN KEY (oId) REFERENCES Depto (oId));

CREATE TABLE Funcionario (oId CHAR(5), rId CHAR(5), codf CHAR(10), PRIMARY KEY (rId), FOREIGN KEY (oId) REFERENCES Pessoa (oId));

CREATE TABLE Funcionario_state (oId CHAR(5), estado CHAR(10), v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(estado, v_timei), FOREIGN KEY (rId) REFERENCES Funcionario (rId));

CREATE TABLE Funcionario_dynamic(rId CHAR(5), role_instance DATE, v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, end_role DATE, PRIMARY KEY(rId), FOREIGN KEY (rId) REFERENCES Funcionario (rId));

CREATE TABLE Funcionario_salario(rId CHAR(5), salario REAL, v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(salario, v_timei), FOREIGN KEY (rId) REFERENCES Funcionario (rId));

CREATE TABLE Funcionario_cargo(rId CHAR(5), cargo CHAR(15), v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(cargo, v_timei), FOREIGN KEY (rId) REFERENCES Funcionario (rId));

CREATE TABLE Funcionario_trabdepto(rId CHAR(5), trabdepto CHAR(5), v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, FOREIGN KEY (rId) REFERENCES Funcionario (rId), FOREIGN KEY (trabdepto) REFERENCES Depto (oId));

CREATE TABLE Instituicao(oId CHAR(5), PRIMARY KEY(oId));

CREATE TABLE Instituicao_state (oId CHAR(5), estado CHAR(10), v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(estado, v_timei), FOREIGN KEY (oId) REFERENCES Instituicao (oId));

CREATE TABLE Instituicao_dynamic(oId CHAR(5), object_instance DATE, v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, end_object DATE, PRIMARY KEY(oId), FOREIGN KEY (oId) REFERENCES Instuicao (oId));

CREATE TABLE Instituicao_nomei(oId CHAR(5), nomei CHAR(15), v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(nomei, v_timei), FOREIGN KEY (oId) REFERENCES Instituicao (oId));

CREATE TABLE Instituicao_enderecoi(oId CHAR(5), enderecoi CHAR(15), v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(enderecoi, v_timei), FOREIGN KEY (oId) REFERENCES Instituicao (oId));

CREATE TABLE Hospital(oId CHAR(5), PRIMARY KEY(oId), FOREIGN KEY (oId) REFERENCES Instituicao (oId));

CREATE TABLE Hospital_dynamic(oId CHAR(5), object_instance DATE, v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, end_object DATE, PRIMARY KEY(oId), FOREIGN KEY (oId) REFERENCES Hospital (oId));

CREATE TABLE Hospital_numleitos(oId CHAR(5), numleitos INTEGER, v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(numleitos, v_timei), FOREIGN KEY (oId) REFERENCES Hospital (oId));

CREATE TABLE Hospital_emergencia(oId CHAR(5), emergencia CHAR(1), v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(emergencia, v_timei), FOREIGN KEY (oId) REFERENCES Hospital (oId), CHECK (emergencia ='S' or emergencia = 'N');

CREATE TABLE Laboratorio(oId CHAR(5), PRIMARY KEY(oId), FOREIGN KEY (oId) REFERENCES Instituicao (oId));

CREATE TABLE Laboratorio _dynamic(oId CHAR(5), object_instance DATE, v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, end_object DATE, PRIMARY KEY(oId), FOREIGN KEY (oId) REFERENCES Laboratorio (oId));

CREATE TABLE Hospital_realizaexames(oId CHAR(5), realizaexames INTEGER, v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(realiazaexames, v_timei), FOREIGN KEY (oId) REFERENCES Laboratorio (oId), FOREIGN KEY (realizaexames) REFERENCES Exame (oId));

CREATE TABLE Exame(oId CHAR(5), PRIMARY KEY(oId));

CREATE TABLE Exame_dynamic(oId CHAR(5), object_instance DATE, v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, end_object DATE, PRIMARY KEY(oId), FOREIGN KEY (oId) REFERENCES Exame (oId));

CREATE TABLE Exame_nomex(oId CHAR(5), nomex CHAR(15), v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(nomex, v_timei), FOREIGN KEY (oId) REFERENCES Exame (oId));

CREATE TABLE Exame_tempoR(oId CHAR(5), tempoR INTEGER, v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(tempoR, v_timei), FOREIGN KEY (oId) REFERENCES Exame (oId));

CREATE TABLE Exame_valor(oId CHAR(5), valorx REAL, v_timei DATE, v_timef DATE, t_timei DATE, t_timef DATE, PRIMARY KEY(valor, v_timei), FOREIGN KEY (oId) REFERENCES Exame (oId));

Anexo 3 Mapeamento de Predicados e Funções

has_class_instance (NomeClasse, Id) - a consulta SQL deve conter uma subconsulta que retorna o oId da tabela NomeClasse, onde oId é igual a Id. Além disso, o operador exists é utilizado para verificar se existe pelo menos uma tupla na tabela NomeClasse;

has_role_instance (IdClasse, NomePapel, IdPapel) - a consulta SQL deve conter uma subconsulta que retorna o oId da tabela NomePapel, cujo atributo oId é igual a IdClasse. Além disso, o operador exists é utilizado para verificar se existe pelo menos uma tupla na tabela NomePapel;

has_role_instance (NomePapel, IdPapel) - a consulta SQL deve conter uma subconsulta que retorna o oId da tabela NomePapel. Além disso, o operador exists deve ser utilizado para verificar se existe pelo menos uma tupla na tabela NomePapel;

active_class (Id) - a cláusula Where da consulta SQL contém a condição que verifica se o Id enviado como parâmetro é um valor válido para o atributo oId da classe considerada (NomeClasse) e se o atributo object_instance da tabela (NomeClasse_dynamic) contém um valor diferente de null;

active_role (Id) - a cláusula Where da consulta SQL deve conter a condição que verifica se o Id enviado como parâmetro é um valor válido para o atributo rId do papel considerado (NomePapel) e se o atributo role_instance da tabela (NomePapel_dynamic) contém um valor diferente de null;

active_class_at (Id, Tempo) - a cláusula Where da consulta SQL deve conter a condição que verifica se o Id enviado como parâmetro é um valor válido para o atributo oId da classe considerado (NomeClasse), se o atributo object_instance da tabela (NomeClasse_dynamic) contém um valor diferente de null e se o v_timei da tabela NomeClasse_dynamic é menor que Tempo e o v_timef é maior que Tempo;

active_role_at (Id, Tempo) - a cláusula Where da consulta SQL deve conter a condição que verifica se o Id enviado como parâmetro é um valor válido para o atributo rId do papel considerado (NomePapel), se o atributo role_instance da tabela (NomePapel_dynamic) contém um valor diferente de null e se o v_timei da tabela NomePapel_dynamic é menor que Tempo e o v_timef é maior que Tempo;

is_valid(Id, NomePropriedade, Valor) - a cláusula Where da consulta SQL deve conter a condição que verifica se o Id enviado como parâmetro é um valor válido para o atributo oId da classe considerada (NomeClasse). Isto no caso da propriedade estar definida no papel base, caso contrário o Id enviado como parâmetro é comparado com o atributo rId do papel considerado. Além disso, a cláusula Where deve conter a condição que verifica

se o valor da propriedade NomePropriedade armazenada na tabela NomeClasse_NomePropriedade ou NomePapel_NomePropriedade é igual a Valor;

is_valid(NomePropriedade, Valor) - a cláusula Where SQL deve conter a condição que verifica se o valor da propriedade NomePropriedade armazenada na tabela NomeClasse_NomePropriedade ou NomePapel_NomePropriedade é igual a Valor.

is_valid_at(Id, NomePropriedade, Valor, Tempo) - a cláusula Where da consulta SQL deve conter a condição que verifica se o Id enviado como parâmetro é um valor válido para o atributo oId da classe considerada (NomeClasse). Isto no caso da propriedade estar definida no papel base, caso contrário o Id enviado como parâmetro é comparado com o atributo rId do papel considerado. Além disso, a cláusula Where contém a condição que verifica se o valor da propriedade NomePropriedade armazenada na tabela NomeClasse_NomePropriedade ou NomePapel_NomePropriedade é igual a Valor. Além disso, verifica se o valor do atributo v_timei da tabela NomeClasse_NomePropriedade ou NomePapel_NomePropriedade é menor que Tempo e se o atributo v_timef é maior que Tempo;

is_valid_at(NomePropriedade, Valor, Tempo) - a cláusula Where deve conter a condição que verifica se o valor da propriedade NomePropriedade armazenada na tabela NomeClasse_NomePropriedade ou NomePapel_NomePropriedade é igual a Valor, se o valor do atributo v_timei da tabela NomeClasse_NomePropriedade ou NomePapel_NomePropriedade é menor que Tempo e se o atributo v_timef é maior que Tempo;

out_role (Id, NomePapel) - o operador not exists deve ser utilizado na cláusula Where para verificar se não existe nenhuma tupla na tabela NomePapel com o atributo oId igual a Id;

role (IdClasse, IdPapel) - o operador exists deve ser utilizado na cláusula Where para verificar se existe uma tupla na tabela do papel considerado (papel utilizado na cláusula FROM) com o atributo oId igual a IdClasse e o atributo rId igual a IdPapel;

before (Instante, Instante) - a cláusula Where da consulta SQL deve conter uma condição onde o primeiro instante é comparado com o segundo através do operador menor que (<);

value (Id, NomePropriedade) - a cláusula Where da consulta SQL deve conter a condição que verifica se o Id enviado como parâmetro é um valor válido para o atributo oId da classe considerada (NomeClasse). Isto no caso da propriedade estar definida no papel base, caso contrário o Id enviado como parâmetro é comparado com o atributo rId do papel considerado. Além disso, a cláusula Where deve conter a condição que verifica se o valor da propriedade NomePropriedade armazenada na tabela NomeClasse_NomePropriedade ou NomePapel_NomePropriedade é igual a constante associada a função, se o valor do atributo v_timei da tabela

NomeClasse_NomePropriedade ou NomePapel_NomePropriedade é menor ou igual a data atual e se o atributo v_timef é maior que a data atual;

value (NomePropriedade) - a cláusula Where deve conter a condição que verifica se o valor da propriedade NomePropriedade armazenada na tabela NomeClasse_ NomePropriedade ou NomePapel_NomePropriedade é igual a constante associada a função, e se o valor do atributo v_timei da tabela NomeClasse_NomePropriedade ou NomePapel_NomePropriedade é menor ou igual a data atual e se o atributo v_timef é maior que a data atual;

value_at (Id, NomePropriedade, Tempo) - o mapeamento é semelhante ao realizado para a função value, a única diferença é que o valor dos atributos v_timei e v_timef devem ser comparadas com o valor temporal enviado como parâmetro e não com a data atual;

valid_time (Id, NomePropriedade) - a cláusula Where deve conter a condição que verifica se o tempo de transação final associado ao atributo NomePropriedade é igual a null (último valor definido para a propriedade) e se o tempo de validade inicial associado ao atributo considerado é igual a constante associada a função. Além disso, o oId ou rId associado ao atributo deve ser igual a Id;

valid_time (NomePropriedade) - a cláusula Where deve conter a condição que verifica se o tempo de transação final associado ao atributo NomePropriedade é igual a null (último valor definido para a propriedade) e se o tempo de validade inicial associado ao atributo considerado é igual a constante associada a função.

transaction_time (Id, NomePropriedade) - a cláusula Where deve conter a condição que verifica se o tempo de transação final associado ao atributo NomePropriedade é igual a null (último valor definido para a propriedade) e se o tempo de transação inicial associado ao atributo considerado é igual a constante associada a função. Além disso, o oId ou rId associado ao atributo deve ser igual a Id;

transaction_time (NomePropriedade) - a cláusula Where deve conter a condição que verifica se o tempo de transação final associado ao atributo NomePropriedade é igual a null (último valor definido para a propriedade) e se o tempo de transação inicial associado ao atributo considerado é igual a constante associada a função;

class_criation_time (id) - a cláusula Where da consulta SQL deve conter a condição que verifica se o Id enviado como parâmetro é um valor válido para o atributo oId da classe considerada (NomeClasse) e se o atributo object_instance da tabela (NomeClasse_dynamic) é igual a constante associada a função;

role_criation_time (Id) - a cláusula Where da consulta SQL deve conter a condição que verifica se o Id enviado como parâmetro é um valor válido para o atributo rId do papel considerado (NomePapel) e se o atributo role_instance da tabela (NomePapel_dynamic) é igual a constante associada a função;

class_end_time(Id) e role_end_time (Id) - o mapeamento é semelhante ao realizado para as funções class_creation_time e role_criation_time, respectivamente. A diferença é que ao invés dos atributos object_instance e role_instance são utilizados os atributos object_end e role_end;

state (Id) - a cláusula Where da consulta SQL deve conter a condição que verifica se o Id enviado como parâmetro é um valor válido para o atributo rId do papel considerado (NomePapel) ou oId da classe considerada e se o valor do atributo estado é igual a constante associada a função;

state_at (Id, Tempo) - o mapeamento é semelhante ao da função anterior. Além das condições estabelecida para a função state, a cláusula Where contém a condição que verifica se o valor do atributo v_timei associado ao atributo estado é menor do que Tempo e se o valor do atributo v_timef é maior que Tempo;

Bibliografia

[AME 92] AMERICAN NATIONAL STANDARDS INSTITUTE. **ANSI X3.135:1992, American National Standard for Information System - Database Language SQL**, 1992.

[ANG 90] ANGELACCIO, M.; CATARCI, T.; SANTUCCI, G. QBD*: a graphical query language with recursion. **IEEE Transactions on Software Engineering**, New York, v.16, n.10, p.1150-1163, 1990.

[AGR 90] AGRAWAL, R. et al. OdeView: The Graphical Interface to Ode. In: ACM - SIGMOD INTERNATIONAL CONFERENCE ON MAGEMENT OF DATA, 1990, Atlantic City, New Jersey. **Proceeding....**New York: ACM, 1990. p.34-43.

[ARR 94] ARRUDA, E. H. P. **Um Modelo de Implementação da Linguagem TF-ORM para o SGBD O2.** Porto Alegre: CPGCC - UFRGS, 1994. (TI - 408).

[BAT 96] BATINI, C. et al. Visual Query Systems: A taxonomy. Disponível por FTP anônimo em disprcs.dis.uniroma1.it, no arquivo /ftp/pub/catarci/taxonomy.ps.gz (21 jun. 1996).

[BOR 95] BORLAND INTERNATIONAL INC. **Delphi User's Guide**. 1995. 452p.

[BOR 95a] BORLAND INTERNATIONAL INC. **Delphi Component Writer's Guide**, 1995. 156p.

[BOR 95b] BORLAND INTERNATIONAL INC. **Delphi Database Application Developer's Guide**, 1995. 189p.

[CAT 93] CATARCI, T. et al. Fundamental Graphical Primitives for Visual Query Languages. **Information Systems**, New York, v.18, n.2, p.75-98, Mar. 1993.

[CAT 94] CATARCI, T.; CHANG, S.; SANTUCCI, G. Query Representation and Management in a Multiparadigmatic Visual Query Enviroment. **Journal of Intelligent Information Systems**, Dordrecht, v.3, 1994.

[CAT 95] CATARCI, T.; TARANTINO, L. A Hypergraph-based Framework for Visual Interaction with Databases. **Journal of Visual Languages with Databases**, [S.l.], v.6, p.135-166, 1995.

[CAT 96] CATARCI, T. et al. **Visual Query Systems for Databases:** A Survey. Disponível por FTP anônimo em disparcs.dis.uniroma1.it, no arquivo /ftp/pub/catarci/QuerySurvey.ps.gz (21 jun. 1996).

[CAV 95] CAVALCANTI, J. M. B. **Implementação de Bancos de Dados Temporais Usando SGBDs Relacionais.** Belo Horizonte: Departamento de Ciência da Computação - UFMG, 1995. Dissertação de Mestrado.

[CLE 90] CLEMENTINI, E.; D'ATRI, A.; DI FELICE, P. Browsing in geographical databases: an object-oriented approach. In: IEE WORKSHOP ON VISUAL LANGUAGES, 1990. **Proceedings...** [S.1.:s.n.],1986.

[CLI 87] CLIFFORD J.; CROCKER, A.The historical relational data model (HRDM) and algebra based on lifespans**. IEEE Transactions on Software Engineering**, New York, v.14, n.7, July 1988. Trabalho apresentado na International Conference on Data Engineering, 1987, Los Angeles, California.

[CLI 93] CLIFFORD J.; CROCKER, A.; YUZHILIN, A. On the completeness of query languages for grouped and ungrouped historical data models. In: **Temporal Databases:** Theory, Design and Implementation. Bridge Parkway: Benjamin/Cummings, 1993. p.496-533.

[CLI 95] CLIFFORD J.; CROCKER, A.; YUZHILIN, A. Recent Trends in Temporal Databases. In: INTERNATIONAL WORKSHOP ON TEMPORAL DATABASES, 1995, Zurich, Switzerland. **Proceedings...** Great Britain: Springer-Verlag, 1995. 360p.

[CON 92] CONCENS, M.P.; CRUZ, I.F.; MEDELZON, A.O. Visualizing Queries and Querying Visualizations. **SIGMOD Record**, New York, v.21, n.1, p.39-46, Mar. 1992.

[CRU 92] CRUZ, I. F. DOODLE: A visual language object-Oriented Databases. In: INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 1992. San Diego, California. **Proceedings...** New York: ACM, 1992. p.71-80.

[DEN 95] DENNEBOUY, Y. et al. SUPER: Visual Interfaces for Object + Relationship Data Models**. Journal of Visual Languages and Computing**, [S.l.], v.6, n.1, p.73-99, 1995.

[DAT 87] DATE, C. **Introduction on standard SQL**. Paris: InterEditions, 1987. 239p.

[DOA 95] DOAN, D. K.; PATON, N. W. Design an user testing of a multi-paradigm query interface to an object-oriented database. **SIGMOD Record**, New York, v. 24, n.3, p. 12-17, 1995.

[EDE 93] EDELWEISS, N; OLIVEIRA, J.Palazzo M. de; PERNICI, B. An Object-Oriented Temporal Model. In: INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION SYSTEMS ENGINEERING (CAISE'93), 5., 1993, Paris. **Proceedings...** Heidelberg: Springer-Verlag, 1993. p.397-415. (Lecture Notes in Computer Science, 685).

[EDE 94] EDELWEISS, N. **Sistemas de Informação de Escritórios:** Um Modelo para Especificações Temporais. Porto Alegre: CPGCC da UFRGS, 1994. Tese de Doutorado.

[EDE 94a] EDELWEISS, N. Modelagem de aspectos temporais de sistemas de informação. In: ESCOLA DE COMPUTAÇÃO, 9., 1994, Recife. **Anais...** Recife: UFPE, 1994.

[EDE 94b] EDELWEISS, N; OLIVEIRA, J.P.M.; PERNICI, B. An Object-oriented approach to a temporal query language. In: DATABASE AND EXPERT SYSTEMS APPLICATIONS CONFERENCE - DEXA, 5., 1994, Athens, Greece. **Proceedings...** 1994. p.225-235. (Lecture Notes in Computer Science, 856).

[ELM 93] ELMASRI, R.; WUU, G.T.J. A temporal model and query language for ERR databases. In: **Temporal Databases**: Theory, Design, and Implementation. Bridge Parkway: Benjamin/Cummings, 1993. p.212-229.

[EPS 90] EPSTEIN, R. G. A Graphical Query Language for Object-Oriented Data Models. In: IEE WORKSHOP ON VISUAL LANGUAGES. **Proceedings...**[S.1]: IEE, 1990. p36-41.

[EPS 91] EPSTEIN, R. G. The TableTalk Query Language**. Journal of Visual Languages and Computing**, [S.l.], v.2, n.2, p.115-141, 1991.

[FER 94] FERNADES, S.; SHIEL, U. Um ambiente gráfico de consultas a um banco de dados temporal orientado a objetos. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, 9., 1994, São Carlos. **Anais...** São Carlos:USP, 1994. 381p.

[GOL 85] GOLDMAN; k.J. et al ISIS: Interface for a semantic information system. In: INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, Austin, Texas. **Proceedings...** New York: ACM, 1985. p. 328-341.

[HAB 94] HABE, E.M. et al. The ambleside survey: important topics in DB/HCI research. In: WORKSHOP ON USER INTERFACES TO DATABASES, 1994. **Proceedings...** [S.l.:s.n.], 1994.

[HAB 94a] HABE, E.M. et al. Foundations of Visual Metaphors for Schema Display. **Journal of Intelligent Information Systems**, Dordrecht, v.3, p.263-298, 1994.

[JEN 92] JENSEN, C.S. et al. A glossary of temporal databases concepts. **SIGMOD Record**, New York, v.21, n.3, p.53-43, Sept. 1992.

[KIM 84] KIM, H.; KORTH, H. F.; SILBERSCHATZ, A. SKI: a semantic knowledgeable Interface. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATABASES, 10., 1984, Singapor. **Proceedings...** [S.l.:s.n.], 1984. p.30-37.

[KIM 88] KIM, H.; KORTH, HF.; SILBERSCHATZ, A. PICASSO: a graphical query language . **Software Practice and Experience**, London, v.13, n.3, p.169-203, Mar. 1988.

[KOU 95] KOURAMAJIAN, V.; GERTZ, M. A graphical query language for temporal databases. In: OOER: OBJECT-ORIENTED AND ENTITY-RELANTIONSHIP MODELING, Berlin. **Proceedings...** Berlin: [s.n.], 1995.

[KOU 95a] KOURAMAJIAN, V.; GERTZ, M. **A visual query language for temporal databases**. [S.l.]: Universidade de Hannover, 1995.

[LEO 89] LEONG, M.;SAM, S.; NARASIMHALU, D. Towards a visual language for an object-oriented multi-media database system. In: WORKING CONFERENCE ON VISUAL DATABASE SYSTEMS, 1989, Tokyo, Japan. **Proceedings...** Amesterdan: North-Holland, 1989. p.465-495.

[MEL 94] MELLO, R. S. **Uma Linguagem Visual de Consulta para o Ambiente STAR.** Porto Alegre: CPGCC da UFRGS, 1994. Dissertação de Mestrado.

[MOH 93] MOHAN, L.; KASHYAP, R. L. A visual query language for graphical Interaction with Schema-Intensive databases. **IEEE Transactions on Knowledge and Data Engineering**, New York, v.5, n.5, p.843-858, Oct. 1993.

[OBE 94] OBERWEIS, A.; SÄNGER, V. GTL - A graphical language for temporal data. In: INTERNATIONAL WORKING CONFERENCE ON SCIENTIFIC AND STATISTICAL DATABASE MANAGEMENT, 17., 1994. **Procedings...** [S.l.]: IEE Computer Press, 1994.

[OLI 93] OLIVEIRA, J.L.; OLIVEIRA, R. Navegação e consulta em banco de dados orientados a objetos. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, 8., 1993, Campina Grande. **Anais...** Paraíba: UFPB, 1993. 430p.p35-49.

[OLI 93a] OLIVEIRA, J.L. On the development of user interface systems for object-oriented databases. In: INTERNATIONAL WORKSHOP ON ADVANCED VISUAL INTERFACES, 1994, Itália. **Proceedings...** [S.l.:s.n.], 1994.

[OLI 95] OLIVEIRA, J.P.M. et al. Implementation of an Object-Oriented Temporal Model. In: DATABASE AND EXPERT SYSTEMS APPLICATIONS CONFERENCE - DEXA, 6., 1995, london, U.K. **Proceedings...** [S.l.:s.n.], 1995.

[RAM 92] RAMOS, H.B. IQL: An integrated graphical user interfaces for relacional queries with genericity. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, 7., 1992, Porto Alegre, RS. **Anais...** Porto Alegre: UFRGS, 1992. p.431-446.

[RUB 95] RUBENKING, Neil J. **Programação em Delphi para Leigos**. São Paulo: Berkeley Brasil Editora, 1995. 372p.

[SAR 93] SARDA, N. L. HSQL: A Historical query language. In: **Temporal Databases**: Theory, Design, and Implementation. Bridge Parkway: Benjamin/Cummings, 1993. p.332-330.

[SIL 92] SILVA, R.C.; TRAINA, C. J. Armazenagem e gerenciamento de informações temporais em um modelo de dados orientado a objetos. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, 7., 1992, Porto Alegre, RS. **Anais...** Porto Alegre: UFRGS, 1992. p.431-446.

[SU 93] SU,S.Y.W.; CHEN, H. Modeling and management of temporal data in object-oriented knowledge bases. In: INTERNATIONAL WORKSHOP ON AN INFRASTRUCTURE FOR TEMPORAL DATABASES, 1993, Arligton, Texas. **Proceedings...** Arligton: [s.n.], 1993.

[SNO 93] SNODGRASS, R. An Overview of TQuel. In: **Temporal Databases:** Theory, Design, and Implementation. Bridge Parkway: Benjamin/Cummings, 1993. p.212-229.

[SNO 87] SNODGRASS, R. The temporal query language Tquel. **ACM Trnasactions on Database Systems**, New York, v.12, n.2, p.247-298, June 1987.

[SNO 95] SNODGRASS, R. **The TSQL2 Temporal Query Language**. Boston: Kluwer Academic, 1995.

[TAN 93] TANSEL, A.U., et al. A Generalized relational framework for modeling temporal data. In: **Temporal Databases**: Theory, Design, and Implementation. Bridge Parkway: Benjamin/Cummings, 1993. p. 183-201.

[TSU 90] TSUDA, K. et al. Iconic Browser: An Iconic Retrivel System for Object-Oriented Databases. **Journal of Visual Languages and Computing**, [S.l.], v.1, n.1, p.59-76, 1990.

[YEN 93] YEN, M. Y.; SCAMELL, R. W. A Human Factors Experimental Comparison of SQL and QBE. **IEEE Transactions on Software Engineering**, IEEE, v.19, n.4, p.390-402, 1993.

[WEI 93] WEILAND, W. J.; SHNEIDERMAN, B. A graphical query interface based on aggregation/generalization hierarchies. **Information Systems**, New York, v.18, n.4, p.215-232, June 1993.

[WU 89] WU, C. T; HSIAO, D.K. Implementation os visual database interface using an object-oriented language. In: WORKING CONFERENCE ON VISUAL DATABASE SYSTEMS, 1989, Tokyo, Japan. **Proceedings...** Amesterdan: North-Holland, 1989. p.105-125.

[WUU 93] WUU, G.T.J.; DAYAL, U. A Uniform model temporal and versioned object-oriented databases. In: **Temporal Databases**: Theory, Design, and Implementation. Bridge Parkway: Benjamin/Cummings, 1993. p230-247.

[ZLO 77] ZLOOF, M. M. Query-by-Example: A Database Language. **IBM Syst. Journal**, New York, v.16, n.4, p.324-343, 1977.

Informática UFRGS

CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Implementação de Consultas para um Modelo de Dados Temporal Orientado a Objetos

por

Tanisi Pereira de Carvalho

Dissertação apresentada aos Senhores:

Profa. Dra. Claudia Maria Bauzer Medeiros (UNICAMP)

Prof. Dr. Duncan Dubugras Alcoba Ruiz (PUC - RS)

Prof. Dr. José Palazzo Moreira de Oliveira

Vista e permitida a impressão.
Porto Alegre, 30/04/97.

Profa. Dra. Nina Edelweiss,
Orientador.