Formalizando Interação Visual com Bancos de Dados Históricos

Sônia Leila Fernandes Silva

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

Área de Concentração: Processamento da Informação

Ulrich Schiel e Tiziana Catarci
Orientadores

Campina Grande, Paraíba, Brasil

S586f Silva, Sonia Leila Fernandes
Formalizando interacao visual com bancos de dados historicos / Sonia Leila Fernandes Silva. - Campina Grande, 1999.
135 f. : il.

Tese (Doutorado em Engenharia Eletrica) - Universidade Federal da Paraiba, Centro de Ciencias e Tecnologia.

1. Bancos de Dados Historicos 2. Sistemas Visuais de Consulta 3. Tese I. Schiel, Ulrich, Dr. II. Catarci, Tiziana, Dra. III. Universidade Federal da Paraiba - Campina Grande (PB) IV. Título

CDU 681.3.07(043)

FORMALIZANDO INTERAÇÃO VISUAL COM BANCO DE DADOS HISTÓRICOS

SÔNIA LEILA FERNANDES DA SILVA

Tese Aprovada em 05.11.1999

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

TIZIANA CATARCI, Ph.D., Univ. La Sapienza-Itália
Orientadora

NINA EDELWEISS, Dr., UFRGS
Componente da Banca

ANA CAROLINA SALGADO, Dr., UFPE
Componente da Banca

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

EDILSON FERNEDA, Dr., UFPB
Componente da Banca

BERNARDO LULA JUNIOR, Dr., UFPB
Componente da Banca

CAMPINA GRANDE - PB
Novembro - 1999

Dedicatória

Dedico ao meu pai **Raimundo Marques** e à minha mãe **Carmen Fernandes**.

Agradecimentos

Existem muitas pessoas que colaboraram para a realização deste trabalho.

Agradeço a **Deus** por tudo.

Agradeço aos **meus familiares** que sempre me apoiaram e me deram força para eu não desistir dos meus objetivos.

Agradeço a **todos os meus amigos** que eu cultivei aqui em Campina Grande, e aos meus amigos que estão longe e que deixo muitas saudades.

Agradeço ao meu orientador **Ulrich Schiel** pelas nossas "polêmicas" e utilíssimas discussões e pelo seu companheirismo.

Agradeço ao **CNPq** pelo suporte financeiro.

Agradeço à **COPELE** pelo apoio que me foi sempre dado, em particular, agradeço à secretária **Ângela** pela sua disponibilidade em sempre me ajudar nos momentos difíceis.

Agradeço à **família Jordão** pela hospedagem, amizade e força, que tornou decisivo para a concretização da tese durante a sua etapa final.

Não tenho palavras para agradecer à minha orientadora **Tiziana Catarci**. Sua objetividade, determinação e paciência foram fundamentais em tornar realidade esta tese.

E, acima de tudo, agradeço ao meu namorado **Giulio** por seu amor, amizade, força, paciência, e por sempre estar presente em corpo e espírito. *"Per lui che sará sempre in mio cuore, un grandissimo BACIO per ringraziarlo".*

Resumo

Aplicações recentes de banco de dados são tipicamente orientadas a um grande número de usuários não especialistas que necessitam ter acesso a interfaces adequadas a facilitar sua interação com o sistema. Adicionalmente, a incorporação do **tempo** em sistemas de banco de dados é uma característica importante em certas aplicações. De fato, diversos modelos de dados temporais e suas correspondentes linguagens de consulta textuais foram propostos. Contudo, existe pouca investigação sobre linguagens visuais para consultar bancos de dados temporais. Nosso trabalho tem como objetivo atender a tal necessidade. Em particular, propomos um ambiente visual de consultas, denominado **TVQE** (**T***emporal* **V***isual* **Q***uery* **E***nvironment*), o qual facilita a interação do usuário com bancos de dados *históricos*. O sistema adota uma representação diagramática do esquema conceitual do banco de dados (incluindo classes e relacionamentos temporais) e uma "agenda gráfica" como metáfora de interação. Usuários não especialistas em banco de dados são liberados de dificuldades sintáticas, típicas de linguagens textuais, e eles podem expressar consultas temporais através de operações gráficas elementares. Diferente de muitas propostas no campo de linguagens visuais de consulta, a linguagem sobre a qual o TVQE é baseado é fornecida com sintaxe e semântica formal. TVQE é baseado em um conjunto mínimo de primitivas gráficas temporais (**TGPs**), as quais são definidas sobre um modelo gráfico de dados temporal (**TGM**), com sintaxe visual e semântica baseada em objeto. Concentramo-nos principalmente nos aspectos formais do TVQE. Fornecemos também algumas idéias de mecanismos visuais de interação e questões relacionadas à implementação.

Abstract

Recent database applications are typically oriented towards a large set of non-expert users, and therefore, they need to be equipped with suitable interfaces facilitating the interaction with the system. Moreover, the incorporation of time in database systems is a desirable feature. Indeed, severals temporal data models and the corresponding textual query languages have been proposed. However, there is a limited amount of research concerning the investigation of user-oriented languages for querying temporal databases. Our proposal addresses such a need. In particular, we propose a visual query environment, namely TVQE (Temporal Visual Query Environment), which provides an easier interaction of the user with temporal databases. The system adopts a diagrammatic representation of the database schema (including temporal classes and relationships) and a "graphical notebook" as interaction metaphor. In our approach, non-database experts are released from syntactical difficulties which are typical of textual languages, and they can easily express temporal queries by means of elementary graphical operations (e.g. click on a node label). Differently from many proposals in the field of visual query languages, the language underlying TVQE is provided with formal syntax and semantics. It is based on a minimal set of temporal graphical primitives (TGPs), which are defined on a Temporal Graph Model (TGM), with visual syntax and object-based semantics. We mainly concentrate on the formal aspects of TVQE, and provide some ideas on the visual interaction mechanisms and implementation issues.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

Um dos componentes problemáticos na maioria dos sistemas de banco de dados é a **interface do usuário**. Alguns pesquisadores justificaram esse fato. Por exemplo, *Ellis et al.* [41] argumentaram que houve uma preocupação da pesquisa direcionada mais para questões como consistência e eficiência do banco de dados, e menos com **usabilidade**. *R. Cooper* [30] declara que tanto o projeto como a implementação de interfaces para um Sistema de Gerenciamento de Banco de Dados (SGBD) exigem investimentos significativos.

Uma vez que que as interfaces de um SGBD tradicional são baseadas no modelo **relacional** [28], a semântica do domínio da aplicação é pobremente representada [71]. Como consequência, perde-se a visão global e a manipulação conceitual dos dados. No modelo relacional, a estrutura complexa de um objeto se encontra distribuída sob forma de valores entre diferentes relações e a manipulação das entidades é feita através de seus valores, ou seja, a "recomposição" de uma entidade necessita de uma ou várias operações de junção.

Realmente, é difícil capturar uma informação estrutural complexa através de tabelas relacionais, principalmente quando o objeto complexo deve ser manipulado como um todo [96]. Desta forma, tal modelo se tornou inadequado para modelar dados de aplicações mais complexas (ex: multimídia), estimulando a geração de outros modelos (semânticos e orientados a objetos).

As linguagens textuais de consulta[1], acabaram sendo restritas, na sua maioria, a usuários especialistas na área, não sendo muito compreensíveis e amplamente utilizadas por usuários novos ou não especialistas que, além de passarem por um período de aprendizado até alcançarem o domínio pleno da linguagem, são forçados a conhecerem o esquema de banco de dados conceitual e alguns conceitos de álgebra relacional, teoria do conjunto e junção. Consultas em SQL (*Structured Query Language*)[2], por exemplo, frequentemente consomem tempo e são suscetíveis a erros [83]. Outros problemas já foram universalmente reconhecidos (ver [9]).

Outro problema surge na fase de desenvolvimento de um banco de dados para um sistema de informação (Figura 1.1), segundo *Carapuça et al.* [16]. Na fase de análise, se definem as entidades do mundo real que são relevantes ao universo do discurso, utilizando o modelo Entidade-Relacionamento (E-R) [26], por exemplo. A segunda fase incorpora o processo de construção da interface, utilizando os recursos do SGBD (gerador de telas, menus e relatórios). A criação geralmente envolve uma codificação que efetua a integração da aplicação com as rotinas externas da interface do SGBD.

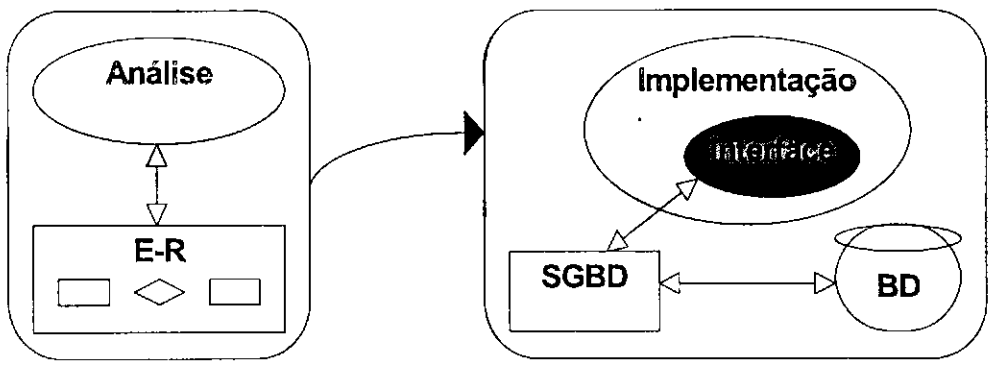

Figura 1.1: Desenvolvimento de um BD para um Sistema de Informação

[1]Uma consulta básica consiste de uma lista alvo e uma condição opcional. A lista alvo especifica os dados de interesse para o resultado da consulta, conhecida como uma projeção sobre os dados. A condição é composta de um ou mais predicados que restringem os dados relevantes à consulta, conhecida como seleção sobre os dados.

[2]SQL é a linguagem de consulta padrão dos SGBDs relacionais, baseada no cálculo relacional e orientada a tupla. Cálculo relacional é um caso particular do cálculo de predicados, usado para operar sobre bancos de dados relacionais.

Carapuça et al. relatam [16] que todo este processo tem a seguinte consequência: a interface obtida, na maioria das vezes, não capta toda a semântica que envolve as entidades do modelo conceitual. Além disso, a criação da interface na última fase reforça a idéia de que ela, na maioria das vezes, é considerada somente do ponto de vista estético [23].

Diante da disponibilidade de *softwares* gráficos que se tornaram mais populares entre a comunidade de usuários de computação, e do advento da **manipulação direta** [99], interfaces visuais para banco de dados surgiram como alternativas para suprir as deficiências das linguagens de consulta não procedurais, considerando dois aspectos: a interação entre o usuário e uma base de dados, e as limitações dos modelos de dados em que elas se baseiam diante de aplicações mais complexas. Desta forma, as interfaces visuais facilitam a especificação de consulta do usuário.

Além das limitações das linguagens convencionais discutidas acima, existem outros fatores que serviram como estímulo à adoção de técnicas visuais nas interfaces de consulta modernas e sua ampla difusão:

- A utilização de imagens se tornou crucial dentro da interação homem-máquina. A abordagem visual atrai a atenção do usuário e estimula-o a explorar todas as funcionalidades disponíveis no sistema.

- O surgimento de diferentes tipos de usuário que acessam o banco de dados, tornando-os mais exigentes em acessar os dados em diferentes formas.

- O próprio processo de formulação de consulta é totalmente adequado para a utilização de técnicas visuais, o qual é constituído de três fases: **Localização**, onde o usuário seleciona a parte do banco de dados sobre a qual ele quer operar; **Manipulação**, onde o usuário define as relações dentro da parte selecionada de forma a produzir o resultado da consulta; e **Visualização**, onde o usuário opera sobre o resultado da consulta, de forma a verificar se as informações correspondem ao que ele especificou na sua consulta.

Propostas iniciais surgiram da exploração gráfica dos modelos semânticos [64], como construtores da linguagem de consulta. Mais tarde, esta abordagem foi aplicada aos modelos orientados a objeto, onde o acesso aos dados é feito através da "navegação" sobre a estrutura complexa de um esquema conceitual (intenção do banco de dados), percorrendo sobre as classes e relacionamentos hierárquicos entre classes. A visualização do domínio da aplicação se torna mais próximo da visão real que o usuário tem sobre os dados.

1.1 Motivação do Trabalho

Com a abordagem **centrada no usuário** para o desenvolvimento de sistemas de informação, diversas atividades de pesquisa sobre sistemas visuais foram realizados nos últimos anos. Entre os sistemas especificamente devotados à **extração da informação** proveniente de um banco de dados, existem os **Sistemas Visuais de Consulta - SVCs** (ver [23] para um resumo sobre SVCs).

Os SVCs incluem uma linguagem para expressar consultas em um formalismo visual ([59]) e uma estratégia de consulta ou interação. Eles são uma alternativa às linguagens textuais de consulta, que são dificilmente exploradas por usuários novatos; e são orientados a um amplo espectro de usuários que têm habilidades técnicas limitadas e geralmente ignoram a estrutura interna do banco de dados acessado.

Contudo, SVCs atuam essencialmente em banco de dados convencionais, enquanto que existe um número limitado de pesquisas em relação à investigação de ambientes amigáveis para consultar banco de dados **temporais**, a despeito de numerosos artigos que foram publicados considerando o fator temporal relevante em diversas aplicações modernas de banco de dados, tais como aplicações bancárias, registros médicos, reservas aéreas, sistemas de informação geográfica (SIG), sistemas multimídia, *data warehouse*, etc (ver [112] para uma bibliografia atualizada sobre bancos de dados temporais).

Realmente, informações temporais tais como valores temporais, restrições temporais e características de evolução temporal estão presentes em um grande número de aplicações do mundo real.

Por outro lado, houve muitas propostas de linguagens textuais temporais, onde cláusulas e predicados são adicionados à linguagem original para manipular aspectos temporais. Existem linguagens textuais temporais para o modelo relacional, modelo E-R e modelos de orientação a objeto (importantes referências podem ser encontradas em [27], [89], [116], [112]).

Estas linguagens apresentam os mesmos problemas de usabilidade das linguagens de consulta originais, tais como:

1. A necessidade de aprender uma sintaxe específica ao invés de simplesmente selecionar alguns ícones ou figuras e declarar alguma condição. Um erro na declaração de uma consulta léxica pode apenas ser detectado depois do término da execução da consulta inteira, ao passo que um erro no ambiente visual é imediatamente reconhecido uma vez que a formulação da consulta é feita passo a passo com *feedback* imediato depois de cada passo.

2. Considerando especificamente as linguagens textuais existentes, o usuário deve conhecer a estrutura interna do banco de dados de forma a declarar a sua consulta. Em um banco de dados temporal, o usuário também deve estar familiarizado com conceitos do tipo *time-stamping* da tupla e do atributo, lógica de intervalos e junções temporais.

Como esforços foram feitos para encontrar novos mecanismos visuais para acessar banco de dados convencionais, isto deveria ser feito em banco de dados temporais. Esquemas conceituais adequados são necessários e novos mecanismos visuais devem ser criados para manipular aspectos temporais.

1.2 Descrição geral da tese

Foi projetado inicialmente um SVC para banco de dados históricos [51], como uma primeira tentativa em formular visualmente consultas temporais. O SVC original foi estendido com novas características, tais como:

- Visualização hierárquica de um esquema complexo;
- Uso de uma metáfora familiar de forma a facilitar a especificação da consulta [50];
- Visualização intensional do resultado da consulta;
- Modelagem de dados sobre a interação do usuário dentro do ambiente de consultas [47], [49].

A versão estendida do sistema é denominada **TVQE** (*Temporal Visual Query Environment*) [48]. A principal idéia do sistema é fornecer ao usuário um ambiente visual de consultas. Em tal ambiente, as diferentes atividades de especificação da consulta são executadas de uma forma homogênea, através de operações gráficas elementares, explorando uma metáfora visual independente de modelo. Desta forma, um usuário não especialista nem precisa entender o modelo de dados temporal interno, nem a sintaxe e semântica de uma linguagem temporal.

Neste trabalho, concentramo-nos sobre os aspectos formais do TVQE. Mais especificamente, apresentamos a formalização de um modelo de dados com sintaxe visual e semântica baseada em objeto, denominada de **TGM** (*Temporal Graph Model*) e das primitivas gráficas temporais (**TGPs** - *Temporal Graphical Primitives*), sobre as quais o ambiente TVQE é baseado. Apresentamos também os itens descritos anteriormente, os quais se referem à atividade de formulação de uma consulta.

1.3 Objetivos da Tese

Os objetivos principais deste trabalho são os seguintes:

- Definir formalmente a sintaxe e semântica de uma linguagem visual de consulta para banco de dados históricos, através da criação do modelo TGM e das primitivas gráficas temporais.
- Unificar o poder expressivo desta linguagem e sua facilidade de uso através da utilização de operações gráficas elementares.

- Propor um sistema visual de consulta com objetivo de validar as primitivas gráficas temporais.

- Permitir que usuários não especializados em banco de dados sejam liberados das dificuldades sintáticas, típicas de linguagens textuais, de forma que eles possam facilmente expressar consultas temporais, utilizando uma metáfora gráfica.

1.4 Estrutura da Tese

O trabalho é organizado da seguinte forma.

O Capítulo 2 apresenta a especificação formal do modelo de dados TGM.

O Capítulo 3 apresenta a especificação formal das primitivas gráficas temporais (TGPs) e descreve o resultado de como a execução da consulta é processada.

O Capítulo 4 apresenta uma análise da evolução de sistemas visuais de consulta nos últimos anos.

O Capítulo 5 apresenta o cenário concreto do ambiente TVQE, de forma a ilustrar como uma consulta temporal pode ser expressa visualmente, e descreve alguns trabalhos relacionados.

O Capítulo 6 descreve os aspectos de implementação do ambiente TVQE.

O Capítulo 7 relata como a história das interações do usuário pode ser acessada e dinamicamente mantida com a utilização das primitivas gráficas temporais, baseado na modelagem dos aspectos temporais de um modelo de usuário.

O Capítulo 8 descreve os resultados alcançados e estabelece as direções de pesquisa para este trabalho.

Capítulo 2

Especificação Formal de Bancos de Dados Históricos

Um banco de dados e uma linguagem de consulta são formalmente definidos em termos de um modelo de dados e um conjunto de operadores, respectivamente. Um modelo de dados fornece um conjunto de mecanismos estruturais que são expressos em termos de uma **representação**. O mesmo se aplica a uma linguagem de consulta, cujos operadores devem ser representados de alguma forma para serem utilizados.

Contudo, em períodos passados, considerando o par ⟨**modelo, representação**⟩, o componente de maior relevância sempre foi o modelo, enquanto que, recentemente, com o crescimento da importância dada à interação homem-máquina, o segundo componente cresceu significativamente em importância. De forma a atender esta exigência, uma possível solução é **unificar** o modelo de dados e sua representação gráfica, aplicando diretamente operações gráficas (com suas próprias semânticas) sobre o modelo.

Neste contexto, apresentamos o modelo de dados temporal TGM (*Temporal Graph Model*), como um formalismo baseado em grafo na representação e no acesso a bancos de dados históricos, onde a representação visual é parte do próprio modelo. O resto do capítulo é organizado da seguinte forma: Na seção 2.1 são descritos alguns conceitos temporais que são incorporados em modelos de dados. Na seção 2.2 é introduzido o modelo TGM.

2.1 Ontologia do Tempo em Modelos de Dados

A noção de tempo, como datas, duração da validade das informações e intervalos temporais, surge em diferentes níveis: na modelagem de dados; na linguagem de recuperação e manipulação dos dados; e no nível de implementação do SGBD. A modelagem de aspectos temporais é um importante tópico dentro dos modelos de dados. A possibilidade de armazenar, manipular e recuperar dados temporais deve ser considerada quando da escolha de um método de modelagem.

2.1.1 Ordem no Tempo

A definição de uma ordem a ser seguida no tempo é fundamental quando utiliza-se alguma representação temporal. O mais comum é o tempo que flui **linearmente** do passado para o futuro. Isto implica em uma ordenação total entre quaisquer dois pontos no tempo. Definido dois pontos diferentes no tempo t e t', representando a ordem de precedência temporal através do operador $\prec$, uma das seguintes expressões é verdadeira: $t \prec t'$ ou $t' \prec t$.

Em alguns casos pode ser considerado um tempo **ramificado** com possíveis passados e futuros, ou seja, o tempo é linear do passado ao tempo corrente, e, a partir deste ponto, o tempo se divide em diversas linhas de tempo, permitindo a possibilidade de dois pontos diferentes serem sucessores imediatos de um mesmo ponto. Neste caso, a estrutura de um tempo ramificado é uma árvore cuja raiz representa o tempo corrente.

Uma última opção de ordenação temporal é considerar o tempo **circular**. Esta forma pode ser utilizada para modelar eventos e processos recorrentes. Um exemplo é uma semana, onde após sete dias, o mesmo dia volta a ocorrer.

2.1.2 Unidade de Tempo

Entre os tipos de unidades temporais, o tipo mais básico é um **instante** que representa um ponto em uma linha de tempo.

Um **evento** é um fato instantâneo ocorrendo em algum instante. O tempo de

ocorrência de um evento é o instante no qual o evento ocorre no mundo real.

Um **período** de tempo é o tempo decorrido entre dois instantes definidos como **início** e **fim**.

Um **intervalo** de tempo é uma duração do tempo. Uma duração é uma quantidade de tempo com tamanho conhecido, mas sem nenhum instante inicial ou final específico (ex. uma semana).

Um *chronon* é um intervalo de tempo com duração mínima que não pode ser decomposto.

Elementos temporais são uniões finitas de períodos disjuntos entre si (uma descrição de elementos temporais podem ser encontrada em [55]).

2.1.3 Densidade no Tempo

Considerando o tempo linear, a densidade temporal pode ser de três tipos: discreta, densa ou contínua.

O tempo **discreto** indica que os instantes são isomorfos aos números naturais, ou seja, cada instante possui um único sucessor. Em um modelo discreto, a linha de tempo é composta de uma sequência de *chronons*.

O tempo **denso** indica que os instantes são isomorfos aos números racionais, ou seja, entre quaisquer dois instantes de tempo, sempre existe outro instante.

O tempo **contínuo** indica que os instantes são isomorfos aos números reais, isto é, ele é denso e diferente dos números racionais, pois não possui espaços e a cada número real corresponde um instante.

2.1.4 Granularidade

A granularidade consiste na duração de um *chronon*. As granularidades mais utilizadas são as que fazem parte do sistema de **calendário** (segundo, minuto, hora, dia, mês, ano, etc) e, dependendo da aplicação, podem ser necessárias várias granularidades. Esta capacidade dá ao usuário a facilidade de tratar informações temporais em vários níveis de abstrações.

2.1.5 Dimensão de Tempo

Existem duas dimensões ortogonais de tempo que podem ser associadas aos fatos de um banco de dados: o tempo válido e o tempo de transação.

O tempo **válido** é o tempo em que um fato é verdadeiro na realidade modelada. O tempo válido de um evento é o tempo no qual o evento ocorreu no mundo real, independente do registro daquele evento no banco de dados. Tempos válidos podem representar tempos no futuro, onde se espera que um fato será verdadeiro a um tempo especificado depois do tempo corrente.

O tempo de **transação** é o tempo em que um fato é armazenado no banco de dados. O tempo de transação de um fato identifica a transação que inseriu o fato no banco de dados e a transação que removeu este fato do banco de dados.

2.1.6 Tempo Absoluto e Relativo

O tempo **absoluto** indica que um tempo válido está associado a um fato e é independente de um outro tempo específico (ex: o salário de Maria aumentou em 30/03/1999).

O tempo **relativo** indica que o tempo válido associado a um fato é determinado pelo tempo válido de um outro fato. O relacionamento entre estes dois tempos pode ser qualitativo (antes, durante, etc) ou quantitativo (três dia antes, vinte anos depois, etc).

2.1.7 Classificação de banco de dados temporais

Os bancos de dados temporais são classificados como: **BDs de tempo válido**, onde ao dado é associado o tempo válido, ou seja, o armazenamento das informações varia com o tempo, podendo obter-se o histórico das mesmas.

BDs de tempo transação, onde são armazenados todos os estados passados do BD, utilizando o tempo de transação.

BDs bitemporais, que suporta tanto o tempo válido quando o tempo de transação.

2.2 O Modelo Temporal baseado em Grafo

Nesta seção apresentamos o modelo **TGM** (*Temporal Graph Model*) como um formalismo baseado em grafo para representar e consultar bancos de dados temporais. Ele representa a **extensão temporal** do modelo *Graph Model*, originariamente proposto em [19] e descrito com mais detalhes em [21].

Um banco de dados representado pelo modelo TGM, denominado ***TGMDB (Temporal Graph Model Database)***, é uma tripla $\langle g, c, m \rangle$, onde g é uma estrutura orientada a grafo, denominada *Typed Graph*, c é um conjunto de **Restrições de Integridade** imposta sobre as classes de objetos representadas em g, e m é uma estrutura que corresponde ao nível extensional do banco de dados, denominada de **Interpretação**.

O esquema de um banco de dados, ou seja, sua parte intensional, é representada no TGM pelo *Typed Graph* e um conjunto de Restrições. As instâncias de um banco de dados, ou seja, sua parte extensional, são representadas pela noção de Interpretação.

Um esquema de banco de dados é expresso no *Typed Graph* em termos de classes e relacionamentos entre classes (denominados de papéis). Uma classe é uma abstração de um conjunto de objetos com características comuns. Um relacionamento entre classes $C_1, \ldots, C_n$ representa associações entre objetos das classes $C_1, \ldots, C_n$.

Em TGM, classes e relacionamentos podem também ser modelados com características de banco de dados **temporais**, permitindo ao usuário recuperar dados não apenas em relação ao estado corrente do banco de dados, mas também seus estados passados (registros históricos). Neste trabalho, consideramos bancos de dados **históricos**, onde aos dados são associados valores de um domínio **ordenado** e **discreto** de intervalos temporais, o qual consiste de pares ordenados de números naturais.

Mais formalmente, o ***Typed Graph*** g é uma tupla $\langle N, E, L_1, L_2, f_1, f_2, f_3 \rangle$, onde:

$N = N_c \cup N_r$ é o conjunto de **nodos**, onde N_c e N_r são mutuamente disjuntos; N_c é o conjunto de **nodos-classe**, representando as classes, e N_r é o conjunto de **nodos-papel** representando os relacionamentos.

N_c é particionado em N_{c_p}, o conjunto de nodos **imprimíveis**, o qual representa o conjunto de classes cujas instâncias são valores de domínio (ex., inteiro, string, etc); N_{c_u}, o conjunto de nodos **não imprimíveis**, o qual representa o conjunto de classes cujas instâncias são identificadores de objeto (ex., Pessoa, Empregado); $N_{c_{t_p}}$, o qual representa o conjunto de nodos **temporais imprimíveis**; e $N_{c_{t_u}}$, o qual representa o conjunto de nodos **temporais não imprimíveis**.

N_r é particionado em N_{r_n}, o conjunto de **relacionamentos não temporais**, e N_{r_t}, o conjunto de **relacionamentos temporais**.

$E \subseteq N \times N$ é um conjunto de **arcos**;

L_1 é um conjunto de **rótulos dos nodos**;

L_2 é um conjunto de **rótulos dos arcos**, incluindo **um rótulo especial** L[1];

f_1 é uma função de N para L_1, associando um rótulo a cada nodo;

f_2 é uma função de E para L_2, associando um rótulo a cada arco;

f_3 é uma função que caracteriza o estado de seleção dos elementos do *Typed Graph*, mapeando cada nodo a um valor em {**não selecionado, selecionado, visualizado**}.

Os rótulos em L_1 são simplesmente nomes de nodos, ao passo que os rótulos em L_2 representam ou operações de conjunto ou expressões booleanas, e são usadas no processo de formulação de uma consulta (ver próximo capítulo).

Em relação ao modelo original, o *Typed Graph* foi estendido com a inclusão dos nodos temporais.

[1] O rótulo especial era representado como T no modelo original, mudamos para L para evitar confusão com o conjunto de pontos de tempo, definido a seguir.

Com o objetivo de descrever a noção de Interpretação em bancos de dados históricos, estendemos o modelo original com a introdução das seguintes notações:

Denotamos com $AD(n)$ o conjunto de **nodos adjacentes** a um dado nodo n. Se n é um nodo-papel, então $\forall n_i \in AD\{n\}(n_i \in N_c)$. Denotamos com $|AD(n)|$ a cardinalidade de $AD(n)$.

$T = \{\ldots, t_1, t_2, \ldots\} \cup \{now\}$ é o conjunto *ordenado* e *discreto* de **pontos de tempo** acrescido do valor especial *now*, onde *now* é o tempo corrente mudando continuamente. Definimos em uma forma natural uma relação de ordem $\leq$ em T ($t_i \leq t_{i+1} \leq \ldots$), que possui as propriedades de *reflexividade, assimetria* e *transitividade.*

Um **intervalo de tempo**[2] I_k é representado como $\langle begin_k, end_k \rangle$, com $begin_k, end_k \in T$ e para todo $t \in T$, se $begin_k \leq t \leq end_k$, então $t \in I_k$. Todos os intervalos no modelo são representados como intervalos **fechados**. Desta forma, um instante t pode ser identificado com o intervalo $\langle t, t \rangle$.

Um *Lifespan ls* é um **elemento temporal**, ou seja, é um conjunto *finito, disjunto* e *ordenado* de intervalos de tempo $\{I_1, \ldots, I_n\}$ Denominamos $LS = \{ls \mid ls = \{I_1, \ldots, I_n\}\}$ o conjunto de todos os *lifespans* sobre T.

$O = O_p \cup O_u \cup O_{t_p} \cup O_{t_u}$ é o conjunto de todos os **objetos atômicos**. O_p são os objetos imprimíveis, O_u são os objetos não imprimíveis, O_{t_p} são os objetos temporais imprimíveis, e O_{t_u} são os objetos temporais não imprimíveis. Note que $O_p, O_u, O_{t_p}, O_{t_u}$ são disjuntos dois a dois.

Definimos o **universo** U como um conjunto de todos os objetos estruturados, definidos como o menor conjunto contendo O, e todas as possíveis tuplas rotuladas

[2] O que denominamos de intervalo de tempo no modelo, corresponde em realidade a um período de tempo na terminologia padrão (ver seção anterior).

(de qualquer aridade) $\langle l_1 : o_1, \ldots, l_k : o_k \rangle$, onde $l_1, \ldots, l_k$ são elementos de L_1, ou seja, rótulos de nodos-classe e $o_1, \ldots, o_k$ são elementos de O.

$U_t \subseteq U$ é um **universo temporal**, isto é, o subconjunto de U constituído pelos objetos temporais O_{t_p} e O_{t_u}, e um conjunto de tuplas rotuladas (de qualquer aridade) $\langle l_1 : o_1, \ldots, l_k : o_k \rangle$, onde $l_1, \ldots, l_k$ são rótulos de nodos-classe, e $o_1, \ldots, o_k$ são elementos de O.

$\theta : U_t \rightarrow LS$ é uma função que associa a cada elemento x de U_t um *lifespan*, denotado por $\theta(x) = ls$. Para cada $I_k \in \theta(x)$, denotamos $I_k = \langle begin_{I_k}(x), end_{I_k}(x) \rangle$, com $x \in U_t$.

A Figura 2.1 ilustra a relação entre o universo U, o universo temporal U_t, o conjunto de todos os *lifespans* LS e o conjunto de pontos de tempo T.

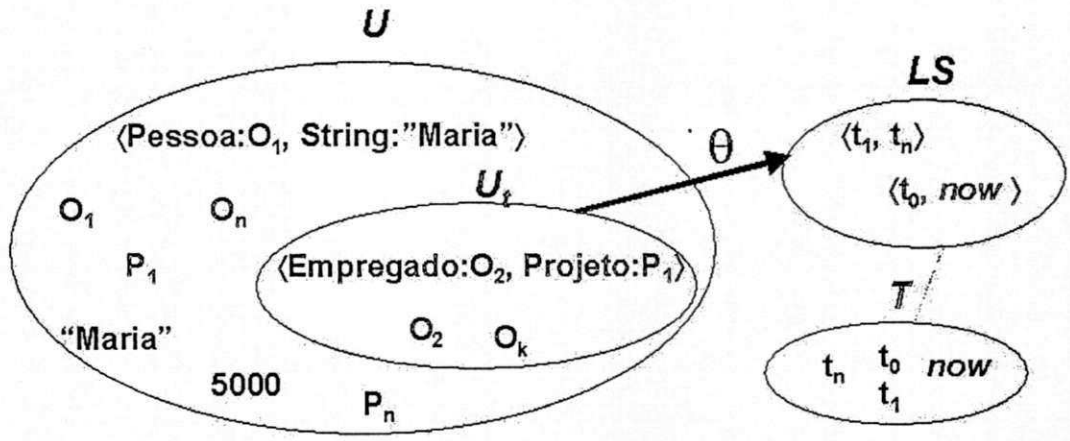

Figura 2.1: Universo U, universo temporal U_t, conjunto LS e o conjunto T

Uma **Interpretação** do Typed Graph g sobre o universo U é uma função mapeando:

- Os nodos-classe imprimíveis de g a um subconjunto de todos os objetos imprimíveis de U;

- Os nodos-classe não imprimíveis de g a um subconjunto de todos os objetos não imprimíveis de U;

- Os nodos-classe temporais imprimíveis de g a um subconjunto de todos os objetos temporais imprimíveis de U_t;

- Os nodos-classe temporais não imprimíveis de g a um subconjunto de todos os objetos temporais não imprimíveis de U_t;

- Os nodos-papel a um subconjunto de todas a tuplas rotuladas de U.

Em particular, dado um nodo-papel n, sua Interpretação é constituída por um conjunto de tuplas rotuladas cuja aridade é igual ao número de nodos-classe que são adjacentes a n.

Formalmente, uma **Interpretação** de um *Typed Graph* é uma função $m : N \rightarrow 2^{(U-U_t)} \cup 2^{(U_t \times LS)}$ mapeando cada nodo $n \in N$ a um subconjunto de U ou de $U_t \times LS$, como segue:

Se $n \in N_{c_p}$ então $m(n) \subseteq O_p$;

Se $n \in N_{c_u}$ então $m(n) \subseteq O_u$;

Se $n \in N_{c_{t_p}}$ então $m(n) \subseteq \{\langle o, \theta(o)\rangle \mid o \in O_{t_p}\}$;

Se $n \in N_{c_{t_u}}$ então $m(n) \subseteq \{\langle o, \theta(o)\rangle \mid o \in O_{t_u}\}$;

Se $n \in N_{r_n}$ e $\{n_1, n_2, \ldots, n_k\} = AD\{n\}$, então $m(n)$ é um conjunto de tuplas da forma $\langle f_l(n_1) : o_1, \ldots, f_l(n_k) : o_k\rangle$, onde $f_l(n_1), \ldots, f_l(n_k) \in L_1$ e $o_1, \ldots, o_k \subset m(n_1) \times m(n_2) \times \ldots \times m(n_k)$;

Se $n \in N_{r_t}$ e $\{n_1, n_2, \ldots, n_k\} = AD\{n\}$, então $m(n)$ é um conjunto de tuplas, cada um tendo um intervalo de tempo associado, ou seja: $m(n) \subseteq \{x, \theta(x)\}$, onde x é uma tupla da forma $\langle f_l(n_1) : o_1, \ldots, f_l(n_k) : o_k \rangle$, onde $f_l(n_1), \ldots, f_l(n_k) \in L$ e $o_1, \ldots, o_k \subset m(n_1) \times m(n_2) \times \ldots \times m(n_k)$.

O conjunto de **Restrições de Integridade** c é especificado pelo projetista de banco de dados, utilizando uma linguagem apropriada (maiores detalhes sobre a linguagem de restrição podem ser vistos em [20]), de forma a expressar condições e propriedades relevantes das classes e relacionamentos. As restrições que são relevantes para este trabalho são as seguintes:

$$n_1 ISA n_2 \Longleftrightarrow m(n_1) \subseteq m(n_2) \wedge (n_1 \in N_{c_{t_p}} \cup N_{c_{t_u}} \Rightarrow n_2 \in N_{c_{t_p}} \cup N_{c_{t_u}});$$

Isto significa que n_1 representa uma subclasse de n_2 em g. Note que se n_1 é uma classe temporal, a superclasse n_2 também tem que ser temporal. De fato, se os objetos de uma subclasse são temporais, eles também o serão na superclasse. O construtor ISA permite representar a hierarquia de **generalização** (relacionamento subclasse-classe), como é definida em modelos semânticos [64].

Esta restrição foi redefinida em relação ao modelo original, de forma a considerar a hierarquia de generalização em nodos temporais.

$ATLEAST(k, n_1, n_2)$ onde $n_1 \in N_c, n_2 \in N_r, k \in Z$ é satisfeito por m se o número de tuplas rotuladas em $m(n_2)$ que contém elementos de $m(n_1)$ no n_1-*componente* é maior ou igual a k.

$ATMOST(k, n_1, n_2)$ onde $n_1 \in N_c, n_2 \in N_r, k \in Z$ é satisfeito por m se o número de tuplas rotuladas em $m(n_2)$ que contém elementos de $m(n_1)$ no n_1-*componente* é menor ou igual a k.

Os construtores $ATLEAST$ and $ATMOST$ permitem expressar **restrições de cardinalidade**.

O construtor $ATLEAST$ significa que todo objeto que é uma instância do nodo-classe n_1 é relacionado a pelo menos k instâncias do nodo-papel n_2, ao passo que o construtor $ATMOST$ significa que todo objeto que é uma instância do nodo-classe n_1 é relacionado ao máximo k instâncias do nodo-papel n_2.

Para realizar a **herança** na generalização, denotamos com $AD'\{n\}$ o conjunto definido como segue:

Se $n \in N_c$ então $AD'\{n\} = AD\{n\} \cup_i AD(n_i)$ onde $n_i \in N_{c_u}$ ou $n_i \in N_{ct_u}$ e $nISA^*n_i$, onde ISA^* é a propriedade transitiva da relação ISA (note que se $n \in N_{c_p}$ ou $n \in N_{ct_p}$, então $AD'\{n\} \equiv AD\{n\}$).

Se $n \in N_r$ então $AD'\{n\} = AD\{n\} \cup \{m \in N_c | \exists n' \in AD\{n\} \wedge mISA^*n'\}$.

Em outras palavras, se n é um nodo-classe, o conjunto $AD'\{n\}$ contém tanto os nodos adjacentes a n como os nodos adjacentes a seus ancestrais, dentro da hieraquia ISA. Se n é um nodo-papel, o conjunto $AD'\{n\}$ contém os nodos adjacentes a n e os descendentes de tais nodos.

Consideramos uma nova restrição em relação ao modelo original, definida como segue:

$\{n_1, \ldots, n_k\} PARTOF(n, r) \iff n_1, \ldots, n_k, n \in N_c \wedge r \in N_r \wedge$

$(\forall o \in m(n)\, \exists!\, x \in m(r) \mid x = \langle f_l(n) : o, f_l(n_1) : o_1, \ldots, f_l(n_k) : o_k \rangle) \wedge$

$(\forall o_i, o_j \in m(n), \text{ com } i \neq j (x_i, x_j \in m(r) \wedge x_i = \langle f_l(n) : o_i, f_l(n_1) : o_1, \ldots, f_l(n_k) : o_k \rangle \wedge x_j = \langle f_l(n) : o_j, f_l(n_1) : o_1, \ldots, f_l(n_k) : o_k \rangle)) \Rightarrow \prod_{f_l(n_1),\ldots,f_l(n_k)}(x_i) \neq \prod_{f_l(n_1),\ldots,f_l(n_k)}(x_j))$.

Note que o operador $\prod$ é o equivalente do operador **projeção** como definido na álgebra relacional [28].

A restrição descrita acima modela a hierarquia de **agregação** (relacionamento componente-agregado), como geralmente definida em modelos de dados semânticos, através do construtor $PARTOF$. Uma agregação é definida pelo produto cartesiano das entidades componentes e sua identidade é determinada pelos valores de seus componentes. Para ser semanticamente válido, a construção deve possuir pelo menos dois componentes.

Redefinimos o seu significado como segue: a) Não existe um objeto agregado sem os seus objetos componentes; b) Não pode existir mais de um objeto agregado compartilhando os mesmos objetos componentes.

Vale a pena ressaltar que usamos a restrição $PARTOF$ **apenas** em conjunção com consultas temporais (ver próximo capítulo).

Para **relacionamentos temporais**, definimos a seguinte restrição:

Para cada $n \in N_{r_t}$, e $m(n) \subseteq \{x, \theta(x)\}$, onde x é o conjunto de tuplas $\langle f_l(n_1) : o_1, \ldots, f_l(n_k) : o_k\rangle$; então, o *lifespan* de n deve ser um subconjunto da intersecção dos lifespan*s* dos objetos relacionados, ou seja: $\theta(\langle f_l(n_1) : o_1, \ldots, f_l(n_k) : o_k\rangle) \subseteq \theta(o_1) \cap \ldots \cap \theta(o_k)$.

Deste ponto em diante, quando referimos a um TGMDB $D = \langle g, c, m\rangle$, implicitamente assumimos que a interpretação m satisfaz todas restrições em c.

Um exemplo de utilização do modelo TGM, de forma a modelar a informação sobre uma agência de emprego é mostrado na Figura 2.2, com a seguinte funcionalidade: uma agência de empregos presta serviços aos seus clientes, procurando empregados para elas. Toda pessoa, antes de ser empregada, é cadastrada como candidata a um emprego, e a agência obtém informações cadastrais dos candidatos.

Assim que as empresas fornecerem certas informações (projetos com equipes que necessitam de novos empregados, salário a ser pago, etc) às agências, estas analisam os candidatos cadastrados para que possam encaminhar empregados qualificados de acordo com a demanda da empresa.

Os candidatos que se adequarem às propostas de empregos fornecidas, serão selecionados para se tornarem novos empregados e já são encaminhados para trabalharem em uma atividade dentro de um projeto específico, com salário e nível compatíveis às suas especialidades.

A Figura 2.2 mostra o *Typed Graph g* e a Tabela 2.1 mostra uma possível Interpretação m para g. Por questões de simplicidade, a Interpretação é listada apenas para um subconjunto de nodos.

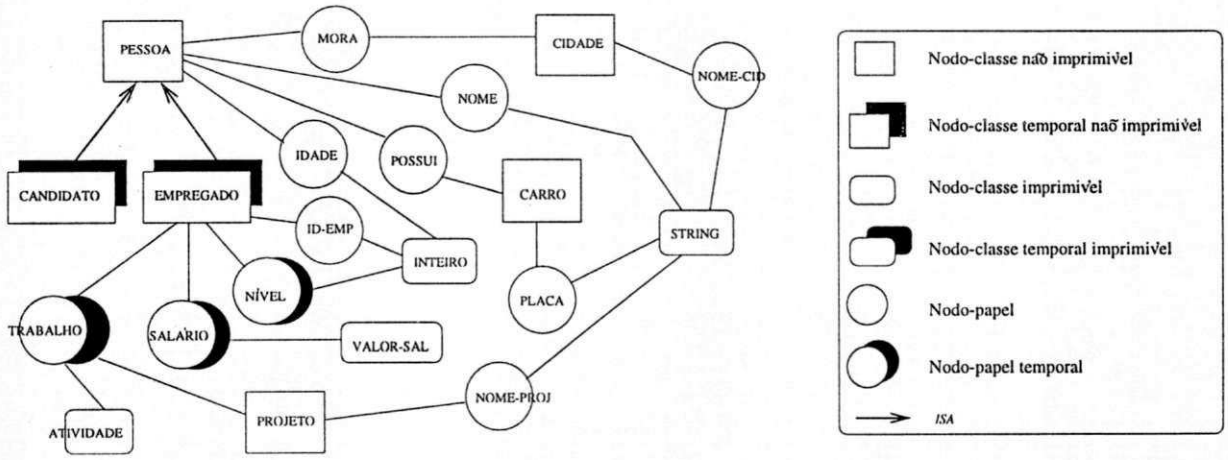

Figura 2.2: Um *Typed Graph g*

2.3 Conclusão deste capítulo

Este capítulo introduziu o modelo TGM como extensão temporal do modelo *Graph Model*, de forma a representar e consultar bancos de dados históricos.

De acordo com a estrutura temporal definida em [57], o modelo TGM compartilha com a maioria dos modelos de dados temporais a mesma estrutura temporal, a qual consiste de dados temporais **discretos, determinados** e **absolutos**. Adicionalmente, TGM suporta uma ordem temporal linear e histórias dos dados em **tempo válido**.

A definição formal do modelo TGM como um grafo permite a unificação do modelo de dados e sua representação gráfica.

$m(Pessoa) = \{\langle o_1, o_2, o_3, o_4\rangle\}$;

$m(Empregado) = \{\langle o_1, \{\langle 1, 10\rangle\}\rangle, \langle o_2, \{\langle 3, 7\rangle\}\rangle, \langle o_3, \{\langle 2, now\rangle\}\rangle\}$;

$m(Cidade) = \{\langle o_5, o_6\rangle\}$;

$m(Inteiro) = Z$;

$m(String) = \{a \ldots z, 1 \ldots 0\}^*$;

$m(Nome) = \{\langle Pessoa : o_1, String : Maria\rangle, \langle Pessoa : o_2, String : Pedro\rangle, \langle Pessoa : o_3, String : John\rangle,$
$\langle Pessoa : o_4, String : Ana\rangle\}$;

$m(Nome - cid) = \{\langle Cidade : o_5, String : SP\rangle, \langle Cidade : o_6, String : RJ\rangle\}$;

$m(Mora) = \{\langle Pessoa : o_1, Cidade : o_5\rangle, \langle Pessoa : o_2, Cidade : o_6\rangle, \langle Pessoa : o_3, Cidade : o_6\rangle,$
$\langle Pessoa : o_4, Cidade : o_5\rangle\}$;

$m(Id\text{-}emp) = \{\langle Empregado : o_1, Inteiro : 0001\rangle, \langle Empregado : o_2, Inteiro : 0010\rangle, \langle Empregado : o_3, Inteiro : 0007\rangle\}$;

$m(Trabalho) = \{\langle Empregado : o_1, Atividade : T1, Projeto : p_1, \{\langle 1, 5\rangle, \langle 8, 10\rangle\}\rangle,$
$\langle Empregado : o_1, Atividade : T2, Projeto : p_2, \{\langle 6, 7\rangle\}\rangle,$
$\langle Empregado : o_2, Atividade : T6, Projeto : p_3, \{\langle 3, 5\rangle\}\rangle,$
$\langle Empregado : o_2, Atividade : T8, Projeto : p_3, \{\langle 6, 7\rangle\}\rangle,$
$\langle Empregado : o_3, Atividade : T1, Projeto : p_1, \{\langle 2, 10\rangle\}\rangle\},$
$\langle Empregado : o_3, Atividade : T4, Projeto : p_2, \{\langle 11, 30\rangle\}\rangle\},$
$\langle Empregado : o_3, Atividade : T3, Projeto : p_1, \{\langle 31, now\rangle\}\rangle\}$;

$m(Salário) = \{\langle Empregado : o_1, valor - sal : 5000, \{\langle 1, 5\rangle, \langle 8, 10\rangle\}\rangle, \langle Empregado : o_1, valor - sal : 6000, \{\langle 6, 7\rangle\}\rangle,$
$\langle Empregado : o_2, valor - sal : 10000, \{\langle 3, 7\rangle\}\rangle, \langle Empregado : o_3, valor - sal : 8000, \{\langle 2, now\rangle\}\rangle\}$.

$m(Nível) = \{\langle Empregado : o_1, Inteiro : 2, \{\langle 1, 10\rangle\}\rangle, \langle Empregado : o_2, Inteiro : 7, \{\langle 3, 7\rangle\}\rangle,$
$\langle Empregado : o_3, Inteiro : 3, \{\langle 2, 10\rangle\}\rangle, \langle Empregado : o_3, Inteiro : 4, \{\langle 11, now\rangle\}\rangle\}$.

Tabela 2.1: Uma Interpretação m para g

A partir desta integração, se permitiu a definição de operações gráficas elementares sobre os componentes da representação gráfica do modelo (ex. seleção de um nodo ou desenho de um arco entre dois nodos), de forma a expressar consultas visuais sobre bancos de dados representado pelo modelo TGM. A sintaxe e a semântica de algumas operações gráficas elementares foram formalmente definidas dentro do modelo original de *Catarci et al.* [19]. Contudo, tais operações gráficas não foram suficientes para expressar consultas históricas.

Portanto, assim como o modelo TGM é uma extensão temporal do modelo *Graph Model*, no próximo capítulo serão introduzidas novas operações gráficas elementares para expressar consultas históricas.

Capítulo 3

As Primitivas Gráficas Temporais

Neste capítulo, definimos as primitivas gráficas **temporais**, com base na idéia de expressar qualquer interação do usuário com um banco de dados, em termos de um conjunto de primitivas gráficas (*Graphical Primitives - GPs*), denominadas de **seleção de um nodo** e **desenho de um arco**, as quais foram definidas baseado no modelo *Graph Model* [19], [21]. É demonstrado que todas as consultas de primeira ordem podem ser expressas através da composição destas duas primitivas, tendo o mesmo poder expressivo das linguagens de consulta relacionais (ver [21]).

Contudo, consultas temporais não podem ser expressas compondo apenas estas duas primitivas, é necessário adicionar primitivas gráficas específicas para a manipulação de informação temporal. Portanto, estendemos o conjunto GP com outras primitivas gráficas, denominadas de primitivas gráficas temporais (*Temporal Graphical Primitives - TGPs*), que são usadas na expressão visual de consultas temporais.

O resto do capítulo é organizado da seguinte forma: Na seção 3.1 é descrito a taxonomia de consultas temporais que podem ser especificadas pelos TGPs. Na seção 3.2 são descritas brevemente as primitivas gráficas. Nas seções 3.3, 3.4, 3.5 e 3.6 são introduzidos os TGPs **seleção de nodo(s) sombreado(s)**, **extensão temporal da mudança do rótulo de um arco**, **seleção do rótulo de um nodo**, e **mudança do rótulo de um nodo**, respectivamente. Na seção 3.7 é descrito o processamento de uma consulta temporal. Na seção 3.8 é descrito um trabalho relacionado.

3.1 Uma Taxonomia para Consultas Históricas

Uma taxonomia para consultas temporais foi proposta em [67]. Aquele documento contém um resumo das principais terminologias utilizadas em banco de dados temporais. De acordo com [67], uma consulta temporal tem dois componentes ortogonais: **Seleção temporal**, o qual é uma condição lógica, baseada em um predicado que envolve o tempo associado com os fatos, e **projeção temporal**, o qual retorna os valores de tempo associados aos dados derivados da seleção temporal.

Uma nova taxonomia para consultas temporais é proposta por *Edelweiss* em [40]. Em [40], as possíveis combinações entre seleção/projeção temporal sobre tempo e dados foram analisadas, resultando em: **seleção/projeção dos dados**, onde condições e resultados se aplicam a valores de dados somente; **seleção/projeção temporal**, onde condições e resultados se aplicam a valores temporais; e **seleção/projeção mista**, onde condições e resultados se aplicam tanto aos dados quanto aos valores temporais.

Esta análise foi combinada com cinco histórias identificadas em um banco de dados **bitemporal**[1]. Em um banco de dados bitemporal, tanto o tempo de transação quanto o tempo válido [103] são armazenados, sobre os quais cinco histórias podem ser identificadas, descritas em [40]:

1. **Dados instantâneos atuais**, representados por todas as informações válidas no momento presente;

2. **Dados instantâneos passados**, representados pelos dados válidos em um determinado instante do passado, de acordo com a atual percepção da história do banco de dados;

3. **Dados instantâneos de história passada**, considerando todas as informações de um determinado momento no passado, de acordo com a história válida naquele momento;

[1] De acordo com [68], existem quatro tipos de banco de dados: instantâneos, de transação, de tempo-válido e bitemporal.

4. **Dados históricos**, nos quais estão incluídas todas as informações armazenadas de acordo com a presente história de dados válidos;

5. **Dados históricos de história passada**, análogos aos anteriores mas considerando uma história anterior à atual, definida por um determinado tempo de transação.

Dentro do contexto de banco de dados de tempo-válido, nosso escopo foi reduzido a duas histórias, como pode ser visto na Tabela 3.1, sem as combinações *seleção dados/projeção dados* e *seleção temporal/projeção temporal*, de acordo com [40] (a primeira representa uma consulta convencional e a última é impossível de representar em uma consulta temporal desde que condições e resultados não se aplicam a valores temporais somente). Esta classificação não inclue consultas que se referem à informação temporal incompleta.

	Dados instantâneos passados	Dados históricos
seleção dados/projeção temporal	ex.3	
seleção dados/projeção mista		ex.4
seleção temporal/projeção dados	ex.1	
seleção temporal/projeção mista		ex.6
seleção mista/projeção dados	ex.2	
seleção mista/projeção temporal		ex.7
seleção mista/projeção mista		ex.5

Tabela 3.1: Diferentes Tipos de Consulta Temporal

Nos seguintes exemplos ilustramos sete casos distintos na Tabela 3.1:

1. Quais foram os salários dos empregados em 10/01/97?

2. Qual foi o salário do João quando ele mudou o seu status pela última vez?

3. Desde quando o grupo de banco de dados trabalha no projeto "Interface para banco de dados temporais"?

4. Qual o histórico salarial dos empregados?

5. Qual o último projeto dos empregados que tiveram um salário inicial maior que 5.000 reais? E desde quando?

6. Qual o histórico dos níveis dos empregados antes do ano de 97?

7. Qual o período que João Silva trabalhou no projeto "Interfaces Visuais Avançadas" durante 1997?

Através da composição de GPs e TGPs, o usuário pode especificar qualquer consulta temporal dentro do domínio de consultas temporais apresentadas na Tabela 3.1.

3.2 Primitivas Gráficas

Como veremos no decorrer deste capítulo, GPs e TGPs consistem de operações gráficas elementares, que podem ser usadas como componentes básicos para expressar visualmente consultas que envolvem seleção e projeção de tempo válido. A semântica dos GPs e TGPs é caracterizada em termos de tranformações do banco de dados, de forma que, na avaliação de uma consulta, a partir de um banco de dados inicial, é obtido um banco de dados *resultante*, contendo exatamente a informação solicitada.

Seja $D = \langle g, c, m \rangle$ um $TGMDB$. Através do GP **seleção de um nodo**, o estado de um nodo é mudado do valor **não selecionado** para o valor **selecionado**, o qual significa incluído no esquema de interesse, ou **visualizado**, o qual significa incluído no resultado da consulta. São mostrados na Figura 3.1 os três diferentes estados do nodo-classe *Projeto*.

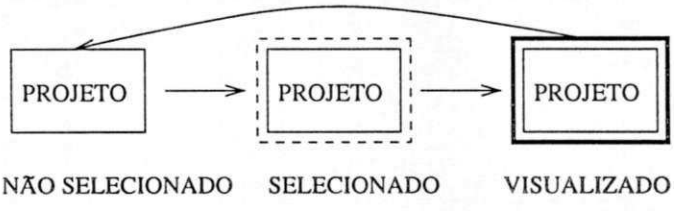

Figura 3.1: GP seleção de um nodo

Seja $D' = \langle g', c', m' \rangle$ o $TGMDB$ resultante da seleção de nodos. O GP **desenho de um arco** é uma função $\mathcal{E}(D', \mathcal{F}, n, q) = D^r$ que cria um arco rotulado com $\mathcal{F}$ entre os nodos n e q, e corresponde à restrição das interpretações dos nodos de acordo com as regras declaradas no rótulo, ou a execução de operações de conjunto sobre eles. Esta primitiva pode ser apenas aplicada quando não existe nenhum arco entre n e q em D'. Seu efeito sobre D' depende de $\mathcal{F}$ e os nodos n e q.

Sejam n e q *nodos-papel*, e seja $\mathcal{F}$ uma *expressão booleana*, durante a construção do banco de dados resultante D° (ver seção 3.7), $\mathcal{E}(D', \mathcal{F}, n, q)$ representa uma restrição da interpretação final. Por exemplo, a Figura 3.2 (a) mostra a restrição de todas as pessoas que possuem carro cuja placa contém o seu nome.

Se $\mathcal{F}$ é um *operador de conjunto*, então n e q são nodos-classe. D^r conterá um novo nodo s e novas restrições ISA (ex. se $\mathcal{F}$ é *união*, então as restrições são n ISA s e q ISA s), como mostra a Figura 3.2 (b). O primeiro nodo s contém todos os objetos que são empregados *ou* estudantes, enquanto que o segundo nodo s contém todos os objetos que são empregados *e* estudantes.

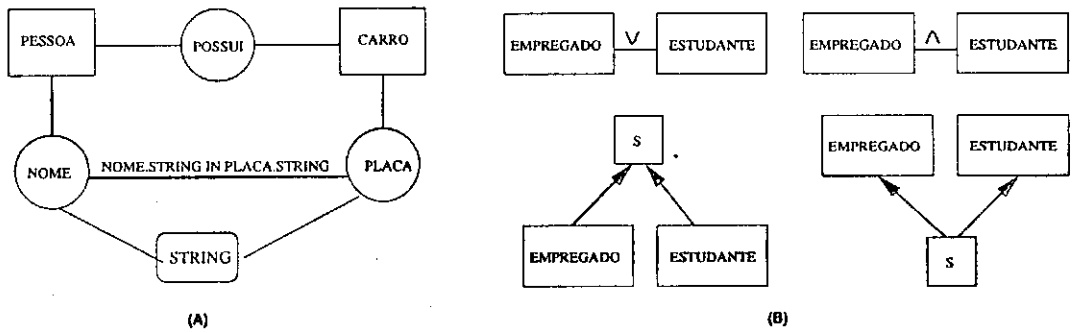

Figura 3.2: GP desenho de um arco

As primitivas descritas anteriormente constituem o conjunto mínimo de ações elementares. Porém, por simplicidade, adicionamos uma nova operação, denominada de **mudança do rótulo de um arco**, relacionando um nodo-classe s a um nodo-papel q. A mudança do rótulo de um arco é denotada com $\mathcal{C}(D', \mathcal{F}_1, s, q)$, onde $\mathcal{F}_1$ é uma fórmula proposicional, cujos elementos são da forma $\alpha R \beta$, onde R é um operador de comparação, α e β representam o componente s das tuplas pertencendo a interpretação de q (referenciada através do rótulo de q) ou constantes.

Durante a construção do banco de dados resultante D°, a presença de rótulos diferente do valor L representa uma restrição da interpretação final.

Vale ressaltar que o mesmo resultado da operação de mudança de um rótulo pode ser obtida através do desenho de um arco reflexivo sobre o nodo-papel q, como mostra a Figura 3.3. Desta forma, a mudança do rótulo de um arco não é definida como uma nova primitiva.

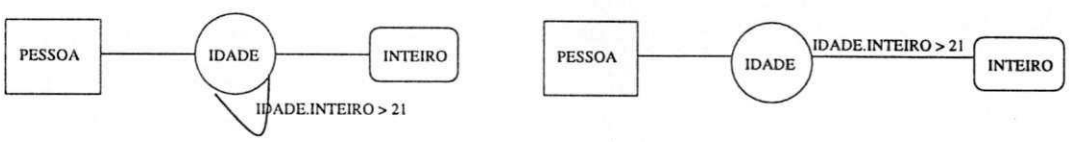

Figura 3.3: GP desenho de um arco e mudança do rótulo de um arco

O formalismo das primitivas gráficas pode ser visto em [19].

3.3 Seleção de nodo(s) sombreado(s)

A seleção da parte sombreada de classes ou relacionamentos temporais corresponde ao TGP **seleção de nodo(s) sombreado(s)**.

A aplicação deste TGP sobre um TGMDB D resulta em um novo TGMDB D', no qual, para os tempos associados aos objetos do nodo sombreado, é criado um nodo explícito contendo intervalos de tempo, que é associado ao nodo original (agora sem sombra), por meio de um nodo de relacionamento, como mostra a Figura 3.4. A classe que representa os intervalos de tempo pode ser manipulada separadamente pelos outros TGPs (os quais são aplicados sobre D'). Pode ser notado a diferença deste operador com a seleção de nodos $n_1, \ldots, n_k$ (se a consulta não envolve aspectos temporais).

Quando este TGP é aplicado a nodos-classe temporais (não) imprimíveis $\{n_i\}$, com $1 \leq i \leq k$, os nodos-classe são transformados em nodos-classe (não) imprimíveis $\{n'_i\}$, contendo todas as instâncias originais sem *lifespans*. Um novo nodo-classe s é criado como uma *agregação* das classes $n'_1, \ldots, n'_k$ com a classe de intervalos de tempo i. A nova interpretação de i contém a **intersecção** dos *lifespans* originalmente associados às classes $n_1, \ldots, n_k$.

Um rótulo especial "all history" é incluído no arco entre os nodos r e i, como mostrado na Figura 3.4 (a), correspondendo à consulta *"Recupere o histórico dos empregados e candidatos"*. Note que o estado do novo nodo-papel assume o estado de *selecionado*, de forma a ser utilizado na construção do banco de dados resultante.

O mesmo procedimento é usado quando a primitiva é aplicada a um conjunto de nodos-papel temporais $\{n_i\}$. A única diferença é que as classes que fazem parte da agregação são as classes $AD(n_1), \ldots, AD(n_k)$ mais o nodo-classe i, como mostrado na Figura 3.4 (b), correspondendo á consulta *"Quais os salários e níveis de cada empregado no ano passado?"*.

Vale ressaltar que as figuras apresentadas representam transformações internas do TGMDB, e são **transparentes** ao usuário final.

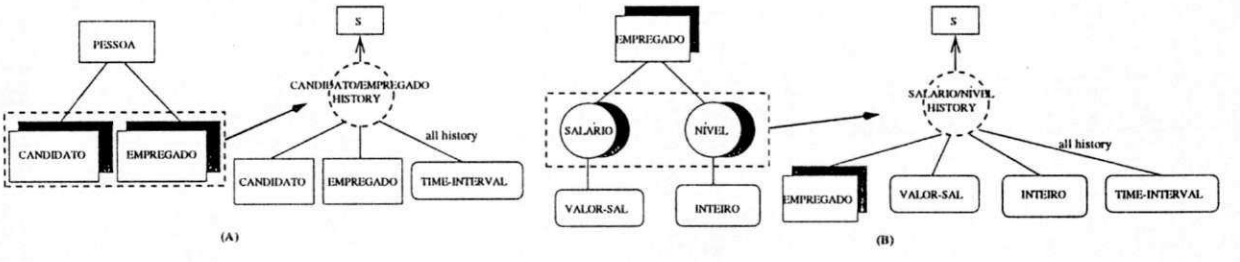

Figura 3.4: TGP Seleção de nodo(s) sombreado(s)

A Tabela 3.2 mostra os efeitos a nível de tupla [55] da seleção dos nodos sombreados dos nodos *Salário* e *Nível*, de forma a obter o nodo-papel não temporal *Salário/Nível-History*.

S	Empregado	Valor-sal	Inteiro	TIME-INTERVAL
J_1	O_1	5000	2	$\langle 1, 5\rangle$
J_2	O_1	6000	2	$\langle 6, 10\rangle$
J_3	O_2	10 000	7	$\langle 3, 7\rangle$
J_4	O_3	8000	3	$\langle 2, 10\rangle$
J_5	O_3	8000	4	$\langle 11, now\rangle$

Tabela 3.2: Interpretação do nodo-papel *Salário/Nível-History*

O operador é definido formalmente como:

Seja $\{n_i\}$ um conjunto de *nodos-classe temporais não imprimíveis*, $\{n_i\} \subset D$, com $1 \leq i \leq k$, a **seleção de nodos sombreados de** $\{n_i\}$ é uma função : $\varpi_n(D, \{n_i\}) = D'$ tal que $D' = D$, exceto que :

$N'_{c_u} = N_{c_u} \cup \{n'_i\} \cup \{s\}$ ($n'_1, \ldots, n'_k, s$ são novos nodos-classe não imprimíveis);
$N'_{c_p} = N_{c_p} \cup \{i\}$ (i é um novo nodo-classe imprimível);
$N'_r = N_r \cup \{r\}$ (r é um novo nodo-papel);
$N'_{c_{t_u}} = N_{c_{t_u}} - \{n_i\}$;
$L'_1 = L_1 \cup \{l_r = l_1 \circ \ldots \circ l_k \circ \text{"}history\text{"} \mid l_i = f_1(n'_i)\}$ ($\circ$ representa concatenação entre rótulos);
$L'_2 = L_2 \cup \{l_e = \text{"}all\ history\text{"}\}$;
$f'_1 = f_1 \cup \{f'_1(r) = l_r, f'_1(i) = \text{"}time - interval\text{"}\}$;
$f'_2 = f_1 \cup \{\langle\langle r, i\rangle, l_e\rangle\}$;
$f'_3 = f_3 \cup \{f'_3(r) = selected\}$;
$c' = c \cup \{\{\{n'_i\}, i\} PARTOF(s, r)\}$;

m' é equivalente a m exceto para $m'(s), m'(n'_1), \ldots, m'(n'_k), m'(i)$ e $m'(r)$ ($m'(s)$ e $m'(r)$ são imediatamente deriváveis de $m'(n'_1), \ldots, m'(n'_k), m'(i)$ e a definição de $PARTOF$):

$m'(s) = \{o_s\}$;
$m'(n'_i) = \{o_{n'_i} \mid \langle o_{n'_i}, \theta(o_{n'_i})\rangle \in m(n_i)\}$;
$m'(i) = \{I \mid \exists o_1 \in m(n_1), \ldots, o_k \in m(n_k)(I = \theta(o_1) \cap \ldots \cap \theta(o_k))\}$.

Um resultado similar é obtido se um subconjunto de $\{n'_i\}$ é composto por nodos-classe imprimíveis temporais. A única diferença é que os nodos deste subconjunto são novos nodos-classe imprimíveis.

Seja $\{n_i\}$ um conjunto de *nodos-papel temporais*, $\{n_i\} \subset D$, com $1 \leq i \leq k$, $AD(n_i) = \{n_{i_1}, \ldots, n_{i_{ci}}\}$, com $ci = |AD(n_i)|$, a **seleção de nodos sombreados de** $\{n_i\}$ é uma função $\varpi_e(D, \{n_i\}) = D'$ tal que $D' = D$, exceto que :

$N'_{c_u} = N_{c_u} \cup \{s\}$ (s é um novo nodo-classe não imprimível);

$N'_{c_p} = N_{c_p} \cup \{i\}$ (i é um novo nodo-classe imprimível);

$N'_r = N_r \cup \{r\}$ (r é um novo nodo-papel);

$N'_{r_t} = N_{r_t} - \{n_i\}$;

$L'_1 = L_1 \cup \{l_r = l_1 \circ \ldots \circ l_k \circ \text{“}history\text{”} \mid l_i = f_1(n_i)\}$;

$f'_1 = f_1 \cup \{f'_1(r) = l_r, f'_1(i) = \text{“}time - interval\text{”}\}$;

$f'_2 = f_1 \cup \{\langle\langle r, i\rangle, \text{“}all\ history\text{”}\rangle\}$; $f'_3 = f_3 \cup \{f'_3(r) = selected\}$;

$c' = c \cup \{\{AD(n_1), \ldots, AD(n_k), i\} PARTOF(s, r)\}$;

$m' = m$ exceto para $m'(s), m'(i)$ ($m'(r)$ é imediatamente derivável de $m'(s), m'(n_{i_1}), \ldots, m'(n_{k_{ck}})$,

$m'(i)$ e a definição de *PARTOF*):

$m'(s) = \{o_s\}$;

$m'(i) = \{I \mid \exists x_i = \langle f_1(n_{i_1}) : o_{i_1}, \ldots, f_1(n_{i_{ci}}) : o_{i_{ci}} \in m(n_i)(I = \theta(x_1) \cap \ldots \cap \theta(x_k))\}$.

Vale ressaltar que as consultas descritas acima envolvem diversos nodos $n_1, \ldots n_k$ para os quais se deseja os tempos **comuns** a eles. Neste caso, a interação visual é uma sequência de seleções sobre os nodos $n_1, \ldots, n_{k-1}$ e uma **dupla** seleção (*double clicking*) sobre o nodo n_k.

Por outro lado, existem consultas em que a informação temporal de cada nodo n_i é manipulada **separadamente**. Neste caso, se deve “selecionar duplamente” sobre cada nodo n_i, como mostrado na figura 3.5, correspondendo a consulta *“Recupere o histórico salarial e o histórico de níveis de cada empregado separadamente”*. Note que os dois nodos-papel resultantes do TGP foram gerados separadamente.

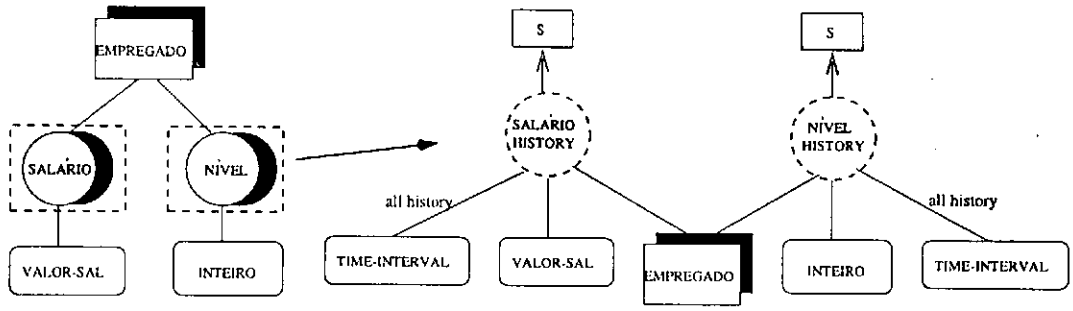

Figura 3.5: Dupla seleção sobre os nodos *Salário* e *Nível*

3.4 Extensão Temporal da Mudança do Rótulo de um Arco

De forma a expressar visualmente um predicado temporal, adicionamos mais uma operação como **extensão temporal do GP mudança do rótulo de um arco**, denotado com $\mathcal{C}_t(D^r, \mathcal{P}, i, r)$, onde o arco relaciona o nodo-papel r e o nodo-classe i, resultantes da seleção de nodo(s) sombreado(s).

A fórmula $\mathcal{P}$ é similar à fórmula original $\mathcal{F}_1$ ($\mathcal{F}_1 = \alpha R \beta$), com a diferença que o operador de comparação R representa os operadores temporais pré-definidos entre intervalos de tempo, definidos por *Allen* [4], da forma $before(I)$, $meets(I)$, $during(I)$, $starts(I)$, $finishes(I)$, $overlaps(I)$, e seus respectivos operadores simétricos $after(I)$, *met-by(I)*, $during(I)$, *started-by(I)*, *finished-by(I)*, *overlaped-by(I)* e $equal(I)$, onde I é um instante ou intervalo de tempo.

Como os predicados de *Allen* são binários (ex. $before(I, J)$), a forma $before(I)$ realmente representa a forma $before(\theta(o), I)$, que significa a recuperação dos objetos com *lifespans* que satisfazem o predicado.

Mostramos na Figura 3.6 a consulta *"Recupere todos os empregados com salário maior que 5000 durante 1987-1988"*. Como consequência desta operação, a presença de rótulos diferentes do valor *"all history"* representará uma restrição da interpretação final.

O operador é definido formalmente como:

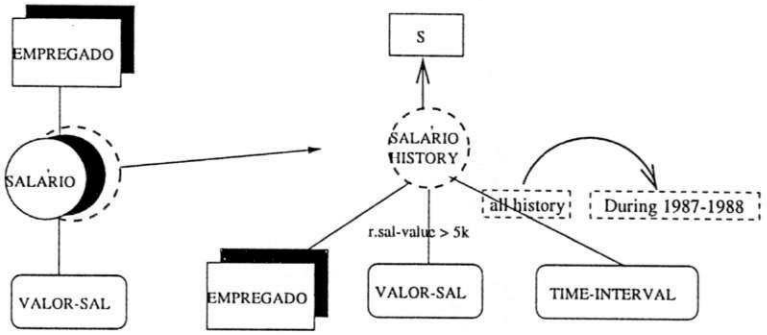

Figura 3.6: Extensão temporal da mudança do rótulo de um arco

Seja r um *nodo-papel resultante da seleção de nodo(s) sombreado(s)*, $r \in D', AD(r) = \{\{n'_i\}, i\}$ ou $AD(r) = \{\{AD(n_1), \ldots, AD(n_k)\}, i\}, AD(n_i) = \{n_{i_1}, \ldots, n_{i_{ci}}\}$, com $ci = |AD(n_i)|$, e $\mathcal{P}(\mathcal{I})$ um *predicado temporal*, a **extensão temporal da mudança do rótulo de um arco associando o nodo-papel r e o nodo-classe** i é uma função $C_t(D', \mathcal{P}(\mathcal{I}), r, i) = D^r$ tal que $D^r = D'$, exceto que:

$f_2^r = f_2' \cup \{\langle\langle r, i\rangle, \mathcal{P}(\mathcal{I})\rangle\}$;

m^r é equivalente a m' exceto para $m^r(r), m^r(i)$:

Se $AD\{r\} = \{\{n'_j\}, i\}$

$m^r(r) = \{x = \langle f_1(s) : o_s, f_1(n'_1) : o_1, \ldots, f_1(n'_k) : o_k, f_1(i) : I'\rangle \mid \exists\{x_j = \langle f_1(s) : o_s, f_1(n'_1) : o_1, \ldots, f_1(n'_k) : o_k, f_1(i) : I\rangle\} \in m'(r) \mid \prod_{f_1(n'_1),\ldots,f_1(n'_k)}(x) = \prod_{f_1(n'_1),\ldots,f_1(n'_k)}\{x_j\} \wedge I' = I \cap \mathcal{P}(\mathcal{I})$;

Se $AD\{r\} = \{\{AD(n_1), \ldots, AD(n_k)\}, i\}$

$m^r(r) = \{x = \langle f_1(s) : o_s, f_1(n_{1_1}) : o_{1_1}, \ldots, f_1(n_{1_{c1}}) : o_{1_{c1}}, \ldots, f_1(n_{k_1}) : o_{k_1}, f_1(n_{k_{ck}}) : o_{k_{ck}}, f_1(i) : I'\rangle \mid \exists\{x_j = langlef_1(s) : o_s, f_1(n_{1_1}) : o_{1_1}, \ldots, f_1(n_{1_{c1}}) : o_{1_{c1}}, \ldots, f_1(n_{k_1}) : o_{k_1}, \ldots, f_1(n_{k_{ck}}) : o_{k_{ck}}, f_1(i) : I\rangle\} \in m'(r) \mid \prod_{f_1(n_{1_1}),\ldots,f_1(n_{1_{c1}}),\ldots,f_1(n_{k_1}),\ldots,f_1(n_{k_{ck}})}(x) = \prod_{f_1(n_{1_1}),\ldots,f_1(n_{1_{c1}}),\ldots,f_1(n_{k_1}),\ldots,f_1(n_{k_{ck}})}\{x_j\} \wedge I' = I \cap \mathcal{P}(\mathcal{I})$;

$m^r(i) = \{I \mid I = \prod_{f_1(i)}(x) \text{ com } x \in m'(r)\}$.

Vale ressaltar que através da composição do TGP seleção de um nodo sombreado e da extensão temporal da operação mudança do rótulo de um arco, expressamos visualmente uma seleção de tempo válida.

3.5 Mudança do Rótulo de um Nodo

Considerando o nodo-papel r, cujas classes relacionadas são $s, \{n'_i\}$ e i ou $s, AD(n_1), \ldots, AD(n_k)$ e i (r resulta da seleção de nodos sombreados de classes ou relacionamento temporais), o usuário pode selecionar um ou mais intervalos de tempo de um *lifespan*, **substituindo o rótulo do nodo** i em um novo rótulo $\mathcal{L}$, o qual representa um número ordinal da forma *1st-interval, 2nd-interval, nth-interval ou last-interval*, como mostrado na Figura 3.7, correspondendo a consulta *"Quando os empregados mudaram os seus trabalhos pela primeira vez?"*.

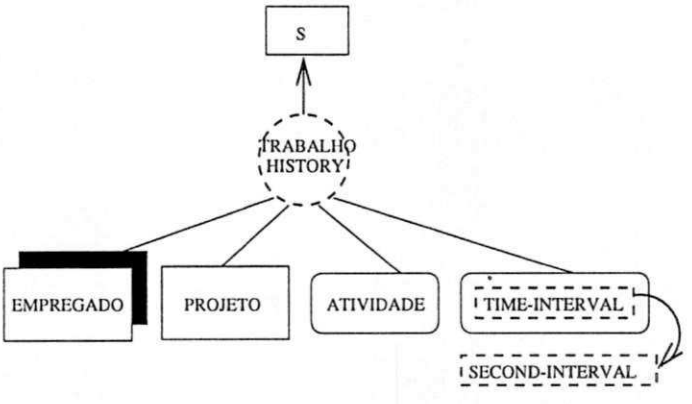

Figura 3.7: TGP mudança do rótulo de um nodo

Desde que os intervalos do *lifespan* dos objetos são ordenados, as tuplas em r que compartilham os mesmos valores, quando projetados sobre as classes $n'_1, \ldots, n'_k$ ou sobre as classes $AD(n_1), \ldots, AD(n_k)$ são *ordenadas no tempo*. Por exemplo, considerando as tuplas $\langle J_1, O_1, T1, P_1\langle 1, 5\rangle\rangle$ e $\langle J_2, O_1, T1, P_1\langle 8, 10\rangle\rangle$ da Tabela 3.3 (esta tabela mostra os efeitos da seleção do nodo sombreado do nodo *Trabalho*, obtendo-se o nodo-papel *Trabalho-History*), com a mudança do rótulo do nodo da Figura 3.7, somente a tupla em destaque permanecerá na interpretação.

O operador é definido formalmente como:

S	Empregado	Atividade	Projeto	TIME-INTERVAL
J_1	O_1	$T1$	P_1	$\langle 1,5\rangle$
J_2	O_1	$T1$	P_1	$\langle 8,10\rangle$
J_3	O_1	$T2$	P_2	$\langle 6,7\rangle$
J_4	O_2	$T6$	P_3	$\langle 3,5\rangle$
J_5	O_2	$T8$	P_3	$\langle 6,7\rangle$
J_6	O_3	$T1$	P_1	$\langle 2,10\rangle$
J_7	O_3	$T4$	P_2	$\langle 11,30\rangle$
J_8	O_3	$T3$	P_1	$\langle 31, now\rangle$

Tabela 3.3: Interpretação do nodo-papel *Trabalho-History*

Seja r um *nodo-papel resultante da seleção de nodo(s) sombreado(s)*, $r \in D', AD(r) = \{\{n'_i\}, i\}$ ou $AD(r) = \{\{AD(n_1), \ldots, AD(n_k)\}, i\}, AD(n_i) = \{n_{i_1}, \ldots, n_{i_{ci}}\}$, com $ci = |AD(n_i)|$, e $\mathcal{L}$ um *parâmetro em {nth-interval, last-interval}*, a **mudança do rótulo do nodo-classe** i é uma função $C_l(D', \mathcal{L}, i) = D^r$ tal que $D^r = D'$, exceto que:

$L_1^r = L_1' \cup \{l_i = \mathcal{L}\}$;

$f_1^r = f_1' \cup \{f_1^i(i) = l_i\}$;

m^r é equivalente a m' exceto para $m^r(r), m^r(i)$:

Se $AD\{r\} = \{\{n'_i\}, i\}$

$m^r(r) = \{x = \langle f_1(s) : o_s, f_1(n'_1) : o_1, \ldots, f_1(n'_k) : o_k, f_1(i) : I'\rangle \mid \exists\{x_j\} \in m'(r) \mid$

$\prod_{f_1(n'_1),\ldots,f_1(n'_k)}(x) = \prod_{f_1(n'_1),\ldots,f_1(n'_k)}\{x_j\} \wedge I' = ord(I_j, \mathcal{L}) \wedge I_j = \prod_{f_1(i)}\{x_j\}\}$;

Se $AD\{r\} = \{\{AD(n_1), \ldots, AD(n_k)\}, i\}$

$m^r(r) = \{x = \langle f_1(s) : o_s, f_1(n_{1_1}) : o_{1_1}, \ldots, f_1(n_{1_{c1}}) : o_{1_{c1}}, \ldots, f_1(n_{k_1}) : o_{k_1}, f_1(n_{k_{ck}}) : o_{k_{ck}}, f_1(i) : I'\rangle \mid \exists\{x_j\} \in m'(r) \mid \prod_{f_1(n_{1_1}),\ldots,f_1(n_{1_{c1}}),\ldots,f_1(n_{k_1}),\ldots,f_1(n_{k_{ck}})}(x) = \prod_{f_1(n_{1_1}),\ldots,f_1(n_{1_{c1}}),\ldots,f_1(n_{k_1}),\ldots,f_1(n_{k_{ck}})}\{x_j\} \wedge I = ord(I_j, \mathcal{L}) \wedge I_j = \prod_{f_1(i)}\{x_j\}\}$;

$m^r(i) = \{I \mid I = \prod_{f_1(i)}(x) \text{ com } x \in m^r(r)\}$.

Se o usuário está interessado em recuperar os instantes iniciais e finais de intervalos selecionados, ele/a pode mudar o rótulo para $begin(\mathcal{L})$ (ou $end(\mathcal{L})$) ou simplesmente *begin* (or *end*), se o usuário não especifica um número ordinal.

Se o usuário também está interessado em recuperar a duração dos intervalos selecionados, ele/a pode mudar o rótulo para $duration(\mathcal{L})$ ou simplesmente *duration*. O resultado é o conjunto de durações $D_1, \ldots, D_n$ com $D_k = end_{I_k} - begin_{I_k}$, onde $I_k = \langle begin_{I_k}, end_{I_k} \rangle$ é uma instância do nodo-classe i. Se o nodo-classe i resulta da aplicação do TGP seleção do rótulo de um nodo (definido a seguir), o resultado é o conjunto de durações $D_1, \ldots, D_n$ com $D_k = \sum_j (end_{I_j} - begin_{I_j})$. Por exemplo, a duração dos intervalos associados com a tupla $\langle O_1, T1, P_1 \rangle$ na Tabela 3.3 é o valor 6.

3.6 Seleção do Rótulo de um Nodo

Com o objetivo de expressar visualmente agregações temporais, utilizamos o TGP **seleção do rótulo de um nodo**.

O objetivo básico é excluir uma classe n_l do conjunto de classes $n_1, \ldots, n_k$ que estão relacionadas por r, a fim de obter os tempos válidos das classes restantes, desconsiderando a classe eliminada. Fazemos isto selecionando o nodo-papel r e **eliminando** o nodo-classe n_l. Como consequência, a **nova interpretação** do nodo-classe i contém a **união** dos *lifespans* originalmente associados com as tuplas em r, compartilhando os mesmos valores quando se faz a projeção sobre as classes $n_1, \ldots, n_{l-1}, n_{l+1}, \ldots, n_k$.

Por exemplo, podemos derivar o *lifespan* de qualquer projeto através da identificação dos *lifespans* das tarefas associadas, ou seja, o usuário pode selecionar o rótulo do nodo-class *Atividade*, como mostrado na Figura 3.8, correspondente à consulta *"Para cada empregado, qual foi o tempo médio para completar um projeto?"*, de forma a agrupar as instâncias das classes *Empregado* e *Projeto*, desconsiderando a classe *Atividade* (ex. o intervalo de tempo do par $\langle O_2, P_3 \rangle$ na Tabela 3.3 ficará sendo $\langle 3, 7 \rangle$).

Esta abordagem é útil em casos onde o *lifespan* de uma classe corresponde ao agregado de *lifespans* de outra classe (ex. o período de um curso de um estudante pode representar um agregado de períodos semestrais).

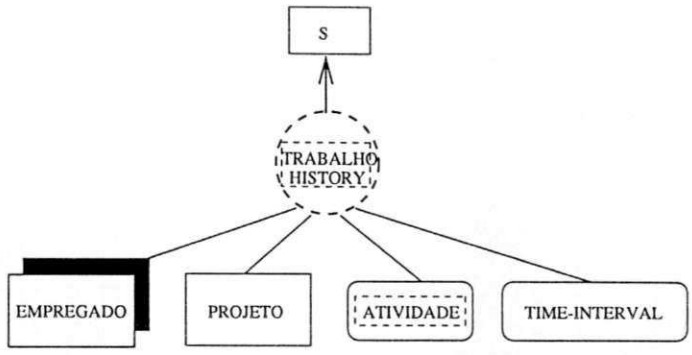

Figura 3.8: TGPs seleção do rótulo de um nodo

Denominamos estas classes como classes de *níveis temporais*. Em TGM, as classes de níveis temporais podem ser modeladas como nodos-classe não temporais $n_1, \ldots, n_k$, os quais são conectadas por um nodo-papel temporal n e um nível l está associado a cada classe n_i, com $1 \leq l \leq k$. Os elementos temporais que compõem o *lifespan* do nodo-papel temporal sempre representam o domínio de tempo associado com a classe relacionada ao mais baixo nível.

Usando o TGP seleção do rótulo de um nodo, podemos automaticamente derivar o *lifespan* da classe a um certo nível l, fazendo a união dos *lifespans* da classe a nível $l-1$.

O operador é definido formalmente como:

Seja r um *nodo-papel resultante da seleção de nodo(s) sombreado(s)*, $r \in D', AD(r) = \{\{n'_i\}, i\}$ ou $AD(r) = \{\{AD(n_1), \ldots, AD(n_k)\}, i\}, AD(n_i) = \{n_{i_1}, \ldots, n_{i_{ci}}\}$, com $ci = |AD(n_i)|$, a **seleção do rótulo do nodo-papel r e do nodo-classe** n_l é uma função $\mathcal{S}_l(D', r, n_l) = D^r$ tal que $D^r = D'$, exceto que:

Se $AD\{r\} = \{\{n'_i\}, i\}$

$m^r(r) = \{x = \langle f_1(s) : o_s, f_1(n'_1) : o_1, \ldots, f_1(n'_{l-1}) : o_{l-1}, f_1(n'_{l+1}) : o_{l+1}, \ldots, f_1(n'_k) : o_k, f_1(i) : I' \rangle \mid \exists \{x_j\} \in m'(r) \mid \prod_{f_1(n'_1),\ldots,f_1(n'_k)}(x) = \prod_{f_1(n'_1),\ldots,f_1(n'_{l-1}),f_1(n'_{l+1}),\ldots,f_1(n'_k)}\{x_j\} \wedge I' = \bigcup_j (I_j) \wedge I_j = \prod_{f_1(i)}\{x_j\}\}$;

Se $AD\{r\} = \{\{AD(n_1), \ldots, AD(n_k)\}, i\}$

$$m^r(r) = \{x = \langle f_1(s) : o_s, f_1(n_{1_1}) : o_{1_1}, \ldots, f_1(n_{1_{c1}}) : o_{1_{c1}}, \ldots, f_1(ni_{l-1}) : o_{i_{l-1}}, f_1(ni_{l+1}) : o_{i_{l+1}}, \ldots, f_1(n_{k_1}) : o_{k_1}, f_1(n_{k_{ck}}) : o_{k_{ck}}, f_1(i) : I' \rangle \mid \exists \{x_j\} \in m'(r) \mid$$

$$\prod_{f_1(n_{1_1}),\ldots,f_1(n_{1_{c1}}),\ldots,f_1(ni_{l-1}),f_1(ni_{l+1}),\ldots,f_1(n_{k_1}),\ldots,f_1(n_{k_{ck}})}(x) =$$

$$\prod_{f_1(n_{1_1}),\ldots,f_1(n_{1_{c1}}),\ldots,f_1(ni_{l-1}),f_1(ni_{l+1}),\ldots,f_1(n_{k_1}),\ldots,f_1(n_{k_{ck}})}\{x_j\} \wedge$$

$$I' = \bigcup_j (I_j) \wedge I_j = \prod_{f_1(i)}\{x_j\}\};$$

Note que expressamos visualmente uma projeção de tempo válido, utilizando o TGP mudança do rótulo de um nodo, de forma a visualizar instantes, intervalos de tempo e durações no resultado da consulta, ou o TGP seleção do rótulo de um nodo, de forma a visulizar agregados temporais no resultado da consulta.

Adicionamos uma operação como extensão temporal do GP desenho de um arco, de forma a expressar visualmente consultas com uma **referência temporal a outro dado**. Neste caso, a extensão temporal é denotada como uma função $\mathcal{E}_t(D', \mathcal{P}, n, q) = D^r$, onde n e q são nodos-papel, cada um resultante da seleção de um nodo sombreado, e $\mathcal{P}$ representa os operadores de *Allen*.

Considerando a consulta *"Quais salários os empregados ganhavam quando eles mudaram de nível pela última vez?"*, Figura 3.9 ilustra a referência temporal ao histórico de níveis.

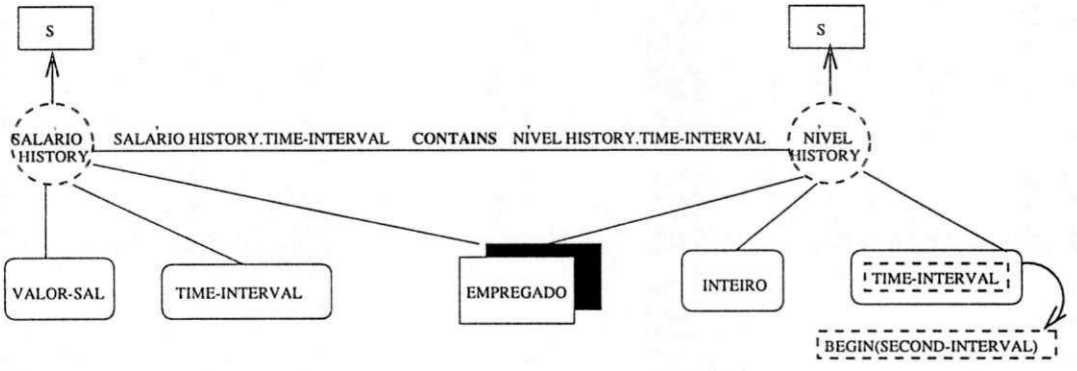

Figura 3.9: Exemplo de uma referência temporal a outro dado

Finalmente, assumimos que muitas visões de um banco de dados podem ser usadas

durante a formulação de uma consulta. De forma a construir tais visões, descrevemos a função $DUPLICATE$, introduzida em [19].

A função $DUPLICATE^k(D)$, onde $D = \langle g, c, m \rangle$ é um TGMDB, resulta em um novo TGMDB $D^k = \langle g^k, c^k, m^k \rangle$ (uma k-cópia de D), o qual é igual a D exceto para os rótulos dos nodos, representando uma concatenação de k com os rótulos originais. Por exemplo, de forma a especificar a consulta *"Recupere o histórico de níveis de todos os empregados cujo nível corrente é maior que 5"*, a função $DUPLICATE$ é aplicada ao *Typed Graph* de forma que um diferente nodo-papel *Nível* é selecionado em cada visão, como mostrado em Figura 3.10.

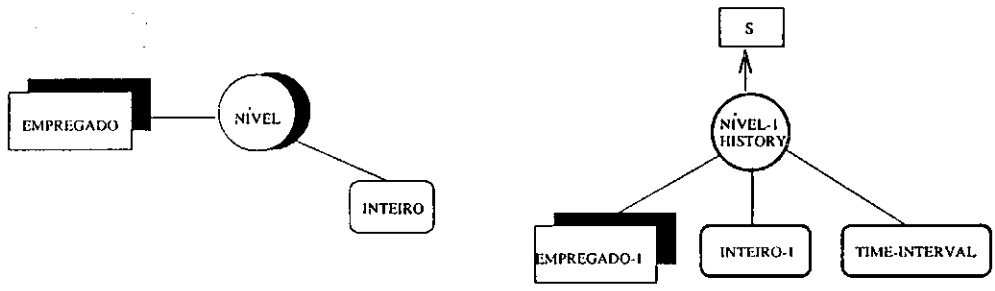

Figura 3.10: Diferentes visões de um TGMDB

3.7 Banco de Dados Resultante

Um novo TGMDB D° representa o **resultado** da execução da consulta expressa através do TGMDB D^r, resultante de aplicações de GPs e TGPs (note que D° pode ser usado para operações posteriores).

D° é um TGMDB composto de um único nodo-classe não imprimível q relacionado, através de nodos-papel binários $r_1, \ldots, r_k$, a um conjunto de nodos imprimíveis que correspondem aos nodos com estado *visualizado* em D^r. Existem condições necessárias e suficientes sobre D^r, de forma que D° possa conter um conjunto não vazio de nodos. Neste caso, D^r é **admissível**, ou seja, ele deve conter ao menos um nodo-papel *visualizado* e cada nodo-papel visualizado deve ser relacionado ao menos a um nodo-classe imprimível *visualizado* e um nodo-classe não imprimível *selecionado* ou *visualizado*.

A interpretação dos nodos-papel binários é computada em dois passos: no primeiro passo, todos os nodos-papel *selecionados* de D são "juntados" (*joined*), dando um significado de um nodo-papel n-ário fictício; no segundo passo, tal significado é projetado sobre os nodos-papel binários de $D°$, levando em consideração as restrições especificadas nos rótulos dos arcos que foram desenhados durante a formulação da consulta.

De forma a especificar $m°$, ou seja, a interpretação do nodo-classe q associado aos nodos-papel binários $r_1, \ldots, r_k$, algumas funções são introduzidas e resultados intermediários são identificados.

Para uma melhor compreensão, definimos as funções através da exploração da seguinte consulta: *"Recupere o nome e emp-id de todos os empregados vivendo em São Paulo (SP) e que tiveram níveis com duração maior que a duração dos seus salários. Recupere também estes níveis, salários e seus correspondentes períodos."* (consideramos o *Typed Graph* e sua Interpretação descritos no capítulo anterior).

Assumindo que GPs e TGPs estão diretamente disponíveis ao usuário, ele/a inicialmente executa a seguinte sequência de operações, como mostra a Figura 3.11: seleção dos nodos *Empregado, Mora, Cidade, Nome-cid*; visualização dos nodos *Nome, String, Id-emp, Inteiro, Nível, Salário* e *Valor-sal* e seleção dos nodos sombreados de *Nível* e *Salário*.

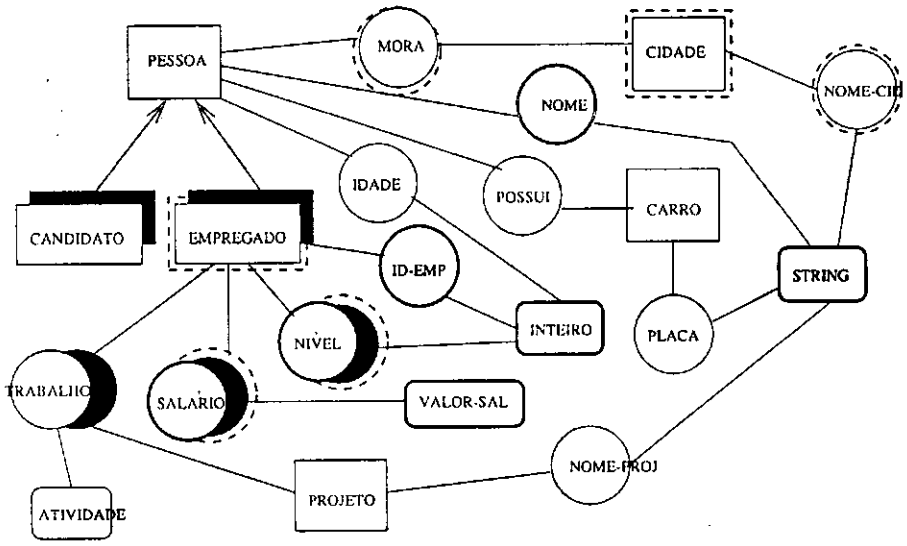

Figura 3.11: Seleção de nodos e nodos sombreados

Seja $HSN(x)$ uma função definida sobre o nodo-classe x e retornando um valor verdadeiro se x é o mais alto nodo selecionado dentro da hierarquia ISA a qual ele pertence (x é o **H***ighest* **S***elected* **N***ode?*): $HSN(x) = True$ se e somente se $f_3(x) \in \{selected, displayed\}$ e não existe um $w \in N_c$ tal que $xISA^*w$ e $w \in \{selecionado, visualizado\}$.

Considerando a hierarquia de generalização existente no *Typed Graph* da Figura 3.11, podemos afirmar que $HSN(Empregado) = True$.

Seja $TAD(y)$ o conjunto de adjacentes verdadeiros (**T***rue* **AD***jacents*) de y uma função definida sobre o nodo-papel y e que retorna o subconjunto de $AD'(y)$ que satisfaz a condição HSN: $TAD(y) = \{x | x \in AD'(y)$ e $HSN(x)\}$.

No exemplo, considerando os nodos-papel do *Typed Graph* da Figura 3.11, podemos afirmar que $TAD(Nome) = \{Empregado, String\}$. O mesmo se aplica ao nodo-papel *Mora*.

Seja $RM(m(y))$ (**R***estrict the* **M***eaning of* y) uma função definida sobre a Interpretação de um nodo-papel y, a qual resulta em uma Interpretação *restrita* de y, de acordo com os nodos-classe selecionados dentro da hierarquia ISA que estão em $TAD(y)$.

Como consequência, esta função altera os rótulos das tuplas dentro da Interpretação de y. Por exemplo, a nova Interpretação do nodo-papel *Nome* será o conjunto de tuplas $\{\langle Employee : o_1, String : Mary\rangle, \langle Employee : o_2, String : Peter\rangle, \langle Employee : o_3, String : John\rangle\}$.

A Figura 3.12 mostra os efeitos da seleção de nodos e nodos sombreados.

Quando y é um nodo-papel temporal, o resultado de $RM(m(y))$ é um conjunto de tuplas, onde o *lifespan* de cada tupla contém ao menos o intervalo de tempo **corrente**. Isto significa que se existe um nodo-papel temporal y em D^r (seu nodo sombreado não foi selecionado), consideramos apenas seu estado *corrente* na Interpretação de D°.

Portanto, a função $RM(m(y))$ é definida como segue:

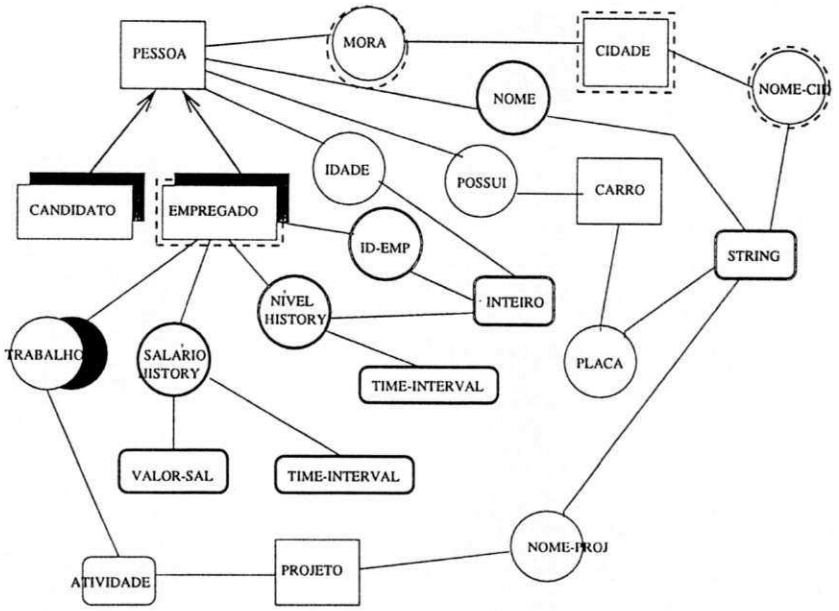

Figura 3.12: *Typed Graph* de interesse da consulta

Se $y \in N_{r_n}$ então $RM(m(y)) = \{\langle l'_1 : o_1, \ldots, l'_k : o_k\rangle | \langle l_1 : o_1, \ldots, l_k : o_k\rangle \in m(y)\}$

Se $y \in N_{r_t}$ então $RM(m(y)) = \{\langle l'_1 : o_1, \ldots, l'_k : o_k\rangle | \langle x, \theta(x)\rangle \in m(y)$, onde x é a tupla da forma $\langle l_1 : o_1, \ldots, l_k : o_k\rangle$, e $\exists I \in \theta(x) | I = \langle begin_I(x), now\rangle\}$,

e $l'_i : o_i$ é tal que $l'_i = f_1(TAD(y) \cap \{z \in N_c | zISA^* f_1^{-1}(l_i)\}$ e $o_i \in m(f_1^{-1}(l'_i)))$.

Em nosso exemplo, não existe nenhum nodo-papel temporal y em D^r, como ilustrado na Figura 3.12.

Seja $RM'(m(y))$ uma função definida sobre a Interpretação do nodo-papel y, a qual, de forma a computar a Interpretação de D°, concatena o rótulo de y aos rótulos das tuplas dentro da interpretação de y:

$RM'(m(y)) = \{\langle l''_1 : o_1, \ldots, l''_k : o_k\rangle | \langle l'_1 : o_1, \ldots, l'_k : o_k\rangle \in RM(m(y))$ e $l''_i = f_1(y) \circ l'_i\}$.

Por exemplo, o $RM'(m(Nome)) = \{\langle Name.Employee : o_1, Name.String : Mary\rangle, \langle Name.Employee : o_2, Name.String : Peter\rangle, \langle Name.Employee : o_3, Name.String : John\rangle\}$.

Baseado nas funções descritas anteriormente, podemos definir a Interpretação de D° como $m^\circ(q) = \{t_1, \ldots, t_s\}$, onde q é o nodo-classe não imprimível que representa o resultado e cada t_i é um novo valor não imprimível, onde s é a cardinalidade de um conjunto RIS (parte extensional da consulta), o qual é definido como segue:

1. Se $N_r^\circ = \emptyset$ então $RIS = \emptyset$;

2. Se $|N_{r_n}^\circ| = 1$ e $|\{k \in N_{r_n} | f_3(k) \in \{selecionado, visualizado\}\}| = 1$ então $RIS = RM'(m(k))$;

3. Caso contrário, $N_r^\circ = \{r_1, \ldots, r_k\}$, com $k \geq 1$ e

   $RIS = inst(eval(n_1, eval(n_2, \ldots, eval(n_{h-1}, n_h))))$, onde $\{n_1, \ldots, n_k\} \equiv \{m \in N_r | f_3(m) \in \{selecionado, visualizado\}\}$ e a função $inst$ extrai o conjunto de instâncias de um nodo fictício computado por $eval$, onde $eval(n_1, n_2)$ retorna um nodo-papel fictício n, cujos adjacentes são a união dos adjacentes de n_1 e n_2, e cuja Interpretação é um conjunto de tuplas:

   $\{\langle l_1 : o_1, \ldots, l_k : o_k, \text{“}nc\text{”} \circ l_{k+1} : o_{k+1}, \ldots, \text{“}nc\text{”} \circ l_h : o_h, l_{h+1} : o_{h+1}, \ldots, l_t : o_t \rangle\}$ tal que para $i = k + 1 \ldots h, f_1^{-1}(l_i) \in N_{c_u}$ e $\langle l_1 : o_1, \ldots, l_k : o_k, f_1(n_1) \circ l_{k+1} : o_{k+1}, \ldots, f_1(n_2) \circ l_h : o_h \rangle \in RM'(m(n_1))$ e $\langle f_1(n_2) \circ l_{k+1} : o_{k+1}, \ldots, f_1(n_2) \circ l_h : o_h, l_{h+1} : o_{h+1}, \ldots, l_t : o_t \rangle \in RM'(m(n_2))$.

   Note que, se os nodos n_1 e n_2 não compartilham os componentes de tupla, a função $eval$ retorna o **produto cartesiano** das interpretações de n_1 e n_2.

O nosso exemplo se aplica ao item 3 desde que existe mais de um nodo-papel selecionado. Desta forma, o conjunto RIS é definido como:

RIS = *inst(eval(Salário-History, eval(Nível-History, eval(Id-emp, eval(Nome, eval(Nome-cid, Mora))))))*.

Mostramos abaixo o resultado da função $eval$ de cada par de nodos.

1. O resultado $e_0 = eval(Nome - cid, Mora)$ é mostrado na Tabela 3.4;

2. O resultado $e_1 = eval(Nome, e_o)$ é mostrado na Tabela 3.5;

3. O resultado $e_2 = eval(Id - emp, e_1)$ é mostrado na Tabela 3.6;

4. O resultado $e_3 =$ *eval(Nível-History,* e_2) é mostrado na Tabela 3.7;

5. O resultado $e_4 =$ *eval(Salário-History,* e_3) é mostrado na Tabela 3.8 (os títulos da coluna desta tabela foram abreviados devido a limitações de espaço).

nc.cidade	nome-cid.string	mora.empregado
O_5	SP	O_1
O_6	RJ	O_2
O_6	RJ	O_3

Tabela 3.4: Resultado de $eval(Nome - cid, Mora)$

nc.cidade	nome-cid.string	nc.empregado	nome.string
O_5	SP	O_1	Maria
O_6	RJ	O_2	Pedro
O_6	RJ	O_3	John

Tabela 3.5: Resultado de $eval(Nome, e_0)$

nc.cidade	nome-cid.string	nc.empregado	nome.string	idemp.inteiro
O_5	SP	O_1	Maria	0001
O_6	RJ	O_2	Pedro	0010
O_6	RJ	O_3	John	0007

Tabela 3.6: Resultado de $eval(Id - emp, e_1)$

Denotaremos com RIS' o conjunto de tuplas da forma $\langle l_1 : o_1, \ldots, l_y : o_y, f_1(q) : o_{y+1} \rangle$ tal que $o_{y+1} \in m^\circ(q), \langle l_1 : o_1, \ldots, l_y : o_y \rangle \in RIS$ e satisfaz todas expressões

nc.cidade	nome-c.string	nc.empregado	nome.string	idemp.inteiro	nível.inteiro	nível.interval
O_5	SP	O_1	Maria	0001	2	$\langle 1, 10\rangle$
O_6	RJ	O_2	Pedro	0010	7	$\langle 3, 7\rangle$
O_6	RJ	O_3	John	0007	3	$\langle 2, 10\rangle$
O_6	RJ	O_3	John	0007	4	$\langle 11, now\rangle$

Tabela 3.7: Resultado de *eval(Nível-History,* e_2)

nc.cid	nome-c.str	nc.emp	nome.str	empid.int	nível-h.intg	nível-h.intv	sal-h.vs	sal-h.intv
O_5	SP	O_1	Maria	0001	2	$\langle 1, 10\rangle$	5000	$\langle 1, 5\rangle$
O_5	SP	O_1	Maria	0001	2	$\langle 1, 10\rangle$	5000	$\langle 8, 10\rangle$
O_5	SP	O_1	Maria	0001	2	$\langle 1, 10\rangle$	6000	$\langle 6, 7\rangle$
O_6	RJ	O_2	Pedro	0010	7	$\langle 3, 7\rangle$	10000	$\langle 3, 7\rangle$
O_6	RJ	O_3	John	0007	3	$\langle 2, 10\rangle$	8000	$\langle 2, now\rangle$
O_6	RJ	O_3	John	0007	4	$\langle 11, now\rangle$	8000	$\langle 2, now\rangle$

Tabela 3.8: Resultado de *eval(Salário-History,* e_3)

booleanas $\mathcal{F}_1, \ldots, \mathcal{F}_n$ e predicados temporais $\mathcal{P}_1, \ldots, \mathcal{P}_n$, tal que diferentes tuplas têm diferentes valores dentro do componente $y + 1 - th$.

Retornando ao exemplo, o usuário realiza as seguintes operações que representam restrições da consulta:

- Mudança do rótulo do arco $\langle Nome - cid, String\rangle$ para *Nome-cid.String="SP"*;
- Um arco com rótulo *Overlaps* é desenhado entre os nodos *Salário-History* e *Nível-History*; mudança do rótulo do nodo *Time-interval* para *duration*, associado aos nodos *Salário e Nível-History*; um arco com o rótulo *Nível-history.duration* > *Salário-history.duration* é desenhado entre os nodos *Salário-history.duration* e *Nível-history.duration.*

O *Typed Graph* resultante é mostrado na Figura 3.13.

Portanto, as únicas tuplas que satisfazem as condições especificadas na consulta estão em destaque na Tabela 3.8.

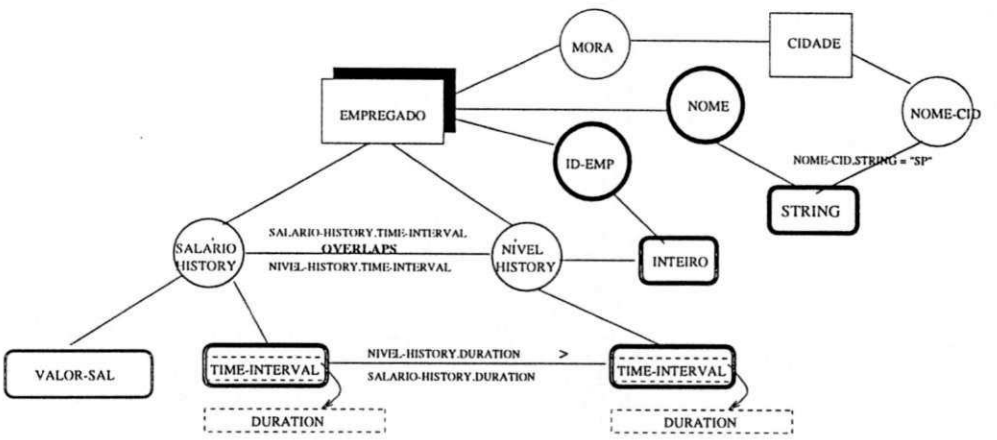

Figura 3.13: *Typed Graph* resultante da aplicação dos operadores

Eventualmente, obtemos de RIS' a Interpretação dos nodos-papel de $D°$:

$m(r_i) = \{\langle l : o, f_1(q) : o_{y+1}\rangle | \langle l_1 : o_1, \ldots, f_1(r_1) \circ l : o, \ldots, l_y : o_y, f_1(q) : o_{y+1}\rangle \in RIS'\}$.

O banco de dados resultante (TGMDB $D°$) é mostrado na Figura 3.14, com a correspondente interpretação na Tabela 3.9.

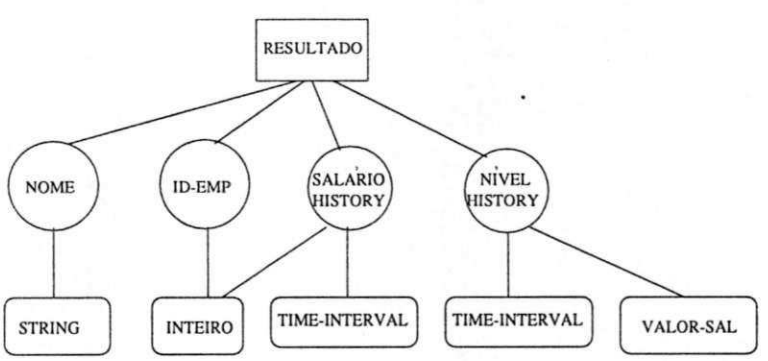

Figura 3.14: Banco de Dados Resultante

3.8 Uma Linguagem Visual de Consulta Temporal

Considerando algumas características relacionadas à consulta temporal, nosso trabalho é relacionado ao trabalho descrito por *Kouramajian e Gertz* [76].

$m(RESULTADO) = \{o_{20}\}$;
$m(Nome) = \{\langle Empregado : o_{20}, String : Maria\rangle\}$;
$m(Id\text{-}emp) = \{\langle Empregado : o_{20}, Inteiro : 0001\rangle\}$;
$m(Salário\text{-}history) = \{\langle Empregado : o_{20}, Valor - sal : 5000, \{\langle 1,5\rangle, \langle 8,10\rangle\}\rangle,$
$\langle Empregado : o_{20}, Valor - sal : 6000, \{\langle 6,7\rangle\}\rangle\}$;
$m(Nível\text{-}history) = \{\langle Empregado : o_{20}, Nvel : 2, \{\langle 1,10\rangle\}$.

Tabela 3.9: A Interpretação resultante

A linguagem visual de consulta de *Kouramajian e Gertz* [76] compreende alguns construtores visuais que permitem a formulação de consultas temporais, baseado em um modelo E-R estendido temporal, o modelo TEER [43]. Este modelo, assim como o modelo TGM, incorpora o conceito de *lifespan* em entidades (classes) e relacionamentos entre entidades.

Os construtores visuais da linguagem são: **expressão booleana temporal**, que é uma expressão condicional sobre atributos e relacionamentos de uma entidade; **tempo verdadeiro**, de uma expressão booleana c, o qual determina o valor de um elemento temporal para cada entidade e; **seleção temporal**, o qual é usado para selecionar entidades particulares baseado em condições temporais; e **projeção temporal**[2], o qual é usado para limitar os dados visualizados das entidades selecionadas a específicos períodos de tempo.

A abordagem de *Kouramajian e Gertz* difere da nossa abordagem em alguns pontos, descritos a seguir.

- Não se visualiza explicitamente as entidades e relacionamentos temporais, como mostra a Figura 3.15;
- Considerando o construtor **projeção temporal**, os predicados de *Allen* [4] são utilizados neste construtor, e não na seleção temporal nos TGPs. Adicionalmente, tal construtor é aplicado sobre **um diagrama E-R completo** (depois da seleção das entidades), como mostra a Figura 3.15, referente à consulta *"Retrieve the*

[2]A denominação destes construtores confunde-se com a terminologia padrão descrita em [68], que denomina seleção/projeção temporal como componentes de uma consulta temporal.

name, salary, and rank of each current computer science instructor during the time period 1985 to 1988" (extraída de [76]).

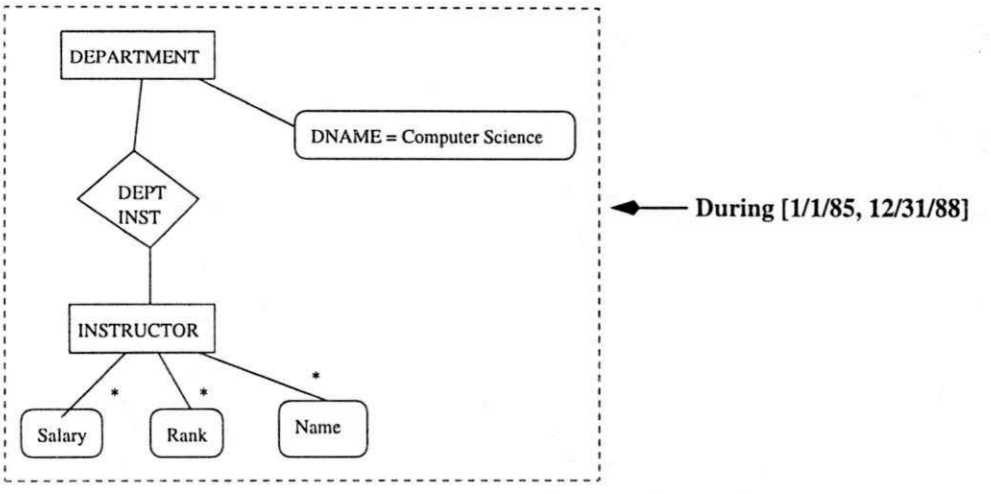

Figura 3.15: Projeção Temporal

O problema desta abordagem é que se pode aplicar somente uma projeção temporal por consulta. Então, não se pode especificar consultas tipo *"Retrieve the name and salary at 1990, and rank during 1985 to 1988, of each current computer science instructor"*. Esta consulta pode ser especificada utilizando os TGPs.

- Considerando o construtor **seleção temporal**, ele é usado na recuperação de entidades baseado nas condições temporais sobre **atributos**. Estas condições envolvem a comparação de dois elementos temporais, utilizando os operadores de comparação $=$, $\neq$, $\subseteq$ e $\supseteq$.

  Portanto, este operador é utilizado em uma parte do diagrama E-R (em contraste à projeção temporal), como mostra a Figura 3.16, com a consulta *"Retrieve the current name and rank of all instructors of the computer science department who were assistant professor during the time period 1987 to 1990"* (extraída de [76]).

  Note que há menos operadores utilizados na seleção temporal do que os utilizados na projeção temporal. Consequentemente, não existem operadores correspondentes na seleção temporal para todos os operadores de *Allen*.

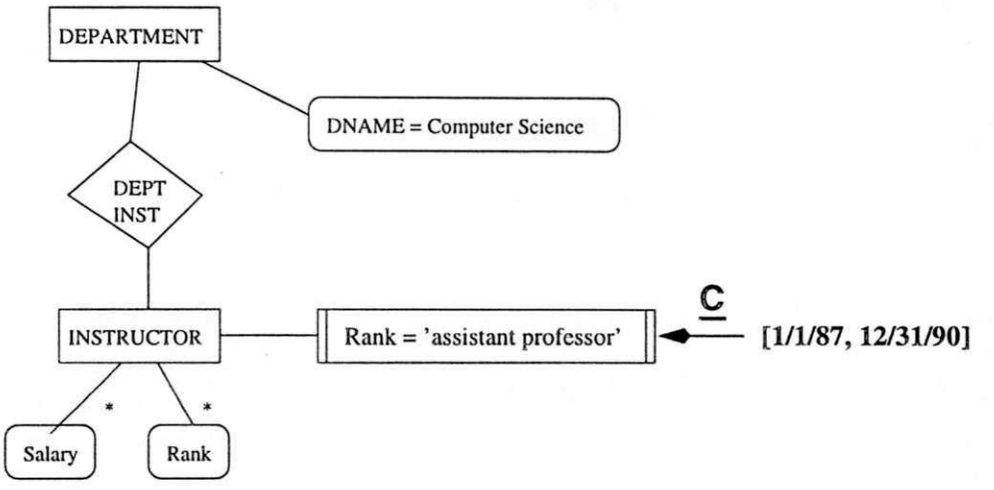

Figura 3.16: Seleção Temporal

- A expressividade de filtrar intervalos temporais em [76] é superior à nossa abordagem. Por exemplo, a consulta *"Retrieve the name, rank and courses of instructors during the time when John Smith taught the compiler course, but except the first time period"*, pode ser especificada utilizando operadores de conjunto, neste caso, o operador de diferença, como mostra a Figura 3.17.

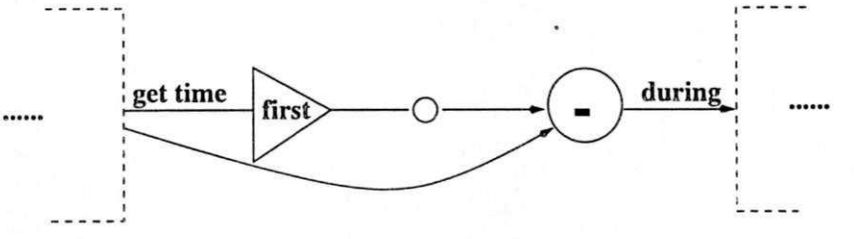

Figura 3.17: Filtragem de intervalos temporais

Por outro lado, consultas que envolvem uma referência temporal a outro dado são especificadas de forma semelhante à nossa abordagem, como mostra a Figura 3.18, com a consulta *"Retrieve the name and rank of all instructors with their courses when John Smith taught a compiler course"* (extraída de [76]).

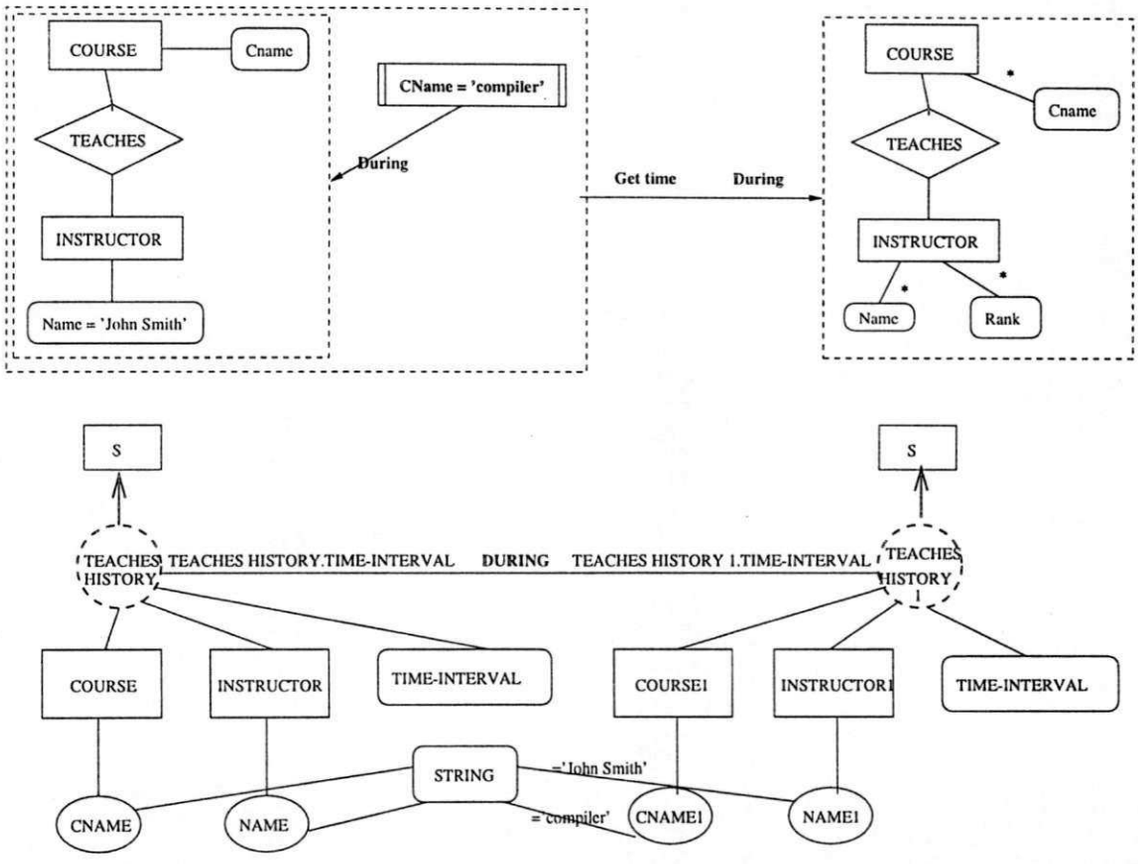

Figura 3.18: Referência temporal a outro dado nas duas abordagens

3.9 Conclusão deste capítulo

Aspectos formais na definição das interfaces para banco de dados, mais especificamente, nas linguagens visuais de consulta, dão uma definição precisa da sua sintaxe, semântica e poder expressivo [23].

Em relação aos sistemas visuais de consulta (SVCs), poucas consideram aspectos formais. O principal desafio das interfaces visuais com formalismo é combinar o poder expressivo desta linguagem com a sua facilidade de uso.

Uma das primeiras linguagens visuais formais surgiu em 88, a linguagem `G+` [33]. Outras linguagens formais surgiram na década de 90, tais como `DOOdle` [32], `Hy+` [29], `VQL` [85], `Gql` [91] e `QBD**` [18]. Contudo, consultas temporais não podem ser especificadas por estas linguagens visuais.

Portanto, a definição formal de operações gráficas elementares para especificar consultas históricas, possibilita que a nossa linguagem visual de consulta (composta de GPs e TGPs) sirva como base para a definição de mecanismos visuais dentro de um SVC, onde tais mecanismos facilitem a interação do usuário não especializado com o sistema. Esta característica foi validada neste trabalho através do mapeamento das primitivas gráficas para mecanismos visuais dentro da nossa proposta de SVC para bancos de dados históricos, o ambiente TVQE.

No próximo capítulo também será feito uma análise da evolução dos SVCs nos últimos anos. Tal análise impulsionou o desenvolvimento do projeto do ambiente TVQE, o qual será descrito com mais detalhes no capítulo 5.

Capítulo 4

Uma Análise da Evolução das Interfaces para Banco de Dados

Em [23] é definida uma taxonomia de SVCs segundo dois critérios: **representação visual** e **estratégia de interação**. Baseado nesta taxonomia e nas novas abordagens utilizadas em interfaces para banco de dados, este capítulo faz uma análise da evolução das interfaces nas décadas de 80 e 90, dentro dos seguintes aspectos:

1. Evolução das principais representações visuais.
2. Evolução e surgimento de novas estratégias de interação.

A evolução dos itens 1 e 2 nas interfaces para banco de dados ilustra o nível de crescimento (ou decrescimento) nas últimas duas décadas, considerando o fator **quantidade** de protótipos desenvolvidos.

O resto do capítulo é organizado da seguinte forma: cada item exposto acima será descrito nas seções 4.1 e 4.2, respectivamente. A análise mais detalhada sobre esta evolução pode ser vista em [46].

4.1 Evolução da Representação Visual

Segundo *Catarci et al.* [23], as representações visuais mais utilizadas nos SVCs são: tabular, diagramática, icônica e híbrida. Descreveremos brevemente a evolução de cada uma nas subseções seguintes.

4.1.1 Interfaces Tabulares

Na representação **tabular**, os dados são organizados e visualmente representados como **tabelas** e os relacionamentos entre os dados como uma justaposição de retângulos [111]. Esta representação é bastante adequada para a estrutura do modelo relacional. A primeira interface tabular conhecida é a linguagem QBE *(Query-By-Example)* [117].

Em QBE as tabelas são mostradas visualmente, sobre as quais o usuário formula sua consulta preenchendo os atributos da tabela com exemplos e restrições necessárias para o resultado da consulta. QBE é melhor que as linguagens textuais pois o usuário não precisa memorizar nomes dos componentes de uma tabela. Devido ao seu sucesso comercial, surgiram novas linguagens *by-example* na década de 80 (um resumo e referências sobre estas linguagens podem ser encontradas em [90]). Por exemplo, TBE [105] é uma extensão do QBE para manipular bancos de dados históricos.

Outras interfaces tabulares surgiram nos anos 80, entre as quais: FormManager [113], FormDoc [75], a interface para o sistema Escher [110], etc.

Na década de 90, mesmo com o paradigma de orientação a objeto integrado ao banco de dados, novas interfaces tabulares surgiram, devido à simplicidade da representação tabular e do modelo relacional. A estas novas interfaces foram incorporadas novas características que possam satisfazer a demanda das novas aplicações, como a interface GRADI [70], que foi desenvolvida para um banco de dados relacional multimídia.

Surgem também outras interfaces tabulares tal como a interface HIBROWSE [41] e Form-Based Visualiser [95]. Outras interfaces foram propostas para acessar bancos de dados remotos, como a descrita em [106], integrando a representação tabular com a página hipertextual. A interface para o sistema Escher foi estendida [111] para servir a ambientes cooperativos, manipulando dados não textuais.

4.1.2 Interfaces Diagramáticas

Diagramas em um SVC representam visualmente entidades e relacionamentos entre entidades, de uma forma similar à estrutura de um grafo. Sua popularidade surgiu a partir da representação visual dos modelos semânticos, em particular, do modelo Entidade-Relacionamento (E-R) [26].

A primeira interface diagramática, surgida em 75, foi `Cupid` [82]. Como o `QBE`, `Cupid` foi projetada como uma interface de alto nível para um banco de dados relacional e utiliza símbolos para representar componentes de uma consulta. O usuário seleciona componentes relevantes e constrói sua consulta a partir destes componentes.

Na década de 80, diversas interfaces surgiram para o modelo E-R, entre as quais, as descritas em [52], [42], [35], além de outras interfaces para outros modelos semânticos tais como a interface `Isis` [56], `Snap` [12] e `Picasso` [72].

Na década de 90, os diagramas emergem como a representação visual mais popular. Surgem as interfaces diagramáticas para os modelos orientados a objeto tais como `G-OQL` [109], `DOOdle` [32], `Super` [36] e continuam a surgir interfaces para o modelo E-R, tais como a interface `QBD*` [5], `AERIAL` [13], [34], e as interfaces para bancos de dados temporais `ERT/vql` [107] e [76] (brevemente descrita no capítulo anterior).

Novas abordagens foram incorporadas às diagramáticas de forma a fortalecer o seu poder de visualisação. Metáforas visuais foram integradas aos diagramas, como a visualisação tridimensional (3D). Entre as interfaces 3D, destacamos `AMAZE` [11] e `WINONA` [94]. Os diagramas são também utilizados para facilitar a visualisação da estrutura dos documentos na *Web* [84] e uma linguagem visual de consulta [14] surgiu para bancos de dados multidimensionais OLAP (*On-line Analytical Processing*).

Em suma, os diagramas conseguem captar a visualisação global dos dados, mostrando o inter-relacionamento entre os mesmos. Adicionalmente, a maioria das interfaces diagramáticas exige que o usuário manipule a estrutura do esquema conceitual. Contudo, o processo de criação da estrutura do esquema conceitual é feito pelo projetista do banco de dados, cuja visão sobre os dados é diversa da visão do usuário final (ver seção seguinte).

4.1.3 Interfaces Icônicas

As interfaces **icônicas** representam um aperfeiçoamento na interação homem-máquina. Isto se deve ao poder de **metáfora** que os ícones proporcionam na representação visual de um conceito ou de uma função. Foi a partir da representação icônica que surgiu a metáfora mais popular dentro das plataformas `Macintosh`, `Ms-Windows` e `X-Windows`, a metáfora ***desktop***. Em uma metáfora *desktop*, arquivos, programas e dispositivos são representados como entidades gráficas que simulam objetos que podem ser encontrados em um ambiente de escritório: documentos, diretórios, caixas de correio, etc.

Os ícones representam tanto as entidades do mundo real como as funcionalidades disponíveis de um SVC. Contudo, a representação icônica não se tornou popular em banco de dados, visto que não consegue representar explicitamente o relacionamento entre as entidades em um esquema conceitual. Realmente este fato se refletiu na pequena quantidade de interfaces icônicas, como ilustra a Figura 4.5, todas surgidas somente na década de 90.

A primeira interface icônica para banco de dados que se tem notícia é a `IconicBrowser` [108]. Depois vieram as interfaces `Iconographer` [39], `QBI` [81] e `Oggetto Desktop` [69].

`Oggetto Desktop` [69] integra a metáfora *desktop* com o paradigma de orientação a objeto. Em `Oggetto desktop`, ícones simples representam itens (arquivos ou documentos) e ícones de grupo representam um grupo de itens (diretórios ou *folders*). Adicionalmente, a interface permite a navegação sobre classes relacionadas pela hierarquia de generalização, além de permitir que os usuários possam utilizar a mesma metáfora na navegação sobre as instâncias do banco de dados (apesar de que ele considera um número irrisório de instâncias, como mostrado na Figura 4.1, extraída de [69]).

`Oggetto desktop` também dá suporte a algumas operações utilizadas na evolução de esquemas, tais como, introdução de novos tipos (classes) e subtipos (subtipos), mudança de tipos em instâncias, alteração dos tipos de atributos que referenciam estas novas classes, entre outras operações.

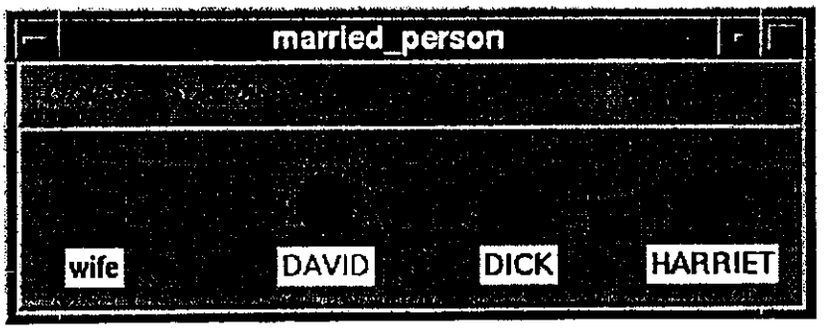

Figura 4.1: Exemplo de uma navegação no Oggetto desktop

Como foi projetado um protótipo inicial de Oggetto desktop, conseguiu-se identificar algumas limitações. Por exemplo, desde que a navegação sobre a estrutura do esquema conceitual é feita através da interação sucessiva de abertura e fechamento de janelas, perde-se a visualização global do esquema. O sistema parece de difícil aplicabilidade em bancos de dados com estruturas complexas e grande quantidade de dados.

Outras interfaces icônicas surgiram, entre as quais, a interface Marmota [15], que realiza consultas na Web, a interface descrita em [31], e a interface 3D Kaleidoscape [87].

4.1.4 Interfaces Híbridas

Devido ao crescimento da complexidade das aplicações e dos diversos tipos de usuários que as utilizam, novas interfaces surgiram permitindo o acesso aos dados através da combinação de diferentes representações visuais. A uma interface interativa que suporta estas características denominamos interface **híbrida**. Ela permite visualizações alternativas da interface em uma das abordagens descritas anteriormente (tabular, diagramática e icônica), ou combinando diferentes estruturas visuais em uma simples representação (por exemplo, ícones relacionados através de arcos, como os diagramas).

Os SVCs híbridos surgiram na década de 80, tais como SKI [74], Sicon [58] e Pasta-3 [77]. Na década de 90 destacamos as interfaces MIGI [80], Medusa [63], IDDS [88], as interfaces descritas em [1], [38] e VisTool [10].

O protótipo `VisTool` é baseado na linguagem visual de consulta `Visionary` [9]. `Visionary` possui uma representação visual híbrida (icônica e diagramática), como mostra a Figura 4.2, extraída de [9], e seu correspondente grafo, ilustrado na Figura 4.3. Conceitos primários são representados visualmente como ícones, enquanto que associações entre conceitos são representados como arcos direcionados. Os círculos e triângulos representam visualmente os relacionamentos *is-a* e *part-of*, respectivamente.

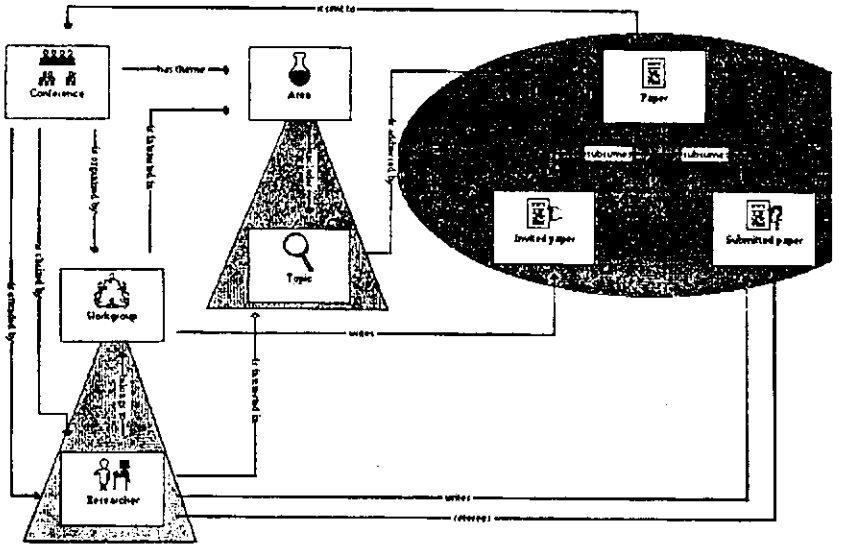

Figura 4.2: Representação visual híbrida de `Visionary`

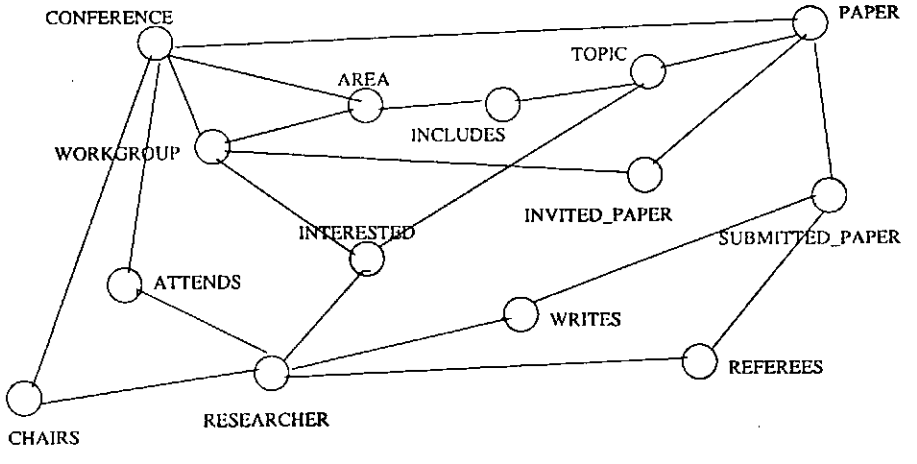

Figura 4.3: Grafo correspondente à representação visual da Figura 4.2

O conceito primário é conectado a qualquer outro conceito através de um caminho de associações. Desta forma, em uma consulta interpretada sob um ponto de vista, é possível referenciar atributos pertencendo a qualquer conceito, sem formular explicitamente as junções necessárias, como mostra a Figura 4.4.

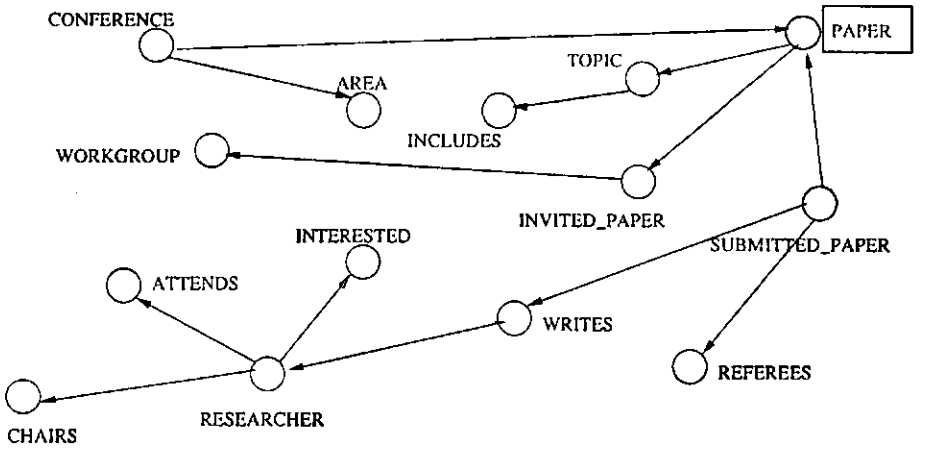

Figura 4.4: Exemplo de um ponto de vista com o conceito primário *paper*

Para finalizar esta seção, ilustramos na Figura 4.5 a evolução de todas as interfaces visuais nestas duas últimas décadas. Note que a única representação visual com decrescimento nesta década foi a representação tabular, até pouco tempo atrás a mais popular representação visual. Nesta análise conclui-se que a interface diagramática atualmente é a mais popular.

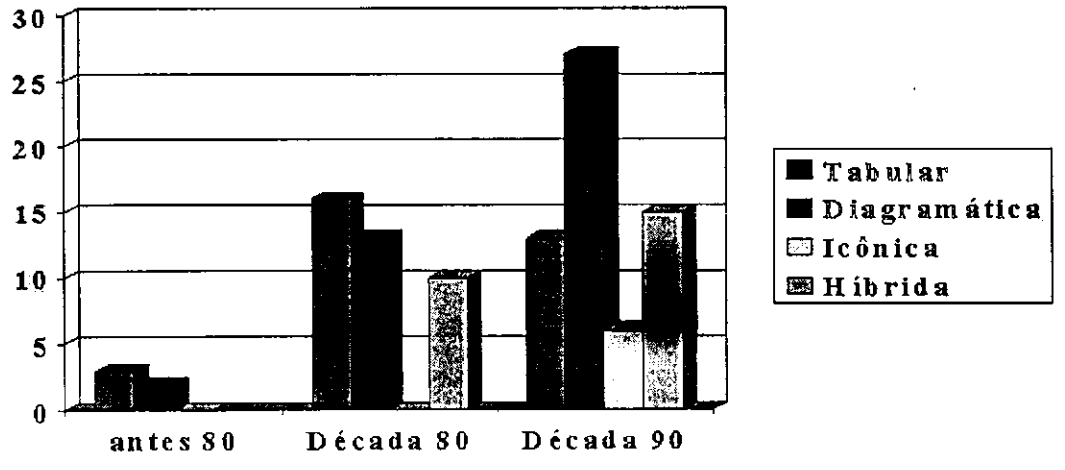

Figura 4.5: Evolução quantitativa das representações visuais

4.2 Evolução das Estratégias de Interação

O trabalho de *Catarci et.al.* [23] define uma taxonomia de estratégias de interação segundo duas atividades que envolvem o acesso aos dados em um SVC: **entendendo a realidade de interesse**, cujo objetivo é a definição do fragmento do esquema envolvido na consulta, denominado esquema de consulta e **formulando a consulta**, onde o esquema de consulta pode ser manipulado de diversas formas, de acordo com os operadores de consulta disponíveis.

Uma classificação de SVCs baseado na estratégia de *entendendo a realidade de interesse* resulta nas seguintes classes: *top-down*, onde aspectos gerais da realidade são percebidos, e então detalhes mais específicos podem ser vistos; e *browsing*, onde o usuário examina um conceito e, através dos seus conceitos associados, um novo conceito pode ser selecionado para ser o corrente, desta forma, os seus conceitos relacionados são mostrados, e assim por diante.

As estratégias para formular uma consulta são classificadas em:

- **Navegação de esquemas**, o qual é similar a estratégia de *browsing* e tem a característica de se concentrar sobre um conceito (ou um grupo de conceitos) de forma a se alcançar outros conceitos de interesse, sobre os quais condições de consulta podem ser especificados;
- **Composição de conceitos**, onde a consulta é formulada através da composição de resultados parciais;
- ***By example***, onde o usuário fornece um exemplo da resposta e o sistema identifica o objetivo generalizando tal exemplo; e
- **Seleção de domínio**, o qual permite uma pesquisa condicionada a um dado domínio sobre um conjunto de dados com múltiplas chaves, onde a abordagem de *consulta dinâmica* [1], [100] é uma implementação desta técnica.

Esta seção apresenta a taxonomia de *Catarci et al.* de forma simplificada, integrando as duas atividades descritas anteriormente (entendendo a realidade de interesse e formulando a consulta).

Desta forma, fazemos uma análise da evolução das interfaces para banco de dados segundo duas estratégias de interação: manipulação da **estrutura** e manipulação do **conteúdo**. Na primeira estratégia, o usuário manipula a estrutura do esquema conceitual de forma a acessar os dados, enquanto que na segunda o usuário percorre sobre o conteúdo do banco de dados, sem precisar ter conhecimento da estrutura dos dados. Descreveremos a evolução das duas estratégias nas subseções seguintes.

4.2.1 Manipulação da Estrutura

Considerando a representação tabular, uma das estratégias de navegação mais popular foi adotada pela interface tabular QBE: a abordagem *by-example*. Todas as linguagens *by-example* adotaram esta estratégia além de outras interfaces tabulares tais como NFQL [44], [115] e [111].

Considerando a representação icônica, a maioria das interfaces utilizou a abordagem de composição de conceitos, através da seleção e sobreposição de ícones. Algumas interfaces diagramáticas ou híbridas também adotaram esta abordagem tais como Cupid [82], Picasso [72], Pasta-3 [77] e VQL [85].

Considerando a representação diagramática, a estratégia de interação mais popular foi a navegação de esquemas. O usuário "navega" sobre o esquema de interesse formando um "caminho" (*path*) correspondente a sua consulta. Um *path* é constituído de uma sequência ordenada de junção entre pares $\langle entidade, relacionamento \rangle$ dentro do esquema conceitual, seguido por uma seleção final e projeção [95].

Contudo, apresentar visualmente a estrutura conceitual como base para formulação de uma consulta ainda está longe do conhecimento conceitual que o usuário tem sobre dados, ou seja, de como ele realmente os "vê". A estrutura lógica dos dados em modelos semânticos está mais direcionada à visão que o projetista tem de uma aplicação específica. Adicionalmente, a criação dos esquemas conceituais pode variar de projetista para projetista, dando margem a diversas interpretações da semântica dos mesmos, como mostra a Figura 4.6, com três abordagens de modelagem de dados do conceito de família.

A Figura 4.7 ilustra um exemplo de consulta dinâmica, extraído de [66]. O lado esquerdo da figura representa uma visualizacao de símbolos geométricos dentro do espaço *salário-idade*, onde cada símbolo representa um empregado com seu correspondente salário e idade. O formato de cada símbolo representa o departamento do empregado.

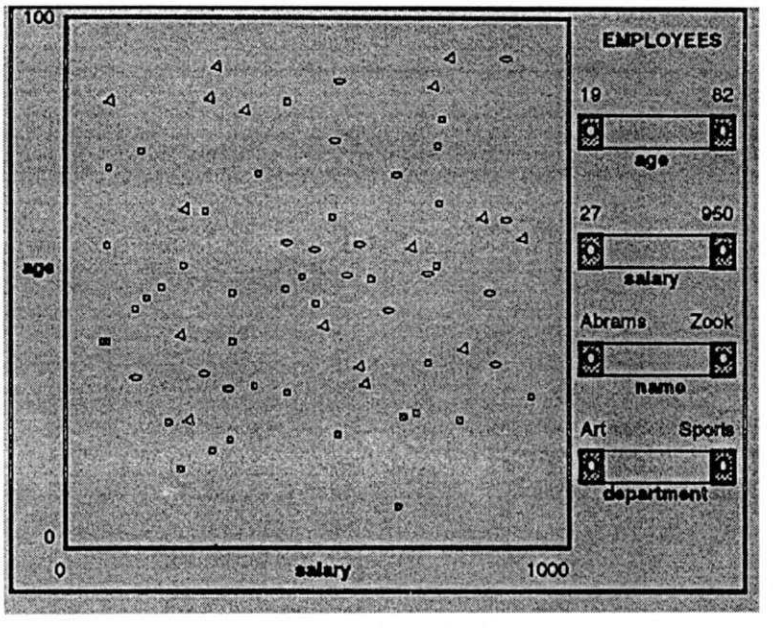

Figura 4.7: Exemplo de uma Consulta Dinâmica

O lado direito da figura é ocupado por *sliders*, um para cada atributo (os atributos são *idade, salário, nome* e *departamento*). Os extremos dos *sliders* representam o valor mínimo e o valor máximo do atributo. A cada manipulação do *slider*, o resultado da consulta se modifica dinamicamente e rapidamente, ou seja, os símbolos desaparecem ou reaparecem, dependendo da mudança de valor dos atributos.

Aplicativos que utilizam consulta dinâmica começaram a surgir somente na década de 90. Aplicações geográficas são perfeitamente adequadas para a consulta dinâmica (referências podem ser encontradas em [100]). Interfaces para bancos de dados foram surgindo como `HIBROWSE` [41], `Guidance` [60], [79], e as interfaces temporais `TVQL` [62] e `Lifelines` [93].

A linguagem visual de consulta `TVQL` (*Temporal Visual Query language*) [62] é principalmente utilizada em dados para vídeo, mais especificamente na identificação de

tendências temporais em dados para vídeo, através das consultas por relacionamentos entre anotações de vídeo, onde os usuários analisam o vídeo em termos de relacionamentos temporais **contínuos** entre eventos.

`Lifelines` [93] é um ambiente de visualização generalizado, utilizado em aplicações que envolvem especificamente resumos de histórias pessoais (dados biográficos). Cada objeto temporal que é instância de um banco de dados é visualizado **individualmente**, como mostra a Figura 4.8, extraída de [93].

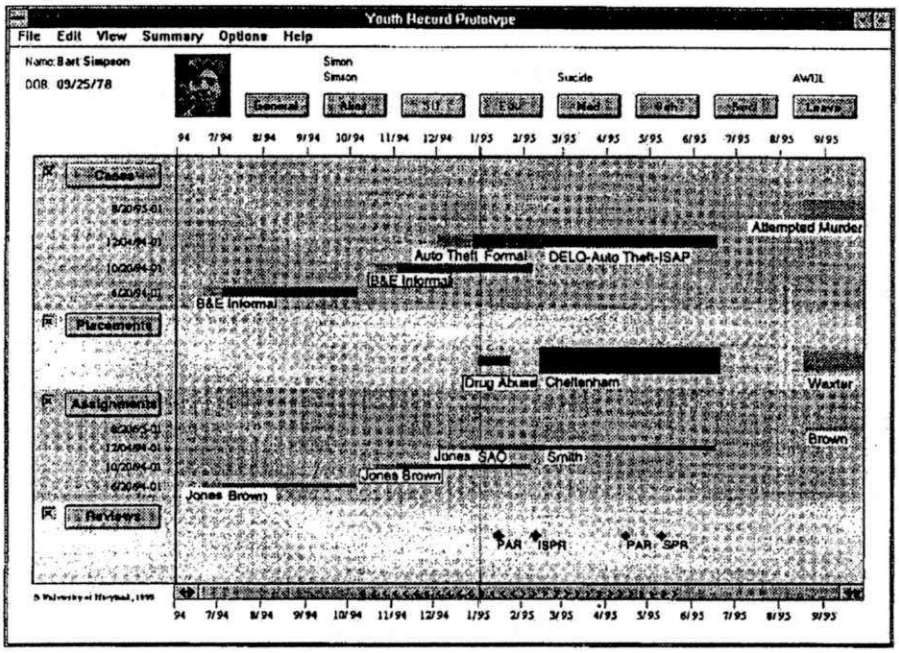

Figura 4.8: Visualização da história de um indivíduo em `Lifelines`

Note que a figura apresenta toda a evolução temporal deste objeto específico, incluindo fatos periódicos (representados visualmente como linhas de tempo) e fatos instantâneos (representados visualmente como ícones). Propriedades visuais (cor, formato, tamanho) relacionados aos fatos possuem um significado semântico (por exemplo, no período de 03/95 a 07/95, o indivíduo *Bart Simpson* esteve em um centro residencial de tratamento por abuso de drogas, em *Cheltenham*, como mostra a Figura 4.8).

Para finalizar esta seção, ilustramos na Figura 4.9 a evolução das duas estratégias de interação nestas duas últimas décadas. Note que na década de 80, havia pouca manipulação do conteúdo, porém esta situação se reverteu na década de 90.

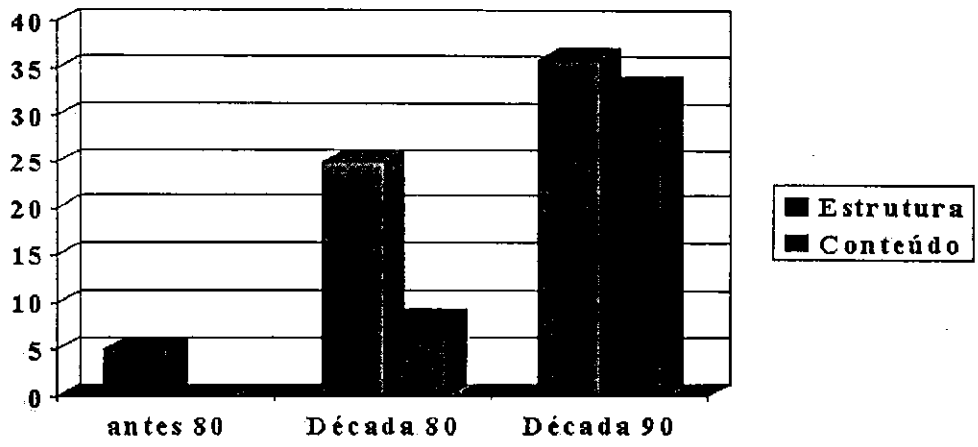

Figura 4.9: Evolução quantitativa das duas estratégias de interação

4.3 Conclusão deste capítulo

Este capítulo fez uma análise evolutiva das interfaces visuais para banco de dados, segundo a representação visual e estratégia de interação adotada.

Baseado nos critérios descritos anteriormente, o ambiente TVQE adota duas representações visuais: uma representação diagramática e uma icônica, a qual explora uma metáfora de uma "agenda gráfica", a qual será descrita no próximo capítulo.

Adicionalmente, o ambiente adota a estratégia *top-down* na localização do esquema de interesse em uma árvore de contextos (descrita no próximo capítulo), a estratégia de *browsing* para determinar o esquema de consulta, e a estratégia de *navegação de esquema* para formular a consulta. Na visualização do resultado da consulta, TVQE adotará a estratégia de *seleção de domínio.*

Detalhes sobre a interface TVQE serão vistos no próximo capítulo.

Capítulo 5

Expressando Visualmente Consultas Temporais

Foi especificado um SVC para bancos de dados históricos, denominado **TVQE** (*Temporal Visual Query Environment*) [48], o qual inclui um conjunto de primitivas gráficas temporais, que foram introduzidas formalmente no capítulo 3.

Neste capítulo, descrevemos o ambiente TVQE dentro dos seguintes aspectos:

1. Breve descrição do projeto conceitual inicial;

2. Descrição do projeto conceitual atual:

   - *Layout* do cenário atual: apresentação, entrada e saída das informações;
   - Descrição das janelas gráficas que compõem o ambiente;
   - Orientação em como utilizar a interface através de um exemplo que se refere a um banco de dados histórico sobre uma agência de emprego (usaremos este exemplo por todo o capítulo).

3. Breve descrição de alguns trabalhos relacionados ao ambiente TVQE.

5.1 Projeto conceitual inicial

A interface TVQE [51] foi projetada inicialmente como uma janela principal contendo um menu *pull-down* para acessar e editar um esquema conceitual e uma área compartilhada por dois painéis, denominados **janela de esquemas** (*schema window*) e **janela de consultas** (*query window*), sobre os quais a estrutura de um esquema conceitual é visualizada. O usuário interage com a janela de esquemas de forma a selecionar a classe **alvo** da consulta [73]. Em seguida, o usuário interage com a janela de consultas.

Antes de ilustrar o *layout* inicial da interface TVQE, definimos o conceito de **contexto** que é utilizado neste projeto inicial e no projeto atual.

Desde que visualizar todas as classes de um esquema complexo em uma simples estrutura visual pode ser inviável, o esquema de banco de dados pode ser visualizado como uma árvore *top-down*, denominada de **árvore de contextos**. A criação da árvore de contextos surgiu de um estudo sobre refinamentos *top-down* de esquema conceituais do modelo E-R durante a fase de projeto, através da utilização das **primitivas *top-down*** de *Batini et. al.* [7].

Mais especificamente, uma primitiva *top-down* é aplicada a um esquema inicial, o qual representa um simples conceito, transformando-o em um novo esquema, o qual representa uma descrição mais detalhada daquele conceito. De fato, a visualização de um esquema como uma árvore de contextos representa seu refinamento como resultado da aplicação de uma primitiva, a qual particiona uma entidade E em um conjunto de novas entidades não relacionadas $E_1, E_2, \ldots, E_n$.

Em nossa abordagem, consideramos tal entidade E como um **contexto** C. Um contexto representa um conceito em uma forma abstrata, mas não contém objetos do mundo real como suas instâncias. Ele pode ser visto como uma abstração de um conjunto de classes (ver exemplo a seguir). Através de uma transformação *top-down*, C é refinado em um conjunto $C_1, C_2, \ldots, C_n$, onde cada C_i pode representar ou um contexto mais específico ou uma classe. Se C_i é um contexto, ele pode ser refinado em um outro conjunto.

Portanto, o usuário acessa o esquema conceitual através do menu *File* e o sistema visualiza o esquema conceitual como uma árvore de dois níveis, tendo um contexto como raiz e seus nodos-folhas representando outros contextos ou classes, como mostrado na Figura 5.1[1]. Note que contextos, classes e classes temporais são visualmente representadas como quadrados escuros, claros e sombreados, respectivamente.

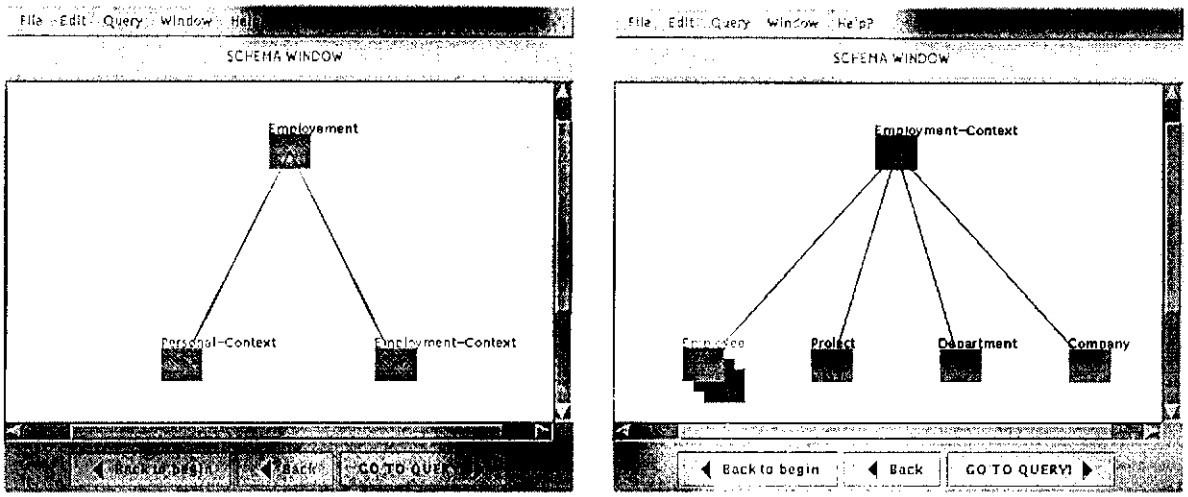

Figura 5.1: Visualização inicial de um esquema conceitual

Desde que o usuário não visualiza a classe desejada no primeiro refinamento do esquema da Figura 5.1 (lado esquerdo), ele/a pode expandir um contexto, apenas selecionando sobre ele, gerando o refinamento do contexto selecionado, como mostra a Figura 5.1 (lado direito). A partir da seleção da classe alvo e o botão *go to query*, o usuário pode especificar a sua consulta. Em seguida, o usuário interage com a janela de consultas, onde a classe alvo é visualizada como nodo-raiz e suas propriedades representam os nodos-folha da árvore, como mostra as Figuras 5.2 e 5.3 (lado esquerdo).

As propriedades representam outras classes, classes temporais, atributos e atributos temporais, os quais são visualmente diferenciados. Note que os relacionamentos binários entre a classe alvo e suas propriedades são visualizados como arcos, enquanto que relacionamentos n-arios são visualizados como classes e podem ser refinados em suas classes associadas. Portanto, o acesso a classes e suas propriedades é feito através da seleção/navegação sobre os nodos de uma árvore.

[1]Por questões de generalidade, todos os exemplos da aplicação estão escritos em inglês.

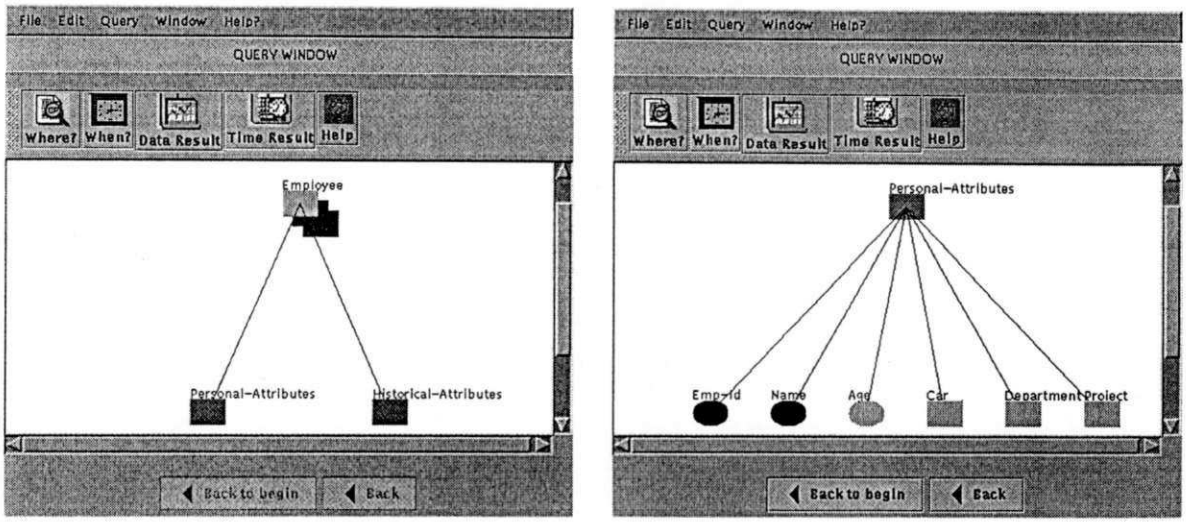

Figura 5.2: Visualização do esquema na janela de consultas

Adicionalmente, os nodos podem assumir três estados: **não selecionado** para a consulta (o qual representa o estado inicial de todos os nodos), **selecionado** (nodo em cor vermelha) para a consulta e **visualizado** (nodo em cor preta) para o resultado da consulta. O usuário pode continuamente mudar o estado de um nodo, como consequência o nodo selecionado assumirá a cor correspondente.

Considerando a filtragem (histórica ou não) de valores das propriedades da classe alvo, a Figura 5.3 (lado direito), por exemplo, ilustra a especificação de uma consulta temporal sobre um período de tempo em um painel de diálogo.

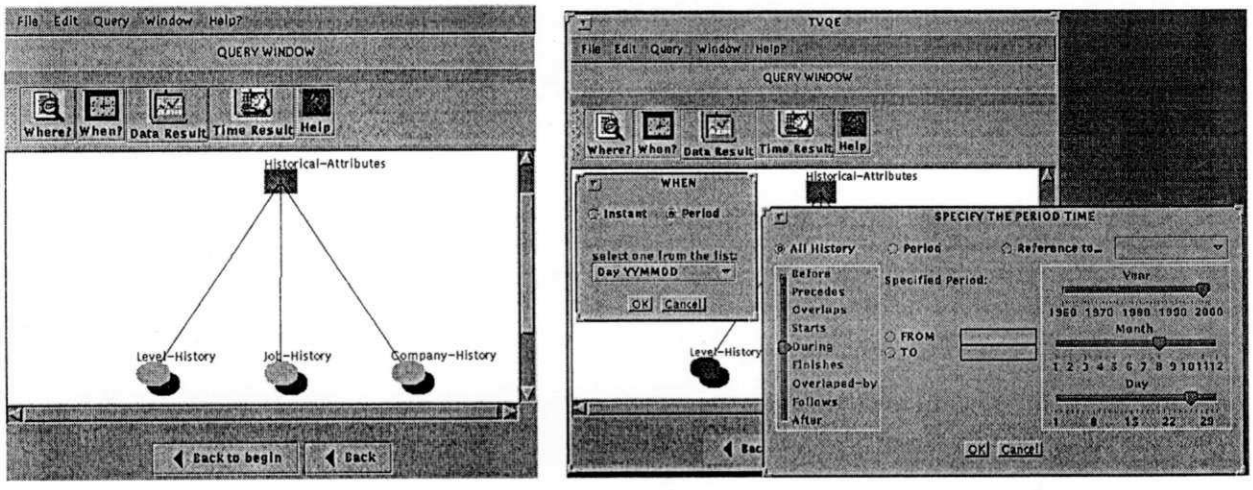

Figura 5.3: Condição temporal sobre um período de tempo

Não entraremos em mais detalhes sobre o projeto inicial visto que a interface sofreu alguma modificações, com veremos na próxima seção.

5.2 Projeto Conceitual Atual

Após o desenvolvimento do projeto inicial, foi necessário avaliar a interface através de sugestões de usuários que tiveram acesso ao protótipo. A partir destas sugestões, alguns problemas foram detectados no projeto inicial, entre os quais:

- Usuários não foram familiarizados em interpretar e manipular nodos do esquema conceitual;
- Como o refinamento do esquema é sempre visualizado em uma árvore de dois níveis por vez, não existe uma visualização global do esquema conceitual.

Desta forma, desenvolvemos um projeto melhorado, incluindo a representação visual icônica, o qual explora uma metáfora de uma **"agenda gráfica"** [50], de forma a facilitar a especificação da consulta.

A especificação de algumas consultas através da seleção de índices em uma agenda gráfica forneceu aos usuários uma maior facilidade de uso do que a manipulação direta sobre a representação diagramática. Adicionalmente, ícones têm um significante poder de metáfora e SVCs icônicos são principalmente direcionados a usuários que não são familiares com conceitos de modelos de dados e podem achar difícil interpretar um diagrama E-R, por exemplo.

Contudo, a representação visual de um esquema de banco de dados como uma agenda gráfica é, em algum sentido, menos rica que a digramática, uma vez que um diagrama favorece a visualização de relacionamentos entre conceitos. Portanto, decidimos integrar as duas representações, destacando o uso da representação icônica como modo de interação.

De forma a definir o novo *layout* da interface TVQE, o projeto conceitual da interface TVQE envolveu os seguintes aspectos:

1. **Definição dos objetos de interação e suas propriedades** - existem dois objetos básicos de interação: as **classes** (entidades) que fazem parte do esquema conceitual e os **dados** que representam as instâncias que fazem parte do resultado da consulta. A classe tem como principais propriedades, nome, descrição, tipo, atributos e relacionamentos, os quais representam outras classes.

2. **Visão conceitual dos objetos de interação** - considerando a visão da estrutura do esquema conceitual, as classes e suas propriedades são visualizadas como **nodos** de uma árvore de contextos, de um grafo, de um subgrafo e como **índices** de uma agenda "gráfica".

3. **Acesso aos objetos de interação** - o acesso a classes e suas propriedades é feito através da **seleção dos índices de uma agenda.**

4. **Principais operações a serem realizadas sobre os objetos** - seleção da classe alvo da consulta e suas propriedades, mudança de estados das propriedades selecionadas, filtragem (histórica ou não) de valores das propriedades da classe alvo, manipulação interativa de dados sobre o tempo. Estas operações são visualmente representadas como botões e ícones.

5. **Estilo de interação** - é a manipulação direta [99]. No ambiente, esta interação é feita sobre todas as funções fornecidas pela interface, tais como: acionamento de botões e ícones, manipulação de painéis e objetos gráficos, sempre apresentando o estado corrente da formulação da consulta através da visibilidade dos objetos e ações sobre eles.

Detalhes dos aspectos citados acima serão descritos nas próximas sub-seções.

5.2.1 Apresentação, entrada e saída das informações

As informações são apresentadas através de uma janela gráfica que compreende menus, ícones, botões (*buttons*), painéis, diagramas e a representação visual de uma agenda "gráfica". O *layout* da janela principal é ilustrado na Figura 5.4, a qual contém as seguintes informações:

1. Área de menu e ícones: contêm botões que representam as operações sobre um esquema conceitual e saída da aplicação, e um conjunto de ícones que representam as operações de uma consulta;

2. Área de Título: contém o título dos painéis da janela principal;

3. Área de Visualização, denominada Janela de Esquemas (*schema window*), a qual contém três sub-painéis: árvore de contextos (*context tree*), esquema gráfico (*graph schema*), e esquema de consultas (*query schema*);

4. Área de interação, denominada Janela de Interação (*interaction window*).

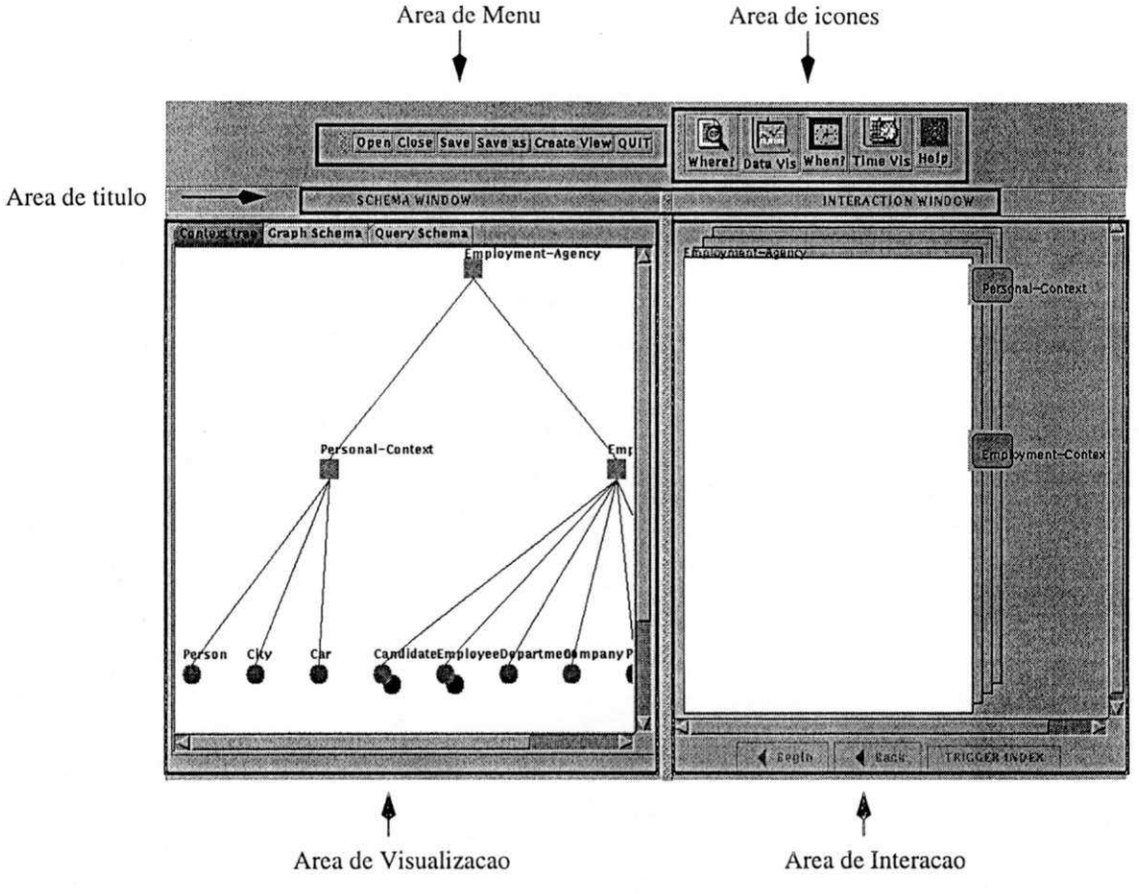

Figura 5.4: *Layout* da Janela Principal

A **entrada** das informações é feita através de painéis de diálogo, que aparecem em decorrência do acionamento dos ícones *Where?* e *When?*.

Estes painéis possuem indagações que esperam alguma resposta do usuário ou coleta de informações, além de possuir elementos gráficos de controle (*widgets* que interagem com o usuário. Por exemplo, a Figura 5.5 corresponde ao *layout* do painel de diálogo utilizado para especificar uma condição temporal. Este *layout* contém as seguintes informações:

1. Painel de opções da condição temporal, incluindo um menu de referências temporais;
2. Painel que contém um menu de granularidades temporais (ano, mês, dia, etc);
3. Painel para especificar uma constante temporal;
4. Painel para especificar um predicado sobre um instante de tempo;
5. Painel para especificar um predicado sobre um período de tempo;
6. Painel de filtragem dos intervalos temporais selecionados;
7. Painel utilizado para especificar agregações temporais;
8. Painel que confirma ou cancela a condição temporal especificada pelo usuário.

A **saída** de uma informação em uma especificação de consultas corresponde ao resultado da mesma. O painel *esquema de consultas* mostra o resultado intensional da consulta, onde o ambiente fornece a representação gráfica do esquema de consultas gerado após a formulação da consulta.

As seções seguintes são ilustradas através de um exemplo de interação do usuário com o sistema, o qual se refere a um banco de dados histórico sobre uma agência de emprego, cuja funcionalidade foi brevemente descrita no capítulo anterior.

5.2.2 Janelas Gráficas

A interface TVQE apresenta ao usuário uma janela contendo items de menu para acessar um esquema de banco de dados, acrescido dos seguintes ícones: ícone *Where*, o

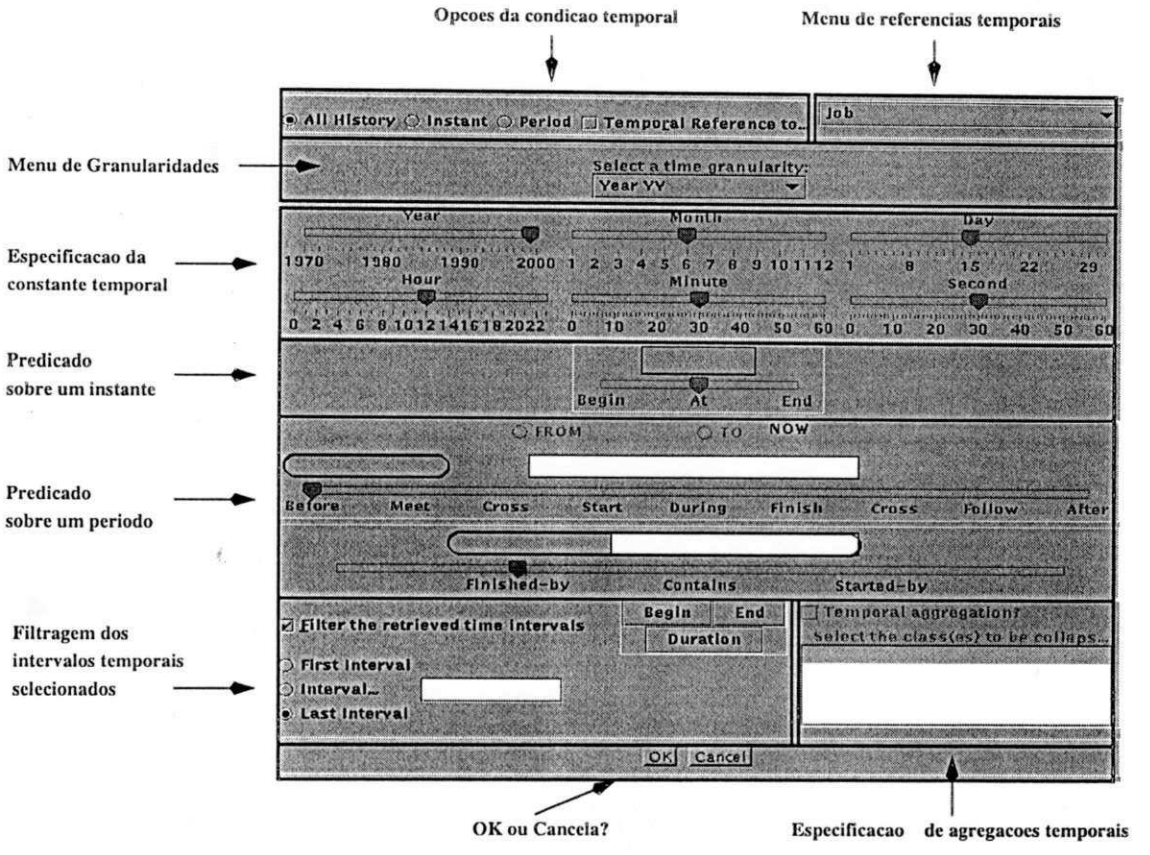

Figura 5.5: *Layout* do painel de especificação de uma condição temporal

qual representa uma seleção corrente sobre dados; ícone *When*, o qual representa uma seleção/projeção temporal; ícones *Data Visualization* and *Time Visualization*, onde os objetos que são instâncias correntes e históricas de uma classe ou relacionamento temporal serão visualizados no painel de visualização dos dados, respectivamente.

Adicionalmente, a interface contém dois painéis: **Janela de Esquemas** e **Janela de Interação**, as quais representam as fases de localização e manipulação da consulta, respectivamente.

A **Janela de esquemas** é basicamente constituída por uma área de trabalho (*workspace*) sobre a qual um esquema de banco de dados é visualmente representado em três diferentes formas: como uma árvore *top-down*, como um grafo e como um subgrafo (as três representações compartilham a mesma área de trabalho). Através da janela de esquemas, o usuário tem uma visão global das classes do esquema e seus inter-relacionamentos.

Considerando a mesma idéia de **contexto**, definida no projeto inicial (ver seção anterior), a janela de esquemas visualiza primeiro o esquema conceitual como uma árvore *top-down*, tendo contextos como raiz e nodos intermediários e seus nodos-folha representando classes, como mostrado na Figura 5.6 (lado esquerdo). Note que dentro da árvore de contextos, contextos, classes e classes temporais são visualmente representadas como quadrados, círculos e círculos sombreados respectivamente.

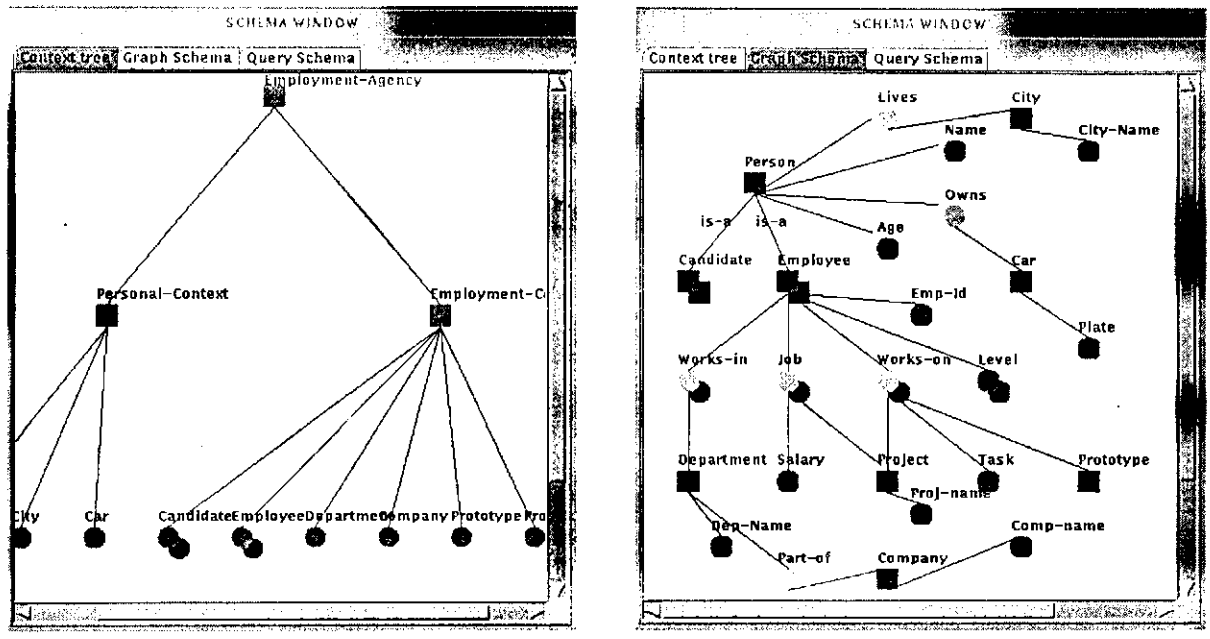

Figura 5.6: Um esquema de banco de dados visualmente representado como uma árvore de contextos e como um grafo

A Janela de esquemas também visualiza o esquema de banco de dados ou como um grafo, denominado de **Esquema Gráfico** (*Graph Schema*), como mostrado na Figura 5.6 (lado direito), ou como um subgrafo, denominado de **Esquema da Consulta** (*Query Schema*), o qual representa o subgrafo de interesse compreendendo apenas as classes e relacionamentos selecionados pelo usuário para a sua consulta (o esquema da consulta fornece uma visão intensional do resultado da consulta, o qual pode ajudar o usuário em abstrair uma grande quantidade de dados recuperados [76]).

O esquema gráfico possui quadrados representando classes de objetos (ex. *Person, Car*), atributos (ex. *Name, Age*), e círculos representando os relacionamentos

entre classes (ex. *Lives, Owns*). Quadrados e círculos sombreados representam classes/atributos temporais (ex. *Employee, Level*) e relacionamentos temporais (ex. *Job*), respectivamente. Adicionalmente, relacionamentos é-um (*is-a*) são expressos através do arco rotulado *"is-a"* (ex. *Employee is-a Person*).

A **Janela de interação** é basicamente composta por uma área de trabalho sobre a qual o esquema de banco de dados é visualizado como uma agenda gráfica (ver Figura 5.7), onde a "folha" da agenda representa ou um contexto ou uma classe. Sempre que a folha representa um contexto, os "índices" da agenda representam outros contextos ou classes, ao passo que se a folha representa uma classe, os índices representam todas as suas propriedades (atributos e relacionamentos).

Através desta janela, o usuário seleciona uma classe (ou melhor, um índice que representa a classe) como a classe alvo da consulta [73]. A partir de então, o esquema é visualmente representado como um grafo na janela de esquemas.

5.2.3 Exemplo

Mostraremos a interface através de um exemplo de uma consulta temporal (a consulta contém apenas uma classe alvo). Assumindo que o usuário está interessado em saber: *Quais salários os empregados ganhavam quando eles mudaram de nível pela última vez?*. Inicialmente, ele/a acessa um esquema de banco de dados (selecionando o item *Open* no menu). Os índices da agenda são dois contextos, denominados *Personal-context* e *Employment-context*, os quais representam informações sobre dados pessoais e dados empregatícios, respectivamente.

Desde que o usuário não visualiza a classe desejada no primeiro refinamento do esquema, ele/a pode expandir um contexto. Como consequência, uma nova folha aparece com o refinamento do contexto selecionado, como mostrado em Figura 5.7, onde o usuário selecionou o contexto *Employment-context*.

Os dois painéis são sincronizados: para cada índice selecionado, o nodo correspondente na janela de esquemas é também selecionado. Depois da seleção de um contexto, o usuário pode mudar para um contexto diferente (ou classe). Adicionalmente, ele/a

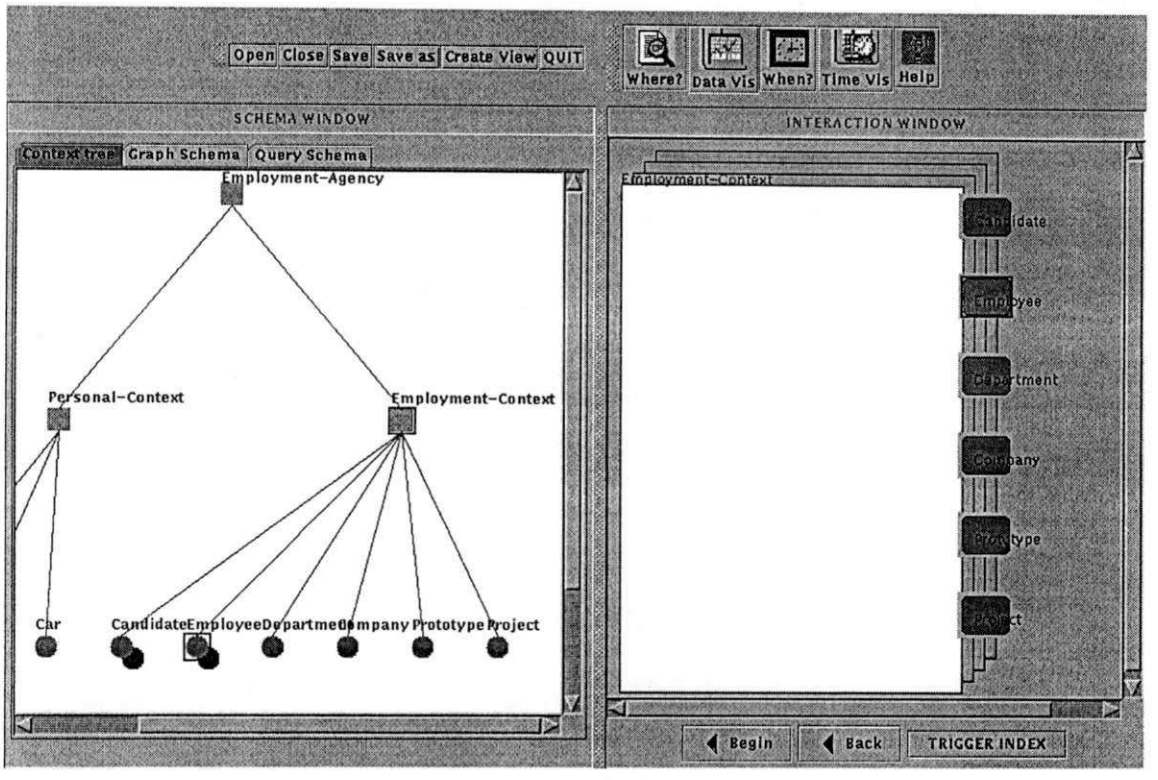

Figura 5.7: Os nodos selecionados *Employment-context* e *Employee*

pode "navegar" sobre a agenda através da seleção dos botões *Back* (refinamento anterior do esquema) ou *Begin* (refinamento inicial do esquema).

Desde que o usuário selecionou a classe *Employee* como a classe alvo da consulta, esta classe aparece como uma folha e os índices representam suas propriedades, como mostrado na Figura 5.8 (note que o sistema automaticamente visualiza o esquema como um grafo na janela de esquemas). O usuário pode expandir um relacionamento de forma com que apareçam as classes e atributos diretamente alcançáveis da classe *Employee*, através daquele relacionamento. Uma classe relacionada pode ser expandida em termos de suas propriedades.

Como no projeto inicial, nodos na janela de esquemas podem assumir três estados: *não selecionado, selecionado* e *visualizado* para o resultado. Os estados *selecionado* e *visualizado* são visualmente representados como um quadrado ao redor do nodo, onde o quadrado com espessura mais grossa representa o estado *visualizado*. Além disso, tal nodo assume a cor vermelha para se destacar dos outros nodos.

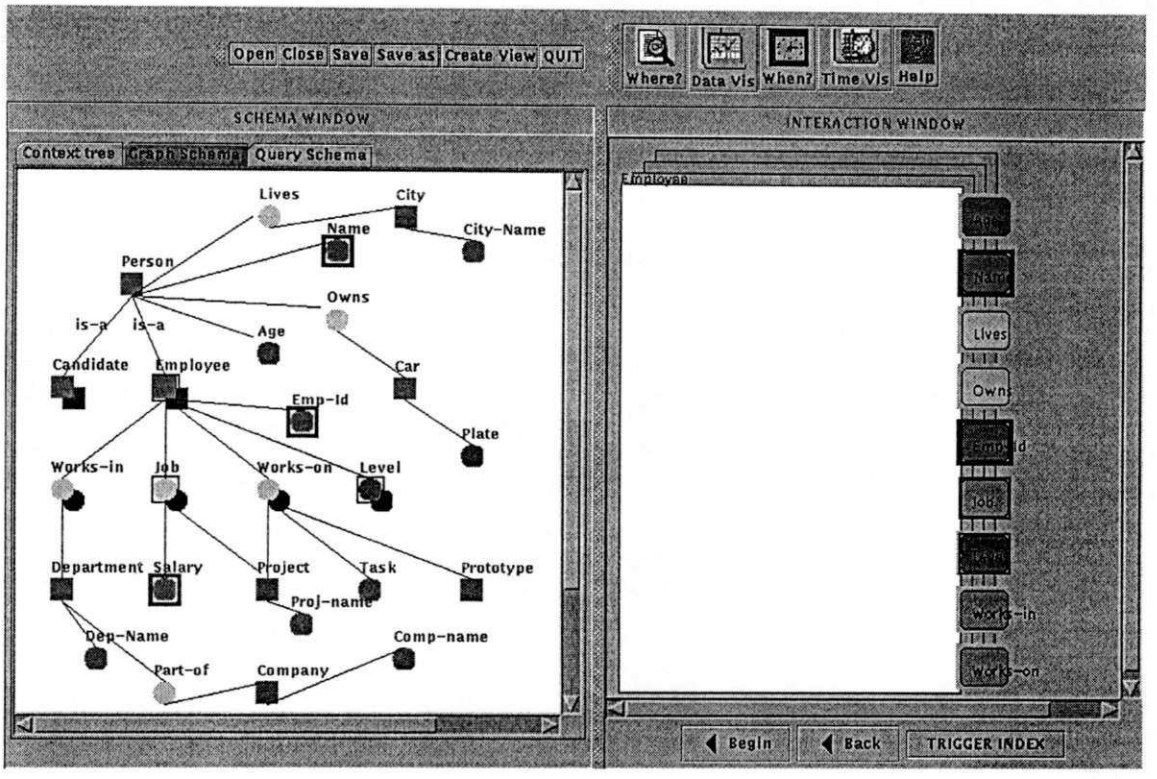

Figura 5.8: Nodos selecionados e visualizados da consulta

O usuário pode continuamente mudar o estado de um nodo através da seleção sucessiva do índice correspondente. Em nosso exemplo, *Job* e *Level* são nodos selecionados, enquanto *Salary*, *Emp-Id* e *Name* são nodos visualizados.

Em seguida, o sistema visualiza o subesquema apropriado, como mostra a Figura 5.9. O usuário pode ou salvar o esquema correspondente para uma posterior manipulação (selecionando o item de menu *Save As*), ou imediatamente usá-lo na fase de manipulação (selecionando o item de menu *Create View*). O usuário vai selecionando as classes de interesse para a consulta até estabelecer um "caminho" (*path*) correspondente a uma consulta elementar. Caso o usuário queira formar caminhos distintos, deve ser selecionado novamente uma classe alvo e as classes envolvidas neste novo caminho.

O usuário pode criar diversas visões através da seleção de outras classes alvo, e relacioná-los na consulta. Esta abordagem facilita a especificação de uma consulta complexa, a qual pode ser particionada em muitas visões.

Nossa consulta envolve uma seleção corrente e de tempo válido com uma **referência**

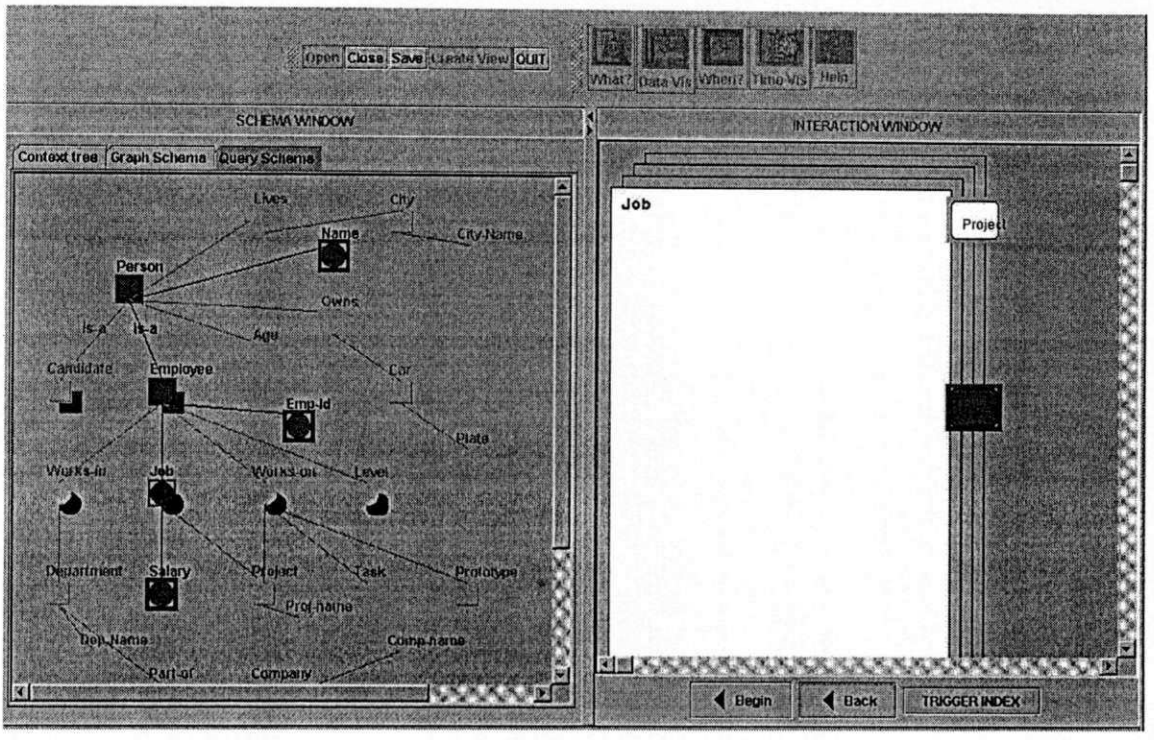

Figura 5.9: Esquema da consulta

temporal a outro dado, desde que se compara a história de um certo dado (histórico salarial) com a história de outro dado (histórico de níveis).

Na especificação da **seleção corrente** de um atributo, o usuário deve selecionar o atributo a ser filtrado e o ícone *Where?*. Como consequência, aparece um painel de diálogo que representa um editor de domínios.

O **editor de domínios** é parametrizado com base no tipo do atributo, tal como, um caractere, um valor numérico ou uma seleção de um menu de valores. O domínio de valores depende do atributo a ser filtrado. O editor também serve na edição de expressões booleanas, para predicados complexos. Por exemplo, a Figura 5.10 ilustra os efeitos da seleção do usuário sobre o atributo *City-name* e o ícone *Where*.

Na especificação da **seleção de tempo válido**, o usuário seleciona o atributo temporal *Level* e o ícone *When?*. Em seguida, as opções *all history, instant, period* e um *check box "temporal reference to..."* aparecem no lado superior de um novo painel.

A opção *all history* é usada para recuperar toda a história de uma classe ou relacionamento temporal, ao passo que as opções *instant* e *period* são usadas na extração

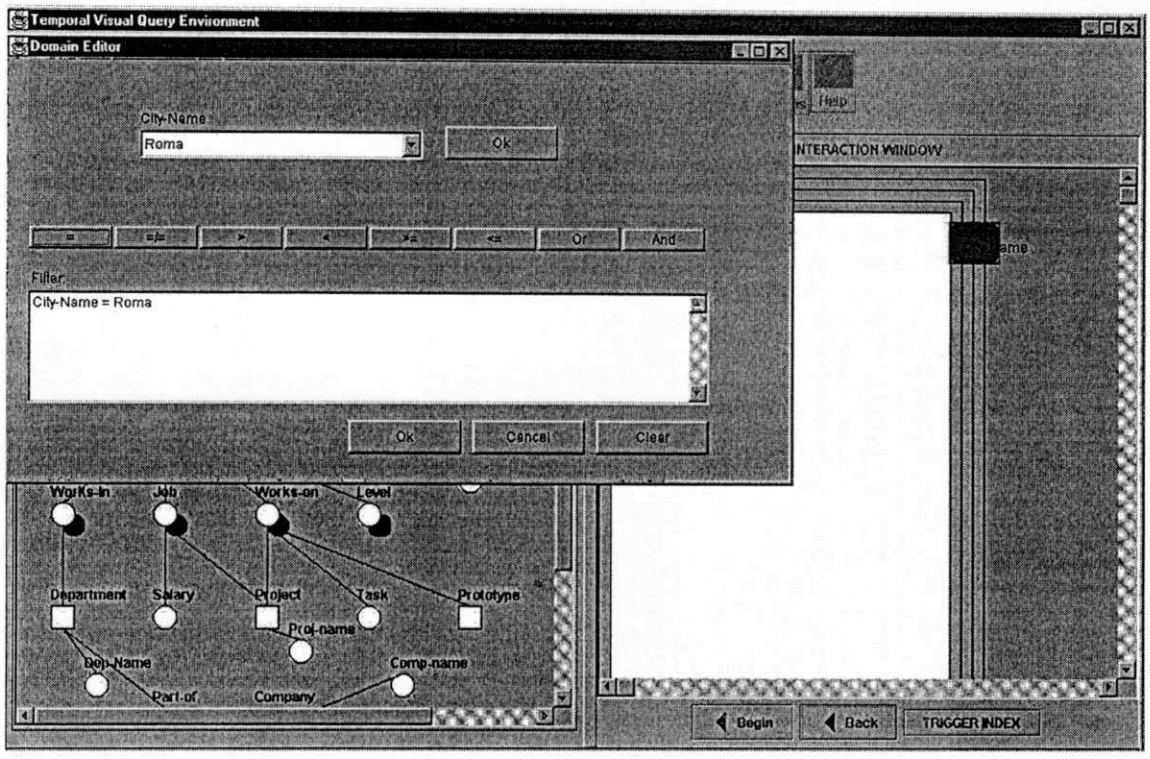

Figura 5.10: Editor de Domínios

de um fato instantâneo e histórico, respectivamente. Em nosso exemplo, o usuário seleciona a opção *all history*, como mostrado em Figura 5.11.

Uma vez que a história de níveis de cada empregado pode conter muitos intervalos temporais, o usuário pode selecionar um ou mais deles, e especificar se a seleção diz respeito ao ponto inicial, ponto final ou duração de cada intervalo. No exemplo, o ponto inicial do último intervalo associado ao nível dos empregados foi escolhido. Se a duração é selecionada, o sistema oferece um menu de funções agregadas *min, max, count, avg* e *sum*, as quais se aplicam ao conjunto dos elementos temporais dentro do relacionamento ou classe.

Os três painéis seguintes são usados para especificar uma condição temporal. O primeiro dos três painéis é usado para especificar ou um instante ou um período, dependendo da granularidade previamente selecionada pelo usuário, no menu de granularidades de tempo.

O próximo painel é ativado quando o usuário seleciona a opção *instant* (quando a

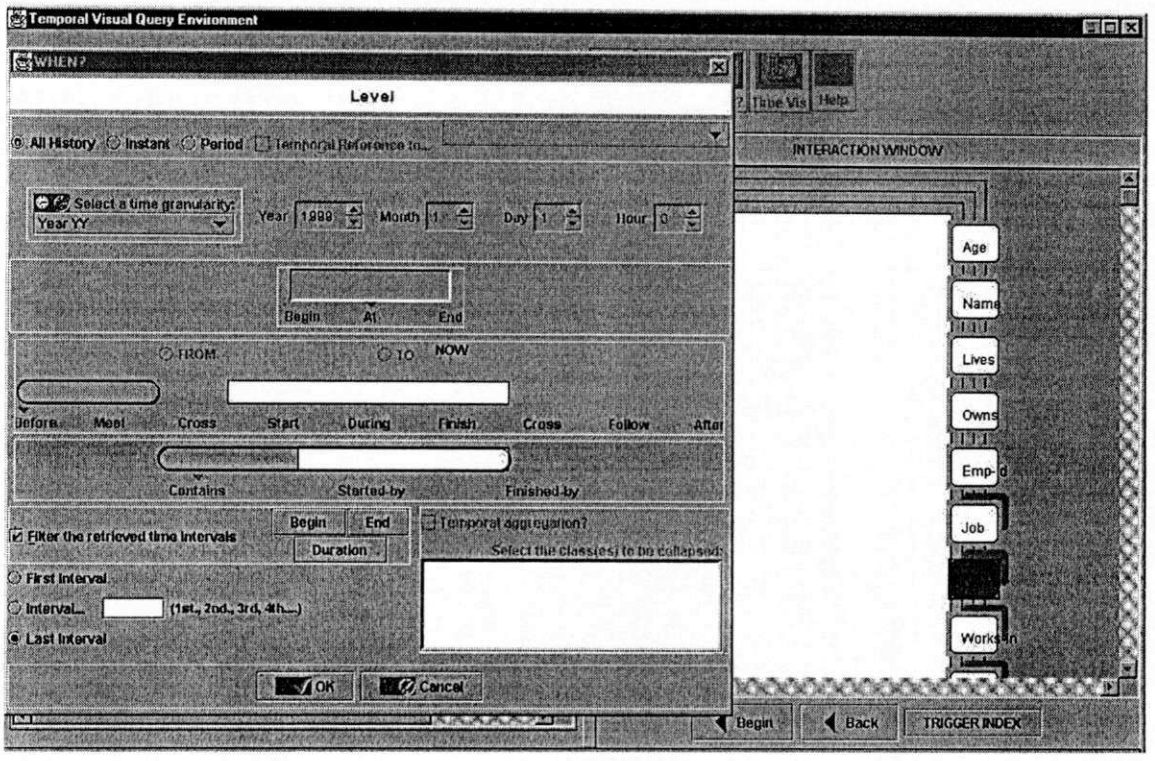

Figura 5.11: Condição de espera de uma consulta temporal

condição temporal refere-se a um instante de tempo). Seja t um instante de tempo. O operador *begin* (*end*) recupera apenas as instâncias cujo tempo de vida inicia (termina) em t, e com *at* o tempo t deve estar dentro do tempo de vida da instância.

O próximo painél é ativado quando o usuário seleciona a opção *period* (quando a condição temporal refere-se a um período de tempo), o qual apresenta dois *sliders* que contêm os predicados entre intervalos de tempo, definidos por *Allen* [4], da forma $before(I)$, $meets(I)$, $during(I)$, $starts(I)$, $finishes(I)$, $overlaps(I)$, e seus respectivos operadores simétricos $after(I)$, *met-by(I)*, $during(I)$, *started-by(I)*, *finished-by(I)*, *overlaped-by(I)* e $equal(I)$, onde I é um intervalo de tempo, indicando os objetos com tempos de vida que satisfazem o predicado.

Depois que o usuário especificou o início e o fim do período desejado, ele/a pode usar ou o *slider* que contém nove operadores, ou o outro *slider* que contém três operadores.

Por exemplo, a Figura 5.12 ilustra as condições temporais *Finishes* e *Cross (overlaps) 26/08/1996 to 19/05/1998* (a figura mostra apenas o fragmento do painel tem-

poral). Note que a área branca acima do *slider* representa o período especificado, e a posição espacial da área cinza com relação a área branca simboliza o relacionamento temporal entre eles (ex: na seleção do valor *Cross* (*Overlaps*), a área cinza parcialmente sobrepõe a área branca).

Figura 5.12: Dois movimentos do *slider* usados em um período

A ordem dos relacionamentos temporais dentro do *slider* com nove operadores é baseada no conceito de **primitivas temporais vizinhas** (*neighbors temporal primitives*), introduzida por *Freksa* em [54] e adicionalmente discutida em [61]. Dois relacionamentos temporais são **vizinhos** (*neighbors*) se uma mudança contínua dos eventos (ex: aumento, diminuição, ou movimento da duração dos eventos) pode ser usada na transformação de uma relacionamento temporal primitivo para outro sem passar através de um relacionamento temporal adicional [61].

Desde que os operadores *finished-by, contains* e *started-by* não fazem parte de tal ordem dentro do slider com nove operadores, eles foram incluídos em um segundo *slider*.

A Figura 5.13 também mostra os efeitos da seleção do ícone *When?* aplicado ao relacionamento temporal *JOB*. No exemplo, o usuário seleciona a opção *instant*. Ele/a também ativa a opção *"temporal reference to..."* e a opção *Level* no menu de classes temporais que podem ser usadas como referências temporais. Como consequência, o nome *"Level"* aparece acima do *slider*, onde ele selecionou o operador *at*, como mostrado na Figura 5.13.

Note que a opção *"temporal aggregation?"* é ativada e uma lista de classes as quais são relacionadas por *Job* aparecem. Com esta opção, o usuário pode selecionar algumas classes da lista, de forma a agrupar as instâncias de classes que não foram selecionadas (ver capítulo 3 para maiores detalhes). No exemplo ele/a seleciona a classe *Project*.

Neste ponto, a condição temporal foi completamente especificada, e o sistema gera o subgrafo de interesse que será usado como entrada para a conexão ao banco de dados.

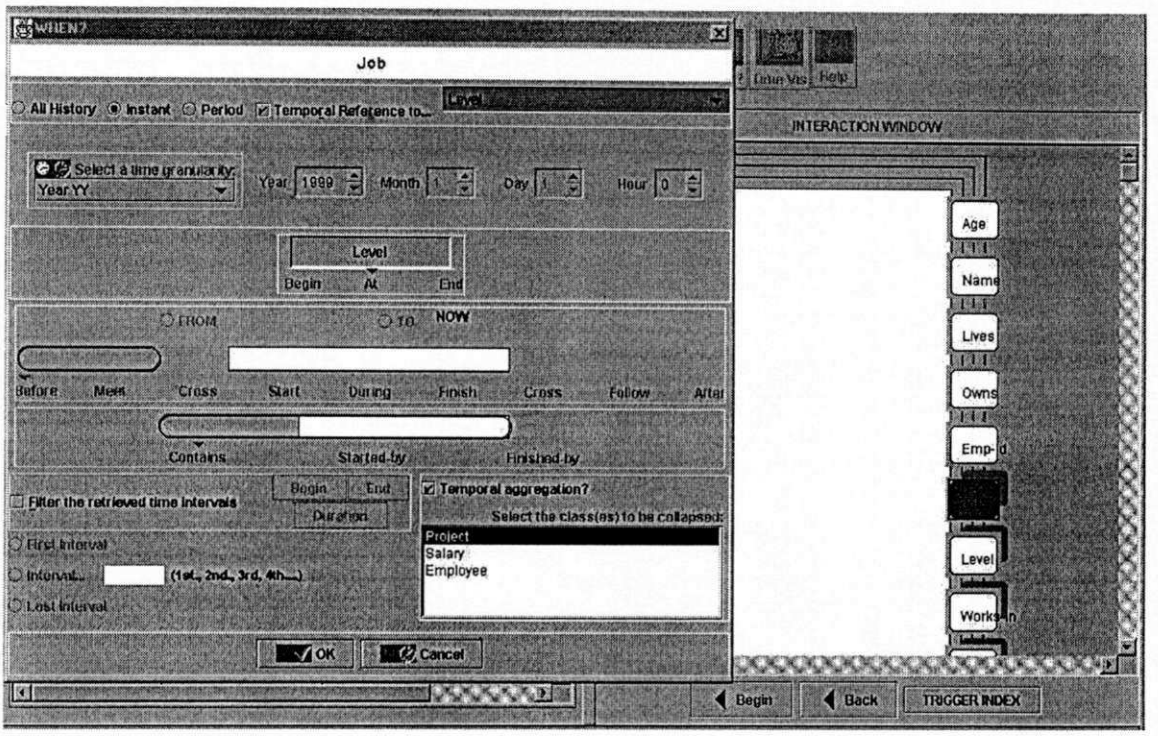

Figura 5.13: Condição de espera de uma consulta temporal

5.3 Trabalhos Relacionados

Descrevemos brevemente alguns trabalhos que compartilham algumas características do sistema TVQE.

5.3.1 Visualização de documentos hipermídia

A visualização inicial de esquemas complexos como uma árvore de contextos é semelhante à estrutura *pre-tree* descrita em [86].

Este trabalho explora um domínio de aplicação diferente do nosso, que se refere à visualização e navegação de documentos em sistemas hipermídia. Assim como a nossa abordagem, a motivação do trabalho de *Mukerjea et al* [86] se deve ao fato de que em qualquer sistema hipermídia com muitos nodos e arcos em uma estrutura diagramática complexa, sua visualização e compreensão se tornam difícil.

A estrutura *pre-tree* é um intermediário entre um grafo e uma árvore. Ele possui um nodo raiz, mas, diferente de uma árvore real, todos os seus descendentes não precisam

ser árvores, eles podem ser grafos arbitrários, como mostra a Figura 5.14, extraída de [86]. Portanto os descendentes formam uma lista de grafos, os quais representam as folhas da árvore e nodos de diferentes folhas não podem estar relacionados por arcos.

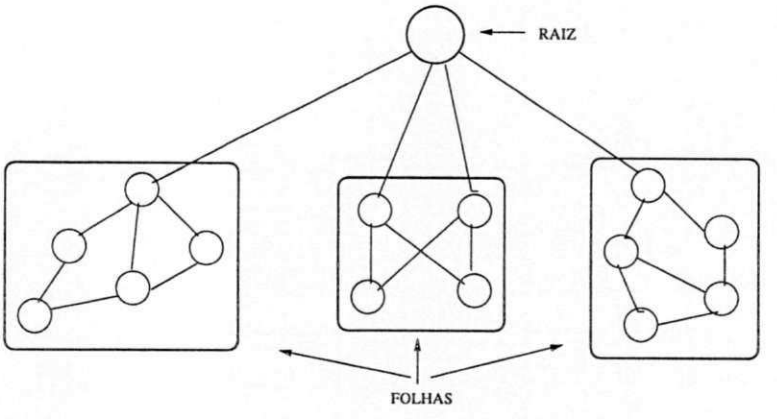

Figura 5.14: Um exemplo da estrutura *pre-tree*

A nível abstrato, o *pre-tree* é similar a nossa árvore de contextos na **representação** hierárquica da informação. Os dois apresentam um nodo raiz, e cada grafo arbitrário no *pre-tree* pode corresponder a um contexto na árvore de contextos. Contudo, a **visualização** hierárquica é diferente nas duas abordagens. Enquanto que a *pre-tree* visualiza um grafo como folhas da árvore, tais folhas na árvore de contextos representam as classes que compõem aquele contexto. O grafo mais detalhado do contexto é visualizado no esquema gráfico separadamente.

Portanto, enquanto que a abordagem de *pre-tree* visualiza diferentes perspectivas hierárquicas de um sistema em uma simples estrutura (mais adequada para a navegação sobre documentos hipermídia), a abordagem de árvore de contextos e esquema gráfico é mais adequada para as fases de *localização* e *manipulação* da consulta, respectivamente.

5.4 A interface $TabWorks^{tm}$

A utilização da metáfora da agenda em interfaces surgiu em outros domínios de aplicação, tais como em sistemas hipertexto [97], [98], [53], e em interfaces para o gerenciamento de informação pessoal utilizado em sistemas operacionais [17], [45].

TabWorks^{tm} (*TabWorks*^{tm} é uma marca registrada da *Xerox Corporation*) surgiu como um aperfeiçoamento da metáfora *desktop* da plataforma `Windows`, na organização de documentos e programas de aplicação.

Em *TabWorks*^{tm}, os índices representam diretórios. Na seleção de um índice aparecem os arquivos que fazem parte daquele diretório. Mais especificamente, os elementos principais da metáfora em *TabWorks*^{tm} são os seguintes: a capa da **agenda** aberta que compreende um conjunto de **índices**, cada um contendo uma ou mais **páginas**. Páginas contêm **ícones** representando documentos ou programas de aplicação, como mostra a Figura 5.15, extraída de [17]. Portanto, esta metáfora reflete visualmente a hierarquia *container* utilizada no gerenciamento de arquivos do sistema `Windows`.

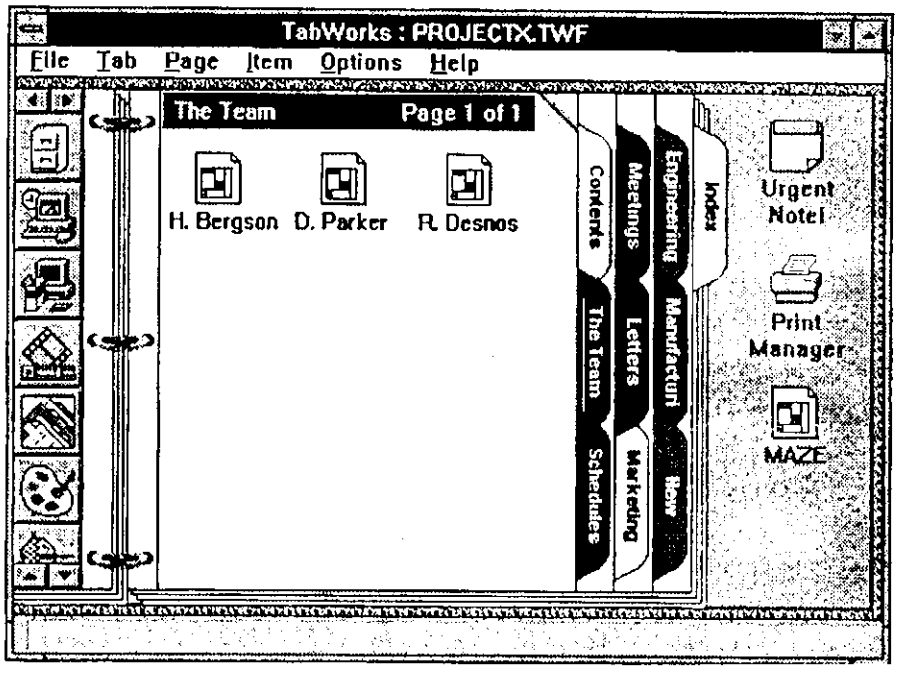

Figura 5.15: Metáfora de agenda utilizada em *TabWorks*^{tm}

Como *TabWorks*^{tm} é uma ferramenta de especificação, ela permite que o usuário possa criar, renomear e organizar índices, além de adicionar, renomear e eliminar páginas. Itens também podem ser adicionados e organizados dentro de uma página. Note que cada componente visual da agenda corresponde a um conceito específico.

Esta abordagem não poderia ser empregada no ambiente TVQE, pelo seguinte motivo: se adaptássemos a abordagem de *TabWorks*^{tm} ao nosso contexto, a nossa

metáfora de agenda apresentaria uma estrutura ilustrada na Figura 5.16.

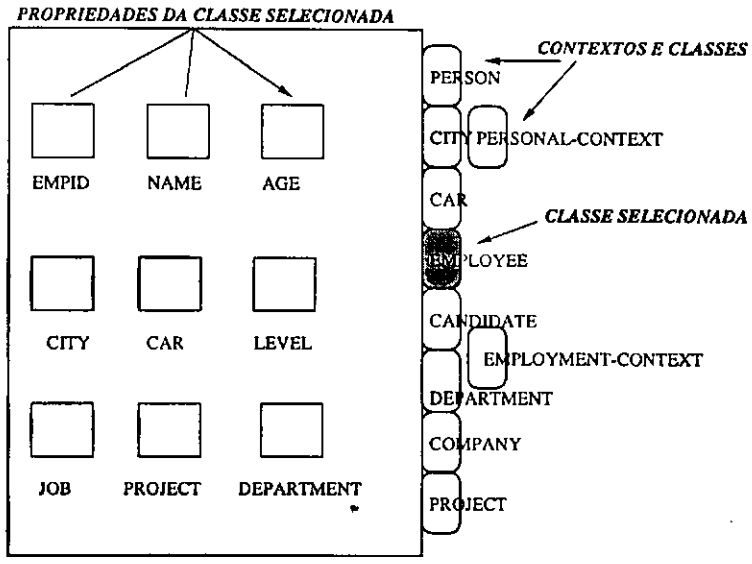

Figura 5.16: Outra estrutura de agenda no ambiente TVQE

Note que os componentes visuais distintos da agenda podem corresponder a um mesmo conceito (ex. índices e itens dentro da página podem corresponder a uma mesma classe). Isto gera redundância da informação, ou seja, uma mesma informação será visualizada mais de uma vez. Além disso, no índice aparecerão dois conceitos distintos ao mesmo tempo (ex. contextos e classes).

Portanto, mesmo que as duas interfaces explorem a mesma metáfora visual em domínios distintos de aplicação, a navegação sobre a informação nas duas abordagens é distinta. Enquanto que em $TabWorks^{tm}$ os índices são fixos, no TVQE a mudança de índices é sensível ao momento da interação.

As próximas interfaces já foram brevemente descritas no capítulo anterior.

5.5 A interface *Oggetto Desktop*

O uso da metáfora de agenda é original no acesso visual a bancos de dados. Contudo, a idéia de utilizar conceitos utilizados na metáfora *desktop* no acesso a bancos de dados orientado a objetos foi explorada pela interface *Oggetto desktop* [69]. O mecanismo de navegação sobre a estrutura do esquema conceitual é similar ao TVQE.

Adicionalmente, assim como a utilização da metáfora de agenda foi comprovadamente considerada como um aperfeiçoamento da metáfora *desktop* [17], [45], dentro do domínio de aplicação de organização da informação em sistemas operacionais, acreditamos que a nossa abordagem de utilizar tal metáfora no acesso ao banco de dados é um passo à frente do uso da metáfora *desktop* em *Oggetto desktop*.

5.6 A interface *VisTool*

O conceito de classe alvo na especificação de consultas em TVQE é relacionado à noção de **conceito primário** de *Benzi et al* [10], utilizado no protótipo `VisTool`. O conceito primário é associado a um **ponto de vista**, ou seja, uma perspectiva de acessar os dados por um caminho (*path expression*) definido.

Uma limitação do TVQE em relação ao *VisTool* é que este possui um mecanismo de solucionar o conflito entre diferentes pontos de vista, encontrados em um grafo cíclico.

Por outro lado, apesar de *VisTool* apresentar uma visualização híbrida para facilitar o relacionamento entre conceitos, sua limitação está na manipulação direta sobre o diagrama (discutido no capítulo anterior). Tal manipulação se torna confusa quando o esquema conceitual envolve muitas classes e associações entre elas. Adicionalmente, consideramos que a representação visual híbrida é complicada de implementar (realmente não foi ilustrado nenhum esboço inicial do protótipo em [10]).

5.7 Interfaces *TVQL* e *Lifelines*

A abordagem visual que os autores do TVQL [62] adotaram em uma parte da especificação de consultas foi base para a nossa abordagem em usar *sliders* para representar os predicados de *Allen* [4].

Apesar do seu grande poder de visualização, *Lifelines* [93] apresenta somente um objeto por vez. Desta forma, utilizaremos a sua abordagem na visualização de um objeto específico, quando o usuário interagir com a nossa visualização *spaghetti*, na janela de visualização dos dados (ver último capítulo).

Mais especificamente, a partir da visualização bidimensional ou tridimensional de diversos objetos como pontos no tempo, o usuário pode selecionar um objeto, e o sistema TVQE apresentaria uma estrutura visual similar ao *Lifelines* para apresentar o histórico deste objeto.

5.8 Conclusão deste capítulo

Neste capítulo foi apresentado uma proposta de um SVC, baseado nas primitivas gráficas temporais as quais foram definidas formalmente no capítulo 3. A Tabela 5.1 mostra as primitivas gráficas e seus correspondentes mecanismos visuais usado na interface TVQE (note que a abreviação $\vec{ET}$ significa *extensão temporal*).

Vale ressaltar que outros mecanismos visuais podem ser definidos a partir das primitivas gráficas.

Através das janelas de **esquemas** e de **interação**, o usuário especifica a sua consulta nas suas fases de localização e manipulação. Contudo, não foi desenvolvida a janela de **Visualização dos Dados**, que corresponde à fase de **Visualização** da consulta, onde o usuário visualiza e interage com os dados que representam o resultado da consulta

Adicionalmente, consultas temporais geram uma grande quantidade de informação, a qual obviamente não serve para ser visualizada em uma forma tabular. Através dos ícones *Data Visualization* and *Time Visualization*, o usuário visualizará a parte extensional do banco de dados em uma representação visual orientada ao tempo, de uma forma bidimensional ou tridimensional . O tipo do dado (propriedades quantitativas e qualitativas do dado) deve ser considerado de forma a produzir uma visualização efetiva.

Considerando aspectos de interação, usaremos a abordagem de consulta dinâmica (ver capítulo prévio) [1], [100] de forma a fornecer uma "navegação" interativa sobre o tempo (ver último capítulo).

PRIMITIVAS GRÁFICAS	TVQE INTERFACE
Seleção de um nodo	Seleção sucessiva sobre o índice representando o nodo.
Desenho de um arco entre os nodos n e q	Seleção sobre os índices representando *n* e *q* e o ícone *Where*.
Mudança do rótulo de um arco entre os nodos s e q	Seleção sobre o índice representando *q* e o ícone *Where*.
Seleção de nodo(s) sombreado(s)	Seleção sobre índice(s) representando nodo(s) e o ícone *When*.
ET da mudança do rótulo de um arco entre o nodo s e o nodo-papel q	Seleção sobre o índice representando o nodo *q* e o ícone *When*; Seleção das opções *instant* ou *period*; Especificação da condição temporal através dos *sliders*.
Mudança do rótulo de um nodo	Seleção sobre o índice representando o nodo e o ícone *When*; Seleção do *check box "filter o retrieved time intervals"*; Seleção do intervalo de tempo correspondente
Seleção do rótulo de um nodo	Seleção sobre o índice representando o nodo e o ícone *When*; Seleção do *check box "temporal aggregation"*; Seleção de classes na lista para serem eliminadas.
ET do desenho de um arco entre os nodos n e q	Seleção sobre o índice representando o nodo e o ícone *When*; Seleção do *check box "temporal reference to..."* e da classe representando *q* no menu de referências temporais; Seleção Seleção das opções *instant* ou *period*; Especificação da condição temporal através dos *sliders*.

Tabela 5.1: TGPs e as correspondentes ações dentro da interface TVQE

Capítulo 6

Aspectos de Implementação

Um aspecto relevante a ser considerado no projeto de uma interface gráfica é a ferramenta de especificação sobre a qual a interface será desenvolvida. Os critérios exigidos na escolha da ferramenta de especificação são a sua flexibilidade, interoperabilidade (multi-plataforma), ambiente horizontal (programação em alto nível), além de rapidez e eficiência.

De acordo com estes critérios, o ambiente está sendo desenvolvido na linguagem orientada a objeto `JAVA`, dentro do ambiente de desenvolvimento `JDK 1.1.6` e `JFC Swing 1.1`, o qual contém uma biblioteca de classes que representam os componentes gráficos utilizados na interface.

Neste capítulo descreveremos aspectos de implementação do ambiente TVQE, considerando:

1. As principais características da linguagem `JAVA` e uma breve descrição do componente `JAVA` de conexão ao banco de dados, `JDBC`;
2. Arquitetura de implementação;
3. Estrutura das classes que foram utilizadas na aplicação.
4. Estrutura de dados que foi utilizada na aplicação.

6.1 Características da Linguagem JAVA

As principais características da linguagem JAVA, que a diferenciam das outras linguagens de programação, são definidas a seguir.

- JAVA é uma linguagem de programação orientada a objeto. Realmente, em JAVA são incluídos os mecanismos de encapsulamento, hereditariedade e polimorfismo.

- As aplicações em JAVA são portáteis em múltiplas plataformas: diferente da maior parte das linguagens de programação que compilam o código fonte e o transformam em um código de máquina (código nativo), que é otimizado por uma plataforma de hardware específica, um compilador JAVA traduz o código fonte em *byte code*, o qual representa um grupo de instruções independente de um microprocessador específico. A vantagem é a possibilidade dos programas executáveis em JAVA funcionarem em qualquer sistema operacional dotado de uma máquina virtual (*Virtual Machine - VM*) JAVA.

  Como se fosse uma verdadeira CPU, ao momento da execução, a máquina virtual interpreta o *byte code* JAVA correspondente a uma plataforma *hardware* e *software* específica, como mostra a Figura 6.1.

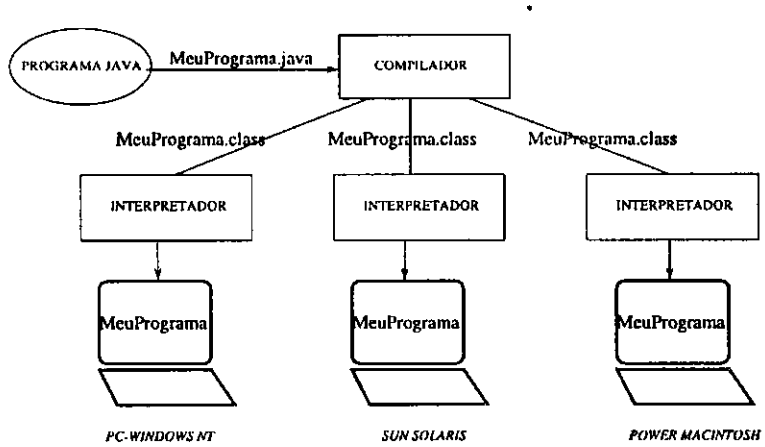

Figura 6.1: Independência de Plataforma

- JAVA é distribuído. JAVA possui uma grande biblioteca de procedimentos para a utilização de protocolos da arquitetura TCP/IP como HTTP e FTP.

As aplicações em JAVA podem acessar objetos remotos através de uma URL, como se estes objetos fossem locais. Considerando o aspecto de segurança em ambientes distribuídos, JAVA possui um próprio sistema de proteção contra vírus e uso indevido de recursos locais do sistema.

- É possível escrever aplicações em JAVA que sejam executáveis seja no interior de um *browser* (ex. *Netscape*), seja como um programa independente (*stand-alone*), em sistemas operacionais dotado da máquina virtual JAVA.

- As aplicações são robustas porque JAVA gerencia diretamente a memória do sistema, através da utilização de um *garbage collector*, um processo automático que executa em *background* e libera a memória quando o programa não a necessita mais. Realmente, o compilador JAVA revela muitos problemas que em outras linguagens se descobriria somente em tempo de execução.

  A máquina virtual JAVA controla automaticamente as exceções relativas a limite de um vetor, uso de apontadores nulos, divisão por zero, conversões não possíveis entre caracteres e números e muitas outras exceções que poderiam causar um *crash* no programa em tempo de execução.

- As aplicações interativas e em tempo real são eficientes desde que o *multithread* intrínseco de JAVA permite a uma aplicação desenvolver atividades simultâneas.

6.1.1 Conexão ao Banco de Dados

O API de JAVA de conexão ao banco de dados, JDBC, fornece as classes que possam gerenciar a maior parte da problemática inerente ao acesso a bancos de dados. De simples comandos de seleção à manipulação de resultados de consultas.

O componente JDBC é organizado em dois níveis: o nível de aplicação, denominado *Application Layer* e o nível de banco de dados, denominado *Driver Layer*. Este último é responsável pela interação direta com o banco de dados, através de chamadas às tabelas relacionais que constituem o banco de dados. A comunicação entre os dois níveis somente é possível, respeitando o protocolo especificado pelo JDBC.

Driver Layer

Esta camada contém duas interfaces: *Driver* e *DriverManager*. A primeira é estreitamente relacionada ao conceito abstrato de acesso ao banco de dados, dentro da aplicação, enquanto que a segunda fornece todos os métodos fundamentais para a conexão ao banco de dados, estabelecendo parâmetros de conexão aos *drivers* do banco de dados. O procedimento de conexão ao banco de dados parte da definição de um **caracter de conexão**, o qual permite definir diversas propriedades de conexão, tais como, o tipo de banco de dados usado, informações da máquina servidora (*host-name, porta, db-name, UID e password*), etc.

Application Layer

Esta camada contém três interfaces: *Connection, Statement* e *ResultSet*.

A interface *Connection* representa uma sessão de conexão ao banco de dados. No momento em que a conexão é criada, a manipulação ao banco de dados é feita através da utilização da classe *Statement*, o qual permite enviar instruções em SQL ao SGBD específico. A classe *ResultSet* é responsável pela utilização das tuplas da tabela que representa o resultado da consulta.

6.2 Arquitetura de Implementação

Esta seção descreve a arquitetura de implementação, considerando a conexão ao banco de dados. Antes de mostrar a arquitetura, descrevemos brevemente o modelo de conexão ao banco de dados, **cliente-servidor** (*2-tier*).

6.2.1 Modelo Cliente-Servidor

Este modelo é constituído de um programa **cliente** que interage com um **servidor** sobre o qual se localiza o banco de dados. O cliente se ocupa de todo o gerenciamento de acesso aos dados e da utilização do *driver* de conexão específico para o banco de dados a ser conectado. O servidor não possui iniciativa autônoma, ele realiza as suas atividades

"por demanda", entre as quais, o fornecimento dos dados requisitados pelo cliente. Por simplicidade, nosso trabalho adota uma arquitetura cliente-servidor, efetuando uma conexão direta ao servidor que contém o banco de dados, como mostra a figura 6.2.

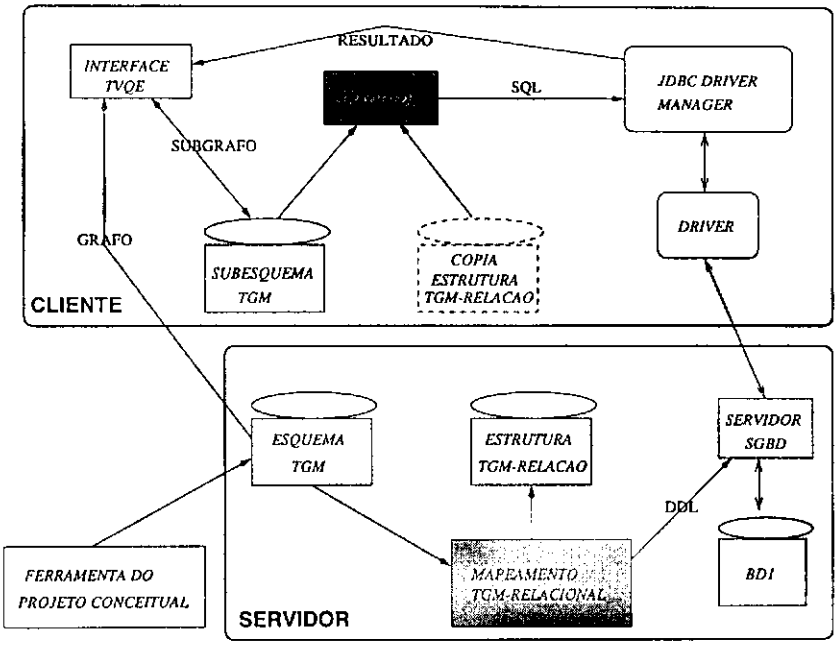

Figura 6.2: Arquitetura de implementação do ambiente TVQE

Dentro do ponto de vista do servidor, um esquema conceitual em formato TGM pode ser criado através do módulo **Ferramenta do Projeto Conceitual**. O esquema TGM serve de entrada para o módulo **Mapeamento TGM-Relacional**, o qual gera uma DDL (*Data Definition Language*) para um SGBD relacional específico. Este módulo também cria uma estrutura que relaciona o esquema gráfico e sua correspondente estrutura relacional. Esta estrutura é duplicada na parte cliente do sistema.

Dentro do ponto de vista do cliente, o usuário acessa um banco de dados através da interface TVQE, sobre o qual o esquema conceitual TGM é visualizado como um grafo. O sistema gera um subgrafo de interesse (que pode ser utilizado para posteriores manipulações) que será convertido em linguagem SQL através do módulo **Tradutor**. Note que este módulo necessita da estrutura que relaciona o esquema gráfico e sua correspondente estrutura relacional, de forma a fazer a conversão em SQL. Através do JDBC, o SGBD retorna com o resultado da consulta.

O modelo cliente-servidor é ótimo em redes de pequenas dimensões (rede em um ambiente de escritório, com poucos clientes conectados em um sistema operacional homogêneo), porém é inadequado em redes de grandes dimensões.

Em uma rede de grandes dimensões, é necessário fornecer e manter o software de conexão (ex: *driver* para cada cliente). Com muitos usuários conectados, é necessário um controle centralizado dos acessos. Tal controle não existe no modelo cliente-servidor. Com este tipo de arquitetura, se sabe exatamente como são memorizados os dados e se conhece toda a lógica do banco de dados. Se existe qualquer mudança em relação à conservação dos dados ou o modo como estes são memorizados, se deve mudar todos os clientes. Portanto, pretendemos futuramente adotar a arquitetura **três níveis** - *3-tier* (cliente-servidor intermediário-servidor).

Uma das vantagens da arquitetura em três níveis é a possibilidade de alterar o modo sobre o qual são memorizados os dados, sem que isto haja efeito sobre o cliente. Com esta arquitetura, não é incluído no cliente o *software de conexão* aos dados, nem a implementação do acesso aos dados. O único ponto de acesso ao banco de dados reside no servidor intermediário. Isto permite simplificar esforços para a realização de várias políticas de controle e segurança. Adicionalmente, o servidor intermediário pode disponibilizar aos clientes dados que são residentes em bancos de dados diversos.

6.3 Estrutura de Classes

Como notação da estrutura de classes do ambiente TVQE, utilizamos a linguagem de modelagem visual UML (***Unified Modeling Language***), para especificação, construção e documentação de aplicações de *software*.

Tal metodologia, através de uma notação gráfica uniforme e independente de linguagem, pode ser utilizada para documentar, analisar os requisitos de um problema, projetar uma solução e facilitar a implementação em uma linguagem de programação (maiores detalhes sobre UML podem ser encontrados em [3]).

Antes de descrever a estrutura de classes do ambiente TVQE, descrevemos brevemente o que é um diagrama de classes em UML.

Este diagrama contém os seguintes elementos: **classes**, as quais representam entidades com características comuns e são visualmente representadas como retângulos. Estas características incluem atributos, operações e associações; e **associações**, as quais relacionam duas ou mais classes e são visualmente representadas como arcos.

Por simplicidade, consideramos três tipos de associações: **composição, agregação** e **generalização**, representadas visualmente como um arco com losango preto, um arco com losango branco e um arco com uma seta branca respectivamente, onde a classe associada com o losango representa a classe composta ou agregada. A agregação existe somente em situações em que a classe agregada pode ser removida sem haver a necessidade de remover as classes componentes, caso contrário, é melhor representar o relacionamento como composição.

Adicionalmente, incluímos no diagrama uma seta direcionada de uma classe A para uma classe B, indicando que a classe B é inicializada a partir da classe A.

As Figuras 6.3, 6.4, 6.5, 6.6, 6.7 e 6.8 ilustram o diagrama de classes em alto nível do ambiente TVQE. O diagrama de classes em alto nível não visualiza as variáveis de instância e os métodos de cada classe. Também não consideramos aqui as classes internas de JAVA.

A classe principal do ambiente TVQE é a classe *applet Agenda* da Figura 6.3. Ela constrói a janela principal sobre a qual são apresentadas a janela de esquemas e a janela de interação. Ela também constrói a estrutura interna do esquema conceitual selecionado pelo usuário.

Basicamente a classe *Agenda* é composta de classes que representam visualmente o esquema conceitual:

- A classe *treeCanvas* visualiza a árvore de contextos;
- A classe *agendaCanvas* visualiza a "agenda gráfica";
- A classe *graphCanvas* visualiza o esquema gráfico;
- A classe *subGraphCanvas* visualiza o esquema da consulta e é uma subclasse da classe *graphCanvas*.

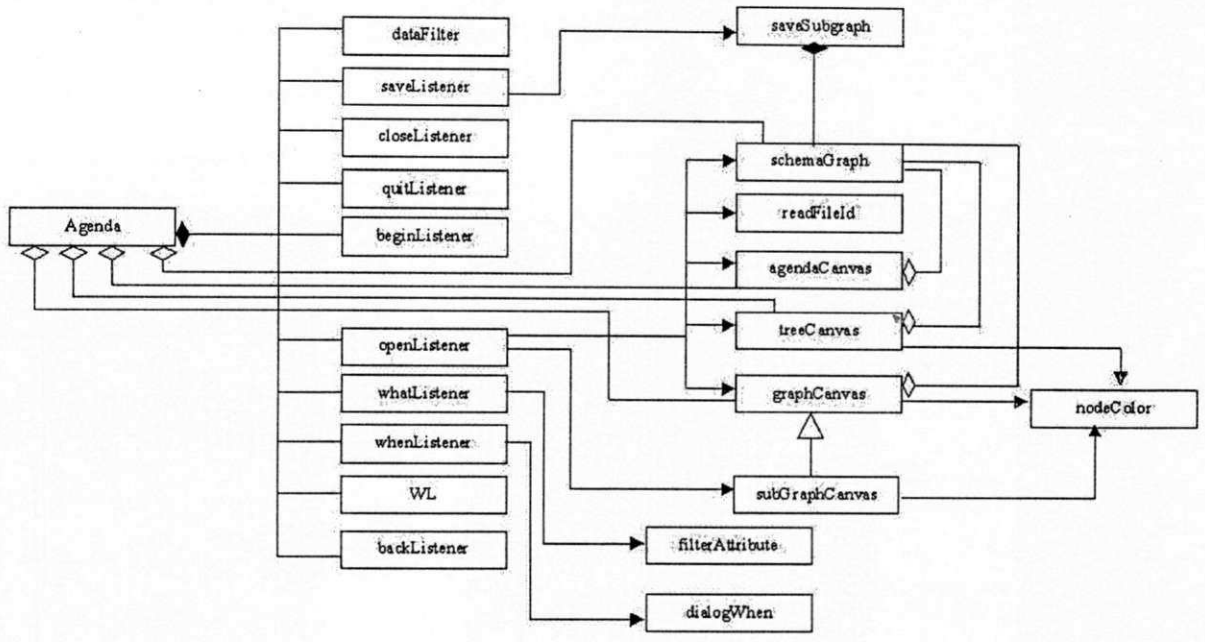

Figura 6.3: Diagrama de Classes 1

Estas classes de visualização possuem a classe *schemaGraph*, a qual representa a estrutura interna do sistema, descrita com mais detalhes na próxima seção.

Vale ressaltar que todas as classes do sistema são compostas de classes de evento (classes que terminam com a palavra *Listener*) associadas aos componentes gráficos que representam as funcionalidades do sistema. A partir das classes de evento, algumas classes são inicializadas, como as classes *saveSubgraph*, que gera uma visão da consulta ou subgrafo de interesse; *nodeColor*, que é responsável pela visualização das cores dos nodos e dos índices da agenda; e as classes *filterAttribute* e *dialogWhen*.

Além disso, a classe *agendaCanvas* inicializa as classes *visualGraph*, a qual contém informações visuais de cada interação do usuário com os índices da agenda; *iconEnable*, que é responsável pela sincronização dos ícones, de acordo com os índices selecionados pelo usuário; e a classe *statusAttribute*, que atualiza dinamicamente o estado de não selecionado para selecionado (ou vice-versa) dentro do arquivo que é utilizado no editor de domínios, como mostra a Figura 6.4.

O diagrama de classes da Figura 6.5 contém as classes que compõem o painel de diálogo (classe *dialogWhen*), utilizado na especificação de uma condição temporal.

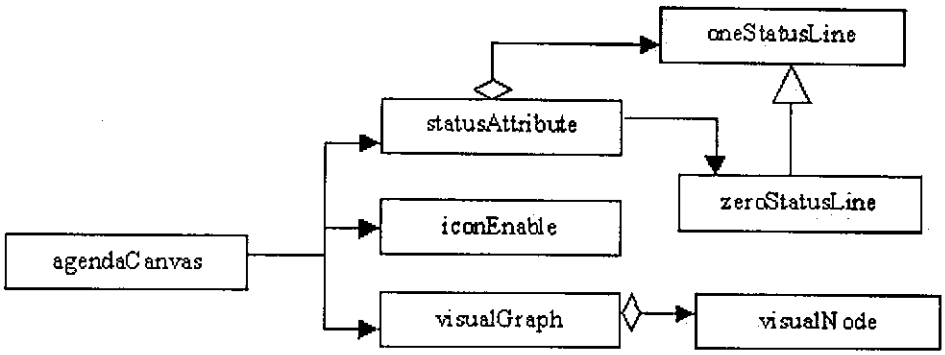

Figura 6.4: Diagrama de Classes 2

Mais especificamente, as classes *Choice*, *slidersPanel*, *granularity*, *instantPanel*, *periodPanel* e *referencePanel* e *listPanel* (mostrado na Figura 6.6) constróem os painéis que compõem o *dialog box When?*, descritos no capítulo anterior.

Outras classes adicionais são:

- A classe *initiallyEnable* é responsável pelo estado inicial dos painéis;
- A classe *EnablePanel* é responsável pela sincronização dos painéis, dependendo da opção selecionada no painel superior do painel de diálogo *When?*;
- A classe *enableGranularity* representa a ativação (ou desativação) da granularidade de tempo selecionada pelo usuário;
- A classe *aggregationNode* visualiza as classes relacionadas por um nodo temporal específico, dentro do painel de agregação temporal;
- A classe *temporalNode* insere nodos temporais dentro do menu de referências temporais;
- A classe *temporalPredicate* insere o predicado temporal especificado pelo usuário dentro do subgrafo de interesse da consulta.
- A classe *periodPanel* que inicializa as classes que representam o componente visual utilizado dentro dos *sliders* dos predicados temporais.

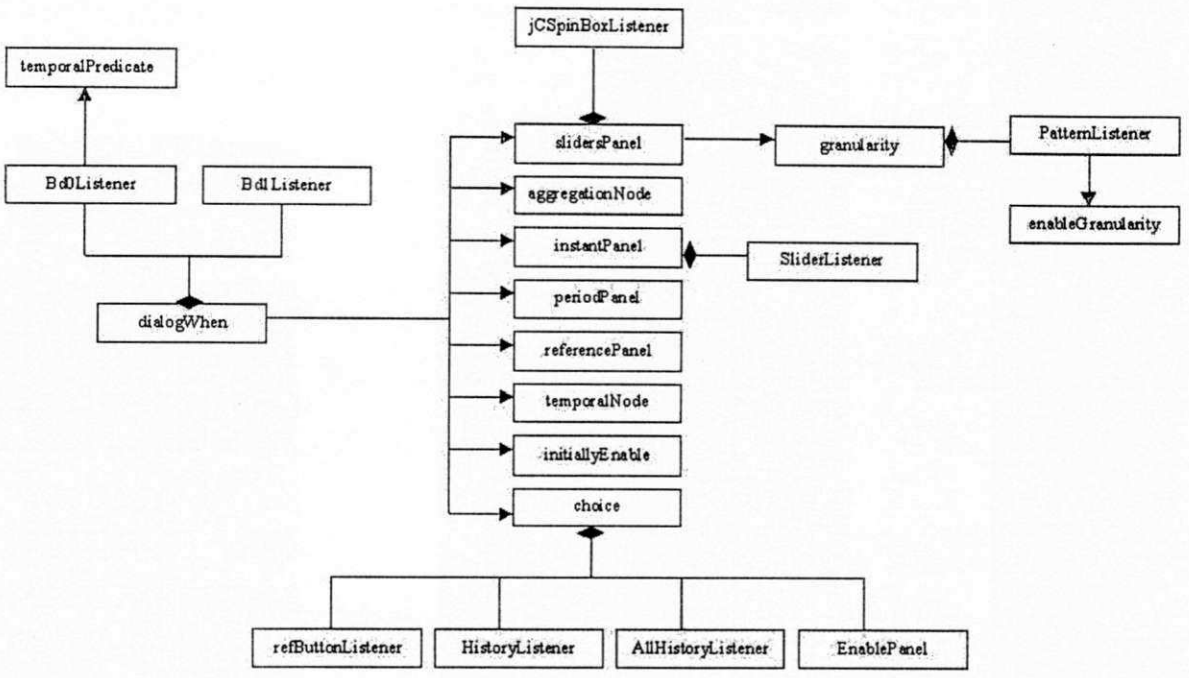

Figura 6.5: Diagrama de Classes 3

O diagrama de classes da Figura 6.7 contém as classes que compõem o editor de domínios (classe *Domain*), utilizado na especificação de uma condição convencional. Esta classe é inicializada a partir da classe *filterAttribute*, que verfica em um arquivo de tipos dos atributos, o atributo que foi selecionado pelo usuário.

Mais especificamente, a classe *Domain* é uma superclasse das classes *editorCaracter*, *editorNumeric* e *editorFile*, os quais compartilham o painel superior do editor de domínios. Cada um é ativado dependendo do tipo do atributo a ser filtrado. Esta classe é basicamente composta dos operadores de comparação e dos predicados booleanos. Adicionalmente, a classe *filterAttribute* visualiza o predicado no editor, enquanto que a classe *Predicado* insere o predicado especificado pelo usuário dentro do subgrafo de interesse da consulta.

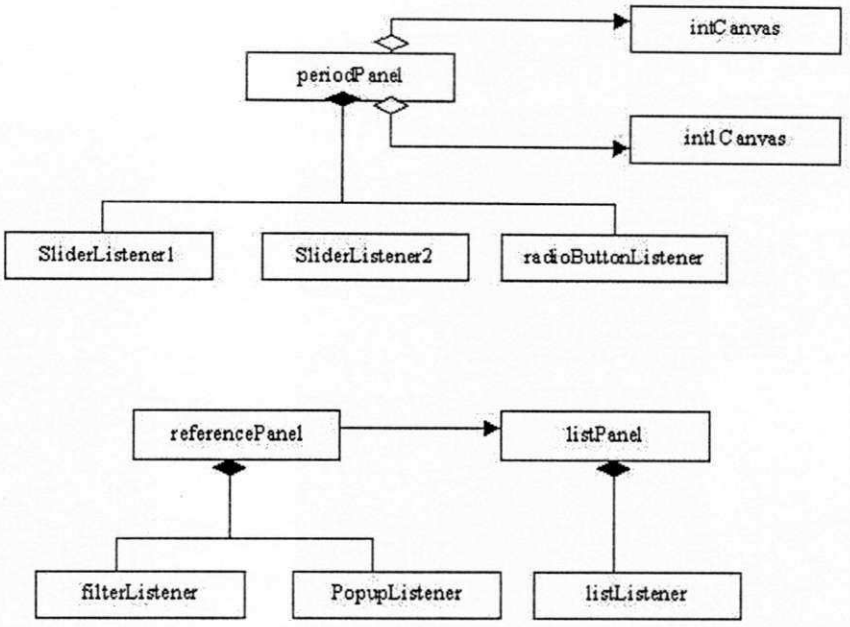

Figura 6.6: Diagrama de Classes 4

6.4 Estrutura de Dados e algoritmos

Utilizamos um **grafo** como estrutura interna e homogênea do esquema de banco de dados, de forma a facilitar a visualização do mesmo como uma árvore *top-down*, como um grafo, como um subgrafo, e como uma "agenda gráfica", onde os nodos representam as classes do esquema e os arcos representam o relacionamento entre as classes (esta estrututura é mantida durante toda a execução da aplicação).

A estrutura do esquema conceitual (classe *schemaGraph*) compreende os seguintes componentes estruturais, como ilustra a Figura 6.8:

- Um **vetor de nodos**;
- Uma **lista de adjacência** (classe *linkList*) para representar os **arcos** do grafo[1]. Uma lista de adjacência é um vetor de listas, onde cada lista individual representa os nodos adjacentes de um determinado nodo (a Figura 6.9 ilustra um exemplo de um grafo e sua correspondente lista de adjacência).

[1] Uma outra abordagem para representar os arcos de um grafo é o uso de uma *matriz de adjacência.*

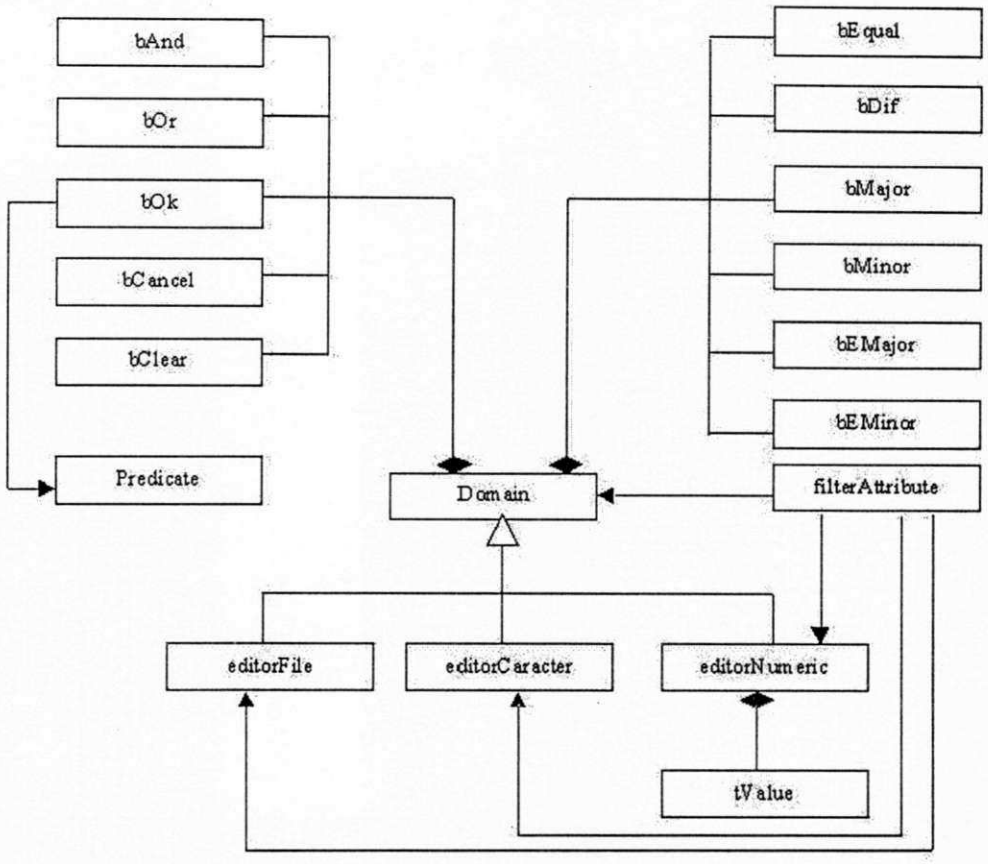

Figura 6.7: Diagrama de Classes 5

- Uma estrutura de **pilha** (classe *schemaStack*), utilizada no algoritmo *Depth-First Search* (definido a seguir), de forma a facilitar a navegação sobre o grafo;
- Uma estrutura de **fila** (classe *schemaQueue*), utilizada no algoritmo *Breadth-First Search* (definido a seguir), para percorrer todos os nodos do grafo;
- Informação visual dos nodos que foram selecionados pelo usuário (classe *visual-Graph* e o vetor *visualNode*).

Cada **nodo** (classe *schemaNode*) do grafo possui informações sobre uma dada classe, tais como:

- Número de identificação do nodo;
- O nome (*label*) da classe;

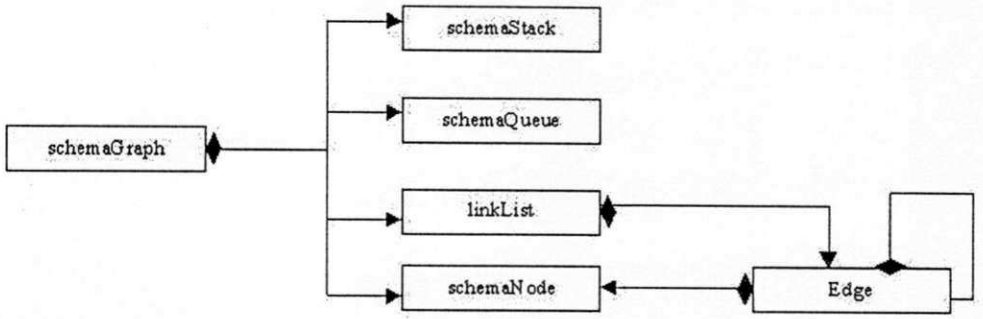

Figura 6.8: Estrutura Interna

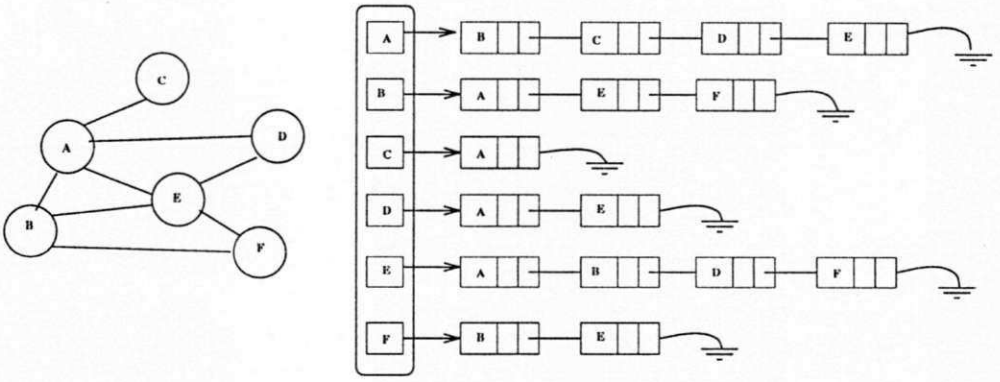

Figura 6.9: Exemplo de um grafo e sua lista de adjacência

- O tipo da classe: se a classe é do tipo *contexto, classe (classe não imprimível no modelo TGM), classe temporal, atributo (classe imprimível no modelo TGM), atributo temporal, relacionamento* ou *relacionamento temporal*;

- O estado da classe: a classe pode assumir os estados de *não selecionado, selecionado* e *visualizado*;

- Um valor booleano indicando se o nodo foi *visitado* na navegação (utilizado somente na árvore de contextos e no esquema gráfico);

- O nível da classe dentro da árvore de contextos;

- A posição ordinal da classe dentro do nível descrito acima;

- As coordenadas X e Y de visualização da classe dentro do esquema gráfico;

- Um valor booleano indicando se o nodo foi *selecionado* (utilizado somente na agenda gráfica).

O **arco** (classe *Edge*) possui as seguintes informações:

- O nodo origem do arco;
- O tipo do arco: um arco pode pertencer a um dos seguintes tipos: *normal, herdado, classe-contexto, subclasse-classe* e *componente-agregado*;
- Um apontador para a classe *Edge*.

6.4.1 Algoritmos DFS e BFS

Umas das operações fundamentais para percorrer sobre um grafo é encontrar quais nodos podem ser alcançados a partir de um nodo específico.

Existem duas abordagens comuns para pesquisar um grafo: algoritmo DFS (*depth-first search*) e algoritmo BFS (*breadth-first search*). Ambos eventualmente alcançarão todos os nodos conectados. A diferença é que a pesquisa DFS é implementada com uma pilha, ao passo que BFS é implementada com uma fila. Desta forma, o grafo é pesquisado em duas formas diferentes.

No ambiente TVQE, o algoritmo DFS é necessário na seguinte situação: a cada seleção de um índice na agenda, deve-se percorrer todos os seus nodos adjacentes (atributos e relacionamentos) e visualizá-los, e para calcular a profundidade máxima da árvore de contextos, enquanto que o algoritmo BFS é necessário na seguinte situação: visualização da árvore de contextos, grafo e subgrafo, e para calcular o número máximo de nodos a cada nível da árvore de contextos.

Capítulo 7

Acesso Homogêneo aos Dados Temporais e Histórico de Interações

Como foi discutido na introdução deste trabalho, recentes aplicações de bancos de dados manipulam dados temporais. Tais aplicações são tipicamente orientadas para um grande número de usuários, e, portanto, precisam ser dotados de interfaces que facilitam a interação do usuário com o sistema. Adicionalmente, informações temporais são também relevantes para modelar a interação do usuário.

Neste capítulo propomos uma modelagem de dados sobre a interação do usuário dentro de um sistema visual de consulta (SVC) [47], [49]. Mais especificamente, o histórico das interações de cada usuário é armazenado, acessado e dinamicamente mantido utilizando os mesmos mecanismos sobre os dados da aplicação, ou seja, a aplicação das primitivas gráficas temporais (TGPs).

Esta abordagem é um esforco para estabelecer claramente uma ligação entre ⟨**interação de usuário, modelagem**⟩ e ⟨**modelagem de dados, consulta**⟩, como uma tentativa de dar suporte à uma análise estruturada dos resultados que são apresentados em testes de usabilidade sobre SVCs.

O resto do capítulo discute como a história das interações do usuário pode ser acessada e dinamicamente mantida.

7.1 A História da Interação do Usuário

A história da interação do usuário pode ser modelada com base no trabalho descrito em [49], sob um modelo de usuário para um sistema visual de consulta adaptativo [22]. Tal modelo é utilizado para fornecer automaticamente ao usuário a representação visual e interação mais apropriada, de acordo com as suas necessidades e habilidades.

O modelo de usuário consiste de três componentes: a **classe estereótipo**, a **assinatura do usuário** e o **modelo de sistema**. A classe estereótipo denota o perfil do usuário, a assinatura do usuário corresponde ao histórico das interações do usuário e o modelo do sistema é orientado ao domínio de aplicação.

Considerando a **classe estereótipo**, nosso esquema de classificação do usuário é o proposto em [25], onde os usuários de banco de dados são profissionais ou não profissionais dependendo do treinamento que eles tiveram. Considerando apenas os usuários não profissionais, a Figura 7.1 ilustra a representação visual mais apropriada em relação a um grupo de usuários, de acordo com quatro características de usuário, relacionada à tarefa de consulta aos dados:

- *Frequência de interação*: identifica se o usuário é ocasional ou frequente;
- *Repetitividade da consulta*: identifica se o usuário faz consultas repetitivas, com padrões similares[1], ou consultas imprevisíveis;
- *Complexidade estrutural da consulta*: identifica se o usuário faz consultas complexas ou não, dependendo das operações utilizadas;
- *Familiaridade com o conteúdo do banco de dados*: identifica se o usuário é familiar ou não com o banco de dados.

Por exemplo, a figura 7.1 indica que a representação visual tabular é a mais apropriada para um usuário "frequente, repetitivo e novato", mas não é totalmente apropriado para um usuário não familiar, apesar dele ser mais apropriado que a representação diagramática.

[1] A similaridade pode envolver ou a estrutura da consulta ou os operadores utilizados na consulta ([22]).

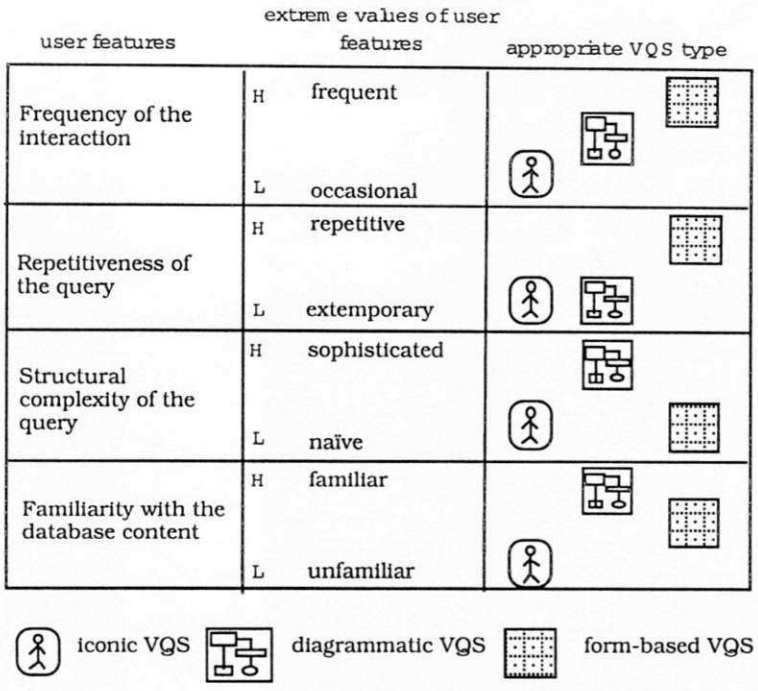

Figura 7.1: Características de Usuários Não Profissionais

Para cada característica, dois valores extremos *High (H)* e *Low (L)* são considerados. Portanto, dezesseis classes de usuário são geradas, da classe "ocasional, atemporal, novato, não familiar" (a primeira vez que o usuário interage com o sistema, ele/a pertence a esta classe, cujos valores das características são todos L) até a classe "frequente, repetitivo, sofisticado, familiar" (quando todas as características são H). Entre as dezesseis classe possíveis, quatro são improváveis, são as classes cujo valor da característica *frequência de interação* é L, enquanto o valor da característica *familiaridade com o conteúdo do banco de dados* é H.

Desde que o mesmo tipo de representação visual é o mais apropriado para algumas das doze classes de usuário, eles foram reduzidos a cinco estereótipos (ver [25], [22] para maiores detalhes) como mostrado na Figura 7.2. A partir destes estereótipos, modelos individuais de usuários de banco de dados podem ser gerados e mantidos, de forma a determinar a adequabilidade de uma representação visual.

Considerando a **assinatura do usuário**, o sistema protocola o conjunto de consultas que o usuário fórmula durante sua interação com um banco de dados.

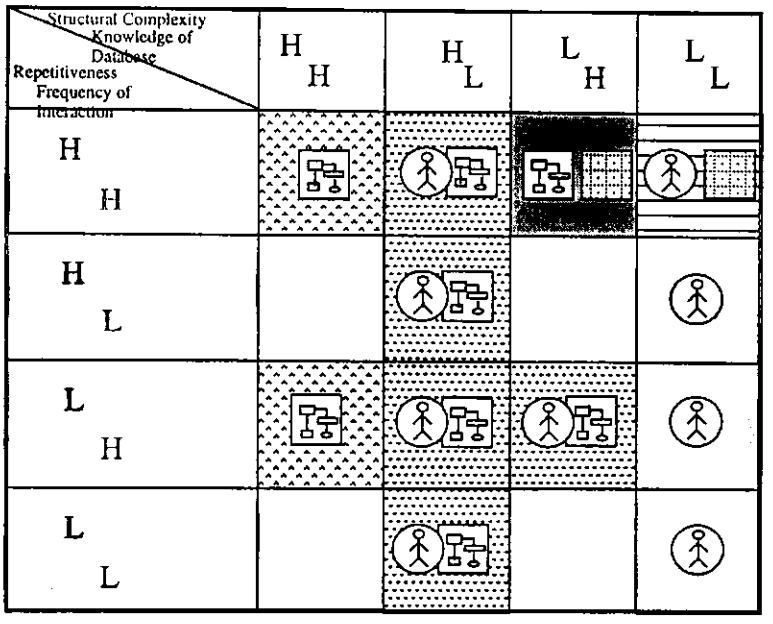

Figura 7.2: Classes de Usuários

Chamamos $Q = (Q_1, Q_2, \ldots, Q_n)$ a sequência de consultas formuladas nos tempos $t_1, t_2, \ldots, t_n$ respectivamente. A sequência Q determina o comportamento do usuário e de Q o sistema pode deduzir o quanto as consultas são complexas, o quanto elas são frequentemente formuladas, etc. Portanto, a partir da assinatura do usuário que os seus estereótipos podem ser alterados.

O terceiro componente do modelo de usuário, o **modelo de sistema**, é a representação do conhecimento que o usuário tem do conteúdo do banco de dados e sua estrutura, ou seja, a visão do banco de dados do usuário. A vantagem de se explorar o modelo de sistema se deve a duas razões: a) as consultas de um usuário na maioria da vezes são limitadas a uma pequena e específica parte de um banco de dados; b) em grandes bancos de dados, o custo de recuperação de uma informação é muito alto.

Neste caso, o sistema constrói um histórico de visões que deve ser analisado de tempos em tempos de forma a sugerir ao usuário a visão mais apropriada para as suas necessidades. Uma estrutura dinâmica foi proposta em [24] para representar o modelo de sistema.

7.1.1 Modelando a história da interação do usuário

A interação do usuário pode ser especificada em termos de três componentes: a **Sessão**, a qual identifica cada interação específica que acontece durante um certo intervalo de tempo; a **Consulta**, que contém a especificação de cada consulta executada durante uma certa sessão; e a **Modalidade**, a qual, para cada consulta, assinala o conjunto de modalidades (tabular, diagramática, icônica, etc) usado para formulá-la.

Em outras palavras, durante uma interação, o usuário executa uma sequência temporal de consultas, especificada por uma ou mais modalidades (cada uma dentro de um intervalo de tempo). O estado de cada consulta pode ser **Completado** ou **Interrompido**, indicando que a consulta foi completada com sucesso ou não.

Dado as suas características temporais, podemos representar o componente *US* como um *TGMDB*. Chamamos este *TGMDB* de *TGMDB Interação*. Figura 7.3 ilustra seu correspondente *Typed Graph*.

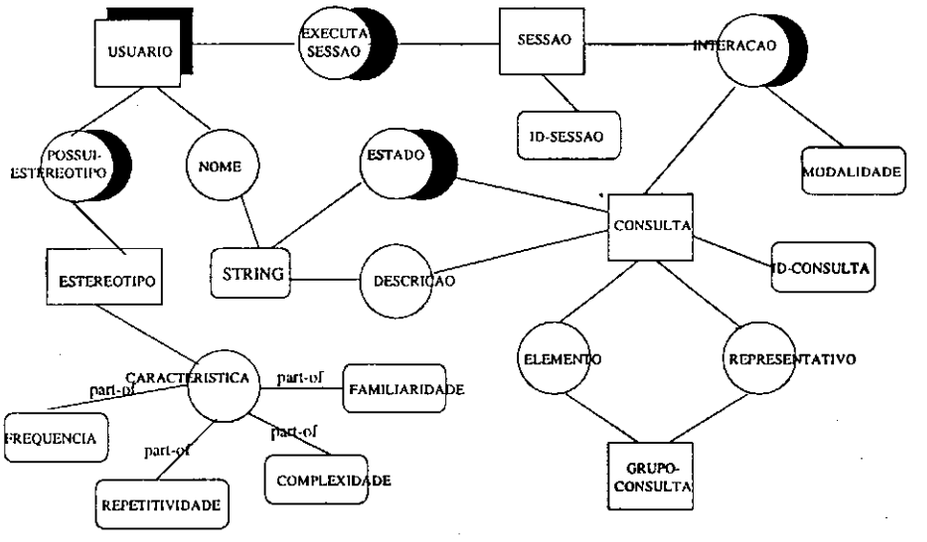

Figura 7.3: *Typed Graph* do TGMDB Interação

O nodo-papel temporal *Executa-Sessão* entre o nodo-classe *Usuário* e o nodo-classe *Sessão* representa o histórico de sessões de cada usuário. Um usuário executa ao menos uma sessão e ao máximo n sessões e uma sessão é executada somente por um único usuário.

O nodo-classe *Estereótipo* é uma agregação dos nodos-classe *Repetitividade*, *Frequência*, *Complexidade* e *Familiaridade* (cujas instâncias são os valores binários L ou H), onde existe o nodo-papel *Característica* entre a classe agregada e as classes componentes. Neste caso, um usuário "repetitivo, ocasional, sofisticado e não familiar" é representado por um identificador dentro da classe *Usuário*, por exemplo, o_1, a tupla $\langle Usurio : o_1, Esteretipo : s_1 \rangle$ no nodo-papel temporal *Possui-estereótipo* e a tupla ⟨ *Estereótipo*:s_1, *Repetitividade:H, Frequência:L, Complexidade:H, Familiaridade:L* ⟩ como instância do nodo-papel *Característica.*

O nodo-classe *Consulta* é associado ao nodo-classe *Grupo-Consulta* através dos nodos-papel *Elemento* e *Representativo.* Isto significa que, de acordo com [22], um conjunto de consultas $\{Q_1, Q_2, \ldots, Q_n\}$ com padrões similares são agrupados em grupos de consulta $\{C_1, C_2, \ldots, C_m\}$, com $m \leq n$, onde cada grupo de consulta C_k tem uma consulta representativa (ver o procedimento definido em [22], o qual identifica se uma consulta pertence a uma grupo particular de consultas).

O nodo-papel temporal *Estado* representa o histórico de estados (Completado ou Interrompido) de cada consulta específica que o usuário formula.

A interpretação do nodo-papel temporal *Interação* é formado de tuplas rotuladas, associando objetos de *Sessão*, *Consulta* e *Modalidade*, a um *lifespan* $ls = \{I_1, \ldots, I_n\}$, onde cada I_k é um intervalo correspondente à duração de uma modalidade (dentro de uma consulta e sessão específica).

Por exemplo, se o estado inicial de uma interação de usuário com um banco de dados é a tripla $\langle S_0, Q_0, M_0 \rangle$ no tempo t_0, e o usuário altera a modalidade M_0 para M_1 no tempo t_1, uma tupla $\langle Sesso : S_0, Consulta : Q_0, Modalidade : M_0, \langle t_0, t_1 \rangle \rangle$ é adicionada à interpretação de *Interação.* As Tabelas 7.1, 7.2 e 7.3 mostram uma representação tabular das histórias de *Interação*, *Sessão* e *Estado* respectivamente, com dados que se referem a dois diferentes usuários.

Adicionalmente, a consulta obtém um estado no instante final da última modalidade sobre a qual a consulta foi expressa. Dentro do *lifespan* associado com as tuplas $\langle S, Q, M \rangle$ na interpretação de *Interação*, ao final de cada intervalo de tempo pode serassociado com um estado.

Sessão	Consulta	Modalidade	Intervalo
S_1	Q_1	M_3	$\langle 0,2\rangle \cup \langle 5,6\rangle$
S_1	Q_1	M_1	$\langle 3,4\rangle$
S_1	Q_2	M_1	$\langle 7,9\rangle$
S_1	Q_2	M_3	$\langle 10,13\rangle$
S_1	Q_3	M_1	$\langle 14,15\rangle$
S_1	Q_3	M_4	$\langle 16,17\rangle \cup \langle 20,21\rangle$
S_1	Q_3	M_3	$\langle 22,25\rangle$
S_2	Q_1	M_1	$\langle 1,5\rangle$
S_2	Q_1	M_3	$\langle 6,7\rangle$
S_2	Q_1	M_2	$\langle 8,9\rangle$
S_2	Q_2	M_3	$\langle 10,14\rangle$
S_2	Q_2	M_2	$\langle 15,16\rangle$
S_2	Q_2	M_1	$\langle 17,18\rangle$

Tabela 7.1: Interpretação do Nodo-Papel Temporal *Interação*

Por exemplo, na Tabela 7.1, a tupla $\langle S_1, Q_1, M_2\rangle$ contém dois intervalos $\langle 0,2\rangle$, $\langle 5,6\rangle$, mas apenas o segundo intervalo é registrado com o estado **Completado** na Tabela 7.3. Note que nesta tabela o instante de um estado é representado como um intervalo de tempo com o ínicio e fim do intervalo assumindo o mesmo valor (desde que um elemento temporal é representado por um conjunto de intervalos de tempo em nosso formalismo).

Usuário	Sessão	Intervalo
U_1	S_1	$\langle 0,17\rangle \cup \langle 20,25\rangle$
U_2	S_2	$\langle 1,18\rangle$

Tabela 7.2: Interpretação do Nodo-Papel Temporal *Executa-Sessão*

Consulta	Status	Intervalo
Q_1	*Completed*	$\langle 6,6\rangle \cup \langle 9,9\rangle$
Q_1	*Interrupted*	$\langle 5,5\rangle$
Q_2	*Completed*	$\langle 13,13\rangle \cup \langle 18,18\rangle$
Q_3	*Interrupted*	$\langle 15,15\rangle$
Q_3	*Completed*	$\langle 25,25\rangle$

Tabela 7.3: Interpretação do Nodo-Papel Temporal *Estado*

7.1.2 Acessando a história da interação do usuário

Através do TGP *seleção do rótulo do nodo*, podemos derivar o *lifespan* de qualquer sessão (consulta), através da identificação dos *lifespans* das consultas (modalidades) associadas, isto é, o usuário pode sucessivamente selecionar o rótulo dos nodos-classe *Modalidade* e *Consulta*. Por exemplo, suponhamos que o usuário é interessado em saber: *"Para cada usuário, qual foi o tempo médio para construir uma consulta nos últimos 2 anos?"*. Primeiramente, ele acessa o esquema conceitual que representa a história da interação do usuário, como mostrado na Figura 7.4, onde ele selecionou a classe *Usuário* como a classe alvo da consulta.

A Figura 7.5 mostra os efeitos do ícone *When?* aplicado ao relacionamento *Interação*. No exemplo, o usuário seleciona a opção *period*. Depois que o usuário escolheu a granularidade *ano* e especificou o período desejado, ele seleciona o predicado *during* do *slider* que contém nove predicados. O usuário também ativa a opção *temporal aggregation?* e seleciona a classe *Modalidade* e *Sessão* dentro da lista de classes relacionadas por *Interação*. De forma a recuperar o tempo médio para especificar uma consulta, o usuário seleciona o botão *duration* e o item *avg* dentro do menu de funções agregadas.

Em suma, através da utilização dos TGPs aplicados sobre a informação temporal gerada durante a interação do usuário com o sistema, diversas consultas temporais podem ser especificadas tais como: *Qual o tempo médio para completar uma consulta para cada modalidade?*, *Desde quando João Silva é um usuário repetitivo?*, ou *Qual a história das sessões de cada usuário durante 1997?*.

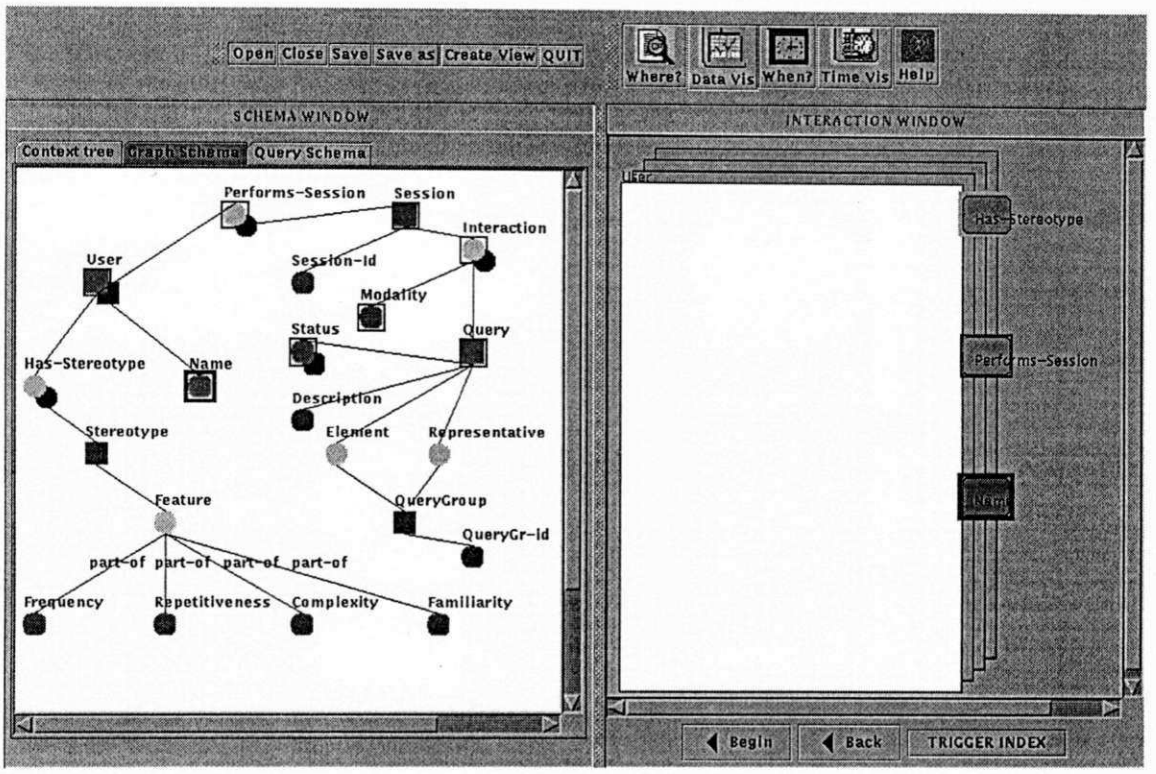

Figura 7.4: Os nodos selecionados e visualizados da consulta

7.1.3 Mantendo o estereótipo do usuário

O estereótipo do usuário pode mudar desde que os valores L e H de cada característica pode mudar a qualquer momento. Portanto, para uma atualização dinâmica do componente CS (estereótipo do usuário), as características do usuário são computadas a partir do componente US (assinatura do usuário).

A seguir, descrevemos como o sistema identifica automaticamente os valores das características do usuário, através dos TGPs.

Considerando a característica **frequência de interação**, ele é computado através da comparação do número de consultas executadas em um dado intervalo de tempo I com um *threshold*, de forma a determinar a baixa ou alta frequência da interação do usuário.

De forma a recuperar o número de consultas executadas por cada usuário durante um periodo de tempo I, o procedimento é como segue: seleção do nodo sombreado do nodo-papel *Interação* (gerando o nodo-papel *Interação-History*).

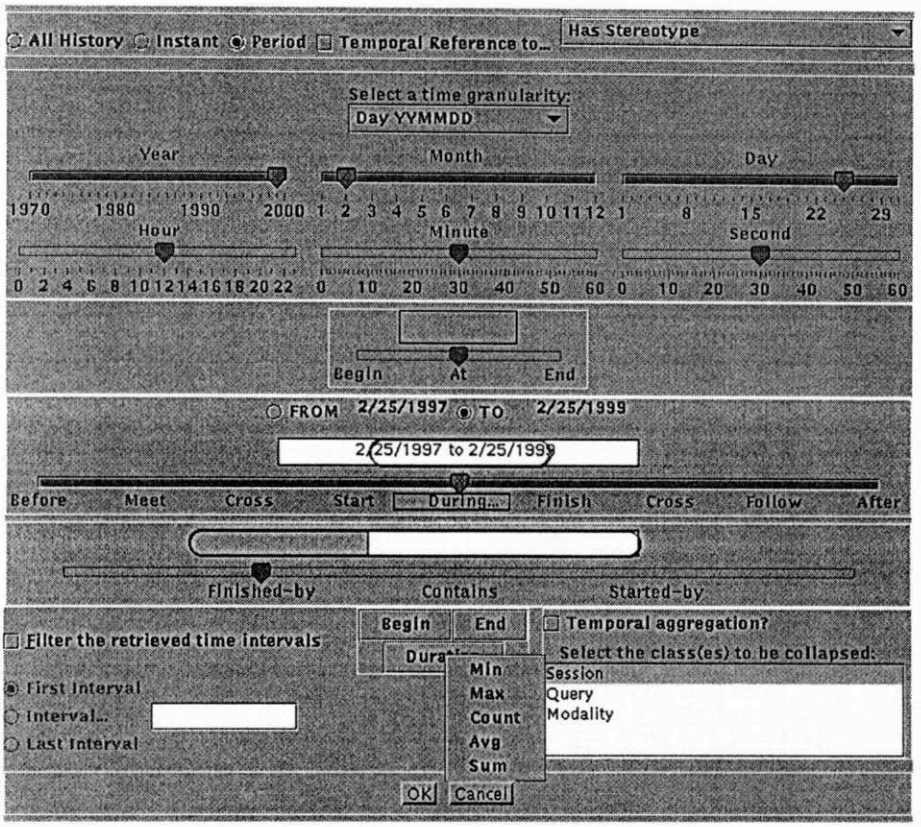

Figura 7.5: Uma condição temporal sobre *Interação*

Em seguida, se aplica a seleção do rótulo dos nodos-classe *Modalidade* e *Sessão* e mudanca do rótulo *all history* para *During I* no arco existente entre o nodo-papel *Interação-History* e o nodo-classe *Time-interval.* De forma a recuperar o número de consultas, uma função agregada *count* é aplicada ao nodo-papel *Interação-History.*

Os três componentes *Consulta, Grupo-Consulta* e *Representativo* são usados para computar as duas características **repetitividade da consulta** e **complexidade estrutural da consulta.**

De acordo com [22], um usuário é considerado um repetitivo se a seguinte entropia é computada: $H(G') = \sum_{i=1}^{m} p_i log(1/p_i)$, e o resultado é menor que um *threshold*, onde $G' = \{G'_1, G'_2, \ldots, G'_m\}$ é o conjunto de consultas representativas e para cada G'_i a probabilidade $p_i = n_i/n$ é computada (n_i indica quantas consultas estão na mesma classe de G'_i). Vale a pena ressaltar que esta entropia é sempre calculada na última sessão que o usuário executa. Este procedimento é mostrado na Figura 7.6

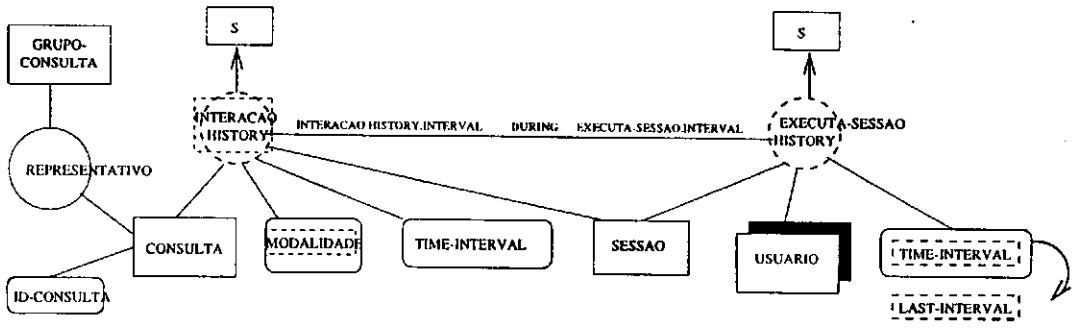

Figura 7.6: Computando *repetitividade e complexidade estrutural da consulta*

Portanto as instâncias da classe *Consulta* são recuperadas de forma a analisar a relação com o nodo-classe *Grupo-Consulta* e *Representativo* para computar a entropia descrita acima.

O mesmo procedimento é aplicado para computar a característica *complexidade estrutural da consulta* utilizando outra fórmula também contendo a variável p_i (uma discussão detalhada sobre como computar a fórmula que determina a complexidade estrutural da consulta pode ser vista em [22]).

Como já foi discutido anteriormente, as dezesseis combinações da característica do usuário determinam cinco estereótipos, como mostrado na Figura 7.2. De forma a manter dinamicamente a dependência entre os cinco estereótipos e as características do usuário, atualizamos o nodo-papel *Possui-estereótipo* correspondente a uma mudança em um dos valores das características, ativando um evento temporal.

Por exemplo, quando um usuário o_1 muda sua característica de "ocasional" para "frequente" no instante t, se aplica uma seleção do nodo sombreado do nodo-papel temporal *Possui-estereótipo* com parâmetro *finishes(now)*, de forma a identificar a tupla $\langle Usurio : o_1, Esteretipo : s_i, \langle begin, now \rangle \rangle$ correspondente ao estereótipo corrente s_i (a classe s_i é uma agregação das quatro classes componentes, onde a classe *Frequência* assume o valor L), e alterar o valor *now* no valor t (contendo o tempo presente) dentro desta tupla.

Como consequência, o sistema adiciona uma nova tupla $\langle Usurio : o_1, Esteretipo : s_j, \langle t, now \rangle \rangle$ (a classe s_j é uma agregação das quatro classes componentes, onde a classe *Frequência* tem o valor H).

7.2 Conclusão deste capítulo

Neste capítulo foi proposto uma forma de dar suporte à avaliação de SVCs, mostrando como modelar, acessar e manter dinamicamete o histórico de interações do usuário dentro do próprio SVC. Realmente, os dados da aplicação e os dados sobre a interação do usuário são uniformemente representados, possibilitando que eles possam ser acessados através dos TGPs.

Para finalizar esta seção, descrevemos algumas considerações descritas por *Silva & Catarci*, em [47], o qual descreve uma modelagem visual de dados temporais em experimentos sobre SVCs.

Diante de uma variedade de interfaces para bancos de dados, é fundamental que os mesmas fossem eficientemente **avaliadas**. A maioria dos experimentos utilizados em interfaces para bancos de dados são comparativos entre os diferentes modos de acesso à informação: interação textual, linguagem natural e interação visual: [8], [114], [100].

Outros experimentos foram executados, de forma a explorar a eficácia de SVCs que fornecem diferentes representações visuais e métodos de acesso aos diversos tipos de usuário. Os resultados experimentais indicam que a adequabilidade de uma representação visual e método de acesso é estritamente relacionada ao nível de habilidade do usuário e o tipo da consulta [79], [92], [6].

Contudo, existem ainda poucos estudos empíricos no teste e validação de interfaces para banco de dados. Uma das principais razões desse fato é o processo de conduzir experimentos com usuário reais. Realmente, este processo envolve muitas atividades, tais como planejar o ambiente em que será realizado o experimento, a seleção de usuários, o planejamento das tarefas, as medidas de desempenho que serão utilizadas, análise estatística dos resultados, etc.

Dentro deste contexto, a idéia de fornecer **uma estrutura uniforme** que possa ser incorporada dentro de qualquer SVC é uma abordagem útil para detectar detectar erros que possam ocorrer nas fases iniciais do desenvolvimento daquele SVC. A escolha dos componentes principais desta estrutura é baseada na literatura existente sobre experimentos de usabilidade em SVCs.

Isto foi comprovado em [47], onde se considerou os dados temporais que são relevantes em experimentos de usabilidade sobre SVCs, utilizando o modelo TGM, e como os TGPs podem ser utilizados para acessar tais dados, de forma a avaliar a facilidade de uso de um SVC. Em particular, em [47] foram analisados três experimentos, brevemente descritos a seguir.

O experimento relatado em [101] avalia o uso do sistema hipertexto `Hyperties`, um sistema interativo de catálogos utilizado em três museus. O objetivo do experimento é fazer uma análise se os usuários exploram melhor a característica hipertexto (movendo de um artigo a outro), em oposição ao acesso por índices.

O experimento relatado em [6] compara duas linguagens visuais de consulta: `QBD*` *(Query by Diagram)* [5], que adota diagramas como formalismo visual, e `QBI` *(Query by Icon)* [81], que adota ícones como formalismo visual. Os resultados do experimento suportam a hipótese de que a eficácia das duas linguagens varia dependendo de duas variáveis: habilidade do usuário e categoria de consulta.

O experimento relatado em [2] compara três diferentes SVCs: uma interface que utiliza a abordagem de consulta dinâmica, uma segunda interface fornecendo uma saída com visualização dos dados, mas usando a modalidade tabular como especificação da consulta e uma terceira interface também usando a modalidade tabular na consulta e fornecendo uma saída textual. O resultado do experimento suporta a hipótese de que os usuários especificam consultas significantemente mais rápido, usando a consulta dinâmica, comparado às outras duas interfaces.

Portanto, a maioria destes experimentos consideram entre as variáveis essenciais, as habilidades do usuário, as classes das consultas, modalidades de interação e métodos de acesso dentro de tarefas específicas.

Vale ressaltar que a disponibilidade de dados temporais permite uma análise mais aprofundada da usabilidade do sistema. Em particular é fundamental monitorar a evolução da interação do usuário e notar como esta evolução afeta a facilidade de uso do sistema. Além disso, medidas temporais caracterizam bem o desempenho do usuário durante a interação com o sistema.

Capítulo 8

Conclusão

Neste trabalho descrevemos uma linguagem visual e formal de consulta a bancos de dados históricos, baseado em uma interface diagramática e icônica na representação visual e interação dos dados, denominada de ambiente TVQE.

O usuário percebe o banco de dados através de uma metáfora de agenda, a qual compreende conceitos (classes) e associações entre conceitos (propriedades da classe). Consultas são formuladas através da escolha de um conceito alvo, e a partir deste ponto, o usuário determina as propriedades (temporais ou não) dos dados a serem recuperados. Esta abordagem facilita ao usuário expressar sua consulta, uma vez que ele não precisa aprender uma sintaxe de uma linguagem textual e nem conhecer a estrutura interna do banco de dados.

Considerando os aspectos formais descritos neste trabalho, o aspecto fundamental não está na nova definição de um modelo de dados temporal e sua correspondente linguagem de consulta. O aspecto mais importante é a representação visual do modelo e de seus correspondentes operadores formais, para a construção de consultas a bancos de dados históricos.

Este capítulo está organizado nas seguintes seções:

1. Potenciais, limitações e dificuldades encontrados no trabalho;

2. Direções futuras de pesquisa.

8.1 Potenciais, Limitações e Dificuldades

Consideramos as seguintes características como **potenciais** do trabalho:

- A interface TVQE é um ambiente genérico, não foi projetada para um domínio de aplicação específico.
- Às ações do usuário corresponde um conjunto bem definido de primitivas formais.
- A visualização estrutural dos dados utiliza uma metáfora de agenda para tentar aumentar o envolvimento do usuário com o sistema, e diminuir a distância entre a visão que o usuário tem do domínio dos dados e a representação deste domínio no sistema, desde que a metáfora representa uma analogia a um conceito conhecido no mundo real (acesso aos índices de uma agenda).
- O usuário especifica (através da árvore de contextos) o que é para ser visualizado, ou um banco de dados total, ou apenas uma parte dele.
- Em uma consulta, perspectivas diferentes sobre um mesmo dado são disponíveis a partir do conceito da escolha de diferentes classes alvo.
- O suporte a um editor de domínios sobre um atributo, removendo qualquer ambiguidade que pode ocorrer na interpretação pessoal que o usuário possui deste domínio.
- A arquitetura TVQE suporta reusabilidade e portabilidade, desde que ele pode ser incorporado no topo de um SGBD relacional que suporte *ODBC* (*Open Database Connectivity*.

Por outro lado, consideramos as seguintes características como **limitações** do trabalho:

- O sistema TVQE é um ambiente projetado somente para consultas a banco de dados, não suportando alterações no banco de dados.

- Mesmo com o crescimento de manipulação do conteúdo na interação com o banco de dados, ainda se utiliza a manipulação da estrutura na especificação de uma consulta, dentro do ambiente TVQE.

- Não foi feito uma avaliação empírica da usabilidade da interface. Foi realizado somente uma avaliação subjetiva sobre o projeto inicial da interface, brevemente descrita no capítulo 5.

- Não foi ainda avaliado o poder expressivo das primitivas gráficas temporais.

Considerando as **dificuldades** encontradas no trabalho, em linhas gerais, é dificil projetar uma interface generalizada e que seja ao mesmo tempo eficiente, interessante e fácil de usar! Uma das maiores dificuldades no início do projeto da interface foi considerar ou não o mecanismo de consulta dinâmica como a única estratégia de interação na especificação de uma consulta.

Consulta dinâmica acarretaria outras dificuldades, entre as quais: a não possibilidade de especificar consultas com múltiplos relacionamentos simultaneamente, e a Utilização de algoritmos complexos para bancos de dados persistentes.

Considerando as dificuldades acima, foi decidido utilizar consulta dinâmica somente na visualização dos dados (que corresponde a trabalhos futuros). Contudo, outras dificuldades surgiram na decisão de projetar a manipulação da estrutura na interface, entre as quais:

- Inicialmente, os relacionamentos eram visualmente representados como arcos. Quando ocorria a seleção de índices relacionados, o arco que os associava era também selecionado, o que gerava um problema quando existia mais de um relacionamento associando as mesmas classes.

- Encontrar uma propriedade visual compreensível (cor, forma, tamanho ou posição) que distinguisse classes, relacionamentos e atributos, e que distinguisse relacionamentos hierárquicos (ex. é-um) dos não hierárquicos.

- Modelar explicitamente classes como agregado de outras classes, geralmente encontrado em esquemas mais complexos.

- Como incluir restrições de cardinalidade no esquema gráfico ou na agenda, sem considerar a notação amplamente utilizada em modelos semânticos (ex. [1,*]), que é mais orientada à visão do projetista de banco de dados?

  Uma primeira idéia foi representar visualmente relacionamentos e atributos multivalorados com um sombreado nos nodos e nos índices, mas isto gerou conflito com a representação visual de relacionamentos e atributos temporais.

  Portanto, deve-se pensar em como representar melhor os relacionamentos e atributos temporais tanto na representação diagramática quanto na agenda gráfica.

- Escolher entre projetar um ícone *When* na especificação de uma condição temporal, ou projetar separadamente dois ícones *Instant* e *Period*, na especificação de uma condição instantânea e periódica respectivamente.

  A escolha da primeira opção na interface atual se deve ao fato da mesma ser melhor adequada a *applets*, desde que não são recomendados diálogos modais em *applets*. Com a segunda opção, seriam necessario dois ou mais diálogos modais na especificação de uma condição temporal.

8.2 Direções Futuras

Um dos trabalhos mais importantes a ser feito será a criação da **Janela de Visualização dos Dados**, utilizando o mecanismo de consulta dinâmica na visualização e manipulação dos dados. Uma proposta de projeto lógico é descrito na subseção 8.2.1. Outro trabalho futuro seria a inclusão da história da interação do usuário dentro do ambiente TVQE, como foi brevemente discutida no capítulo anterior.

Considerando os aspectos relacionados à interface TVQE, sugerimos outros trabalhos futuros, entre os quais:

- Incluir o mecanismo de *undo* nas funcionalidades da interface.

- Executar a aplicação como um *Applet* em um *Web browser*.

- Explorar a interação visual na especificação de consultas mais complexas, que compreendam duas ou mais visões dentro de uma mesma consulta, e que possa incluir os operadores de conjunto, união (consultas disjuntivas), intersecção (consultas conjuntivas) e diferença.

- Atualmente, o editor de domínios é restrito para alguns tipos de dados. É necessário estender este editor para suportar outros domínios.

- Eliminar o painel *When*, que serve para relacionamentos e atributos temporais, e substituí-lo por um editor de domínio (com granularidade temporal específica) daquele atributo temporal.

- Incluir imagens nos índices da agenda, cada imagem representando a semântica da classe, relacionamento ou um atributo.

- Realizar a migração do ambiente dentro da arquitetura cliente-servidor para a arquitetura em três níveis (*3-tier*) (brevemente descrita no capítulo 6).

Em seguida será desenvolvido o mapeamento dos subgrafos de interesse em comandos da linguagem SQL, a partir da criação da estrutura TGM x relação[1] (ver arquitetura de implementação no capítulo 6).

Considerando os aspectos relacionados ao modelo TGM e as primitivas gráficas temporais, algumas extensões devem ser feitas, entre as quais:

- Incluir a modelagem de granularidades temporais dentro do modelo TGM.

- Assim como foram feitos mapeamentos dos bancos de dados expressos nos diversos modelos de dados relacionais, semânticos e orientado a objetos para os GMDBs, seguiremos a abordagem de [21] para fazer mapeamentos do modelo TGM para os diversos modelos de dados relacionais e orientados a objeto que possuam aspectos temporais.

- Relacionar as primitivas TGPs com um álgebra temporal existente [104].

[1]Ela faz parte de uma dissertação de mestrado no departamento de sistemas e computação da Universidade Federal da Paraíba.

8.2.1 Resultado da consulta no TVQE

Como foi descrito anteriormente, toda a especificação da consulta é feita sobre a parte intensional do banco de dados, ou seja, o esquema conceitual. O sistema fornece uma metáfora de agenda como uma interface entre o modelo de dados e o usuário, de forma a facilitar a especificação de sua consulta. Contudo, o usuário pode ter conhecimento do domínio que nem sempre é refletido pela metáfora. Desta forma, o usuário quer verificar a informação que ele extraiu do banco de dados, passando ao seu nível extensional. Isto pode resultar em uma mudança de metáfora ou representação visual dos dados.

Adicionalmente, algumas funcionalidades devem ser definidas de forma a facilitar a compreensão visual do conteúdo do banco de dados, principalmente quando este conteúdo compreende milhares de dados. Os dados serão visualizados de duas formas: forma tabular dos dados ou métodos de visualização adequados em apresentar uma informação quantitativa (*bar chart, plot chart, line chart*, etc).

Considerando a representação tabular, os dados que compõem o resultado da consulta podem apresentar uma estrutura aninhada. Foi observado em [95] que dois atributos de classes selecionadas para consulta, e que são conectadas por um relacionamento com cardinalidade *n-m*, serão apresentadas em uma forma aninhada. Outra possibilidade de uma estrutura aninhada é a informação temporal de atributos que foram selecionados para a consulta, caso a informação temporal de um atributo seja um agregado de informações temporais do outro atributo.

Considerando a visualização dos dados em uma forma multidimensional, *Ahlberg* identificou em [1] as seguintes vantagens em se utilizar este tipo de visualização:

- Permite uma visualização de mais de dois atributos simultaneamente;
- Melhor utilização do espaço de tela disponível;
- Fornece visualizações mais naturais de domínios de aplicação que são inerentemente tridimensionais (SIGs, CAD, imagens médicas, etc).

Adicionalmente, esta forma de visualização se beneficia das propriedades de percepção visual inerentes aos seres humanos.

Um mecanismo de interação poderoso sobre a visualização desta informação é a consulta dinâmica. As vantagens de se utilizar consulta dinâmica em interfaces para bancos de dados foi comprovada em [2], [1], [102], [100].

Considerando o domínio temporal, mecanismos de visualisação bidimensional (2D) devem ser definidos de forma a identificar a evolução temporal dos objetos de uma forma mais intuitiva. Como o domínio do tempo é ordinal, a visualisação 2D se restringe a:

- Apresentação somente de um objeto de uma classe temporal (mostrando a sua evolução temporal);
- Apresentação de um objeto de um relacionamento temporal (mostrando a evolução temporal da propriedade de um objeto específico).

Obviamente isto não pode ser aplicado para visualizar a evolução temporal de todas as propriedades (ou classes) de todos os objetos, desde que isto corresponde a uma relação ternária. Na representação da relação ternária, outros mecanismos de visualisação, tais como a visualização tridimensional (3D), devem ser analisados.

Desde que consideramos os objetos visualizados como pontos em uma dimensão espacial *starfieds* [1], nossa proposta é apresentar um versão temporal de *starfields*, denominada visualização ***spaghetti***.

O princípio geral da nossa proposta é mapear o tempo sobre uma representação espacial 2D ou 3D. Na representação 3D, uma coordenada será o tempo e as outras serão parâmetros que representam atributos que forma visualizados na especificação da consulta. Cada "corte" (*slice*) a um tempo particular é uma superfície representando *starfields*. Os pontos correspondentes em cada "corte" do tempo são conectados por linhas contínuas. Através da coloração dos pontos baseados nos valores dos atributos, o efeito da visualização será um conjunto de espaguetes coloridos.

O problema desta visualização é a densidade das informações. Ou seja, se um fio de espaguete contiver muitos pontos, será difícil distingui-los. Desta forma, uma idéia seria reduzir a densidade da visualização 3D. Reduzindo o número de pontos, o espaguete será menos denso, então se distinguirá mais facilmente os fios do espaguete.

O problema agora é como reduzir o número de fios do espaguete. Existem algumas opções:

- Utilização de um critério de seleção através de *sliders* dinâmicos de forma a reduzir o número de fios em um espaguete. Para isto, os valores dos critérios de seleção devem ser aproximados.
- Usar técnicas estatísticas ou de agrupamento (*clusters*) de forma a criar pontos representativos a serem visualizados.

Isto seria um ponto de partida para a visualização dos dados temporais.

Para finalizar esta tese, mostramos graficamente todo o processo de desenvolvimento desta tese. Note que o tempo deste processo não foi linear e sim ramificado, como mostra a Figura 8.1 (a sigla TOM (*Temporal Object Model*) representa o modelo de dados inicial e as siglas CD e MU significam a estratégia de consulta dinâmica e modelo de usuário, respectivamente).

A figura mais abaixo ilustra como seria este processo de desenvolvimento se o tempo fosse linear.

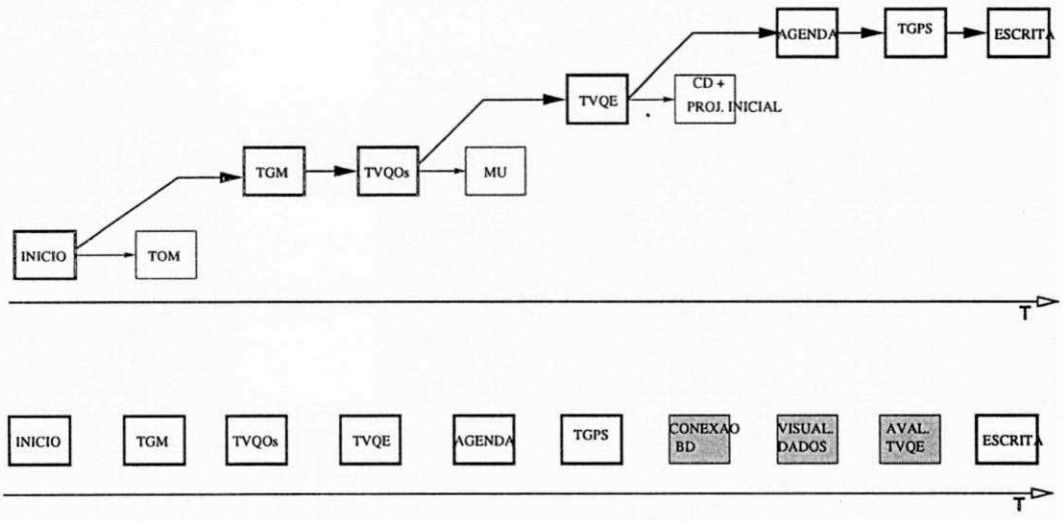

Figura 8.1: Processo de Desenvolvimento da Tese

Portanto, a realização das atividades em destaque (conexão ao banco de dados, visualização dos dados e avaliação da interface) correspondem a trabalhos futuros de pesquisa.

Bibliografia

[1] A. Ahlberg and B. Shneiderman. Visual information seeking: Tight couping of dynamic query filters with starfield displays. In *Proc. of CHI94, Boston*, pages 313–317, 1994.

[2] C. Ahlberg, C. Williamson, and B. Shneiderman. Dynamic queries for information exploration: an implementation and evaluation. In *ACM CHI 92, Monterey CA*, pages 619–626, May 1992.

[3] S. Alhir. *UML in a Nutshell. A Desktop Quick Reference.* O'Reilly Publisher & Associates Inc., Sebastopol, CA, 1998.

[4] J.F. Allen. Maintaining knowledge about temporal interval. *Communications of ACM*, 26(1):832–843, 1983.

[5] M. Angelaccio, T. Catarci, and G. Santucci. QBD*: A graphical query language with recursion. *IEEE Transaction on Software Engineering*, 16(10):1150–1163, 1990.

[6] A.N. Badre, T. Catarci, and G. Santucci. Comparative ease of use of a diagrammatic versus an iconic query language. In Kennedy J., editor, *Third Int. Workshop on User Interfaces to Database Systems, Edinburgh*, 1996.

[7] C. Batini, S. Ceri, and S. Navathe. *Conceptual Database Design. An Entity-Relationship Approach.* Redwood City, CA, 1992.

[8] J.E. Bell. The experiences of new users of a natural language interface to a relational database in a controlled setting. In Cooper R., editor, *Interface to Database*

Systems Glasgow 1992, Workshop in Computing, Springer-Verlag, London, pages 433–454, 1993.

[9] F. Benzi, D. Maio, and S. Rizzi. Visionary: A visual query language based on the user viewpoint approach. In Kennedy J., editor, *Third International Wokshop on User Interfaces to Database Systems - Edinburgh*, 1996.

[10] F. Benzi, D. Maio, and S. Rizzi. Vistool: A visual tool for querying relational databases. In T. Catarci et al., editor, *Proc. of the Working Conference on Advanced Visual Interfaces (AVI'98)*, pages 258–260. ACM Press, May 1998.

[11] J. Boyle, J.E. Fothergill, and P.M.D. Gray. Design of a 3d user interface to a database. In P. Sawier, editor, *Interfaces to Database Systems, Lancaster 1994*, Workshop in Computing, Springer-Verlag, London, pages 127–142, 1995.

[12] D. Bryce and R. Hull. Snap: A graphics-based schema manager. *Proc. of the IEEE Conference on Data Engineering, Los Angeles USA*, pages 151–164, 1986.

[13] L.M. Burns et al. Aerial: Ad hoc entity-relationship investigation and learning. In *Proc. of the IEEE Int, Conf. on Systems, Man, and Cybernetics*, 1991.

[14] L. Cabibblo and R. Torlone. From a procedural to a visual query language for olap. In *10th Int. Conf. on Scientific and Statistical Database Management (SSDBM'98), Capri, Italy*, 1998.

[15] F. Capobianco, M. Mosconi, and L. Pagnin. Lookin for convenient alternatives to forms for querying remote databases on the web: A new iconic interface for progressive queries. In T. Catarci et al., editor, *Proc. of the Workshop on Advanced Visual Interfaces (AVI96), Gubbio Italy*, pages 119–124, May 1996.

[16] R. Carapuça, A. Serrano, and J. Farinha. Automatic derivation of graphical human-machine interfaces for databases. In London Workshop in Computing, Springer-Verlag, editor, *Interface to Database Systems, Glasgow 1992*, pages 176–192, 1993.

[17] H.J. Carrillo et al. Articulating a metaphor through user-centered design. In I. Katz et al., editor, *Proc. of the Human Factors Computing Systems Conference (CHI'95), Denver, Colorado*, pages 566–572. ACM / SIGCHI, May 1995.

[18] T. Catarci and G. Santucci. A visual approach to multilinear recursion. In Sawier P., editor, *Interface to Database Systems, Lancaster 1994*, Workshop in Computing, Springer-Verlag, London, pages 65–83, 1995.

[19] T. Catarci, G. Santucci, and M. Angelaccio. Fundamental graphical primitives for visual query languages. *Information Systems*, 18(2):75–98, 1993.

[20] T. Catarci, G. Santucci, and J. Cardiff. Knowledge-based schema integration in a heterogeneus environment. In *Proc. of the 2nd NGTIS Conference, Naharia, Israel*, 1995.

[21] T. Catarci, G. Santucci, and J. Cardiff. Graphical interaction with heterogeneus databases. *VLDB Journal*, 6(2):97–120, 1997.

[22] T. Catarci et al. A graph-based framework for multiparadigmatic visual access to databases. *IEEE Transactions Knowledge on Data Engineering*, 8(3):455–475, 1996.

[23] T. Catarci et al. Visual query systems: Analysis and comparison. *Journal of Visual Languages and Computing*, (8):215–260, 1997.

[24] S.K. Chang. A visual language compiler for information retrieval by visual reasoning. *IEEE Transactions on Software Engineering*, pages 1136–1149, 1990. Special Section on Visual Programming.

[25] S.K. Chang, M.F. Costabile, and S. Levialdi. A framework for intelligent visual interface design for database systems. In Cooper R., editor, *Interface to Database Systems, Glasgow 1992*, Workshop in Computing, Springer-Verlag, London, pages 377–391, 1993.

[26] P.P. Chen. The entity-relationship model toward a unified view of data. *ACM Transaction on Database Systems*, 1(1):9–36, 1976.

[27] J. Chomicki. Temporal query languages: a survey. In *Proc. 1st Int. Conference on Temporal Logic*, Springer-Verlag, pages 506–534, 1994.

[28] E.F. Codd. *Relational Completeness of Database Sub-Languages, Database Systems*, pages 65–98. Prentice-Hall, Englewwod Cliffs, 1972.

[29] M.P. Consen and A.O. Mendelzon. Hy+: A hygraph-based query and visualization system. In ACM Press, editor, *Proc. SIGMOD 93*, pages 511–516, 1993.

[30] R. Cooper. The interaction between dbms and user interface research. In Cooper R., editor, *Interface to Database Systems, Glasgow 1992*, Series Workshop in Computing, Springer-Verlag, London, pages 1–5, 1993.

[31] J.M. Corridoni, A. Bimbo, and S. Magistris. Image query by semantic color content. In T. Catarci et al., editor, *Proc. of the Workshop on Advanced Visual Interfaces (AVI96), Gubbio, Italy*, pages 213–222, 1996.

[32] I.F. Cruz. Doodle: A visual language for object-oriented databases. In *ACM SIGMOD Conf. on Management of Data*, 1992.

[33] I.F. Cruz, A.O. Mendelzon, and P.T. Wood. G+: Recursive queries without recursion. In *Proc. of the 2nd Int. Conference on Expert Database Systems*, pages 355–368, 1988.

[34] B. Czedjo et al. A graphical data manipulation language for an extended entity-relationship model. *IEEE Computer*, 23(3), 1990.

[35] B. Czejdo et al. A visual query language for an e-r data model. In *Proc. of the Int. Workshop on Visual Languages, Roma, Italy*, 1989.

[36] Y. Dennebouy et al. Super:visual interfaces for object + relationship data models. *Journal of Visual Languages and Computing*, 6(1):73–99, 1995.

[37] A. Dix and A. Patrick. Query by browsing. In Sawier P., editor, *Interfaces to Database Systems, Lancaster 1994*, Workshop in Computing, Springer-Verlag, London, pages 236–248, 1995.

[38] K. Doan et al. A multi-paradigm query interface to an object-oriented database. In *INTERACT 95*, 1995.

[39] S.W. Draper and K.W. Waite. Iconographer as a visual programming system. In Diaper D. and Hammond N., editors, *HCI 91 People and Computers VI: Usability Now!*, Cambridge University Press, pages 171–185, 1991.

[40] N. Edelweiss and J. Oliveira. Modelagem de aspectos temporais de sistemas de informação. In *IX Escola de Computação, Recife*, 1994.

[41] G.P. Ellis, G.E. Finlay, and A.S. Pollitt. Hibrowse for hotels: Bridging the gap between user and system views of a database. In Sawier P., editor, *Interface to Database Systems, Lancaster 1994*, Workshop in Computing, Springer-Verlag, London, pages 46–92, 1995.

[42] L. Elmasri and J.A. Larson. A graphical query facility for er databases. In *Proc. of the 4th Int. Conf. on Entity-Relationship Approach, Chicago USA*, pages 236–245, 1985.

[43] R. Elmasri et al. *Temporal Databases. A Temporal Model and Query Language for EER Databases*, pages 212–229. Benjamin/Cummings. Redwood City CA, 1993.

[44] D.W. Embley. Nfql: The natural forms query language. *ACM Transactions on Database Systems*, 14(2):168–211, 1989.

[45] T. Erickson. The design and long-term use of a personal electronic notebook: A reflective analysis. In M. Tauber et al., editor, *Proc. of the Human Factors Computing Systems Conference (CHI'96), Vancouver, Canada*, pages 11–18. ACM / SIGCHI, 1996.

[46] S. Fernandes Silva. Uma análise da evoluçao das interfaces para banco de dados. Projeto de pesquisa, Coordenação de Pós-Graduação em Engenharia Elétrica (COPELE), Universidade Federal da Paraíba (UFPB), Campina Grande PB, 1997.

[47] S. Fernandes Silva and T. Catarci. Visual modeling of temporal data in usability experiments. In Y. Ioannidis and W. Klas, editors, *Fourth Working Conference on Visual Database Systems (VDB4), L'Aquila, Italy*, pages 295–316. IFIP, May 1998.

[48] S. Fernandes Silva and T. Catarci. Graphical interaction with historical databases. In *Eleventh Int. Conf. on Scientific and Statistical Database Management (SSDBM'99), Ohio, USA*, 1999.

[49] S. Fernandes Silva and T. Catarci. Homogeneous access to temporal data and interaction histories in visual interfaces. In *Workshop on User Interfaces to Data Intensive Systems (UIDIS'99), Edinburgh, UK*, 1999.

[50] S. Fernandes Silva, T. Catarci, and U. Schiel. A graphical notebook as interaction metaphor for querying databases. In B. Neto and M. Camargo, editors, *XIV Simposio Brasileiro de Banco de Dados, Florianópolis, SC*, pages 171–185. SBC Sociedade Brasileira de Computação, 1999.

[51] S. Fernandes Silva, U. Schiel, and T. Catarci. Visual query operators for temporal databases. In R. Morris and L. Khatib, editors, *Fourth Workshop on Temporal Representation and Reasoning (TIME'97), Florida, USA*, pages 46–54. IEEE Computer Society, 1997.

[52] D. Fogg. Lessons from a "living in a databases" graphical query interface. In *ACM SIGMOD Conf. on the Management of Data*, pages 100–106, 1984.

[53] J. Fowler et al. Experience with the virtual notebook system: Abstraction in hypertext. In *Proc. of CSCW'94*, pages 133–143. ACM Press, 1994.

[54] C. Freksa. Temporal reasoning based on semi-intervals. *Artificial Intelligence*, 54:199–227, 1992.

[55] S.K. Gadia. A homogeneous relational and query language for temporal databases. *ACM TODS*, 13(4), 1988.

[56] K.J. Goldman et al. Isis: Interface for a semantic information system. In *ACM SIGMOD Conf. on the Management Data*, pages 328–342, 1985.

[57] I.A. Goralwalla, M.T. Ozsu, and D. Szafron. *An Object-Oriented Framework for Temporal Data Models. Temporal Databases: Research and Practice*, pages 1–35. Springer-Verlag Berlin Heidelberg. 1998.

[58] I.P. Groette and E.G. Nilsson. Sicon: An icon presentation module for an e-r database. In *Proc. of the 7th Conf. on Entity-Relationship Approach, Roma, Italy*, pages 271–289, 1988.

[59] D. Harel. On visual formalism. *Communication of the ACM*, 5:514–530, 1988.

[60] D. Haw, C. Goble, and A. Rector. Guidance: Making it easy for the user to be an expert. In P. Sawier, editor, *Interface to Database Systems, Lancaster 1994*, Workshop in Computing, Springer-Verlag, London, pages 25–48, 1995.

[61] S. Hibino and E. Rundensteiner. A visual query language for identifying temporal trends in video data. In IEEE Computer Press, editor, *Int. Workshop on Multimedia Database Management Systems, New York*, pages 74–81, 1995.

[62] S. Hibino and E. Rundensteiner. *A Visual Multimedia Query Language for Temporal Analysis of Video Data*, pages 123–159. Kluwer Academic Publishers, Norwell MA. 1996.

[63] P. Hietala and J. Nummenmaa. A multimodal database user interface and framework supporting user learning and user interface evaluation. In Cooper R., editor, *Interfaces to Database System, Glasgow 1992*, Workshop in Computing, Springer-Verlag, London, pages 392–405, 1993.

[64] R. Hull and R. King. Semantic database modeling: Survey, applications and research issues. *ACM Computing Surveys*, 19(3):201–260, 1987.

[65] R. Inder and J. Stader. Bags and viewers: A metaphor for intelligent database access. In Sawier P., editor, *Interface to Database Systems, Lancaster 1994*, Workshop in Computing, Springer-Verlag, London, pages 215–235, 1995.

[66] Y.E. Ioannidis. Dynamic information visualization. *ACM Sigmod Record*, 25(4), December 1996.

[67] C.S. Jensen et al. A consensus glossary of temporal database concepts. *SIGMOD RECORD*, 23(1), 1994.

[68] C.S. Jensen et al. *The Consensus Glossary of Temporal Database Concepts. Temporal Databases: Research and Practice.* Springer Verlag Berlin Heidelberg, 1998.

[69] A. Kadyamatimba, J. Mariani, and P. Sawyer. Desktop objects: Directly manipulating data and metadata. In Kennedy J., editor, *Third International Workshop on User-Interfaces to Database Systems, Edinburgh*, 1996.

[70] D.A. Keim and V. Lum. Gradi: A graphical database interface for a multimedia dbms. In Cooper R., editor, *Interfaces to Database System, Glasgow 1992*, Workshop in Computing, Springer-Verlag, London, pages 95–112, 1993.

[71] W. Kent. Limitation of record-based information models. *ACM Transaction on Database Systems*, 4(1):107–131, 1979.

[72] H.J. Kim, H.F. Korth, and A. Silberschatz. Picasso: A graphical query language. *Software Practice and Experience*, 18(3):169–203, 1988.

[73] W. Kim. A model of queries for object-oriented databases. In *Proc. 15th Int. Conf. on VLDB, Amsterdam*, 1989.

[74] R. King and S. Meiville. Ski; a semantic knowledgeable interface. In *Proc. of the 10th Int. Conf. on Very Large Databases, Singapore*, pages 30–37, 1984.

[75] H. Kitagawa et al. For ngraphics: A form-based graphics architecture providing a database workbench. *IEEE CGA*, 4(6):38–56, 1984.

[76] V. Kouramajian and M. Gertz. A visual query editor for temporal databases. In *Proc. of the 14th Int. Conf. on OO and E-R Modeling*, pages 388–399, 1995.

[77] M. Kuntz and R. Melchert. Pasta-3's graphical query language: Direct manipulation, cooperative queries, full expressive power. In *Proc. of the 15th Int. Conf. on Very Large Databases, Amsterdam, Holland*, 1989.

[78] J.A. Larson. A visual approach to browsing in a database environment. *IEEE Computer*, 19(6):62–71, 1986.

[79] L. Leventhal et al. Searching without a keyboard in a multimedia environment. In Nordby K. et al., editor, *INTERACT 95*, Chapmen & Hall: Oxford, pages 241–246, 1995.

[80] J. Littlehales and P. Hancox. The problems of integrating to publicily available databases. In Cooper R., editor, *Interfaces to Database System, Glasgow 1992*, Workshop in Computing, Springer-Verlag, London, pages 41–55, 1993.

[81] A. Massari and P.K. Chrysanthis. Visual query of encapsulated objects. In *Proc. of the 5th Int. Workshop on Research Issues on Data Engineering, Taipei, Taiwan*, pages 18–25, 1995.

[82] N. McDonald and M. Stonebraker. Cupid - the friendly query language. In *Proc. ACM Pacific 75 Conf.*, pages 127–131, 1975.

[83] M. MdSap and D. McGregor. Sfqi: Semi-formal query language interface to relational databases. In Sawier P., editor, *Interface to Database Systems, Lancaster 1994*, Workshop in Computing, Springer-Verlag, London, pages 104–124, 1995.

[84] A. Mendelzon. Visualizing the world wide web. In T. Catarci et al., editor, *Proc. of the Workshop on Advanced Visual Interfaces (AVI96), Gubbio, Italy*, Workshop in Computing, Springer-Verlag, London, pages 13–19, May 1996.

[85] L. Mohan and R.L. Kashyap. A visual query language for graphical interaction with schema-intensive databases. *IEEE TKDE*, 5(5):843–858, 1993.

[86] S. Mukherjea, J. Foley, and S. Hudson. Visualizing complex hypermedia networks through multiple hierarchical views. In I. Katz et al., editor, *Proc. of the Human Factors Computing Systems Conference (CHI'95), Denver, Colorado*, pages 331–337. ACM / SIGCHI, May 1995.

[87] N. Murray, C. Goble, and N. Paton. Kaleidoscape: a 3d environment for querying odmg compliant databases. In Y. Ioannidis and W. Klas, editors, *4th Working Conference on Visual Database Systems (VDB4), L'Aquila, Italy*, pages 85–102. IFIP, May 1998.

[88] M.C. Norrie. An interactive system for object-oriented data model with relations. In R. Cooper, editor, *Interfaces to Database System, Glasgow 1992*, Workshop in Computing, Springer-Verlag, London, pages 9–24, 1993.

[89] G. Ozsoyoglu and R. Snodgrass. Temporal and real-time databases: A survey. *IEEE TKDE*, 4(7):513–532, 1995.

[90] G. Ozsoyoglu and H. Wang. Example-based graphical database query languages. *IEEE Computer*, 26(5):25–38, 1993.

[91] A. Papantonakis and P.J.H. King. Syntax and semantic of gql, a graphical query language. *Journal of Visual Languages and Computing*, 6(1):3–25, 1995.

[92] N. Paton et al. Techniques for the effective evaluation of database interfaces. In Cooper R., editor, *Interfaces to Database System, Lancaster 1994*, Workshop in Computing, Springer-Verlag, London, pages 306–328, 1995.

[93] C. Plaisant et al. Lifelines: Visualizing personal histories. In M. Tauber et al., editor, *Proc. of the Human Factors Computing Systems Conference (CHI'96), Vancouver, Canada*, pages 221–227. ACM / SIGCHI, 1996.

[94] M. Rapley and J. Kennedy. Three dimensional interface for an object-oriented database. In Cooper R., editor, *Interfaces to Database System, Lancaster 1994*, Workshop in Computing, Springer-Verlag, London, pages 143–167, 1995.

[95] G. Santucci and F. Palmisano. A dynamic form-based data visualiser for semantic query languaged. In P. Sawier, editor, *Interface to Database Systems, Lancaster 1994*, Workshop in Computing, Springer-Verlag, London, pages 249–265, 1995.

[96] P. Sawyer et al. Object-oriented database systems: A framework for user interface development. In R. Cooper, editor, *Interfaces to Database System, Glasgow 1992*, Workshop in Computing, Springer-Verlag, London, pages 25–38, 1993.

[97] J.L. Schnase and J.J. Leggett. Computational hypertext in biological modeling applications. In *Proc. of Hypertext'89 Conference*, pages 181–197. ACM Press, 1989.

[98] F.M. Shipman, R.J. Chaney, and G.A. Gorry. Distributed hypertext for collaborative research: The virtual notebook system. In *Proc. of Hypertext'89 Conference*, pages 129–135. ACM Press, 1989.

[99] B. Shneiderman. Direct manipulation, a step beyond programming languages. *IEEE Computer*, 16(8):57–69, 1983.

[100] B. Shneiderman. Dynamic queries for visual information seeking. *IEEE Software*, 11:70–77, 1994.

[101] B. Shneiderman et al. Evaluating three museum installations of a hypertext system. *Journal of the American Society for Information Science*, 40(3):172–182, 1989.

[102] B. Shneiderman et al. *Sparks of Innovation, Human-Computer Interaction*, pages 281–294. Ablex Publ.., Norwood, NJ, 1993.

[103] R.T. Snodgrass. The tenporal query language `tquel`. *ACM Transactions on Database Systems*, 12(2):247–298, 1987.

[104] Y.W.S. Stanley, J.H. Soon, and M. Chen. Temporal association algebra: A mathematical foundation for processing object-oriented temporal databases. *IEEE Transactions on Knowledge and Data Engineering*, 10(3), 1998.

[105] A.U. Tansel, M.E. Arkun, and Ozsoyoglu G. Time-by-example: Query language for historical databases. *IEEE TSE*, 15(4):464–478, 1989.

[106] L. Tarantino. Hypertabular representation of database relations in world wide web front-ends. In Kennedy J., editor, *Third International Workshop on User-Interfaces to Database Systems, Edinburgh*, 1996.

[107] B. Theodoulidis et al. Interactive querying and visualization in temporal databases. In *Reasoning Workshop of the 4th DOOD Conf., Singapore*, 1995.

[108] K. Tsuda et al. Iconic browser: An iconic retrieval system for object-oriented databases. *Journal of Visual Languages and Computing*, 1(1):59–76, 1990.

[109] F. Ty. G-oql: Graphical interface to the object-oriented query language oql. Master's thesis, University of Florida, 1988.

[110] L.M. Wegner. Escher: Interactive, visual handling of complex objects in the extended nf2 database model. In Kunji T.L., editor, *Visual Database Systems*, North-Holland, 1989.

[111] L. M. Wegner et al. A visual interface for synchronous collaboration and negotied transactions. In T. Catarci et al., editor, *Proc. of the Workshop on Advanced Visual Interfaces (AVI96), Gubbio, Italy*, Workshop in Computing, Springer-Verlag, London, pages 155–165, May 1996.

[112] Y. Wu, S. Jajodia, and X.S. Wang. *Temporal Database Bibliography Update. Temporal Databases: Research and Practice.* Springer Verlag Berlin Heidelberg, 1998.

[113] S.B. Yao et al. Formanager: An office forms mangement system. *ACM Transactions on Office Information System*, 2(3):235–262, 1984.

[114] M.Y. Yen and R.W. Scamell. A human factors experimental comparison of `sql` and `qbe`. *IEEE Transactions on Software Engineering*, 19(4):390–402, 1993.

[115] J. Zhao, B. Kostka, and A. Muller. An integrated approach to task-oriented database retrieval interfaces. In Cooper R., editor, *Interface to Database Systems, Glasgow 1992*, Workshop in Computing, Springer-Verlag, London, pages 56–73, 1993.

[116] E. Zimanyi, C. Parent, and S. Spaccapietra. Terc+: A temporal conceptual model. In *International Symposium on Digital Media Information Base (DMIB'97), Nara, Japan*, 1997.

[117] M. Zloof. Query-by-example. *IBM Systems Journal*, 21(3):324–343, 1977.