O conteúdo do presente relatório é de única responsabilidade do(s) autor(es).
(The contents of this report are the sole responsibility of the author(s).)

Incorporação do Tempo em um SGBD Orientado a Objetos

Ângelo Roncalli Alencar Brayner
Claudia Bauzer Medeiros

Relatório Técnico DCC–94-02

Abril de 1994

Incorporação do Tempo em um SGBD Orientado a Objetos

Ângelo Roncalli Alencar Brayner
Claudia Bauzer Medeiros*

Sumário

Este artigo descreve a Camada de Gerenciamento Temporal, um subsistema de gerenciamento temporal implementado para o banco de dados orientado a objetos O_2. Este subsistema permite a definição de esquemas temporais e a manipulação (consultas e atualizações) de objetos nesses esquemas. Estes comandos são traduzidos pelo subsistema em programas e consultas executados pelo banco de dados subjacente. O trabalho apresentado contribui para a discussão sobre a implementação de sistemas temporais orientados a objetos, pouco explorado na literatura.

1 Introdução

Há vários anos vêm sendo realizadas pesquisas ligadas à incorporação do tempo a sistemas de bancos de dados. Uma revisão bibliográfica recente [Soo91] permite verificar a atividade de pesquisa na área, com cerca de 300 trabalhos citados, todos publicados nos últimos anos. A maioria destes trabalhos limita-se a incorporar facilidades temporais a bancos de dados relacionais. O primeiro trabalho mais abrangente que trata do tempo em bancos de dados orientados a objetos data de 1991 [SC91].

*Departamento de Ciência da Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP

Desde então, poucos trabalhos foram realizados neste contexto, devido a dois fatores principais:

- as dificuldades introduzidas pelo tratamento de dimensões temporais, tanto do ponto de vista de modelagem quanto de implementação, mesmo se limitado a sistemas relacionais. Em sistemas orientados a objetos, tais dificuldades são acrescidas pela introdução de herança e composição, que permitem combinações arbitrárias de objetos e suas propriedades;

- a pequena proporção de sistemas orientados a objetos existentes no mercado, aliada à não padronização do conceito de objetos. Com isto, torna-se mais difícil realizar um trabalho de tratamento temporal em sistemas orientados a objetos.

Este trabalho apresenta uma contribuição para o desenvolvimento das pesquisas nesta área. A contribuição consiste na descrição da implementação de um sistema de gerenciamento de tempo para um banco de dados orientados a objetos a partir do modelo TOODM proposto por [Lin93]. Este sistema, a *Camada de Gerenciamento Temporal*, foi implementado sobre o sistema O_2 [dddO91] e permite a definição e o gerenciamento de dados temporais, bem como o processamento de consultas temporais neste banco de dados.

As principais contribuições deste trabalho são portanto as seguintes:

- incorporação de um sistema de gerenciamento e consultas de dados temporais a um SGBD orientado a objetos.

- descrição da implementação realizada, identificando problemas existentes.

O restante do texto está organizado da seguinte forma. A seção 2 apresenta uma revisão bibliográfica da pesquisa e implementação em bancos de dados temporais, evidenciando a quase inexistência de trabalhos relativos a orientação a objetos. A seção 3 descreve as principais propriedades e características do modelo TOODM e da linguagem TOOL.

A seção 4 apresenta a Camada de Gerenciamento Temporal, responsável pela interface entre o usuário e o SGBD e descreve como é realizado o gerenciamento temporal de objetos. A seção 5 descreve o processamento de consultas realizado pela Camada. A seção 6 descreve as estruturas de dados implementadas e apresenta exemplos do mapeamento realizado pela Camada entre comandos do usuário e comandos em O_2. Finalmente, a seção 7 apresenta conclusões e extensões.

2 Bancos de Dados Temporais

Sistemas temporais têm sido propostos nos mais diversos contextos, como por exemplo, em automação de escritórios [Ede94] ou em sistema de suporte à decisão. Este trabalho se ocupa exclusivamente da incorporação de tempo em bancos de dados, sem se preocupar com as aplicações que manipularão o banco.

2.1 Representação Temporal de Dados

As propostas de incorporação de tempo em bancos de dados utilizaram basicamente as seguintes abordagens: *(i)* introduzir o conceito de tempo na semântica de um modelo de dados; ou *(ii)* introduzir, em um modelo de dados qualquer, atributos que representem as dimensões temporais.

Na primeira abordagem, torna-se necessária a especificação de uma lógica para formalizar a semântica temporal [CW83]. Na segunda abordagem, é necessário apenas definir funções para mapear operações temporais em operações básicas do modelo. Esta segunda abordagem tem sido a mais utilizada.

A maioria dos modelos temporais limita-se a incorporar o contexto temporal no modelo relacional. O número de propostas que incorporam o conceito de tempo no modelo orientado a objetos é reduzido.

Pode-se caracterizar as propostas de modelos temporais de dados através dos seguintes fatores:

Modelo de Dados Suporte : Nas propostas de modelos de dados temporais apresentadas em [S. 79, SA85a, SA85b, Tan86, TA86,

Ari86, NA87, NA89, Sad87, Gad88], o modelo relacional é estendido para incorporar o contexto temporal. A incorporação de tempo no modelo orientado a objetos é proposta em [WD92, Wol92, Lin93]. Em [EW90] a incorporação de tempo é feita sobre o modelo *EER* (*Enhanced Entity-Relationship*);

Número de Dimensões Temporais : Existem modelos que incorporam apenas uma dimensão temporal, na maioria das vezes a dimensão *Tempo Válido* como em [S. 79, EW90, Tan86, Sad87]. Outros modelos representam a evolução temporal de dados em duas dimensões temporais, *Tempo de Transação* e *Tempo Válido* [SA85a, NA87, WD92, Wol92, Lin93]. Há ainda modelos que propõem um número um número infinito de dimensões temporais [Ari86, Gad88];

Nível de incorporação do Tempo : As dimensões temporais podem ser incorporadas em dois níveis: nível de um objeto do mundo real (o que corresponde a tupla no modelo relacional, entidade no modelo *ER* e objeto no modelo orientado a objetos) [S. 79, SA85a, Ari86, Wol92]; ou nível de atributos de um objeto (o que corresponde a atributos nos modelos relacional e *ER* e componentes de um objeto no modelo orientado a objetos) [Tan86, NA87, Gad88, EW90, WD92, Lin93].

2.2 Consultas Temporais

As linguagens de consulta e álgebras temporais propostas introduzem, na realidade, extensões às operações básicas dos modelos de dados suporte.

A maioria das propostas introduz as operações de *Seleção Temporal* e *Projeção Temporal*. A operação de seleção temporal retorna dados em função de um predicado temporal. Por exemplo, a linguagem TOOL [Lin93] introduz as cláusulas *TSlice* e *VSlice* para processar operações de seleção temporal sobre os eixos temporais *Tempo-de-Transação* e *Tempo-Válido*, respectivamente. A linguagem TQUel [Sno87] introduz

as cláusulas *When* para operar sobre o eixo *Tempo-Válido* e *As-of* para operar sobre o eixo *Tempo-de-Transação*.

A operação de projeção temporal retorna valores de tempo. Estes valores são representados através de atributos específicos. Na linguagem TOOL implementada neste trabalho, operações de projeção temporal são realizadas através das cláusulas *TWhen* e *VWhen* que retornam valores de *Tempo de Transação* e de *Tempo Válido*, respectivamente.

Os predicados temporais são especificados através de construtores e operadores temporais. Os operadores temporais (operadores que retornam um valor *booleano*) mais comuns são: *Before*, *After*, *Overlap*, *During*. Estes construtores são propostos em [Sno87, Ari86, Gad88, Wol92, Lin93]. Os construtores temporais (operadores que retornam valores de tempo) propostos na maioria das linguagens de consultas temporais [Sno87, Gad88, EW90, Lin93] são: *Last_Instant*, *Last_Interval*, *First_Instant*, *First_Interval*, *Adjacent*.

Independente do modelo de dados e da linguagem de consulta, o processamento de consultas temporais apresenta o seguinte conjunto de problemas:

- O volume de dados armazenados em um BDT é de uma ordem de grandeza bastante superior ao volume de dados de um BD convencional. Isto implica que novos métodos de indexação (estruturas e algoritmos de busca) devem ser desenvolvidos;
- Os métodos tradicionais de indexação só podem ser utilizados para valores sobre os quais pode ser executado algum tipo de ordenação. Valores do tipo intervalo não suportam ordenação completa. Torna-se necessário o desenvolvimento de novas estruturas de acesso para intervalos (por exemplo, a estrutura de acesso *Índice Temporal* proposta por [DW92]);
- Manipulação de *informações incompletas*:
  - Valores de objetos desconhecidos ou inexistentes. Em certos casos, incertezas quanto à existência de um objeto em certos pontos no tempo;

– Indeterminação temporal.

3 O Modelo TOODM

O TOODM é uma extensão ao modelo de dados orientado a objetos, introduzindo, neste modelo, as dimensões temporais *Tempo de Transação* e *Tempo Válido*. Estas dimensões temporais estão representadas através de atributos implícitos. O modelo TOODM apresenta três tipos de objetos:

- objetos do tipo TIME: objetos utilizados para representar as dimensões temporais;
- objetos invariantes no tempo: objetos que não são compostos por nenhum objeto do tipo TIME;
- objetos variantes no tempo: objetos complexos compostos por pelo menos um objeto do tipo TIME.

Segundo [Lin93], *categoria temporal* de uma classe é a classificação desta quanto ao número de dimensões temporais incorporadas a seus objetos. Assim, podem existir quatro tipos de categoria temporal em um banco de dados: classes *Instantâneas*, classes *Tempo-de-Transação*, classes *Tempo-Válido* e classes *Bitemporais*.

Os relacionamentos entre objetos são representados no modelo orientado a objetos através dos grafos de herança e composição. As propriedades de herança e composição podem gerar conflitos durante a evolução de esquemas. Ao se incorporar o contexto temporal em um banco de dados orientado a objetos, o número destes conflitos pode crescer como decorrência do que denominamos de *conflitos temporais*. O TOODM apresenta heurísticas que tentam resolver os conflitos temporais. O processo de execução destas heurísticas é denominado de *temporalização* [Lin93]. Este processo deve ser executado durante a evolução de esquemas (criação e alteração).

O processo de temporalização é composto de duas fases: resolução de conflitos temporais pelo grafo de herança e pelo grafo de composição.

Para definir o processo de temporalização foi proposta uma propriedade temporal para os grafos de herança e de composição. Denominamos esta propriedade como *invariante temporal* dos grafos de herança e de composição (o conceito de invariantes no modelo orientado a objetos é definido em [Jay87]). A invariante garante que uma classe deve manter no mínimo as dimensões temporais incorporadas por suas antecessoras.

Para consultar o TOODM, [Lin93] propôs a linguagem TOOL. Esta linguagem permite consultas que retornam:

- valores de tempo (objetos do tipo TIME);
- valores invariantes no tempo;
- objetos, classes ou estados temporais.

Consultas que retornam valores de tempo correspondem às operações de *VWhen* (para *Tempo Válido*) e *TWhen* (para *Tempo de Transação*). Consultas que retornam objetos, classes ou estados temporais referem-se a operações de seleção temporal (*TSlice, VSlice*). A linguagem permite combinar arbitrariamente estas operações.

O processador de consultas descrito neste trabalho implementou estas operações através de uma *interface* interativa.

4 A Camada de Gerenciamento Temporal

A incorporação do tempo a bancos de dados se faz de duas maneiras: modificando o SGBD ou criando uma interface entre o SGBD e o usuário.

O trabalho desenvolvido segue esta segunda linha, por impossibilidade de modificar o SGBD utilizado. Desta forma, a incorporação do tempo foi implementada através da criação da *Camada de Gerenciamento Temporal*, que se comporta como uma *interface* entre o usuário e o SGBDOO O_2. Sua função básica é suprir o O_2 com uma plataforma temporal.

O princípio básico de funcionamento da *Camada de Gerenciamento Temporal* é o de aliar as funções básicas de gerenciamento temporal às

funções de evolução de esquema e de gerenciamento de objetos apresentadas pelo O_2. Assim, especificações de operações no TOODM e na TOOL são mapeadas em comandos O_2.

Para representar a incorporação do contexto temporal em um SGBD, o modelo TOODM pressupõe a existência prévia de um modelo de dados e de uma linguagem de consulta neste SGBD. Com base nesta premissa, o mapeamento de operações temporais solicitadas pelo usuário em comandos pertencentes à sintaxe O_2 é principal tarefa da Camada de Gerenciamento Temporal.

Este mapeamento pode gerar comandos que podem ser agrupados em três conjuntos distintos de comandos O_2. Estes conjuntos são os seguintes:

Comandos de manipulação de esquema: têm a função de definir *esquemas, bases, classes, aplicações, programas, funções* e *métodos*;

Instruções imperativas: são utilizadas para codificar *programas, métodos, funções* e *trechos de códigos*. A principal função das instruções imperativas é alterar objetos e valores;

Comandos de consulta: têm a função de recuperar objetos e valores no BD;

Através destes mapeamentos, a *Camada de Gerenciamento Temporal* interage indiretamente com dois componentes da arquitetura do O_2: *o gerenciador de esquema* e *o gerenciador de objetos*.

A figura 1 ilustra o processo de interação da *Camada de Gerenciamento Temporal* com o usuário e com o O_2. A representação do O_2 foi feita através de sua arquitetura funcional (composta das três camadas especificadas na figura 1) [dddO91].

Durante o processamento de mapeamento dos comandos do usuário em comandos O_2, a *Camada de Gerenciamento Temporal* precisa garantir:

1. a manutenção da semântica das especificações do usuário (por exemplo, garantir que, em uma consulta especificada pelo o usuário, os comandos O_2 gerados corespondam à execução da solicitação);

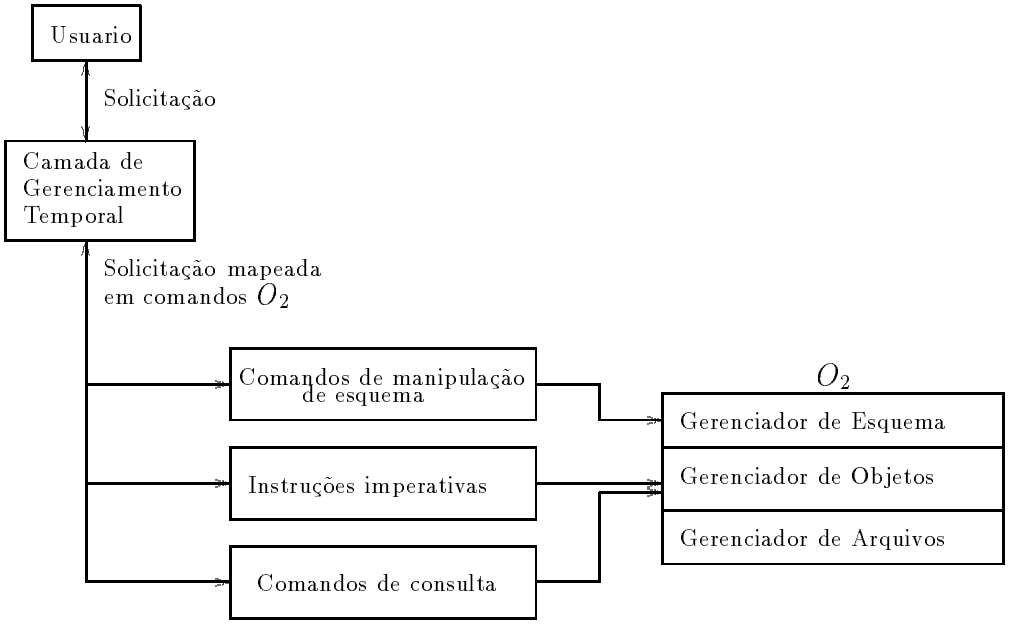

Figura 1: Processo de interação da *Camada de Gerenciamento Temporal* com o usuário e o O_2

2. as restrições de integridade temporal do modelo TOODM (por exemplo, garantir que uma classe *Bitemporal* não possa ser declarada como superclasse de uma classe *Instantânea*);

3. a geração de comandos O_2 sintaticamente corretos.

Através da *Camada de Gerenciamento Temporal* é possível manipular qualquer objeto do modelo de dados do O_2 dentro do plano temporal. Em outras palavras, é possível criar, alterar e remover esquemas, bases e classes de um banco de dados O_2.

5 Gerenciador de Consultas

O *Gerenciador de Consultas Temporais* é um módulo da *Camada de Gerenciamento Temporal* que traduz consultas TOOL em consultas O_2, processadas pelo O_2query (processador de consultas apresentado pelo O_2).

O *Gerenciador de Consultas* permite as seguintes operações:

- Consultas que operam sobre o eixo *Tempo-de-Transação*. Muitos autores denominam este tipo de consultas como operação de **Rollback**. Estas consultas retornam valores de tempo ou objetos, dependendo da cláusula temporal a ser utilizada;

- Consultas que operam sobre o eixo *Tempo-Válido*. Este tipo de consulta também é denominada **Histórica**. Estas consultas retornam valores de tempo ou objetos;

- Consultas atemporais

O princípio básico de funcionamento do *Gerenciador de Consultas* é o seguinte: através de janelas, o gerenciador solicita ao usuário a especificação de parametros necessários à construção de uma consulta temporal. Com base nestes parametros, o *Gerenciador de Consultas* faz o mapeamento da consulta temporal em uma consulta da sintaxe O_2SQL. Após isto, submete a consulta ao processador de consultas do O_2.

Os parametros de consulta necessários à formulação de uma consulta temporal são os seguintes:

- *Nome da classe* a ser consultada (Obrigatório);
- *Nome da extensão* da classe a ser consultada (Obrigatório);
- *Cláusula Temporal* a ser utilizada - *TWhen, VWhen, TSlice, VSlice, Atemp*[1] (Obrigatório);
- *Predicado atemporal* (Opcional);
- *Predicado temporal* (Opcional);
- *Operadores temporais* (Opcional) (por exemplo, *Before, After*);
- *Construtores temporais* (Opcional) (por exemplo, *Min_interval*).

O parâmetro *predicado atemporal* tem a função de selecionar um subconjunto de objetos da extensão da classe a ser consultada. Por exemplo, um usuário deseja aplicar uma consulta temporal aos funcionários de *sobrenome "Vitorino"*. Para formular esta consulta, o usuário deve especificar um predicado *em sintaxe* O_2SQL para restringir a consulta ao subconjunto de funcionários de sobrenome "Vitorino". Caso o usuário não especifique nenhum predicado atemporal, o conjunto de objetos ao qual será aplicada a consulta temporal corresponderá à extensão da classe a ser consultada.

O parâmetro *predicado temporal* tem a função de restringir *o intervalo de tempo* a ser consultado. Por exemplo, um usuário pode desejar aplicar uma consulta temporal a um conjunto de funcionários quando estes ganhavam um salário maior que US$ 1,200. Para formular esta consulta, é necessário que o usuário especifique um predicado para restringir o intervalo de tempo sobre o qual será aplicado a consulta para o intervalo correspondente ao período no qual cada funcionário recebia salário maior que 1,200.

[1] Esta opção refere-se à aplicação de consultas não temporais, caracterizadas pela ausência de cláusulas temporais.

O trabalho descrito em [ea93] propõe um conjunto básico de consultas que um SGBD temporal deve poder prover. Esta proposta de benchmark foi realizada para sistemas temporais relacionais. No entanto, o mesmo tipo de consulta pode ser definido em sistemas orientados a objetos. A *Camada de Gerenciamento Temporal* permite executar todas as consultas especificadas neste benchmark, além de vários outros tipos de consulta não contemplados. Assim sendo, o sistema implementado não apenas satisfaz estes requisitos, mas provê uma maior flexibilidade em termos de navegação temporal.

Através do *Gerenciador de Consultas* proposto, conseguiu-se efetuar consultas do tipo:

- *Qual o funcionário que permaneceu o maior período sem receber aumentos de salário ?*
- *Quando foi o último aumento de salário para cada funcionário antes de 10/11/1992 ?*
- *Quando foi o primeiro aumento de salário após 25/12/1991 ?*
- *Listar o histórico salarial dos funcionário lotados no departamento DCC após 15/03/90;*
- *A partir de quando e quais foram os funcionários têm salários maior que US$ 1,500 ?*
- *Quando os objetos de sobrenome "Vitorino" foram incluídos no BD ?*

6 Tempo: armazenamento e representação em BDOO

Nesta seção são apresentadas as estruturas propostas para armazenar tempo em Bancos de Dados Orientados a Objetos, dando-se ênfase à implementação destas estruturas no O_2. É apresentado também como o conceito de Tempo está representado nestas estruturas.

6.1 Estruturas para incorporar o Plano Temporal em um BDOO

A *Camada de Gerenciamento Temporal* manipula objetos O_2 estendidos com estruturas específicas que permitem gerenciamento temporal. Estas estruturas foram definidas de forma a garantir a eficiência na navegação de tempo dando suporte a todas as operações especificadas no TOODM.

A inclusão das dimensões temporais em objetos foi feita com base em dois parametros: *tipo de construtor utilizado* e *nível de encapsulamento do objeto*.

- **Tipo de Construtor.** A incorporação de dimensões temporais *Tempo de Transação* e *Tempo Válido* a objetos depende do tipo de construtor utilizado na definição destes objetos. As duas dimensões podem ser associadas a objetos definidos pelo construtor **tuple**. Para objetos definidos pelos construtores **set** ou **list**, é associada somente a dimensão *Tempo de Transação*. A justificativa para a utilização desta estratégia é de ordem prática, pois não identificamos um único tipo de aplicação que necessitaria controlar a dimensão temporal *Tempo Válido* para conjuntos de objetos;

- **Nível de Encapsulamento.** Para o nível mais externo de um objeto complexo, foi incorporada somente uma dimensão temporal, dependendo do construtor utilizado. Por exemplo, se for utilizado o construtor **tuple** para definir o nível mais externo de um objeto, então será incorporada a dimensão *Tempo Válido* neste nível. Caso seja utilizado o construtor **set** ou **list**, a dimensão a ser incorporada será *Tempo de Transação*. Em nível de componentes de um objeto (ou seja, em nível de atributos) podem ser associadas as duas dimensões temporais, conforme a regra anterior. A definição desta estratégia se deve aos seguintes fatores:

  (i) Permite redução na replicação de dados, já que não se armazenará os estados de um objeto, mas sim, de seus atributos (componentes);

(ii) Permite recuperar um estado qualquer de um objeto a partir de uma operação de união dos valores dos atributos deste objeto para o dado estado.

Assim, um objeto de tipo *set(tuple(x:set(integer), y:tuple(z:string)))* poderá ter os seguintes eixos temporais associados: TT, ao nível mais externo (set); TT e TV ao nível de tuple; TT ao componente x da tupla; e TT e TV ao componente y da tupla.

A representação da evolução temporal de um objeto ao longo de um único eixo, *Tempo-Válido* ou *Tempo-de-Transação*, utiliza a estrutura de **listas encadeadas**. Cada elemento da lista é uma tupla, do tipo: <**valor, valor_tempo**> onde *valor* é um ponteiro para um objeto (OID) ou um valor atômico, e *valor_tempo* é um valor de tempo no eixo temporal que está sendo representado. Este valor de tempo pode ser um número real (para representar valores no eixo *Tempo-de-Transação*) ou um intervalo (caso o eixo representado seja *Tempo-Válido*).

Para incorporar as duas dimensões temporais simultaneamente (objetos bitemporais), efetua-se o aninhamento da lista de TV dentro da lista de TT. Com este aninhamento, a incorporação das duas dimensões temporais ocorre da seguinte forma: *cada estado de um atributo no BD é acompanhado do histórico deste atributo no mundo real.* No nível mais externo de objeto a estrutura se mantem inalterada, pois neste nível há a incorporação de apenas uma dimensão temporal. Em nível de atributo há a seguinte alteração: para cada elemento da lista que representa a dimensão *Tempo de Transação*, é embutida a lista que representa a dimensão *Tempo Válido* no lugar do atributo *valor* da tupla <valor, valor_tempo>.

A seguir, é apresentada uma sequência de figuras que ilustra a incorporação do contexto temporal em um objeto através destas estruturas. Para facilitar o entendimento, os valores para a dimensão *Tempo Válido* representam a quantidade de segundos após a marca de tempo utilizada como origem do eixo *Tempo-Válido*. Os valores para a dimensão *Tempo-de-Transação* foram expressos em valores simbólicos para destacar a ortogonalidade dos dois eixos temporais.

Considere que os seguintes eventos ocorreram na sequência cronológica (no eixo *Tempo-de-Transação*) em que são apresentados na figura 2:

1. em um instante **TT1**, um objeto, o *Objeto_1*, é incluído na classe Bitemporal **C**. Este objeto começa a existir no mundo real um segundo após a marca de tempo definida como origem para o eixo *Tempo-Válido*. O objeto *Objeto_1* é composto pelos atributos temporais *atributo-1* (atômico) e *atributo-2* (não atômico, ou seja, é um objeto pertencente a uma outra classe qualquer), e pelo atributo *histórico* (lista encadeada gerida pela *Camada de Gerenciamento Temporal* representando a evolução temporal do objeto no eixo *Tempo-Válido*). Neste mesmo instante **TT1** são atribuídos valores aos atributos *atributo-1* e *atributo-2*. O valor atribuído a *atributo-1* passa a ser válido no momento em que se inicia a validade do objeto *Objeto_1* no mundo real. O valor atribuído para *atributo-2* só começa a ser válido dois segundos após a origem do eixo *Tempo-Válido* (considerando que o objeto correspondente a OID1 já é válido neste instante). O símbolo OID no histórico do *atributo-2* significa a presença de um ponteiro que aponta para o objeto definido como valor para este atributo;

2. em um instante **TT2**, o valor do *atributo-1* é alterado de *valor-1* para *valor-2*. Seu período de validade no mundo real inicia-se dois segundos após a origem do eixo *Tempo-Válido*. Com isso, é inserido um novo estado na lista de estados de *atributo-1*, o estado TT2. Na lista de histórico do mundo real deste atributo é inserido também um novo elemento que representa o período de validade (o intervalo $[2,\infty)$) deste novo valor.

6.2 O Armazenamento do Contexto Temporal no O_2

As estruturas para armazenar a evolução temporal de objetos nos eixos *Tempo-de-Transação* e *Tempo-Válido* foram implementadas a partir dos construtores de tipos **list** e **tuple** existentes no O_2. Estes construtores

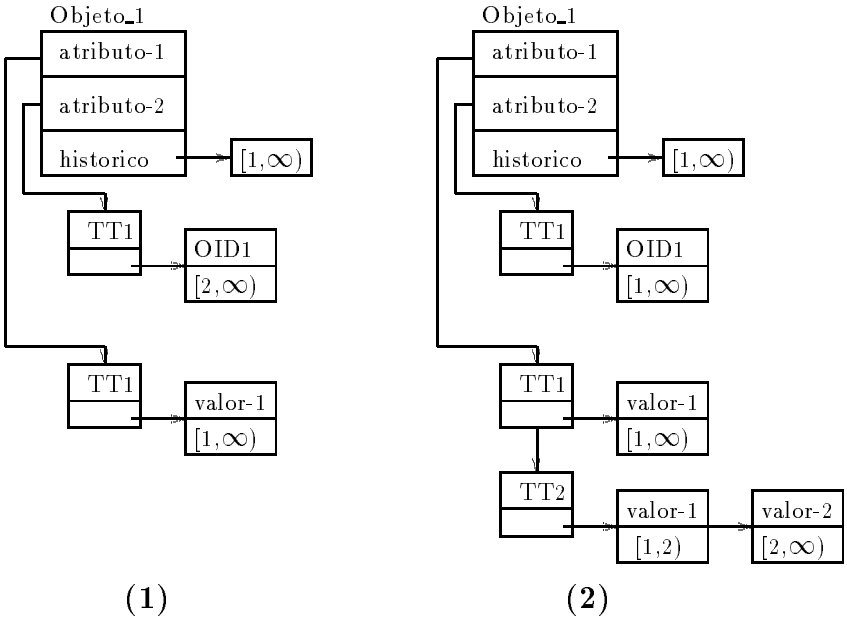

Figura 2: Os estados de Objeto_1 nos instantes **TT1** e **TT2**.

foram utilizados por serem a forma mais eficiente para reduzir espaço de armazenamento e tempo de consulta, pois são otimizados internamente pelo O_2. A seguir, são apresentadas estas implementações para cada tipo de classe temporal.

Classe Bitemporal. Esta é a mais genérica no plano temporal, já que apresenta as duas dimensões temporais possíveis.

Para implementar a estrutura que armazena o histórico de objetos do tipo **tuple**, a *Camada de Gerenciamento Temporal* especifica automaticamente o atributo <**historico_objeto**>, cuja definição é a seguinte:

```
historico_objeto: list( tuple( inicio_tv: real,
                               fim_tv : real))
```

A implementação da estrutura que armazena os estados de objetos do tipo **set, unique set, list** apresenta a seguinte definição:

```
list(tuple( valor: <tipo-especificado-usuário>,
        tt_set_list: real))
```

Dependendo do tipo de um atributo temporal, foram definidas diferentes estratégias para implementar a estrutura que armazena sua evolução em cada eixo temporal. Existem estratégias distintas para os seguintes tipos de atributos:

- *(i)* um tipo primitivo do O_2 (*integer, real, string, char, boolean*);
- *(ii)* um tipo complexo (atributos que correspondem a objetos complexos).

Todos atributos são implementados com a seguinte definição:

```
nome_atributo: list(tuple( tt_nome_atributo: real,
                           historico_nome_atributo:
                           list(tuple(valor_nome_atributo: TIPO | OID,
                                      inicio_tv_nome_atributo: real,
                                      fim_tv_nome_atributo : real))))
```

No caso de atributos atômicos, *valor_nome_atributo* indica o tipo diretamente (por exemplo *integer*). No caso de atributos complexos, *valor_nome_atributo* contém o OID do objeto correspondente.

Classe Tempo-Válido. Este tipo temporal de classe só apresenta a dimensão temporal *Tempo-Válido*. Sua implementação é semelhante à das classes Bitemporais, com a remoção da estrutura que armazena a dimensão *Tempo de Transação*.

```
nome_atributo: list(tuple (valor_nome_atributo: TIPO | OID,
                           inicio_tv_nome_atributo: real,
                           fim_tv_nome_atributo : real))
```

Classe Tempo-de-Transação. É necessária apenas a remoção das listas que armazenam a dimensão *Tempo Válido*. Um atributo definido como temporal em uma classe deste tipo apresenta a seguinte especificação:

```
nome_atributo: list (tuple (tt_nome_atributo: real,
                            valor_nome_atributo: TIPO | OID ))
```

6.3 Representação do Tempo

As estruturas descritas conseguem armazenar valores de tempo representados através de um *modelo linear* (onde os valores de tempo ocorrem de forma linear e ordenada, no sentido *passado* → *futuro*) e *discreto* de tempo, apresentando as duas dimensões temporais, *Tempo de Transação* e *Tempo Válido*, definidas para o TOODM.

Para a definição da granularidade das marcas de tempo nos eixos temporais *Tempo-de-Transação* e *Tempo-Válido*, considerou-se a existência de dois níveis de abstração na representação de granularidade:

1. nível do usuário (nível de abstração mais alto), e;
2. nível interno da *Camada de Gerenciamento Temporal*.

Para o nível de abstração mais baixo (o nível interno da *Camada de Gerenciamento Temporal*), as marcas de tempo nos dois eixos temporais

foram representadas por uma granularidade do tipo *simples* e que utiliza a métrica *segundo*. A vantagem desta padronização de granularidade no nível interno da *Camada de Gerenciamento Temporal* é evitar a incompatibilidade de granularidades entre objetos do BD (que pode ocorrer em operações sobre o eixo *Tempo-Válido*). Os valores de tempo neste nível são definidos como sendo do tipo *real*.

Para representar a granularidade no nível do usuário, foi introduzido o conceito de **granularidade virtual**. Granularidade virtual é a propriedade que permite ao usuário "enxergar" e manipular uma granularidade distinta da granularidade interna da *Camada de Gerenciamento Temporal*. Em outras palavras, com base na propriedade da granularidade virtual, a *Camada de Gerenciamento Temporal* garante a independência da implementação temporal.

Para cada eixo temporal, foram definidas estratégias distintas para implementar o conceito de granularidade virtual. O princípio básico da propriedade da granularidade virtual consiste na transformação do tipo de granularidade de uma marca de tempo.

Por definição, o usuário não exerce nenhum tipo de controle sobre a evolução temporal de objetos no eixo *Tempo-de-Transação*. Em outras palavras, ao usuário é permitido apenas consultar valores de tempo sobre este eixo temporal. Desta forma, padronizou-se que, neste eixo, a granularidade virtual (com a qual o usuário irá trabalhar) é do tipo *composta* e formada pelos seguintes elementos: *<ano, mês, dia, hora, minuto, segundo>*. Isto implica que as marcas de tempo sobre o eixo *Tempo-de-Transação* são apresentadas ao usuário através desta granularidade virtual.

Muitas vezes, o usuário deseja recuperar estados do BD especificando valores de *Tempo de Transação* através de uma granularidade virtual que apresenta um refinamento menor. Por exemplo, o usuário pode desejar recuperar um estado do BD especificando uma marca de tempo com granularidade virtual: < ano,mês,dia>. Para viabilizar este tipo de operação, foi introduzida um novo tipo de operação temporal que denominou-se de **Projeção Temporal sobre Granularidade** (PTG).

Por exemplo, considere que um evento **e** é representado segundo a

granularidade <ano, mês, dia, hora, minuto, segundo>. Considere ainda que **e** apresenta os seguintes valores de tempo <1993,10,12,10,30,45>. A operação de PTG *G[dia, segundo]* para o evento **e** retorna os valores <12,45>.

A implementação do conceito de granularidade virtual (no nível do usuário) para representar valores de tempo sobre o eixo *Tempo-Válido* baseou-se na seguinte estratégia: durante a especificação de atributos temporais, a *Camada de Gerenciamento Temporal* solicitará ao usuário que defina uma granularidade virtual para estes atributos.

Marcas de tempo no eixo temporal *Tempo-de-Transação* são representadas como *eventos*. Em outras palavras, para a *Camada de Gerenciamento Temporal*, um evento no eixo *Tempo-de-Transação* representa a mudança de um estado **E1** do Banco de Dados para um estado **E2**. Cada evento representa o valor do **Tempo de Máquina** correspondente ao tempo de *compromisso* (*commit*) de uma transação.

Para representar marcas de tempo no eixo temporal *Tempo-Válido*, foi definido o formato de *intervalos*.

6.4 Exemplo

Em função das estruturas descritas, mostramos a seguir o mapeamento de comandos gerados automaticamente pela *Camada de Gerenciamento Temporal*. Suponha a classe *Funcionário* definida (esta definição também é gerada pela Camada) como:

```
class Funcionario public type
      tuple(nome: list(tuple(tt_nome:real,
                             historico_nome:list(tuple(valor_nome:string,
                                                       inicio_tv_nome:real,
                                                       fim_tv_nome:real)))),
            salario: list(tuple(tt_salario:real,
                                historico_salario:list(tuple(valor_salario:real,
                                                             inicio_tv_salario:real,
                                                             fim_tv_salario:real)))),
            historico_objeto: list(tuple(inicio_tv:real,
                                         fim_tv:real)))
end;
```

name Funcionarios: set(Funcionario); /* Repositório de dados*/

A consulta *Quando os funcionários de sobrenome "Vitorino" receberam salário maior que US$ 1,200* gera uma consulta O_2SQL em quatro etapas: duas seleções sucessivas, uma projeção e uma operação de agrupamento dos valores de tempo por objeto.

→ Consulta TOOL

*VWhen_View (nome like "*Vitorino*")*
From Funcionário
INDB (during (salario > 1,200))

→ Consulta O_2SQL gerada

1. Seleção

```
define q1 as select o from o in Funcionarios
             where (element( select (last(last(p.nome).historico_nome)).valor_nome
                             from p in Funcionarios
                             where o=p)) like "*Vitorino*" and
                   (element( select (last(q.historico_objeto)).fim_tv
                             from q in Funcionarios
                             where o=q)) = 0 and
                   (element( select (last(q.historico_objeto)).inicio_tv
                             from q in Funcionarios
                             where o=q)) != 0
```

2. Seleção e Projeção

```
define q2 as select tuple(objeto: t, inicio:l.inicio_tv_salario, fim:l.fim_tv_salario)
             from t in q1,
                  p in (select last(o.salario) from o in q1
                        where o=t),
                  l in p.historico_salario
             where l.valor_salario > 1,200
```

4. Agrupamento

```
q3: group resposta in q2 by (objeto: resposta.objeto)
          with (intervalos: select tuple(inicio:p.inicio, fim:p.fim)
                            from p in partition)
```

7 Conclusões e extensões

Este artigo apresentou uma descrição da implementação de um sistema de gerenciamento temporal, a *Camada de Gerenciamento Temporal*. As

principais características desta implementação são: *(i)* utilizar o TOODM como modelo temporal de dados, bem como a linguagem de consulta TOOL proposta por [Lin93]; *(ii)* funcionar como uma camada sobreposta a um SGBDOO já existente, o O_2; *(iii)* apresentar um módulo específico para processamento de consultas temporais, o *Gerenciador de Consultas*. Uma descrição mais detalhada pode ser encontrada em [Bra94].

As principais contribuições apresentadas neste trabalho são:

- a identificação e a resolução de problemas inerentes à incorporação do conceito de tempo em um BDOO.
- a apresentação de estruturas que viabilizam a representação da evolução temporal de objetos. Estas estruturas foram definidas a partir dos construtorese de tipos *Tuple* e *List*. Isto implica que estas estruturas podem ser implementadas em qualquer SGBDOO que apresente estes construtores;
- a descrição de um processador de consultas temporais implementado capaz de efetuar consultas nos dois eixos temporais;

Aliado à portabilidade apresentada pela implementação, deve-se ressaltar que as consultas temporais por ela processadas superam o conjunto de consultas temporais apresentado pelo *benchmark* de [ea93].

Como extensões a este trabalho, podemos citar:

- Incorporação do contexto temporal no gerenciamento da evolução de esquemas;
- Incorporação do contexto temporal no controle de versões de métodos;
- A partir da álgebra que suporta o processamento de consultas do O_2, definição de novas operações sobre os atributos que representam as dimensões temporais para introduzir operações temporais nesta álgebra;

Referências

[Ari86] G. Ariav. A Temporally Oriented Data Model. *ACM Transactions on Database Systems*, 11(4):499–527, December 1986.

[Bra94] Angelo Roncalli Alencar Brayner. Implementação de um Sistema Temporal em um Banco de Dados Orientado a Objetos. Master's thesis, Universidade Estadual de Campinas, 1994.

[CW83] James Clifford and David S. Warren. Formal semantics for time in databases. *ACM Transactions on Database Systems*, 8(2):214–254, June 1983.

[dddO91] Grupo de desenvolvimento do O2. O2. *Communications of the ACM*, 34(10):35–48, 1991.

[DW92] Umeshwar Dayal and Gene T.J. Wuu. A Uniform Approach to Processing Temporal Queries. In *Proceedings of the 18th VLDB Conference*, pages 407–418, 1992.

[ea93] Christian S. Jensen et al. The TSQL Benchmark. In *Proceedings of the International Workshop on a Infrastructure for Temporal Databases*, pages QQ–1–QQ–28, June 1993.

[Ede94] N. Edelweiss. *Sistemas de Informação de Escritórios: Um Modelo para Especificações Temporais.* PhD thesis, DI-UFRGS, 1994.

[EOP93] N. Edelweiss, J. P. Oliveira, and B. Pernici. An Object-oriented Temporal Model. In *Proc CAISE Conference*, 1993.

[EW90] Ramez Elmasri and Gene T. J. Wuu. A TEMPORAL MODEL AND QUERY LANGUAGE FOR ER DATABASES. In *Proceedings of the 6th Conference on Data Engineering*, pages 76–83, February 1990.

[Gad88] Shashi K. Gadia. A Homogeneous Relational Model and Query Languages for Temporal Databases. *ACM Transactions on Database Systems*, 13(4):418–448, December 1988.

[Jay87] Jay Banerjee, Hong-T Chou, Jorge Garza, Won Kim, Darrell Woelk, Nat Ballou and H. J. Kim. Data Model Issues for Object-Oriented Applications. *ACM TOIS*, 1987.

[Lin93] Lincoln Cesar Medina de Oliveira. Incorporação da Dimensão Temporal em Bancos de Dados Orientados a Objetos. Master's thesis, Universidade Estadual de Campinas, 1993.

[NA87] Shamkant B. Navathe and Rafi Ahmed. TSQL - A language interface for history databases. In *Proceedings of the Conference on Temporal Aspects in Information Systems*, pages 113–128, 1987.

[NA89] Shamkant B. Navathe and Rafi Ahmed. A temporal relational model and a query language. *Information Sciences*, (49):147–175, 1989.

[S. 79] S. Jones, P. Mason and R. Stamper. LEGOL 2.0: A relational specification language for complex rules. *Information Systems*, 4(4):293–305, November 1979.

[SA85a] Richard Snodgrass and I. Ahn. A Taxonomy of Time in Databases. In *Proceedings of the ACM SIGMOD International Conference on Manegement of Data*, pages 236–246, May 1985.

[SA85b] Richard Snodgrass and I. Ahn. Temporal Databases. *IEEE Computer*, pages 35–42, September 1985.

[Sad87] Sadeghi,R., Samson, W.B., Deen, S.M. Hql: A historical query language. *Technical report, Dundee College of Technology*, 1987.

[SC91] Stanley Y. W. Su and Hsin-Hsing M. Chen. A temporal knowledge representation model OSAM*/T and its query language

OQL/T. In *Proceedings of the International Conference on Very Large Data Bases*, pages 431–442, september 1991.

[Sno87] Richard Snodgrass. The Temporal Query Language TQuel. *ACM Transactions on Database Systems*, 12(2):247–298, June 1987.

[Soo91] Michael D. Soo. Bibliography on Temporal Database. *ACM SIGMOD Record*, 20(1):14–23, March 1991.

[TA86] Adullah U. Tansel and M.E. Arkun. HQUEL: A query language for historical relational databases. In *Proceedings of the 3rd International Workshop on Statistical and Scientific Databases*, 1986.

[Tan86] Adullah U. Tansel. Adding time dimension to relational model and extending relational algebra. *Information Systems*, 11(4):343–355, 1986.

[WD92] Gene T. J. Wuu and Umeshwar Dayal. A uniform model for temporal object-oriented databases. In *Proceedings of the International Conference on Data Engineering*, pages 584–593, 1992.

[Wol92] Wolfgang Käfer and Harald Schöning. Realizing a Temporal Complex-Object Data Model. In *Proceedings of the ACM SIGMOD International Conference on Management Data*, pages 266–275, 1992.

Relatórios Técnicos – 1992

92-01 **Applications of Finite Automata Representing Large Vocabularies,** *C. L. Lucchesi, T. Kowaltowski*

92-02 **Point Set Pattern Matching in d-Dimensions,** *P. J. de Rezende, D. T. Lee*

92-03 **On the Irrelevance of Edge Orientations on the Acyclic Directed Two Disjoint Paths Problem,** *C. L. Lucchesi, M. C. M. T. Giglio*

92-04 **A Note on Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams,** *W. Jacometti*

92-05 **An (l, u)-Transversal Theorem for Bipartite Graphs,** *C. L. Lucchesi, D. H. Younger*

92-06 **Implementing Integrity Control in Active Databases,** *C. B. Medeiros, M. J. Andrade*

92-07 **New Experimental Results For Bipartite Matching,** *J. C. Setubal*

92-08 **Maintaining Integrity Constraints across Versions in a Database,** *C. B. Medeiros, G. Jomier, W. Cellary*

92-09 **On Clique-Complete Graphs,** *C. L. Lucchesi, C. P. Mello, J. L. Szwarcfiter*

92-10 **Examples of Informal but Rigorous Correctness Proofs for Tree Traversing Algorithms,** *T. Kowaltowski*

92-11 **Debugging Aids for Statechart-Based Systems,** *V. G. S. Elias, H. Liesenberg*

92-12 **Browsing and Querying in Object-Oriented Databases,** *J. L. de Oliveira, R. de O. Anido*

Relatórios Técnicos – 1993

93-01 **Transforming Statecharts into Reactive Systems,** *Antonio G. Figueiredo Filho, Hans K. E. Liesenberg*

93-02 **The Hierarchical Ring Protocol: An Efficient Scheme for Reading Replicated Data,** *Nabor das C. Mendonça, Ricardo de O. Anido*

93-03 **Matching Algorithms for Bipartite Graphs,** *Herbert A. Baier Saip, Cláudio L. Lucchesi*

93-04 **A lexBFS Algorithm for Proper Interval Graph Recognition,** *Celina M. H. de Figueiredo, João Meidanis, Célia P. de Mello*

93-05 **Sistema Gerenciador de Processamento Cooperativo,** *Ivonne. M. Carrazana, Nelson. C. Machado, Célio. C. Guimarães*

93-06 **Implementação de um Banco de Dados Relacional Dotado de uma Interface Cooperativa,** *Nascif A. Abousalh Neto, Ariadne M. B. R. Carvalho*

93-07 **Estadogramas no Desenvolvimento de Interfaces,** *Fábio N. de Lucena, Hans K. E. Liesenberg*

93-08 **Introspection and Projection in Reasoning about Other Agents,** *Jacques Wainer*

93-09 **Codificação de Seqüências de Imagens com Quantização Vetorial,** *Carlos Antonio Reinaldo Costa, Paulo Lício de Geus*

93-10 **Minimização do Consumo de Energia em um Sistema para Aquisição de Dados Controlado por Microcomputador,** *Paulo Cesar Centoducatte, Nelson Castro Machado*

93-11 **An Implementation Structure for RM-OSI/ISO Transaction Processing Application Contexts,** *Flávio Morais de Assis Silva, Edmundo Roberto Mauro Madeira*

93-12 **Boole's conditions of possible experience and reasoning under uncertainty,** *Pierre Hansen, Brigitte Jaumard, Marcus Poggi de Aragão*

93-13 **Modelling Geographic Information Systems using an Object Oriented Framework,** *Fatima Pires, Claudia Bauzer Medeiros, Ardemiris Barros Silva*

93-14 **Managing Time in Object-Oriented Databases,** *Lincoln M. Oliveira, Claudia Bauzer Medeiros*

93-15 **Using Extended Hierarchical Quorum Consensus to Control Replicated Data: from Traditional Voting to Logical Structures,** *Nabor das Chagas Mendonça, Ricardo de Oliveira Anido*

93-16 $\mathcal{LL}$ **– An Object Oriented Library Language Reference Manual,** *Tomasz Kowaltowski, Evandro Bacarin*

93-17 **Metodologias para Conversão de Esquemas em Sistemas de Bancos de Dados Heterogêneos,** *Ronaldo Lopes de Oliveira, Geovane Cayres Magalhães*

93-18 **Rule Application in GIS – a Case Study,** *Claudia Bauzer Medeiros, Geovane Cayres Magalhães*

93-19 **Modelamento, Simulação e Síntese com VHDL,** *Carlos Geraldo Krüger e Mário Lúcio Côrtes*

93-20 **Reflections on Using Statecharts to Capture Human-Computer Interface Behaviour,** *Fábio Nogueira de Lucena e Hans Liesenberg*

93-21 **Applications of Finite Automata in Debugging Natural Language Vocabularies,** *Tomasz Kowaltowski, Cláudio Leonardo Lucchesi e Jorge Stolfi*

93-22 **Minimization of Binary Automata,** *Tomasz Kowaltowski, Cláudio Leonardo Lucchesi e Jorge Stolfi*

93-23 **Rethinking the DNA Fragment Assembly Problem,** *João Meidanis*

93-24 EGOLib — **Uma Biblioteca Orientada a Objetos Gráficos,** *Eduardo Aguiar Patrocínio, Pedro Jussieu de Rezende*

93-25 **Compreensão de Algoritmos através de Ambientes Dedicados a Animação,** *Rackel Valadares Amorim, Pedro Jussieu de Rezende*

93-26 GeoLab: **An Environment for Development of Algorithms in Computational Geometry,** *Pedro Jussieu de Rezende, Welson R. Jacometti*

93-27 **A Unified Characterization of Chordal, Interval, Indifference and Other Classes of Graphs,** *João Meidanis*

93-28 **Programming Dialogue Control of User Interfaces Using Statecharts,** *Fábio Nogueira de Lucena e Hans Liesenberg*

93-29 EGOLib – **Manual de Referência,** *Eduardo Aguiar Patrocínio e Pedro Jussieu de Rezende*

Relatórios Técnicos – 1994

94-01 **A Statechart Engine to Support Implementations of Complex Behaviour,** *Fábio Nogueira de Lucena, Hans K. E. Liesenberg*

Departamento de Ciência da Computação — IMECC
Caixa Postal 6065
Universidade Estadual de Campinas
13081-970 – Campinas – SP
BRASIL

reltec@dcc.unicamp.br