UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

ALEX MARTINS DE OLIVEIRA

Funcionalidades Temporais do Banco de Dados Oracle e Mecanismos para Consultas de Tempo de Validade.

Trabalho de Conclusão de Mestrado apresentado como requisito parcial para a obtenção do grau de Mestre em Informática

Profa. Dra. Nina Edelweiss
Orientadora

Porto Alegre, novembro de 2003

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Oliveira, Alex Martins de

Normas para Apresentação de Monografias do Instituto de Informática e do PPGC / Alex Martins de Oliveira, - Porto Alegre: Programa de Pós-Graduação em Computação, 2003.

105 f.: il.

Trabalho de Conclusão (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2003. Orientador: Nina Edelweiss.

1. ABNT. 2. Processadores de texto. 3. Formatação eletrônica de documentos. I. Edelweiss, Nina. VI.Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitora: Profª. Wrana Maria Panizzi
Pró-Reitor de Ensino: Prof. José Carlos Ferraz Hennemann
Pró-Reitora Adjunta de Pós-Graduação: Profa. Jocélia Grazia
Diretor do Instituto de Informática: Prof. Philippe Olivier Alexandre Navaux
Coordenador do PPGC: Prof. Carlos Alberto Heuser
Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

SUMÁRIO

LISTA DE ABREVIATURAS

BD	Banco de Dados
BDT	Banco de Dados Temporais
IOT	*Index organized table*
ORDTS	*Time series Schema*
PL/SQL	*Procedural Language/Structured Query Language*
PTV	Pacote de Tempo de Validade
SGBD	Sistema Gerenciador de banco de dados
SQL	*Structured Query Language*
TSC	*Time Series Cartrigdes*
TSQL2	*Temporal Structured Query Language* 2

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Apesar das vantagens das funcionalidades de banco de dados temporais, já amplamente demonstradas na literatura, ainda não existe, comercialmente, um SGBD totalmente temporal. Algumas propostas já foram feitas, embora um pouco discretas, dando ênfase em apenas parte das funcionalidades temporais, já sinalizando que em breve será possível existir um SGBD puramente temporal.

Uma dessas propostas se constitui na implementação de uma camada de software que simula um banco de dados temporal, chamada Pacote de Tempo de Validade – PTV. Ela foi desenvolvida para demonstrar algumas funções temporais de banco de dados de tempo de validade.

Embora o PTV tenha funções para garantir a integridade de dados temporais na inserção de tuplas, além de outros controles, não apresenta funções de consultas temporais. Essas funções foram desenvolvidas neste trabalho com base no TSQL2, aumentando, portanto, as funcionalidades temporais do PTV. Elas foram desenvolvidas para o SGBD *Oracle 9i* e consistem da principal proposta desse trabalho, permitindo inclusive validar as funções de consultas temporais de outras propostas da literatura.

A segunda proposta desse trabalho é prover aos desenvolvedores e pesquisadores dessa área, uma interface visual apropriada para o uso do PTV, permitindo, assim, a exploração máxima dos recursos temporais desse pacote.

Palavras-chave: Banco de dados, Banco de Dados Temporal, Séries de Tempo, Consulta Temporal.

Oracle Database Temporal Functionalities and Mechanism to Valid-time Query

ABSTRACT

In spite of the temporal database advantages, widely demonstrated in the literature, nowadays there is no completely commercial version of a temporal DBMS. Few and discretes proposals have been made, but only with emphasis in part of the possible temporal funcionalities. There are some evidences that, in a near future, a purely temporal DBMS will be available.

One of these temporal database implementation proposals is known as PTV (Valid-Time Package), and was developed to demonstrate some valid-times database temporal functions.

Although presenting functions that insure temporal data integrity during tuple insertion and some other controls, PTV does not present support for temporal queries. Based on TSQL2, this functions were developed in this work, thus increasing the PTV temporal funcionalities. These functions, the main proposal of this work, were developed for the Oracle 9i DBMS, allowing to validate temporal query functions defined in the literature.

The second proposal of this work is to supply an easy-to-use visual interface, oriented to the area programmers and researchers, providing the maximum exploration of this package temporal resources.

Keywords: Data Bases, Temporal Data Bases, Time Series, Temporal Query.

1 INTRODUÇÃO

A maior parte das aplicações atuais têm necessidade de manipular, de alguma maneira, informações históricas – dados relativos a estados passados da aplicação [EDE 98]. São afirmações como essa que demonstram a importância do estudo de banco de dados temporais. Hoje em dia, cada vez mais são exigidas aplicações que tratam de temporalidade. Apesar disso, ainda não existem banco de dados comerciais totalmente temporais [CAV 95], apesar da crescente necessidade desse modelo.

Diversas aplicações têm se mostrado bastante adaptadas a esse tipo de modelagem. Como exemplos, temos: aplicações financeiras, sistemas de reservas, sistemas de suporte a decisão, etc. Outra importante utilização de modelagem temporal é o uso em *Data Warehouse* na utilização da dimensão tempo.

Uma solução para essa questão é a implementação de parte de temporalidade nos Banco de dados comerciais [TIM 99], ou seja, a partir dos recursos do próprio SGBD, são construídos pacotes ou módulos que simulam e realizam de forma transparente aos usuários as funcionalidades temporais.

O objetivo principal desse trabalho é implementar funções de consultas temporais baseadas em *TSQL2* [SNO 95] para o Pacote de Tempo de Validade descrito em [PIN 2003]. Esse trabalho está estruturado da seguinte forma:

Inicialmente são mostrados conceitos teóricos sobre banco de dados temporais e suas diversas estratégias de implementação e modelagem. No capítulo seguinte, é feita uma comparação dos aspectos temporais entre as versões 8i e 9i do banco de dados comercial *Oracle*. Em seguida, é apresentado o Pacote de Tempo de Validade – PTV – dissertação de mestrado desenvolvido em [PIN 2003] que implementa do Tempo de Validade no SGBD *Oracle*.

Nos capítulos a seguir são apresentadas as funções de consultas temporais desenvolvidas neste trabalho para o PTV, bem como uma interface *front-end* para manipulação das funções do Pacote de Tempo de Validade. Finalmente é descrito um estudo de caso para as funções de consulta temporais desenvolvidas.

No final desse trabalho são apresentadas as conclusões, bem como a avaliação deste trabalho. Ainda neste capítulo são feitas sugestões de trabalhos futuros.

2 BANCO DE DADOS TEMPORAIS

Uma das principais características de banco de dados não-temporais é a instantaneidade de seus estados. Eles suportam operações de modificação que facilitam a transição de um estado para outro, substituindo dados velhos por novos. Nas consultas, é assumido que seus valores são válidos para aquele momento.

Por outro lado, existem diversas aplicações que necessitam, não somente do estado atual dos seus dados, mas de momentos ou instantes anteriores. Os problemas vêm desde como definir uma estratégia para adicionar períodos de tempos válidos até como definir consultas e verificação de integridade.

Os estudos sobre Bancos de Dados temporais buscam resolver esses problemas. Diferentemente dos Bancos de Dados convencionais, os Bancos de Dados temporais permitem armazenar e recuperar todos os estados dos objetos ao longo do tempo [EDE 94].

[EDE 94] classifica os bancos de dados em quatro tipos diferentes: Bancos de Dados instantâneos, de tempo de transição, de tempo de validade e bitemporais. Nas seções seguintes esses tipos serão mostrados com mais detalhes. Outros conceitos importantes, como o tipo de dados *TimeStamp,* serão detalhado nas seções seguintes.

Além dos tipos de bancos de dados mencionados anteriormente, outro importante conceito utilizado no desenvolvimento desse trabalho é o TSQL2, que consiste em extensão temporal da linguagem padrão SQL-92, para manipular dados temporais em banco de dados relacionais [SNO 95].

2.1 Banco de dados Instantâneo

Bancos de Dados instantâneos têm como característica principal a capacidade de guardar somente momentos instantâneos de suas informações. Em bancos de dados convencionais, na medida em que suas informações são alteradas ou substituídas, elas não podem ser mais recuperadas, pois não é possível manter histórico de suas informações. Existe uma pequena exceção nesses bancos de dados com relação ao histórico, mas que corresponde na verdade a um procedimento administrativo. Em uma operação de mudança de valores seus estados são perdidos, a menos do registro de transações anteriores feito para permitir a recuperação de erros de processamento (“logs”) [EDE 94].

A figura 2.1 [EDE 94] apresenta mudanças de estados em um banco de dados instantâneos. Esses instantes são representados por inclusão de atributos ou alterações feitas por programas aplicativos.

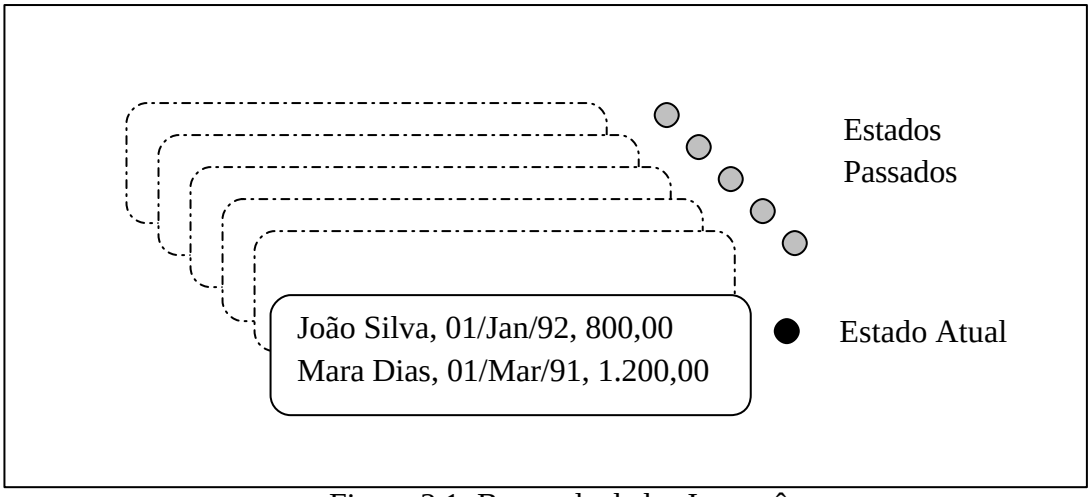


Figura 2.1: Banco de dados Instantâneo.

2.2 Banco de dados de Tempo de Transação

Nesse tipo de banco de dados para cada informação é associado um rótulo de tempo (*TimeStamp*), fornecido automaticamente pelo próprio SGBD (Sistema Gerenciador de banco de dados). Agora uma informação não é mais substituída por outra, pois a cada alteração é feita uma operação de inserção do novo valor, com um rótulo temporal.

O conceito de estado atual é definido pelos tempos de transações mais recentes para cada propriedade. Nesse modelo temporal a validade da informação inicia no momento da sua introdução no banco de dados.

A figura 2.2 [EDE 94] ilustra a manutenção em um banco de dados de tempo de transação.

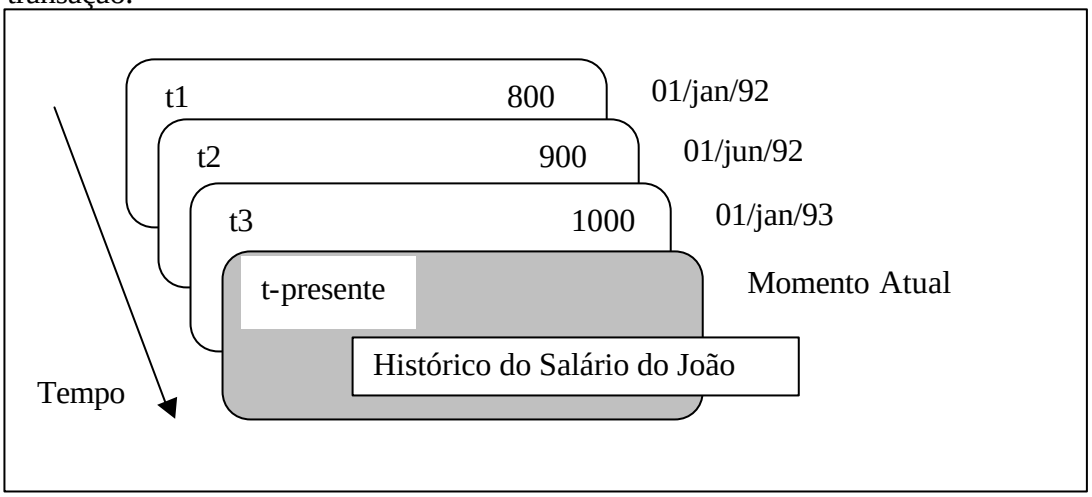


Figura 2.2: Banco de dados de Tempo de Transação.

O tempo de validade da informação pode não coincidir com o tempo de transação. Para este caso, coloca-se explicitamente um atributo para a validade. O funcionário João teve seu salário alterado para 900, referente a 01/Jun/92. Essa alteração pode ter sido feita no dia 03 de junho de 1992. Internamente no banco de dados, a data de transação é 03/Jun/92 e a validade da informação de 01/Jun/92.

2.3 Banco de dados de Tempo de Validade

Esse modelo de banco de dados tem uma certa semelhança com o modelo de tempo de transação. A diferença básica é que nesse modelo o tempo de validade é definido independente do tempo em que a informação é introduzida no banco de dados. O tempo de validade é fornecido pelo usuário e o tempo de transação é definido automaticamente pelo SGBD. Nesse modelo o tempo de transação não é armazenado. Uma importante característica desse modelo é a capacidade de alteração ou correções de informações do passado.

A figura 2.3 [EDE 94] mostra um modelo de banco de dados de tempo de validade. Nesse exemplo é mostrada a evolução de salário para o funcionário João. Seu salário em um momento t1 vale 800. No momento t2 vale 900 e no tempo t3 é igual a 1000. Com isso é possível verificar a evolução no tempo.

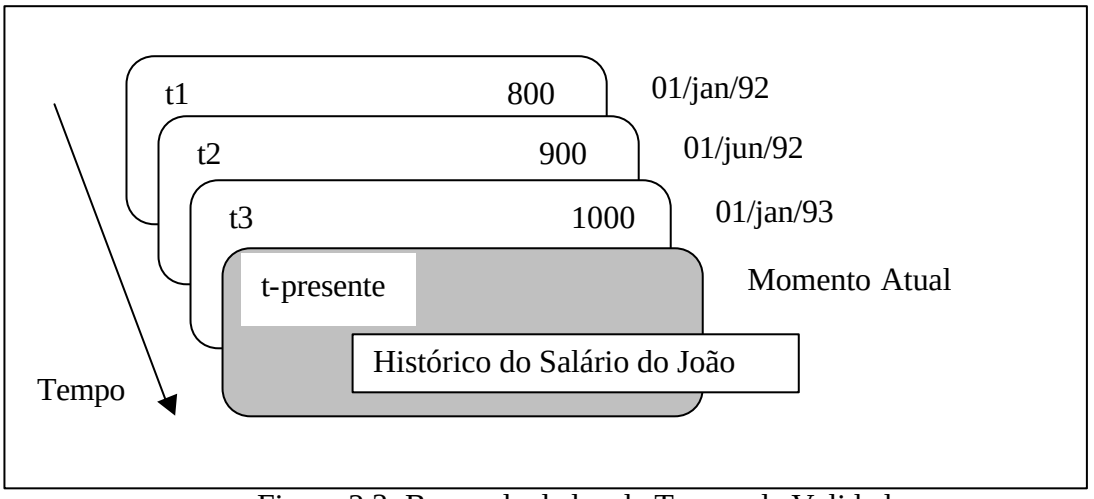


Figura 2.3: Banco de dados de Tempo de Validade.

2.4 Banco de dados Bitemporal

O modelo bitemporal incorpora as características dos modelos de Tempo de Transação e Tempo de Validade. Todo histórico do banco de dados fica armazenado e é possível ter acesso a todos os estados do banco. Através da característica de tempo de validade é possível, por exemplo, registrar valores e, como mencionado anteriormente, alterar valores do passado. A figura 2.4 [EDE 94] mostra um exemplo do modelo bitemporal.

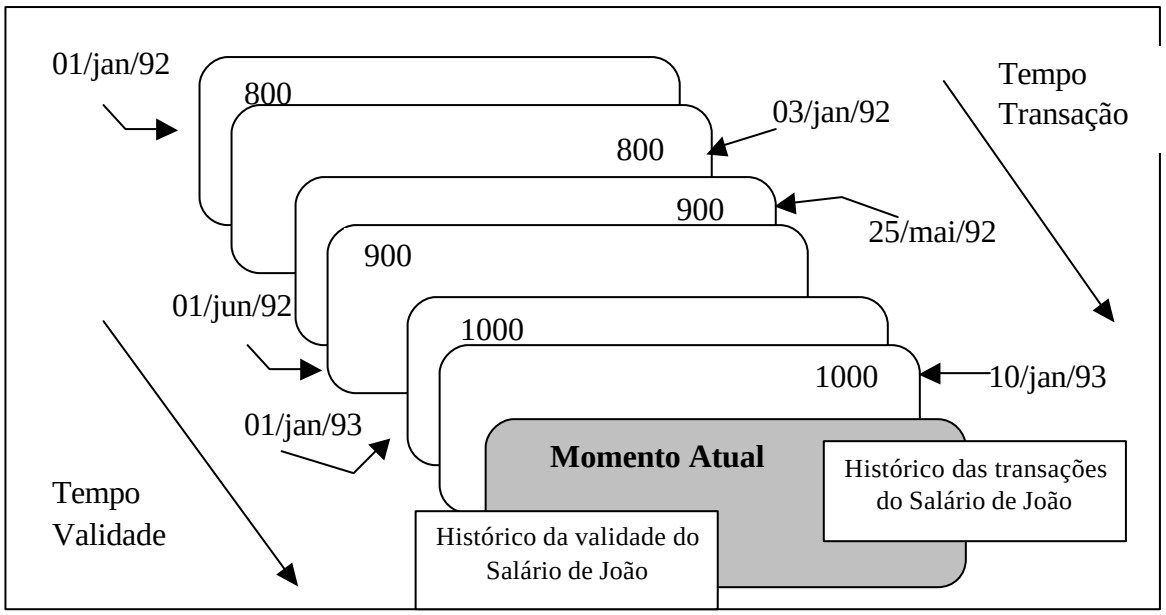


Figura 2.4: Banco de dados Bitemporal.

2.5 Linguagem de Manipulação de Dados Temporais *TSQL2*

O TSQL2 é uma linguagem de consulta temporal, que tem por objetivo a consulta e a manipulação dos dados que variam de acordo com o tempo. Estes dados são armazenados em bases de dados relacionais [SNO 95].

TSQL2 suporta três linhas de tempo: tempo definido pelo usuário, tempo válido e tempo de transação. No modelo TSQL2, um ponto na linha de tempo é chamado de instante e um tempo entre dois instantes é chamado de um período de tempo. Um período é a descrição única de dois instantes: um instante inicial e um instante final. Uma característica importante de um instante é que ele tem duração zero. O tempo é armazenado em estruturas de dados denominados rótulos temporais (*timestamps*).

Um conceito bastante importante é o *chronon* [JEN 98], que define a menor escala de tempo suportada pelo TSQL2. Ele pode ser expresso por qualquer duração, por exemplo: nanosegundos, anos, minutos, etc. Através da escolha apropriada de uma escala, um usuário pode ver a linha do tempo como um conjunto discreto de segundos, dias ou anos.

O *chronon* pode ser representado por um rótulo temporal (*timestamp*) que é utilizado para modelar um instante. O *timestamp* de período também contém uma escala associada (dia, mês, ano, etc.). Ele é composto por dois timestamps de instante [SNO 95].

2.5.1 Funções TSQL2

Em [SNO 95] são definidas funções para a criação de uma base relacional com atributos temporais. A tabela 2.1 mostra as funções para conversão de dados.

O conjunto de funções criadas pode ser classificado em extrator de dados, construtor de dados e funções variadas. O primeiro conjunto de funções pode ser exemplificado pelas funções *BEGIN* e *END*, pois as mesmas desmembram um período, retornando apenas um instante (*timestamp*). O segundo conjunto de funções, chamados construtores de dados, representados pelas funções *PERIOD*, *INTERSECT* e *INTERVAL*, constroem novos valores de dados a partir de seus argumentos. Finalmente, exemplos das funções variadas são *FIRST* e *LAST*, que aceitam dois pontos do tempo como argumento.

Tabela 2.1: Funções Construídas para Conversão de Dados.

Funções	Interpretação
datetime ✍ BEGIN(period)	Retorna o ponto no tempo que representa o início do período.
Datetime ✍ END(period)	Retorna o ponto no tempo que representa o fim do período.
period ✍ PERIOD(datetime1, datetime2)	Retorna o período válido obtido através da diferença de datetime1 - datetime2, diferente de nulo.
period ✍ INTERSECT(period1, period2)	Retorna a interseção do period1 sobre o period2, e esta sobreposição deve ser diferente de nulo.
Interval ✍ ABSOLUTE(interval)	Retorna um valor absoluto de um intervalo
datetime ✍ FIRST(datetime1, datetime2)	Retorna datetime1 se este for menor ou igual a datetime2, caso contrário, retorna datetime2.
datetime ✍ LAST(datetime1, datetime2)	Retorna datetime1 se este for maior ou igual a datetime2, caso contrário, retorna datetime2.

As funções de agregação *COUNT* (número de escalares em uma coluna), *SUM* (soma dos escalares em uma coluna), *AVG* (média dos escalares em uma coluna), *MAX* (maior escalar em uma coluna) e *MIN* (menor escalar em uma coluna) tiveram suas semânticas estendidas para acomodar tipos de dados temporais. A tabela 2.2 [SNO 95], apresenta as funções de agregação relacionadas com os tipos de dados temporais.

Tabela 2.2: Funções de Agregação

Função de Agregação	***Ponto no tempo***	***Período***	***Intervalo***
COUNT	✍	✍	✍
SUM			✍
AVG	✍	✍	✍
MAX	✍	✍	✍
MIN	✍	✍	✍

O símbolo "✍" indica quais funções de agregação são definidas para quais tipos de dados temporais. O TSQL2 implementa outras funções temporais como *scale*, *nobind*, *union*, etc, entretanto por questão de relevância, apenas as funções apresentadas nas tabelas 1 e 2 serão desenvolvidas nesse trabalho. A exceção é a função *absolute* que não será implementada por utilizar um tipo de dados não definido no PTV - o *interval*.

3 FUNÇÕES TEMPORAIS DO ORACLE 8 X ORACLE 9I

Esse capítulo aborda as funcionalidades temporais de SGBD Oracle versões 8 e 9i. Será mostrado que no Oracle 8, as funções temporais estão agrupadas em um pacote chamado Cartucho de Séries de Tempo (Time Series Cartridge). Na versão 9i, a Oracle, fabricante do SGBD, optou por incorporar a maioria das funções do Time Series diretamente nas sentenças SQL, tendo, com isso, ganhos em performance e simplificação no uso.

3.1 *Oracle 8i Time Series Cartridge*

No Oracle 8, a modelagem temporal é feita através de um pacote chamado *Oracle* 8 *Time Series Cartridge* ou Cartucho de Séries de Tempo que é formado por um conjunto de procedimentos, funções e objetos com características próprias, que foram adaptados ao modelo relacional. Por exemplo, as aplicações podem usar o *Time Series* para processar dados históricos de transações o mercado financeiro, como no comércio de ações [ORA 99].

Com o *Time Series Cartridge* é possível usar ou adaptar tabelas para aplicações que envolvam séries de tempo ou criar novas tabelas, bem como estender as funcionalidades do *Cartridge* adicionando ou modificando suas funções e criar *calendários* customizados. A tabela 3.3 mostra a arquitetura do *Time Series* e como esse pacote interage com o restante do SGBD.

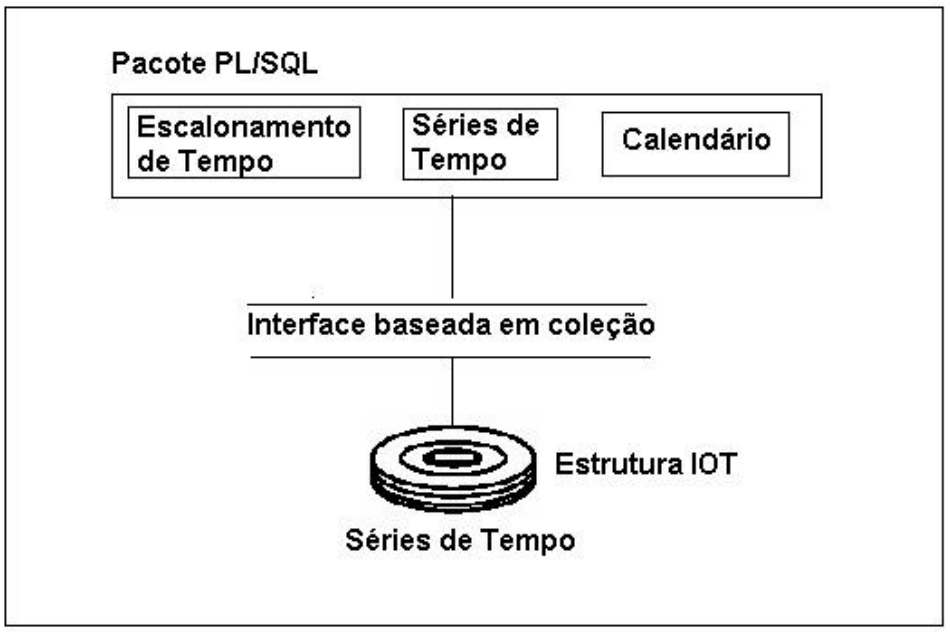


Figura 3.5: Arquitetura do *Time Series Cartrige*.

O nível mais baixo é representado por uma tabela organizada por índices (IOT - *Index Organized Tabl*e), uma tabela plana (*flat table*) ou uma tabela aninhada (*nested* IOT). A diferença entre as três tabelas é como os dados são estruturados. Em seguida existe um nível de interface responsável pela ligação entre a camada mais baixa (armazenamento) e o nível superior. Através dessa ligação, as funções de escalonamento temporal (*time scaling functions*) inserem e recuperam dados de séries de tempo. A interface pressupõe que os dados de séries temporais são armazenados em tabelas relacionais, que os calendários são armazenados em tabelas de objetos e que calendários e dados de séries de tempo sejam tratados como objetos [ORA 99]. A camada superior consiste em pacotes de código em PL/SQL, que determinam os calendários, as séries de tempo, etc.

A estrutura do *Time Series Cartridge* consiste de um mecanismo de controle de integridade através de gatilhos em operações de inserção, remoção, etc, e de um conjunto de tabelas e visões responsáveis pelo armazenamento e pela consistência dos dados. A Figura 3.6, apresentada em [PIN 2003], mostra as estruturas de dados do *Time Series*, para a tabela de funcionários. As descrições das estruturas mencionadas são as seguintes:

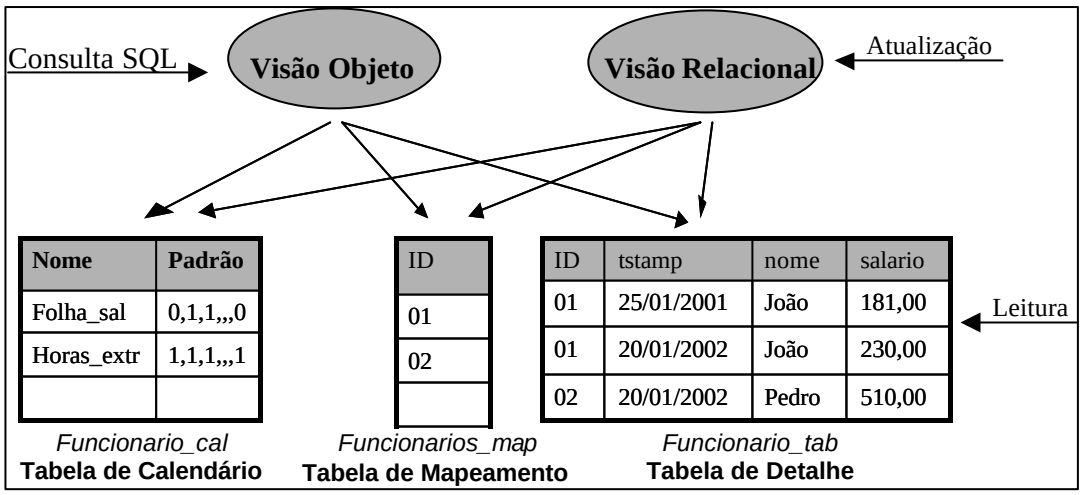


Figura 3.6: Estrutura de Dados do *Time Series*.

- **visão objeto (funcionário)** é um objeto sobre o qual serão manipulados os dados. Pode ser usada somente para acessos de leitura usando as funções de tempo do TSC;
- **visão relacional (funcionario_rvw)** é uma visão relacional responsável pela integridade dos dados, onde operações de inserção, modificação e remoção são aplicadas e controladas;
- **tabela funcionario_tab** é a tabela que contém os dados (tuplas) propriamente ditos;
- **tabela funcionario_map** armazena o identificador da tupla, que neste caso pode ser o código do funcionário;
- **tabela funcionario_cal** armazena informações sobre os padrões do calendário, se este existir (maiores detalhes na seção 3.2).

Um importante elemento do *Time Series* é o calendário que permitem vincular as informações a datas ou a períodos de tempo definidos. Esse permite ao desenvolvedor criar aplicações com características temporais, vinculando informações a datas [ORA 99]. A modelagem com o *Time Series* pode ser feita com ou sem o uso de calendários. Quando a modelagem utiliza calendário, ela é chamada de regular. A modelagem dita irregular não utiliza de calendários. O Oracle também disponibiliza técnicas objeto-relacional, que consistem em usar o paradigma de orientação a objetos sobre o modelo relacional.

A Figura 3.7 [PIN 2003] mostra um exemplo da utilização de calendários para a tabela de Funcionários.

Um calendário é formado por cinco elementos. O primeiro é o nome do calendário seguido do valor “0”(tipo), indicando que é um calendário baseado em exceção.

O segundo elemento é a freqüência, que consiste em definir a granularidade da representação do calendário. Existem diversos níveis para essa granularidade. Em [ORA 99] são mostrados diversos tipos, como minuto, segundo, etc.

O terceiro elemento de um calendário é o padrão no qual são definidos os valores válidos para o rótulo temporal A representação é feita através de um ou mais zeros (0) ou números positivos. O número ‘zero’ especifica rótulos temporais inválidos e o número ‘um’ especifica valores válidos.

O tempo de validade é o próximo elemento do calendário. Através dele, é possível definir valores limites para rótulos temporais. Esse tempo é representado por um valor inicial e um valor final.

Finalmente, o quinto e último elemento de um calendário é a exceção, utilizada para representar, como o próprio nome sugere, valores que estão fora do padrão. O uso de exceções em calendários é facultativo, mas, quando utilizadas, devem ser declaradas ordenadas de forma crescente no tempo.

```
INSERT INTO funcionario_cal VALUES (
ORDSYS.ORDTCalendar(
0,-- tipo do Calendario (0 = Standard)
'folha_salarial',       -- nome do Calendário
4,-- freqüência para o Dia
ORDSYS.ORDTPattern (-- definição do Padrão
(exigido)
ORDSYS.ORDTPatternBits(0,1,1,1,1,1,0),
TO_DATE('15/08/2001','DD/MM/YYYY') ),
'01/01/2001','01/01/2003', -- rótulos
temporais limitantes
NULL, NULL ); -- Semfinal e exceções
```

Figura 3.7: Exemplo de um calendário.

3.2 *Oracle 9i*

O *Time Series* tem sido utilizado como estratégia no uso das funcionalidades temporais no SGBD Oracle. O *Time Series* foi absorvido dentro do próprio SQL nas versões mais recentes desse SGBD. Este capítulo mostrará a estratégia da Oracle na implementação dessas funções.

As funcionalidades do Time Series (ORDTS), foram incorporadas no SQL, depois da *release* 8.1.7 do Oracle 8, incluindo, portanto, a versão 9i. As classes de funções temporais foram implementadas diretamente no SQL, entre as quais se destacam [ORA 2003]:

- funções acumulativas;
- funções de *ranking*;
- funções *LEAD*/*LAG*;
- funções de Inteligência Empresarial.

Embora disponível dentro do pacote do *Time Series*, os objetos calendários (*Calendars*), não estarão disponíveis no SQL. Com essa mudança de implementação, é possível destacar algumas diferenças entre as duas estratégias no uso de funcionalidades temporais. Estas funções serão chamadas das instruções SQL e implementadas em *frameworks* através de PL/SQL ou objetos Oracle, eliminando assim a necessidade do *Time Series*. Essa modificação traz benefícios para as funcionalidades temporais, como por exemplo, maior performance e maior integração com ferramentas de desenvolvimento e banco de dados [ORA 2003].

Tabela 3.3: Diferenças entre o *Time Series* e SQL.

Funções	**ORDTS**	**SQL**
Múltiplos símbolos qualificadores	Habilitado somente em extensões para a tabela *map*	Suportado diretamente
Suporte a frações de segundos	Não suportado	Suportado pelo tipo de dados *Timestamp*
Funções Janelas sobre expressões	Não suportada	Suportada
Classificação de data segundo tamanho	Não suportada	Suportada

Apenas para ilustrar, na Tabela 3.3 há um resumo das diferenças entre as implementações do *Time Series* (ORDTS) e funcionalidades temporais do Oracle 9i (funções analíticas SQL).

Outro aspecto importante no que tange à diferença entre o *Time Series* e o SQL é a sintaxe entre as duas implementações. A tabela 4, disponível em [ORA 2000], dá uma idéia das mudanças na programação das funções de tempo.

Tabela 3.4: Diferença de sintaxe do *Time Series* e do SQL.

	Time Series	**SQL**
Funções de tempo	Referenciadas na cláusula *Table*	Referenciada na lista *Select*
Dados do *Time Series*	Referenciado através de *Views*	Referenciado diretamente no nome da Tabela
Data Inicial e Final	Parâmetro para funções	Referenciado na cláusula *WHERE*
Tamanho da Janela	Parâmetro para funções	Referenciado na cláusula *OVER*

Funções do ORDTS são referenciadas na cláusula *TABLE* enquanto que as funções SQL são referenciadas na cláusula *SELECT*. As consultas do ORDTS requerem que os dados do *Time Series* sejam referenciados através de visões (*VIEWS*). Informações de datas, no ORDTS são parâmetros das funções, enquanto que no SQL elas são referenciadas na cláusula *WHERE*. Por fim, as funções janelas são passadas para o ORDTS através de parâmetros das funções *Time Series*, enquanto que no SQL elas são referenciadas na cláusula *OVER* das funções analíticas [ORA 2000].

A seguir serão mostrados exemplos equivalentes entre funções do ORDTS e de funções SQL. O objetivo é mostrar tecnicamente as informações mencionadas anteriormente. Não serão abordados aspectos de desempenho entre as duas implementações Esses exemplos serão mostrados em tópicos específicos para conjuntos ou famílias de funções.

3.2.1 Funções *Lead* e *Lag*

As funções *Lead* e *Lag* têm funções específicas no *Time Series.* Elas são utilizadas para adiantar ou atrasar unidades de tempo, que refletem a freqüência do calendário do *Time Series*. A Figura 3.8 mostra um exemplo do uso da função LAG sendo utilizada no *Time Series* [ORA 2000].

```
SELECT T.tstamp, T.value
FROM tsdev.stockdemo_ts ts,
      TABLE (CAST(ORDSYS.TimeSeries.ExtractTable(
            ORDSYS.TimeSeries.Lag(ts.open, 1,
            to date('11-OCT-99', 'DD-MON-YY'),
            to_date('15-OCT-99', 'DD-MON-YY'))
            ) AS ORDSYS.ORDTNumTab)) T
WHERE ticker = 'ACME';
```

Figura 3.8: Função Lag no ORDTS.

O exemplo da Figura 3.9 abaixo mostra o mesmo exemplo da Figura 3.8, só que agora utilizando funções SQL.

```
1   SELECT LAG(tstamp,1) OVER (ORDER BY tstamp) as tstamp, open
2   FROM tsdsev.stockdemo_tab
3   WHERE ticker = 'ACME'
4         AND tstamp >= to_date('11-OCT-99', 'DD-MON-YY')
5         AND tstamp <= to_date('15-OCT-99', 'DD-MON-YY');
```

Figura 3.9: Função Lag no SQL.

É possível verificar que no exemplo da Figura 3.8, a função do *Time Series Lag* é referenciada na cláusula *TABLE* (linha 4), enquanto que no SQL, Figura 3.9, a função *Lag* é referenciada na cláusula *SELECT* (linha 1). Outros aspectos de diferenças de sintaxe nos dois exemplos são:

- ?? Dados *Time Series*
  - o ORDTS (linha 2): uma visão *Time Series* (*tsdev*.*stockdeo_ts*)
  - o SQL (linha 2): a tabela *tsdev*.*stockdeo_tab*
- ?? Datas de entrada
  - o ORDTS (linhas 4-6): parâmetro para a função *Time Series*
  - o SQL (linhas 3-5): especificado na cláusula *WHERE*

3.2.2 Funções Acumulativas

As funções de seqüência acumulativas (Média, Máximo, Mínimo, Produto e Soma) calculam um resultado sobre uma janela de registros. Por exemplo, uma seqüência acumulativa sobre 100 elementos deveria emitir um resultado com base nos 100 registros

definidos na cláusula de critério das funções. No caso do ORDTS essa informação é passada como parâmetro da função na cláusula TABLE. A figura 3.10 mostra um exemplo do uso de funções acumulativas no ORDTS, o qual se calcula a média (AVG) a partir de um determinado período para o *Time Series* ACME[ORA 202000].

```
1   SELECT T.tstamp, T.value
2         FROM tsdev.stockdemo_ts ts ts,
3               TABLE (CAST(ORDSYS.TimeSeries.ExtractTable(
4                     ORDSYS.TimeSeries.Cavg(ts.open,
5                           to_date('01-DEC-94', 'DD-MON-YY'),
6                           to_date('15-DEC-94', 'DD-MON-YY'))
7                     ) AS ORDSYS.ORDTNumTab)) T
8         WHERE ticker = 'ACME';
```

Figura 3.10: Cálculo de Média no *Time Series*.

A Figura 3.11 [ORA 2000] mostra o mesmo exemplo do cálculo de média (Avg), usando desta vez funções SQL.

```
1   SELECT tstamp,
2               AVG(open) OVER (ORDER BY tstamp
3                           ROWS UNBOUNDED PRECEDING) as value
4         FROM tsdev.stockdemo_tab
5         WHERE ticker = 'ACME'
6               AND tstamp >= to_date('01-DEC-94', 'DD-MON-YY')
7               AND tstamp <= to_date('15-DEC-94', 'DD-MON-YY');
```

Figura 3.11: Cálculo de Média no SQL.

3.2.3 Funções de Janela Móveis

A função de média móvel (Magv) retorna uma série de tempo que contém a média dos valores de cada rótulo temporal sucessivo, para um especificado intervalo sobre uma faixa de datas definida. Esse mesmo raciocínio é utilizado para a função de soma móvel (Msum).

As figuras 3.12 e 3.13 mostram o mesmo exemplo no uso de funções móveis para o ORDTS e SQL, respectivamente. Os dois exemplos utilizam a função Mavg [ORA 2000] para o mesmo intervalo de tempo.

```
1   SELECT T.tstamp, T.value
2   FROM tsdev.stockdemo_ts ts,
3         TABLE (CAST(ORDSYS.TimeSeries.ExtractTable(
4               ORDSYS.TimeSeries.Mavg(ts.open, 5,
5                     to_date('08-OCT-99', 'DD-MON-YY'),
6                     to_date('15-OCT-99', 'DD-MON-YY'))
7               ) AS ORDSYS.ORDTNumTab)) T
8    WHERE ticker = 'ACME';
```

Figura 3.12: Função Magv para o ORDTS.

```
1   SELECT tstamp,
2         AVG(open) OVER (ORDER BY tstamp ROWS 4 PRECEDING) as
    value
3   FROM tsdev.stockdemo_tab
4   WHERE ticker = 'ACME'
5         AND tstamp >= to_date('08-OCT-99','DD-MON-YY')
6         AND tstamp <= to date('15-OCT-99','DD-MON-YY');
```

Figura 3.13: Função Magv para o SQL

No exemplo acima, o tamanho cinco da janela no ORDTS corresponde no SQL à sintaxe ROWS 4 PRECEDING, que define uma janela de tamanho 5, corresponde ao registro atual mais quatro linhas anteriores.

ORDTS e SQL têm adotado diferentes convenções para manipulação das k-1 linhas, onde k é o tamanho da janela. No ORDTS é feito uma média entre as k-éssimos últimas tuplas. Na tabela 3.5 o valor 6 do ORDTS é uma média das linhas 1 a 5, o valor 8, das linhas 2 a 6 e assim por diante. No SQL, esse cálculo é feito de uma forma diferente. Com uma janela de tamanho 5, são feitos sub-totais parciais a partir de uma janela de tamanho 1, começando na data inicial sendo incrementado em uma unidade a cada iteração até o tamanho 5, não utilizando os valores fora do intervalo mencionado. No exemplo é possível perceber que o valor 10 do SQL (linha 5) é uma média dela mesma (janela tamanho 1). Para o valor 11, próxima iteração, a média tem janela de tamanho 2 (linhas 5 e 6). Essa regra vai até o tamanho de janela igual a 5.

Tabela 3.5: Comparação entre ORDTS e SQL.

Linha	DATA	VALOR	ORDTS (Mavg)	SQL(Mavg)
1	04/10/99	2		
2	05/10/99	4		
3	06/10/99	6		
4	07/10/99	8		
5	08/10/99	10	6	10
6	11/10/99	12	8	11
7	12/10/99	14	10	12
8	13/10/99	16	12	13
9	14/10/99	18	14	14
10	15/10/99	20	16	16

Na Tabela 3.5 há um exemplo que compara as diferentes convenções utilizadas. Ainda neste exemplo, são mostrados cinco períodos de médias móveis, iniciados em 08/10/99.

Para que o SQL tenha a mesma saída do ORDTS é preciso fazer uma alteração, uma vez que as implementações são diferentes. A Figura 3.14 [ORA 2000] mostra a sentença SQL que deve ser utilizada para ter a mesma saída de dados do exemplo da tabela 12.

```
1   SELECT tstamp, value
2   FROM
3         (SELECT tstamp,
4               AVG(open) OVER (ORDER BY tstamp ROWS 4 PRECEDING) as
    value
5               FROM tsdev.stockdemo_tab ts
6               WHERE ticker = 'ACME'
7                     AND tstamp >= to_date('04-OCT-99','DD-MON-YY')
8                     AND tstamp <= to_date('15-OCT-99','DD-MON-YY'))
9   WHERE tstamp >= to_date('08-OCT-99','DD-MON-YY');
```

Figura 3.14: Sentença SQL adaptada.

A tabela 3.6 mostra um sumário das sintaxe no SQL para as três categorias de funções.

Tabela 3.6: Resumo das sintaxes do SQL

Categoria da Função	Sintaxe SQL
Deslocamento	SELECT LAG(tstamp,1) OVER (ORDER BY tstamp)
Seqüência Acumulativa	SELECT AVG(open) OVER (ORDER BY tstamp ROWS UNBOUNDED PRECEDING)
Janela Móvel	SELECT AVG(open) OVER (ORDER BY tstamp ROWS 4 PRECEDING)

3.2.4 Comparação de performance

Em [ORA 2000] são descritos testes de desempenho de performance entre o ORDTS e as funções analíticas do SQL.8.1.6. Portanto, não é o *Oracle* 9i, embora ele implemente as mesmas funções SQL mencionadas.

Os testes realizados com as funções *Lag*, média acumulativa e média móvel são executados com 140.000 registros para os dados de séries de tempo. Para média móvel são realizados testes adicionais com 1.000.000 de registros de séries de tempo. Todos esses testes, exceto o de 1.000.000 de registros, foram realizados em um processador dual (200 megahertz) Sun Ultra 2 com 1 Gigabyte de memória RAM. Os tempos são registrados em segundos [ORA 2000].

Para a função *Lag*, com dados de séries de tempo de um período, a função SQL teve um desempenho por um fator acima de 40. A tabela 3.7 [ORA 2000] mostra os resultados:

Tabela 3.7: Resultado no uso da Função Lag.

Quant. De Período	ORDTS	SQL	Desempenho
1	1909	40	47,7

As tabelas 3.8, 3.9 e 3.10 [ORA 2000] resumem os resultados para os cálculos realizados com as funções de média acumulativa e média móvel (com 140.000 e 1.000.000 de registros).

Tabela 3.8: Resultado no uso da Função Média Acumulativa.

ORDTS	SQL	Desempenho
100	63	1,6

A média acumulativa, realizada com 140.000 registros, mostra um fator de desempenho de 1,6 sobre o ORDTS.

Muitos fatores influenciam a performance da função de média móvel.

Tabela 3.9: Resultado no uso da Função Média Móvel para 140.000 Registros.

Tamanho da Janela	ORDTS	SQL	Desempenho
10	107	38	2,8
100	107	40	2,7
1000	109	160	0,7
50000	102	153	0,7

Para pequenas janelas móveis, as funções SQL têm um desempenho melhor. Entretanto, para situações em que o tamanho da janela é superior a 100, o ORDTS tem um desempenho melhor.

Tabela 3.10: Resultado no uso da Função Média Móvel para 1.000.000 tuplas.

Tamanho da Janela	ORDTS	SQL	Desempenho
100	3017	197	18

Com 1.000.000 de registros o teste foi realizado com um computador com a seguinte configuração: processador dual (300 megahertz), Sun Ultra 2 com 256 megabytes de memória RAM. O resultado mostrado é que neste caso, as funções SQL superam em 18 vezes a função ORDTS.

Na maioria dos casos, as funções analíticas do SQL tiveram um desempenho superior bastante significativo em relação às funções ORDTS. Na verdade, esses resultados já eram esperados uma vez que as funções do *Time Series* são implementadas do pacote PL/SQL, enquanto que as funções analíticas do SQL estão embutidas diretamente do *kernel* do SGBD.

Nesta seção foram mostradas diversas diferenças entre o ORDTS e as funções analíticas do SQL. Foi mostrado também como re-escrever algumas famílias ou conjunto de funções do ORDTS para seus respectivos equivalentes ao SQL. Por fim, foram realizados testes de desempenho entre as duas implementações.

As funções analíticas do SQL mostraram ter muitas vantagens sobre as funções do ORDTS, incluindo funcionalidades avançadas, melhor performance e melhor integração com os novos tipos de dados do SGBD Oracle [ORA 2000].

4 PACOTE DE TEMPO DE VALIDADE (PTV)

O Pacote de Tempo de Validade (PTV) é uma proposta de implementação de uma camada de software sobre o SGBD Oracle. Consiste de funções, procedimentos e gatilhos que fornecem recursos para inserção e administração de tempo de validade [PIN 2003].

Neste capítulo serão abordados detalhes da arquitetura, funcionalidades e implementações do PTV, bem como uma comparação em relação ao pacote *Time Series Cartridge* do Oracle.

Como mencionado anteriormente, o PTV é um pacote que implementa tempo de validade no SGBD Oracle. Desta forma, a representação temporal do PTV é feita através de um intervalo de tempo, diferentemente do *Time Series* que implementa o tempo de transação, representado por um rótulo temporal (*timestamp*). O intervalo de tempo que representa o tempo de validade no PTV foi padronizado através da utilização de dois atributos do tipo data: *TvalidadeI* que representa a validade inicial para um determinado registro e *TvalidadeF* que representa a validade final. Através do esquema do banco de dados não é possível identificar quais tabelas são temporais. Para tanto, é necessário visualizar a estrutura da tabela.

cpf	idFunção	nome	salario	Data
66062244074	5	João	381,00	25/01/2001
66062244074	4	João	430,00	25/07/2001
70062244070	5	Pedro	381,00	27/01/2001
81162244071	3	Maria	200,00	25/01/2001

A

id	tstamp	TvalidadeI	TvalidadeF	Nome	salario	idFunção
66062244074	25/01/2001	27/01/2001	24/07/2001	João	381,00	5
66062244074	25/07/2001	25/07/2001	Null	João	430,00	4
70062244070	25/01/2001	27/01/2001	Null	Pedro	381,00	5
81162244071	25/07/2001	25/07/2001	25/08/2001	Maria	200,00	3

B

Figura 4.15: Execução do *Time Series* e do PTV em tabela atemporal

O PTV funciona independentemente do *Time Serie*, uma vez que implementam funcionalidades temporais distintas. A figura 4.15 [PIN 2003] mostra uma tabela atemporal – "A" - que foi transformada em uma tabela temporal após a execução do *Time Series* e do PTV – "B".

4.1 Arquitetura do PTV

Um dos principais objetivos do PTV é a consistência de seus registros. Essa funcionalidade é utilizada sempre que se faz alguma operação de inserção, alteração ou remoção de dados temporais. Para um melhor entendimento, na figura 4.16 [PIN 2003] é mostrada uma estrutura que representa a arquitetura do PTV. Semelhante ao *Time Series*, o PTV utiliza recursos como gatilhos, funções e objetos. Os principais módulos do PTV são:

i) mecanismo de consistência, responsável pelo controle de entrada de dados,

ii) mecanismo de controle, responsável pela ativação, inserção e desativação do tempo de validade no banco de dados e

iii) ferramenta de consulta que objetiva facilitar a recuperação de dados temporais [PIN 2003]:

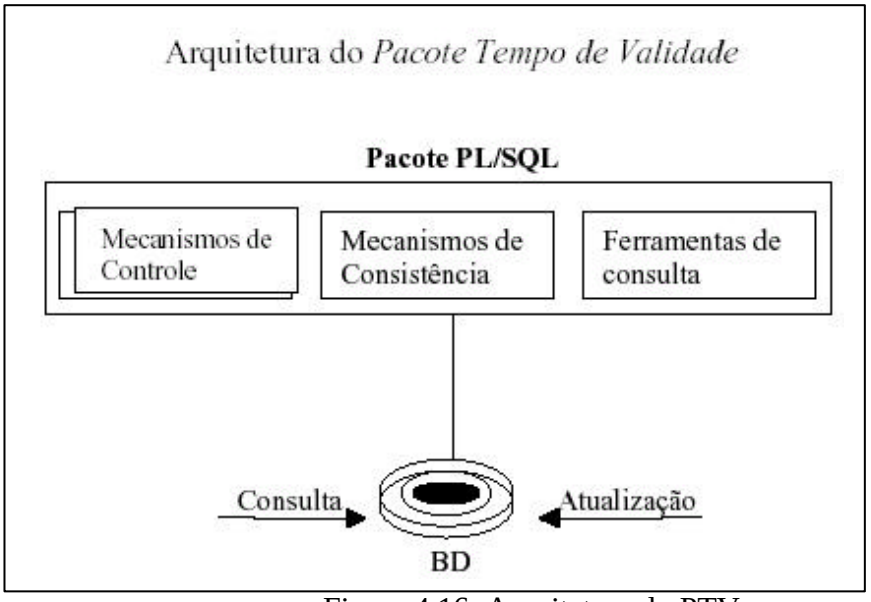


Figura 4.16: Arquitetura do PTV.

4.2 Gerenciamento do PTV

A seguir são mostradas as regras utilizadas pelo PTV para manutenção da consistência de seus dados e analogias eventuais ao *Time Series.*

O primeiro ponto a ser observado é a questão da remoção física de registros no PTV. Apesar de não ser possível conceitualmente em banco de dados Temporais, o PTV permite exclusão (chamado *vacuuming*) física de registros, desde que não comprometa a consistência dos dados, trabalhando semelhantemente ao *Time Series.* Este processo é necessário quando os dados não interessam mais, devendo ser realizada pelo administrador do banco de dados. As regras que serão mostradas em seguida são utilizadas, além da exclusão, em operações de inclusão de dados temporais.

A operação de atualização de dados no PTV não é permitida, diferentemente dos BDT onde é possível atualizar dados não temporais. Já o *Time Series* permite atualização de dados temporais e não temporais. O tempo de validade do PTV não é inserido sobre o pacote do *Time Series*. Sendo assim, os dois pacotes são independentes entre si, de forma que o PTV pode ser utilizado sem a presença do *Time Series.*

4.3 Consistência de Dados de Tempo de Validade

A consistência dos dados temporais no PTV é realizada através de procedimentos e gatilhos que procuram manter ao máximo os conceitos básicos de BDT. Isto implica em que algumas funcionalidades, como exclusão de registros, ficarem fora da implementação.

Para melhor compreensão das regras de validação apresentadas a seguir, é mostrado abaixo um conjunto de termos que serão utilizados nas regras:

- *tstamp* - representa o rótulo temporal;
- *tvi* - representa o tempo de validade inicial;
- *tvf* - representa o tempo de validade final;
- *new* - representa o novo valor de tempo de validade da tupla;
- *old* - representa o valor já inserido do tempo de validade da tupla;
- *maxOld* - representa o maior valor de tempo de validade existente no BD;
- *minOld* - representa o menor valor de tempo de validade existente no BD;
- *tvfOldNext* - representa o tempo de validade final da próxima tupla ;
- *tviOldPrior* - representa o tempo de validade inicial da tupla anterior;
- *Null* – representa um valor Nulo.

Os dados de tempo de validade, diferentemente dos dados de tempo de transação que são fornecidos automaticamente pelo SGBD, devem ser fornecidos pelo usuário. Deste modo, foram definidos dois tipos de regras para controlar o tempo de validade: as regras de usuário e as de banco de dados [PIN 2003].

As regras de usuários são aplicadas sobre dados fornecidos pelo usuário. Já as regras de banco de dados são aplicadas sobre os dados armazenados. A tabela 4.11 mostra um resumo das regras de usuários definidas para o PTV.

Tabela 4.11: Regras de usuários

Nº da Regra	**Sintaxe**	**Descrição**
1	tvi <> NULL	O tempo de validade inicial não pode ser nulo
2	tvi < tvf	O tempo de validade final deve ser maior do que o tempo de validade inicial
3	UPDATE (tvi e tvf)	Não é permitida operação de atualização sobre atributos de tempo

As regras de banco de dados são aplicadas caso exista uma tupla identificada pelo atributo *ID*. A seguir é apresentado, na tabela 12, um resumo das principais situações relativas à definição de tempos de validade, em comparação com os valores já definidos no banco de dados. Para cada uma destas são definidas regras de integridade temporal para os dados. A descrição completa de todas as regras pode ser vista em [PIN 2003].

Em determinadas situações, o PTV realiza atualizações, sobre valores fornecidos ou mesmo armazenados, de forma transparente ao usuário, a fim de corrigir ou evitar inconsistências [PIN 2003].

Tabela 4.12: Regras de banco de dados

Número	Regra	Descrição
1	*TVI.NEW > TVI.MaxOLD*	Tempo de validade inicial da nova tupla é maior que o maior valor do tempo de validade existente.
2	*TVI.NEW < TVI.MinOLD*	Tempo de validade inicial da nova tupla é menor que o menor valor do tempo de validade existente.
3	*TVI.NEW = TVI.OLD (não extremo)*	Tempo de validade inicial da tupla a ser inserida é igual ao valor de tvi de qualquer outra tupla que já tenha sido inserida no BDT, desde que este valor não seja o maior ou o menor valor de tvi da tupla de mesmo "ID" existente na tabela.
4	*TVI.NEW <> TVI.OLD (não extremo)*	Tempo de validade inicial de uma nova tupla não existe no BDT, e este valor não é um valor de extremidades.
5	*TVI.NEW = TVI.MaxOLD*	Tempo de validade inicial da nova tupla coincide com o valor de *tvi* da tupla de maior extremidade
6	*TVI.NEW = TVI.MinOLD*	Essa regra não será aplicada, pois assim como em BDT, as tuplas não podem ser alteradas.

Como exemplo, apresentamos a seguir as regras relativas a situação 2 da tabela 4.12 [PIN 2003].

Regra 2.1
SE *TVI.NEW < TVI.MinOLD* **E**
SE (tvf.new > tvi.old) E (tvf.old + (uma unidade de tempo) > tvi.old)
ENTÃO tvi.old := tvf.new + (uma unidade de tempo)
SENÃO exceção

Se *tvf* de uma nova tupla for maior que o *tvi* de uma tupla pré-existente, então isso caracteriza uma inconsistência nos dados. Com isso, o valor de *tvi* da tupla é substituído pelo valor de tvf da nova tupla acrescido de uma unidade de tempo.

Regra 2.2
SE *TVI.NEW < TVI.MinOLD* **E**
SE tvf.new = tvi.old
ENTÃO tvi.old := tvf.new + (uma unidade de tempo)

Regra 2.3
SE *TVI.NEW < TVI.MinOLD* **E**
SE tvf.new = NULL
ENTÃO exceção

4.4 Funcionamento do PTV

O PTV, apresentado em [PIN 2003], é composto por quatro procedimentos e uma função, descritos a seguir.

?? *PROCEDURE* ativaTV(tabela *VARCHAR2*)
- o tarefa: inclui uma determinada tabela no PTV, atribuindo-lhe mecanismos de controle de consistência. Essa tabela deve possuir

os atributos ID – Identificação, Tvalidadei – tempo de validade inicial e Tvalidadef – tempo de validade final.
- argumentos: tabela ->nome da tabela

?? *PROCEDURE* insereAtributoTV(tabela *VARCHAR2*)
- tarefa: esse procedimento insere os atributos ID – identificação, Tvalidadei – tempo de validade inicial e Tvalidadef – tempo de validade final, em uma determinada tabela.
- argumentos: tabela ->nome da tabela

?? *PROCEDURE* desativaTV(tabela *VARCHAR2*)
- tarefa: remove de uma determinada tabela o mecanismo de consistência de dados temporais.
- argumentos: tabela -> nome da tabela

?? *PROCEDURE* verConsistenciaTV(tabela *VARCHAR2*)
- tarefa: através desse procedimento, o PTV faz uma verificação da consistência dos dados temporais de todas as tuplas da tabela.
- argumentos:tabela -> nome da tabela

?? *FUNCTION* período (tabela *VARCHAR2*, periodoINI *DATE*, periodoFIM *DATE*) return colecaoTV
- tarefa: a função período executa uma consulta a uma determinada tabela, simulando a função *Period* do TSQL2 vista na seção 2.5. O resultado dessa consulta é o conjunto de tuplas em que o tempo de validade inicial é maior ou igual ao argumento periodoINI e o tempo de validade final seja menor ou igual ao argumento periodoFIM.
- argumentos: tabela -> nome de tabela
  periodoINI -> Data que define a data inicial
  periodoFIM -> Data que define a data final

Os códigos desses procedimentos e da função estão em [PIN 2003]. Apenas para ilustração, na figura 4.17 apresentada em [PIN 2003] é mostrado o código do procedimento *insereAtributoTV*:

```
PROCEDURE insereAtributoTV(tabela VARCHAR2) AS
BEGIN
      -- cria os atributos de tempo de validade
      EXECUTE IMMEDIATE ' ALTER TABLE '||tabela||' ADD ID VARCHAR2';
      EXECUTE IMMEDIATE ' ALTER TABLE '||tabela||' ADD TvalidadeI DATE';
      EXECUTE IMMEDIATE ' ALTER TABLE '||tabela||' ADD TvalidadeF DATE';
      -- cria indice
      EXECUTE IMMEDIATE ' CREATE INDEX idxPTV ON '||tabela||'
                                                    (ID,TvalidadeI)';
END insereAtributoTV;
```

Figura 4.17: Código do procedimento insereAtributoTV

O PTV foi desenvolvido para atuar em um banco de dados já existente ou em novos. As funcionalidades do PTV são executadas sobre tabelas individuais e não sobre o banco de dados como um todo. O mecanismo funcional ou ciclo do uso do PTV é iniciado com a inserção dos atributos. Em seguida, a tabela é adicionada ao mecanismo de consistência temporal. A partir desse momento, os dados dessa tabela específica vão ficar submetidos aos mecanismos de controles temporais já mencionados. Finalmente, caso se deseje retirar essa tabela do PTV, através de um único procedimento, o mecanismo de controle e consistência e retirado dessa tabela.

A figura 4.18 mostra o ciclo de vida no uso do PTV em um banco de dados atemporal. Através desta figura, é possível ver o comportamento de uma tabela dentro de um banco de dados, desde o momento em que ela ainda é atemporal, passando a ser uma tabela temporal e voltando a ser novamente atemporal, a medida em que essa funcionalidade não seja mais necessária. Esse exemplo é usado apenas para mostrar que a temporalidade usada pelo PTV é reversível e pode ser usada em banco de dados já existentes, evitando, com isso, a necessidade de se fazer uma modelagem totalmente nova.

O mecanismo de funcionamento do PTV prevê que suas funções devem seguir determinada ordem ou hierarquia. Assim como o *Time Series*, o PTV pode ser instalado em qualquer banco de dados Oracle, independente de sua utilização ou não. Caso o administrador dos dados deseje utilizar as funcionalidades do PTV para uma determinada tabela ou conjunto de tabelas, este deve seguir, individualmente para cada tabela, os passos que seguem:

1. ***inserção do tempo de validade***. Esse é o primeiro passo para a inserção do mecanismo de controle de tempo de validade do PTV em uma tabela. Através do procedimento *insereAtributoTV*, o administrador deve submeter a tabela à criação dos atributos *ID*, *TvalidadeI* e *TvalidadeF*;
2. ***ativação do mecanismo de controle de tempo de Validade*.** Em seguida, submete-se a tabela ao procedimento *ativaTV* que insere as regras de consistências. A partir desse momento essa tabela já está usando os mecanismos de controle temporal do PTV;
3. ***remoção do mecanismo de controle de tempo de validade.*** Caso se deseje voltar ao estado inicial, é necessário utilizar a funcionalidade *desativaTV*, que remove os controles de tempo de validade mas mantém os atributos inseridos pelo procedimento *insereAtributoTV*.

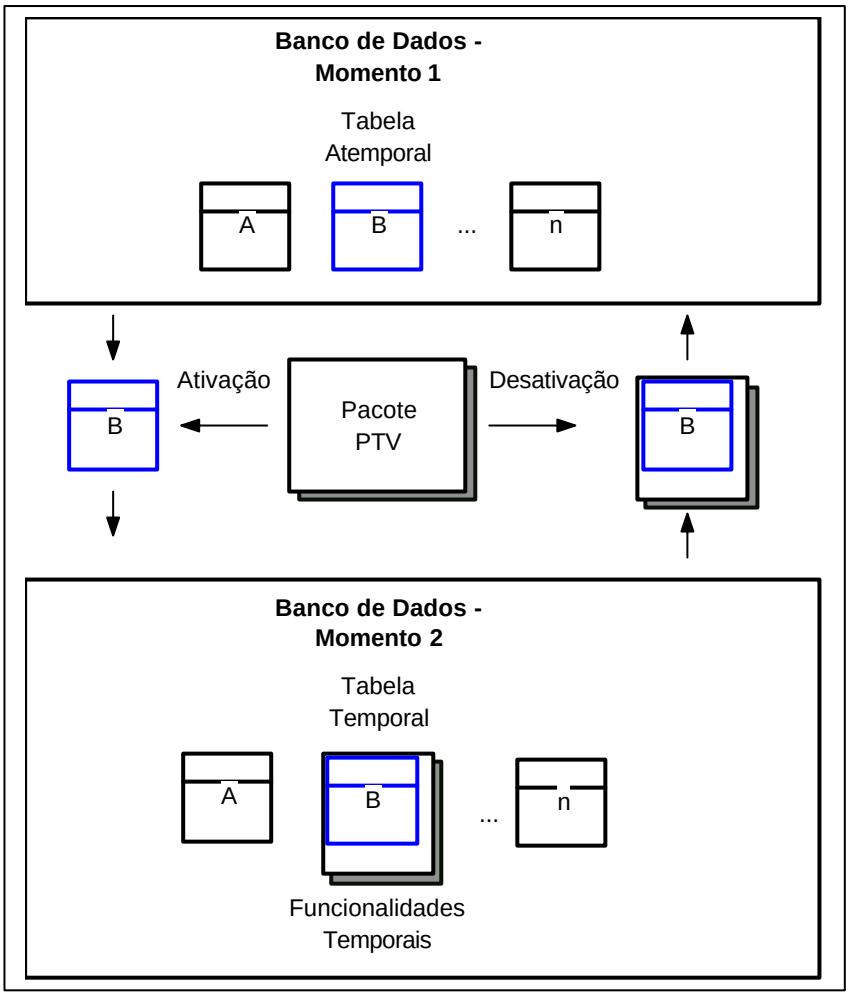


Figura 4.18: Funcionamento do PTV.

A seqüência das figuras 4.19, 4.20, 4.21 e 4.22 mostra o procedimento de ativação do PTV em uma tabela de funcionários denominada FUNC. Esse procedimento foi realizado através da ferramenta Oracle SQLPLUS.

Inicialmente, a tabela FUNC tem a estrutura mostrada na tabela 13.

Tabela 4.13: Estrutura inicial da tabela de funcionários.

Atributo	Tipo	Tamanho
Código	VARCHAR2	10
Nome	VARCHAR2	35
Tstamp	DATE	

A figura 4.19 mostra a inserção dos atributos ID, TvalidadeI e TvalidadeF na tabela de funcionários.

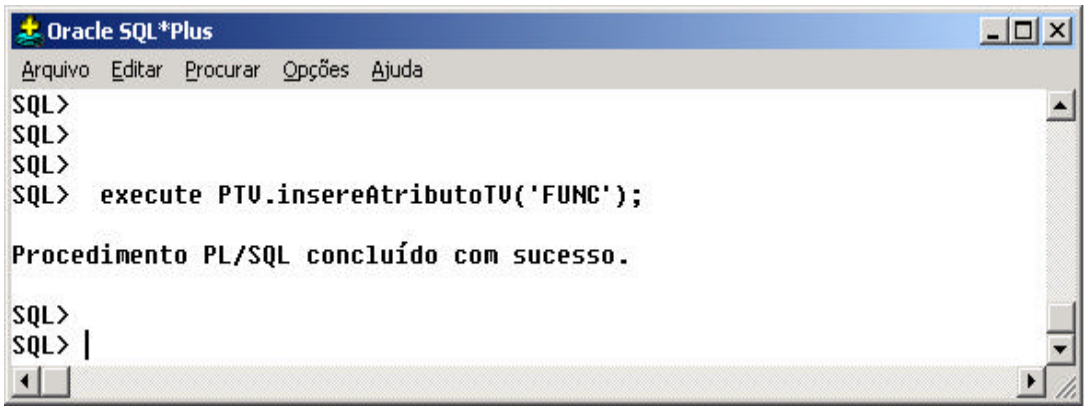


Figura 4.19: Inserção de atributos temporais

Após a inserção dos atributos temporais, o próximo passo é a ativação do controle de tempo de validade. A figura 4.20 ilustra essa etapa. Uma vez executado este procedimento, a tabela FUNC começa a sofrer as restrições mencionadas no item 4.3 deste capítulo. O preenchimento dos dados dos atributos inseridos deve ser feito de forma manual. O PTV disponibiliza uma função para a verificação de todos os dados da tabela temporal. Essa função é realizada através do procedimento *verConsistenciaTV*.

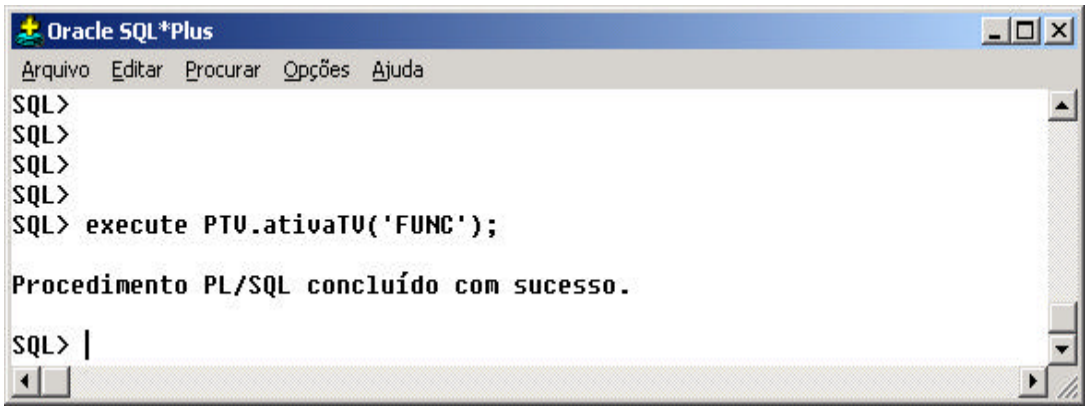


Figura 4.20: Ativação do controle de tempo de validade.

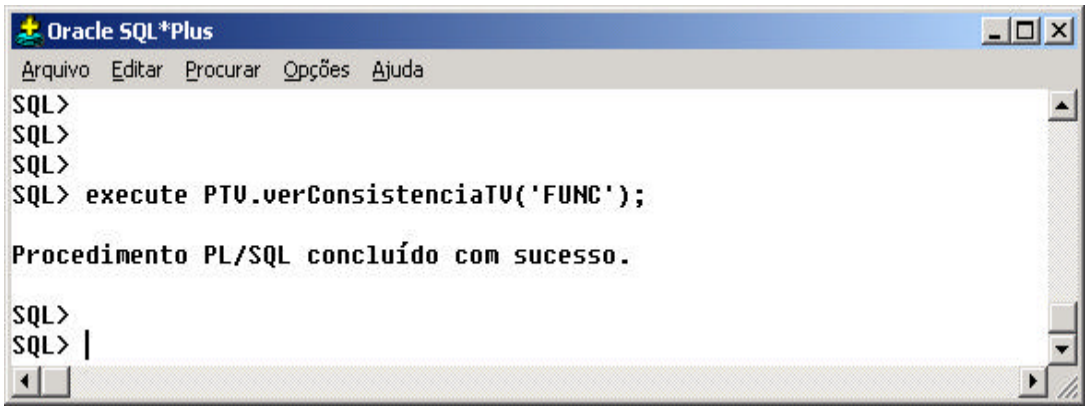


Figura 4.21: Verificação da consistência dos dados temporais

Por fim, se o usuário desejar desfazer todo processo de temporalização da tabela, ele pode utilizar a função para remoção do controle de tempo de validade do PTV. A figura 4.22 ilustra essa desativação.

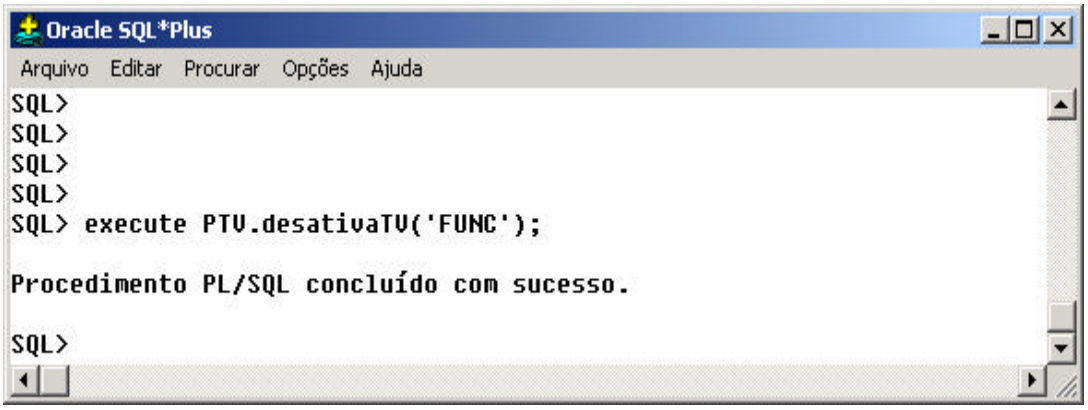


Figura 4.22: Desativação do controle de tempo de validade

Tanto a remoção quanto a inserção do controle de tempo de validade, devem ser feitas tabela a tabela, não sendo possível, portanto, a inserção ou remoção em um conjunto de tabelas ao mesmo tempo.

5 EXTENSÃO AO PTV – FUNÇÕES DE CONSULTA

O PTV tem se mostrado uma interessante ferramenta para prática do uso de BDT, uma vez que não existem ainda BDT puros [CAV 95]. Outro aspecto importante do PTV é que ele foi desenvolvido para um SGBD bastante difundido comercialmente, dando com isso, a possibilidade de sua utilização de forma comercial. Apesar dos pontos positivos mencionados, o PTV ainda tem alguns pontos a serem melhorados. Um destes pontos é a falta de uma análise de performance para a execução de seus procedimentos. As ferramentas de apoio a banco de dados em geral, normalmente devem ter boa performance.

Outro ponto em que o PTV poderia ser melhorado é a disponibilidade de mais funções de consulta temporal. Em sua versão original, o PTV só disponibilizava uma função de consulta temporal propriamente dita.

O principal objetivo desse trabalho foi incorporar ao PTV um conjunto de funções de consulta temporal para dar mais funcionalidade a esse pacote, bem como aproximá-lo cada vez mais ao que deveria ser um BDT. Um dos maiores desafios para a criação dessas funções foi definir exatamente quantas e quais funções de consulta temporal deveriam ser disponibilizadas.

O primeiro problema levantado foi a identificação dessas funções. Elas poderiam vir de alguma proposta já disponível pela literatura. Foram analisadas diversas propostas de modelos de dados temporais, e de suas linguagens de consulta:

- HRDM (*The Historical Relational Data Model*) [CLI 93, CLI 87]
- HSQL(*Historical SQL*) [SAR 90, SAR 93]
- TQuel (*Temporal Query Language*) [SNO 93]
- TEER (*Temporal EER*) [ELM 93]
- TSQL2 [SNO 95]

Após esse estudo, foi definido que a linguagem a ser adotada como modelo básico para o PTV deveria ser o TSQL2 devido, principalmente, ao fato do Oracle [ORA 202002] implementar em sua linguagem SQL, o padrão ANSI. Tendo o TSQL2 sido originado do mesmo modelo, obtém-se desta forma uma maior proximidade entre o SGBD e a linguagem de consulta temporal.

5.1 Funções de consulta

Nesta seção serão apresentadas as novas funções de consultas implementadas para o PTV. Assim como o TSQL2 é uma extensão temporal para o padrão SQL92, as funções desenvolvidas nesse trabalho formam uma extensão ao PTV, que nada mais é do que um pacote SQL em um SGBD que segue o padrão SQL92. Um aspecto relevante nesse trabalho é o fato tentar aplicar na prática as funcionalidades desenvolvidas no

TSQL2. Nesse caso, o TSQL2 está para o SQL92 assim como as funções de consultas temporais desenvolvidas estão para o PTV.

A tabela 5.14 mostra o conjunto de todas as funções do PTV. Nesta mesma tabela, as funções em negrito correspondem à proposta desse trabalho. Na seqüência dessa seção, essas funções serão apresentadas com mais detalhes.

5.1.1 Procedimento *tpreencheTV*

Esse procedimento é a única exceção no desenvolvimento dessas funções, pois não seguiu o padrão do TSQL2. A origem dessa função surgiu de uma necessidade no uso do próprio PTV. Depois do uso do procedimento *insereAtributoTV* e *ativaTV*, havia a necessidade do preenchimento manual dos atributos ID, TvalidadeI e TvalidadeF. O preenchimento do atributo ID é bastante simples, pois sua origem deve ser feita através de um atributo chave, como código ou matrícula. Utilizando o exemplo da figura 3.12, o ID deve ser preenchido através do atributo CÓDIGO. Com relação aos atributos TvalidadeI e TvalidadeF, o preenchimento não é tão simples, pois, como podemos ver na tabela 15, existe uma dificuldade ao se fazer o relacionamento entre os atributos, no sentido de atribuir os valores de TVALIDADEI e TVALIDADEF. No exemplo são apresentados apenas os atributos envolvidos no processo de atualização dos dados temporais, excluindo-se, portanto, atributos não relevantes para esse procedimento, como NOME, SALARIO, etc.

Tabela 5.14: Funções do PTV.

Pacote de Tempo de Validade - PTV
procedure ativaTV(tabela VARCHAR2);
procedure insereAtributoTV(tabela VARCHAR2);
procedure desativaTV(tabela VARCHAR2);
procedure verConsistenciaTV(tabela VARCHAR2);
Function periodo(tabela VARCHAR2, periodoINI DATE, periodoFIM DATE);
procedure tpreencheTV(tabela VARCHAR2);
Function tfirst(tabela VARCHAR2, dataini date, datafim date);
Function tlast(tabela VARCHAR2, dataini date, datafim date);
Function tbegin(tabela VARCHAR2, dataini date, datafim date);
Function tend(tabela VARCHAR2, dataini date, datafim date);
Function tperiod(tabela VARCHAR2, dataini date, datafim date);
Function tintersect(tabela VARCHAR2, dataini1 date, datafim1 date, dataini2 date, datafim2 date);
function tcount(tabela VARCHAR2, atrib1 NUMBER, atrib2 NUMBER, atrib3 NUMBER, dataini date, datafim date);
function tsum(tabela VARCHAR2, atrib1 NUMBER, atrib2 NUMBER, atrib3 NUMBER, soma varchar2, dataini date, datafim date);
function tavg(tabela VARCHAR2, atrib1 NUMBER, atrib2 NUMBER, atrib3 NUMBER, media varchar2, dataini date, datafim date);
function tmax(tabela VARCHAR2, atrib1 NUMBER, atrib2 NUMBER, atrib3 NUMBER, maior varchar2, dataini date, datafim date);
function tmin(tabela VARCHAR2, atrib1 NUMBER, atrib2 NUMBER, atrib3 NUMBER, menor varchar2, dataini date, datafim date);

Tabela 5.15: Atualiza de atributos de tempo de validade.

Linha	B – Tempo de Transação		A – Tempo de Validade		
	CODIGO	TSTAMP	ID	TVALIDADEI	TVALIDADEF
1	1	1/1/1995	1	1/1/1995	9/4/1995
2	1	10/4/1995	1	10/4/1995	11/4/1996
3	1	12/4/1996	1	12/4/1996	30/11/1996
4	1	1/12/1996	1	1/12/1996	30/4/1997
5	1	1/5/1997	1	1/5/1997	Null
6	10	10/1/1995	10	10/1/1995	9/1/1996
7	10	10/1/1996	10	10/1/1996	Null
8	11	11/1/1995	11	11/1/1995	10/1/1997
9	11	11/1/1997	11	11/1/1997	Null
10	12	12/1/1995	12	12/1/1995	11/11/1995
11	12	12/11/1995	12	12/11/1995	Null
12	13	13/1/1995	13	13/1/1995	12/1/1998
13	13	13/1/1998	13	13/1/1998	Null
14	14	1/4/1990	14	1/4/1990	31/3/1992
15	14	1/4/1992	14	1/4/1992	Null
16	15	2/4/1990	15	2/4/1990	1/4/1992
17	15	2/4/1992	15	2/4/1992	Null
18	16	3/4/1990	16	3/4/1990	2/4/1993
19	16	3/4/1993	16	3/4/1993	Null
20	17	4/4/1990	17	4/4/1990	3/5/1990
21	17	4/5/1990	17	4/5/1990	3/6/1990
22	17	4/6/1990	17	4/6/1990	Null
23	18	5/4/1990	18	5/4/1990	4/4/1996
24	18	5/4/1996	18	5/4/1996	4/4/1998
25	18	5/4/1998	18	5/4/1998	Null

Na tabela 5.15-B são mostrados os atributos CODIGO e TSTAMP. Nesse momento a tabela de funcionários se encontra em um estado semelhante ao tempo de transação, pois para cada *tupla* é associado um rótulo temporal. Uma vez inseridos os atributos de tempo de validade na tabela 5.15-A (TvalidadeI e TvalidadeF) era necessário fazer uma atualização manual em que, para cada tupla, devia-se realizar a seguinte tarefa: o TvalidadeI receber o rótulo temporal (TSTAMP) e o TvalidadeF receber, se existir, o próximo TSTAMP para aquele mesmo ID menos uma unidade de tempo; caso contrário receber valor nulo. Para as tuplas de ID igual a 16 (linhas 18 e 19) é feito o seguinte: na linha 18 o TvalidadeI recebe o valor de TSTAMP diretamente, ou seja 03/04/1990. O TvalidadeF dessa mesma linha será o TSTAMP menos uma unidade de tempo do próximo registro de ID igual a 16, se existir, senão receberá Nulo. Nesse caso o valor atribuído será 03/04/1993 – (uma unidade de temp), resultando em 02/04/1993. Para o próximo registro de ID igual a 16 (linha 19), é feito o mesmo raciocínio. O TvalidadeI recebe o TSTAMP, ou seja, 03/04/1993. O TvalidadeF recebe nulo, pois não há mais tuplas com ID igual a 16.

O procedimento *tpreencheTV* realiza exatamente esse procedimento de preenchimento dos valores correspondente aos atributos ID, TvalidadeI e TvalidadeF, inseridos pelo procedimento *insereAtributoTV*. Esse procedimento torna essa tarefa automática. A figura 5.23 mostra o código do procedimento *tpreencheTV*, que utiliza os recursos das funções analíticas do SQL, mencionadas no capítulo 3, item 3.2.1.

```
PROCEDURE tpreencheTV(tabela VARCHAR2) AS
TYPE Cursores is REF CURSOR;
c1 Cursores;
tvi date;
tvf date;
v_id varchar2(15);
sql_sentenca varchar2(300);
sql_sentenca1 varchar2(300);
BEGIN
-- Atualização do atributo ID a partir do CODIGO
sql_sentenca1 := 'update ' || tabela || ' set ID = CODIGO';
execute immediate sql_sentenca1;
execute immediate 'commit ';
-- Atualização dos atributos TvalidadeI e TvalidadeF
OPEN c1 FOR  'SELECT id,tstamp, LEAD(tstamp-1,1,null)
            OVER (PARTITION BY id ORDER BY tstamp)
            FROM ' || tabela || ' ';
 LOOP
    FETCH c1 INTO v_id,tvi,tvf;
     sql_sentenca := 'update ' || tabela || ' set tvalidadei = '||""|| tvi ||""||'
     ,tvalidadef = '||""|| tvf ||""||
     ' where id = '||""||v_id||""|| ' and tstamp = ' ||"" || tvi ||""|| ' ';
     execute immediate sql_sentenca;
     execute immediate 'commit ';
    EXIT WHEN c1%NOTFOUND;
  END LOOP;
  CLOSE c1;
END tpreencheTV;
```

Figura 5.23: Procedimento *tpreencheTV*

Nas figuras 5.24 a 5.29 a seguir, são mostradas as etapas do funcionamento do procedimento *tpreencheTV* através do Oracle SqlPlus para a tabela de Funcionários - FUNC.

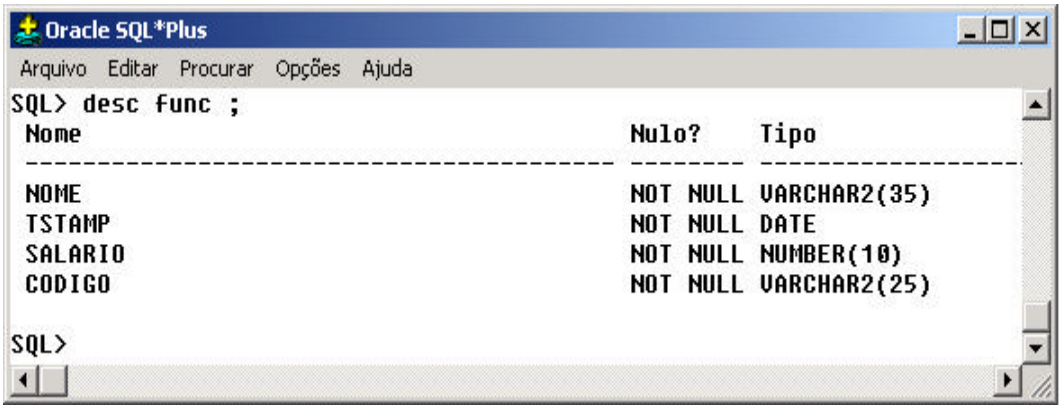


Figura 5.24: Estrutura original da tabela FUNC – Funcionários

Na figura 5.24 a tabela FUNC ainda não está utilizando o PTV. Na figura 5.25 a tabela FUNC utiliza um rótulo temporal como um tempo de transação. Na figura 5.26, são adicionados os atributos ID, TvalidadeI e TvalidadeF através do procedimento *insereAtributoTV*.

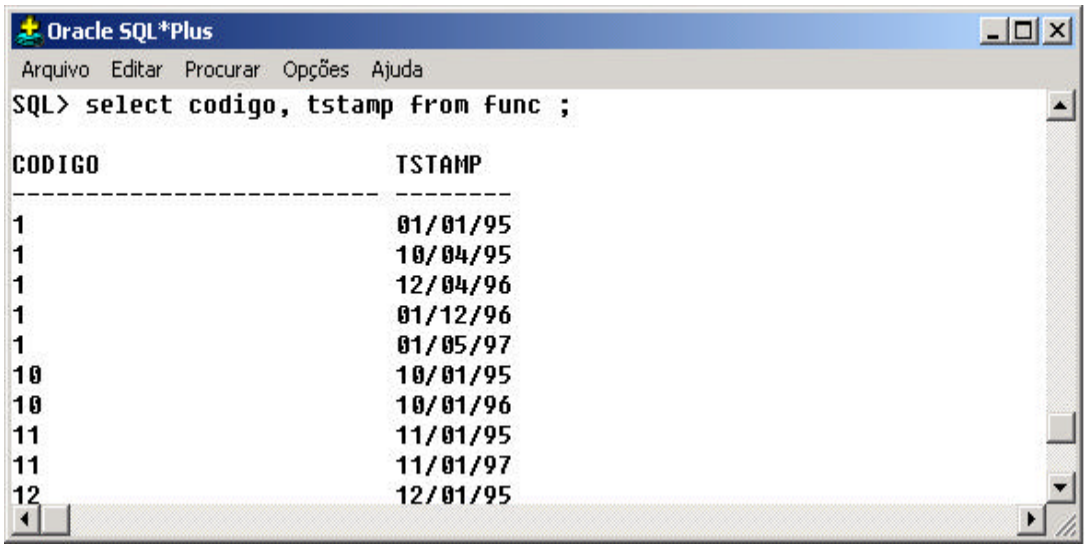


Figura 5.25: Parte dos dados da tabela FUNC.

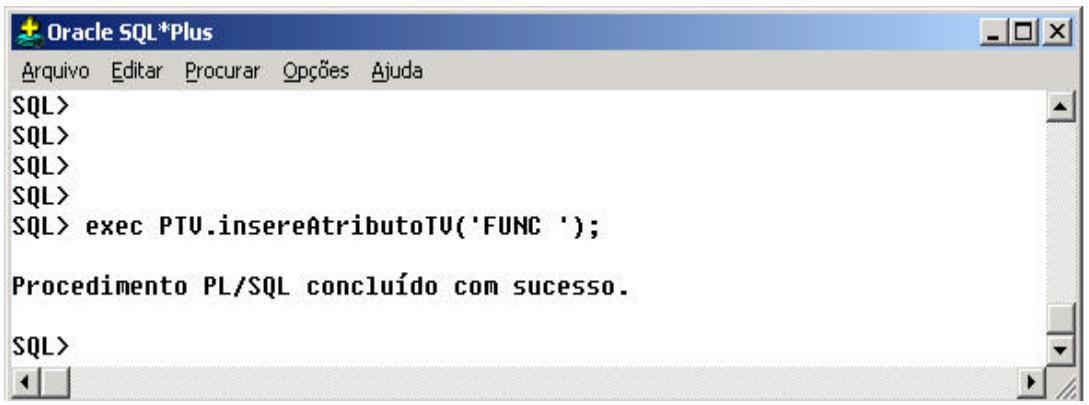


Figura 5.26: Inserção dos atributos ID, TvalidadeI e TvalidadeF

Logo em seguida, é feita uma verificação de que realmente os atributos inseridos ainda estão sem dados. O comando SELECT da figura 5.27, mostra apenas parte dos dados da tabela FUNC.

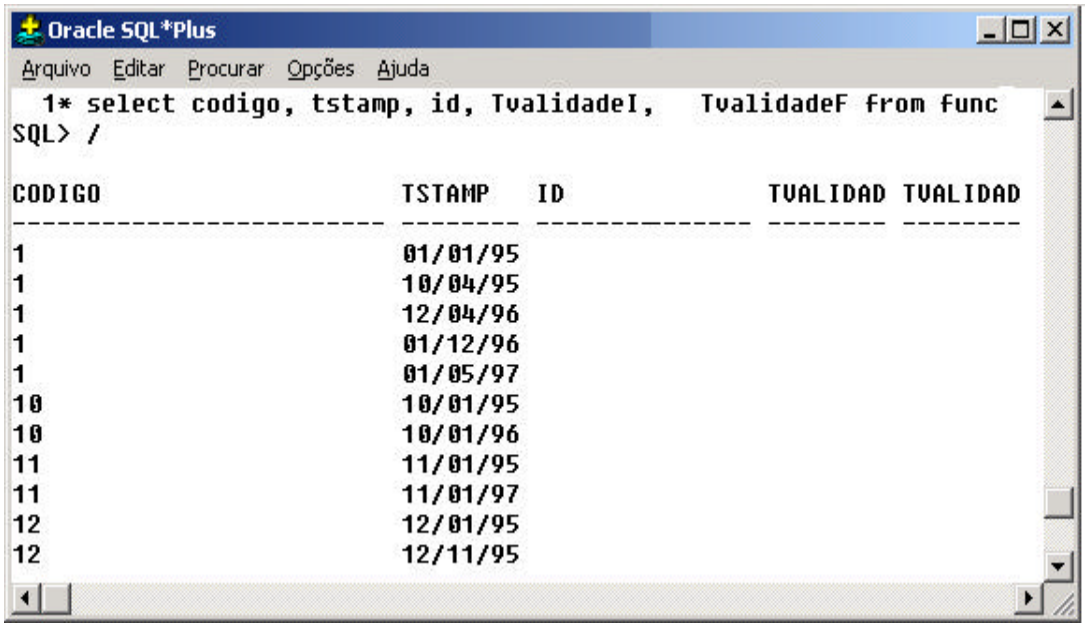


Figura 5.27: Listagem de parte dos dados da Tabela FUNC.

Em seguida, o procedimento *preencheTV* é acionado. Esse passo não é obrigatório para o funcionamento do PTV, mas traz bastante comodidade ao usuário administrador, pois dependendo da quantidade de dados na tabela original, essa tarefa pode necessitar de muito tempo se realizada manualmente.

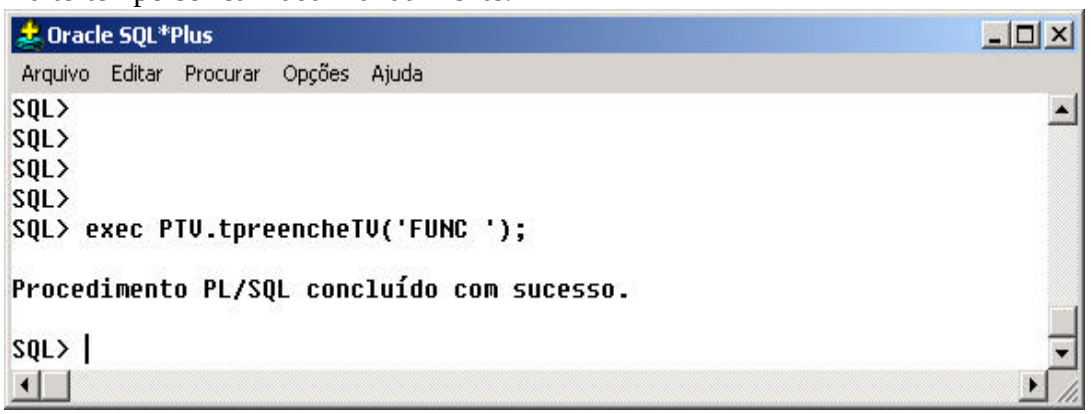


Figura 5.28: Execução do procedimento *preencheTV*

Após sua execução, os atributos ID, TvalidadeI e TvalidadeF foram preenchidos automaticamente, como mencionados. A figura 5.29, mostra agora os dados preenchidos, em contraste com a figura 5.27 antes da execução do procedimento *preencheTV*.

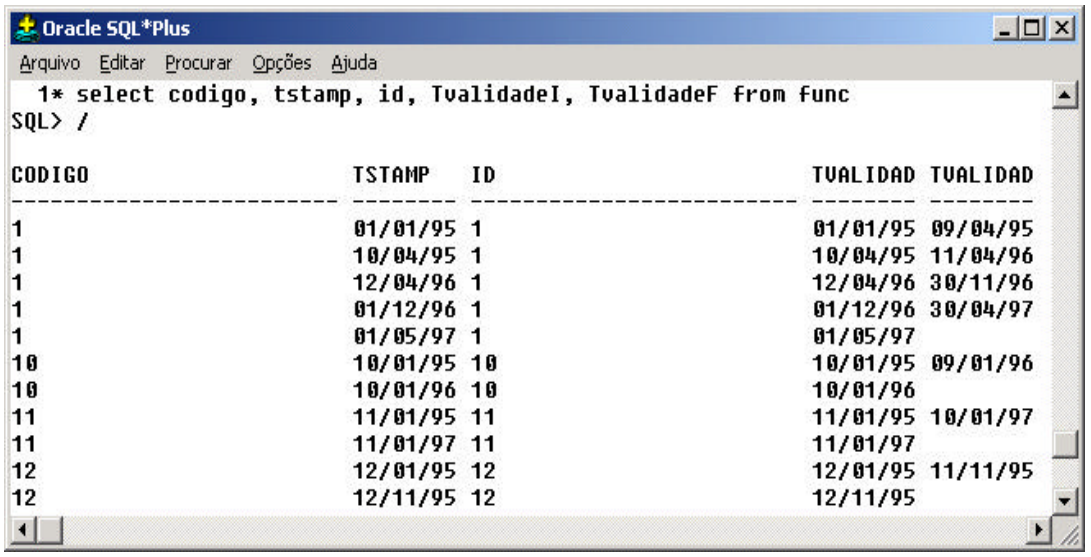


Figura 5.29: Visualização dos dados depois da execução do *preencheTV*.

5.1.2 Função *tfirst*

A função *tfirst* retorna a primeira data do atributo de TvalidadeI que satisfaça o critério de período passado pelo parâmetro da função, ou seja TvalidadeI >= tvi e TvalidadeF <= tvf.

?? Código fonte: anexo A

?? Sintaxe: tfirst(<tabela>,<tvi>, <tvf>)
Tabela -> Nome da tabela onde será realizada a consulta
tvi -> Tempo de validade inicial
tvf -> Tempo de validade final

?? Modo de uso: cláusula FROM da sentença SQL

?? Exemplo: Figura 5.30

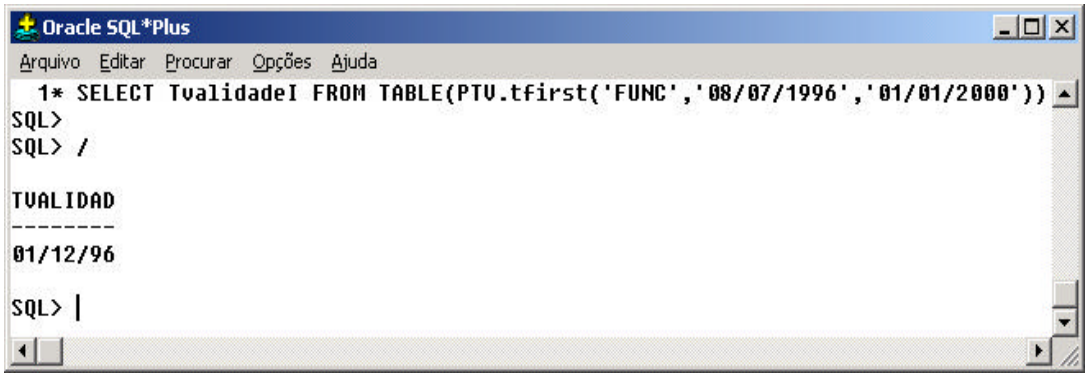


Figura 5.30: Exemplo da função *tfirst*

5.1.3 Função *tlast*

A função *tlast* retorna a última data do atributo TvalidadeF que satisfaça o critério de período passado pelo parâmetro da função, ou seja TvalidadeF >= tvi e TvalidadeF <= tvf.

?? Código Fonte: anexo A

?? Sintaxe: tlast (<tabela>,<tvi>, <tvf>)
Tabela -> Nome da tabela onde será realizada a consulta

tvi -> Tempo de validade inicial
tvf -> Tempo de validade final

?? Modo de uso: cláusula FROM da sentença SQL.

?? Exemplo: figura 5.31

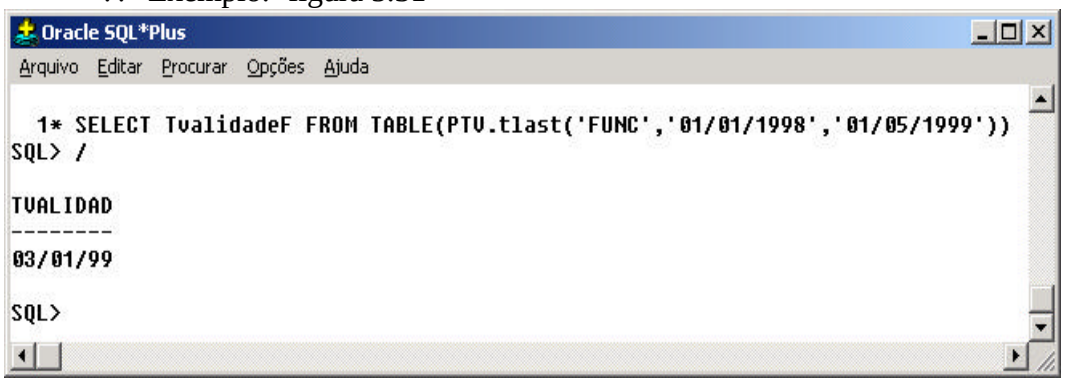


Figura 5.31: Exemplo da função *tlast*.

5.1.4 Função *tbegin*

A função *tbegin* retorna as tuplas em que o conjunto o atributo TvalidadeI satisfaça o critério de período passado pelo parâmetro da função, ou seja TvalidadeI >=tvi e TvalidadeI <= tvf.

?? Código Fonte: anexo A

?? Sintaxe: tbegin (<tabela>,<tvi>, <tvf>)
Tabela -> Nome da tabela onde será realizada a consulta
tvi -> Tempo de validade inicial
tvf -> Tempo de validade final

?? Modo de uso: cláusula FROM da sentença SQL.

?? Exemplo: figura 5.32

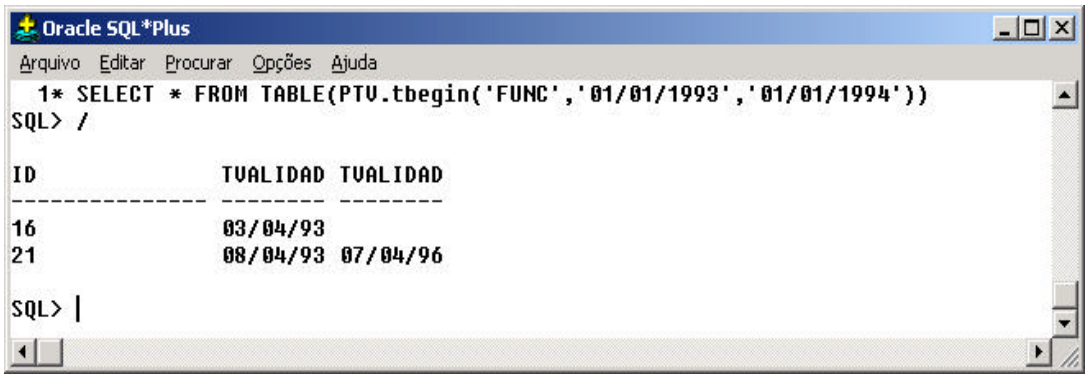


Figura 5.32: Execução da função *tbegin*.

5.1.5 Função *tend*

A função *tend* retorna as tuplas em que o atributo TvalidadeF satisfaça o critério de período passado pelo parâmetro da função, ou seja TvalidadeF >=tvi e TvalidadeF <= tvf.

?? Código Fonte: anexo A

?? Sintaxe: tend (<tabela>,<tvi>, <tvf>)
Tabela -> Nome da tabela onde será realizada a consulta
tvi -> Tempo de validade inicial

tvf -> Tempo de validade final

?? Modo de uso: cláusula FROM da sentença SQL.

?? Exemplo: figura 5.33

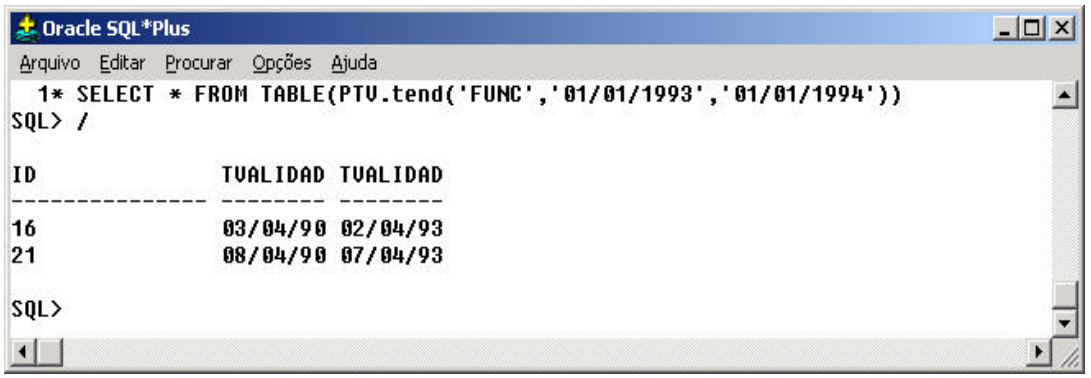


Figura 5.33: Execução da função *tend*.

5.1.6 Função *tperiod*

A função *tperiod* retorna as tuplas em que o conjunto de atributos TvalidadeI e TvalidadeF satisfaçam o critério de período passado pelo parâmetro da função, ou seja TvalidadeI >=tvi e TvalidadeF <= tvf.. Tperiod conceitualmente realiza a mesma tarefa que a função PERIODO do PTV. A única diferença é que tperiod implementa o conceito de *Pipelined* [ORA 2002], recurso que permite maior desempenho na execução da função.

?? Código Fonte: anexo A

?? Sintaxe: tperiod (<tabela>,<tvi>, <tvf>)
Tabela -> Nome da tabela onde será realizada a consulta
tvi -> Tempo de validade inicial
tvf -> Tempo de validade final

?? Modo de uso: cláusula FROM da sentença SQL.

?? Exemplo: figura 5.34

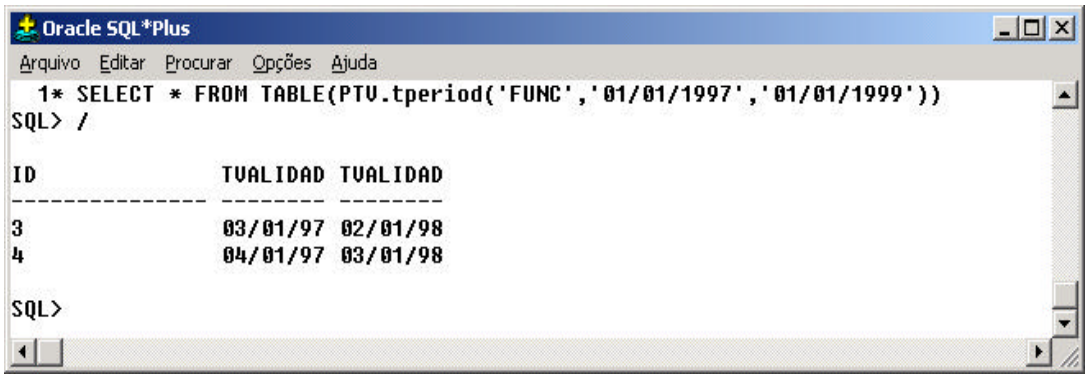


Figura 5.34: Execução da função *tperiod*.

5.1.7 Função *tintersect*

A função *tintersect* retorna as tuplas de um determinado período que corresponda a intersecção de dois períodos passados por parâmetro.

?? Código Fonte: anexo A

?? Sintaxe: tperiod (<tabela>,<tvi-1>, <tvf-1>,<tvi-2>,<tvf-2>)

Tabela -> Nome da tabela onde será realizada a consulta
tvi-1 -> Tempo de validade inicial do primeiro período
tvf-1 -> Tempo de validade final do primeiro período
tvi-2 -> Tempo de validade inicial do segundo período
tvf-2 -> Tempo de validade final do segundo período

?? Modo de uso: cláusula FROM da sentença SQL.
?? Exemplo: figura 5.35

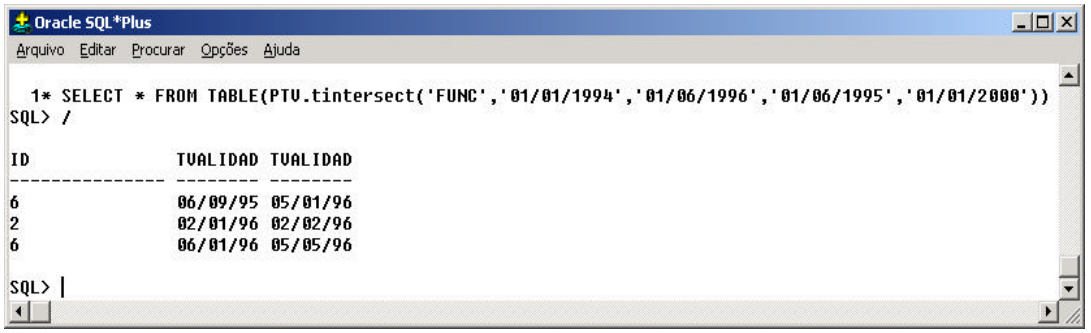


Figura 5.35: Execução da função *tintersect*.

5.1.8 Função *tcount*

A função *tcount* retorna as tuplas com as quantidades que satisfaçam aos critérios passados a função como parâmetro.

?? Código Fonte: anexo A
?? Sintaxe: tcount (<tabela>,<atrib1>,<atrib2>,<atrib3>,<tvi>, <tvf>)
Tabela -> Nome da tabela onde será realizada a consulta
Atrib1 -> Se igual a 1, o TvalidadeI é usado para contagem
Atrib2 -> Se igual a 1, o TvalidadeF é usado para contagem
Atrib3 -> Se igual a 1, o ID é usado para contagem
tvi -> Tempo de validade inicial do período
tvf -> Tempo de validade final do período
?? Modo de uso: cláusula FROM da sentença SQL.
?? Exemplo: figura 5.36

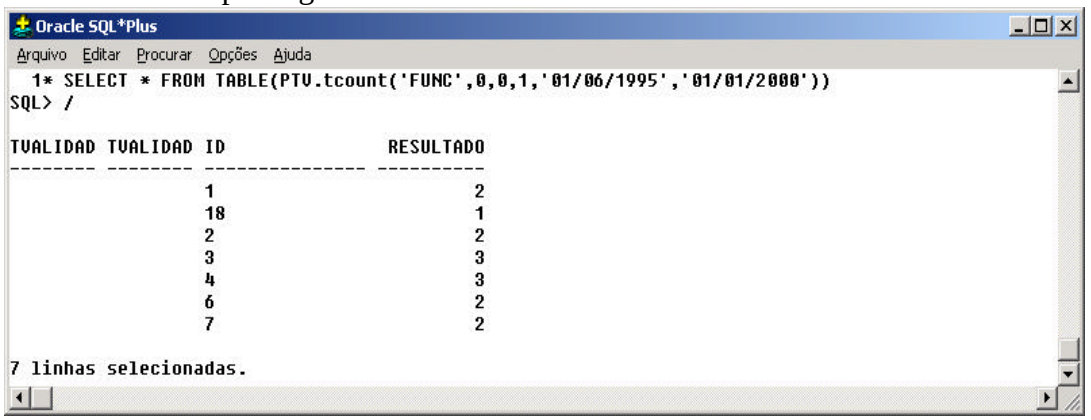


Figura 5.36: Execução da função *tcount*.

5.1.9 Função *tsum*

A função *tsum* retorna as tuplas com as somas que satisfaçam aos critérios passados à função como parâmetro.

?? Código Fonte: anexo A

?? Sintaxe: tcount (<tabela>,<atrib1>,<atrib2>,<atrib3>,<soma>,<tvi>, <tvf>)
Tabela -> Nome da tabela onde será realizada a consulta
Atrib1 -> Se igual a 1, o TvalidadeI é usado para agrupamento
Atrib2 -> Se igual a 1, o TvalidadeF é usado para agrupamento
Atrib3 -> Se igual a 1, o ID é usado para agrupamento
Soma -> Atributo no qual será feito a soma.
tvi -> Tempo de validade inicial do período
tvf -> Tempo de validade final do período

?? Modo de uso: cláusula FROM da sentença SQL.

?? Exemplo: figura 5.37

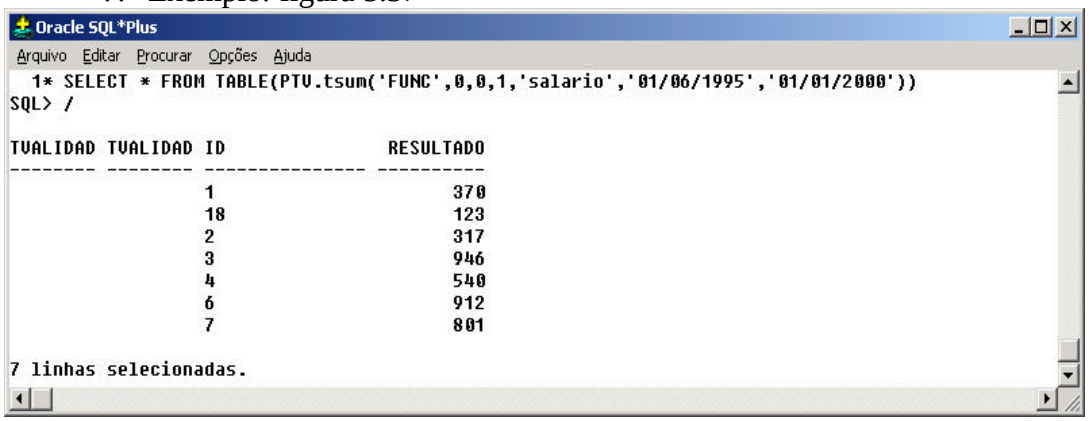


Figura 5.37: Execução da função *tsum*.

5.1.10 Função *tavg*

A função *tavg* retorna as tuplas com as médias que satisfaçam aos critérios passados à função como parâmetro.

?? Código Fonte: anexo A

?? Sintaxe: tavg (<tabela>,<atrib1>,<atrib2>,<atrib3>,<media >,<tvi>, <tvf>)
Tabela -> Nome da tabela onde será realizada a consulta
Atrib1 -> Se igual a 1, o TvalidadeI é usado para agrupamento
Atrib2 -> Se igual a 1, o TvalidadeF é usado para agrupamento
Atrib3 -> Se igual a 1, o ID é usado para agrupamento
Média -> Atributo no qual será feito a média.
tvi -> Tempo de validade inicial do período
tvf -> Tempo de validade final do período

?? Modo de uso: cláusula FROM da sentença SQL.

?? Exemplo: figura 5.38

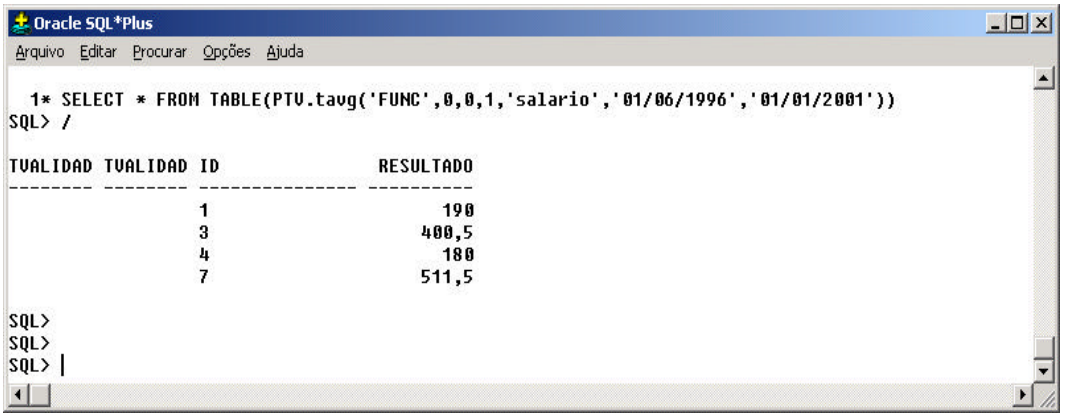


Figura 5.38: Execução da função *tavg*.

5.1.11 Função *tmax*

A função *tmax* retorna as tuplas com os valores máximos que satisfaçam aos critérios passados à função como parâmetro.

?? Código Fonte: anexo A

?? Sintaxe: tavg (<tabela>,<atrib1>,<atrib2>,<atrib3>,<max >,<tvi>, <tvf>)
Tabela -> Nome da tabela onde será realizada a consulta
Atrib1 -> Se igual a 1, o TvalidadeI é usado para agrupamento
Atrib2 -> Se igual a 1, o TvalidadeF é usado para agrupamento
Atrib3 -> Se igual a 1, o ID é usado para agrupamento
Max -> Atributo no qual será verificado o valor máximo.
tvi -> Tempo de validade inicial do período
tvf -> Tempo de validade final do período

?? Modo de uso: cláusula FROM da sentença SQL.

?? Exemplo: figura 5.39

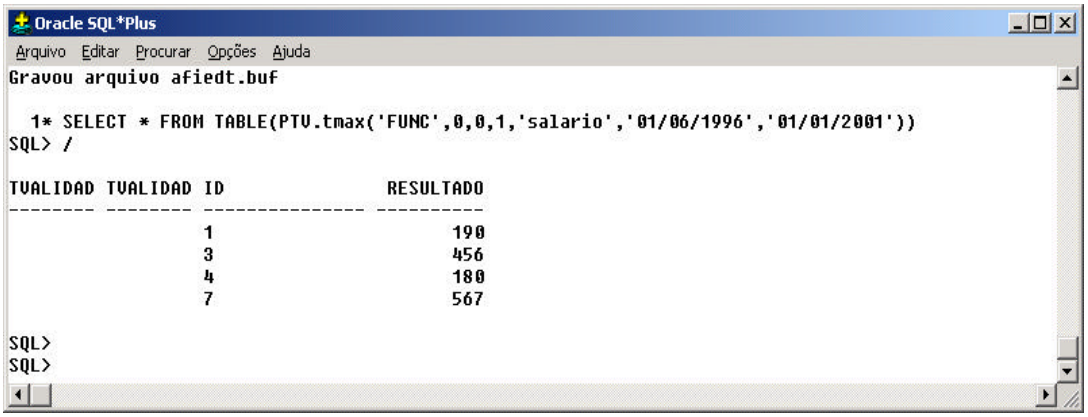


Figura 5.39: Execução da função *tmax*.

5.1.12 Função *tmin*

A função *tmin* retorna as tuplas com os valores mínimos que satisfaçam aos critérios passados à função como parâmetro.

?? Código Fonte: anexo A

?? Sintaxe: tavg (<tabela>,<atrib1>,<atrib2>,<atrib3>,<min >,<tvi>, <tvf>)
Tabela -> Nome da tabela onde será realizada a consulta

Atrib1 -> Se igual a 1, o TvalidadeI é usado para agrupamento
Atrib2 -> Se igual a 1, o TvalidadeF é usado para agrupamento
Atrib3 -> Se igual a 1, o ID é usado para agrupamento
Min -> Atributo no qual será verificado o valor mínimo.
tvi -> Tempo de validade inicial do período
tvf -> Tempo de validade final do período

?? Modo de uso: cláusula FROM da sentença SQL.
?? Exemplo: figura 5.40

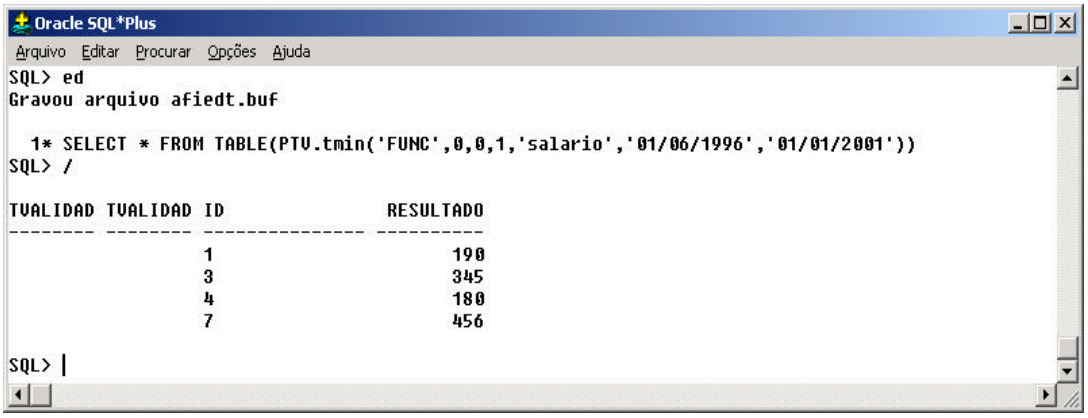


Figura 5.40: Execução da função *tmin*.

5.2 Considerações finais

As funções apresentadas neste capítulo ampliam as funcionalidades do PTV, principalmente no que diz respeito às consultas temporais. Essas funções foram desenvolvidas com base na linguagem de consulta temporal TSQL2. Três aspectos relevantes podem ser destacados desta implementação. O primeiro é a flexibilidade no uso das funções temporais, uma vez que elas podem ser utilizadas dentro de sentenças SQL. A exceção é o procedimento *preencheTV*. O segundo ponto diz respeito à diversidade das funcionalidades implementadas, possibilitando, com isso, maior variação em seu uso. Por fim, essas funções consolidam a importância da linguagem de consulta temporal TSQL2 .

6 INTERFACE PTV FONT-END

Os capítulos 4 e 5 mostraram o funcionamento e o mecanismo de utilização do PTV com a extensão realizada neste trabalho. Através das figuras é possível verificar que essas funcionalidades são sempre utilizadas pela ferramenta Oracle SQLPlus, que funciona de forma interativa, e através da linguagem SQL. Isso significa que o manuseio do PTV é bastante sensível, ou seja, pequenos erros de digitação podem gerar erros nas funções. Outro ponto que dificulta sua utilização, é o fato de que as ordens dos comandos não são claras, pois não existe um ambiente integrado que dê ao PTV uma forma de produto acabado.

Em virtude desses problemas, foi desenvolvida neste trabalho uma interface *front-end*, que tem como função básica facilitar o manuseio dos recursos do PTV e oferecer uma ferramenta de estudos em BDT ao interessados no assunto. Esse ambiente foi desenvolvido através da linguagem Delphi versão 5.

A interface desenvolvida, denominada PTV Front-end possui três macro-níveis funcionais onde são agregadas suas sub-funções. O primeiro nível é o Arquivo que deverá ser desenvolvido em futuras extensões do PTV Front-end O segundo é o nível de Controle onde estão dispostos os procedimentos:

- ?? Insere Tempo de Validade
- ?? Ativa Tempo de Validade
- ?? Preenche Tempo de Validade
- ?? Verifica Consistência de Tempo de Validade
- ?? Desativa Tempo de Validade

O outro macro-nível é chamado Consultas, o qual por sua vez é subdividido em dois outros sub-níveis: Agregação e Conversão. O sub-nível de Agregação é formado por:

- ?? Função de Contador (Tcount)
- ?? Função de Soma (Tsum)
- ?? Função de Média (Tavg)
- ?? Função de Valor Máximo (Tmax)
- ?? Função de Valor Mínimo (TMin)

O sub-nível de Conversão é formado pelas funcionalidades:

- ?? Função Tbegin
- ?? Função TEnd
- ?? Função Tperiod
- ?? Função Tintersect
- ?? Função Tfirst
- ?? Função Tlast

A figura 6.41 mostra a tela principal do PTV Front-End. Através desse formulário principal é possível acessar as opções e funcionalidades do PTV através do menu de opções. A área de trabalho do formulário não tem funcionalidades.

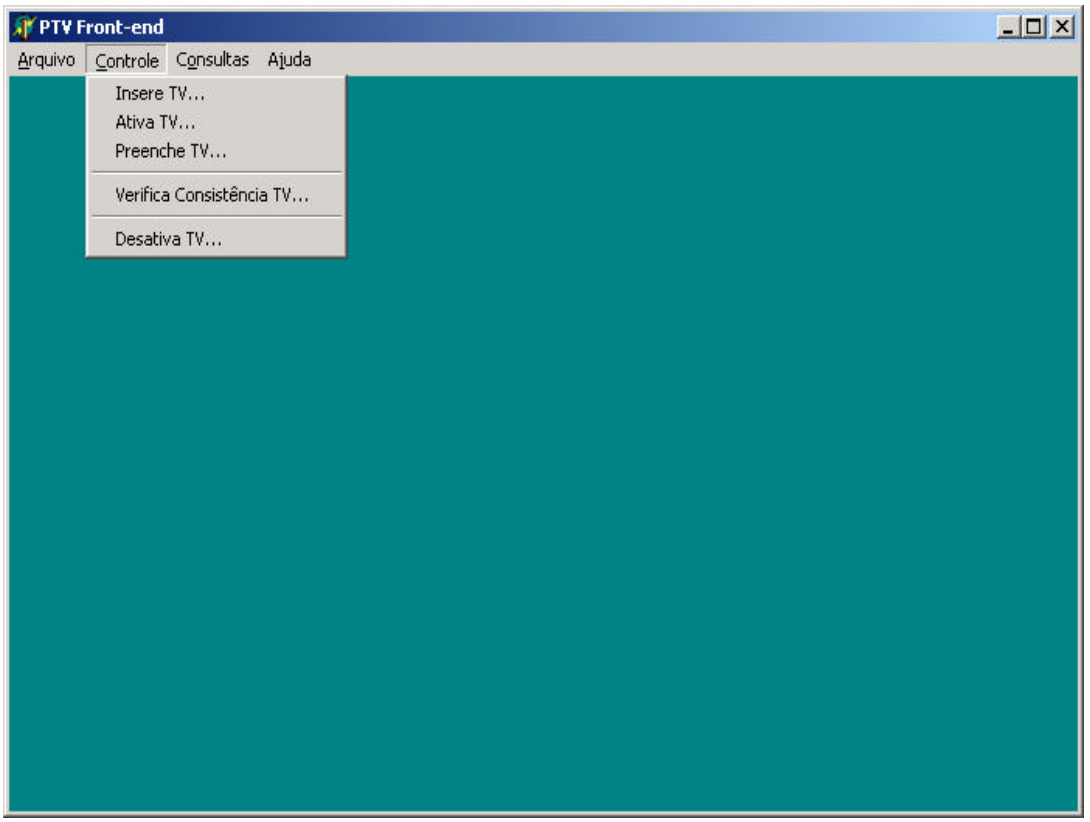


Figura 6.41: Formulário principal do PTV Front-End.

6.1 Menu Controle

O menu Controle é responsável pela ativação e desativação do PTV em uma tabela. O funcionamento dessas funcionalidades já foi discutido em momentos anteriores. Nesse capítulo serão mostradas apenas as interfaces e como utilizá-las. As opções desse menu são formadas pelas opções:

- *Insere TV*
- *Ativa TV*
- *Preenche TV*
- *Verifica Consistência TV*
- *Desativa TV*

Para facilitar o entendimento das interfaces do PTV Front-end, vai ser usado o mesmo exemplo da tabela de Funcionários que foi utilizado nos capítulos anteriores. Como será mostrada a seguir, a ordem da execução das funções segue a estrutura do menu para uma mesma tabela, embora todas opções mantenham-se disponíveis. O usuário deve atentar para detalhes como não aparecer no formulário *PreencheTV* tabelas que não tiveram o tempo de validade ativado, bem como não é possível ativar o tempo de validade (*Ativa TV*) em uma tabela que não teve os atributos temporais inseridos (*Insere TV*).

No exemplo a seguir, a premissa inicial é que a tabela FUNC tem a mesma estrutura da Tabela 12. Inicialmente será executada a função *Insere Atributo TV*, como pode ser visto na figura 6.42:

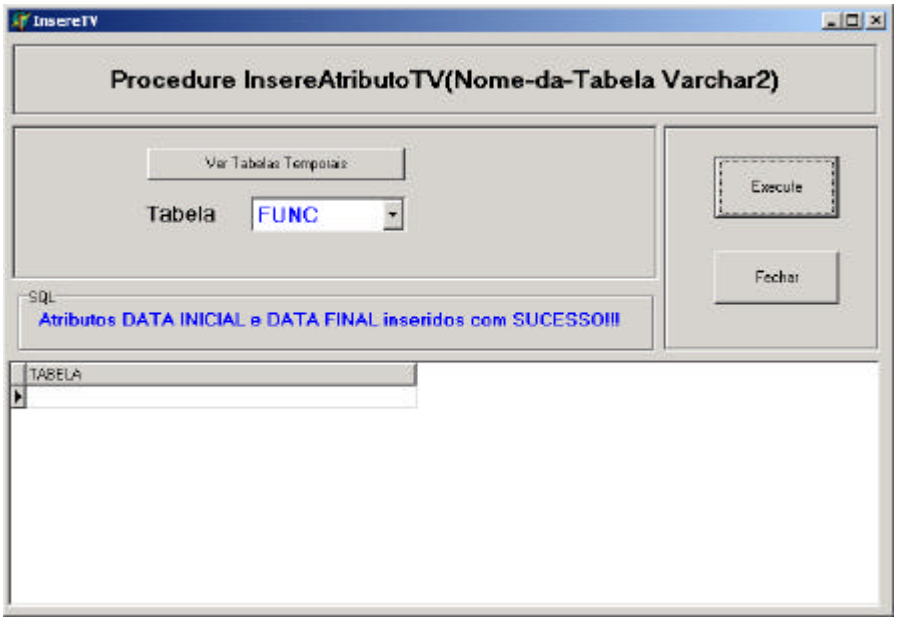


Figura 6.42: Insere Atributo de Tempo de Validade

Esse procedimento é equivalente ao mostrado na figura 4.19.

O próximo passo é ativar o controle de consistência do tempo de validade do PTV. Esse procedimento foi realizado manualmente na figura 4.20. Agora, na figura 6.43, o mesmo processo se repete de forma visual.

O procedimento para ativação do tempo de validade é bastante simples, bastando selecionar a tabela na qual se deseja ativar o tempo de validade, na caixa de listagem TABELA. Em seguida, deve ser pressionado o botão *EXECUTE*. Para verificar se a tabela foi ativada, basta verificar a caixa de mensagem chamada SQL e ver o resultado. No exemplo, é possível ver que o procedimento foi realizado com sucesso. O botão *VER TABELAS TEMPORAIS* tem como função mostrar as tabelas que tem ativado o mecanismo de tempo de validade.

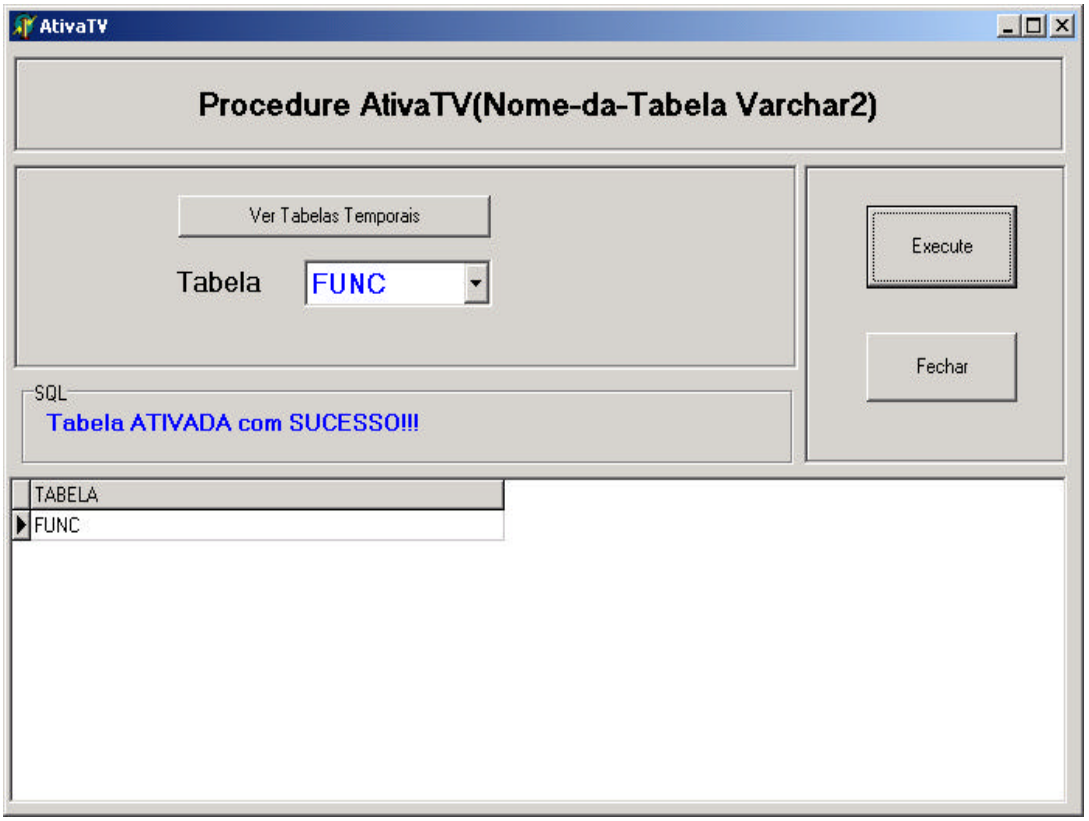


Figura 6.43: Ativação do tempo de validade.

Uma vez ativado o tempo de validade, o próximo passo será o preenchimento dos atributos *ID*, *TvalidadeI* e *TvalidadeF* inseridos na figura 6.42. Esse procedimento, visto na figura 6.43, tem como principal objetivo facilitar ao usuário o preenchimento desses dados. Essa função pode não ser tão relevante caso a tabela tenha poucas tuplas, mas há situações em que essas tabelas possuem milhares de registros. Para esses casos, ela se mostra bastante eficiente.

O procedimento consiste em selecionar a tabela na caixa de listagem *TABELA* que está em destaque na figura 6.44, e acionar o botão *EXECUTE*. Nessa caixa de listagem aparecem todas as tabelas que têm ativado o controle de tempo de validade. Se o procedimento for executado sem erros, aparecerá a mensagem “Dados preenchidos com sucesso!!!” na caixa de texto *SQL*.

Outro aspecto importante é a visualização dos dados preenchidos. Na figura 6.44, é possível ver os dados dos atributos *ID*, *TvalidadeI* e *TvalidadeF*.

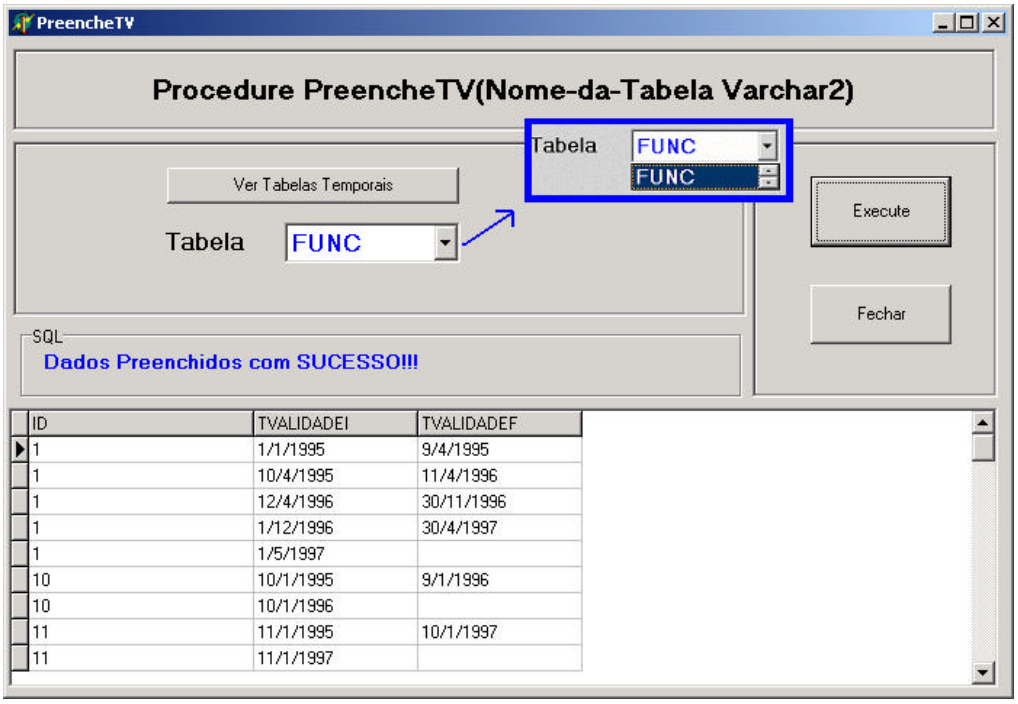


Figura 6.44: Execução da função Preenche Tempo de Validade.

Tecnicamente, essa tabela está pronta para ser usada, com os recursos temporais.

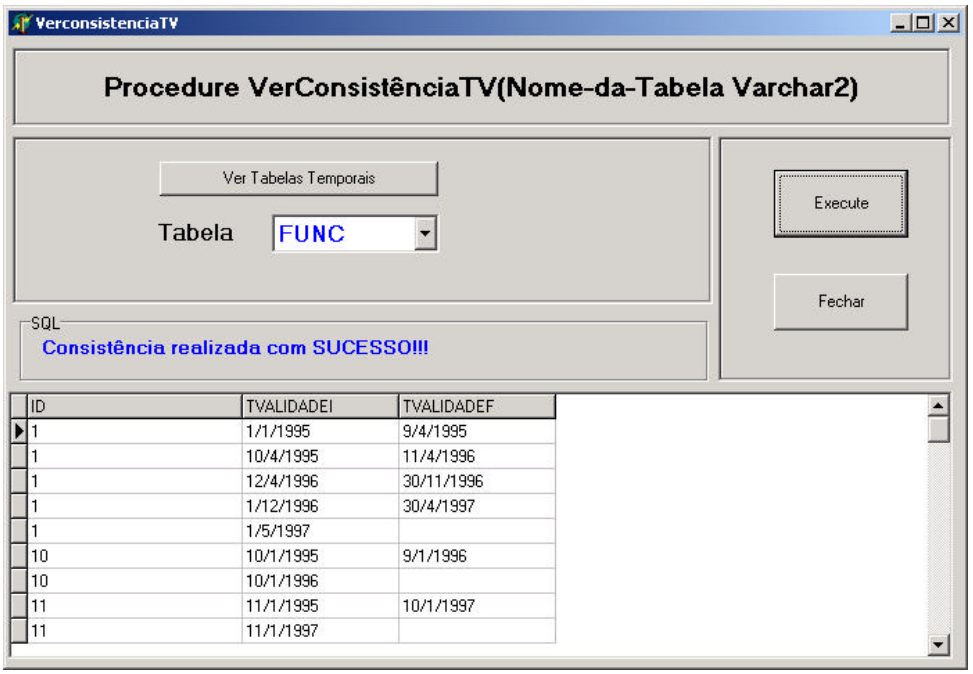


Figura 6.45: Função para verificação de consistências temporais.

A figura 6.45 mostra um procedimento do PTV que tem como objetivo fazer uma verificação em suas tuplas, conforme feito na figura 4.21 de forma interativa. As regras utilizadas por esse procedimento foram comentadas em seções anteriores.

6.2 Menu Consultas

A partir dessa seção serão mostradas as funções criadas para o PTV, apresentadas no capítulo 5. Essas funções foram dividas em dois grupos maiores chamados Agregação e Conversão. Dentro do grupo de Agregação, as seguintes consultas estão disponíveis: Tcount, Tsum, Tavg, Tmax e Tmin. Já dentro do grupo Conversão há as seguintes funções: Tbegin, Tend, Tperiod, Tintersect, Tfirst e Tlast.

As interfaces visuais para todas as funções são bastante semelhantes. A figura 6.46 mostra o formulário que visualiza a execução da função Tcount.

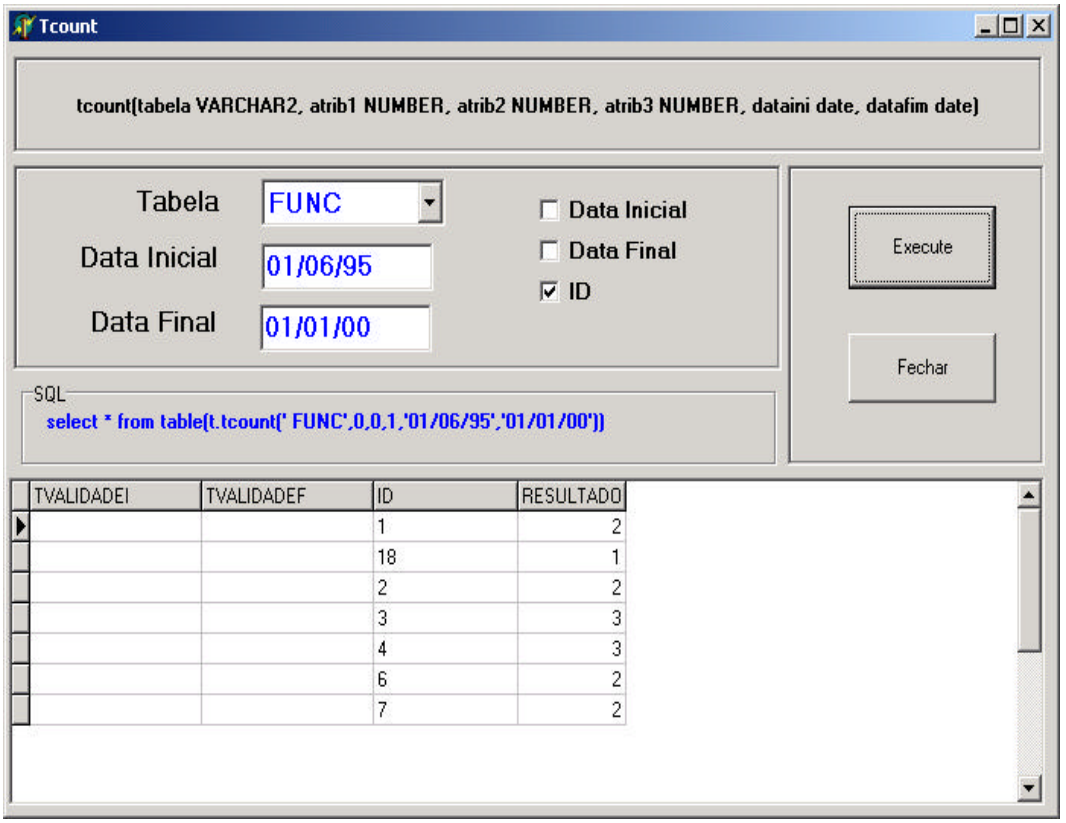


Figura 6.46: Execução da função *Tcount*.

Esta figura mostra o mesmo resultado apresentado na figura 5.36. Na caixa SQL é mostrada a sentença SQL que originou o resultado. As caixas de texto Data Inicial e data final representam o período escolhido para o filtro. Os *checkbox* Data Inicial, Data Final e ID indicam os parâmetros para a contagem das tuplas. Caso o usuário não selecione nenhum desses, na caixa SQL é mostrada uma mensagem de advertência: "Você deve marcar pelo menos um dos CAMPOS de checkbox". As demais funções de consultas para o grupo agregação são mostradas no anexo B, por se tratarem de uma repetição de telas com funções já discutidas nas seções anteriores.

A figura 6.47 mostra a interface visual para a função *Tbegin*, do grupo de funções de Conversão. É possível verificar que esse formulário é bastante semelhante ao mostrado na figura 6.46. A estrutura básica é a mesma, alterando-se apenas os parâmetros das funções. Com relação às mensagens, botões de execução, saída, e resultado da consulta são iguais para todas as funções de consulta.

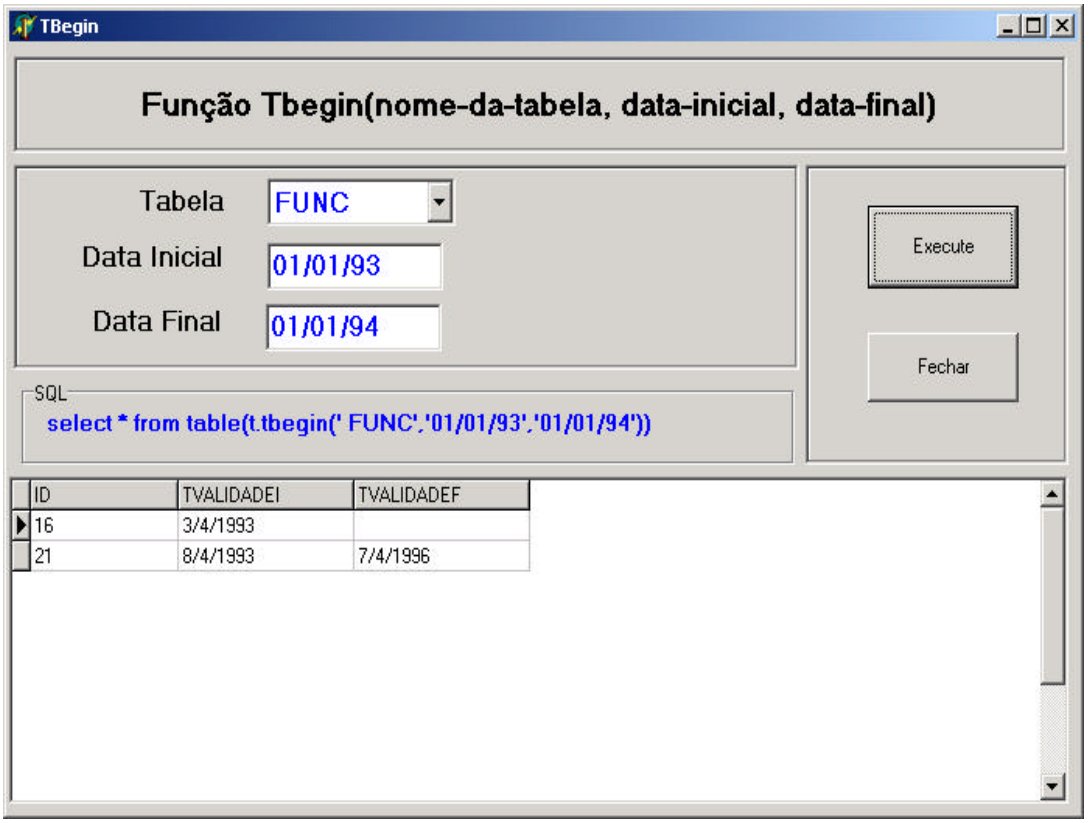


Figura 6.47: Função de consulta *Tbegin*.

O resultado dessa função foi comentado no capítulo anterior. Serão feitos comentários apenas dos objetos do formulário. A lista Tabela define qual tabela será utilizada pela consulta. O conjunto Data Inicial e Data Final representam o período escolhido para a execução. Pela semelhança dos formulários que se seguem e pela explanação sobre as consultas mencionadas no capítulo anterior, as funções visuais dessa interface serão mostradas no anexo C.

6.3 Considerações finais

Neste capítulo foi mostrado o uso operacional do PTV Front-end. Com os exemplos apresentados foi possível verificar através de objetos como *listbox* e *checkbox*, a facilidade de manipulação dessa ferramenta, tanto nos procedimentos quanto nas funções de consultas temporais, maximizando o uso das funções do PTV.
Com o PTV Front-end, o usuário não necessita de conhecimento profundo de SQL para manipulação de dados temporais. Para cada uma das tarefas, há sempre a sentença SQL que é executada no banco de dados, permitindo ao usuário saber quais comandos estão sendo realizado. Finalmente, com o PTV Front-end não é necessário o uso de outras ferramentas adicionais de acesso ao banco de dados.

7 ESTUDO DE CASO

Neste capítulo será apresentado um estudo de caso para exemplificar o uso das funções de consulta desenvolvidas neste trabalho. Não será apresentada a modelagem completa do sistema em estudo. A figura 6.48 mostra um diagrama de classe simplificado apenas com os atributos e as principais tabelas ou classes.

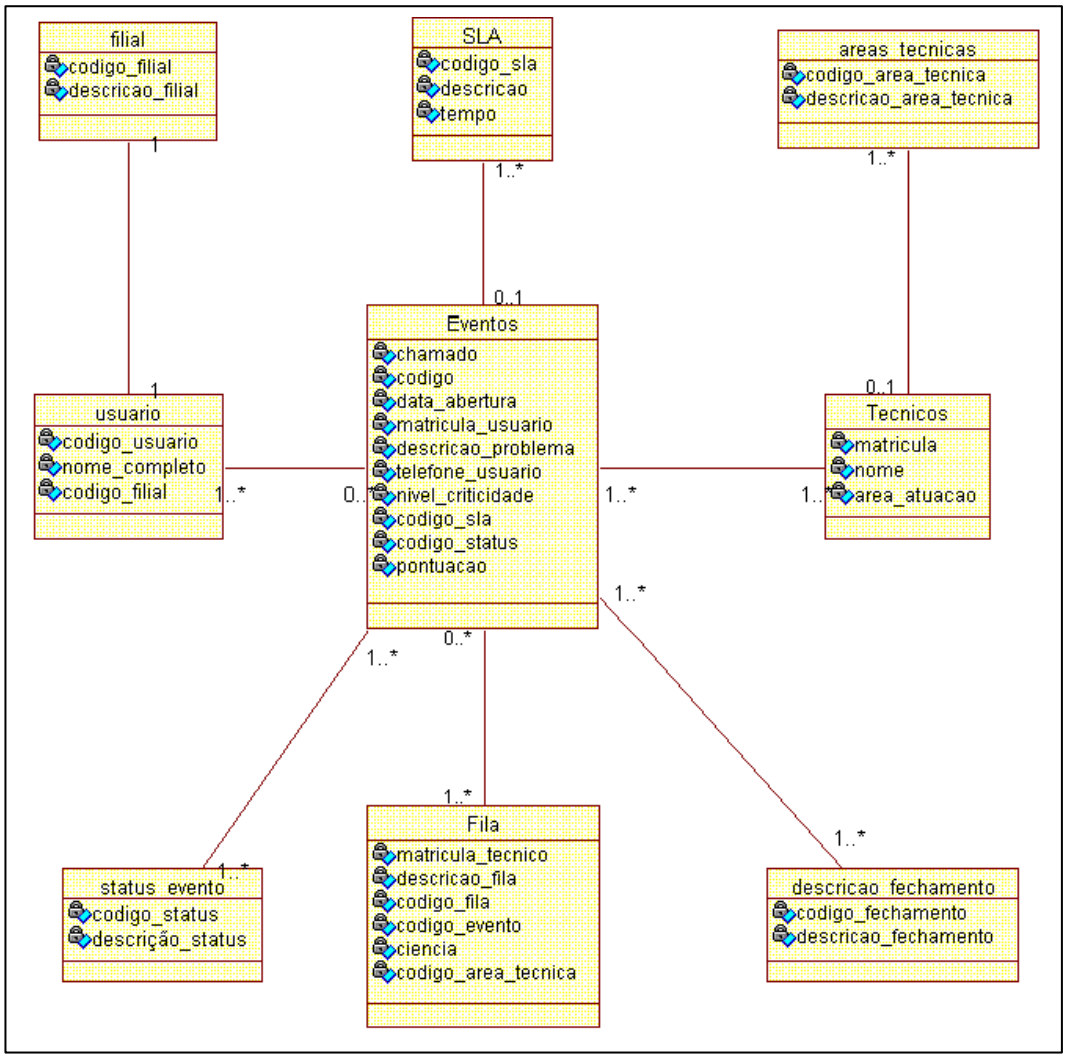


Figura 6.48: Diagrama de Classe

O sistema cujo diagrama de classe corresponde a uma aplicação de *HelpDesk* para atendimento de chamados de suporte técnico de informática. Esse modelo foi baseado no sistema *Remmedy* desenvolvido pela IBM, cujo funcionamento é o seguinte: à medida em que os chamados vão sendo registrados, um operador encaminha cada um a um técnico. Cada técnico só recebe um chamado por vez, recebendo o próximo somente quando o chamado atual é encerrado.

Abaixo segue a estrutura das tabelas que serão utilizadas no uso do PTV.

Eventos

Nome	Nulo?	Tipo
CODIGO	NOT NULL	VARCHAR2(10)
CHAMADO	NOT NULL	VARCHAR2(10)
MATRICULA_USUARIO	NOT NULL	VARCHAR2(10)
DESCRICAO_PROB	NOT NULL	VARCHAR2(50)
TELEFONE_USUARIO	NULL	VARCHAR2(50)
DATA_ABERTURA	NOT NULL	DATE
NIVEL_CRITICIDADE	NOT NULL	VARCHAR2(2)
CODIGO_SLA	NOT NULL	VARCHAR2(2)
CODIGO_STATUS	NOT NULL	VARCHAR2(2)
PONTUACAO	NULL	NUMBER(2)

O atributo CODIGO é chave estrangeira do atributo MATRÍCULA da tabela TECNICO. Por questão de simplificação e adaptações para uso do PTV, apenas os atributos sublinhados serão utilizados nos exemplos desse capítulo. O atributo DATA_ABERTURA será renomeado para TSTAMP e representa a data do encaminhamento do chamado ao técnico. O atributo PONTUACAO define o grau de importância daquele chamado, que pode variar de 1 a 10.

Tecnicos

Nome	Nulo?	Tipo
MATRICULA	NOT NULL	VARCHAR2(10)
NOME	NOT NULL	VARCHAR2(50)
AREA_ATUACAO	NOT NULL	VARCHAR2(10)

A tabela TECNICOS será implementada com foi mostrado anteriormente, sem modificação alguma. A seguir serão mostrados os passos para o uso do PTV.

7.1 Inserção dos atributos de Tempo de Validade

A figura 6.49 mostra o procedimento para inserção dos atributos ID, TvalidadeI e TvalidadeF que serão utilizados pelas funções temporais.

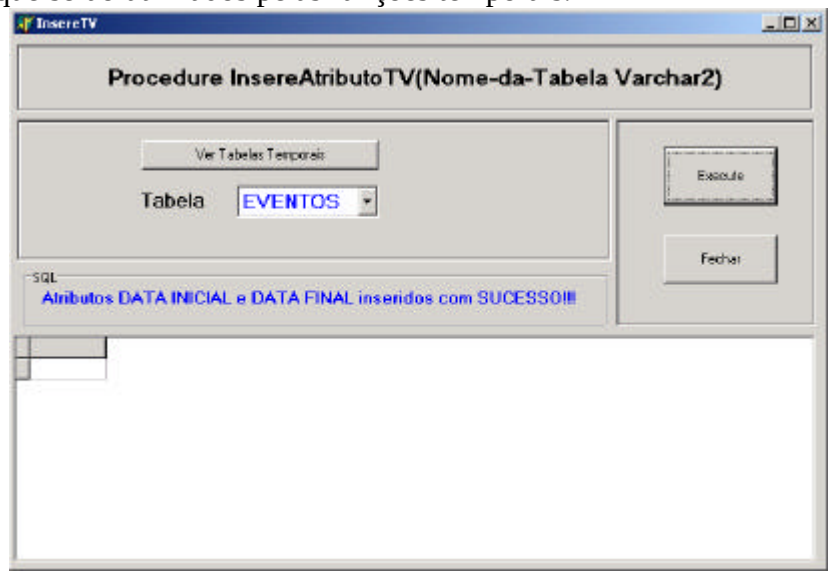


Figura 6.49: Inserção de atributos para uso do PTV.

7.2 Ativação do PTV

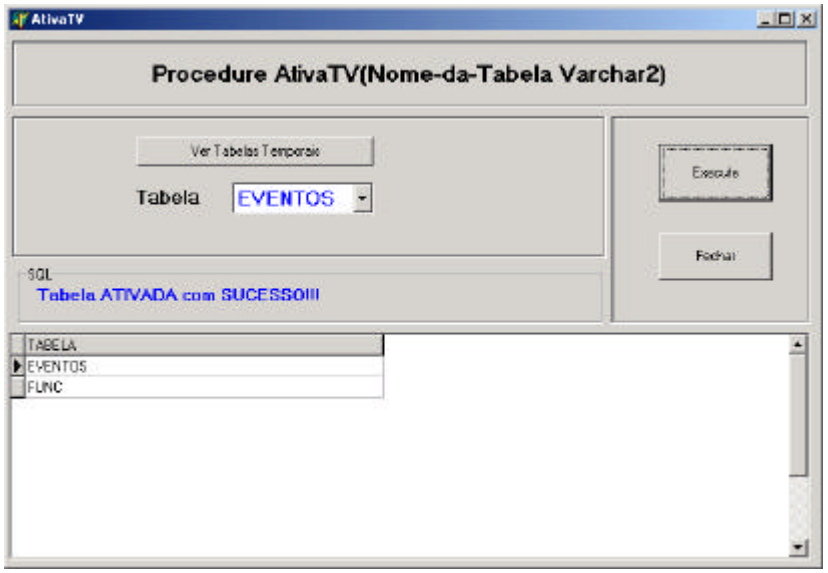


Figura 6.50: Ativação do PTV

A figura 6.50 mostra a ativação do PTV para a tabela EVENTOS. A partir desse momento todo mecanismo de inserção temporal será feito pelo PTV, bem como será possível o uso das funções de consultas temporais, pois o nome da tabela torna-se disponível para a ferramenta.

7.3 Preenchimento dos atributos temporais

A função PreencheTV mostrada na figura 6.51 insere os dados nos atributos ID, TvalidadeI e TvalidadeF. O funcionamento dessa função pode ser vista no capítulo anterior.

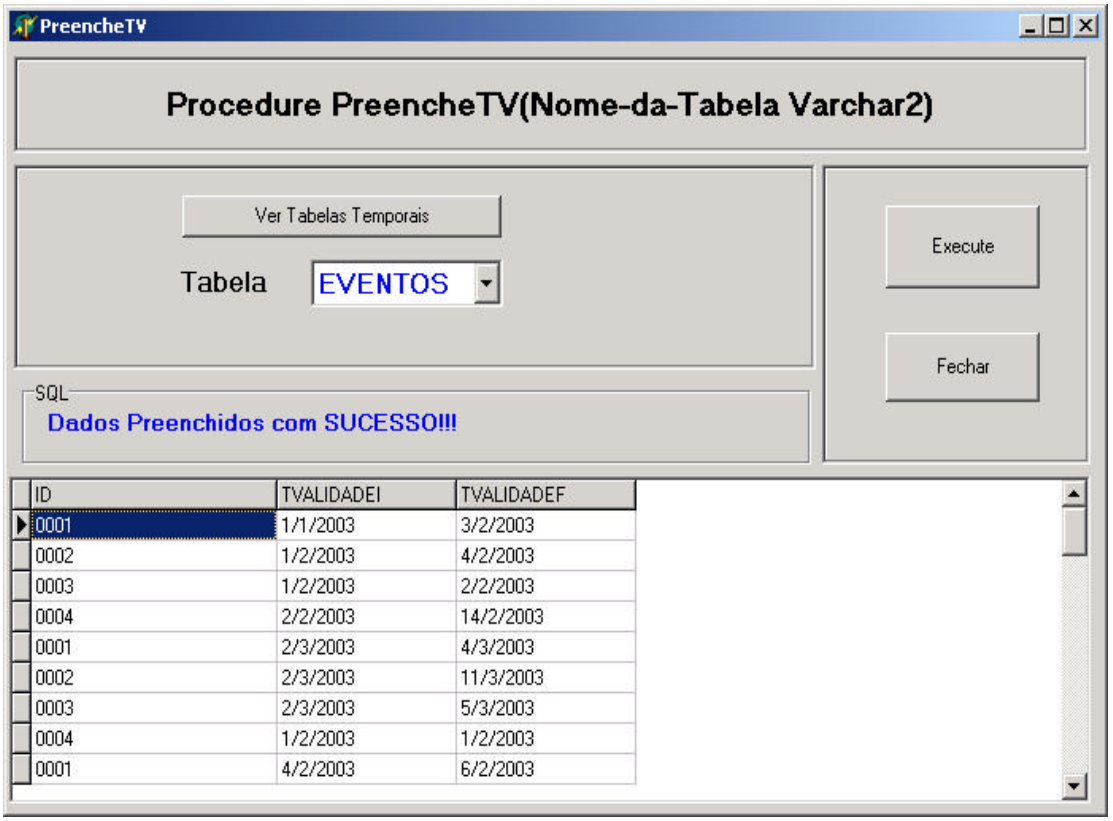


Figura 6.51: Função PreencheTV.

7.4 Verificação de Consistência

A função mostrada na figura 7.52 tem como objetivo fazer uma verificação dos dados temporais já inseridos. Ela é utilizada principalmente quando o mecanismo do PTV é acionado após a inserção de dados.

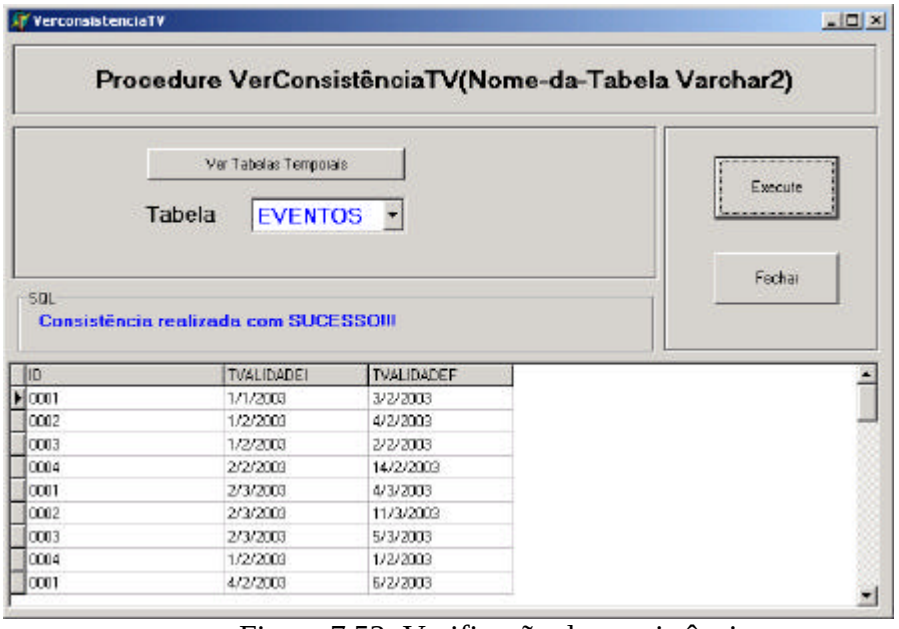


Figura 7.52: Verificação de consistência

7.5 Quantidade de chamados por técnico

Uma vez realizadas as tarefas do item 3.1 até 3.3, serão realizados testes de consultas temporais, nos quais serão exploradas suas funcionalidades. A figura 7.53 ilustra o uso da função TCOUNT. Neste exemplo é possível ver como resultado a quantidade de chamados por técnicos.

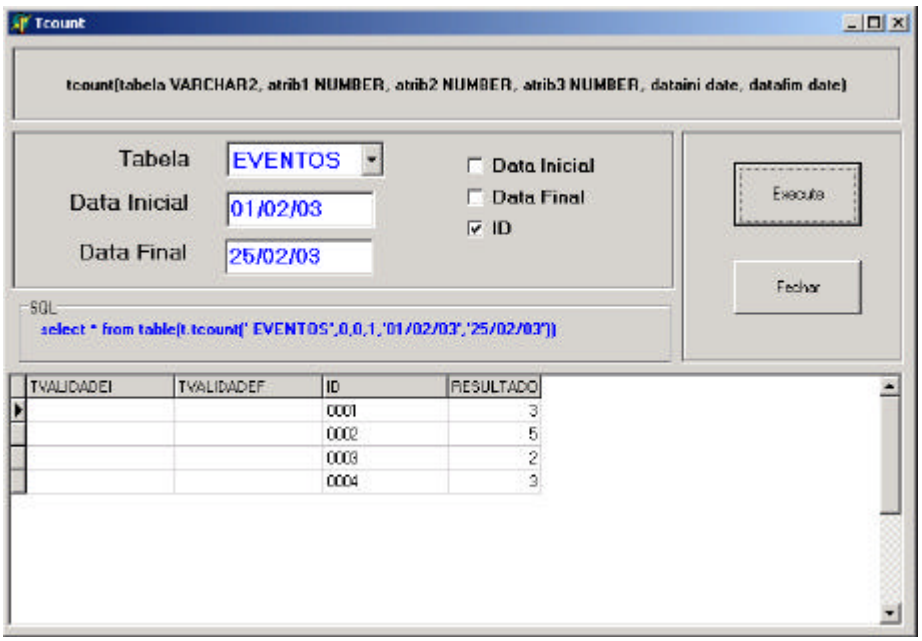


Figura 7.53: Função Tcount

7.6 Quantidade de chamados

A figura 7.54 mostra o uso da função Tcount que nesse momento apresenta a lista com as quantidades de chamados encaminhados por dia em um dado período. Através dessa função ainda é possível visualizar a consulta anterior, só que agrupada pelo tempo de encerramento do chamado.

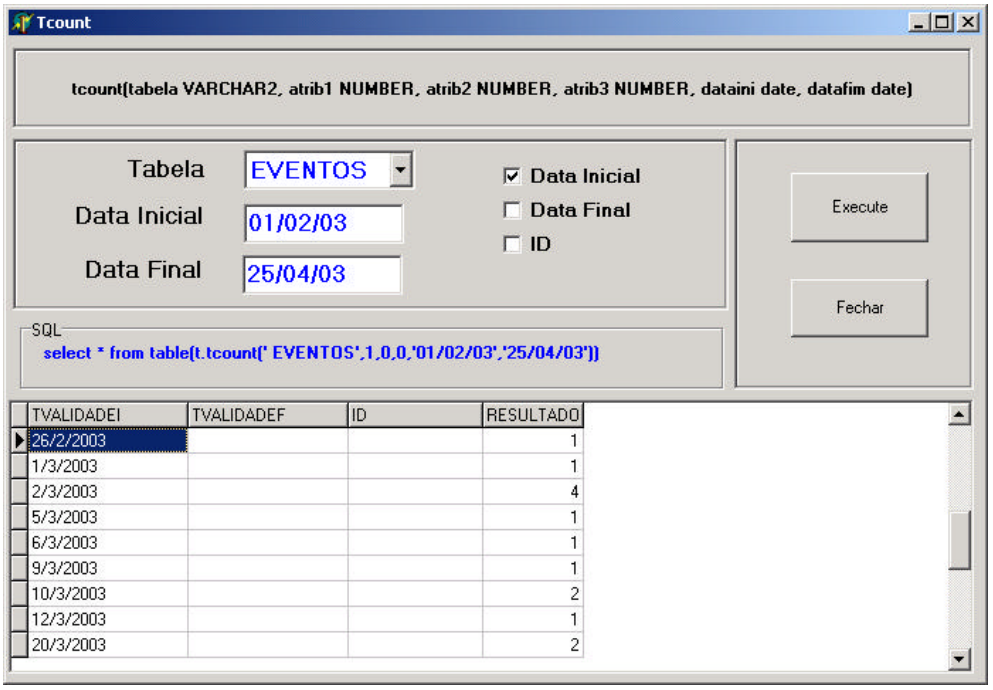


Figura 7.54: Chamados por data de encaminhamento.

7.7 Chamados abertos em determinado período

O próximo exemplo ilustra o uso da função Tbegin. Neste caso, a figura 7.55 apresenta os chamados que foram encaminhados aos técnicos no período de 01/03/2003 a 05/03/2003.

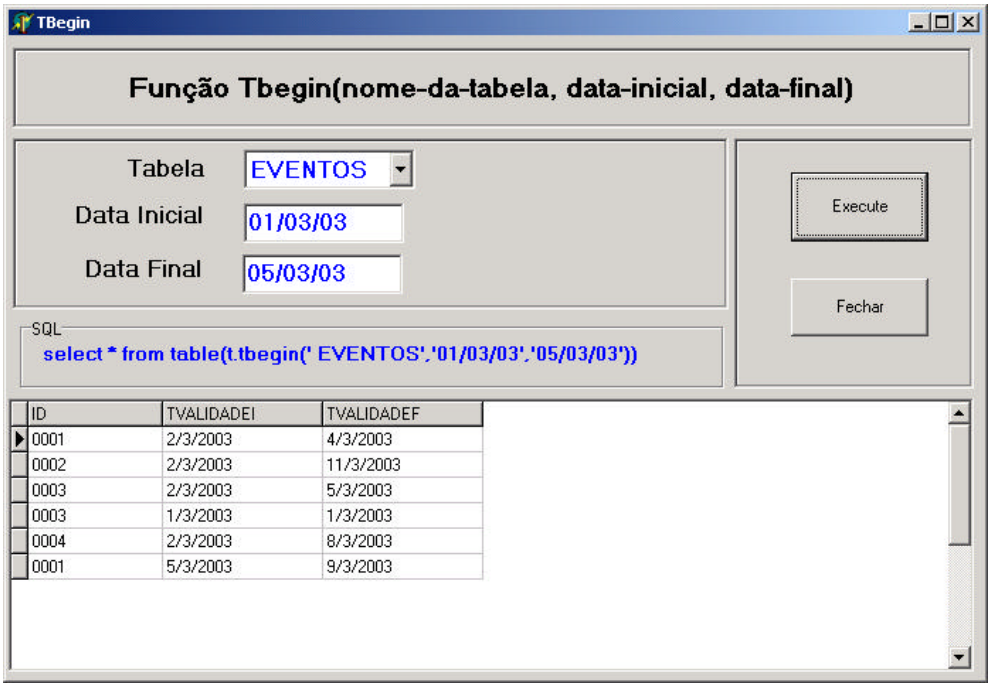


Figura 7.55: Eventos encaminhados em determinado período.

7.8 Chamados abertos em determinado período

Na figura 7.56 é mostrado um exemplo de consulta em que se utiliza a função Tend. Nesse exemplo, é mostrada a relação de eventos que foram encerrados em um dado intervalo de tempo.

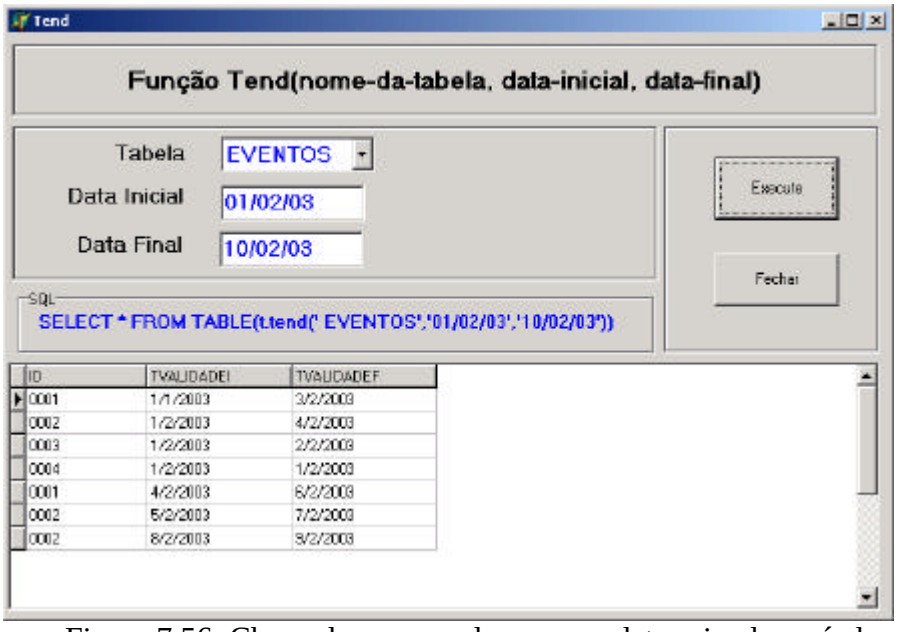


Figura 7.56: Chamados encerrados em um determinado período

7.9 Chamados abertos em determinado período

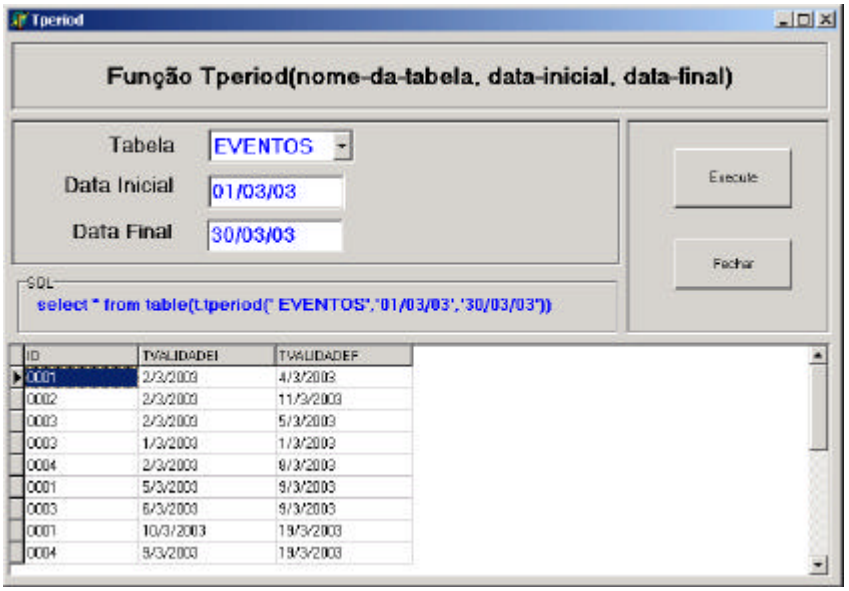


Figura 7.57: Chamados por período

A figura 7.57 acima mostra uma relação de chamados que foram encaminhados e encerrados em um dado período.

7.10 Chamados abertos em um determinado período

O próximo exemplo mostra o uso da função Tfirst. Nesse caso é mostrada a primeira data de encaminhamento de chamado a partir de um dado intervalo de tempo.

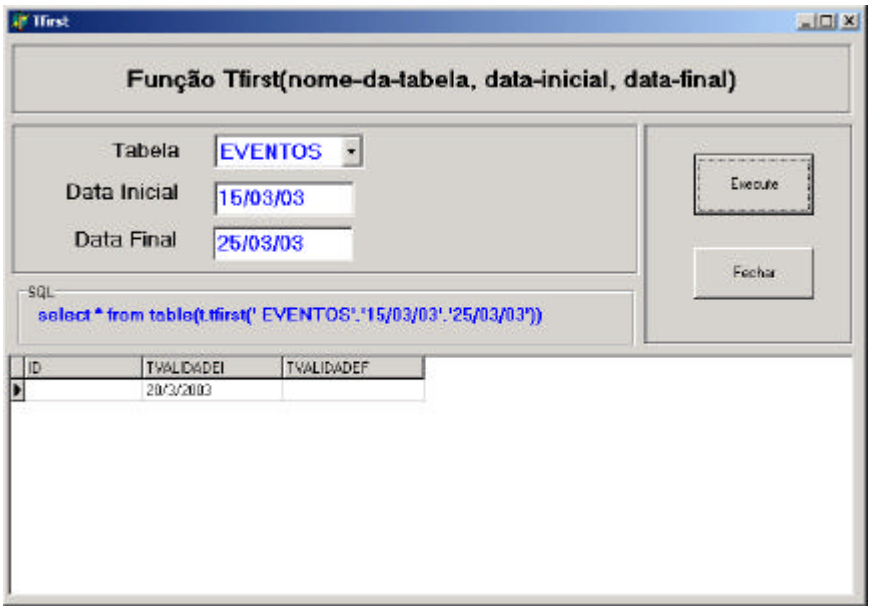


Figura 7.58: Primeira data de encaminhamento.

7.11 Chamados abertos em um determinado período

O próximo exemplo, apresentado na figura 7.59, ilustra um exemplo semelhante ao mostrado na figura 7.58. Neste caso é possível verificar, como resultado, a última data de encerramento de chamado que satisfaça a condição ter data limite no período de 25/02/2003 e 10/03/2003, ou seja a data de encerramento de chamado que esteja no intervalo acima e que mais se aproxime da data final.

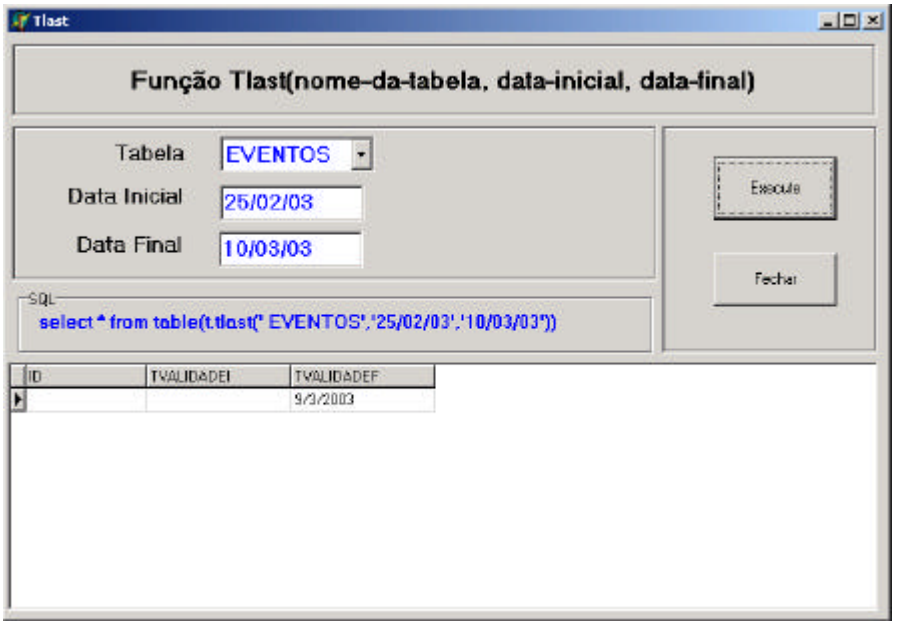


Figura 7.59: Data de encerramento.

7.12 Função Tintersect

A figura 7.60 mostra um exemplo no uso da função Tintersect, mostrando os chamados que têm suas datas de encaminhamento e encerramento dentro da intersecção entre os intervalos 1 e 2.

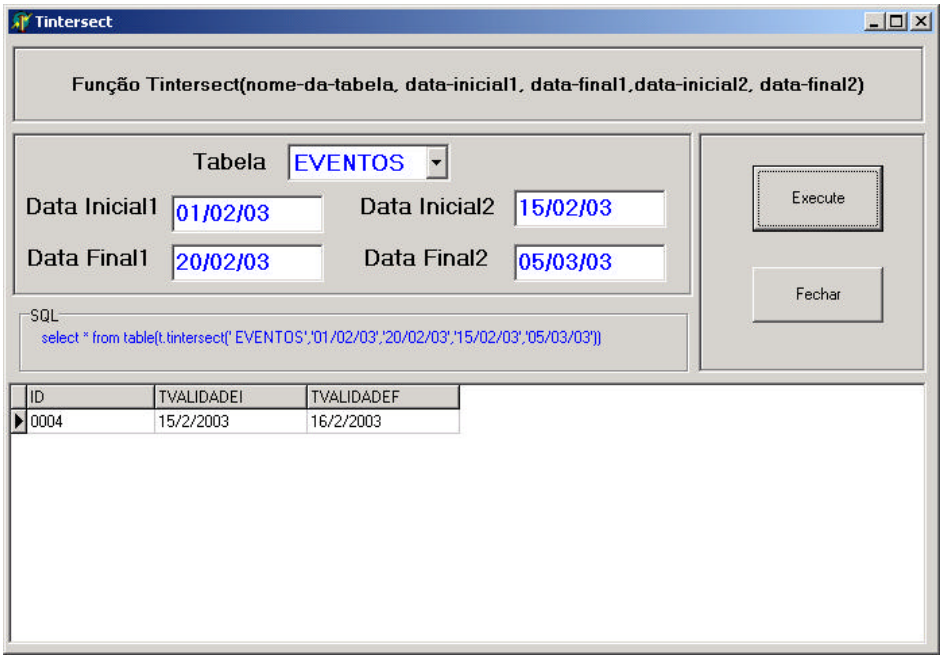


Figura 7.60: Uso da função Tintersect

7.13 Média da Pontuação dos Chamados

No exemplo da figura 7.61 é mostrada a média da pontuação dos chamados dos técnicos no período de 10/02/2003 a 20/03/2003.

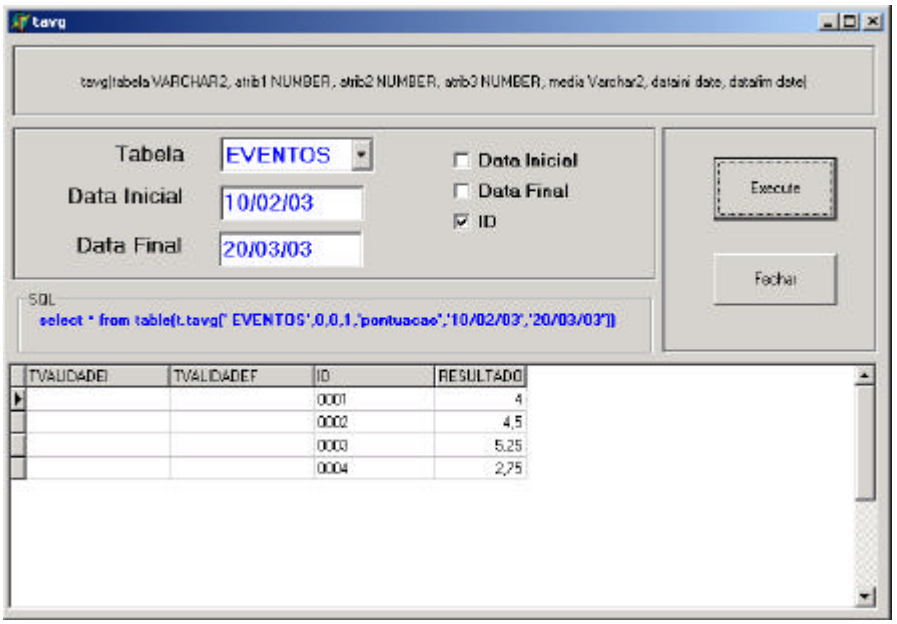


Figura 7.61: Média da pontuação dos chamados

A consulta da figura 7.62 mostra os chamados de maior pontuação no período de 01/03/2003 a 30/03/2003, para os técnicos que atuaram nesse período.

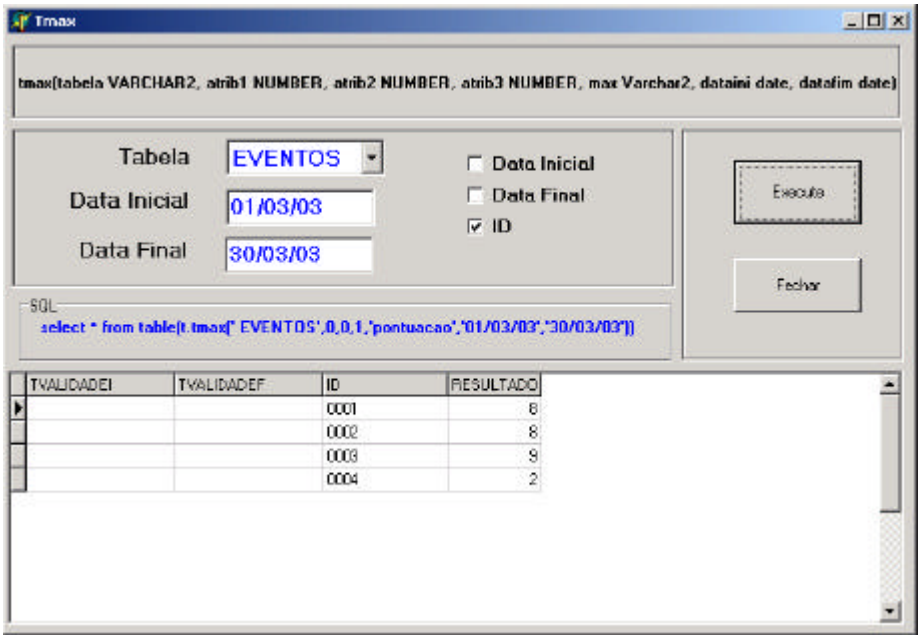


Figura 7.62: Pontuação máxima do período

7.14 Considerações finais

O objetivo desse capítulo foi mostrar, de uma forma prática, o uso da ferramenta visual do PTV, bem como as funcionalidades das consultas temporais desenvolvidas nesse trabalho. Algumas funções ficaram de fora do estudo de caso por realizarem tarefas equivalentes às mostradas nas figuras anteriores. Nesse exemplo não foram avaliados aspectos de performance e nem foi explorado ao máximo o uso das funções.

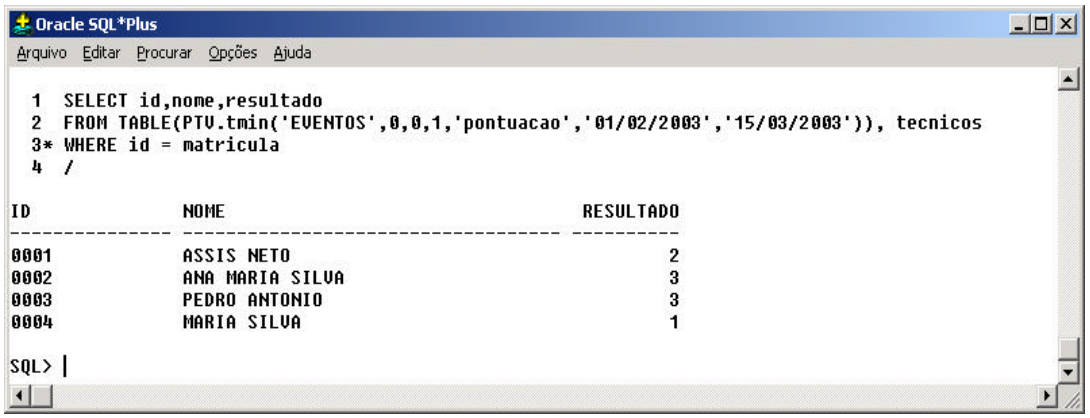


Figura 7.63: SQL com a função Tmin

Por fim, apenas para ilustrar mais uma utilização desse pacote, é possível utilizar as funções desenvolvidas nesse trabalho como parte de uma sentença SQL. A figura 7.63 mostra uma sentença SQL na qual é feito um *join* entre a função Tmin e a tabela TECNICOS, resultando nos chamados de menor valor de pontuação para cada um dos técnicos que tiveram chamados no período de 01/02/2003 e 15/03/2003.

8 CONCLUSÃO

Apesar de avanços sob o aspecto temporal, o SGBD *Oracle* 9i ainda está bastante incipiente no que diz respeito a funções de SGBDT, embora seja possível implementar pacotes e funções com essa finalidade. Em comparação com a versão 8 desse SGBD, é possível perceber que a funções temporais tiveram um ganho em termos de performance e flexibilidade no uso das funções.

O PTV é uma proposta de uma camada de software sobre um SGBD relacional que simula as funções temporais de tempo de validade.. Esse pacote foi desenvolvido no intuito de fazer e garantir a consistência dos dados temporais, permitindo também que uma tabela pudesse ter seu mecanismo temporal ativado e desativado sem prejuízo de seus dados originais. Entretanto, o PTV ainda apresentava carências de funções que permitissem o uso de consultas temporais de uma forma mais transparente ao usuário.

Em virtude dessa carência, a proposta desse trabalho foi desenvolver funções temporais baseadas em um modelo de consultas de tempo já disponível na literatura, o TSQL2. Com isso, pode-se ampliar as funcionalidades do PTV e mostrar a eficiência de parte da proposta (*subset*) do *TSQL2* na prática, pois essa modelagem não é implementada comercialmente. O *TSQL2* foi escolhido como modelo para a construção das funções de tempo por ser uma proposta de modelagem bastante aceita e por seguir o padrão ANSI que é implementado pelo SGBD *Oracle* 9i, permitindo, com isso, maior facilidade na implementação desse modelo.

Esse trabalho também sugeriu uma interface visual para manipulação dos dados temporais a partir do PTV. Seu objetivo foi disponibilizar aos estudiosos da área e aos desenvolvedores, uma ferramenta que facilitasse o estudo de BDT de uma forma prática na qual seja possível verificar resultados. Essa interface visual foi desenvolvida na linguagem Delphi versão 5 com acesso ao SGBD *Oracle* 9i.

Com a adição de funções que permitem realizar consultas temporais e um *Font-End* para facilitar a manipulação de seus dados, o PTV torna-se uma proposta de ferramenta temporal para uso de DBT com Tempo de Validade bastante completa, pois, a partir de agora, além do controle de consistência é possível também analisar através de consultas temporais seus resultados, de uma forma centralizada e visual.

8.1 Contribuições principais

Esse trabalho acrescentou novas funcionalidades temporais ao PTV, basicamente sob o aspecto de funções temporais. Outra contribuição foi a implementação de uma interface para o PTV, facilitando a elaboração de consultas temporais.

8.2 Propostas de Trabalhos Futuros

A implementação no PTV do mecanismo de Tempo de Transação consiste em uma possibilidade de ampliação das funções de tempo desse pacote, permitindo assim um acréscimo de suas funcionalidades. Outro ponto a ser trabalhado é a ampliação das funções da Interface visual do PTV. Esta pode evoluir para uma interface gráfica de administração desse pacote, assim como o *Enterprise Manager Console* do Oracle 9i.

Finalmente, outra possibilidade de trabalho nessa linha seria a implementação das funções do PTV em um SGBD de software livre, como o PostgreSQL e o MySQL, pois uma vez que se tenha acesso ao código fonte do SGBD, seria bastante interessante a implementação do PTV diretamente no Kernel do Sistema.

REFERÊNCIAS

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

[CLI 93] CLIFFORD, J.; CROCKER, A. The Historical Relational Data Model (HRDM) Reviseted. In: **Temporal Databases**: theory, design, and implementation. Redwood City: The Benjamin/Cummings, 1993.

[EDE 94] EDELWEISS, N.; OLIVEIRA, J. P. M**. Modelagem de aspectos temporais de sistemas de informação.** Recife: Universidade Federal de Pernambuco, 1994. Trabalho apresentado na 9. Escola de Computação, 1994,Recife.

[EDE 98] EDELWEISS, N. Banco de Dados Temporais: Teoria e Prática. In: JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA,JAN,17., 1998, Belo Horizonte. **Anais...** Belo Horizonte: SBC, 1998. 335p. p.225-282.

[ELM 93] ELMASRI, R.; WUU, G.T.J. A temporal model and query language for EER databases. In: **Temporal Databases:** Theory, Design, and Implementation. Redwood City: Benjamin/Cummings, 1993.

[JEN 98] JENSEN, C.S. et al. The Consensus Glossary of Temporal Database Concepts. In: ETZION, O.; JAJODIA, S.;SRIPADA, S.(Ed.). **Temporal Databases**: research and practice. Berlin: Springer, 1998. p.367-405

[ORA 99] ORACLE 8i Time Series User´s Guide: Release 8.1.5 A67294-01. Disponível em: <www.oracle.com>. Acesso em: 05 jun.2003.

[ORA 2000] MIGRATING from Oracle Time Series to SQL Analytic Functions: Time Series Functions. Disponível em: <http://www.oracle.com/database/option>. Acesso em: 20 maio 2003.

[ORA 2002] ORACLE 9i SQL Reference: Release 2 (9.2) Part No. A96540-01. Disponível em: <http://metalink.oracle.com>. Acesso em: 30 maio 2003.

[ORA 2003] TIME Series, Release 8.1.6 DOC ID 93837.1. Disponível em: < http://metalink.oracle.com>. Acesso em: 30 maio 2003.

[PIN 2003] PINHEIRO, S. F. **PTV – Pacote de Tempo de Validade para o SGBD Oracle**. 2001. 75p. Dissertação (Mestrado em Ciência da Computação – Instituto de Informática, UFRGS, Porto Alegre.

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

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

[SAR 90] SARDA, N. L. Extensions to SQL for historical databases. **IEEE Transactions on Knowledge and Data Engineering**, New York, v.2, n.2, June 1990.

[SAR 93] SARDA, N. L. HSQL: A Historical query language. In: **Temporal Databases:** Theory, Design, and Implementation. Redwood City: Benjamin/Cummings, 1993.

[TIM 99] TIME DB, A temporal Relational DBRMS. Disponível em: <http://www.timeconsult.com/software/software.html>. Acesso em: jun. 2002.

ANEXO A CÓDIGOS FONTE

ANEXO A1 Código Fonte da Função TFIRST

```
FUNCTION tfirst(tabela VARCHAR2, dataini date, datafim date)  RETURN
COLECAOTV PIPELINED is
TYPE Cursores is REF CURSOR;
c1 Cursores;
resultado TVALIDADE := TVALIDADE(NULL,NULL,NULL);
tvi date;
BEGIN
 OPEN c1 FOR
  CASE
    WHEN (dataini is not null) AND (datafim is not null) then
    'select distinct first_value(tvalidadei) over (order by tvalidadei asc rows unbounded
preceding)
    from '||tabela||' where
    (tvalidadei >='||''''||TO_CHAR(dataini,'dd/mm/yyyy')||''''|| 'and
    tvalidadei <='||''''||TO_CHAR(datafim,'dd/mm/yyyy')||''''||')'
    WHEN (dataini is null) AND (datafim is null) then
    'select distinct first_value(tvalidadei) over (order by tvalidadei asc rows unbounded
preceding)
    from '||tabela||''
    WHEN (dataini is null) then
    'select distinct first_value(tvalidadei) over (order by tvalidadei asc rows unbounded
preceding)
    from '||tabela||' where
    (tvalidadei <='||''''||TO_CHAR(datafim,'dd/mm/yyyy')||''''||')'
    WHEN (datafim is null) then
    'select distinct first_value(tvalidadei) over (order by tvalidadei asc rows unbounded
preceding)
    from '||tabela||' where
    (tvalidadei >='||''''||TO_CHAR(dataini,'dd/mm/yyyy')||''''||')'
  END;
  LOOP
     FETCH c1 INTO tvi;
     EXIT WHEN c1%NOTFOUND;
     resultado.tvalidadei := tvi;
```

```
    PIPE ROW(resultado);
  END LOOP;
  CLOSE c1;
  RETURN;
END;
```

ANEXO A2 Código Fonte da Função TLAST

```
FUNCTION tlast(tabela VARCHAR2, dataini date, datafim date)  RETURN
COLECAOTV PIPELINED is
TYPE Cursores is REF CURSOR;
c1 Cursores;
resultado TVALIDADE := TVALIDADE(NULL,NULL,NULL);
tvf date;
BEGIN
 OPEN c1 FOR
  CASE
     WHEN (dataini is not null) AND (datafim is not null) then
    'select distinct first_value(tvalidadef) over (order by tvalidadei desc rows unbounded
preceding)
    from '||tabela||' where
    (tvalidadef >='||""||TO_CHAR(dataini,'dd/mm/yyyy')||""|| 'and
    tvalidadef <='||""||TO_CHAR(datafim,'dd/mm/yyyy')||""||')'
     WHEN (dataini is null) AND (datafim is null) then
    'select distinct first_value(tvalidadef) over (order by tvalidadei desc rows unbounded
preceding) from '||tabela||"
     WHEN (dataini is null) then
    'select distinct first_value(tvalidadef) over (order by tvalidadei desc rows unbounded
preceding) from '||tabela||' where
    (tvalidadef <='||""||TO_CHAR(datafim,'dd/mm/yyyy')||""||')'
     WHEN (datafim is null) then
    'select distinct first_value(tvalidadef) over (order by tvalidadei desc rows unbounded
preceding) from '||tabela||' where
    (tvalidadef >='||""||TO_CHAR(dataini,'dd/mm/yyyy')||""||')'
  END;
  LOOP
     FETCH c1 INTO tvf;
     EXIT WHEN c1%NOTFOUND;
     resultado.tvalidadef := tvf;
     PIPE ROW(resultado);
  END LOOP;
  CLOSE c1;
  RETURN;
END;
```

ANEXO A3 Código Fonte da Função TBEGIN

```
FUNCTION tbegin(tabela VARCHAR2, dataini date, datafim date)  RETURN
COLECAOTV PIPELINED is
TYPE Cursores is REF CURSOR;
c1 Cursores;
resultado TVALIDADE := TVALIDADE(NULL,NULL,NULL);
tvi date;
tvf date;
vid varchar2(10);
BEGIN
 OPEN c1 FOR
  CASE
    WHEN (dataini is not null) AND (datafim is not null) then
    'select id,tvalidadei, tvalidadef
    from '||tabela||' where
    (tvalidadei >='||""||TO_CHAR(dataini,'dd/mm/yyyy')||""|| 'and
    tvalidadei <='||""||TO_CHAR(datafim,'dd/mm/yyyy')||""||')'
    WHEN (dataini is null) AND (datafim is null) then
    'select id,tvalidadei, tvalidadef
    from '||tabela||"
    WHEN (dataini is null) then
    'select id,tvalidadei, tvalidadef
    from '||tabela||' where
    (tvalidadei <='||""||TO_CHAR(datafim,'dd/mm/yyyy')||""||')'
     WHEN (datafim is null) then
    'select id,tvalidadei, tvalidadef
    from '||tabela||' where
    (tvalidadei >='||""||TO_CHAR(dataini,'dd/mm/yyyy')||""||')'
  END;
  LOOP
     FETCH c1 INTO vid,tvi,tvf;
     EXIT WHEN c1%NOTFOUND;
     resultado.id := vid;
     resultado.tvalidadei := tvi;
     resultado.tvalidadef := tvf;
     PIPE ROW(resultado);
  END LOOP;
  CLOSE c1;
  RETURN;
END;
```

ANEXO A4 Código Fonte da Função TEND

```
FUNCTION tend(tabela VARCHAR2, dataini date, datafim date)  RETURN
COLECAOTV PIPELINED is
TYPE Cursores is REF CURSOR;
c1 Cursores;
resultado TVALIDADE := TVALIDADE(NULL,NULL,NULL);
vid varchar2(15);
tvi date;
tvf date;
BEGIN
 OPEN c1 FOR
  CASE
     WHEN (dataini is not null) AND (datafim is not null) then
    'select id,tvalidadei, tvalidadef
    from '||tabela||' where
    (tvalidadef >='||""||TO_CHAR(dataini,'dd/mm/yyyy')||""|| 'and
    tvalidadef <='||""||TO_CHAR(datafim,'dd/mm/yyyy')||""||')'
    WHEN (dataini is null) AND (datafim is null) then
    'select id,tvalidadei, tvalidadef from '||tabela||"
    WHEN (dataini is null) then
    'select id,tvalidadei, tvalidadef from '||tabela||' where
    (tvalidadef <='||""||TO_CHAR(datafim,'dd/mm/yyyy')||""||')'
    WHEN (datafim is null) then
    'select id,tvalidadei, tvalidadef from '||tabela||' where
    (tvalidadef >='||""||TO_CHAR(dataini,'dd/mm/yyyy')||""||')'
  END;
  LOOP
     FETCH c1 INTO vid, tvi, tvf;
     EXIT WHEN c1%NOTFOUND;
     resultado.id := vid;
     resultado.tvalidadei := tvi;
     resultado.tvalidadef := tvf;
     PIPE ROW(resultado);
  END LOOP;
  CLOSE c1;
  RETURN;
END;
```

ANEXO A5 Código Fonte da Função TPERIOD

```
FUNCTION tperiod(tabela VARCHAR2, dataini date, datafim date)  RETURN
COLECAOTV PIPELINED is
TYPE Cursores is REF CURSOR;
c1 Cursores;
resultado TVALIDADE := TVALIDADE(NULL,NULL,NULL);
tvi date;
tvf date;
id varchar2(15);
BEGIN
  OPEN c1 FOR
  CASE
    WHEN (dataini is not null) AND (datafim is not null) then
          'SELECT tvalidadei,tvalidadef,id FROM '||tabela||' WHERE (
          ( tvalidadei>='||""||TO_char(dataini,'dd/mm/yyyy')||""||' ) AND
            ( tvalidadef <= '||""||TO_char(datafim,'dd/mm/yyyy')||""||' ) AND
            ( tvalidadef is not NULL ) ) OR
            ((tvalidadei >= '||""||TO_char(dataini,'dd/mm/yyyy')||""||' ) AND
            ( tvalidadei < '||""||TO_char(datafim,'dd/mm/yyyy')||""||' ) AND
            ( tvalidadef = NULL ))'
    WHEN (dataini is null) AND (datafim is null) then
    'select tvalidadei,tvalidadef,id from '||tabela||''
    WHEN (dataini is null) then
    'select tvalidadei,tvalidadef,id from '||tabela||' where
    (tvalidadef <='||""||TO_CHAR(datafim,'dd/mm/yyyy')||""||')'
    WHEN (datafim is null) then
    'select tvalidadei,tvalidadef,id from '||tabela||' where
    (tvalidadei>='||""||TO_CHAR(dataini,'dd/mm/yyyy')||""||')'
  END;
  LOOP
    FETCH c1 INTO tvi,tvf,id;
    EXIT WHEN c1%NOTFOUND;
    resultado.tvalidadei := tvi;
    resultado.tvalidadef := tvf;
    resultado.id := id;
    PIPE ROW(resultado);
  END LOOP;
  CLOSE c1;
  RETURN;
END;
```

ANEXO A6 Código Fonte da Função TINTERSECT

```
FUNCTION tintersect(tabela VARCHAR2, dataini1 date, datafim1 date, dataini2 date,
datafim2 date)  RETURN COLECAOTV Pipelined is
TYPE Cursores is REF CURSOR;
c1 Cursores;
c2 Cursores;
resultado TVALIDADE := TVALIDADE(NULL,NULL,NULL);
tvi date;
tvf date;
dataini date;
datafim date;
vdataini1 date;
vdatafim1 date;
vdataini2 date;
vdatafim2 date;

id varchar2(15);
BEGIN
  vdataini1 := dataini1;
  vdatafim1 := datafim1;
  vdataini2 := dataini2;
  vdatafim2 := datafim2;

-- Verifica se alguma data estah nula.
  if dataini1 is null then
    OPEN c2 FOR 'select distinct first_value(tvalidadei)
    over (order by tvalidadei asc rows unbounded preceding)
    from '||tabela||'';
    LOOP
      FETCH c2 INTO tvi;
      EXIT WHEN c2%NOTFOUND;
      if tvi is not null then
        vdataini1 := tvi;
      end if;
    END LOOP;

    CLOSE c2;
  end if;
  if dataini2 is NULL then
    OPEN c2 FOR 'select distinct first_value(tvalidadei)
    over (order by tvalidadei asc rows unbounded preceding)
   from '||tabela||'';
    LOOP
      FETCH c2 INTO tvi;
      EXIT WHEN c2%NOTFOUND;
      vdataini2 := tvi;
    END LOOP;
    CLOSE c2;
```

```
  end if;
--    raise_application_error(-20006, vdataini1 || vdataini2);
  if datafim1 is NULL then
    OPEN c2 FOR 'select distinct first_value(tvalidadef)
    over (order by tvalidadef desc rows unbounded preceding)
    from '||tabela||' where tvalidadef is not null ';

    LOOP
      FETCH c2 INTO tvf;
      EXIT WHEN c2%NOTFOUND;
      vdatafim1 := tvf;
    END LOOP;
    CLOSE c2;
  end if;
  if datafim2 is NULL then
    OPEN c2 FOR 'select distinct first_value(tvalidadef)
    over (order by tvalidadef desc rows unbounded preceding)
    from '||tabela||' where tvalidadef is not null ';
    LOOP
      FETCH c2 INTO tvf;
      EXIT WHEN c2%NOTFOUND;
      vdatafim2 := tvf;
    END LOOP;
--    raise_application_error(-20010, vdatafim1 || vdatafim2);
    CLOSE c2;
  end if;

-- Verifica os intervalos de inicio

  dataini :=
  case
   when (vdataini1 >= vdataini2 and vdataini1 <= vdatafim2) THEN
     vdataini1
   when (vdataini2 >= vdataini1 and vdataini2 <= vdatafim1) THEN
     vdataini2
   when (vdatafim1 < vdataini2 and vdatafim1 < vdatafim2) THEN
     null
   when (vdataini1 > vdataini2 and vdataini1 > vdatafim2) THEN
     null
  end;

  datafim :=
  case
   when (vdatafim1 >= vdataini2 and vdatafim1 <= vdatafim2) THEN
     vdatafim1
   when (vdatafim2 >= vdataini1 and vdatafim2 <= vdatafim1) THEN
     vdatafim2
   when (vdatafim1 < vdataini2 and vdatafim1 < vdatafim2) THEN
     null
   when (vdataini1 > vdataini2 and vdataini1 > vdatafim2) THEN
```

```
      null
  end;

--    raise_application_error(-20008, dataini ||' '|| datafim );
  OPEN c1 FOR
  'SELECT tvalidadei,tvalidadef,id FROM '||tabela||' WHERE (
  ( tvalidadei>='||""||TO_char(dataini,'dd/mm/yyyy')||""||' ) AND
  ( tvalidadef <= '||""||TO_char(datafim,'dd/mm/yyyy')||""||' ) AND
  ( tvalidadef is not NULL ) ) OR
  ((tvalidadei >= '||""||TO_char(dataini,'dd/mm/yyyy')||""||' ) AND
  ( tvalidadei < '||""||TO_char(datafim,'dd/mm/yyyy')||""||' ) AND
  ( tvalidadef = NULL )) order by tvalidadei ';

  LOOP
    FETCH c1 INTO tvi,tvf,id;
    EXIT WHEN c1%NOTFOUND;
    resultado.tvalidadei := tvi;
    resultado.tvalidadef := tvf;
    resultado.id := id;
    PIPE ROW(resultado);
  END LOOP;
  CLOSE c1;
 RETURN;
END;
```

ANEXO A7 Código Fonte da Função TCOUNT

```
FUNCTION tcount(tabela VARCHAR2, atrib1 NUMBER, atrib2 NUMBER, atrib3
NUMBER, dataini date, datafim date)  RETURN COLECAOTVRETORNO PIPELINED
is
TYPE Cursores is REF CURSOR;
c1 Cursores;
resultado TVALIDADERETORNO :=
TVALIDADERETORNO(NULL,NULL,NULL,NULL);
retorno number;
tvi date;
tvf date;
id varchar2(15);
vcondicao varchar2(500);
BEGIN
vcondicao :=
CASE
    WHEN (dataini is not null) AND (datafim is not null) THEN
    ' COUNT(*) FROM '||tabela||' WHERE (
            ( tvalidadei>='||''''||TO_char(dataini,'dd/mm/yyyy')||''''||' ) AND
    ( tvalidadef <= '||''''||TO_char(datafim,'dd/mm/yyyy')||''''||' ) AND
    ( tvalidadef is not NULL ) ) OR
    ((tvalidadei >= '||''''||TO_char(dataini,'dd/mm/yyyy')||''''||' ) AND
    ( tvalidadei < '||''''||TO_char(datafim,'dd/mm/yyyy')||''''||' ) AND
    ( tvalidadef = NULL )) GROUP BY '
   WHEN (dataini is null) AND (datafim is null) then
   ' COUNT(*) FROM '||tabela||' GROUP BY '
   WHEN (dataini is null) then
   ' COUNT(*) FROM '||tabela||' where
   (tvalidadef <='||''''||TO_CHAR(datafim,'dd/mm/yyyy')||''''||') GROUP BY '
   WHEN (datafim is null) then
   ' COUNT(*) FROM '||tabela||' where
   (tvalidadei>='||''''||TO_CHAR(dataini,'dd/mm/yyyy')||''''||') GROUP BY '
  END;

   IF (atrib1 = 1) AND (atrib2 = 1) AND (atrib3 = 1) THEN
      OPEN c1 FOR 'SELECT tvalidadei, tvalidadef, id, ' || vcondicao ||
      ' tvalidadei, tvalidadef, id ';
      LOOP
        FETCH c1 INTO tvi,tvf,id,retorno;
        EXIT WHEN c1%NOTFOUND;
        resultado.tvalidadei := tvi;
        resultado.tvalidadef := tvf;
        resultado.id := id;
        resultado.resultado := retorno;
        PIPE ROW(resultado);
      END LOOP;
   END IF;
   IF (atrib1 = 1) AND (atrib2 = 1) AND (atrib3 = 0) THEN
```

```
    OPEN c1 FOR 'SELECT tvalidadei, tvalidadef, ' || vcondicao ||
    ' tvalidadei, tvalidadef ';
    LOOP
      FETCH c1 INTO tvi,tvf,retorno;
      EXIT WHEN c1%NOTFOUND;
      resultado.tvalidadei := tvi;
      resultado.tvalidadef := tvf;
      resultado.resultado := retorno;
      PIPE ROW(resultado);
    END LOOP;
  END IF;
  IF (atrib1 = 1) AND (atrib2 = 0) AND (atrib3 = 0) THEN
    OPEN c1 FOR 'SELECT tvalidadei, ' || vcondicao ||
    ' tvalidadei ';
    LOOP
      FETCH c1 INTO tvi,retorno;
      EXIT WHEN c1%NOTFOUND;
      resultado.tvalidadei := tvi;
      resultado.resultado := retorno;
      PIPE ROW(resultado);
    END LOOP;
  END IF;
  IF (atrib1 = 0) AND (atrib2 = 1) AND (atrib3 = 1) THEN
    OPEN c1 FOR 'SELECT tvalidadef, id, ' || vcondicao ||
    ' tvalidadef, id ';
    LOOP
      FETCH c1 INTO tvf,id,retorno;
      EXIT WHEN c1%NOTFOUND;
      resultado.tvalidadef := tvf;
      resultado.id := id;
      resultado.resultado := retorno;
      PIPE ROW(resultado);
    END LOOP;
  END IF;
  IF (atrib1 = 0) AND (atrib2 = 1) AND (atrib3 = 0) THEN
    OPEN c1 FOR 'SELECT tvalidadef, ' || vcondicao ||
    ' tvalidadef ';
    LOOP
      FETCH c1 INTO tvf,retorno;
      EXIT WHEN c1%NOTFOUND;
      resultado.tvalidadef := tvf;
      resultado.resultado := retorno;
      PIPE ROW(resultado);
    END LOOP;
  END IF;
  IF (atrib1 = 0) AND (atrib2 = 0) AND (atrib3 = 1) THEN
    OPEN c1 FOR 'SELECT id, ' || vcondicao ||
    ' id ';
    LOOP
      FETCH c1 INTO id,retorno;
```

```
          EXIT WHEN c1%NOTFOUND;
          resultado.id := id;
          resultado.resultado := retorno;
          PIPE ROW(resultado);
        END LOOP;

    END IF;
    IF (atrib1 = 1) AND (atrib2 = 0) AND (atrib3 = 1) THEN
        OPEN c1 FOR 'SELECT tvalidadei, id, ' || vcondicao ||
        ' tvalidadei, id ';
        LOOP
          FETCH c1 INTO tvi,id,retorno;
          EXIT WHEN c1%NOTFOUND;
          resultado.tvalidadei := tvi;
          resultado.id := id;
          resultado.resultado := retorno;
          PIPE ROW(resultado);
        END LOOP;
--  ELSE
--       raise_application_error(-20008, 'PARÂMETROS INVÁLIDOS' );
    END IF;
  CLOSE c1;
  RETURN;
END;
```

ANEXO A8 Código Fonte da Função TSUM

```
FUNCTION tsum(tabela VARCHAR2, atrib1 NUMBER, atrib2 NUMBER, atrib3
NUMBER, soma varchar2, dataini date, datafim date)  RETURN
COLECAOTVRETORNO PIPELINED is
TYPE Cursores is REF CURSOR;
c1 Cursores;
resultado TVALIDADERETORNO :=
TVALIDADERETORNO(NULL,NULL,NULL,NULL);
retorno number;
tvi date;
tvf date;
id varchar2(15);
vcondicao varchar2(500);
BEGIN
vcondicao :=
CASE
    WHEN (dataini is not null) AND (datafim is not null) THEN
    ' SUM('||soma||') FROM '||tabela||' WHERE (
            ( tvalidadei>='||""||TO_char(dataini,'dd/mm/yyyy')||""||' ) AND
    ( tvalidadef <= '||""||TO_char(datafim,'dd/mm/yyyy')||""||' ) AND
    ( tvalidadef is not NULL ) ) OR
    ((tvalidadei >= '||""||TO_char(dataini,'dd/mm/yyyy')||""||' ) AND
    ( tvalidadei < '||""||TO_char(datafim,'dd/mm/yyyy')||""||' ) AND
    ( tvalidadef = NULL )) GROUP BY '
   WHEN (dataini is null) AND (datafim is null) then
   ' SUM('||soma||') FROM '||tabela||' GROUP BY '
   WHEN (dataini is null) then
   ' SUM('||soma||') FROM '||tabela||' where
   (tvalidadef <='||""||TO_CHAR(datafim,'dd/mm/yyyy')||""||') GROUP BY '
   WHEN (datafim is null) then
   ' SUM('||soma||') FROM '||tabela||' where
   (tvalidadei>='||""||TO_CHAR(dataini,'dd/mm/yyyy')||""||') GROUP BY '
  END;

   IF (atrib1 = 1) AND (atrib2 = 1) AND (atrib3 = 1) THEN
      OPEN c1 FOR 'SELECT tvalidadei, tvalidadef, id, ' || vcondicao ||
      ' tvalidadei, tvalidadef, id ';
      LOOP
        FETCH c1 INTO tvi,tvf,id,retorno;
        EXIT WHEN c1%NOTFOUND;
        resultado.tvalidadei := tvi;
        resultado.tvalidadef := tvf;
        resultado.id := id;
        resultado.resultado := retorno;
        PIPE ROW(resultado);
      END LOOP;
   END IF;
   IF (atrib1 = 1) AND (atrib2 = 1) AND (atrib3 = 0) THEN
      OPEN c1 FOR 'SELECT tvalidadei, tvalidadef, ' || vcondicao ||
```

```
    ' tvalidadei, tvalidadef ';
    LOOP
      FETCH c1 INTO tvi,tvf,retorno;
      EXIT WHEN c1%NOTFOUND;
      resultado.tvalidadei := tvi;
      resultado.tvalidadef := tvf;
      resultado.resultado := retorno;
      PIPE ROW(resultado);
    END LOOP;
  END IF;
  IF (atrib1 = 1) AND (atrib2 = 0) AND (atrib3 = 0) THEN
    OPEN c1 FOR 'SELECT tvalidadei, ' || vcondicao ||
    ' tvalidadei ';
    LOOP
      FETCH c1 INTO tvi,retorno;
      EXIT WHEN c1%NOTFOUND;
      resultado.tvalidadei := tvi;
      resultado.resultado := retorno;
      PIPE ROW(resultado);
    END LOOP;
  END IF;
  IF (atrib1 = 0) AND (atrib2 = 1) AND (atrib3 = 1) THEN
    OPEN c1 FOR 'SELECT tvalidadef, id, ' || vcondicao ||
    ' tvalidadef, id ';
    LOOP
      FETCH c1 INTO tvf,id,retorno;
      EXIT WHEN c1%NOTFOUND;
      resultado.tvalidadef := tvf;
      resultado.id := id;
      resultado.resultado := retorno;
      PIPE ROW(resultado);
    END LOOP;
  END IF;
  IF (atrib1 = 0) AND (atrib2 = 1) AND (atrib3 = 0) THEN
    OPEN c1 FOR 'SELECT tvalidadef, ' || vcondicao ||
    ' tvalidadef ';
    LOOP
      FETCH c1 INTO tvf,retorno;
      EXIT WHEN c1%NOTFOUND;
      resultado.tvalidadef := tvf;
      resultado.resultado := retorno;
      PIPE ROW(resultado);
    END LOOP;
  END IF;
  IF (atrib1 = 0) AND (atrib2 = 0) AND (atrib3 = 1) THEN
    OPEN c1 FOR 'SELECT id, ' || vcondicao ||
    ' id ';
    LOOP
      FETCH c1 INTO id,retorno;
      EXIT WHEN c1%NOTFOUND;
```

```
          resultado.id := id;
          resultado.resultado := retorno;
          PIPE ROW(resultado);
        END LOOP;

    END IF;
    IF (atrib1 = 1) AND (atrib2 = 0) AND (atrib3 = 1) THEN
        OPEN c1 FOR 'SELECT tvalidadei, id, ' || vcondicao ||
        ' tvalidadei, id ';
        LOOP
          FETCH c1 INTO tvi,id,retorno;
          EXIT WHEN c1%NOTFOUND;
          resultado.tvalidadei := tvi;
          resultado.id := id;
          resultado.resultado := retorno;
          PIPE ROW(resultado);
        END LOOP;
--      ELSE
--        raise_application_error(-20008, 'PARÂMETROS INVÁLIDOS' );
    END IF;
  CLOSE c1;
  RETURN;
END;
```

ANEXO A9 Código Fonte da Função TAVG

```
FUNCTION tavg(tabela VARCHAR2, atrib1 NUMBER, atrib2 NUMBER, atrib3
NUMBER, media varchar2, dataini date, datafim date)  RETURN
COLECAOTVRETORNO PIPELINED is
TYPE Cursores is REF CURSOR;
c1 Cursores;
resultado TVALIDADERETORNO :=
TVALIDADERETORNO(NULL,NULL,NULL,NULL);
retorno number;
tvi date;
tvf date;
id varchar2(15);
vcondicao varchar2(500);
BEGIN
vcondicao :=
CASE
    WHEN (dataini is not null) AND (datafim is not null) THEN
    ' AVG('||media||') FROM '||tabela||' WHERE (
            ( tvalidadei>='||""||TO_char(dataini,'dd/mm/yyyy')||""||' ) AND
    ( tvalidadef <= '||""||TO_char(datafim,'dd/mm/yyyy')||""||' ) AND
    ( tvalidadef is not NULL ) ) OR
    ((tvalidadei >= '||""||TO_char(dataini,'dd/mm/yyyy')||""||' ) AND
    ( tvalidadei < '||""||TO_char(datafim,'dd/mm/yyyy')||""||' ) AND
    ( tvalidadef = NULL )) GROUP BY '
   WHEN (dataini is null) AND (datafim is null) then
   ' AVG('||media||') FROM '||tabela||' GROUP BY '
   WHEN (dataini is null) then
   ' AVG('||media||') FROM '||tabela||' where
   (tvalidadef <='||""||TO_CHAR(datafim,'dd/mm/yyyy')||""||') GROUP BY '
   WHEN (datafim is null) then
   ' AVG('||media||') FROM '||tabela||' where
   (tvalidadei>='||""||TO_CHAR(dataini,'dd/mm/yyyy')||""||') GROUP BY '
  END;

   IF (atrib1 = 1) AND (atrib2 = 1) AND (atrib3 = 1) THEN
      OPEN c1 FOR 'SELECT tvalidadei, tvalidadef, id, ' || vcondicao ||
      ' tvalidadei, tvalidadef, id ';
      LOOP
        FETCH c1 INTO tvi,tvf,id,retorno;
        EXIT WHEN c1%NOTFOUND;
        resultado.tvalidadei := tvi;
        resultado.tvalidadef := tvf;
        resultado.id := id;
        resultado.resultado := retorno;
        PIPE ROW(resultado);
      END LOOP;
   END IF;
   IF (atrib1 = 1) AND (atrib2 = 1) AND (atrib3 = 0) THEN
      OPEN c1 FOR 'SELECT tvalidadei, tvalidadef, ' || vcondicao ||
```

```
    ' tvalidadei, tvalidadef ';
    LOOP
      FETCH c1 INTO tvi,tvf,retorno;
      EXIT WHEN c1%NOTFOUND;
      resultado.tvalidadei := tvi;
      resultado.tvalidadef := tvf;
      resultado.resultado := retorno;
      PIPE ROW(resultado);
    END LOOP;
  END IF;
  IF (atrib1 = 1) AND (atrib2 = 0) AND (atrib3 = 0) THEN
    OPEN c1 FOR 'SELECT tvalidadei, ' || vcondicao ||
    ' tvalidadei ';
    LOOP
      FETCH c1 INTO tvi,retorno;
      EXIT WHEN c1%NOTFOUND;
      resultado.tvalidadei := tvi;
      resultado.resultado := retorno;
      PIPE ROW(resultado);
    END LOOP;
  END IF;
  IF (atrib1 = 0) AND (atrib2 = 1) AND (atrib3 = 1) THEN
    OPEN c1 FOR 'SELECT tvalidadef, id, ' || vcondicao ||
    ' tvalidadef, id ';
    LOOP
      FETCH c1 INTO tvf,id,retorno;
      EXIT WHEN c1%NOTFOUND;
      resultado.tvalidadef := tvf;
      resultado.id := id;
      resultado.resultado := retorno;
      PIPE ROW(resultado);
    END LOOP;
  END IF;
  IF (atrib1 = 0) AND (atrib2 = 1) AND (atrib3 = 0) THEN
    OPEN c1 FOR 'SELECT tvalidadef, ' || vcondicao ||
    ' tvalidadef ';
    LOOP
      FETCH c1 INTO tvf,retorno;
      EXIT WHEN c1%NOTFOUND;
      resultado.tvalidadef := tvf;
      resultado.resultado := retorno;
      PIPE ROW(resultado);
    END LOOP;
  END IF;
  IF (atrib1 = 0) AND (atrib2 = 0) AND (atrib3 = 1) THEN
    OPEN c1 FOR 'SELECT id, ' || vcondicao ||
    ' id ';
    LOOP
      FETCH c1 INTO id,retorno;
      EXIT WHEN c1%NOTFOUND;
```

```
          resultado.id := id;
          resultado.resultado := retorno;
          PIPE ROW(resultado);
        END LOOP;

    END IF;
    IF (atrib1 = 1) AND (atrib2 = 0) AND (atrib3 = 1) THEN
        OPEN c1 FOR 'SELECT tvalidadei, id, ' || vcondicao ||
        ' tvalidadei, id ';
        LOOP
          FETCH c1 INTO tvi,id,retorno;
          EXIT WHEN c1%NOTFOUND;
          resultado.tvalidadei := tvi;
          resultado.id := id;
          resultado.resultado := retorno;
          PIPE ROW(resultado);
        END LOOP;
--      ELSE
--        raise_application_error(-20008, 'PARÂMETROS INVÁLIDOS' );
    END IF;
  CLOSE c1;
  RETURN;
END;
```

ANEXO A10 Código Fonte da Função TMAX

```
FUNCTION tmax(tabela VARCHAR2, atrib1 NUMBER, atrib2 NUMBER, atrib3
NUMBER, maior varchar2, dataini date, datafim date)  RETURN
COLECAOTVRETORNO PIPELINED is
TYPE Cursores is REF CURSOR;
c1 Cursores;
resultado TVALIDADERETORNO :=
TVALIDADERETORNO(NULL,NULL,NULL,NULL);
retorno number;
tvi date;
tvf date;
id varchar2(15);
vcondicao varchar2(500);
BEGIN
vcondicao :=
CASE
    WHEN (dataini is not null) AND (datafim is not null) THEN
    ' MAX('||maior||') FROM '||tabela||' WHERE (
            ( tvalidadei>='||""||TO_char(dataini,'dd/mm/yyyy')||""||' ) AND
    ( tvalidadef <= '||""||TO_char(datafim,'dd/mm/yyyy')||""||' ) AND
    ( tvalidadef is not NULL ) ) OR
    ((tvalidadei >= '||""||TO_char(dataini,'dd/mm/yyyy')||""||' ) AND
    ( tvalidadei < '||""||TO_char(datafim,'dd/mm/yyyy')||""||' ) AND
    ( tvalidadef = NULL )) GROUP BY '
   WHEN (dataini is null) AND (datafim is null) then
   ' MAX('||maior||') FROM '||tabela||' GROUP BY '
   WHEN (dataini is null) then
   ' MAX('||maior||') FROM '||tabela||' where
   (tvalidadef <='||""||TO_CHAR(datafim,'dd/mm/yyyy')||""||') GROUP BY '
   WHEN (datafim is null) then
   ' MAX('||maior||') FROM '||tabela||' where
   (tvalidadei>='||""||TO_CHAR(dataini,'dd/mm/yyyy')||""||') GROUP BY '
  END;

   IF (atrib1 = 1) AND (atrib2 = 1) AND (atrib3 = 1) THEN
      OPEN c1 FOR 'SELECT tvalidadei, tvalidadef, id, ' || vcondicao ||
      ' tvalidadei, tvalidadef, id ';
      LOOP
         FETCH c1 INTO tvi,tvf,id,retorno;
         EXIT WHEN c1%NOTFOUND;
         resultado.tvalidadei := tvi;
         resultado.tvalidadef := tvf;
         resultado.id := id;
         resultado.resultado := retorno;
         PIPE ROW(resultado);
      END LOOP;
   END IF;
   IF (atrib1 = 1) AND (atrib2 = 1) AND (atrib3 = 0) THEN
      OPEN c1 FOR 'SELECT tvalidadei, tvalidadef, ' || vcondicao ||
```

```
    ' tvalidadei, tvalidadef ';
    LOOP
      FETCH c1 INTO tvi,tvf,retorno;
      EXIT WHEN c1%NOTFOUND;
      resultado.tvalidadei := tvi;
      resultado.tvalidadef := tvf;
      resultado.resultado := retorno;
      PIPE ROW(resultado);
    END LOOP;
  END IF;
  IF (atrib1 = 1) AND (atrib2 = 0) AND (atrib3 = 0) THEN
    OPEN c1 FOR 'SELECT tvalidadei, ' || vcondicao ||
    ' tvalidadei ';
    LOOP
      FETCH c1 INTO tvi,retorno;
      EXIT WHEN c1%NOTFOUND;
      resultado.tvalidadei := tvi;
      resultado.resultado := retorno;
      PIPE ROW(resultado);
    END LOOP;
  END IF;
  IF (atrib1 = 0) AND (atrib2 = 1) AND (atrib3 = 1) THEN
    OPEN c1 FOR 'SELECT tvalidadef, id, ' || vcondicao ||
    ' tvalidadef, id ';
    LOOP
      FETCH c1 INTO tvf,id,retorno;
      EXIT WHEN c1%NOTFOUND;
      resultado.tvalidadef := tvf;
      resultado.id := id;
      resultado.resultado := retorno;
      PIPE ROW(resultado);
    END LOOP;
  END IF;
  IF (atrib1 = 0) AND (atrib2 = 1) AND (atrib3 = 0) THEN
    OPEN c1 FOR 'SELECT tvalidadef, ' || vcondicao ||
    ' tvalidadef ';
    LOOP
      FETCH c1 INTO tvf,retorno;
      EXIT WHEN c1%NOTFOUND;
      resultado.tvalidadef := tvf;
      resultado.resultado := retorno;
      PIPE ROW(resultado);
    END LOOP;
  END IF;
  IF (atrib1 = 0) AND (atrib2 = 0) AND (atrib3 = 1) THEN
    OPEN c1 FOR 'SELECT id, ' || vcondicao ||
    ' id ';
    LOOP
      FETCH c1 INTO id,retorno;
      EXIT WHEN c1%NOTFOUND;
```

```
          resultado.id := id;
          resultado.resultado := retorno;
          PIPE ROW(resultado);
        END LOOP;

    END IF;
    IF (atrib1 = 1) AND (atrib2 = 0) AND (atrib3 = 1) THEN
        OPEN c1 FOR 'SELECT tvalidadei, id, ' || vcondicao ||
        ' tvalidadei, id ';
        LOOP
          FETCH c1 INTO tvi,id,retorno;
          EXIT WHEN c1%NOTFOUND;
          resultado.tvalidadei := tvi;
          resultado.id := id;
          resultado.resultado := retorno;
          PIPE ROW(resultado);
        END LOOP;
--      ELSE
--        raise_application_error(-20008, 'PARÂMETROS INVÁLIDOS' );
    END IF;
  CLOSE c1;
  RETURN;
END;
```

ANEXO A11 Código Fonte da Função TMIN

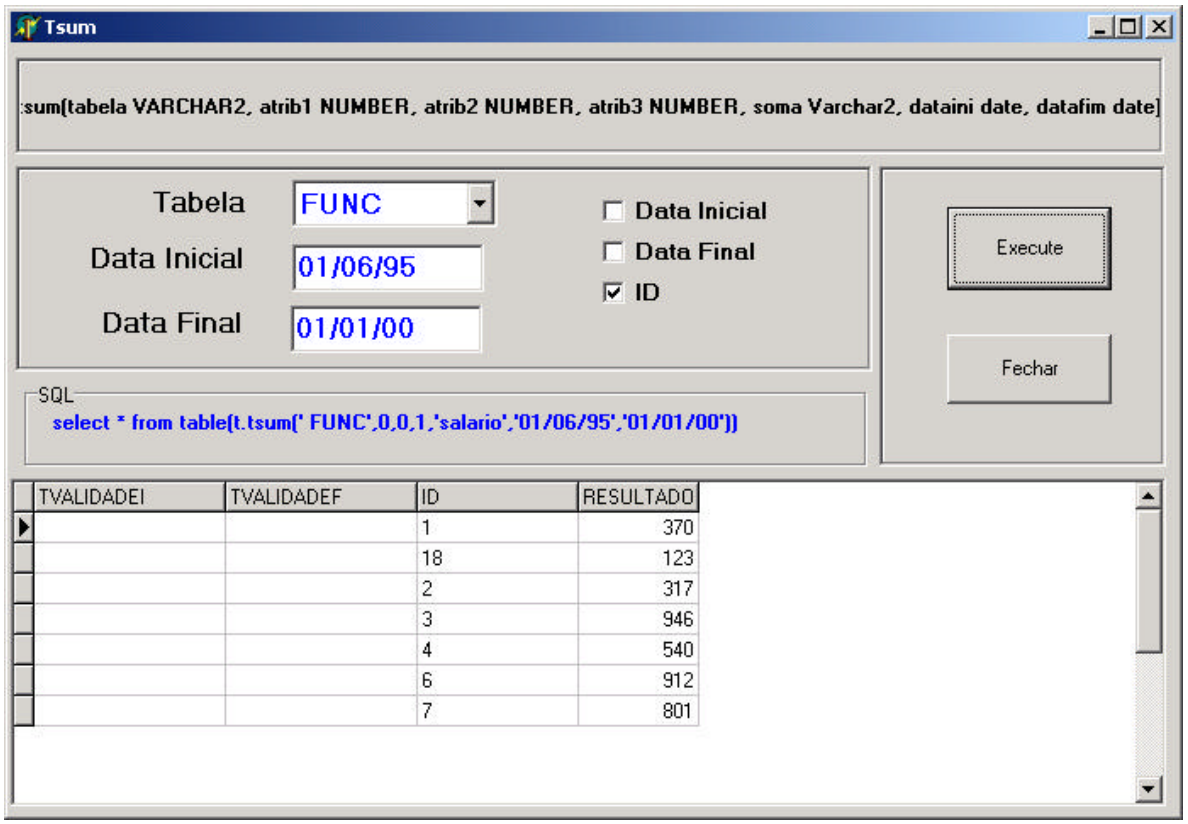
```
FUNCTION tmin(tabela VARCHAR2, atrib1 NUMBER, atrib2 NUMBER, atrib3
NUMBER, menor varchar2, dataini date, datafim date)  RETURN
COLECAOTVRETORNO PIPELINED is
TYPE Cursores is REF CURSOR;
c1 Cursores;
resultado TVALIDADERETORNO :=
TVALIDADERETORNO(NULL,NULL,NULL,NULL);
retorno number;
tvi date;
tvf date;
id varchar2(15);
vcondicao varchar2(500);
BEGIN
vcondicao :=
CASE
   WHEN (dataini is not null) AND (datafim is not null) THEN
    ' MIN('||menor||') FROM '||tabela||' WHERE (
           ( tvalidadei>='||""||TO_char(dataini,'dd/mm/yyyy')||""||' ) AND
    ( tvalidadef <= '||""||TO_char(datafim,'dd/mm/yyyy')||""||' ) AND
    ( tvalidadef is not NULL ) ) OR
    ((tvalidadei >= '||""||TO_char(dataini,'dd/mm/yyyy')||""||' ) AND
    ( tvalidadei < '||""||TO_char(datafim,'dd/mm/yyyy')||""||' ) AND
    ( tvalidadef = NULL )) GROUP BY '
   WHEN (dataini is null) AND (datafim is null) then
   ' MIN('||menor||') FROM '||tabela||' GROUP BY '
   WHEN (dataini is null) then
   ' MIN('||menor||') FROM '||tabela||' where
   (tvalidadef <='||""||TO_CHAR(datafim,'dd/mm/yyyy')||""||') GROUP BY '
   WHEN (datafim is null) then
   ' MIN('||menor||') FROM '||tabela||' where
   (tvalidadei>='||""||TO_CHAR(dataini,'dd/mm/yyyy')||""||') GROUP BY '
 END;

   IF (atrib1 = 1) AND (atrib2 = 1) AND (atrib3 = 1) THEN
      OPEN c1 FOR 'SELECT tvalidadei, tvalidadef, id, ' || vcondicao ||
      ' tvalidadei, tvalidadef, id ';
      LOOP
        FETCH c1 INTO tvi,tvf,id,retorno;
        EXIT WHEN c1%NOTFOUND;
        resultado.tvalidadei := tvi;
        resultado.tvalidadef := tvf;
        resultado.id := id;
        resultado.resultado := retorno;
        PIPE ROW(resultado);
      END LOOP;
   END IF;
   IF (atrib1 = 1) AND (atrib2 = 1) AND (atrib3 = 0) THEN
      OPEN c1 FOR 'SELECT tvalidadei, tvalidadef, ' || vcondicao ||
```

```
    ' tvalidadei, tvalidadef ';
    LOOP
      FETCH c1 INTO tvi,tvf,retorno;
      EXIT WHEN c1%NOTFOUND;
      resultado.tvalidadei := tvi;
      resultado.tvalidadef := tvf;
      resultado.resultado := retorno;
      PIPE ROW(resultado);
    END LOOP;
  END IF;
  IF (atrib1 = 1) AND (atrib2 = 0) AND (atrib3 = 0) THEN
    OPEN c1 FOR 'SELECT tvalidadei, ' || vcondicao ||
    ' tvalidadei ';
    LOOP
      FETCH c1 INTO tvi,retorno;
      EXIT WHEN c1%NOTFOUND;
      resultado.tvalidadei := tvi;
      resultado.resultado := retorno;
      PIPE ROW(resultado);
    END LOOP;
  END IF;
  IF (atrib1 = 0) AND (atrib2 = 1) AND (atrib3 = 1) THEN
    OPEN c1 FOR 'SELECT tvalidadef, id, ' || vcondicao ||
    ' tvalidadef, id ';
    LOOP
      FETCH c1 INTO tvf,id,retorno;
      EXIT WHEN c1%NOTFOUND;
      resultado.tvalidadef := tvf;
      resultado.id := id;
      resultado.resultado := retorno;
      PIPE ROW(resultado);
    END LOOP;
  END IF;
  IF (atrib1 = 0) AND (atrib2 = 1) AND (atrib3 = 0) THEN
    OPEN c1 FOR 'SELECT tvalidadef, ' || vcondicao ||
    ' tvalidadef ';
    LOOP
      FETCH c1 INTO tvf,retorno;
      EXIT WHEN c1%NOTFOUND;
      resultado.tvalidadef := tvf;
      resultado.resultado := retorno;
      PIPE ROW(resultado);
    END LOOP;
  END IF;
  IF (atrib1 = 0) AND (atrib2 = 0) AND (atrib3 = 1) THEN
    OPEN c1 FOR 'SELECT id, ' || vcondicao ||
    ' id ';
    LOOP
      FETCH c1 INTO id,retorno;
      EXIT WHEN c1%NOTFOUND;
```

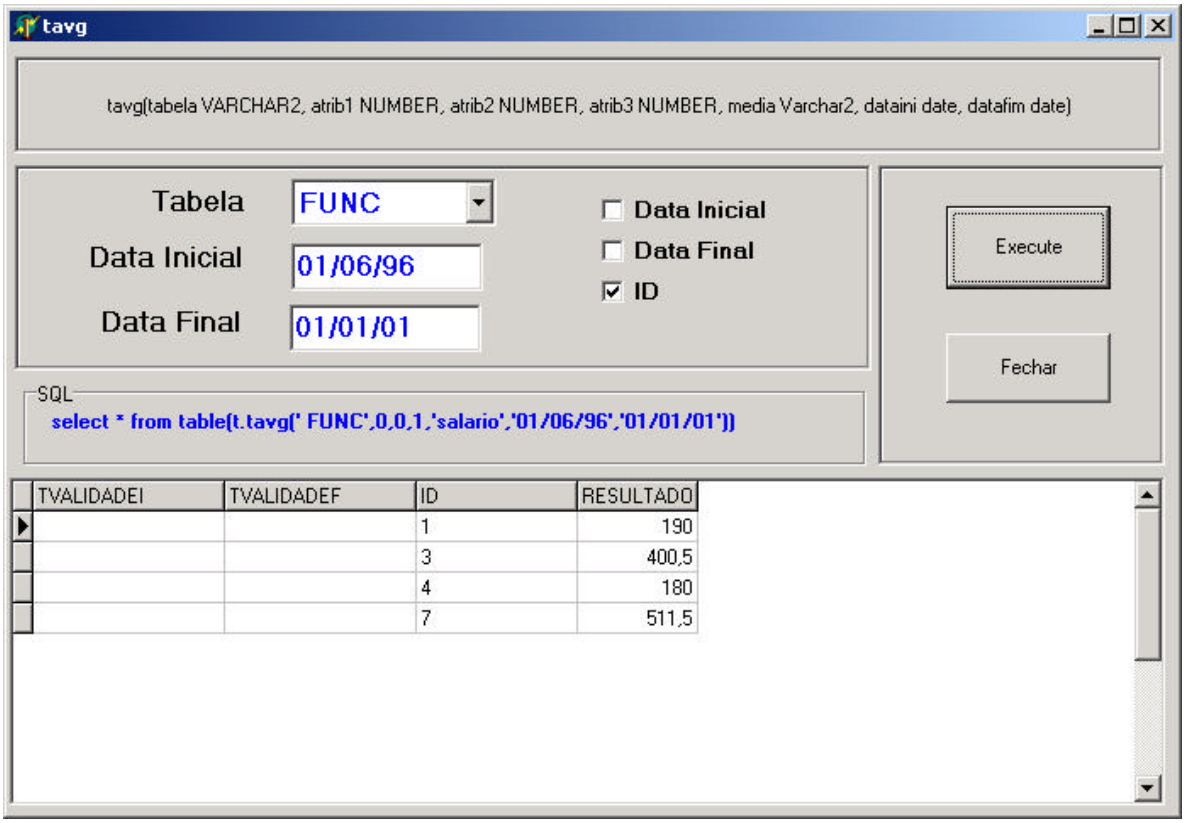
```
          resultado.id := id;
          resultado.resultado := retorno;
          PIPE ROW(resultado);
        END LOOP;

    END IF;
    IF (atrib1 = 1) AND (atrib2 = 0) AND (atrib3 = 1) THEN
        OPEN c1 FOR 'SELECT tvalidadei, id, ' || vcondicao ||
        ' tvalidadei, id ';
        LOOP
          FETCH c1 INTO tvi,id,retorno;
          EXIT WHEN c1%NOTFOUND;
          resultado.tvalidadei := tvi;
          resultado.id := id;
          resultado.resultado := retorno;
          PIPE ROW(resultado);
        END LOOP;
--      ELSE
--        raise_application_error(-20008, 'PARÂMETROS INVÁLIDOS' );
    END IF;
  CLOSE c1;
  RETURN;
END;
```

ANEXO B INTERFACES VISUAIS

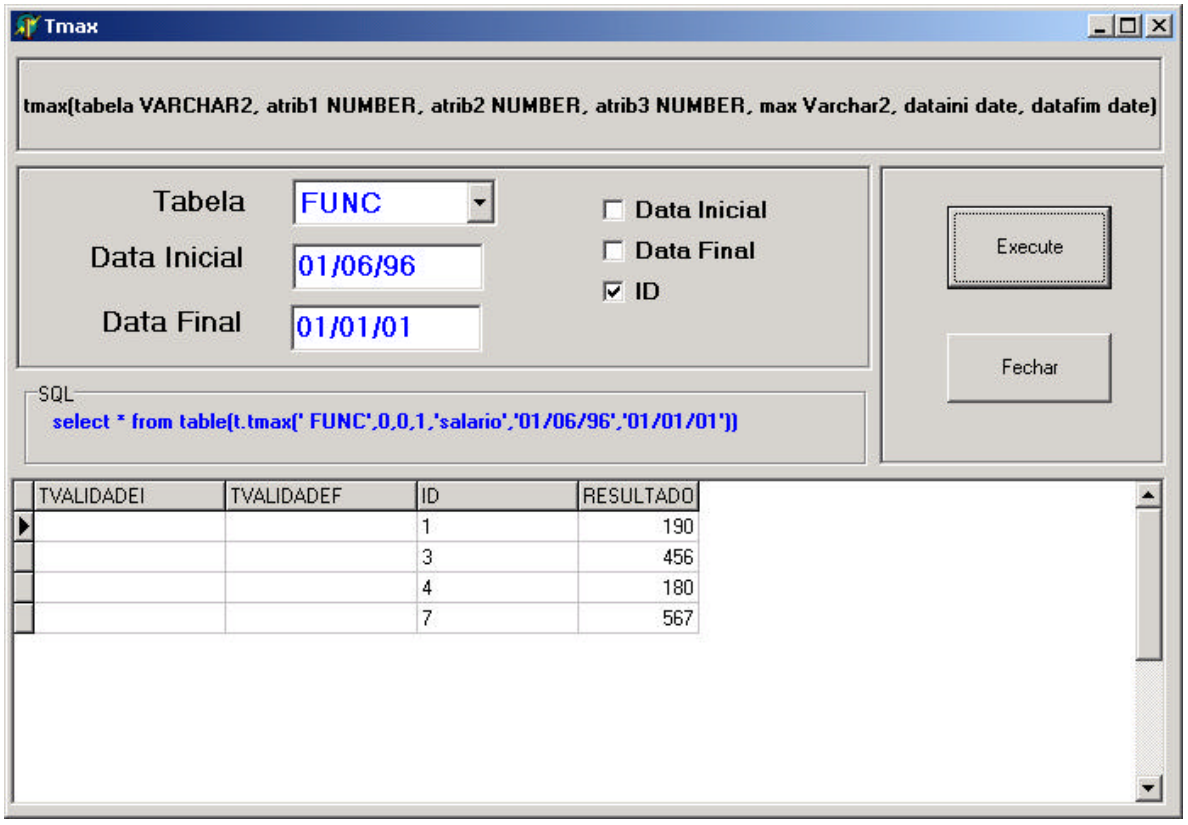
ANEXO B1 Interface Visual da Função TSUM



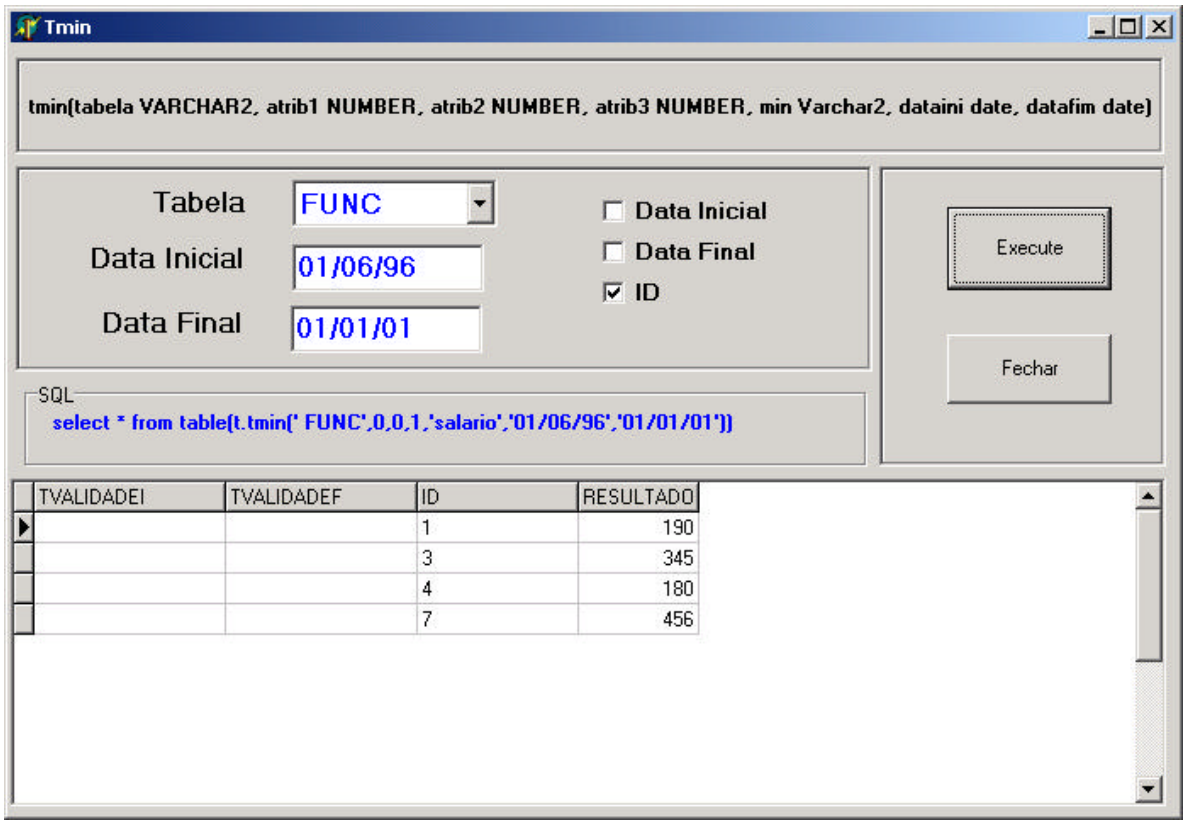
ANEXO B2 Interface Visual da Função TAVG



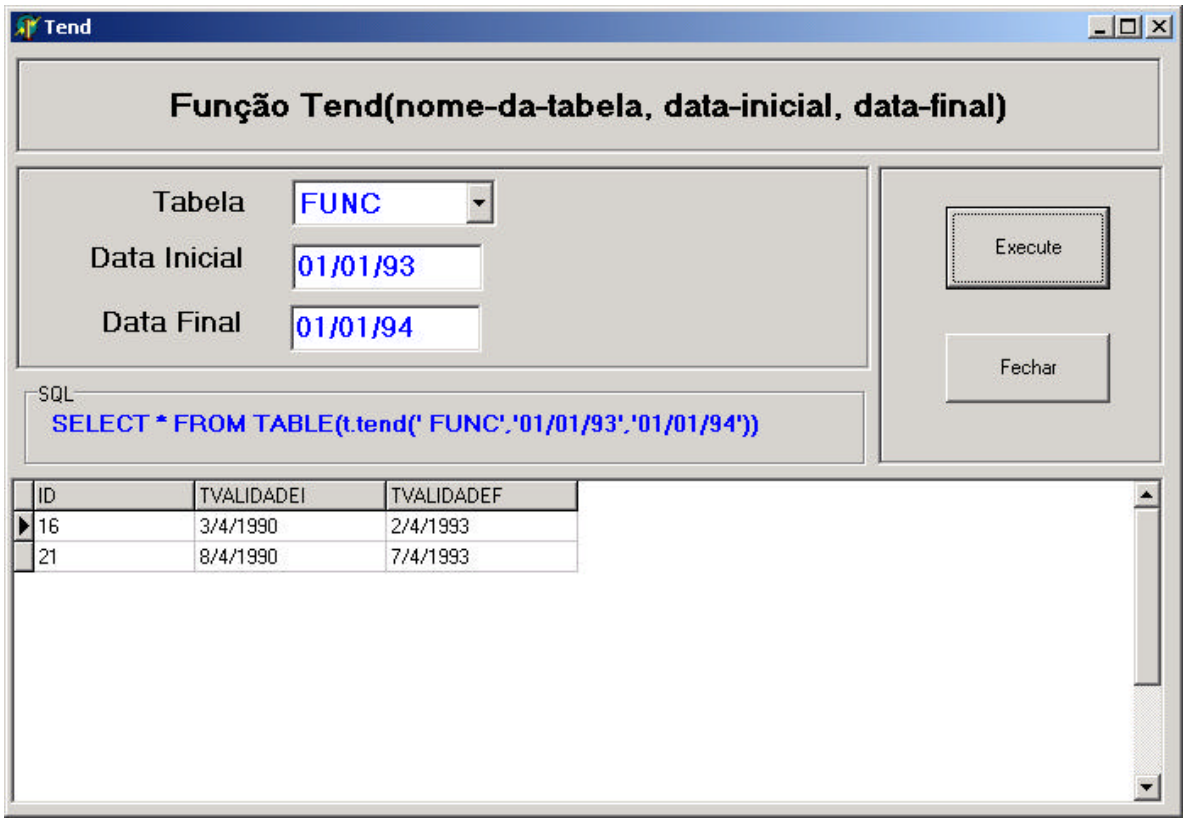
ANEXO B3 Interface Visual da Função TMAX



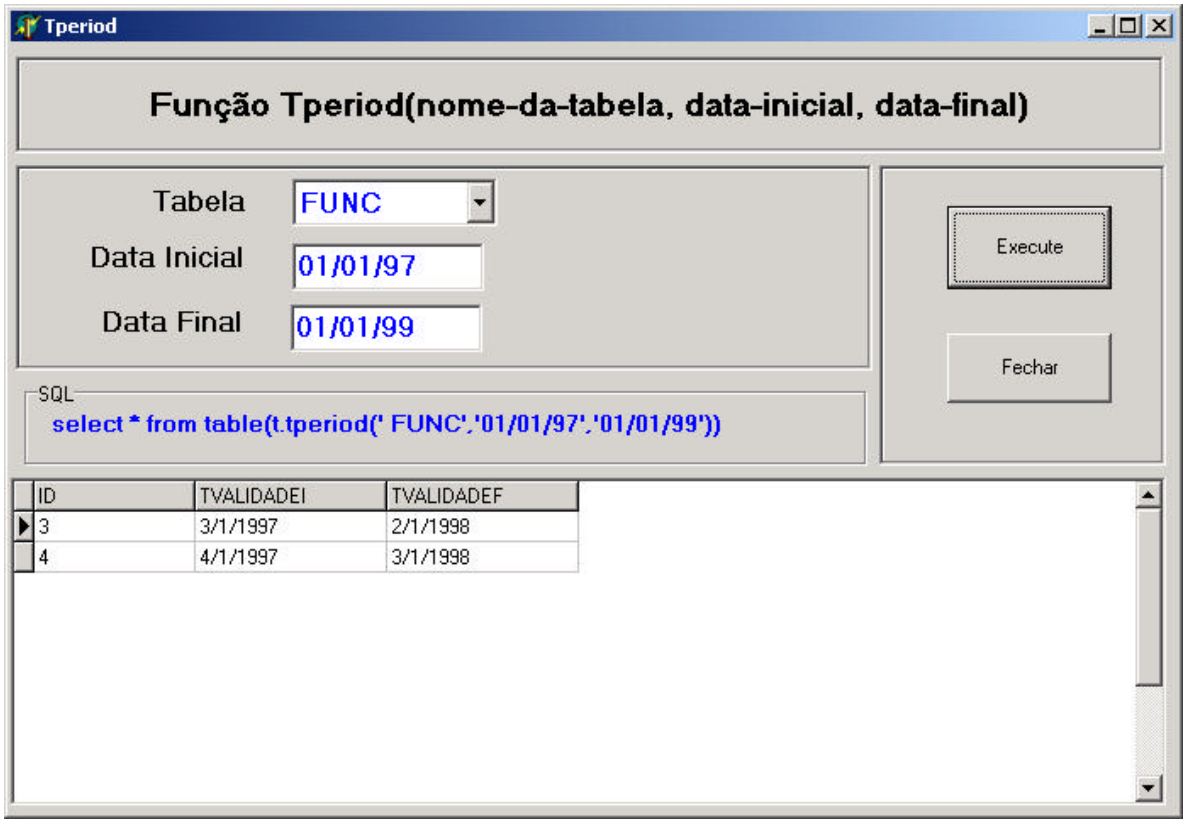
ANEXO B4 Interface Visual da Função TMIN



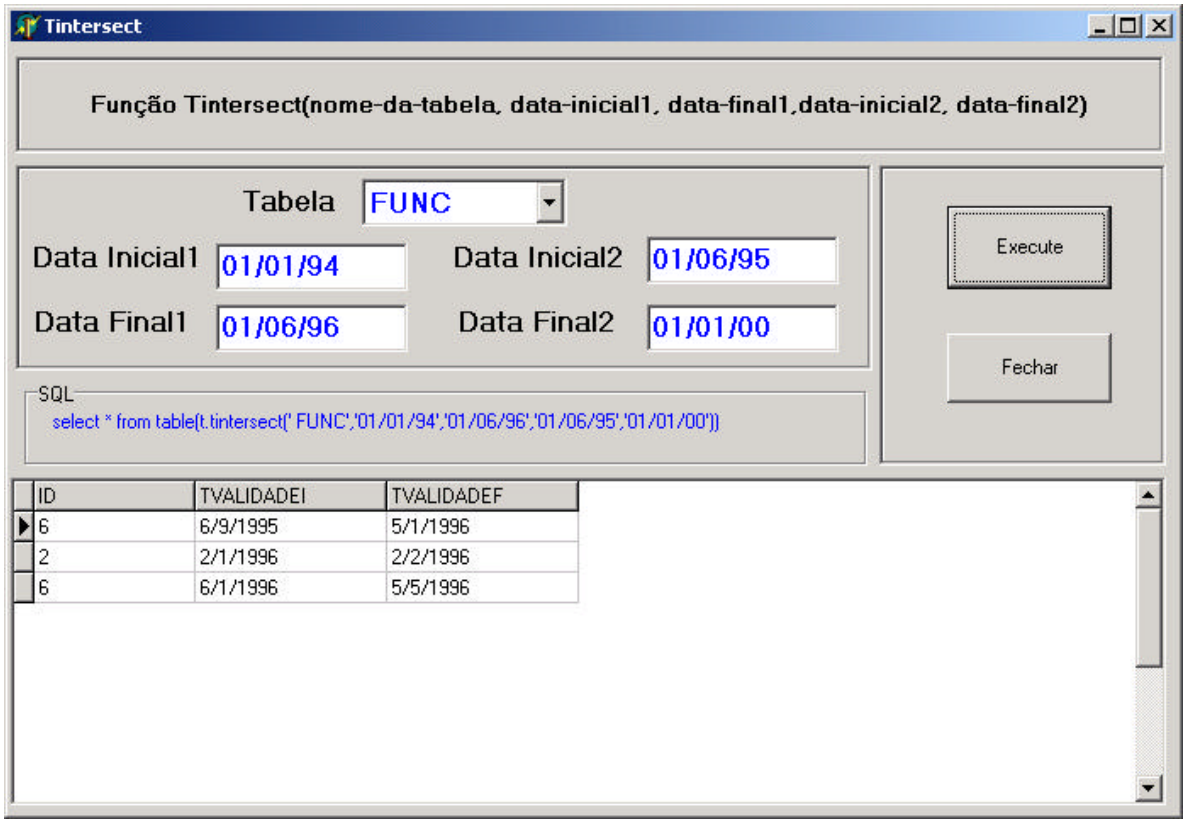
ANEXO B5 Interface Visual da Função TEND



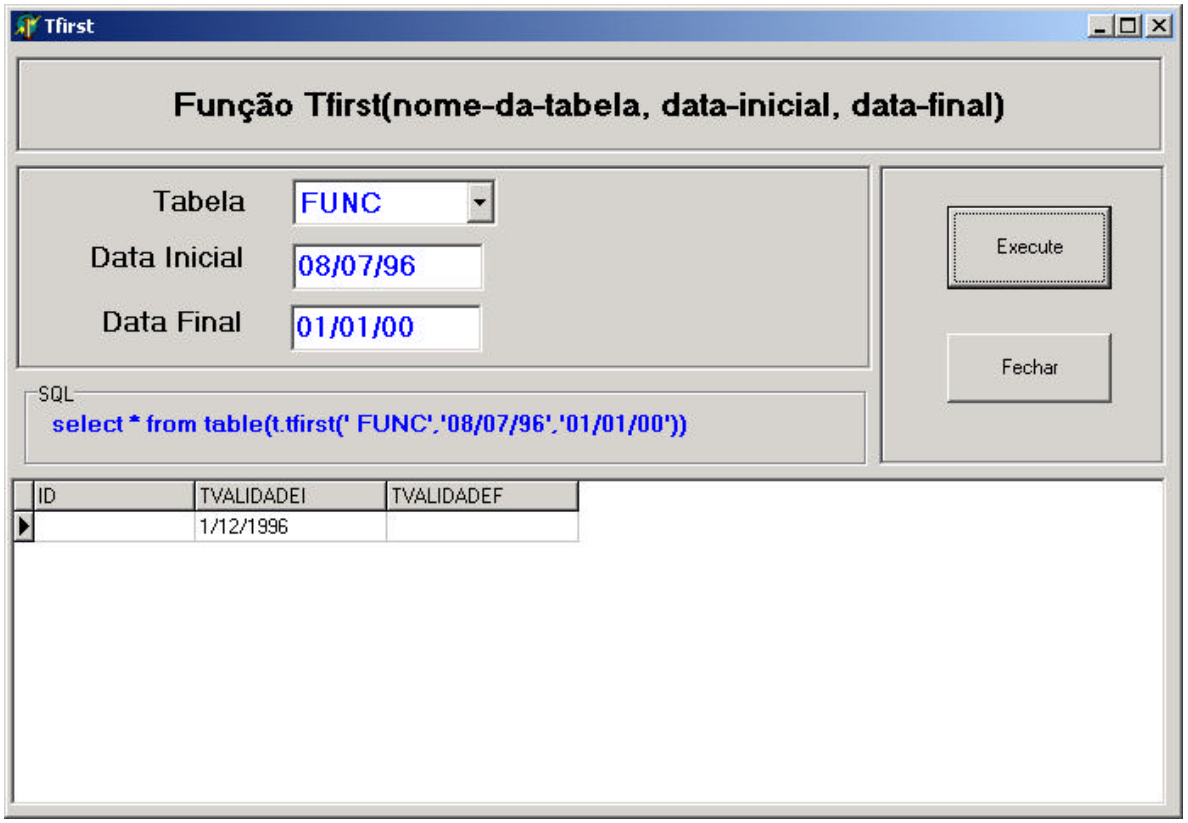
ANEXO B6 Interface Visual da Função TPERIOD



ANEXO B7 Interface Visual da Função TINTERSECT



ANEXO B8 Interface Visual da Função TFIRST



ANEXO B9 Interface Visual da Função TLAST