INTERFACE DE CONSULTAS TF-ORM PARA BANCO DE DADOS RELACIONAIS

Mirella Moura Moro
Nina Edelweiss

PROGRAMA DE PÓS GRADUAÇÃO EM COMPUTAÇÃO – INSTITUTO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Av. Bento Gonçalves, 9500 – Bloco IV – Agronomia
Caixa Postal 15064 – CEP 91501 – Porto Alegre – RS
[mirella, nina]@inf.ufrgs.br

Resumo

TF-ORM (*Temporal Functionality in Objects with Roles Model*) é um modelo de dados temporal orientado a objetos que utiliza papéis representando os comportamentos dos objetos. Como não existe SGBD para o TF-ORM, foi necessário mapear o modelo para um BD relacional. A partir desse mapeamento, deve-se permitir a execução de consultas TF-ORM sobre o BD. Para tal, a consulta deve ser traduzida para SQL (*Structured Query Language*). O objetivo deste artigo é apresentar uma interface de consultas TF-ORM para banco de dados relacionais. A consulta TF-ORM pode ser fornecida a partir de um arquivo texto ou escrita pelo usuário no próprio ambiente. O resultado é a consulta em SQL correspondente, a qual é gravada e mostrada ao usuário. Após o mapeamento, a ferramenta permite sua execução sobre a base de dados.

Palavras-chave: banco de dados, consulta temporal e relacional, mapeamento TF-ORM/SQL.

Abstract

TF-ORM (Temporal Functionality in Objects with Roles Model) is an object oriented temporal data model which uses roles to represent the object behaviors. Because there is no DBMS for TF-ORM, it was necessary to map the model to a relational database. Once this mapping is implemented, the next step is to allow the TF-ORM queries run on the database. To do this, the query must be mapped to SQL (Structured Query Language). The objective of this paper is to present an interface for executing TF-ORM queries on a relational database. The TF-ORM query can be read from a text file or written by the user through the interface. The result is the correspondent SQL query, which is recorded and presented to the user. After the mapping, the interface allows the execution of the query on the database.

Keywords: database, temporal and relational query, TF-ORM/SQL mapping.

1. Introdução

Bancos de dados temporais permitem armazenar todos os estados de um objeto, registrando sua evolução com o tempo [EDE 98, ETZ 98, ÖZS 95, TAN 93, ZAN 97]. Uma vez que a utilização de um modelo de dados temporal não implica, necessariamente, na utilização de um SGBD específico, pode-se usar bancos de dados comerciais através do mapeamento do modelo temporal para o banco em questão. Esse enfoque foi adotado para o modelo TF-ORM (*Temporal Functionality in Objects With Roles Model*) [EDE 93, 94], modelo de dados orientado a objetos que utiliza o conceito de papéis para representar diferentes comportamentos dos objetos.

Tendo-se o mapeamento entre o modelo temporal e o banco de dados comercial, o próximo passo é permitir que o usuário faça consultas TF-ORM [EDE 94a] sobre a base de dados. Assim como o modelo, a consulta também deve ser traduzida para uma linguagem que permita sua execução sobre um BD relacional. A linguagem escolhida foi a SQL (*Structured Query Language*) por ser a linguagem aceita pela maioria dos bancos de dados comerciais e por possuir padrão ANSI. O mapeamento da consulta TF-ORM para SQL foi implementado baseando-se no que foi proposto em [CAR 97].

Esse trabalho apresenta uma ferramenta que realiza o mapeamento de consultas TF-ORM para o banco de dados relacionais. A consulta TF-ORM pode ser lida a partir de um arquivo texto ou escrita pelo usuário via teclado. A ferramenta então realiza as análises léxica, sintática e semântica da consulta e, se a mesma estiver correta, faz o mapeamento. Caso haja algum erro na consulta, o usuário é notificado e o processo é suspenso. O resultado do mapeamento é a consulta em SQL que é gravada em outro arquivo com extensão .SQL e mostrada ao usuário na tela. Também é permitida a execução da consulta SQL sobre a base de dados desejada previamente selecionada. Essa ferramenta foi projetada de modo que sua extensão para a utilização de bases de dados de diferentes fabricantes (Oracle, Sybase, SQL Server, ...) seja possível sem maiores dificuldades, uma vez que o mapeamento é feito para SQL ANSI.

Esse trabalho está dividido da seguinte maneira: na segunda seção é apresentada, resumidamente, a consulta SQL; na terceira, está o modelo TF-ORM com suas características gerais, sua linguagem de consulta e os resumos dos mapeamentos do modelo para o BD relacional e da consulta para SQL; a seguir está o estudo feito para a realização das análises léxica e sintática da consulta TF-ORM; na quinta seção está a descrição do ambiente de mapeamento de consultas.

2. Consulta SQL

De acordo com o ANSI (*American National Standards Institute*), SQL é uma linguagem para SGBDs relacionais. Sentenças SQL são usadas para atualizar e recuperar informações de um banco de dados. Uma vez que o TF-ORM possui apenas uma linguagem de recuperação de dados, essa apresentação da linguagem SQL está restringida a seu comando de seleção. O modelo básico de uma instrução de consulta em SQL é [TOR 99]:

```
SELECT  coluna1 [, coluna2, ...]
FROM    tabela1 [, tabela2, ...]
[WHERE  condição];
```

Os nomes que estão na cláusula `SELECT` determinam quais colunas irão compor o resultado. O usuário pode escrever quantas colunas quiser ou colocar o operador “*” para selecionar todas. Pode-se também colocar expressões de soma, subtração, multiplicação e divisão no resultado das consultas, inserindo-as na cláusula `SELECT`. A cláusula `FROM` determina qual ou quais tabelas do banco de dados devem ser consideradas na consulta.

A cláusula `WHERE` (opcional) especifica a quais condições as linhas retornadas devem obedecer. As condições podem conter uma série de operadores lógicos. Além desses, o operador `LIKE` permite que o usuário selecione somente as tuplas que seguem o padrão especificado. O SQL permite ainda uma série de facilidades, entre elas: junção de duas tabelas, aninhamento de consultas, funções sobre caracteres, números e datas, funções sob grupo de linhas (`COUNT`, `SUM`, `AVG`, `MAX`, `MIN`) e cláusulas especiais como `EXIST`, `GROUP BY`, `HAVING` e `ORDER BY`.

3. Modelo TF-ORM

3.1. Características Gerais

TF-ORM (*Temporal Functionality in Objects with Roles Model*) [EDE 93, 94] é um modelo de dados temporal orientado a objetos que utiliza o conceito de papéis para representar os diferentes comportamentos dos objetos. Esse modelo foi criado com o objetivo de armazenar não somente os valores atuais dos dados, mas toda a sua história – valores passados, atuais e previsões de valores futuros.

Dois diferentes conceitos de tempo podem ser identificados em uma aplicação [JEN 98]: *tempo de transação* no qual uma informação foi introduzida no banco de dados; e *tempo de validade* no qual a informação modela a realidade. No modelo TF-ORM esse tempos são utilizados como rótulos associados às propriedades que descrevem as instâncias e os atributos que variam com o tempo. O tempo é modelado variando de forma discreta, estando linearmente ordenado.

Com o objetivo de separar os aspectos estáticos dos aspectos dinâmicos, o TF-ORM apresenta o conceito de papéis. O objeto continua a ser instância de uma única classe, mas poderá assumir ao longo de sua existência um ou mais papéis (comportamentos), inclusive diversas instâncias de um mesmo papel simultaneamente.

O modelo apresenta dois tipos de propriedades: *propriedade estática*, mantém seu valor no tempo; *propriedade dinâmica*, pode alterar seu valor no tempo, sendo que todos os valores devem ficar armazenados permitindo a recuperação do histórico de uma instância.

Cada classe é definida por um nome e papéis que sua instância pode assumir. São três tipos de classes: *recursos*, modelam as informações e documentos; *processos*, atividades que podem ser feitas com as informações e os documentos; e *agentes*, pessoas que desempenham as atividades. Cada objeto de classe possui um identificador único que é definido no instante de sua criação e armazenado na propriedade estática pré-definida *oId*. Todo objeto apresenta um papel básico que descreve as características iniciais de uma instância e as propriedades globais que controlam sua evolução. A criação de um objeto de uma classe instancia automaticamente o seu papel básico que, ao contrário dos demais papéis, só pode ser instanciado uma vez.

Cada papel é representado por um nome, propriedades, estados possíveis para as instâncias desse papel, mensagens que podem ser enviadas ou recebidas, regras de transição de estado e restrições de integridade. Cada papel também possui um identificador único armazenado na propriedade estática pré-definida *rId*.

Além das propriedades estáticas pré-definidas *oId* e *rId*, o modelo TF-ORM apresenta também propriedades dinâmicas pré-definidas: *object_instance*, intervalos de tempo em que a instância está ativa; *end_object*, instante de tempo em que a instância foi descontinuada; *role_instance*, eventuais suspensões e reativações da instância do papel; *end_role*, tempos de transação e validade do final da vida da instância de um papel.

3.2. Linguagem de Consulta TF-ORM

A linguagem de consulta do modelo TF-ORM baseia-se na linguagem SQL e apresenta extensões para suportar os aspectos temporais [EDE 94a]. A estrutura geral do comando SQL é mantida: cláusulas de especificação (`SELECT`), de identificação (`FROM`) e de busca (`WHERE`). Adicionalmente, a linguagem apresenta a cláusula `WHEN`, permitindo assim duas formas alternativas para a consulta:

```
SELECT  <cláusula de especificação>
FROM    <cláusula de identificação>
[ { WHERE | WHEN }  <cláusula de busca> ]
[ AS ON <cláusula de instante temporal> ]
```

A cláusula de especificação define a saída da consulta: de dados, temporal ou mista. As saídas temporal e mista são obtidas apenas com o uso da cláusula `WHEN`, pois a cláusula `WHERE` recupera apenas os dados da base atual (válidos hoje). Quando a cláusula de especificação contiver uma data de transação, uma data de validade ou período, a saída será temporal. Essas informações temporais estão associadas a cada uma das propriedades dinâmicas do modelo, já as propriedades estáticas são válidas enquanto a instância estiver ativa, por isso não possuem data associada. Na cláusula de identificação são especificados as classes e papéis sobre os quais se deseja recuperar informações. Na cláusula de busca estão as condições que devem ser satisfeitas pelos objetos recuperados.

A cláusula `WHEN` permite a recuperação de toda a história do banco de dados (presente, passado e futuro). Nesse caso, duas situações podem ocorrer: a cláusula de busca não possui nenhuma restrição temporal, ou seja, serão analisadas todas as informações do banco de dados; a cláusula de busca contém as palavras `DATE`, `PERIOD`, `TRANSACTION_DATE` e/ou `TRANSACTION_PERIOD` e serão analisadas apenas as informações que satisfazem a essa restrição temporal.

A cláusula de instante temporal (`AS ON`) define a data correspondente a uma história passada, mudando o referencial de tempo da consulta para o momento definido e utilizando como base as informações conhecidas naquele instante. O modelo TF-ORM possui também operadores temporais, predicados e funções que podem ser utilizados na cláusula de busca determinando o momento do tempo no qual a expressão deve ser avaliada.

Na figura 1 são apresentados três exemplos de consulta TF-ORM onde seleciona-se a propriedade estática `CPF` e a propriedade dinâmica `salario` do papel `medico` da classe `pessoa` cujo valor do salário é maior que 2000. Os resultados das três consultas são totalmente diferentes: a consulta A retorna os valores referentes aos médicos cujos salários são maiores que 2000 hoje; a consulta B retorna o CPF e o salário de todos os médicos que alguma vez na história tiveram salários maiores que 2000, valores de salários do passado, do presente e do futuro; a consulta C retorna o CPF e o salário dos médicos conforme os valores da base no dia primeiro de janeiro de 1999.

```
SELECT CPF, salario
FROM   pessoa.medico
WHERE  salario > 2000
```

Consulta A

```
SELECT CPF, salario
FROM   pessoa.medico
WHEN   salario > 2000
```

Consulta B

```
SELECT CPF, salario
FROM   pessoa.medico
WHEN salario > 2000  AS ON 01/01/199
```

Consulta C

Figura 1 – Exemplos de consulta TF-ORM

3.3. Mapeamento do Modelo TF-ORM para Banco de Dados Relacional

O mapeamento do esquema TF-ORM para BD relacional é realizado mapeando cada classe, subclasse, papel, propriedade dinâmica e estado para uma tabela no BD relacional [CAR 97].

Para mapear as propriedades dinâmicas, quatro informações temporais são associadas: tempo de validade inicial (`v_timei`), tempo de validade final (`v_timef`), tempo de transação inicial (`t_timei`) e tempo de transação final (`t_timef`).

Cada classe é mapeada para uma tabela de mesmo nome que contém as propriedades estáticas como atributos e `oId` como identificador. As propriedades dinâmicas pré-definidas (`object_instance`, `v_timei`, `v_timef`, `t_timei`, `t_timef`) juntamente com o oId (faz o relacionamento com a tabela da classe) são colocadas em outra tabela (`NomeClasse_dynamic`). As demais propriedades dinâmicas são mapeadas uma para cada tabela, cujo nome é formado por `NomeClasse_NomePropriedade`, que está relacionada à tabela base da classe através do atributo `oId`. O mapeamento do papel difere do mapeamento da classe com a inclusão do atributo `rId` como identificador do papel, e o atributo `oId` que relaciona essa tabela à tabela da classe.

3.4. Mapeamento da Consulta TF-ORM para SQL

O mapeamento da linguagem de consulta TF-ORM para a linguagem SQL é realizado em duas etapas: definição das tabelas relacionamentos e restrições sobre dados, e definição das restrições temporais [CAR 97].

Definição das tabelas, relacionamentos e restrições sobre dados: o objetivo é identificar em que tabelas do banco de dados as propriedades estão armazenadas, como essas tabelas se relacionam e quais restrições sobre dados podem ser colocadas diretamente na consulta SQL.

A análise começa na cláusula `SELECT`, onde são consideradas apenas as propriedades. O sistema deve descobrir em que tabela do banco de dados está armazenada a propriedade. O nome da tabela deve ser incluído na cláusula `FROM` da consulta SQL. Além disso, devem ser incluídas a tabela da classe e do papel que armazenam o identificador da classe com as propriedades estáticas, e o identificador do papel com suas propriedades estáticas. Na consulta SQL, os atributos da cláusula `SELECT` são escritos na forma `NomeTabela.atributo`.

Para cada propriedade da cláusula `WHERE` da consulta TF-ORM é feita uma consulta ao sistema do BD para descobrir em que tabela essa propriedade está armazenada. Se o nome da tabela não estiver na cláusula `FROM` da consulta SQL, o sistema inclui esse nome. A ferramenta também inclui na cláusula `FROM` as tabelas que serão utilizadas por alguns predicados e funções.

Depois que a cláusula `FROM` estiver completa, a ferramenta monta a cláusula `WHERE` da consulta SQL. São definidos todos os relacionamentos existentes entre as tabelas da cláusula `FROM` da consulta SQL. Depois a ferramenta acrescenta as restrições especificadas pela consulta TF-ORM. Por último é feito o mapeamento das funções e predicados encontrados na cláusula `WHERE` da consulta TF_ORM.

Definição das restrições de tempo: essa é a última etapa do mapeamento e é realizada através de um conjunto de regras. Palavras especiais que permitem a indicação de informações temporais como `DATE`, `PERIOD`, `TRANSACTION_DATE` e `TRANSACTION_PERIOD`, a cláusula especial `AS ON`, operadores temporais como `ALWAYS PAST` e `ALWAYS FUTURE`, são tratados separadamente e estão todos completamente especificados em [CAR 97].

4. Análises Léxica e Sintática

Um compilador é uma ferramenta que lê um programa escrito em uma linguagem (*fonte*) e o converte para um programa equivalente em outra linguagem (*alvo*). Como parte importante do processo, o compilador reporta ao seu usuário a presença de erros. A principal função da ferramenta desse trabalho é servir de "tradutor" de consultas TF-ORM para SQL. Nos termos de compilação, a ferramenta recebe o arquivo *fonte* em TF-ORM e o traduz para o arquivo *alvo* em SQL ou retorna os erros encontrados no arquivo *fonte*.

O processo de compilação consumia bastante tempo até Lesk e Johnson publicarem seus artigos sobre o Lex e o Yacc – [LES 75] e [JOH 75]. Esses dois utilitários simplificaram enormemente a tarefa de construir um compilador [NIE 98]. Essas ferramentas geram um componente de análise sintática tendo como base uma gramática especificando a sintaxe da linguagem. O Lex gera código para o analisador léxico, ou *scanner*, e o Yacc gera código C para o analisador sintático, ou *parser*.

Novas pesquisas têm sido feitas nessa área desde então. Um dos resultados é o PRECCX (*PREttier Compiler Compiler eXtended*), um tradutor que converte um *script* de definição de gramática em código C ANSI [BRE 94]. O código de saída pode ser compilado em qualquer compilador ANSI. Ele prolonga as facilidades do Yacc do Unix e dos outros equivalentes para PC, incluindo aqueles baseados no Bison da GNU. Entre elas: *lookahead* infinito em vez do *lookahead* de um *token* do Yacc; compilação e lincagem incrementais de *scripts* arranjados em módulos separados; suporte para descrições de BNFs extendidas; definição de gramática parametrizadas, permite gramáticas dependente de contexto; variáveis que servem para a gramática como parâmetros, permitindo "macros" para substituir construções gramaticais repetitivas; permite o uso de colchetes para representar segmentos opcionais da gramática. Outra vantagem do PRECCX é o tamanho do arquivo de entrada aceito: o Yacc tem limitação de 255 caracteres enquanto o PRECCX aceita arquivos maiores.

A desvantagem do PRECCX em relação ao Yacc é que, até o presente momento (versão 2.42), não há equivalência para as declarações de precedência e associatividade existentes no Yacc. Em vez disso, é necessário codificar explicitamente a ordem requerida.

5. Descrição do Ambiente

5.1. Detalhes da implementação

A interface do sistema foi implementada utilizando-se a ferramenta para desenvolvimento de aplicações Delphi. Optou-se por implementar o modelo sobre um banco de dados relacional, por ser este o tipo de banco de dados mais utilizado atualmente. Inicialmente foi implementada uma ferramenta para interação com o banco de dados *Sybase SQL Anywhere*. Com o intuito de melhorar o desempenho da ferramenta, diminuindo o tempo de retorno das informações da base e o consumo de recursos do sistema, fez-se uma adaptação para o banco de dados *Personal Oracle 7.3 for Windows 95*.

5.2. Características

A principal função da ferramenta é servir de interface para consultas TF-ORM em um banco de dados relacional. Adicionalmente, a ferramenta pode apenas verificar se a consulta TF-ORM está correta léxica, sintática e semanticamente, como também executar a consulta sobre a base de dados e mostrar o resultado em forma de tabela. A consulta TF-ORM pode ser lida a partir de um arquivo texto ou escrita pelo usuário via teclado.

A ferramenta também informa se o arquivo contem erro, relatando o tipo e onde ele ocorreu. O resultado do mapeamento, a consulta em SQL, é gravado em outro arquivo com extensão .SQL e

mostrado ao usuário na tela. Com o arquivo SQL, o usuário pode executar a consulta na base de dados e a ferramenta retorna os resultados.

Desde o princípio, fundamentou-se que a consulta SQL resultante do mapeamento deveria seguir o padrão ANSI, para poder ser executada em qualquer banco de dados relacional comercial. Desse modo, a linguagem utilizada na consulta TF-ORM deve ser aceita pelo banco de dados evitando confusões com o conjunto de caracteres para suporte de linguagem adotado pelo mesmo. Por exemplo, na implementação realizada a consulta TF-ORM não pode apresentar acentuação ou a letra "ç" em qualquer uma das cláusulas.

Para evitar problemas de versões de arquivos diferentes, estipulou-se que o arquivo com a consulta TF-ORM deve estar salvo antes de realizar o mapeamento. Caso o usuário faça qualquer modificação no arquivo, ele deverá salvá-lo (com o mesmo nome ou não) para poder realizar a tradução ou a verificação da consulta. A figura 2 mostra como ficou a interface final.

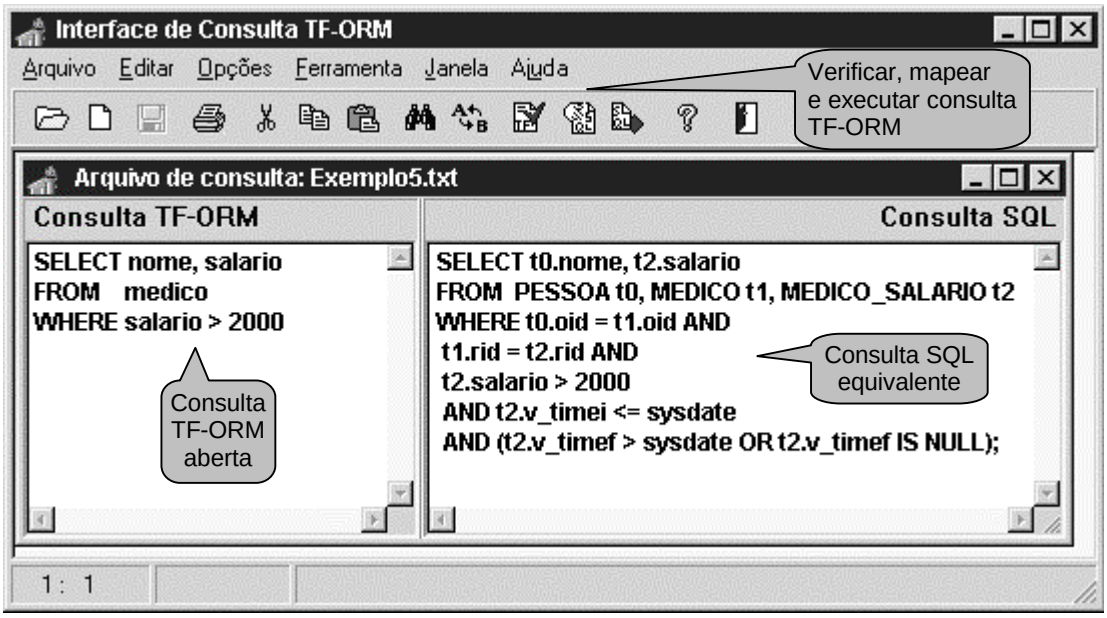

Figura 2 – Interface de consultas TF-ORM

A tela principal apresenta um menu, uma barra de botões com as principais opções do menu e uma área para as consultas. A maioria dos comandos do menu apresenta teclas de atalho. Os botões de *Verificação* e *Mapeamento*, bem como seus comandos no menu, são habilitados quando há alguma consulta TF-ORM aberta. Já o botão e o comando de *Executar a Consulta* é habilitado quando existe uma consulta SQL aberta, como mostra a figura 2.

5.3. Funcionamento

Ao iniciar a execução da ferramenta implementada, o usuário deve escolher qual base de dados (conjunto de dados armazenados) será considerada durante a verificação ou mapeamento da consulta TF-ORM. A primeira tarefa da ferramenta é transparente ao usuário. Ela armazena as informações da base de dados em uma estrutura própria. Desse modo, os nomes das classes, papéis e propriedades do esquema TF-ORM usado são armazenados em variáveis locais. É importante salientar que o nome das tabelas do sistema do banco de dados mudam conforme o produto que está sendo usado. No banco de dados da *Sybase*, as tabelas usadas são: *sys.systable, sys.syscolumns e sys.sysforeignkeys*. Já as tabelas

usadas pelo produto da *Oracle* são*: all_tables* e *all_tab_columns*, sendo analisadas só as que são propriedades de usuários.

O usuário pode abrir um arquivo contendo a consulta TF-ORM ou digitá-la na parte esquerda da interface. Quando a consulta aberta pelo usuário já possui uma consulta SQL equivalente gravada, ou seja, com o mesmo nome da consulta TF-ORM com extensão .SQL, a ferramenta também a mostra no espaço da direita (figura 2).

Caso o usuário queira apenas verificar se o arquivo contém uma consulta TF-ORM, após a seleção no menu ou no botão de *Verificação da Consulta*, a ferramenta realiza as análises léxica, sintática e semântica. O mapeamento pode, opcionalmente, ser realizado. Em ambos os casos (sendo ou não realizado o mapeamento da consulta), uma janela é aberta mostrando o andamento do processo. Caso o mapeamento ocorra com sucesso, a ferramenta apresenta ao usuário a consulta SQL resultante. Nesse caso, o comando e o botão *de Execução da Consulta* ficam habilitados. O usuário tem a opção de salvar a consulta SQL no menu *Arquivo*. Ele pode salvar com o mesmo nome do arquivo da consulta TF-ORM ou com outro. A consulta SQL também pode ser executada no banco de dados. Após sua execução, a ferramenta mostra uma tela com as tuplas resultantes da consulta em forma de tabela.

5.4. Exemplos

A figura 3 mostra as três consultas TF-ORM apresentadas na figura 1 com seus respectivos mapeamentos para consulta SQL retornados pela ferramenta.

Consultas TF-ORM	Consultas SQL

`SELECT CPF, salario`
`FROM pessoa.medico`
`WHERE salario > 2000`
 	
A
	`SELECT t0.cpf, t2.salario`
	`FROM PESSOA t0, MEDICO t1, MEDICO_SALARIO t2`
	`WHERE t0.oid = t1.oid AND t1.rid = t2.rid AND`
	`t2.salario > 2000`
	`AND t2.v_timei <= sysdate`
	`AND (t2.v_timef > sysdate OR t2.v_timef IS NULL);`

`SELECT CPF, salario`
`FROM pessoa.medico`
`WHEN salario > 2000`
 	
B
	`SELECT t0.cpf, t2.salario`
	`FROM PESSOA t0, MEDICO t1, MEDICO_SALARIO t2,` **`PESSOA_DYNAMIC t3`**
	`WHERE t0.oid = t1.oid AND` **`t0.oid = t3.oid`** `AND t1.rid = t2.rid AND`
	`t2.salario > 2000`
	`AND (t2.v_timei <= t3.v_timef OR t3.v_timef IS NULL)`
	`AND (t2.v_timef >= t3.v_timei OR t2.v_timef IS NULL);`

`SELECT CPF, salario`
`FROM pessoa.medico`
`WHEN salario > 2000`
`AS ON 01/01/1999`
 	
C
	`SELECT t0.cpf, t2.salario`
	`FROM PESSOA t0, MEDICO t1, MEDICO_SALARIO t2, PESSOA_DYNAMIC t3`
	`WHERE t0.oid = t1.oid AND t0.oid = t3.oid AND t1.rid = t2.rid AND`
	`t2.salario > 2000 AND`
	`AND (t2.v_timei <= t3.v_timef OR t3.v_timef IS NULL)`
	`AND (t2.v_timef >= t3.v_timei OR t2.v_timef IS NULL)`
	`t3.t_timei <= '01/01/1999' AND t2.t_timei <= '01/01/1999';`

Figura 3 – Exemplos de mapeamento de consulta TF-ORM para SQL

Na consulta SQL resultante de A, pode-se observar os relacionamentos entre as tabelas da classe `pessoa` e do papel `medico`. É requisitado que o tempo de validade inicial seja menor que a data de hoje bem como o tempo de validade final seja maior que a data de hoje ou nulo (em negrito), pois a consulta tem de considerar os valores válidos das propriedades dinâmicas na base de dados hoje. Na consulta resultante de B, pode-se verificar as diferenças da consulta A em negrito. Como temos uma

cláusula `WHEN`, deve-se consultar toda a história da base, verificando os intervalos no tempo onde há intersecção dos valores das propriedades `cpf` e `salario` (últimas duas linhas da consulta). Como na consulta C foram requeridos valores válidos na base em 01/01/1999, devem ser consideradas apenas as informações que foram inseridas na base até a data especificada, então o tempo de transação inicial das propriedades deve ser menor ou igual a essa data.

5.5. Verificações e Erros

A primeira verificação que a ferramenta faz é se a consulta TF-ORM apresenta as cláusulas `SELECT`, `FROM` e `WHERE` ou `WHEN`. A segunda verificação é feita pelo compilador gerado pelo PRECCX com as análises léxica e sintática, retornando a linha e o símbolo onde ocorreu o erro (figura 4). A terceira verificação é a análise semântica executada pela própria ferramenta. Essa análise verifica se as tabelas e atributos da consulta existem no banco de dados. Caso o erro seja no parâmetro de alguma função ou predicado, a tela de andamento apresenta a propriedade que não pertence à base de dados e o nome da função na qual é mencionada (figura 5). Em todos os casos de erro, o mapeamento é cancelado.

Exemplos de Erros Retornados: caso a consulta A da figura 3 apresente o símbolo "[" no lugar de ">", a tela de mensagens com o andamento do mapeamento apresenta o conteúdo da figura 4.

```
>>> Mapeamento de C:\Users\Mirella\Ferramenta\Exemplo5.txt

 Preparando arquivo
 Verificando se a consulta é TF-ORM
 Iniciando análises léxica e sintática
ERRO na linha 3: análise falhou perto de [
   >>> Mapeamento cancelado <<<
```

Figura 4 – Exemplo de erro léxico ou sintático na consulta TF-ORM

Caso o usuário se engane na hora da escrita da consulta e escreva `salari` ao invés de `salario`, a tela de mensagens aparece como na figura 5.

```
>>> Mapeamento de C:\Users\Mirella\ Ferramenta\Exemplo5.txt

 Preparando arquivo
 Verificando se a consulta é TF-ORM
 Iniciando análises léxica e sintática
    analises léxica e sintática concluídas com sucesso
 Iniciando análise semântica
ERRO: a Propriedade SALARI não é válida para essa base de dados.
   >>> Mapeamento cancelado <<<
```

Figura 5 – Exemplo de erro semântico na consulta TF-ORM

5.6. Gramática utilizada

A gramática usada como entrada no PRECCX foi escrita baseada em [CAR 97]. Foi feito um estudo para adequá-la ao formato de entrada do compilador. Entre as modificações e ajustes estão: modificação dos operadores de comparação de ≥, ≤ e ≠ para >=, <= e <>, seguindo as normas do padrão ANSI para SQL; visto que os BDs comerciais aceitam tabelas e colunas cujos nomes começam com sublinhado ("_"), mudou-se o identificador para que aceite como primeiro caracter uma letra ou um sublinhado; exclusão da recursividade à esquerda; redefinição de *property value* (valores das propriedades) para aceitar números inteiros.

6. Conclusão

Esse trabalho apresentou uma ferramenta de interface de consultas TF-ORM para base de dados relacionais. A consulta TF-ORM pode ser lida a partir de um arquivo texto ou escrita pelo usuário via teclado. O resultado é a consulta SQL correspondente, a qual é gravada em outro arquivo e mostrada ao usuário. Após o mapeamento, a ferramenta permite sua execução sobre a base de dados desejada.

Como principal resultado desse trabalho, tem-se a consulta TF-ORM mapeada para SQL/ANSI, que pode ser executada em qualquer BD relacional comercial que aceite o padrão. Como resultado adicional, tem-se a possibilidade de apenas verificar se a consulta TF-ORM é válida. Um compilador gerado pelo PRECCX executa as análises léxica e sintática da consulta TF-ORM. Se não ocorrerem erros, a ferramenta realiza a análise semântica da consulta. Uma tela de mensagens é apresentada com o andamento da verificação e com os erros encontrados na consulta.

Para trabalhos futuros estão a extensão para que a ferramenta aceite mais de uma classe e papel sendo consultados. Isso implicaria no estudo da sintaxe do TF-ORM para aceitar tal modificação. Além disso, essa ferramenta foi planejada de modo que extensões para outras bases de dados, como SQL Server, pudessem ser feitas sem maiores dificuldades. Todas as consultas às tabelas do sistemas são realizadas logo que o usuário seleciona o BD. As informações que interessam, nome das tabelas e de suas colunas, são armazenadas em uma estrutura própria da ferramenta. O único trabalho que se teria para mudar de base, seria o de adaptar o nome das tabelas e atributos do sistema que a ferramenta consulta.

Outra extensão a ser realizada é permitir que o usuário escolha a forma de apresentação do resultado da consulta. Na ferramenta, o resultado é apresentado em forma de tabela. Algumas consultas seriam mais compreensíveis se o resultado viesse em forma de gráficos ou mostrando as propriedades desenhadas sobre a linha do tempo.

7. Bibliografia

[AHO 86] AHO, Alfred V.; SETHI, Ravi. **Compilers – Principles, Techniques, and Tools**. Stanford: Addison-Wesley Publishing Company, 1986. 796 p.

[BRE 94] BREUER, Peter T.; BOWEN, Jonathan. P. **PRECCX User Manual**. Oxford, Reino Unido, 1994. Disponível por WWW em http://www.comlab.ox.ac.uk/archive/redo/precc/user-manual.html (12/04/1999).

[CAR 97] CARVALHO, Tanisi. P. **Implementação e Consultas para um Modelo de Dados Temporal Orientado a Objetos**. Porto Alegre: CPGCC – UFRGS, 1997. Dissertação de Mestrado.

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

[EDE 94] EDELWEISS, Nina**. Sistemas de Informações de Escritórios: um modelo para Especificações Formais**. Porto Alegre: CPGCC - UFRGS, 1994. Tese de doutorado.

[EDE 94a] EDELWEISS, N.; OLIVEIRA, J.P.M. de; PERNICI, B. An Object-oriented approach to a temporal query language. In: DATABASE AND EXPERT SYSTEMS APPLICATIONS - DEXA'94, 5., Sept. 7-9, 1994, Athens, Greece. **Proceedings**. Heidelberg: Springer-Verlag, 1994. p.225-235. (Lecture Notes in Computer Science 856).

[EDE 98] EDELWEISS, N. Bancos de Dados Temporais: Teoria e Prática. In: XVII Jornada de Atualização em Informática, do XVIII Congresso Nacional da Sociedade Brasileira de Computação, v.2. Ed.: H.P. MOURA. **Anais**. Recife: Sociedade Brasileira de Computação, 1998. p.225-282.

[ETZ 98] ETZION, O.; JAJODIA, S.; SRIPADA, E. (Eds.) **Temporal Databases: Research and Practice.** Berlin: Springer-Verlag, 1998. LNCS1399.

[JEN 98] JENSEN, C.S. et al. The Consensus Glossary of Temporal Database Concepts - February 1998 Version. In: **Temporal Databases Research and Practice**. O. Etzion, S. Jajodia and S. Sripada (eds.) Springer-Verlag. Berlin Heidelberg 1998. pp. 367-405.

[JOH 75] JOHNSON, Stephen C. **Yacc: Yet Another Compiler Compiler**. Technical Report 32, New Jersey, United States of America. Bell Laboratories, Murray Hill, 1975. Disponível por WWW em http://members.xoom.com/thomasn/y_yacc.pdf (12/04/1999).

[LES 75] LESK, M. E.; SCHMIDT, E. **Lex – A Lexical Analyzer Generator**. Technical Report 39, New Jersey, United States of America. Bell Laboratories, Murray Hill, 1975. Disponível por WWW em http://members.xoom.com/thomasn/y_lex.pdf (12/04/1999).

[NIE 98] NIEMANN, Thomas. **A Guide to Lex & Yacc**. Portland, Oregon. Disponível por WWW em http://members.xoom.com/thomasn/y_man.htm (12/04/1999).

[ÖZS 95] ÖZSOYOGLU, G.; SNODGRASS, R. T. Temporal and real-time databases: a survey. **IEEE Transactions on Knowledge and Data Engineering**, New York, v.7, n.4, p.513-532, Aug.1995.

[TAN 93] TANSEL, Clifford G.; et al. (editores). **Temporal Databases - theory, design and implementation**. Redwood City: The Benjamin/Cummings Publishing Company, Inc., 1993.

[TOR 99] TORRES, Frank. **SQL Tutorial & Interpreter w/ Live Practice Database**. Disponível por WWW em http://torresoft.netmegs.com/ (10/04/1999).

[ZAN 97] ZANIOLO, C. et al. **Advanced Database Systems**. San Francisco: Morgan Kaufmann Publishers, 1997. 574p.