EMap - Uma Interface de Consultas Temporais em SGBDs Relacionais

Edimar Manica[1,*], Cristiano R. Cervi[1, 2], Carina F. Dorneles[3], Renata Galante[1]

[1]Instituto de Informática - Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

[2]Instituto de Ciências Exatas e Geociências - Universidade de Passo Fundo (UPF)
Bairro São José – BR 285 – Km 171 – 99001-970 – Passo Fundo – RS – Brasil

[3]Departamento de Informática e Estatística - Universidade Federal de Santa Catarina (UFSC)
Campus Universitário Trindade - 88049-900 - Florianópolis – SC – Brasil

{edimar.manica, galante}@inf.ufrgs.br, cervi@upf.br, dorneles@inf.ufsc.br

Resumo. *Este artigo apresenta a ferramenta EMap, a qual oferece uma interface Web que possibilita que usuários executem consultas escritas na linguagem de consulta temporal TSQL2 sobre SGBDs sem suporte a consulta temporal. Isto é possível através de um mapeamento de TSQL2 para SQL-92. O artigo ainda apresenta alguns experimentos que comprovam a viabilidade e corretude da ferramenta, através da avaliação de usuários. Os experimentos possuem duas contribuições: validar a ferramenta e analisar a qualidade dos resultados através de testes com usuários.*

Abstract. *This paper presents a tool that implements a proposal for mapping temporal query statements in TSQL2 to SQL-92, allowing query databases stored in DBMSs that do not have temporal query support. This work still presents the mapping specification and some experiments that show the viability and correctness of the tool, by means of user evaluation, are presented. The experiments have, basically, two contributions: validate the tool and analyze the results quality using tests executed with users.*

1. Introdução

Bancos de dados temporais, em amplo sentido, cercam todas as aplicações de banco de dados com algum aspecto de tempo na organização de suas informações (Elmasri e Navathe 2005). Quando se utiliza um modelo de dados temporal para especificar uma aplicação, não se restringe, necessariamente, à utilização de um SGBD (Sistema de Gerenciamento de Banco de Dados) específico para o modelo adotado. SGBDs convencionais podem ser utilizados desde que haja um mapeamento adequado entre o modelo de dados temporal e o modelo adotado pelo SGBD utilizado. Neste caso, a recuperação de informações utilizando a linguagem de consulta do próprio SGBD exige que o usuário conheça em detalhes este mapeamento. Contudo, a utilização de um banco de dados convencional para armazenar as informações do modelo não deve eliminar a possibilidade de utilização de uma linguagem de consulta temporal (Edelweiss 1998). Estas linguagens permitem a descrição de consultas temporais em alto nível, eliminando a necessidade do usuário conhecer os aspectos de implementação da temporalidade. Além da possibilidade de definir filtros sobre os dados envolvidos na consulta, o usuário pode definir faixas de tempo sobre as quais os dados devem ser considerados.

O problema abordado neste artigo é a necessidade dos usuários conhecerem a implementação da temporalidade para realizar consultas temporais sobre um SGBD que não oferece suporte a nenhuma linguagem de consulta temporal. A diferença no nível de abstração de uma consulta utilizando-se a linguagem SQL-92 e uma linguagem de consulta temporal pode ser observada na Figura 1. Ela apresenta duas consultas que visam retornar os mesmos dados, mas uma foi definida na linguagem SQL-92 e a outra na linguagem TSQL2, definida por (Snograss 1995). As duas consultas retornam o histórico de níveis de bolsa do pesquisador que possui código igual a 2 (nível de bolsa e período). Observa-se na Figura 1 que em SQL convencional o usuário precisa indicar quais atributos

[*] Este trabalho foi desenvolvido enquanto o autor era aluno de graduação da Universidade de Passo Fundo.

representam o tempo, enquanto que em TSQL2 utiliza apenas a função `VALID()` para obter as informações temporais desejadas. Além disso, em SQL convencional é necessário saber em qual tabela o atributo temporal `nivel_bolsa` está fisicamente implementado, diferentemente da TSQL2 onde é necessário apenas conhecer em qual tabela ele foi modelado.

Consulta em SQL-92

```
SELECT t1.nivel_bolsa ,
       '[' || t1.vt_begin || ',' || t1.vt_end || ')' AS periodo
  FROM pessoa pessoa, pesquisador_bolsa t1
 WHERE t1.vt_begin >= '-infinity'  AND  pessoa.cd_pessoa = 2
   AND t1.cd_pessoa = pessoa.cd_pessoa AND t1.tt_stop = 'infinity'
```

Consulta em TSQL2

```
SELECT nivel_bolsa, valid(nivel_bolsa) AS periodo
  FROM pessoa
 WHERE begin(valid(nivel_bolsa)) >= '-infinity'
   AND cd_pessoa = 2
```

Figura 1. Exemplo de consulta em SQL convencional e consulta em TSQL2

Este artigo descreve a ferramenta *EMap* (Extrai e Mapeia), que possui as seguintes contribuições: (i) disponibiliza uma interface amigável para o usuário realizar consultas temporais em TSQL2, apresentando a sintaxe equivalente em SQL-92; e (ii) efetua o mapeamento da TSQL2 para SQL-92, a fim de que consultas temporais possam ser executadas sobre SGBDs que não possuem linguagens de consulta com suporte ao tratamento de temporalidade. Para isso, foi definido um conjunto de regras de mapeamento para funções e operadores temporais. Foram realizados também estudos de caso sobre os SGBDs PostgreSQL e Mysql. Para a implementação foi utilizado o modelo bitemporal definido pelo `Btpgsql` (Howard 2008), que emula um banco de dados bitemporal no PostgreSQL, mas que não provê suporte a uma linguagem de consulta temporal. No PostgreSQL, o Btpgsql controla as restrições temporais durante a atualização dos dados, porém nos demais SGBDs este controle deve ser realizado pela aplicação. O presente trabalho é uma extensão de trabalhos prévios, diferenciando-se destes por especificar os mapeamentos e implementar a ferramenta *EMap* e suas funcionalidades enquanto que (Cervi et al. 2008) apresenta uma visão geral do contexto onde o trabalho está inserido e (Manica et al. 2009) descreve as adaptações da implementação do mapeamento para ser independente de um SGBD específico.

O artigo está organizado como segue. Na Seção 2 são apresentados os principais conceitos de banco de dados temporais utilizados no trabalho. A Seção 3 descreve a ferramenta *EMap*, enquanto os experimentos e resultados obtidos são apresentados na Seção 4. Os trabalhos relacionados são apresentados na Seção 5. Finalmente, na Seção 6, são descritas as considerações finais e as direções futuras.

2. Conceitos de Bancos de Dados Temporais

Os principais conceitos de bancos de dados temporais utilizados para concepção da ferramenta *EMap* são os seguintes (Edelweiss 1998): **(i) tempo de transação**: é o tempo em que o fato é armazenado no banco de dados, sendo definido pelo próprio SGBD; **(ii) tempo de validade**: é o tempo durante o qual um fato do banco de dados é verdadeiro na realidade modelada. São definidos pelo próprio usuário; e **(iii) banco de dados bitemporal**: são os bancos de dados que possuem suporte ao tempo de transação e ao tempo de validade.

Elmasri e Navathe (2005) definem alguns conceitos referentes à implementação de relações temporais em bancos de dados relacionais: **(i) atributos síncronos**: são os atributos que quando o valor de um atributo evolui, o valor dos demais também evolui; **(ii) atributos assíncronos**: são os atributos que quando o valor de um atributo muda, o valor dos demais pode não mudar; e **(iii) particionamento vertical**: é a forma de implementar relações temporais utilizada neste trabalho, onde para cada entidade, com atributos temporais assíncronos, presente no modelo conceitual (entidade denominada aqui de **tabela origem**) criam-se no modelo lógico N tabelas (denominadas aqui de **tabelas físicas**) de modo a deixar em uma mesma tabela apenas os atributos síncronos e replicando a chave primária em cada tabela. Esta maneira evita a redundância de criar uma nova versão das tuplas inteiras sempre que qualquer um dos atributos for atualizado. Quando dois atributos temporais, particionados em tabelas distintas, precisam ser acessados ao mesmo tempo, é necessário combinar a informação com uma variação de junção denominada junção de interseção temporal, a qual é responsável pela junção de duas tabelas pelo tempo. Esta junção pode ser realizada explicitamente através do operador temporal INTERSECT, como é o caso dos trabalhos relacionados, ou implicitamente como é o caso da ferramenta *EMap*.

3. Ferramenta EMap

O objetivo principal da ferramenta *EMap* é o mapeamento de consultas escritas em TSQL2 para consultas em SQL-92 de forma transparente ao usuário. Com isso, é possível que as consultas sejam executadas sobre SGBDs convencionais (sem suporte à temporalidade) que representem o tempo segundo o modelo físico implementado pelo Btpgsql, porém sem necessitar que o usuário conheça a implementação da temporalidade. Na Figura 2 é apresentada a arquitetura da ferramenta *EMap*, onde os seguintes passos são executados: **(i)** o usuário escreve uma consulta em TSQL2, que é lida pelo módulo de mapeamento; **(ii)** o módulo de mapeamento substitui a consulta em TSQL2 por uma consulta em SQL convencional equivalente e a submete ao SGBD; e **(iii)** o SGBD processa a consulta SQL e retorna o conjunto-resposta para o usuário.

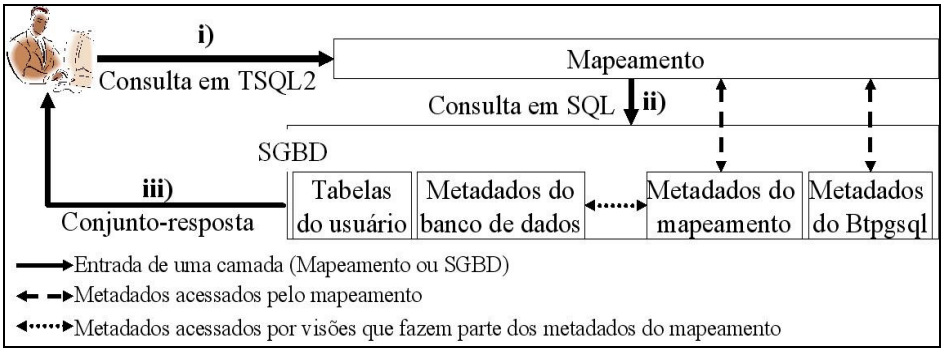


Figura 2. Arquitetura da ferramenta EMap

A ferrramenta EMap implementa os principais operadores e funções temporais, permitindo ao usuário consultar as versões correntes, as versões passadas e as versões que se acreditava serem válidas em um tempo passado. Para isso, foi definido um subconjunto de TSQL2 que contém os elementos apresentados na Tabela 1, onde se destacam as funções e os predicados (ou operadores) temporais, os quais não são aceitos em SQL-92. O conjunto de elementos definido é suficiente para prover suporte à maioria das consultas temporais necessárias em uma aplicação.

Tabela 1. Elementos contidos

Cláusulas	SELECT, FROM, WHERE
Operadores aritméticos	+, -, *, /
Operadores condicionais	<, >, =, <>, <=, >=, IS NULL, IS NOT NULL., LIKE, ~*
Operadores lógicos	AND, OR, NOT
Operadores Temporais	OVERLAPS, INTERSECT, PRECEDES
Funções Temporais	VALID**, TRANSACTION**, BEGIN, END
Granularidade	Única - *timestamp*
Particionamento Vertical	Sim
Palavra reservada AS	Apenas no SELECT

3.1 Apresentação da Ferramenta

A ferramenta *EMap* está acessível para testes no site http://mosaico.upf.br:8080/EMap. Sua implementação foi realizada utilizando a linguagem Java, o *framework* Struts 2 e o IDE Netbeans versão 5.5. É uma ferramenta portável, funcionando tanto em Linux quanto em Windows. Além disso, por ser uma ferramenta Web permite que seja instalada em um servidor e acessada remotamente via *browser* por diversos usuários, de modo que estes necessitam apenas de um navegador Web para utilizar a ferramenta.

EMap possui duas telas principais: `login` e `consultas`. A tela de `login` é apresentada na Figura 3, onde o usuário informa o endereço, o nome da base de dados que ele deseja consultar, o usuário, a senha e em qual SGBD a base se encontra armazenada. Ainda na tela de `login`, há um *link* para um arquivo no formato PDF com a documentação da ferramenta. A tela de `consultas` é ilustrada na Figura 4, na qual o usuário deve digitar a consulta em TSQL2 na `Área 1` e em seguida clicar no botão `Executar`. Com isso, a ferramenta mapeia a consulta para SQL-92 e submete ao SGBD. Quando o SGBD retorna o conjunto-resposta da consulta, a ferramenta exibe na `Área 2` a consulta equivalente em SQL-92 e apresenta os resultados na `Área 3`. A consulta em TSQL2 e sua equivalência em SQL, apresentadas na Figura 4, são as mesmas ilustradas na Figura 1.

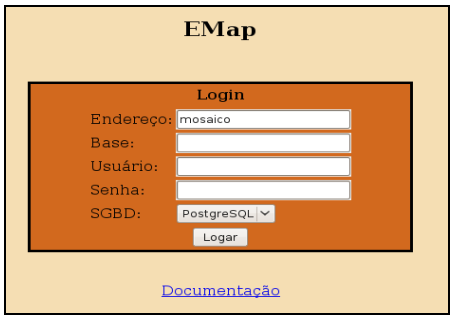


Figura 3. Tela de Login

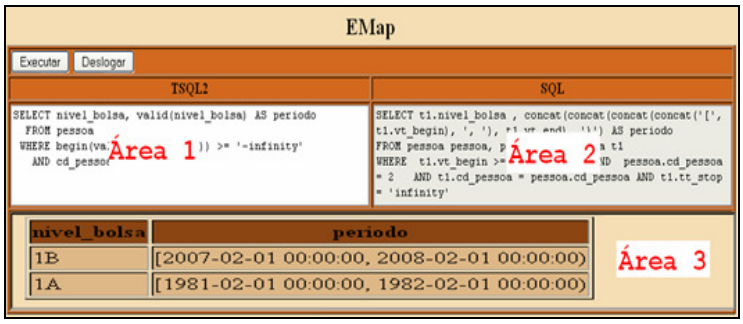


Figura 4. Tela de Consultas

3.2 Etapas do mapeamento

O processo de mapeamento é dividido em 3 etapas, conforme descrito a seguir.

- **Leitura do FROM**: armazena em um vetor o nome e *alias* de cada tabela do FROM;
- **Mapeamento SELECT:** o problema deste passo está em substituir apenas o que é temporal por SQL convencional, sem alterar a parte não temporal. São executados os seguintes passos: **(i)** mapear todas as funções temporais; e **(ii)** mapear todos os atributos temporais. Um exemplo simples, do mapeamento da função `(BEGIN(VALID(p.vl_salario)))` em TSQL2 para um atributo SQL-92 `(psal.vt_begin)` é:

TSQL2	***SQL-92***

`SELECT p.vl_salario,`
`  BEGIN(VALID(p.vl_salario))`
`FROM pessoa p`
	`SELECT psal.vl_salario, psal.vt_begin FROM`
	`pessoa p, pessoa_salario psal WHERE`
	`p.cd pessoa = psal.cd pessoa`

- **Mapeamento WHERE:** para cada condição do WHERE deve-se executar os seguintes passos: **(i)** mapear todos os operadores temporais; **(ii)** mapear todas as funções temporais; e **(iii)** mapear todos os atributos temporais. Um exemplo do mapeamento de um operador temporal (INTERSECT) para SQL-92 é:

TSQL2	***SQL-92***

`SELECT <qualquer_coisa>`
`FROM pessoa pesj, pessoa pesm`
`WHERE VALID(pesj.vl_salario)`
`  INTERSECT`
`  VALID(pesm.vl_salario)`
	`SELECT <qualquer_coisa>`
	`FROM pessoa pesj, pessoa pesm,`
	`  pessoa_salario psalj, pessoa_salario psalm`
	`WHERE psalj.cd_pessoa = pesj.cd_pessoa AND`
	`  psalm.cd_pessoa = pesm.cd_pessoa AND`
	`  psalj.vt_begin < psalm.vt_end AND`
	`  psalj.vt_end > psalm.vt_begin`

Para a realização do mapeamento é necessário acessar os metadados do Btpgsql que possuem as informações temporais armazenadas em tabelas auxiliares (estes metadados são especificados em Howard (2008)). Além das tabelas auxiliares do Btpgsql, a ferramenta *EMap* necessita de outras duas tabelas auxiliares e duas visões (Figura 5): **(i) particionamento_vertical**: tabela utilizada para identificar que conceitualmente um atributo temporal (`nm_atributo`) pertencia a uma tabela (`nm_tabela_origem`), porém foi particionado para outra (`nm_tabela_fisica`) por ser assíncrono aos demais atributos; **(ii) configuração**: tabela utilizada para definir o nome dos atributos que representam o tempo inicial e final de validade e de transação, bem como, os valores que representam as variáveis temporais "início da aplicação" e "tempo corrente a medida que ele corre"; **(iii) vw_tabela**: visão utilizada para identificar que um atributo pertence a uma determinada tabela, necessária quando acessado um atributo temporal sem informar o *alias* da tabela a qual ele pertence; **(iv) vw_referencia**: visão utilizada para identificar as chaves estrangeiras, necessária quando acessado um atributo que pertencia conceitualmente a uma tabela (tabela origem) e que foi particionado para outra (tabela física) por ser assíncrono aos demais atributos da tabela origem, pois neste caso é necessário fazer a junção da tabela física com a tabela origem. As informações presentes nas visões são obtidas dos metadados do próprio banco de dados, porém a ferramenta não deve acessar diretamente os metadados do banco de dados uma vez que cada banco armazena seus metadados de uma forma diferente. Estas visões garantem a independência de um SGBD específico e são detalhadas em Manica et al. (2009).

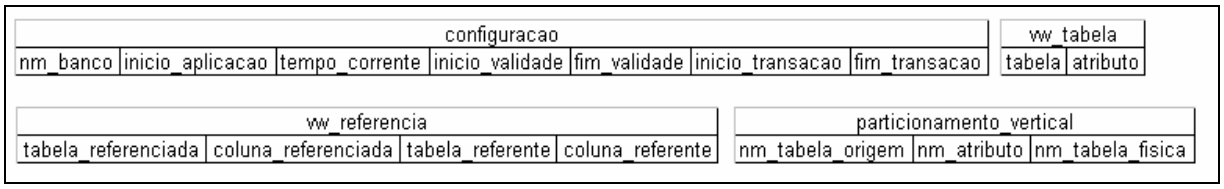
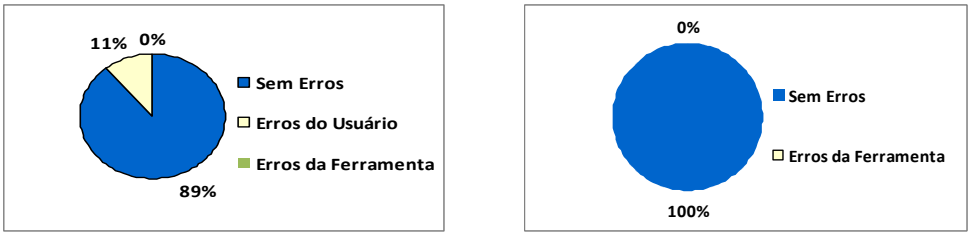


Figura 5. Tabelas auxiliares utilizadas pela ferramenta

4. Experimentos e Resultados

Para a realização dos testes utilizou-se parte de uma modelagem de currículo de pesquisadores. Esta modelagem possui 5 tabelas, sendo 3 tabelas bitemporais, 1 tabela com atributos temporais e não temporais e 1 tabela apenas com atributos não temporais. O banco de dados foi criado em dois SGBDs: um no PostgreSQL e outro no MySQL. As bases de dados foram populadas com 46 currículos de pesquisadores extraídos do Lattes (http://lattes.cnpq.br). Para a validação da ferramenta foram executadas 100 consultas por 4 alunos de graduação, sobre as bases de dados.

A Figura 6 ilustra os resultados obtidos a partir da execução, nas duas bases, das 100 consultas. A primeira execução das consultas gerou os resultados observados em **(a)**, onde se percebe que não ocorreram erros da ferramenta, porém foram constatados 11% de erros do usuário. Para uma melhor avaliação da ferramenta os erros do usuário foram corrigidos. Estes erros eram na grande maioria erros de digitação, como, por exemplo, digitar INT**RE**SECT em vez de INT**ER**SECT. Depois de corrigidos os erros, as consultas foram novamente executadas nas duas bases através da ferramenta e o resultado é apresentado em (**b**), onde se observa que 100% das consultas foram executadas corretamente pela ferramenta.



(a) Com Erros do Usuário (b) Sem Erros do Usuário

Figura 6. Resultados dos Experimentos

O principal diferencial encontrado na implementação dos casos de uso nos dois SGBDs é o tratamento dos atributos de tempo. O PostgreSQL possibilita a definição dos valores "`-infinity`" e "`infinity`". Estes valores, no aspecto temporal, representam, respectivamente, o início da aplicação e o tempo corrente à medida que ele corre. Como o MySQL não permite esses valores, eles são mapeados para o menor e para maior valor permitido pelo tipo de dado (`` `0000-00-00' `` e `` `9999-12-31' ``, respectivamente). Com isso, no MySQL o usuário não pode usar o valor `` `9999-12-31' `` como data real, pois ele significa o tempo corrente à medida que ele corre, logo a transparência da implementação da temporalidade fica comprometida.

5. Trabalhos Relacionados

O trabalho de Zaupa (2002) apresenta uma forma de realizar consultas temporais usando o modelo TVM (*Temporal Versions Model*) em um banco de dados convencional (estudo de caso no DB2). Green e Johnson (2003) apresentam o protótipo de uma ferramenta denominada ProSQL para suporte ao desenvolvimento de extensões para SQL, voltada principalmente para extensões temporais. Carvalho (1997) apresenta o VQS TF-ORM, que é um ambiente de recuperação de informações temporais para o modelo TF-ORM (*Temporal Functionality in Objects With Roles Model*) em um banco de dados convencional (estudo de caso no Watcom).

As principais contribuições da ferramenta *EMap* estão relacionadas com o particionamento vertical (forma de implementação de relações temporais em bancos de dados relacionais definida por Elmasri e Navathe (2005)): (*i*) permite que os atributos temporais assíncronos sejam particionados para outras tabelas, agrupando em uma mesma tabela os atributos síncronos e mantendo a implementação das relações temporais transparente ao usuário. ProSQL permite que os atributos sejam particionados da mesma forma que a ferramenta *EMap*, porém a implementação das relações

temporais não fica transparente ao usuário, assim, ele deve ter conhecimento que determinados atributos foram particionados para outras tabelas. Zaupa (2002) e Carvalho (1997) particionam os atributos temporais em tabelas separadas, armazenando os atributos assíncronos em tabelas diferentes, porém os atributos síncronos também ficam em tabelas diferentes, replicando neste último caso, desnecessariamente, os atributos que representam o tempo. Tanto Zaupa (2002) quanto Carvalho (1997) mantêm implícita a implementação das relações temporais; (**ii**) implementa a junção de interseção temporal implícita por tempo de validade entre tabelas de mesma origem (tabelas físicas criadas a partir do particionamento de uma tabela origem) enquanto que nos trabalhos relacionados esta junção deve ser realizada de forma explícita; e (**iii**) implementa o mapeamento de forma independente do SGBD, enquanto que os trabalhos relacionados, apesar de definirem um modelo genérico de mapeamento, realizam uma implementação específica para um determinado SGBD.

6. Considerações Finais

Este artigo apresentou a ferramenta *EMap* que realiza o mapeamento de uma linguagem de consulta temporal (TSQL2) para SQL-92, visando permitir aos usuários a realização de consultas de mais alto nível sobre os aspectos temporais utilizando SGBDs relacionais convencionais. O objetivo é possibilitar consultas sobre os dados atuais e passados, e sobre os dados que eram considerados válidos em instantes passados, mesmo depois de terem sido alterados. Uma das principais contribuições deste trabalho está no fato de tornar a consulta transparente ao usuário. Outra contribuição importante é a implementação do operador INTERSECT temporal implícito por tempo de validade entre tabelas de mesma origem, bem como, o agrupamento de atributos síncronos em uma mesma tabela e a característica da implementação do mapeamento ser independente de um SGBD específico.

Atualmente está sendo elaborado um conjunto de testes de usabilidade a fim de comprovar que a transparência da implementação da temporalidade interfere positiva e significativamente na facilidade de criação das consultas temporais. Nos testes será verificado o tempo de realização de cada consulta em TSQL2 e em SQL-92, bem como o número de erros do usuário em cada tarefa usando cada uma das linguagens. Também, será realizado um questionário pós-teste, onde o usuário irá informar para cada tarefa qual das linguagens ele achou mais fácil e intuitiva. Espera-se, como resultado final, uma análise crítica da usabilidade da ferramenta a partir das hipóteses levantadas na primeira etapa do experimento.

Como trabalho futuro destaca-se a implementação de uma funcionalidade que gere o modelo conceitual temporal, a partir das tabelas relacionais do banco de dados convencional e dos metadados utilizados pelo mapeamento, a fim de auxiliar o usuário na criação das consultas.

Referências

Carvalho, T. P. Implementação de Consultas para um Modelo de Dados Temporal Orientado a Objetos. Dissertação (Mestrado em Ciência da Computação) – UFRGS, Porto Alegre, 1997.

Cervi, C. R.; Manica, E.; Dorneles, C. F.; Galante, R. M.. Mapeamento de um Modelo Lógico Temporal para um Modelo Físico de um SGBD. In: II Sessão de Pôsters - SBBD. Campinas, SP, 2008.

Edelweiss, N. Banco de Dados Temporais: Teoria e Prática. In: XVII Jornada de Atualização em Informática – JAI. XVII CSBC. Belo Horizonte 1998. p 225-282.

Elmasri, R.; Navathe, S. B. Sistemas de Banco de Dados. 4. ed. Pearson Education, 2005.

Green, J.; Johnson, R. ProSQL: A Prototyping Tool For SQL Temporal Language Extensions. In: James, A. E. et al., *BNCOD*, v.2712 of LNCS, pages 190–197. Springer, 2003.

Howard, A. Bi-temporal PostgreSQL Module 0.2.4. Disponível em: <http://raa.ruby-lang.org/project/btpgsql/>. Acesso em 25/05/2008.

Manica, E.; Cervi, C. R.; Dorneles, C. F.; Galante, R. M. Ferramenta para Suporte a Consultas Temporais em SGBDs Convencionais. In: V Escola Regional de Banco de Dados, 2009, Ijuí.

Snodgrass, R. (Ed.). The TSQL2 Temporal Query Language. Kluwer, 1995.

Zaupa, A. P. Suporte a Consultas no Ambiente Temporal de Versões. Dissertação(Mestrado em Ciência da Computação) – Universidade Federal do Rio Grande do Sul, Porto Alegre, 2002.