Implementação de Aspectos Temporais em SGBDs Livres

Eugênio de Oliveira Simonetto

Curso de Sistemas de Informação – Centro Universitário Franciscano (UNIFRA)
Andradas, 1614 – Centro - 97015-032 – Santa Maria – RS – Brasil

eosimonetto@unifra.br

Resumo - *O trabalho apresenta um estudo sobre a viabilidade de implementação de aspectos temporais, nos SGBD open-source PostgreSQL, Ingres, Firebird e MySql, tratando três modalidades diferentes: (1) tabela instantânea para BD de tempo de transação, (2) visão para BD de tempo de transação, e (3) tempo de validade. Foram consideradas as funcionalidades oferecidas por estes SGBD e o grau de transparência no tratamento dos dados para dois perfis de usuários: (a) os interessados nos aspectos temporais dos dados e (b) os que tratam os dados como em bancos de dados instantâneos. Como contribuição, é mostrado que, em MySql, muito pouco pode ser implementado, por enquanto, e em PostgreSQL podem ser implementadas características temporais, com total transparência na atualização dos aspectos temporais dos dados. Também é apresentada neste artigo uma ferramenta de apoio ao desenvolvimento de bancos de dados temporais em SGBDs livres.*

1. Introdução

A necessidade da representação das características temporais dos dados em bancos de dados não é recente. Diversos trabalhos têm sido desenvolvidos tanto no nível de modelagem, como no nível de implementação em banco de dados (Clifford, 1993; Snodgrass, 2000). No entanto, os SGBD dominantes do mercado atualmente, como Sybase e SQL Server, pouco oferecem nesse sentido. Por sua vez, o Oracle oferece um módulo específico, porém limitado, para tratamento de dados temporais, chamado *Time Series Cartridge*. E, por sua vez, tratando-se de bancos de dados livres nenhum SGBD, tais como, PostgreSQL, Ingres, Firebird e MySQL traz implementado, dentre suas funções, procedimentos para o desenvolvimento e a manipulação de aspectos temporais.

Por outro lado, o fato dos SGBDs livres pouco oferecerem em termos de funcionalidades para o tratamento adequado de informações com características temporais não significa que os sistemas de informação deixem de tratá-las. O que tem ocorrido é que, sendo necessário, a modelagem de uma aplicação representa essas características de maneira empírica já no nível conceitual.

O objetivo deste trabalho foi o de verificar a viabilidade de incorporar aspectos temporais no desenvolvimento de bancos de dados, considerando as atuais funcionalidades oferecidas pelos SGBDs *open-source*, bem como o de implementá-las nos SGBD utilizados para o desenvolvimento da pesquisa. Para a verificação da viabilidade e implementação foram utilizados os modelos propostos por Simonetto & Ruiz (2000) e Simonetto & Ruiz (2001), no caso dos bancos de dados de tempo de transação e, foram utilizadas as regras propostas por Hübler & Edelweiss (1999) para a implementação dos bancos de dados de tempo de validade.

O artigo está estruturado da seguinte forma: Na seção 2 são apresentadas as características básicas de um software livre, devido ao fato do estudo concentrar-se no desenvolvimento do aspecto tempo em SGBDs com as mesmas características, na seção 3 são apresentados os modelos de representação temporal a serem utilizados quando da verificação da viabilidade de

implementação dos mesmos nos SGBDs livres. Na seção 4, são apresentados os resultados da verificação proposta, na seção 5 é apresentada uma ferramenta de apoio ao desenvolvimento de bancos de dados temporais utilizando-se SGBDs livres. Por fim na seção 6, são apresentadas as considerações finais do estudo.

2. Software Livre

O conceito de software livre se refere à liberdade dos usuários executarem, copiarem, distribuírem, estudarem, modificarem e aperfeiçoarem o software. Mais precisamente, ele se refere a quatro liberdades, para os usuários: a liberdade de executar o programa, para qualquer propósito; a liberdade de estudar como o programa funciona, e adaptá-lo para as suas necessidades; a liberdade de redistribuir cópias de modo que você possa ajudar ao seu próximo e; a liberdade de aperfeiçoar o programa, e liberar os seus aperfeiçoamentos, de modo que toda a comunidade se beneficie.

Neste trabalho foram previamente estabelecidos os produtos livres a serem analisados, que são: PostgreSQL (Momjian, 2001), MySQL (MySQL, 2005), Ingres (Ingres, 2005) e Firebird (Firebird, 2005). A escolha de tais produtos deu-se pelo fato de os mesmos serem os mais utilizados, no que se refere ao armazenamento de dados em produtos de software livre.

3. Modelos Avaliados para Implementação Tempo nos SGBDs Livres

Para o desenvolvimento das aplicações utilizando características temporais nos produtos de bancos de dados livres, foram estabelecidas modalidades a serem verificadas quanto à possibilidade ou não de implementação utilizando o PostgreSQL, o MySQL, o Ingres e o Firebird. Para os bancos de dados de tempo de transação foram verificadas as modalidades Tabela Instantânea (Simonetto & Ruiz, 2000) e Visão (Simonetto & Ruiz, 2001), já para os bancos de dados de tempo de validade foi, primeiramente, desenvolvido um modelo de tempo de válido utilizando dois atributos adicionais representando início e o fim de validade dos dados. Também, no caso de tempo de validade, foi verificada a possibilidade de serem implementadas as regras propostas por (Hübler & Edelweiss, 1999).

Para todas as modalidades foram verificadas as três operações básicas de atualização em bancos de dados, quais sejam: a inserção (*insert*), a alteração (*update*) e a exclusão (*delete*) de dados. Para a representação dos aspectos temporais foi utilizado o conceito de intervalo de tempo (Snodgrass, 2000).

3.1 - Modalidade Tabela Instantânea

Nesta modalidade define-se cada relação somente com os atributos a serem representados no banco de dados, sem a presença dos atributos referentes ao aspecto tempo, a qual foi chamada *tabela instantânea*. Após a definição desta tabela, classifica-se seus atributos como estáticos (que não variam com o correr do tempo) e temporais (os quais variam com o correr do tempo), pois conforme esta classificação, o atributo será tratado de diferentes formas.
Para os atributos classificados como estáticos é definida uma só tabela, chamada *tabela estática*, a qual contém o atributo chave da tabela instantânea, todos os atributos estáticos e mais os atributos adicionais referentes ao tempo de transação *(I_TT* e *F_TT)*. Para cada atributo temporal da tabela instantânea, é definida uma nova tabela, composta do atributo chave da tabela instantânea, o atributo temporal em questão e mais os atributos adicionais referentes ao tempo de transação (*I_TT* e *F_TT)*.

O tipo de esquema relacional resultante do modelo é o seguinte:

```
Tabela_Instantanea(Chave1,..., Chaven, AtEstat1,..., AtEstatn, AtTemp1,..., AtTempn,
primary key (Chave1,..., Chaven));

Tabela_Estatica(Chave1,..., Chaven, AtEstat1,..., AtEstatn, I_TT, F_TT, primary key
(Chave1,..., Chaven, I_TT));

Tabela_Temp1(Chave1,..., Chaven, AtTemp1, I_TT, F_TT, primary key (Chave1,..., Chaven,
I_TT));
   .....

Tabela_Tempn(Chave1,..., Chaven, AtTempn, I_TT, F_TT, primary key (Chave1,..., Chaven,
I_TT));
```

Onde :

- `Chave1,..., Chaven` – são os atributos que fazem parte da chave primária da tabela instantânea;
- `AtEstat1, ... AtEstatn` - são atributos estáticos, que não variam com o correr do tempo;
- `AtTemp1, ..., AtTempn` - são atributos temporais;
- `I_TT` - é o atributo que representa o início de um intervalo temporal, ou seja, momento que a informação foi incluída na base de dados;
- `F_TT` - é o atributo que representa o fim de um intervalo temporal, ou seja, o momento que a informação deixa de ser válida para a base de dados.

Nas tabelas estática e temporal a chave primária é composta pelos atributos `Chave1,..., Chaven` e `I_TT`.

Para a implementação das operações de atualização (*insert, update* e *delete*) foi utilizado o conceito de *triggers.* Um exemplo da implementação da modalidade tabela instantânea é apresentado na tabela 1.

Tabela Instantânea	**Tabela Temporal NOME_ALUNO**	**INSERT na Tabela NOME_ALUNO**
create table aluno(cod_aluno integer not null, nome_aluno varchar(40), ender_aluno varchar(40) , email_aluno varchar(40), primary key (cod_aluno));	create table aluno_nome_aluno (cod_aluno integer not null , nome_aluno varchar(40) not null , i_tt timestamp without time zone DEFAULT now() not null , f_tt timestamp without time zone , primary key (cod_aluno , i_tt));	create function f_aluno_nome_aluno _insert () returns opaque as 'begin insert into aluno_nome_aluno values (new.cod_aluno , new.nome_aluno, now(), NULL); return new; end;' language 'plpgsql'; create trigger aluno_nome_aluno_trgi after insert on aluno for each row execute procedure f_aluno_nome_ aluno _insert ();

Tabela 1 – Exemplo da implementação da modalidade Tabela Instantânea

3.2 - Modalidade Visão

Para o desenvolvimento do segundo modelo de implementação de bancos de dados de tempo de transação foi utilizado o conceito de visões de banco de dados com a finalidade de garantir a transparência do modelo de dados, a manipulação automática do dado temporal e, também, de minorar a redundância de dados ocasionada pela modalidade tabele instantânea. O usuário do banco de dados teria o direito de acesso apenas às visões, onde toda e qualquer atualização teria de ser feita através destas.

Primeiramente, define-se a estrutura das relações que comporiam a resolução do problema da seguinte maneira: uma tabela estática, composta por todos os atributos que não variam com o

correr do tempo e mais a rotulação temporal e uma tabela para cada atributo temporal juntamente com os atributos referentes à rotulação temporal. O tipo de esquema relacional resultante é o seguinte:

```
Tabela_Estatica(Chave1,..., Chaven, AtEstat1, ... , AtEstatn, I_TT, F_TT, primary key
(Chave1,..., Chaven, I_TT));

Tabela_Temp1(Chave1,..., Chaven, AtTemp1, I_TT, F_TT, primary key (Chave1,..., Chaven,
I_TT));

.....

Tabela_Tempn(Chave1,..., Chaven, AtTempn, I_TT, F_TT, primary key (Chave1,..., Chaven,
I_TT));
```

E a visão correspondente ao esquema relacional:

```
Tabela_Visao(Chave1,...,Chaven, AtEstat1, ... , AtEstatn, AtTemp1, ... AtTempn);
```

A definição das tabelas na modalidade visão de Simonetto & Ruiz (2001), é feita tirando-se proveito da opção INSTEAD OF de declaração de regras ativas (Momjiam, 2000). Com isso, não existe fisicamente a tabela Instantânea (apresentada no modelo anterior), somente a tabela estática e as tabelas temporais, para cada atributo temporal. As notações utilizadas para as chaves e atributos foram as mesmas utilizadas na modalidade Tabela Instantânea. Para a implementação das operações de atualização (*insert, update* e *delete*) foi utilizado o conceito de *triggers.*

3.3 – Regras de Tempo de Validade

Quando uma operação de inserção (*insert*) é realizada, além do novo valor a ser inserido, o valor temporal correspondente ao tempo de validade inicial da tupla deve, também, ser fornecido, sendo armazenado no atributo referente à validade inicial da tupla (*I_TV)*. Um conjunto de regras foram propostas por Hübler & Edelweiss (1999) para análise do valor temporal, de forma a garantir a consistência temporal do banco de dados, porque duas tuplas diferentes, de um mesmo atributo, nunca poderão ser válidas ao mesmo tempo. Assim, estas regras devem ser satisfeitas: (i) o tempo de validade incial (*I_TV*) deve ser maior ou igual à data e; (ii) a validade inicial da tupla (*I_TV*) deve ser maior ou igual ao maior *I_TV* de todas as tuplas inseridas.

A operação de atualização (*update*) somente é executada após um conjunto de regras serem verificadas. Estas regras são: (i) somente o tempo de validade inicial pode ser atualizado; (ii) um novo tempo de validade inicial anterior ao atual não é aceito; (iii) tempo de validade pertencente ao passado não pode ser modificado e; (iv) somente a última tupla definida pode ser atualizada. A atualização do passado pode levar o banco de dados a um estado inconsistente.

A operação de remoção (*delete*) apresenta regras similares às desenvolvidas para a operação de atualização. As tuplas do passado e tuplas atualmente válidas não podem ser excluídas – pois, sendo um banco de dados temporal, todo o passado é preservado. A operação é substituída por um conjunto de regras que substituem a validade da tupla e alteram os rótulos correspondentes: o tempo de validade final da tupla é atualizado para a data atual. Todas as regras criadas para *insert, update* e *delete* foram implementadas utilizando-se *triggers*.

4. Verificação da Viabilidade de Implementação de Aspectos Temporais em SGBD Livres

Após definidos os tipos de modelos de implementação a serem utilizados foi executada a verificação propriamente dita. Os resultados de cada modalidade de implementação em relação ao SGBD analisado são apresentados na tabela 2, onde o símbolo ✓ representa a possibilidade de implementação do modelo no SGBD e X, em caso contrário.

	Modalidade Tabela Instantânea	**Modalidade Visão**	**Atributos de Tempo de Validade**	**Regras de Tempo de Validade**
PostgreSQL	✓	✓	✓	✓
MySQL	X	X	✓	X
Ingres	✓	X	✓	✓
Firebird	✓	X	✓	✓

Tabela 2 – Viabilidade de Implementação de Aspectos Temporais nos SGBDs Livres

Analisando-se os dados apresentados na tabela 2 constata-se que todas as modalidades de implementação são passíveis de representação utilizando-se o SGBD PostgreSQL para tal e, que as modalidades que necessitam de regras ativas (*triggers*) para serem implementadas ainda não são possíveis de serem desenvolvidas utilizando o MySQL, devido ao fato de o mesmo não possuir *triggers* dentre as suas funções. Constata-se, também, que o Ingres e o Firebird implementam a maioria das modalidades com exceção da modalidade visão. Ambos não implementam esta modalidade devido ao fato de não possuírem a opção INSTEAD OF na declaração de regras ativas. Na próxima seção é apresentada uma ferramenta de apoio ao desenvolvimento de bancos de dados temporais para os SGBDs PostgreSQL, Firebird e Ingres. Porém, cabe ressaltar que a mesma implementa as modalidades tabela instantânea e atributos de tempo de validade. Estão sendo desenvolvidas as regras de tempo de validade para todos estes SGBDs e a implementação da modalidade visão para o PostgreSQL.

5. A Ferramenta Desenvolvida

Para a especificação da ferramenta de apoio ao desenvolvimento do modelo de dados temporais utilizando bancos de dados livres, procuramos identificar as diferentes etapas que constituem o processo de desenvolvimento do modelo temporal e, que poderiam vir a ser apoiadas por algum tipo de ferramenta computacional. Foram identificadas as seguintes etapas:

1. *Análise do Esquema de Dados Inicial;*
2. *Classificação dos Atributos;*
3. *Geração das Estruturas Componentes do Modelo***.**

Após a execução destas etapas é obtido o *script* final, conforme o tipo de banco de dados temporal (tempo de transação e/ou tempo de validade), para ser submetido ao SGBD PostgreSQL, Ingres ou Firebird. Este *script* é composto por todos os gatilhos e tabelas derivados do esquema inicial do banco de dados. Das três etapas, a única que há a interação com o usuário é a de classificação dos atributos, onde estes são classificados como temporais ou estáticos, conforme o ponto de vista (ou a necessidade) do projetista do sistema. Tempo de

Transação foi implementado em conformidade com o modelo proposto por Simonetto (2000), e tempo de validade foi implementado conforme Snodgrass (2000).

5.1 - Análise do Esquema de Dados Inicial

A primeira etapa envolve o reconhecimento do banco de dados a ser submetido à ferramenta, bem como para identificar para qual SGBD deve ser gerado o *script* do banco de dados. Na *ClockTool*, a interface para seleção do SGBD e da localização do arquivo que contém o *script* original é apresentada na figura 5.1. Também nesta etapa, são identificadas todas as tabelas componentes do banco de dados e, dentro de cada tabela todos os seus atributos, bem como sua chave primária *(primary key)*. Esta etapa tem por finalidade preparar os atributos para sua posterior classificação.

O esquema do banco de dados a ser reconhecido pela ferramenta, deve ser fornecido através de um arquivo *script*, gerado por uma ferramenta CASE ou pelo projetista do sistema. É dada, na interface inicial (figura 5.1), a opção de o usuário criar seu *script* original a partir da própria *ClockTool*.

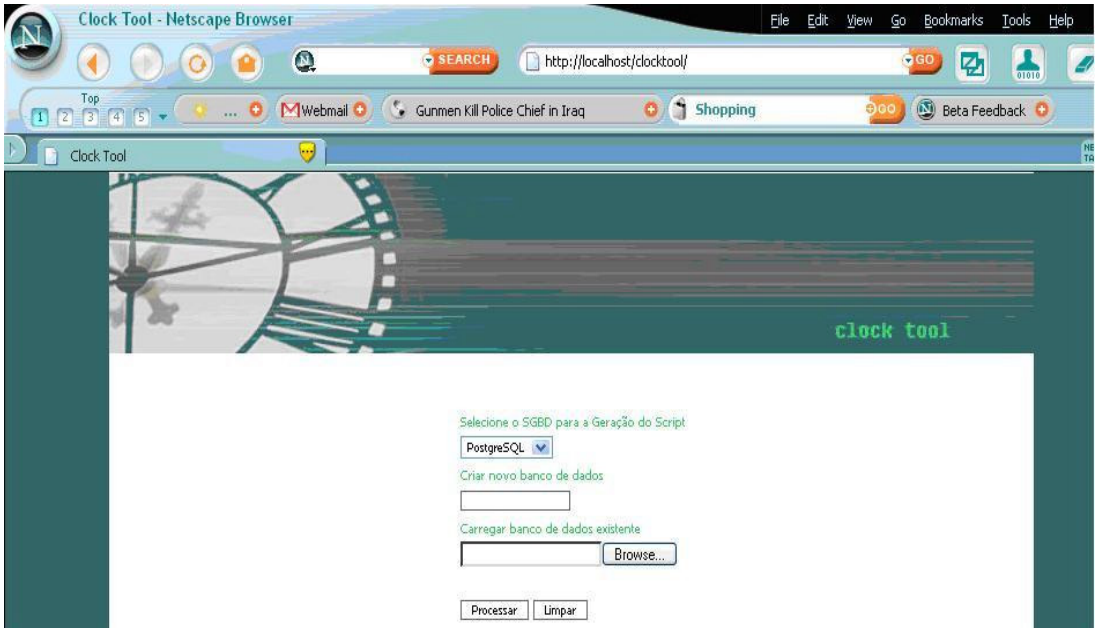

Figura 5.1 – Interface da *ClockTool* para a seleção do SGBD e do *script* original.

5.2 - Classificação dos Atributos

Nesta etapa, o projetista deve classificar cada atributo dentro de cada tabela como estáticos ou temporais. Nesta etapa, também, o projetista define o tipo de banco de dados temporal a ser gerado pela ferramenta: tempo de transação e/ou tempo de validade (Snodgrass, 2000). Quando o tipo de banco de dados a ser gerado for bitemporal, ambos os *checkbox* deverão ser selecionados. A interface de classificação dos atributos e do tipo de banco de dados temporal pode ser visualizada nas figuras 5.2 e 5.3.

Na interface para classificação dos atributos, a ferramenta disponibiliza ao projetista do sistema todos os atributos de cada tabela componente do sistema, para estes serem submetidos à classificação. Após a classificação dos atributos, é dada ao usuário a opção **Gerar Script**, a qual disponibilizará ao usuário o novo *script* (produto final da ferramenta), para ser submetido ao SGBD. Após a classificação dos atributos por parte do projetista do sistema se dá à parte

de criação das estruturas. A interface da ferramenta para todo o processo de classificação dos atributos, tipo de banco de dados temporal a ser gerado e geração de *script,* é apresentada na figura 5.4.

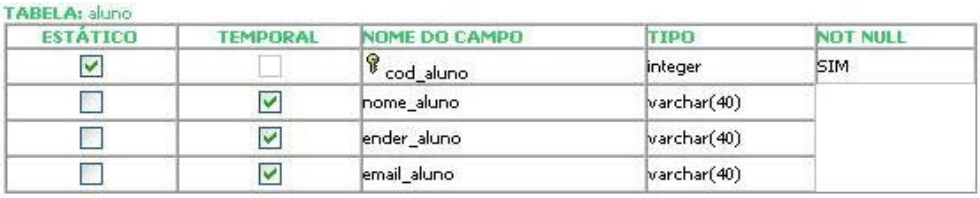

Figura 5.2 – Interface para a classificação dos atributos

Figura 5.3 – Interface para a seleção do tipo de banco de dados a ser gerado

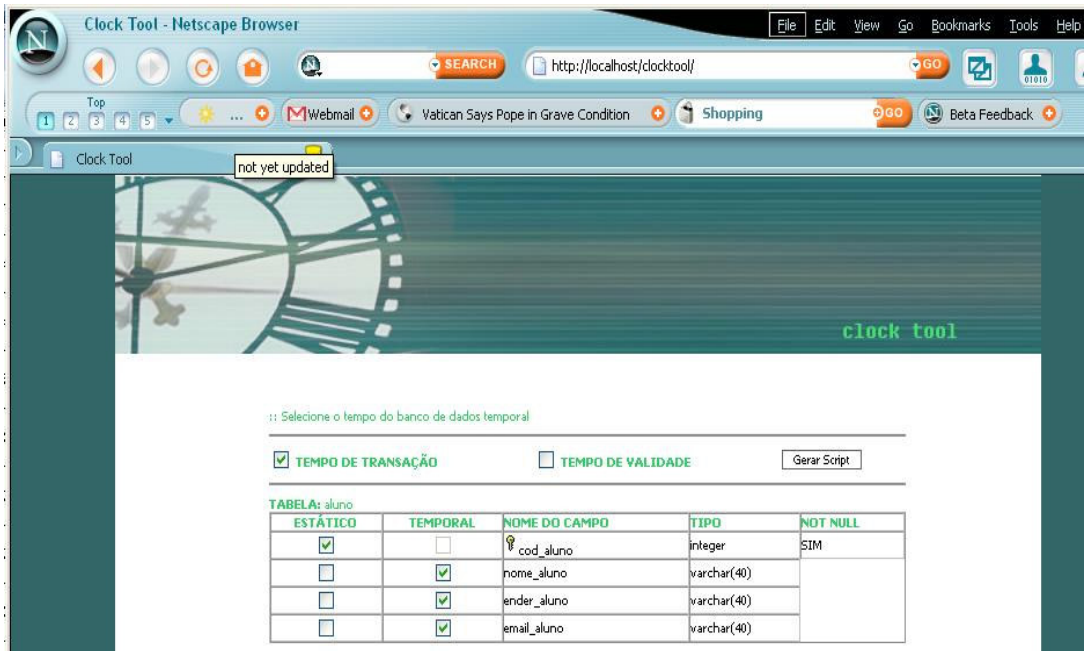

Figura 5.4 – Interface da *ClockTool* para as classificações e geração do *script*

5.3 - Geração das Estruturas Componentes do Modelo

Nesta etapa de funcionamento da ferramenta, são construídas as estruturas que garantem a manutenção e o armazenamento automático do dado temporal. Conforme a classificação para os atributos de cada tabela, executada na etapa anterior, são determinadas as estruturas correspondentes. As novas estruturas implicam na criação de *triggers*, procedimentos e novas tabelas, na forma de um arquivo *script*.

Para cada tabela do modelo, a primeira estrutura a ser gerada pela ferramenta é a própria relação definida pelo usuário que, no modelo proposto por Simonetto (2000), chama-se *tabela instantânea*. Para cada atributo classificado como temporal, é gerada uma *nova tabela* e *um*

conjunto de rotinas de manutenção do dado temporal que manipule este tipo de dado a cada inserção, atualização e remoção.

Já para os atributos, de uma tabela, classificados como estáticos é definida *somente uma tabela*, bem como suas respectivas *rotinas de manutenção do dado temporal*, no caso de inserção e remoção (pois o atributo estático não varia com o correr do tempo) de algum valor de atributo. O modo como o novo *script* é apresentado pela ferramenta pode ser visualizado na figura 5.5.

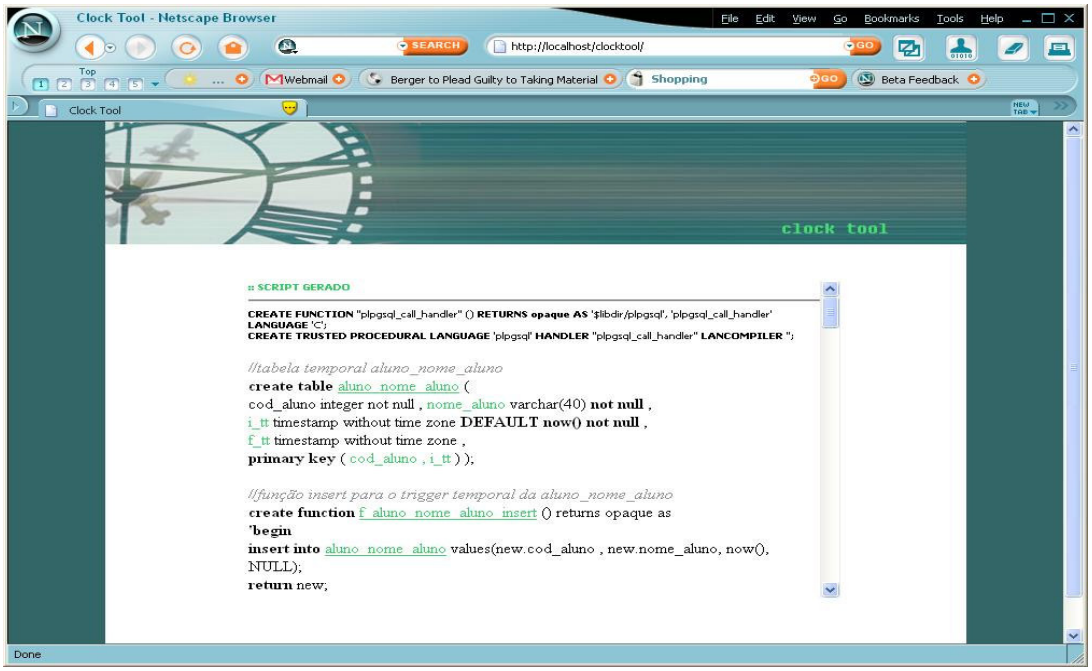

Figura 5.5 - Interface para Apresentação do *script* final pela *ClockTool*

5. Considerações Finais

O objetivo principal do artigo foi o de apresentar um estudo sobre a viabilidade de implementação de aspectos temporais nos SGBDs livres, no caso o PostgreSQL e o MySQL, utilizando modelos previamente definidos para tal. Após as análises verificou-se que o SGBD PostgreSQL implementou todos os modelos propostos, enquanto que com o MySQL é possível implementar apenas o tempo de validade sem as regras de validação do mesmo, ou seja, o único modelo que não necessita regras ativas (*triggers*) para a sua implementação. O Firebird e o Ingres somente não implementaram a modalidade visão.

Porém, o MySQL prevê para a sua versão 5.1 a inclusão de *triggers* dentre as suas funcionalidades (MySQL, 2005), o que garantiria, pelo menos, a implementação da modalidade tabela instantânea e das regras de tempo de validade, pois para a implementação da modalidade visão é necessário que as regras suportem a declaração INSTEAD OF.

Por fim, cabe ressaltar que os resultados obtidos utilizando-se a ferramenta apresentada foram considerados satisfatórios, visto que transformaram fielmente os *scripts* originais dos modelos de dados dos usuários, em modelos de dados temporais, de acordo com as classificações executadas.

Referências Bibliográficas

CLIFFORD, J.; COCKER, A. The Historical Relational Data Model (HRDM) revisited. In: ***Temporal Databases: Theory, Design and Implementation***. B. Cummings, 1993.

FIREBIRD. **Firebird-Relational Database for the New Millenium.** Disponível em firebird.sourceforge.net/. Acessado em 24/05/2005.

GUIA LIVRE. **Referência de Migração para Software Livre do Governo Federal.** Disponível em www.governoeletronico.gov.br. Acessado em 25/04/2005.

HÜBLER, P.N.; EDELWEISS, N.K.; CARVALHO, T.P. Implementação de um banco de dados temporal utilizando um SGBD convencional. In: XXV Conf. Latinoamericana de Informatica. **Anais do CLEI'99,** p.99-110.Asunción, Paraguai.1999.

INGRES. **Ingres Open Source – The SQL**. Disponível em www3.ca.com/Solutions. Acessado em 25/04/2005.

MOMJIAN, B. **PostgreSQL – Introduction and Concepts**. Addison-Wesley, 2001.

MySQL**. MySQL Homepage.** Disponível em www.mysql.com. Acessado em 25/04/2005.

SIMONETTO, E.O.; RUIZ,D.D. A Proposal Model to Incorporation of Temporal Aspects to the Relational DBMS Using Active Databases Concept. In: IEEE Fourth International Workshop on Databases and Information Systems, **Proceedings…** v1, p.52-59, Vilnius-Lithuania, 2000.

SIMONETTO,E. O., RUIZ, D.D.A. Using Views To Implement Time-Transactions Databases on Relational DBMS In: The 5th World Multi-Conference on Systemics, Cybernetics and Informatics, 2001, Orlando-Florida. **Proceedings of The 5th World Multi-Conference on Systemics, Cybernetics and Informatics** , 2001.

SNODGRASS, R.T. **Developing Time-Oriented Database Applications in SQL**. Morgan Kaufmann Publishers, 2000.