INTRODUÇÃO DO TEMPO NO AMBIENTE PARA DESENVOLVIMENTO DE BANCOS DE DADOS DEDUTIVOS

por

Nina Edelweiss

RP nº 114 Julho 1989

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
Av. Osvaldo Aranha, 99
90.210 - Porto Alegre - RS - BRASIL
Telefone: (0512) 21-84-99
Telex: (051) 2680 - CCUF BR

Correspondência: UFRGS-CPGCC
Caixa Postal 1501
90001 - Porto Alegre - RS - BRASIL

BANCO DE DADOS - SBU
Banco: Dados dedutivos
Tempo: Banco: Dados

ÍNDICE

RESUMO

Neste relatório são apresentadas as modificações efetuadas no **Ambiente para Desenvolvimento de Protótipos de Bancos de Dados Dedutivos**, descrito no RP 97 do CPGCC/UFRGS. Foi incluído o **tempo** no modelo de dados, resultando um **Banco de Dados Dedutivo Temporal**. É apresentado, ainda, um exemplo de utilização deste ambiente, na modelagem de uma agência de locação de fitas de vídeo.

ABSTRACT

This report presents the modifications that were done in the **Prototype Development Environment for Deductive Data Bases**, described in the CPGCC/UFRGS's 97º Technical Report. The **time** was included in the data model, being the result a **Temporal Deductive Data Base**. An example of the use of this environment in the modeling of a video-retal agency, is also presented.

1. INTRODUCÃO

Este trabalho descreve como foi feita a inclusão do **tempo** na Base de Dados do AMBIENTE PARA DESENVOLVIMENTO DE BANCOS DE DADOS DEDUTIVOS, descrito no RP 97 do CPGCC da UFRGS. O tempo deverá modelar a natureza dinâmica de uma parte do mundo real - a semântica temporal. A base de dados guardará as informações e, para cada informação armazenada, o instante de sua definição ou remoção. Desta forma, em qualquer momento poderá ser reconstituído todo o passado do Banco de Dados, que passa a ser um ***Banco de Dados Histórico*** [SNO86].

Com uso de um exemplo de aplicação, foi modelada a dinâmica de um sistema, através da definição de condições de violação de integridade da base de dados. São permitidas e efetivamente executadas quaisquer alterações solicitadas por um usuário. Se uma determinada alteração tiver como efeito a violação da integridade da base de dados, ela será desfeita. Deste modo, as condições de violação limitam as transições entre estados possíveis deste sistema.

O sistema que gerencia o ambiente aqui descrito, foi totalmente implementado, tendo sido utilizada a linguagem de programação em lógica PROLOG [STE86].

O texto está dividido em seis seções. Na seção 2 é feita a apresentação de algumas idéias quanto ao conceito de Bancos de Dados Temporais. A seção 3 apresenta como o **tempo** foi incluído no ambiente anteriormente implementado, e a seção 4 mostra as consequentes altera-

ções nas regras de violação das condições de integridade da base de dados. A seção 5 cita as alterações ocorridas na utilização do ambiente. Na seção 6 é modelado um exemplo através da utilização do ambiente, ficando a seção 7 para conclusões e colocação de algumas idéias para posteriores extensões do ambiente.

2. BANCOS DE DADOS TEMPORAIS

O armazenamento de informações em uma base de dados tem por finalidade a representação de um sub-conjunto do mundo real, de um modo o mais fiel e completo possível. O modelo utilizado para os dados deverá, portanto, possuir estas características. O **tempo** associado às informações é muito importante pois no mundo quase todos os eventos se relacionam de uma maneira temporal. Através da associação de informações ao tempo pode-se ter uma visão mais completa da realidade, estando representadas, inclusive, as propriedades dinâmicas destas informações.

Em [SNO86] foi descrita uma taxonomia de bases de dados, no que diz respeito ao tempo. São classificados em quatro classes, a saber:

a) **Bases de Dados "Snapshot"** : onde é representada uma instância da base de dados, não sendo armazenadas informações a respeito de estados anteriores;

b) **Bases de Dados "Rollback"** : onde são guardados todos os estados de uma base de dados, associando a cada um deles o tempo de ocorrência da transação que o originou;

c) **Bases de Dados Históricas** : quando o tempo é associado a cada informação e não a um estado completo, ficando todas as informações anteriores armazenadas;

d) **Bases de Dados Temporais** : nestas o tempo é associado como no caso anterior, sendo permitidas atualizações não só no momento presente como também em qualquer momento passado ou futuro.

Os Gerenciadores de Bases de Dados tradicionais não apresentam ferramentas para armazenar e processar aspectos temporais. A implementação de Bancos de Dados Históricos era inviável até pouco tempo, devido ao grande custo de armazenamento das informações. Como estes custos tem decrescido com o desenvolvimento de novas tecnologias, a idéia tem se tornado mais atraente. Isto é observado através do aumento de pesquisas nesta área nos últimos anos, principalmente nos aspectos de modelagem, linguagens de consulta e semântica temporal [CLI83]. Entretanto, segundo [AHN86], poucos tem sido os trabalhos de implementação efetiva de gerenciadores de bases de dados deste tipo.

A necessidade de inclusão do tempo associado às informações nas bases de dados foi reforçada com o crescimento do uso de Sistemas de Suporte à Decisão. Nestes, é essencial a capacidade de manipulação do tempo e de informações históricas, pricipalmente para o planejamento

empresarial, a investigação de relações causais e a análise retrospectiva [ARI86].

Na implementação de Bancos de Dados Temporais é interessante que a semântica temporal seja incorporada diretamente dentro do esquema conceitual do banco de dados e, por consequência, na linguagem de definição de dados associada, deixando o usuário livre desta tarefa. Deste modo pode-se garantir que o tempo seja tratado de maneira uniforme e consistente em todo o sistema.

3. TRATAMENTO DO TEMPO

Para o tratamento do tempo na base de dados foi utilizada a idéia colocada em [KOW86]. Foi assumido que todas as atualizações da base de dados (inclusão ou retirada de informações) seriam feitas na ordem temporal de sua ocorrência no mundo real. Não é possível, portanto, fazer alguma atualização referente ao passado. Isto possibilitou a utilização das funções embutidas no sistema operacional para fornecerem o **data** (dia, mes e ano) e o **horário** (hora, minuto e segundo) atuais, que são então associados a cadá modificação efetuada.

Segundo a taxonomia apresentada no item anterior, o tratamento adotado para o tempo caracteriza a base de dados como Histórica. Como, entretanto, existe a intenção de permitir a edição de dados já gravados em futuras extensões do sistema, resolveu-se chamar a base de dados de Temporal.

O tempo não foi incorporado diretamente aos dados, mas através da definição de fatos específicos. Isto não significa que o tempo esteja dissociado da informação: o sistema se encarrega de fazer a associação. A cada inclusão de uma informação à base de dados, será também incluído um fato associando os dados desta informação à data e ao horário daquele instante. A cada solicitação de retirada será incluído um fato semelhante. O efeito da operação de remoção será, portanto, não uma retirada mas o acréscimo de novos conhecimentos a respeito do fim do tempo de validade desta informação. A informação propriamente dita será incluída somente no momento de sua primeira definição, ficando presente a partir deste momento. O fato de ela estar, em algum momento posterior, definida ou não, será determinado pela informação correspondente aos momentos de definição ou remoção.

Para a manipulação das informações contidas na base de dados foi necessária a definição de atributos que indicassem a sua existência ou não no instante considerado. Cada referência a alguma informação deverá conter uma verificação de sua validade neste momento, o que é feito através da utilização destes atributos.

É importante notar que um usuário do sistema não tem acesso aos fatos que determinam a definição ou não de uma informação, sendo eles controlados exclusivamente pelo sistema.

O instante de definição de informações é representado na base de dados através dos seguintes fatos:

```
cRclasse(NomeClasse,Data,Horário).
cRentidade(Classe,NomeEntidade,Data,Horário).
cRrelacao(NomeRelacao,Data,Horário).
cRtupla(Relacao,Entidade1,Entidade2,Data,Horário).
```

A remoção de informações é representada pelos fatos:

```
dRclasse(NomeClasse,Data,Horário).
dRentidade(Classe,NomeEntidade,Data,Horário).
dRrelacao(NomeRelacao,Data,Horário).
dRtupla(Relacao,Entidade1,Entidade2,Data,Horário).
```

Quanto à implementação destes conceitos no ambiente anterior, apresentado em [EDE88], em alguns aspectos ocorreram poucas modificações enquanto em outros, estas foram mais marcantes. É importante notar, entretanto, que a introdução do tempo na base de dados não alterou a forma anterior de representação de informações, nem o modelo lógico utilizado para os dados. Na definição de dados apenas foram acrescentadas mais algumas informações.

A implementação foi substancialmente modificada na forma de remover informações, uma vez que estas não são mais removidas, somente tornadas indefinidas. A remoção resultou, portanto, no acréscimo e não na retirada de informações da base de dados.

As maiores mudanças ocorreram na formas de consultas oferecidas, com a inclusão do instante que deverá ser considerado e com o acréscimo de novas opções, conforme explicado no item 5. A forma em que as pesquisas são efetuadas leva em consideração a maneira como os dados são armazenados na base de dados. Os fatos que indicam a definição ou não das informações (cR... e dR...) são incluídos no início da base de dados e as informações propriamente ditas (classe..., entidade..., relacao... e tupla...) ficam no seu final. Ao fazer as pesquisas, portanto, os primeiros fatos recuperados, relativos a definições, serão os últimos a terem sido definidos, o mesmo acontecendo com os de remoções de informações.

4. RESTRIÇÕES DE INTEGRIDADE

Como descrito no relatório anterior, poderão ser incluídas restrições de integridade no Banco de Dados. Estas deverão ser escritas na forma de condições de violação - regras que, quando satisfeitas, violem a integridade da base de dados.

Existem diversos tipos de restrições de integridade. Segundo [WEB83], as que se relacionam a estados, transições e pré-condições, por exemplo, seriam as seguintes:

a) colocar restrições a valores de informações, restringindo os estados permitidos na base de dados (ex: salários de determinada qualificação devem ser superiores a um certo valor);

b) criar dependência entre diferentes informações através da colocação de pré-condições para uma transição, sendo esta pré-condição um valor fixo, externo à base de dados (ex: uma pessoa só pode ser contratada se nasceu antes de determinada data);

c) fazer restrições às transições da base de dados, restringindo as permitidas, restrições estas descritas em função de pré e pós-estados da base de dados (ex: salários não podem diminuir).

Nestes tres exemplos apresentados, os dois primeiros traduzem restrições de integridade estáticas, indicando que dados podem ser armazenados. A aplicação destas restrições exige somente a existência do estado atual da base de dados. O último, entretanto, representa uma restrição de integridade dinâmica, indicando como os dados podem ser atualizados, exigindo a existência de estados anteriores para sua aplicação..

No nosso caso, com a introdução do **tempo** associado a cada informação tornou-se possível escrever, além das restrições de integridade estáticas, presentes no ambiente anterior, também restrições de integridade dinâmicas. Estas comparam informações de instantes diferentes na base de dados, controlando portanto a dinâmica dos dados. Não só as restrições dinâmicas mas também as estáticas podem apresentar necessidade de informações associadas ao seu tempo de definição [CAS82].

No sistema que estamos apresentando as condições de violação deverão ser escritas em uma linguagem de lógica de 1º ordem, cujos ope-

radores são aqueles reconhecidos pelo *PROLOG*, e cujos símbolos predicativos são aqueles que guardam informações na base de dados (*classe*, *entidade*, *relacao*, *tupla*). A inclusão do tempo tornou necessária a verificação da validade de qualquer informação consultada na base de dados, o que deverá ser também efetuado nas condições de violação. Para facilitar esta verificação foram criados predicados específicos, que são os seguintes:

```
existe(Relacao,Entidade1,Entidade2,Tempo).
existe(Classe,Entidade,Tempo).
```

O primeiro verifica a existência de uma tupla no momento atual, em que é feita a verificação da integridade da base de dados. O segundo, a existência de uma entidade em uma classe. Em ambos, o Tempo indica o instante de definição da informação encontrada, englobando o dia e a hora desta definição. Deste modo, os instantes de definição de informações poderão ser comparados pelas condições de violação, além da verificação de sua existência no momento atual.

Para auxiliar a escrita destas condições de violação foram criados ainda outros predicados. O que compara dois diferentes instantes de definição de uma informação na base de dados, verificando se o primeiro é anterior ao segundo, é o seguinte:

```
anterior(Tempo1,Tempo2).
```

O intervalo de tempo (em Dias) entre duas datas diferentes poderá ser obtido através deste predicado:

```
intervalo(Tempo1,Tempo2,Dias).
```

Em todos eles o Tempo engloba a data e o horário da definição da informação.

Quando o Tempo não precisa ser recuperado, podem ser utilizados os seguintes dois predicados para a verificação da validade de uma tupla ou de uma entidade no momento atual:

```
definida(Relacao,Entidade1,Entidade2).
definida(Classe,Entidade).
```

Para auxiliar o entendimento de uma eventual violação, foi criado um predicado que possibilita a escrita de mensagens específicas. Estas mensagens deverão ser incluídas nas condições de violação, informando ao usuário do ambiente qual a condição que foi satisfeita (ou seja, qual a restrição de integridade que foi violada). Este predicado é o seguinte:

```
mensagem(' ..... ').
```

Outro predicados deverão ser criados para auxiliar ainda mais a escrita das condições de violação.

Como exemplo da escrita de uma restrição de integridade estática que define o salário de um engenheiro como sendo no mínimo NCz$600,00 temos a seguinte condição de violação:

```
viola :- tupla(profissao,Nome,engenheiro),
         definida(profissao,Nome,engenheiro),
         tupla(salario,Nome,Valor),
         definida(salario,Nome,Valor),
         Valor < 600.0,
         mensagem('SALARIO DE ENGENHEIRO MENOR QUE O MINIMO').
```

Uma restrição de integridade dinâmica, por exemplo, seria não permitir diminuição de salários, representada pela seguinte condição de violação:

```
viola :- tupla(salario,Nome,V1),
         existe(salario,Nome,V1,T1),
         tupla(salario,Nome,V2),
         existe(salario,Nome,V2,T2),
         anterior(T1,T2),
         V2 < V1,
         mensagem('SALARIO NAO PODE DIMINUIR').
```

5. UTILIZAÇÃO DO AMBIENTE

O ambiente permite a interação de um usuário com a base de dados através de um sistema de menus, ou através de uma interface de linguagem natural, ou ainda, através de um interpretador de programas de aplicação [EDE88]. A utilização do ambiente não foi alterada pelas modificações feitas no sistema para a inclusão das informações referentes ao tempo na base de dados. Foram, entretanto, acrescentadas mais algumas opções de consultas às informações da base de dados, relacionadas ao instante de tempo considerado. Além das consultas anteriores, relacionadas ao estado da base de dados no instante atual da consulta, foram implementadas as seguintes opções:

- todas as definições e remoções de uma determinada classe, ou de uma relação, ou de uma entidade, ou de uma tupla;
- todas as informações definidas no instante atual;
- todas as informações definidas na base de dados em um determinado instante passado, fornecido pelo usuário;

6. EXEMPLO DE UTILIZAÇÃO DO AMBIENTE

Como exemplo da utilização deste ambiente, vamos modelar um subconjunto das tarefas executadas em uma locadora de fitas de vídeo. Para esta modelagem utilizaremos o modelo T.A.& A., apresentado em [HOL88].

As tarefas a serem modeladas são o cadastramento de clientes e de fitas disponíveis, e uma parte da locação (retirada e devolução) de fitas.

A dinâmica dos dados é modelada através de restrições de integridade impostas à base de dados. Estas restrições são verificadas pelo sistema sempre que alguma operação de alteração do estado desta base de dados é efetuada. Deste modo, qualquer operação é permitida mas, se a mesma leva a base de dados a uma situação inválida, a operação é desfeita pelo próprio sistema. Só operações que resultem em um estado válido para a base de dados são efetivadas. Vemos, portanto, que a dinâmica dos dados é modelada através de verificações das propriedades estáticas do modelo.

6.1 CADASTRO DE CLIENTES

O cadastro de clientes será desdobrado em dois cadastros. O primeiro vai armazenar o nome e as informações a respeito de clientes que podem alocar fitas. O segundo vai guardar o nome de clientes indesejáveis, aos quais será vetada a entrada ho primeiro cadastro. Esta tarefa constitui-se, portanto, de duas atividades: a inclusão de nomes em cada um dos dois cadastros.

A atividade de inclusão do nome de um novo cliente no cadastro dos clientes da agência requer as seguintes ações:

a) verificação da pertinência do nome deste cliente no cadastro de clientes indesejáveis, o que bloquearia a sua inclusão no cadastro desejado;

b) verificação da pertinência do nome deste cliente no cadastro normal da agência, com a finalidade de impedir repetição de informações neste cadastro;

c) caso os itens anteriores indiquem que o cliente possa ser cadastrado, armazenar seu nome e informações a seu respeito (código, endereço, telefone, etc.).

Para incluir o nome de um cliente indesejável no cadastro de maus clientes, as ações a executar serão as seguintes:

a) verificação da pertinência do nome deste cliente no cadastro de clientes normais da agência; caso esta ocorrer, o nome deste cliente deverá ser retirado deste cadastro antes de sua inclusão no cadastro de mau cliente;

b) verificação da pertinência deste nome no cadastro de maus clientes, impedindo assim a duplicidade de informações neste cadastro;

c) inclusão do nome no cadastro desejado, caso o item (b) não seja satisfeito.

A modelagem destas informações no modelo de dados lógico utilizado neste ambiente requer que sejam criadas **classes** (simbólicas ou numéricas) para armazenar as entidades válidas e **relações** binárias entre entidades destas classes. Foram criadas as seguintes **classes**:

- **nome_cliente** (simbólica) : nomes dos clientes do primeiro cadastro (que podem alocar fitas);
- **mau_cliente** (simbólica) : nomes dos clientes do segundo cadastro (que não podem ser aceitos como clientes da locadora);
- **cod_cliente** (numérica inteira) : códigos dos clientes;
- **end_rua** (simbólica) : nomes de ruas;
- **end_nr** (numérica inteira) : número de endereços;
- **end_bairro** (simbólica) : bairros.

A associação de informações de um determinado cliente é formada através de relações que envolvem sempre o seu código. Este código deverá ser numérico e único para cada cliente. As **relações** e as duas classes relacionadas por cada uma delas são as seguintes:

- **cliente (cod_cliente , nome_cliente)**
- **rua (cod_cliente , end_rua)**
- **numero (cod_cliente , end_nr)**
- **apto (cod_cliente , end_nr)**
- **bairro (cod_cliente , end_bairro)**

6.2 CADASTRO DE FITAS

Na tarefa de cadastramento das fitas disponíveis para locação deverá ser armazenado o nome do filme que a fita contém e o tipo em que pode ser classificado (aventura, terror, infantil, etc.). Estas informações serão armazenadas nas seguintes classes:

- **nome_filme** (simbólica) : nome de filme;
- **tipo_filme** (simbólica) : tipo do filme;
- **cod_fita** (numérica inteira) : código de uma fita.

Cada fita terá um código próprio, o qual associa todas as informações a seu respeito. Como é possível que estejam disponíveis mais de uma cópia de um determinado filme, a cada uma delas corresponderá um código diferente pois o código está associado à fita e não ao filme.

As **relações** que associam o código de uma fita às informações a seu respeito são as seguintes:

- **fita (cod_fita , nome_filme)**
- **tipo (cod_fita , tipo_filme).'**

6.3 LOCAÇÃO E DEVOLUÇÃO DE FITAS

A **tarefa** de locação de fitas é a mais complexa deste exemplo apresentado. Ela é desdobrada em duas **atividades básicas**, a verifica-

ção de condições que permitam a locação e a locação propriamente dita.

Um pedido de locação de uma fita por uma determinada pessoa somente será atendido se as seguintes condições forem atendidas:

a) a pessoa for um cliente cadastrado no escritório;
b) a pessoa não estiver alugando alguma fita por um período superior a 30 dias;
c) a fita estiver cadastrada (existir esta fita neste escritório); e
d) a fita estiver disponível para aluguel.

As condições dos itens (a) e (c) acima mencionados poderão ser verificadas através de **ações** que pesquisem os cadastros antes explicados. A condição (b) poderá ser verificada através de uma pesquisa das fitas alocadas por este cliente. Uma pesquisa nas informações relativas a fitas alugadas indicará se a fita está ou não disponível (d).

Na efetivação de locação de uma fita deverão ser guardados o código da fita alugada e o código do cliente que estiver efetuando este aluguel, além de informações adicionais a respeito desta atividade de aluguel. Neste exemplo foi considerada somente a data de locação. Para esta última informação foi criada a seguinte classe:

- data (numérica inteira) : datas válidas (ddmmaa).

As informações a respeito de uma locação efetivada são armazenadas nas seguintes **relações**:

- **alugado (cod_fita , cod_cliente)**
- **data_loc (cod_fita , data).**

A **tarefa** de devolução de uma fita será executada através da remoção das informações relativas a esta locação. Ao remover a tupla correspondente de **alugado**, a fita estará novamente disponível para posterior locação.

6.4 CONDIÇÕES DE VIOLAÇÃO

A dinâmica dos dados nesta agência foi modelada através das condições de violação que representam a integridade da base de dados. As condições de violação são as seguintes:

a) incluir duas vezes o mesmo nome na classe **nome_cliente**;
b) incluir duas vezes o mesmo nome na classe **mau_cliente**;
c) incluir o nome de um novo cliente na classe **nome_cliente**, estando o nome deste cliente definido na classe **mau_cliente**;
d) incluir o nome de um cliente indesejável na classe **mau_cliente**, estando o nome deste cliente definido na classe **nome_cliente**;
e) incluir uma tupla na relação **cliente** associando um nome de um cliente a um código já existente em outra tupla desta mesma relação;

f) incluir uma tupla na relação **cliente**, associando um código a um nome já existente em outra tupla desta mesma relação;

g) incluir uma tupla na relação **rua**, associando uma rua a um código que não aparece em nenhuma tupla da relação **cliente**;

h) incluir uma tupla na relação **numero**, associando um número a um código que não aparece em nenhuma tupla da relação **cliente**;

i) incluir uma tupla na relação **apto**, associando o número de um apartamento a um código que não aparece em nenhuma tupla da relação **cliente**;

j) incluir uma tupla na relação **bairro**, associando o nome de um bairro a um código que não aparece em nenhuma tupla da relação **cliente**;

k) incluir uma tupla na relação **filme**, associando o nome de um filme a um código de fita já existente em outra tupla desta mesma relação;

l) incluir uma tupla na relação **tipo**, associando o tipo de um filme a um código de fita já existente em outra tupla desta mesma relação;

m) incluir uma tupla na relação **tipo**, associando o tipo de um filme a um código de fita que não aparece em nenhuma outra tupla da relação **filme**;

n) incluir uma tupla na relação **alugado**, associando o nome de um cliente ao código de uma fita que já está alugada, isto é, cujo código já aparece definido em alguma outra tupla desta relação;

o) incluir uma tupla na relação **alugado** quando existe alguma outra tupla em **alugado** com o código do mesmo cliente, tendo sido efetuada esta última locação em uma data anterior a 30 dias da atual;

p) incluir uma tupla na relação **data_loc**, associando a data de locação de uma fita ao código desta mesma fita, sendo que o código desta fita não aparece em nenhuma tupla da relação **alugado**;

q) remover uma entidade da classe **nome_cliente**, quando existir alguma locação em seu nome, ou seja, quando existir alguma tupla da relação **alugado** na qual aparece o código associado a este nome em alguma tupla de **cliente**;

r) remover alguma rua da classe **end_rua**, se esta estiver em alguma tupla da relação **rua**;

s) remover algum nome de bairro da classe **end_bairro**, se este estiver em alguma tupla da relação **bairro**;

t) remover o nome de um filme da classe **nome_filme**, se existir locação de alguma fita com este filme, isto é, se existir alguma tupla em **alugado** com o código de alguma fita que está associado a este nome de filme através de uma tupla da relação **fita**;

u) remover alguma entidade da classe **tipo_filme**, se este tipo estiver associado a algum filme através de uma tupla da relação **tipo**.

v) incluir dois nomes iguais na classe **end_rua**;

w) incluir dois nomes iguais na classe **end_bairro**;

x) incluir dois nomes iguais na classe **nome_filme**;

y) incluir dois tipos de filmes iguais na classe **tipo_filme**;

As verificações de unicidade de informações a respeito de nome de clientes definidos na classe **nome_cliente** e **mau_cliente** (a,b), de nomes de ruas na classe **rua** (v), de nomes de bairros na classe **bairro** (w), de nomes de filmes na classe **nome_filme** (x) e de tipos de filmes na classe **tipo_filme** (y) são efetuadas pelo próprio sistema que não permite duas entidades com nome igual em uma mesma classe.

No modelo utilizado é permitido que um mesmo cliente alugue mais de uma fita, em datas iguais ou diferentes, além de poder alugar a mesma fita mais de uma vez. É possível, também, que um mesmo filme esteja presente em mais de uma fita, associado a códigos de fita diferentes.

Como as tuplas de informações de endereço de um cliente foram amarradas na definição do código deste cliente (relação **cliente**) através das condições de violação (g) a (j), a retirada de um cliente do cadastro só poderá ser feita após a remoção de todas estas informações. O mesmo acontece no término de uma operação de locação de uma fita, quando deverá ser removido antes a informação correspondente à data de locação da fita, para depois remover a tupla correspondente de **alugado**. Deste modo, não ficará definida a informação de data de locação destà fita após a sua devolução.

As condições de violação mencionadas são traduzidas em regras escritas em uma linguagem de lógica de 1ª ordem que utiliza a representação lógica das informações [EDE88] e os predicados auxiliares citados no item 4. Por exemplo, as condições (c) e (e) seriam representadas pela seguinte regra:

```
viola :- entidade(nome_cliente,X),
         definida(nome_cliente,X),
         entidade(mau_cliente,X),
         definida(mau_cliente,X),
         mensagem('NOME JA CONSTA DE OUTRO CADASTRO.').
```

Já a condição (n), onde o tempo está claramente envolvido, seria representada por:

```
viola :- tupla(alugado,F1,Cliente),
         existe(alugado,F1,Cliente,T1),
         tupla(alugado,F2,Cliente),
         existe(alugado,F2,Cliente,T2),
         antes(T2,T1),
         intervalo(T2,T1,Dias),
         Dias)30,
         mensagem('ESTE CLIENTE ESTÁ COM FITA EM ATRASO.').
```

No Anexo 1 são apresentadas algumas listagens do exemplo apresentado. Durante a execução deste exemplo de aplicação são efetuadas algumas definições de informações, algumas remoções (operação de tornar indefinidas algumas informações armazenadas na base de dados), é apresentado o *menu* de consultas e são efetuadas algumas consultas. Para mostrar como as informações ficam armazenadas na base de dados são apresentados, no Anexo 2, os conteúdos dos arquivos correspondentes à base de dados e às condições de violação.

7. CONCLUSÃO

A inclusão do tempo no ambiente implementado tornou-o muito mais versátil, mais representativo da realidade que se quer modelar. Principalmente no que diz respeito à possibilidade de criar restrições de integridade dinâmicas, permitindo a representação da semântica dos dados. A implementação desta nova idéia no ambiente anterior foi bastante simples devido às facilidades apresentadas pela linguagem Prolog. Isto mostra que o sistema é de fácil extensão e manutenção.

O exemplo modelado mostrou ser o sistema de aplicabilidade para prototipação de casos reais, validando a intenção inicial quando do seu desenvolvimento. A metodologia T.A.& A. facilitou sobremaneira o modo de analizar o ambiente que se objetivou modelar.

Existem diversos pontos do sistema que ainda podem ser desenvolvidos. Entre eles, a criação da opção de edição de informações, opções de verificação sobre os arquivos de condições de violação e de regras da G.C.D., e a extensão da linguagem dos programas de aplicação.

Muito importante seria a criação de predicados mais abrangentes para facilitar a escrita das condições de violação, escondendo os testes de validade das informações. Como pode ser observado no exemplo do item 6, para um pequeno conjunto de tarefas resultou um número expressivo de condições de violação, escritas de uma maneira bastante complexa.

AGRADECIMENTO

Agradecemos ao Prof. José Mauro Volkmer Castilho pelas indicações a respeito do tratamento do tempo na base de dados e pela revisão deste trabalho.

ANEXO 1

LISTAGEM DE TELAS DO EXEMPLO DO ITEM 6

```
                    CRIACAO DE CLASSES
                    ==================

     Forneca o nome de cada classe que quer definir,
     seguido do tipo permitido para suas entidades.

     Tipos permitidos  : inteiro
                         real
                         simbolico
                         derivada

     Para terminar, digite fim como nome de classe.
                            ===

Classe : nome_cliente.
Tipo   : simbolico.
--) A classe foi incluida no B.D.

Classe : mau_cliente.
Tipo   : simbolico.
--) A classe foi incluida no B.D.

Classe : cod_cliente.
Tipo   : inteiro.
--) A classe foi incluida no B.D.

Classe : end_rua.
Tipo   : simbolico.
--) A classe foi incluida no B.D.

Classe : end_nr.
Tipo   : intgiro.
--) A classe foi incluida no B.D.

Classe : end_bairro.
Tipo   : simbolico.
--) A classe foi incluida no B.D.

Classe : nome_filme.
Tipo   : simbolico.
--) A classe foi incluida no B.D.

Classe : tipo_filme.
Tipo   : simbolico.
--) A classe foi incluida no B.D.

Classe : cod_fita.
Tipo   : inteiro.
--) A classe foi incluida no B.D.

Classe : data.
Tipo   : inteiro.
--) A classe foi incluida no B.D.

Classe : fim.

      ----) Fim do processo de criacao de classes.
            ======================================

            DIGITE QUALQUER TECLA PARA CONTINUAR
```

DEFINICAO DE RELACOES

Forneca o nome e o tipo de cada relacao, seguidos dos nomes
das duas classes que devera relacionar, separados por virgula.

Tipos de relacoes:

  total   : relacao basica que envolve todas as entidades das duas classes
  parcial : relacao basica, com tuplas definidas por extensao
  derivada : relacao derivada de outras relacoes

Para terminar, forneca fim como nome de relacao.

```
Nome da relacao : cliente.
Tipo da relacao : parcial.
Nome das classes a serem relacionados : cod_cliente,nome_cliente.
--) A relacao foi definida.

Nome da relacao : rua.
Tipo da relacao : parcial.
Nome das classes a serem relacionados : cod_cliente,end_rua.
--) A relacao foi definida.

Nome da relacao : numero.
Tipo da relacao : parcial.
Nome das classes a serem relacionados : cod_cliente,end_nr.
--) A relacao foi definida.

Nome da relacao : apto.
Tipo da relacao : parcial.
Nome das classes a serem relacionados : cod_cliente,end_nr.
--) A relacao foi definida.

Nome da relacao : bairro.
```

INSERCAO DE ENTIDADES

Forneca a classe na qual deseja inserir entidades: mau_cliente.
Forneca as entidades.

Para terminar : fim

alexandre.
--) Registrado.
fim.
---) Termino da insercao de entidades.

DIGITE QUALQUER TECLA PARA CONTINUAR

INSERCAO DE ENTIDADES

Forneca a classe na qual deseja inserir entidades: nome_cliente.
Forneca as entidades.

Para terminar : fim

anamaria.
--) Registrado.
berenice.
--) Registrado.
alexandre.

--) NOME JA CONSTA DO OUTRO CADASTRO.
--) A relacao nao foi incluida no B.D. pois violaria
suas restricoes de integridade.
fernando.
--) Registrado.
fim.
---) Termino da insercao de entidades.

DIGITE QUALQUER TECLA PARA CONTINUAR

INSERCAO DE ENTIDADES

Forneca a classe na qual deseja inserir entidades: nome_filme.
Forneca as entidades.

Para terminar : fim

rain_man.
--> Registrado.
roger_rabbit.
--> Registrado.
secretaria_do_futuro.
--> Registrado.
fim.
---> Termino da insercao de entidades.

DIGITE QUALQUER TECLA PARA CONTINUAR

INSERCAO DE ENTIDADES

Forneca a classe na qual deseja inserir entidades: tipo_filme.
Forneca as entidades.

Para terminar : fim

drama.
--> Registrado.
comedia.
--> Registrado.
guerra.
--> Registrado.
desenho.
--> Registrado.
fim.
---> Termino da insercao de entidades.

DIGITE QUALQUER TECLA PARA CONTINUAR

INSERCAO DE TUPLAS

Forneca o nome da relacao na qual vai inserir tuplas : cliente.
Entre com as tuplas, separadas por virgula.

Classes relacionadas : cod_cliente e nome_cliente

Para terminar : fim.

123,anamaria.
--) A tupla foi definida.
124,berenice.
--) A tupla foi definida.
125,fernando.
--) A tupla foi definida.
126,alexandre.
--) A segunda entidade nao esta definida.
fim.

---) Termino da insercao de tuplas.

DIGITE QUALQUER TECLA PARA CONTINUAR

INSERCAO DE TUPLAS

Forneca o nome da relacao na qual vai inserir tuplas : fita.
Entre com as tuplas, separadas por virgula.

Classes relacionadas : cod_fita e nome_filme

Para terminar : fim

33,rain_man.
--) A tupla foi definida.
34,rain_man.
--) A tupla foi definida.
35,roger_rabbit.
--) A tupla foi definida.
36,secretaria_do_futuro.
--) A tupla foi definida.
fim.

---) Termino da insercao de tuplas.

DIGITE QUALQUER TECLA PARA CONTINUAR

INSERCAO DE TUPLAS

Forneca o nome da relacao na qual vai inserir tuplas : tipo.
Entre com as tuplas, separadas por virgula.

Classes relacionadas : cod_fita e tipo_filme

Para terminar : fim

33,drama.
--) A tupla foi definida.
34,drama.
--) A tupla foi definida.
35,desenho.
--) A tupla foi definida.
36,comedia.
--) A tupla foi definida.
fim.

---) Termino da insercao de tuplas.

DIGITE QUALQUER TECLA PARA CONTINUAR

INSERCAO DE TUPLAS

Forneca o nome da relacao na qual vai inserir tuplas : alugado.
Entre com as tuplas, separadas por virgula.

Classes relacionadas : cod_fita e cod_cliente

Para terminar : fim

33,123.
--) A tupla foi definida.
33,124.

ESTA FITA JA ESTA ALUGADA PARA OUTRO CLIENTE.
--) A tupla nao foi incluida no B.D. pois violaria
suas restricoes de integridade.
34,123.
--) A tupla foi definida.
35,124.
--) A tupla foi definida.
fim.

---) Termino da insercao de tuplas.

DIGITE QUALQUER TECLA PARA CONTINUAR

REMOCAO DE TUPLAS

Forneca o nome da relacao da qual quer remover uma tupla : alugado.

Forneca a tupla : 33,123.

--) A tupla foi removida.

DIGITE QUALQUER TECLA PARA CONTINUAR

INSERCAO DE TUPLAS

Forneca o nome da relacao na qual vai inserir tuplas : alugado.
Entre com as tuplas, separadas por virgula.

Classes relacionadas : cod_fita e cod_cliente

Para terminar : fim

33,124.
--> A tupla foi definida.
fim.

---> Termino da insercao de tuplas.

DIGITE QUALQUER TECLA PARA CONTINUAR

REMOCAO DE TUPLAS

Forneca o nome da relacao da qual quer remover uma tupla : alugado.

Forneca a tupla : 33,124.

--> A tupla foi removida.

DIGITE QUALQUER TECLA PARA CONTINUAR

INSERCAO DE TUPLAS

Forneca o nome da relacao na qual vai inserir tuplas : alugado.
Entre com as tuplas, separadas por virgula.

Classes relacionadas : cod_fita e cod_cliente

Para terminar : fim

33,123.
--> Esta tupla ja tinha sido definida e foi removida.
Vai ser agora redefinida.
--> A tupla foi redefinida.
fim.

---> Termino da insercao de tuplas.

DIGITE QUALQUER TECLA PARA CONTINUAR

CONSULTAS AO BANCO DE DADOS

1 : quais as classes definidas neste instante
2 : quais as relacoes definidas neste instante
3 : quais as entidades de uma classe neste instante
4 : quais as tuplas de uma relacao neste instante
5 : qual o tipo associado ao nome de uma classe
6 : quais as classes associadas a uma relacao
7 : todas definicoes e remocoes de uma classe
8 : todas definicoes e remocoes de uma entidade
9 : todas definicoes e remocoes de uma relacao
10 : todas definicoes e remocoes de uma tupla
11 : todas as informacoes atuais do B.D.
12 : todas as informacoes do B.D. em determinado instante
13 : comparacao entre duas classes definidas
14 : comparacao entre duas relacoes definidas

CONSULTA A TODAS AS DEFINICOES E REMOCOES DE UMA TUPLA

Forneca o NOME DA RELACAO a qual esta tupla pertence : alugado.
Forneca agora a TUPLA (virgulas separando entidades) :33,123.

DEFINICOES -
DATA : 5/5/1989 HORARIO : 21:50:12
DATA : 5/5/1989 HORARIO : 21:35:19

REMOCOES -
DATA : 5/5/1989 HORARIO : 21:37:14

DIGITE QUALQUER TECLA PARA CONTINUAR

CONSULTA A TODAS AS INFORMACOES ATUAIS DA BASE DE DADOS

DATA ATUAL : 5 / 5 / 1989

CLASSES :

```
nome_cliente ( simbolico )
mau_cliente ( simbolico )
cod_cliente ( inteiro )
end_rua ( simbolico )
end_nr ( inteiro )
end_bairro ( simbolico )
nome_filme ( simbolico )
tipo_filme ( simbolico )
cod_fita ( inteiro )
data ( inteiro )
```

ENTIDADES :

```
alexandre ( classe mau_cliente )
anamaria ( classe nome_cliente )
berenice ( classe nome_cliente )
fernando ( classe nome_cliente )
rain_man ( classe nome_filme )
roger_rabbit ( classe nome_filme )
secretaria_do_futuro ( classe nome_filme )
drama ( classe tipo_filme )
comedia ( classe tipo_filme )
guerra ( classe tipo_filme )
desenho ( classe tipo_filme )
```

RELACOES :

```
cliente ( tipo parcial )
rua ( tipo parcial )
numero ( tipo parcial )
apto ( tipo parcial )
bairro ( tipo parcial )
fita ( tipo parcial )
tipo ( tipo parcial )
alugado ( tipo parcial )
data_loc ( tipo parcial )
```

TUPLAS :

```
123,anamaria ( relacao cliente )
124,berenice ( relacao cliente )
125,fernando ( relacao cliente )
33,rain_man ( relacao fita )
34,rain_man ( relacao fita )
35,roger_rabbit ( relacao fita )
36,secretaria_do_futuro ( relacao fita )
33,drama ( relacao tipo )
34,drama ( relacao tipo )
35,desenho ( relacao tipo )
36,comedia ( relacao tipo )
33,123 ( relacao alugado )
34,123 ( relacao alugado )
35,124 ( relacao alugado )
```

DIGITE QUALQUER TECLA PARA CONTINUAR

ANEXO 2

LISTAGEM DE ARQUIVOS

```
copy

      /*                    ARQUIVO DE CONDICOES DE VIOLACAO                    *
/*                                                                              */
viola :- entidade(nome_cliente,X),definida(nome_cliente,X),
         entidade(mau_cliente,X), definida(mau_cliente,X),
         mensagem('--> NOME JA CONSTA DO OUTRO CADASTRO.').

viola :- tupla(cliente,Cod,N1),definida(cliente,Cod,N1),
         tupla(cliente,Cod,N2),definida(cliente,Cod,N2),
         N1\=N2,
         mensagem('ESTE CODIGO JA ESTA ASSOCIADO A OUTRO CLIENTE.').

viola :- tupla(cliente,C1,Nome),definida(cliente,C1,Nome),
         tupla(cliente,C2,Nome),definida(cliente,C2,Nome),
         C1\=C2,
         mensagem('ESTE NOME JA ESTA ASSOCIADO A OUTRO CODIGO.').

viola :- tupla(rua,Cod,R),definida(rua,Cod,R),
         not(tupla(cliente,Cod,N),definida(cliente,Cod,N)),
         mensagem('ESTE CODIGO NAO ESTA ASSOCIADO A NENHUM CLIENTE.').

viola :- tupla(numero,Cod,Nr),definida(numero,Cod,Nr),
         not(tupla(cliente,Cod,N),definida(cliente,Cod,N)),
         mensagem('ESTE CODIGO NAO ESTA ASSOCIADA A NENHUM CLIENTE.').

viola :- tupla(apto,Cod,A),definida(apto,Cod,A),
         not(tupla(cliente,Cod,N),definida(cliente,Cod,N)),
         mensagem('ESTE CODIGO NAO ESTA ASSOCIADO A NENHUM CLIENTE.').

viola :- tupla(bairro,Cod,B),definida(bairro,Cod,B),
         not(tupla(cliente,Cod,N),definida(cliente,Cod,N)),
         mensagem('ESTE CODIGO NAO ESTA ASSOCIADO A NENHUM CLIENTE.').

viola :- tupla(fita,Cod,N1),definida(fita,Cod,N1),
         tupla(fita,Cod,N2),definida(fita,Cod,N2),
         N1\=N2,
         mensagem('ESTE CODIGO JA ESTA ASSOCIADO A OUTRA FITA.').

viola :- tupla(tipo,Cod,TP1),definida(tipo,Cod,TP1),
         tupla(tipo,Cod,TP2),definida(tipo,Cod,TP2),
         TP1\=TP2,
         mensagem('ESTE CODIGO JA FOI ASSOCIADO A OUTRO TIPO.').
```

```
viola :- tupla(tipo,Cod,TP),definida(tipo,Cod,TP),
         not(tupla(filme,Cod,N),definida(filme,Cod,N)),
         mensagem('ESTE CODIGO NAO ESTA ASSOCIADO A NENHUMA FITA.').

viola :- tupla(alugado,Cod,N1),definida(alugado,Cod,N1),
         tupla(alugado,Cod,N2),definida(alugado,Cod,N2),
         N1\=N2,
         mensagem('ESTA FITA JA ESTA ALUGADA PARA OUTRO CLIENTE.').

viola :- tupla(data_loc,Cod,D1),definida(data_loc,Cod,D1),
         not(tupla(alugado,Cod,N),definida(alugado,Cod,N)),
         mensagem('O CODIGO DESTA FITA NAO APARECE COMO ALUGADO.').

viola :- tupla(alugado,CF,CC),definida(alugado,CF,CC),
         tupla(cliente,CC,NC),definida(cliente,CC,NC),
         not entidade(nome_cliente,NC),
         mensagem('NOME DE CLIENTE DEVE ESTAR DEFINIDO.').

viola :- tupla(rua,CC,R),definida(rua,CC,R),
         not entidade(end_rua,R),
         mensagem('NOME DA RUA DEVE ESTAR DEFINIDO.').

viola :- tupla(bairro,CC,B),definida(bairro,CC,B),
         not entidade(end_bairro,B),
         mensagem('NOME DO BAIRRO DEVE ESTAR DEFINIDO.').

viola :- tupla(alugado,CF,CC),definida(alugado,CF,CC),
         tupla(fita,CF,NF),definida(fita,CF,NF),
         not entidade(nome_filme,NF),
         mensagem('O NOME DO FILME DEVE ESTAR DEFINIDO.').

viola :- tupla(tipo,CF,TF),definida(tipo,CF,TF),
         not entidade(tipo_filme,TF),
         mensagem('ESTE TIPO DEVE ESTAR DEFINIDO.').

viola :- tupla(alugado,F1,Cliente),existe(alugado,F1,Cliente,T1),
         tupla(alugado,F2,Cliente),existe(alugado,F2,Cliente,T2),
         antes(T2,T1),
         intervalo(T2,T1,Dias),
         Dias>30,
         mensagem('ESTE CLIENTE ESTA COM FITA EM ATRASO.').
```

```
classe(nome_cliente,simbolico).
classe(mau_cliente,simbolico).
classe(cod_cliente,inteiro).
classe(end_rua,simbolico).
classe(end_nr,inteiro).
classe(end_bairro,simbolico).
classe(nome_filme,simbolico).
classe(tipo_filme,simbolico).
classe(cod_fita,inteiro).
classe(data,inteiro).
cRclasse(data,5051989.0,211549.0).
cRclasse(cod_fita,5051989.0,211539.0).
cRclasse(tipo_filme,5051989.0,211527.0).
cRclasse(nome_filme,5051989.0,211516.0).
cRclasse(end_bairro,5051989.0,211500.0).
cRclasse(end_nr,5051989.0,211448.0).
cRclasse(end_rua,5051989.0,211427.0).
cRclasse(cod_cliente,5051989.0,211418.0).
cRclasse(mau_cliente,5051989.0,211406.0).
cRclasse(nome_cliente,5051989.0,211356.0).
entidade(cod_cliente,_00E5) :- not var(_00E5) , integer(_00E5).
entidade(end_nr,_00E5) :- not var(_00E5) , integer(_00E5).
entidade(cod_fita,_00E5) :- not var(_00E5) , integer(_00E5).
entidade(data,_00E5) :- not var(_00E5) , integer(_00E5).
entidade(mau_cliente,alexandre).
entidade(nome_cliente,anamaria).
entidade(nome_cliente,berenice).
entidade(nome_cliente,fernando).
entidade(nome_filme,rain_man).
entidade(nome_filme,roger_rabbit).
entidade(nome_filme,secretaria_do_futuro).
entidade(tipo_filme,drama).
entidade(tipo_filme,comedia).
entidade(tipo_filme,guerra).
entidade(tipo_filme,desenho).
cRentidade(tipo_filme,desenho,5051989.0,212829.0).
cRentidade(tipo_filme,guerra,5051989.0,212824.0).
cRentidade(tipo_filme,comedia,5051989.0,212820.0).
cRentidade(tipo_filme,drama,5051989.0,212817.0).
cRentidade(nome_filme,secretaria_do_futuro,5051989.0,212702.0).
cRentidade(nome_filme,roger_rabbit,5051989.0,212554.0).
cRentidade(nome_filme,rain_man,5051989.0,212505.0).
cRentidade(nome_cliente,fernando,5051989.0,212325.0).
cRentidade(nome_cliente,berenice,5051989.0,212310.0).
cRentidade(nome_cliente,anamaria,5051989.0,212302.0).
cRentidade(mau_cliente,alexandre,5051989.0,212221.0).
cRentidade(data,_00E5,5051989.0,211549.0).
cRentidade(cod_fita,_00E5,5051989.0,211539.0).
cRentidade(end_nr,_00E5,5051989.0,211448.0).
cRentidade(cod_cliente,_00E5,5051989.0,211418.0).
```

```
relacao(cliente,parcial,cod_cliente,nome_cliente).
relacao(rua,parcial,cod_cliente,end_rua).
relacao(numero,parcial,cod_cliente,end_nr).
relacao(apto,parcial,cod_cliente,end_nr).
relacao(bairro,parcial,cod_cliente,end_bairro).
relacao(fita,parcial,cod_fita,nome_filme).
relacao(tipo,parcial,cod_fita,tipo_filme).
relacao(alugado,parcial,cod_fita,cod_cliente).
relacao(data_loc,parcial,cod_fita,data).
cRrelacao(data_loc,5051989.0,212005.0).
cRrelacao(alugado,5051989.0,211943.0).
cRrelacao(tipo,5051989.0,211920.0).
cRrelacao(fita,5051989.0,211901.0).
cRrelacao(bairro,5051989.0,211833.0).
cRrelacao(apto,5051989.0,211808.0).
cRrelacao(numero,5051989.0,211748.0).
cRrelacao(rua,5051989.0,211722.0).
cRrelacao(cliente,5051989.0,211701.0).
tupla(cliente,123,anamaria).
tupla(cliente,124,berenice).
tupla(cliente,125,fernando).
tupla(fita,33,rain_man).
tupla(fita,34,rain_man).
tupla(fita,35,roger_rabbit).
tupla(fita,36,secretaria_do_futuro).
tupla(tipo,33,drama).
tupla(tipo,34,drama).
tupla(tipo,35,desenho).
tupla(tipo,36,comedia).
tupla(alugado,33,123).
tupla(alugado,34,123).
tupla(alugado,35,124).
tupla(alugado,33,124).
cRtupla(alugado,33,123,5051989.0,215012.0).
cRtupla(alugado,33,124,5051989.0,214036.0).
cRtupla(alugado,35,124,5051989.0,213622.0).
cRtupla(alugado,34,123,5051989.0,213559.0).
cRtupla(alugado,33,123,5051989.0,213519.0).
cRtupla(tipo,36,comedia,5051989.0,213400.0).
cRtupla(tipo,35,desenho,5051989.0,213349.0).
cRtupla(tipo,34,drama,5051989.0,213337.0).
cRtupla(tipo,33,drama,5051989.0,213329.0).
cRtupla(fita,36,secretaria_do_futuro,5051989.0,213229.0).
cRtupla(fita,35,roger_rabbit,5051989.0,213214.0).
cRtupla(fita,34,rain_man,5051989.0,213152.0).
cRtupla(fita,33,rain_man,5051989.0,213142.0).
cRtupla(cliente,125,fernando,5051989.0,213026.0).
cRtupla(cliente,124,berenice,5051989.0,213017.0).
cRtupla(cliente,123,anamaria,5051989.0,212958.0).
dRtupla(alugado,33,124,5051989.0,214421.0).
dRtupla(alugado,33,123,5051989.0,213714.0).
end_of_file     1 Arquivo(s) copiado(s)
```

REFERÊNCIAS BIBLIOGRÁFICAS

[AHN86] AHN,I. Towards an Implementation of Database Management Systems with Temporal Support. IEEE Transactions on Database Systems, 11(2):374-381, June 1896.

[ARI86] ARIAV,G. A Temporally Oriented Data Model. ACM Transactions on Database Systems, New York, 11(4):499-527, Dec. 1986.

[CAS82] CASTILHO,J.M.V.; CASANOVA,M.A.; FURTADO,A.L. A Temporal Framework for Database Specifications. Rio de Janeiro, PUCRJ, 1982. Relatório Técnico.

[CLI83] CLIFFORD,J. & WARREN,D.S. Formal Semantics for Time in Databases. ACM Transactions on Database Systems, New York, 8(2):214-154, June 1983.

[EDE88] EDELWEISS,N. & COSTA,A.C.R. Um Ambiente para Desenvolvimento de Protótipos de Bancos de Dados Dedutivos. Porto Alegre, CPGCC/UFRGS, 1988. RP nº 97.

[HOL88] HOPPEN,N.; OLIVEIRA,J.P.M.; LIMA,L.M.R. Metodologia para Modelagem de Escritório Baseada no Nível de Estruturação das Atividades. In: REUNIÃO ANUAL da ASSOCIAÇÃO NACIONAL de PROGRAMAS de PóS-GRADUAÇÃO em ADMINISTRAÇÃO, 12, Natal 26-28, set. 1988. Anais. Belo Horizonte, ANPAD, 1988. v.1 p.247-63.

[KOW86] KOWALSKI,R. & SERGOT,M. A Logic-Based Calculus of Events. New Generation Computing, Toquio, 4(1): 214-254, 1986.

[SNO86] SNODGRASS,R. & AHN,I. Temporal Databases. Computer, Los Alamitos, 19(9): 35-42, 1986.

[STE86] STERLING,L & SHAPIRO,E. The Art of Prolog = Advanced Programming Thechniques. Harvard, The MIT Press, 1986.

[WEB83] WEBER,W.; STUCKY,W.; KARSZT,J. Integrity Checking in Data Base Systems. Information Systems, Oxford, 8(2):125-136, 1983.