Comparação entre a Teoria da Revisão de Crença e a Teoria de Atualização de Banco de Dados

Thiago Sousa
thiago@linux.ime.usp.br

Departamento de Ciência da Computação
Universidade de São Paulo, Brasil

1 Introdução

A principal motivação dos estudos nas áreas de *Revisão de Crença* e de *Atualização de Banco de Dados* é a mesma. Dado um conjunto finito (pode ser infinito no caso da revisão de crença) de informações consistentes e uma nova informação para ser inserida nesse conjunto, é preciso garantir que o novo conjunto continue consistente mesmo que para isso seja necessário descartar algumas informações e todas as dependências relacionadas a elas. No presente trabalho iremos mostrar como a teoria da revisão de crença se relaciona com a teoria de atualização de banco de dados, abordando as propostas principais por trás de cada teoria, dando ênfase aos pontos em comum, sem deixar de ressaltar suas diferenças. Primeiramente serão detalhadas as idéias referentes às duas teorias, mostrando a semelhança entre uma proposta de *Revisão de Crença* e outra de *Atualização de Banco de Dados*. Em seguida iremos através de simulações mostrar a relação entre as duas idéias. Concluiremos o nosso trabalho mostrando que apesar de haver estudos na área de banco de dados que tratam o problema de atualização, os sistemas gerenciadores atuais não as utilizam.

2 A Atualização de Banco de Dados

Um dos problemas fundamentais em um banco de dados está relacionado à atualização, inserção e remoção de tuplas quando se é necessário satisfazer algumas restrições (chamadas mais comumente de restrições de integridade). Uma tupla inserida, atualizada ou removida pode deixar o sistema inconsistente.

Várias propostas para solucionar esse problema foram feitas. Uma delas, feita por Reiter [7], relaciona tal problema como pertencente a classe dos Problemas de Planejamento e utiliza-se da solução do Situation Calculus da área de Inteligência Artificial. Uma proposta de Katsuno e Mendelzon [2] diferencia atualização de revisão e mostra que os banco de dados se enquadram no modelo das atualizações. Atualizações consistem em trazer para a base de conhecimento o que há de novo quando o mundo descrito sofre mudanças, enquanto a revisão é usada quando obtemos uma nova informação sobre um mundo estático. Por exemplo, quando se diz que o salário de uma pessoa sofre um aumento de 10%, trata-se de uma atualização. Quando tentamos diagnosticar uma falha em um circuito e queremos incorporar na base de conhecimento os resultados de sucessivos testes, onde os novos resultados podem contrastar com as expectativas, trata-se do caso de se fazer uma revisão.

Já Revesz [5] propõe um novo operador chamado Arbitragem para a solução, onde a informação que se deseja inserir ao conjunto não é melhor e nem pior que as demais e mostra que os três operadores(revisão, atualização e arbitragem) se complementam, já que dependendo do problema de atualização um dos três operadores é melhor indicado. Winslett [8] enfatiza que um banco de dados deva ser tratado com uma semântica baseada em modelos e dentre estes devemos escolher o que melhor satisfaz as restrições de integridade.

Mas a abordagem que iremos melhor detalhar será a de Fagin et al. [6] que é bastante semelhante a idéia do *Epistemic Entrechment* da revisão de crença, que será relatada posteriormente. Como solução, Fagin et al. propõe que a base de dados seja tratada como uma teoria (conjunto consistente de proposições baseados em lógica de primeira ordem). O que se deseja é que todo tipo de inserção, remoção ou atualização provoque uma mudança mínima nessa teoria. Mas pode ocorrer que mais de uma teoria atenda a essa mudança mínima, tornando a escolha mais complicada, na medida que se pode escolher uma teoria que viole alguma restrição de integridade. Para resolver tal problema, é necessário estabelecer prioridades para cada sentença da base de dados. Para implementar tal ranking de sentenças, Fagin at al idealizaram a *Tagged Sentence* (ou Sentença Rotúlada), um par <i,a>, onde

i seria um número natural que representasse a prioridade da proposição a no banco de dados. Quanto menor o valor de i, mais importante é a sentença para banco de dados. Sendo assim, todas as restrições de integridade teriam i = 0. As demais teriam um i maior que zero (ex: sentenças existenciais, i=1; tuplas sem variáveis, i=2). Com esse ranking de prioridades, sempre que o banco de dados é atualizado, as mudanças são feitas nas sentenças de pouca importância, preservando as restrições de integridade.

3 Simulando a Atualização de um Banco de Dados

Vista a teoria por trás da semântica da atualização de bancos de dados, iremos agora fazer uma pequena simulação utilizando a idéia de Fagin et al.. Utilizaremos mais adiante esse mesmo exemplo de dados para se fazer uma simulação com base nas idéias da revisão de crença. Peguemos um banco de dados relacional POD(*Professor, Orientado, Departamento*) com a dependência funcional Professor $\rightarrow$ Departamento e atualmente com os seguintes dados:

Professor	Orientado	Departamento
Renata	Thiago	Computação
Renata	Flávio	Computação
Polcino	Pedro	Matemática
Simonis	Leo	Estatística

Nesse caso temos uma restrição de integridade dada pela dependência funcional com todos os elementos distintos. Seja R o banco de dados com as *Tagged Sentences*. Temos que R^0 (conjunto de sentenças com i = 0):

{<0,$\forall y_1 ... y_5$(POD(y_1, y_2, y_3)&POD(y_1, y_4, y_5) $\rightarrow$ $y_3 = y_5$ >,<0,Renata $\neq$ Polcino>,...,<0,Computação $\neq$ Matemática>}.

R^1:

$R^0 \bigcup$

{<1,$\exists$x(POD(Renata,Thiago,x))>,
<1,$\exists$x(POD(Renata,Flávio,x))>,
<1,$\exists$x(POD(Polcino,Pedro,x))>,
<1,$\exists$x(POD(Simonis,Leo,x))>}.

R^2:

$R^1 \bigcup$

{<2,POD(Renata, Thiago, Computação>,
<2,POD(Renata, Flávio, Computação>,
<2,POD(Polcino, Pedro, Matemática>,
<2,POD(Simonis, Leo, Estatística>}.

Suponha que desejamos inserir em POD a sentença ∃x(POD(Renata, x,Filosofia)). Vamos agora analisar como o banco de dados faz essa inserção mínimamente. Primeiro olhamos se R^0 é consistente com a operação. Como é, vamos para o próximo passo. Vemos se R^1 também é consistente. Como é, verificamos a sentença rotúlada com um i maior. No caso, analisaremos se R^2 é consistente. Notamos que nem <2,POD(Renata,Thiago,Computação)> e nem <2,POD(Renata,Flávio,Computação)> são consistentes com a inserção desejada. Eliminamos então as duas sentenças rotúladas inconsistentes e acrescentamos a nova sentença com um i = 3. O banco de dados ficaria assim:

Professor	Orientado	Departamento
Renata	Thiago	Filosofia
Renata	Flávio	Filosofia
Polcino	Pedro	Matemática
Simonis	Leo	Estatística

4 A Revisão de Crença

A teoria da revisão de crença é focada nos estados epistêmicos (de conhecimento, crença) e nas mudanças ocorridas nesses estados, também conhecidas como mudanças de crença. Os tipos de mudanças mais importantes nessa teoria, que foram propostos por Alchourrón et al. [1], são a expansão, a revisão e a contração de um determinado estado. O principal desafio da revisão de crença é representar através de modelos lógicos essas três mudanças. Daremos aqui enfoque a revisão e a contração, já que a expansão nada mais é do que um acréscimo de uma proposição a um estado. Dois tipos de abordagens são apresentadas para a construção de funções que, dado um estado epistêmico e uma proposição como entrada, realizam as operações de revisão e contração, mantendo o estado consistente.

A primeira abordagem é a "Maxichoice", que é baseada na mínima alteração de um estado quando sujeito a mudanças, atendendo assim a um dos principais requisitos da revisão de crença. Essa abordagem funciona assim : dado um conjunto de crenças K e uma proposição a, o novo conjunto K' (K contraído em relação a proposição a) deve ser um dos subconjuntos maximais de K que não implica a. Uma outra abordagem para tentar implementar esses operadores (contração,revisão,expansão) é o *Epistemic Entrechment Ordering*, no qual são definidos os graus de prioridade das sentenças e quando é feita alguma operação, as proposições menos importantes são revisadas primeiramente.

Dois problemas surgem no *Epistemic Entrechment Ordering* no momento da implementação. O primeiro é que por trás dessa idéia é possivel ranquear um número infinito de sentenças, o que é inviável do ponto de vista computacional. Um outro problema é que *Epistemic Entrechment Ordering* não suporta uma função de revisão/contração iterativa, já que a ordenação é perdida logo após a primeira operação. Isso é muito ruim pois as aplicações atuais recebem uma grande quantidade de novas informações. Sendo assim, a implementação computacional se utiliza da idéia do *Finite Partial Epistemic Ranking*, que é como se fosse um *Epistemic Entrechment Ordering* com uma quantidade finita de sentenças, ranqueadas num intervalo de [0,1], onde as sentenças com ranking igual a zero são inconsistentes e as com ranking igual a um são tautologias. A função iterativa fica por conta do *Adjustment of Partial Epistemic Ranking*, onde o ranking das sentenças é reajustado a cada iteração. Com os algoritmos feitos por Williams [3] para representar essa função iterativa, é possivel implementar e simular de uma maneira razoável a revisão de crença com base na idéia de proposições ordenadas.

5 Simulando a Revisão de Crença

Iremos agora realizar uma simulação dos operadores da revisão de crença com a base de dados anterior e utilizando-se dos algoritmos desenvolvidos por Williams [4]. Seja o mesmo banco de dados R e com os seguintes graus de prioridades definidos para as sentenças :

$\{\forall y_1 ... y_5(\text{POD}(y_1, y_2, y_3)\&\text{POD}(y_1, y_4, y_5) \rightarrow y_3 = y_5)$,Renata $\neq$ Polcino,...,Computação $\neq$ Matemática$\} = 0.9$; (1)

$\{\exists$x(POD(Renata,Thiago,x))$\} = 0.3$; (2)

$\{\exists$x(POD(Renata,Flávio,x))$\} = 0.3$;

$\{\exists$x(POD(Polcino,Pedro,x))$\} = 0.3$;

$\{\exists$x(POD(Simonis,Leo,x))$\} = 0.3$;

{POD(Polcino, Pedro, Matemática)} = 0.1; (3)
{POD(Simonis, Leo, Estatística)} = 0.1;
{POD(Renata, Thiago, Computação)} = 0.1; (4)
{POD(Renata, Flávio, Computação)} = 0.1;

Gostariamos de inserir a sentença ∃x(POD(Renata,x,Filosofia)) com um grau de prioridade 0.4. Faremos uma simulação dessa inserção usando a função do *Adjustment of Partial Epistemic Ranking* descrita em forma algorítmica abaixo. Sem perda de semântica podemos eliminar os quantificadores das sentenças para efeitos práticos.

Temos a função *Adjustment of Partial Epistemic Ranking*:

$$R^*(a,i) = \begin{cases} R^-(a,i) & \text{se } i < \text{grau(R,}a\text{)} \\ ((\text{R}^-(\neg a,0))^+(a,i) & \text{c.c.} \end{cases}$$

onde,

$$R^-(a,i)(b) = \begin{cases} i & \text{se grau(R,}a\text{) = grau(R,}a\text{) e R}(b) > i \\ \text{R(b)} & \text{c.c.} \end{cases}$$

para todo $b \in \text{dom(R)}$, e

$$R^+(a,i)(b) = \begin{cases} \text{R}(b) & \text{se R}(b) > i \\ \text{i} & \text{se } a \equiv b \text{ ou R}(b) \leq i < \text{grau(R,)} \\ \text{grau(R,)} & \text{c.c.} \end{cases}$$

para todo $b \in \text{dom(R} \cup \{a\})$.

Deseja-se fazer R^*(POD(Renata,x,Filosofia),0.4). Para facilitar o entendimento da simulação, (1), (2), (3) e (4) substituirão as suas respectivas sentenças como referenciado acima. Seguindo o algoritmo, temos:

1)R^-(¬POD(Renata,x,Filosofia),0);

2)R^-(¬POD(Renata,x,Filosofia),0)(1)
→ (R^-(¬POD(Renata,x,Filosofia),0)(1))$^+$(POD(Renata,x,Filosofia),0.4)(1) = 0.9 pois R(1) > 0.4.

3)R^-(¬POD(Renata,x,Filosofia),0)(2)
→ (R^-(¬POD(Renata,x,Filosofia),0)(2))$^+$(POD(Renata,x,Filosofia),0.4)(2) = 0.4 já que $0.3 \leq 0.4 < 1$.

O mesmo vale para
POD(Renata, Flávio, x);
POD(Polcino, Pedro, x);
POD(Simonis, Leo, x);

4)R^{-}(¬POD(Renata,x,Filosofia),0)(3)
$\rightarrow$ (R^{-}(¬POD(Renata,x,Filosofia),0)(3))$^{+}$(POD(Renata,x,Filosofia),0.4)(3) = 0.4 pois $0.1 \leq 0.4 \leq 1$.

O mesmo vale para POD(Simonis, Leo, Estatística);

5)R^{-}(¬POD(Renata,x,Filosofia),0)(5)
$\rightarrow$ (R^{-}(¬POD(Renata,x,Filosofia),0)(5))$^{+}$(POD(Renata,x,Filosofia),0.4)(5) = 0 pois R(POD(Renata,x,Filosofia) $\rightarrow$ POD(Renata,Thiago,Computação)) = 0.

O mesmo vale para POD(Renata,Flávio,Computação);

O banco de dados ficaria com a seguinte ordenação :

R*($POD(Renata, x, Filosofia), 0.4)(POD(y_1, y_2, y_3)\&POD(y_1, y_4, y_5) \rightarrow y_3 = y_5, Renata \neq Polcino >, ..., Computação \neq Matemática) = 0.9$;
R*($POD(Renata, x, Filosofia), 0.4)(POD(Renata, Thiago, x)) = 0.4$;
R*($POD(Renata, x, Filosofia), 0.4)(POD(Renata, Flávio, x)) = 0.4$;
R*($POD(Renata, x, Filosofia), 0.4)(POD(Polcino, Pedro, x)) = 0.4$;
R*($POD(Renata, x, Filosofia), 0.4)(POD(Simonis, Leo, x)) = 0.4$;
R*($POD(Renata, x, Filosofia), 0.4)(POD(Polcino, Pedro, Matemática)) = 0.4$;
R*($POD(Renata, x, Filosofia), 0.4)(POD(Simonis, Leo, Estatística)) = 0.4$;
grau(R*($POD(Renata, x, Filosofia), 0.4)(Renata, Thiago, Computação)) = 0$;
grau(R*($POD(Renata, x, Filosofia), 0.4)(Renata, Flávio, Computação)) = 0$;

6 Os Bancos de Dados na Prática

Apesar das várias propostas na literatura para o tratamento do problema de atualização(Katsuno e Mendelzon; Winslett; Fagin, Ullman e Vardi; Reiter; Revesz, etc), os sistemas gerenciadores de banco de dados atuais(SGBDs) se utilizam do modelo relacional e da sintaxe do SQL para realizar as operações de inserção, deleção e atualização na base de dados. Vejamos um exemplo prático de como o SQL "trabalha" com os dados e como as inconsistências são tratadas. Abaixo, descrevemos um mini banco de dados no modelo relacional sem nos preocuparmos com a eficiência e normalização :

Professor	Orientado	Departamento
Renata	Thiago	Computação
Renata	Flávio	Computacão
Polcino	Pedro	Matemática
Simonis	Leo	Estatística

Orientado	Bolsa
Thiago	CNPq
Flávio	CNPq
Pedro	Fapesp
Leo	CAPES

Professor	Endereço	Nascimento
Renata	Rua Ricardo de Abreu	12/12/1973
Polcino	Rua da Vida	03/08/1946
Simonis	Rua Cardoso de Almeida	26/09/1955

O esquema acima foi criado a partir do seguinte código SQL usando o gerenciador MySQL versão 3.23.41 no Linux Red Hat 7.2:

```
create table POD (
professor char(20),
orientado char(20),
departamento char(15),
foreign key (professor) references PEN on delete cascade
on update cascade,
foreign key (orientado) references OB on delete cascade
on update cascade);
```

```
create table OB (
orientado char(20) not null, bolsa char(10),
primary key (orientado));

create table PEN (
professor char(20) not null,
endereco char(30),
nascimento Date,
primary key (professor));
```

Criamos a seguinte asserção para verificar a dependência funcional de Professor → Departamento na tabela POD :

```
create assertion depfuncional
check (not exists (select * from POD as P
where not exists(select * from POD as D
where P.professor = D.professor
and P.departamento = D.departamento)));
```

Desejamos agora inserir em POD que a *professora* Renata pertence ao *departamento* de Filosofia. Uma das maneiras seria através do comando {insert into POD(professor,departamento) values ("Renata","Filosofia")}. Mas o gerenciador rejeitaria essa inserção porque violaria a dependência funcional de Professor → Departamento da tabela. O jeito correto de se fazer isso seria através do comando de atualização {update POD set departamento="Filosofia" where professor="Renata"}. A tabela POD ficaria assim:

Professor	Orientado	Departamento
Renata	Thiago	Filosofia
Renata	Flávio	Filosofia
Polcino	Pedro	Matemática
Simonis	Leo	Estatística

Suponha que verificamos que a professora Renata não pertence mais ao quadro de professores e portanto devemos remover todas as tuplas relacionadas ao seu nome na tabela PEN e POD. Para se fazer isso utilizamos o comando {delete from PEN where professor="Renata"}. As tabelas PEN e POD ficariam assim:

Professor	Endereço	Nascimento
Polcino	Rua da Vida	03/08/1946
Simonis	Rua Cardoso de Almeida	26/09/1955

Professor	Orientado	Departamento
Polcino	Pedro	Matemática
Simonis	Leo	Estatística

Mas por que a tabela POD foi atualizada se nenhum comando relacionado a ela foi executado? Devido à cláusula {on delete cascade} associada à declaração da chave estrangeira, se a remoção de uma tupla de PEN resultar na violação de uma regra de integridade anterior, a remoção não será rejeitada. Ao contrário, a remoção é feita em cascata, de modo que as tuplas pertencentes a PEN que foram removidas sejam também removidas da tabela POD. De modo similar, a atualização de um campo referenciado por uma regra de integridade não será rejeitada se ela violar essa regra devido à cláusula {on update cascade}. Não faremos o caso da atualização em cascata por ser trivialmente parecido com o caso anterior.

7 Discussão

Um dos objetivos principais desse trabalho é traçar uma comparação entre os problemas de atualização de banco de dados e a teoria da revisão de crença. No lado téorico, notamos claramente que uma base de dados apresenta comportamento semelhante quando se deseja inserir uma nova sentença utilizando os operadores da revisão de crença com o *Finite Partial Epistemic Ranking* ou utilizando a idéia das sentenças rotúladas da teoria de atualização de banco de dados . Para ambos, no exemplo de banco de dados apresentado anteriormente, a inserção eliminou as tuplas (Renata,Flávio,Computação) e (Renata,Thiago,Computação) e conferiu um grau de prioridade maior para a nova sentença e suas inferências. Apesar da grande semelhança, não se pode afirmar que a idéia de Fagin et al. se enquadra totalmente nos Postulados de AGM, pois o Principio da Irrelevância de Sintaxe é quebrado. Por outro lado, a idéia de dar graus de prioridade as sentenças(*Tagged Sentences*) é, dentre as várias sobre atualização de banco de dados, a que mais se aproxima das propostas da Teoria de Revisão de Crença. Uma discussão interessante seria de onde vêm os números das prioridades em cada uma das teorias.Fagin et al. propõe que todas as restrições de integridade tenham máxima prioridade, isto é, grau zero, o que é bastante razoável. Mas não explica de onde vêm os

demais valores, deixando a critério do administrador do banco de dados definir quais as tuplas mais prioritárias. Já Williams nada menciona a respeito de como os números são escolhidos, presupondo que a base já venha com os graus de prioridades definidos. Fica para estudos posteriores definir de onde vêm esses números, se é possível dar graus de prioridades automaticamente ou só mesmo manualmente.

No lado prático, foi feito uma análise de como os bancos de dados relacionais, que dominam mais de 90% do mercado, utilizando a linguagem SQL tratam o problema. Notamos que o SQL trata a atualização e a revisão do mesmo jeito. Na verdade para o gerenciador de dados pouco importa se a informação recebida é nova para a base de dados ou se é apenas uma mudança de uma informação já presente. O que importa é que a base de dados sempre fique consistente. E para isso um bom administrador de banco de dados cria as restrições de integridade em SQL de forma a evitar inconsistências. Os chamados banco de dados dedutivos, que conseguem tratar as inferências nas sentenças, por usarem Datalog (linguagem derivada do ProLog) possuem pouquíssima aceitação no mercado. Fica para estudos posteriores tentar implementar operadores que usufruam das facilidades da linguagem SQL e consigam ao mesmo tempo colocar em prática os verdadeiros operadores da revisão de crença.

Referências

[1] Alchourrón C., Gardenfors P. and Makison D. On the logic of theory change : Partial meet functions for contraction and revision. *Journal of Symbolic Logic*, 1(50):510–530, 1985.

[2] Katsuno H. and Mendelzon A. On the difference between updating a knowlegde base and revising it. In *Proceedings of SICPKRR*, 1991.

[3] Williams Mary-Anne. Transmutations of theory bases. Technical report, University of Newcastle, 1993.

[4] Williams Mary-Anne. Iterated theory base change : A computational model. In *Proceedings of IJCAI*, 1995.

[5] Revesz Peter. On the semantics of theory change : Arbitration between old and new information. Technical report, University of Nebraska, 1993.

[6] Fagin R., Ulman J. and Vardi M. On the semantics of updates in databases - preliminary report. In *Proceedings of Second ACM SIGACT-SIGMOD*, 1983.

[7] Reiter Raymond. On formalizing database updates. Technical report, Department of Computer Science - University of Toronto, 1992.

[8] Marianne Winslett. *Updating Logical Databases.* Cambridge, 1990.