

Síntese de Operações de Atualização sobre Banco de Dados.

27 de julho de 1984

L. Tucherman

Instituto Latino Americano
de Pesquisa em Sistemas
IBM do Brasil Ltda.
Caixa Postal 1830
22.000 Rio de Janeiro, RJ
Tel. 322-4040 ramal 258

M.A. Casanova

Centro Científico de Brasília
IBM do Brasil Ltda.
Caixa Postal 853
70.000 Brasília, DF
Tel. 225-7670 r.5203

Idioma: Português

Total de páginas: 25 páginas

RESUMO

Título: Síntese de Operações de Atualização sobre Banco de Dados.

Autores:

- L. Tucherman
- M.A. Casanova

O presente trabalho apresenta algumas considerações sobre o projeto de aplicações de banco de dados sujeitas a restrições de integridade. O enfoque básico é o de tipos abstratos de dados conjugado com uma extensão do modelo de entidades e relacionamentos. No trabalho é discutido também como prever quais os objetos que devem ser encapsulados pelas operações da aplicação a fim de se garantir as restrições de integridade.

Palavras chaves: banco de dados relacional, restrições de integridade, encapsulamento, tipos abstratos de dados.

CURRICULUM VITAE

Luiz Tucheran

é pesquisador do Instituto Latino Americano de Pesquisa em Sistemas da IBM do Brasil Ltda onde é responsável pela área de banco de dados. Formou-se em Engenharia Civil-Eletrotécnica pela Universidade Federal de Juiz de Fora - MG, obteve o grau de Mestre em Ciência da Computação pela Pontifícia Universidade Católica do Rio de Janeiro em 1983 e presentemente cursa o programa de Doutorado em Ciência da Computação da mesma universidade.

Marco Antonio Casanova

é pesquisador do Centro Científico da IBM em Brasília. Formou-se em Engenharia Eletrônica pelo Instituto Militar de Engenharia em 1974, obteve o grau de Mestre em Ciências pela Pontifícia Universidade Católica do Rio de Janeiro em 1976, e os graus de Mestre em Ciências (1977) e Doutor em Filosofia (1979), ambos em Matemática Aplicada, pela Universidade de Harvard. Seus interesses acadêmicos incluem teoria de banco de dados, sistemas de gerencia de bancos de dados, e projeto e análise de algoritmos concorrentes.

BIBLIOGRAFIA:

- [CHEN76] - Chen P.P., "The Entity-Relationship Model: Toward a Unified View of Data", ACM Trans. on Database Systems, Vol. 1, No. 1, March 1976, pp. 3-36.
- [DATE81] - Date, C.J. "An Introduction to Database Systems vol I e II" Addison-Wesley (1981 e 1983).
- [FURT78] - Furtado, A.L. "Specifying External Data Base Schemas". Proceedings of International Computer Symposium, (1978)
- [FURT79] - Furtado, A.L. e Santos, C.S. "Organização de banco de dados", Editora Campus (1979).
- [SCHE79] - Scheuermann, P.; Schiffner, G.; Weber H. "Abstraction Capabilities and Invariant Properties Modelling Within The Entity-Relationship Approach", Proc. of 1st International Conference on Entity-Relationship Approach, Los Angeles, Dec. 1979
- [WEBE76] - Weber, H. "The D-Graph Model of Large Shared Data Bases: A Representation of Integrity Constraints and Views as Abstract Data Type". IBM Report RJ1875, IBM Reserach, San Jose, CA (Nov. 1976).

INTRODUÇÃO

O presente trabalho apresenta algumas considerações sobre o projeto de aplicações de banco de dados sujeitas a restrições de integridade, tais como "todo departamento tem que ter um gerente" ou "todo curso tem que ter no mínimo dois alunos", por exemplo. O enfoque básico é o de tipos abstratos de dados conjugado com uma extensão do modelo de entidades e relacionamentos [CHEN76, SCHE79].

O enfoque de tipos abstratos de dados postula o encapsulamento das estruturas de dados pelas operações. Em termos de banco de dados, esta idéia é aproveitada permitindo que o banco de dados seja atualizado apenas por um grupo de operações pré-definidas, que necessariamente deverão preservar a consistência do banco. Consultas, por outro lado, continuam livres, diferentemente do enfoque puro de tipos abstratos de dados.

Resumidamente, a extensão do modelo de entidades-relacionamentos usada tem as seguintes características. Começamos observando que o modelo original de Entidade-Relacionamento, conforme proposto por Chen [CHEN76], somente permite a especificação de relacionamentos do tipo (1:1, 1:N, N:1, N:M) para um relacionamento particular. A extensão adotada propõe como construtores adicionais relacionamentos parciais e totais, e relacionamentos fracos. Associados a estes construtores intensionais existem também diversas propriedades invariantes que devem ser preservadas durante as operações de atualização. A extensão adotada permite ainda determinar a

propagação das operações de atualização sobre o esquema conceitual, descrito por um diagrama de entidades-relacionamentos.

No trabalho é discutido também, através de um exemplo de implementação da abordagem de tipos abstratos de dados aplicados a banco de dados [FURT78], como prever quais objetos deverão ser encapsulados pelas operações da aplicação a fim de se garantir as restrições de integridade. A metodologia sugerida baseia-se em uma análise do caminho de propagação das operações de atualização sobre o diagrama de entidades-relacionamentos no modelo estendido.

SÍNTESE DAS CONDIÇÕES E EFEITOS COLATERAIS

O presente capítulo apresenta algumas considerações que devem ser verificadas no projeto de aplicações com banco de dados que utilizem a abordagem de tipos abstratos de dados para a preservação de integridade.

Estas considerações são baseadas no trabalho feito por Scheuermann [SCHE79] que estendeu o modelo original de Entidade-Relacionamento conforme proposto por Chen [CHEN76] com alguns construtores intensionais necessários para expressar restrições de integridade. Como construtores foram incluídos relacionamentos parciais e totais e relacionamentos fracos. Associados a estes construtores intensionais existem diversas propriedades invariantes que devem ser preservadas durante as operações de atualização.

Nesse trabalho é mostrado que o tipo de propriedade invariante e o tipo de operação de atualização univocamente determinam a forma como as atualizações se propagam pelo banco de dados.

1. Construtores

O modelo E-R original somente permite especificar relacionamentos do tipo (1:1, 1:N, N:1, N:M) para um relacionamento particular. Para modelar uma restrição de integridade como, por exemplo, todo empregado deve possuir pelo menos uma habilitação, foram introduzidos os conceitos de relacionamento total e parcial.

Definição 3.1:

- (a) Um relacionamento R_i é uma relação entre n entidades, onde cada elemento pertence a um dos conjuntos de entidades participantes do relacionamento, isto é,
- $$R_i = \{ (e_1, e_2, \dots, e_n) \mid e_i \in E, i = 1, \dots, n \}$$
- onde cada tupla $(e_1, e_2, e_3, \dots, e_n)$ é uma instância do relacionamento, e E_i ($i=1 \dots n$) são as entidades envolvidas.
- (b) Um relacionamento R_i é total em um conjunto de entidades E_k se cada instância do conjunto E_k ocorre em alguma tupla de R_i . Caso contrário o relacionamento R_i é parcial em E_k .

Exemplo:

Relacionamento total

Todo empregado
deve ter uma
habilitação

Podem existir
habilitações
sem empregados



Relacionamento parcial

Podem existir empregados
que não estejam trabalhando
em algum projeto

Podem existir
projetos inativos



Definição 3.2:

Seja R_i um relacionamento e E_j e E_k conjuntos de entidades participantes de R_i . Dizemos que R_i é fraco de E_j para E_k sss

(i) R_i é total em E_j

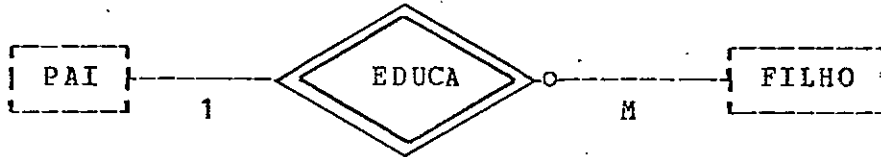
(ii) há uma dependência funcional de E_j para E_k em R_i

Dizemos ainda que E_j é fraca e E_k é forte.

Esta definição não corresponde exatamente à definição original em Chen [CHEN76], no entanto ela parece mais ou menos com a sua representação diagramática de "dependência existencial". Acreditamos, entretanto, que esta definição expressa mais

precisamente o fato de que fraqueza é uma propriedade de um relacionamento e não de uma entidade.

Exemplo:



2. Propriedades invariantes e operações de atualização

Uma operação de atualização tem o objetivo de causar uma modificação em um conjunto de instâncias de uma entidade ou de um relacionamento. Em relação a estas operações, entidades e relacionamentos são de naturezas bem diferentes: 1) relacionamentos somente podem existir se as entidades relacionadas também existirem, e 2) a existência de entidades é regulada pela natureza do relacionamento envolvido. Dependendo do tipo de relacionamento envolvido, as entidades podem existir totalmente independentes (relacionamento parcial) ou condicionadas à existência de outras entidades (relacionamento total ou relacionamento fraco).

Chamamos de propriedades invariantes às regras de existência que devem ser satisfeitas de forma a que o modelo represente o mais precisamente possível a situação do mundo real.

Existem duas formas de preservar estas propriedades durante as operações de atualização. A primeira não permite certas operações de atualização, a segunda permite todas as operações de

atualização corretas e, se necessário, estende uma intenção de atualização às entidades e relacionamentos adjacentes. Essas extensões de intenção são chamadas de propagações de atualização. Neste trabalho são definidas várias regras de propagação para os diferentes construtores intensionais no modelo E-E-R e mostrado como elas podem ser efetivamente cumpridas.

De acordo com a natureza das propriedades invariantes a serem observadas, uma propagação de atualização pode ter que continuar sobre uma série de objetos no diagrama E-E-R. Estas cadeias de possíveis propagações são chamadas de caminhos de propagação.

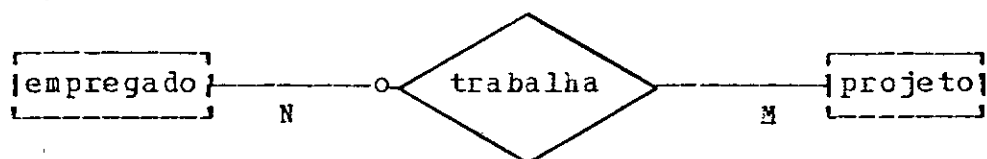
Algumas considerações sobre os caminhos de propagação:

- Ponto de acesso é o objeto no diagrama por onde se inicia a propagação de atualização.
- O resultado geral da propagação não depende da ordem de execução dos caminhos de propagação.
- Entidades que podem existir independentemente são candidatas para o encerramento da propagação

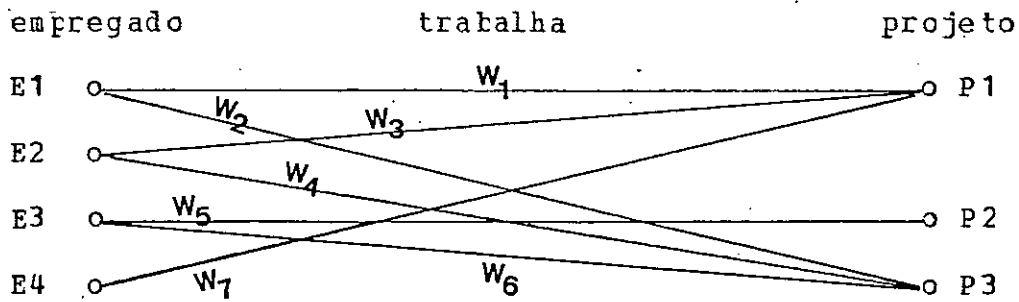
É importante notar a diferença existente entre os relacionamentos totais e fracos quando estamos atualizando uma entidade.

Vamos assumir que os seguintes diagramas são verdadeiros:

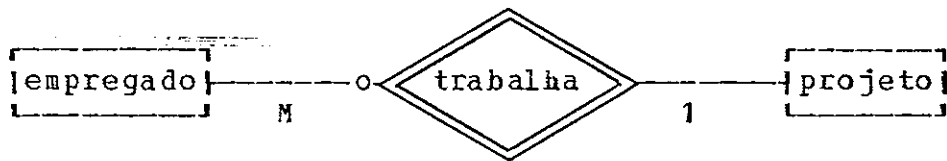
a) caso 1



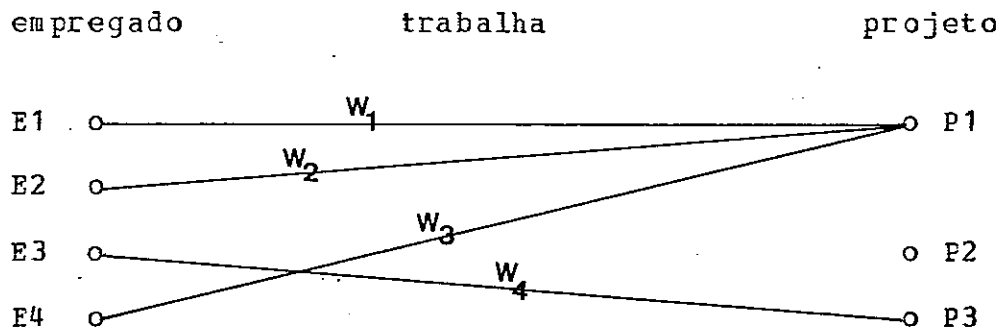
com a seguinte configuração de instâncias.



b) caso 2



com a seguinte configuração de instâncias.



Para o caso 1 de relacionamento total vale a seguinte regra:

Se deletarmos P1 então teremos que deletar w₁, w₃, w₇ e que deletar E4, pois temos de deletar todas as instâncias de empregados que não estão relacionadas com nenhum projeto.

Para o caso 2 de relacionamento iraco vale a seguinte regra:

Se deletarmos P1 então teremos que deletar w1 , w2 , w3 e que deletar E1 , E2 , E4 pois temos de deletar todas as instâncias de empregados que são dependentes daquele projeto.

3. Regras de propagação para redes E-E-R

Conforme vimos na seção anterior os caminhos de propagação nos indicam que objetos podem vir a ser acessados, para verificação ou atualização, durante a execução de uma operação.

Definiremos nesta seção como os caminhos de propagação são formados.

- A propagação inicia-se em um ponto de acesso, ou seja, um objeto da rede E-E-R.
- A propagação pode continuar nos objetos adjacentes chamados de pontos de continuação.
- A propagação termina em um ponto de terminação.

Um caminho de propagação pode ser descrito como:

PONTO DE ACESSO	nome
OPERAÇÃO	nome
CAMINHO	ponto de acesso.objeto1.objeto2....nil

Embora o grafo E-E-R não seja dirigido, deve-se direcionar as arestas quando se analisa a propagação de atualização.

Assim podemos definir:

- Seja D um grafo E-E-R,
- Um objeto de D é um ponto de acesso se uma solicitação externa (remoção, adição ou modificação), feita por um usuário, se refere a este objeto
- Um objeto X de D é um ponto de continuação de um caminho P se:
 - a) uma solicitação foi propagada até X através de uma aresta do caminho P e
 - b) X tem uma ou mais arestas divergentes e
 - c) existe um objeto Y conectado com X por uma aresta divergente para o qual a atualização deve ser propagada e que sua propagação ainda não foi incluída neste caminho P .
- Um objeto X é um ponto de terminação do caminho P se:
 - a) uma solicitação foi propagada até X através de um caminho P e é verdade que
 - b) não existe aresta divergente de X ou
 - c) não existe necessidade de se propagar a atualização ao longo de nenhuma aresta divergente de X ou
 - d) todas as propagações a partir de X através de suas arestas divergentes já foram incluídas neste caminho P .

Estas definições implicam que:

- Um ponto de acesso pode ser um ponto de continuação ou um ponto de terminação.
- Se um ponto de acesso é também um ponto de terminação, então o caminho de propagação é circular.
- Se um ponto de acesso é também um ponto de continuação, ou se um ponto de continuação aparece duas vezes na expressão do caminho, então o caminho contém um ciclo.

Se devemos ou não propagar uma atualização a partir de um objeto X do grafo E depende da natureza da solicitação e das propriedades invariantes associadas com a construção intensional em que este objeto participa.

4. Regras de propagação aplicadas a propriedades invariantes

Distinguiremos quatro configurações básicas para a propagação de atualizações, conforme a figura abaixo.

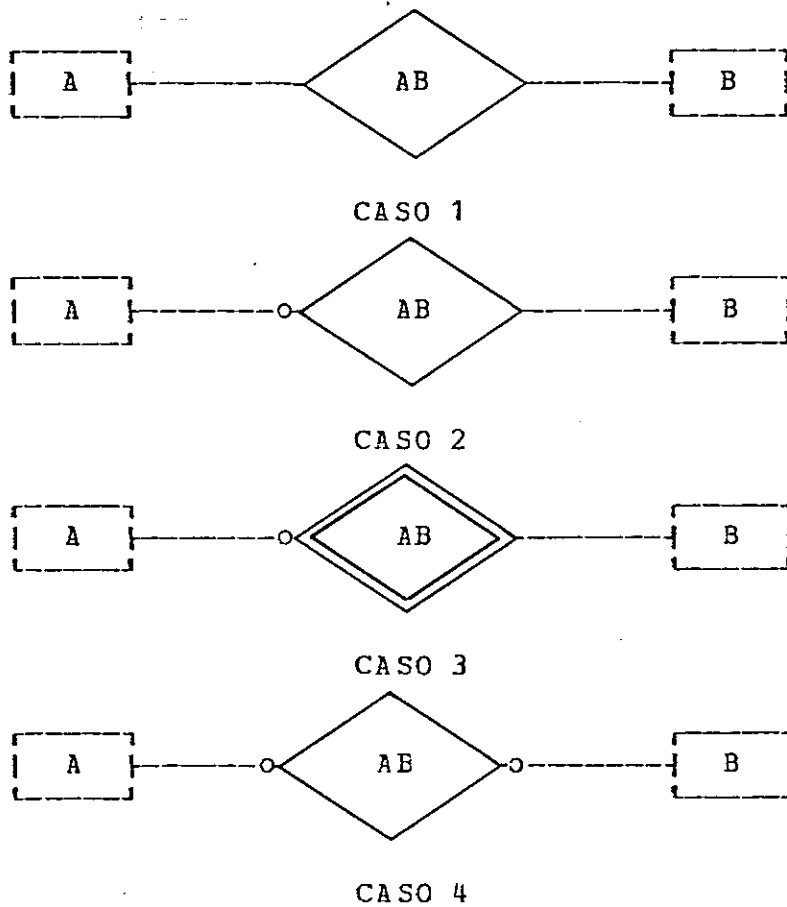


Figura 1. Configurações básicas para propagação

Nos restringimos a relacionamentos binários simplesmente por conveniência de ilustração.

As tabelas abaixo resumem as regras de propagação para estas quatro configurações. Consideramos separadamente o tipo de ponto de acesso, isto é, se é entidade ou relacionamento e, para cada um deles, discriminaremos o tipo de operação. Convém notar que o ponto de acesso nestas tabelas pode ser considerado qualquer objeto que já tenha sido acessado no caminho de propagação.

Ponto de acesso	A D I C A O		D E L E C A O	
	relaciona- mento	entidade oposta	entidade oposta	relaciona- mento
A ou B (caso 1)	—	—	—	*
A (caso 2)	*	*	—	*
A (caso 3)	*	*	—	*
B (caso 2)	—	—	*	*
B (caso 3)	—	—	*	*
A ou B (caso 4)	*	*	*	*

Figura 2. Regras de propagação para entidades

Acesso ao relacionamento AB	A D I C A O		D E L E C A O		M O D I F I C A Ç Ã O	
	entid. A	entid. B	entid. A	entid. B	entid. A	entid. B
	caso 1	*	*	—	—	*
caso 2	*	*	*	—	*	—
caso 3	*	*	*	—	—	—
caso 4	*	*	*	*	*	*

Figura 3. Regras de propagação para relacionamentos

Algumas observações nestas tabelas são necessárias. O asterisco indica que o objeto deve ser incluído no caminho de propagação que começa no ponto de acesso, pois o objeto pode vir a ser acessado, durante a execução, para atualização ou para alguma verificação. Deve-se lembrar que nesta abordagem somente é permitido operações de atualização corretas. Por exemplo, no caso de adicionarmos um relacionamento, este se traduz em também adicionar as entidades correspondentes, caso elas não existam.

O sentido de se ter operação de modificação na tabela de relacionamento significa que pode-se modificar valores dos atributos do relacionamento assim como o próprio relacionamento.

5. Rede de caminhos de propagação

Conforme já apresentado, as propagações de atualização podem ocorrer simultaneamente ao longo de vários caminhos de propagação ou ao longo de um caminho circular. Assim sendo, devemos garantir que as propagações de atualização devem satisfazer às seguintes propriedades: (i) todas as propagações de atualização devem eventualmente terminar; (ii) o resultado geral é independente da ordem em que os caminhos de propagação são executados.

De forma a garantir estas propriedades, devemos tomar cuidado na restrição dos caminhos de propagação. Assim, após aplicarmos interativamente as regras básicas de propagação, devemos decidir sobre o número de caminhos de propagação e seus objetos constituintes.

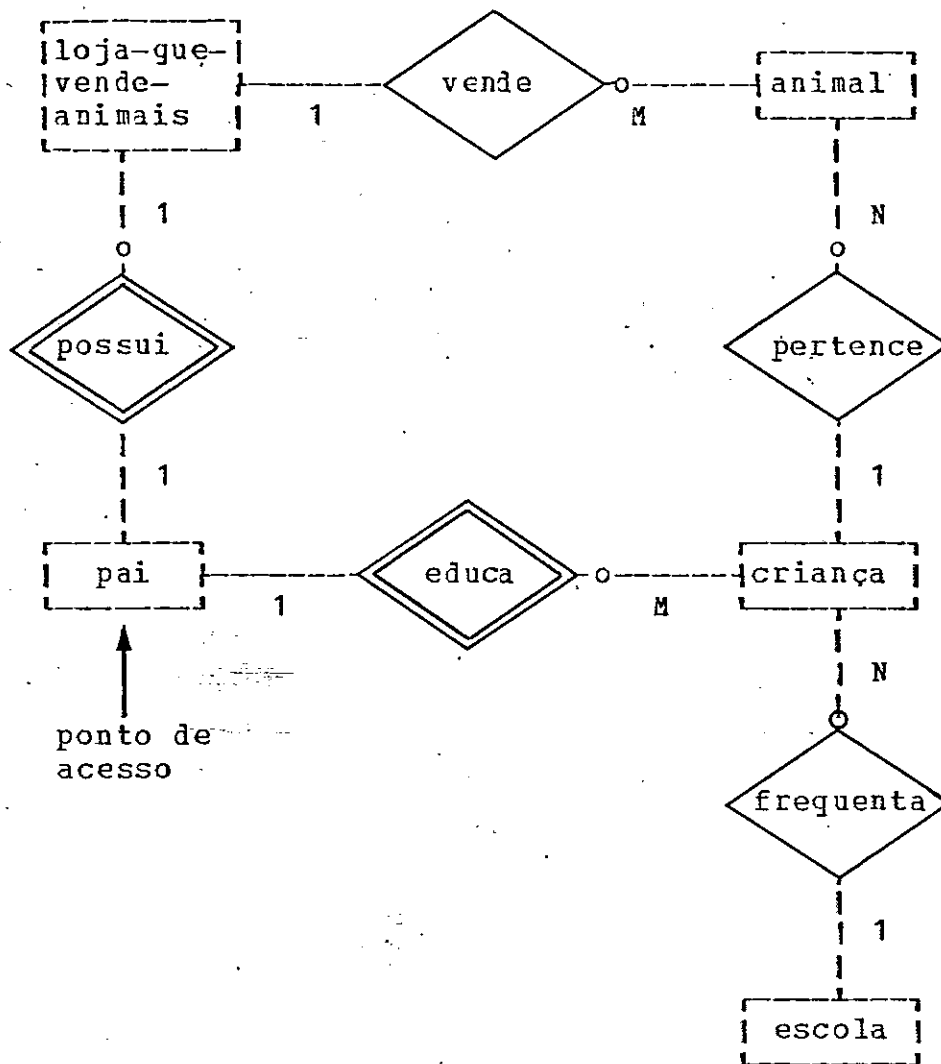
Quantos caminhos de propagação devemos ter de forma a garantir que o resultado global seja o mesmo?

Para resolver este problema, precisamos observar as seguintes regras:

- i) Um caminho de propagação deve ser construído para cada aresta divergente de um ponto de acesso ao longo da qual uma atualização deve ser propagada se esta aresta não pertencer a um ciclo de propagação de atualização que contenha o ponto de acesso. (Se todas as arestas divergentes pertencem a um ciclo, somente um caminho de acesso deve ser construído).
- ii) Um sub-caminho de propagação deve ser construído para cada aresta divergente de um ponto de continuação ao longo da qual uma atualização deve ser propagada se esta aresta não pertencer a um ciclo de propagação de atualização que contenha este ponto de continuação.

Chamamos de rede de caminhos de propagação a todos os caminhos de propagação de uma operação de atualização que partem de um ponto de acesso.

Exemplo:



Ponto de acesso: PAI

Operação : DELEÇÃO de uma instância de PAI

Obtemos a seguinte rede de caminhos de propagação:

Caminho 1:

PAI.POSSUI.LOJA-QUE-VENDE-ANIMAIS.VENDE.ANIMAL.PERTENCE.NIL

Caminho 2:

PAI.EDUCA.CRIANÇA. (sub-caminho 2.1 + sub-caminho 2.2)

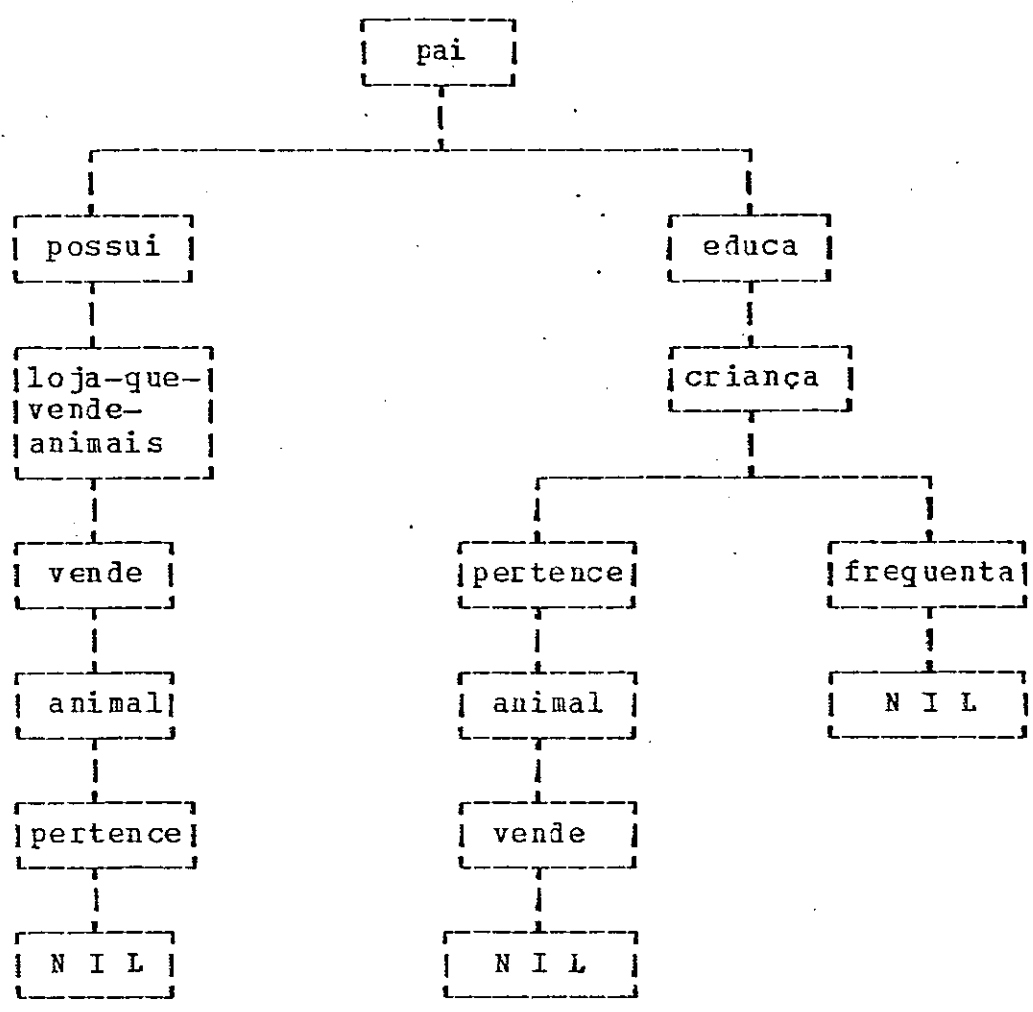
Sub-caminho 2.1:

CRIANÇA.PERTENCE.ANIMAL.VENDE.NIL

Sub-caminho 2.2:

CRIANÇA.FREQUENTA.NIL

Esta rede poderia ser representada por uma árvore de caminhos de propagação, onde a raiz é o ponto de acesso.



Um outro ponto de interesse neste trabalho é que, de forma a poder suportar a propagação das atualizações, o usuário deve acessar, não somente o ponto de acesso, como também todos os objetos do caminho de propagação.

IMPLEMENTAÇÃO E AVALIAÇÃO DE UM CASO

O objetivo deste capítulo é apresentar um exemplo da implementação da abordagem de tipos abstratos de dados aplicados

a banco de dados, conjugando-a com a discussão do capítulo anterior.

1. Descrição do exemplo

O exemplo escolhido é o da área de pessoal de uma pequena companhia industrial. Este exemplo é bastante simplificado e certamente não cobre todas as situações que podem ocorrer em descrições mais detalhadas [FURT79].

Após um estudo de sistemas de informação, foi identificado o seguinte diagrama de entidades e relacionamentos estendido (os atributos foram omitidos por simplicidade):

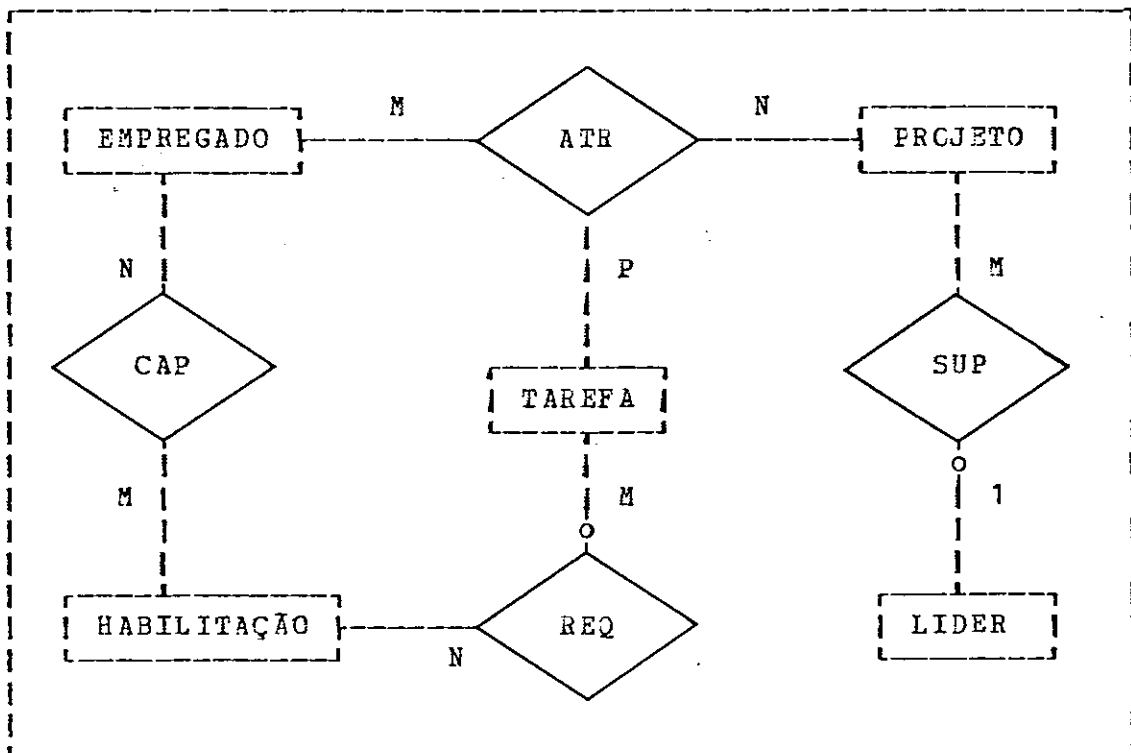


Figura 4. Diagrama Entidade-Relacionamento Estendido

A partir deste diagrama podemos obter, em dois passos, o esquema conceitual relacional descrito abaixo. Inicialmente obtem-se um esquema relacional a partir do diagrama E-E-R através das transformações usuais. Em seguida, simplifica-se este primeiro esquema eliminando-se os esquemas de relação de PROJETO, TAREFA, LIDER e HABILITAÇÃO por terem somente um atributo. Note que o diagrama também gera restrições de integridade, primariamente pelo fato de que as entidades participantes de um relacionamento devem necessariamente existir no banco.

- Esquemas de Relação

EMP (N,S,C) - nome, salário e cargo do empregado

REQ (T,H) - requisito de habilitação para executar tarefas

ATR (N,T,P) - atribuição de empregado em projeto a tarefa

SUP (P,L) - supervisão de projeto por um líder

CAP (N,H) - capacitação (habilitações) possuídas ou adquiridas por empregados

- Restrições de Integridade

- ATR[N] é um subconjunto de EMP[N] (ou seja, somente empregados contratados podem ser associados a projetos)

- ATR[T] é um subconjunto de REQ[T]

- CAP[N] é um subconjunto de EMP[N] (ou seja, somente empregados contratados podem ter suas habilitações registradas no banco de dados)

Analisando em seguida as atividades da empresa, restrições de integridade adicionais, além daquelas já induzidas pelo diagrama, são então formuladas:

- 1) Os salários devem ser pelo menos iguais ao salário mínimo
- 2) Um empregado contratado deve ter um e apenas um salário e cargo
- 3) Um projeto não pode ter mais do que um líder em um dado momento
- 4) Um projeto deve ter um líder inicial no momento em que é criado
- 5) Somente projetos sem líder podem ser encerrados
- 6) Empregados só podem estar associados com projetos que no momento tem líder
- 7) Somente empregados que não estão no momento associados a nenhum projeto podem ser demitidos
- 8) Para desempenhar uma tarefa um empregado deve ter todas as habilitações requeridas para a tarefa

Porém, o projeto do banco de dados não deve se restringir a determinar o seu esquema conceitual (que inclui as restrições de integridade). A abordagem de tipos abstratos de dados determina que também as operações sobre as estruturas do banco sejam definidas e que elas preservem as restrições de integridade. Mais ainda, segundo esta abordagem, o usuário somente poderá modificar o banco de dados através das operações definidas no projeto. Desta forma, o usuário não mais precisará se preocupar com o aspecto de consistência lógica do banco: este aspecto está embutido na própria definição das operações.

As consultas continuam livres, no entanto, no contexto de bancos de dados, já que elas não corrompem a consistência do banco. Embora esta posição seja um desvio do conceito básico de tipos abstratos, ela é justificável no ambiente de banco de dados pois não convém restringir em princípio a flexibilidade de consulta ao banco.

A filosofia de tipos abstratos se harmoniza com o projeto clássico de banco de dados através da definição de visões, acrescidas de operações de atualização [WEBE76].

Assim, por exemplo, poderíamos definir as visões do gerente de engenharia da pequena empresa da seguinte forma. Intuitivamente, o gerente de engenharia inicia novos projetos especificando seu nome e o do líder inicial. Ele pode substituir o líder de um projeto, ou suspender um projeto deixando-o sem líder. Nenhum empregado pode continuar ligado a um projeto suspenso. Um projeto suspenso pode ser permanentemente encerrado pelo gerente de engenharia, ou pode ser reiniciado pela atribuição de um novo líder. Várias tarefas compõem cada projeto, e o gerente de engenharia é responsável pela indicação das habilitações de um empregado que são requeridas para realizar cada tarefa. O gerente de engenharia também pode associar empregados com projetos (que não estejam suspensos) e encerrar tais associações.

Mais precisamente, o esquema do gerente de engenharia seria composto de quatro visões:

V-NECESS (T,H) - habilitação necessária para desempenhar uma tarefa
V-QUAL (N,C,H) - qualificação dos empregados
V-FROJ (P,L) - projetos e seus líderes
V-DISTR (N,T,P) - projetos e seus líderes

Estas visões seriam definidas como:

VISÃO V-NECESS

DISPLAY

TUPLAS (T,H)
OPERACÕES requeira (t,h)
OPERACÕES remova (t,h)

EXPRESSÕES

TUPLAS tuplas da relação REQ
OPERACÕES

requeira

CONDIÇÕES nenhuma
EFEITOS inserir <t,h> na relação REQ
EFEITOS COLATERAIS se existir uma tupla
<n,t,p> para algum n e p na relação
ATR e <n,h> não existe em CAP, então
retirar* <n,t,p> da relação ATR.

remova

CONDIÇÕES nenhuma
EFEITOS retirar <t,h> da relação REQ
EFEITOS COLATERAIS nenhum

FIM V-NECESS

VISÃO V-QUAL

DISPLAY

TUPLAS (N,J,H)

EXPRESSÕES

TUPLAS das relações EMP e CAP concatenandos
as que tem o mesmo n.

FIM V-QUAL

VISÃO V-PROJ

DISPLAY

TUPLAS (P,L)

OPERACÕES inicie (p,l)
 substitua (p,l)
 suspenda (p)
 reinicie (p,l)
 encerre (p,l)

EXPRESSÕES

TUPLAS da relação SUP

OPERACÕES

inicie

CONDIÇÕES não existe tupla na relação
 SUP com p

EFEITOS insere <p,l> na relação SUP

EFEITOS_COLATERAIS nenhuma

substitua

CONDIÇÕES nenhuma

EFEITOS modifique <p,-> em <p,l>
 na relação SUP

EFEITOS_COLATERAIS nenhum

suspenda

CONDIÇÕES nenhuma

EFEITOS modifique <p,l> em <p,*>
 na relação SUP

EFEITOS_COLATERAIS retire <-, -, p>
 da relação ATR

reinicie

CONDIÇÕES existe uma tupla <p,*>
 na relação SUP

EFEITOS modifique <p,*> em <p,l>
 na relação SUP

EFEITOS_COLATERAIS nenhum

encerre

CONDIÇÕES existe <p,*> na relação SUP

EFEITOS retire <p,*> da relação SUP

EFEITOS_COLATERAIS nenhuma

FIM V-PROJ

VISÃO V-DISTR

DISPLAY

TUPLAS (N,T,P)

OPERACÕES associe (n,p)
 desassocie (n,p)

EXPRESSÕES

TUPLAS da relação ATR
OPERAÇÕES

associe

CONDIÇÕES existe uma tupla $\langle n, -, - \rangle$ em EMP
e uma tupla $\langle p, l \rangle$ em SUP com $l \neq *$
e não existe a tupla $\langle n, -, p \rangle$ em ATR
EFEITOS insere $\langle n, *, p \rangle$ em ATR
EFEITOS COLATERAIS nenhum

desassocie

CONDIÇÕES nenhuma
EFEITOS retire $\langle n, -, p \rangle$ de ATR
EFEITOS COLATERAIS nenhum

FIM V-DISTR

Nota:

A seguinte notação será usada:

"*" - valores indefinidos

"-" - nos casos em que qualquer valor do atributo correspondente
é aceitável.

- a operação $\text{retira}^* \langle n, t, p \rangle$ verifica se a tupla $\langle n, t, p \rangle$ é
única em ATR para os valores de n e p . Se for, modifica
 $\langle n, t, p \rangle$ para $\langle n, *, p \rangle$ caso contrário, $\text{retira} \langle n, t, p \rangle$ de ATR.

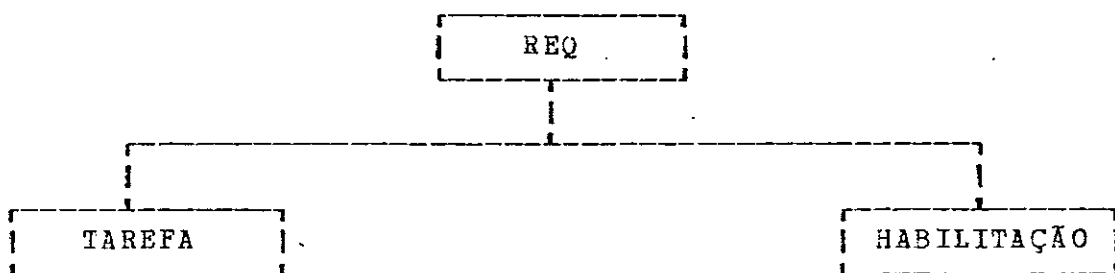
2. Aplicação do método de propagação de atualizações

Mencionamos brevemente na seção anterior que as operações de
uma visão devem preservar a consistência do banco. Se assim não
o fizerem, estarão incorretas. Nesta seção abordaremos
brevemente o problema de construir operações corretas, ou provar
a correção de operações pré-definidas.

Começemos observando que há duas classes de restrições: aquelas oriundas diretamente do diagrama de entidades e relacionamentos e aquelas obtidas por uma análise direta da aplicação. As restrições do primeiro tipo são simples de se obter bastando observar o diagrama, enquanto que as do segundo tipo surgem de uma análise da aplicação.

Da mesma forma, construir uma operação correta, ou verificar a correção de uma operação, com relação às restrições de segundo tipo depende da aplicação. No entanto, o processo de verificar se uma determinada operação preserva as restrições oriundas do diagrama, ou o processo de construí-la corretamente com respeito a estas restrições, pode ser mecanizado através do uso dos caminhos de propagação.

Considere, como exemplo, a operação requeira. Esta operação é essencialmente uma inserção em REQ. Como o esquema de relação REQ foi gerado do tipo de entidade também chamado de REQ, construamos o caminho de propagação para uma inserção em REQ:



Analisando o caminho de propagação, uma inserção em REQ pode forçar uma inserção nos conjuntos de entidades TAREFA e HABILITAÇÃO, ou ser bloqueada, caso não existam as entidades apropriadas. Suponhamos que a primeira opção seja a adotada.

Note-se que TAREFA e HABILITAÇÃO foram eliminadas ao se criar o esquema relacional por possuírem somente um atributo. Assim, a primeira opção é coerente com o projeto do esquema relacional e a operação regueira está naturalmente correta com relação às restrições oriundas do diagrama E-E-R.

Passemos agora a analisar as restrições obtidas através da análise direta da aplicação. A única restrição afetada por uma inserção em REQ seria a restrição 8. Portanto, é necessário verificar se regueira a satisfaz. Para este propósito foram então introduzidos os efeitos colaterais de regueira.

Para concluir esta breve discussão, caminhos de propagação ajudam na definição correta das operações oferecidas para os usuários dentro do enfoque de tipos abstratos, da mesma forma que o próprio diagrama E-E-R auxilia o projeto do esquema conceitual no modelo de dados final. Mas a definição destas operações requer ainda outras ferramentas para testar a preservação de restrições não oriundas do diagrama E-E-R.

CONCLUSÕES

Estudamos uma metodologia que possibilita ao projetista analisar, durante a fase de projeto do banco de dados, as propagações das operações de atualização sobre o esquema conceitual que é descrito por um diagrama entidade-relacionamento. Esta análise nos auxilia na previsão dos objetos que deverão ser encapsulados pelas operações de atualização.

Através de um caso exemplo estudamos o emprego da metodologia de caminhos de propagação de atualizações e comparamos os caminhos obtidos com as operações de atualização descritas pela aplicação, o que nos permitiu concluir da viabilidade da técnica de encapsulamento como um mecanismo eficaz para garantir estas propagações.

Somos de opinião que, independentemente da inclusão das restrições de integridade, o uso da técnica de encapsulamento como interface através da qual se atualiza o banco de dados e como mecanismo de autorização, continua sendo uma estratégia viável no sentido de se definir operações com maior significado para o usuário sem referência a comandos primitivos da linguagem do SGBD.