INFORMÁTICA 82

XV CONGRESSO NACIONAL DE INFORMÁTICA
II FEIRA INTERNACIONAL DE INFORMÁTICA

ANAIS

INFORMÁTICA 82

XV CONGRESSO NACIONAL DE INFORMÁTICA
II FEIRA INTERNACIONAL DE INFORMÁTICA

ANAIS

MÉTODOS PARA GARANTIR A INTEGRIDADE EM BANCO DE DADOS

L. Tucherman / A. L. Furtado
IBM do Brasil Ltda. / Pontifícia Universidade Católica
Rio de Janeiro, RJ

Palavras-chave: banco de dados relacional, integridade semântica, restrições de integridade, asserções de integridade, monitoramento.

O presente trabalho apresenta diferentes métodos para garantir a Integridade Semântica em Banco de Dados, dividindo-os em três áreas: métodos para restrições de uso freqüente e que são básicos para o modelo relacional, método para restrições generalizadas e métodos que utilizam a abordagem de tipos abstratos de dados.

1. INTRODUÇÃO

Um banco de dados é uma coleção de ítens de dados, com o objetivo de modelar uma empresa em um mundo real. Um esquema de banco de dados é a sua estrutura lógica; um estado do banco de dados é uma instância do seu esquema, que modela a empresa em um determinado momento. O esquema do banco de dados pode ser considerado como invariante, enquanto o estado do banco de dados se modifica com o tempo. Uma transação que modifica um estado do banco de dados é chamada de atualização.

Em um sistema integrado de banco de dados, todos os usuários dependem da correção dos dados, ou seja, que as operações de atualização mantenham o banco de dados em um estado consistente. Existem vários motivos para que dados incorretos ocorram em um banco de dados. Estes motivos podem ser: entrada incorreta dos dados, programas de atualização incorretos, falta de entendimento da parte do usuário, etc.

Em todo Banco de Dados devem ser consideradas regras que nos permitam determinar a correção dos dados. Estas regras são conhecidas como *Restrições de Integridade Semânticas* que tratam da prevenção dos erros semântico feitos, através de atualizações autorizadas ou não, devidos em especial, à falta de conhecimento do usuário. Elas procuram assegurar que o estado do banco de dados permaneça correto mesmo que o usuário ou os programas de aplicação tentem modificá-lo incorretamente.

O presente trabalho apresenta diferentes métodos de garantir a Integridade Semântica em Banco de Dados, alguns já implementados em Sistemas de Gerência de Banco de Dados (SGBDs) Relacionais e outros somente em estágio de estudo e/ou pesquisa.

Apresentaremos estes métodos divididos em três grandes áreas:

1. Métodos para restrições de uso frequente, tais como: chaves primárias, domínios, integridade referencial e valores nulos.
2. Métodos para restrições generalizadas, tais como: asserções, modificação de consultas, gatilhos e monitoramento.
3. Métodos que utilizam a abordagem de tipos abstratos de dados.

Conclusões e considerações de custo-benefício além de direções futuras de pesquisa encerrarão o trabalho.

Tomamos como premissa para a elaboração deste trabalho, que o leitor esteja familiarizado com os conceitos de Banco de Dados Relacionais, em particular com as noções de relação, domínios, tratamento mais detalhado do modelo sugerimos [DATE81c].

MÉTODOS PARA RESTRIÇÕES DE INTEGRIDADE DE USO FREQUENTE

Neste capítulo apresentamos os métodos de uso mais frequente, que são básicos para o modelo relacional.

1. Chave Primária

Em uma relação, todas as tuplas devem ser distintas, ou seja, deve existir pelo menos uma combinação de atributos com as seguintes propriedades:

. O valor desta combinação em qualquer tupla da relação distingue esta tupla de todas as outras.
. Se qualquer atributo desta combinação é retirado, a propriedade acima fica sem efeito.

A esta combinação de atributos, com as propriedades acima descritas, denominamos de *chave candidata.*

Quando definimos um esquema conceitual, escolhemos entre as chaves candidatas de cada uma das relações, uma para ser a chave primária. Além daquelas propriedades, uma chave primária não pode ter nenhum componente cujo valor seja nulo, e qualquer modificação no valor de um dos seus componentes deve ser cuidadosamente controlada.

As chaves primárias desempenham um papel importante no sentido de permitirem representarmos objetos do mundo real de uma maneira simples e direta, assim como para que possamos manter a integridade de tal representação.

Se um Sistema de Gerência de Banco de Dados não suporta chaves primárias, podemos ter situações de ambigüidade, decorrentes da impossibilidade de se determinar se duas tuplas representam ou não o mesmo objeto. Assim, a possibilidade de se escolher uma chave primária para cada uma das relações de um esquema, evita termos estados do banco de dados com situações dúbias.

No tocante a modificações de valores das chaves primárias, convém lembrar que podemos ter, no banco de dados, numerosas ocorrências do antigo valor referenciando o valor da chave primária que estamos querendo modificar. Se isso for verdade, estas ocorrências (que referenciam a chave primária) devem de alguma forma ser atualizadas com o novo valor ou então nulificadas. Estas alterações devem ser realizadas em uma mesma transação, de forma a mantermos a atomicidade da operação, assim como mantermos a integridade referencial [Codd81].

Três métodos são apresentados para o problema de atualização de chaves primárias:
a. A utilização de um mecanismo de autorização, que permita somente a um grupo selecionado de usuários atualizar atributos componentes de uma chave primária. Este método não é totalmente satisfatório pois mesmo este grupo poderia cometer enganos.
b. A delegação da tupla que contém a chaves primária a ser modificada, seguida da inclusão de uma nova tupla que contém o novo valor da chave primária, além dos outros atributos da tupla deletada.
c. A implementação de um comando para modificar o valor da chave primária e um outro para modificar os atributos não pertencentes à chave primária. Assim os usuários teriam uma distinção lingüística e operacional entre atualizar o valor de um atributo da chave primária e qualquer outro atributo. A desvantagem seria um maior número de comandos.

2. Domínios

Em um sistema de banco de dados relacional, cada relação é definida sobre um ou mais domínios. Assim, é importante que possamos especificar precisamente o conjunto de objetos que constituem um domínio, uma vez que os valores dos atributos (colunas) de uma tupla da relação são obtidos a partir dos domínios sobre os quais eles foram definidos.

Erros de integridade semântica nos domínios podem ser ocasionados quando o usuário tenta atualizar ou adicionar atributos de uma ou mais tuplas da relação.

A definição de domínios é um dos aspectos da especificação de integridade semântica no contexto de um sistema de banco de dados relacional. Se a virmos como uma restrição de integridade, a definição de um domínio deve incluir: (a) a especificação do conjunto de objetos de dados atômicos que constituem o domínio, (b) quando esta definição deve ser verificada e (c) o que deve ocorrer se ela não for satisfeita.

Na especificação da linguagem proposta por McLeod [MCLE76], a definição dos domínios é verificada sempre que um valor de um atributo (coluna) de uma relação é criado ou modificado, por uma operação de atualização ou adição. Então tudo que necessita aparecer explicitamente na definição de um domínio é a descrição precisa dos objetos que compõem este domínio e as ações que devem ocorrer quando o valor de algum atributo da relação é criado ou modificado e ele não pertence ao domínio sobre o qual aquele atributo foi definido.

Assim, a definição de cada domínio consiste de quatro cláusulas:
a. Nome do domínio
b. Descrição – que permite especificar o conjunto de objetos de dados atômicos que constituem o domínio.

Esta especificação pode, descrever as sub-unidades que compõe um objeto, definir restrições sobre este conjunto e fixar predicados que o descrevam.
c. Ordenação – que indica como os objetos do domínio são ordenados com respeito a comparações com outros objetos do mesmo domínio. Podemos especificar ordenação numérica, lexicográfica ou nenhuma ordenação. Quando nenhuma ordenação é especificada, somente comparações por igualdade são permitidas. Nos outros casos podemos comparar os objetos por >, <, < = e > =. É importante notar que esta ordenação faz parte da definição do domínio, e não a especificação da ordenação de colunas (atributos) de alguma relação, que pode ter a sua ordenação própria.
d. ação de violação – que especifica os procedimentos que devem ser tomados ao se criar ou modificar o valor de um atributo definido. Pode se especificar: (a) que a operação seja rejeitada e uma mensagem de erro seja enviada ao usuário, (b) que um valor particular deve ser assumido como o novo valor do atributo e (c) que um procedimento seja ativado, passe o valor errado como argumento e receba de volta um novo valor para o atributo.

Exemplo da especificação de um domínio:

```
domain DATA
description
dia: number where integer and > = 1 and < = 31 '/'
mes: number where > = 1 and < = 12 '/197'
ano: number where integer and > = and < = 9
where (if mes=4 or =6 or =9 or =11)
   then dia < =30
   and (if mes=2 then dia < =29)
   and (if (mes=2 and ano < > 6) then dia <= 28)
orderig
ano, mes, dia
violation-action
error 'data errada'
```

Assim, os domínios são definidos independentes das relações, que, na sua definição, informam para cada coluna o seu nome, o nome do domínio, a especificação da

unidade, uma descrição narrativa do papel ("role") da coluna e uma indicação da possibilidade de ocorrência de valores nulos nesta coluna.

A importância dos domínios num modelo relacional, também se justifica pelas seguintes razões [CODD81]:

. Fornecer ao sistema informações se duas ocorrências de um mesmo valor do banco de dados denotam o mesmo objeto do mundo real;

A intenção do modelo relacional, é somente tratar com domínios semânticos, isto é, domínios onde os seus valores representam objetos do mundo real. Assim, sempre que existir uma distinção semântica entre os domínios, teremos como consequência operações distintas.

. Restringir operações de união, interseção e diferença;

Estas operações somente são efetuadas se os atributos são definidos sobre o mesmo domínio, fazendo com que o resultado faça sentido, ou seja, represente um significado pertinente.

. Restringir os acoplamentos ("join") baseados em ordenação ou comparação;

Para esta operação, as considerações feitas acima são válidas, uma vez que não faz sentido acoplarmos duas relações por atributos definidos sobre domínios semanticamente diferentes.

. Permitir que as informações comuns dos domínios sejam fatoradas da declaração dos atributos da relação, e associadas com a declaração do domínio que é comum a estes atributos.

3. Integridade Referencial

Conforme visto no ítem de chaves primárias, devemos reconhecer em um banco de dados a existência de uma dependência entre atributos de diferentes relações do seu esquema. Para suportar esta dependência é importante ao SGBD conhecer quais os atributos que são dependentes em sua existência, e de que outros atributos eles o são.

O problema de integridade referencial é de assegurar que o banco de dados satisfaça um conjunto de restrições de referência predefinidas, que aparecem sempre que uma relação inclui referência a outra. Por exemplo, uma relação EMPREGADO inclui referência, através do atributo DEPT, à relação DEPARTAMENTO; a restrição neste exemplo seria EMPREGADO. DEPT c DEPARTAMENTO. DEPT, ou seja, todo valor do atributo. DEPT da relação EMPREGADO deve ocorrer como valor do atributo DEPT de ALGUMA tupla da relação DEPARTAMENTO.

Uma proposta para a especificação de regras de restrições de referências, foi feita por Date [DATE81a]. Nela, integridade referencial é definida como sendo:

"Seja A um atributo de R1 definido por um domínio primário D, i.e., um domínio, especificado opcionalmente, em que alguma relação do banco de dados tem uma chave primária (atributo simples) definida sobre ele".

Então a qualquer tempo, cada valor de A em R1 deve ser (a) nulo ou (b) igual a por exemplo k, onde k é o valor de uma chave primária de alguma tupla de uma relação R2 (R1 e R2 não necessariamente distintas) com chave primária definida por D. R1 é denominada nesta regra, de relação que referencia e R2 de relação referenciada.

Para a formulação de regras específicas de restrições de integridade referencial em um ambiente relacional, a proposta baseada na Unified Database Language – UDL [DATE80], além de garantir a regra básica acima definida, ainda resolve algumas restrições tais como:

. podem existir situações em R1. A é também a chave primária de R1 e desejamos insistir na existência de uma relação referenciada R2 distinta de R1.

. podem existir casos em que a relação referenciada tem chave primária composta de múltiplos atributos enquanto a regra especifica atributo simples.

. a regra básica se apoia no conceito de domínio, que pode não ser suportado pelo SGBD.

. a regra básica assegura que uma relação referenciada R2 deve existir, mas não a identifica. É essencial que conheçamos exatamente quais as relações que são referenciadas por uma dada relação.

. a regra básica não especifica o que o SGBD deve fazer se uma operação de violação é tentada; por exemplo, atualizar um valor de um atributo que é referenciado. Três tipos de ações podem ocorrer: (1) rejeitar a operação, (2) aceitar a operação e fazer uma operação de atualização em cascata nas tuplas da relação que referencia e (3) aceitar a operação mas tornar nulo o valor do atributo das tuplas da relação que referencia.

4. Valores Nulos

A importância da existência de suporte para valores nulos é devida ao fato de que, frequentemente, as informações do mundo real são incompletas e, de alguma forma, devemos poder registrar estas situações no banco de dados. Precisamos poder mostrar quando os valores de atributos de uma relação ainda não são conhecidos. O caso de atributos que não são aplicáveis a todas as tuplas da relação não será considerado.

A existência de valores nulos permite que em um ambiente de banco de dados possamos resolver situações do tipo:

. Durante a criação de uma tupla nova, o usuário não dispõe de todos os valores para certos atributos. Assim o sistema coloca valores nulos nos atributos desta tupla que não tem valores conhecidos.

. No caso de adicionarmos um novo atributo a uma relação existente, o sistema colocaria valores nulos neste novo atributo para todas as tuplas da relação.

. Nas funções de agregação aplicadas sobre algum atributo de uma relação, é importante que o sistema reconheça e ignore tuplas para as quais valores nulos existam. Um exemplo é o da média, pois o usuário poderia ficar surpreso ao verificar que o valor obtido não é igual à soma dividida pelo número dos elementos deste atributo.

. Também nas funções de agregação, se estas forem aplicadas sobre atributos de relações cujas tuplas estão vazias, seria interessante que o sistema retornasse um valor nulo.

Um estudo sobre valores nulos pode ser encontrado em [DATE81b], onde as propriedades dos valores nulos são analisadas, e também são descritas, informalmente, algumas extensões às definições do modelo relacional para incorporar estes valores. São discutidas também algumas das dificuldades que ocorrem em conexão com o conceito de valores nulos. Ele esboça ainda uma alternativa baseada no conceito de valores por ausência ("default").

Nesta alternativa, é associada à declaração de cada domínio a designação de um determinado valor que servirá como valor por ausência. Nos atributos de uma relação especifica-se se cada atributo deve ou não receber valores por ausência.

Quando uma nova tupla é inserida na relação, (a) o

usuário deve fornecer os valores de todos os atributos que não tem valores por ausência; (b) para os outros atributos o sistema fornecerá o valor por ausência se o usuário não fornecer nenhum valor.

Para as funções agregadas aplicadas a um atributo, o usuário deve especificar um predicado explícito para excluir os valores por ausência se isto for o desejado. As funções agregadas foram também estendidas para a inclusão de um argumento opcional, que define o valor a ser retornado se o primeiro argumento é avaliado para um domínio vazio.

MÉTODOS DE RESTRIÇÕES DE INTEGRIDADE GENERALIZADAS

Neste capítulo apresentamos alguns métodos de especificação de restrições de integridade baseados: em asserções e modificação de consultas, gatilhos e monitoramento.

1. Asserções e Modificação de Consultas

A um banco de dados podemos associar um conjunto de restrições semânticas ou asserções, que expressam algumas propriedades semânticas do banco de dados, rejeitando qualquer operação de atualização que viole estas asserções, garantindo assim a consistência do banco de dados.

Uma classificação de asserções de integridade, assim como a especificação de um subsistema para a manutenção de integridade em um banco de dados foi feita por Eswaran [ESWA75]. Nesta classificação podemos:

. Distinguir as asserções em relação ao estado do banco de dados.

Elas podem ser aplicadas em estados estáticos assim como em estados dinâmicos ou transicionais.

. Distinguir se as asserções se aplicam sobre as propriedades de tuplas individuais ou sobre as propriedades de um conjunto de tuplas.

. Definir se as asserções são seletivas ou gerais em relação as operações de atualização, ou seja, em que operações estas asserções devem ser verificadas.

Basicamente, as asserções deveriam ser verificadas em qualquer operação de atualização ao banco de dados. Entretanto o usuário pode desejar especificar que uma asserção seja verificada somente no caso da adição de uma nova tupla, na eliminação/modificação de uma tupla existente ou numa combinação destas operações.

. Definir em que ocasião as asserções devem ser verificadas. Se somente ao fim da execução de uma transação, isto é, após uma sequência de comandos de consulta ou atualização, que é especificada como uma unidade de interação com o SGBD, ou se em estágios intermediários da transação, isto é, após cada atualização de tuplas, individuais ou conjunto. No primeiro caso dizemos que a asserção é adiada e no segundo que ela é imediata.

As quatro classificações acima são independentes entre si e todas as combinações são válidas.

O subsistema discutido, protege o banco de dados contra erros semânticos por permitir ao usuário: (a) definir asserções de integridade que garantem a correção do banco de dados e (b) especificar que procedimentos devem ser tomados quando as asserções não são satisfeitas.

Em sistemas em que a unidade de integridade é a transação [ASTR76, CHAM76, ZLOO78], as asserções são verificadas após o conjunto de operações que a compõem ter sido executado. A razão para que as asserções sejam verificadas após a execução da transação, e não após a execução de cada operação particular, é que, em certos casos, uma operação por si só pode violar uma asserção, mas o conjunto das operações que compõem a transação não. Adiando a verificação das asserções até o término da transação, e no caso de uma das asserções não ser verdadeira, o sistema tem que desfazer todas as modificações feitas no banco de dados pela transação. Assim, podemos ter temporariamente o banco de dados em um estado inconsistente, mas após a execução da transação ele se tornará consistente.

Em sistemas em que a unidade de integridade é um comando único [STON75], as asserções são verificadas antes do comando ser executado. No estudo realizado por Stonebraker [STON75, STON75a, STON76] é proposta uma técnica denominada de modificação de consulta. A palavra consulta, neste caso, deve ser interpretada com um sentido mais amplo, de forma a incluir também comandos de atualização.

Nesta proposta, cada comando de atualização que possa vir a violar as restrições de integridade, é imediatamente combinado com a asserção apropriada, ou seja, são adicionados ao comando de atualização os predicados da asserção. Após esta combinação, o comando é executado é, se não houver violação da asserção, o banco de dados é atualizado.

No trabalho feito por Walker [WALK81], é apresentada uma técnica que transforma automaticamente qualquer transação que o usuário entre no sistema de banco de dados em uma nova transação. Esta, só modificará o banco de dados se for determinado, através de uma sequência de consultas, que as modificações desejadas não lhe acarretam nenhuma inconsistência. Caso essa inconsistência ocorra, a transação transformada apenas envia ao usuário uma mensagem de aviso. Assim, esta técnica evita que necessitemos desfazer as modificações feitas, pelas transações, no banco de dados.

2. Gatilhos e Monitoramento

O gatilho e o monitoramento podem ser vistos como uma extensão das asserções, onde os predicados destas são expressos como condições para os gatilhos. Cada um deles é uma sequência de comandos de atualização, que são executados sempre que a condição pré-especificada pelo predicado é satisfeita.

Na proposta [ESWA76, ASTR76] o usuário pode definir que um gatilho seja executado a partir da ocorrência de uma ação específica: leitura, adição, deleção e modificação de uma tupla de uma relação. Se uma operação, por exemplo de modificação, é realizada sobre várias tuplas e esta operação ativa um gatilho, então ele será executado repetidamente a cada modificação de uma tupla.

O gatilho é sempre imediatamente executado, não podendo ser adiado até o fim da transação. Gatilhos são considerados como parte da transação que os ativa. As asserções, se existirem, serão avaliadas após todos os comandos da transação e de todos os gatilhos, que tais comandos ativam, serem executados. No caso de uma das asserções falhar, as atualizações feitas pela transação e pelos seus gatilhos são restauradas.

Hammer [HAMM78] apresenta uma maneira eficiente de detetar violações de asserções ocasionadas por atualizações no banco de dados. A técnica apresentada é

baseada na premissa de que a estrutura da operação de atualização de um banco de dados pode ser geralmente antecipada, e que uma análise do efeito potencial que uma atualização tem sobre uma asserção, permite que esta seja eficientemente testada. O método permite testar as asserções antes da operação de atualização ser executada, evitando assim a necessidade dos procedimentos de restauração no caso de ocorrer uma violação.

A análise proposta é executada, em tempo de compilação, por um processador de asserções que assume que as possíveis operações de atualização no banco de dados foram pré-definidas. Para cada tipo de operação de atualização, este processador sintetizará uma interação eficiente com o banco de dados (um teste), que, quando for executada pela ativação da operação, determinará que asserções, se existirem, foram violadas pela operação.

Bunemam [BUNE79] propõem um vigia ("alerter"), que é um programa que monitora o banco de dados, indicando a ocorrência de alguma condição pré-especificada para um usuário ou um programa específico. Um vigia pode ser simples, se somente é sensível à tupla que está sendo atualizada, ou complexo, quando monitora o conteúdo de várias relações simultâneas.

Neste texto, um esquema é apresentado de colocação de vigias em consultas complexas envolvendo um banco de dados relacional, e um método é demonstrado para reduzir a quantidade de computação envolvida na verificação da necessidade de ativação de um vigia, ou seja, se a modificação no banco de dados é de interesse para o vigia.

Uma estrutura básica para a especificação de integridade semântica, é proposta por Hammer [HAMM76], com o objetivo de impor uma estrutura na formulação de regras para descrição do ambiente da aplicação. Esta proposta auxilia o Administrador do Banco de Dados a expressar as restrições de integridade relevantes ao banco de dados, assim como o orienta na implementação e execução destas restrições, seja através de um sistema automático, ou através da codificação de rotinas.

MÉTODOS QUE UTILIZAM A ABORDAGEM DE TIPOS ABSTRATOS DE DADOS

Neste capítulo apresentamos alguns estudos que empregam a abordagem de tipos abstratos de dados, para garantir a integridade semântica do banco de dados.

Os tipos abstratos de dados descrevem uma coleção de entidades abstratas e as operações sobre elas, ou seja, são uma especificação da implementação de uma entidade abstrata e suas operações. Este conceito, chamado encapsulamento, permite ocultar do usuário a representação da entidade, assim como limita o acesso ou a modificação de instâncias da entidade apenas através das operações aplicáveis sobre o tipo abstrato.

Baseados neste princípio, podemos pré-definir um conjunto de operações de atualização, que sejam utilizadas por qualquer programa que atualize o banco de dados. Esta estratégia nos permite incluir, dentro do conceito de encapsulamento dos tipos abstratos de dados, as restrições de integridade. Assim, os usuários estariam limitados a modificar o banco de dados usando somente as operações pré-definidas.

Com esta abordagem, podemos incluir na definição de uma visão da aplicação, que pode ser considerada como um objeto abstrato, as operações de manutenção, sem nos referirmos às suas traduções em comandos primitivos da linguagem do SGBD. Além disso, significa que podemos definir operações com maior sentido para o usuário.

Um estudo com este tipo de abordagem, foi feito por Furtado [FURT78] onde é proposta a construção de uma visão, que se constitui de relações derivadas e suas operações. A ênfase, neste estudo é dada às operações de atualização. Cada uma destas operações tem um nome, e sua expressão inclui uma ou mais chamadas para as operações primitivas de atualização do SGBD que serão aplicadas sobre as relações base. Podem no entanto existir situações em que estas operações primitivas se apliquem sobre relações diferentes da relação base da qual a visão é derivada.

O objetivo destas operações de mais alto nível, é garantir que as restrições de integridade sejam mantidas. A expressão de uma operação inclui, além dos efeitos diretamente desejados pelo usuário, as condições que devem ser verificadas para a execução da operação e os efeitos colaterais da operação, ou seja, as atualizações não diretamente desejadas pelo usuário. As condições e os efeitos colaterais preservam as restrições de integridade do banco de dados.

O estudo assume que as operações de interrogação ao banco de dados estão disponíveis em todas as visões e se referem de forma irrestrita aos atributos definidos na visão.

Weber [WEBE76a], propõe um modelo baseado em rede semântica, onde são representados os objetos do banco de dados, as dependências entre os objetos, as visões e as propriedades de modificações dos objetos.

Os objetos do banco de dados são distintos. Identificam-se por símbolos e estão associados a um tipo em cujo especificação definimos as informações sobre o objeto, assim como suas propriedades de modificação.

A especificação de um tipo é baseada no conceito de tipo abstrato de dados, onde ele é caracterizado por todas as operações que lhe são aplicáveis.

Na proposta, a especificação tem a seguinte forma:

identificador-tipo (operação-1,. . ., operação-n)
  especificação da representação do tipo
  especificação da operação-1 ⎫ [INTERFACE]
  :
  : ⎬→ [CORPO]
  especificação da operação-n. ⎭

Weber [EBE76] distingue dois tipos de restrições de integridade basicamente diferentes: as que fornecem caracterizações adicionais sobre os objetos do banco de dados e as que podem especificar as dependências entre estes objetos.

A interface, que é a única parte visível ao usuário, identifica a classe de todos os objetos igualmente estruturados no banco de dados, assim como o conjunto das operações permitidas sobre estes objetos. O corpo descreve a estrutura deste objeto e as modificações ocasionadas nesta estrutura pela execução das operações definidas. A estrutura do objeto está aí representada em termos de tipos mais primitivos.

As restrições do primeiro tipo podem ser incluídas na especificação do tipo, pois este oferece ao usuário a capacidade de restringir as possíveis combinações dos componentes dos objetos, definir as condições que devem ser verificadas, e satisfeitas, antes da execução das operações sobre os objetos deste tipo.

As restrições do segundo tipo também podem ser incluídas na especificação do tipo, uma vez que este permite a definição de objetos compostos, que são os que contém, como componentes, objetos dependentes.

CONSIDERAÇÕES FINAIS

Apresentamos neste trabalho uma série de métodos para garantir a integridade em banco de dados. Devido ao impacto no desempenho que um SGBD Relacional teria para monitorar todos os comandos de atualização, de forma a verificar uma possível violação de uma das restrições de integridade, a maioria dos sistemas comercialmente disponíveis implementam o controle das restrições de integridade de maneira bem limitada [CODD81].

Estes sistemas implementam, com algumas diferenças, as restrições que se aplicam ao modelo relacional, tais como: chave primária, domínios e valores nulos; a restrição de integridade referencial não é incluída em nenhum deles.

O sistema INGRES [STON75a] implementa asserções de integridade que não envolvem funções de agregação, e que se aplicam sobre tuplas de uma relação simples. O sistema ORACLE anuncia, para sua nova versão, a implementação de asserções e gatilhos [DIEC81].

Uma análise de custos/benefícios deverá ser feita, a fim de avaliarmos a viabilidade de implementarmos um dos médicos descritos neste trabalho e que classes de restrições devem ser consideradas. É possível que em algumas situações políticas, uma política de auditoria seja mais conveniente.

BIBLIOGRAFIA

(ASTR76) ASTRAHAN M.M. et al. System R: A Relational Approach to Data Base Management., ACM Transactions on Database Systems 1, 2 (Jun. 1976)

(BUNE79) BUNEMAN, O.P. and Clemons, E.K. Efficiently Monitoring Relational Databases. ACM Transactions on Database Systems, Vol. 4, Nº3, (Sep. 1979)

(CHAM76) CHAMBERLIN, D.D. et al. SEQUEL2: A Unified Approach to Data Definition, Manipulation, and Control, IBM Jounal of Research and Development, (Nov. 1976) pp. 560-575.

(CODD81) CODD, E.F. The Capabilities of Relational Database Management Systems. IBM Research Report RJ3132, IBM Research, San Jose, CA, (May 1981)

(DATE80) DATE, C.J. An Introduction to the Unified Database Language (UDL). Proc. 6th International Conference on Very Large Data Bases (Oct. 1980)

(DATE81) DATE, C.J. An Introduction to Database Systems (3rd edition). Addison-Wesley (1981)

(DATE81a) DATE, C.J. Referential Integrity. IBM Santa Teresa Technical Report 03.132 (Jan. 1981)

(DATE81b) DATE, C.J. The Problem of Null Values. IBM Santa Teresa Technical Report 03.168 (Oct. 1981)

(DIEC81) DIECKMANN, E.M. Three Relational DBMS. Datamation (Setp. 1981) pp. 137-148.

(ESWA75) ESWARAN, K.P. and Chamberlin, D.D. Functional Specification of a Subsystem for Data Base Integrity. Proc. Ist International Conference on Very Large Data Bases (Sept. 1975), pp. 48-68.

(ESWA76) ESWARAN, K.P. Specifications, Implementations and Interactions of a Trigger Subsystem in an Integrated Database System. IBM Research Report, RJ1820, IBM Research, San Jose, CA, (Nov. 1976).

(FURT78) FURTADO, A.L. Specifying External Data Base Schemas. Proceedings of International Computer Sumposium, (1978).

(HAMM76) HAMMER, M. and McLeod D.J. A Framework for Data Base Semantic Integrity. Proc. 2nd International Conference on Software Engineering, San Francisco, CA, (Oct. 1976).

(HAMM78) HAMMER, M. and Sarin, S.K. Efficient Monitoring of Database Assertions. Proc. of ACM-SIGMOD International Conference on Management of Data, (Jun. (1978), pp. 38-49.

(McLE76) MCLEOD, D.J. High Level Domain Definition in a Relational Data Base System. IBM Reserarch Report RJ1716, IBM Research, San Jose, CA, (Feb. 1976).

(STON75) STONEBRAKER, M.R. Implementation of Integrity Constraints and Views by Query Modification. Proc. of ACM-SIGMOD International Conference on Management of Data, (May 1975), pp. 65-78.

(STON75a) STONEBRAKER, M.R. and Held, G.D. and Wong, W., INGRES: A Relational Data Base Management. Proc. AFIPS National Computer Conference, Anaheim, CA. (May 1975).

(STON76) STONEBRAKER, M.R. et al The Design and Implementation of INGRES. ACM Transactions on Database Systems 1, 3 (Sept. 1976), pp. 189-122.

(WALK81) WALKER A. and Salveter S.C. Automatic Modification on Transactions to Preserve Data Base Integrity Without Undoing Updates. Technical Report 81/026, Dept. of Computer Science, State University of New York at Stony Brook, NY, (Jun. 81).

(WEBE76) WHEBER, H. A Semantic Model of Integrity Constraints on a Relational Data Base. Proc. IFIP-TC2-Working Conference, Frendenstadt, Germany (Jan. 1976).

(WEBE76a) 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 Research, San Jose, CA, (Nov. 1976).

(ZLOO78) ZLOOF, M.M. Security and Integrity Whithin the Queryby-Example Data Base Management Language. IBM Research Report RC6982, IBM Thomas J. Watson Research Center, Yorktown Heights, (Feb. 1978).