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

XV CONGRESSO NACIONAL DE INFORMÁTICA
SUCESU RIO DE JANEIRO – OUTUBRO 1982

PROJETO DE BANCO DE DADOS BASEADO EM RESTRIÇÕES DE INTEGRIDADE

Clesio Saraiva dos Santos / José Mauro Voekmer de Castilho / Antonio Luz Furtado
Universidade Federal do Rio Grande do Sul / Pontifícia Universidade Católica do Rio de Janeiro
Av. Osvaldo Aranha, 99 – Porto Alegre, RS / Rua Marques de São Vicente, 225 – Rio de Janeiro, RJ

Palavra-chaves: Projeto de banco de dados, restrições de integridade, descrição formal de aplicações de banco de dados.

As restrições de integridade impostas a um banco de dados podem servir como base para o projeto do próprio banco de dados e da aplicação correspondente. Com este objetivo, esboçamos neste trabalho uma metodologia de projeto de aplicação de banco de dados, e apresentamos parte de uma taxonomia para restrições de integridade e alguns formalismos que permitem expressar rigorosamente restrições de integridade complexas.

1. INTRODUÇÃO

O projeto de um banco de dados, como o de sistemas de um modo geral, se inicia por uma fase de análise de requisitos. Os resultados desta análise são organizados e detalhados convenientemente até chegar-se a um conjunto de descrições ou especificações que caracterizam as informações que estarão armazenadas no banco de dados e os programas que irão consultar ou alterar estas informações.

A caracterização das informações armazenadas é feita usualmente com o emprego de um modelo de dados. Um modelo de dados é constituído por um conjunto de conceitos com forma, significado e comportamento definidos, com os conceitos de relação, atributos e chave, por exemplo, no modelo relacional [Codd70]. No entanto, nem sempre os conceitos do modelo de dados são suficientes para caracterizar os estados e as transições que são válidas para um banco de dados. É preciso restringir-se as definições feitas com uso dos conceitos puros, para que se tenha a caracterização exata das informações, estados e transições. Isto é feito estabelecendo-se regras adicionais, as restrições de integridade, a serem satisfeitas pelo banco de dados e pelos programas que o manipulam.

É fácil admitir que as duas restrições:

(a) "a idade de um empregado não pode ser inferior a 15 anos"

(b) "o salário de um empregado não pode diminuir"

são de tipos diferentes, pois enquanto (a) pode ser verificada por uma simples inspeção da configuração atual do banco de dados, (b) requer que se examine a transição de um estado para o seguinte. A classificação das restrições de integridade apresentada neste artigo e que deriva daquela apresentada em [Santos 80] distingue várias classes de restrições, de acordo com vários critérios.

Dentre as vantagens de dispor-se de tal classificação, está a de se poder usá-la com uma lista ("check-list") a ser seguida durante a fase de análise de requisitos.

A especificação das informações componentes incluindo as restrições de integridade sobre elas vigentes constitui o *esquema conceitual* do banco de dados dentro do espírito da proposta ANSI SPARC [ANSI 75].

Existem propriedades que podemos associar aos esquemas conceituais e que dão uma medida da qualidade dos mesmos [Casanova 81]. Assim, dispondo de um esquema conceitual com um conjunto de restrições de integridade, não devem existir contradições entre estas restrições, e as transições de estado previstas ou desejadas para o banco de dados não devem violar as restrições de transição. É desejável que o conjunto de restrições seja mínimo, isto é, que não haja no conjunto restrições que sejam conseqüência de outras. Quando for associado ao esquema um conjunto de operações ou programa para realização das transições de estado desejadas para o banco de dados, estas operações não devem violar as restrições de integridade, e é desejável que elas sejam aplicáveis, isto é, que exista pelo menos um estado válido ao qual a operação possa ser aplicada com sucesso.

Existem pesquisas sobre linguagens de definição de esquemas baseadas em lógica [Veloso81, Casanova 81, Castilho 82] que possibilitam descrever de maneira rigorosa as características estáticas e, especialmente, dinâmicas de aplicações de banco de dados, sem que seja preciso descrever explicitamente os programas ou operações que manipulam o banco de dados. Estas linguagens permitem

o desenvolvimento do projeto de esquemas conceituais com a construção de descrições da aplicação com diferentes graus de detalhamento.

Neste trabalho é apresentada e discutida uma aproximação ao problema de projetar aplicações de banco de dados, baseada na enumeração e expressão de restrições de integridade. Na seção 2 é delineada e discutida uma técnica de construção de descrições formais de aplicações. A seção 3 apresenta uma taxonomia para restrições de integridade que pode facilitar o trabalho de identificação das restrições associadas às aplicações cuja descrição se quer construir. Na seção 4 apresentam-se, com exemplos, linguagens adequadas à descrição de aplicações de banco de dados. A seção 5 conclui o trabalho, indicando algumas direções para trabalho futuro.

2. PROJETO DE APLICAÇÕES DE BANCO DE DADOS

Normalmente o projeto de uma aplicação de banco de dados se inicia com uma análise do sistema de informação existente (ou previsto) da empresa, usando técnicas como a construção de diagramas de entidades e relacionamentos [Chen 76], ou a síntese de um esquema relacional em terceira forma normal a partir de formulários e boletins de entrada ou saída [Codd 72]. Estas técnicas delineiam a estrutura dos dados do banco, e são complementadas por descrições informais das operações, transações ou movimentos que farão parte da aplicação a ser implantada. A imprecisão das descrições que se obtém é grande, e a maior parte da semântica da aplicação fica registrada apenas em documentação "off-line". Muito poucas condições que se gostaria de impor sobre os dados e sobre suas mudanças, como reflexo de situações da realidade, são expressas. A definição de atributos "chave primária" para determinado tipo de representação de objeto, a existência de relacionamento 1:N, N:M ou N:1 entre objetos, são algumas das condições, ou restrições de integridade, que são analisadas e expressas com as técnicas usuais.

Vamos seguir neste trabalho uma aproximação diferente onde é dada grande ênfase a restrições de integridade.

O processo inicia com um levantamento de todos os fatos e situações que dizem respeito a aplicação a ser implementada. Os fatos e situações são apresentados em linguagem natural. Estes fatos e situações falam de *estados válidos* do banco de dados da aplicação, e de mudanças ou *transições de estado válidas* do banco. Desta apresentação se extraem elementos lingüísticos: sujeitos e objetos, seus atributos e predicados. Com estes elementos se define uma linguagem formal, que é então usada para expressar de maneira precisa os fatos e situações anteriormente apresentados. É possível que numa primeira etapa apenas se descreva estados e transições válidas sem mencionar como os estados são construídos ou como as transições ocorrem, ou seja, sem mencionar operações ou programas que manipulam o banco.

Temos então uma descrição da aplicação em um nível de abstração bem grande, mas no entanto rigorosa, sem os problemas de uma descrição informal. O conjunto de restrições de integridade expressas desta maneira inclui, é claro, as restrições comuns de *chave* e de relacionamentos 1:N e outros, e outras restrições menos comuns, além de fixar o comportamento mínimo esperado das operações sobre o banco de dados. A enumeração das restrições é crítica do ponto de vista da fidelidade da aplicação às intenções do projetista: busca-se um conjunto *completo* de restrições, onde, tanto quanto possível, nada foi esquecido ou colocado a mais. Neste processo, é de grande utilidade a existência de uma caracterização ou classificação para restrições de integridade, que atue como guia. Uma taxonomia de restrições de integridade, desenvolvida em parte com esta finalidade, é apresentada na seção 3.

A descrição da aplicação obtida neste processo tem dois componentes: uma *linguagem formal,* definida com os elementos lingüísticos identificados na análise da apresentação inicial dos fatos e situações sobre a aplicação, e um *conjunto de fórmulas* escritas nesta linguagem, que corresponde ao conjunto de restrições de integridade da aplicação.

As principais vantagens da descrição de uma aplicação como apresentada acima são: (1) a descrição é *formal,* portanto não-ambígua; (2) a aplicação é descrita em um *nível de abstração alto,* usando termos e símbolos próprios do ambiente onde a aplicação vai ser usada; (3) a descrição pode ser analisada, verificando-se ou deduzindo-se propriedades que ela possui, como por exemplo a ausência de contradições entre as restrições; (4) implementações da aplicação descrita podem ser verificadas como corretas em relação a descrição construída, usando técnicas formais; (5) por ser uma descrição "mínima", contendo apenas detalhes relevantes da aplicação (inclusive comportamento mínimo desejado das operações ou programas de manipulação do banco) há normalmente um grande número de alternativas de implementação, o que possibilita a escolha de uma alternativa ótima.

Desvantagens: (1) é difícil construir as descrições. É preciso estabelecer técnicas e ferramentas que ajudem no levantamento das restrições e na verificação de propriedades desejadas para as descrições. (A taxonomia de seção 3 é uma importante contribuição); (2) a manipulação das descrições, para fins de verificação de correção de implementações ou de verificação de propriedades das descrições, é complexa (computacionalmente falando), conforme o formalismo de descrições adotado. Se se adota formalismos baseados em lógica (como proposto na seção 4) é possível que se tenha de considerar problemas de indecibilidade associados à lógica de primeira ordem.

Uma vez estabelecida uma descrição de aplicação, que contém a caracterização dos estados válidos do banco de dados correspondente e que estabelece limites para a gama de transições de estado válidas para este banco, pode-se definir um conjunto de operações ou programas de aplicação que vão realizar as alterações ao banco, mudando seu estado. O comportamento destas operações não deve violar a descrição já construída, ou seja as operações não devem gerar estado inválidos, nem realizar transições de estado inválidas. É possível construir nova descrição da aplicação, definindo as operações por suas propriedades, propriedades estas expressas com uso de outra linguagem formal, entre cujos elementos lingüísticos se incluem símbolos correspondentes às operações.

Se as propriedades das operações contidas nesta segunda descrição não desrespeitam a descrição original, então dizemos que esta segunda descrição é um "refinamento" da primeira. Uma implementação das operações, sob forma de programas que satisfazem as propriedades contidas na segunda descrição, pode

constituir uma terceira descrição da aplicação. É interessante notar que as descrições citadas tem diferentes graus de abstração: a primeira é a mais abstrata, é a que estabelece o mínimo necessário de restrições (não se considera qualquer detalhe de implementação, nem sequer qual o conjunto de operações de manipulação adotado); a segunda fala de operações e suas propriedades e é menos abstrata porque já fala sobre um conjunto de operações por último, programas são associadas às operações, obtendo-se uma descrição "física" onde aparecem detalhes de implementação.

A construção da primeira descrição (a mais abstrata) coincide com a fase de análise de dados; a construção das descrições seguintes (segunda e terceira) são etapas do projeto da aplicação e do banco de dados correspondente. Considerações sobre representação física de informações, organizações de arquivo, métodos de acesso, atendimento a requisitos de segurança e proteção aos dados, são tratados na construção da terceira descrição.

3. RESTRIÇÕES DE INTEGRIDADE

Nesta seção apresentamos um esquema de classificação de restrições de integridade quanto aos aspectos de *origem, substância, especificação e aplicação.*

Esta classificação apresenta interesse sob vários pontos de vista. Em primeiro lugar, permite um melhor conhecimento das restrições de integridade e dos diversos aspectos a elas referentes. O interesse maior, no entanto, decorre de sua utilidade na execução de tarefas de grande relevância, tais como a especificação do esquema conceitual de um banco de dados e a definição de um sistema de gerência de bancos de dados.

Na especificação de um esquema conceitual, a classificação constitui uma "check-list" a ser seguida na captação e formulação das restrições. Na definição de um sistema de gerência, é necessária ao serem estabelecidos os recursos a serem oferecidos para a especificação e a manutenção das restrições de integridade.

No estabelecimento desta classificação, vários trabalhos foram considerados. Alguns deles definem classes de restrições de integridade tomando como base o modelo relacional [Furtado79, Hammer 76, Weber 76], ou o modelo CODASYL Melo 79], sendo as restrições diferenciadas principalmente quanto à complexidade das condições que expressam.

Vários trabalhos dedicados ao estudo de classes particulares de restrições podem ser citados, tais como as dependência s de junção [Aho 77, Bernstein 76, Delobel 78, Fagin 77, Housel 79, Mendelzon 79, Rissanem 77], sendo que alguns destes tratam especificamente de dependências funcionais e multivaloradas, estabelecendo formas normais e métodos de obtenção de um esquema conceitual normalizado, por síntese ou decomposição, a partir destas dependências, no modelo relacional.

Mais recentemente encontramos alguns trabalhos que tratam de restrições de integridade em modelos de nível mais alto, como o modelo de entidades e relacionamentos [Poonen 78, Santos 80a, Santos 81] e o modelo de redes semânticas [Mylopoulos 78]. A abordagem das restrições no contexto destes modelos é mais adequada, uma vez que as restrições definem propriedades semânticas do banco de dados, as quais podem ser expressas de modo mais natural com o uso de conceitos mais próximos dos objetos reais modelados.

3.1 ASPECTOS CONSIDERADOS NA CLASSIFICAÇÃO

O esquema de classificação apresentado a seguir pode ser visto como quatro esquemas independentes:
– quanto aos agentes que impõem as restrições (*origem*);
– quanto ao condicionamento imposto ao BD (*substâncias*);
– quanto às formas usadas para a incorporação das restrições à especificação do banco de dados (*especificação*);
– quanto aos mecanismos de manutenção da integridade e às características dinâmicas das restrições (*aplicação*).

3.2 EXEMPLOS DE RESTRIÇÕES

Apresentamos um pequeno conjunto de restrições de integridade para uma aplicação de banco de dados de uma empresa onde empregados trabalham em projetos e estão ligados a departamentos. A aplicação de banco de dados vai ajudar a controlar o pessoal e os projetos da empresa.

Restrições:

(1) O salário de um empregado não pode descrever.

Este Basic é interpretado, de forma que o usuário não precisa efetuar compilações para fazer desenhos. Os comandos podem ser executados dentro de um programa, ou então, através de comandos imediatos, de forma que um desenho pode ser desenvolvido através de um processo de experimentação, tentativas e erro.

A relação dos comandos do Basic é a seguinte:
Não gráficos: REM (comentário), DIM (dimensão), DATA (dados) READ (leitura de dados), RESTORE (reler dados), LET (atribuição), FOR. . .TO. . .STEP (loop), NEXT (fim de loop), END (fim de programa), GOTO (desvio), IF (desvio condicional), PRINT (escrita), GOSUB (chamada de rotina) e RETURN (retorno de rotina);
Funções. TAB (tabulação), ABS (valor absoluto), INT (valor inteiro), RND (randômico), SGN (sinal da expressão), SQR (raiz quadrada), SIN (seno), COS (cosseno), TAN (tangente), ARG e CALL (para chamar rotinas assembler);
Gráficos: COLOR (cor), SCREEN (limpa tela), MOVE (desloca cursor), PLOT (retas), LOCATE (posiciona cursor por meio externo), WHERE (sabe X, Y do cursor), Write (escreve frases e expressões), CHAR (informa tipo do caractere), AREA (desenha área para histogramas) e GRID (traça reticulado).

O Basic ocupa 9kb, podendo ser usado como uma "feature" adicional de terminais gráficos. Foi desenvolvido no NCE, porém um trecho do parser foi baseado num compilador cedido para universidades, da PROTEC, e utilizou um conjunto de rotinas de ponto flutuante já pronto, da MICROTECH.

O Basic Gráfico contribuiu em muito para o ensino de gráficos, e para aplicações não muito complexas é preferido ao pacote compatível com CALCOMP, que embora bem mais completo é mais difícil de ser utilizado.

Foi implementado por um analista e um programador em aproximadamente 3 meses. Essa rapidez se deveu ao fato da simplicidade da linguagem e ao know-how já obtido no desenvolvimento das outras rotinas gráficas.

11. TERMINAL GRÁFICO

O terminal gráfico é um equipamento periférico acoplável a computadores através de uma interface de comunicações RS-323 ou Loop de corrente, e que se

destina a produzir desenhos coloridos na tela de TV.

O terminal consta de uma tela de TV colorida, um teclado alfanumérico, um teclado de funções e um joy-stick (light-pen opcional). O usuário usando o teclado se comunica com o computador que comanda os desenhos na tela. Usando o joy-stich (ou o light-pen) ele aponta para posições da tela que devem sofrer qualquer mudança e o terminal envia para o computador a posição indicada.

O terminal gráfico é composto das seguintes partes (placas):

- CPU e memória: Z-80 com 16kb de ROM e 16 de RAM;
- Interface de comunicação: duas USART (Intel 8251), uma para comunicação como o computador principal, e outra para comunicação com um periférico hardcopy (p.e. impressora gráfica ou plotter eletrostático);
- Vídeo gráfico: interface idêntica à descrita anteriormente.

12. FIRMWARE DO TERMINAL GRÁFICO

Existe um paralelo entre a programação de controle dos terminais alfanuméricos e gráficos. Ambos recebem comandos por uma linha de comunicação e os executam, além de transmitirem dados de meios externos (por exemplo, teclado) e internos (por exemplo, vídeo) para o computador principal. Comumente são implementados em algoritmos relativamente simples, que controlam o fluxo de dados e a seleção de função. A execução das funções, controlada por tabelas de seleção ou estruturas do tipo CASE.

No terminal gráfico do NCE, tanto o teclado quanto os links de comunicação operam sob interrupção de forma que as funções gráficas, que são relativamente complexas, podem ter grandes tempos de execução, sem prejuízo para o fluxo de entrada para o sistema. Essa complexidade nos leva a pensar em uma UCP mais rápida para outras implementações de terminais, porém, é bem aceitável o tempo que uma UCP de 8 bits (Z-80 no nosso caso) leva para executar essas funções. Há a exigência de um buffer de tamanho razoável, para haver um certo balanço no sistema (2kb no mínimo).

O terminal ao receber sequências alfanuméricas, se comporta como um terminal de caracteres comum, desenhando as letras correspondentes na tela. Os comandos gráficos são sequências de escape (caracter ESC seguido de uma cadeia de números representáveis em 7 bits, especificando as funções, parâmetros e modificadores de função). Com isso, ganhamos uma portabilidade maior, pois o terminal pode ser ligado sem problemas no mesmo lugar e com a mesma linha de comunicação que um terminal alfanumérico.

A existência ou não de protocolo de comunicação foi para nós uma opção séria. Optamos por um funcionamento semelhante a um terminal burro, ligado com especificações RS-232 ou loop de corrente, a 4 fios (full-duplex), com atendimento a XON-XOFF, para não haver transbordamento de caracteres no buffer.

As funções gráficas implementadas em firmware foram as seguintes: transmissão e recepção de mapas de bits, seleção de cor de fundo, seleção de cor de traçado, posicionamento do cursor X-Y, traçado de retas, traçado de círculos, traçado de circunferências e arcos de circunferências, barras coloridas (para histogramas), letras orientadas em múltiplos de 90 graus e com tamanho múltiplo de 8 x 5 ou 12 x 9 e geração de caracteres com padrão fornecido pelo usuário.

Obs.: Caso haja interesse de obter os algoritmos destas funções deverá ser mantido um contato com nossa instituição.

13. SOFTWARE BÁSICO PARA O COMPUTADOR PRINCIPAL

O terminal pode ser comandado por vários pacotes de software gráfico existente. No caso de nossa instituição, fizemos a adaptação no software gráfico do PDP-11/70 para geração de desenhos no plotter CALCOMP incluindo "drivers" específicos para o nosso equipamento.

Pretendemos num futuro breve fazer a implantação de um software padrão (provavelmente o GKS alemão), mas ainda não temos previsão para isso, dependendo também de contatos externos e de documentação.

14. ESTAÇÃO DE CAD DE BAIXO CUSTO

Um sistema de CAD tem por funções armazenar, buscar, tratar e apresentar visualmente informações gráficas. Consiste normalmente de um "pacote" de hardware e software especializado.

O coração de um sistema de CAD é a estação de trabalho, onde o engenheiro interage com o sistema para desenvolver um projeto em detalhe, sempre monitorando o seu trabalho numa tela do tipo TV. Através de comandos apropriados pode definir, modificar e retirar dados do projeto, sem necessidade de desenhá-lo no papel.

Um sistema típico de CAD inclui um processador central com memória de massa, onde são armazenados os desenhos e programas. A estação de trabalho consiste de uma unidade de visualização com um tubo de raios catódicos, um teclado e um dispositivo de saída gráfico.

Pelo que já se pode perceber, o hardware SDE com vídeo gráfico é o hardware de uma estação de trabalho (a menos de um meio para entrada de dados gráficos, por exemplo, um digitalizador). Nossa idéia é desenvolver uma estação de CAD inteligente (principalmente software), orientado para o projeto de circuitos integrados. Esta estação terá baixíssimo custo, cerca de 10.000 dólares (lembrando que uma estação de CAD de maior capacidade, importada, custa de 100.000 a 500.000 dólares).

15. OBSERVAÇÕES FINAIS

Da experiência obtida no desenvolvimento da interface colorida do SDE-40, conclui-se que a utilização de microprocessador de 16 bits seria o adequado para desenvolver o terminal gráfico colorido. No nosso caso, fomos obrigados, em face da pequena equipe e de razões de tempo, a usar uma UCP já pronta, de 8 bits. Isto tornaria o terminal muito mais rápido, e além disso, pelo espaço de endereçamento maior, a paginação poderia ser dispensada.

(2) Os empregados do departamento número 10 tem categoria maior ou igual a 15.

A restrição a seguir supõe que tenha sido adotado o modelo relacional de dados (Codd 70), e que tenham sido definidas, entre outras, as relações "empregado" e "Departamento", ambas com um atributo "Número de Departamento".

(3) Todo Número de Departamento que aparece na relação "empregado" deve aparecer na relação "Departamento".

(4) O número de um departamento não pode ser mais do que dois dígitos decimais.

(5) Os salários somente podem ser laterados entre os dias 20 e 25 de cada mês.

(6) Toda vez que um empregado é promovido para uma categoria superior, recebe um aumento de 10% em seu salário.

(7) No final de cada ano, todos os empregados do departamento 20 devem ter categoria igual ou superior a 15.

(8) Cada empregado tem apenas um salário.

(9) Um empregado só pode deixar a empresa se não estiver desempenhando tarefas em algum projeto.

(10)Só é possível atribuir tarefas em um projeto a um empregado que tenha as capacitações requeridas.

Algumas destas restrições são citadas como exemplo e enquadradas na classificação que se segue. Outras vão ser usadas na seção 4 para ilustrar os formalismos de descrição.

3.3 CLASSIFICAÇÃO

Nesta seção, são discriminadas as classes de restrições de integridade de acordo com os aspectos e critérios selecionados.

Na indicação das classes, à cada uma delas é associado um código Hw, x, y, zK onde w adota o aspecto e subclasse, respectivamente. O número entre parênteses, à direita, indica uma restrição da seção 3.2 que se enquadra na classe.

3.3.1 ORIGEM

0.1 origem
- 0.1.1 ambiente (1)
- 0.1.2 aplicação (2)
- 0.1.3 modelo (3)
- 0.1.4 implementação (4)

As restrições do ambiente são impostas por agentes externos ao sistema real modelado pelo banco de dados e devem ser observados por toda aplicação no mesmo ambiente.

As restrições da aplicação refletem normas internas à instituição na qual se desenvolve a aplicação e que se referem às informações registradas no banco de dados. À medida que novas restrições vão sendo incorporadas à especificação do banco de dados, é aumentado o grau de automatização das decisões da empresa.

As restrições originadas no modelo surgem como conseqüência de sua inadequação à informação representada ou de um mau projeto de esquema.

As restrições originadas na implementação tem o objetivo de fazer com que os dados fornecidos estejam de acordo com as opções de representação física estabelecida.

3.3.2 SUBSTÂNCIA

O aspecto de substância permite a diferenciação das restrições de acordo com vários critérios que refletem características relevantes do seu significado. Estes aspectos evidencia o tipo de condicionamento imposto à evolução do banco de dados.

Como ilustração, seguem as classificações quanto à abrangência de estados (configurações instantânea de banco de dados) e quanto ao propósito. Outros critérios são considerados em (Santos 80).

s.1. abrangência de estados
- s.1.1 intra-estado (2)
- s.1.2 inter-estados (1)
- s.1.3 transição (5)

Infra-estado são as restrições que envolvem a situação dos objetos em um mesmo estado. Inter-estados envolvem as situações dos objetos em diferentes estados. As restrições de transição condicionam as transições entre estados. De um modo geral, estas últimas estabelecem condições cuja validade depende de fatos não registrados no BD.

s.2 propósito
- s.2.1 condicional (1)
- s.2.2 causativo (6)
- s.2.3 verificativo (7)

As restrições condicionais especificam condições que impedem ou permitem a efetivação de determinadas operações sobre o BD. As causativas determinam a execução de operações complementares em situações particulares do BD. Podemos ter, ainda restrições com propósito verificativo, isto é, restrições que não influenciam diretamente a execução de operações mas estabelecem condições que devem se verificar quando da ocorrência de um determinado evento.

3.3.3 ESPECIFICAÇÃO

Este aspecto trata da incorporação de uma restrição à especificação do BD. O critério considerado é a forma de declaração das restrições.

e.1 declaração
- e.1.1 implícita
- e.1.2 explícita
- e.1.3 derivada

Restrições implícitas são aquelas inerentes às construções do modelo de dados utilizados. Explícitas são aquelas às quais correspondem declarações na especificação do modelo conceitual do BD. Restrições derivadas são conseqüências lógicas das restrições explícitas, combinadas ou não as implícitas.

3.3.4 APLICAÇÃO

Este aspecto trata das formas pelas quais a integridade do BD é montada. Neste caso não estamos classificando as restrições em si, mas sim as alternativas e mecanismos de manutenção de integridade.

Na verdade, diferentes restrições podem induzir diferentes procedimentos para a sua verificação e manutenção. No entanto, de modo geral, podemos dizer que as classes indicadas representam alternativas de manutenção da integridade.

Como ilustração, seguem as classificações quanto ao tratamento e quanto à inspeção. Outros critérios são considerados em (Santos 80).

a.1 tratamento
- a.1.1 prevenção
- a.1.2 complementação
- a.1.3 reconstituição

A prevenção implica no estabelecimento de medidas

que impeçam ocorrência de violação à restrição. Na complementação, a ocorrência de uma violação dispara a execução de uma operação que restaura a integridade do BD. Na reconstituição, são desfeitas as operações que causaram a violação. Apesar de dispendioso, o tratamento para reconstituição é muitas vezes, o único viável, mormente nos casos em que a violação somente é detectada ao final de execução de uma transação complexa.

a.2 inspeção
  a.2.1 direta
  a.2.2 indireta
  a.2.3 mista

A verificação da integridade do BD envolve o exame dos objetos envolvidos. Normalmente os objetos são inspecionados diretamente. Em alguns casos, no entanto, isto pode ser muito dispendioso e é possível tornar-se mais eficiente o processo de verificação pela definição de objetos que são visíveis apenas aos procedimentos de manutenção de integridade (inspeção indireta). Em alguns casos ainda, alguns objetos são inspecionados, diretamente e outros indiretamente (mista) para a verificação de uma mesma restrição.

Os aspectos de origem e susbtância dizem respeito à fase de análise de dados, durante o desenvolvimento de uma aplicação de banco de dados. A observação destes aspectos auxilia a identificação de restrições de integridade. Já os aspectos de especificação e aplicação das restrições de integridade são elementos a serem observados durante as fases de projeto e implementação da aplicação de banco de dados.

Na próxima seção discutimos o problema de expressar de maneira rigorosa as restrições de integridade, e nos preocupamos especialmente com restrições que abrangem mais de um estado do banco de dados.

4. FORMALISMOS DE DESCRIÇÃO

Nesta seção vamos apresentar alguns formalismos que parecem adequados para a descrição de aplicações de banco de dados como discutido na seção 2. As descrições compreendem uma linguagem formal, contendo elementos lingüísticos adequados a cada aplicação, e um conjunto de fórmulas nesta linguagem, que expressam as restrições de integridade de aplicação. Por simplicidade vamos considerar apenas parte do exemplo da seção 3.2, e definir uma linguagem formal que seja suficiente para descrever apenas as restrições (8), (9) e (10).

Como parte da definição desta linguagem formal, dizemos que ela deverá ser capaz de expressar situações envolvendo *empregados* e seus *salários*, a *atribuição* de empregados a tarefas em projetos, as *capacitações* de empregados, e os *requisitos* exigidos para o exercício de tarefas. Se quisermos apenas expressar restrições estáticas, ou seja, aquelas que caracterizam os estados válidos do banco de dados, é possível adotar como linguagem formal uma linguagem polisortida simples, de lógica de primeira ordem [Enderson 72]. Seria adequado para expressar a única restrição estática (8) dentre as três restrições escolhidas, uma linguagem polisortida L com sortes *nome, salário, tarefa, capacitação* e *projeto*, e que inclui, além dos símbolos lógicos usuais (*e, ou, não,* variáveis para cada sorte, parênteses, símbolo "=", quantificadores universal e existencial V e E, símbolo de implicação "⇒"), os símbolos predicativos EMP, ATR, CAP, REQ, com sortes *nome* x *salário, nome* x *tarefa* x *projeto, nome* x capacitação, tarefa *x* capacitações, respectivamente.

A interpretação intuitiva para a fórmula EMP (n,s) diz que esta fórmula será verdadeira quando a pessoa identificada pelo nome *n* for um empregado da empresa, com salário *s*.

A restrição (8) seria expressa como segue:

(8) Vn Vsl Vs2 ((EMP (n, s1) e EMP (n, s2)) U s1O s2)

e é lida como

"para todo nome n e para todos os salários s1 e s2, se EMP (n, s1) e EMP (n, s2) são verdadeiras então s1 é igual a s2, ou seja n só tem associado a si um valor de salário".

Um estado válido do banco de dados com relação a esta restrição será um estado em que a interpretação e avaliação desta fórmula resulta num valor "verdade".

A linguagem usada para expressar a restrição (8) não é suficiente para expressar (9) e (10), porque ela é adequada para falar apenas sobre *estados* e não sobre *transições* de estado. Adotamos então uma outra linguagem, LT, que contém todos os elementos da linguagem L apresentada antes, mais os símbolos □, □*, ◇ e ◇⁻. Se P é uma expressão escrita em L (como (8) acima) damos à expressão □P a seguinte interpretação intuitiva: ela será verdadeira se em todos os estados futuros que se pode alcançar em um só passo, seja este qual for, P é verdadeira. Por sua vez, □* P será verdadeira se em todos os estados futuros (alcançados èm um ou mais passos) e no estado atual, P for verdadeira. Além disso, ◇ P será verdadeira se existir algum estado futuro, alcançável num passo só, em que P é verdadeira, e : ◇P será verdadeira se existir algum estado futuro (não importa o número de passos em que ele é alcançado) em que P é verdadeira. Distinguimos então transições de estado efetuadas num único passo, ou *atômicas*, de transições que envolvem a passagem por estados intermediários. As restrições (9) e (10) seriam expressas em LT como segue:

(9) ∀n ((∃ T ∃p ATR(m, t, p) ⇒□ (∃ s EMP (n,s)))

que é lida como: "para todo nome n, se existe alguma tarefa t atribuída ao empregado de nome n no projeto p então em qualquer estado futuro alcançado num passo só, existe um salário associado a n, ou seja o empregado continua na empresa".

(10) ∀n∀t∀p (((*não* ATR(n, t, p) *e* ◇ ATR (n, t, p))) ⇒∀c(REQ (t, c) ⇒CAP (n, c)))

é lida como: "Para todo n, t e p, se o empregado n não exercia tarefa t no projeto p e se em algum estado futuro alcançável num passo só ele passa a exercê-la, isto significa que atualmente ele possui todas as capacitações requeridas".

As linguagens L e LT vistas acima não mencionaram operações. LT fala de transições de estado num passo só, mas não faz menção a como estas transições acontecem. Vamos considerar a existência de operações de alteração ao banco de dados, por exemplo a operação de atribuir uma tarefa a um empregado num projeto. Para que esta operação seja bem sucedida é necessário, que a pessoa envolvida seja empregado na empresa e que além disso ela tenha as capacitações requeridas. Vamos usar para descrever o comportamento esperado da operação de atribuição, uma

linguagem LO que tem os mesmos elementos de L, mais o símbolo "[atribuir (n, t, p)]. Uma fórmula [atribui (n, t, p)] p, onde P exprime uma restrição estática, é interpretado intuitivamente como: ela será verdadeira, se P for verdadeira em todos os estados alcançados pela aplicação da operação que faz a atribuição da tarefa t dentro do projeto p ao empregado n (denotada por "atribui (n, t, p)".

A seguinte fórmula de LO expressa uma propriedade *de atribui*:

$\forall n \forall t \forall p$ (($\exists$ s EMP (n, s) $\wedge \forall c$(REQ (t, c) $\Rightarrow$CAP (n, c))) $\Rightarrow$[Atribui (n, t, p)] ATR (n, t, p))

Esta fórmula diz que se existe o empregado n e se ele satisfaz os pré-requisitos para exercer a tarefa, então, depois de executada a operação denotada por "atribui (n, t, p)", o empregado n foi atribuído à tarefa t no projeto p.

Outras fórmulas devem ser escritas dizendo que a operação não efeito algum quando n não é empregado ou quando n não tem as capacitações requeridas, e dizendo também que a operação apenas cria a atribuição mencionada, e que nada mais afetado pela operação.

Se a operação de atribuição tem este comportamento, a restrição (10) não vai ser violada. Assim, esta operação pode estar associada a uma descrição de aplicação que é *refinamento* da descrição abstrata construída com LT.

Pode-se agora construir para cada operação programas em alguma linguagem, de tal forma que o comportamento destes programas respeite as propriedades apresentadas.

Em SEQUEL [SEQUEL 74] a operação de atribuição poderia ser descrita como segue (considerando a existência das relações EMP [NOME, SALÁRIO], REQ [TAREFA, CAPACITAÇÃO], CAP[NOME, CAPACITAÇÃO] e ATR[NOME, TAREFA, PROJETO]):

```
INSERT INTO ATR:
    SELECT EMP. NOME, REQ. TAREFA, ATR.
PROJETO
    FROM EMP, CAP, REQ, ATR
    WHERE EMP. NOME = 'n'
    AND REQ.TAREFA = 't'
    AND ATR.PROJETO = 'p'
AND (SELECT CAPACITAÇÃO
FROM CAP
WHERE CAP.NOME O
WHERE CAP.NOME = 'n')
CONTAINS
(SELECTE CAPACITAÇÃO
FROM REQ
WHERE REQ.TAREFA = 't'
```

Uma análise cuidadosa desta instrução em SEQUEL constata que ela respeita as propriedades da atribuição apresentada antes.

5. CONCLUSÕES

Neste trabalho examinamos o papel e a importância das restrições de integridade no projeto de aplicações de banco de dados. Foi esboçada uma metodologia para projeto de aplicações de banco de dados, e apresentaram-se alguns aspectos e critérios para classificação de restriçoes. Finalmente foram mostrados alguns formalismos que permitem expressar de forma rigorosa restrições de integridade que abrangem um ou mais estados dos bancos de dados.

A classificação de restrições ainda está sendo pesquisada por nós, com vistas à obtenção de uma metodologia de projeto de banco de dados.

Os formalismos também estão sendo estudados, e buscamos formas de tratar automaticamente, ou pelo menos sistematicamente, as descrições construídas segundo cada formalismo, visando a verificação de propriedades das descrições, como a ausência de contradições entre restrições, a satisfação de descrições mais abstratas (correção de implementação), entre outras.

Especialmente se pretende explorar a possibilidade de descrever aplicações de banco de dados em diferentes graus de abstração, buscando sistematizar, tanto quanto possível, o processo de "refinar" descrições. A construção modular de descrições, por integração de descrições ou "visões" parciais é outro ponto a ser explorado. O objetivo final é o desenvolvimento de técnicas e ferramentas de "software" que assistam ao projetista de banco de dados em seu trabalho, desde a face de análise de dados até a implementação de aplicação.

BIBLIOGRAFIA:

(AHO 77) AHO, A. V., BEERI. C & ULLMAN, J.D. The theory of joins in relational detabases – In: 18th SYMPOSIUM OF FOUNDATIONS OF COMPUTER SCIENCE, 18., Oct. *Proceedings.* 1977.

(ANSI 75) ANSI/x3/SPARC. *Interim report of the study group on data management systems.* FDT Bulletim ACM, 1975.

(BERNSTEIN 76) BERNSTEIN, P. A. Synthesizing third normal form relations from functional dependencies. *ACM TODS. 1*(4):, 277-98, 1976.

(CASANOVA 81) CASANOVA, M.A; CASTILHO, J.M.V. DE & FURTADO, A.L. Properties of conceptual and External Database Schemas. *Monografias em Ciência da Computação.* (11)1981.

(CASTILHO 82) CASTILHO, J.M.V. DE; CASANOVA M.A. & FURTADO, A.L. A Temporal Framework for Database Specifications. In: INTERNATIONAL CONFERENCE on VERY LARGE DATABASES, Mexico, 1982. (a ser publicado).

(CHEN 76) CHEN, P. The Entity-Relationship model – Towards a unifiel view of data. *ACM TODS.* 1(1):9-36, 1976.

(CODD 70) CODD, E. F. A relational model for large shared data banks *CACM 13*(6): 377-87, 1970.

(CODD 72) CODD, E. F' Further normalization of the database relational model. In: RUSTIN, R., ed. *Database systems.* Englewood Cliffs, Prentice Hall, p. 33-64, 1972.

(DELOBEL 78) DELOBEL, C. Normalization and hierarchical dependencies in the relational data model. *ACM TODS, 3*(3), 1978.

(ENDERTON 72) ENDERTON, H.B. *A mathematical introduction to logic* New York, Academic Press, 1972.

(FAGIN 77)FAGIN, R. Multivalued dependencies and a new normal form for relational detabases. *ACM TODS. 2*(3): 262-78, 1977.

(FURTADO 79) FURTADO, A.L., SEVCIK,

K.C. [1] SANTOS, S. S. dos. Permiting updates through views of data bases. *Information systems, 4*(4): 269-83, 1979.

(FURTADO 81) FURTADO, A.L., SANTOS, C.S. dos & CASTILHO, J.M. V. de. Dynamic modelling of a simple existence constraint. *Information system, 6*(1): 73-80, 1981.

(HAMMER 76) HAMMER, M.M. & MCLEOD, D.J. A framework for semantic integrity. In: INTERNATIONAL CONFERENCE SOFTWARE ENGINEERING, 2, Out. 1976. *Proceedings.* 1976.

(HOUSEL 79) HOUSEL, R., WADDLE, V. & YAO, S.B. Functional dependency model for logical database design. In: INTERNATIONAL CONFERENCE on VERY ALRGE DATABASES. 5., Rio de Janeiro, Out. 3-5, 1979. *Proceddings.* Rio de Janeiro, Sucesu, 1979.

(MELO 79) MELO, R.N. Monitoring integrity constraints in a CODASYL-Like DBMS. In: INTERNATIONAL CONFERENCE on VERY LARGE DATABASES, 5., Rio de Janeiro, Out. 3-5, 1979. *Proceedings.* Rio de Janeiro, Sucesu, 1979.

(MENDELZON 79) MENDELZON, J.M. & MAIER D. Generalized mutual dependencies and the decomposition od database relations. In: INTERNATIONAL CONFERENCE on VERY LARGE DATABASES, 5., Rio de Janeiro, Out. 3-5, 1979. *Proceedings.* Rio de Janeiro, Sucesu, 1979.

(MYLOPOULOS 78) MYLOPOULOS, J., BERNSTEIN, P.A. & WONG, H.K.T. *A preliminary specification of TAXIS: A language for designing interactive information systems.* Computer Corporation of America, 1978. (TR CCA-78-02).

(POONEN 78) POONEN, G. CLEAR: A conceptual language for entities and relationships. ICMOD Conf. 1978. *Proceedings.* 1978.

(SANTOS 80) SANTOS C.S. dos. *Uma caracterização sistemática para restrições de integridade.* Rio de Janeiro, PUC, 1980. (Tese de Doutorado).

(SANTOS 80a) SANTOS, S.S. dos. NEUHOLD E.J. & FURTADO, A.L. A datatype approach to the Entity-Relationship model. In: CHEN, P., ed. *Entity-relationship approach to System Analysis and Design.* Amsterdam, North Holland, 1980.

(SANTOS 81) SANTOS, C.S., MAIBAUM, T.S.E., & FURTADO, A.L. Conceptual Modelling of Database Operations. *International Journal of Computer and Information Sciences, 105*), 1981.

(SEQUEL 74) CHAMBERLIN, D.D. & BOYCE, R.F. SEQUEL: a Structured English Query Language. In: ACM SIGMODWORKSHOP on DATA DESCRIPTION ACCESS AND CONTROL, Michigan, May 1-3, 1974. *Proceedings.* New York, ACM, 1976. p. 249-64.

(VELOSO) VELOSO, P.A.S., CASTILHO, J.M.V. de & FURTADO A.L. Systematic Derivation of Complementary Specifications. In: INTERNATIONAL CONFERENCE on VERY LARGE DATABASE, 7., Cannes, Setp. 9-11, 1981. *Proceedings.* 1981. p. 409-20.

(WEBER 76) WEBER, H. A semantic model of integrity constraints on a relational databases. In: NIJSSEN, A.M., ed.. *Modelling in database Mangement Sys tems.* Amsterdam, North-Holland, 1976. p. 269-92.