Recuperação de informação

P. BAXENDALE, Editor

Um modelo relacional de dados para Grandes bancos de dados compartilhados

EF CODD
Laboratório de Pesquisa IBM, San Jose, Califórnia

Os futuros usuários de grandes bancos de dados devem ser protegidos contra ter que saber como os dados são organizados na máquina (o representação interna). Um serviço de alerta que fornece essas informações não são uma solução satisfatória. Atividades de usuários nos terminais e a maioria dos programas de aplicativos deve permanecer não afetado quando a representação interna dos dados é alterada e mesmo quando alguns aspectos da representação externa são alterados. Mudanças na representação de dados serão frequentemente necessária como resultado de mudanças na consulta, atualização e relatório tráfego e crescimento natural dos tipos de informação armazenada.

Os sistemas de dados formatados não inferenciais existentes fornecem aos usuários com arquivos estruturados em árvore ou rede um pouco mais geral modelos dos dados. Na Seção 1, inadequações desses modelos são discutidos. Um modelo baseado em relações n-árias, uma normal formulário para relações de banco de dados, e o conceito de um universal sublinguagem de dados são introduzidos. Na Seção 2, certas operações ções sobre as relações (além da inferência lógica) são discutidas e aplicado aos problemas de redundância e consistência no modelo do usuário.

PALAVRAS-CHAVE E FRASES: **banco de dados, banco de dados, estrutura de dados, dados organização, hierarquias de dados, redes de dados, relações, derivabilidade, redundância, consistência, composição, junção, linguagem de recuperação, predicado cálculo, segurança, integridade de dados**
CATEGORIAS *CR* : 3,70, 3,73, 3,75, 4,20, 4,22, 4,29

1. Modelo Relacional e Forma Normal

1.1. INTRODUÇÃO

Este artigo está preocupado com a aplicação de elementos teoria da relação mentária para sistemas que fornecem acesso a grandes bancos de dados formatados. Exceto para um papel por Childs [1], a principal aplicação das relações com dados sistemas tem sido os sistemas de resposta a perguntas dedutivas. Levein e Maron [2] fornecem inúmeras referências para trabalhar nesta área.

Em contraste, os problemas tratados aqui são os de *dados independência - a* independência dos programas de aplicação e atividades terminais de crescimento em tipos de dados e mudanças na representação de dados - e certos tipos de *dados inconsistência* que se espera que se torne problemática mesmo em sistemas não-redutores.

A visão relacional (ou modelo) dos dados descritos em Seção 1 parece ser superior em vários aspectos ao gráfico ou modelo de rede [3, 4] atualmente em voga para sistemas inferenciais. Ele fornece um meio de descrever dados com sua estrutura natural apenas - isto é, sem superim- apresentando qualquer estrutura adicional para representação da máquina finalidades. Consequentemente, fornece uma base para um alto nível linguagem de dados que irá render a máxima independência ser- entre os programas, por um lado, e a representação da máquina e organização de dados, por outro.

Uma outra vantagem da visão relacional é que forma uma base sólida para o tratamento de derivabilidade, redundância, e consistência das relações - estes são discutidos na Seção 2. O modelo de rede, por outro lado, gerou um número de confusões, a menos das quais é o engano a derivação de conexões para a derivação de rela- (consulte as observações na Seção 2 sobre ″armadilha de *conexão* ").

Finalmente, a visão relacional permite uma avaliação mais clara do escopo e limitações lógicas do presente formatado sistemas de dados, e também os méritos relativos (de uma lógica ponto de vista) de representações concorrentes de dados dentro de um sistema único. Exemplos dessa perspectiva mais clara são citado em várias partes deste artigo. Implementações de sistemas para apoiar o modelo relacional não são discutidos.

1.2. DEPENDÊNCIAS DE DADOS NOS SISTEMAS ATUAIS

O fornecimento de tabelas de descrição de dados em recentemente de- sistemas de informação desenvolvidos representam um grande avanço em direção ao objetivo de independência de dados [5, 6, 7]. Tais tabelas facilitar a mudança de certas características dos dados representados sentação armazenada em um banco de dados. No entanto, a variedade de características de representação de dados que podem ser alteradas *sem prejudicar logicamente alguns programas de aplicativos* é ainda bastante limitado. Além disso, o modelo de dados com o qual a interação dos usuários ainda está desordenada com propriedades representacionais direitos, particularmente no que diz respeito à representação de col- lições de dados (em oposição a itens individuais). Três de os principais tipos de dependências de dados que ainda precisam a serem removidos são: dependência de ordenação, dependência de indexação e dependência do caminho de acesso. Em alguns sistemas, estes as dependências não são claramente separáveis umas das outras.

1.2.1. *Dependência de pedidos.* Elementos de dados em um banco de dados pode ser armazenado de várias maneiras, algumas envolvendo não se preocupando com o pedido, alguns permitindo que cada elemento para participar de apenas um pedido, outros permitindo cada elemento para participar em vários pedidos. Vamos considerar os sistemas existentes que requerem ou permitem dados elementos a serem armazenados em pelo menos uma ordem total que é intimamente associado ao pedido determinado por hardware de endereços. Por exemplo, os registros de um arquivo relativo as peças podem ser armazenadas em ordem crescente por série de peças número. Esses sistemas normalmente permitem a aplicação gramas para assumir que a ordem de apresentação dos registros de tal arquivo é idêntico (ou subordinado) ao

pedidos armazenados. Esses programas de aplicação que levam vantagem da ordem armazenada de um arquivo provavelmente falhará operar corretamente se por algum motivo for necessário para substituir esse pedido por um diferente. Observações semelhantes espera para um pedido armazenado implementado por meio de ponteiros.

É desnecessário destacar qualquer sistema como exemplo, porque todos os sistemas de informação conhecidos que são comercializados hoje não conseguem fazer uma distinção clara entre ordem de apresentação por um lado e pedido armazenado no outro. Problemas de implementação significativos devem ser resolvido para fornecer este tipo de independência.

1.2.2. Dependência de indexação. No contexto de for- dados emaranhados, um índice é geralmente considerado como um componente orientado para o desempenho da representação de dados ção Tende a melhorar a resposta a consultas e atualizações e, ao mesmo tempo, desacelerar a resposta às inserções e exclusões. Do ponto de vista informativo, um índice é um componente redundante da representação de dados. Se um sistema usa índices em tudo e se é para ter um bom desempenho em um ambiente com mudanças nos padrões de atividade nos dados banco, a capacidade de criar e destruir índices de vez em quando provavelmente será necessário tempo. A questão então surge: Os programas de aplicativos e atividades do terminal podem permanecer invariante à medida que os índices vêm e vão?

Os sistemas de dados formatados atuais levam amplamente diferentes abordagens para indexação. TDMS [7] incondicionalmente pro vides indexação em todos os atributos. O atualmente lançado versão do IMS [5] fornece ao usuário uma escolha para cada arquivo: uma escolha entre nenhuma indexação (o se- hierárquico organização quencial) ou indexação na chave primária apenas (a organização sequencial indexada hierárquica). Em nenhum dos casos é a lógica do aplicativo do usuário dependente do existência dos índices fornecidos incondicionalmente. IDS [8], no entanto, permite que os designers de arquivo selecionem atributos para ser indexado e incorporar índices na estrutura de arquivo por meio de correntes adicionais. Programas de aplicação vantagem significativa do benefício de desempenho destes cadeias dexing devem se referir a essas cadeias pelo nome. Tal pro gramas não funcionam corretamente se essas correntes forem posteriores removido.

1.2.3. *Dependência do caminho de acesso.* Muitos dos existentes sistemas de dados formatados fornecem aos usuários uma estrutura de árvore arquivos ou modelos de rede um pouco mais gerais dos dados. Programas de aplicativos desenvolvidos para trabalhar com esses sistemas sistemas tendem a ser logicamente prejudicados se as árvores ou redes são alterados na estrutura. Segue um exemplo simples.

Suponha que o banco de dados contenha informações sobre as peças e projetos. Para cada peça, o número da peça, o nome da peça, descrição da peça, quantidade disponível e quantidade do pedido são gravados. Para cada projeto, o número do projeto, projeto nome, descrição do projeto são registrados. Sempre que um projeto faz uso de uma determinada parte, a quantidade dessa parte comp atribuídas ao projeto em questão também são registradas. Suponha que o sistema requer que o usuário ou designer de arquivo declare ou definir os dados em termos de estruturas de árvore. Então, qualquer um das estruturas hierárquicas podem ser adotadas para a informação mação mencionada acima (ver Estruturas 1-5).

Estrutura 1. Projetos subordinados às partes

Arquivo	*Segmento*	*Pldds*
F	PAPEL	papel #
		nome da parte
		Descrição parcial
		quantidade disponível
		quantidade sob encomenda
	PROJETO	projeto #
		Nome do Projeto
		Descrição do Projeto
		quantidade comprometida

Estrutura 2. Peças subordinadas a projetos

Arquivo	*Segmento*	*gidds*
F	PROJETO	projeto #
		Nome do Projeto
		Descrição do Projeto
	PAPEL	papel #
		nome da parte
		Descrição parcial
		quantidade disponível
		quantidade sob encomenda
		quantidade comprometida

Estrutura 3. Partes e projetos como pares
Relacionamento de compromisso subordinado a projetos

Arquivo	*Segmen ~*	*Campos*
F	PAPEL	papel #
		nome da parte
		Descrição parcial
		quantidade disponível
		quantidade sob encomenda
G	PROJETO	projeto #
		Nome do Projeto
		Descrição do Projeto
	PAPEL	papel #
		quantidade comprometida

Estrutura 4. Partes e projetos como pares
Relação de Compromisso Subordinada às Partes

Arquivo	*Segmento*	*Fidds*
P	PAPEL	parte #
		Descrição parcial
		quantidade disponível
		quantidade sob encomenda
	PROJETO	projeto #
		quantidade comprometida
G	PROJETO	projeto #
		Nome do Projeto
		Descrição do Projeto

Estrutura 5. Peças, Projetos e
Relacionamento de Compromisso como Pares

F ~ é	*Segmento*	*Campos*
F	PAPEL	papel #
		nome da parte
		Descrição parcial
		quantidade disponível
		quantidade sob encomenda
G	PROJETO	projeto #
		Nome do Projeto
		descrição do projeto
H	COMPROMETER-SE	papel #
		projeto #
		quantidade **comprometida**

Agora, considere o problema de imprimir a peça número, nome da peça e quantidade comprometida para cada peça usado no projeto cujo nome do projeto é "alfa". O raio que representa uma relação n-ária R tem o seguinte propriedades:

(1) Cada linha representa uma n-tupla de R.

as seguintes observações podem ser feitas, independentemente de quais sistema de informação orientado por árvore disponível é selecionado para resolver este problema. Se um programa P for desenvolvido para isso problema assumindo uma das cinco estruturas acima - que é, P não faz nenhum teste para determinar qual estrutura está em vigor fect-then P falhará em pelo menos três dos restantes estruturas. Mais especificamente, se P tiver sucesso com a estrutura 5, falhará com todos os outros; se P tiver sucesso com a estrutura 3 ou 4, ele falhará com pelo menos 1, 2 e 5; se P tiver sucesso com 1 ou 2, ele falhará com pelo menos 3, 4 e 5. O motivo é simples em cada caso. Na ausência de um teste para determinar qual estrutura está em vigor, P falha porque uma tentativa é feito para executar uma referência a um arquivo inexistente (disponível sistemas tratam isso como um erro) ou nenhuma tentativa é feita para execute uma referência a um arquivo que contém as informações necessárias. O leitor que não está convencido deve desenvolver uma amostra programas para este problema simples.

Uma vez que, em geral, não é prático desenvolver aplicativos programas de instalação que testam todas as estruturas de árvore permitidas pelo sistema, esses programas falham quando uma mudança no estrutura torna-se necessária.

Sistemas que fornecem aos usuários um modelo de rede de os dados encontram dificuldades semelhantes. Tanto na árvore quanto casos de rede, o usuário (ou seu programa) é obrigado a explorar uma coleção de caminhos de acesso do usuário aos dados. Faz não importa se esses caminhos estão em correspondência próxima com caminhos definidos por ponteiros na representação armazenada - em IDS a correspondência é extremamente simples, em TDMS é exatamente o oposto. A consequência, independentemente do armazenado representação, é que as atividades e programas terminais sejam vêm dependentes da existência continuada do usuário caminhos de acesso.

Uma solução para isso é adotar a política de que uma vez o caminho de acesso do usuário é definido, não se tornará obsoleto até que todos os programas de aplicativos que usam esse caminho se tornem obsoleto. Essa política não é prática, porque o número de vias de acesso no modelo total para a comunidade de usuários de um banco de dados eventualmente se tornariam excessivamente ampla.

1.3. UMA VISÃO RELACIONAL DOS DADOS

O termo *relação* é usado aqui em sua matemática aceita sentido matemático. Conjuntos dados $ 1, $ 2, • • •, S ~ (não necessariamente distinto), R é uma relação nesses n conjuntos se for um conjunto de n-tuplas, cada uma das quais tem seu primeiro elemento de S ~, seu segundo elemento de $ 2 e assim por diante. 1 Vamos nos referir a $ i como o jº *domínio* de R. Conforme definido acima, R é dito ter *grau n.* Relações de grau 1 são frequentemente chamadas de *unárias,* degree 2 *binário,* grau 3 *ternário* e grau *n n-ário.*

Por razões expositivas, devemos freqüentemente fazer uso de uma representação de matriz de relações, mas deve ser remembro que esta representação particular não é um esparte sensível da visão relacional que está sendo exposta. Um ar-

i Mais concisamente, R é um subconjunto do produto cartesiano $ 1 X S ~ X """ x S ..

(2) A ordem das linhas é irrelevante.
(3) Todas as linhas são distintas.
(4) A ordem das colunas é significativa - ela corresponde responde ao pedido S ~, $ 2, -.-, S ~ do dorede em que R é definido (ver, no entanto, as observações abaixo em ordenado e não ordenado por domínio relações).
(5) A significância de cada coluna é parcialmente con visto, rotulando-o com o nome do correspondente domínio esponjoso.

O exemplo na Figura 1 ilustra uma relação de grau 4, chamado de *fornecimento,* que reflete as remessas em andamento de peças de fornecedores especificados para projetos específicos em quantidades especificadas.

fornecimento (quantidade do projeto da peça do fornecedor)

1	2	5	17
1	3	5	23
2	3	7	9
2	7	5	4
4	1	1	12

FIGO. 1. Uma relação de grau 4

Alguém pode perguntar: se as colunas são rotuladas com o nome de domínios correspondentes, por que a ordenação de columns importa? Como mostra o exemplo na Figura 2, duas cores umns podem ter cabeçalhos idênticos (indicando idênticos domínios), mas possuem significados distintos em relação ao relação. A relação descrita é chamada de *componente.* É um relação ternária, cujos primeiros dois domínios são chamados de *parte* e o terceiro domínio é chamado de *quantidade.* O significado de *com ponente (x, y, z)* é que a parte x é um componente imediato (ou subconjunto) da parte y, e unidades z da parte x são necessárias para montar uma unidade da peça y. É uma relação que joga um papel crítico no problema de explosão de peças.

componente (parte	*papel*	*quantidade)*
1	*5*	*9*
2	*5*	*7*
3	*5*	*2*
2	6	12
3	6	3
4	7	1
6	**7**	**1**

FIG. 2. Uma relação com dois domínios idênticos

É um fato notável que várias informações existentes sistemas (principalmente aqueles baseados em arquivos estruturados em árvore) falham para fornecer representações de dados para relações que têm dois ou mais domínios idênticos. A versão atual de IMS / 360 [5] é um exemplo de tal sistema.

A totalidade dos dados em um banco de dados pode ser vista como um coleção de relações que variam no tempo. Essas relações são de graus variados. Conforme o tempo avança, cada relação n-ária pode estar sujeito à inserção de n-tuplas adicionais, exclusão dos existentes, e alteração de componentes de qualquer um de seus n-tuplas existentes.

Página 4

Em muitos dados comerciais, governamentais e científicos bancos, no entanto, algumas das relações são de degree (um grau de 30 não é incomum). Os usuários devem normalmente não se preocupa em lembrar o domínio pedido de qualquer relação (por exemplo, o *fornecedor* do pedido , depois *parte,* depois *projeto,* depois *quantidade* na relação *oferta).* Assim, propomos que os usuários lidem, não com as relações que são ordenados por domínio, mas com *relacionamentos* que são suas contrapartes de domínio não ordenado. 2 Para conseguir isso, os domínios devem ser identificáveis de forma única, pelo menos em qualquer dada relação, sem usar posição. Assim, onde nomes e números de peça são. Vamos chamar o conjunto de valores representados em algum momento o *domínio ativo* naquele instante.

Normalmente, um domínio (ou combinação de domínios) de um dada relação tem valores que identificam exclusivamente cada elemento mento (n-tupla) dessa relação. Tal domínio (ou combination) é chamada de *chave primária.* No exemplo acima, o número da peça seria uma chave primária, enquanto a cor da peça não seria. Uma chave primária não é *redundante* se for um domínio simples (não uma combinação) ou uma combinação de modo que nenhum dos domínios simples participantes seja

são dois ou mais domínios idênticos, exigimos em cada caso que o nome de domínio seja qualificado por uma *função* distinta *nome,* que serve para identificar o papel desempenhado por aquele domínio na relação dada. Por exemplo, na relação *componente* da Figura 2, a primeira *parte do* domínio pode ser qualificado pelo nome da função sub e o segundo por *super,* então que os usuários podem lidar com o *componente de* relacionamento e *seus domínios - sub.part super.part, quantidade - sem* consideração a qualquer ordem entre esses domínios.

Para resumir, é proposto que a maioria dos usuários deve interagir com um modelo relacional dos dados consistindo em uma coleção de relações que variam no tempo (ao invés de relações). Cada o usuário não precisa saber mais sobre qualquer relacionamento do que seu nome junto com os nomes de seus domínios (função quali- sempre que necessário) .3 Mesmo essas informações podem ser oferecido em estilo de menu pelo sistema (sujeito a segurança e restrições de privacidade) mediante solicitação do usuário.

Normalmente, existem muitas maneiras alternativas em que um re- modelo nacional pode ser estabelecido para um banco de dados. Em a fim de discutir uma forma preferida (ou forma normal), nós deve primeiro introduzir alguns conceitos adicionais (ativos domínio, chave primária, chave estrangeira, domínio não simples) e estabelecer alguns links com a terminologia atualmente em uso na programação de sistemas de informação. No restante de neste artigo, não devemos nos preocupar em distinguir entre re- relações e relações, exceto onde parece vantagens tageful para ser explícito.

Considere um exemplo de um banco de dados que inclui relações ções relativas a peças, projetos e fornecedores. Um rela- ção chamada *parte é* definida nos seguintes domínios:

(1) número da peça
(2) nome da peça
(3) cor da parte
(4) peso parcial
(5) quantidade disponível
(6) quantidade no pedido

e possivelmente outros domínios também. Cada um desses domínios é, com efeito, um conjunto de valores, alguns dos quais ou todos podem ser representado no banco de dados a qualquer instante. Enquanto é concebível que, em algum momento, todas as cores das peças estão presentes ent, é improvável que todos os pesos de parte possíveis, parte

Em termos matemáticos, um relacionamento é uma classe de equivalência de aquelas relações que são equivalentes sob a permutação de domínios (consulte a Seção 2.1.1).

Naturalmente, como acontece com quaisquer dados colocados e recuperados de um computador sistema de computador, o usuário normalmente fará um uso muito mais eficaz dos dados se ele estiver ciente de seu significado.

supérfluo na identificação única de cada elemento. A rela- ção pode possuir mais de um primário não redundante chave. Este seria o caso no exemplo se partes diferentes sempre receberam nomes distintos. Sempre que uma relação tem duas ou mais chaves primárias não redundantes, uma delas é selecionada arbitrariamente e chamada de chave primária daquele re- ção.

Um requisito comum é para elementos de uma relação com referência cruzada de outros elementos da mesma relação ou elemento mentos de uma relação diferente. As chaves fornecem uma função orientada ao usuário meios (mas não o único meio) de expressar tal cruz referências. Chamaremos de domínio (ou combinação de domínio ção) da relação R uma *chave estrangeira* se não for a chave primária de R, mas seus elementos são valores da chave primária de alguns relação S (a possibilidade de S e R serem idênticos não é excluídos). Na relação de *oferta* da Figura 1, a combinação ção do *fornecedor, parte, projeto* é a chave primária, enquanto cada desses três domínios considerados separadamente é uma chave estrangeira.

Em trabalhos anteriores, tem havido uma forte tendência para tratar os dados em um banco de dados como consistindo em duas partes, uma parte que consiste em descrições de entidades (por exemplo, descrições de fornecedores) e a outra parte consistindo de relações ções entre as várias entidades ou tipos de entidades (para exemplo, a relação de *oferta*). Esta distinção é difícil para manter quando alguém pode ter chaves estrangeiras em qualquer relação ção qualquer. No modelo relacional do usuário, há peras não há vantagem em fazer tal distinção (pode haver alguma vantagem, no entanto, quando se aplica conceitos relacionais para representações de máquina do usuário conjunto de relacionamentos).

Até agora, discutimos exemplos de relações que são definido em domínios simples - domínios cujos elementos são valores atômicos (não decomponíveis). Valores não atômicos podem ser discutido dentro da estrutura relacional. Assim, alguns domínios podem ter relações como elementos. Essas relações pode, por sua vez, ser definido em domínios não simples e assim por diante. Por exemplo, um dos domínios em que a relação *em- ployee é* definido pode ser o *histórico de salários. Um* elemento do domínio de histórico de salário é uma relação binária definida no *data* principal e o *salário* do domínio . O domínio do *histórico de salários* é o conjunto de todas essas relações binárias. A qualquer momento existem tantas instâncias da relação de *histórico de salários* no banco de dados, pois há funcionários. Em contraste, lá é apenas uma instância da relação de *funcionário* .

Os termos atribuem e repetem o grupo nos dados presentes terminologia de base é aproximadamente análoga ao domínio simples

Página 5

e domínio não simples, respectivamente. Muito da confusão na terminologia atual é devido à falha em distinguir entre tipo de interpolação e instância (como em "registro") e entre componentes de um modelo de usuário dos dados, por um lado e suas contrapartes de representação de máquina no outro lado (novamente, citamos "registro" como exemplo).

1.4. FORMA NORMAL

Uma relação cujos domínios são todos simples pode ser representada enviado em armazenamento por um homo-coluna bidimensional matriz genética do tipo discutido acima. Um pouco mais estrutura de dados complicada é necessária para uma relação com um ou mais domínios não simples. Por este motivo (e outros a ser citado abaixo) a possibilidade de eliminar o não-simples domínios parece que vale a pena investigar. 4 Existe, de fato, um procedimento de eliminação muito simples, que chamaremos normalização.

Considere, por exemplo, a coleção de relações ex- hibitado na Figura 3 (a). *História de trabalho* e *filhos* não são domínios simples da relação *empregado. A história do salário* é um

Se a normalização conforme descrito acima for aplicável, a coleção não normalizada de relações deve satisfazer o seguintes condições:

(1) O gráfico das inter-relações do não simples domínios é uma coleção de árvores.

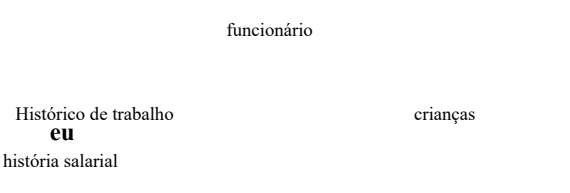
(2) Nenhuma chave primária tem um domínio de componente que seja não simples.

O escritor não conhece nenhum aplicativo que exigiria qualquer relaxamento dessas condições. Outras operações de um tipo de normalização são possíveis. Estes não são discutidos em este papel.

A simplicidade da representação do array que se torna viável quando todas as relações são lançadas na forma normal não é apenas uma vantagem para fins de armazenamento, mas também para comunicação de dados em massa entre sistemas que usam amplamente diferentes representações dos dados. A comunicação forma seria uma versão adequadamente compactada da matriz representação e teria as seguintes vantagens:

(1) Seria desprovido de ponteiros (com valor de endereço ou valor de deslocamento).

domínio não simples do *histórico de empregos* da relação . A árvore em A Figura 3 (a) mostra apenas essas inter-relações dos não domínios simples.



funcionário *(nº homem,* nome, data de nascimento, histórico profissional, filhos)
jobhistory *(jobdate,* title, salaryhistory)
histórico salarial *(data* salarial , salário)
filhos *(nome da criança, ano de* nascimento)

FIGO. 3 (a). Conjunto não normalizado

funcionário ' *(martS,* nome, data de nascimento)
jobhistory ' *(man #, iobdate,* title)
história salarial ' *(homem #, iobdate, data* salarial , salário)
filhos ' *(homem #, nome da criança, ano de* nascimento)

Fro. 3 (b). Conjunto normalizado

A normalização procede da seguinte maneira. Começando com o reinstalação no topo da árvore, pegue sua chave primária e expand cada uma das relações imediatamente subordinadas por inserir este domínio de chave primária ou combinação de domínio. A chave primária de cada relação expandida consiste no chave primária antes da expansão aumentada pela primária chave copiada da relação pai. Agora, dê o fora da relação pai todos os domínios não simples, remova o nó superior da árvore, e repita a mesma sequência de operações em cada subárvore restante.

O resultado da normalização da coleção de relações em A Figura 3 (a) é a coleção da Figura 3 (b). O primário a chave de cada relação está em itálico para mostrar como tais chaves são expandidos pela normalização.

'ME Sanko da IBM, San Jose, reconheceu independentemente o conveniência de eliminar domínios não simples.

(2) Isso evitaria toda dependência de endereçamento hash esquemas.

(3) Não conteria índices ou listas de pedidos.

Se o modelo relacional do usuário estiver configurado na forma normal, nomes de itens de dados no banco de dados podem tomar uma forma mais simples forma do que seria o caso. Um nome geral tomaria uma forma como

R (g) .rd

onde R é um nome relacional; g é um identificador de geração (opcional); r é um nome de função (opcional); d é um nome de domínio. Uma vez que g é necessário apenas quando várias gerações de um dado relação existe, ou está prevista para existir, e r é necessário apenas quando a relação R tem dois ou mais domínios nomeados d, a forma simples *Rd* será freqüentemente adequada.

1,5. ALGUNS ASPECTOS LINGUÍSTICOS

A adoção de um modelo relacional de dados, conforme descrito acima, permite o desenvolvimento de um sub-conjunto de dados universal linguagem baseada em um cálculo de predicado aplicado. Um primeiro- o cálculo do predicado da ordem é suficiente se a coleção de relações está na forma normal. Tal linguagem forneceria um pátio vara de poder linguístico para todas as outras línguas de dados propostas indicadores, e seria um forte candidato para incorporar ding (com modificação sintática apropriada) em uma variedade de linguagens de host (programação, comando ou problema orientado). Embora não seja o objetivo deste artigo descrever tal linguagem em detalhes, suas características salientes seria a seguinte.

Vamos denotar a sublinguagem de dados por R e o host linguagem por H. R permite a declaração de relações e seus domínios. Cada declaração de uma relação identifica o chave primária para essa relação. Relações declaradas são adicionadas ao catálogo do sistema para uso por qualquer membro do usuário comunidade com a devida autorização. H permits apoiando declarações que indicam, talvez menos permanentemente, como essas relações são representadas no armazenamento

Página 6

idade. R permite a especificação para recuperação de qualquer subconjunto de dados do banco de dados. Ação em tal recuperação de recuperação a busca está sujeita a restrições de segurança.

A universalidade do sublangnage de dados reside em seu capacidade descritiva (não sua capacidade de computação). Em grande banco de dados, cada subconjunto de dados tem um número muito grande de descrições possíveis (e sensatas), mesmo quando assuponha (como fazemos) que existe apenas um conjunto finito de funções sub-rotinas às quais o sistema tem acesso para uso em dados de qualificação para recuperação. Assim, a classe de qualificação expressões que podem ser usadas em uma especificação de conjunto devem têm o poder descritivo da classe de bem formados fórmulas de um cálculo de predicado aplicado. É bem conhecido que para preservar este poder descritivo é desnecessário expressar (em qualquer sintaxe escolhida) cada fórmula de o cálculo de predicado selecionado. Por exemplo, apenas aqueles em a forma normal prenex é adequada [9].

Funções aritméticas podem ser necessárias na qualificação ou outras partes das instruções de recuperação. Essas funções podem ser definido em H e invocado em R.

Um conjunto assim especificado pode ser obtido para fins de consulta apenas, ou pode ser retido para possíveis alterações. Tomada de inserções a forma de adicionar novos elementos às relações declaradas com em relação a qualquer pedido que possa estar presente em seus representação da máquina. Exclusões que são eficazes para a comunidade (em oposição ao usuário individual ou sub comunidades) assumem a forma de remoção de elementos de relações esclarecidas. Algumas exclusões e atualizações podem ser trig-

Fornecimento de relação 4-ária da Figura 1, que envolve 5 nomes em notação n-ária, seria representada na forma

P (fornecedor, Q (parte, R (projeto, quantidade)))

em notação binária aninhada e, portanto, emprega 7 nomes.

Outra desvantagem deste tipo de expressão é a sua assimetria. Embora esta assimetria não proíba exploração simétrica, certamente cria algumas bases de interrogatório muito estranho para o usuário expressar (con sider, por exemplo, uma consulta para essas peças e quantidades relacionados a determinados projetos fornecidos por meio de Q e R).

1.6. RELAÇÕES EXPRESSÍVEIS, NOMEADAS E ARMAZENADAS

Associados a um banco de dados estão duas coleções de relações ções: o *conjunto nomeado* e o *conjunto expressável.* O conjunto nomeado é a coleção de todas as relações que a comunidade de os usuários podem se identificar por meio de um nome simples (ou identificador). Uma relação R adquire associação no conjunto nomeado quando um usuário devidamente autorizado declara R; perde adesão quando um usuário devidamente autorizado cancela a declaração de **R.**

O conjunto expressável é a coleção total de relações que podem ser designados por expressões na linguagem de dados. Tal expressões são construídas a partir de nomes simples de relações no conjunto nomeado; nomes de gerações, funções e domínios; conectivos lógicos; os quantificadores do predicado calculus; 6 e certos símbolos de relação constante, como =,>. O conjunto nomeado é um subconjunto do conjunto expressável - geralmente um subconjunto muito pequeno.

gerada por outros, se as dependências de exclusão e atualização forem entre relações especificadas são declaradas em R.

Um efeito importante que a visão adotou em relação aos dados tem no idioma usado para recuperá-lo está na nomenclatura de elementos e conjuntos de dados. Alguns aspectos disso foram dis- discutido na seção anterior. Com a rede normal vista, os usuários muitas vezes serão sobrecarregados com cunhagem e uso mais nomes de relação do que o absolutamente necessário, uma vez que nomes são associados a caminhos (ou tipos de caminho), em vez do que com as relações.

Uma vez que um usuário está ciente de que uma determinada relação está armazenada, ele esperará ser capaz de explorá-lo usando qualquer combinação de seus argumentos como "conhecidos" e o argumento restante mentos como "desconhecidos", porque as informações (como Everest) está lá. Este é um recurso do sistema (ausente em muitos sistemas de informação atuais) que chamaremos *exploração* (logicamente) *simétrica* das relações. Naturalmente, simetria no desempenho não é esperada.

Para apoiar a exploração simétrica de um único binário re- ção, dois caminhos direcionados são necessários. Para uma relação de de- gree n, o número de caminhos a serem nomeados e controlados é n fatorial.

Novamente, se uma visão relacional for adotada em que cada n- relação ária (n> 2) deve ser expressa pelo usuário como um expressão aninhada envolvendo apenas relações binárias (ver Sistema LEAP de Feldman [10], por exemplo) então 2n - 1 nomes devem ser cunhados em vez de apenas n -b 1 com direto notação n-ária, conforme descrito na Seção 1.2. Por exemplo, o

5 Explorar uma relação inclui consulta, atualização e exclusão.

Uma vez que algumas relações no conjunto nomeado podem ser indefinidas pelo tempo combinações pendentes de outras nesse conjunto, é útil para considere associar ao conjunto nomeado uma coleção de declarações que definem essas restrições independentes do tempo. Devemos adiar uma discussão mais aprofundada sobre isso até que tenhamos introduziu várias operações sobre relações (ver Seção 2).

Um dos principais problemas enfrentados pelo designer de um sistema de dados que deve apoiar um modelo relacional para o seu usuários é determinar a classe de representantes armazenados ções a serem apoiadas. Idealmente, a variedade de permitidos representações de dados devem ser apenas adequadas para cobrir o espectro de requisitos de desempenho da cor total seleção de instalações. Uma variedade muito grande leva a sobrecarga necessária no armazenamento e reinterpretação contínua ção das descrições das estruturas em vigor.

Para qualquer classe selecionada de representações armazenadas, os dados o sistema deve fornecer um meio de traduzir as solicitações do usuário expresso na linguagem de dados do modelo relacional em correspondentes - e eficientes - ações na corrente representação armazenada. Para uma linguagem de dados de alto nível, apresenta um problema de design desafiador. No entanto, é um problema que deve ser resolvido - à medida que mais usuários obtêm acesso atual a um grande banco de dados, responsabilidade por pro- fornecendo resposta eficiente e mudanças de rendimento do usuário individual para o sistema de dados.

s Porque cada relação em um banco de dados prático é um conjunto finito em a cada instante do tempo, os quantificadores existenciais e universais pode ser expresso em termos de uma função que conta o número de elementos em qualquer conjunto finito.

Página 7

2. Redundância e consistência

2.1. OPERAÇÕES DE RELACIONAMENTO

Uma vez que as relações são conjuntos, todas as operações de conjunto usuais são aplicável a eles. No entanto, o resultado pode não ser um relação; por exemplo, a união de uma relação binária e um relação ternária não é uma relação.

As operações discutidas abaixo são especificamente para relações ções. Essas operações são introduzidas por causa de sua chave papel em derivar relações de outras relações. Seus a aplicação principal é em sistemas de informações não inferenciais sistemas-tems que não fornecem inferência lógica serviços - embora sua aplicabilidade não seja necessariamente destruída quando tais serviços são adicionados.

A maioria dos usuários não estaria diretamente preocupada com esses operações. Projetistas de sistemas de informação e pessoas preocupada com o controle do banco de dados deve, no entanto, ser mantida bastante familiarizado com eles.

2.1.1. *Permutação.* Uma relação binária tem uma matriz representação com duas colunas. Trocando essas cores umns produz a relação inversa. Mais geralmente, se um permutação é aplicada às colunas de uma relação n-ária, a relação resultante é considerada uma *permutação* do dada relação. Existem, por exemplo, 4! - 24 permuta- ções da relação de *fornecimento na* Figura 1, se incluirmos o permutação de identidade que deixa a ordem das colunas inalterado.

Uma vez que o modelo relacional do usuário consiste em uma coleção de relacionamentos (relações não ordenadas de domínio), permuta- ção não é relevante para tal modelo considerado isoladamente. É, no entanto, relevante para a consideração de armazenadas representações do modelo. Em um sistema que fornece exploração simétrica de relações, o conjunto de consultas respondível por uma relação armazenada é idêntico ao conjunto respondível por qualquer permutação dessa relação. Embora é logicamente desnecessário armazenar uma relação e alguns permutação dele, as considerações de desempenho podem fazer relação ternária que preserva todas as informações em as relações dadas?

O exemplo na Figura 5 mostra duas relações R, S, que são juntáveis sem perda de informações, enquanto a Figura 6 mostra uma junção de R com S. Uma relação binária R é *juntável* com uma relação binária S se existe uma relação ternária U de modo que 7r12 (U) = R e ~ '23 (U) - S. Qualquer um desses ternários relação é chamada de *junção* de R com S. Se R, S são relações binárias ções tais que v2 (R) = ~ i (S), então R é juntável com S. Uma junção que sempre existe nesse caso é o *natural junção* de R com S definido por

$$R * S = \{(a, b, c): R (a, b) \wedge S (b, c)\}$$

onde R (a, b) tem o valor *verdadeiro* se (a, b) é um membro de R e da mesma forma para S (b, c). É imediato que

$$\bullet\ \text{"12 (R, S) = R}$$

e

$$\sim 23 (R, S) = S.$$

Observe que a junção mostrada na Figura 6 é a junção natural de R com S da Figura 5. Outra junção é mostrada na Figura 7

II31 (fornecimento)	*(projeto*	*fornecedor)*
	5	*1*
	5	*2*
	1	*4*
	7	*2*

FIGO. 4 Uma projeção permutada da relação na Figura 1

R *(fornecedor*	*papel)*	*S (projeto parcial)*	
1	*1*	*1*	*1*
2	*1*	*1*	*2*
2	*2*	*2*	*1*

FIGO. 5. Duas relações unificáveis

é aconselhável.

2.1.2. *Projeção.* Suponha agora que selecionamos certas cores umns de uma relação (eliminando os outros) e, em seguida, remova da matriz resultante qualquer duplicação nas linhas. A matriz final representa uma relação que se diz ser um *projeção* da relação dada.

Um operador de seleção 7r é usado para obter qualquer permutação, projeção ou combinação das duas operações ções. Assim, se L é uma lista de índices lc 7 L = / 1, / 2, • • •, ik e R é uma relação n-ária (n _> k), então 7rL (R) é o k-ário relação cuja j-ésima coluna é a coluna ij de R (j = 1, 2, ..., k) exceto que a duplicação nas linhas resultantes é removida. Vigarista- Considere a relação de *fornecimento* da Figura 1. Uma projeção permutada desta relação é exibida na Figura 4. Observe que, neste caso particular, a projeção tem menos n-tuplas do que o relação da qual é derivado.

2.1.3. *Juntar.* Suponha que recebamos duas relações binárias ções, que têm algum domínio em comum. Sob o que circunstâncias podemos combinar essas relações para formar um

7 Ao lidar com relacionamentos, usamos nomes de domínio (função- qualificado sempre que necessário) em vez de posições de domínio.

R * S	*(fornecedor*	*papel*	*projeto)*
	1	*1*	*1*
	1	*1*	*2*
	2	*1*	*1*
	2	*1*	*2*
	2	*2*	*1*

FIGO. 6 A junção natural de R, com S (da Figura 5)

você	*(fornecedor*	*papel*	*projeto)*
	1	*1*	*2*
	2	*1*	*1*
	2	*2*	*1*

FIGO. 7. Outra junção de R com S (da Figura 5)

A inspeção dessas relações revela um elemento (element 1) da *parte* do domínio (o domínio no qual o deve ser feito) com a propriedade que possui mais de um parente em R e também em S. É este elemento

Página 8

que dá origem à pluralidade de junções. Tal elemento no domínio de junção é chamado de *ponto de ambigüidade* com respeito à junção de R com S.

Se ~ r21 (R) ou S é uma função, 8 nenhum ponto de ambigüidade pode ocorrer na junção de R com S. Nesse caso, o natural junção de R com S é a única junção de R com S. Observe que o qualificação reiterada "de R com S" é necessária, porque S pode ser juntável com R (bem como R com S), e este junção seria uma consideração inteiramente separada. Na figura 5, nenhuma das relações R, ml (R), S, ~ r21 (S) é uma função.

A ambigüidade na junção de R com S às vezes pode ser resolvido por meio de outras relações. Suponha que recebamos, ou pode derivar de fontes independentes de R e S, um relação T no *projeto de* domínios e *fornecedor* com o seguinte propriedades ing:

(1) ~ - ~ (T) = ~ -2 (S),

(2) w2 (T) = ~ (R),

(3) *T (j, s) - + 3p (R (S, p) AS (p, j)),*

(4) *R (s, p) ----> 3j (S (p, j) AT (j, s)),*

(5) *S (p, j) ----> 3s (T (j, s) AR (s, p)),*

então podemos formar uma junção de três vias de R, S, T; Aquilo é um relação ternária tal que

~ (v) = R, ~ (U) = S, ~ (V) = T.

Essa junção será chamada *de junção cíclica de 3* para distingui-la de uma *junção linear de 3* que seria uma relação quaternária V tal que

~ r12 (V) = R, ~ r ~ 3 (V) = Z, 7r34 (V) = T.

Embora seja possível que exista mais de uma junção cíclica de 3 (ver Figuras 8, 9, para um exemplo), as circunstâncias sob que isso pode ocorrer implicam em restrições muito mais severas

R (8 p)	s (pi)	T ~ 8)
la	de Anúncios	dl
2a	ae	d2
2b	bd	e2
	ser	e2

FIGO. 8 Relações binárias com pluralidade de 3 ~ oins cíclicos

U (8 pj) U '(8 pi)

y), e T com R (digamos *z),* e, além disso, y deve ser um relativo de x sob S, za relativo de y sob T e xa relativo de z sob R. Observe que na Figura 8 os pontos x = a; y = d; z = 2 tem essa propriedade.

A junção 3 linear natural de três relações binárias R, S, T é dado por

RST = {(a, b, c, d): R (a, b) AS (b, c) *AT (c, d)}*

onde parênteses não são necessários no lado esquerdo porque a junção 2 natural (*) é associativa. Para obter o contrapartida cíclica, apresentamos a operadora .y que pro produz uma relação de grau n - 1 a partir de uma relação de grau n amarrando suas pontas. Assim, se R é uma relação n-ária (n> _ 2), o *empate* de R é definido pela equação

"I (R) = {(al, a2, ..., a, _l): R (al, a2, ..., a ~ -l, an) A al = an}.

Podemos agora representar a junção 3 cíclica natural de *R, S, T* pela expressão

(RST).

Extensão das noções de 3-join linear e cíclico e suas contrapartes naturais para a junção de n relações binárias (onde n> 3) é óbvio. Algumas palavras podem ser adequadas apropriado, no entanto, quanto à articulação de relações que não são necessariamente binários. Considere o caso de dois relações R (grau r), S (graus s) que devem ser unidas em p de seus domínios (p <r, p <s). Para simplificar, suprepresentam que esses p domínios são os últimos p dos r domínios de R, e o primeiro p dos domínios s de S. Se assim não fosse, nós sempre poderia aplicar permutações apropriadas para torná-lo assim. Agora, pegue o produto cartesiano do primeiro *rp* doprincipal de R e chame esse novo domínio de A. Pegue o carroproduto tesiano dos últimos p domínios de R, e chame isso de B. Considere o produto cartesiano dos últimos domínios *sp* de S e chame isso de C.

Podemos tratar R como se fosse uma relação binária no domínios A, B. Da mesma forma, podemos tratar S como se fosse um birelação nária nos domínios B, C. As noções de linear e 3-join cíclico agora são diretamente aplicáveis. Um aplicativo semelhante proach pode ser feito com as n-joins lineares e cíclicas de n relações de graus variados.

2.1.4. *Composição.* O leitor provavelmente está familiarizado

2ae	2ae
2bd	2bd
2be	2be

FIGO. 9 Duas junções cíclicas de 3 das relações na Figura 8

do que aqueles para uma pluralidade de 2 junções. Para ser específico, o re-
Iações R, S, T devem possuir pontos de ambigüidade com
em relação a unir R com S (diga o ponto x), S com T (diga

8 Uma função é uma relação binária, que é um-um ou muitos-um, mas não um-muitos.

com a noção de composição aplicada às funções. Nós
deve discutir uma generalização desse conceito e aplicá-lo
primeiro para relações binárias. Nossas definições de composição
e a composibilidade são baseadas muito diretamente nas definições
de junção e capacidade de junção fornecidas acima.

Suponha que recebamos duas relações R, S. T é uma *com posição* de R com S se houver uma junção U de R com S tal que T = 7913 (U). Assim, duas relações são composíveis se e apenas se eles forem j oináveis. No entanto, a existência de mais de uma junção de R com S não implica a existência de mais de uma composição de R com S.

Correspondendo à junção natural de R com S é o

Página 9

composição natural 9 de R com S definido por

*RS = ~ 'i ~ (R * S).*

Tomando as relações R, S da Figura 5, seu com natural:
posição é exibida na Figura 10 e outra composição
é exibida na Figura 11 (derivada da junção exibida
na Figura 7).

R. S	*(projeto*	*fornecedor)*
	1	*1*
	1	*2*
	2	*1*
	2	*2*

Fro. 10 A composição natural de R com S (da Figura 5)

T	*(projeto*	*fornecedor)*
	1	*2*
	2	1 ~:

Fro. 11. Outra composição de R com S (da Figura 5)

Quando existem duas ou mais junções, o número de
composições podem ser tão poucos quanto um ou tantos quanto o número
ber de associações distintas. Figule 12 mostra um exemplo de dois
relações que têm várias junções, mas apenas uma composição.
Observe que a ambigüidade do ponto c é perdida na composição de R
com S, por causa de associações inequívocas feitas por meio do
pontos a, b, d, e.

R (parte do fornecedor)		*S (projeto parcial)*	
1	***uma***	***uma***	***g***
1	***b***	***b***	***f***
1	***c***	*c*	*f*
2	*c*	*c*	*g*
2	*d*	*d*	*g*
2 '	*e*	*e*	*f*

Fio. 12. Muitas junções, apenas uma composição

Extensão da composição a pares de relações que são
não necessariamente binário (e que pode ser de diferentes de-
grees) segue o mesmo padrão da extensão de pares
aderir a tais relações.

A falta de compreensão da composição relacional levou
vários designers de sistemas no que pode ser chamado de
armadilha de conexão. Esta armadilha pode ser descrita em termos de
seguinte exemplo. Suponha que cada descrição do fornecedor seja
ligados por ponteiros às descrições de cada parte fornecida
por esse fornecedor, e cada descrição da peça é similarmente
ligado às descrições de cada projeto que usa aquele
papel. Uma conclusão é agora tirada que é, em geral, er-
roneous: ou seja, se todos os caminhos possíveis são seguidos de
um determinado fornecedor por meio das peças que ele fornece para os projetos
usando essas partes, obter-se-á um conjunto válido de todos os projetos
fornecidos por esse fornecedor. Essa conclusão é correta
apenas no caso muito especial em que a relação de destino seja
entre projetos e fornecedores é, de fato, a compensação natural
posição das outras duas relações - e devemos normalmente
adicione a frase "para sempre", porque isso geralmente é im-

2.1.5. *Restrição.* Um subconjunto de uma relação é uma relação.
Uma maneira pela qual uma relação S pode agir em uma relação R para
gerar um subconjunto de R é através da *restrição de* operação
de R por S. Esta operação é uma generalização da restrição
de uma função para um subconjunto de seu domínio, e é definido
do seguinte modo.

Sejam L, M listas de índices iguais de tamanho que
L = / 1, / 2, "", ik, M = jl, j2, "", jk onde k ~ grau
de R e k ~ grau de S. Então a restrição L, M de R por
S denotado *RdMS* é o subconjunto máximo R 'de R tal que

~ L (R ') = ~ M (S).

A operação é definida apenas se a igualdade for aplicável
entre elementos de ~ 'h (R) por um lado e ~' h (S) por
o outro para todo h = 1, 2, • • •, k.

As três relações R, S, R 'da Figura 13 satisfazem a equação
ção *R '= R ¢ ~, 3) [¢ i, 2iS.*

R 0 p J)	S (pj)	R '(spj)
1 a A	**a A**	**1 a A**
2 a A	**c B**	**2 a A**
2 a B	**b B**	2 b B
2 b A		
2 b B		

FIGO. 13. Exemplo de restrição

Agora estamos em posição de considerar várias aplicações
dessas operações nas relações.

2.2. REDUNDÂNCIA

A redundância no conjunto nomeado de relações deve ser dis-
distinguido da redundância no conjunto armazenado de representação
ções. Estamos principalmente preocupados aqui com o primeiro.
Para começar, precisamos de uma noção precisa de derivabilidade para
relações.

Suponha que 0 seja uma coleção de operações sobre relações e
cada operação tem a propriedade de que, a partir de seus operandos,
produz uma relação única (assim, a junção natural é elegível, mas
juntar não é). Uma relação R é *O-derivável* de um conjunto S de rela-
ções se existe uma sequência de operações das col-
seleção 0 que, para sempre, produz R de membros de S.
A frase "para sempre" está presente, pois estamos tratando
com relações que variam no tempo, e nosso interesse é em derivabil-
que se mantém por um período de tempo significativo. Para o
conjunto nomeado de relacionamentos em sistemas não inferenciais, ele ap-
peras que uma coleção adequada 01 contém o seguinte
operações: projeção, junção natural, empate e restrição.
A permutação é irrelevante e a necessidade de composição natural
não ser incluído, porque pode ser obtido tomando um natural
junte-se e, em seguida, uma projeção. Para o conjunto armazenado de representantes
ções, uma coleção adequada de operações incluiria
permutação e operações adicionais relacionadas com sub-
estabelecer e fundir relações, ordenar e conectar
seus elementos.

2.2.1. *Redundância Stronq.* Um conjunto de relações é *fortemente redundante* se contiver pelo menos uma relação que possua

aplicado em reivindicações relativas a técnicas de seguimento de caminho.

9 Outros escritores tendem a ignorar composições diferentes do na-
estrutural e, consequentemente, referir-se a esta composição em particular como
a composição - consulte, por exemplo, "Topologia Geral" de Kelley.

Página 10

ampla, a coleção de relações consiste apenas no seguinte
relação de formação:

funcionário (número de série, nome, gerente #, managername)

com *serial #* como chave primária e *gerenciador #* como estrangeiro
chave. Vamos denotar o domínio ativo por, ~, e supor
que

Em (nº gerente) c A, (nº de série)

e

Em (managername) C At (name)

para sempre t. Neste caso, a redundância é óbvia: o
o *managername do* domínio é desnecessário. Para ver que é um
redundância forte, conforme definido acima, observamos que

~ 34 (funcionário) = ~ '12 (funcionário) l [lm (funcionário).

No segundo exemplo, a coleção de relações inclui um
relação S descrevendo fornecedores com chave primária s #, um re-
lação D que descreve departamentos com chave primária d #, a
relação J que descreve projetos com chave primária j #, e o
seguintes relações:

P (s #, d #, ...), Q (s #, j #, .-.), R (d #, j #, ...),

onde em cada caso .-. denota domínios diferentes de s #, d #,
j #. Vamos supor que a seguinte condição C é conhecida por
manter independente do tempo: departamento de suprimentos do fornecedor
d (relação P) se e somente se o fornecedor fornecer algum projeto
j (relação Q) à qual d é atribuído (relação R). Então nós
pode escrever a equação

~ '12 (P) = 71-12 (Q). 71 "21 **(R)**

e, portanto, exibem uma grande redundância.

Uma razão importante para a existência de forte re-
Dundâncias no conjunto nomeado de relacionamentos é o usuário con-
venience. Um caso particular disso é a retenção de semi-
relacionamentos obsoletos no conjunto nomeado de modo que os antigos pro-
gramas que se referem a eles pelo nome podem continuar a funcionar corretamente
retamente. Conhecimento da existência de fortes redundâncias
no conjunto nomeado permite que um sistema ou banco de dados administre
maior liberdade na seleção do representante armazenado
para lidar de forma mais eficiente com o tráfego atual. Se o
redundâncias fortes no conjunto nomeado são refletidas diretamente
em redundâncias fortes no conjunto armazenado (ou se outro
redundâncias são introduzidas no conjunto armazenado), então, gen-
de modo geral, o espaço de armazenamento extra e o tempo de atualização são
consumido com uma possível queda no tempo de consulta para alguns
consultas e em carga nas unidades centrais de processamento.

2.2.2. *Redundância fraca.* Um segundo tipo de redun-
dança pode existir. Em contraste com a redundância forte, não é
caracterizado por uma equação. Uma coleção de relações é
fracamente redundante se contiver uma relação que tenha um projeto
ção que não é derivada de outros membros, mas está em
todas as vezes uma projeção de *alguma* junção de outras projeções de
relações na coleção.

Podemos exibir uma redundância fraca tomando o segundo
exemplo (citado acima) para uma redundância forte, e como-
supondo agora que a condição C não é válida o tempo todo.

As relações ~ 12 (P), ~ r12 (Q), "12 (R) são relações l ° complexas
com a possibilidade de pontos de ambigüidade ocorrendo a partir de
de vez em quando na união potencial de quaisquer dois. Debaixo
nestas circunstâncias, nenhuma delas é derivável da
outros dois. No entanto, existem restrições entre eles,
uma vez que cada um é uma projeção de alguma junção cíclica dos três
eles. Uma das redundâncias fracas pode ser caracterizada
pela afirmação: para sempre, ~ 12 (P) é *alguma* composição
de ~ -12 (Q) com "~ '21 (R). A composição em questão pode
ser natural em algum momento e não natural em
outro instante.

De um modo geral, redundâncias fracas são inerentes a
as necessidades lógicas da comunidade de usuários. Eles não são
removível pelo sistema ou administrador de banco de dados. Se
eles aparecem, eles aparecem no conjunto nomeado e
o conjunto armazenado de representações.

2.3. CONSISTÊNCIA

Sempre que o conjunto nomeado de relações é redundante em
em qualquer sentido, devemos associar a esse conjunto uma coleção de
declarações que definem todas as redundâncias que detêm
independente do tempo entre as relações dos membros. Se o
sistema de informação carece - e provavelmente irá - de
informações semânticas de cauda sobre cada relação nomeada,
não pode deduzir as redundâncias aplicáveis ao nomeado
definir. Pode, ao longo de um período de tempo, fazer tentativas de
induzir as demissões, mas tais tentativas seriam falhadas
lible.

Dada uma coleção C de relações que variam no tempo, um as-
conjunto associado Z de declarações de restrição e um instantâneo
valor V para C, devemos chamar o estado *(C, Z, V) consistente*
ou *inconsistente de* acordo com V satisfaz ou não satisfaz Z.
Por exemplo, dadas as relações armazenadas R, S, T junto com
a declaração de restrição *'% '12 (T)* é uma composição de
7r ~ 2 (R) com 7r ~ 2 (S) ", podemos verificar de vez em quando que
os valores armazenados para R, S, T satisfazem essa restrição. Um al-
goritmo para fazer esta verificação examinaria os dois primeiros
colunas de cada um de R, S, T (de qualquer maneira que eles são representados
enviado no sistema) e determinar se

(1) Tente (T) = ~ -i (R),

(2) ~ (T) = ~ (S),

(3) para cada par de elementos (a, c) na relação ~ 2 (T)
existe um elemento b tal que (a, b) está em ~ -12 (R)
e (b, c) está em 7r12 (S).

Existem problemas práticos (que não vamos discutir
aqui) ao tirar um instantâneo de uma coleção
de relações, algumas das quais podem ser muito grandes e altamente
variável.

É importante notar que a consistência, conforme definido acima
é uma propriedade do estado instantâneo de um banco de dados, e
é independente de como esse estado surgiu. Assim, em
particular, não há distinção feita com base em
se um usuário gerou uma inconsistência devido a um ato de
omissão ou um ato de comissão. Exame de um simples

10 Uma relação binária é complexa se nem ela, nem o seu inverso for um
função.

exemplo mostrará a razoabilidade disso (possivelmente convencional) para a consistência.

Suponha que o conjunto nomeado C inclui as relações S, J, D, P, Q, R do exemplo na Seção 2.2 e que P, Q, R possuem as redundâncias fortes ou fracas descritas nele (no caso particular agora em consideração, não importa que tipo de redundância ocorra). Mais distante, suponha que em algum momento t o estado do banco de dados seja consistente e não contém nenhum projeto j de modo que o fornecedor 2 forneça o projeto j e j é atribuído ao departamento 5. Consequentemente, não há elemento (2, 5) em ~ -i2 (P). Agora, um usuário apresenta o elemento (2, 5) em ~ ri2 (P) inserindo alguns apropriados elemento ate em P. O estado do banco de dados agora é inconsistente. A inconsistência pode ter surgido de um ato de omissão são, se a entrada (2, 5) estiver correta, e houver um projeto j de modo que o fornecedor 2 fornece j e j é atribuído a departamento 5. Neste caso, é muito provável que o usuário pretende, em um futuro próximo, inserir elementos em Q e R que terá o efeito de introduzir (2, j) em ~ r12 (Q) e (5, j) em 7rI2 (R). Por outro lado, a entrada (2, 5) pode ter sido defeituoso. Pode ser que o usuário pretende inserir algum outro elemento em P - um elemento cuja inserção transformaria um estado consistente em um estado consistente. A questão é que o sistema irá normalmente não tem como resolver esta questão sem interrogando seu ambiente (talvez o usuário que criou a inconsistência).

Existem, é claro, várias maneiras possíveis em que um sistema pode detectar inconsistências e responder a elas. Em uma abordagem, o sistema verifica possíveis inconsistências ency sempre que ocorrer uma inserção, exclusão ou atualização de chave. Naturalmente, essa verificação tornará essas operações mais lentas. Se uma inconsistência foi gerada, os detalhes são registrados internamente, e se não for corrigido dentro de alguns intervalo de tempo, seja o usuário ou alguém responsável por a segurança e integridade dos dados são notificadas. Outro abordagem é conduzir a verificação de consistência como um lote operação uma vez por dia ou com menos frequência. Entradas causando o inconsistências que permanecem no estado do banco de dados em verificar o tempo pode ser rastreado se o sistema principal mostra um diário de todas as transações que mudam de estado. este última abordagem certamente seria superior se poucos não ocorreram inconsistências transitórias.

2.4. RESUMO

Na Seção 1, um modelo relacional de dados é proposto como um base para proteger os usuários de sistemas de dados formatados de as mudanças potencialmente perturbadoras na representação de dados causados pelo crescimento do banco de dados e mudanças no tráfego. Uma forma normal para a coleção variável no tempo de relação navios é introduzido.

Na Seção 2, operações sobre relações e dois tipos de redundância são definidas e aplicadas ao problema de manter os dados em um estado consistente. Isso está vinculado a tornar-se um sério problema prático à medida que mais e mais dife- diferentes tipos de dados são integrados em comum bancos de dados.

Muitas questões são levantadas e deixadas sem resposta. Para exemplo, apenas algumas das propriedades mais importantes de o sublinguagem de dados na Seção 1.4 são mencionados. Nenhum os detalhes puramente lingüísticos de tal língua, nem o problemas de implementação são discutidos. No entanto, o o material apresentado deve ser adequado para experientes programadores de sistemas para visualizar várias abordagens. Isto também se espera que este artigo possa contribuir para uma maior pré- cisão no trabalho em sistemas de dados formatados.

Reconhecimento. Era CT Davies, da IBM Pough- Keepie que convenceu o autor da necessidade de dados independência em futuros sistemas de informação. O autor deseja agradecer a ele e também a FP Palermo, CP Wang, EB Altman e ME Senko da IBM San Jose Re- Laboratório de pesquisa para discussões úteis.

RECEBIDO EM SETEMBRO DE 1969; REVISADO EM FEVEREIRO DE 1970

REFERÊNCIAS

1. CHILDS, DL Viabilidade de uma estrutura de dados teórica definida - uma estrutura geral baseada em uma definição reconstituída de relação. Proc. IFIP Cong., 1968, North Holland Pub. Co., Amsterdã, p. 162-172.
2. LEVEIN, RE, AND MARON, ME Um sistema de computador para execução de inferência e recuperação de dados. *Com. ACM 10,* 11 (novembro de 1967), 715–721.
3. BACHMAN, CW Software para processamento de acesso aleatório. *Datamation* (abril de 1965), 36-41.
4. McGEE, WC Processamento de arquivo generalizado. Em *Re anual visualizar na Programação Automática 5,* 13, Pergamon Press, Nova York, 1969, pp. 77-149.
5. Sistema de gerenciamento de informações / 360, descrição do aplicativo Manual de instalação H20-0524-1. IBM Corp., White Plains, NY, Julho de 1968.
6. GIS (Sistema de Informação Generalizado), Descrição do Aplicativo- Manual de instalação H20-0574. IBM Corp., White Plains, NY, 1965.
7. BLEIER, RE Tratamento de estruturas de dados hierárquicas no Sistema de gerenciamento de dados compartilhados por tempo SDC (TDMS). Proc. ACM 22nd Nat. Conf., 1967, MDI Publications, Wayne, Pa., Pp. 41-49.
8. IDS Reference Manual GE 625/635, GE Inform. Sys. Div., Pheonix, Ariz., CPB 1093B, fevereiro de 1968.
9. CHURCH, A. *Uma introdução à lógica matemática I.* Prince- ton U. Press, Princeton, NJ, 1956.
10. FELDMAN, JA, E ROVNEa, PD An Algol-based associ- linguagem ativa. Stanford Artificial Intelligence Rep. AI-66, 1º de agosto de 1968.

[]