INTRODUÇÃO AOS MODELOS DE SISTEMAS ADMINISTRADORES DE BASE DE DADOS

RENÉ LAPYDA

TESE APRESENTADA AO INSTITUTO DE MATEMÁTICA E ESTATÍSTICA DA UNIVERSIDADE DE SÃO PAULO, PARA OBTENÇÃO DO GRAU DE MESTRE EM MATEMÁTICA APLICADA

ORIENTADOR: Prof. Dr: VALDEMAR WAINGORT SETZER

Maio de 1977.

A meus pais,

à Leila,

ao Nelson

Gostaria de agradecer ao Prof. Dr. V. W. Setzer por ter aceitado orientar este trabalho, por tê-lo feito de maneira altamente valiosa para mim e por ter se excedido nesta função, dispondo-se a ouvir e discutir mesmo as dúvidas não diretamente ligadas ao assunto da dissertação e não poupando incentivos para que os obstáculos fossem enfrentados.

O meu reconhecimento para Joana D'Arc Silva e Manoel Peres Garcia do Setor de Publicações do IME-USP, pela afabilidade com que colaboraram comigo.

R. L.

S U M Á R I O

No Capítulo I faz-se uma apresentação dos sistemas de informação computarizados, através de uma perspectiva histórica paralela à dos sistemas de computação propriamente ditos, isto é, equipamentos e sistemas de programação. O aparecimento dos Sistemas Administradores de Base de Dados (SABD) é situado como conseqüência dessa evolução. No Capítulo II, os principais conceitos e a terminologia relativos aos SABD são apresentados através de uma estrutura constituída de modelos: os modelos externos, o modelo geral, o modelo conceitual, o modelo interno e o modelo físico. Nos demais capítulos, alguns modelos são descritos e comentados, tendo como referência a estrutura do capítulo II: o modelo Entidade-Relacionamento (de Chen), os modelos do DIAM (de Senko e outros) e o modelo Relacional (de Codd).

A B S T R A C T

Chapter I presents the computerized information systems, through a historical perspective, in parallel with that of the computing systems as such, that is, hardware and software. The appearence of Data Base Management Systems is viewed as a consequence of that historical evolution. Chapter II covers the main concepts and terminology of DBMS, using a model structure, namely: the conceptual, the general, the external, the internal and the physical models. In the remaining chapters, some models are described and commented, following the structure used in chapter II, that is: Chen's Entity-Relationship Model, the DIAM models of Senko and others, and Codd's Relational Model.

Í N D I C E

C A P Í T U L O I

SISTEMAS DE INFORMAÇÃO - EVOLUÇÃO HISTÓRICA - APARECIMENTO DOS SISTEMAS ADMINISTRADORES DE BASE DE DADOS *

I - INTRODUÇÃO

Chamaremos de organismo, neste trabalho, a todo conjunto de pessoas, recursos materiais, aparelhos, etc. reunidos para desempenhar alguma função, para atingir algum objetivo definido. Os membros desse conjunto são os componentes do organismo. Então, qualquer empresa, instituição, associa-

---

* Não se pode dizer que haja, em Português, um nome consagrado para "Data Base Management Systems". "Data Base" tem sido comumente traduzido para "Banco de Dados". Um dos motivos pelos quais não será usado esse nome é a utilização de "Banco de Dados" (aí sim, de maneira adequada) também como tradução de "Data Bank", que os autores de lingua inglesa distinguem de "Data Base". "Data Bank" seria o repositório geral dos dados de um organismo, independentemente de outras considerações (v.g., relações entre os dados, formas de armazenamento, etc.), enquanto que "Data Base" possui características mais definidas, conforme será visto.

Preferiu-se, então, "Base de Dados".

Foi escolhido, também, "Sistemas Administradores", em lugar de, v.g., "Sistemas Gerenciais" ou "Sistemas Administrativos", porque reflete melhor o fato de que os sistemas devem administrar a Base de Dados, enquanto que os dois últimos podem dar a idéia de sistemas que administram o organismo, com a utilização de Bases de Dados.

ção, enfim, qualquer organização administrativa, educacional, social, econômica, etc. é um organismo.

Nenhum organismo pode funcionar sem um sistema de informação. Cada componente do organismo, para exercer sua função específica, necessita de informações que podem ser obtidas de algum outro componente, podem ser fruto de sua própria elaboração ou podem ser obtidas do ambiente onde o organismo exerce suas atividades.

Essas informações variam em complexidade, de acordo com a função que o componente exerce. Assim, por exemplo, um encarregado das salas de aula de uma Universidade, utiliza informações tais como: o horário em que uma determinada sala deve ser aberta ou fechada, a quantidade de cadeiras que ela deve conter e assim por diante, informações que podem ser obtidas, por exemplo, do seu chefe imediato. Nesse caso, a visão que esse funcionário tem do sistema de informação da Universidade se resume a esse chefe. Já um Chefe de Departamento necessita receber e obter informações em muito maior quantidade * e de fontes muito mais variadas (funcionários do Departamento, professores, direção do Instituto a que pertence o Departamento, Centro de Processamento de Dados, outras Universidades, etc.). Para este componente, o "seu" sistema de informação é muito mais amplo e variado. Evidentemen

---

* Por ora, "informação", "quantidade de informação" e "sistema de informação" estão sendo usados com seus sentidos não-técnicos usuais.

te, não é raro dois ou mais componentes necessitarem de certas informações em comum (v.g., um professor precisa saber o número de alunos que assistirão às suas aulas - para planejar o curso, por exemplo -, enquanto que a administração da Universidade necessita da mesma informação por motivos contábeis, administrativos, estatísticos, etc.).

Um Sistema Administrador de Base de Dados (SABD), quando incorporado a um organismo, passa a fazer parte do sistema de informação desse organismo.

Para iniciar a apresentação do que seja um SABD e qual o seu papel no sistema de informação de um organismo, vamos precisar melhor os conceitos de dado, informação e sistema de informação.

Analogamente a [SENK 73], podemos dizer que sistema de informação é aquele capaz de registrar, elaborar, manter e fornecer informações.

A distinção entre dado e informação é sutil e podemos ilustrá-la através de uma analogia com as linguagens naturais, conforme faz, também, [LANG 75]. Nessas linguagens, informações são comunicadas através de sentenças, que são formadas por combinações de palavras. Cada palavra isolada, em geral, não transmite informação, embora possa ter um sentido. A palavra "cão", por exemplo, embora transmita a idéia do animal, não carrega nenhuma informação.

Já a sentença "meu cão é preto" em primeiro lugar individualiza "meu cão" entre todos os cães possíveis e, em

segundo lugar, especifica a cor "preta" entre todas as cores possíveis, constituindo então informação.

A analogia é feita dizendo-se que dado está para sentença assim como informação está para a mensagem contida na sentença.

Assim, "meu cão é preto" e "my dog is black" são dados distintos contendo, basicamente, a mesma informação.

Em outras palavras, dado é o fato bruto, não interpretado, enquanto informação pressupõe um contexto já estruturado onde ela se insere. Por exemplo, para uma pessoa que conheça apenas a lingua inglesa, o dado "meu cão..." existe mas não transmite informação, pois ela não possui o contexto linguístico onde a frase possa se encaixar.

Voltando aos sistemas de informação, concluímos que os dados são constituintes fundamentais desses sistemas, na medida em que é a partir do recebimento, armazenamento, organização, classificação, ordenação, etc. dos dados que estes sistemas podem cumprir as funções de registro, elaboração, manutenção e fornecimento de informação.

Um subconjunto dos sistemas de informação é o dos sistemas de informação computarizados, onde existem tarefas executadas automáticamente. É entre os elementos desse subconjunto que se inserem os SABD, cujo surgimento pode ser melhor compreendido à luz de uma abordagem da evolução histórica dos sistemas de informação e de sua interação com a evolução histórica dos sistemas computacionais ("hardware" e

"software").

Para esta abordagem tomamos como referência o modelo proposto em [BENJ 72] para a evolução dos sistemas de informação.

Esta evolução pode ser melhor compreendida quando dividida em fases. Fixar-nos-emos apenas nas fases posteriores ao aparecimento dos sistemas computacionais.

II - FASES DE DESENVOLVIMENTO

1ª fase

(corresponde, aproximadamente, ao início da década de 60)

A introdução do computador nos sistemas de informação se caracterizou, no início, pela mecanização de funções específicas, como a elaboração de folhas de pagamento, o controle de materiais, etc. A paisagem típica era a de uma porção de pequenos sistemas praticamente isolados entre si. Já se notava, porém, uma tendência ao aproveitamento, por um novo sistema, dos dados armazenados por aplicações anteriores.

Do ponto de vista computacional, foi uma época caracterizada por:

- o programador tinha que entender o computador ("hardware") para escrever seus programas (em linguagem de máquina ou "assembler").
- o "software" existente cuidava apenas de operações entra

- arquivos sequenciais, estrutura lógica dos dados praticamente idêntica à organização física, isto é, os programas definiam os dados praticamente com a mesma disposição em que eram armazenados fisicamente.
- a organização dos dados era otimizada tendo em vista uma única aplicação; havia grande redundância de dados entre programas.
- mudanças na organização dos dados ou nas unidades de armazenamento forçavam alterações nos programas.

Sintetizando, nessa etapa a mecanização visou apenas substituir o trabalho humano em certas funções, buscando maior velocidade e exatidão na execução das tarefas.

2ª fase

(fins da década de 60)

A mecanização de funções, que caracterizou a 1ª fase, engendrou o interesse por uma maior racionalização no desenvolvimento de sistemas de informação. As consequências disso foram, principalmente:

1. A centralização do desenvolvimento de sistemas: no início da fase anterior era comum os próprios usuários serem encarregados do desenvolvimento do sistema. Com a crescente computarização dos sistemas, a necessidade de racionalização levou à formação de grupos especialmente voltados para projeto e desenvolvimento. Sistemas semelhantes passaram a ser projetados de maneiras semelhantes e a centralização trouxe consi

go

2. A tendência para integração de sistemas: a possibilidade de coordenar os pequenos sistemas em grandes sistemas multifuncionais ficou patente e tornou-se possível graças

ao crescente "know-how" em projeto e desenvolvimento de sistemas e

ao aumento muito grande nas capacidades de "hardware" e "software" surgidas com as novas gerações de computadores.

O conhecimento adquirido trouxe, até mesmo, tentativas de formalização do processo de desenvolvimento de sistemas e, consequentemente, a idéia de automatizar o próprio controle desse processo. Programas para a construção de fluxogramas e documentação de sistemas são exemplos de tentativas nesse sentido.

Todo esse aumento de "know-how" trouxe, paradoxalmente, um aumento na complexidade dos sistemas. Os projetos passaram a ser mais demorados; mais demoradas e custosas passaram a ser a implementação e a substituição dos sistemas, a inércia aumentou também, trazendo consigo os problemas de manutenção, crescimento e reorganização — os sistemas não podiam mais ser jogados fora após pouco tempo de uso, passou a ser necessário mantê-los em atividade pelo maior prazo possível.

Do ponto de vista técnico, as principais características eram:

- deu-se o aparecimento efetivo dos sistemas operacionais, promovendo a independência do programador em relação ao conhecimento íntimo da máquina.
- métodos de acesso sequencial e aleatório foram incorporados ao "software".
- as mudanças nas unidades de armazenamento já não afetam os programas.
- surgem as primeiras medidas para proteção dos dados.
- a organização dos dados continuava voltada para aplicações específicas, havendo ainda grande redundância de dados.

Estas características técnicas respondem principalmente a necessidades surgidas na 1ª fase. Facilitar a vida do programador, libertando-o da necessidade de um conhecimento mais profundo da máquina ou do "lay-out" dos dispositivos de armazenamento, permitindo-lhe dedicação à lógica dos programas. Surgem, também, algumas características exigidas pelos novos sistemas multifuncionais: uma separação, ainda que ligeira, entre estrutura lógica e organização física dos dados; uma preocupação com a segurança de arquivos a serem compartilhados por vários usuários; etc.

Em síntese, então, a 2ª fase se caracteriza pela crescente importância dos sistemas mecanizados na administração. As tomadas de decisão já exigem informações que somente esses sistemas podem fornecer. Estruturas e organizações mais eficientes, nos dados, passam a merecer maior atenção nos projetos.

3ª fase

(anos 70)

O desejo de mecanização atinge os escalões administrativos mais altos.

Os sistemas de um organismo podem ser separados em três níveis, de acordo com sua posição em relação à administração:

- sistemas operativos: são os sistemas básicos para o funcionamento do organismo.
- sistemas de controle ou decisão: são os que regulam os sistemas operativos.
- sistemas estratégicos: são usados para tomadas de decisões gerais e formulação de políticas de controle.

As 1ª e 2ª fases de desenvolvimento dos sistemas computarizados se restringiram ao nível dos sistemas operativos (simples ou multifuncionais), com restritos exemplos ao nível de sistemas de controle. O reconhecimento dos três níveis de sistemas engendrou a tentativa de integração do nível estratégico.

Um bom sistema de informação, além de consolidar o maior número possível de sub-sistemas funcionais em um único sistema multifuncional integrado, deve agora, também, promover a integração dos dados de maneira a satisfazer as necessidades de informação dos distintos níveis administrativos.

Esse alargamento de objetivos trouxe as seguintes conseqüências:

1. Estruturas de dados adequadas para o funcionamento dos sistemas operativos, por exemplo, podem se mostrar inadequadas para prover a necessidade de informação dos sistemas estratégicos. Os sistemas operativos, em geral, utilizam os dados de um modo estático, com poucas alterações no decorrer do tempo. Mesmo que alterações ocorram, elas são de pequeno porte, não exigem uma estrutura de dados muito flexível para atendê-las. Já os sistemas estratégicos são, por sua própria natureza, "surpreendentes". Necessitam, frequentemente, fazer consultas que nunca foram formuladas anteriormente, seja pela variedade de problemas que podem surgir nesse nível, seja como consequência de mudanças no "staff" administrativo. Considera-se um bom sistema de informação não aquele que foi "educado" por um administrador antigo, durante certo tempo, para responder às suas perguntas, mas sim um sistema flexível a ponto de satisfazer a curiosidade de um novo administrador, desejoso de conhecer o organismo que vai dirigir. Esta multiplicidade de necessidades é uma das causas principais do estágio em que se encontram os projetos de SABD e muitas vezes não é apontada como tal.

2. O grau de interação humana com o sistema também é afetado pela multiplicidade de níveis. A interação a nível de sistema operativo é relativamente bem definida, já que, em geral, esses sistemas exigem pouca intervenção humana, o que não acontece a nível estratégico, onde os sistemas (de planejamen

to, por exemplo) se modificam constantemente. O sistema de informação deve, evidentemente, facilitar a intervenção humana, sempre que necessária. Outro aspecto a ser considerado é o da aproximação entre a parte mecanizada do sistema de informação e componentes do organismo sem conhecimentos de programação de computadores.

3. Uma conseqüência, que talvez venha a se revelar a mais importante com relação aos SABD, é o caráter fundamental que passa a ser atribuído aos dados nos projetos de sistemas de informação. Na 1ª fase, tudo girava em torno das funções (ou processos, aplicações). Os dados eram organizados obedecendo às necessidades das funções a serem mecanizadas. Os projetos de sistemas multifuncionais sentiram a necessidade de dar um tratamento diferente aos dados. Se uma nova aplicação devia utilizar dados já registrados para aplicações existentes, era patente o desperdício em se duplicar os arquivos apenas porque pequenas diferenças existiam. Da mesma forma, se a organização e armazenamento dos dados de um programa tinham que ser alterados, era de interesse manter o programa inalterado, desde que a estruturação lógica desses dados não fosse modificada. Com o crescimento dos sistemas, uma porcentagem cada vez maior dos esforços de programação passou a ser dedicada à manutenção de programas antigos, à tentativa de mantê-los em uso apesar de mudanças nos arquivos. É nesse ponto que entra um conceito fundamental quando se fala em SABD e que será analisado amplamente mais adiante: a chama

da independência de dados. Basicamente, independência de dados é a separação entre a estrutura lógica dos dados e a sua organização nos dispositivos de armazenamento, de modo que mudanças na última não acarretem, obrigatoriamente, mudanças na outra. Com essa separação, um programador poderia, por exemplo, estruturar um registro lógico em seu programa, contendo campos de diferentes registros físicos. Se esses registros físicos fossem modificados, o programa não sentiria os efeitos, desde que nem o conteúdo, nem estruturas mais fundamentais (os campos, v.g.) fossem alterados. Além de facilitar a manutenção, o uso desse conceito implica numa diminuição na redundância de dados, o que, além da economia, contribui para um melhor controle da consistência dos dados. De fato, se um mesmo dado é armazenado em mais de um lugar, quando um programa atualiza seu valor em um dos locais, o sistema deve ter recursos para atualizar os demais, sob pena de tornar a base de dados inconsistente.

Essa foi a contribuição da 2ª fase (sistemas multifuncionais) na criação de esforços para uma definição mais racional dos dados de um sistema, visando integrá-los da maneira mais estável possível. Com a 3ª fase, com a necessidade de se estruturar os dados tendo em vista os níveis administrativos, o problema se tornou mais complexo. Uma mesma organização de dados deveria ser capaz de atender o nível operativo (para o qual havia sido dirigida toda a experiência anterior)

e o nível estratégico, conforme foi dito em 1. . Os sistemas estratégicos necessitam, em geral, de informações sobre o organismo a que pertencem. O caráter imprevisível da maioria das informações necessárias levou à concepção de que a estrutura de dados deveria refletir, ao máximo, a própria estrutura do organismo. Essa concepção se ajusta perfeitamente às exigências dos sistemas multifuncionais na busca de estruturas de dados comuns a várias funções, pois, em geral, o relacionamento que estas funções tem entre si reflete suas posições na estrutura geral do organismo. O projetista do sistema deve ter um conhecimento profundo do organismo e de seu relacionamento com o ambiente, e esse conhecimento deve ser expresso na estruturação de uma base de dados sobre a qual funcionará o sistema de informação e, consequentemente, o próprio organismo. Essa estruturação deve ser anterior a e independente dos processos que a utilizarão. A orientação do projeto se transfere das aplicações para os dados. O reconhecimento dessa prioridade da base de dados deu origem a um novo papel, necessário tanto no projeto como na manutenção do sistema de informação, qual seja o do administrador de dados ou administrador da base de dados, centralizador do trabalho de administração dos dados do sistema, refletindo bem a importância que agora assume esse trabalho.

Vejamos ainda as características técnicas ligadas à 3ª fase e os prováveis desenvolvimentos futuros:

- diferentes arquivos lógicos podem ser associados ao mesmo arquivo físico.
- os mesmos dados podem ser recuperados através de diferentes caminhos de acesso (dados dispostos, em sequência, por exemplo, podem ser manipulados pelos programas como se estivessem armazenados segundo alguma forma de "hashing").
- o "software" é enriquecido pela administração de dados.
- a organização física dos dados é independente das definições lógicas presentes nos programas de aplicação e, portanto, pode ser modificada para aumentar a eficiência global sem provocar alterações nos programas.
- possibilidade de estruturas de dados extremamente complexas, sem que os programas se tornem, necessariamente, mais complexos.
- possibilidade de recuperação de dados por chaves múltiplas (mais de um campo).
- desenvolvimento de técnicas de tele-processamento para sistemas de acesso simultâneo e sistemas de resposta e atualização imediatas ("real-time").
- linguagens (para definição e manipulação de dados) adequadas; atenção para usuários programadores ou não.
- (futuro) base de dados dinâmica, isto é, evolui sem grandes custos.
- (futuro) meios efetivos para avaliação do desempenho e instrumentos para o administrador de dados fazer as alterações necessárias, de acordo com a avaliação.

- (futuro) proteção adequada para a segurança e a integridade de dados (definidas mais adiante).

III - CONCLUSÃO

Essa apresentação sob o ponto de vista evolutivo visou dar elementos para uma melhor compreensão do contexto em que surgiram os SABD. Essa evolução, no entanto, deve ser devidamente apreciada. A divisão em fases é nítida tão somente no que se refere à vanguarda do desenvolvimento dos sistemas e o mesmo acontece com as épocas citadas. Um quadro mais de acordo com a realidade mostra, na verdade, uma interpenetração das várias fases, ou, melhor ainda, cada fase englobando a(s) fase(s) anterior(es).

Em nosso país, por exemplo, a predominância ainda é de sistemas de informação de 1ª fase, embora já se tenha muitos exemplos de 2ª fase e alguns raros organismos estejam tentando ingressar na 3ª fase.

Por outro lado, em relação aos sistemas computacionais se nota uma proporção inversa, marcando um contraste que não pode passar desapercebido. Para a predominância de sistemas de informação de 1ª fase, existe uma proliferação de sistemas computacionais de recursos mais modernos. Só recentemente começou-se a notar um aumento de interesse em relação a equipamentos de menor porte (mini-computadores, por exemplo).

E não deve ser esquecido o lado dos recursos humanos. A passagem de uma fase para outra mais avançada foi fruto também de uma expansão da potencialidade técnica do elemento humano envolvido. Quando os recursos disponíveis em uma fase são explorados ao máximo por técnicos de alto nível, a passagem para um sistema mais avançado se torna natural, diminuindo o perigo dos traumas comuns a uma modernização artificial, praticamente imposta pelas técnicas de "marketing", onde a formação de elementos qualificados vem a reboque (quando vem!) e não na vanguarda das mudanças.

A orientação deste trabalho consiste em apresentar os Sistemas Administradores de Base de Dados em suas origens, em seus objetivos e naquilo que trouxeram (ou ainda pretendem trazer) de mais inovador no campo dos Sistemas de Informação.

O assunto é bastante controverso e o número de pontos sobre os quais existe um consenso generalizado é bastante restrito. As tentativas de padronização ([CODA 71], [ANSI 75], etc.) parecem longe de promover uma união de opiniões, até mesmo com relação à terminologia.

Mais ainda, o tema é por demais extenso, tendo ramificações em várias outras áreas como linguagens, compilação, sistemas operacionais, programação, etc.

O capítulo seguinte procura dar uma visão global de um SABD, fixando os principais conceitos e aproveitando para introduzir a terminologia.

Nos demais capítulos são desenvolvidas, em maior detalhe, algumas das propostas que mais têm contribuído para que os novos conceitos venham a se concretizar.

C A P Í T U L O II

O SISTEMA ADMINISTRADOR DE BASE DE DADOS

Em caráter de síntese: os SABD tiveram seu primeiro alento no desejo de integração dos dados relevantes a um organismo, com o objetivo de reduzir a redundância nos dados utilizados pelos usuários do sistema de informação do organismo.

Os SABD deveriam ser instrumentos para que um conjunto dinâmico (no sentido de mudança e crescimento) de dados fosse compartilhado por uma comunidade dinâmica (no sentido de diversidade de necessidades) de usuários.

Essa "definição" implica, entre outras coisas, em:

a) permitir que o acesso à base de dados se dê através de várias linguagens, desde as linguagens de programação mais conhecidas até linguagens adequadas para usuários casuais (e não-programadores), sem deixar de incluir linguagens que permitam ao administrador de dados definir e manter a base de dados lógica e fisicamente.

b) permitir que cada usuário possa ter uma visão própria dos dados que pretende utilizar, sem que isto implique em re-

dundância e de modo que o desempenho global do sistema seja o melhor possível.

c) oferecer independência de dados, ou seja, proteção ao usuário contra interferências em sua utilização dos dados, provocadas por alterações estranhas a essa utilização.

d) oferecer proteção adequada quanto à segurança e integridade dos dados e quanto à privacidade (melhor explicado em seguida).

A integração de dados através de um SABD traz consigo novos problemas, relacionados à manutenção desses dados, e que podem ser classificados em problemas de segurança, integridade e privacidade. Quando a maior parcela de responsabilidade sobre os dados é retirada de cada usuário e centralizada na função do administrador de dados, esses problemas não são triviais.

Manter a segurança de uma base de dados significa evitar acessos não autorizados. Um usuário que "entrega" seus dados ao SABD, evidentemente deseja que eles não possam ser consultados por elementos indevidos. Mais ainda, ele quer que seus dados estejam protegidos contra tentativas, deliberadas ou não, de destruição ou adulteração.

Por outro lado, a integridade de uma base de dados pode ser ameaçada por:

- falhas mecânicas do sistema.
- tentativas de acesso simultâneo a determinada porção da base de dados.

- falhas em programas que atualizam dados.
- e assim por diante.

O SABD deve então conter instrumentos para sanar estes problemas, que podem ser desastrosos para o sistema de informação do organismo.

Já as questões relacionadas com a privacidade têm a ver mais com a parte "filosófica" dos SABD, pois estes, com sua capacidade de reunir e fornecer informações, podem ser instrumentos de violação da liberdade individual, na medida em que contribuam para aumentar o controle sobre a vida de um cidadão.

Seguindo a orientação expressa em [ANSI 75], vamos precisar melhor a idéia de sistema de informação.

Um sistema de informação consiste dos seguintes conceitos:

- mensagens
- procedimentos
- recursos
- processos
- registros.

Toda comunicação de dados relacionados com entrada/saída real (cartões, impressora, terminais) e com transferências de dados entre processos faz parte da administração de mensagens.

As atividades relacionadas com a preparação de um

programa até (exclusive) a fase de execução, formam a administração de procedimentos.

As atividades ligadas com os recursos físicos, ou sejam, alocação de memória, "swapping", ordenação de prioridades, etc., constituem a administração de recursos.

Tudo o que for relacionado com o estado de um processo, ou seja, variáveis locais, áreas de trabalho, contadores de instrução e assim por diante, é reunido sob o nome de administração de processos.

Finalmente, temos o que é chamado de administração de dados: registros, arquivos, campos e suas descrições, métodos de acesso, organização de arquivos, linguagens para usuários, técnicas de aplicação ("mapping"), técnicas de proteção e recuperação, etc.

Seria ótimo poder discutir os assuntos relacionados com a administração de dados através de conceitos e proposições matematicamente definidos. O estado atual da pesquisa nessa área, porém, ainda é bastante incipiente no que diz respeito à formalização, estando distante da precisão desejada. Usando o precedente de [SENK 73], preferimos, ao invés de restringir este trabalho aos poucos aspectos que podem ser convenientemente descritos do ponto de vista matemático, procurar construir uma descrição mais completa e apropriada à compreensão do assunto, utilizando conceitos não tão precisamente definidos, mas deixando idéias sobre caminhos de investigação a serem seguidos em busca de melhores definições.

O problema se apresenta ainda mais complexo quando se sabe que não há sequer um consenso generalizado sobre a terminologia a ser empregada.

[CODA 71] e [ANSI 75] são os dois trabalhos mais conhecidos como tentativas de padronização, tanto da estrutura como do vocabulário relacionado com os SABD. [CODA 71] acabou se revelando uma padronização apenas de um modelo particular de SABD (o chamado "modelo de rede"), assim mesmo sujeita a muitas críticas. O trabalho do "Study Group" da ANSI/SPARC é mais recente e o seu alcance ainda não pode ser bem avaliado, mas em seu próprio relatório ficam claras as dificuldades encontradas. O "Study Group" achou por bem não aprofundar a tentativa de padronização até o nível da descrição dos componentes de um SABD, preferindo se restringir a uma descrição funcional das interfaces entre esses componentes, isto é, sem entrar em considerações tecnológicas.

Neste trabalho, sempre que possível, utilizaremos uma terminologia próxima à proposta pelo "Study Group", no intuito de evitar propor mais uma terminologia.

A base de dados de um organismo deve ser compartilhada entre usuários que têm distintas visões em relação a ela. Conforme já foi visto, o que há de comum entre todos os usuários possíveis é justamente o organismo ao qual estão ligados. O ponto de partida para a obtenção de um terreno comum de trabalho deve, portanto, ser o organismo.

A figura 1 apresenta uma visão esquemática da ma-

neira como é feita essa integração.

O organismo é parte da realidade, do mundo real. Uma descrição completa, formal, precisa e não ambígua do mundo real é um dos objetivos da Ciência em geral e se afigura como um problema de extrema complexidade, ainda que recortemos essa realidade e tentemos representar uma porção arbitrariamente isolada do resto: um organismo ou uma parte de um organismo.

Para interagir com essa realidade, o ser humano executa processos mentais de redução e simplificação, visando obter esquemas racionais que lhe permitam agir logicamente. Estes esquemas variam de pessoa para pessoa e são função de uma série de fatores: dos mecanismos de percepção, do histórico de formação das estruturas mentais, até mesmo da motivação individual em relação ao objeto. O vale de um rio pode ser visualizado como um sistema de componentes de uma barragem em potencial (por um engenheiro) ou como um conjunto de movimentos e cores (para um pintor), por exemplo [FROM 62]. Numa empresa, um psicólogo pode ter uma visão do relacionamento afetivo entre os funcionários que seja absolutamente transparente às demais pessoas.

Para que se possa obter uma base de dados comum, é imprescindível partir-se de uma visão do organismo que seja comum a todas as pessoas envolvidas, que conte com uma concordância geral.

Essa visão unificada constitui um modelo simplifica

do, onde o organismo pode ser representado por um conjunto de frases simples (em linguagem natural), envolvendo tão somente um número finito de entidades, propriedades, relacionamentos, etc. e haja um consenso geral sobre a avaliação de cada uma destas frases como verdadeira ou falsa. Não vamos precisar, por enquanto, os conceitos de entidade, propriedade e relacionamento, sendo suficientes as idéias intuitivas que, em geral, se tem destas palavras.

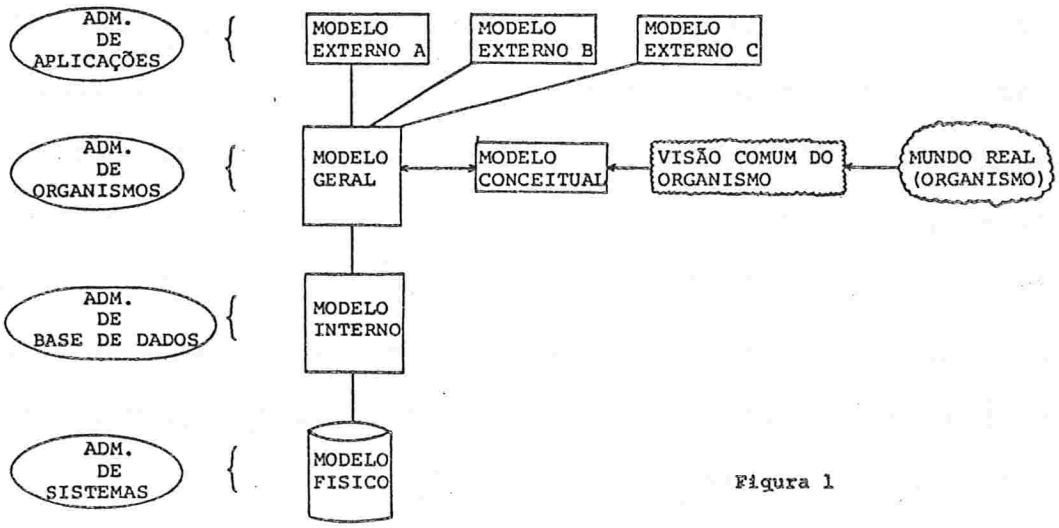


Figura 1

Ex: "o aluno de nome José Silva cursa a disciplina MAP-211", "125322 é número de aluno", "uma das propriedades de um funcionário é a sua idade", "um aluno é identificado univocamente pelo seu número de R.G.", etc.

Esse modelo ainda é inadequado para um tratamento formal e, em geral, é apenas subentendido na prática, não sendo realmente formulado. O que se dá é uma passagem direta para um modelo conceitual do organismo, mais adequado para o tratamento formal.

O modelo conceitual deve ser independente de suas possíveis realizações, da representação em computador, por exemplo (embora a representação em computador esteja sujeita ao modelo conceitual). Pelo esquema da figura 1, pode-se notar que este modelo está associado a uma função humana, a do administrador do organismo (as funções humanas podem ser executadas por um ou mais indivíduos; um mesmo indivíduo, ou grupo de indivíduos, pode ser encarregado de mais de uma função humana; a separação das funções visa chamar a atenção para a importância de se considerar a maior ou menor independência que deve existir entre elas, o que está diretamente ligado aos níveis de independência de dados). Esta função deve representar as necessidades e desejos do organismo como um todo, sem se prender a aspectos particulares de usuários. O modelo de dados elaborado por esta função será o modelo central do sistema, ao qual se deverão referir os demais. A sua estabilidade deve depender apenas dos aspectos reais do

organismo, isto é, ele só deve ser alterado como resposta a alterações realmente introduzidas no organismo e não como resposta a necessidades particulares.

Por exemplo, uma nova representação de entidade, nesse modelo, só deveria ser introduzida quando uma nova entidade fosse, efetivamente, adicionada a ou identificada no organismo, enquanto que o desejo de se melhorar a "perfomance" de processamento não deveria interferir no modelo. Este modelo já deve conter, também, as fórmulas de autorização e proteção, segundo a visão do administrador do organismo, isto é, a definição dos "quem pode ter acesso a que porção de dados".

Os objetos desse modelo não precisam ser materializados fisicamente. Ele assume um papel de guia para a formulação e manutenção dos outros modelos.

O modelo conceitual deve exprimir restrições de ordem semântica que, atualmente, são difíceis de serem incorporadas em outros modelos (por motivos técnicos e econômicos).

Alguns exemplos de fatos relativos a isso são:

a) se uma pessoa é identificada como sendo do sexo feminino, em nenhum outro momento poderá ser identificada como sendo do sexo masculino.
b) se uma pessoa tem 25 anos de idade, em épocas posteriores somente poderá ter idade maior ou igual a 25 anos.
c) quantidades de maçãs e quantidades de alunos não devem ser somadas (geralmente).

d) o nome de uma pessoa não pode ser armazenado em uma base de dados para identificar, por exemplo, uma cidade.

e) se consta da base de dados que

1. Fornecedor F fornece a peça PE

2. Projeto PR recebeu 15 unidades da peça PE,

não deve ser permitido concatenar esses dois fatos a fim de formar

3. Fornecedor F forneceu 15 unidades da peça PE, ao projeto PR,

pois PR pode ter recebido as 15 unidades de vários fornecedores.

O passo seguinte é a formulação do modelo geral, que é uma adaptação do modelo conceitual voltada para dois objetivos principais:

a) servir de referência mais direta do que o modelo conceitual para a formulação dos modelos externos, aqueles efetivamente empregados pelos usuários; em sistemas onde o modelo conceitual não comparece para uso automático (é o caso da grande maioria dos sistemas atuais, onde não há, explicitamente, modelo conceitual), o modelo geral o substitui parcialmente

b) servir de referência mais direta para a formulação do modelo interno, o modelo que já contém informações mais detalhadas sobre a representação dos dados.

O modelo geral deve conter apenas objetos que serão

efetivamente materializados e deve ser coerente com o modelo conceitual.

O modelo geral é o que se pode chamar, essencialmente, de um modelo lógico de dados. Ele mostra como a base de dados está estruturada, que entidades são representadas, como são descritas essas entidades (por propriedades, por relacionamento com outras entidades, etc.), que tipos de relacionamentos entre entidades existem (hierarquia, rede, etc.), se os valores que representam propriedades são numéricos, qual seu comprimento, se a base de dados é formada por um conjunto de relações matemáticas e assim por diante.

O modelo não especifica quais os caminhos de acesso para os dados, se através de índices, de busca sequencial, "hashing", se é possível se ter acesso a uma hierarquia diretamente em qualquer nível ou somente a partir do nível mais alto, etc. Não especifica também, evidentemente, como os dados estão armazenados nos dispositivos físicos.

O modelo interno de dados é o último nível de representação antes da representação física. Partindo da estrutura lógica, esse modelo apresenta características que orientarão a organização física.

A função humana responsável pelo modelo interno é o administrador de base de dados (o nome "administrador de base de dados" é mais comumente usado englobando também a função do administrador do organismo e, às vezes, a do administrador de aplicações). Enquanto os modelos conceitual e

geral (bem como os modelos externos, como veremos) devem ser formulados tendo em vista uma melhor compreensão da estrutura lógica dos dados, o modelo interno é voltado principalmente para o desempenho, em termos de processamento. Através da formulação e reformulações do modelo interno, o administrador de base de dados deve tentar obter a organização que proporcione o melhor desempenho global do sistema, embora possa haver diminuição de desempenho em algumas aplicações. O modelo interno é o que apresenta alterações mais frequentes, pois o desempenho global depende da utilização que se faz do sistema. A organização do modelo interno deve se modificar a fim de aumentar a eficiência dos processos que mais sobrecarreguem o sistema e essa sobrecarga varia com o tempo.

O modelo interno pode ser visto como um espaço linear (apenas para fixar melhor a idéia), ilimitado, de endereçamento (a unidade de deslocamento nesse espaço pode ser bits, bytes, registros, trilhas, cilindros, etc.) e os dados estão distribuídos por esse espaço. Estão especificados os caminhos de acesso (índices, índices secundários, fórmulas de "hashing", cadeias de apontadores, etc.) e características tais como comprimentos, esquemas de codificação (binário, decimal, caracteres) de campos, forma de aglutinação de campos (os campos referentes a um registro do modelo geral, por exemplo, podem estar distribuídos por diferentes registros internos).

O modelo interno tem que ser coerente com o modelo

geral e, portanto, com o modelo conceitual.

O modelo físico é aquele onde comparecem as características dos dispositivos físicos que armazenam os dados: a maneira como os dados são dispostos nos volumes, como é tratado o "overflow", interação entre memórias primária e secundária. É prática generalizada que o projeto do modelo físico seja feito de maneira que sua manipulação possa ser entregue ao sistema operacional.

Os modelos externos representam as visões que os usuários ou conjuntos de usuários têm dos dados. Representam estruturas lógicas que não devem ir de encontro ao modelo geral, embora possam conter variações em relação a ele. Por exemplo, a representação de uma entidade em um modelo externo pode ser obtida pela concatenação de representações de duas ou mais entidades no modelo geral; se, porém, está representado no modelo geral o fato de que "um departamento possui vários funcionários, mas um funcionário pertence a apenas um departamento", não seria aceitável, em um modelo externo, que "um funcionário trabalhe em dois departamentos".

A formulação e manutenção dos modelos externos é feita pelos administradores de aplicações, cada um ligado a um grupo de aplicações que possam compartilhar o mesmo modelo externo.

A dinâmica de funcionamento do sistema (uma vez definidos os modelos) pode ser entendida através da figura 2:

1) O usuário faz uma solicitação de dados ao SABD. Para essa solicitação, o usuário toma como referência o modelo externo apropriado e utiliza uma extensão da sua linguagem de programação (usuários programadores) ou uma linguagem especial. Em ambos os casos se pode falar em uma Linguagem de Manipulação de Dados (LMD).

2) Através de um componente que faz a transformação entre o modelo externo usado em 1) e o modelo geral, o SABD obtém os objetos do modelo geral que correspondem aos objetos do modelo externo especificados pelo usuário.

3) As especificações, em termos de objetos do modelo geral, são utilizadas por um transformador modelo geral/modelo interno para a obtenção dos endereços (no espaço de endereçamento do modelo interno) dos objetos do modelo interno que correspondem às referidas especificações.

4) Os endereços obtidos em 3) são, então, passados ao componente que manipula o modelo físico (muitas vezes é o próprio sistema operacional) para que sejam feitos os acessos físicos necessários para a obtenção efetiva dos dados.

5) Obtidas as representações físicas dos dados, inicia-se um processo inverso, no qual essas representações são, sucessivamente, transformadas em objetos dos modelos interno, geral e externo, e o resultado é entregue à área de trabalho do usuário (na divisão de dados de um programa, ou na

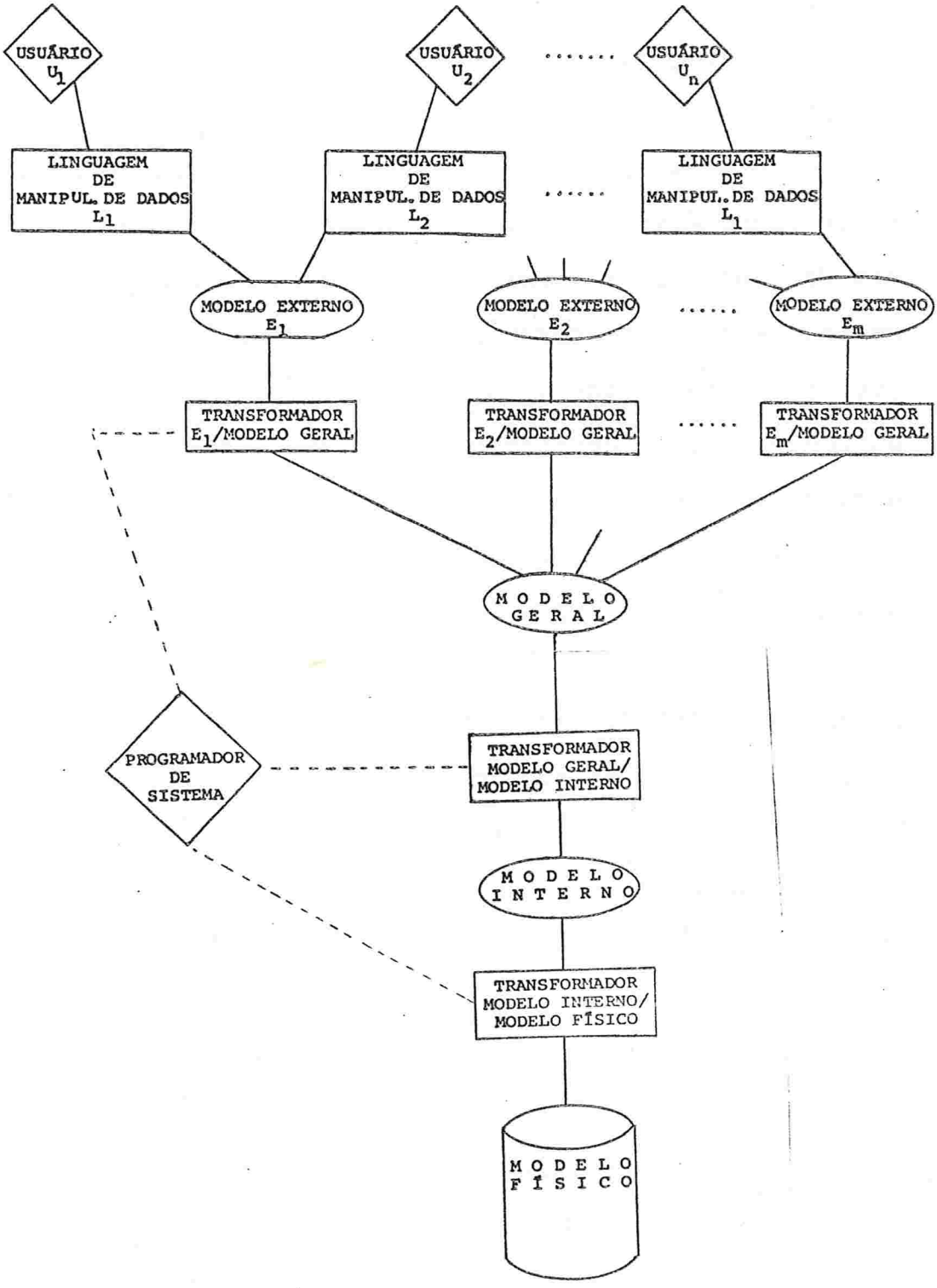

USUÁRIO U_1
USUÁRIO U_2
.....
USUÁRIO U_n
LINGUAGEM DE MANIPUL. DE DADOS L_1
LINGUAGEM DE MANIPUL. DE DADOS L_2
.....
LINGUAGEM DE MANIPUL. DE DADOS L_1
MODELO EXTERNO E_1
MODELO EXTERNO E_2
.....
MODELO EXTERNO E_m
TRANSFORMADOR E_1/MODELO GERAL
TRANSFORMADOR E_2/MODELO GERAL
.....
TRANSFORMADOR E_m/MODELO GERAL
MODELO GERAL
PROGRAMADOR DE SISTEMA
TRANSFORMADOR MODELO GERAL/ MODELO INTERNO
MODELO INTERNO
TRANSFORMADOR MODELO INTERNO/ MODELO FÍSICO
MODELO FÍSICO

Figura 2

tela de um terminal, etc.).

O esquema de funcionamento é fundamentalmente o mesmo para operações de armazenamento de dados.

O esquema acima proposto para a administração de base de dados é voltado quase inteiramente para a independência de dados. Os sistemas existentes atualmente (e, provavelmente, os que aparecerem em futuro próximo), são implementações restritas desse esquema. Uma análise do esquema, à luz do conceito de independência de dados, pode servir de base para a compreensão dos compromissos envolvidos nessas restrições. Sempre que uma característica do esquema é abandonada, o grau de independência é diminuído.

Independência de dados é, basicamente, isolar um usuário dos efeitos estranhos à utilização que faz dos dados, efeitos advindos da evolução do ambiente onde estão inseridos os "seus" dados. Independência de dados não é isolar um programa de alterações em sua lógica interna ou na sua particular visão dos dados. Não é apenas a capacidade de um sistema em fornecer várias visões dos mesmos dados, mas, principalmente, a capacidade de manter essas visões.

"A necessidade de independência de dados não pode ser evitada tentando-se estabelecer e manter compatibilidade, isto é, assegurar que todas mudanças e usos sejam compatíveis entre si. Independência de dados não é uma disciplina; é uma flexibilidade" ([ANSI 75]).

O conceito parte do princípio de que as mudanças

são inevitáveis e a intenção é diminuir o impacto das mudanças, não evitá-las.

As mudanças que ocorrem em um ambiente de dados podem ser classificadas de acordo com os modelos propostos anteriormente:

1) Mudanças nos dispositivos físicos: no atual estágio dos sistemas de computação, a independência em relação a essas mudanças já é obtida, em geral, a nível de sistema operacional.

2) Mudanças no modelo interno - são específicas deste nível as mudanças que visam melhorar o desempenho global, quanto ao processamento, mas o modelo interno pode, também, sofrer alterações como reflexo de mudanças em outros modelos, principalmente no modelo geral:
   - quando dois organismos se fundem, tem que haver uma uniformização das estratégias de armazenamento.
   - o administrador de base de dados resolve introduzir uma certa redundância, a fim de obter um maior isolamento entre áreas de aplicações (para maior eficiência ou maior segurança).
   - o comprimento máximo para um campo interno tem que ser aumentado.
   - mudanças de localização de dados (caminhos de acesso) no modelo interno para melhorar a atualização, recuperação, controle de segurança ou controle de integridade.
   - etc.

Mudanças como as mencionadas podem ser executadas no modelo interno sem que o modelo geral (e consequentemente, os modelos conceitual e externos) seja perturbado. O que deve ser alterado é o transformador modelo geral/modelo interno.

3) Mudanças no modelo geral - são específicas deste nível as mudanças que refletem alterações no modelo conceitual e, portanto, no organismo, mas o modelo geral pode mudar por exigência de modelos externos (por exemplo, quando um modelo externo, a ser criado, necessita de dados presentes no modelo conceitual, mas não no modelo geral):
   - quando dois organismos se fundem, os dois modelos gerais devem ser fundidos em um só, de maneira que os programas existentes possam ser mantidos (desde que seus modelos externos não tenham que ser reformulados).
   - quando duas bases de dados têm que ser fundidas com a descoberta de um relacionamento entre elas.
   - novos tipos de entidades apareceram (ou foram identificadas) no organismo, provocando mudanças no modelo conceitual.
   - novas propriedades são acrescentadas a entidades já existentes ou há uma migração de propriedades (acréscimo de campos ou migração de campos entre registros).
   - etc.

Alterações nos transformadores modelo externo/mode-

lo geral previnem que essas mudanças provoquem alterações nos modelos externos e, portanto, afetem programas que os utilizam.

4) Mudanças em modelo externo - essas mudanças são frutos de mudanças correspondentes na visão particular que uma área de aplicações tem dos dados e, portanto, é impossível que os programas não sejam afetados.

As implementações devem, portanto, basicamente, balancear a complexidade dos transformadores (e a correspondente sobrecarga em processamento) com a perda de independência de dados.

Em um extremo, pode-se ter uma implementação em que há praticamente congruência entre os modelos. Um registro no modelo externo é sempre congruente com um registro no modelo geral, que, por sua vez, é sempre congruente com um registro no modelo interno, por exemplo. Nesse caso pode haver transformações diretas modelo externo/modelo interno, ou mesmo, modelo externo/modelo físico. A eficiência aumenta, mas a flexibilidade desaparece.

Do outro lado, pode-se ter uma implementação que permita um registro do modelo externo corresponder a (partes de) vários registros do modelo geral e permita um registro do modelo geral ser obtido por concatenação vertical e/ou horizontal de registros do modelo interno. Exemplo de concatenação vertical: suponhamos registros representando "funcionários" e um dos campos seja "salário"; pode-se ter um

campo em um registro geral que se refira ao "total de salários", sem que este campo exista no modelo interno. Exemplo de concatenação horizontal: suponhamos campos do modelo geral representando propriedades "pessoais", como "endereço", "esposa", "filhos", etc., de um "funcionário"; estes campos podem estar em registros internos distintos de registros internos onde estejam armazenadas as "propriedades profissionais", como "salário", "departamento", etc.; estes campos podem ser reunidos em um só tipo de registro no modelo geral.

Entre os dois extremos há toda uma gama de variação nas possíveis implementações.

Algumas delas oferecem um bom grau de independência entre o modelo geral e o modelo interno, mas restringem os modelos externos a selecionar tão somente porções do modelo geral, sem poder fazer maiores alterações. É, por exemplo, o caso do modelo de [CODA 71] .

Em outras, impõem-se restrições ao próprio modelo geral ou ao modelo interno, para simplificar os transformadores. Assim, o modelo geral permite apenas tipos selecionados de representações de entidades ou relacionamentos entre entidades, enquanto o modelo interno tem à sua disposição um número selecionado de caminhos de acesso ou formas de representação. Quase todos os sistemas existentes atualmente possuem restrições desse teor.

Alguns sistemas oferecem a possibilidade de "cur-

to-circuitos", isto é, a capacidade de, por exemplo, os usuários poderem manipular diretamente recursos do modelo interno, em seus programas (é o que se chama de "navegação", muito bem descrita em [BACH 73], novamente ao preço da independência.

Em [MART 75], encontramos uma sugestiva ilustração dos graus de independência de dados, através de uma analogia com uma pessoa (o usuário) que quer um táxi (o SABD) para levá-lo a um cinema para assistir a um filme (os dados). No melhor dos casos, ela diria, simplesmente, ao motorista: "- O Poderoso Chefão!", e seria levada ao cinema apropriado, sem que precisasse saber qual o cinema ou onde está localizado. Mesmo que o filme mude constantemente de cinema, o pedido (programa) pode ser feito sempre da mesma forma. Se a pessoa fosse obrigada a pedir pelo nome do cinema: "- Cine Metro!", o pedido teria que ser alterado cada vez que o filme mudasse de cinema, mas a pessoa ainda está desobrigada de saber os endereços dos cinemas. Com um motorista menos esperto, o pedido teria que ser: "- Av. S. João, nº tal" e, portanto, se o Cine Metro mudasse de endereço, o pedido teria que ser reformulado. Finalmente, no pior caso, a pessoa teria que "navegar" : "- ... pegue a 1ª direita, siga em frente,... " e o seu pedido, evidentemente, teria que ser adaptado a cada mudança de mão em ruas que fazem parte do roteiro.

No projeto de um SABD, o ponto de partida é a escolha dos tipos adequados de modelos, para os vários níveis.

Isso não significa, necessariamente, que a cada nível do esquema apresentado deva corresponder exatamente um modelo. Como já foi dito, aquele esquema tem um caráter funcional. Considerações de ordem prática podem levar a projetos onde, por exemplo, dois ou mais níveis formem um componente único (é o caso de um SABD que só permita aos usuários utilizar o modelo geral, sem poder definir modelos externos particulares); outros projetos permitem que atribuições definidas para um determinado nível façam parte de outro (na proposta apresentada em [CODA 71], por exemplo, o modelo geral contém indicações sobre caminhos de acesso aos dados, áreas de armazenamento, etc.); em alguns projetos, um modelo conceitual pode ser definido e armazenado para processamento automático (para verificação da coerência do modelo geral, por exemplo), em outros ele será trabalhado em nível de uma função humana; alguns sistemas podem incorporar a flexibilidade de diferentes transformações modelo-para-modelo (é o caso de um modelo externo que seja muito usado, quando se justificaria, em termos de melhoria de desempenho, a definição de um transformador especial que ligue, diretamente, esse modelo ao modelo interno, sem passar pelo modelo geral) ou, simplesmente, "curto-circuitar" certas transformações.

Essas considerações de ordem prática são feitas, de modo geral, à luz do balanço economia (eficiência, armazenamento) x independência de dados.

A escolha dos modelos deve levar em conta, basica-

mente:

1) Facilidade de uso - os modelos escolhidos devem fornecer clareza compatível com as funções a que se destinam; o modelo interno deve ser de fácil compreensão e utilização para o administrador de base de dados, os modelos externos devem ser dirigidos aos usuários externos, o modelo conceitual deve favorecer a representação dos fatos do organismo e assim por diante. Devem permitir a utilização de linguagens simples para sua definição e manipulação, devem permitir aos usuários aprender rapidamente a representar a sua particular visão dos dados, quais operações podem e não podem fazer (inserções, eliminações, atualizações, etc.).

2) Facilidade de transformações - preservando o máximo de independência de dados, os modelos devem, no entanto, permitir que as transformações entre os vários níveis sejam as menos complexas possíveis, a fim de não comprometer a eficiência e facilitar a manutenção desses processos (é bom lembrar que são as transformações que devem ser alteradas quando se deseja que mudanças em um nível não perturbem os demais).

3) Uma qualidade desejável em um modelo (e que está de acordo com 1) e 2) é a capacidade de representar estruturas e/ou organizações complexas através de composições entre um mínimo de elementos básicos simples e que:

a) a complexidade do modelo seja proporcional à complexidade do que ele representa;

b) que em modelos complexos se consiga isolar porções simples (é uma vantagem, por exemplo, para usuários externos que necessitam de modelos externos simples, retirados de um modelo geral complexo).

A idéia de, nos vários níveis, os modelos serem constituídos de um mínimo de unidades básicas simplifica o problema das transformações, além da própria definição e manipulação dos modelos. Essa idéia será melhor esclarecida na apresentação do DIAM (CAP. IV).

4) Consistência - as regras para definição e manutenção de modelos devem incorporar restrições que impeçam o usuário de executar operações que violem a integridade do modelo. Estas regras podem estar implícitas no modelo, ou podem se tornar explícitas através de mensagens que acusem uma operação indevida.

5) Consultas - este ítem se refere mais aos níveis externo e geral, e diz respeito à flexibilidade do modelo quanto aos tipos de consultas à base de dados. Como já vimos, uma das características mais importantes de um SABD é a capacidade de resposta aos pedidos mais variados. É importante, então, que um modelo sugira ao usuário toda essa gama de variação. Aqui também é desejável que o modelo (e a linguagem de manipulação associada) estabeleça uma cer-

ta proporcionalidade entre a complexidade das operações envolvidas no atendimento ao pedido e a dificuldade de formulá-lo. Por exemplo, seja um modelo onde estejam representadas entidades do tipo "aluno", "professor", "disciplina".

Seja um pedido como "informações sobre o aluno nº 5033" e outro como "nomes dos alunos que cursam a disciplina Map-111". Se este último envolver operações mais complexas que o primeiro, o esquema abaixo é o exemplo de uma forma onde esta diferença estaria clara ao usuário:

PROFESSOR

NOME	ANO DE ADMISSÃO	ENDEREÇO	etc
.	.	.	.
.	.	.	.
.	.	.	.

ALUNO

Nº	NOME	ENDEREÇO	ANO	etc
.	.	.	.	.
.	.	.	.	.
5033	.	.	.	.
.	.	.	.	.
.	.	.	.	.

DISCIPLINA

SIGLA	DEPTO	SEMESTRE	etc
.	.	.	.
.	.	.	.
.	.	.	.

DIVISÃO DE TURMAS

SIGLA - DA - -DISCIPLINA	Nº - DO - -ALUNO	NOME - DO - -PROFESSOR	etc
.	.	.	.
.	.	.	.
MAP-111	.	.	.
.	.	.	.
.	.	.	.
MAP-111	.	.	.
.	.	.	.
.	.	.	.

De fato, esta forma sugere que "informações sobre o aluno nº 5033" podem ser obtidas através das seguintes operações:

- busca do nº 5033 na coluna Nº da tabela ALUNO;
- ler as colunas, no cruzamento com a linha encontrada.

enquanto o segundo pedido envolve:

- busca da sigla MAP-111 na coluna DISCIPLINA da tabela DIVISÃO DE TURMAS;
- cada vez que for encontrada, deve ser anotado o Nº - DE - ALUNO correspondente, na mesma tabela;
- para cada número obtido na operação anterior, buscá-lo na coluna Nº da tabela ALUNO e ler o NOME correspondente.

A decisão fundamental, quanto às características dos modelos, está na escolha do tipo de modelo geral. Ao modelo geral devem estar sujeitos os modelos externos. O tipo esco-

lhido para o modelo geral tem influência decisiva na escolha do tipo de modelo interno e, consequentemente, nas decisões relativas ao armazenamento físico dos dados.

Assim, não é de estranhar que, quando se fala em "modelo de dados", atualmente, se pense em "modelo lógico", ou "modelo do usuário" e que nesse nível estejam mais concentradas as pesquisas e discussões.

O primeiro modelo a ser apresentado é o Modelo Entidade-Relacionamento (Entity-Relationship Model) ([CHEN 76]). É o modelo que mais se aproxima de um modelo conceitual, atualmente, na medida em que fornece mais informações semânticas, embora lhe falte algumas características que seriam desejáveis nesse nível. Através de sua descrição são introduzidas as principais idéias relativas às estruturas lógicas de dados e o modo como algumas noções semânticas são incorporadas.

Os modelos que constituem o DIAM ([SENK 73]) - Modelo Conjunto-de-Entidades (Entity-Set Model), Modelo Cadeia ou Modelo Caminhos-de-Acesso (String Model ou Access-Path Model), Modelo Codificação (Encoding Model) e o Modelo Nível-de-Dispositivo-Físico (Physical Device Level Model) - formam um bom e raro exemplo da concatenação entre os vários níveis de um SABD.

O Modelo Relacional é um tipo característico de modelo geral. É o melhor exemplo da possibilidade de formalização e da utilização de conceitos matemáticos no estudo dos

SABD.

Como já foi dito, as terminologias propostas são inúmeras. A fim de estabelecer um quadro de referência, para a compreensão dos vocabulários ligados aos vários modelos a serem descritos, apresentamos algumas definições, sempre que possível de acordo com [ANSI 75]. Muitas delas se referem a termos já clássicos em computação, mas nem sempre bem definidos. Podem ser utilizadas para os vários modelos ou, mesmo, para o mundo real.

Um objeto é algo do mundo real ou de algum modelo, onde esse "algo" implica em uma ocorrência.

Uma representação é uma imagem ou símbolo para um objeto. Um mapa representa uma cidade. O valor de um campo representa o valor de uma propriedade de uma entidade. Um registro (em algum modelo) pode representar uma entidade.

Um atributo é a representação de uma propriedade de uma entidade ou de um relacionamento (cf. com o "valor de uma propriedade de uma entidade"). Ex: o atributo "projeto em que trabalha" relativo à entidade "funcionário".

Um conjunto-de-valores é a população de valores, dentre os quais podem ser selecionados os valores para um atributo.

Conjunto-ativo-de-valores é a população dos valores que, em um dado momento, representam efetivamente algum fato sobre alguma entidade ou relacionamento.

Um valor é uma ocorrência de uma propriedade de uma entidade ou relacionamento, ou a representação dessa ocorrência em

algum modelo.

Um identificador é uma propriedade ou combinação de propriedades, cujos valores servem de nome para uma entidade ou relacionamento. Nos modelos, a representação de um identificador é uma chave.

Uma classe é uma coleção de objetos similares, no sentido de que possuem o mesmo protótipo.

Um tipo é o protótipo da coleção de objetos similares.

Um relacionamento (no mundo real) é uma conexão entre duas ou mais entidades, duas ou mais coleções de entidades, ou duas ou mais propriedades. Envolve os objetos conectados (ex: entidades "João" e "José"), o caráter da conexão (ex: "pai de"), e a direção da conexão (ex: "João é o pai"), quando houver.

Um relacionamento (em um modelo) é uma conexão entre dois ou mais objetos desse modelo.

Uma associação é uma coleção de zero ou mais objetos, conectados por um relacionamento não dirigido. Ex: os alunos de uma escola.

Uma estrutura é uma coleção de zero ou mais objetos conectados por relacionamentos dirigidos. Ex: uma hierarquia (João é pai de José, que é pai de Carlos e Maria, etc.).

Uma organização é um arranjo de dados voltado para o armazenamento físico (envolvendo, por exemplo, rótulos de volumes, índices, apontadores).

Um campo é o menor objeto identificável por um nome, em um

modelo.

Um grupo é uma associação ou estrutura (possuindo um nome) de campos e/ou grupos, definida para simplificar o endereçamento.

Um registro é uma associação ou estrutura (possuindo um nome) de campos e/ou grupos. Um registro é o objeto física ou logicamente: armazenado, eliminado ou recuperado.

Um complexo é uma associação ou estrutura (possuindo um nome) de registros e/ou complexos, que atua como mecanismo de referência (ex: os "sets" no modelo da [CODA 71], os "segments" do IMS, etc.).

Um conjunto-de-registros é uma associação ou estrutura (possuindo um nome) de registros e/ou complexos. Um conjunto-de-registros é o maior objeto que pode incluir algum outro, em um modelo. Por exemplo, no modelo interno, é o objeto que pode ser aberto ou fechado ("open", "close").

Uma entidade (mundo real) é uma pessoa, lugar, coisa, conceito ou evento, de interesse para o organismo. Seus limites são arbitrários: uma parte de uma entidade pode ser definida separadamente, receber um nome e passar a ser também uma entidade.

Uma propriedade (mundo real) é uma característica de uma entidade (ou de um relacionamento entre entidades). Uma propriedade desempenha um papel na descrição da entidade (ou do relacionamento).

C A P Í T U L O III

O MODELO ENTIDADE - RELACIONAMENTO

O Modelo Entidade-Relacionamento (MER) ([CHEN 76]) parte de uma visão do mundo real onde entidades e relacionamentos são os objetos fundamentais. Com pequenas variações, essa concepção tem sido usada na grande maioria dos trabalhos sobre SABD, ainda que com outras terminologias.

Evidentemente, é impossível obter-se uma descrição completa de todas entidades e relacionamentos, tais como são no mundo real. Impõe-se uma primeira simplificação, na tentativa de tornar "representável" essa visão do mundo, que consiste na escolha das entidades e relacionamentos relevantes para o sistema de informação do organismo, descritos tão somente através de suas propriedades relevantes.

A representação de uma entidade no MER será chamada registro-entidade e a representação de um relacionamento entre entidades será chamada relacionamento.

A escolha dos registros-entidade e dos relacionamentos é um tanto arbitrária. Por exemplo, uma "dupla" no jogo de tênis, pode ser representada por um registro-entidade

ou como o relacionamento entre dois registros-entidade "tenistas". Uma norma que pode, na prática, atenuar esse caráter arbitrário é: se houver registros-entidade do tipo "tenista", no modelo, então, "dupla" é adotada como relacionamento.

Registros-entidade (e_i) são agrupados em conjuntos-de-registros-entidade (E_j) segundo propriedades comuns.

Um conjunto-de-relacionamentos (R_i) é uma relação matemática *

$$\{[e_1, e_2, \ldots, e_n] \mid e_1 \epsilon E_1, e_2 \epsilon E_2, \ldots, e_n \epsilon E_n\}$$

envolvendo n conjuntos-de-registros-entidade e cada tupla é um relacionamento entre n registros-entidade. A cada E_j ($j = 1, 2, \ldots, n$) está associado um papel (ρ_j) que indica a função desempenhada por um $e_i \epsilon E_j$, em cada relacionamento. Ex.: em um relacionamento envolvendo dois "funcionários", um pode ter o papel de "gerente" e o outro "subordinado".

Se, em cada tupla de R_i, cada elemento for denotado por ρ_1/e_1, ρ_2/e_2, etc., então a ordem dos E_j, em R_i, se torna irrelevante.

Informações sobre entidades são representadas, no MER, por pares atributo/valor. Informações sobre relaciona

---

* do ponto de vista estritamente matemático, se um R_i sofrer alguma alteração, uma nova relação estará sendo criada. Para não se precisar definir R_i em função do tempo, contorna-se o problema admitindo-se que, em instantes de tempo distintos, relações distintas podem ter o mesmo nome. Assim, se, por exemplo, R_i sofrer uma atualização, nos instantes seguintes a nova relação terá o nome de R_i.

mentos são representadas por pares papel/registro-entidade e pares atributo/valor.

Um <u>atributo</u> (A_i) é uma função que leva um E_j ou um R_j a um <u>conjunto-de-valores</u> (V_k) ou a um produto cartesiano de conjuntos-de-valores:

$$A_i: \begin{matrix} E_j \\ \text{ou} \\ R_j \end{matrix} \longrightarrow \begin{matrix} V_k \\ \text{ou} \\ V_1 \times V_2 \times \ldots \end{matrix}$$

Nome/João Silva, Número-USP/41523, etc. são exemplos de pares atributo/valor relativos a um conjunto-de-registros-entidade ALUNO.

"Nome" é um atributo que leva ALUNO ao produto cartesiano $V_1 \times V_2$, onde V_1 = NOMES e V_2 = SOBRENOMES (conjuntos-de-valores).

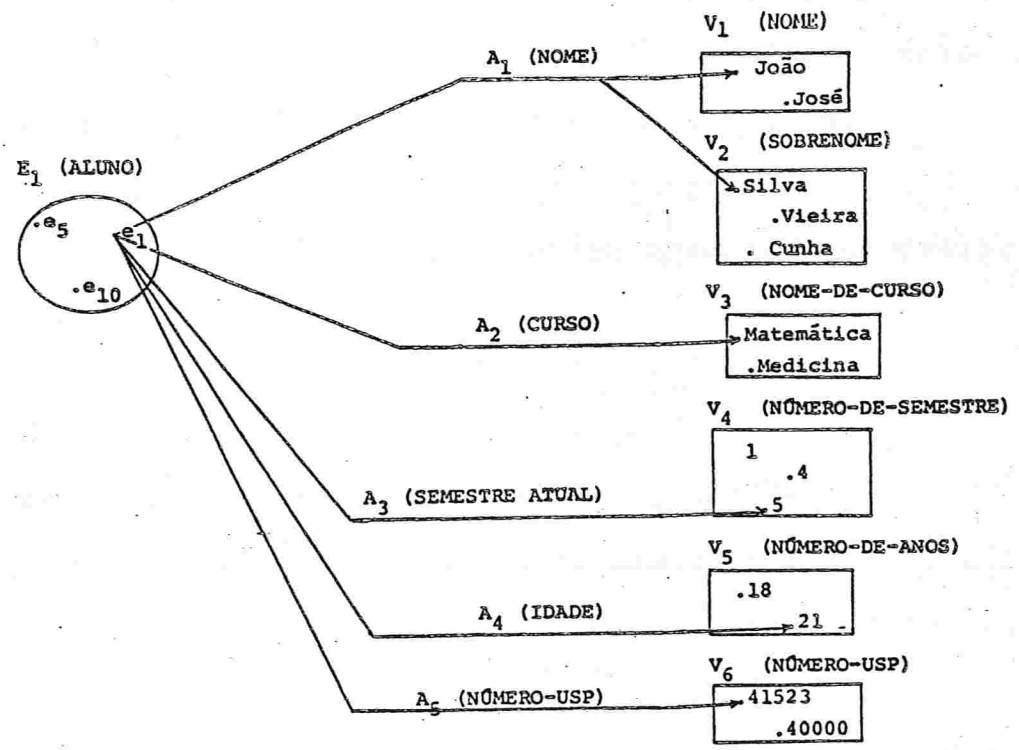
Vejamos um exemplo da descrição de um registro-entidade e_1, do conjunto-de-registros-entidade ALUNO (E_1) e de um relacionamento (r_1), do conjunto-de-relacionamentos ALUNO-DISCIPLINA (R_1):

e_1 [
- nome/João Silva
- número-USP/41523
- curso/Matemática
- semestre-atual/5
- idade/21

r_1 [
- matriculado/e_1 (papel/registro-entidade)
- disciplina/e_2 (papel/registro-entidade)
- nota/8 (atributo/valor)

Nota: o modelo deve conter também um registro-entidade e_2 (disciplina).

Uma outra descrição, esquemática, é a da figura 3.


E1 (ALUNO)
.e5
e1
.e10
A1 (NOME)
A2 (CURSO)
A3 (SEMESTRE ATUAL)
A4 (IDADE)
A5 (NÚMERO-USP)
V1 (NOME)
João
.José
V2 (SOBRENOME)
Silva
.Vieira
. Cunha
V3 (NOME-DE-CURSO)
Matemática
.Medicina
V4 (NÚMERO-DE-SEMESTRE)
1
.4
5
V5 (NÚMERO-DE-ANOS)
.18
21
V6 (NÚMERO-USP)
41523
.40000

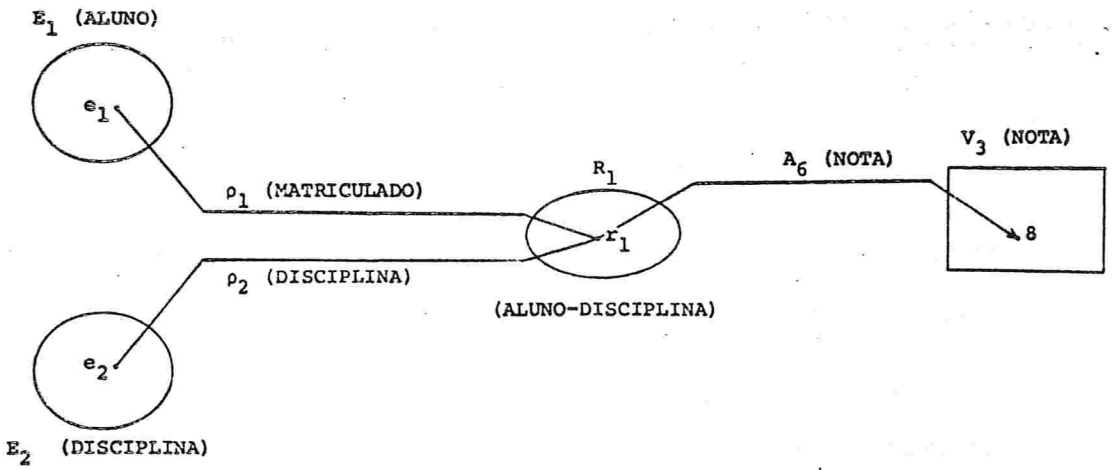

E1 (ALUNO)
e1
ρ1 (MATRICULADO)
R1
r1
A6 (NOTA)
V3 (NOTA)
8
ρ2 (DISCIPLINA)
(ALUNO-DISCIPLINA)
e2
E2 (DISCIPLINA)

Figura 3

Então, por exemplo, o atributo A_3 (SEMESTRE-ATUAL) é uma função que leva o E_1 (ALUNO) ao V_4 (NÚMERO-DE-SEMESTRE), que é o conjunto de todos os números que podem ser número de semestre (1, 2, 3, 4, ...).

Como existem mais de um atributos com o mesmo domínio (E_1), podem existir mais de um atributo com o mesmo contra-domínio. Poderíamos ter, por exemplo, A_7 (IDADE-DE-INGRESSO) levando E_1 a V_5.

A_1 é exemplo de um atributo cujo contra-domínio é um produto cartesiano de conjuntos-de-valores.

A_6 é exemplo de um atributo cujo domínio é um conjunto-de-relacionamentos, pois não é atributo de E_1 nem de E_2, isoladamente. NOTA se refere ao aluno cursando uma disciplina.

A_5 e V_6 são objetos diferentes, embora tenham o mesmo nome (NÚMERO-USP), o mesmo acontecendo com E_2 e ρ_2 (DISCIPLINA).

Visando obter uma forma do modelo mais dirigida para uma possível representação no SABD, introduziremos, mais adiante, o conceito de chave primária e uma forma tabular para representar conjuntos-de-registros-entidade e conjuntos-de-relacionamentos.

Antes, porém, cabem algumas considerações sobre as estratégias possíveis para o administrador de organismo chegar aos modelos conceitual e geral.

Uma abordagem seria a chamada "top-down". O admi-

nistrador parte de uma visão global do organismo e vai, gradativamente, chegando aos níveis mais detalhados.

No extremo oposto, está a abordagem "bottom-up". Este caminho parte de observações locais, que vão se juntando, sempre que possível, em busca de uma síntese estruturada.

Historicamente, este último tipo foi sempre o mais utilizado, provavelmente por ser mais natural e porque, como já vimos, a automatização em sistemas de informação iniciou-se em processos localizados. Para as idéias ligadas a um SABD, porém, a abordagem analítica pode oferecer instrumentos poderosos, principalmente quanto à obtenção de modelos mais estáveis.

Por esta razão, introduzimos aqui o "diagrama entidade-relacionamento" [CHEN 76], que fornece um tipo de visão global para a abordagem analítica e contém indicações importantes para se chegar aos E_i e R_i do MER: os tipos de conjuntos-de-relacionamentos.

Um exemplo desse diagrama está na figura 4.

O primeiro passo para a sua formulação é a identificação, pelo administrador de organismo, dos conjuntos-de-registros-entidade relevantes (representados por retângulos).

Em seguida, são identificados os conjuntos-de-relacionamentos (representados pelos losangos e pelos traços que os unem aos retângulos).

O tipo do conjunto-de-relacionamentos é indicado pelos caracteres que aparecem junto aos traços. Os tipos possí

veis são:

a) Um-para-um (1:1) - indica que cada registro-entidade, e_i, de um conjunto-de-registros-entidade E_1, pode estar associado a, no máximo, um registro-entidade, e_j, do outro conjunto-de-registros-entidade E_2, neste conjunto-de-relacionamentos e vice-versa. Mais precisamente, se um conjunto-de-relacionamentos R_1, que liga E_1 e E_2, é do tipo 1:1, então um $e_i \epsilon E_1$ pode estar associado a, no máximo, um $e_j \epsilon E_2$, em R_1 e vice-versa. Na figura, DEPARTAMENTO-CHEFE, que liga DEPARTAMENTO e PROFESSOR, é 1:1. Isto indica que cada departamento tem um só professor como chefe e cada professor pode ser chefe de um só departamento *. R_1 pode ser expresso como uma função bijetora que leva E_1 a E_2 e a sua inversa:

$$R_1 : E_1 \longleftrightarrow E_2$$

b) Um-para-vários (1:n) - cada $e_i \epsilon E_1$ pode estar associado a mais de um registros-entidade de E_2, mas cada $e_j \epsilon E_2$ pode estar associado a, no máximo, um registro-entidade de E_1. No exemplo, DEPARTAMENTO-PROFESSOR é 1:n. Cada departamento pode ter mais de um professor em seu qua-

---

* Mais uma consideraçao sobre a variaçao do modelo com o tempo: essa restrição se refere ao estado do modelo em um determinado instante (um "instantâneo" do modelo). Um professor poderia ser chefe de departamentos diferentes em diferentes épocas e, na verdade, essa restrição não o impede. Este modelo reflete, em seus "instantâneos", "instantâneos" do organismo.

dro, mas cada professor pertence a um só departamento.

$$R_1 : E_1 \longrightarrow E_2$$

c) Vários-para-vários (m:n) - não há restrições quanto ao número. R_1 não é associável a uma função. DISCIPLINA-PERÍODO é m:n. Cada disciplina pode estar presente em mais de um período escolar dos históricos de alunos. Um período escolar de um aluno pode abranger mais de uma disciplina.

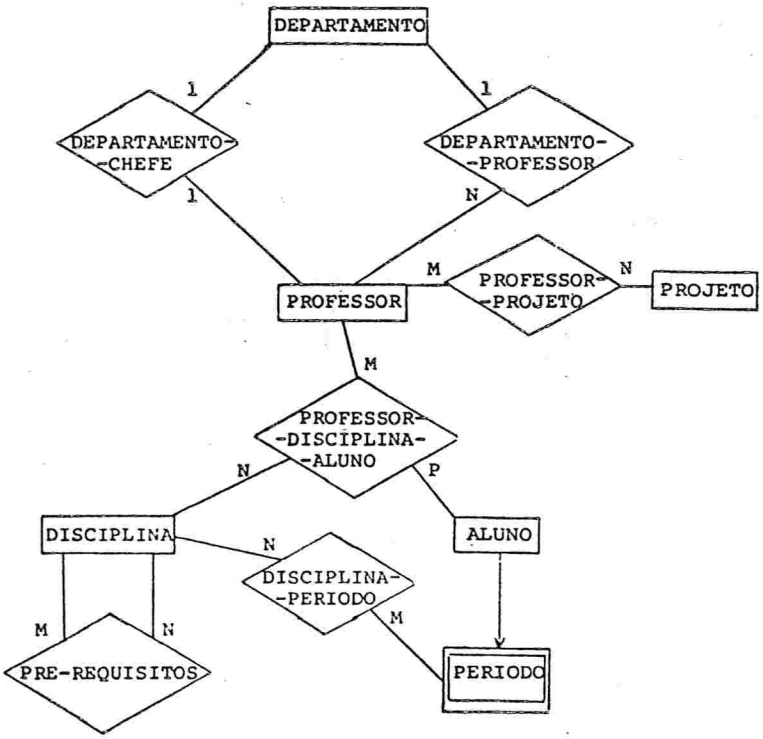
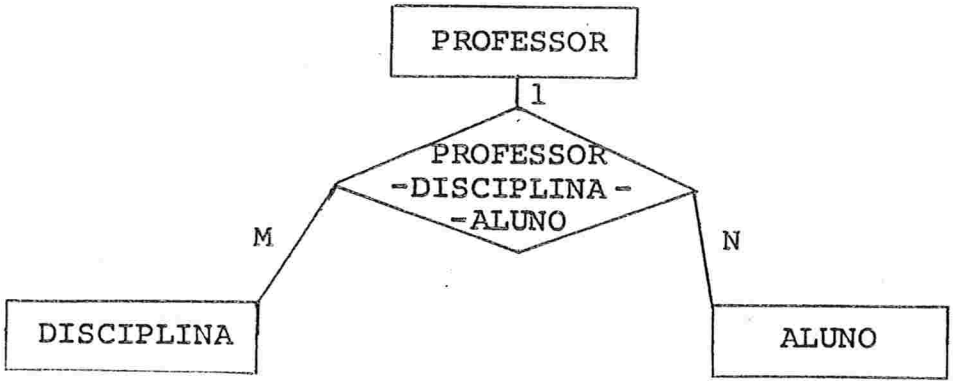


Figura 4

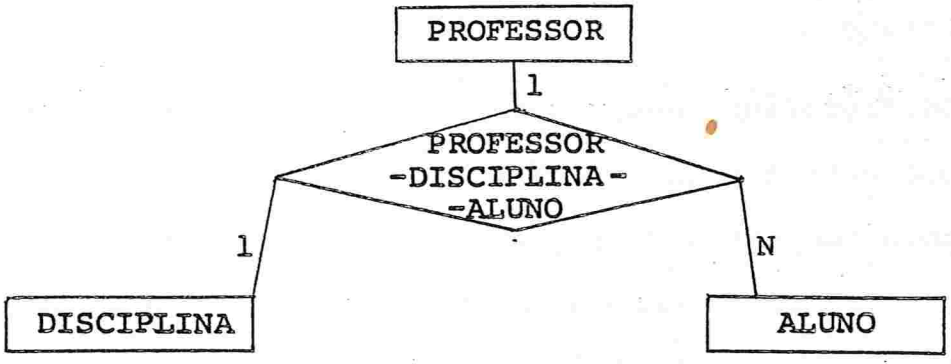
Observações:

1) Os tipos apresentados em a), b) e c) podem ser estendidos para conjuntos-de-relacionamentos envolvendo mais de dois conjuntos-de-registros-entidade. Por exemplo, PROFESSOR-DISCIPLINA-ALUNO é do tipo m:n:p, uma extensão do tipo m:n. Cada aluno, em cada disciplina, pode ter mais de um professor. Cada professor pode ministrar uma disciplina a mais de um aluno. Cada aluno pode ter aulas com um professor em mais de uma disciplina

Se, no exemplo acima, fosse introduzida a restrição adicional de que cada aluno recebe aulas sobre uma disciplina de apenas um professor, a representação deveria ser:

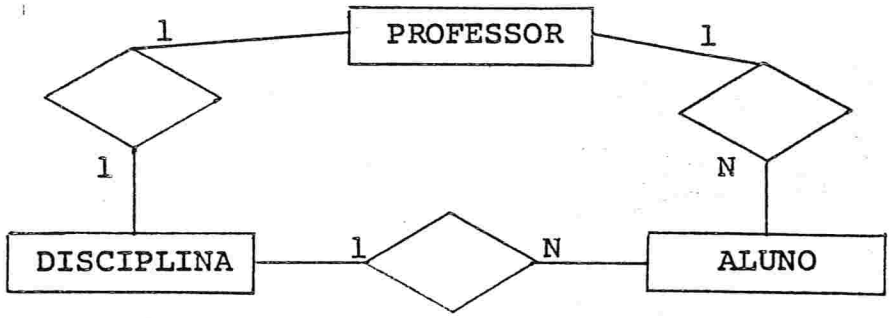


Neste caso, o M indica que, apesar da nova restrição, um aluno pode ainda receber aulas sobre mais de uma disciplina, de um mesmo professor. Se fosse exigido que um aluno tivesse aulas com um professor em apenas uma disciplina, então:

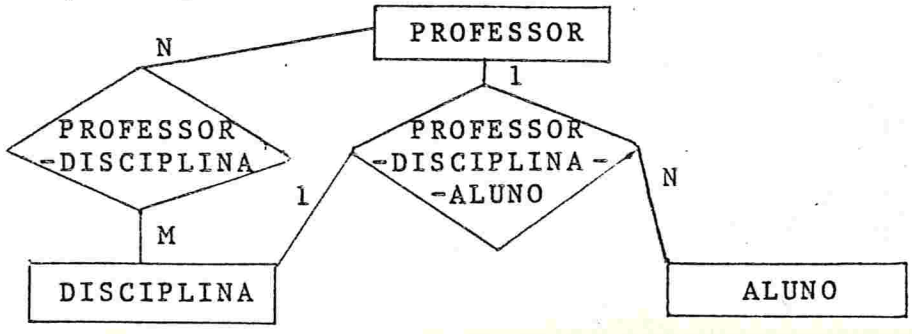


ou seja, cada aluno cursa cada disciplina com, no máximo, um professor; um aluno recebe aulas de um professor em, no máximo, uma disciplina; um professor pode ministrar uma disciplina para mais de um aluno. *

Observe-se as diferenças entre o último esquema e um esquema que relacionasse os conjuntos-de-registros-entidade dois a dois, embora com associações do mesmo tipo:



* um fato adicional, como "um professor pode ministrar várias disciplinas (embora não para um mesmo aluno) e uma disciplina pode ser ministrada por vários professores (não para o mesmo aluno)", exigiria um conjunto-de-relacionamentos adicional que ligasse tão somente PROFESSOR e DISCIPLINA.



Nesta última forma, tem-se que:

- uma disciplina pode ser ministrada a mais de um aluno, mas cada aluno só pode cursar uma disciplina, no máximo;
- um professor pode dar aulas a mais de um aluno, mas um aluno só pode ter um professor, no máximo;
- um professor pode ministrar uma disciplina, no máximo, e uma disciplina só pode ser ministrada por um professor, no máximo,

restrições totalmente diversas das do esquema anterior. Além disso, é fácil notar que nenhum esquema seria semanticamente equivalente a esse (o de ligação tripla), se tivesse apenas ligações dois a dois.

2) Um mesmo grupo de conjuntos-de-registros-entidade pode estar envolvido em diferentes conjuntos-de-relacionamentos. Ex.: DEPARTAMENTO e PROFESSOR, em DEPARTAMENTO-CHEFE e DEPARTAMENTO-PROFESSOR.

3) Um conjunto-de-relacionamentos pode envolver um único conjunto-de-registros-entidade. Ex.: PRÉ-REQUISITO liga DISCIPLINA com ele próprio. Uma disciplina pode ter outras como pré-requisito e cada disciplina pode ser pré-requisito de mais de uma outra (m:n).

4) O diagrama permite a representação de uma propriedade que tem grande influência na formulação do MER e na sua utilização prática: a "dependência existencial". Um E_1 é

dependente de E_2, quando todo $e_i \epsilon E_1$ depende da existência de algum $e_j \epsilon E_2$, isto é, e_i só pode existir ligado a e_j; se e_j for eliminado do modelo, e_i também o deve ser.

Em nosso exemplo, PERÍODO é dependente de ALUNO. Semanticamente, um período consiste na história escolar de um aluno durante um intervalo de tempo (um semestre, por exemplo): média geral, quantidade de disciplinas cursadas, quantidade de disciplinas em que foi aprovado, etc. Então, um período só tem significado quando ligado a algum aluno. Se o registro-entidade representativo deste aluno tiver que ser eliminado (após a formatura, por exemplo), todos os registros-entidade de PERÍODO que forneciam a história escolar desse aluno devem ser eliminados também.

Um conjunto-de-registros-entidade dependente é representado no diagrama por um retângulo duplo; um traço com seta mostra o conjunto-de-registros-entidade do qual ele depende. No diagrama-exemplo, PERÍODO, além da ligação com ALUNO, tem uma ligação com DISCIPLINA, através do conjunto-de-relacionamentos DISCIPLINA-PERÍODO, que informa quais as disciplinas cursadas em um período, podendo inclusive possuir atributos específicos aos relacionamentos (ex.: nota final, freqüência, etc., obtidas pelo aluno, em uma disciplina, em um período escolar).

A seguir, apresentamos alguns exemplos de como res-

trições de ordem semântica podem ser introduzidas, neste nível, através de uma linguagem próxima à da teoria dos conjuntos, utilizando o fato de que os conjuntos-de-valores são, na verdade, conjuntos no sentido matemático do termo:

a) NOTA-FINAL (dp) ϵ [0,10], onde dp ϵ DISCIPLINA-PERÍODO, isto é, a imagem do atributo NOTA-FINAL (cujo domínio é DISCIPLINA-PERÍODO) deve conter valores entre 0 e 10, apenas.

b) {NOME (a) | aϵALUNO} $\cap$ {NOME (ex) | exϵEX-ALUNO} = $\emptyset$

c) MÉDIATM(t) = Σ MÉDIA (a_i)/Nº-ALUNOS (t), onde tϵTURMA, $a_i \epsilon$ALUNO, [a_i,t] ϵ ALUNO-TURMA

d) DESCONTOS (p) < SALÁRIO (p), onde pϵPROFESSOR

O MER pode ser colocado sob uma forma mais adequada à obtenção, por exemplo, de um modelo geral e, portanto, dirigida para a formulação, também, dos modelos externos. É uma forma tabular, onde registros-entidade, atributos, relacionamentos são reunidos em tabelas, cada uma formando um conjunto-de-registros-entidade (ou um conjunto-de-relacionamentos). Cada linha de tabela descreve um registro-entidade (ou um relacionamento) deste conjunto-de-registros-entidade (conjunto-de-relacionamentos) e cada registro-entidade (relacionamento) deve estar descrito em uma só linha.

Para isto, cada conjunto-de-registros-entidade deve possuir um atributo especial, que seja uma função injetora, isto é, a distintos registros-entidade correspondem valores

distintos no conjunto-de-valores que é contra-domínio do atributo. Esse atributo recebe o nome de chave primária; "chave", porque cada registro-entidade pode ser reconhecido pelo valor deste atributo (que representa um identificador da entidade correspondente) e "primária", para distingui-lo de outros atributos que poderiam funcionar como chave. Ex.: NÚMERO-USP como chave primária de ALUNO.

Na falta de um atributo com essas características, duas alternativas podem ser seguidas:

a) um grupo de atributos pode ser escolhido como chave primária. Ex.: se NOME e SOBRENOME forem dois atributos de PROFESSOR, poderiam formar a chave primária, desde que seja contornado o problema de homônimos.
b) criar-se um atributo artificial, especialmente para desempenhar a função de chave primária. Ex.: atribuir-se uma numeração aos professores.

Formalmente, uma chave primária (CP) seria:

$$CP:E \rightarrow \underset{m\geq 1}{V_1 \times \ldots \times V_m}$$

$$\text{tal que } \exists\, CP^{-1}:V_1 \times \ldots \times V_m \rightarrow E.$$

Na figura 5 pode-se ver um exemplo da representação do conjunto-de-registros-entidade ALUNO.

Quanto aos conjuntos-de-relacionamentos, basta que cada linha contenha os valores das chaves dos registros-entidade envolvidos no relacionamento correspondente (estas cha-

ves primárias formam a chave do conjunto-de-relacionamentos), mais os valores dos atributos específicos aos relacionamentos (figura 6).

Atenção deve ser tomada com os conjuntos-de-registros-entidade dependentes. Sua chave primária deve conter a chave primária do conjunto-de-registros-entidade do qual depende (chave auxiliar) (figura 7).

Com essa prática, a condição de dependência existencial fica implícita no modelo. Quando um registro-entidade é eliminado de ALUNO, todos os registros-entidade de PERÍODO, dependentes daquele, devem ser automaticamente eliminados, já que os valores da chave primária conterão o nº-USP de um aluno "inexistente".

chave primária

ALUNO

atributo	NUMERO-USP	NOME		CURSO	SEMESTRE-ATUAL	IDADE
conjunto-de-valores	NUMERO-USP	NOME	SOBRENOME	NOME-DE-CURSO	NUMERO-DE-SEMESTRES	NUMERO-DE-ANOS
	41523	João	Silva	Matemática	5	21
	42324	José	Ayres	Matemática	3	20
	30001	Carla	Gomes	Engenharia	1	18
	⋮	⋮	⋮	⋮	⋮	⋮

Figura 5

chave

conjunto-de--registros-entidade	ALUNO	DISCIPLINA	
papel	MATRICULADO	DISCIPLINA	
atributo	NÚMERO-USP	SIGLA	NOTA
conjunto-de-valores	NÚMERO-USP	SIGLA	NOTA
	41523	MAP-111	7,5
	42324	MAP-111	8,0
MATRICULADO-DISCIPLINA	31521	MAP-121	—
	53122	MAP-211	2,5
	⋮	⋮	⋮

FIGURA 6

chave primária

chave auxiliar

conjunto-de--registros-entidade	ALUNO	PERÍODO		
atributo	NUMERO-USP	SEMESTRE		MEDIA-GERAL
conjunto-de-valores	NUMERO-USP	Nº-DE-ANO	NUMERO-DE--SEMESTRE	NOTA
	41523	74	2	6.0
	41523	75	1	6.5
	41523	75	2	6.2
	41523	76	1	6.0
	30001	⋮	⋮	⋮
	⋮	⋮	⋮	⋮

Figura 7

Damos, a seguir, uma idéia dos tipos de informação que podem ser extraídos com base em um modelo entidade-relacionamento, através de uma linguagem apoiada, fundamentalmente, na teoria de conjuntos.

Para isso, definamos um modelo-entidade-relacionamento constituído dos conjuntos-de-registros-entidade PROF, DEPTO, PROJ e dos conjuntos-de-relacionamentos PD (relaciona professores e os departamentos em que trabalham), CHEF (relaciona professores e os departamentos que chefiam) e PP (relaciona professores e os projetos em que trabalham). Os atributos relativos a cada um estão apontados na figura 8. São omitidas as caracterizações dos conjuntos-de-valores e os papéis, por serem irrelevantes para o nosso propósito.

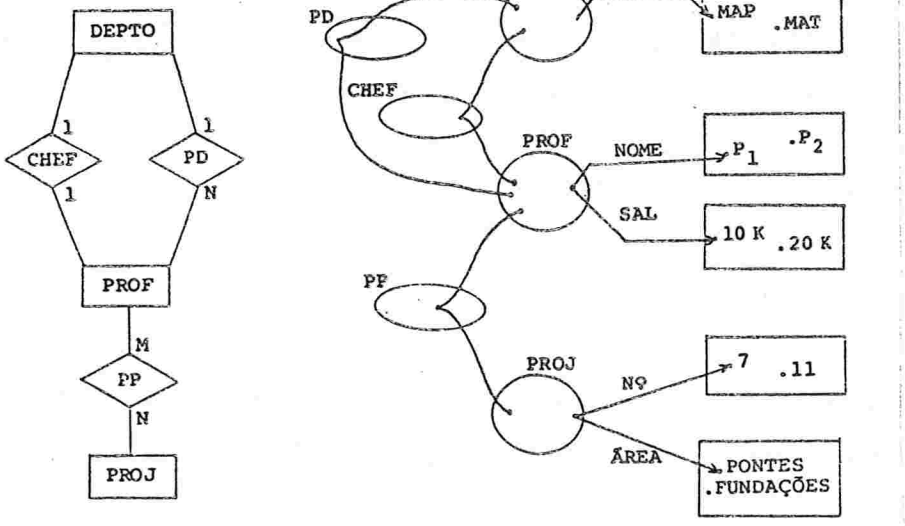


FIGURA 8

(1) "Nomes dos professores que trabalham em algum projeto"

$\{NOME(e) \mid e \in PROF,\ [e_i, e] \in PP\ ^{*},\ e_i \in PROJ\}$

(2) "Nomes e salários dos professores que não trabalham em nenhum projeto"

$\{(NOME(e), SALÁRIO(e)) \mid e \in PROF,\ PP \cap (PROJ \times \{e\}) = \emptyset\}$

(3) "Nomes dos professores que trabalham no projeto nº 7"

$\{NOME(e) \mid e \in PROF,\ [e_i, e] \in PP,\ e_i \in PROJ,\ Nº(e_i) = 7\}$

(4) "Nomes dos professores que trabalham em algum projeto, mas não no nº 7"

(1) - (3)

(5) "Nomes dos professores que não trabalham no projeto nº 7"

$\{NOME(e) \mid e \in PROF\}$ - (3)

(6) "Nomes dos professores que trabalham em outro projeto, além do nº 7"

$\{NOME(e) \mid e \in PROF,\ [e_i, e] \in PP,\ PP \cap ((PROJ - \{e_i\}) \times \{e\}) \neq \emptyset,\ e_i \in PROJ,\ Nº(e_i) = 7\}$

(7) "Nomes dos professores que trabalham apenas no projeto nº 7"

(3) - (6)

(8) "Nomes dos professores que trabalham pelo menos nos projetos nºs 5, 7, 8"

$\{NOME(e) \mid e \in PROF,\ (\{e_i, e_j, e_k\} \times \{e\}) \subseteq PP,\ e_i, e_j, e_k \in PROJ,$

---

* É irrelevante a ordem dos registros-entidade em cada relacionamento.

$N^{\circ}(e_i) = 5$, $N^{\circ}(e_j) = 7$, $N^{\circ}(e_k) = 8\}$

(9) "Nomes dos professores que trabalham em todos os projetos"

$\{NOME(e) \mid e \in PROF,\ (PROJ \times \{e\} \subseteq PP\}$

(10) "Nomes dos professores que trabalham nos projetos nºs 5, 7, 8 e em nenhum outro"

$(8) \cap \{NOME(e) \mid e \in PROF,\ PP \cap ((PROJ - \{e_i, e_j, e_k\}) \times \{e\}) = \emptyset,\ e_i, e_j, e_k \in PROJ,\ N^{\circ}(e_i) = 5,\ N^{\circ}(e_j) = 7,\ N^{\circ}(e_k) = 8\}$

(11) "Nomes dos professores que trabalham no projeto nº 7 e ganham mais do que 20 K"

$\{NOME(e) \mid e \in PROF,\ SAL(e) > 20\ K,\ [e_i, e] \in PP,\ e_i \in PROJ,\ n^{\circ}(e_i) = 7\}$

(12) "Nomes de todos os profesoores"

$\{NOME(e) \mid e \in PROF\}$

(13) "Nome(s) do(s) professor(es) que recebe(m) o maior salário"

$\{NOME(e) \mid e \in PROF,\ \{SAL(e_j) \mid e_j \in PROF,\ SAL(e_j) \leq SAL(e)\} = \{SAL(e_i) \mid e_i \in PROF\}\}$

(14) "Nomes dos professores que trabalham em pelo menos um projeto em que o professor de nome P_1 trabalha"

$\{NOME(e) \mid e \in PROF,\ [e_i, e] \in PP,\ e_i \in PROJ,\ [e_i, e_j] \in PP,\ e_j \in PROF,\ NOME(e_j) = P_1\}$

(15) "Nomes dos professores que trabalham em pelo menos todos os projetos em que o professor P_1 trabalha"

$\{NOME(e) \mid e \in PROF, \ (\{e_i \mid e_i \in PROJ, [e_i,e_j] \in PP, \ e_j \in PROF, NOME(e_j) = P_1\} \times \{e\}) \subseteq PP\}$

(16) "Nomes de todos professores ativos (em projetos) e as áreas em que trabalham"

$\{(NOME(e), AREA(e_j) \mid e \in PROF, e_j \in PROJ, [e,e_j] \in PP\}$

(17) "Nomes dos professores que trabalham sob a chefia do professor P_1 e ganham mais que ele"

$\{NOME(e) \mid e \in PROF, [e,e_j] \in PD, e_j \in DEPTO, [e_i,e_j] \in CHEF, e_i \in PROF, NOME(e_i) = P_1, SAL(e) > SAL(e_i)\}$

(18) "Nomes dos professores que trabalham no departamento MAP e ganham mais que todos os professores do departamento MAT"

$\{NOME(e) \mid e \in PROF, [e,e_j] \in PD, e_j \in DEPTO, SIGLA(e_j) = MAP, SAL(e) > \underline{MAX} \{SAL(e_i) \mid e_i \in PROF, [e_i,e_k] \in PD, e_k \in DEPTO, SIGLA(e_k) = MAT\}\}$

(MAX foi usada para abreviar a notação, correspondendo a um cálculo de "máximo" análogo ao apresentado em (13)).

(19) "Nomes dos professores que ganham mais do que seus chefes"

$\{NOME(e) \mid e \in PROF, [e,e_j] \in PD, e_j \in DEPTO, [e_i,e_j] \in CHEF, e_i \in PROF, SAL(e) > SAL(e_i)\}$

(20) "Nomes dos professores que são chefes de departamentos em que o salário mínimo é menor do que 5K ou o salário

máximo é maior do que 20K"

$\{NOME(e) \mid e \in PROF, [e,e_j] \in CHEF, e_j \in DEPTO, \underline{MIN}\{SAL(e_i) \mid e_i \in PROF, [e_i,e_j] \in PD\} < 5K\} \cup \{NOME(e) \mid e \in PROF, [e,e_j] \in CHEF, e_j \in DEPTO, \underline{MAX}\ \{SAL(e_i) \mid e_i \in PROF, [e_i,e_j] \in PD\} > 20\ K.$

(21) "Nomes dos professores que trabalham no MAP e ganham mais do que o chefe do MAP"

$\{NOME(e) \mid e \in PROF, [e,e_j] \in PD, e_j \in DEPTO, SIGLA(e_j) = MAP, [e_i,e_k] \in CHEF, e_i \in PROF, e_k \in DEPTO, SIGLA(e_k) = MAT, SAL(e) > SAL(e_i)\}.$

Cada consulta aqui apresentada não tem, evidentemente, forma única. O critério adotado para a escolha foi tão somente tentar se restringir a um mínimo de termos da linguagem usual da teoria dos conjuntos, usando, como conectivo lógico, apenas a conjunção e, como quantificador, apenas o quantificador existencial (implícitos).

Em consultas como as de nºs (13), (18) e (20), problemas podem surgir com conjuntos vazios, que satisfaçam trivialmente certas propriedades. Este, no entanto, é um problema inerente à teoria dos conjuntos e deve ser resolvido em outro nível.

C A P Í T U L O IV

DIAM - DATA INDEPENDENT ACCESSING MODEL [SENK 73]

O DIAM não é propriamente um modelo de dados, mas sim um conjunto de quatro modelos e suas interligações. Melhor ainda, é uma estrutura com quatro níveis de descrição de uma base de dados.

Vejamos, em linhas gerais, quais são esses níveis e qual sua correspondência com os níveis apresentados no esquema do capítulo II.

1) Modelo CONJUNTO-DE-ENTIDADES (MCE) (Entity Set Model): trata-se de um modelo lógico de dados, podendo ser utilizado a nível de modelo conceitual (restrito) ou como modelo geral, pois não inclui considerações sobre armazenamento de dados.
2) Modelo CADEIA (MCa) (String Model): no DIAM, as funções características do nível do modelo interno divididas entre o Modelo Cadeia e o Modelo Codificação. O MCa é o modelo onde estão descritos os caminhos de acesso aos dados.
3) Modelo CODIFICAÇÃO (MCo) (Encoding Model): é o modelo que descreve a disposição dos dados em um espaço de endereçamento virtual (independente dos dispositivos físicos de armazenamento).

4) Modelo NÍVEL-DE-DISPOSITIVO-FÍSICO (MNDF) (Physical Device Level Model): é o nível mais "baixo", correspondendo ao modelo físico. É o modelo onde está descrita a disposição dos dados nos dispositivos físicos de armazenamento.

No DIAM, a subdivisão em quatro níveis é justificada, não só pelas considerações apresentadas para a subdivisão nos modelos do capítulo anterior (independência de dados, principalmente, a possibilidade de resolver isoladamente os problemas específicos de cada nível, etc.), como também pela busca de padrões mínimos em cada modelo. As descrições, em cada nível, se baseiam em conceitos comuns, básicos, em torno dos quais são construídos os modelos. Assim, funções gerais, envolvendo esses conceitos, podem ser programadas e fazer parte do sistema, evitando que funções praticamente semelhantes sejam programadas cada vez que se façam necessárias. Com isto, diminui-se a redundância de programação, aumenta-se a compatibilidade entre sistemas, simplifica-se a verificação de programas, etc.

No MCE, a padronização se dá ao se descrever tudo em termos de "entidades".

O MCa envolve três tipos básicos de cadeias (caminhos de acesso).

O MCo é construído a partir de Unidades Básicas de Codificação - UBC.

E o MNDF se baseia nas Especificações da Subdivisão Física.

O MODELO CONJUNTO-DE-ENTIDADES (MCE)

Para o DIAM, o mundo real é visto como um conjunto de entidades. Cada ser humano é uma entidade, cada cor é uma entidade, cada número é uma entidade, uma associação entre duas ou mais entidades é uma entidade, etc.

A cada entidade estão associados um ou mais nomes. Ex.: cada parte de uma máquina tem um nome: eixo, roda, engrenagem. Uma entidade pode ter um nome emprestado de outra(s) entidade(s). Ex.: uma entidade do tipo DATA tem seu nome constituído por nomes de três outras entidades, tipos DIA, MÊS e ANO (por sua vez, o nome de uma entidade DIA é obtido através do nome de uma entidade NÚMERO).

DATA - DIA/30, MÊS/abril, ANO/1976.

Uma entidade do tipo ALUNO pode ter como nome, o NÚMERO-USP, o REGISTRO-GERAL, uma combinação NOME-SOBRENOME, etc.

Deve-se ressaltar que o nome não identifica, necessariamente, uma entidade. O nome "roda" é usado para todas as rodas, desde que não seja importante distinguir uma roda da outra.

No MCE, cada entidade relevante está representada por um registro-entidade *, univocamente identificado pe-

* Mais uma vez, usaremos uma terminologia que acompanhe as definições já apresentadas. No caso do MCE, isto será particularmente vantajoso em relação à terminologia de [SENK 73], onde coexistem termos como: Entity Name, Entity Name Set, Entity Name Set Name, Entity Set, Entity Set Name e assim por diante.

lo valor de uma ou mais chaves. Um caso como o de "rodas" é resolvido agrupando-se as rodas em um só registro-entidade (a entidade representada é, na verdade, o tipo de peça).

Para maiores detalhes, vamos partir de um exemplo ([SENK 73]). Primeiramente, veremos parte de um modelo sob uma forma tabular (figura 9).

conjunto-de-registros-entidade PARTE

chaves : PARTE/PARTE

conjuntos-de-registros-entidade associado	PARTE	COR	PESO
papel	PARTE	COR-DA-PARTE	PESO-DA-PARTE
	engrenagem	azul	17
	roda	vermelho	6
	serra	vermelho	3
	⋮	⋮	⋮

conjunto-de-registros-entidade REMESSA

chaves : Nº/Nº-DA-REMESSA;
COMPANHIA/FORNECEDOR, PROJETO/PROJETO-SERVIDO, PARTE/PARTE-FORNECIDA

Nº	COMPANHIA	PROJETO	PARTE	PESO
Nº-DA-REMESSA	FORNECEDOR	PROJETO-SERVIDO	PARTE-FORNECIDA	PESO-DA-REMESSA
2716	Jones	17	engrenagem	36
1693	Smith	17	roda	27
21	Brown	10	serra	4
431	Jones	75	engrenagem	18
⋮	⋮	⋮	⋮	⋮

Figura 9

O conjunto-de-registros-entidade PARTE representa o fato de que uma entidade do tipo PARTE é descrita através de entidades dos tipos COR e PESO, que assumem os papéis de COR-DA-PARTE e PESO-DA-PARTE, respectivamente, na descrição. Analogamente, REMESSA é descrita através de Nº (Nº-DA-REMESSA), COMPANHIA (FORNECEDOR), PROJETO (PROJETO-SERVIDO), PARTE (PARTE-FORNECIDA) e PESO (PESO-DA-REMESSA).

No MCE, um conjunto-de-registros-entidade como COR é, na verdade, fictício (ou virtual), pois ele não comparece no modelo numa forma semelhante à apresentada na figura 9. Isto representa o fato de que uma COR é descrita tão somente pelo seu nome. O mesmo acontece com Nº e PESO. A escolha dos conjuntos-de-registros-entidade "reais" e dos "fictícios" é um problema do administrador do organismo. Pode-se ter uma base de dados onde se queira, por exemplo, uma tabela relacionando pesos em várias unidades de medida; nesse caso, PESO obteria um "status" de conjunto-de-registros-entidade "real" e seria descrito por QUILOS, LIBRAS, etc.

PROJETO, COMPANHIA e PARTE são conjuntos-de-registros-entidade descritos explicitamente no modelo e que comparecem também na descrição de registros-entidade do tipo REMESSA, através de uma de suas chaves.

A todo conjunto-de-registros-entidade associado à descrição cabe um papel que dá indicações semânticas sobre a

associação.

Note-se, ainda, que fica omitido, por exemplo, o fato de que os nomes de PESO são também emprestados do conjunto-de-registros-entidade NÚMERO. Na verdade, os fatos relativos a esses conjuntos "fictícios" devem merecer um tipo especial de descrição (provavelmente semelhante à descrição dos conjuntos-de-valores, no MER).

Para passar do MCE para o Modelo Cadeia, devemos antes mencionar os três tipos de combinações entre os dados expressos pelos conjuntos-de-registros-entidade acima descritos. Estas combinações (que são facilmente extensíveis a outros modelos, como o MER, o relacional, etc.) sugerem as operações sobre o MCE, necessárias para gerar os tipos de informação solicitados a um SABD.

1) Combinações entre triplas: conjunto-de-registros-entidade associado/papel/valor (valor da chave do registro-entidade associado), em um mesmo registro-entidade, permitem resposta a questões sobre características de um registro-entidade identificado. Ex.: "qual o projeto e a companhia envolvidos na remessa de nº 2716?"

2) Combinações entre registros-entidade de um mesmo conjunto-de-registros-entidade permitem satisfazer buscas a registros-entidade com características especificadas. Ex.: "quais as partes vermelhas?"

3) Combinações entre registros-entidade de diferentes conjun

tos-de-registros-entidade permitem respostas a quesitos de vários graus de complexidade. Ex.: "qual o peso das partes vermelhas fornecidas por Smith a projetos de nº maior que 15?"

Apesar de ter sido apresentado, aqui, sob forma tabular, nada há de explícito no MCE que dê indicação sobre armazenamento ou sobre caminhos para acesso aos dados nele contidos.

As questões envolvidas na formulação do MCE (que são pertinentes ao MER e ao modelo relacional, também) têm como objetivo:

- evitar redundância
- manter consistência e integridade da base de dados

nos pontos possíveis de serem considerados em um modelo lógico:

- escolha dos conjuntos-de-registros-entidade associados à descrição de um tipo de registro-entidade (no MER: escolha dos atributos). Ex.: em várias REMESSAS, a PARTE-FORNECIDA é "engrenagem"; a COR de "engrenagem" deve ser associada à descrição de uma PARTE e não à descrição da REMESSA, pois nesse último caso, haveria redundância na representação de que "uma engrenagem é azul". A redundância, além de anti-econômica *, aumenta o perigo de inconsistência: um engano de atualização e poderemos, em pontos

---

* quando estendida até o nível físico.

diferentes, ter a informação de que "uma engrenagem é azul" e "uma engrenagem é preta", v.g. Se a cor estiver representada apenas na descrição da PARTE "engrenagem", esse perigo não ocorre. *

- como representar os relacionamentos entre entidades (como os 1:n, m:n, etc. do MER)? Em um relacionamento 1:n, como em DEPARTAMENTO e PROFESSOR, é mais natural que DEPARTAMENTO seja utilizado na descrição de PROFESSOR. Já um relacionamento m:n obriga a criação de um novo conjunto-de-registros-entidade, que pode ter também outros conjuntos-de-registros-entidade associados à sua descrição. É o caso de REMESSA, que representa o relacionamento entre três entidades, PARTE, COMPANHIA e PROJETO (PESO é um conjunto-de-registros-entidade associado a esse relacionamento). É fácil notar a redundância envolvida, se cada tipo de registro-entidade participasse da descrição do outro. Um relacionamento 1:1 pode ser de dois tipos:

  a) entre dois conjuntos-de-registros-entidade efetivamente presentes no modelo, como DEPARTAMENTO e PROFESSOR (chefe de departamento). Neste caso, apenas a semântica do relacionamento indica que PROFESSOR deve entrar na

---

* Caso análogo se verifica com relaçao a PROFESSOR e DEPARTAMENTO. Associado a cada PROFESSOR devemos ter o DEPARTAMENTO e a cada DEPARTAMENTO, um PROFESSOR (o chefe). Se a cada PROFESSOR associássemos outro PROFESSOR (o seu chefe), a manutenção seria sobrecarregada, já que cada professor tem uma associação mais estável com o departamento do que com o chefe.

descrição de DEPARTAMENTO (no papel de CHEFE), já que todo DEPARTAMENTO tem um chefe. Se fosse adotado o inverso, isto é, DEPARTAMENTO entrasse na descrição de PROFESSOR, muitos registros-entidade PROFESSOR (relativos aos professores não-chefes) teriam campo vazio.

b) entre dois conjuntos-de-registros-entidade associados à descrição de um terceiro:

b1) caso de múltiplas chaves (ex.: NÚMERO-USP e REGISTRO-GERAL).

b2) caso em que destes dois conjuntos-de-registros-entidade associados, um apenas deveria permanecer e outro conjunto-de-registros-entidade deveria ser criado para representar o relacionamento. Ex.: se na descrição de PARTE, tivéssemos PESO-EM-QUILOS e PESO-EM-LIBRAS (relacionados 1:1), os problemas de redundância e coexistência seriam evitados com a criação de um conjunto-de-registros-entidade representando o relacionamento PESO-EM-QUILOS e PESO-EM-LIBRAS.

O MODELO CADEIA (MCa)

O MCa é o primeiro passo para se chegar do MCE ao armazenamento físico.

Seguindo a filosofia do DIAM de buscar unidades básicas comuns, o MCa consta de três qualidades de cadeias,

representando as três combinações definidas anteriormente. Associações entre cadeias destas três qualidades podem representar os principais caminhos de acesso necessários para um SABD e permitem que um estudo seja feito tão somente nesse nível, abstraindo-se tanto as considerações pertinentes ao modelo lógico (MCE) quanto outras considerações relativas ao armazenamento.

1) Cadeias atômicas (CA): unem triplas conjunto-de-registros-entidade/papel/valor, associados a um mesmo registro-entidade, em uma ordem específica. ALFA, BETA e GAMA são três tipos de cadeias atômicas apresentadas na figura 10, que é exemplo já descrito, sob uma forma mais adequada para ilustrar as cadeias.

   ALFA une uma tripla PARTE/PARTE-FORNECIDA/valor com uma tripla Nº/Nº-DA-REMESSA/valor (nesta ordem), em REMESSA.

   BETA une PARTE/PARTE/valor, COR/COR-DA-PARTE/valor e PESO/PESO-DA-PARTE/valor, em PARTE.

   GAMA une COR/COR-DA-PARTE/valor e PARTE/PARTE/valor, em PARTE.

Na definição, por exemplo, do tipo ALFA, não se entra em considerações sobre se a representação física será feita através de contiguidade física, por apontadores, etc. (figura 11). O que importa na definição das cadeias são:

- os elementos que a cadeia une
- em que ordem
- a quais outras cadeias ela está ligada.

2) Cadeias-entidade (CE): unem cadeias atômicas do mesmo tipo. Os elementos a serem unidos são especificados através de qualificações booleanas.

   ENGRNG e INDX-COR são tipos de CEs (figura 10).

   ENGRNG une cadeias ALFA, onde PARTE/PARTE-FORNECIDA = 'engrenagem', ordenadas por Nº/Nº-DA-REMESSA.

   INDX-COR une todas cadeias GAMA, por ordem alfabética de COR/COR-DA-PARTE.

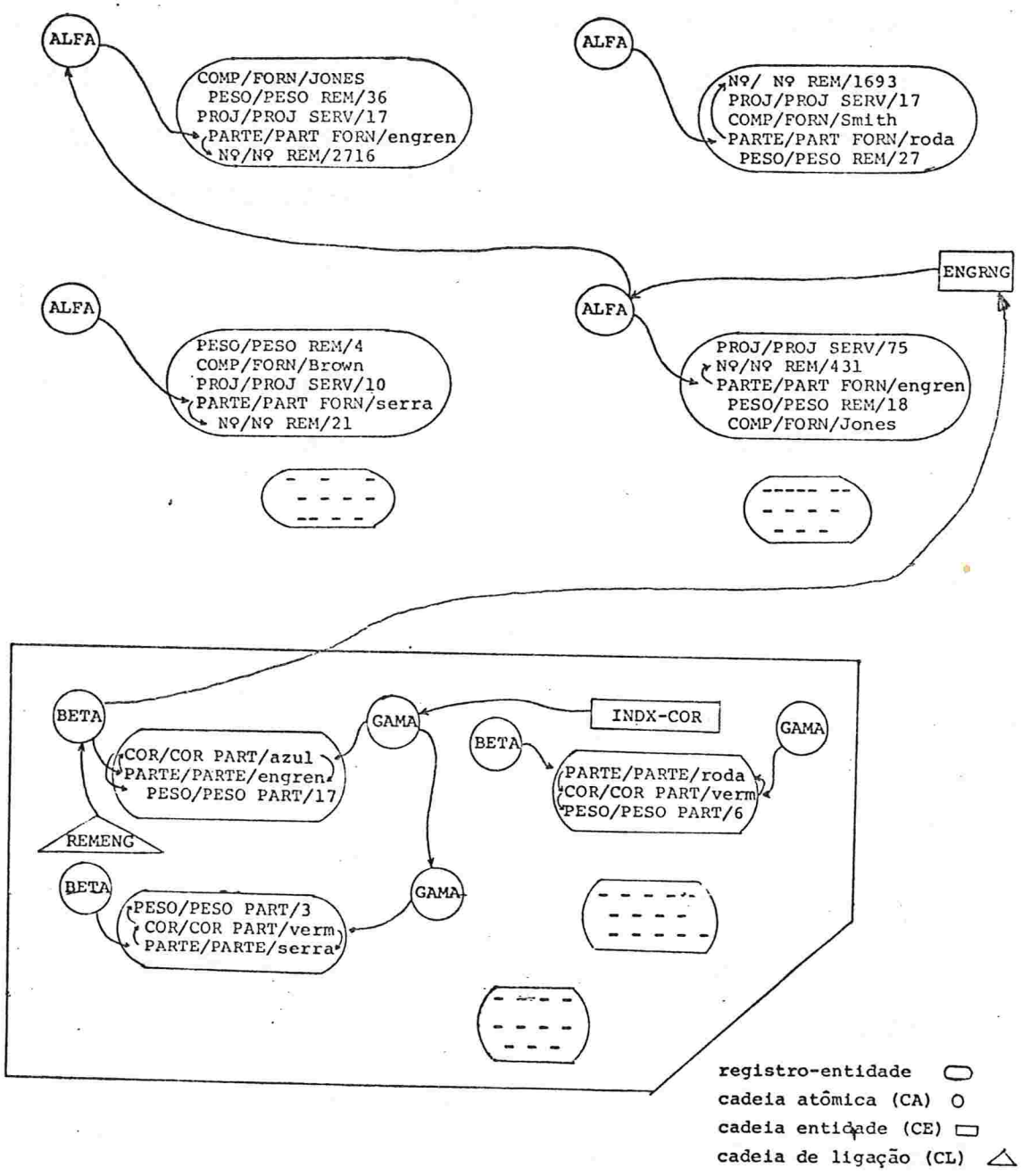
3) Cadeias de Ligação (CL): unem elementos (que podem ser CAs, CEs ou CLs) que têm em comum o valor da chave de um mesmo registro-entidade.

   REMENG é um tipo de CL (figura 10).

   REMENG une uma CA BETA com uma CE ENGRNG, nesta ordem. Note-se que, através de REMENG, BETA, ENGRNG e ALFA, estabelece-se um caminho de acesso hierárquico de um registro-entidade de PARTE para registros-entidade de REMESSA. Hierárquico no sentido de que só se pode atingir as REMESSAs a partir de uma PARTE, justamente a PARTE que desempenha o papel de PARTE-FORNECIDA nessas REMESSAs.

A CE INDX-COR representa o estabelecimento de um índice secundário para PARTE. Um índice primário (através da chave de PARTE) poderia ser estabelecido analogamente, com uma CE que una as BETA, por exemplo, o que seria, também, uma forma de representar PARTE como um arquivo seqüencial.

conjunto-de-registros-entidade: REMESSA
chaves: Nº/Nº REM; COMP/FORN, PROJ/PROJ SERV, PARTE/PART FORN.



conjunto-de-registros-entidade: PARTE
chaves: PARTE/PARTE.

Figura 10

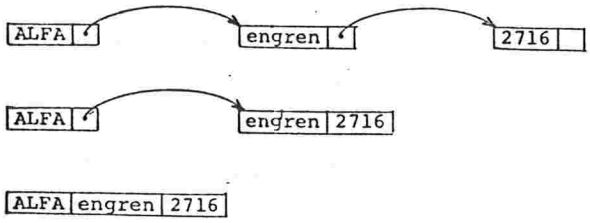


Figura 11

Três esquemas possíveis para representação física de uma CA.

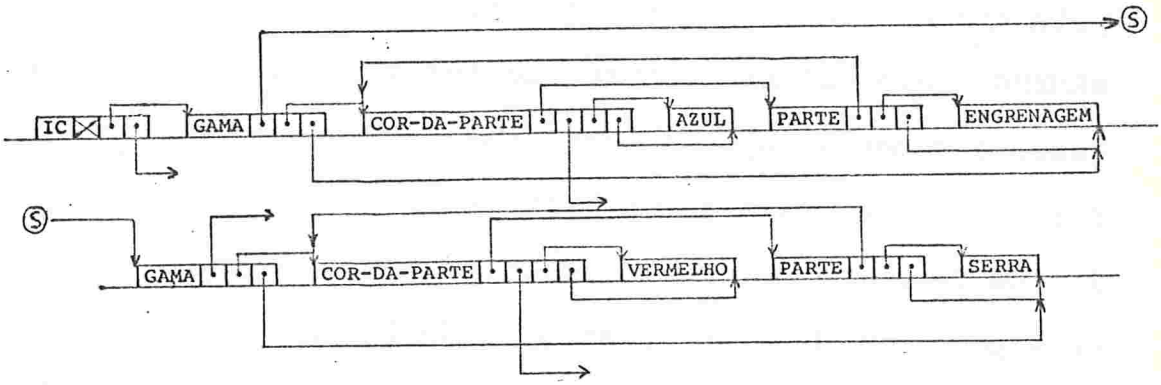
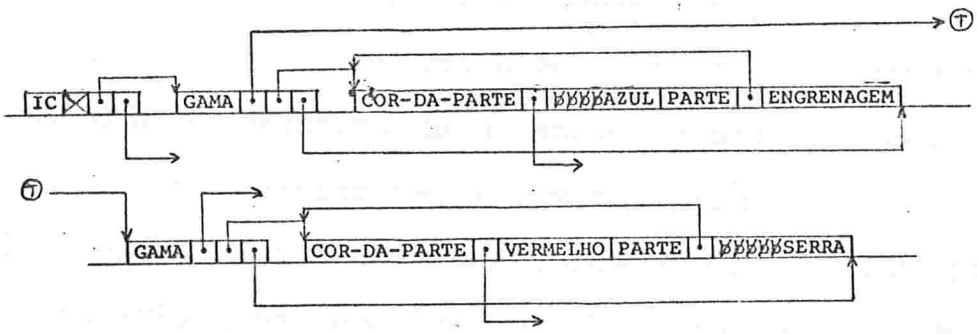


Figura 12



CATÁLOGO contém: COR-DA-PARTE | APTA1: adjacência
APTV: adjacência
FIM: 8

PARTE | APTV: adjacência
FIM: 10

Figura 13

As cadeias BETA, não estando unidas a outras cadeias (exceto à que participa em REMENG), são acessáveis por fórmulas de endereçamento direto ("hashing").

Como se pode perceber, a escolha das cadeias é sujeita a considerações de desempenho. Devem ser estabelecidos caminhos de acesso simples para as operações mais executadas pelo sistema. Assim, por exemplo, a escolha de cadeias como REMENG será valiosa sempre que for alta a freqüência de acesso a remessas após o exame das características da parte fornecida. Da mesma forma, INDX-COR será justificável se um índice de cores for muito usado para acesso a PARTE. Note-se a possibilidade de construção de índices com múltiplos níveis e, de resto, os principais tipos de caminhos de acesso são representáveis através das cadeias definidas.

Freqüentemente, existem mais de um caminhos de acesso para atingir os dados procurados. Um amplo campo de pesquisas consiste na busca de um processo de seleção automático da melhor alternativa, a ser incorporado no transformador MCE/MCa. Aparentemente, a adoção dos três tipos básicos de cadeias e a construção de catálogos adequados para a descrição desses modelos devem facilitar esse trabalho.

O MODELO CODIFICAÇÃO (MCo)

Neste nível, o modelo representa a forma como as cadeias do MCa e seus elementos se distribuem em um espaço de

endereçamento (tomado aqui como linear) constituído por bits ou bytes, cada qual com um endereço identificador.

O elemento básico comum é, agora, a chamada Unidade Básica de Codificação (UBC) que, em nossa terminologia, é um registro interno.

Cada UBC é constituída de campos, cujos valores podem representar nomes, comprimentos, apontadores ou valores de chave. O sistema também possui um catálogo que descreve o MCo. Este catálogo contém as informações sobre os formatos das UBC de cada tipo. Essas informações são, na verdade, parametrizações de um modelo padrão de UBC:

RÓTULO	APTA 1	APTA 2	...	APTAn	APTV	FIM

RÓTULO: é o nome do tipo de cadeia ou valor de chave ou papel, etc. a que se refere esta UBC (ex.: ALFA, REMENG, CORDA-PARTE, VERMELHO).

APONTADOR DE ASSOCIAÇÃO (APTA): há um apontador de associação para cada coleção à qual esta UBC pertença (mas não encabece). O valor desse apontador corresponde ao endereço da próxima UBC da coleção.

APONTADOR DE VALOR (APTV): é o apontador para a primeira UBC de uma coleção encabeçada por esta UBC (ou para a UBC que contém o valor da chave, no caso de um papel).

FIM: é o apontador para o final da última UBC da coleção encabeçada por esta UBC.

Na figura 12, vemos ilustrada uma parte de um MCo

correspondente ao exemplo anteriormente apresentado para o MCE e MCa. As notas que se seguem referem-se a essa figura.

1) rótulo=IC (INDX-COR) - inicia, portanto, uma ocorrência de CE.

   apta - não tem apontadores de associação pois não pertence a nenhuma outra coleção de UBCs.

   aptv - aponta para 2), que é uma UBC para a ocorrência de GAMA correspondente à parte cuja cor é a primeira em ordem alfabética.

   fim - aponta para o final da coleção correspondente à cadeia INDX-COR.

2) rótulo=GAMA - corresponde à ocorrência de uma CA tipo GAMA.

   apta 1 - aponta para a próxima UBC da coleção encabeçada por 1).

   aptv - aponta para 3), que é uma UBC para uma ocorrência de COR-DA-PARTE, que é o primeiro papel de uma cadeia tipo GAMA.

   fim - aponta para o final da coleção correspondente a esta cadeia GAMA.

3) rótulo=COR-DA-PARTE - corresponde à ocorrência de um papel.

   apta 1 - aponta para a próxima UBC (correspondente ao papel PARTE) da coleção encabeçada por 2).

   apta 2 - aponta para a próxima UBC (papel PESO-DA-PAR-

TE) de uma CA BETA . (não aparece na figura).

aptv - aponta para a UBC correspondente ao valor associado ao papel COR-DA-PARTE.

fim - aponta para o final da UBC indicada pelo aptv.

4) rótulo=AZUL.

(este tipo de UBC só possui o rótulo).

5) apta 1 - não tem apontador de associação referente à cadeia GAMA, pois corresponde ao último papel de uma CA desse tipo (O apontador que aparece refere-se à cadeia BETA).

No exemplo da figura 12, quase todas as informações estão presentes no espaço de endereçamento *. A figura 13 ilustra o mesmo exemplo, quando se utiliza um catálogo que contenha informações comuns às UBCs do mesmo tipo, possibilitando que essas informações sejam omitidas no espaço de endereçamento.

O MODELO NÍVEL DE DISPOSITIVO FÍSICO (MNDF)

Finalmente, o Modelo Nível de Dispositivo Físico mostra a disposição dos dados nos dispositivos físicos reais e propriedades desses dispositivos no tratamento dos dados.

A transformação do MCo para o MNDF é a transfor

---

* falta apenas identificar qual apontador de associaçao pertence a qual coleção (em COR-DA-PARTE, por exemplo), e os comprimentos dos rótulos.

mação dos endereços virtuais do MCo em indicações dos locais efetivos, correspondentes a esses endereços.

Em sistemas mais simples, quanto a este aspecto, os endereços virtuais já podem conter as indicações para os endereços reais. Por exemplo, quando o endereço virtual do MCo é um número composto pelos número de volume, número de cilindro, número de trilha, número de bloco e número de byte (ou bit). Nesse caso, o MCo e o MNDF praticamente se confundem.

Como já foi dito, é comum também que o projeto do SABD chegue apenas a uma interface com o sistema operacional (através do endereço virtual) e fique a cargo deste toda a manipulação do espaço físico real (alocação de espaço, manutenção de espaços livres, tratamento de "overflow", etc.).

Dado que o MNDF está diretamente ligado a características físicas do ambiente em que está implementado o SABD, é mais difícil o estabelecimento de conceitos básicos que funcionem como unidades semelhantes às estabelecidas para os outros modelos. De uma forma geral, pode-se fixar como padrões, para a descrição de MNDF, os seguintes tópicos:

a) Especificação dos tipos de subdivisão do espaço físico, contendo:
   - nome
   - descrição dos componentes (podem ser outros tipos de subdivisões)
   - processo de inserção de dados

- administração do espaço disponível (alocação de novos recursos)
- tratamento de overflow.

Exemplos de tipos de subdivisão: trilhas, cilindros, cilindros de dados, cilindros de índices, cilindros de overflow, volumes, etc.

b) Designação dos tipos para as subdivisões reais. Exemplo: assinalar os números dos cilindros de overflow, os números das trilhas que conterão índices.

c) Alocação do espaço físico, isto é, atribuir nomes às áreas que circunscrevem porções do espaço formatado em b). Exemplo: AREA1 será constituída de todos os blocos das trilhas nº 2 a nº 5 do cilindro nº 8, no volume nº 1.

d) Estabelecimento das regras de transformação dos endereços virtuais do MCo para os endereços do espaço físico preparado em a), b) e c). Devem estar incluídos aqui, além de fórmulas para o cálculo do endereço físico apropriado, a política de tratamento de sinônimos (endereços do MCo que geram endereços idênticos no MNDF), a política para a obtenção de um novo endereço físico no caso do anterior apontar um local de tamanho insuficiente para a inserção de um dado, e outras especificações semelhantes.

Como exemplo de descrição de um MNDF, [SENK 73] indica a maneira como é descrito o ISAM (Indexed Sequential Access Method) nos manuais da IBM.

C A P Í T U L O V

O MODELO RELACIONAL ([CODD 70])

Um modelo relacional de dados é constituído por conjuntos-de-registros. Cada conjunto-de-registros R é uma relação matemática definida do seguinte modo:

dados n conjuntos $C_1, C_2, \ldots, C_n$, não necessariamente distintos,

$$R \subseteq C_1 \times C_2 \times \ldots \times C_n \text{, onde}$$

$$C_1 \times C_2 \times \ldots \times C_n = \{(c_1, c_2, \ldots, c_n) \mid c_1 \in C_1,\ c_2 \in C_2, \ldots,\ c_n \in C_n\}.$$

Cada tupla de R corresponde a um registro.

$C_1, C_2, \ldots, C_n$ recebem o nome de domínios. Assim, C_i é o i-ésimo domínio de R.

Assim definido, o modelo relacional apresenta inúmeras características adequadas para figurar como o modelo geral de um SABD, conforme veremos adiante. Além disso, é o modelo que forneceu campo para os maiores avanços na tentativa de formalizar o assunto.

O modelo relacional é, tipicamente, um modelo lógico de dados, pois se presta a uma manipulação da base de da-

dos que independe da maneira como esses dados são armazenados fisicamente. Essa manipulação pode ser vista, simplificadamente, como operações sobre tabelas, independentes dos caminhos de acesso, do modo como os dados das tabelas estão guardados nos dispositivos de armazenamento, dos índices adotados ou das seqüências escolhidas (independência de dados).

Eis algumas conseqüências implícitas no modelo:

- a ordem das tuplas é irrelevante: não há uma seqüência de registros imposta "a priori" e, assim, nenhuma operação tomará por base uma ordenação de registros.
- todas as tuplas são distintas.
- a ordem dos domínios é significativa para a identificação dos elementos nas tuplas, mesmo que os domínios sejam identificados por nomes (devido à possibilidade de domínios idênticos em uma mesma relação); essa ordem pode ser tornada irrelevante (a exemplo do MER e do MCE) caso se atribua ainda um papel a cada domínio.

Exemplo:

R

domínios	professor	professor
	P_1	P_5
	P_2	P_5
	P_3	P_7
	.	.
	.	.

R é uma relação que informa quais os orientadores de professores. Os professores orientados aparecem na colu-

na da esquerda e os orientadores na outra.

Para que a ordem deixe de ser significativa, o 1º domínio deve receber o papel de ORIENTADO e o 2º, de ORIENTADOR.

Se os papéis passam a fazer parte do modelo, um conjunto-de-registros passa a ser, na verdade, uma classe de equivalência das relações que diferem apenas pela ordem dos domínios. Sempre que possível, essa distinção entre as duas abordagens será omitida.

De um ponto de vista matemático, quando se altera qualquer elemento de qualquer tupla de uma relação, quando se insere (elimina) uma tupla ou quando se insere (elimina) um domínio, uma nova relação é criada. Quando se trata de um modelo de dados, não é conveniente introduzir um novo nome a cada alteração feita. Um conjunto-de-registros deve, então, ser visto como uma relação variante no tempo ou, melhor, duas relações diferentes podem ter o mesmo nome em diferentes instantes de tempo. O mesmo se aplica aos domínios.

No modelo relacional, como, de resto, em todo tipo de modelo de dados, dois aspectos principais devem ser abordados:

1) A estruturação do modelo: no caso do modelo relacional, esse ponto corresponde à escolha das relações que comporão o modelo e dos domínios que participarão em cada relação. Os critérios que regem essa escolha e as maneiras pelas quais se pode chegar à estrutura final são reunidos no

estudo das chamadas formas normais de relações.

2) A operação do modelo: para o modelo relacional, esse ponto será abordado através da apresentação de dois tipos principais de linguagens para manipulação das relações, do estudo da capacidade seletiva dessas linguagens (ou seja, a capacidade de selecionar informações a partir do modelo) e de um pequeno estudo comparativo entre elas. Esse ponto corresponde à maneira como o modelo se comporta com respeito a consultas, atualizações, inserções, eliminações, etc.

A abordagem desses dois aspectos será feita na ordem inversa da numeração acima, por ser mais natural.

A OPERAÇÃO DO MODELO

[CODD 72a] introduziu uma álgebra de relações e um cálculo de predicados como bases para dois tipos de linguagens de manipulação de dados. Essas linguagens podem ter uma identidade própria dentro do sistema (linguagens de consulta *) ou podem constituir sublinguagens que são incorporadas às linguagens de programação usuais, provendo-as de uma extensão para manipulação da base de dados relacional. As relações a serem operadas por essas linguagens serão relações

---

* "query language", em inglês. Esse nome é, tradicionalmente usado, mesmo que, além da capacidade de consulta, a linguagem contenha recursos de atualização.

normalizadas (ou relações na 1ª forma normal), isto é, relações cujos domínios contêm apenas elementos simples, não decomponíveis (portanto, um domínio não pode ter uma relação entre seus elementos) (ver formas normais, na estruturação do modelo).

A seleção de dados por uma linguagem baseada em uma álgebra ou em um cálculo de relações pode ser encarada como a formação de uma nova relação, a partir das relações definidas no modelo (saliente-se que um simples valor pode ser visto como uma relação unária com uma só tupla).

ÁLGEBRA DE RELAÇÕES

Sejam $r = (c_1,\ldots,c_n)$ uma tupla de $R \subseteq C_1 \times \ldots \times C_n$

e $s = (d_1,\ldots,d_m)$ uma tupla de $S \subseteq D_1 \times \ldots \times D_m$

Definição: A concatenação de r com s é a tupla definida por

$$r \frown s = (c_1,\ldots, c_n, d_1,\ldots, d_m) .$$

Operações:

1) PRODUTO CARTESIANO entre duas relações R e S.

O produto cartesiano usual entre duas relações R e S é uma relação binária definida por:

$$R \times S = \{(r,s) \mid r \in R,\ s \in S\} .$$

Na álgebra de relações, no entanto, é usado um pro-

duto cartesiano especial, definido por:

$$R \otimes S = \{r \frown s \mid r \epsilon R,\ s \epsilon S\} .$$

As operações 2), 3) e 4) que se seguem, são operações usuais da teoria dos conjuntos, mas são definidas apenas para relações de mesmo grau (Grau de uma relação é o número de domínios que a definem). Se, em uma relação-resultado, houver ambiguidade quanto aos nomes dos domínios, prevalecem, por convenção, os nomes da relação mais à esquerda.

2) UNIÃO entre duas relações, R e R', de grau n.

$$R \cup R' = \{r \mid r \epsilon R \vee r \epsilon R'\} .$$

3) INTERSECÇÃO entre duas relações, R e R', de grau n.

$$R \cap R' = \{r \mid r \epsilon R \wedge r \epsilon R'\} .$$

4) DIFERENÇA entre duas relações, R e R', de grau n.

$$R - R' = \{r \mid r \epsilon R \wedge r \notin R'\} .$$

5) PROJEÇÃO.

É uma operação efetuada sobre uma relação, tendo, como parâmetros, domínios escolhidos. Ela permite que sejam isolados os domínios desejados, que sua ordem de apresentação seja alterada, etc.

Para se identificar um elemento de uma tupla, a notação mais completa seria algo como:

r(t).p.D

onde r é a tupla em questão (pertencente à relação R),

t é o instante de tempo (indica qual versão de R está sendo considerada),

D é o domínio considerado,

p é o papel desempenhado pelo domínio D em R.

Em geral, t pode ser omitido, ficando convencionado que a versão de R a ser considerada é a última; p só é realmente necessário quando existem mais de um domínios com o mesmo nome. Então, na maior parte dos casos, basta uma notação como:

r.D

para identificar o elemento.

Uma outra notação utiliza uma ordenação dos domínios. Então, para i = 1, 2,..., n

r[i] denota o i-ésimo componente de r.

(r[i], para i < 1 ou i > n, não é definido).

Já que elas não se confundem, usaremos ambas as notações, de acordo com a conveniência.

Duas extensões podem ser feitas a essas notações:

Seja A uma lista de domínios de R,

$$A = (C_{i_1}, C_{i_2}, \ldots, C_{i_k})$$

ou uma lista de inteiros

$$A = (i_1, i_2, \ldots, i_k) .$$

Então,

a) $r.A = (r.C_{i_1}, r.C_{i_2}, \ldots, r.C_{i_k})$ ou

$r[A] = (r[i_1], r[i_2], \ldots, r[i_k])$.

Se A for vazia, r.A (ou r[A]) = r .

b) É a definição da operação de projeção:

Projeção de R sobre A é definida por

$R[A] = \{r[A] \mid r \epsilon R\}$ ou

$R.A = \{r.A \mid r \epsilon R\}$.

Na figura 14 vemos exemplos das operações 1) a 5).

6) JUNÇÃO.

É uma operação que, a partir de duas relações, forma uma terceira, através de correlações entre domínios.

Vamos dar uma definição geral da operação e introduzir, também, a chamada junção natural que, por ser o caso mais comum, merece uma definição simplificadora.

Seja θ qualquer um dos operadores relacionais $=$, $\neq$, $>$, $\geq$, $<$, $\leq$.

A junção da relação R sobre A, com a relação S sobre B (onde A e B são listas de domínios de R e S, respectivamente), segundo θ, é definida por:

$$R[A\ \theta\ B]S = \{r \frown s \mid r \epsilon R \wedge s \epsilon S \wedge (r.A\ \theta\ s.B)\} .$$

desde que todo r.A seja comparável, segundo θ, com todo s.B. Diz-se que r.A é comparável, segundo θ, com s.B quando $r.A\ \theta\ s.B$ é verdadeiro ou falso (não indefinido).

O caso mais freqüente de junção é segundo =, quando a junção deve, naturalmente, ser seguida de uma projeção que elimine a duplicação relativa aos domínios de A e B.

Costuma-se, então, definir a junção natural:

para $A = (C_{i_1}, C_{i_2}, \ldots, C_{i_k})$ e $B = (D_{j_1}, D_{j_2}, \ldots, D_{j_k})$

$$R[A*B]S = \{r^\frown(s.\overline{B}) \mid r\epsilon R \wedge s\epsilon S \wedge (r.C_{i_q} = s.D_{i_q},\ q = 1,2,\ldots,k)\}$$

onde $\overline{B}$ é a lista de domínios de S que não pertencem a B.

A junção é exemplificada na figura 15.

7) DIVISÃO

Para definir a divisão da relação R, sobre a lista A, pela relação S, sobre a lista B (domínios de A pertencem a R e domínios de B pertencem a S), vamos utilizar o produto cartesiano especial definido em 1).

$$R[A \div B]S = \{r.\overline{A} \mid r\epsilon R \wedge (r.\overline{A} \otimes S.B \subseteq R)\}$$

(R.A e S.B são do mesmo grau) .

Portanto, $R[A \div B]S \subseteq R.\overline{A}$.

A divisão seleciona as tuplas de $R.\overline{A}$ que comparecem em R concatenadas com todas as tuplas de S.B.

A divisão pode ser definida através de operações já definidas:

$$R[A \div B]S = R.\overline{A} - ((R.\overline{A} \otimes S.B - R).\overline{A}) .$$

Exemplo de divisão aparece na figura 16.

Fig. 14

ALUNOS

nome	nº	curso
João	5	Eng
Orlando	10	Mat
Ney	11	Fis

ALUNAS

nome	nº	curso
Maria	6	Quim
Alice	12	Fis

ESTUDANTES = ALUNOS ∪ ALUNAS

nome	nº	curso
João	5	Eng
Orlando	10	Mat
Ney	11	Fis
Maria	6	Quim
Alice	12	Fis

R

P1	5
P2	7
P3	2

S

A2	V	2
A6	X	2

R ⊗ S

P1	5	A2	V	2
P2	7	A2	V	2
P3	2	A2	V	2
P1	5	A6	X	2
P2	7	A6	X	2
P3	2	A6	X	2

ENG

Ney
Joaquim
Nelson

MAT

Estanislau
Joaquim
Marcos
Orlando

ENG ∩ MAT

Joaquim

ENG - MAT

Ney
Nelson

ESTUDANTES . (NOME,CURSO) ou
ESTUDANTES [1,3]

nome	curso
João	Eng
Orlando	Mat
Ney	Fis
Maria	Quim
Alice	Fis

ESTUDANTES [3]

curso
Eng
Mat
Fis
Quim

Fig. 15

FUNC

Nº	ANO	MES	DIA
1	74	2	9
2	75	12	12
3	73	1	14
4	77	2	11

FUNC [(2,3,4) < (2,3,4)] FUNC

Nº	ANO	MES	DIA	Nº	ANO	MES	DIA
1	74	2	9	2	75	12	12
1	74	2	9	4	77	2	11
2	75	12	12	4	77	2	11
3	73	1	14	1	74	2	9
3	73	1	14	2	75	12	12
3	73	1	14	4	77	2	11

R1

Nº	CURSO
53	ENG
22	FIS
11	FIS

R2

Nº	SEM
22	5
33	1
41	1
11	3

R1[1 * 1] R2

Nº	CURSO	SEM
22	FIS	5
11	FIS	3

Fig. 16

ALUNOS

Nº	DISC
51	MAP-111
43	MAP-111
22	MAP-111
22	MAP-121
43	MAP-121
22	MAP-211

DISC

SIGLA
MAP-111
MAP-121
MAP-211

ALUNOS [2 ÷ 1] DISC

Nº
22

Fig. 17

FUNC

Nome	Horas-P1	Horas-P2
Hugo	21	19
Mario	0	40
Chico	40	0
Fernando	20	20
Luis	31	9

FUNC [2 > 3]

Nome	Horas-P1	Horas-P2
Hugo	21	19
Chico	40	0
Luis	31	9

8) RESTRIÇÃO.

É uma operação efetuada sobre uma relação e que seleciona tuplas de acordo com comparações entre certos elementos dessas tuplas.

Seja θ um dos operadores relacionais $=$, $\neq$, $>$, $<$, $\geq$ ou $\leq$.

A restrição de R, segundo θ, sobre as listas A e B, é definida por:

$$R[A\ \theta\ B] = \{r \mid r \in R \wedge (r.A\ \theta\ r.B)\}$$

onde R.A e R.B devem ter o mesmo grau e todo r.A seja comparável, segundo θ, com todo r.B .

A restrição pode ser definida através de junções e projeções:

$$R[A\ \theta\ B] = (R[(A,B) * (A,B)]\ ((R.A)[A\ \theta\ B](R.B))) . C_k,$$

onde C_k é a lista dos domínios de R, na mesma ordem.

E a junção pode, agora, ser definida através do produto cartesiano especial e da restrição:

$$R[A\ \theta\ B]S = (R \otimes S)[A\ \theta\ B] .$$

Exemplo de restrição aparece na figura 17.

A seguir apresentamos uma série de consultas a uma base de dados, formuladas através de expressões dá álgebra de relações.

Nas expressões algébricas, as prioridades de operações podem ser indicadas por parênteses. Para evitar a proliferação exagerada dos parênteses, porém, define-se a seguinte ordem de prioridade:

1. Restrição e projeção.
2. Divisão, junção e produto cartesiano.
3. União, intersecção, diferença.

e, para operações do mesmo nível de prioridade, a ordem que prevalece é a da esquerda para a direita.

Uma linguagem (ou sublinguagem) baseada na álgebra de relações pode ainda conter, entre outros, recursos para:

1. Inserir novas tuplas em relações já existentes: isso pode ser obtido através de um comando que execute uma operação de união.

   Ex: (com uma relação R contendo 4 domínios)

   INSERT R , (<C1/'35', C3/'João', C4/'P3'>) * .

---

* Este tipo de notação será usada também para indicar constantes (que são relações com um domínio e uma tupla).

A omissão de C2 indica que não será inserido valor para o elemento correspondente ao 2º domínio.

2. Eliminar tuplas de relações já existentes: pode-se utilizar um comando que execute uma operação de diferença.
   Ex: DELETE R , (<C1/'35'>) .

3. Atualizar tuplas: pode ser obtido através de combinação adequada de inserções e eliminações.

4. Expressar funções muito utilizadas em consultas, como: soma ou média de elementos de um domínio, contagem do nº de tuplas de uma relação, máximos, mínimos, etc, além das funções e operações matemáticas usuais.

5. Definir relações auxiliares (temporárias).

Um problema de notação pode surgir quando se utiliza os nomes dos domínios e não a sua ordem na relação: pode-se evitar nomes repetidos de domínios em uma mesma relação e, até mesmo, de uma relação para outra (e, com isso, evitar a obrigatoriedade de usar nomes de "papéis"); na junção de uma relação com ela própria, porém, isso já não é mais possível. Nos exemplos que se seguem, esse problema foi contornado com indicações da ordem em que esses nomes repetidos aparecem na relação.

Ex: R(Nome , Idade) .

Em R[Idade > Idade]R, aparecem dois domínios "Idade". Se houver necessidade, serão usados Idade(1) e Ida-

de(2), onde, de acordo com a definição de junção, Idade(1) se refere ao domínio mais à esquerda, isto é, aquele onde aparece o valor maior (em cada tupla).

A parte do modelo sobre a qual são feitas as consultas em questão é:

PF(NOME, DEP, SAL) - professor

PP(NPROJ, NPROF) - professor, projeto (NPROJ: nº do projeto, NPROF: nome do professor)

PJ(NPROJ, AREA) - projeto (NPROJ: nº do projeto).

D(SIGLA, CHEFE) - departamento.

(1) "Nomes de professores que trabalham em algum projeto"

PP.NPROF

(2) "Nomes e salários de professores que não trabalham em nenhum projeto"

(PF[NOME * NOME](PF.NOME - PP.NPROF)).(NOME,SAL)

(3) "Nomes de professores que trabalham no projeto nº 7"

(PP[NPROJ * C1]<'7'>).NPROF

(4) "Nomes de professores que trabalham em algum projeto, mas não no nº 7"

(1) - (3) ((1) e (3) indicam as relações resultantes das consultas (1) e (3), respectivamente)

(5) "Nomes de professores que não trabalham no projeto nº 7"

PF.NOME - (3)

(6) "Nomes de professores que trabalham em outro projeto além do nº 7"

((PP[NPROJ ≠ C1]<'7'>) . NPROF) ∩ (3)

(7) "Nomes dos professores que trabalham apenas no projeto nº 7"

(3) - (6)

(8) "Nomes dos professores que trabalham, pelo menos, nos projetos nºs 5, 7 e 8"

(PP[NPROJ ÷ C1]{<'5'> , <'7'> , <'8'>}) .

(9) "Nomes dos professores que trabalham em todos os projetos"

(PP[NPROJ ÷ NPROJ]PJ)

(10) "Nomes dos professores que trabalham nos projetos nºs 5, 7, 8 e em nenhum outro

(8) - (PP[NPROJ * NPROJ](((PJ[NPROJ ≠ C1]<'5'>)[NPROJ ≠ C1]<'7'>)[NPROJ ≠ C1]<'8'>)) . NPROF

Esta expressão seria simplificada substituindo-se a parte assinalada por uma relação auxiliar.

(11) "Nomes dos professores que trabalham no projeto nº 7 e

ganham mais do que 20K"

((3)[NPROF * NOME](PF[SAL > C1]<'20K'>)) . NPROF

(12) "Nomes de todos os professores"

PF . NOME

(13) "Nome(s) do(s) professor(es) que recebe(m) o maior salário"

((PF[SAL ≥ SAL]PF)[NOME(2) ÷ NOME]PF) . NOME

ou

PF[SAL * C1]<MAX(PF.SAL)>

(14) "Nomes dos professores que trabalham em pelo menos um projeto em que o professor P1 trabalha"

(PP[NPROJ * NPROJ](PP[NPROF * C1]<'P1'>)) . NPROF

(15) "Nomes dos professores que trabalham em, pelo menos, todos os projetos em que o professor P1 trabalha"

PP[NPROJ ÷ NPROJ](PP[NPROF * C1]<'P1'>)

(16) "Nomes de todos os professores ativos e as áreas em que trabalham"

(PP[NPROJ * NPROJ]PJ) . (NPROF,AREA)

(17) "Nomes dos professores que trabalham sob a chefia do professor P1 e ganham mais que ele"

(PF[DEP * SIGLA](D[CHEFE * C1]<'P1'>)
[SAL > SAL](PF[NOME * C1]<'P1'>)) . NOME(1)

(18) "Nomes dos professores que trabalham no departamento MAP e ganham mais dos que todos os professores do departamento MAT"

(PF[DEP * C1]<'MAP'>[SAL > SAL](PF[DEP * C1]<'MAT'>)
[NOME(2) ÷ NOME](PF[DEP * C1]<'MAT'>)) . NOME

ou

(PF[DEP * C1]<'MAP'>[SAL > C1]
<MAX((PF[DEP * C1]<'MAT'>) . SAL)>) . NOME

(19) "Nomes dos professores que ganham mais do que seus chefes"

((PF[DEP * SIGLA] D [CHEFE * NOME]PF)
[SAL(1) > SAL(2)]) . NOME(1)

(20) "Nomes dos professores que são chefes de departamentos onde o salário mínimo é menor do que 5K ou o salário máximo é maior do que 20K .

(PF[NOME * CHEFE]((D [SIGLA * DEP] PF[SAL < C1]<'5K'>) ∪
∪ (D[SIGLA * DEP] PF[SAL > C1]<'20K'>))) . NOME(1)

Observação: Na forma aqui apresentada, a linguagem não permite formular uma consulta análoga, onde "salário mínimo" fosse substituído por "salário médio". Se a álgebra estiver contida em uma linguagem de programação, esta pode efetuar o

trabalho. Senão, a linguagem deveria ser estendida (ver, por exemplo, o "BY" da linguagem QUEL, no cálculo de relações, adiante).

(21) "Nomes dos professores que trabalhem no MAP e ganhem mais do que o chefe do MAT"

(PF[DEP * C1]<'MAP'>[SAL > SAL]
(PF[NOME * CHEFE](D[SIGLA * C1]<'MAT'>))) . NOME(1)

(22) "Soma dos salários pagos a professores ativos (em projetos) *

Suponhamos, para ilustrar, que a relação PP contenha ainda um domínio PORC, que indique a porcentagem de tempo que o professor dedica ao projeto. A resposta à consulta teria que ser, então, uma somatória dos produtos de cada porcentagem pelo salário do respectivo professor

(Σ(porc × sal)) .

Esta consulta também não pode ser formulada nesse estado de linguagem. Poder-se-ia formar a nova relação:

PP[NPROF * NOME]PF

mas deve haver um recurso adicional que permita somar os produtos de 2 domínios, ou que permita formar um novo domínio com os valores desses produtos.

---

* soma dos salários pagos a professores pelo trabalho em projetos.

CÁLCULO DE RELAÇÕES

Em [CODD 72a] é definido um cálculo de predicados para ser utilizado na formulação de consultas a uma base de dados constituída de relações na 1ª forma normal. A maior parte das tentativas de implementação de SABDs baseados no modelo relacional usa o cálculo relacional * como base para linguagens de consulta (QUEL [HELD 75], SQUARE [BOYC 75], SEQUEL [CHAM 74] são exemplos destas linguagens).

Em linhas gerais, as consultas formuladas através do cálculo relacional constituem-se nas chamadas expressões-α. Codd estabelece uma equivalência entre o potencial que as expressões-α têm em consultar um modelo relacional, com o que chama de "completude relacional" de linguagens, ou seja, a capacidade de uma linguagem extrair todo o conteú do informacional presente em um modelo relacional.

Uma expressão-α é definida a partir de termos e fórmulas.

DEFINIÇÃO: Um termo de alcance tem a seguinte forma

$$P_j r_i \qquad i,j = 1,2,3,...$$

onde P_j é um predicado e r_i é uma variável.

O significado de um termo de alcance pode ser enten

---

* usaremos o nome consagrado por Codd ("relational calculus") para identificar este cálculo de predicados aplicado ao modelo relacional.

dido a partir de:

a) Cada predicado da forma P_j está em correspondência biunívoca com uma relação R_j $(j = 1,2,...,M)$ de um modelo relacional constituído por M relações.

b) As variáveis representam tuplas de relações.

c) Um termo de alcance $P_j r_i$ estabelece o campo de uma variável r_i como sendo a relação R_j.

DEFINIÇÃO: Um <u>termo de junção</u> tem a seguinte forma

$$A\,\theta\,B \quad \text{ou} \quad A\,\theta\,k$$

onde A,B são variáveis indexadas da forma $r_j[I]$ $(I = 1,2,3,...)$

k é uma constante

e θ é um dos símbolos $=, \neq, <, >, \leq,$ ou $\geq$.

Significado:

a) $r_j[I]$ corresponde ao I-ésimo componente da tupla r_j.

b) k e θ correspondem ao significado usual.

Um termo de cálculo relacional só pode ser um termo de alcance ou um termo de junção.

DEFINIÇÃO: As <u>fórmulas bem-formadas</u> ("well formed formulae") do cálculo relacional são definidas a seguir, de maneira recursiva. Como todas as demais fórmulas, que não obedecem à definição, não são de interesse aqui

e não serão abordadas neste trabalho, a palavra "fórmula" será usada como abreviação para "fórmula bem-formada".

1. Um termo é uma fórmula.
2. Se Γ é fórmula, $\neg\Gamma$ é fórmula.
3. Se Γ_1 e Γ_2 são fórmulas, $(\Gamma_1 \vee \Gamma_2)$ é fórmula, $(\Gamma_1 \wedge \Gamma_2)$ é fórmula.
4. Se Γ é uma fórmula em que r_i ocorre como <u>variável livre</u> (isto é, não aparece precedida imediatamente pelos símbolos $\forall$ ou $\exists$ e nenhum termo de alcance em que participe aparece precedido por $\forall$ ou $\exists$ *), então

   $\exists\, r_i(\Gamma)$ é fórmula e

   $\forall\, r_i(\Gamma)$ é fórmula, respeitadas as convenções usuais que evitam duplicações de uso

   $\exists, \forall, \vee, \wedge, \neg$ têm seus significados usuais.

Com esta definição fica estabelecida uma primeira restrição às fórmulas do cálculo que interessam às expressões-α .

O passo seguinte é a obtenção, entre as fórmulas, daquelas em que toda variável tem o seu campo de alcance bem definido.

DEFINIÇÃO: <u>Uma fórmula de alcance sobre r_i</u> é uma fórmula com as seguintes características:

---

* o oposto é <u>variável limitada</u>.

a) sem quantificadores

b) todos seus termos são termos de alcance

c) tem apenas uma variável livre - r_i

d) $\neg$ só ocorre precedido por $\wedge$

e) se r_i ocorre em mais de um termos de alcance, estes termos devem estar associados a relações com o mesmo número de domínios.

A característica d) elimina a possibilidade de uma fórmula como $\neg P_j r_j$, que apenas afirma que R_j não está no campo de r_j.

DEFINIÇÃO: Uma fórmula delimitadora de alcance tem a seguinte forma

$$A_1 \wedge A_2 \wedge \ldots \wedge A_n \wedge F \text{ , onde}$$

1) $n \geq 1$

2) $A_1, \ldots, A_n$ são fórmulas de alcance sobre n variáveis distintas.

3) F é uma fórmula em que:

   a) $\forall$ e $\exists$ sempre precedem uma fórmula de alcance sobre uma variável; esta variável não é nenhuma das n variáveis a que se referem $A_1, \ldots, A_n$.

   b) as variáveis livres têm seus campos especificados em $A_1, A_2, \ldots, A_n$.

   c) não existem termos de alcance (exceto os necessários em a)).

Finalmente, então:

DEFINIÇÃO: Uma <u>expressão-α</u> é definida por

1) Uma expressão da forma

$$(v_1,\ldots, v_k) : f, \text{ onde}$$

cada v_i $(i = 1,\ldots, k)$ é uma variável (indexada ou não) e f é uma fórmula delimitadora de alcance,

é uma expressão-α ($(v_1,\ldots, v_k)$ é a <u>lista-objetivo</u>).

2) Se $(v_1,\ldots, v_k) : f_1$ e $(v_1,\ldots, v_k) : f_2$ são expressões-α, então

$$(v_1,\ldots,v_k) : (f_1 \vee f_2)$$
$$(v_1,\ldots,v_k) : (f_1 \wedge f_2)$$
$$(v_1,\ldots,v_k) : (f_1 \wedge \neg f_2) \quad \text{são expressões-α}$$

3) Nenhuma outra expressão é expressão-α .

Da mesma forma que na álgebra de relações, uma expressão-α, $(v_1,\ldots,v_k) : f$, representa uma relação, o resultado da consulta.

Esta relação é o conjunto

$$\{(v_1 \frown v_2 \frown \ldots \frown v_k) \mid f\} .$$

De outro modo, se $r_1,\ldots,r_n$ são n variáveis (não indexadas) distintas que aparecem em $(v_1,\ldots,v_k)$, e que são associadas às relações $R_1,\ldots,R_n$ (não necessariamente distintas), então:

$(v_1,\ldots,v_k)$: f é uma projeção (sobre os domínios indicados pela lista-objetivo) da relação formada pela concatenações de tuplas de $R_1,\ldots, R_n$ que satisfazem f.

Segue-se uma lista de consultas formuladas através de expressões-α do cálculo de relações definido. *

(1) $r_2[2] : P_2r_2$

(2) $(r_1[1],r_1[3]) : P_1r_1 \wedge \forall P_2r_2(r_2[2] \neq r_1[1])$

(3) $r_2[2] : P_2r_2 \wedge r_2[1] = 7$

(4) $r_2[2] : P_2r_2 \wedge \forall P_2r_3(r_2[1] \neq 7 \wedge (r_3[2] \neq r_2[2] \vee r_3[1] \neq 7))$

(5) $r_1[1] : P_1r_1 \wedge \forall P_2r_2(r_1[1] \neq r_2[2] \vee r_2[1] \neq 7)$

(6) $r_2[2] : P_2r_2 \wedge r_2[1] = 7 \wedge \wedge \exists P_2r_3(r_3[2] = r_2[2] \wedge r_3[1] \neq 7)$

(7) $r_2[2] : P_2r_2 \wedge r_2[1] = 7 \wedge \wedge \forall P_2r_3(r_3[2] \neq r_2[2] \vee r_3[1] = 7)$

---

* São as mesmas consultas apresentadas para a álgebra, com a mesma numeração, o modelo é o mesmo, tendo sido feita a seguinte correlação de nomes para as relações:

PF → R1 PJ → R3
PP → R2 D → R4

(8) $r_2[2] : P_2r_2 \wedge r_2[1] = 5 \wedge$

$\wedge \exists P_2r_3(r_3[2] = r_2[2] \wedge r_3[1] = 7) \wedge$

$\wedge \exists P_2r_4(r_4[2] = r_2[2] \wedge r_4[1] = 8)$

(9) $r_1[1] : P_1r_1 \wedge \forall P_3r_3 \exists P_2r_2(r_2[1] = r_3[1] \wedge$

$\wedge r_2[2] = r_1[1])$

(10) é a mesma expressão-α de (8), acrescida, à direita, por:

$\wedge \forall P_2r_5(r_5[2] \neq r_2[2] \vee r_5[1] = 5 \vee r_5[1] = 7 \vee r_5[1] = 8)$

(11) $r_1[1] : P_1r_1 \wedge \exists P_2r_2(r_2[2] = r_1[1] \wedge r_2[1) = 7 \wedge$

$\wedge r_1[3] > 20K)$

(12) $r_1[1] : P_1r_1$

(13) $r_1[1] : P_1r_1 \wedge \forall P_1r_2(r_1[3] \geq r_2[3])$

(14) $r_2[2] : P_2r_2 \wedge \exists P_2r_3(r_3[2] = \text{'P1'} \wedge r_3[1] = r_2[1])$

(15) $r_1[1] : P_1r_1 \wedge \forall P_2r_2 \exists P_2r_3(r_2[1] \neq \text{'P1'} \vee$

$\vee (r_3[1] = r_2[1] \wedge r_3[2] = r_1[1]))$

(16) $(r_1[1],r_3[2]) : P_1r_1 \wedge P_3r_3 \wedge$

$\wedge \exists P_2r_2(r_2[2] = r_1[1] \wedge r_2[1] = r_3[1])$

(17) $r_1[1] : P_1r_1 \wedge \exists P_4r_4 \exists P_1r_2(r_1[2] = r_4[1] \wedge$

$\wedge r_4[2] = \text{'P1'} \wedge r_2[1] = \text{'P1'} \wedge r_1[3] > r_2[3])$

(18) $r_1[1] : P_1r_1 \wedge \forall P_1r_2(r_1[2] = \text{'MAP'} \wedge$

$\wedge (r_2[2] \neq \text{'MAT'} \vee r_1[3] > r_2[3])$ *

(19) $r_1[1] : P_1r_1 \wedge \exists P_4r_4 \exists P_1r_2(r_1[2] = r_4[1] \wedge$

$\wedge r_2[1] = r_4[2] \wedge r_1[3] > r_2[3])$

(20) $r_1[1] : P_1r_1 \wedge \exists P_4r_4 \exists P_1r_2(r_1[1] = r_4[2] \wedge$

$\wedge r_2[2] = r_4[1] \wedge (r_2[3] < 5K \vee r_2[3] > 20K))$

(21) $r_1[1] : P_1r_1 \wedge \exists P_1r_2 \exists P_4r_4 (r_4[1] = \text{'MAT'} \wedge$

$\wedge r_4[2] = r_2[1] \wedge r_1[2] = \text{'MAP'} \wedge r_1[3] > r_2[3])$

(22) A exemplo do caso da álgebra, a formulação desta consulta necessita de extensões à linguagem que serão ilustradas na linguagem QUEL (o mesmo acontece com a consulta (20), se "salário mínimo" fosse substituído

---

* Se não houver professores no MAT, a resposta a esta consulta será a lista dos nomes de <u>todos</u> os professores do MAP. A origem desse fato está em que

$$r_2[2] \neq \text{'MAT'} \vee r_1[3] > r_2[3]$$

é logicamente equivalente a uma implicação:

$$r_2[2] = \text{'MAT'} \Longrightarrow r_1[3] > r_2[3] ,$$

que é "falsa" sempre que o termo da esquerda é "falso" independentemente do termo da direita. O usuário de uma linguagem baseada no cálculo deve estar alerta a problemas como este.

por "salário médio", por exemplo).

A apresentação das consultas, formuladas através do cálculo relacional, segue aqui a forma definida em [CODD 72a], que se aproxima ao máximo das notações consagradas para cálculos de predicados.

No cálculo deste cálculo relacional, porém, algumas simplificações poderiam ser introduzidas, tendo por objetivo uma maior afinidade com o significado das consultas, ou seja, uma maior "legibilidade".

Por exemplo, os predicados da forma P_j têm sempre o mesmo significado, ou seja, "a variável que os segue (r_i,v.g.) é uma tupla da relação R_j". A notação poderia ser, então, $r_i \epsilon R_j$.

Da mesma forma, os próprios R_j poderiam ser substituídos pelos nomes das relações, bem como as indicações de domínios.

Um par de chaves, envolvendo a consulta, segere melhor a idéia de se estar buscando um conjunto de valores.

Eis algumas das consultas já apresentadas, escritas sob esta nova forma:

(1) $\{r_2[\text{NPROF}] : r_2 \epsilon \text{PP}\}$

(3) $\{r_2[\text{NPROF}] : r_2 \epsilon \text{PP} \wedge r_2[\text{NPROJ}] = 7\}$

(6) $\{r_2[\text{NPROF}] : r_2 \epsilon \text{PP} \wedge r_2[\text{NPROJ}] = 7 \wedge$

$\wedge \exists\, r_3 \epsilon \text{PP}(r_3[\text{NPROF}] = r_2[\text{NPROF}] \wedge$

$$\wedge\ r_3[\text{NPROJ}] \neq 7)\}\ .$$

Outra contribuição à "legibilidade" das consultas poderia ser a introdução da implicação lógica ($\Longrightarrow$) e da negação do quantificador existencial ($\nexists$) . Essa modificação não traria conseqüências quanto aos resultados obtidos, já que "$\Longrightarrow$" e "$\nexists$" têm equivalentes lógicos na notação primitiva:

$\Gamma_1 \Longrightarrow \Gamma_2$ é logicamente equivalente a $\neg\Gamma_1 \vee \Gamma_2$

$\nexists\ P_j r_i(\Gamma)$ é logicamente equivalente a $\forall\ P_j r_i(\neg\Gamma)$

(2) "Nomes e salários de professores que não trabalhem em nenhum projeto" seria mais claramente expressa por:

$$\{(r_1[\text{NOME}], r_1[\text{SAL}]) : r_1 \in \text{PF} \wedge \nexists\ r_2\ \text{PP}(r_2[\text{NPROF}] = r_1[\text{NOME}])\}$$

(4) "Nomes de professores que trabalhem em algum projeto mas não no nº 7"

$$\{r_2[\text{NPROF}] : r_2 \in \text{PP} \wedge r_2[\text{NPROJ}] \neq 7 \wedge \nexists\ r_3 \in \text{PP}(r_3[\text{NPROF}] = r_2[\text{NPROF}] \wedge r_3[\text{NPROJ}] = 7)\}$$

ou

$$\{r_2[\text{NPROF}] : r_2 \in \text{PP} \wedge r_2[\text{NPROJ}] \neq 7 \wedge \forall\ r_3 \in \text{PP}(r_3[\text{NPROF}] = r_2[\text{NPROF}] \Longrightarrow$$

$$\Longrightarrow \; r_3[\text{NPROJ}] \neq 7)\}$$

(5) "Nomes de professores que não trabalhem no projeto nº 7"

$$\{r_1[\text{NOME}] : r_1 \in \text{PF} \wedge \nexists \; r_2 \in \text{PP}(r_2[\text{NPROF}] = r_1[\text{NOME}] \wedge$$

$$\wedge \; r_2[\text{NPROJ}] = 7)\}$$

(15) "Nomes dos professores que trabalhem em pelo menos todos os projetos em que o professor P1 trabalha"

$$\{r_1[\text{NOME}] : r_1 \in \text{PF} \wedge$$

$$\wedge \; \forall \; r_2 \in \text{PP} \; \exists \; r_3 \in \text{PP}(r_2[\text{NPROF}] = \text{'P1'} \Longrightarrow$$

$$\Longrightarrow (r_3[\text{NPROJ}] = r_2[\text{NPROJ}] \wedge$$

$$\wedge \; r_3[\text{NPROF}] = r_1[\text{NOME}])\}$$

ou

$$\{r_1[\text{NOME}] : r_1 \in \text{PF} \wedge \forall \; r_2 \in \text{PP}(r_2[\text{NPROF}] = \text{'P1'} \Longrightarrow \exists \; r_3 \in \text{PP}$$

$$(r_3[\text{NPROJ}] = r_2[\text{NPROJ}] \wedge$$

$$\wedge \; r_3[\text{NPROF}] = r_1[\text{NOME}]))\}$$

(18) "Nomes dos professores do departamento MAP que ganham mais que todos os professores do MAT"

$$\{r_1[\text{NOME}] : r_1 \in \text{PF} \wedge r_1[\text{DEP}] = \text{'MAP'} \wedge$$

$$\wedge \; \forall \; r_2 \in \text{PF}(r_2[\text{DEP}] = \text{'MAT'} \Longrightarrow$$

$$\Longrightarrow r_1[\text{SAL}] > r_2[\text{SAL}])\}$$

Em [CODD 72a], uma linguagem é chamada relacionalmente completa quando é, pelo menos, equivalente ao cálculo de relações, no sentido de que é capaz de formular todas as consultas possíveis de serem formuladas através de expressões-α .

Pode-se, então, mostrar que uma linguagem é relacionalmente completa, mostrando que qualquer expressão-α possui equivalente nessa linguagem.

Em [CODD 72a] é sugerido um algoritmo que transforma uma expressão-α em uma expressão da álgebra de relações, na tentativa de mostrar que a álgebra de relações também é relacionalmente completa.

III - ALGORITMO DE REDUÇÃO

A definição de expressão-α anteriormente apresentada é construída em três passos. O algoritmo de redução refere-se, inicialmente, às expressões-α do passo 1) e é, posteriormente, generalizado.

Apresentamos, inicialmente, uma idéia intuitiva do algoritmo e, em seguida, uma caracterização mais formal.

As expressões algébricas produzidas devem ser vistas como constituindo um procedimento. Essa característica das linguagens "algébricas" será citada quando da comparação entre álgebra e cálculo de relações.

1) Inicialmente são geradas as relações que correspondem aos

campos das variáveis que aparecem na expressão-α. Isso é feito aplicando-se as operações de união, intersecção e diferença às relações do modelo.

2) Forma-se o produto cartesiano (especial para relações) dessas relações (relação S).

3) Na relação S, mantêm-se apenas as tuplas que satisfazem os termos de junção da expressão-α, através de operações algébricas de redução, intersecção, união, etc.

4) Na relação resultante de 3), aplicam-se divisões e projeções formando uma nova relação que satisfaça as quantificações da expressão-α .

5) São efetuadas as projeções adequadas para atender às especificações da lista-objetivo.

Mais detalhadamente:

ALGORITMO A:

Seja a expressão-α

$$v' : f'$$

onde $v' = (v'_1,\ldots,v'_k)$ é a lista-objetivo;

$f' = A'_1 \wedge A'_2 \wedge \ldots \wedge A'_p \wedge F'$ é uma fórmula delimitadora de alcance;

existem $q\ (\geq 0)$ variáveis quantificadas em F';

todas as p variáveis livres de f' aparecem em v'.

A1. /A expressão-α original é transformada em uma outra equivalente (no sentido de denotar a mesma consulta), em uma forma mais adequada para o desenvolvimento do algoritmo/ .

a) converter F' para a forma normal prenexa ([CHUR 56]), sem expandir os quantificadores que possuem fórmulas de alcance.

b) converter a parte de F' isenta de quantificadores para a forma normal disjuntiva ([CHUR 56]) .

c) eliminar os ¬ que precedem termos de junção, substituindo-se os operadores relacionais $=, \neq, \geq, \leq, >, <$ por seus complementares $\neq, =, <, >, \leq, \geq$, respectivamente.

d) mudar os nomes das variáveis, atribuindo-lhes os nomes $r_1, r_2, \ldots, r_{p+q}$, de acordo com a ordem de sua primeira ocorrência em f'.

A nova expressão-α fica sendo:

$$v : f \quad \text{ou} \quad v : A_1 \wedge A_2 \wedge \ldots\ldots \wedge A_p \wedge F \ .$$

As variáveis livres de F são r_j, $j = 1,2,\ldots,p$ e as variáveis limitadas, r_j, $j = p+1, p+2, \ldots, p+q$, pois todas as variáveis livres aparecem antes em F (em $A_1, A_2, \ldots, A_p$) .

A variável r_j é associada a A_j $(j = 1,2,\ldots,p)$ e seja Q_j ($Q_j = \exists$ ou $\forall$) o quantificador associado a r_j $(j = p+1, p+2, \ldots, p+q)$.

A2. /Formar as equações que definem os S_j ($j = 1,\ldots, p+q$), relações correspondentes aos campos de r_j / .

Para cada variável r_j ($j = 1,\ldots, p+q$) formar uma equação

$$S_j = L_j$$

onde L_j é obtido a partir das fórmulas de alcance relativas a r_j, segundo as seguintes transformações:

a) $P_i r_j$ passa a R_i

b) $\vee$ passa a $\cup$

c) $\wedge\neg$ passa a $-$

d) $\wedge$ passa a $\cap$ quando c) não pode ser aplicada .

A3. /Formar a relação S, um produto cartesiano que engloba os campos de todas as variáveis presentes em F/ .

Para cada relação R de um modelo relacional,

DEFINIÇÃO: Seja U(R) a união de todas as relações do modelo que possuem o mesmo número de domínios que R

e convenciona-se

a) $U(T) = U(R)$ para todo $T \subseteq R$

b) $U(R_i \cup R_j) = U(R_i) \cup U(R_j)$

Então

$$S = X_1 \otimes X_2 \otimes \ldots \otimes X_{p+q}$$

onde

$$X_j = \begin{cases} S_j & \text{se } r \text{ é livre ou quantificada existencialmente ou} \\ U(S_j), & \text{em caso contrário.} \end{cases}$$

Esta definição de S ao invés de, simplesmente,

$$S = S_1 \otimes S_2 \otimes \ldots \otimes S_{p+q}$$

previne o caso em que $S_j = \emptyset$ e r_j é quantificada por $\forall$ (de acordo com a última definição, S seria vazia, bem como o resultado da consulta).

Seja o grau de X_j, g_j $(j = 1,2,\ldots, p+q)$.

Então, defina-se

$$o_j = \sum_{i=1}^{j-1} g_i \ , \quad (o_i = 0)$$

que indica a origem da contagem dos domínios correspondentes a X_j, em S. Por exemplo, o z-ésimo domínio de X_j corresponde ao z'-ésimo domínio de S, onde

$$z' = o_j + z$$

A4. /Eliminar de S as tuplas que não satisfaçam os termos de junção de F/ .

Eliminar, de F, os quantificadores .

Se o que restar for vazio, formar a equação

$$T_{p+q} = S$$

senão, formar uma equação de definição para T_{p+q}, onde o outro termo é obtido a partir de F (sem os quantificadores)

de acordo com as seguintes transformações:

a) $\vee$ passa a $\cup$

b) $\wedge$ passa a $\cap$

c) $r_j[z]\ \theta\ r_\ell[w]$ passa a $S[(o_j + z)\ \theta\ (o_\ell + w)]$ (restrição)

d) $r_j[z]\ \theta\ K$ passa a $S[(o_j + z)\ \theta\ 1]\{K\}$ (junção)

onde K é uma constante.

A5. /Formar equações que exprimam os limites impostos pelos quantificadores/ .

Formar equações de definição para T_{p+q-1}, T_{p+q-2},....., T_p, correspondentes aos quantificadores Q_{p+q}, Q_{p+q-1},..., Q_{p+1}, respectivamente, isto é, numa ordem que vai do mais interno para o mais externo.

Para $j = p+q, p+q-1, \ldots, p$

$$T_{j-1} = T_j[1,2,\ldots, o_j] \quad \text{se} \quad Q_j = \exists$$

(isto é, a existência ou não de uma tupla de X_j satisfazendo uma certa condição já foi levada em conta em A4; esta equação, então, simplesmente elimina de T_j os domínios correspondentes aos de X_j, através de uma operação de projeção, pois r_j, sendo uma variável limitada, encerrou o seu papel),

$$\text{ou} \quad T_{j-1} = T_j[E_j \div D_j]S_j \quad \text{se} \quad Q_j = \forall$$

onde $E_j = (o_j+1, o_j+2, \ldots, o_j+g_j)$

e $D_j = (1,2,\ldots, g_j)$

(também elimina de T_j os domínios correspondentes aos de X_j, mas, dessa projeção são mantidas apenas as tuplas que, em T_j, apareciam concatenadas a todas as tuplas de S_j, de acordo com a exigência do quantificador $\forall$) .

A6. /Formar a equação que leve em conta a lista-objetivo/ .

$$Z = T_p[H_1, H_2, \ldots, H_k]$$

onde, H_i é $o_j+1, o_j+2, \ldots, o_{j+1}$

se v_i em F' corresponde a r_j em F (variável não indexada) ou

$$H_i \text{ é } o_j + z$$

se v_i em F' corresponde a $r_j[z]$ em F (variável indexada) .

A7. /Voltar aos nomes da expressão-α original/ .

Formar uma expressão algébrica para definir Z, reintroduzindo os nomes originais das relações do modelo (R_1, R_2, etc.) e os números de ordem dos seus domínios, e eliminando os X_i ($i = 1, 2, \ldots, p+q$), S_i ($i = 1, 2, \ldots, p+q$), S, T_i ($i = p, \ldots, p+q$), $U(R_i)$ (valores de i de acordo com o que foi convencionado).

Para estender esse algoritmo a uma expressão-α qualquer, falta considerar o caso em que a parte da expressão-α à direita do sinal ":" (f), é formada não por uma simples fórmula delimitadora de alcance mas por mais de uma, unidas por conectivos lógicos. Nesse caso, aplica-se o algoritmo de redução, isoladamente para cada

$$v : f_i \qquad i = 1, 2, \ldots, m$$

onde m é o número de fórmulas delimitadoras de alcance presentes.

Teremos então m equações definindo $Z_1, Z_2, \ldots, Z_m$.

Forma-se, então, uma equação para definir Z, onde o lado direito é formado a partir da expressão de f, segundo as seguintes transformações:

a) $\vee$ passa a $\cup$

b) $\wedge\neg$ passa a $-$

c) $\wedge$ passa a $\cap$ quando b) não pode ser aplicada

d) f_i passa para Z_i

e) cada Z_i é substituído pelo lado direito da equação que o define.

EXEMPLO: Relações no modelo

$R_1(\overset{1}{\text{Nº-PROJ}}, \overset{2}{\text{NOMPROF}})$ - professor

$R_2(\overset{1}{\text{SIGLA}})$ - departamento

$R_3(\overset{1}{\text{NOME}}, \overset{2}{\text{DEPTO}})$ - professor

Consulta: "Números dos projetos que possuem professores de todos os departamentos" .

Expressão-α: $r_1[1] : P_1r_1 \wedge \forall P_2r_2 \exists P_3r_3(r_1[2] = r_3[1] \wedge$
$\wedge r_2[1] = r_3[2])$

Algoritmo de redução:

A1. A expressão já está nas formas normais prenexa e disjuntiva e as variáveis têm denominações segundo a ordem exigida.

A2. $S_1 = R_1 \ (o_1 = 0 \ , \ g_1 = 2)$

$S_2 = R_2 \ (o_2 = 2 \ , \ g_2 = 1)$

$S_3 = R_3 \ (o_3 = 3 \ , \ g_3 = 2)$

A3. $S = S_1 \otimes S_2 \otimes S_3 \quad (U(S_2) = S_2)$

A4. $p = 1$ (número de variáveis livres - r_1)

$q = 2$ (número de variáveis limitadas - r_2 , r_3)

$T_{p+q} = T_3 = S[2 = 4] \cap S[3 = 5]$

(Restrições a S impostas pelos termos de junção $r_1[2] = r_3[1] \wedge r_2[1] = r_3[2]$)

A5. $T_2 = T_3[1,2,\ldots,o_3] = T_3[1,2,3]$, pois $Q_3 = \exists$

$T_1 = T_2[E_2 \div D_2]S_2$, pois $Q_2 = \forall$

mas $E_2 = (o_2+1, \ldots, o_2+g_2) = (3)$

e $D_2 = (1, \ldots, g_2) = (1)$

então $T_1 = T_2[3 \div 1]S_2$

A6. $Z = T_1[1]$, pois a lista-objetivo é $(r_1[1])$

A7. As expressões obtidas até agora são:

$S_1 = R_1$

$S_2 = R_2$

$S_3 = R_3$

$S = S_1 \otimes S_2 \otimes S_3$

$T_3 = S[2 = 4] \cap S[3 = 5]$

$T_2 = T_3[1,2,3]$

$T_1 = T_2[3 \div 1]S_2$

$Z = T_1[1]$

Formando-se, então, uma expressão algébrica única,

$$Z = T_1[1] = (T_2[3 \div 1]R_2)[1] = (T_3[1,2,3][3 \div 1]R_2)[1] =$$

$$= ((S[2 = 4] \cap S[3 = 5])[1,2,3][3 \div 1]R_2)[1] ,$$

e, portanto,

$$Z = (((R_1 \otimes R_2 \otimes R_3)[2 = 4] \cap$$

$$\cap (R_1 \otimes R_2 \otimes R_3)[3 = 5])[1,2,3][3 \div 1]R_2)[1] .$$

IV - COMPARAÇÃO

Uma breve comparação entre o cálculo e a álgebra de relações, como bases para linguagens de consulta, serve para ilustrar vários pontos que são, em geral, examinados na escolha de linguagens para um SABD.

É bem verdade que, no estado em que foram apresentados, tanto a álgebra quanto o cálculo carecem de uma série de extensões imprescindíveis em uma linguagem de manipulação de dados. Recursos para atualizar relações, armazenar e eliminar dados, funções especiais para cálculo de médias, máximos, contagens, etc são algumas dessas extensões. Mais adiante será apresentada, em linhas gerais, uma linguagem de manipulação de dados baseada no cálculo de relações, que incorpora algumas dessas extensões. Na comparação que se pretende fazer, porém, o papel dessas extensões ficará colocado.

Uma linguagem de consulta, no modelo relacional, é, como foi dito, uma selecionadora de novas relações. O resultado de uma consulta é uma relação formada a partir das relações presentes no modelo. Na formação desse conjunto, pode-se resumir as características da álgebra e do cálculo de relações, dizendo-se que a álgebra funciona "construtivamente", isto é, através de operações sucessivas sobre as relações originais e sobre relações auxiliares, chega ao resultado desejado. Já no cálculo, o conjunto-resultado é estabelecido através de sua "definição" sob a forma de um predicado.

Isso caracteriza a álgebra como uma linguagem do tipo "procedimento", mais próxima às linguagens de programação conhecidas, onde o programador é um construtor de seqüências de operações. No cálculo, o "programa" é baseado fundamentalmente nas propriedades dos dados e a sua execução é mais tarefa do sistema do que propriamente do usuário. Essa diferença gera algumas considerações:

a) a álgebra é de implementação mais fácil; o "know-how" para implementação de linguagens "procedimentais" é muito mais desenvolvido, enquanto que linguagens tipo cálculo são de "mais alto nível", isto é, diminuem a tarefa do usuário e aumentam a do sistema.

b) o cálculo facilita a obtenção de uma maior independência de dados, já que os dados desejados são pedidos segundo suas propriedades e fica a cargo do sistema a escolha do melhor caminho para obtê-los; na álgebra, o usuário pode se defrontar com uma série de alternativas e nem sempre escolher a melhor.

c) o final do item b) sugere também que o cálculo é de utilização mais fácil para leigos (não programadores); apesar da dificuldade, também, da linguagem de predicados, ela oferece campo mais propício para linguagens mais acessíveis a quem só conhece linguagens naturais: inúmeras linguagens têm sido propostas que, entre outras facilidades, eliminaram o uso de quantificadores e introduziram

termos em língua inglesa.

d) os dispositivos para autorização de uso são mais facilmente incorporados a uma linguagem que se refere aos dados através de suas propriedades, sem ter que prever as possíveis alternativas para a formulação de expressões algébricas.

e) quanto às extensões mencionadas anteriormente, [CODD 72a] sugere três lugares potenciais para acomodar funções especiais:
   - na lista-objetivo, possibilitando transformações na relação-resultado.
   - substituir um termo de junção por uma função lógica.
   - substituir uma variável indexada de um termo de junção por uma função (cujo resultado é um número ou uma cadeia, etc.).

No contexto da álgebra, essas funções teriam que ser incorporadas através da definição de novas operações,o mesmo acontecendo com as funções de atualização, armazenamento e eliminação.

V - QUEL

QUEL é a linguagem de consulta do INGRES, um SABD baseado no modelo relacional, projetado e implementado na Universidade da Califórnia ([HELD 75]).

Esta linguagem é um exemplo de linguagem de manipulação baseada no cálculo de relações, acrescida de extensões do tipo já discutido anteriormente, e colocada sob forma que possibilite maiores facilidades para o usuário.

As principais alterações (e/ou extensões) são:

a) eliminação dos quantificadores.
b) introdução de palavras-chave em inglês.
c) possibilidade de atualização, armazenamento, eliminação, etc. de dados.
d) presença de inúmeras funções especiais.

A estrutura de uma consulta ("query") em QUEL é indicada por:

```
query := {range-statement} {retrieve-statement}
                        ({ } indica "um ou mais")

range-statement := RANGE OF {variable} IS relation

retrieve-statement := RETRIEVE [INTO result-name]
                      (target-list) [WHERE
                      qualification]
                  ([ ] indica 0 ou 1 ocorrências)

target-list := {result-domain = function}
```

Os "range-statement" fazem o papel que as fórmulas de alcance desempenham nas expressões-α, isto é, delimitam os campos de ação das variáveis. É uma delimitação mais

restrita, no sentido de que cada variável pode estar associada a uma só relação definida na base de dados (embora essa definição possa ter sido obtida através de algum "query" anterior).

Os "retrieve-statement" indicam o nome da relação-resultado ("result-name"), se for o caso, os domínios que a constituirão ("target-list") e as propriedades que identificam as tuplas da relação-resultado ("qualification").

Cada par "result-domain" - "function" na "target-list" indica o nome do novo domínio e a maneira como são obtidos seus valores. Esse par pode ser substituído por "variable.domain", apenas, o que significa que o novo domínio conserva o nome e o valor correspondente a algum domínio já existente em outra relação.

Seguem alguns exemplos de consultas em QUEL.

Modelo usado:

PROFESSOR (NOME, DEPTO, SAL)

PROJETO-PROFESSOR (PROJ, PROF)

(1) "Nomes e salários dos professores que trabalham em algum projeto".

```
RANGE OF P IS PROFESSOR
RANGE OF PF IS PROJETO-PROFESSOR
RETRIEVE INTO R (PF.PROF, P.SAL) WHERE PF.PROF = P.NOME
```

(2) "Nomes dos professores que não trabalham em nenhum projeto".

```
RANGE OF PF IS PROJETO-PROFESSOR
RETRIEVE INTO R (PF.PROF) WHERE COUNT
                              (PF.PROJ BY PF.PROF) = 0
```

A função COUNT representa o número de elementos do conjunto ao qual se aplica (no caso, PF.PROJ BY PF.PROF).

O conjunto PF.PROJ BY PF.PROF é o conjunto dos valores do domínio PROJ referentes a tuplas onde o domínio PROF tem o valor PF.PROF .

(3) "Nomes dos professores que trabalham nos projetos nº 5, 7 e 8".

```
RANGE OF PF IS PROJETO-PROFESSOR
RETRIEVE INTO R (PF.PROF) WHERE COUNT
                    (PF.PROJ BY PF.PROF WHERE PF.PROJ = 5
                     OR PF.PROJ = 7 OR PF.PROJ = 8) ≥ 3
```

Esse exemplo mostra a possibilidade do uso de uma qualificação com BY e se serve do fato de que o conjunto formado não contém duplicações.

Às vezes, porém, as duplicações devem ser mantidas:

(4) "Lista contendo departamento e o salário-médio de seus professores".

```
RANGE OF P IS PROFESSOR
```

```
RETRIEVE INTO R (P.DEPTO, SALMED =
                = AVG'(P.SAL BY P.DEPTO))
```

AVG é a função que calcula a média de um conjunto sem duplicações; AVG' mantém as duplicações no cálculo da média. O mesmo acontece com COUNT', SUM' (soma), etc.

(5) "Professores que trabalham em pelo menos um projeto em que o professor "Arno" trabalha".

```
RANGE OF PF1 IS PROJETO-PROFESSOR

RANGE OF PF2 IS PROJETO-PROFESSOR

RETRIEVE INTO R (PF1.PROF) WHERE PF1.PROJ =
                = PF2.PROJ AND PF2.PROF = 'ARNO'
```

Para efetuar outras tarefas de manipulação de dados, QUEL apresenta três comandos: REPLACE, DELETE, APPEND. A sintaxe que norteia seu emprego não necessita maiores esclarecimentos.

(6) "Os professores com salário maior que 30K passam a trabalhar no projeto nº 5".

```
RANGE OF P IS PROFESSOR

RANGE OF PF IS PROJETO-PROFESSOR

APPEND TO PROJETO-PROFESSOR (PROJ = 5, PROF = P.NOME)
                WHERE P.SAL > 30K
```

(7) "Aumentar o professor Arno em 20%".

```
RANGE OF P IS PROFESSOR

REPLACE P (SAL = 1.2* P.SAL) WHERE P.NOME = 'ARNO'
```

(8) "Eliminar de PROJETO-PROFESSOR as tuplas do professor Walter".

```
RANGE OF PF IS PROJETO-PROFESSOR

DELETE PF WHERE PF.PROF = 'WALTER'
```

QUEL possui ainda outros comandos, como DEFINE (para possibilitar ao usuário definir relações que correspondam ao seu modelo externo), como INTEGRITY CONSTRAINT (para introduzir condições de consistência ou segurança), etc., sobre os quais não entraremos em maiores detalhes *.

A completude relacional do QUEL pode ser examinada a partir das operações da álgebra de relações:

UNIÃO.

Álgebra $R \cup S$

QUEL (supondo que $D_1, D_2, \ldots, D_n$ sejam os domínios de R e $C_1, \ldots, C_n$ os de S)

```
RANGE OF VR IS R

RANGE OF VS IS S
```

---

* para maiores informações sobre QUEL, consultar bibliografia adicional, ao final deste trabalho.

RETRIEVE INTO W (E_1 = VR.D_1,..., E_n = VR.D_n)

APPEND TO W (E_1 = VS.C_1,..., E_n = VS.C_n)

INTERSECÇÃO.

Álgebra R ∩ S

QUEL RANGE OF VR IS R

RANGE OF VS IS S

RETRIEVE INTO W (E_1 = VR.D_1,..., E_n = VR.D_n)

WHERE VR.D_1 = VS.C_1 AND ... AND

VR.D_n = VS.C_n

DIFERENÇA.

Álgebra R-S

QUEL RANGE OF VR IS R

RANGE OF VS IS S

RETRIEVE INTO W (E_1 = VR.D_1,..., E_n = VR.D_n)

WHERE VR.D_1 ≠ VS.C_1 ... OR

VR.D_n ≠ VS.C_n

PROJEÇÃO.

Álgebra R.A *

---

* na bibliografia consultada não fica claro se, em QUEL, é possível indicar operações sobre uma lista de domínios, sem fazê-lo explicitamente para cada domínio. Acreditemos ser uma extensão de fácil incorporação à linguagem, possivelmente através das "attribute-functions" ([HELD 75]), que dão origem a valores que dependem apenas dos dados contidos em uma tupla.

QUEL RANGE OF VR IS R

RETRIEVE INTO W (VR.A)

PRODUTO CARTESIANO

Álgebra $R \otimes T$

QUEL RANGE OF VR IS R

RANGE OF VT IS T (onde T consiste de (F_1, ..., F_m))

RETRIEVE INTO W ($VR.D_1$,..., $VR.D_n$, $VT.F_1$, ..., $VT.F_m$)

JUNÇÃO

Álgebra $R[D_i \ \theta \ F_j]T \quad 1 \leq i \leq n \quad e \quad 1 \leq j \leq m$

QUEL RANGE OF VR IS R

RANGE OF VT IS T

RETRIEVE INTO W ($VR.D_1$,..., $VR.D_n$, $VT.F_1$, ..., $VT.F_n$) WHERE $VR.D_i \ \theta \ VT.F_j$ (Ver * na página anterior)

RESTRIÇÃO

Álgebra $R[D_i \ \theta \ D_j] \quad 1 \leq i \leq n \quad e \quad 1 \leq j \leq n$

QUEL RANGE OF VR IS R

RETRIEVE INTO W ($VR.D_1$,..., $VR.D_n$) WHERE $VR.D_i \ \theta \ VR.D_j$ (ver * na página anterior)

DIVISÃO.

Álgebra $R[D_i \div F_j]T \quad 1 \leq i \leq n \quad e \quad 1 \leq j \leq m$

QUEL

```
RANGE OF VR IS R
RETRIEVE INTO W (VR.D1,..., VR.Di-1,
                 VR.Di+1,..., VR.Dn) WHERE
                 COUNT (VR.Di BY (VR.D1,...,
                 VR.Di-1, VR.Di+1,..., VR.Dn)
                 WHERE VR.Di = VT.Fj  ≥
                 ≥ COUNT (VT.Fj)
```

(ver * na página 136)

A ESTRUTURAÇÃO DO MODELO

Uma série de critérios deve presidir o trabalho de estruturação de um modelo relacional de dados, já que não é única a solução para o problema de se encontrar as relações (e respectivos domínios) que podem constituir o modelo:

1) Independência de dados - este fator, como sempre, é da maior importância. O modelo deve ser estruturado tendo em vista o caráter dinâmico de uma base de dados. As relações devem comportar, com o máximo de flexibilidade, reestruturações impostas por essa dinamicidade, com um mínimo de impacto sobre os programas já existentes. Essas mudanças devem-se, principalmente, à introdução de novos tipos de dados e à alteração nas estatísticas de utilização (que pode comprometer a eficiência de uso).

2) Informação - o modelo deve se apresentar de forma a ser o mais informativo possível aos usuários. Desempenhando a função de modelo geral (ou modelo externo), o modelo relacional deve permitir que os vários tipos de usuários possam se familiarizar rapidamente com o seu conteúdo.

3) Dependências - as relações devem estar organizadas de modo a minimizar efeitos colaterais nas operações de inserção, eliminação e atualização.

4) Flexibilidade - o modelo deve facilitar operações de formação de relações desejadas pelos usuários, principalmente nos sistemas que permitam a definição de modelos externos.

Embora não haja provas formais sobre o assunto, há um certo consenso de que uma "coleção ótima de relações na 3ª forma normal" é a que melhor se adapta aos critérios acima.

[CODD 72b] definiu as 1ª, 2ª e 3ª formas normais para relações. Definições apresentadas posteriormente ([CODD 74] e Sharman, citado em [CHAM 76]) evitam essas etapas de normalização, concentrando-as em uma única definição. Basear-nos-emos, no entanto, no desdobramento proposto por Codd que dá campo para uma melhor compreensão do relacionamento entre a normalização e os objetivos anteriormente enumerados.

Dois novos conceitos devem ser introduzidos para

facilitar a compreensão do que se segue: dependência funcional e chave.

A partir deste ponto, ao invés de falarmos em domínios, passaremos a usar atributo , nome que representará a associação entre um domínio e o papel que este domínio desempenha no modelo. Na definição de modelo relacional foi dito que os domínios não necessitam ser distintos entre si. Do ponto de vista semântico, porém, domínios iguais, em geral, desempenham papéis distintos, daí a propriedade em se introduzir os atributos.

DEFINIÇÃO: Um atributo A_2 é dependente funcional de um atributo A_1, quando, em qualquer instante de tempo, existe uma função que leva do conjunto-de-valores-ativos referentes a A_1 ao conjunto-de-valores-ativos referentes a A_2. Em outras palavras, A_2 é dependente funcional de A_1 quando, em qualquer instante de tempo, a cada valor de A_1, efetivamente presente no modelo, pode estar associado, no máximo, um valor de A_2.

A dependência funcional tem como símbolo:

$$A_1 \longrightarrow A_2$$

Exemplo: um atributo DEPARTAMENTO é dependente funcional de um atributo NOME-DE-PROFESSOR (NOME-DE-PROFESSOR $\longrightarrow$ DEPARTAMENTO) em um modelo onde se representa o fato de que cada professor pertence a um só departamento, em cada instan

te (e supondo que não haja nomes repetidos de professores).

Ressalte-se a importância do trecho "em qualquer instante de tempo", na definição acima. Isso significa que se, em um determinado instante, for constatado que a cada valor de A_1 está associado um único valor de A_2, este fato não basta para consolidar a dependência funcional. Esta reflete considerações semânticas subjacentes ao modelo, que devem estar presentes sempre.

Por exemplo, seja, em um dado instante, a relação CARGA-DIDATICA (PROFESSOR, DISCIPLINA).

PROFESSOR	DISCIPLINA
P1	D5
P2	D7
P3	D6

Este "instantâneo" da relação, por si só, não implica na dependência funcional

PROFESSOR ⟶ DISCIPLINA

A dependência funcional deveria ser imposta pelo fato de, em qualquer instante, cada professor poder ministrar uma única disciplina.

O conceito de dependência funcional pode ser estendido para listas de atributos:

$$\{A_1, A_2, \ldots, A_m\} \longrightarrow \{B_1, \ldots, B_n\}$$

Exemplo:

$$\{SALA, HORA\} \longrightarrow \{DISCIPLINA, PROFESSOR\}$$

DEFINIÇÃO: Uma lista de atributos L tem <u>dependência funcional completa</u> de outra lista de atributos M se L é dependente funcional de M e não é dependente funcional de nenhum sub-conjunto próprio de atributos de M .

Exemplo: no exemplo anterior a dependência funcional indica que em uma sala, em uma certa hora, estão determinados o professor e a disciplina que a ocupam. Essa dependência funcional é completa, pois nem SALA, isoladamente, nem HORA, isoladamente, determinam {PROFESSOR, DISCIPLINA} . Isso pode ser indicado por:

$$\{SALA, HORA\} \longrightarrow \{PROFESSOR, DISCIPLINA\}$$

$$SALA \not\longrightarrow \{PROFESSOR, DISCIPLINA\}$$

$$HORA \not\longrightarrow \{PROFESSOR, DISCIPLINA\}$$

<u>Propriedade da dependência funcional</u>: [DELO 73]

1) Transitiva.

$$L \longrightarrow M,\ M \longrightarrow N \Longrightarrow L \longrightarrow N$$

2) Reflexiva

$$L \longrightarrow L$$

3) Projetiva.

$$L' \subset L \Longrightarrow L \longrightarrow L'$$

4) Aditiva.

$$L \longrightarrow M \quad e \quad L \longrightarrow N \Longrightarrow L \longrightarrow \{M, N\}$$

5) Pseudo-transitiva.

$$L \longrightarrow M \quad e \quad \{M, N\} \longrightarrow P \Longrightarrow \{L, N\} \longrightarrow P$$

6) Aumentativa.

$$L \longrightarrow M \Longrightarrow \{L, N\} \longrightarrow M, \text{ qualquer } N$$

L, M, N, P são listas de um ou mais atributos,

{M, N} representa uma lista constituída pelos atributos de M e pelos atributos de N.

DEFINIÇÃO: Uma lista K de atributos de uma relação R é uma <u>chave</u> de R, quando

$$K \longrightarrow U,$$

(onde U é a lista de todos atributos de R) e esta dependência funcional é completa.

Exemplo:

R (Nº-PEÇA, PROJETO, FORNECEDOR, QUANTIDADE, END-FORN, COR, CONTRATO, NOME-PEÇA, ESTOQUE)

Algumas das dependências funcionais entre estes atributos são:

Nº-PEÇA $\longrightarrow$ NOME-PEÇA

NOME-PEÇA $\longrightarrow$ Nº-PEÇA (não há nomes de peça repetidos)

FORNECEDOR ⟶ END-FORN (cada fornecedor tem um só endereço)

NOME-PEÇA ⟶ COR

Nº-PEÇA ⟶ COR

{FORNECEDOR, PROJETO} ⟶ CONTRATO

{FORNECEDOR, Nº-PEÇA} ⟶ ESTOQUE

{FORNECEDOR, NOME-PEÇA} ⟶ ESTOQUE

{FORNECEDOR, Nº-PEÇA, PROJETO} ⟶ QUANTIDADE (QUANTIDADE representa a quantidade de uma determinada peça fornecida a um determinado projeto por um determinado fornecedor)

etc.

É fácil de ver que apenas duas listas de atributos são chaves de R:

{FORNECEDOR, Nº-PEÇA, PROJETO} e

{FORNECEDOR, NOME-PEÇA, PROJETO}

pois são as únicas listas das quais QUANTIDADE é dependente funcional.

Da definição de chave, ressaltam algumas propriedades:

1) Todo atributo (ou lista de atributos) de R é dependente funcional de toda chave de R.

2) Não é necessário que cada atributo ou cada conjunto de atributos de R tenha dependência funcional completa da chave. A dependência funcional deve ser completa apenas para o conjunto de <u>todos</u> os atributos de R (cf. CONTRATO,

ESTOQUE, etc.).

3) Nenhuma lista de atributos L de R, tal que

$L \subset K$, é dependente funcional de qualquer outra lista de atributos L' de R, tal que

$L' \subset K$ e $L \not\subseteq L'$.

E mais, nenhum atributo de R pode ser introduzido em K sem tornar falsa essa propriedade.

4) Existe sempre pelo menos uma chave, em qualquer relação (em um caso extremo, sempre existe a possibilidade de K = U)

Prova de 1).

Suponhamos que a lista L de atributos de R não seja dependente funcional de uma chave K de R.

Mas, $L \subset U$, logo $U \longrightarrow L$ (propriedade projetiva).

$K \longrightarrow U$ (por definição de chave), logo

$$\left.\begin{array}{l} K \longrightarrow U \\ U \longrightarrow L \end{array}\right\} \Longrightarrow K \longrightarrow L \quad \text{(propriedade transitiva)}$$

e, portanto, a suposição é falsa.

Prova de 3).

Suponhamos que existam L e L' (listas de atributos de R) e uma chave K de R, tais que

$L \subset K$

$L' \subset K$

$L' \longrightarrow L$

$L \not\subseteq L'$

Seja A a lista dos atributos de K que não pertencem nem a L, nem a L' .

Por definição de chave, $K \longrightarrow U$, portanto $\{L,L',A\} \longrightarrow U$ (propriedade aumentativa) .

Se $L' \longrightarrow L$, então $\{L',L',A\} \longrightarrow U$ (propriedade pseudo-transitiva), isto é,

$$\{L',A\} \longrightarrow U,$$

ou seja, a dependência funcional de U em relação a K não é completa, contrariando a definição de chave.

Seja, agora, D um atributo de R tal que $D \notin K$ e seja $K' = \{K,D\}$.

Chamando-se K de L' e D de L, é fácil de ver que a propriedade não mais vale, em relação a K'.

1ª forma normal.

Uma relação está na 1ª forma normal quando seus atributos têm apenas valores simples, isto é, valores não decomponíveis em outros (por exemplo, um valor de um atributo não pode ser um conjunto com mais de um valores, não pode ser uma outra relação).

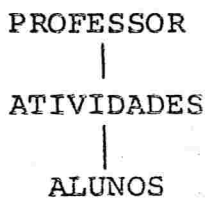
Este primeiro passo de normalização visa, tão so-

mente, dar uma forma simples, homogeneizada, às relações, em busca de simplicidade na visualização do modelo e de maiores facilidades para implementação.

Exemplo: Seja

relação	atributos
PROFESSOR	(NOME, DEPTO, ATIVIDADES)
onde, ATIVIDADES	(SEMESTRE, DISCIPLINA, ALUNOS)
e ALUNOS	(Nº, NOTA)

A relação PROFESSOR não está na 1ª forma normal. Um dos atributos de PROFESSOR é constituído por relações do tipo ATIVIDADES. Cada relação ATIVIDADES, por sua vez, também tem um atributo constituído por relações (ALUNOS). Está subjacente uma hierarquia entre relações que pode ser ilustrada por:



[CODD 72b] apresenta um procedimento bastante simples para trazer uma relação para a 1ª FN. O procedimento faz as transformações necessárias utilizando, em cada relação, uma chave (a chave primária) que, teoricamente, pode ser escolhida de modo arbitrário mas que, na prática, deve ser a chave que será utilizada no sentido que se dá ao termo em processamento de dados, isto é, como identificador de uma linha (tupla) nas operações de busca:

1) Na árvore que representa a hierarquia de relações, expandem-se as chaves primárias das relações que são "filhas" da relação-raiz, incorporando a chave primária desta.

2) Os domínios não simples da relação-raiz são eliminados desta.

3) Elimina-se a raiz da árvore.

4) Repete-se o processo para cada uma das sub-árvores restantes.

O procedimento exige que as chaves primárias sejam formadas apenas de atributos simples e que os atributos compostos se relacionem através de hierarquias puras (sem formar ciclos), exigências estas que podem ser satisfeitas por praticamente todos os casos reais.

Um modelo equivalente ao do exemplo, mas constituído de relações em 1ª FN somente, obtido através do procedimento, é: (foram adotadas as seguintes chaves primárias -

em PROFESSOR, NOME

em ATIVIDADES, (SEMESTRE, DISCIPLINA)

em ALUNOS, Nº . No novo modelo, as chaves primárias estão sublinhadas.

PROFESSOR' (NOME, DEPTO)

ATIVIDADES' (NOME, SEMESTRE, DISCIPLINA)

ALUNOS' (NOME, SEMESTRE, DISCIPLINA, Nº, NOTA) .

Então, se, de acordo com o modelo original, tivéssemos os seguintes dados:

PROFESSOR

Nome	depto	ATIVIDADES		ALUNOS	
		semestre	disciplina	nº	nota
Joaquim	MAP	1	MAP-111	53	8.0
				71	7.7
				62	0.0
				41	5.5
		1	MAP-211	53	0.0
				63	2.0
				75	4.5
Ney	MAT	1	MAT-111	50	2.5
				51	3.5
		2	MAT-121	33	10.0
				32	5.5
				31	9.5

no novo modelo, passaríamos a ter:

PROFESSOR' ATIVIDADES'

Nome	depto	nome	semestre	disciplina
Joaquim	MAP	Joaquim	1	MAP-111
Ney	MAT	Joaquim	1	MAP-211
		Ney	1	MAT-111
		Ney	2	MAT-121

ALUNOS'

nome	semestre	disciplina	nº	nota
Joaquim	1	MAP-111	53	8.0
Joaquim	1	MAP-111	71	7.7
Joaquim	1	MAP-111	62	0.0
Joaquim	1	MAP-111	41	5.5
Joaquim	1	MAP-211	53	0.0
Joaquim	1	MAP-211	63	2.0
Joaquim	1	MAP-211	75	4.5
Ney	1	MAT-111	50	2.5
Ney	1	MAT-111	51	3.5
Ney	2	MAT-121	33	10.0
Ney	2	MAT-121	32	5.5
Ney	2	MAT-121	31	9.5

No novo modelo, nenhuma informação foi perdida em relação ao original. Uma operação de junção natural entre PROFESSOR' e ATIVIDADES', sobre nome e uma junção natural entre o resultado e ALUNOS', sobre (nome, semestre, disciplina), reconstituem toda informação presente em PROFESSOR.

2ª fase normal.

Vejamos, agora, o seguinte exemplo, com uma relação na 1ª FN :

Professor	Disciplina	Sala	Créditos
André	MAP-111	25	6
André	MAP-121	24	8
Matana	MAP-111	25	6
Sabrina	MAP-111	28	6

Nesta relação, identifiquemos algumas dependências funcionais:

$$\{\text{PROFESSOR, DISCIPLINA}\} \longrightarrow \text{SALA}$$

$$\text{DISCIPLINA} \not\longrightarrow \text{SALA}$$

$$\text{PROFESSOR} \not\longrightarrow \text{SALA}$$

$$\text{DISCIPLINA} \longrightarrow \text{CRÉDITOS}$$

A única chave desta relação é {PROFESSOR, DISCIPLINA} que, portanto, é também a chave primária.

As dependências funcionais assinaladas indicam que cada disciplina tem um número de créditos que independe de outros atributos e cada professor ministra uma disciplina em uma determinada sala, mas que um professor pode ocupar várias salas (para diferentes disciplinas) e cada disciplina também pode ocupar várias salas (para diferentes professores).

Suponhamos agora que:

1) Deseja-se alterar o número de créditos de MAP-111 para 8. O que se observa é que 3 tuplas da relação devem ser alteradas, embora uma única informação esteja sendo modificada. Isto, em geral, não é desejável,

a) não só pela multiplicidade de trabalho (mesmo que todas menos uma alterações sejam efetuadas automaticamente pelo sistema).

b) mas também por expor o modelo a inconsistências, se uma ou mais tuplas não chegarem a ser alteradas devidamente (por falhas de programa, por alguma interrupção no sistema, etc.).

2) O professor André deixa de ministrar a disciplina MAP-121. A tupla que contém essa informação deve ser eliminada (é procedimento normal, em processamento de dados, que um registro não possa permanecer no arquivo com valor de chave indefinido, no nosso caso, chave primária), o que acarretará a perda da informação de que MAP-121 corresponde a 8 créditos.

3) Deseja-se introduzir a informação de que a disciplina MAP-321 corresponde a 7 créditos. Por motivo análogo ao exposto em 2), esse armazenamento não pode ser efetuado enquanto não houver algum professor que ministre essa disciplina.

4) Deseja-se introduzir novos professores para ministrar a disciplina MAP-111. Para cada novo professor, deve ser repetida a informação do número de créditos de MAP-111, trazendo inconvenientes análogos aos apontados em 1).

Anomalias como estas motivaram a definição da 2ª forma normal:

DEFINIÇÃO: Uma relação R está na 2ª forma normal quando está na 1ª FN e todo atributo (ou lista de atributos) não pertencente a nenhuma chave tem dependência funcional completa de toda chave.

De acordo com esta definição, nota-se que, no exemplo, a relação não está na 2ª FN, pois

DISCIPLINA ⟶ CREDITOS

isto é, CREDITOS não tem dependência funcional completa em relação à chave.

Um modelo equivalente, em que só haja relações na 2ª FN, pode ser obtido através de projeções adequadas:

R1 (PROFESSOR, DISCIPLINA, SALA) e R2 (DISCIPLINA, CREDITOS)

R1

professor	disciplina	sala
André	MAP-111	25
André	MAP-121	24
Matana	MAP-111	25
Sabrina	MAP-111	28

R2

disciplina	créditos
MAP-111	6
MAP-121	8

Novamente, nenhuma informação foi perdida (uma junção natural sobre disciplina recupera a relação original) e as anomalias são superadas:

1) Apenas uma tupla (de R2) é alterada para modificar o número de créditos de MAP-111 .

2) Eliminando-se uma tupla de R1 (André, MAP-121, 24), con-

serva-se a informação de que MAP-121 corresponde a 8 créditos (R2) .

3) Pode-se introduzir a tupla (MAP-321, 7) em R2, já que a chave primária contém apenas o atributo disciplina.

4) Podem ser introduzidos tantos novos professores quanto se queira, sem que seja necessário mexer com informações referentes ao número de créditos das disciplinas (a não ser para o primeiro professor de cada disciplina)

3ª forma normal.

Tomemos outro exemplo, com um modelo que só contenha a seguinte relação (na 2ª FN) :

Nº-prof	Nome-prof	Depto	Chefe
1	Joaquim	MAP	Hugo
2	Nelson	MAP	Hugo
3	Estanislau	MAT	Cristovão
4	André	MAP	Hugo
5	Ney	MAP	Hugo
7	Matana	MAE	Luiza

e seja Nº-prof a chave primária.

Algumas de suas dependências funcionais são:

Nº-PROF ⟷ NOME-PROF

Nº-PROF ⟶ DEPTO

DEPTO ⟶ CHEFE

Esta relação está, trivialmente, na 2ª FN pois

suas chaves (Nº-prof e Nome-prof) são constituídas por um só atributo e, portanto, todas as dependências funcionais das quais participam, são completas.

Tomemos as seguintes situações:

1) Deseja-se mudar o chefe do departamento MAP. Várias tuplas devem sofrer alterações, dando margem ao aparecimento de inconsistências no modelo.

2) O professor de nº 3 não mais pertence ao MAT. Vai ser perdida a informação sobre quem chefia o MAT.

3) Deseja-se introduzir a informação de que se criou um novo departamento (MAC) e será chefiado por Homero. Enquanto nenhum professor estiver designado para este departamento, a informação não pode ser armazenada.

4) Novos professores foram admitidos no MAE. Em cada nova tupla a ser criada, deve ser adicionado o nome do chefe do departamento.

DEFINIÇÃO: Uma relação R está na <u>3ª forma normal</u> quando está na 2ª FN e não se verifica uma situação do tipo:

$$K \longrightarrow A$$

$$A \longrightarrow B$$

$$A \not\longrightarrow K \quad \text{onde}$$

K é uma chave de R, B é um atributo de R (ou lista de atributos de R) que não pertence a nenhuma chave de R, A

é um atributo (ou lista de atributos) de R e B ⊈ A .

No exemplo, Nº-PROF ⟶ DEPTO

DEPTO ⟶ CHEFE

DEPTO ↛ Nº-PROF

e, portanto, a relação não está na 3ª FN .

Através de projeções convenientes podemos obter um novo modelo:

R1 (Nº-PROF, NOME-PROF, DEPTO) e R2 (DEPTO, CHEFE)

R1

Nº-prof	Nome-prof	Depto
1	Joaquim	MAP
2	Nelson	MAP
3	Estanislau	MAT
4	André	MAP
5	Ney	MAP
7	Matana	MAE

R2

Depto	Chefe
MAP	Hugo
MAT	Cristovão
MAE	Luiza

que, analogamente aos casos anteriores, conserva toda a informação contida na relação original.

Observe-se que a dependência funcional

NOME-PROF ⟶ Nº-PROF (ou Nº-PROF ⟶ NOME-PROF)

dá condições a que R1 esteja na 3ª FN, pois

Nº-PROF ⟶ NOME-PROF (ou NOME-PROF ⟶ Nº-PROF e

NOME-PROF ⟶ DEPTO Nº-PROF ⟶ DEPTO) .

Na verdade, este tipo de condição foi incluída na

definição para possibilitar que mais de uma chaves compareçam em uma relação na 3ª FN, situação que é bastante usual, na prática, e não conduz a anomalias do tipo 1), pois, em uma relação, cada valor da chave comparece uma única vez (em cada instante de tempo).

O novo modelo resolve problemas ligados às anomalias citadas, como pode ser facilmente verificado.

A restrição (na definição da 3ª FN) feita a B para que não pertença a nenhuma chave pode ser ilustrada pelo seguinte exemplo:

R (PROF, DISC, ALUNO) onde {DISC, ALUNO} $\longrightarrow$ PROF

PROF $\longrightarrow$ DISC

PROF $\not\longrightarrow$ {DISC, ALUNO}

R

prof	disc	aluno
Eurico	Física	A1
Eurico	Física	A2
João	História	A1
Calvito	História	A2

Se não houvesse a restrição a B, esta relação não estaria na 3ª FN e deveria ser projetada. Isto, no entanto, acarretaria perda de informação, mesmo que se tomassem as três projeções

R1 (PROF, ALUNO), R2 (PROF, DISC) e R3 (DISC, ALUNO)

R1		R2		R3	
prof	aluno	prof	disc	disc	aluno
Eurico	A1	Eurico	Física	Física	A1
Eurico	A2	João	História	História	A1
João	A1	Calvito	História	Física	A2
Calvito	A2			História	A2

De fato, como responder à pergunta: "Com qual professor o aluno A1 estuda História?".

De qualquer modo, porém, essa relação R constitui um caso um tanto "patológico", pois, apesar de estar na 3ª FN, não evita uma anomalia: se o aluno A1 deixar de assistir à disciplina História, esta tupla deveria ser eliminada e seria perdida a informação de que João é professor de História. Mesmo que se adotasse como chave primária Prof, aluno o problema não seria resolvido (e se o professor João deixasse de ensinar História e se quisesse conservar a informação de que A1 está matriculado em História?).

Este caso teria que receber uma solução não ortodoxa:

R1 (PROF, ALUNO, DISC) e R2 (PROF, DISC), por exemplo, acarretando o surgimento de redundâncias.

Existem pelo menos duas definições de 3ª FN que não necessitam da 2ª FN .

1) [CODD 74] : "Uma relação R está na 3ª FN se está na 1ª FN e, para toda coleção de atributos C, de R, se algum atributo não pertencente a C é dependente funcio

nal de C, então todos atributos de R são dependentes funcionais de C".

2) Sharman (cit. em [CHAM 76]) : "Uma relação esta na 3ª FN quando todo determinante é uma chave". (Determinante é uma coleção de atributos da qual alguma outra coleção de atributos, não contida no determinante, é dependente funcional). Admite-se que a definição se refere apenas a relações na 1ª FN .

É fácil notar a equivalência entre ambas (só diferem quanto à nomenclatura).

Confrontemos as duas definições, 1) e 2), com a definição apresentada anteriormente (e que será indicada por 3)).

a) Se uma relação R está na 3ª FN, segundo 1) e 2), também está na 3ª FN segundo 3)?

Uma relação na 1ª FN não está na 3ª FN, segundo 3), se e somente se pelo menos uma das duas situações seguintes se verificar:

1. Um atributo de A, não pertencente a nenhuma chave, não tem dependência funcional completa de uma chave K (nestas condições, a relação não está na 2ª FN e, consequentemente, não está na 3ª FN, segundo 3)).

2. $K \longrightarrow C$

   $C \longrightarrow A$

   $C \not\longrightarrow K$, onde K é chave, C é uma lista de atributos

da relação, A é um atributo não pertencente a nenhuma chave da relação e $A \notin C$ (é a situação que, segundo 3), impede uma relação na 2ª FN de estar na 3ª FN).

Se 1. ocorrer, então $\exists$ C, uma coleção de atributos tal que $C \longrightarrow A$, onde $C \subset K$ e, portanto, $C \not\longrightarrow K$ (senão K não seria chave); mas $A \in C$ (senão pertenceria à chave K), portanto A é um atributo não pertencente a C, dependente funcional de C e existe algum atributo da relação que não é dependente funcional de C (senão K não seria chave). Assim, a relação não está na 3ª FN segundo 1), o que contraria a suposição inicial. Logo, 1. não pode ocorrer.

Se 2. ocorrer, então

$$C \longrightarrow A \quad \text{e} \quad C \not\longrightarrow K;$$

como $A \notin C$, então A é: um atributo não pertencente a C, dependente funcional de C e existe algum atributo (de K) que não é dependente funcional de C. Assim, a relação não está na 3ª FN segundo 1), o que contraria a suposição inicial. Logo, 2. também não pode ocorrer.

A resposta à pergunta a) é, portanto, sim.

b) Se uma relação R está na 3ª FN, segundo 3), também está na 3ª FN, segundo 1) e 2)?

Vejamos, novamente, o exemplo:

R (PROF, DISC, ALUNO), onde $\{DISC, ALUNO\} \longrightarrow PROF$

$PROF \longrightarrow DISC$

$PROF \not\longrightarrow \{DISC, ALUNO\}$

Então, DISC $\notin$ {PROF} e {PROF} $\longrightarrow$ DISC; mas, {PROF} $\not\longrightarrow$ ALUNO .

Portanto, R está na 3ª FN segundo 3), mas não segundo 1) e 2).

Já foi dito, no entanto, que o exemplo acima é excepcional. Na grande maioria dos casos, as 3 definições se comportam de modo equivalente, podendo ser intercambiadas.

Podemos, então, sintetizar algumas das vantagens em se estruturar um modelo relacional através de relações na 3ª FN:

1) Para o administrador do organismo, encarregado da formulação dos modelos lógicos, a 3ª FN, principalmente no que diz respeito às dependências funcionais, fornece importantes subsídios de ordem semântica, além de sugerir uma estratégia para uma sistematização na construção do modelo. Se a abordagem escolhida for do tipo "top-down", quando, por exemplo, se parte de um diagrama entidade-relacionamento para chegar em relações na 3ª FN, o processo de normalização pode servir de aferição ao diagrama. Se a abordagem for do tipo "bottom-up", a constatação das dependências funcionais existentes entre atributos ainda não organizados constitui o primeiro passo da estruturação do modelo.

2) Para os usuários, trabalhar com relações na 3ª FN tem a vantagem de evitar as anomalias já citadas. Quando o usuá

rio opera diretamente com o modelo geral, ele praticamente tem que ser alertado apenas para a proibição em duplicar valores da chave primária numa mesma relação. A própria apresentação dos dados em forma relacional facilita a compreensão da base de dados, pela sua naturalidade. A 3ª FN evita para o usuário muito do treino a que teria de ser submetido, se o modelo pudesse conter relações em outras formas. Na 2ª FN, por exemplo, em uma situação vulnerável a anomalias do tipo 2) (ver 3ª forma normal), o usuário deveria ser alertado para o fato de que a operação de eliminação não pode ser efetuada, pois acarretaria perda de informação.

Apesar das formas normais, porém, a estruturação de um modelo relacional não é um problema de solução única. Alguns autores ([CODD 72b], [WANG 75], [DELO 73], etc.) examinaram a possibilidade de se definir uma coleção ótima de relações na 3ª FN e até mesmo de se encontrar algoritmos que levassem a essa otimização. Do ponto de vista prático, porém, os resultados ainda não são compensadores e, por enquanto, cabe ao bom senso do administrador do organismo a tarefa de eleger relações e atributos (com a normalização servindo como instrumento auxiliar, como já foi comentado).

EXEMPLO DE ESTRUTURAÇÃO DE UM MODELO

Baseado em um exemplo de [WANG 75], serão ilustrados, agora, dois caminhos que podem ser seguidos pelo ad-

ministrador do organismo, em busca de um modelo relacional de dados.

A porção do mundo real (ou organismo) que deve ser representada pelo modelo, pela base de dados, é um sistema para reserva de passagens aéreas. Os usuários são os funcionários encarregados de:

- fornecer, aos possíveis passageiros, informações sobre roteiros de viagem, datas, número de lugares disponíveis, etc.
- introduzir, no sistema, informações pertinentes ao passageiro e à reserva, uma vez efetuada.

I) Abordagem "bottom-up".

Neste tipo de abordagem, o administrador de organismo efetua um estudo deste organismo através do exame de pontos isolados, isto é, sem se preocupar em, inicialmente, apreender o todo do problema.

No caso do modelo relacional, este estudo consiste:

a) no reconhecimento de atributos de interesse para o problema.
b) na constatação de dependências funcionais entre estes atributos.

Para o sistema em questão, os atributos de interesse são:

- nome do passageiro (PNOM)
- endereço do passageiro (PEND)

- telefone do passageiro (PTEL)
- número de lugares pedidos (NLUG)
- número do voo (NVOO)
- classe de acomodações (CLSS)
- tipo de avião (APAR)
- capacidade do aparelho, em lugares de 1ª classe (CAP1)
- capacidade do aparelho em lugares, classe turística (CAP2)
- local de partida para um trecho de rota (LOCP)
- local de chegada de um trecho de rota (LOCC)
- horário de partida (HORP)
- horário de chegada (HORC)
- data (DATA)
- número de lugares de 1ª classe, reservados (RES1)
- número de lugares, classe turística, reservados (RES2)

Na enumeração das dependências funcionais, que refletem a semântica do relacionamento entre esses atributos, o administrador do organismo deve se ater a um mínimo. Este mínimo é, inicialmente, constituído pelas dependências funcionais que formam uma cobertura mínima necessária para o fechamento transitivo ([WANG 75]). Na referência citada, esses conceitos são definidos com maior detalhe, sendo inclusive apresentado um algoritmo para a obtenção dessa cobertura mínima. Em linhas gerais, uma cobertura mínima é um conjunto minimal de dependências funcionais completas, cujo lado direito é constituído de um só atributo e das quais todas as dependências funcionais completas possíveis (o fechamento

transitivo) são deriváveis através das propriedades da dependência funcional (ex: se A ⟶ B e B ⟶ C comparecem na cobertura mínima, então A ⟶ C é uma possível dependência funcional completa, mas que não pertence à cobertura).

No exemplo, então, uma cobertura mínima pode ser:

(a lista de atributos (LOCP, LOCC, HORP, HORC) está representada, abreviadamente, por TRCH)

PNOM ⟶ PEND

PNOM ⟶ PTEL

NVOO ⟶ APAR

APAR ⟶ CAP1

APAR ⟶ CAP2

TRCH ⟶ NVOO

{TRCH, DATA} ⟶ RES1

{TRCH, DATA} ⟶ RES2

{PNOM, TRCH, DATA} ⟶ NLUG

{PNOM, TRCH, DATA} ⟶ CLSS

ou seja,

- um vôo (identificado por NVOO) é constituído por vários trechos de rota (TRCH) e é sempre efetuado num mesmo tipo de avião (APAR).
- dois trechos podem ter os mesmos locais de chegada e partida, mas os horários serão diferentes (e cada trecho pertence a um determinado vôo).
- um passageiro faz reservas para trechos, e datas em que os

P-V

PASSAGEIRO	VIAGEM			
PNOM	TRCH	DATA	NLUG	CLSS

chave primária

O modelo relacional é, então, o seguinte:

AVIÃO (APAR, CAP1, CAP2)

VÔO (NVOO)

A-V (NVOO, APAR)

TRECHO (TRCH)

V-T (TRCH, NVOO)

PASSAGEIRO (PNOM, PEND, PTEL)

VIAGEM (TRCH, DATA, RES1, RES2)

P-V (PNOM, TRCH, DATA, NLUG, CLSS)

[CHEN 76] indica que a abordagem II produz relações similares às relações em 3ª FN obtidas pelo processo de normalização já descrito. Um tipo de discrepância nesta similaridade é que acontece, por exemplo, com o conjunto-de-relacionamentos A-V (fenômeno análogo se dá com V-T): A-V é do tipo 1 : N, logo há uma dependência funcional

$$NVOO \longrightarrow APAR \quad (e \quad APAR \not\longrightarrow NVOO)$$

e, portanto, de acordo com a definição de chave utilizada pa

II) Abordagem "top-down".

Este tipo de abordagem será exemplificado através da utilização do diagrama entidade-relacionamento como ponto de partida.

Tentando uma visão global do organismo, o administrador identifica, inicialmente, os conjuntos-de-registros-entidade relevantes:

Passageiro	Avião	Vôo	Trecho	Viagem

onde PASSAGEIRO representa uma entidade que não necessita maiores explicações,

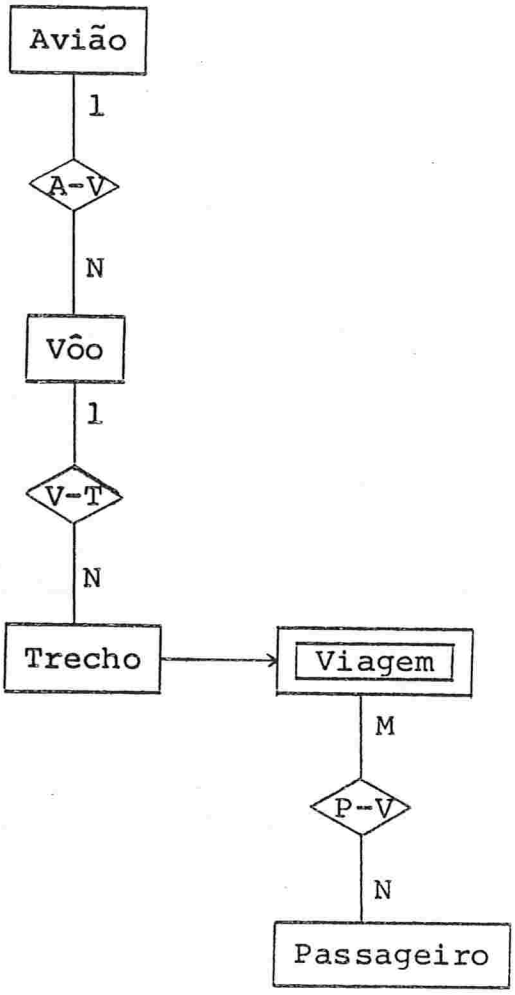
AVIÃO se refere ao aparelho de vôo,

VÔO é um conjunto de trechos,

TRECHO é indicado por dois locais e dois horários e

VIAGEM representa um evento, no qual um trecho é percorrido.

A seguir devem ser identificados os conjuntos-de-relacionamentos, representando os relacionamentos entre entidades, completando-se o diagrama com os tipos desses conjuntos-de-relacionamentos:



Do ponto de vista semântico, o diagrama denota, entre outras coisas que:

- cada vôo é realizado em um só tipo de aparelho.
- cada trecho pertence a um único vôo.
- o conjunto-de-registros-entidade VIAGEM é dependente de TRECHO porque uma viagem só tem existência quando associada a algum trecho.
- os passageiros estão relacionados diretamente com as viagens, para as quais são efetuadas as reservas.

Colocando-se o modelo sugerido por este diagrama sob a forma tabular anteriormente apresentada (ver figuras 5, 6, 7), praticamente está estruturado o modelo relacional:

(para maior simplicidade e por não apresentar maiores problemas, serão omitidas as indicações sobre os conjuntos-de-valores e os papéis, supondo que os nomes dos atributos sejam suficientemente explicativos)

AVIAO

APAR	CAP1	CAP2

chave primária

VOO

NVOO

chave primária

TRECHO

TRCH

chave primária

PASSAGEIRO

PNOM	PEND	PTEL

chave primária

VIAGEM

TRECHO			
TRCH	DATA	RES1	RES2

chave auxiliar

chave primária

A-V

AVIAO	VOO
APAR	NVOO

chave primária

V-T

VOO	TRECHO
NVOO	TRCH

chave primária

percorrerá (um trecho pode ser percorrido diariamente, por exemplo).

- um trecho é percorrido no máximo uma vez por dia.
- um passageiro pode reservar passagens para uma única classe, em cada vez que percorre um trecho.
- etc.

A partir dessa cobertura mínima, a obtenção do modelo relacional é imediata:

PASSAGEIRO (PNOM, PEND, PTEL)

AVIÃO (APAR, CAP1, CAP2)

VÔO (NVOO, APAR)

TRECHO (TRCH, NVOO)

VIAGEM (TRCH, DATA, RES1, RES2)

RESERVA (PNOM, TRCH, DATA, NLUG, CLSS)

O procedimento usado é o seguinte:

1) Os atributos presentes em uma dependência funcional da cobertura mínima formam uma relação. O lado esquerdo da dependência funcional é a chave primária da relação (Note se que, assim formadas, as relações estão, trivialmente, na 3ª FN).

2) Se R1 (A, B) e R2 (A, C), substitui-se R1 e R2 por

R (A, B, C)

(corresponde à propriedade aditiva das dependências funcionais e, em [WANG 75], demonstra-se que R também está na 3ª FN).

ra o processo de normalização, APAR não deve fazer parte de uma chave em que compareça NVOO .

De qualquer maneira, os modelos obtidos através das abordagens I e II são bastante semelhantes, mas nada fornece indicação segura de que isto sempre aconteça. Em ambas as abordagens intervêm fatores subjetivos que podem fazer com que dois administradores distintos cheguem a modelos distintos (ainda que usem o mesmo tipo de abordagem), embora, com as sistematizações propostas, as diferenças dificilmente cheguem a ser significativas.

De qualquer maneira, as duas abordagens constituem instrumentos valiosos para o administrador visualizar o organismo, na busca de um modelo de dados.

Uma outra alternativa, talvez a mais indicada enquanto não se chega a um procedimento melhor definido, seria uma combinação das duas abordagens, uma atuando como verificador para a outra.

R E F E R Ê N C I A S

[ANSI 75]: ANSI/X3/SPARC (Study Group on DBMS) - "Interim Report" - ACM SIGMOD Newsletter, 1975.
Proposta de padronização para SABDs.

[BACH 73]: Bachman, C. W.- "The programmer as navigator" - CACM 16, 11 Nov 1973.
Artigo, já clássico, sobre a mudança do centro no mundo do processamento de dados: do processo para os dados.

[BENJ 72]: Benjamin, R. I. - "A generational perspective of information system development" - CACM 15, 7 July 1972.
Evolução histórica dos sistemas de informação sob uma perspectiva "geracional", a exemplo do que é feito comumente com "gerações" de computadores.

[BOYC 75]: Boyce, R. F. & Chamberlin, D. D. & King III, W. F. & Hammer, M. M. - "Specifying Queries as Relational Expressions: the SQUARE sublanguage" - CACM 18, 11 Nov 1975.
Linguagem de manipulação de dados para modelos relacionais. Adaptação do cálculo relacional para uma notação mais "natural" (sem quantificadores, por exemplo). Prova de que é relacionalmente completa.

[CHAM 74]: Chamberlin, D. D. & Boyce, R. F. - "SEQUEL: a Structured English Query Language" - IBM Research Lab., San Jose, Cal. 1974.
Linguagem de manipulação de dados para modelos relacionais. Resultado de mudanças na linguagem SQUARE para dar-lhe uma sintaxe mais próxima à da língua inglesa. É a linguagem utilizada no Sys R, da IBM.

[CHAM 76]: Chamberlin, D. D. - "Relational Data Base Management Systems" - ACM Computing Surveys 8, 1 March 1976.
Apanhado suscinto dos principais conceitos do modelo relacional. Bibliografia extensa.

[CHEN 76]: Chen, Peter Pin-Shan - "The Entity-Relationship Model - Toward a unified view of data" - ACM TODS 1, 1 March 1976.
O Modelo Entidade-Relacionamento. Diagrama entidade-relacionamento. Transformações do MER para outros modelos.

[CHUR 56]: Church, A. - "Introduction to Mathematical Logic" - Princeton Univ. Press, Princeton, N.J. 1956.

[CODA 71]: Codasyl Programming Language Committee - "DBTG Report" - ACM 1971.
O relatório básico sobre o modelo da Codasyl.

* [CODD 70]: Codd, E.F. - "A relational model of data for large shared data banks" - CACM 13, 6 June 1970.
O primeiro artigo sobre o modelo relacional. Vantagens sobre outros modelos.

[CODD 72a]: Codd, E.F. - "Relational completeness of Data Base sublanguages" - Courant Computer Science Symp. May 1971 - ed. Rustin, R. - Prentice Hall 1972.
Artigo básico sobre a álgebra e o cálculo de relações. Completude relacional. Comparações.

[CODD 72b]: Codd, E.F. - "Further normalization of the Data Base relational model" - Courant Computer Science Symp. May 1971 - ed. Rustin, R. - Prentice Hall 1972.
Definição rigorosa das 3 formas normais.

[CODD 74]: Codd, E.F. - "Recent investigations in relational Data Base Systems" - Proc. IFIP Cong. Stockholm, Aug 1974 - ed. Rosenfeld, J.L. - North Holland.
Revisão do modelo relacional: novos aspectos em alguns tópicos (normalização, linguagens, modelos externos) e propostas para investigação.

[DELO 73]: Delobel, C. & Casey, R.G. - "Decomposition of a Data Base and the theory of Boolean switching functions" - IBM Journal of R&D Sep 1973.
Estruturação de modelo relacional. Equivalência entre operações com dependências funcionais e operações com funções booleanas, como auxílio no tratamento computacional do problema.

[FROM 62]: Fromm, E. - "A linguagem esquecida" - Zahar Editores, 1962.

[HELD 75]: Held, G.D. & Stonebraker, M. & Wong, E. - "INGRES - a relational Data Base System" Proc. AFIPS NCC, Anaheim, May 1975 - AFIPS Press.
Apresentação condensada dos principais componentes do INGRES, com ênfase na linguagem QUEL.

[LANG 75]: Langefors, B. & Sundgren, B.-"Information Systems Architecture" - Petrocelli/Charter, N. Y. 1975.

[MART 75]: Martin, J. - "Computer Data Base Organization" - Prentice Hall 1975.
Livro introdutório aos SABD. Estruturas lógicas de dados. Organização física. Detalhes de implementação. IMS. Modelo da Codasyl. Modelo relacional.

[SENK 73]: Senko, M.E. & Altman, E.B. & Astrahan, M.M. & Fehder, P.L. - "Data structure and accessing in Data Base Systems" - IBM Systems Journal 12, 1 1973.
Evolução dos sistemas de informação. Justificativa e descrição dos modelos do DIAM (Data Independent Accessing Model): Modelo Conjunto-de-Entidades, Modelo Cadeia, Modelo Codificação e Modelo Nível-de Dispositivo-Físico.

[WANG 75]: Wang, C.P. & Wedekind, H. - "Segment synthesis in logical Data Base design" - IBM Journal of R&D 19, 1 Jan 1975.
Procedimentos para estruturação de um modelo relacional de dados, através da 3ª forma normal, partindo de um conjunto minimal de dependências funcionais.

B I B L I O G R A F I A A D I C I O N A L

Segue-se uma lista de obras que, sem serem referenciadas diretamente no texto, de alguma forma contribuíram para este trabalho.

ASTRAHAN, M.M. & CHAMBERLIN, D.D. - "Implementation of a structured English Query Language" - CACM 18, 10 Oct 1975.
Considerações sobre a implementação da linguagem SEQUEL (baseada no cálculo de relações). Problemas de otimização de desempenho.

ASTRAHAN, M.M. & al. - "SYSTEM R: a relational approach to Data Base Management" - ACM TODS 1, 2 June 1976.
Documento básico sobre o Sys R da IBM. Descrição da arquitetura e do projeto do sistema. Detalhes de utilização e da implementação. O Sys R é experimental, não tendo ainda sido comercializado.

BACHMAN, C.W. - "The evolution of storage structures" - CACM 15, 7 July 1972.
Artigo bastante original, onde uma técnica gráfica representativa de estruturas de dados (proposta pelo autor em outro trabalho) é usada para ilustrar a evolução histórica das estruturas de armazenamento, incluindo os novos conceitos ligados aos SABD.

BILLER, H. & NEUHOLD, E.J. - "Formal view on Schema-Subschema correspondence" - Proc. IFIP Cong. Stockholm, Aug 1974 - ed. Rosenfeld, J.L. - North Holland.
Modelo formal de base de dados e da noção de base de dados, como referência para análise do "mapping" entre dois modelos de bases de dados. Crítica ao relacionamento entre Schema e Subschema no modelo da Codasyl.

BILLER, H. & NEUHOLD, E.J. - "The semantic interrelationship of data models" - Universitat Stuttgart. Institut für Informatik - 1976.
Consistência de bases de dados. Tentativas de formalização. Linguagens.

BIRKHOFF, G. & BARTEE, T.C. - "Modern applied algebra" - Mc Graw Hill, 1970.

BLASER, A. & SCHMUTZ, H. - "Data Base Research: a Survey" - Lecture Notes in Computer Science n. 39 - Springer-Verlag (Berlim), Heidelberg (N.Y.) - 1975.
Resenha histórica sobre os SABD, resumindo os principais conceitos e aspectos. Bibliografia com 198 ítens.

CARDENAS, A.F. - "Analysis and performance of inverted Data Base structures" - CACM 18, 5 May 1975.
Implementação de base de dados através de organização invertida, considerada principalmente sob o ponto de vista de eficiência. Um dos artigos mais indicados sobre o assunto.

CHAMBERLIN, D.D. & GRAY, J. N. & TRAIGER, I. L. - "Views, authorization and locking in a relational Data Base System" - Proc. AFIPS NCC, Anaheim - May 1975 - AFIPS Press.
Breve introdução aos problemas mencionados no título, através da linguagem SEQUEL.

CODASYL SYSTEMS COMMITTEE - "A survey generalized Data Base Management Systems" - Technical Report, May 1969 - cópias disponíveis na ACM.
Descrição minuciosa de alguns SABD em uso comercial na época. Trabalho pioneiro de apoio ao relatório [CODA 71].

CODASYL SYSTEMS COMMITTEE - "Feature Analysis of Generalized Data Base Management Systems" - Technical Report, May 1971 - cópias: ACM, BCS, IFIP.
Reformulação do relatório do ítem anterior. Outros sistemas foram incorporados (IMS, Codasyl). Apresentação por tópicos e não por sistema.

CODASYL SYSTEMS COMMITTEE - "Introduction to "Feature Analysis of Generalized Data Base Management Systems" - CACM 14, 5 May 1971.

COHEN, L.J. - "Data Base Management Systems" - Q.E.D. Information Sciences Inc. 1973.
Apreciação de 5 SABD em uso comercial (IMS, SYS 2000, IDMS, TOTAL, ADABAS) sob os mais variados aspectos. Boa descrição desses sistemas. Avaliação e comparação.

DATE, C. J. - "An introduction to Data Base Systems" - Addison-Wesley 1975.
Texto didático introdutório aos SABD. Apresentação detalhada dos modelos relacional, hierárquico (IMS) e de rede (Codasyl). Bibliografia e exercícios.

DEHENEFFE, C. & HENNEBERT, H. & PAULUS, W. - "Relational Model for a Data Base". - Proc. IFIP Cong. Stockholm, Aug 1974 - ed. Rosenfeld, J. L. - North Holland.
Divisão das relações de um modelo relacional em dois tipos, semelhantes à divisão entre entidades e relacionamentos. Linguagem de consulta.

EARLEY, J. - "On the semantics of data structures" - Courant Computer Science Symp. May 1971 - ed. Rustin, R. - Prentice Hall 1972.
Um dos primeiros artigos a examinar as estruturas de dados através da separação em níveis lógico, de caminhos de acesso e físico. Exemplos.

ENGLES, R. W. - "An analysis of the April 1971 DBTG Report" - BCS Conf. Oct 1971 - cópias: ACM, BCS.
Críticas da IBM ao modelo da Codasyl e propostas.

FELDMAN, J. A. & ROVNER, P. D. - "An ALGOL-based associative language" - CACM 12, 8 Aug 1969.
Artigo pioneiro sobre linguagens de especificação de dados através de suas propriedades. Linguagem LEAP.

KING, P. F. H. - "Some comments on the DBTG Report" - BCS Conf. Oct 1971 - cópias: ACM, BCS.
Exemplos de utilização, chamando atenção para alguns pontos controversos.

KNUTH, D. E. - "The art of computer programming", vol. 3 ("Sorting and Searching") - Addison-Wesley, 1973.
Estruturação de e busca em: árvores binárias, árvores balanceadas, "multiway trees", etc. "Hashing".

LANGEFORS, B. - "Information Systems" - Proc. IFIP Cong. Stockholm, Aug 1974 - ed. Rosenfeld, J.L. - North Holland.
Visão filosófica dos sistemas de informação. Problemas em projetos. Justificativa das visões "infológica" (ponto de vista apenas informacional, semântico) e "datalógica" (estruturação e implementação dos dados) em um SABD.

LINDGREEN, P. - "Basic operations on information as a basis for Data Base design" - Proc. IFIP Cong. Stockholm, Aug 1974 - ed. Rosenfeld, J.L. - North Holland.
Proposta de estruturação de base de dados através de "conjuntos de informação" e operações definidas sobre esses

LUCKING, J.R.- "A brief summary of the Codasyl DBTG Report" - BCS Conf. Oct 1971 - cópias: ACM, BCS.

LYON , J.K. - "An introduction to Data Base Design" - Wiley & Sons 1971.
Livro introdutório a alguns conceitos de base de dados. Dirigido mais para aspectos administrativos de um organismo.

McDONALD, N. & STONEBRAKER, M. - "CUPID - the friendly query language" - Electronics Research Lab., College of Engineering, Univ. of California, Berkeley - 1974.
"Casual User Pictorial Interface Design", linguagem gráfica interativa, manipulada através de um terminal de vídeo e destinada a usuários leigos em programação (INGRES).

MAGALHÃES, G.C. & ZIVIANI, N. & PEQUENO, T.H.C.- "Um modelo para implantação de um Banco de Dados Relacional" - CAPRE bol. informa. abr/jun 1975.
Estruturas de armazenamento voltadas para manipulação de dados através de álgebra relacional.

McLEOD, D. & MELDMAN, M.- "RISS - a generalized minicomputer relational Data Base Management System" - Proc. AFIPS NCC, Anaheim, May 1975 - AFIPS Press.
Esquema de como um SABD relacional pode ser implementado em minicomputadores. Linguagens (programadores e não-programadores).

MICHAELS, A.S. & MITTMAN, B. & CARLSON, C.R.- "A comparison of the Relational and Codasyl approaches to Data Base Management" - ACM, Computing Surveys 8, 1 March 1976.
Definição, manipulação, proteção e independência de dados e desempenho do sistema como tópicos de comparação.

MYLOPOULOS, J. & SCHUSTER, S. & TSICHRITZIS, D.-"A multi-level relational system" - Proc. AFIPS NCC, Anaheim, May 1975 - AFIPS Press.
Descrição do desenvolvimento do projeto ZETA/TORUS (Toronto). Divisão do projeto em níveis que podem ser desenvolvidos quase independentemente. Um SABD relacional.

OLLE, T.W. - "Recent Codasyl Reports on Data Base Management" - Courant Computer Science Symp. May 1971 - ed. Rustin, R. - Prentice Hall 1972.
Explanação sintetizada dos principais conceitos envolvidos no modelo da Codasyl.

OLLE, T.W. - "Current and future trends in Data Base Management Systems" - Proc. IFIP Cong. Stockholm, Aug 1974 - ed. Rosenfeld, J.L. - North Holland.
Evolução dos conceitos ligados aos SABD. Estruturação ló-

gica. Independência de dados. Níveis de usuários (administrador, programador, não-programador, paramétrico) e suas interfaces. Integridade e privacidade. O modelo Codasyl. Propostas de desenvolvimento futuro.

OZKARAHAN, E. A. & SCHUSTER, S. A. & SMITH, K. C.-"RAP-an associative processor for Data Base Management" - Proc. AFIPS NCC, Anaheim, May 1975 - AFIPS Press.
Arquitetura para um processador associativo, voltado para Sistemas Administradores de Base de Dados.

SENKO, M.E. - "Details of a scientific approach to information systems" - Courant Computer Science Symp. May 1971 - ed. Rustin, R. - Prentice Hall 1972.
Um dos artigos preliminares à apresentação do DIAM, examina os problemas envolvidos com a concepção lógica de sistemas de informação.

SIBLEY, E. H. & TAYLOR, R. W. - "A data definition and mapping language" - CACM 16, 12 Dec 1973.
Linguagem de definição de dados que inclua também características físicas, para que usuários com condições, possam melhorar a eficiência de seus programas, fazendo-os dependentes dos dados.

SIBLEY, E. H. - "The development of Data Base technology" - ACM Computing Surveys 8, 1 March 1976.
Artigo conciso dando um apanhado geral sobre as condições em que surge a necessidade dos SABD.

STEEL, T. B. - "Data Base standartization - a Status Report" - Lecture Notes in Computer Science n. 39 - Springer-Verlag (Berlin), Heidelberg(N.Y.) - 1975.
Introdução ao relatório [ANSI 75]. Alguns princípios filosóficos.

STONEBRAKER, M. & HELD, G. - "Networks, Hierarchies and Relations in DBMS" - Electronics Research Lab., College of Engineering, Univ. of California, Berkeley - 1975.
Três concepções de modelos de dados. Linguagens de manipulação de dados segundo o grau de "procedimentaliddde" ("procedurality). Tenta mostrar porque os dois aspectos não devem ser confudidos.

STONEBRAKER, M. & WONG, E. & KREPS, P. & HELD, G.-"The design and implementation of INGRES" ACM TODS 1, 3 Sep 1976.
INGRES (Interactive Graphics and Retrieval System). SABD relacional, parcialmente implementado em computadores PDP (sistema operacional UNIX). Estrutura de processamento do sistema. Linguagem QUEL. Métodos de acesso. Catálogos. CUPID (linguagem gráfica para não-programadores). Decomposição de consultas. EQUEL (QUEL como sublinguagem de mani

SUNDGREN, B. - "Conceptual foundation of the infological approach to data bases" - Proc. IFIP WC, Cargese, Apr 1974 ed. Klimbie & Koffman - North Holland.
Uma concepção de modelo lógico de dados, ainda praticamente restrita a pesquisadores suecos. Ênfase em problemas semânticos.

TSICHRITZIS, D. C. & LOCHOVSKY, F. H. - "Hierarchical Data Base Management: a survey" - ACM Computing Surveys 8, 1 March 1976.
O modelo Hierárquico. Métodos de acesso. Estratégias de implementação. Exemplos. Bibliografia. Comparação com outros modelos.

WEDEKIND, H. - "On the selection of access paths in a Data Base System" - Proc. IFIP WC, Cargese, Apr 1974 - ed. Klimbie & Koffman - North Holland.
"B-trees" como caminho de acesso básico na formulação de critérios de seleção de caminhos de acesso. "B*-trees". Comparação com método sequencial e "hashing".

WEDEKIND, H. - "Structured Database Programming" - Carl Hanser Verlag, Munchen, Wien, 1977.
Extensão de conceitos de programação estruturada para as estruturas de dados, orientada para bases de dados. Considerações sobre performance e consistência. Linguagem SEQUEL usada para exemplificar diversos casos.

WONG, E. & YOUSSEFI, K. - "Decomposition —a strategy for query processing" - ACM TODS 1, 3 Sep 1976.
Estratégia para processamento de consultas, usada no INGRES. A consulta é decomposta em consultas mais simples (uma variável). Algoritmos de redução e de escolha de variáveis a serem substituídas (minimização de custo).

YOURDON, E. - "Design of on-line computer systems" - Prentice Hall 1972.
Sistemas "on-line" em geral. Problemas na estruturação e operação de arquivos (e bases de dados) em sistemas desse tipo. Problemas de acesso simultâneo. Segurança.

ZLOOF, M. M. - "Query by example" - Proc. AFIPS NCC, Anaheim, May 1975 - AFIPS Press.
Linguagem de manipulação de dados em modelo relacional, voltada para usuários não-programadores. O usuário trabalha com tabelas visualizadas em terminal de vídeo.

ZOOK, W. & YOUSSEFI, K. & KREPS, P. & HELD, G. & FORD, J. - "INGRES - reference manual" - Electronics Research Lab., College of Engineering, Univ. of California, Berkeley - 1975.