VISÕES ESTENDIDAS

UMA PROPOSTA PARA EXTENSÃO DE BANCOS DE DADOS RELACIONAIS

Este exemplar corresponde à redação final da tese devidamente corrigida e defendida por Henrique Nou Schneider e aprovada pela Comissão Julgadora.

Campinas, 21 de agosto de 1989.

Profa. Dra : Claudia Maria Bauzer Medeiros

Dissertação apresentada ao Instituto de Matemática, Estatística e Ciência da Computação, UNICAMP, como requisito parcial para obtenção do Título de Mestre em Ciência da Computação.

Sch57v
11382/BC

UNICAMP
BIBLIOTECA CENTRAL

A Erwin Henrique

AGRADECIMENTOS

- A meus pais, por minha alfabetização e formação moral, sem os quais eu seria ninguém.

- A Teca, espôsa e companheira, pela compreensão e estímulo nesta jornada.

- A Profa. Claudia Bauzer Medeiros, mestra querida, por ter me ensinado Banco de Dados e me orientado pacientemente, neste e em outros trabalhos.

- Aos Profs. Antônio Monteiro e Manuel Bernardino Lino Salvador pelo estímulo e apoio.

- A IBM Brasil, na pessoa do Sr. J. Paulo Schiffini, pelo recebimento de uma Bolsa de Estudos que tanto ajudou na realização deste trabalho.

- A CAPES e CNPQ pela contribuição para a realização deste curso e confecção deste trabalho.

RESUMO

A tese apresenta uma proposta para estender facilidades de SGBDs relacionais através de mecanismos de atualização de visões. A extensão se baseia na definição de interfaces que permitam o suporte a um novo tipo de visão : as *"visões estendidas" (VEs)*, utilizadas como mecanismo para definir e manipular tipos de dados não suportados pelo SGBD original.

As visões são definidas segundo a filosofia de *Tipos Abstratos de Dados*, onde a função geradora de visão, as operações permitidas sobre a visão e um conjunto de mapeamentos para cada operação estão encapsulados em um único *Módulo*. Como cada atualização pode ter mais que uma tradução correta, o usuário pode optar por especificar o mapeamento quando da definição da visão, ou no momento em que ele estiver processando atualizações.

A validação da proposta é feita através da implementação de um protótipo que permite a definição e manipulação de alguns tipos de VEs.

VISÕES ESTENDIDAS

UMA PROPOSTA PARA EXTENSÃO DE BANCOS DE DADOS RELACIONAIS

INDICE

1 - Tipos de Dados e Sistemas de Apoio à Decisão 01

1.1 - Introdução 01

1.2 - Tipos de Dados e Manipulação de Objetos em Banco de Dados 01

1.3 - Bancos de Dados Estendíveis e o Modelo Relacional 07

1.4 - Sistemas de Apoio à Decisão 11

1.5 - Resumo 15

1.6 - Organização da Tese 15

2 - Visões, TADs e Agregados 17

2.1 - Introdução 17

2.2 - Atualizações em Visões 17

2.3 - Definições e Notação-Agregados 22

2.4 - Visões como Tipos Abstratos de Dados 26

2.5 - Resumo 35

3 - Sistema Proposto 37

3.1 - Introdução 37

3.2 - Visões Estendidas e Atributos Estruturados 39

3.3 - Arquitetura Geral da Interface para Suporte a VEs 42

3.4 - Módulo de Interface Universal 44

3.5 - Módulo de Geração 47

3.5.1 - Mapeamento de Consulta 48

3.5.2 - Mapeamento de Atualização 50

3.6 - Módulo de Atualização 60

3.7 - Módulo de Consistência 68

3.8 - Restrições Impostas 73

3.9 - Resumo 75

4 - Sistema Implementado 76

4.1 - Características da Implementação 77

4.2 - Estruturas de Dados 78

4.2.1 - Arquivos Utilizados 79

4.3 - Restrições Adotadas na Implementação 85

4.4 - Exemplo 87

5 - Conclusões e Extensões 97

5.1 - Conclusões 97

5.2 - Extensões 100

Bibliografia 103

Anexo 1 - Mapeamentos de Atualizações sobre alguns Agregados 110

Anexo 2 - Representação Gráfica dos Arquivos Utilizados 117

Anexo 3 - Manual do Usuário - Protótipo Implementado 119

CAPÍTULO 1

TIPOS DE DADOS E SISTEMAS DE APOIO À DECISÃO

1.1 - INTRODUÇÃO

Esta tese propõe um sistema voltado a usuários pertencentes à categoria de "*usuários da informação*" (nível externo na classificação dos sistemas de gerência de base de modelos), isto é, usuários tomadores de decisões que se interessam pelas saídas e suas validades, sem se preocuparem com a representação e o funcionamento dos modelos. Dentre os pontos implementados salientam-se o uso de novos tipos de dados e sua utilização em Sistemas de Apoio à Decisão. Este capítulo fornece os conceitos necessários e uma breve revisão bibliográfica na área.

1.2 - TIPOS DE DADOS E MANIPULAÇÃO DE OBJETOS EM BANCO DE DADOS

As noções de tipos de dados e abstração em banco de dados têm uma ligação muito íntima com as mesmas noções em linguagens de programação. Em ambos os casos, os problemas se referem a como definir novos tipos de dados e suas operações e incorporá-los a linguagens (sistemas) já existentes.

[BIS86], falando no contexto de linguagens, aponta que historicamente um tipo de dados era um conjunto de valores que se referia a uma das (poucas) classes disponíveis (por exemplo, inteiro). Posteriormente, surgiu a noção de abstração e um tipo passou a ser um "mecanismo de uma linguagem para garantir autenticação e segurança" ([BIS86] pp. 11). São identificados três pontos de vista no tratamento de tipos de dados : pragmático, operacional e matemático. O pragmático afirma que um tipo é uma maneira de organizar informação sobre programas, expressando as categorias de operações que podem ser aplicadas sobre valores. No caso operacional, busca-se classificar conjuntos de valores de acordo com as operações que comportam (em outras palavras, de acordo com as propriedades destes conjuntos). Finalmente, no enfoque matemático, um tipo é definido de acordo com um conjunto de operações e uma especificação utilizando (alguma) teoria formal.

Enfoques semelhantes podem ser encontrados na área de banco de dados, com ênfase no aspecto operacional. Assim é que a literatura correlata atualmente trata de tipos "padrão" em oposição a "tipos não convencionais".

Um tipo padrão segue a noção histórica de tipos em linguagens de programação, correspondendo a um tipo básico, disponível em qualquer SGBD convencional (por exemplo, inteiro, real ou string). Tipos não convencionais são tipos construídos a partir de tipos básicos e que permitem ao usuário maior flexibilidade na definição de suas aplicações. Tipos não convencionais podem ser

adicionados a um sistema de banco de dados através de rotinas especiais (por exemplo, utilizando noções de tipos abstratos de dados) ou já fazer parte do rol de tipos de um SGBD, que neste caso pode também ser "não convencional", incorporando tipos que permitam suporte eficiente a novas aplicações.

Um exemplo de tipo de dado usualmente necessário nas aplicações ditas não convencionais são os "*campos longos*". Campos longos são itens de dados não formatados e muito extensos que devem ser tratados pelo SGBD como objetos atômicos. O suporte a campos longos pode ser feito pela introdução de uma facilidade de armazenamento de textos embutida no sistema [MEL88]. Uma das formas encontradas para implementar esse recurso é fazer com que os SGBDs representem os campos longos como cadeias de caracteres, cuja interpretação será feita por programas que fazem interface com o SGBD. Porém, além de ser descrito como sendo uma solução ineficiente, provoca redundância do código nos vários programas que o utilizam e cria dependência de dados (se o objeto mudar, modificar-se-ão todos os programas que o interpretam).

A necessidade de suportar novos tipos gerou a noção de "sistema estendível (extensible) de banco de dados", referente a um SGBD que permita a um usuário especializado adicionar novos tipos e operações a este SGBD [OSB87].

O conceito de tipo não convencional em bancos de dados estendíveis aproxima-se, a grosso modo, do conceito de tipos abstratos de dados em linguagens de programação. Analogamente, são utilizados os conceitos de encapsulamento de operações, de

forma a permitir um conjunto de operações sobre os valores do tipo "não convencional".

Bancos de dados orientados a objetos são também estendíveis, mas se distinguem da categoria anterior por incorporarem a definição de um modelo semântico.

O conjunto de tipos embutidos permite operações sobre objetos complexos, suas hierarquias e agregações. O propósito de "orientação para objetos" é procurar uma forma de melhor capturar a semântica das novas aplicações de banco de dados. Um objeto complexo é um objeto cujos atributos podem ser atômicos ou objetos complexos; possuem, assim, uma raiz e uma hierarquia de sub-objetos.

Bancilhon [BAN88] aponta como principais características da área de banco de dados orientados a objetos a falta de um modelo básico e de formalização aliados a uma atividade experimental interna. No seu trabalho, aponta como um dos obstáculos a vencer nesta área o fato de que não há consenso quanto à definição de conceitos. Segundo ele, as principais características de um sistema orientado a objetos são : encapsulamento (em que cada objeto tem uma interface, que é sua parte visível e sua implementação, que esconde dados e operações); identificação; tipos (conjunto de objetos com as mesmas características) e classes (usadas durante execução do sistema); herança de propriedades; "overloading" e "late binding".

Esta tese diferencia sistemas estendíveis baseados em SGBDs tradicionais de sistemas estendíveis orientados a objetos.

Um exemplo híbrido é o sistema POSTGRES [STO87], que embora baseado no modelo relacional, incorpora operações que facilitam a geração e manipulação de objetos complexos incluindo agregação e generalização. É dado suporte a hierarquia de objetos compartilhados, permitindo que objetos compartilhem objetos e que atributos de uma relação referenciem tuplas de outras relações.

Os mecanismos utilizados na extensão proposta se baseiam em tipos abstratos de dados, incluindo operadores e procedimentos definidos pelo usuário, atributos do esquema de relação do tipo "procedimento" e hereditariedade destes procedimentos e atributos.

É facultado ao usuário manipular tipos atômicos e estruturados de dados. Os dados de tipo atômico são definidos como TADs, especificando-se o nome do tipo, o comprimento de sua representação interna, funções para converter a representação interna em externa e vice-versa e um valor "default". Pode-se, ainda, definir operadores de comparação. O tipo estruturado pode ser do tipo "vetor" ou "procedimento". O tipo procedimento é uma expressão da linguagem de consulta. Assim um atributo do tipo procedimento pode ter, no seu domínio, valores retirados de outras relações. O autor chama de "multirelação" a relação que tem em seu esquema atributos com essas características. Por exemplo, um tipo quadrado, pode ser definido e incorporado a uma aplicação no sistema POSTGRES. Conjuntos de valores que formam um quadrado podem sofrer rotação, translação etc. Uma ocorrência de um tipo "expressão SQL" é uma relação que corresponde à

materialização da consulta representada pela expressão. Este mecanismo pode ser usado, inclusive, para definição de relações aninhadas.

POSTGRES mantém os dados eliminados e modificados (dados históricos), permitindo consultas a estados anteriores do banco de dados. O sistema só não salvará os estados do banco de dados quando for explicitamente especificado.

POSTGRES permite, também, definição de versões. Uma versão de uma relação pode ser criada sobre uma relação-base ou sobre um instantâneo. O segundo caso deverá ser escolhido quando o usuário quiser que as atualizações sobre a relação-base não se propaguem para a versão. No primeiro caso, as atualizações sobre a relação-base se propagam para a versão. Em ambos os casos, as atualizações sobre a versão não afetam a relação-base a não ser quando for explicitamente especificado.

Outra solução do tipo híbrido é o aprofundamento de estudos de banco de dados não normalizados de forma a modelar algumas das novas aplicações. Segundo este enfoque, uma tupla de uma relação NF2 corresponde à ocorrência de um objeto. Por exemplo, [KEM87] mostra como integrar o conceito de TADs ao sistema AIM [PIS86] para obter objetos utilizados em aplicações de robótica.

A literatura recente contém vários exemplos de desenvolvimento de SGBDs específicos para suportar os requisitos das novas aplicações. Assim, por exemplo, [BAN89] apresenta um sistema de banco de dados especialmente projetado e construído para suportar orientação a objetos. O novo sistema incorpora as

características de um SGBD relacional e de um OOS (Object-Oriented System). Exemplos de outros trabalhos que se encaixam nesta área são o GERPAC [DEL89], o sistema DAMOKLES [DIT87] e o GEMSTONE [MAI85]. Vieira [VIE88] faz uma análise sucinta de alguns sistemas orientados para objetos.

1.3 - BANCOS DE DADOS ESTENDÍVEIS E O MODELO RELACIONAL

O enfoque analisado nesta tese corresponde ao dos bancos de dados estendíveis, cuja arquitetura e funcionamento interno não sejam projetados para orientação a objetos. Mais especificamente, tratar-se-á de extensões ao modelo relacional.

A definição de um novo tipo de dado para extensão de bancos de dados relacionais envolve uma sequência de fases [DAT88].

a) Requisitos Semânticos, ou seja, dados básicos (que estão na raiz da hierarquia do novo tipo de dado) e operações primitivas necessárias.
b) Detalhes Sintáticos, ou seja, constantes, variáveis, operadores, expressões e formatos de E/S.
c) Detalhes de Implementação, ou seja, representação interna e interface com a linguagem hospedeira.

Sabe-se que o grande sucesso do Modelo Relacional de Banco de Dados deveu-se ao fato de ser um modelo elegante, simples e com

fundamentação matemática. No entanto, a Teoria Relacional tem-se mostrado insuficiente para suportar sistemas "*orientados a objetos*" como os citados anteriormente. De fato, o conceito de abstração de dados é central ao modelo de dados voltado a objetos.

[MIR88] distingue os problemas a dois níveis : estrutura de dados e operacional. A nível de estrutura de dados, o Modelo Relacional tem dificuldades em representar, de uma forma natural, alguns conceitos de entidades (por exemplo, uma entidade precisa de <u>várias</u> relações normalizadas para ser representada). Apresenta, ainda, problemas em representar ligações entre domínios e relações e definir operadores sobre domínios. A nível operacional, torna-se difícil manipular ligações semânticas por escassez de operadores semânticos.

Um dos caminhos adotados para solucionar esse problema foi adaptar os sistemas relacionais existentes para manipular objetos através de ferramentas específicas. Esta solução causou sérios problemas de desempenho, pois nos sistemas relacionais a forma de armazenamento dos dados é apropriada para processamento de tabelas (relações) e não de objetos com estrutura mais complexa. O modelo de dados orientados para registros, tal como é o relacional, tem pouco poder de expressão para manipular aplicações complexas orientadas para objetos.

A representação típica de objetos no modelo relacional é feita grupando-se os sub-objetos de mesmo tipo em uma única relação (ou em tuplas com aninhamento, no modelo não normalizado)

e os sub-objetos de tipos diferentes através de atributos comuns. Assim, os objetos complexos são entidades estruturadas que são armazenadas como coleção de tuplas inter-relacionadas, as quais são manipuladas pelo sistema como uma unidade lógica (objeto atômico), ou seja, trata-se uma coleção de tuplas heterogêneas como um "objeto molecular" [BUC84].

Uma das estratégias utilizadas é estender o modelo relacional afim de torná-lo "orientado a objetos" através da utilização de Tipos Abstrato de Dados (TAD) ([BUC84], [MED87], [MED89], [NEU88], [OSB86], [STO82], [STO83]). Porém, por serem objetos passivos, estes não são suficientes para tratar os objetos como entidades dinâmicas (objetos ativos). Um TAD é um tipo de objeto que define um certo domínio de valores (objetos) e um conjunto de operadores aplicáveis a esses valores.

TADs têm sido usados para modelagem semântica de dados, como base para linguagens de programação e mais recentemente, na implementação de aplicações em SGBDs [STO83]. TADs promovem modularidade já que mudanças na implementação do tipo de dado são escondidas, não afetando os níveis superiores que utilizam esse dado. A utilização de TADs em Banco de Dados é vista no capítulo 2.

[STO82] propõe dois mecanismos para adicionar informação semântica a um sistema de banco de dados relacional : a) um sistema de regras orientadas semelhante a Inteligência Artificial, pois muitos serviços de um SGBD podem ser interpretados como sistemas de regras e b) tratamento de

atributos do esquema de uma relação como TADs. Em [STO83] o autor mostra como utilizar TADs para aplicações PAC.

Para se estender um SGBD relacional usando-se TADs, deve-se considerar três aspectos [MEL89] :

i) Relações ou Domínios como TADs (o que será discutido no capítulo 2).

ii) Relacionamento entre TAD e SGBD.

CASO 1 : o TAD é implementado com uma linguagem de programação de uso geral e os acessos ao banco de dados são feitos pelo SGBD.

CASO 2 : o TAD é implementado através da própria linguagem relacional do SGBD.

CASO 3 : Uma linguagem de programação de propósito geral é estendida com facilidades do SGBD relacional.

iii) Suporte a hierarquia de TAD.

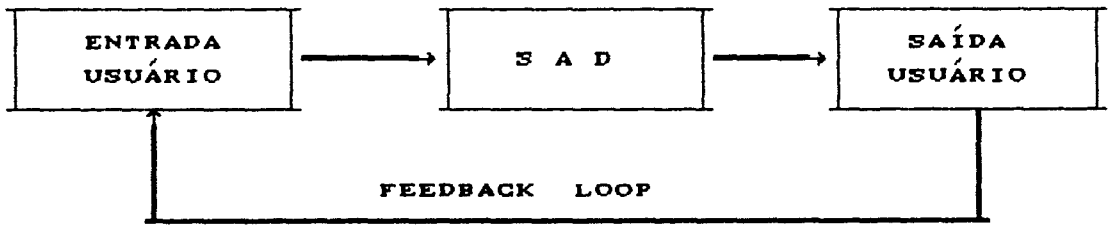
- Hierarquia de Tipos é uma propriedade muito desejada nos sistemas de banco de dados orientados a objetos. O uso de TAD permite a implementação dessa propriedade tendo como base da hierarquia o relacionamento de sub-tipos que existe entre tipos de dados. A raiz dessa hierarquia é um tipo padrão do sistema (inteiro, real, caracter, etc).

Como se verá posteriormente, o enfoque dado nesta tese encaixa-se no CASO 1.

1.4 - SISTEMAS DE APOIO À DECISÃO

SADs são sistemas surgidos nos anos 70 que se encaixam na categoria de Sistemas Gerenciadores de Informação. Estes sistemas visam reunir os recursos tecnológicos do computador com o poder de julgamento do usuário, de modo a melhorar a atividade de tomada de decisões.

Eles têm por objetivo permitir, de uma forma eficiente, o acesso aos dados importantes para o planejamento, análise e controle das atividades de uma organização. Um SAD deve prover meios de entrada, processamento e saída de dados, para que o usuário tome decisões que irão afetar a organização a curto, médio ou longo prazo. Mittra [MIT86] chama este processo de "*feedback loop*" , mostrado na figura abaixo.



Um Sistema de Apoio à Decisão pode ser classicamente definido [MEL88] como sendo um sistema que visa a aplicação, baseada em computador, de tecnologias adequadas para ajudar a melhorar a efetividade das tomadas de decisão gerencial em tarefas semi-estruturadas. Entende-se por problemas não-estruturados ou

semi-estruturados aqueles onde existe mais de uma regra de decisão que resolve as situações onde não se pode assegurar certeza em relação aos dados, aos procedimentos utilizados ou à sua sequenciação.

Atualmente um Sistema de Apoio à Decisão é conceitualmente dividido em quatro módulos ([MEL88], [MIT86]): módulo de controle, módulo de armazenamento, módulo de manipulação dos dados e módulo de construção de modelos.

O módulo de controle funciona como interface com o usuário e com os outros módulos. É tipicamente do tipo dirigido por menus, fornecendo mensagens para guiar o usuário a formular o problema e a solucioná-lo. O módulo de controle recebe as informações provenientes do usuário, as interpreta e dispara gatilhos para execução de um particular comando ou rotina. Por razões óbvias, o módulo de controle deve ser implementado como uma interface "on-line" e amigável.

O módulo de armazenamento de dados contém todos as informações necessárias ao SAD, correspondendo geralmente a um SGBD.

O módulo de manipulação é responsável pela recuperação de dados. Contém uma linguagem de consulta não procedural e geradores de relatórios e gráficos. Como geralmente estes recursos são aproveitados das facilidades do SGBD, alguns autores condensam os módulos de armazenamento e manipulação em um único módulo.

O módulo de construção de modelos (ou base de modelos) é

utilizado via sistema gerenciador de base de modelos, que permite o acesso e execução de modelos aplicáveis aos problemas existentes e garante independência das aplicações, existência de diferentes visões do modelo, compatibilidade com o SGBD e capacidade de armazenar conhecimento. O módulo de construção de modelos utiliza ferramentas de modelagem analítica, programação linear e outros princípios de otimização, análise estatística e métodos de análise de decisão. Quando o problema não permite uma modelagem analítica, utiliza-se princípios de simulação. Para tanto, é requerido um suporte computacional complexo.

Como recurso, o gerenciador de base de modelos deve ter uma Linguagem de Descrição de Modelos, para que o usuário especifique novos modelos; e uma Linguagem para Manipulação de Modelos, para permitir atualização e resolução de modelos, consultas a modelos e ligação entre modelos.

Basicamente, a construção de um Modelo segue as etapas que enumeraremos a seguir : a) formulação do problema; b)construção, se possível for, de uma Modelo Matemático; c) derivação, teste e controle da solução e d) implementação da solução.

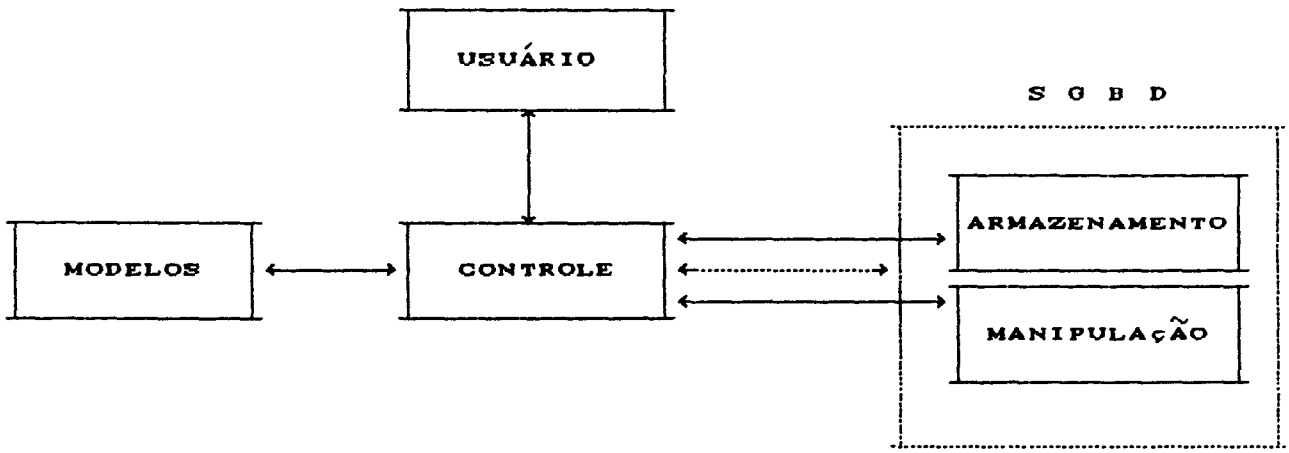
Dentre as limitações dos Modelos Matemáticos, podem ser citadas :

a) A complexidade do modelo ou o grande número de funções envolvidas exige a utilização de computador.

b) As soluções são geralmente válidas somente para um período de tempo determinado, ou seja, não são soluções definitivas.

c) Os modelos se baseiam em dados históricos e assumem que as condições do passado valerão no futuro.

A figura abaixo mostra a conexão dos módulos entre si e com o usuário.



Dentre as características de um Sistema de Apoio à Decisão pode-se citar :

i) Auxilia a tomada de decisões.

ii) Acessa dados em larga escala, inclusive dados históricos, o que exige um SGBD não tradicional para a sua implementação.

iii) Responde com rapidez as perguntas do usuário.

iv) Proporciona cenários "what if" para guiar o usuário.

v) É flexível, pois se adapta ao estilo da atividade de decidir de cada usuário.

Um SAD acessa as informações fornecidas pelo usuário, as processa e exibe o resultado para averiguação do usuário. Caso um resultado não seja satisfatório, o usuário pode repetir o processo, alterando os dados de entrada ou o meio (modelo) para produzir os resultados, até que estes sejam satisfatórios. Finalmente, o usuário pode pedir a impressão gráfica dos resultados.

Existem duas categorias de SADs [MIT86] : aqueles que executam as funções associadas a procedimentos quantitativos e aqueles que executam tarefas mais criativas e utilizam análise qualitativa. Os primeiros são os mais comuns no mercado.

O sistema discutido na tese pode ser usado, como se verá, para desenvolver SADs do primeiro tipo.

1.5 - RESUMO

Este capítulo descreveu alguns trabalhos na área de suporte a tipos de dados não convencionais em SGBDs e apresentou conceitos de SAD suficientes para o entendimento do resto da tese.

1.6 - ORGANIZAÇÃO DA TESE

O restante da tese está dividido da seguinte forma : o próximo capítulo fornece uma revisão bibliográfica sobre o

problema de atualização de visões e define a terminologia que será usada; o terceiro capítulo apresenta a arquitetura de um sistema estendível que permite a utilização e definição de novos tipos de dados a partir de visões do usuário; o quarto capítulo descreve um protótipo que implementa a arquitetura proposta; e o quinto capítulo contém conclusões e extensões. Nos anexos pode ser encontrado um exemplo de utilização do protótipo implementado.

CAPÍTULO 2

VISÕES, TADs E AGREGADOS

2.1 - INTRODUÇÃO

A tese propõe uma arquitetura de banco de dados (relacionais) estendíveis onde as facilidades para extensão são providas através de visões. Os tipos analisados são baseados em funções agregadas. Este capítulo fornece a base para a arquitetura proposta.

2.2 - ATUALIZAÇÕES EM VISÕES

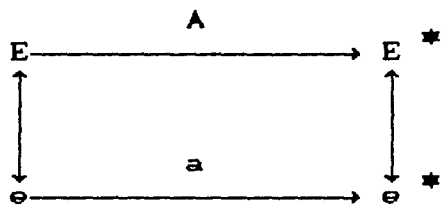
Uma "visão" é uma relação derivada que, ao contrário de "instantâneos" (snapshots), caracteriza-se por espelhar o estado corrente das relações sobre as quais ela é definida [FUR85]. Existem dois tipos de enfoque para implementar uma visão : virtuais e materializadas. Visões virtuais têm suas funções geradoras armazenadas no Dicionário de Dados e são materializadas quando referenciadas pelo usuário. Visões materializadas constituem relações derivadas armazenadas no banco de dados sendo atualizadas através de uma política de manutenção de visões materializadas, sempre que as relações-base usadas para sua

definição são modificadas. [SHM84] e [BLA85] apresentam métodos eficientes para atualizar visões materializadas, baseados em algoritmos que detectam quaisatualizações sobre as relações-base são relevantes para a formação da visão e em algoritmos diferenciais que reprocessam somente as tuplas necessárias à atualização.

O usuário interage com uma visão requisitando consultas e atualizações sobre ela [FUR85]. Nesta tese, o termo visão aparece não como "tabique" [FUR79], que impede ao usuário acesso a dados, mas sim como "janela", que mostra ao usuário apenas dados de seu interesse em determinado momento. Esta tese se aterá a estudar os processos de atualização sobre visões, sem considerar algoritmos para sua materialização.

Qualquer pedido de atualização sobre uma visão requer a tradução das operações sobre a visão para operações correspondentes sobre o banco de dados, caracterizando o chamado "problema de atualização de visões". Este problema tem sido analisado por diversos autores ([BAN81], [BLA85], [BLA87], [DAT86],[DAY82], [FUR79], [FUR85], [KEL85], [KEL86a] [KEL86b], [MED85], [MED86]). A grande dificuldade apontada é a de traduzir as modificações solicitadas para o banco de dados sem perder a integridade. A tradução de uma atualização sobre uma visão é dita exata quando não provoca nenhum tipo de efeito colateral nem viola a consistência semântica do banco de dados. A definição de maneiras de escolher a tradução correta de uma atualização - isto é, aquela que faz a visão mudar precisamente de acordo como

requerido pelo usuário - acaba por limitar o conjunto de visões que podem ser modificadas, já que poucas visões se sujeitam às regras impostas. Uma tradução correta pode depender, inclusive, da intenção do usuário - ou seja, da semântica da atualização. A figura resume os problemas que se pode encontrar ao aplicar a atualização "A" à visão externa "E", visando estado externo final "E*". Os caracteres minúsculos correspondem ao estado interno do banco de dados, sendo que "a" é uma possível tradução para "A" [BAN81].



Problemas :

- *Mais de 1 mapeamento*
- e^* *ñao corresponde a* E^*
- e^* *inconsistente*

A imposição de unicidade de tradução é questionável. A partir do momento em que o usuário faz sua solicitação, as operações são traduzidas em operações correspondentes sobre a extensão do esquema conceitual. A forma de obtenção das traduções pode ser classificada em dois tipos de tratamento [MED85] : estabelecendo critérios gerais de mapeamento ou aproveitando a noção da TADs.

No primeiro caso, são definidos procedimentos generalizados para traduzir uma atualização, que tem como entradas a definição da visão, a requisição do usuário e o estado corrente do banco de dados. A atualização é modificada, sendo acrescida das restrições constantes da visão e, se aceitável, é feita a tradução da

requisição modificada em operações sobre o banco de dados. Havendo ambiguidade na tradução, ou sendo impossível classificar a atualização dentro das classes (restritas) de atualizações permitidas, ela não pode ser atendida. Bancilhon e Spyratos [BAN81] introduzem o conceito de visão complemento e mostram que existirá um único tradutor para uma determinada visão se o complemento desta for mantido constante (inalterado). Keller ([KEL86a], [KEL86b]) propõe uma alternativa para acabar com a ambiguidade semântica das traduções dos pedidos de atualização sobre visões, introduzindo a semântica necessária através de diálogos com o usuário no momento da definição da visão.

No segundo caso, a definição da visão engloba as operações que são permitidas sobre ela, bem como sua tradução. Esta tradução é especificada no momento de definição da visão, sendo então considerados não apenas regras sintáticas, como no primeiro caso, mas também, semânticas ([SEN88], [TUC85]).

Estas regras seguem, basicamente, os seguintes princípios [DAT83] :

a) Monitoração das transações, especialmente as de atualização, para detecção de possíveis violações à integridade.
b) Na ocorrência de uma violação (sintática e/ou semântica), deve-se avisar o usuário, podendo-se tanto rejeitar a transação como tentar-se validá-la, adicionando-se semântica a ela, através de interação com o usuário.

Um resumo recente dos problemas na área é dado por Keller [KEL86a]; Medeiros e Tompa [MED86] discutem maneiras de liberalizar mapeamentos e consequentemente ampliar o universo de visões passíveis de atualização; Blakeley [BLA87] analisa um problema correlato - o de manutenção de integridade em visões materializadas quando as relações-base são atualizadas. Lafue ([LAF82a], [LAF82b]) discute o gerenciamento de integridade de banco de dados e faz uma análise de verificação protelada e verificação imediata.

O sistema de mapeamento sugerido para a tese optou por dar flexibilidade ao usuário : cada pedido de atualização tem uma tradução "default", mas o usuário pode optar por uma das alternativas apresentadas. Para que o mapeamento seja mantido sob controle, o número de alternativas é reduzido, considerando-se que cada solicitação pode corresponder a uma classe de operações, cada qual associada a um mapeamento distinto, cabendo ao usuário indicar a operação desejada. Este trabalho se concentra em analisar atualizações sobre um tipo especial de visões : aquelas que contêm campos agregados. Enquanto formas de cálculo de agregados já aparecem na área de otimização de consultas ([FRE86], [BUL87], [DAY87]), há pouquíssimas referências na literatura sobre atualizações, geralmente no sentido inverso : o efeito sobre uma visão quando atributos base para cálculo de um agregado são modificados.

2.3 - DEFINIÇÕES E NOTAÇÃO-AGREGADOS

Segundo a notação de Maier [MAI83], Date [DAT84], o esquema de uma relação é denotado $R(A_1, \ldots, A_n)$, onde R é o nome da relação e A_i , para $i = 1 .. n$, são atributos do esquema. Cada atributo "A_i" tem um domínio "$DOM(A_i)$" de valores associados.

Uma tupla t é um conjunto de valores ordenados segundo os atributos do esquema. Cada valor pertence ao domínio do atributo correspondente. Uma relação é uma ocorrência de esquema R e é composta de um conjunto de tuplas.

Um esquema de uma visão V definida sobre um banco de dados R é um conjunto dos nomes dos atributos que compõem a visão. Estes atributos correspondem àqueles obtidos do esquema de R a partir do que se costuma chamar <u>função geradora da visão</u> : uma sequência de operações que, aplicadas a um banco de dados, fornece a imagem de uma consulta (visão). Além dos operadores seleção, projeção e junção natural, este trabalho considerará igualmente alguns operadores agregados como parte de uma função geradora.

Um operador agregado corresponde à aplicação de alguma função ou rotina sobre atributo(s) de um banco de dados, de forma a obter um novo atributo, cuja restrição de integridade sobre o domínio não necessariamente coincide com as restrições de integridade dos domínios dos <u>atributos fonte</u>, ou seja atributos que participam da definição do agregado. Assim como é feita a distinção entre esquemas e ocorrências, aqui se introduz uma notação, correspondendo à diferença entre agregado (definição da função a aplicar) e valor do agregado (resultado da aplicação da

função a ocorrências dos atributos). Dir-se-á que um agregado tem parâmetros (nomes das variáveis/atributos); cada instanciação dos parâmetros resultará num valor específico do agregado. Assim, por exemplo, o agregado $AG = A1 + A2$ (soma de dois atributos) tem $DOM(AG) = (x + y)$, onde $x \in DOM(A1)$ e $y \in DOM(A2)$.

Um agregado pode ser referenciado pela tupla $AS = \langle AP, AG \rangle$ onde AP é o conjunto de atributos que são parâmetros do agregado; AS é o nome do atributo que receberá o valor do agregado e AG é a função geradora do agregado, isto é, a função que calcula o agregado.

Este trabalho distingue vários tipos de agregados :

* agregado simples é calculado usando um único atributo.
* agregado composto é calculado a partir de mais de um atributo.
* agregado multi-relação é um agregado composto onde os parâmetros provêm de mais de um esquema.

Para os agregados sobre um único esquema, definimos :

* agregado horizontal é aquele cujos valores são calculados a partir de tuplas (ou seja, cada instanciação é obtida a partir dos valores dos atributos de uma dada tupla da relação-base).
* agregado vertical é aquele cujos valores são calculados sobre valores de um mesmo atributo (projeção) de uma relação (a partir de todas as tuplas de uma relação-base

para um determinado atributo atômico).

Muitos sistemas comerciais de banco de dados relacionais aceitam apenas agregados sobre uma única relação e em geral apenas verticais.

Sejam $R1 = (A1, A2, A3)$ e $R2 = (A4, A5)$ esquemas e AG um agregado. $AG = (A1 * cte)$ é um agregado simples e $AG = (A1 + A5)$ é um agregado composto multi-relação. Seja ϕ uma instanciação de AG para uma ocorrência $r1$ de $R1$, com n tuplas. Seja $t_k(A_j)$ o valor do atributo j para alguma tupla t_k da ocorrência $r1$. O agregado horizontal $AG = (A1 + A2)$ obedece $(\phi_k(AG)) = (t_k(A1) + t_k(A2))$ (para $k = 1 .. n$); o agregado vertical $AG = SOMA (A1)$ obedece $(\phi (AG)) = \sum t_k(A1)$ (para $k = 1 .. n$).

O exemplo a seguir mostra alguns problemas existentes na atualização de um agregado. Seja $R = (DISciplina, ALuno, Nota1, Nota2, Nota3)$ e sejam as visões $V1 = (DIS, AL, MED)$ e $V2 = (DIS, AL, NOTAMAX)$, onde $MED = (N1 + N2 + N3) / 3$ e $NOTAMAX = MAX(N1, N2, N3)$ são agregados horizontais. Seja o estado do banco de dados e as visões correspondentes.

BD :

DIS	AL	N1	N2	N3
c10	joão	3.0	7.5	4.5
c10	maria	8.0	8.0	2.0
c15	rita	3.0	9.0	6.0

V1 :	DIS	AL	MEDIA
	c10	joão	5.0
	c10	maria	6.0
	c15	rita	6.0

V2 :	DIS	AL	NMAX
	c10	joão	7.5
	c10	maria	8.0
	c15	rita	9.0

A modificação através de *V1* da média de "joão" de 5.0 para 6.0 pode ser conseguida, por exemplo, atribuindo a *N1*, *N2* e *N3* os seguintes valores :

⟨Ni ← Ni + 1⟩; ou *⟨Ni ← 6.0⟩*; ou *⟨N1 = 0.0, N2 = 8.0, N3 = 10.0⟩*; ect.

Em cada alternativa, a mudança seria refletida na visão *V2*. Do mesmo modo, aumentar a nota *MAX* de "rita" em *V2* de 9.0 para 10.0 pode ter várias interpretações, inclusive até a de atribuir a uma das notas o valor 10.0 e tornar as demais iguais a zero. O problema, como já mencionado antes, é que uma atualização de agregado é bastante ambígua e o mapeamento pode variar, para um mesmo pedido, dependendo de quem (e quando) fizer a solicitação.

Todos os agregados mencionados até agora têm como parâmetros atributos de esquemas de relações-base do banco de dados; chamemos a estes agregados de 1ª ordem. Estes últimos podem por sua vez ser parâmetros de agregados, e assim sucessivamente. Para facilitar a notação, conveciona-se aqui chamar de agregados de i-ésima ordem aqueles que têm, ao menos, um parâmetro que é um agregado de ordem (i - 1).

Nesta tese, os agregados correspondem a atributos de esquemas de visões. Cada atualização em uma visão é mapeada para uma

sequência de atualizações no banco de dados; em particular, cada atributo agregado atualizado requer modificações em vários atributos fonte. Atualizações de agregados de ordem 2, por exemplo, exigem mapeamento para seus parâmetros (ao menos um dos quais agregado) e destes para os atributos da relação-base. Quanto mais alto o nível de um agregado, maior a complexidade do mapeamento, já que a cada nível pode haver várias opções válidas para traduzir um pedido de atualização.

O anexo 1 contém exemplos de diferentes mapeamentos de atualização de agregados.

2.4 - VISÕES COMO TIPOS ABSTRATOS DE DADOS

Afim de minorar os problemas de atualização através de visões sobre as relações-base de um banco de dados, surgiu um novo enfoque, em geral classificado como tratamento de visões como Tipos Abstratos de Dados (TADs). Um dos trabalhos mais detalhados na área é o que define visões a partir de módulos ([TUC83], [TUC85],[SEN88]). O sistema RAD [OSB86] implementa TADs em banco de dados relacionais. Ambos trabalhos são discutidos nesta seção.

Modularização é parte de uma estratégia que visa o projeto estruturado de banco de dados do ponto de vista de esquema conceitual. A estratégia visa, sobretudo, diminuir a complexidade das descrições dos banco de dados. A definição de um módulo consiste na descrição em alto nível das suas estruturas e

operações, bem como das restrições de integridade impostas. Um módulo pode englobar uma visão e um ou mais esquemas. Ao usuário é concedido acesso ao módulo, e assim à visão nele definida. A definição de cada visão deve conter não apenas a descrição de como acessar os dados da visão - ou seja, sua função geradora - mas também a descrição do conjunto de atualizações permitidas através delas, cada qual acompanhada das respectivas traduções em atualizações sobre os módulos que lhe deram origem.

Este enfoque requer como suporte uma linguagem que permita definir visões, bem como metodologias para se verificar quais traduções são corretas, de acordo com critérios previamente definidos. As relações manipuladas por um módulo *M* devem ser somente atualizadas por operações contidas em *M*. Isto corresponde a noção de encapsulamento em um tipo abstrato de dados.

As definições da visão, suas operações e respectivas traduções seriam embutidas em módulos com a seguinte estrutura :

MÓDULO *M* estende "outros módulos"
  Esquemas (*E*)
  Restrições (*R*)
  Operações (*O*)
  Função Geradora (*V*)
  Traduções "exatas" para *O* (*T*)
FIM-MÓDULO

onde :

i) A tripla *M* = (*E*, *R*, *O*) define um módulo propriamente dito, sendo :

a) "*E*" o conjunto de esquemas de relações definidas em *M*.

b) "*R*" é o conjunto de restrições de integridade sobre os esquemas relacionais em "*E*".

c) "*O*" é o conjunto de operações sobre relações com esquemas em "*E*".

ii) "*V*" é o conjunto de expressões através das quais visões serão materializadas sobre as relações-base, com esquemas em "*E*". Cada expressão é uma função geradora de visão.

iii) "*T*" contém, para cada operação definida uma tradução com uma ou mais operações sobre "*E*".

Pode-se descrever as seguintes vantagens quando se adota uma visão como TAD :

a) As atualizações não produzirão efeitos colaterais indesejáveis.

b) Serão evitadas ambiguidades na tradução das operações.

c) As restrições impostas pela linguagem de manipulação de dados (LMD) não irão afetar as operações de atualização, pois estas já foram previamente definidas.

d) Será preservada a integridade sintática e consistência semântica do banco de dados : as restrições de integridade serão automaticamente preservadas, já que as atualizações na visão são restritas a operações predefinidas, com efeitos

limitados e conhecidos.

e) Se o banco de dados é atualizado somente através das operações visíveis aos usuários, o estado do banco de dados sempre permanecerá consistente.

f) Os usuários podem interagir com o sistema de banco de dados como se realmente eles estivessem atualizando as visões.

g) O chamado "problema de atualização de visões" é evitado, já que as traduções das operações de atualização são definidas na fase de projeto (nível de especificação) e então embutidas na definição dos módulos.

Como desvantagens, pode-se citar :

a) É muito trabalhoso definir as operações. Além disto, não está descartada a possibilidade de ter que redefini-las durante a vida útil do banco de dados.

b) SGBDs comerciais não permitem fazer declarações nem determinar restrições como as requeridas na definição de módulos. A implantação de mecanismos deste tipo só pode ser feita através de rotinas ativadas por gatilhos.

Segue-se um exemplo de como definir um módulo contendo agregados MÉDIA e MAX e suas atualizações.

Seja **M1** o módulo que contém a descrição do esquema de relação *R1 (DIS, AL, N1, N2, N3)*.

Notação : $\Pi_{N1}\ \sigma_{AL=a1} R1 = a1.R1(N1)$

O símbolo ";" indica composição de operações, onde a ordem da sequência das operações é relevante e o símbolo "||" indica concatenação.

Supõe-se que as operações permitidas sobre R1 definidas em **M1** são *INS(R1,t)* e *EL(R1,t)*.

MÓDULO **M** estende **M1**

- *Esquemas* : V1 [DIS, AL, MEDIA]

  V2 [DIS, AL, NMAX]

- *Restrições* :

  DOMÍNIO : $\forall$ p, $\forall$ q, $\forall$ r, V1(p,q,r), $r \geq 0.0 \wedge r \leq 10.0$

- *Operações* :

  * **MODIFMED (V1(D,A,MD))** :

    Se V1(DIS) = D & V1(AL) = A então V1(MEDIA) $\leftarrow$ MD;

  * **MODIFMAX (V2(D,A,MX))** :

    Se V2(DIS) = D & V2(AL) = A então V2(NMAX) $\leftarrow$ MX;

- *Funções Geradoras* :

  V1 [DIS,AL,MEDIA] : $\Pi_{DIS,AL}^{R1}$ || S1; onde S1 : MEDIA = $\sum_{i=1}^{3} ni/3$

  V2 [DIS,AL,NMAX] : $\Pi_{DIS,AL}^{R1}$ || S2; onde S2 : NMAX = MAX(N1,N2,N3).

■ *Operações Substitutas :*

MODIFMED (V1(D,A,MD)) :

MAP1 :

EL(R1,(D,A,N1,N2,N3)) ; INS(R1,(D,A,N1+MD,N2+MD,N3+MD)).

MODIFMAX (V2(D,A,MX)) :

MAP3 :

x ← A.V2(NMAX);

EL(R1,(D,A,N1,N2,N3)) ; INS(R1,(D,A,N1 * MX/x, N2 * MX/x,
N3 * MX/x)).

OBS : As operações substitutas descrevem mapeamentos para as operações em função de operações *EL*, *INS* de *M1*, sendo a notação de [TUC83] adaptada para facilitar o exemplo. Caso se desejasse mais de uma opção de mapeamento, como se sugere na tese, as operações substitutas poderiam aparecer como se segue.

■ *Operações Substitutas :*

MODIFMED (V1(D,A,MD)) :

Caso

MAP1 :

EL(R1,(D,A,N1,N2,N3)) ; INS(R1,(D,A,N1+MD,N2+MD,N3+MD)).

MAP2 :

x ← A.R1(N1);

$$y \leftarrow 3 * MD - (A.R1(\sum_{i=1}^{3} Ni)) - x;$$

i ← 1;

```
        Enquanto y ≥ 10.0 faça
         z ← y - 10.0;
         Ni' = A.R1(Ni) ← 10.0;
         i ← i + 1;
         y ← A.R1(Ni) + z;
        Fim-Enquanto
        Ni' = A.R1(Ni) ← y;
        EL(R1,CD,A,N1,N2,N3)) || INS(R1,CD,A,N1',N2',N3')).
    MODIFMAX (V2(D,A,MX)) :
    Caso
        MAP3 :
        x ← A.V2(NMAX);
        EL(R1,CD,A,N1,N2,N3)) || INS(R1,CD,A,N1 * MX/x, N2 * MX/x,
                                     N3 * MX/x)).
        MAP4 :
        x ← A.V2(NMAX);
        EL(R1,CD,A,N1,N2,N3)) || INS(R1,CD,A,N1 + (MX - x), N2 +
                                     (MX - x), N3 + (MX - x))).
FIM-MÓDULO
```

Ao se requerer a modificação da média de umaluno em V1, pode-se levar a alteração ao agregado NMAX referente àquele aluno em V2. O efeito colateral é previsível pois alterando-se os atributos fonte o valor do máximo pode mudar.

Se uma atualização em uma das visões provocar modificações em outras visões, elas não serão tratadas como efeitos colaterais,

pois são modificações necessárias para manter a integridade do banco de dados. A definição de módulo constituído por extensão garante que as modificações efetuadas mantêm a consistência das demais visões [TUC83].

Em relação aos mapeamentos descritos em **M**, MAP1 representa uma distribuição uniforme, MAP2 uma distribuição não uniforme automática, MAP3 processa modificação proporcional nos valores dos atributos fonte para o máximo e MAP4 altera os valores dos atributos fonte do mesmo valor absoluto. O anexo 1 define estes mapeamentos, todos incorporados ao protótipo implementado.

Uma implementação que flexibiliza o modelo relacional para suportar aplicações que requerem tipos de dados não convencionais usando TADs é o sistema RAD [OSB86]. O sistema permite a definição de novos tipos de dados e operações mantendo as características de um SGBD tradicional, pois tanto pode ser utilizado interativamente como através de uma interface que contenha uma linguagem hospedeira embutida. RAD é um SGBD relacional que permite a definição de novos domínios como TADs.

Afim de facilitar a definição de novos tipos de dados o sistema RAD foi organizado em dois níveis : nível de abstração e nível de implementação.

No nível de abstração, os novos tipos e as operações sobre eles são definidos através de LDD. Neste nível os dados são vistos como objetos. Para que o SGBD possa manusear os valores dos novos tipos é necessário incorporar suas representações internas ao sistema e definir suas operações primitivas. Estas

tarefas são realizadas no nível de implementação. Oferecendo-se rotinas padrão já definidas (inserir-tupla, eliminar-tupla, recuperar-tupla, etc) o usuário não precisa saber como as relações que representam os novos tipos foram implementadas pelo sistema. Esta organização garante a preservação da integridade do banco de dados, pois as operações estão sob o controle do SGBD.

Para definir novos domínios em RAD o usuário dispõe de três classes de operações : primitivas, agregado e transformação.

Operações primitivas são utilizadas para se definir constantes, operadores, operações de inserção e atualização de valores, "layout" de saída para os novos dados e predicados.

Agregado é qualquer operação que mapeia uma relação em um valor pertencente ao domínio de um tipo, ou seja, um agregado recebe uma relação como parâmetro e retorna um valor como resultado.

Por exemplo, a consulta mostrada a seguir sobre a relação *EMPLOYEE* recupera o empregado que ingressou mais recentemente na empresa [OSB86].

SELECT *EMPLOYEE* WHERE
(DATE_OF_BIRTH = LATEST_DATE (PROJECT *EMPLOYEE* OVER DATE_OF_BIRTH))

onde DATE_OF_BIRTH é um atributo do tipo *DATE* pertencente ao esquema de *EMPLOYEE* e LATEST_DATE é um procedimento, definido pelo usuário, que retorna um agregado.

Do ponto de vista desta tese, LATEST_DATE é um **agregado vertical** e o parâmetro do procedimento que calcula o agregado pode ser interpretado como sendo o atributo de uma **visão**, cuja função geradora é a expressão (PROJECT *EMPLOYEE* OVER DATE_OF_BIRTH).

Transformação é uma operação que recebe como entrada uma relação e produz, como saída, outra relação. Para definir uma operação de transformação é necessário declarar a nova operação no nível de abstração e codificá-la, no nível de implementação, para sua posterior incorporação no dicionário de dados.

Por exemplo, uma operação de transformação pode ser usada para converter uma relação contendo imagens gráficas em uma determinada representação para uma outra relação contendo uma representação diferente para as mesmas imagens [OSB86].

A linguagem de consulta de RAD é baseada em álgebra relacional e a autora faz uma comparação desta linguagem com uma outra, provada "relacionalmente completa", para demonstrar a "completude relacional" da primeira.

2.5 - RESUMO

Este capítulo apresentou alguns problemas relativos à atualização de visões e manuseio de agregados, bem como a incorporação de TADs a banco de dados.

O capítulo seguinte utiliza estas noções para propor um novo tipo de visão (VE) para sistemas (relacionais) estendíveis, que

incorporam módulos e agregados para permitir aos usuários definir os seus tipos de dados.

CAPÍTULO 3

SISTEMA PROPOSTO

3.1 - INTRODUÇÃO

Esta tese descreve a construção de uma interface para o usuário de bancos de dados relacionais que ofereça mecanismos de definição e manutenção de certos tipos de dados não convencionais. A forma pela qual esta facilidade é implementada é a de permitir ao usuário a definição e manipulação de visões que incorporem o uso de tipos não suportados diretamente pelo SGBD. A tese denomina de "*visão estendida*", ou "*VE*", esse tipo de visão especial.

O sentido do termo "visão" é aqui ampliado, similarmente ao especificado por [MED87], ou seja, um conjunto de dados e operações a eles associados. Desta forma, não é essencial que o SGBD hospedeiro suporte a noção de visão, uma vez que o sistema proposto se encarrega de suportar esta facilidade dentro dos limites estabelecidos (ver seção 3.8).

Este capítulo fornece a descrição geral da interface proposta para viabilizar a definição de VEs. O capítulo 4 descreve a implementação de um protótipo a partir desta especificação, que permite a definição de tipos não convencionais para uma família específica de aplicações.

Este trabalho se restringirá à análise e solução de mapeamentos de atualizações de visões que contenham atributos agregados com as seguintes características :

a) A visão é definida usando junção, projeção, seleção, união de fragmentos horizontais sobre um mesmo esquema e agregados.
b) Os agregados são calculados sobre atributos de uma única relação (mesmo que a visão seja gerada a partir de junções), não sendo considerados agregados multi-relação.
c) Para uma dada relação serão consideradas apenas restrições de integridade do tipo *dependência de chave*. Uma restrição é do tipo *dependência de chave* quando é uma dependência funcional e qualquer atributo da relação depende exclusivamente da(s) chave(s). Em outras palavras as relações estão em *BCNF*. Esta restrição visa eliminar problemas de mapeamentos de atualização de visões, seguindo a linha de ([KEL85], [KEL86a], [KEL86b]), e tornar os esquemas independentes (integridade local implicando integridade global).
d) As atualizações em agregados serão consideradas isoladamente, sem levar em consideração seus efeitos em outros campos da mesma visão ou de outras visões. Parte-se do pressuposto que estes reajustes serão visíveis a partir do momento em que as relações-base são atualizadas - os efeitos colaterais da atualização de um agregado podem ser tratados da mesma forma que efeitos colaterais da atualização da visão em geral.

e) Devido aos problemas inerentes ao mapeamento de atualização sobre visões formadas a partir de funções não monotônicas, supõe-se a motonicidade não só das funções geradoras de visão como também das geradoras de agregados

3.2 - VISÕES ESTENDIDAS E ATRIBUTOS ESTRUTURADOS

Uma VE é uma facilidade proposta para estender bancos de dados relacionais que incorpora as noções de tipos abstratos de dados as visões, permitindo assim, aos usuários, a definição de novos tipos a partir de um conjunto de tipos e operações básicas.

Uma VE tanto pode ser definida a partir de relações-base quanto a partir de outras VEs, à semelhança de [FUR79]. Para salientar este fato, usar-se-á o termo tabela-base de uma VE, ao invés de relação-base para denotar as relações e as VEs usadas para gerar uma visão estendida. Uma visão estendida é uma tabela cujo esquema tem atributos (atômicos e monovalorados) de dois tipos : simples e estruturados. Os primeiros correspondem a mapeamento direto (1:1) sobre atributos da tabela-base. Os últimos são atributos correspondentes a novos tipos, definidos de forma semelhante a TADs, interativamente, pelo usuário. A eles são associadas rotinas de definição e mapeamento de suas atualizações para as tabelas-base. O valor de um atributo estruturado de uma tupla de VE é obtido a partir da aplicação de funções que combinam um ou mais valores de atributos base (ou estruturados) das tuplas das tabelas-base.

Visões estendidas podem ser tanto ativadas durante uma sessão como estar permanentemente materializadas. Se materializadas, as visoẽs passam a ser relações (derivadas) com atributos também derivados, porém estáticas. Como se verá posteriormente, uma visão permanentemente materializada é tratada como um instantâneo.

Um atributo simples reflete diretamente valores de atributos de tabelas-base; um atributo estruturado é resultante da aplicação de uma função geradora de agregado, discutida no capítulo 2, sobre atributos de tabelas-base.

Um atributo estruturado é baseado em conceitos semelhantes ao que [ALB85] chama de tipo abstrato. Ao descrever sua linguagem, os autores definem tipo como um conjunto de valores e as operações primitivas a eles associados. Tipos embutidos na linguagem são chamados conceitos. Tipos abstratos são aqueles definidos pelo usuário, para facilitar a representação de novos conceitos e correspondem a um mecanismo para permitir encapsulamento de dados e operações. Formalmente, cada atributo estruturado de uma VE é definido pela tupla

$$< AE_j, (O_k, [TE_k]) >,$$

$$\text{sendo} \quad AE_j = < (AY_i), AG_j >$$

onde AE_j é a definição da formação do atributo como agregado usando a notação de capítulo 2 ; (AY_i) é o conjunto de atributos fonte - simples e estruturados - que servem de parâmetros para a

função geradora do agregado AG_j; os pares $\langle O, TE \rangle$ representam as operações de atualização aceitáveis sobre as tuplas da VE e as traduções destas operações em operações sobre as tabelas-base. Para um dado atributo, as traduções TE_k dizem respeito apenas ao comportamento do atributo AE_j quando solicitada a atualização O_k da tupla contendo o atributo.

Medeiros [MED89] define formalmente uma visão-estendida como sendo a tupla

$$V_j = \langle \xi, G_j, \langle O_k, T_k \rangle \rangle,$$

onde G_j é a função geradora de VE e que inclui seleção, junção natural, projeção e funções agregados, ξ é o conjunto de esquemas de tabelas-base para V_j e $\langle O_k, T_k \rangle$ é o conjunto de operações permitidas sobre a visão com os respectivos mapeamentos associados. O mapeamento T_k de uma atualização de uma tupla de VE não inclui a tradução da operação quanto aos atributos estruturados da visão. Este segundo mapeamento é deixado à definição de cada atributo.

Os domínios não convencionais tratados aqui se referem a valores agregados. Na verdade, pode se considerar todos os atributos como agregados, sendo a identidade a função que mapeia atributos simples. Em outras palavras, atributos simples são definidos por

$$\langle AS_j, \langle O_k, [T_k] \rangle \rangle$$

sendo AS_j = < A_i, I > onde I é a identidade de A_i para algum atributo da tabela-base.

3.3 - ARQUITETURA GERAL DA INTERFACE PARA SUPORTE A VEs

O sistema proposto consiste em três módulos principais: módulo de geração, módulo de consistência e módulo de manipulação. O diálogo com o usuário é feito acionando-se um quarto módulo, o de interface universal.

O módulo de geração permite ao usuário especificar VEs, incorporando novos tipos às aplicações.

O módulo de manipulação contém as rotinas que permitem consultar e modificar as tuplas de uma VE, propagando as atualizações para as tabelas-base.

O módulo de consistência se responsabiliza por garantir a validade das operações de atualização sobre a visão e os respectivos mapeamentos para as tabelas-base, invalidando-as se ferirem as regras de integridade do Banco de Dados.

O módulo de interface universal é responsável por interpretar as solicitações do usuário e acionar um dos outros dois módulos. Ele é baseado em princípios de relação universal.

Esta seção trata da descrição de cada um destes módulos. A figura *fig-1* mostra esquematicamente sua interação.

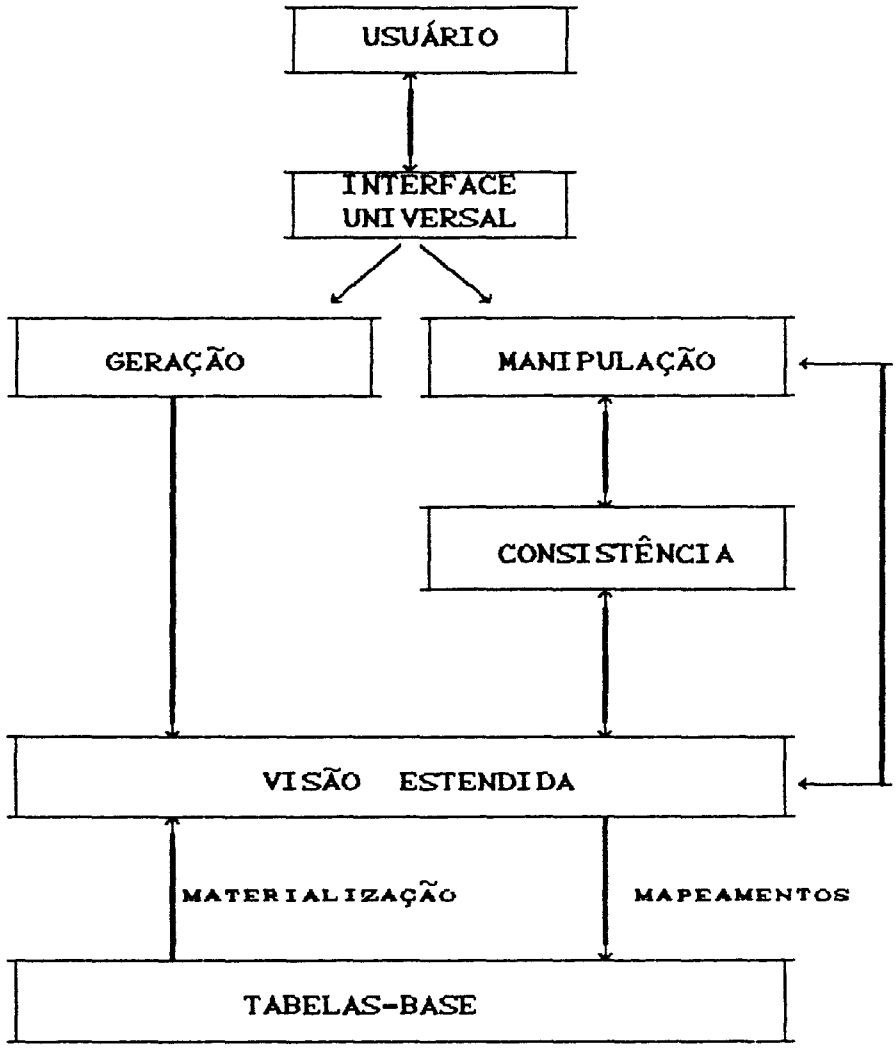


fig-1

As idéias básicas da proposta deste sistema pretendem permitir as seguintes atividades :

- análise de mecanismos associados à manipulação de visões para permitir a utilização de dados não convencionais.

- utilização das estruturas inerentes a TADs e MÓDULOS (descritos no capítulo 2) como mecanismos de construção e suporte das operações de consulta e atualização, com mapeamentos escolhidos pelo usuário e não pelo sistema.

- especificação de uma interface flexível no que diz respeito a mapeamentos de atualização. Como se verá a seguir, estas opções tanto podem vir embutidas no módulo de definição de VEs, aproximando-se das noções de *TADs* e *MÓDULOS*, como serem escolhidas no momento da atualização, como em um *Sistema de Apoio à Decisão (SAD)*.

3.4 - MÓDULO DE INTERFACE UNIVERSAL

Há várias formas de se especificar uma visão. Em geral, ela é feita através da definição de uma consulta sobre as relações base. Para dar maior flexibilidade ao sistema, já que muitos de seus atributos não têm correspondentes nos atributos das relacões base, optou-se por especificar a função geradora de visão indiretamente, através da enumeração dos atributos que compõem a visão. Para atributos simples, basta fornecer seu nome; para atributos estruturados, a especificação é mais complexa. Na relação universal, o usuário enxerga os atributos como pertencentes a uma única relação e se refere a eles pelos seus nomes, sem especificar as relações-base onde são definidos. Uma Interface Universal é aquela que dá ao usuário uma visão do tipo

relação universal do banco de dados; cabe à interface fazer a tradução da solicitação do usuário, através de nomes de atributos e o acesso às relações apropriadas.

A interface proposta se assemelha à usada em [NEU88], por usar a noção de relação universal. Afim de evitar ambiguidades e garantir respostas corretas às consultas, os atributos do esquema conceitual devem ter semântica bem definida e nomes únicos.

Outro exemplo de interface universal é o trabalho de [BRO88], que considera uma interface baseada na hipótese de relação universal fraca ("weak instance assumption"). Nesta hipótese, não se considera como tuplas válidas para a formação da visão aquelas cuja projeção sobre os atributos da tabela-base contenha valores nulos (chamada também projeção total).

A VE, na arquitetura proposta é uma visão definida sobre a visão de interface universal. Como explicado em [BRO88], a solicitação do usuário é respondida através de mecanismos do tipo janelas (windows); o problema na implementação de uma interface universal é determinar como traduzir uma expressão de janela em solicitação ao banco de dados. Como apontam os mesmos autores, existem vários algoritmos para solucionar o problema. O trabalho mencionado salienta que a interface nele proposta difere das demais porque permite atualização e consultas (ao passo que os demais permitem apenas consultas).

Nesta tese, a interface universal é bem simplificada e portanto muitos dos problemas apontados (como por exemplo, tratamento de nulos) deixam de existir. Dentre as simplificações

mais importantes estão : (1) todas as junções são consideradas sem perdas; (2) as chaves de todas as tabelas-base de uma VE precisam aparecer na VE; e (3) não se considera o modelo universal fraco, que ignora nulos. Além disto, as únicas restrições de integridade existentes são dependências de chave, não havendo dependência entre relações.

Desta forma, a correspondência entre a especificação do usuário e a definição dos atributos é discutida a seguir.

Quando um usuário referencia um atributo que aparece em uma única tabela-base, supõe-se projeção daquele atributo; se o atributo é definido em mais de uma tabela-base, supõe-se a junção sem perdas das relações correspondentes com projeção sobre o atributo. Condições de seleção são condições impostas a valores de atributos. Finalmente, como chaves não podem ser modificadas, então atributos de junção por serem chaves de uma das relações, não podem ser modificados. No caso de atributos estruturados, não há problema em determinar a origem de seus parâmetros para cada atributo, já que por hipótese não se consideram agregados multi-relação.

Desta forma, os parâmetros de cada atributo estruturado só podem estar em uma única tabela-base e não devem corresponder a atributos de junção.

Esta restrição pode ser contornada parcialmente pelo fato de que VEs podem servir de base para VEs. Considere, por exemplo, *R1* = (A, *B*, *C*) e *R2* = (C, *D*) esquemas de um banco de dados.

Seja *VE1* = (A, *C*, *N*) o esquema de uma VE, onde *N* é um

atributo estruturado definido por *AG(D)*. Os valores materializados correspondem a $\Pi_{A,C,N}{}^{R1}{}_{C,N}{}^{\infty}$ R2 , correspondendo os valores de *N* aos obtidos da relação *(C, AG(D))*. É válido definir visões estendidas usando *N*, mas não é válido definir atributos estruturados que tenham *B* e *N* como parâmetros (porque são oriundos de esquemas distintos) ou *C* e *N* como parâmetros (embora presentes em um único esquema, *C* é chave de R2 e aparece em mais de uma relação).

No caso da interface aqui proposta, não há preocupação em esconder nulos do usuário. Desta forma, caso uma atualização através de uma visão V1 gere valores nulos para atributos que não pertencem à visão, estes valores poderão ser posteriormente mostrados em outra visão V2.

3.5 - MÓDULO DE GERAÇÃO

O módulo de geração é aquele que permite ao usuário especificar e gerar suas VEs.

Os esquemas de visões poderão ser constituídos por atributos atômicos projetados das tabelas-base e/ou por atributos estruturados calculados sobre atributos atômicos ou outros estruturados (ou seja, constituindo agregados de grau variável).

Sugere-se que a definição de VEs seja armazenada em uma estrutura de dados específica, o que permitirá a materialização da visão a qualquer instante. Uma possível estrutura a ser

utilizada é discutida no capítulo 4.

Para que o usuário possa criar as visões de seu interesse, faz-se mister que ele conheça a parte do esquema de Banco de Dados necessária à definição dos esquemas de visões (ou seja, nomes de atributos e suas semânticas) e que ele forneça as funções geradoras das visões. A geração de visões comporta dois tipos de mapeamento : de consulta e atualização.

3.5.1 - Mapeamento de Consulta

Um mapeamento de consulta corresponde à resposta a uma consulta feita à interface universal, através de uma visão.

Quando um atributo é simples, as restrições a que ele é submetido são aquelas herdadas do banco de dados. Além disto, a operação de seleção é indicada pelo usuário ao especificar as condições impostas a cada atributo da visão que corresponda a um atributo da tabela-base.

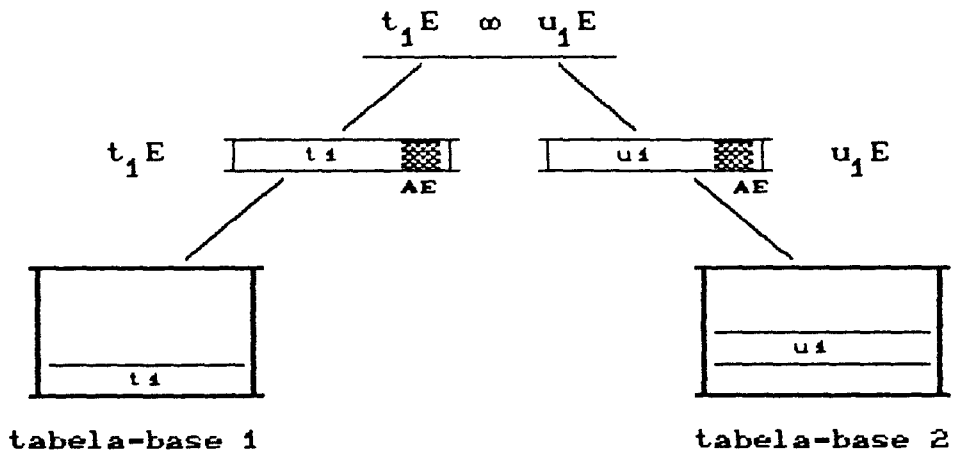
Para a especificação de cada atributo estruturado, o usuário deve indicar a função para o seu cálculo e também fornecer os nomes dos atributos fonte sobre os quais será aplicada a função que produzirá o agregado. Esta forma de especificação torna uniforme a indicação de todos os atributos.

Uma vez definido o esquema de uma VE, ela pode ser materializada e ter fragmentos consultados.

A visualização de uma tupla de uma VE deve ser feita em duas etapas. Na primeira etapa, para cada tabela-base são criadas

tuplas intermediárias contendo valores de atributos estruturados. Na segunda etapa é feita a junção natural destas tuplas intermediárias.

No caso de visões formadas a partir de uma única tabela-base, a segunda etapa não é necessária.



EXEMPLO : Seja R1 = (A, B, C, E) , R2 = (C, D) e V1 = (A, C, B + 1, √D) e considere o seguinte estado do banco de dados.

r1 :

A	B	C	E
1	1	1	1
2	2	2	2

r2 :

C	D
1	4
2	25
7	16

A materialização da visão se faria em duas etapas :

ETAPA 1

r1E :

A	B + 1	C
1	2	1
2	3	2

r2E :

C	√D
1	2
2	5
7	4

ETAPA 2

A	B + 1	C	√D
1	2	1	2
2	3	2	5

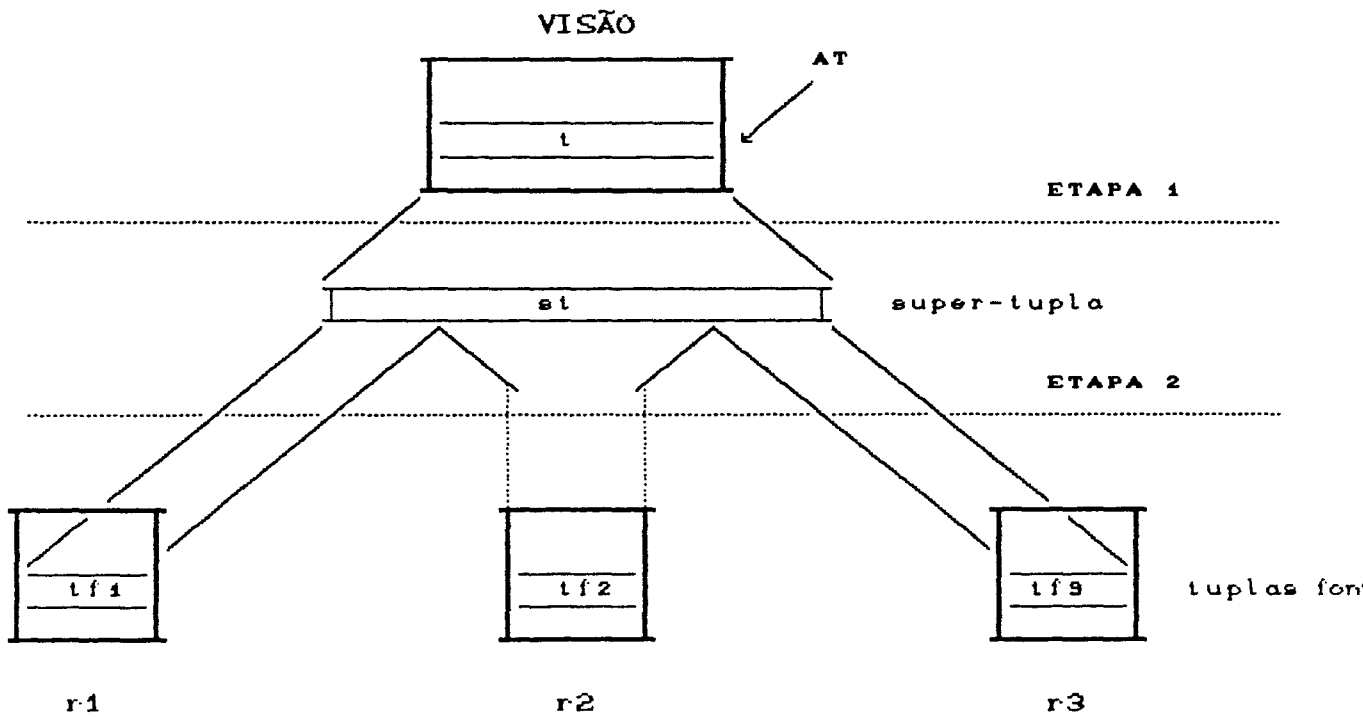
3.5.2 - Mapeamento de Atualização

As VEs aceitam atualizações. Desta forma, é necessário definir os mecanismos de mapeamento utilizados. A interface proposta deixa a cargo do usuário a definição destes mapeamentos. É concebível, assim, que uma mesma visão comporte, por exemplo, vários tipos de inserção de tuplas. Os mapeamentos de atualização precisam ser definidos a dois níveis: mapeamentos das tuplas de uma VE e,independentemente, dos atributos estruturados das tuplas.

A filosofia de VEs estende o conceito de visões como TADs, já que o mapeamento de atualização sobre a VE tem que levar em conta o mapeamento dos seus atributos estruturados. Desta forma, uma

atualização sobre uma VE deve ser considerada como feita em duas etapas. No primeiro estágio são atualizados os atributos estruturados, sendo aplicada a função de mapeamento inverso, criando uma tupla estendida (aqui chamada super-tupla) contendo os valores dos atributos fonte (simples/estruturados) já atualizados. No segundo estágio, a super-tupla é mapeada para as tabelas-base, seguindo os conceitos de mapeamento de atualização de visões. A quebra do mapeamento das operações sobre a visão em duas fases aumentou o nível de complexidade do conhecido "problema de atualização de visões". A figura a seguir esquematiza o processo.

ETAPAS DE MAPEAMENTO



com st = tf1 ∞ tf2 ∞ tf3 e

AT (Visão, t) = AT (U, st) = ‹AT (ri, tfi)›,

onde AT é do tipo inserção de tupla, remoção de tupla ou modificação de tupla e U é a relação universal. A notação AT(E,t) deve ser lida como "atualização do tipo AT sobre tabela de nome E, referente à tupla t".

Note que se a VE contém um atributo estruturado correspondente a um agregado vertical, então se deve considerar um conjunto de super-tuplas. No entanto, para simplificação, será suposto sempre a existência de uma única super-tupla.

Note, também, que o caminho para mapear uma atualização não corresponde ao inverso do caminho de materialização. Neste último, os valores resultado de agregados são determinados antes de se juntar as tuplas. No primeiro, os valores base são determinados e só então as atualizações são propagadas.

EXEMPLO : Sejam $R1 = (\underline{A}, B, C, D)$, $R2 = (\underline{D}, E, F)$ e $V1 = (A, Y, D, W)$ onde $Y = B + C$ e $W = E * F$. Suponha que toda atualização tenha mapeamentos com distribuição uniforme.

A inserção da tupla (4,6,8,30) em $V1$ gera pedido de inserção de super-tupla $A,B,C,D,E,F = (4,3,3,8,\sqrt{30},\sqrt{30})$, transformado em inserção das tupla (4,3,3,8) em $R1$ e $(8,\sqrt{30},\sqrt{30})$ em $R2$.

A especificação do mapeamento de atributos estruturados pode ser feita de duas maneiras: quando da geração da visão ou quando de sua atualização. No primeiro caso, os mapeamentos são

incorporados à definição da visão; no segundo caso, são decididos pelo usuário à medida que processa suas atualizações. Mapeamentos da segunda fase (ou seja, de super-tupla para tabelas-base) são todos do primeiro tipo; mapeamentos de atributo (ou seja, de VEs para a tupla estendida) podem ser dos dois tipos. As características do primeiro tipo de tratamento são descritas nesta subseção, pois dizem respeito ao módulo gerador; as características do segundo tipo são discutidos no módulo de atualização.

O usuário poderá especificar, durante a definição de VEs, o mapeamento de cada atualização sobre a visão para operações sobre as tabelas-base. No caso de VEs serem construídas sobre outras VEs, o usuário deve especificar como serão traduzidas as operações sobre estas últimas, que por sua vez já devem ter incorporadas à sua definição o mapeamento de suas atualizações, e assim por diante, até atingir as relações-base.

a) Mapeamento de Atualização de Super-Tupla

Para simplificar o problema de multiplicidade de traduções, supõe-se que, para cada super-tupla de visão, só existe uma tradução possível para cada tipo de atualização (inserção, modificação e eliminação). Em outras palavras, não existem vários tipos de inserção ou eliminação.

Desta forma, um pedido de inserção de tupla de VE é traduzido para pedido de inserção de super-tupla, na qual os valores dos

atributos fonte de atributos estruturados já estarão definidos. Identicamente, eliminação de tupla de VE é traduzida para eliminação de super-tupla (neste caso, o atributo estruturado só influencia se foi gerado a partir de um agregado vertical). Finalmente modificação de atributo simples é mapeada para modificação do atributo correspondente na tabela-base, e de atributo estruturado em modificação do conjunto de atributos fonte da super-tupla.

O usuário pode optar por adiar a definição da formação da super-tupla (mapeamento de atualização de atributo estruturado) mas não tem liberdade para definir as regras para mapear sua atualização para as tabelas-base (segunda etapa de mapeamento).

As regras propostas para a segunda etapa de mapeamento são as especificadas por Keller [KEL86] e Dayal e Bernstein [DAY82] : O mapeamento para eliminação de uma super-tupla deve ser o da eliminação da tupla que corresponda à "fonte limpa" desta super-tupla. Uma tupla é considerada fonte limpa de uma tupla de visão quando a sua eliminação provoca apenas a eliminação da tupla de visão da qual ela é fonte. A inserção de uma super-tupla corresponde à inserção das tuplas que a formam nas tabelas-base correspondentes, com nulos marcados para os atributos eliminados por projeção. Estas regras só são possíveis de aplicar a partir de um conjunto de hipóteses básicas, além das já discutidas na introdução. Estas hipóteses estão descritas na seção referente ao módulo de consistência. Modificação de atributos da super-tupla são mapeados em modificação dos mesmos atributos nas

tabelas-base.

b) Mapeamento de Atualização de Atributos Estruturados

Ao definir uma VE, o usuário pode definir imediatamente como atualizar seus atributos estruturados. O sistema, desta forma, funciona como se cada visão fosse um módulo similar a TAD como descrito no capítulo 2. O esquema do módulo são os atributos enumerados. As operações permitidas são aquelas cujo mapeamento é definido pelo usuário. Além disto, se uma VE é definida sobre VEs, suas operações são traduzidas como operações sobre as tabelas-base, que por sua vez já tiveram suas definições especificadas, sendo estes mapeamentos cascateados escondidos do usuário.

Os exemplos que se seguem pressupõem que o rol de possíveis agregados e seus mapeamentos já estão embutidos no sistema.

EXEMPLO :

Consideremos o conjunto de visões estendidas onde as atualizações de todos os atributos estruturados têm os respectivos mapeamentos especificados quando da definição da visão. Neste caso, a função geradora de visão, as operações de atualização e as respectivas traduções estão "encapsuladas" em um único bloco, de forma semelhante aos trabalhos de [TUC83] e [TUC85] descritos no capítulo 2. A descrição modular é adaptada destes trabalhos.

Seja R1 [A, B] uma relação base com chave A definida em um módulo M1 e o seguinte estado :

r1 :

A	B
002	10

Adotando-se as seguintes funções como geradoras de agregados e suas inversas como respectivos mapeamentos, a estrutura modular que incorporaria a definição da visão 'V', a operação 'MODIFAG' (modificar agregado) sobre V e seu mapeamento são mostrados a seguir. Supõe-se que as operações EL e INS foram definidas em M1.

Funções que estão disponíveis :

$F1 : Ag = cte1 * fte. \rightarrow F1^{-1} : fte = Ag / cte1$

$F2 : Ag = fte / cte2 \rightarrow F2^{-1} : fte = Ag * cte2$

$F3 : Ag = fte + cte3 \rightarrow F3^{-1} : fte = Ag - cte3$

MÓDULO M estende M1

Esquema : V [A,Ag]

Restrições : ∀ a, ∀ a', V (a,ag) ∧ V (a,ag') → ag = ag'

Operações :

MODIFAG (a,ag) :

Se a ∋ V [a] então V [Ag] ← ag

Função Geradora :

V [A,Ag] : R1 (A) || S onde S : Ag = F1 [S, R1 (B)]

Mapeamento :

MODIFAG (a,ag) : EL (R1,(a,-)) ; INS (R1,(a,$F1^{-1}$(ag)))

Fim-Módulo

A visão V para o exemplo contém a tupla (002, 50). A modificação da tupla (002, 20) é proibida por violação da restrição (mesma chave). A inserção de (005, 10) é mapeada em inserção de (005, 10/5) em R1.

Vale considerar que, em se tratando de visões *virtuais*, o mapeamento é considerado uma operação substituta, definida em termos de R1.

O próximo exemplo mostra o caso em que o usuário define o agregado a partir do grupamento de funções padrão (ou seja, não escolhe nenhuma das funções previamente definidas).

Tomando-se os mesmos esquemas de relação e visão, o módulo **M** seria modificado nas seções de definição da *função geradora* e *mapeamento*.

Função Geradora : R1(A) || S

onde S : Ag = F1(10, F2(F3(R1(B), 50), 100))

Mapeamento :

MODIFAG (a,ag) : EL (R1,(a,-)) ; INS (R1,(a,$F3^{-1}$($F2^{-1}$($F1^{-1}$(ag))))).

Em outras palavras, o usuário especifica a função (10 * ((B + 50) / 100).

Este exemplo não apresenta maiores problemas pois o usuário utilizou funções "default" oferecidas pelo sistema para compor a função que calcula o agregado. Além disto, as funções são tais que a inversa da nova função composta é obtida através da composição das inversas das funções geradoras, na ordem inversa

de sua aplicação. Porém, esta regra de geração de inversa nem sempre pode ser aplicada.

Cada particular função tem suas próprias regras de consistência e estas são verificadas no momento da sua aplicação, como se o agregado fosse calculado por cada uma das funções isoladamente. A consistência, no entanto, deve ser verificada somente após o cálculo do valor final e não considera valores intermediários.

No caso mais geral, a permissão para o usuário criar suas próprias funções geradoras de agregados é bem mais complexa. Além do problema da semântica da nova função (deixada sob inteira responsabilidade do usuário), existe o problema de violação da consistência do Banco de Dados, pois o usuário terá que ditar as regras de consistência válidas para a nova função as quais podem contradizer regras já existentes. O problema adquire um grau de complexidade maior quando se compartilha atributos fonte para servirem de base para o cálculo de mais de um agregado. Este problema é discutido com mais detalhes na seção 3.6.

Idealmente, o usuário poderia criar seus mapeamentos, além daqueles cuja implementação já esteja embutida no sistema, se nenhuma opção do menu atender a suas necessidades. Isto pode ser permitido pelo sistema, desde que esta função seja restrita a determinados casos (por exemplo, combinação linear de algumas das funções do menu padrão).

Ressalte-se que a facilidade acima descrita é praticamente impossível de implementar de forma a garantir a consistência dos

dados, tendo em vista que vários agregados podem ser definidos em cima do mesmo conjunto de parâmetros. É concebível que o sistema tenha gatilhos que mantenham a consistência entre agregados gerados a partir das funções do menu e os definidos sobre os mesmos atributos fonte (como no exemplo anterior). No entanto, é praticamente inviável supor que seja possível manter consistência mútua entre funções arbitrariamente especificadas pelo usuário ao definir suas visões estendidas.

A solução neste último caso seria que o usuário, ao definir uma função não embutida, fosse obrigado a especificar sua interação com os demais agregados do sistema. Isto implicaria que o sistema permitisse adição incremental de rotinas e que, além disto, estas novas rotinas pudessem modificar as demais, o que obviamente desvirtua o propósito da interface, que é o de tornar as interações entre mapeamentos transparentes ao usuário. Desta forma, se o usuário tem necessidade de funções que não são fornecidas pelo sistema, ele deve especificar estas funções ao encarregado de manutenção do sistema, que o modificará para satisfazer as necessidades do usuário.

Estas observações são feitas para ressaltar que, embora a descrição do sistema vise seu uso geral, pressupõe-se sua adequação a classes de aplicações. A cada classe corresponde um conjunto de agregados e possíveis mapeamentos de atualização.

3.6 - MÓDULO DE MANIPULAÇÃO

Uma vez definidas as VEs, o usuário pode executar tanto consultas quanto atualizações. Neste último caso, é ativado o módulo de atualização.

Quando o sistema recebe solicitação de atualização de visões, ele a executa, se possível, efetuando o mapeamento para as tabelas-base.

Como se permite mais de um atributo estruturado em uma mesma VE, é possível que as funções geradoras destes atributos sejam as mesmas mas os mapeamentos de atualização difiram. Por exemplo, uma tupla pode conter dois agregados SOMA, um refente a soma de salários ao longo de um ano e outro referente a soma de descontos. A modificação do primeiro agregado pode ser distribuida uniformemente e a do segundo retratar a modificação de política governamental e se referir a um desconto específico (distribuição não uniforme).

O mapeamento de atualização de atributos estruturados pode ser fixo (predefinido pelo usuário durante a geração) ou variável (definido durante solicitação de atualização).

O primeiro caso corresponde à noção de TAD's e foi discutido na seção anterior. O segundo caso permite ao usuário atividade do tipo SAD, em que várias opções de mapeamentos são tentadas até que se atinja o resultado desejado. Este segundo tratamento é intermediado pelo módulo de atualização.

Se o usuário optar por escolher os mapeamentos quando da

geração da VE, o sistema processa automaticamente o conjunto de operações correspondentes à tradução da atualização sobre a visão. Trata-se, assim, de um tratamento em que a VE encapsula operações cujos mapeamentos pré-fixados fazem parte das opções permitidas sobre a visão.

Caso o usuário não opte por pré-definir os mapeamentos para atributos estruturados durante a geração, a escolha do mapeamento será feita de modo interativo, quando o usuário estiver processando atualizações sobre suas VEs. Esta opção difere conceitualmente do mapeamento fixo por duas razões principais :

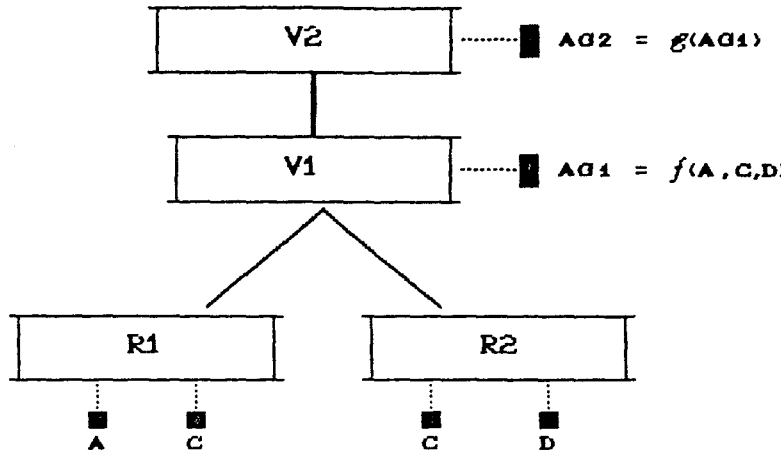
a) Um mesmo tipo de atualização sobre uma mesma VE (por exemplo, inserção de tupla) pode ter mapeamentos diferentes especificados para um mesmo atributo durante uma sessão. Pode-se argumentar que isto torna indefinida a semântica implícita na especificação de tuplas de uma visão. Por exemplo, uma certa tupla contendo o agregado *SOMA* pode ser inserida de forma a distribuir o valor da soma uniformemente pelos atributos fonte. A tupla seguinte pode concentrar o valor da soma em algum atributo fonte e manter os demais valores fonte nulos.
b) Como o mapeamento não é fixado na geração, é permitido ao usuário testar várias opções até atinjir um resultado satisfatório . Neste caso, exige-se do usuário que tenha conhecimento da existência e composição das tabelas-base. A atividade de várias tentativas deve ser acompanhada da

exibição ao usuário do resultado da cada mapeamento sobre estas tabelas. Assim sendo, o sistema tem características SAD:

dispondo de várias iterações de *erro/acerto*, o sistema auxilia o usuário a decidir a opção mais adequada de mapeamento.

Tanto para mapeamento pré-definido como variável, o sistema oferecerá o mesmo leque de opções de mapeamentos. A seguir serãoapresentados exemplos de cascateamento de operações e de interação SAD. Para estes exemplos e as próximas seções é feita a definição de "árvore genealógica de visão". A árvore genealógica de uma VE é a representação gráfica hierárquica que indica as tabelas-base (relações base ou derivadas) tomadas para sua especificação e materialização. Uma árvore genealógica é a grosso modo o grafo correspondente à execução da consulta que materializa uma visão. A árvore genealógica é usada para determinar o nível de um agregado e consequentemente a extensão de propagação de uma atualização até as relações-base.

Seja V2 uma visão cuja árvore genealógica é mostrada a seguir.



onde :

- A, C e D são atributos *atômicos* e *fontes* de AG1.

- AG1 e AG2 são atributos *agregados* com AG1 *fonte* de AG2.

- f e g são *funções geradoras de agregados* AG1 e AG2 respectivamente.

- as inversas de "f" e "g", "f^{-1}" e "g^{-1}", com algumas variações, formam o menu de opções de mapeamentos.

$$f \rightarrow \{ f_1^{-1}, f_2^{-1}, f_9^{-1} \}$$

$$g \rightarrow \{ g_1^{-1}, g_2^{-1} \}$$

O estado inicial das tabelas-base para os atributos fonte dos agregados é mostrado a seguir.

r1 :

A	C
a_1	c_1
:	:
:	:

r2 :

C	D
c_1	d_1
:	:
:	:

v1 :

A	C	AG1
a_1	c_1	$ag1_1$
:	:	:
:	:	:

v2 :

A	C	AG2
a_1	c_1	$ag2_1$
:	:	:
:	:	:

Suponha que o usuário, quando das definições de V1 e V2, haja optado pelos mapeamentos f_1^{-1} e g_1^{-1} .

Suponha-se, então, o pedido de atualização de modificação de $ag2_1$ para $ag2_1^*$. O pedido terá o cascateamento de mapeamentos dados a seguir.

$$\langle\ \mathrm{MOD}\ [\mathrm{AG2}(ag2_1),\ \mathrm{AG2}(ag2_1^*)]\ ;\ f_1^{-1}(g_1^{-1}(ag2_1^*))\ \rangle$$

A execução da atualização provoca o novo estado mostrado abaixo, onde cada nível corresponde à atualização do agregado correspondente.

$$v2 : \frac{A \quad C \quad AG}{a'_1 \quad c'_1 \quad ag2_1^*}$$

x-x-x-x-x **MAPEAMENTO (1° NÍVEL)** x-x-x-x-x

$$v1 : \frac{A \quad C \quad AG1}{a'_1 \quad c'_1 \quad ag1'_1}$$

x-x-x-x-x **MAPEAMENTO (2° NÍVEL)** x-x-x-x-x

$$r1 : \frac{A \quad C}{a'_1 \quad c'_1} \qquad r2 : \frac{C \quad D}{c'_1 \quad d'_1}$$

Depois do sistema processar a operação de atualização requerida e as operações referentes à tradução desta, o usuário poderá pedir para ver os novos valores dos atributos fonte, confirmando-os ou não. Em caso negativo, ele dispõe dos outros tipos de mapeamentos e os utilizará até satisfazer-se com os

valores produzidos.

Baseando-se na árvore genealógica de visão mostrada acima, suponha-se um banco de dados de uma micro-empresa, onde:

- R1 e R2 são tabelas de receita e despesa, respectivamente.

- V1 é uma VE, cujo esquema contém um atributo agregado (AG1), resultante da aplicação da função f = A + C - D; ou seja, cada tupla de V1 espelha o rendimento líquido mensal.

- V2 é uma VE definida sobre V1, cujo esquema é composto pelo atributo agregado AG2 que é calculado pela função "g", com a seguinte lei de formação: g = 0.2 * AG1; ou seja, imposto de 20% sobre o lucro.

Tomando-se como inicial, o estado das tabelas mostrado abaixo, e considerando apenas os atributos estruturados.

r1:

A	C
25	10
8	12

r2:

C	D
10	8
12	2

v1:

AG1
27
18

v2:

AG2
5.4
3.6

Imagine um sistema que manipule este banco de dados, exibindo as visões estendidas aos gerentes interessados e permitindo que estes processem operações de atualização sobre estas visões. Um sistema com essas facilidades caracteriza uma interface SAD, a qual teria incorporada a si os mapeamentos para cada tipo de operação de atualização sobre as VEs (ou seja, mapeamento de super-tupla).

Tomemos a operação de atualização MODIF-AG2 e os mapeamentos

UNICA BIBLIOTECA CENTRAL

MAP1 e MAP2, cujas definições são dadas abaixo:

MODIF-AG2 :

1) Acessar tupla requerida em V2

2) Solicitar novo valor de AG2

3) Repetir

4) Exibir menu com opções de mapeamento

5) Executar opção escolhida

6) Exibir V2 e folhas da árvore genealógica de V2

até (usuário satisfeito) ou (fim de operação)

7) Se (usuário satisfeito)

então gravar novo estado do BD

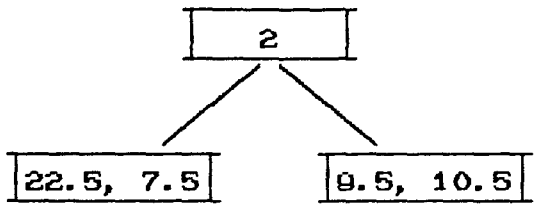
Mapeamentos:

MAP1 : *Distribuir uniformemente o incremento/decremento sobre os atributos fonte do agregado.*

MAP2 : *Concentrar a distribuição sobre alguns dos atributos fonte.*

Suponha-se o seguinte pedido de atualização: modificar valor de AG2 na primeira tupla para 2. Quando exibido um menu com opções de mapeamento, suponha que o usuário optou por MAP1.

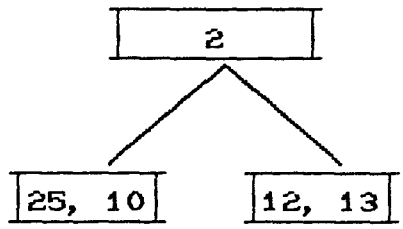
Os resultados produzidos seriam mostrados em uma estrutura secundária para avaliação e possível concretização da operação pelo usuário.



Vale observar que V1 teria, como consequência do efeito-colateral de MODIF-AG2, o valor de sua primeira tupla modificada para 10. Porém, este usuário específico só veria a visão V2, pois não lhe interessa os valores dos atributos fonte intermediários.

A princípio é estranha a semântica de MAP1, pois está se requerendo uma diminuição de receita e aumento de despesa, já que "f" não é uma função monotônica. Este exemplo, porém, não se preocupa com coerência nas semânticas dos mapeamentos e visa somente ilustrar os conceitos discutidos. Note que o exemplo viola várias hipóteses básicas, já que a chave C é parâmetro de agregado e o agregado é multi-relação.

Suponha que o usuário não ficou satisfeito com os valores resultantes da aplicação do mapeamento escolhido. O sistema reexibiria a tela com o menu de opções de mapeamento e o usuário escolheria, por exemplo, a opção MAP2, especificando concentração da distribuição somente sobre os atributos fonte da tabela R2. Depois de executar a operação que traduz a atualização sobre V2, o sistema exibiria, para apreciação do usuário, a estrutura secundária com os novos valores produzidos.



Não há dúvidas que esta opção de mapeamento tem uma semântica mais próxima da realidade. Por exemplo, pode se tratar de uma escola que, ao oferecer bolsas de estudo para alunos carentes, aumentaria suas despesas, reduzindo-se o lucro e consequentemente o imposto a pagar.

3.7 - MÓDULO DE CONSISTÊNCIA

As únicas dependências analisadas são dependências funcionais. Para facilitar a análise supõe-se que as relações base estejam em BCNF. O sistema cuidará também da consistência dos dados a nível de integridade de domínio sempre que o usuário solicitar uma operação de atualização sobre a visão, estendendo a verificação às relações-base armazenadas, quando da aplicação das operações que traduzem a operação de atualização. A garantia de outras restrições de integridade tais como chaves únicas e tipos de dados podem ser deixadas ao SGBD. Desta forma, o sistema deve interceptar as possíveis mensagens de erro a nível de relação-base e transmiti-las aos usuários.

O sistema só modificará fisicamente o Banco de Dados

(equivalente a uma operação de *COMMIT*) quando forem verificadas e acatadas todas as restrições de integridade e após confirmação do usuário.

O sistema deverá ter condições de retroceder (efeito *backtracking*) durante as operações de atualização sobre atributos estruturados, inclusive aqueles com nível de formação de agregado maior do que 1.

O sistema deve verificar, para cada nível da árvore genealógica da visão, as regras de consistência e, se estas forem violadas, deve retornar ao nó com agregado de nível maior.

Se a definição da visão já contiver os mapeamentos pré-definidos (ou seja, definição durante geração) e ocorrendo uma situação de desrespeito às regras de integridade, nada poderá ser feito, a não ser a emissão de uma mensagem de erro para o usuário seguida do abandono daquela operação de atualização. Em outras palavras, trata-se do que [CAS88] chama de bloqueio de propagação. Se o mapeamento, ao contrário, for especificado durante a operação, o sistema pode solicitar novos dados ao usuário, afim de que tome alguma atitude, optando, por exemplo, por outro tipo de mapeamento.

Será, também, da responsabilidade do sistema fazer o controle de atributos atômicos que sejam atributos fonte de mais de um atributo agregado para uma mesma VE. Assim, se ocorrer a situação de haver dois ou mais agregados cujas funções de cálculo sejam aplicadas sobre o mesmo parâmetro (simples ou estruturados), o sistema manterá o usuário informado, como também controlará e

cuidará dos efeitos colaterais surgidos quando da atualização de um destes agregados. Ressalta-se mais uma vez que este tipo de consistência só pode ser realizado quando as funções geradoras de agregados são limitadas e predefinidas pelo sistema, bem como seus mapeamentos inversos.

A seguir é mostrado um exemplo de atualização de visões que compartilham atributos fonte para cálculo de seus atributos estruturados.

Seja R uma relação base, V1 e V2 visões definidas sobre R cujos estados são mostrados abaixo.

Sejam F1 e F2 as funções geradoras de AG1 e AG2 e $F1^{-1}$ e $F2^{-1}$ os respectivos mapeamentos.

F1 : AG1 = B * 10 → $F1^{-1}$: B = AG1 / 10

F2 : AG2 = B / 2 → $F2^{-1}$: B = AG2 * 2

r :

A	B
001	4
002	10

v1 :

A	AG1
001	40
002	100

v2 :

A	AG2
001	2
002	5

Seja o seguinte pedido de modificação de tupla sobre V1 :

V1 (001,40) → V1 (001,20)

O novo estado de V1 é :

v1 :	A	AG1
	001	20
	002	100

Aplicando-se $F1^{-1}$ temos a tradução da operação de atualização sobre V1 para operação sobre R. O novo estado de R é :

r :	A	B
	001	2
	002	10

Porém, como V1 e V2 têm em seus esquemas atributos agregados (AG1 e AG2) cujas funções geradoras são aplicadas sobre o mesmo atributo fonte (B), um pedido de atualização sobre AG1 de V1 afeta AG2 de V2 e vice-versa. Então, após a aplicação da operação sobre R que traduz a operação de atualização sobre a visão, uma nova ativação de V2 mostrará a tupla de V2 com o novo valor.

O novo estado de V2 é :

v2 :	A	AG2
	001	1
	002	5

Vale aqui salientar que este problema é extremamente

complexo, pois é preciso que o sistema garanta a integridade final das relações e das visões. Causa problemas, por exemplo, ter dois agregados horizontais em uma mesma visão, AG1 e AG2, de forma que AG1 seja de 1° nível e AG2 de 2° nível e utilizando AG1 como parte de seus atributos fonte. Sejam G1 e G2 as funções geradoras de AG1 e AG2 e (A1...An) atributos da tabela-base . Seja o esquema de visão (A1,...,Aj,AG1,AG2), onde AG1 = G1(A1 ... Ak) e AG2 = G2(AG1,A1 ... Ah). Uma modificação em AG1, por exemplo, deve ser mapeada para os atributos A1,...,Ak. Por outro lado, esta modificação também altera AG2, tendo em vista que dentre seus atributos fonte estão AG1 e A1 ... Ah. No entanto, dependendo das restrições sobre AG2, é possível que esta modificação force mudanças nos atributos fonte, o que por sua vez pode afetar AG1, criando um ciclo de dependência AG1 ↔ AG2.

Nesta tese pressupõe-se que esta interação não poderá ocorrer. Um primeiro passo para evitar este tipo de problema é restringir a formação de atributos estruturados de uma visão. É possível, ainda assim, que agregados interajam destrutivamente se houver intersecção não nula nos seus conjuntos de atributos fonte. Em outras palavras, a atualização de um agregado AG1 em uma visão exige modificação de AG2 na mesma ou em outra visão. O que se deseja prevenir é a interação destrutiva AG1 ↔ AG2. Algumas das restrições adotadas na seção seguinte visam eliminar problemas neste sentido. Além disto, como se está atualizando visões, o sistema deve tentar evitar problemas clássicos de indefinição semântica destas operações. Seguem-se algumas

restrições para contornar os problemas apontados.

3.8 - RESTRIÇÕES IMPOSTAS

* Cada visão deve ter entre seus atributos ao menos uma chave de cada uma de suas tabelas-base, sendo formada através de junção sem perdas. Isto garante que cada visão tenha uma chave que também é chave de uma das relações do banco de dados. Com isto, evita-se uma série de problemas em mapeamentos de inserções e alterações e se garante que uma eliminação tenha fonte limpa [DAY82]. A existência ou não de atributos estruturados na visão não viola esta propriedade: os únicos atributos deste tipo permitidos são agregados, monovalorados e oriundos da aplicação de funções sobre atributos fonte, que por sua vez dependem funcionalmente das chaves das tabelas-base, que devem estar em BCNF. Caso se permitisse também atributos não convencionais multivalorados (resultado, por exemplo, da aplicação de procedimentos de consulta sobre relações base [STO87]) este tipo de raciocínio não seria mais válido.

* O sistema proposto não pretende descobrir, quando da definição do usuário, se os mapeamentos são inviáveis (por exemplo, se o mapeamento de AG1 conflita com o de AG2). Tal responsabilidade cabe ao usuário; o sistema apenas assinala a ocorrência de interferência de agregados quando da execução das

operações. Ou seja, quando o usuário requerer uma operação de atualização sobre a visão que modifique o valor do agregado, o outro agregado que é calculado sobre os mesmos atributos fonte terá o seu valor também alterado, sendo o usuário notificado. Se a segunda alteração violar as restrições de domínio deste segundo agregado, a atualização será interrompida e não será efetivada.

Em suma, o sistema realiza propagação de atualização de uma visão sobre outra (ou a mesma visão), mas interrompe a propagação assim que encontra uma violação de domínio. Nenhum outro tipo de restrição é levado em consideração : as dependências funcionais são herdadas das relações base e sua manutenção é garantida pelo fato de que cada visão é obrigada a ter ao menos uma chave de cada uma de suas tabelas-base. É igualmente ignorada a possibilidade da existência de outras dependências como MVDs ou IDs.

* Atributos pertencentes às chaves não podem ser atributos fonte para geração de atributos estruturados (evitando, assim, que dependências de chave possam ser violadas).

* Tendo em vista as restrições acima, os agregados verticais não podem fazer parte de uma visão estendida, caso contrário esta não poderia ter chaves identificando tuplas.

3.9 - RESUMO

Este capítulo discutiu a arquitetura geral de um sistema que suporta definição de tipos não convencionais a partir de visões. O próximo capítulo apresenta um protótipo desenvolvido que suporta uma classe específica de aplicação.

CAPÍTULO 4

SISTEMA IMPLEMENTADO

Este capítulo descreve a implementação de um protótipo segundo o sistema proposto no capítulo 3. O protótipo foi desenvolvido para suportar aplicações que requeiram manipulação de algumas funções estatísticas, tendo sido implementadas as operações *soma*, *média*, *máximo* e *mínimo*.

A implementação deste sistema teve como objetivo primordial validar a especificação. Além disto, buscou verificar, através do desenvolvimento de uma aplicação, a validade dos conceitos analisados no capítulo 3 e que são centrais à proposta da tese.

Estes conceitos dizem respeito ao uso de visões estendidas como mecanismo para incorporar tipos de dados não convencionais a um banco de dados relacional. Como se verá pela leitura deste capítulo, o sistema desenvolvido permite a especificação de visões que incorporem módulos (similares a TADs) ou que suportem aplicações do tipo SAD.

Em paralelo a estas noções, o sistema é um exemplo de suporte a atualização de visões que contenham atributos agregados.

As visões suportadas por esta implementação são voláteis. Em outras palavras, se armazenadas, passam a fazer parte do conjunto de arquivos base e podem ser usadas como instantâneos,

mas seus atributos agregados não são mais passíveis de atualização, a não ser como reflexo de mudanças nas tabelas-base ("refresh" do instantâneo). Vale dizer que VEs armazenadas não podem ser usadas pelo módulo de atualização.

4.1 - CARACTERÍSTICAS DA IMPLEMENTAÇÃO

O sistema foi implementado em PASCAL, podendo ser executado em qualquer equipamento do tipo PC. Ao invés de um banco de dados, o sistema está implementado sobre arquivos (tabelas) acessados através de índices em árvore B . Como um dos objetivos principais era verificar a utilização de visões como suporte a tipos de dados e o aumento de flexibilidade que isto acarreta para os usuários, não houve preocupação em utilizar um SGBD real.

Para uso de um SGBD real, a geração de visões do protótipo deveria ser modificada, necessitando ser efetuada em dois passos. O primeiro passo corresponderia à geração padrão da visão do SGBD hospedeiro; o segundo passo exigiria que a esta visão fossem adicionados os atributos não padrão, artificialmente, através da interface.

O protótipo implementado considera apenas visões formadas por projeção e/ou seleção, sem se preocupar com junção de relações. A razão desta simplificação é que a tese se concentra na primeira fase de mapeamento (geração de super-tupla), correspondendo a segunda fase ao problema já estabelecido de atualização de

visões. O protótipo tem, assim, por objetivo verificar a viabilidade da implementação de atributos estruturados.

Outra opção de geração de visões estendidas seria a que não aproveitasse qualquer mecanismo de visões do SGBD. Nesta segunda solução, bastaria substituir os comandos de E/S do protótipo por comandos LMD correspondente. Se esta segunda solução apresenta a vantagem de permitir a utilização integral do protótipo, por outro lado dificulta o controle do SGBD sobre a manipulação das visões, uma vez que o SGBD não reconhece as VEs como visões.

4.2 - ESTRUTURAS DE DADOS

O protótipo implementado está baseado em duas classes de estruturas de dados : principal e secundária. As estruturas principais de dados são compostas dos esquemas de relação e VEs, dados e índices. Os esquemas de relações, dados e índices se encontram em arquivos (RELBASEi, VISÃOj, RELINDi e VISINDj). Os esquemas de VEs são gerados em memória e, ao término de uma sessão, armazenados em arquivos (ESQ_VISÃO).

As estruturas de dados secundárias são compostas, basicamente, por vetores. Um vetor é utilizado como "buffer" para guardar os valores produzidos pelo mapeamento de uma atualização, já que estes só serão gravados nos arquivos de dados após autorização do usuário. "Vet_Dif" é um vetor diferencial que contém os atributos das tabelas-base que não foram selecionados

pelo usuário e que serão valorados com "nulo" pelo sistema, quando da composição da super-tupla (primeira etapa de mapeamento de atualização).

O Anexo 2 mostra graficamente os registros de cada um dos arquivos das estruturas de dados principais, segundo a metodologia de [COW80]. A estrutura secundária não foi mostrada pois é óbvia a sua representação.

4.2.1 - Arquivos Utilizados

Para facilitar a leitura, esta seção descreve os arquivos e as estruturas manipulados pelo sistema do ponto de vista de uma única relação e visão estendida. Ao final do capítulo, são feitas as generalizações necessárias para estender as noções para multi relação e várias visões.

Para o funcionamento do sistema, é preciso que os arquivos a serem utilizados estejam corretamente formatados (o que, no caso das relações-base, corresponde a grosso modo à carga do banco de dados). Desta forma, os dados para as relações-base estão inicialmente armazenados em arquivos do tipo TEXT de PASCAL.

Estes arquivos devem ser processados de forma a "criar" relações-base. Esta fase do processamento só é necessária quando o usuário deseja acrescentar novas relações ao sistema, ou modificar relações existentes. Para tal, o protótipo provê a função "Gerar Banco de Dados". Para cada arquivo tipo TEXT de uma relação-base são gerados dois arquivos correspondentes a uma

relação : arquivo do esquema e arquivo de dados.

Os registros do arquivo TEXT são armazenados sequencialmente, sendo cada registro, a partir do segundo registro, interpretado como uma tupla da relação, onde os valores dos atributos estão dispostos sequencialmente. O primeiro registro descreve o esquema de relação. A informação constante no primeiro registro é processada e armazenada em um arquivo particular (Esq_Relação), com número de atributos seguido de seus nomes. A relação propriamente dita é transformada num arquivo RELBASE.

Tentou-se com esta arquitetura flexibilizar o sistema no tocante à criação do banco de dados. A figura abaixo mostra a estrutura dos arquivos TEXT que dão origem a relações.

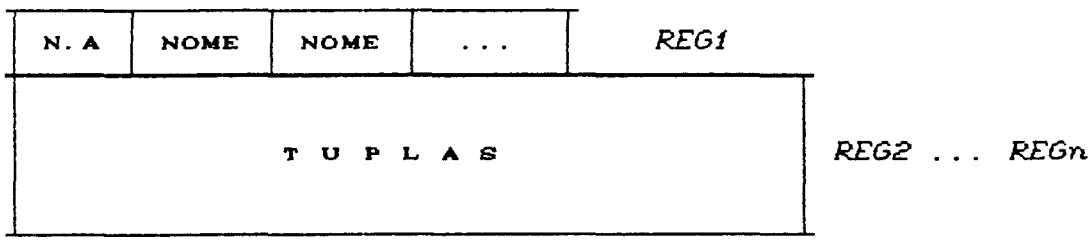


FIG1

Os arquivos "Esq_Relação" e "Esq_Visão" contêm, respectivamente, a descrição dos esquemas de relação-base e VE. Estes arquivos são arquivos PASCAL do tipo "FILE OF ...".

O esquema de uma visão é formatado interativamente na rotina de geração de visão, através de especificação do usuário. Ao final de uma sessão, é armazenado no arquivo correspondente, para ser acessado em sessões posteriores.

As figuras FIG2 e FIG3 mostram, respectivamente, os registros dos arquivos Esq_Relação e Esq_Visão.

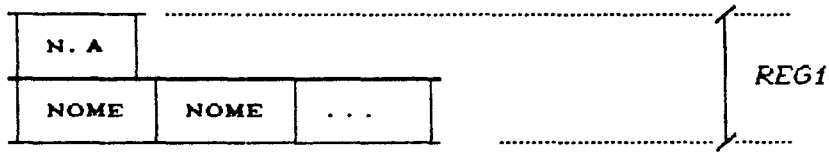


FIG2

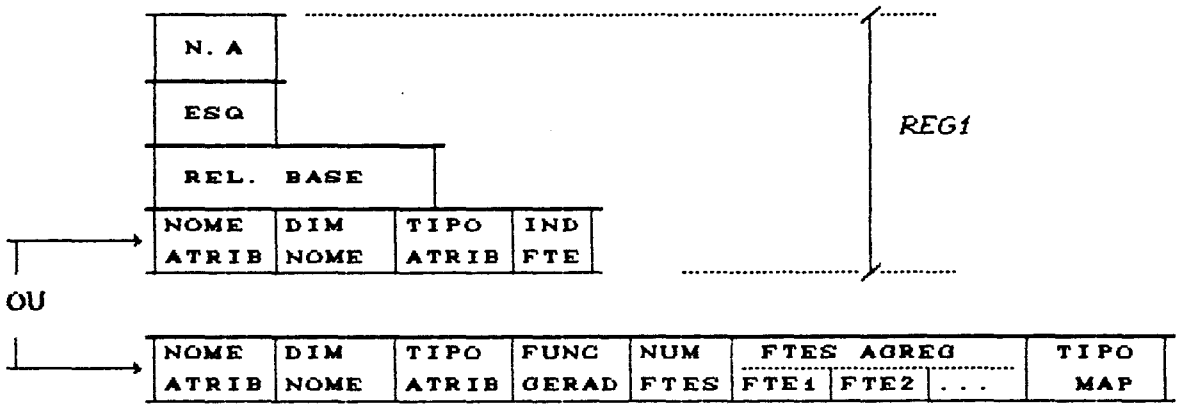


FIG3

FIG3 é um registro de tamanho variável contendo uma parte fixa e outra variável. A parte fixa contém exclusivamente informações a respeito do esquema sem se preocupar com os atributos, isto é, o número de atributos que comporão o esquema, o nome do esquema e a relação-base sobre a qual a visão é definida. A parte variante do registro pode se referir a atributos simples ou estruturados. O campo "TIPO_ATRIB" funciona como seletor de atributo: "1" para atributo simples e "2" se

tratar-se de atributo estruturado. Caso o atributo seja do tipo simples, o campo "IND_FTE" conterá um apontador para o atributo da relação-base a ser projetado. Se tratar-se de atributo estruturado contém, entre outros, a indicação da função geradora de agregado (FUNC_AGREG) e as posições, dentro do esquema de relação-base, dos atributos fonte para o cálculo do agregado (FTE_AGREG).

"Relbase" é um arquivo sequencial de dados corresponde à relação propriamente dita. Como se pode ver no Anexo 2, cada tupla da relação-base é, na verdade, um registro composto dos campos "Status", "Obs", "Cod" (chave primária) e "Outros_Campo". Este último é um vetor contendo os valores dos atributos do esquema de relação.

O campo "Status" é de uso interno do gerenciador e lhe permite controlar as eliminações de registros, já que elas são do tipo eliminações lógicas. O campo "Obs" indica qual o tipo da operação que atualizou por último cada tupla da relação-base. Como o sistema suporta atualização através de visões, a relação-base pode ser formada de tuplas já existentes, tuplas inseridas/modificadas através de alguma operação de atualização sobre a visão e tuplas eliminadas via visão. A figura FIG4 esquematiza este registro.

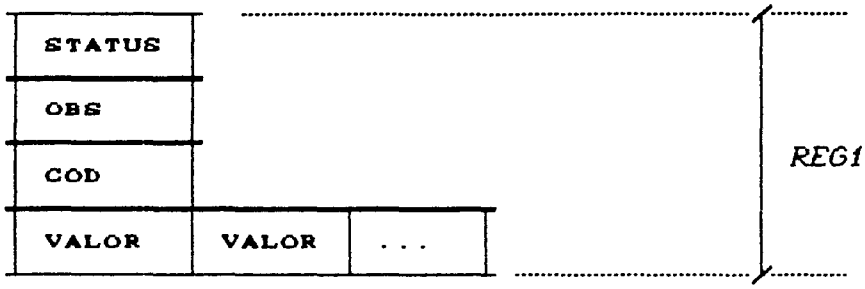


FIG4

Assim como os dados de uma relação-base são armazenados sob a forma do arquivo RELBASE, visões podem ter sua materialização simulada através do arquivo "Visão". Este é também um arquivo sequencial de dados, acessado a partir de índices em árvore B. Para materialização da visão, a rotina responsável primeiramente lê o arquivo "Esq_Visão" afim de conhecer o esquema de visão e sua função geradora e a seguir lê a relação-base correspondente formatando as tuplas e as gravando em "Visão". FIG5 mostra a estrutura deste registro.

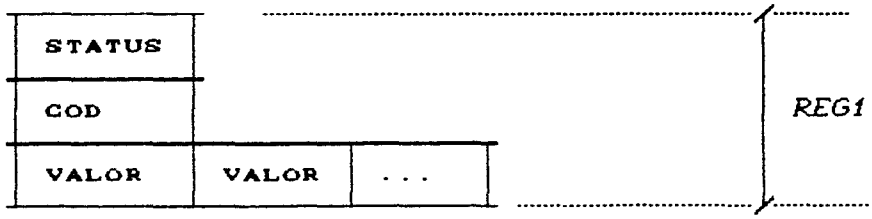


FIG5

Da mesma forma que em "Relbase", o campo "Status" serve para marcar os registros (tuplas) eliminadas. O campo "Cod" é a

projeção da chave primária de "Relbase". O último campo é um vetor que armazena os valores dos atributos do esquema de visão. Cada registro corresponde a uma tupla da visão.

Associados aos arquivos de dados "Relbase" e "Visão" existem os arquivos "Relind" e "Visind", respectivamente, que servem de estrutura de acesso. O método de indexação utilizado pelo subsistema de acesso do gerenciador é a árvore B . Os registros podem ser indexados por uma ou mais chaves e há a opção de ter, ou não, chaves duplicadas. No caso do sistema implementado, adotou-se chaves únicas.

A unidade fundamental da árvore B adotada pelo protótipo é o *ITEM*. Um item representa a conexão entre a chave e seu registro físico de dados. Um item é composto por uma chave (para processar busca na árvore B), uma referência de dados (que aponta para o local no arquivo de dados onde o registro associado com aquela chave se encontra) e uma referência de página (aponta para uma página que contém as chaves maiores do que a chave do item corrente).

Um registro do arquivo de índices é chamado de PÁGINA. Uma página é um registro contendo três estruturas : um vetor de itens, uma referência de página extra e um contador. O contador serve para indicar quantos itens existem na página, isto é, aponta para o último item válido do "vetor de itens". A referência de página extra é usada para armazenar chaves cujos valores são menores do que qualquer chave da página corrente.

Para qualquer consulta ou atualização sobre visões ou

relações (correspondendo a atualização nos arquivos correspondentes), o sistema antes faz uma busca nos respectivos arquivos de índices e ao encontrar a chave procurada obtém a posição física do registro (tupla) no arquivo de dados (relação-base/visão).

A tabela abaixo resume os arquivos utilizados.

TEXT - dados para carga de relações

RELBASE, VISÃO - arquivos que simulam relações e visões materializadas

ESQ_RELAÇÃO, ESQ_VISÃO - arquivos com esquemas de relação e visão

RELIND, VISIND - arquivos para indexar RELBASE e VISÃO

4.3 - RESTRIÇÕES ADOTADAS NA IMPLEMENTAÇÃO

Tendo em vista os problemas relativos à manutenção de integridade e considerando que se está testando um protótipo, foram adotadas as seguintes restrições:

a) O usuário só pode materializar uma única visão a cada sessão e esta visão será resultante da aplicação de projeção precedida opcionalmente por seleção sobre uma relação-base armazenada.

b) O conjunto de atributos que servirão de fonte para a especificação do esquema de visão é formado por todos os atributos da relação-base, exceto o atributo "OBS", já que

este é um atributo artificialmente criado para auxiliar o administrador do banco de dados.

c) Limitou-se a geração de visões àquelas com agregados de 1° grau.

d) Pressupõe-se que os agregados em uma mesma visão não podem interagir interativamente de forma a anular resultados de atualização.

Para estender a especificação do sistema afim de suportar manipulação de várias relações - ou seja, a visão ser definida sobre mais de uma relação-base - ter-se-ia que modificar a estrutura de dados que representa o esquema de visão (FIG3) nos seguintes pontos :

a) O nome de cada relação-base acompanhará o nome do atributo no esquema de visão.

b) Se um determinado atributo existir em mais de uma relação (junção sem perdas), então os nomes destas (duas) relações acompanharão o nome do atributo no esquema de visão.

O protótipo pode ser otimizado para permitir que VEs com árvore genealógica maior do que 1 possam ser materializadas, desde que os seus atributos estruturados tenham nível igual a 1.

O Anexo 3 descreve passo a passo os módulos ativados durante uma sessão.

4.4 - EXEMPLO

- Suponha a seguinte relação com esquema <A, B, C, D, E> e estado dado a seguir.

R1 :	A	B	C	D	E
	01	2.0	4.0	6.0	8.0
	02	10.0	5.0	0.0	3.0
	03	8.0	2.0	6.0	7.0

Suponha uma visão com esquema <A, X, Y>, onde X = (B+C)/2 e Y = E. Sua materialização gera o seguinte estado.

V1 :	A	X	Y
	01	3.0	8.0
	02	7.5	3.0
	03	5.0	7.0

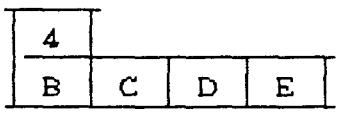
As estruturas internas geradas são as seguintes.

ARQUIVO CONTENDO DADOS PARA GERAÇÃO DA RELAÇÃO :

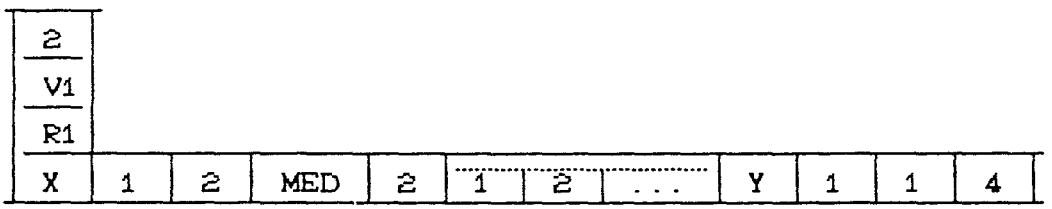
4	B	C	D	E		
ORIG		01	02.0	04.0	06.0	08.0
ORIG		02	10.0	05.0	00.0	03.0
ORIG		03	08.0	02.0	06.0	07.0

Note que a primeira linha do arquivo só contém os atributos passíveis de operações. Assim sendo, "A" não foi citado por tratar-se de chave primária. O status "ORIG" indica que a tupla não foi, ainda, atualizada via visão.

ESQ_RELAÇÃO :



ESQ_VISÃO :



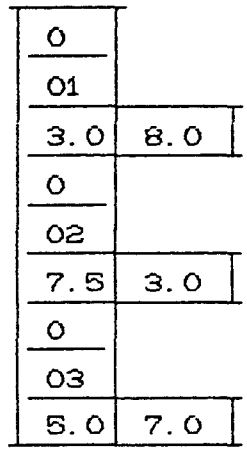
Na parte fixa do registro o valor "2" indica o número de atributos do esquema de visão, "V1" é o nome do esquema e "R1" a relação-base. A parte variante é uma estrutura do tipo "array of records". O valor "X" indica o nome do primeiro atributo, o valor "1" indica que seu nome ocupa 1 byte, o valor "2" especifica o seu tipo (atributo estruturado), "MED" indica que a função geradora de agregado é "Média Aritmética" e o outro valor "2" diz o número de parâmetros que serão usados no cálculo do agregado,

que por sua vez têm as suas posições no esquema de relação-base indicadas no vetor com valores que se segue. Os próximos campos descrevem o segundo atributo, cujo nome é "Y". O valor "1" indica o comprimento do seu nome, o valor "1" seguinte especifica o seu tipo (atributo simples) e o valor "4" indica a posição do atributo fonte no esquema de relação-base.

RELBASE :

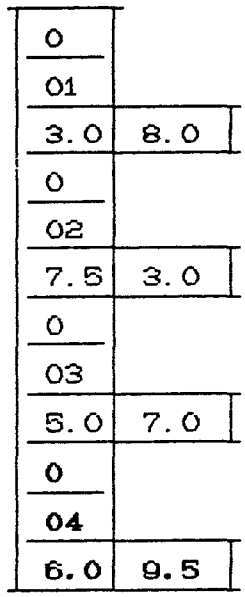
0 ORIG 01			
2.0	4.0	6.0	8.0
0 ORIG 02			
10.0	5.0	0.0	3.0
0 ORIG 03			
8.0	2.0	6.0	7.0

VISÃO :



Uma atualização do tipo *INS[V1(04,6.0, 9.5)]*, com mapeamento do tipo *distribuição uniforme* provocaria as seguintes alterações no conteúdo das estruturas, onde 04 é a identidade da nova tupla.

VISÃO :



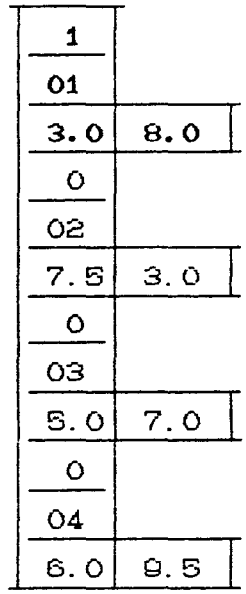
RELBASE :

O ORIG 01			
2.0	4.0	6.0	8.0
O ORIG 02			
10.0	5.0	0.0	3.0
O ORIG 03			
8.0	2.0	6.0	7.0
O I V1 04			
6.0	6.0	NULO	9.5

Vale observar que o valor "I V1" indica que a tupla foi inserida na relação-base através da visão cujo esquema tem nome "V1".

Considere, agora, a atualização *ELI [V1(01)]*, ou seja, eliminar a tupla de V1 identificada pela chave 01. As estruturas internas sofreriam as seguintes alterações.

VISÃO :



Onde o valor "1" indica eliminação lógica da tupla. As rotinas de exibição de visão e relação filtram estas tuplas, só exibindo as que tiverem valor de "status" igual a zero. Os registros (tuplas) com valor de "status" igual a "1" podem ser reutilizados pelo gerenciador.

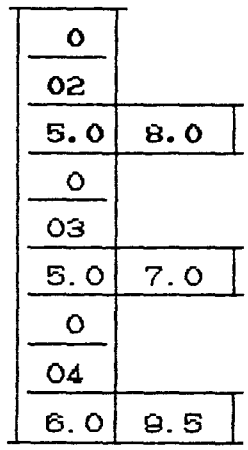
RELBASE :

1 E V1 01			
2.0	4.0	6.0	8.0
0 ORIG 02			
10.0	5.0	0.0	3.0
0 ORIG 03			
8.0	2.0	6.0	7.0
0 I V1 04			
6.0	6.0	NULO	9.5

Enquanto o espaço do registro eliminado não for reutilizado, o valor "1" indica que trata-se de uma tupla eliminada e o valor "E V1" mostra, para o Administrador do Banco de Dados, a visão que a eliminou.

Uma atualização de tipo *MOD (V1 [02, (5.0,8.0)])*, com mapeamento do tipo *distribuição uniforme* provocaria as seguintes alterações no conteúdo das estruturas, onde "02" é a chave da tupla a modificar.

VISÃO :



RELBASE :

0 M V1 02			
5.0	5.0	0.0	8.0
0 ORIG 03			
8.0	2.0	6.0	7.0
0 I V1 04			
6.0	6.0	NULO	9.5

O valor "**M V1**" indica, para o Administrador do Banco de Dados, que a tupla foi modificada através da visão "V1".

CAPÍTULO 5

CONCLUSÕES E EXTENSÕES

5.1 - CONCLUSÕES

A tese apresentou uma arquitetura para estender bancos de dados relacionais afim de permitir ao usuário a definição de novos tipos de dados através de visões. Parte da arquitetura proposta foi validada através da implementação de um sistema que permite a definição de novos atributos para uma família de funções de formação.

Foram analisados determinados tipos de mapeamentos de atualização em visões que contêm campos agregados, que é um tipo de transação não permitido na literatura. O mecanismo adotado no tratamento do "problema de atualização de visões" foi conceber a visão como um Tipo Abstrato de Dados. Usando o conceito de módulos, que somente permite o acesso às visões através das operações definidas nos mesmos,os problemas de inconsistência e efeitos colaterais indesejáveis são minorados. No caso da visão conter um agregado, nada impede ao usuário examinar o efeito das atualizações sobre os valores dos parâmetros (que seriam atributos de outros esquemas do módulo, correspondentes às tabelas-base).

A discussão desta noção de "visões estendidas", em cujos

esquemas se definem novos tipos de dados, foi estendida para abordar o "problema de atualização de visões", sendo as traduções das operações sobre VEs para operações sobre as tabelas-base feitas em duas etapas.

Deve se salientar a aplicação do trabalho em casos específicos, como, por exemplo, proporcionando uma interface que apresenta cenários "what if" para suporte de Sistemas de Apoio à Decisão (SADs). Em estudos de planejamento, é interessante saber para se produzir um determinado valor, quais os valores que os campos base deverão assumir. No sistema apresentado, basta mudar o valor do agregado, forçando a atualização correspondente dos parâmetros, segundo o mapeamento escolhido.

Apresentou-se, além da análise geral do problema, um conjunto de mapeamentos sugeridos para tipos específicos de agregados horizontais e verticais (SOMA, MÉDIA, MÁXIMO e MÍNIMO). Agregados verticais já fazem parte da especificação de alguns sistemas ([OSB86], [STO87]), porém o tratamento de agregados horizontais não costuma ser discutido. Saliente-se, ainda, que esta tese é um dos poucos trabalhos encontrados na literatura que analisa os problemas surgidos quando permite-se atualização de atributos agregados via visão e propõe mapeamentos para as atualizações. Estes mapeamentos foram implementados em uma interface conversacional em PASCAL. Utilizou-se um sistema gerenciador de arquivos para simular os acessos a um SGBD convencional. Optou-se por implementar estes agregados porque são aqueles que, junto com COUNT, se encontram disponíveis para especificações de visões em

grande número dos SGBDs disponíveis no mercado.

Uma das utilidades de visões é restringir o acesso de usuários não credenciados a partes do banco de dados. Tomando-se a árvore genealógica de uma visão "V", pode-se afirmar que quanto mais alta for essa árvore mais desprestigiado será o usuário de "V", no sentido que ele verá menos atributos do esquema de banco de dados do que aqueles usuários das visões definidas abaixo de "V" na árvore genealógica.

Utilizando-se, porém, os conceitos de VEs, o cenário descrito acima passa a ter outra interpretação : Sendo "V" uma VE, o usuário de "V" terá mais privilégios do que os usuários das VEs que foram utilizadas para a definição de "V", pois as atualizações processadas sobre os atributos estruturados de "V" irão afetar as VEs que se encontram nos níveis abaixo de "V". A árvore genealógica de VE mostra então a hierarquia que confere maior ou menor poder de decisão a um usuário.

Este cascateamento de atualizações sobre os vários níveis de uma árvore genealógica que se propagam até atingir suas folhas (relações-base) introduz nos bancos de dados relacionais características de planilhas de dados eletrônicas.

Uma planilha eletrônica é um programa que recebe como entrada dados numéricos e os armazena dentro de células, que são áreas formadas pela interseção das linhas horizontais e colunas da planilha. Além de números, as células podem conter textos e fórmulas. Uma fórmula pode se referir a várias células.

O usuário pode especificar relacionamentos entre linhas ou

colunas. Se o valor incluso numa célula ou um relacionamento entre linhas ou colunas mudar, modificar-se-ão outras partes da planilha e as alterações serão exibidas ao usuário.

Assim como SADs, a utilização de planilhas eletrônicas aumenta a produtividade das atividades desenvolvidas em um escritório pois propicia uma grande flexibilidade para a construção de modelos financeiros e matemáticos.

5.2 - EXTENSÕES

Muitas das extensões se referem às hipóteses que determinam regras de formação de atributos estruturados, tanto do ponto de vista teórico quanto de implementação. As extensões poderiam ser :

(1) Considerar a utilização de visões e agregados "multirelação". O termo multirelação refere-se ao definido em [STO87] onde as tuplas da visão são compostas por valores de atributos projetados de mais de uma relação-base e o agregado é calculado sobre atributos de mais de uma relação.

(2) Oferecer uma interface para cadastramento de novas funções e definições dos respectivos mapeamentos, que seriam adicionados ao dicionário de dados do sistema de banco de dados. Todavia, a corretude sintática e a semântica dessas novas funções ficariam sob responsabilidade do usuário (para ambiente mono-usuário) e do administrador do banco de dados (para ambiente multi-usuário).

(3) Liberalizar os atributos das tabelas-base para que pudessem ser parâmetros de distintas funções geradoras de agregados, ou em outras palavras, permitir que haja interseção de conjuntos de atributos fonte para cálculo de agregados.

(4) Considerar, na implementação, a definição de agregados com grau m (m > 1) e a capacidade de propagar as alterações para os agregados de grau n (n < m).

(5) Permitir o suporte a visões materializadas. Para isso, seria preciso definir uma política eficiente de manutenção de visões materializadas para permitir que as modificações nas relações-base fossem propagadas para as visões materializadas. Quando o usuário não precisasse mais de uma específica visão materiliazada, esta seria eliminada do banco de dados. Poder-se-ia, opcionalmente, eliminar a função geradora da visão, as funções geradoras de agregados para esta visão e os mapeamentos.

(7) Permitir gravação de um "LOG", afim de que todas as transações processadas no banco de dados fiquem catalogadas. Desta forma, adicionar-se-á a semântica necessária ao banco de dados para que o Administrador do Banco de Dados possa conhecer a estória de cada tupla da relação-base.

Continua sem sentido definir, numa mesma visão, agregados horizontais e verticais, já que agregados horizontais correspondem a cálculos sobre fragmentos horizontais, tendo como resultado uma relação com o número de tuplas igual ao de

fragmentos utilizados; enquanto que agregados verticais são calculados sobre um único atributo do esquema de relação, usando-se toda a relação e produz, como resultado, uma relação derivada com uma única tupla.

Finalmente, é necessário fazer um estudo mais aprofundado de mapeamentos alternativos da atualização em duas etapas, bem como identificar famílias de agregados utilizados por tipo de aplicação. Um exemplo encontrado frequentemente na literatura de orientação a objetos é o de manipulação de formas geométricas. Seria interessante explorar este tipo de aplicação utilizando VEs, onde tuplas da relação-base conteriam coordenadas de pontos num espaço (R2 ou R3) e as funções de agregados forneceriam, por exemplo, áreas e volumes.

BIBLIOGRAFIA

[ALB85] - Albano,A., Cardelli,L. & Osini,R., "GALILEO - A Strongly Typed, Interactive, Conceptual Language", ACM TODS, Vol.10, No. 2, 1985, 230-260.

[BAN81] - Bancilhon,F. & Spyratos,N., "Update Semantics and Relational Views", ACM TODS, Vol. 6, No. 4, 1981.

[BAN88] - Bancilhon,F., "Object-Oriented Database Systems", Relatório Técnico Altair, Inria, 1988, 16-18.

[BAN89] - Bancilhon,F., "Query Languages for Object-Oriented Database Systems : Analysis and a Proposal", Anais 4º SBBD, 1989, 22-37.

[BIS86] - Bishop,J.M., "Data Abstraction in Programming Languages", Addison Wesley, Internatinal Computer Science Series, 1986.

[BLA85] - Blakeley,J.A., Larson,P.A. & Tompa,F.W., "Efficiently Updating Materialized Views", Univ. Waterloo,1985.

[BLA87] - Blakeley,J.A., "Updating Materialized Database Views". Rel.Tec. Univ.Waterloo,CS87-32,1987.

[BLAC83] - Blacwell,P.R., Jajodia,S.R. & NG,P.A., "A View of Database Management Systems as Abstract Data Types", Entity-Relationship Software Engineering, 1983,661-668.

[BRO88] - Brosda,V. & Vossen,G., "Update and Retrieval in a Relational Database Through a Universal Schema Interface", ACM TODS, Vol.13, No.4, 1988, 449-487.

[BUC84] - Buchmann,A.P. & Batory,D.S., "Molecular Objects, Abstract Data Types and data Models : A Framework", Anais X VLDB, 1984, 172-184.

[BUL87] - von Bultzingloewen,G., "Translating and Optimizing SQL Queries Having Aggregates", Anais XIII VLDB, 1987, 235-244.

[CAS88] - Casanova,M.A.,Furtado,A.L. & Tucherman,L., "A Monitor Enforcing Referential Integrity", Anais 3º SBBD, 1988, 1-16.

[COW80] - Cowan,D.D., Graham,J.W., Welch,J.W, & Lucena,C.J.P., "A Data-Oriented Approach to Program Construction", Software Practice & Experience, Vol. 10, 1980, 355-371.

[DAT83] - Date,C.J., "An Introduction to Database Systems", Vol 2, Addison Wesley, 1983.

[DAT84] - Date,C.J., "Introdução a Sistemas de Banco de Dados", Vol 1, Ed. Campus, 4º Edição.

[DAT86] - Date,C.J., "Updating Views", in Relational Database Select Writings, Addison Wesley, 1986.

[DAT88] - Date,C.J., "Defining Data Types in a Database Language", SIGMOD Record, Vol. 17 No. 2, 1988, 53-76.

[DAY82] - Dayal,U. & Bernstein,P., "On the Correct Translation of Update Operations on Relational Views", ACM TODS, Vol. 8, No. 3, 1982, 381-416.

[DAY87] - Dayal,U., "Of Nests and Trees : A Unified Approach to Processing Queries that Contain Nested Subqueries, Aggregates and Quantifiers", Anais XIII VLDB, 1987, 197-208.

[DEL89] - Delgado,A.L., Magalhães,L.P., Ricarte,I.L.M., Ruschel,R.C. & Olguin,C.J.M., "Implementação de um Banco de Dados não Convencional", 4º SBBD, 1989, 77-89.

[DIT86] - Dittrich,K.R., "Object-Oriented Database Systems - A workshop Report", Anais V ER Conference, 1986.

[DIT87] - Dittrich,K.R. & Gotthard,W. & Lockemann,P., "DAMOKLES - The Database System for the UNIBASE Software Engineering Environment", Database Engineering, Mar-1987, 37-47.

[FRE86] - Freytag,J.C., Translating Aggregate Queries into Interative Programs", Almaden Research Center IBM, 1986.

[FUR79] - Furtado,L.A., Sevick,K.C. & Santos,C.S., "Permitting Updates Through Views of Databases", Inform. Systems, Vol 4, 269-283.

[FUR85a] - Furtado,A.L. & Casanova,M.A., "Updating Relational Views", in Query Processing in Database Systems, Springer-Verlag, 1985.

[FUR85b] - Furtado,A.L., & Santos,C.S., "Organização de Banco de Dados", Editora Campus, 5º Edição, 1985.

[GHO86] - Ghosh,S.P., "Statistical Relational Tables for Statistical Database Management", IEEE Trans. Soft. Eng., SE-12(12), 1986, 1106-1116.

[GHO87] - Ghosh,S.P., "Numerical Operations on Relational Database", IBM Almaden Research Lab., 1987.

[KEL85] - Keller,A.M., "Algorithms for Translating View Updates to Database Updates for Views Involving Selection, Projection and Joins", ACM, 1985, 154-163.

[KEL86a] - Keller,A.M., "The Role of Semantics in Translating Views Updates", Computer, Vol. 19, No. 1, 1986, 63-73.

[KEL86b] - Keller,A.M., "Choosing View Update Translator by Dialog at View Definition Time", Anais XII VLDB, 1986, 467-474.

[KEM87] - Kemper,A. & Lockemann,P. & Wallrath,M., "An Object-Oriented Database System for Engineering Applications", Anais ACM SIGMOD, 1987, 299-310.

[KEN79] - Kent,W., "Limitations of Record-based Information Models", ACM TODS, Vol. 4, No. 1, 1979, 107-131.

[KLU85] - Klausner,A. & Goodman,N., "Multirelations - Semantics and Languages", Anais XI VLDB, 1985, 251-258.

[LAF82a] - Lafue,G.M.E., "Semantic Integrity Dependencies and Delayed Integrity Checking", Anais VIII VLDB, 1982, 292-297.

[LAF82b] - Lafue,G.M.E., "Semantic Integrity Management of Database : A Survey", Lab. Comp. Science Research, Rutgers Univ., New Jersey, 1982.

[MAI83] - Maier,D., "The Theory of Relational Databases", Comp. Science Press, 1983.

[MAI85] - Maier,D., Otis,A. & Purcly,A., "Object-Oriented Database Development at Servio Logic", Database Engineering, Vol.8, No. 4, 1985, 58-67.

[MED85] - Medeiros,C.M.B., "A Validation Tool for Designing Database Views that Permites Updates", Phd Thesis, Univ. Waterloo, 1985.

[MED86a] - Medeiros,C.M.B., "Alternativas para Liberalizar Atualizações em Banco de Dados", Anais VI Cong. SBC, 1986, 105-115.

[MED86b] - Medeiros,C.M.B. & Tompa,F,W., "Understanding the Implications of View Update Policies", Algorithmica, Vol. 1, No. 2, 1986, 337-360.

[MED87] - Medeiros,C.M.B., "Um Modelo para Visões-Objetos", Anais 2º SBBD, 1987, 143-150.

[MED89] - Medeiros,C.M.B., "Views as Mechanism for Adding Types to a Database", a ser publicado em Anais IX Conferência da Sociedade Chilena de Computação, Santiago, 1989.

[MEL88] - Melo,R.N., "Banco de Dados não Convencionais : A Tecnologia do BD e Suas Novas Áreas de Aplicação", VI Escola de Computação, 1988.

[MIR88] - Miranda,S., "State of the Art of Data Models for Complex Objects", 3º SBBD, 1988, 50-61.

[MIT86] - Mittra,S.S., "Design Support Systems", John Wiley, 1986, New York.

[MOT81] - Motro,A. & Buneman,P., "Constructing SuperViews", ACM-SIGMOD, 1981, 56-64.

[MOT87] - Motro,A., "Superviews : Virtual Integration of Multiple Database", IEEE Trans. Soft. Eng., SE-13(7), 1987, 785-798.

[NAV86] - Navathe,S., Elmasri,R. & Honeywell,J.L., "Integrating User Views in Database Design", IEEE Trans. Soft. Eng.,SE-12(1), 1986, 50-62.

[NEU88] - Neuhold,E.J. & Schrefl,M., "Dynamic Derivation of Personalized Views", Anais XIV VLDB, 1988, 183-194.

[OSB86] - Osborn,S.L. & Heaven,T.E., "The design of a Relational Database System With Abstract Data Types for Domains", ACM TODS, Vol. 11, No. 3, Set-1986, 357-373.

[PIS86] - Pistor,P. & Anderson,F., "Designing a Generalized NF2 Data Model with an SQL - Type Language Interface", Anais XII VLDB, 1986, 278-285.

[PRO86] - Coleção Informática PRODESP, Série Software N. 4, Jul-1986.

[SCI83] - Sciore,E., "Improving Database Schemas by Adding Attributes", Anais ACM-PODS, 1983, 379-383.

[SCH88] - Schneider,H.N. & Medeiros,C.M.B., "Atualização de Atributos Agregados em Visões", Anais 3. SBBD, 1988, 127-137.

[SEN88] - Sena,G.J. & Furtado,A.L., "Um Prototipo de Ferramentas para Projeto de Banco de Dados", Anais 3. SBBD, 1988, 83-93.

[SHM84] - Shmueli,O. & Itai,A., "Maintenance of Views", Anais ACM-SIGMOD, 1984, 240-255.

[STO82] - Stonebraker,M., "Adding semantic Knowledge to a Relational Database System", Rel. Tec., Univ. California, 1982.

[STO83a] - Stonebraker,M., Fogg,D. & Ong,J., "Implementation of Data Abstraction in the Relational Database System INGRES", SIGMOD Record, ACM, New York, 1983, 1-13.

[STO83b] - Stonebraker,M., Rubenstein,B. & Guttman,A. "Application of Abstract Data Types and Abstract Indices to CAD Data", Anais ACM-SIGMOD, 1983, 317-333.

[STO87] - Stonebraker,M., "The POSTGRES Data Model", Anais XII VLDB, 1987, 83-96.

[TUC83] - Tucherman,L., Furtado,A.L. & Casanova,M.A., "A Pragmatic Approach to Structured Database Design", Anais IX VLDB, 1983, 219-231.

[TUC85] - Tucherman,L., Furtado,A.L. & Casanova,M.A., "A Tool for Modular Database Design", Anais XI VLDB, 1985, 436-447.

[UZS86] - Ozsoyoglu,G. & Ozsoyoglu,Z.M., "Statistical Database Query Languages", IEEE Trans. Soft. Eng., SE-12(12), 1986, 1106-1116.

[VIE88] - Vieira,M.T.P. & Melo,R.N., "Uma Análise de Banco de Dados Orientados para Objetos", 3º SBBD, 1988, 262-268.

ANEXO 1

MAPEAMENTOS DE ATUALIZAÇÃO SOBRE ALGUNS AGREGADOS

Esta anexo descreve alternativas para mapeamento de atualização de atributos estruturados gerados a partir de SOMA, MÉDIA ARITMÉTICA, MÁXIMO e MÍNIMO. Optou-se por estes quatro agregados por serem disponíveis em consultas (e consequentemente em visões) em grande parte dos SGBDs relacionais.

Os mapeamentos para agregados MÉDIA e SOMA são análogos, módulo o fato de que MÉDIA é a divisão de SOMA por um inteiro; da mesma forma, os mapeamentos para MAX e MIN são idênticos. Por este motivo, os exemplos e definição das regras a seguir serão expressos em termos de MÉDIA e MAX, para simplificar os detalhes. Os tipos de atualização considerados são :

1) Agregados MÉDIA (aritmética) / SOMA
   a) Modificar o valor de uma determinada MÉDIA / SOMA
   b) Inserir uma nova MÉDIA / SOMA
   c) Eliminar uma dada MÉDIA / SOMA

2) Agregados MAX / MIN
   a) Modificar o valor do MAX / MIN
   b) Estabelecer novo MAX/ MIN
   c) Inserir / Eliminar MAX / MIN

d) A operação modificar opera apenas sobre os valores de parâmetros, enquanto estabelecer pode requerer inserção ou eliminação de tupla.

A discussão será efetuada como se a visão fosse baseada numa única relação - a que dá os parâmetros para o agregado.

1 - Modificação do agregado MÉDIA

A modificação do valor do agregado tem duas modalidades : aumentar / diminuir a MÉDIA para um valor determinado; e aumentar / diminuir a MÉDIA de uma certa porcentagem. Ambas as modalidades são tratadas de forma idêntica. O mapeamento é feito obedecendo-se a noção de que um aumento da média é tratado por aumento de um ou mais valores dos parâmetros e que sua diminuição é mapeada para diminuição do valor dos parâmetros. O mapeamento pode ter duas opções : a) Distribuição Uniforme ; e b) Distribuição Não Uniforme.

a) Distribuição Uniforme :

A modificação é distribuída uniformemente pelos valores dos parâmetros do agregado, mantendo as restrições de domínio. Em outras palavras, seja a variação solicitada *MÉDIA* = *MÉDIA* + Δ, onde Δ ∈ [-10.0, 10.0]. Se o agregado *MÉDIA* for dado por $(\sum_i N_i)$ /

n, para i = 1..n, então a atualização para cada valor N_i é N_i + Δ.

b) Distribuição Não Uniforme :

A variação não uniforme corresponde a modificar os valores dos parâmetros segundo pesos distintos. Para agregados verticais, exige que o usuário identifique os pesos em função das tuplas, enquanto que para agregados horizontais deve indicar os atributos. No primeiro caso, é preciso que a visão contenha as chaves de todas as tuplas das tabelas-base para permitir a identificação; no segundo caso, o usuário deve conhecer os parâmetros. A distribuição não uniforme pode ocorrer por determinação expressa do usuário (que indica sobre quais parâmetros concentrar a alteração) ou quando a distribuição uniforme não é possível (por violar a restrição de domínio). No exemplo anterior, suponha que seja especificado um mapeamento com distribuição não uniforme, onde *N1* e *N2* devem contribuir cada um com 20% na nova média. O mapeamento seria então *N1* = *N2* = *n* * *0.2* * (*MÉDIA* + Δ); e todas as demais (n - 2) notas variariam de δ, onde δ = (*MEDIA* + Δ - *2* * *N1* / *n*) (ou seja, a distribuiçao uniforme pelo restante dos parâmetros).

No sistema descrito no capítulo 4, há ainda uma variante de distribuição não uniforme, (distribuição não uniforme automática) que ocorre quando uma variação uniforme não pode ser aplicada. Neste caso, varia-se o valor de um parâmetro até o limite

permitido, sendo a diferença entra a variação efetuada e a desejada distribuida pelos parâmetros restantes (dentre os submetidos a variação uniforme). Este processo é repetido até se obter o valor de agregado desejado ou constatar-se a impossibilidade da atualização. No exemplo anterior, caso um determinado atributo, por exemplo *N4*, pudesse apenas variar de $|\delta'| < |\delta|$, a diferença $|\delta'| - |\delta|$ seria distribuída entre os atributos *N3*, *N5* ... *Nn*. Por outro lado, se a composição de pesos especificada pelo usuário (no caso, para *N1* e *N2*) não puder ser atendida, o sistema se encarrega de informá-lo do fato e não realiza a operação.

2 - Inserção / Eliminação (sobre visão contendo MÉDIA) :

No caso de MÉDIA horizontal, a inserção na visão corresponde a inserção de uma nova tupla na relação. Para um dado valor do agregado, pode-se especificar distribuição uniforme ou não uniforme (tratadas de forma análoga a modificação). Para eliminação, remove-se a "fonte-limpa" da tupla da visão a ser eliminada.

No caso de agregado vertical, as operações não fazem sentido, pois corresponderiam a inserir ou eliminar uma relação inteira. No caso especial em que a visão é definida a partir de fragmentos horizontais de relação, cada um dando origem a uma tupla na visão, a inserção corresponde a definição de um novo

fragmento contendo uma única tupla e a eliminação destrói o fragmento que gerou a tupla na visão.

3 - Modificação do agregado MAX

Existem vários mapeamentos para as operações de atualização sobre o máximo. A tese oferece duas opções de tradução : a) modificar os valores dos parâmetros do agregado; e b) modificar o valor do parâmetro que deu origem ao valor do agregado.

a) Modificar valores dos parâmetros :

Isto significa que todos os valores dos parâmetros são modificados, havendo duas opções : na mesma proporção do valor do agregado; ou do mesmo valor absoluto. Se é solicitada a variação $MAX = MAX + \Delta$, onde MAX é um agregado horizontal (sobre n parâmetros A_i, para i = 1..n) ou vertical (sobre um parâmetro A_j), o primeiro caso implica $\rho(A_k) = \rho(A_k) * (MAX + \Delta) / MAX$ e o segundo $\rho(A_k) = (\rho_k) + \Delta$ (para todos os A_k que sejam parâmetros de MAX).

b) Modificar valor do parâmetro que originou agregado :

Isto significa a modificação do(s) valor(es) de todo(s) o(s) parâmetros A_k onde $\rho(A_k) = MAX$. Quando da modificação de um MAX para um valor menor, pode ser que a modificação dos valores dos parâmetros que lhe deram origem não seja suficiente para mapear a

atualização, já que outros parâmetros que mantiveram seus valores antigos podem passar a ter valor maior que o *MAX* desejado. Neste caso, estes valores deverão ser igualmente atualizados para o novo *MAX*. Seja o banco de dados e a visão

BD :	NOME	N1	N2	N3
	claudia	4.5	2.3	8.7
	eduardo	2.8	9.3	7.5

V :	NOME	NMAX
	claudia	8.7
	eduardo	9.3

A modificação de *NMAX* de "eduardo" de 9.3 para 7.0 requer, no caso (a), modificação do valor dos atributos para (2.8-2.3; 9.3-2.3; 7.5-2.3); e no caso (b) para (2.8, 7.0, 7.0).

4 - Estabelecer novo máximo

Quando se trata de máximo horizontal, é idêntico ao caso 3. Quando se trata de agregado vertical, dois tipos de tratamento podem ser adotados : o sistema insere na relação uma nova tupla que reflita o valor do agregado e elimina todas as tuplas onde o valor do parâmetro correspondente exceda o novo máximo; ou o sistema modifica a(s) tupla(s) que deram origem ao agregado (como em 3.b) e elimina aquelas onde o valor seja maior (ao invés de modificá-las como em 3.b).

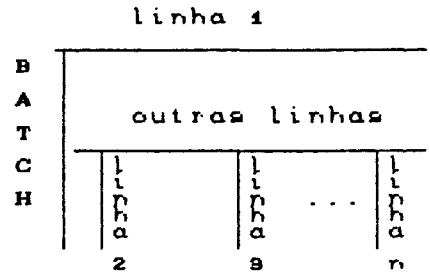
5 - Inserir / Eliminar (tupla de visão contendo) MAX

No caso de máximo vertical aplicam-se as mesmas considerações que em 2. No caso de máximo horizontal, a eliminação destrói a(s) tupla(s) que o originaram e a inserção é tratada como um estabelecimento de novo máximo.

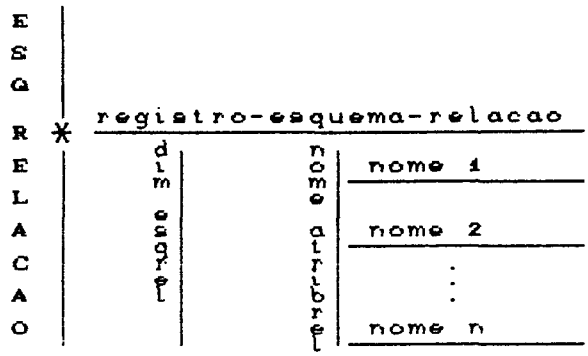
ANEXO 2

REPRESENTAÇÃO GRÁFICA DOS ARQUIVOS UTILIZADOS

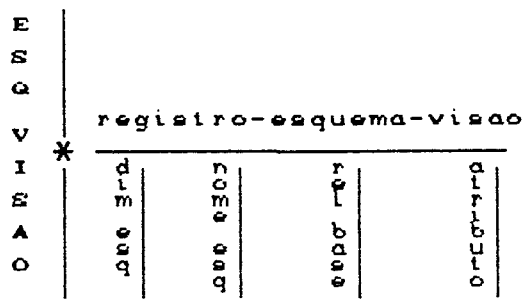
a)



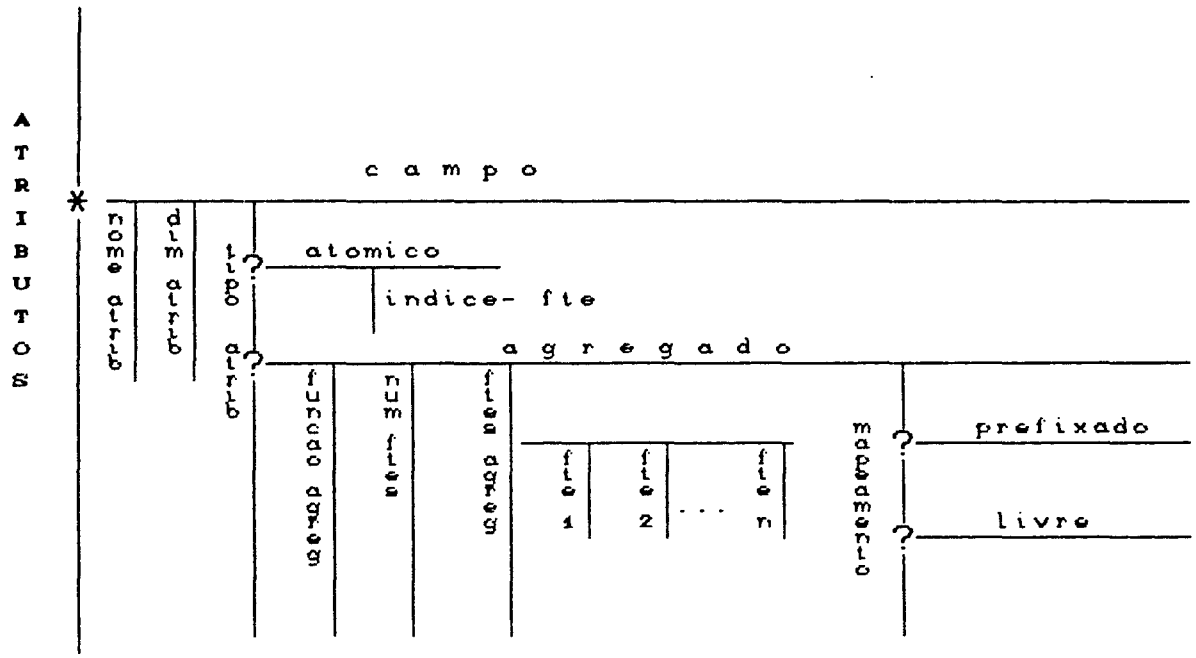
b)



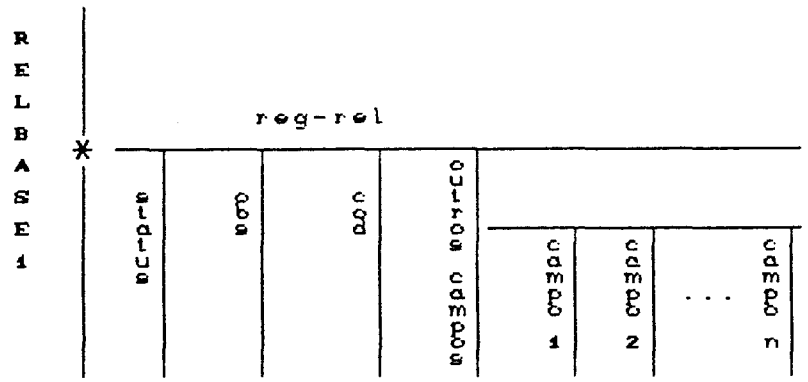
c-1)



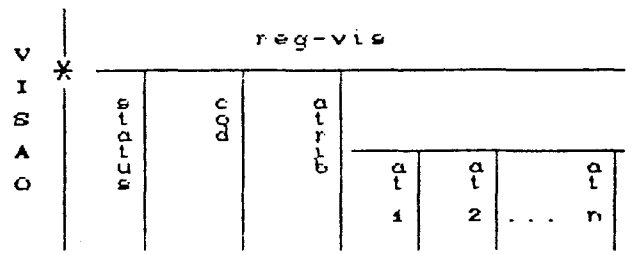
c-2)


ATRIBUTOS
campo
nome atrib
dim atrib
tipo atrib
atomico
indice- fte
agregado
funcao agreg
num fies
fies agreg
fie 1
fie 2
... fie n
mapeamento
prefixado
livre

d)


RELBASE1
reg-rel
status
cod
cod
outros campos
campo 1
campo 2
... campo n

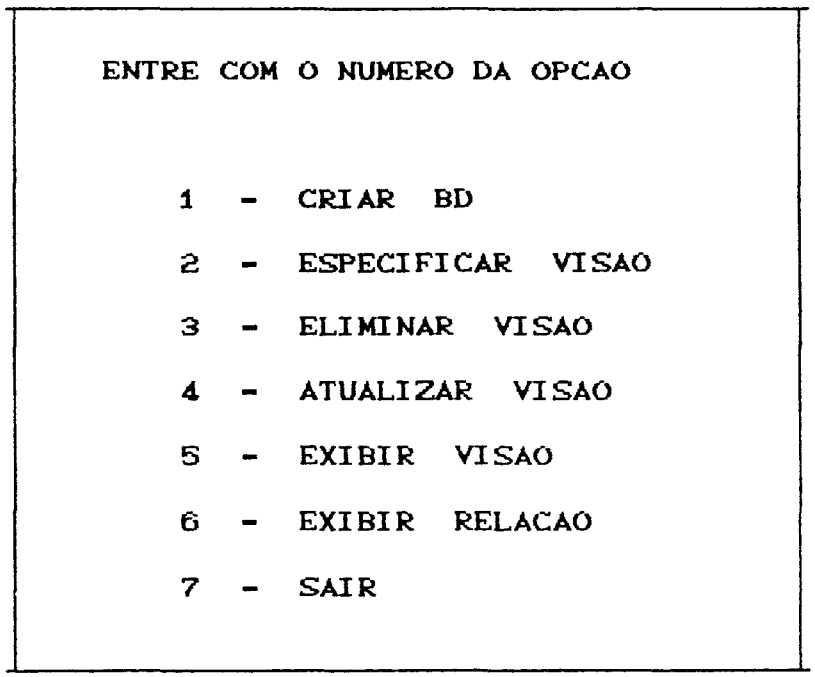
e)


VISAO
reg-vis
status
cod
atrib
at 1
at 2
... at n

ANEXO 3

MANUAL DO USUÁRIO - PROTÓTIPO IMPLEMENTADO

Quando o sistema é executado, aparece uma tela contendo o menu com os tipos de operações disponíveis para o usuário.



Tela1 - Menu Principal

Esta seção descreve os sub-módulos que são ativados quando é escolhida cada uma das opções.

1 - CRIAR BD :

SUB-MÓDULOS ATIVADOS :

* *Gerar_Relação*

DESCRIÇÃO DOS SUB-MÓDULOS :

* *Gerar_Relação :*

a) OBJETIVO :

- Materializar a relação-base a partir do arquivo "Batch".

b) ENTRADA :

"Batch".

c) SAÍDA :

"Esq_Relação", "Relbase1", "Rel1ind" e "Vídeo".

d) MENSAGENS :

i) Após materializar a relação-base, a rotina emite a seguinte mensagem para o usuário.

"RELACAO GERADA

NOME FISICO : Relbase1

TOTAL TUPLAS : x ; onde x é o número de registros de "Relbase1" calculado pela rotina.

FOI CRIADA ESTRUTURA DE ACESSO INDEXADA POR CHAVE PRIMARIA DE RELBASE1".

2 - ESPECIFICAR VISÃO :

SUB-MÓDULOS ATIVADOS :

* *Def_Esq_Vis*
* *Mat_Visão*

DESCRIÇÃO DOS SUB-MÓDULOS :

* *Def_Esq_Vis*

a) OBJETIVO :

- Definir o esquema de visão.

b) ENTRADA :

Informações via "Teclado".

c) SAÍDA :

"Esq_Visão".

d) LÓGICA :

- O usuário interage com a rotina interativamente respondendo às perguntas exibidas no vídeo. É perguntado o nome do esquema de visão, quantos atributos compõe o esquema, o nome da relação-base sobre a qual a visão será definida. Por exemplo, as telas exibidas são do tipo "*Qual o Nome do Esquema ? (Maximo 2 Caracteres)*", "*Quantos Atributos Comporao o Esquema ? (<= N_MAX_ATRIB)*" e assim por diante. Para cada atributo do esquema de visão é pedido : nome, tipo (atômico/agregado). Caso o atributo seja atômico, o usuário informará a posição do atributo da relação-base que será projetado na visão; caso seja agregado, o usuário indicará a função geradora do agregado, o número de

parâmetros usados no cálculo e a posição que cada parâmetro (atributo fonte) ocupa no esquema de relação-base. Quanto ao mapeamento, é solicitado que o usuário decida por ter mapeamento prefixado ou se prefere especificá-lo durante as transações. No primeiro caso, o sistema apresenta uma tela (tela2) com o menu de opções de mapeamentos para o agregado especificado. No segundo caso, a tela2 só será mostrada no momento da tradução da operação de atualização sobre a visão para operações sobre o banco de dados.

```
                ENTRE COM O NUMERO DA OPCAO

1 - DISTRIBUICAO UNIFORME
2 - DISTRIBUICAO NAO  UNIFORME  :  Especificacao de pesos por
                                   atributo
3 - DISTRIBUICAO NAO UNIFORME : Automatica
```

Tela2 - Mapeamentos para agregados

* *Mat_Visão*

a) OBJETIVO :

- Materializar a visão.

b) ENTRADA :

"Esq_Visão" e "Relbase1".

c) SAÍDA :

"Visão", "Visind" e "Vídeo".

d) LÓGICA :

- Depois de ler o arquivo "Esq_Visão", a rotina lê a relação-base e, para cada atributo do esquema de visão, ou projeta na visão o atributo atômico ou calcula o agregado requerido. Para cada atributo estruturado há uma rotina específica para seu cálculo.

e) MENSAGENS :

i) Depois de gravar a última tupla no arquivo "VISÃO" o sistema emite a mensagem :

"VISAO MATERIALIZADA"

ii) Quando do cálculo de agregados, se o usuário tiver especificado uma função desconhecida, o sistema emitirá a seguinte mensagem seguida da descontinuação do sistema.

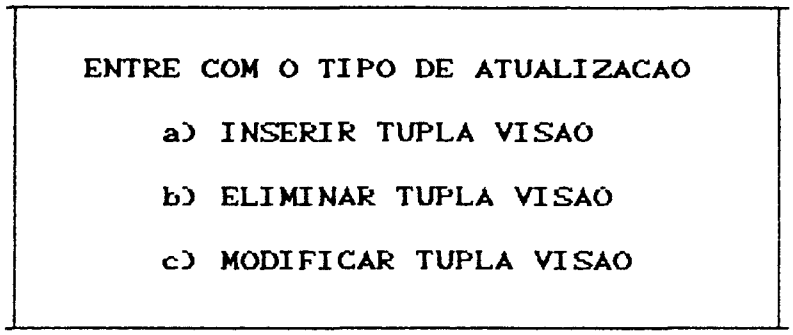
"ERRO : FOI ESPECIFICADA UMA FUNCAO NAO DEFINIDA".

3 - ELIMINAR VISÃO :

- Este módulo é utilizado para eliminar fisicamente os arquivos "Visão", "Visind" e "Esq_Visão" do diretório, afim de que o usuário especifique novo esquema de visão.

4 - ATUALIZAR VISÃO :

- O sistema apresenta uma tela (tela3) contendo os tipos de transações que o usuário poderá processar sobre a visão.



Tela3 - Operações Permitidas sobre a Visão

Descreveremos a seguir a computação realizada quando da escolha de cada uma das operações.

INSERIR TUPLA VISÃO :

SUB-MÓDULOS ATIVADOS :

* *Ins_Tupla_Vis*
* *Ins_Tupla_Rel*
* *Commit*

DESCRIÇÃO DOS SUB-MÓDULOS :

* *Ins_Tupla_Vis*

a) OBJETIVO :

- Formatar a tupla a ser inserida na visão.

b) ENTRADA :

"Esq_Visão" e Informações via "Teclado".

c) SAÍDA :

"RAM" e "Vídeo".

d) LÓGICA :

- A rotina lê o arquivo "Esq_Visão" e identifica através do esquema de visão o tipo de cada atributo, pedindo então que o usuário forneça o valor dos atributos. A cada valor especificado, é feita a consistência de domínio. Caso o valor especificado viole a restrição de domínio, o sistema emite uma mensagem para o usuário.

e) MENSAGENS :

i) Se o valor especificado para o atributo (atômico/agregado) violar restrição de domínio.

"ERRO : VALOR ESPECIFICADO VIOLA RESTRICAO DE DOMINIO

LIMITE SUPERIOR DE DOMINIO = x " ;

onde x é o valor limite superior de domínio para o atributo em questão.

* *Ins_Tupla_Rel*

a) OBJETIVO :

- Formatar a tupla da relação-base, inclusive valorando os atributos fonte de agregados como resultado da execução do mapeamento escolhido.

b) ENTRADA :

"Esq_Visão" e Estrutura secundária em "RAM" contendo a tupla a ser inserida na visão.

c) SAÍDA :

"RAM" e "Vídeo".

d) LÓGICA :

- Para cada atributo do esquema de visão, a rotina verifica se é do tipo atômico ou agregado. Se tratar-se de atributo atômico, a rotina valora o atributo fonte apontado pelo campo de "Esq_Visão" que contém a sua posição no esquema de relação. Se tratar-se de atributo agregado, há duas alternativas a analisar.

i) Mapeamento Prefixado.

- A rotina verifica no arquivo que contém o esquema de visão qual foi o mapeamento escolhido previamente e o executa.

ii) Mapeamento escolhido no momento da atualização.

- A rotina chama a função que exibe o menu de opções de mapeamentos para o agregado específico e, em seguida, o usuário opta pelo mapeamento.

A *tela2* já apresentada mostra o menu com opções de mapeamentos para os agregados SOMA e MÉDIA.

Depois de escolhida a opção de mapeamento (caso (ii)) ou imediatamente após ler o tipo de mapeamento pré-estabelecido (caso (i)), a rotina que traduz a operação de atualização sobre a visão é chamada e calcula os valores atômicos dos atributos fonte. Para as opções "distribuição uniforme" e "distribuição automática" não há necessidade de fazer consistência de domínio, pois o valor que o atributo agregado recebeu foi comparado com os

limites de domínio permitidos no sub-módulo "Ins_Tupla_Vis" e é tal que garante estas distribuições sobre os atributos fonte.

Porém, para "distribuição ponderada" é necessário fazer consistência de domínio, pois o usuário pode escolher uma ponderação que produza valores que extrapolem os limites de domínio dos atributos atômicos tomados como fonte do agregado.

Novamente, se tratar-se de mapeamento especificado a cada operação e acontecer a violação das regras de consistência de domínio, o sistema avisará o usuário (mensagem (i)) e reexibirá a tela com o menu de opções de mapeamentos. Em se tratando de mapeamento prefixado e ocorrer a violação da restrição de domínio, o sistema avisará, também, o usuário (mensagens (i) e (ii)) e abandonará a operação de inserção. Se os valores produzidos no mapeamento são válidos, é mostrado ao usuário o conjunto de valores produzidos pela operação de tradução. Caso o usuário não fique satisfeito com os valores mostrados, pode optar por um dos seguintes caminhos :

1) Mapeamento Prefixado.

   - O sistema abandona a operação de inserção.

2) Mapeamento Livre.

   - O sistema repete o ciclo de mapeamento : exibe menu com opções de mapeamentos, executa mapeamento escolhido, questiona sobre aceitação dos valores calculados.

No caso do usuário confirmar os valores calculados no mapeamento, o sistema pergunta se ele quer concretizar a operação de atualização, gravando as novas tuplas na visão e na

relação-base correspondente (mensagem (v)). Se a resposta for negativa, o sistema abandonará a operação de inserção e, se for positiva, chamará o módulo "COMMIT" descrito a seguir.

e) MENSAGENS :

i) Quando a opção de mapeamento para agregados "SOMA" e "MÉDIA" for "distribuição ponderada" e os pesos especificados produzir algum valor que extrapole a região permitida para o domínio do atributo atômico fonte, o sistema emitirá a mensagem :

"ATRIBUTO FONTE NAO SUPORTA CONCENTRACAO DE y % SOBRE SI" ;

onde y é o valor do peso especificado pelo usuário.

ou

"ERRO : VALOR DISTRIBUIDO PELOS ATRIBUTOS REMANECENTES VIOLA RESTRICAO DE DOMINIO".

ii) Para o mapeamento "livre", depois que o controle é devolvido à rotina INS_TUPLA_REL pela rotina de mapeamento "DIST_POND", se a restrição de domínio foi violada, o sistema emitirá a mensagem:

" FACA OUTRA ESCOLHA DE MAPEAMENTO "

iii) Para o mapeamento prefixado, na situação descrita em (ii), além da mensagem (i), o sistema emitirá a mensagem :

"OPERACAO DE INSERCAO SERA ABANDONADA"

iv) No caso de mapeamento prefixado não conhecido, o sistema avisará com a seguinte mensagem seguida da descontinuação do

programa.

"ATENCAO !

FOI ESPECIFICADO MAPEAMENTO NAO DEFINIDO"

v) Quando, finalmente, o usuário aceita os valores calculados no mapeamento, o sistema emitirá a mensagem :

" 'COMMIT' TRANSACAO PROCESSADA ? (S/N) "

vi) Caso ocorrer, por algum erro do subsistema de acesso, uma situação em que é permitido ao usuário inserir uma tupla na visão com chave já existente na relação-base, o sistema emitirá a mensagem abaixo, seguida da descontinuação do programa.

"ATENCAO - INSERCAO SEM SUCESSO

JA EXISTE TUPLA NA RELACAO BASE IDENTIFICADA POR CHAVE x ;

onde x é a chave da tupla inserida na visão.

ARQUIVOS DE INDICES DA RELACAO BASE E DA VISAO ESTAO INCOMPATIVEIS

OS ARQUIVOS PRECISAM SER REVISTOS"

* *Commit*

a) OBJETIVO :

- Gravar/Eliminar/Regravar tuplas na visão e relação-base.

b) ENTRADA :

- Estrutura auxiliar na "RAM" contendo as tuplas da visão e da relação-base.

c) SAÍDA :

"Visão", "Visind", "Relbase1", "Rel1ind" e "Vídeo".

d) LÓGICA :

- Dependendo do sub-módulo que fez a chamada a "Commit", será executado a parte do código que grava tupla na visão e relação-base (*Inserir Tupla Visão*) ou elimina tupla da visão e relação-base (*Eliminar Tupla Visão*) ou regrava tupla modificada na visão e relação-base (*Modificar Tupla Visão*).

e) MENSAGENS :

i) Depois de processada a gravação/eliminação/regravação, o sistema emite a seguinte mensagem :

"ATUALIZACAO FISICA PROCESSADA NO BANCO DE DADOS".

ELIMINAR TUPLA VISÃO

SUB-MÓDULOS ATIVADOS :

* *Del_Tupla_Vis*
* *Del_Tupla_Rel*
* *Commit*

DESCRIÇÃO DOS SUB-MÓDULOS :

* *Del_Tupla_Vis*

a) OBJETIVO :

- Procurar, na estrutura de acesso, a chave da tupla a eliminar e se encontrá-la, guardar em memória a localizaçaõ física da tupla na visão.

b) ENTRADA :

Informações via "Teclado" e "Visind".

c) SAÍDA :

"Vídeo" e "RAM".

d) LÓGICA :

- A rotina pede que o usuário entre com o valor da chave primária da tupla a eliminar da visão, a procura no arquivo de índices associado à visão e emite as mensagens devidas.

e) MENSAGENS :

i) Se chave por encontrada na busca :

"TUPLA IDENTIFICADA POR CHAVE z FOI ENCONTRADA NA VISAO" ;

onde z é o valor da chave fornecida pelo usuário.

ii) Se a rotina interna de busca não encontrar a chave requerida:

"ATENCAO - ELIMINACAO SEM SUCESSO
NAO EXISTE NA VISAO TUPLA IDENTIFICADA POR CHAVE z ".

* *Del_Tupla_Rel*

a) OBJETIVO :

- Verificar se a chave da tupla a eliminar da visão existe no arquivo de índices associado à relação-base.

b) ENTRADA :

- Informações na "RAM".

c) SAÍDA :

"Vídeo" e "RAM".

d) LÓGICA :

- A rotina importa da memória principal o valor da chave passado pelo sub-módulo anterior e verifica se ela existe na estrutura de acesso da relação-base.

e) MENSAGENS :

i) Sucesso na busca :

"TUPLA IDENTIFICADA POR CHAVE z FOI ENCONTRADA NA RELACAO-BASE" ;

onde z é a chave lida da memória.

ii) No insucesso da busca, o sistema emitirá a mensagem seguida da descontinuação do programa.

"ATENCAO - ELIMINACAO SEM SUCESSO
NAO EXISTE TUPLA NA RELACAO-BASE IDENTIFICADA POR CHAVE z
OS ARQUIVOS DE INDICES DA RELACAO-BASE E DA VISAO ESTAO INCOMPATIVEIS
O ARQUIVOS PRECISAM SER REVISTOS ".

MODIFICAR TUPLA VISÃO

SUB-MÓDULOS ATIVADOS :

* *Mod_Tupla_Vis*
* *Mod_Tupla_Rel*
* *Commit*

DESCRIÇÃO DOS SUB-MÓDULOS :

* *Mod_Tupla_Vis*

a) OBJETIVO :

- Modificar o valor dos atributos especificados pelo usuário.

b) ENTRADA :

"Esq_Visão", "Visind" e Informações via "Teclado".

c) SAÍDA :

"RAM" e "Vídeo".

d) LÓGICA :

- A rotina pede ao usuário que entre com o valor da chave da tupla a modificar e, se esta for encontrada no arquivo de índices, o sistema acessará, na visão, a tupla indexada por aquela chave e depois de ler o arquivo que contém a descrição do esquema de visão, o sistema exibirá, um a um, os atributos (nome, tipo e valor) perguntando se o usuário quer modificar o valor de cada atributo exibido.

Se a resposta for positiva, o sistema pede ao usuário que informe o novo valor do atributo (atômico/agregado) e depois fará os testes de consistência de domínio. Quando o usuário acabar as alterações, a tupla modificada fica armazenada na memória principal para posterior regravação na visão (Commit).

e) MENSAGENS :

i) Se o novo valor do atributo (atômico/agregado) ferir a restrição de integridade do domínio, o sistema emitirá a mensagem :

" ERRO : VALOR ESPECIFICADO VIOLA RESTRICAO DE DOMINIO
LIMITE SUPERIOR DE DOMINIO = x " ;

onde x é o maior valor permitido para o atributo.

ii) Se não existir tupla identificada por chave especificada.

" ATENCAO - MODIFICACAO SEM SUCESSO

NAO EXISTE TUPLA NA VISAO IDENTIFICADA POR CHAVE z " ;

onde z é o valor da chave especificada pelo usuário.

* *Mod_Tupla_Rel*

a) OBJETIVO :

- Repassar para a relação-base as modificações feitas na visão pelo usuário.

b) ENTRADA :

"Esq_Visão", "Relind" e "RAM".

c) SAÍDA :

"RAM" e "Vídeo".

d) LÓGICA :

- Primeiramente é chamada a rotina interna de busca afim de verificar se a chave passada pelo sub-módulo *Mod_Tupla_Vis* existe no arquivo de índices associado à relação-base. Se a chave for encontrada, a rotina acessa as informações acerca do esquema de visão (Esq_Visão) para conhecer o tipo de cada atributo que compõe o esquema.

Se o atributo for "atômico", o seu valor será repassado para o atributo fonte da relação-base (o sistema o conhece, pois existe um campo no registro de Esq_Visão que aponta o(s) atributo(s) fonte de cada atributo do esquema de visão).

No caso de tratar-se de atributo agregado, há duas situações a analisar :

i) *Mapeamento Prefixado.*

- O sub-módulo de modificação de tupla da relação-base chama a rotina de mapeamento especicificada no registro do esquema de visão para este atributo. O processo é igual ao descrito em INS_TUPLA_REL, inclusive as mesmas mensagens serão emitidas, pois os módulos de mapeamentos são compartilhados.

ii) *Mapeamento Livre.*

- Como em INS_TUPLA_REL, o sistema apresentará a *tela2* para que o usuário escolha a opção de mapeamento, sendo o processo idêntico a INS_TUPLA_REL.

e) MENSAGENS :

As mesmas emitidas por INS_TUPLA_REL trocando-se, onde aparecer, "inserção" por "modificação".

5 - EXIBIR VISÃO :

SUB-MÓDULOS ATIVADOS :

* *Exibir_Visão*

DESCRIÇÃO DOS SUB-MÓDULOS ATIVADOS :

* *Exibir_Visão*

a) OBJETIVO :

- Mostrar o conteúdo da visão para o usuário.

b) ENTRADA :

"Esq_Visão", "Visão".

c) SAÍDA :

"Vídeo".

d) LÓGICA :

- O sub-módulo faz uma leitura em "Esq_Visão" para formatar o cabeçalho com os nomes dos atributos do esquema de visão. Depois lê exaustivamente o arquivo "Visão" e coloca no formato de saída as tuplas que tiverem o campo "status" com valor igual a zero.

Serão mostradas 17 tuplas por vez no vídeo, devendo o usuário responder à mensagem emitida (mensagem (i)) para visualizar outra página de saída. A cada exibição de página, o cabeçalho será reeditado.

e) MENSAGENS :

i) Quando a página é exibida.

" PRESSIONE QUALQUER TECLA PARA CONTINUAR ...".

6 - EXIBIR RELAÇÃO :

SUB-MÓDULOS ATIVADOS :

* *Exibir_Relação*

DESCRIÇÃO DOS SUB-MÓDULOS :

* *Exibir_Relação*

a) OBJETIVO :

- Exibir a relação-base armazenada para o DBA ou usuário autorizado.

b) ENTRADA :

"Esq_Relação", "Reelbase1" e Informações via "Teclado".

c) SAÍDA :

"Vídeo".

d) LÓGICA :

É pedido que o usuário forneça a "senha" que o qualifica como autorizado ou não a ver o Banco de Dados. Caso se trate de um usuário autorizado, a rotina acessa o arquivo que contém a descrição do esquema de relação-base e formata o cabeçalho com os nomes dos atributos do esquema. Depois é executado o "loop" que lê cada tupla da relação-base, a coloca no formato de saída e exibe aquelas não marcadas como eliminadas (status = 0).

Será mostrada no vídeo uma página com o máximo de 17 tuplas e para continuar o usuário responderá a mensagem emitida.

e) MENSAGENS :

i) Para exibir nova página ou voltar ao menu principal :

" PRESSIONE QUALQUER TECLA PARA CONTINUAR ... ".

ii) Caso o usuário não pertença à classe de usuários autorizados a ver o Banco de Dados :

" DESCULPE-ME !

VOCE NAO ESTA AUTORIZADO A VER O BANCO DE DADOS" .

7 - SAIR :

a) OBJETIVO :

- Sair do sistema e ir para ambiente Turbo-Pascal.

b) LÓGICA :

- Ao escolher a opção "7" do menu principal, a variável lógica, que serve como condição de fim do "loop" que exibe a *telaI* com o menu principal, é setada com "true" e, a seguir, são fechados os arquivos "Visão", "Relbase1" e respectivos arquivos de índices.