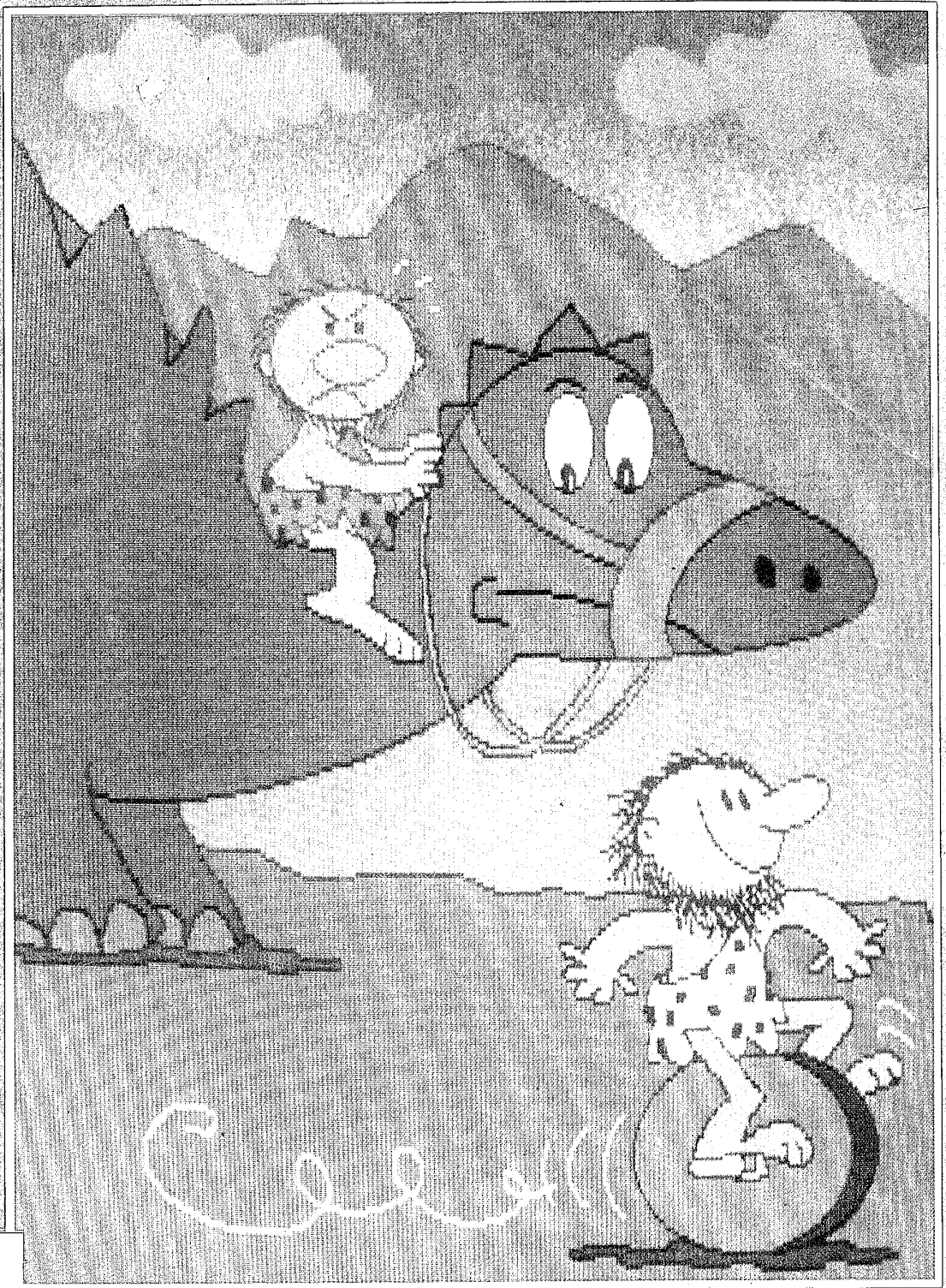
ANAIS

XXI CONGRESSO NACIONAL DE INFORMÁTICA



SUCESU'88

RIO DE JANEIRO

FOCO

SUCESU'88

XXI CONGRESSO NACIONAL DE INFORMÁTICA

Rio de Janeiro de 22 a 26 de agosto de 1988

ANAIS

VOLUME 1

Sociedade dos Usuários
de Computadores e Equipamentos
Subsidiários - RJ
Rua do Carmo, 57/6º and.
Rio de Janeiro - CEP 20011
Tel.: (021) 221-5183
Telex.: (21) 32522 - SUSU-BR

FOCO
Feiras, Exposições
e Congressos Ltda.
Rua da Ajuda, 35/7º and.
Rio de Janeiro - CEP 20040
Tel.: (021) 210-3237
TELEX.: (21) 21864 - FOCO-BR
FAX.: (021) 533-0892

IMPLEMENTAÇÃO DE UM PROTÓTIPO DE FERRAMENTA PARA PROJETO/REPROJETO DE BANCOS DE DADOS

AUTORES:
GALENO JOSÉ DE SENA
ANTÔNIO LUZ FURTADO

ENDEREÇOS:
GALENO JOSÉ DE SENA
Faculdade de Engenharia de Guaratinguetá - FEG/UNESP
Depto. de Matemática
Av. Dr. Ariberto Pereira da Cunha, 333
CEP 12500 - GUARATINGUETÁ - SP
Fone: (0125) 22-2800
ANTÔNIO LUZ FURTADO
Pontifícia Universidade Católica do Rio de Janeiro - PUC/RJ
Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
CEP 22453 - RIO DE JANEIRO - RJ
Fone: (021) 529-9396

CURRICULUM VITAE:
GALENO JOSÉ DE SENA é Engenheiro Mecânico (1983) pela FEG, Mestre em Ciências em Informática (bancos de dados) (1987) pela PUC/RJ e doutorado em Informática também na PUC/RJ. Suas áreas de interesse compreendem bancos de dados e inteligência artificial.
ANTÔNIO LUZ FURTADO é Bacharel em Economia (1964) pela Universidade do Estado da Guanabara, Mestre em Informática (1969) pela PUC/RJ e Doutor em Ciência da Computação (1974) pela Universidade de Toronto, Canadá. Áreas de interesse: estruturas de informação, bancos de dados e teoria dos grafos.

RESUMO:
Uma disciplina de modularização para especificação de esquemas de bancos de dados modulares é descrita. A estrutura de um protótipo de geração de planos é apresentada. Uma ferramenta especialista para projeto/reprojeto de bancos de dados, suportando a metodologia e a integrando com o gerador de planos, é então descrita.

PALAVRAS-CHAVE:
Projeto Lógico de Bancos de Dados, Restrições de Integridade, Preservação de Consistência, Programação Lógica, Sistemas Especialistas, Geração de Planos.

1. INTRODUÇÃO

O que motiva, basicamente, o desenvolvimento de uma disciplina de modularização [CFT, FCT], a ser usada na especificação de esquemas de bancos de dados, e a construção de uma ferramenta suportando a sua aplicação, é o problema de organização do projeto e manutenção de esquemas de bancos de dados complexos, considerado no sentindo amplo de como se descrever as estruturas de dados (parte estática) e as transações (parte dinâmica) de um sistema de informação (SI) baseado em um banco de dados. A utilização de *ferramentas* [BGM, CFT, Ce, FCT] para auxiliar no projeto de sistemas de informação [LM] baseados em bancos de dados é largamente reconhecida, entre outras razões elas podendo constituir o único modo efetivo de se conduzir métodos de projeto formais para o domínio das aplicações. *Sistemas especialistas* [BGM, FM, Ni], incorporando o conhecimento de especialistas humanos em alguma área, estão sendo aplicados na solução de um grande número de problemas. Uma combinação dos dois conceitos — ferramentas de projetos e sistemas especializados, daí resultando as *ferramentas especialistas* [BGM, CFT, Ce, FCT] — constitui, como se poderia esperar, uma área de pesquisa intensa [BGM].

O conhecimento a incorporar na ferramenta deve refletir, sobretudo, um método para projeto/reprojeto de bancos de dados [FCT]. A capacidade de inferência [FM, Ko, Ni, Wa] pode, em especial, usar regras gerais para sintetizar seqüências de aplicações de funções (ativação de um *sistema de geração de planos*), capazes de conduzir a aplicação do estado corrente a um estado caracterizado (parcialmente) por uma conjunção de fatos dada.

O presente trabalho descreve uma ferramenta especialista, efetivamente implementada de programação micro-PROLOG [CM] estendida com o sistema APES [HS], suportando a disciplina de modularização descrita em [CFT], e ativando, na fase de projeto, um sistema de geração de planos baseado no módulo de processamento da ferramenta de [FM, VF], para facilitar o processo de verificação de alguns dos requisitos semânticos da metodologia.

2. A DISCIPLINA DE MODULARIZAÇÃO

A fase de *projeto* da metodologia é caracterizada pela edição sistemática de módulos, de modo estruturado, a um núcleo de banco de dados, possivelmente vazio, definindo-se, a nível de módulo, os conjuntos de objetos compatíveis com o seu tipo. O processo de modificação ou de introdução de novos objetos em módulos já especificados caracteriza a fase de *reprojeto* da metodologia.

Há no método três tipos de módulos: *primitivo, subordinante* e *externo*. O ABD escolhe o tipo de primitivo para descrever porções da aplicação que têm sentido por si mesmas. Módulos subordinantes são construídos a partir de módulos já especificados, os módulos *subordinados*, o construtor de subordinação sendo usado pelo ABD quando desejar introduzir, entre outros objetos, uma restrição de integridade que pode ser violada por alguma operação definida previamente em algum módulo do esquema, a operação tornando-se escondida e o módulo que a contém, subordinado. Módulos externos são também definidos sobre módulos previamente introduzidos, os módulos *estendidos*, o ABD devendo fazer uso deles quando desejar introduzir visões e operações sobre visões. Um módulo é *conceitual* se ele é primitivo ou subordinante. Um *esquema de banco de dados modular* é caracterizado por um conjunto de módulos, cada módulo sendo um módulo conceitual ou externo.

Fazendo uso do modelo de dados relacional [Da], um módulo pode conter, não importando o seu tipo, definições de *relações* (tabelas), *restrições de integridade* e *operações.* As restrições de integridade estabelecem quais os estados que são válidos no banco de dados (restrição estática) e que transições de estados são válidas (restrições de transição). Para especificar, num módulo conceitual, que o corpo de uma operação contém testes para garantir a não violação de uma restrição de integridade, introduz-se uma *cláusula de garantia* no esquema, que corresponde a uma declaração da forma:

"operação *garante* restrição de integridade"

os objetos participando do relacionamento devendo ter sido introzidos no mesmo módulo. Para o caso de específico de módulos subordinantes deve haver a especificação de *cláusulas de recobrimento.* Para especificar que a aplicação de uma operação pode violar uma restrição de integridade, introduz-se uma destas cláusulas, a qual corresponde a uma declaração da forma:

"operação *pode-violar* restrição de integridade"

os objetos participando do relacionamento devendo ter sido especificados em módulos diferentes: a operação em um módulo subordinado e a restrição de intergridade no módulo subordinante.

Em módulos externos dever haver a introdução de *mapeamentos de definição de visão* e de *procedimentos substitutivos.* Um mapeamento de definição de visão, introduzido para cada relação do módulo, permite defini-la em termos das relações dos módulos estendidos. Um procedimento substitutivo, especificado para cada operação do módulo, é uma operação definida sobre a união das relações dos módulos estendidos, devendo ser uma tradução exata e aceitável da operação correspondente do módulo externo [FC].

Uma das características da metodologia é a existência de uma ordem natural [FCT] para criar ou modificar uma especificação. A nível de módulos, se o módulo M sobordina ou estende o módulo M', então M' deve ser definido antes de M. A nível de componentes a ordem é: (1) relações, (2) restrições de integridade, (3) cláusulas de recobrimento, (4) operações, (5) cláusulas de garantia. A ordem natural caracteriza a *ordem de definição* [CFT] da metodologia, na qual o ABD deve definir ou modificar os objetos do esquema.

A consistência no banco de dados resultante da aplicação dos construtores da metodologia é obtida adotando-se um conjunto de *requisitos* de projeto, que definem o que é um projeto modular consistente e formam o núcleo de disciplina de modularização. A formalização dos requisitos de projetos gera as *leis de integridade,* que são implementadas no protótipo para constituir um módulo de verificação dos requisitos nas fases de projeto e reprojeto do esquema. A formalização dos requisitos usados pela ferramenta de reprojeto inclui as leis de propagação. Estas são usadas para otimizar o reprojeto por dirigir a atenção para aqueles componentes que podem ter sido afetados por alguma alteração efetuada pelo ABD e que podem, como conseqüência, necessitar também de alguma alteração por sua vez. A ordem de definição garante que a propagação de alterações somente ocorre na direção ascendente (ou seja, do objeto que sofreu alteração para objetos definidos depois dele segundo ordem de definição).

3. O PROTÓTIPO DE GERAÇÃO DE PLANOS

O problema de formação de planos [Ko] é caracterizado por um *estado inicial*, um *estado objetivo* e um conjunto de *ações* utilizadas nas transições de estados da aplicação. O objetivo é obter um *plano:* seqüência apropriada de ações capazes de transformar o estado inicial num estado final que satisfaça a descrição do estado objetivo (este será um subconjunto do estado final).

A implementação do protótipo, efetuada em micro-PROLOG [CM], está baseada no sistema WARPLAN [Wa] — um gerador de planos de propósito geral para domínios descritos em um formalismo próximo do usado pelo sistema STRIPS [Ni], e no protótipo apresentado nos trabalhos de [FM, VF].

Uma aplicação pode ser vista como um tipo de dado abstrato, podendo, como conseqüência, ser especificada em termos das funções neste definidas [BZ]. As funções para consultas correspondem aos fatos conceituais ao passo que as funções para alterações são definidas especificando-se:
(a) os domínios de que vêm os valores de seus parâmetros;
(b) os fatos que estas adicionam;
(c) os fatos que estas excluem e
(d) as pré-condições para a sua aplicação.
Este estilo de especificação, orientado por funções [FM], define explicitamente as funções de alterações, os fatos somente comparecendo implicitamente nas condições e efeitos das funções. Designa-se por "fra-

me problem" [Ko, Ni, Wa] o problema de como se especificar os fatos que têm validade no estado que antecede a aplicação de alguma função e que continuam válidos no estado que sucede à aplicação.

O protótipo é composto de dois módulos: um específico, o *módulo de aplicação* [FM], que se refere a um particular sistema de informação de bancos de dados e corresponde à especificação do esquema conceitual da aplicação, e outro, geral, o *módulo de processamento* [FM], que permite a execução de experimentos com fatos do banco de dados, a nível do esquema conceitual da aplicação.

O módulo de aplicação é estruturado com a introdução de cláusulas para os predicados seguintes:

"operation-on": para especificação dos domínios de onde são obtidos os valores dos parâmetros das funções;

"added": para especificação dos fatos adicionados pelas funções;

"deleted": para especificação dos fatos excluídos pelas funções;

"conditions": para especificação das pré-condições para a aplicação de funções e

"imposs": para especificação de conjunções de fatos que violam as restrições de integridade do banco de dados.

Os axiomas do módulo de processamento do protótipo adaptam um algoritmo [Wa] para a geração de seqüências alternativas de aplicações de funções (ações) que transformam a especificação de uma dada aplicação de um estado inicial (ou de qualquer estado satisfazendo a descrição inicial) para um estado objetivo (ou algum estado satisfazendo a descrição do objetivo).

4. O PROTÓTIPO DE PROJETO/REPROJETO

O protótipo resultante da implementação de método de projeto/reprojeto e da integração deste com o de geração de planos está organizado na forma de duas unidades [CFT, FCT, Se]: uma *ferramenta de projeto* — auxiliando o ABD na especificação de novos módulos, e uma *ferramenta de reprojeto* — ajudando-o na especificação de alterações para o esquema modular. A integração da ferramenta de projeto com o protótipo de geração de planos foi realizada por meio de um programa de ligação [Se]. O protótipo foi implementado em micro-PROLOG [CM] estendido com o sistema APES [HS]. A organização das ferramentas é analisada a seguir.

4. 1 A Ferramenta de Projeto

A figura 1 (ver anexo 1) mostra os componentes da ferramenta de projeto e o fluxo de controle, uma seta simples indicando a ativação de um componente e uma seta dupla, saída de resultados.

Os componentes têm as funções seguintes:

(a) o *gerenciador de projeto (GPR)* rege a criação de módulos e de seus objetos em uma ordem compatível com a ordem de definição;

(b) o *administrador de diálogos (AD)* controla a comunicação com o ABD, verifica a correção de suas respostas e armazena, no dicionário de dados, cláusulas descrevendo os objetos de esquema modular;

(c) o *dicionário de dados (DD)* contém a descrição do esquema modular na forma de cláusulas;

(d) o *analisador sintático (AS)* verifica a sintaxe livre do contexto e constrói as árvores de análise para fórmulas bem formadas do cálculo de predicados de primeira ordem e para operações da linguagem de programação formal descrita em [CB].

(e) o *dicionário de leis de integridade (DLI)* contém uma implementação das leis de integridade da metodologia;

(f) o *integrador com o gerador de planos (IGP)* gera as cláusulas do módulo de aplicação do protótipo de geração de planos e

(g) o *gerador de planos (GPL)* contém o módulo de processamento do protótipo de geração de planos e, usando as cláusulas do módulo de aplicação, verifica a suficiência das pré-condições das operações do esquema.

O fluxo de controle é comentado brevemente no que segue.

O GPR ativa o AD para comunicação com o ABD, desta resultando a especificação de módulos e de seus objetos, resultado da comunicação sendo armazenado no DD. O DLI ativa o AD para comunicação com o ABD, estabelecida com o propósito de verificar os requisitos semânticos da metodologia. Deve ser observado que o AD implementa uma forma amigável de comunicação com o ABD.

O DLI é invocado pelo GPR e pelo AD para a verificação dos requisitos de projeto, verificação esta que garantirá a consistência do banco de dados.

O DLI ativa as AS para dois propósitos: (1) verificar a sintaxe livre do contexto de fórmulas descrevendo mapeamentos de visões ou restrições de integridade e de programas definindo operações ou procedimentos substitutivos e (2) percorrer as árvores de análise geradas buscando por subexpressões de forma apropriada, para verificação de condições sensitivas ao contexto.

O GPR ativa o IGP para a construção de módulo de aplicação do protótipo de geração de planos. Para isto, o IGP necessita das árvores de análise das operações do esquema, que são obtidas chamando-se o AS, e de se comunicar com o ABD para o estabelecimento das conjunções de fatos caracterizando estados de violação das restrições de integridade, quando então chama o AD. Concluída a estruturação do módulo de aplicação, o IGP ativa o GPL com o propósito de confirmar a suficiência das pré-condições das operações do esquema, o ABD sendo advertido caso isto não se comprove.

A ferramenta não implementa diretamente os construtores da metodologia: ela estabelece um diálogo com o ABD do qual resulta a descrição de novos módulos. Para criar um novo módulo, o ABD ativa a ferramenta entrando com:

"module-name M"

com M sendo o nome do módulo. Um esquema modular é armazenado no dicionário na forma de um conjunto de cláusulas definindo um predicado designado "tab", que toma parte então no programa lógico que estrutura a ferramenta.

4.2 Integração da Ferramenta de Projeto com o Módulo de Geração de Planos

Foi mencionado que a integração da ferramenta de projeto com o gerador de planos foi realizada atra-

vés de um programa de ligação [Se], este correspondendo ao componente IGP da figura 1 (ver anexo 1). Este estrutura o módulo de aplicação para o gerador de planos a partir das operações e das restrições de integridade do módulo considerado. A estrutura desse programa de ligação é apresentada na figura 2 (ver anexo 1), as setas indicando a ordem de ativação de cada uma das funções que o constituem, estas sendo descritas no que segue:

(a) *cláusulas sintáticas (CST):* abarca as subfunções seguintes:
(a.1) *cláusulas de adição (CA):* gera as cláusulas do predicado "added",
(a.2) *cláusulas de exclusão (CE):* gera as cláusulas do predicado "deleted";
(a.3) *cláusulas de domínios (CD):* gera as cláusulas do predicado "operation-on";
(a.4) *cláusulas de condições (CC):* comporta as subfunções seguintes:
(a.4.1) *condições de comandos (CCo):* estrutura as listas de condições das cláusulas do predicado "conditions";
(a.4.2) *condições de fórmulas (CF):* identifica os elementos dessas listas de condições.
(b) *cláusulas semânticas (CSM):* constrói as cláusulas do predicado "imposs" mediante diálogo com o ABD.
(c) *entradas no dicionário (ED):* gera entradas do dicionários de relações [CM] para os predicados do módulo de aplicação.
(d) *exclui operações (EOP):* exclui do módulo de aplicação as cláusulas de operações escondidas, introduzindo cláusulas semanticamente equivalentes, definidas sobre as operações que chamam essas operações escondidas.
(e) *lista cláusulas (LC):* exibe o módulo de aplicação construído.
(f) *edita o módulo de aplicação (EMA):* nesta, o ABD, fazendo uso das facilidades de edição do micro-PROLOG, substitui convenientemente constantes por variáveis nas cláusulas dos predicados do módulo de aplicação.
(g) *remove cláusulas (RC):* responsável pela exclusão do programa, após retornar do gerador de planos, das cláusulas do predicados do módulo de aplicação.

O IGP, após a execução da função EMA, ativa o GPL, que procura encontrar seqüências (alternativas) de aplicações de operações (ações) capazes de conduzir o banco de dados aos estados caracterizados pelas conjunções de fatos do predicado "imposs". Caso seja possível identificar tais seqüências, há pelo menos uma operação no esquema com pré-condições não suficientes para a preservação das restrições de integridade, situação esta que é comunicada pelo IGP ao ABD.

4.3 A Ferramenta de Reprojeto

A figura 3 (ver anexo 2) apresenta os componentes da ferramenta de reprojeto e o fluxo de controle, setas simples e duplas tendo o mesmo significado que na ferramenta de projeto e a seta tracejada indicando que as cláusulas do dicionário de alterações são usadas para produzir a saída final, a saber, a nova versão do dicionário.

Alguns dos componentes da ferramenta de reprojeto são também usados na fase de projeto. Os componentes exclusivos da fase de reprojeto têm as funções seguintes:

(a) o *gerenciador de reprojeto (GRP)* coordena a realização de mudanças nas definições de objetos, a introdução de novos objetos e gerencia a propagação da alteração pretendida para os objetos posteriores segundo a ordem de definição;
(b) o *dicionário de leis de propagação (DLP)* contém uma implementação das leis de propagação da metodologia;
(c) o *dicionário de alterações (DA)* contém o registro das alterações que tiveram lugar na sessão de reprojeto.

O fluxo de controle é comentado brevemente no que segue.

O GRP ativa o AD para a especificação da alteração pretendida pelo ABD; ativa o DLP para verificar o efeito desta alteração sobre os conjuntos de objetos posteriores segundo a ordem de definição; ativa ainda o AS para a geração das árvores de análise para fórmulas e programas do esquema, estas sendo utilizadas no DLP.

Pela mesmas razões que na ferramenta de projeto, o AD chama o DLI, o DLI ativa o AS e o AD.

O DLP verifica, em resumo, a satisfação dos requisitos de projeto pelos objetos posteriores (segundo a ordem de definição) àquele que sofreu alguma alteração. Para tal finalidade ativa o AD (verificação dos requisitos semânticos) e o DLI (verificação dos requisitos sintáticos). Para verificação dos requisitos semânticos há a aplicação de testes pelo DLI com o objetivo de reduzir o número de questões colocadas ao ABD.

Da comunicação com o ABD resultam as cláusulas de registros de alterações, armazenadas no DA. As cláusulas deste são usadas para produzir a saída final: uma nova versão do esquema modular. As alterações levadas a efeito estruturam as cláusulas de definição de um predicado designado por "was-applied", que toma parte então no programa lógico que constitui a ferramenta.

O ABD invoca a ferramenta de reprojeto para mudar a definição de algum objeto especificando:

"change (nome do objeto)";

caso deseje inserir um novo objeto no esquema, deve então ativá-la especificando:

"new (classe do objeto)s of (nome do módulo)"

as constantes identificando a (classe do objeto) sendo obtidas do conjunto (scheme, constraint, operation), os elementos deste correspondendo aos conceitos relacionais de relação, restrição de integridade e operação, respectivamente.

5. EXEMPLOS DE UTILIZAÇÃO DO PROTÓTIPO

No trabalho de [Se] são dados exemplos de uso do protótipo na especificação de um pequeno banco de dados, cuja estrutura é mostrada na figura 4 (ver anexo 2). Como se observa, há a descrição, neste banco de dados, das entidades *PRODUTO* e *DEPÓSITO*, do relacionamento *REMESSA* entre PRODUTO e DEPÓSITO, e ainda de uma visão *ENTREGA*, construída sobre REMESSA.

A figura 5 mostra o uso do construtor de módulo PRIMITIVO da metodologia para a especificação dos

objetos do módulo PRODUTO [Se]. Nesta, a relação PROD tem como atributos o código (pnum) e o nome (nome) do produto; a restrição de integridade UNICO-N assinala o código como chave de PROD; a operação INCLUAPROD é de inserção, contendo um teste (estrutura condicional) para garantir a não violação de UNICO-N por final, a operação EXCLUAPROD é de retirada de tuplas de PROD pelo valor de chave.

São apresentados, nas seções seguintes, exemplos de uso do protótipo nas fases de projeto e reprojeto do esquema modular. Nestes, as respostas dadas pelo ABD aparecem destacadas.

5.1 Fase de Projeto

Para a introdução dos objetos do módulo PRODUTO, a ferramenta estabelece um diálogo com o ABD, sendo mostrada, como exemplo, na figura 6, a comunicação levada a efeito para a especificação da restrição de integridade UNICO-N.

No atual estágio do protótipo os requisitos semânticos são verificados mediante consulta ao ABD e confiando-se em suas respostas. Mostra-se como exemplo, na figura 7, o modo de verificação do requisito semântico P7 [CFT, Se], requisito este estabelecendo que INCLUAPROD deve ser invariante com relação a UNICO-N.

Para ilustrar o uso do GPL [Se], supor que o ABD tenha introduzido a operação INCLUAPROD sem o teste de garantia, ou seja:

INCLUAPROD ((p,n): insert (p,n) into PROD).

Da ativação do IGP pelo GPR resulta, incialmente, o estabelecimento de um diálogo com o ABD para a introdução de uma conjunção de fatos caracterizando um estado de violação da restrição de integridade UNICO-N, a qual poderia ser:

((PROD p n) (PROD p m)).

O IGP constrói então o módulo de aplicação para o GPL mostrado na figura 8.

Da ativação do GPL resultam planos (alternativos) capazes de produzir uma inconsistência no banco de dados com relação à restrição de integridade UNICO-N. A figura 9 apresenta a comunicação do estado de violação no banco de dados da ferramenta para o ABD. Observe-se que, caso não houvesse o erro, a cláusula de definição do predicado "conditions" para INCLUAPROD seria:

(INCLUAPROD p n) conditions ((not PROD p m)).

Um dos aspectos de importância da implementação é a apresentação de mensagens de erro apropriadas quando da violação de algum dos requisitos da metodologia. Supor, como exemplo, que para a questão de verificação do requisito P7, mostrada anteriormente, o ABD tenha respondido "yes" ao invés de "no". Neste caso haveria a violação do requisito P7 e a ferramenta mostraria a mensagem da figura 10.

5.2 Fase de Reprojeto

Supor, na discussão seguinte, que há somente o módulo PRODUTO no esquema modular.

Como na fase de projeto, estabelece-se, na fase de reprojeto, um diálogo entre a ferramenta de reprojeto e o ABD para a especificação da alteração desejada e a sua propagação para os objetos posteriores segundo a ordem de definição. A figura 11 mostra a especificação de uma alteração para a relação PROD, com a introdução de um domínio adicional, a saber, peso.

Como conseqüência de uma alteração efetuada pelo ABD, os objetos posteriores, segundo a ordem de definição, àquele que sofreu a alteração podem ter suas definições tornadas inválidas, pelo que também podem necessitar, por sua vez, de alguma alteração. Para esta verificação são utilizadas, como se disse, as leis de propagação. A figura 12 mostra a implementação de uma destas leis, nesta a ferramenta consultando o ABD sobre a validade da restrição de integridade UNICO-N, dado que esta faz referência à relação PROD, que sofreu alteração.

A ferramenta verifica, a seguir, o efeito da alteração introduzida sobre a definição de INCLUAPROD, perguntando ao ABD:

(a) se INCLUAPROD faz referência de modo correto à relação PROD, que foi alterada e
(b) se INCLUAPROD, com relação ao requisito semântico P7, continua invariante relativamente a UNICO-N, que também sofreu alteração, como mostrado na figura 12.

Questões similares são então formuladas para a operação EXCLUAPROD.

Conclui-se a sessão de reprojeto com a ferramenta consultando o ABD sobre a necessidade de introdução de cláusulas de garantia apropriadas, justificando-se isto no fato de que tanto a restrição de integridade como as operações foram previamente alteradas.

6. CONCLUSÕES

Observa-se que sistemas especialistas são muito adequados para o projeto de bancos de dados, introduzindo um novo estilo no modo de condução de diálogos, correção de inconsistências e justificação de resultados [BGM].

O enfoque do processo de projeto com programação lógica faz com que uma aplicação de bancos de dados consista de um único programa lógico, constituído de uma coleção uniforme de cláusulas descrevendo o dicionário de dados, as leis de integridade, as leis de propagação, etc., isto corroborando a argumentação de [CMo], segundo a qual a programação lógica oferece uma representação uniforme para os mais importantes conceitos de bancos de dados.

Como assinalado em [FCT], a potência e a flexibilidade da linguagem PROLOG a tornam conveniente para implementação da metodologia. Embora tudo na ferramenta seja uniformemente expresso por sentenças do PROLOG, a orientação difere em cada parte:

- o DLI e o DLP são *declarativos;*
- o GPR e o GRP são *procedurais;*
- o AS é *orientado por regras de produção;*
- o DD e o DA são *factuais,* sendo tratados como dados;
- o IGP e o GPL são *procedurais.*

O sistema de programas desenvolvido, considerado como um ambiente de especificação de aplicações de bancos de dados, pode ser visto, respeitadas as proporções de escopo e extensão, como um ambiente do tipo CASE ("Computer-aided softwere engineering") [Ca], uma tecnologia para melhorar a produtividade no desenvolvimento de sistemas, os componentes do protótipo podendo ser associados a com-

ponentes de um "kit" CASE de ferramentas para o desenvolvimento de sistemas.

Como assinalado em [FCT], um aspecto crítico do protótipo é a distribuição da carga de trabalho entre o usuário e a ferramenta especialista. Embora as regras implementadas coordenem com alguma extensão os esforços de projeto e reprojeto do usuário, muito permanece em aberto para este, não apenas em ternos das alternativas válidas (o que é desejável) como também de cometer erros (o que não é desejável), principalmente no que se refere à implementação dos requisitos semânticos. Como se pode observar, obteve-se com este trabalho algum progresso nos aspectos relacionados com a verificação deste tipo de requisito mediante a integração da ferramenta de projeto com o protótipo de geração de planos.

Na implementação atual do protótipo, efetivou-se a integração da ferramenta de projeto com o protótipo de geração de planos, para verificar/testar a suficiência das pré-condições das operações. Como pesquisa futura, para se obter uma ferramenta completa com relação aos objetivos atuais, deve-se executar as etapas de:

(a) integrar a ferramenta de reprojeto com o gerador de planos;
(b) fazer uso do gerador de planos para verificar a correção da tradução de operações (que consta dos substitutivos) de módulos externos.

REFERÊNCIAS BIBLIOGRÁFICAS

[BGM] M. Bouzeghoub, G. Gardarin, E. Metais. "Database Design Tools; An Expert System Approach". Proc. of the 11th Int l. Conf. on Very Large Data Bases, Stockholm, Sweden (1985), 436-447.

[BZ] M. L. Brodie, S. N. Zilles (eds.). Proc. of the Workshop on Data Abstraction, Databases and Conceptual Modelling. SIGMOD Record, vol. 11, n.2 (1981)

[Ca] A. F. Case. "Computer-aided softwere engineering (CASE): Technology for improving software development productivity". ACM SIGBDP vol. 17, n.1 (1985).

[CB] M.A. Casanova, P. A. Bernstein. A Formal System for Reasoning about Programs Accessing a Relational Data Base". ACM Trans. on Programming Languages and Systems 2:3 (1980), 386-414.

[Ce] S. Ceri *Methodology and tools for database design*. North-Holland (1985).

[CFT] M. A. Casanova, A.L. Furtado, L. Tucherman. "A Softwere Tools for Modular Database Design". Centro Científico IBM Brasil, Relatório Técnico CCB033 (1985)

[CM] K. L. Clark, F. G. McCabe. *micro-PROLOG: programmning in logic,* Prentice-Hall (1984).

[CMo] M. A. Casanova. C.M.O. Moura. "An Invitation to the design of Database applications in Logic Programming". Centro Científico IBM Brasil, Relatório Técnico CCBO32 (1985).

[Da] C.J. Date. An *Introduction to Database Systems*. Addison Wesley Publishing Company (1976).

[FC] A. L. Furtado, M. A. Casanova. "Updating Relational Views", em *Query Processing in Database System*. W. Kim, D. S. Reiner, D. S. Batory (eds.), Springer-Verlag, (1985), 127-142.

[FCT] A. L. Furtado, M. A. Casanova, L. Tucherman. "A Framework for Design/Redesign Experts". Proc. of the 1st. Int 1 Conf. on Expert Database Systems, South-Carolina, Abril (1986), 313-328.

[FM] A. L. Furtado, C. M. O. Moura. "Expert helpers to data-based information systems". Proc. of the First International Workshop on Expert Database systems, (1984), 298-313.

[HS] P. Hammond, M. Sergot. *apes: augmented PROLOG for expert systems = reference manual.* Logic Based Systems Ltd. (1984).

[Ko] R. Kowalski. *Logic for problem solving.* North Holand (1979).

[LM] P. C. Lockemann, H. C. Mayr. "Information system design: techniques and software support" em *Information Processing 86*. H. J. Kugler (ed.), North-Holland (1986) 617-634.

[Ni] N. J. Nilsson. *Principles of artificial intelligence".* Springer-Verlag (1982).

[Se] G. J. Sena. *Implementação de protótipo de ferramenta para projeto/reprojeto de bancos de dados.* Dissertação de Mestrado, Departamento de Informática, PUC/RJ, Rio de Janeiro-RJ (1987).

[FV] P. A. S. Veloso, A. L. Furtado. "Towards simpler and yet complete formal specifications", em *information System: Theoretical and Formal Aspects.* A Sernadas, J. Bubenko, A. Olive (eds.), North-Holland (1985), 175-189.

[Wa] D. H. D. Warren. "WARPLAN: a system for generating plans". Memo 76. University of Edinburgh (1974).

ANEXO I

Figura 1 : Componentes e fluxo de controle na ferramenta de projeto

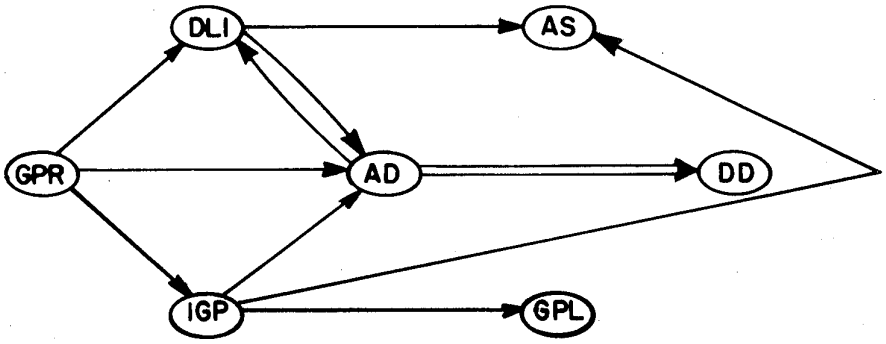
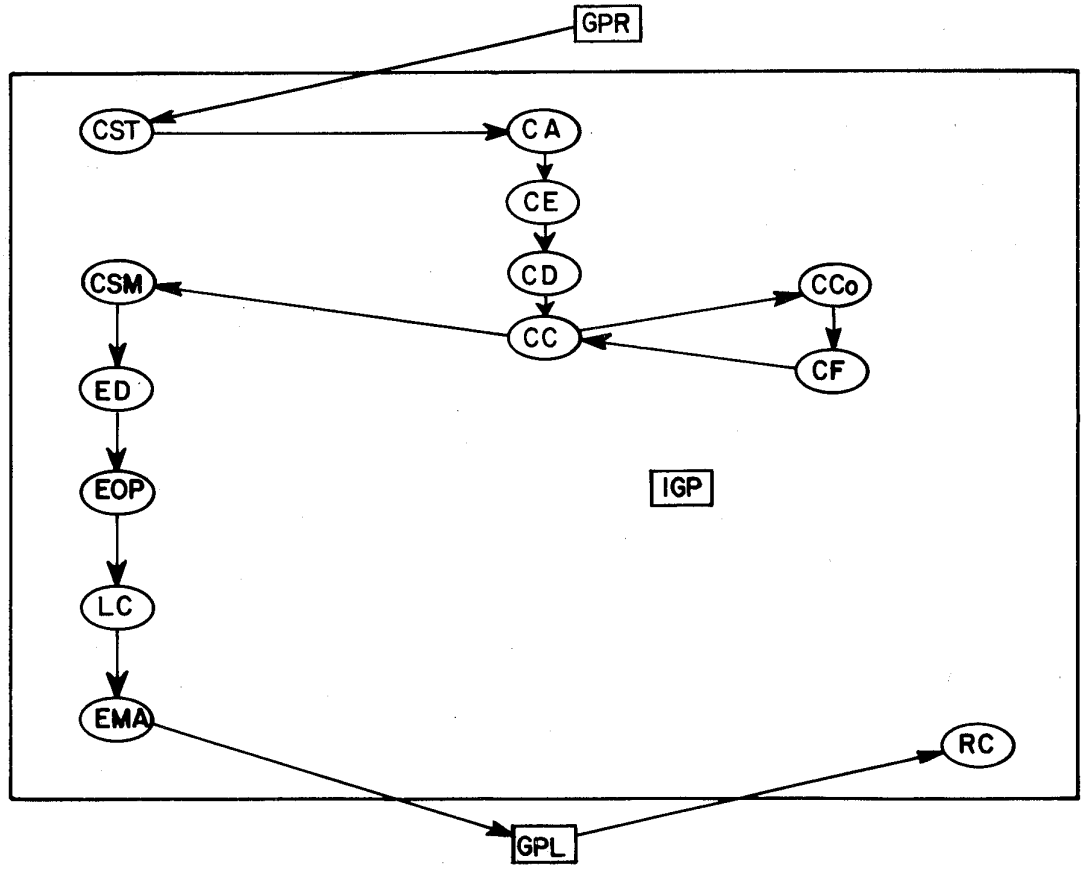


Figura 2 : Estrutura do programa de ligação da ferramenta de projeto com o gerador de planos



ANEXO 2

Figura 3 : Componentes e fluxo de controle na ferramenta de reprojeto

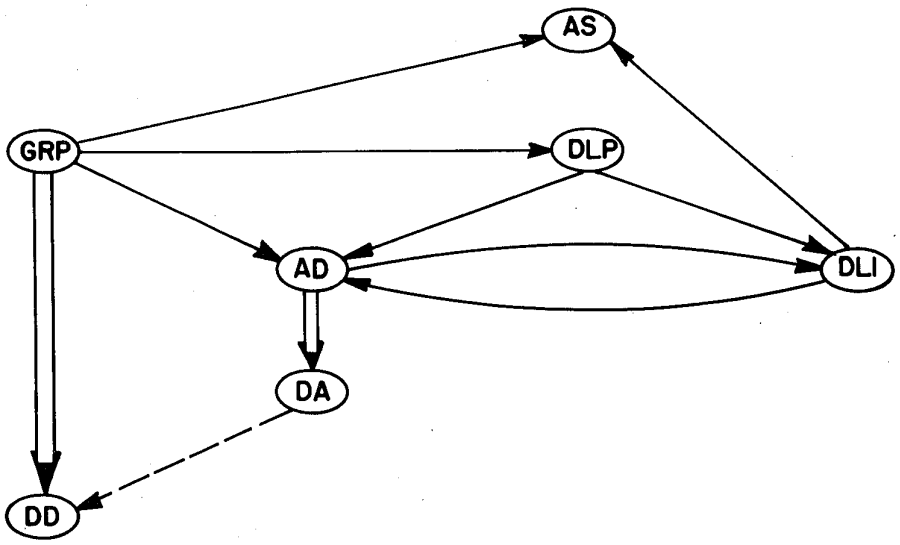
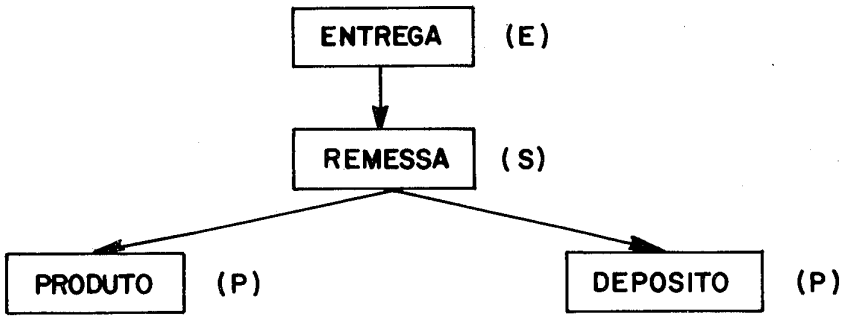


Figura 4 : Estrutura de um pequeno banco de dados



P : módulo primitivo S: módulo subordinante E: módulo externo

Figura 5: Uso do construtor de módulo primitivo para especificação do módulo PRODUTO

```
módulo PRODUTO
     relações
          PROD [pnum,nome];
     restrições de integridade
          UNICO_N: ∀p∀n∀m(PROD(p,n) & PROD(p,m) => m=n);
     operações
          INCLUAPROD (p,n): if -∃m(PROD(p,m)) then insert (p,n)
                                                    into PROD;
          EXCLUAPROD ((p) : delete PROD(x,y) where x=p;
     cláusulas de garantia
          INCLUAPROD garante UNICO_N
fim módulo.
```

Figura 6: Ilustração do diálogo estabelecido entre a ferramenta de projeto e o ABD: introdução de UNICO_N

```
* constraints --- (<name> (<definition>)) ?
     Answer is (UNICO_N (\/p\/n\/m(PROD(p,n) & PROD(p,m) => m=n)))
     Answer is enough.
```

Figura 7: Ilustração do modo de implementação de requisitos semânticos: requisito P7 para INCLUAPROD e UNICO_N

```
* operation INCLUAPROD of module PRODUTO:
* ((p,n) : if -"]"m(PROD(p,m)) then insert (p,n) into PROD)

* constraint UNICO_N of module PRODUTO:
* (\/p\/n\/m(PROD(p,n) & PROD(p,m) "=>" m=n))

* if used directly, may INCLUAPROD violate UNICO_N ? no.
```

Figura 8: Módulo de aplicação para o protótipo de geração de planos: módulo PRODUTO

```
(INCLUAPROD p n) operation-on (pnum nome)
(EXCLUAPROD p) operation-on (pnum)

(PROD p n) added (INCLUAPROD p n)
(PROD p y) deleted (EXCLUAPROD p)

(INCLUAPROD p n) conditions ()
(EXCLUAPROD p) conditions ((PROD p y) (equal p p))

imposs ((PROD p n) (PROD p m))
```

Figura 9: Ilustração do modo de comunicação de estado de inconsistência no banco de dados: violação de UNICO_N

```
=== ">" VIOLATION-OF-CONSTRAINTS-ANALYSIS "<" ===

* inconsitent state
(PROD p n) (PROD p m)

* may be reached by the sequence (s) of function applications:
(s0 (INCLUAPROD p n) (INCLUAPROD p m))
(s0 (INCLUAPROD p m) (INCLUAPROD p n))

* you may have to reespecify the set of operations of module
PRODUTO
```

Figura 10: Ilustração do tipo de mensagem produzida quando da violação dos requisitos: violação do requisito P7

```
*** SEMANTICAL error ***
*** P7 requiremente violated ***
* constraint UNICO_N must be an invariant of operation INCLUAPROD
```

Figura 11: Ilustração do modo de especificação de uma alteração para o esquema modular: modificação da relação PROD

```
change PROD

* scheme PROD
 (<change> (<domains>))?
     Answer is (MOD (pnum,nome,peso))
```

Figura 12: Ilustração do modo de propagação da alteração pretendida para objetos posteriores: verificação de UNICO_N

```
=== ">" (NEEDS-MANUAL-CHANGE) "<" ===

* constraint UNICO_N of module PRODUTO
(\/p\/n\/m(PROD(p,n) & PROD(p,m) "=>" m=n))

* constraint UNICO_N of module PRODUTO
(\/p\/n\/m(PROD(p,n) & PROD(p,m) "=>" m=n))

* you have to do additional changes in the constraint UNICO_N
because it has not a valid definition at the modular schema:

* delete or modify constraint UNICO_N in module PRODUTO since the
definition of UNICO_N references a scheme PROD (of module PRODUTO)
that was modified

* constraint UNICO_N
 (<change> (<definition>))?
     Answer is (MOD (\/p\/n\/m\/w\/v(PROD(p,n,w) & PROD(p,m,v) =>
                                    m=n & w=v)))
```