UNIVERSIDADE FEDERAL DA PARAÍBA

CENTRO DE CIÊNCIAS E TECNOLOGIA

Curso de Mestrado em Sistemas e Computação

UMA FERRAMENTA PARA A DEFINIÇÃO E VERIFICAÇÃO DE REGRAS DE INTEGRIDADE EM BANCO DE DADOS.

Gabriela Ferreira Drumond

Campina Grande - Pb

Fevereiro - 1990

Gabriela Ferreira Drumond

UMA FERRAMENTA PARA A DEFINIÇÃO E VERIFICAÇÃO DE REGRAS DE INTEGRIDADE EM BANCO DE DADOS.

Area de Concentração : Ciência da Computação

Marcus da Costa Sampaio

Orientador

Campina Grande - Pb

Fevereiro - 1990

D795f Drumond, Gabriela Ferreira
Uma ferramenta para a definicao e verificacao de regras de integridade em banco de dados / Gabriela Ferreira Drumond. - Campina Grande, 1990.
68 f.

1. Dissertacao 2. Banco de Dados 3. Sistemas e Computacao I. Universidade Federal da Paraiba (PB) II. Marcus Costa Sampaio, Dr. (orientador) III. Título

CDU 004.65(043)

UMA FERRAMENTA PARA DEFINIÇAO E VERIFICAÇAO DE REGRAS DE INTEGRIDADE EM BANCO DE DADOS

GABRIELA FERREIRA DRUMOND

DISSERTAÇAO APROVADA EM 20.02.1990

MARCUS COSTA SAMPAIO
Orientador

SONIA SCHEOTMAN SETTE
Componente da Banca

ULRICH SCHIEL
Componente da Banca

JOSELUCE DE FARIAS CUNHA
Componente da Banca

Campina Grande - Pb.
Fevereiro - 1990

SUMÁRIO

1 - INTRODUÇÃO 1

2 - CONSIDERAÇÕES SOBRE CONSISTÊNCIA EM BANCO DE DADOS 4

3 - ARQUITETURA E COMPONENTES DO SISTEMA 7

3.1 - Arquitetura do Sistema 7

3.2 - Componentes do Sistema 8

4 - A LINGUAGEM DE DEFINIÇÃO DE REGRAS DE INTEGRIDADE 10

4.1 - A Versão em Portugues 10

4.1.1 - A Especificação da NRI 11

4.2 - A Versão em Lógica 13

4.2.1 - A Especificação da LRI 13

5 - O CATALOGADOR 16

5.1 - Execução do Sistema de Catalogação 16

5.1.1 - Catalogando uma Regra em NRI 21

5.1.2 - Tradutor 1 21

5.1.2.1 - O Analisador Léxico 23

5.1.2.2 - O Analisador Sintático 25

5.1.2.3 - O Mapeador 26

5.1.3 - Catalogando uma Regra em LRI 34

5.2 - O Armazenamento da Regra 37

5.2.1 - Armazenamento de Regras Tipo Definição .. 37

5.2.2 - Armazenamento de Regras Para Atualizações . 39

6 - O GERADOR 42

6.1 - Módulos Competentes do Gerador 43

6.2 - Tradutor 2 44

6.2.1 - Operadores Relacionais 46

6.2.2 - O Mapeador Lógica --> Álgebra Relacional . 47

6.2.2.1 - Classificando os Relacionamentos 44

6.2.2.2 - O Mapeamento Lógica ---> Álgebra 49

6.2.2.3 - Geração da Condição 54

6.3 - Tradutor 3 55

6.3.1 - As Tabelas Fornecidas por Tradutor 2 55

6.3.2 - O Mapeamento Para o SQl 57

6.4 - O Gerador de Rotinas 59

6.5 - O Programa de Crítica 60

7 - CONCLUSÕES 63

8 - BIBLIOGRAFIA 67

LISTA DE ILUSTRAÇÕES

3.1 - Arquitetura do Sistema 7

3.2 - Módulos do Sistema 9

5.1 - Menu 1 16

5.2 - Menu 2 17

5.3 - Menu 3 18

5.4 - Menu 4 18

5.5 - Menu 5 18

5.6 - Menu 6 19

5.7 - Menu 7 20

5.8 - Menu 8 20

5.9 - Menu 9 21

5.10 - Tradutor 1 22

5.11 - Tabela de Simbolos 23

5.12 - Tabela de Sinônimos 24

5.13 - O Analisador para a Regra em LRI 35

5.14 - Tabelas Gerenciadas pelo Analisador 36

5.15 - Estrutura de Dados da Tabela de Definições 37

5.16 - Catálogo de RIs 40

6.1 - O Gerador 42

6.2 - Módulos do Gerador 43

6.3 - Tradutor 2 45

6.4 - Tradutor 3 55

6.5 - As Tabelas Fornecidas por Tradutor 2 56

1. INTRODUÇÃO

Poucos Gerenciadores de Banco de Dados (GBDs), comercialmente disponíveis, oferecem linguagens de definição de Regras de Integridade (RI) [1]. A razão principal é que a verificação on-line de RIs é quase sempre uma operação onerosa. Resolve-se, então, o problema da integridade de um Banco de Dados (BD), através do uso de Programas de Crítica, escritos em uma linguagem hospedeira de comandos do GBD particular. Tais programas verificam um conjunto de RIs para uma operação específica de atualização antes da mesma, e, em caso de violação da integridade do BD, uma mensagem apropriada é transmitida e a operação é rejeitada; em caso contrário, a operação é realizada. Estes programas são executados em modo batch, a partir de um arquivo de transações.

Nosso trabalho é a descrição de uma ferramenta desenvolvida com o objetivo de gerar automaticamente programas de crítica.

O sistema, por nós desenvolvido, permite a definição de RIs, usando uma linguagem em duas versões, uma versão em portugues e uma versão em lógica, permitindo ainda a definição de novos ítens de dado (Definições), a partir de ítens de dado reais do BD.

As RIs fornecidas em portugues são mapeadas para a linguagem em lógica e, juntamente com as RIs fornecidas em lógica, são armazenadas em um Catálogo de RIs.

As RIs armazenadas no Catálogo, de acordo com o tipo da transação (Inclusão, Exclusão, Modificação) e o local (estamos considerando somente GBDs relacionais: o local será uma única

relação do esquema do BD, embora as RIs relativas a uma determinada relação possam envolver várias relações, são mapeadas , então, para a Álgebra Relacional . Este passo intermediário aumenta a portabilidade do nosso sistema : as RIs em AR podem ser facilmente mapeadas para instruções em qualquer linguagem de GBDs relacionais (SQL, QUEL, etc). No nosso caso, esse mapeamento é feito para a SQL [2].

O passo seguinte consiste na geração de rotinas compostas por instruções de decisão, instruções de consulta e a instrução de atualização referentes a cada par (tipo , local) da transação. De posse dessas rotinas, um programa de crítica em Pascal e SQL é gerado.

Um programa de crítica, para um determinado conjunto de RIs, é finalmente compilado, e só será alterado caso haja mudanças no Catálogo de RIs (inclusão ou exclusão de RI).

O capítulo 2 apresenta algumas considerações sobre Restrições de Integridade em Banco de Dados.

O capítulo 3 detalha os componentes da arquitetura do nosso sistema.

O capítulo 4 apresenta a Linguagem de Definição de Regras (Definições e RIs) nas suas duas versões.

No capítulo 5 é apresentado o Catalogador, responsável pelo processamento e armazenamento de Regras fornecidas nas duas versões da linguagem de definição de regras.

O capítulo 6 descreve o Gerador, responsável pela geração do programa de crítica e pelos passos intermediários entre o fornecimento da RI e a sua transformação em uma instrução de consulta, forma em que ela será utilizada no programa de

crítica.

O capítulo 7 apresenta as conclusões do nosso trabalho e possíveis aperfeiçoamentos que poderão vir a ser feitos.

2 - CONSIDERAÇÕES SOBRE CONSISTENCIA EM BANCO DE DADOS

No mundo da informática, Bancos de Dados são cada vez mais utilizados para armazenar dados relativos ao mundo real; destas informações é exigido que elas reflitam exatamente o universo a que se propõem a modelar. Para que isto aconteça, o BD deve ser consistente e não estar sujeito a alterações que possam destruir a integridade dos dados armazenados.

A importância da consistência de BDs pode ser sentida no 3o Simpósio Brasileiro de Banco de Dados - 1988 no qual vinte por cento dos artigos publicados tratavam de Regras de Integridade em BDs e de formas de garantir as mesmas sem onerar o sistema como um todo. Nestes artigos [10] [11] [12], definem-se duas tendências principais para solucionar o problema: existência de um Subsistema de Regras de Integridade e Programas de Crítica definidos e implementados pelo usuário.

Um Subsistema de Regras de Integridade possui normalmente uma linguagem de definição de restrições sobre os dados e mecanismos de verificação das regras de integridade. O motivo pelo qual poucos GBDs possuem um Subsistema de Integridade, e, mesmo na existência do mesmo, este ter como característica cobrir uma pequena gama de restrições, é a degradação da eficiência do sistema; existirão acessos adicionais ao BD além de consultas ao Catálogo de Regras de Integridade. A principal vantagem neste caso seria a de não haver possibilidade de acesso ao BD sem o acionamento do subsistema de regras de integridade, evitando que o mesmo sofra alterações na sua consistência.

No segundo caminho, os programas de crítica funcionam como interfaces para o BD antes da atualização do mesmo. Não demandam acessos adicionais nem consultas a catálogos de regras de integridade. As desvantagens são o tempo e os recursos gastos na confecção destes programas e a não garantia de que o BD possa ser atualizado através de outros caminhos, pondo em risco a sua consistência. Dependendo da quantidade de aplicativos que usem o BD e do número de "donos" destes aplicativos e dependendo ainda do grau de interação entre estes "donos", corre-se o risco de haver uma grande quantidade de programas de consistência para cada um destes aplicativos.

O nosso trabalho, cujas idéias iniciais estão contidas em [11], propõe um programa de crítica único para qualquer aplicativo que tenha acesso ao BD, gerado automaticamente e compilado uma única vez. O programa é baseado em um conjunto determinado de Regras de Integridade e só sofrerá modificações se este conjunto sofrer alterações. Com isto o trabalho não possui as desvantagens inerentes a este tipo de solução:

. A alta demanda de tempo e mão-de-obra não existe, já que o programa é gerado automaticamente e compilado uma única vez para um determinado conjunto de Regras de Integridade;

. O problema de acessos indevidos pode ser solucionado com o acionamento automático do programa gerado, para qualquer atualização do BD;

. O conjunto de Regras de Integridade sendo definido em conjunto por todos os usuários cobriria toda e qualquer atualização indevida.

No nosso sistema, as Regras de Integridade são verificadas independentemente de sua classificação (Integridade de Entidade, Referencial, Domínio, etc). Não existem procedimentos específicos, dentro do programa gerado, para cada tipo de restrição de integridade.

3. ARQUITETURA E COMPONENTES DO SISTEMA

3.1 ARQUITETURA DO SISTEMA

A figura 3.1 apresenta os principais elementos que compoẽm a arquitetura do nosso sistema :

- CATÁLOGO DE RI's : Contém as RIs que devem ser obedecidas por uma transação qualquer, para que , se efetivada, a integridade do BD seja preservada ;

- TRANSAÇÃO DE ATUALIZAÇÃO : Pode ser uma inclusão de dados ou uma modificação de dados já existentes no BD, ou então uma exclusão de dados do BD;

- BD OPERACIONAL : É o BD armazenado, sobre o qual incidirão as atualizações permitidas pelas RIs.

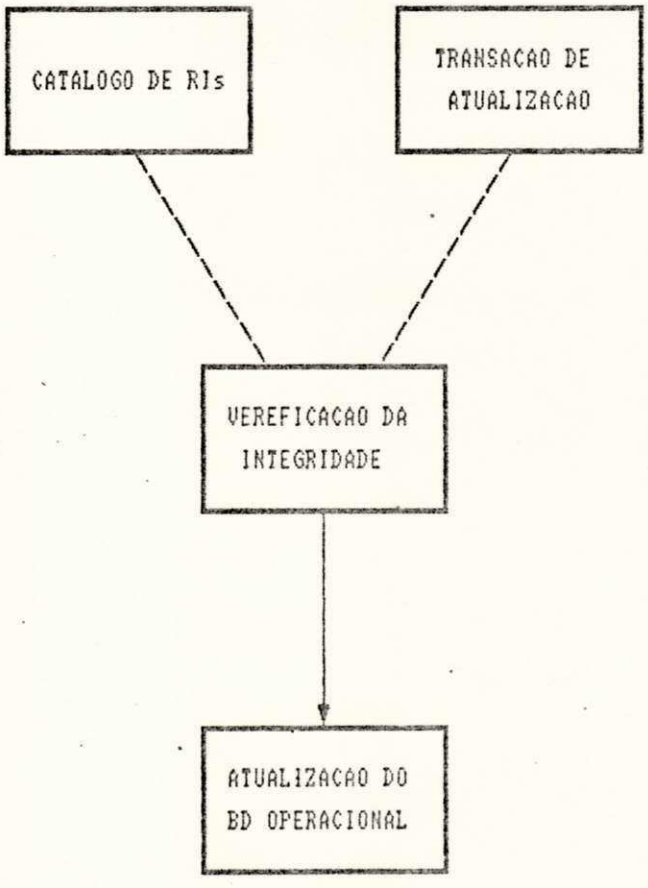

FIGURA 2.1 - ARQUITETURA DO SISTEMA

3.2 COMPONENTES DO SISTEMA

De acordo com a figura 3.2 , os módulos componentes do sistema são :

CATALOGADOR - Tem como entrada um conjunto de regras que podem ser de dois tipos : as RIs ou as Definições. Estas regras podem ser definidas usando a Linguagem em Lógica de Definição de Regras (LRI) ou usando a Linguagem Natural de Definição de Regras (NRI).

As RIs, independentemente da linguagem de definição utilizada, devem vir acompanhadas do tipo e do local, que é o arquivo (relação) sobre o qual ocorrerá a atualização. Consideraremos atualizações de tipos diferentes em um mesmo local, como operações independentes - isto implica em que, para cada atualização, deverá haver a verificação das RIs de seu tipo e local.

A saida do Catalogador é o Catálogo de RIs em LRI, organizado por tipo e local. O acesso ao Catálogo de RIs é feito através de um Índice, cuja chave de acesso é determinada pela concatenação do tipo e do local de uma RI. Cada entrada no Índice é uma cabeça de uma lista de RIs para um particular tipo i e local j.

No caso da regra ser uma definição ela é armazenada pelo Catalogador na tabela de definições .

GERADOR - tem como entrada o Catálogo e o Esquema do BD Operacional. A partir dos dados contidos nestes arquivos, o Programa de Crítica é gerado, objetivo final do sistema.

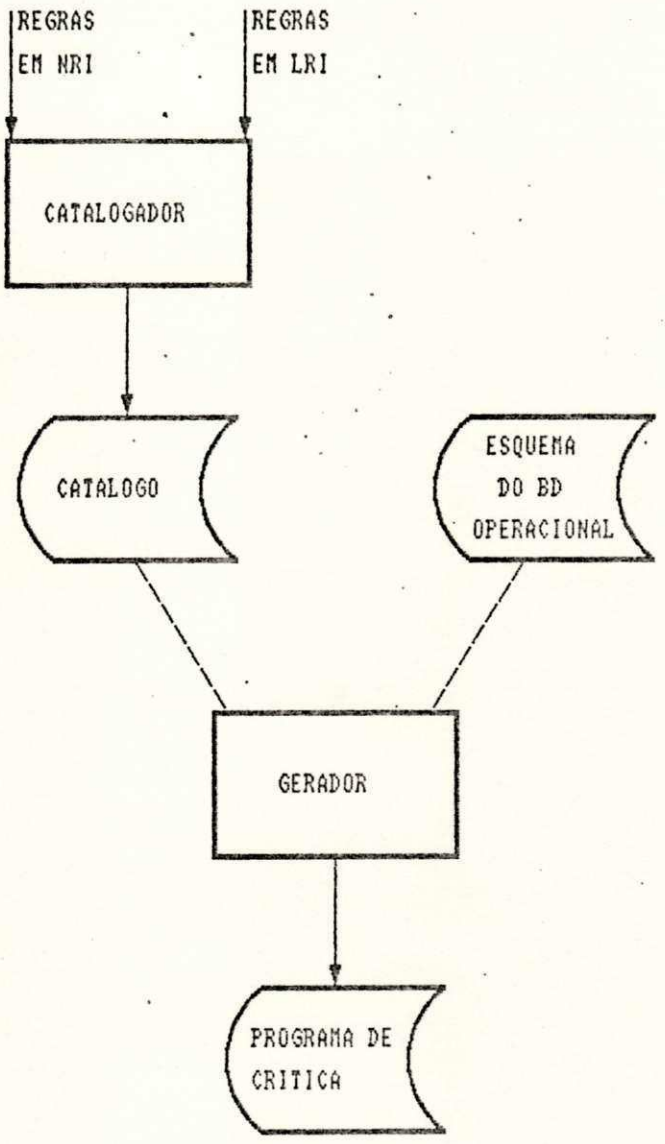

FIGURA 3.1 - MODULOS DO SISTEMA

4. A LINGUAGEM DE DEFINIÇÃO DE REGRAS

O nosso trabalho propõe uma linguagem de definição de regras em duas versões :

- uma versão em português (NRI), cujo compilador reconhece locuções e palavras sinônimas.

- uma versao em Lógica (LRI). Uma RI em LRI é, basicamente, uma cláusula de Horn [4,5], isto é, uma cabeça constando do predicado OP e dos argumentos tipo e local e o corpo que é uma conjunção de proposições Bi que têm sempre que ser verdadeiras (OP(tipo,local) :- B1, B2, ... ,Bn, onde ":-" é lido como "se"). Já para uma definição, a cabeça é modificada : o predicado é o nome da definição e os argumentos são atributos da mesma.

Para exemplificar as gramáticas de ambas as versões utilizadas, consideraremos os símbolos terminais, relações e atributos, como as relações e atributos pertencentes ao Esquema Relacional [6] abaixo :

EMP(Matrícula,Nome,Endereço,Salário,Depto,Data_Admissão,Certi)

DEP(Depto,Nome,Chefe)

PROJ(cod,Gerente,Orçamento,Depto)

Consideramos os nomes das relações e seus atributos como auto-explicativos a menos do atributo Certi da relação EMP, corresponde ao número da certidão de casamento do empregado.

4.1 A VERSÃO EM PORTUGUÊS

Definimos esta linguagem (NRI) através de sua gramática [7] GN = (T, N, P, S), onde GN é o nome da gramática.,T é o conjunto dos símbolos terminais ; N é o conjunto dos símbolos

não terminais ; P é o conjunto (finito) de produções e S é o símbolo inicial.

A linguagem NRI será a linguagem gerada por GN e representada por : L(GN) = { W / W pertence T* e S ===> * W }, que pode ser traduzido para : a linguagem gerada pela gramática GN (NRI), é igual ao conjunto de palavras W, tal que W pertence ao conjunto dos símbolos terminais de GN, inclusive o vazio, e, partindo-se do símbolo inicial S, através de zero ou mais derivações, chega-se a W.

4.1.1 A ESPECIFICAÇÃO DA NRI

Daremos a seguir a sintaxe da NRI, através das Regras de Produção (P) da sua gramática :

⟨regra⟩ ::= ⟨definição⟩ / ⟨ri⟩

⟨definição⟩ ::= ⟨quantificador⟩ *⟨identificador⟩* ⟨pertinência⟩ ⟨condição⟩

⟨ri⟩ ::= ⟨quantificador⟩ ⟨relação⟩ ⟨pertinência⟩ ⟨atrib⟩/
⟨quantificador⟩ ⟨atrib⟩ ⟨pertinência⟩ ⟨atrib⟩/
⟨quantificador⟩ ⟨atrib⟩ ⟨pertinência⟩ ⟨condição⟩

⟨atrib⟩ ::= ⟨atributo⟩/⟨atributo⟩ de [⟨quantificador⟩] ⟨relação⟩/
⟨atributo⟩ de [⟨quantificador⟩] ⟨definição⟩/
⟨atributo⟩ de [⟨quantificador⟩] ⟨atributo⟩ /
⟨definição⟩

⟨condição⟩ ::= ⟨atributo⟩ ⟨op⟩ ⟨atrib⟩ / ⟨atributo⟩ ⟨op⟩ ⟨valor⟩/
⟨atributo⟩ ⟨op⟩ ⟨constante⟩

⟨pertinência⟩ ::= ⟨pert_negativa⟩ / ⟨pert_positiva⟩

⟨quantificador⟩ ::= ⟨quant_positivo⟩ / ⟨quant_negativo⟩

⟨constante⟩ ::= "⟨nome⟩"

⟨nome⟩ ::= ⟨char⟩ (⟨char⟩)

⟨char⟩ ::= qualquer caractere ASCII diferente branco

⟨identificador⟩ ::= l(l/d) [_l(l/d)]

⟨valor⟩ ::= ⟨d⟩ (⟨d⟩) [,⟨d⟩ (⟨d⟩)]

⟨d⟩ ::= 0 / 1 /.../ 9

⟨l⟩ ::= a / b /.../ z /A / B /.../ Z

Os símbolos não terminais quant_positivo, quant_negativo, pert_positiva, pert_positiva, operador, são determinados de acordo com as necessidades do úsuario do sistema. Os símbolos utilizados por nós foram :

⟨quant_positivo⟩ ::= um / algum / qualquer / mais de um / uns / menos de um / outro / alguns / quaisquer

⟨quant_negativo⟩ ::= nenhum

⟨pert_negativa⟩ ::= não deve ter / não deve ser / não existe / não deve existir / não pode ser / não pode ter / não pode existir

⟨pert_positiva⟩ ::= deve ser / deve ter / existe / deve existir / pode ser / pode ter / pode existir

⟨operador⟩ ::= maior que, menor que, igual, diferente, maior ou igual, menor ou igual

Os símbolos não terminais ⟨relação⟩ e ⟨atributo⟩ tiveram seus valores determinados pelo Esquema do BD definido como exemplo no Ínicio deste capítulo :

⟨relação⟩ ::= EMP / DEP / PROJETO

⟨atributo⟩ ::= matrícula / nome / endereço / salário / depto / data_admissão / certi / chefe / proj / gerente / orçamento

Exemplos de RIs em linguagem natural:

1 - Inclusão na relação Empregado

Um empregado deve ter salario maior que 10000.

2 - Modificação na relação Departamento

Um chefe de departarmento deve ser empregado.

Quanto ao símbolo não terminal ⟨definição⟩, consideraremos que existem duas delas definidas : Grande_Projeto e Cônjuge, definidos como : Um Grande-Projeto deve ter orçamento maior que 10000000 Milhoões e Um Cônjuge deve ter Certi igual a Certi de outro empregado.

. Um grande-projeto deve ter orçamento maior que "100 Milhões"

. Um conjuge deve ter certi igual a certi igual ao de outro empregado.

Vale observar que os atributos e relações podem ser representados ainda por seus sinônimos que são inicialmente carregados em uma tabela, consistindo dos nomes das relações e atributos que poderão ser utilizados pelo úsuario em substituição aos nomes oficiais. Na nossa tabela de sinônimos, por exemplo, empregado e mat são sinônimos do atributo matrícula.

4.2 A VERSÃO EM LÓGICA

A LRI é definida por sua gramática GL, para a qual valem as observações feitas para a GN na seção 4.1.

4.2.1 ESPECIFICAÇÃO DA LRI

Daremos a seguir a sintaxe da LRI, através das regras de produção (P) de GL :

⟨regra⟩ ::= ⟨definição⟩ / ⟨ri⟩

⟨definição⟩ ::= ⟨identificador⟩ (⟨atrs⟩) ⟨--- ⟨proposição⟩
(⟨proposição⟩)

⟨ri⟩ ::= OP [⟨tipo_atualização⟩, ⟨local⟩] ⟨--- ⟨proposição⟩
(⟨proposição⟩).

⟨local⟩ ::= ⟨relação⟩

⟨proposição⟩ ::= ⟨relação⟩(⟨atrs⟩ / ⟨var⟩) (,⟨relação⟩(⟨atrs⟩ /
⟨var⟩)),⟨relacionamento⟩ (,⟨relacionamento⟩)

⟨atrs⟩ ::= ⟨atr_proj⟩ (,⟨atr_proj⟩)

⟨var⟩ ::= - (,-)

⟨atr_proj⟩ ::= Xdd / xdd

⟨relacionamento⟩ ::= ⟨atr_proj⟩ ⟨op⟩ ⟨atr_proj⟩ /
⟨atr_proj⟩ ⟨op⟩ ⟨constante⟩ /
⟨constante⟩ ⟨op⟩ ⟨atr_proj⟩

⟨identificador⟩ ::= l(l/d) [_l (l/d)]

⟨d⟩ ::= 0/1/ ... /9

⟨tipo_atualização⟩ ::= I/M/E

⟨l⟩ ::= a/b/...z/A/B/...Z

⟨op⟩ ::= < / > / <> / >= / <= / =

⟨constante⟩ ::= "⟨nome⟩"

⟨nome⟩ ::= ⟨char⟩ (⟨char⟩)

⟨char⟩ ::= qualquer caractere ASCII diferente de branco

O símbolo não terminal ⟨relação⟩, como na NRI, é determinado pelo Esquema do BD, e de acordo com o BD dado no início deste capítulo, teremos :

⟨relação⟩ ::= EMP / DEP / PROJ

Os exemplos em NRI vistos anteriormente seram representados em LRI :

1 - OP[I,EMP] <--- EMP(X01,-,-,X02,-,-,-),X02>"10000".

2 - OP[M,DEP] <--- DEP(X01,-,X02),EMP(X03,-,-,-,-,-,-),X02=X03.

Vale como observação que ambas as gramáticas são gramáticas do tipo livre de contexto.

5 . O CATALOGADOR

O Catalogador tem como função armazenar regras (Definições e RIs) no Catálogo de regras. O Catalogador tem como entrada, regras definidas em LRI e/ou regras definidas em português (NRI).

5.1. EXECUÇÃO DO SISTEMA DE CATALOGAÇÃO

Antes do início do fornecimento das regras, o usuário faz várias opções através de menus. O primeiro menu a ser exibido é o que trata das opções de execução do sistema (figura 5.1).

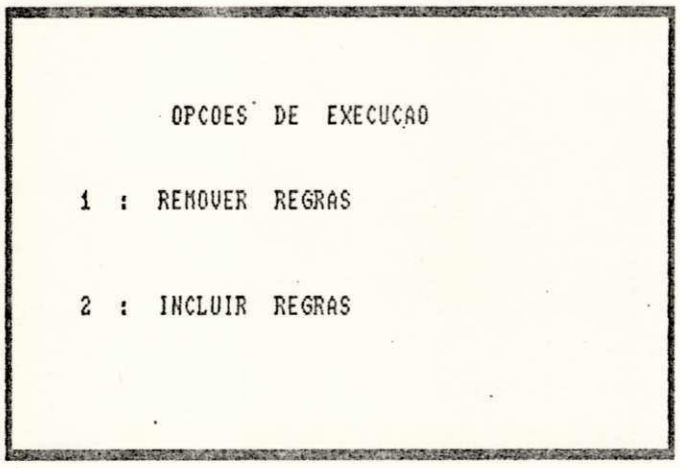

FIGURA 5.1 - MENU 1

Se o usuário desejar remover alguma regra (opção 1), um novo menu é mostrado (figura 5.2) e nele é escolhido o tipo de regra que o usuário deseja remover : Regra de Integridade (opção 1) ou uma Definição (opção 2). No caso da opção 1 são listadas na tela todas as RIs catalogadas em LRI, e o usuário faz a escolha pelo número da regra . No caso da opção 2, também são listadas as definições, só que pelo nome, e o usuário faz a

sua opção de remoção através do cursor. Em ambos os casos, as regras só serão efetivamente removidas depois que o usuário confirmar para o sistema que deseja realmente fazer a remoção.

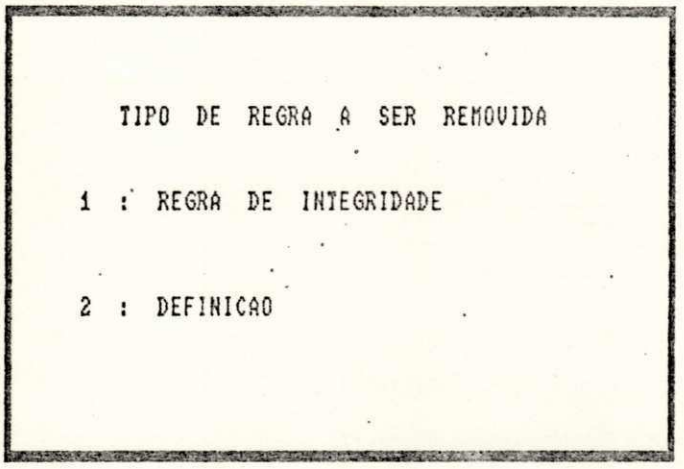

FIGURA 5.2 - MENU 2

Se a opção do usuário for incluir regras (opção 2) um novo menu é exibido na tela (figura 5.3); através dele, o usuário define qual será a linguagem utilizada para definir a regra que ele irá fornecer. Este menu voltará a ser exibido sempre que o Catalogador terminar de armazenar uma RI e não for feita a opção 3 pelo usuário.

Escolhida a linguagem em que a regra será fornecida, novos menus serão exibidos. Inicialmente o usuário tem opção, através do menu da figura 5.4 , de ter acesso ao esquema do BD às regras já definidas, que poderão ser utilizadas na formulação de novas regras. Se a opção escolhida foi a 1, de acordo com o esquema de BD que estamos considerando, o menu terá o aspecto mostrado na figura 5.5.

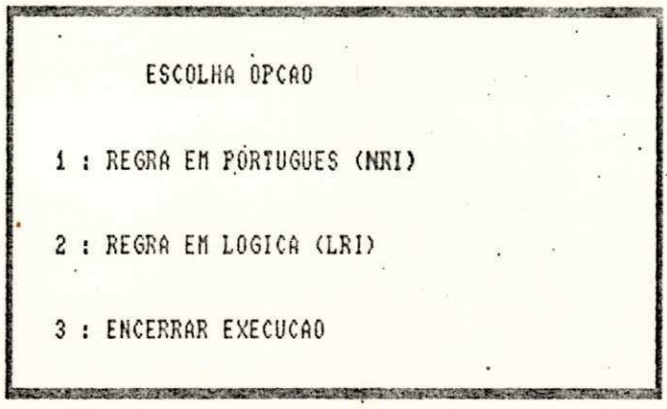

FIGURA 5.3 - MENU 3

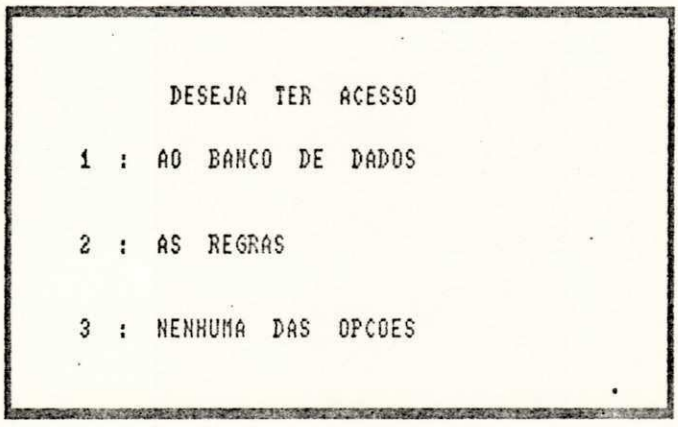

FIGURA 5.4 - MENU4

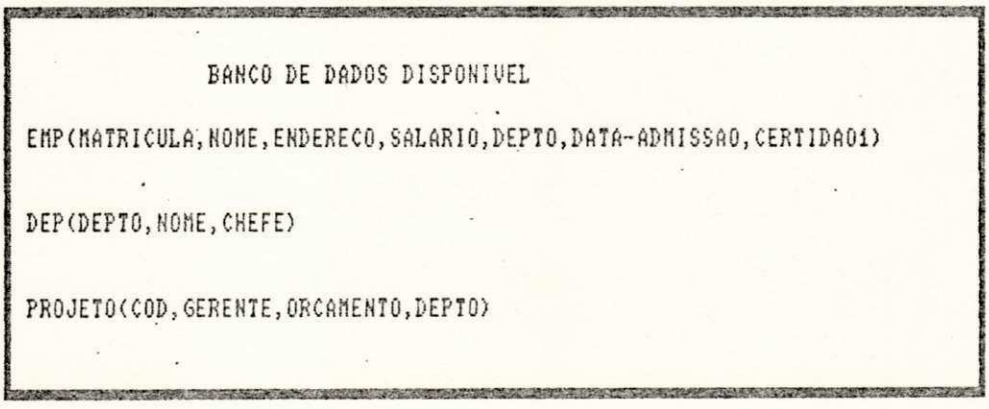

FIGURA 5.5 - MENU 5

Considerando as regras Grande-Projeto e Cônjuge definidas no capítulo anterior , se a opção escolhida foi a 2, o menu terá o aspecto da figura 5.6. Através deste menu o usuário pode ter acesso, via cursor, as versões em português das definições (ver 5.2.1).

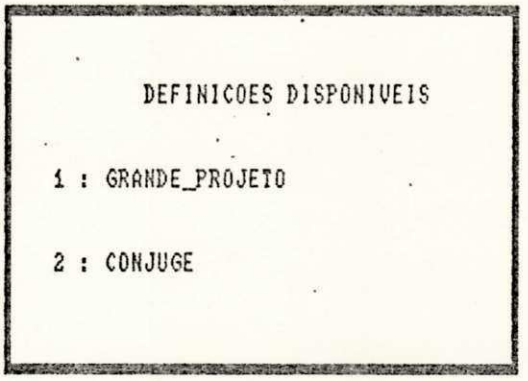

FIGURA 5.6 - MENU6

O usuário indicará ao sistema, através do menu da figura 5.7, o tipo de regra que ele fornecerá.

No caso da opção 1, o usuário diz ao sistema que a regra que ele irá fornecer é uma condição a ser satisfeita por uma determinada atualização em um determinado local (arquivo sobre o qual a operação incidirá).

Neste caso, o usuário determina, através de menus, o tipo de atualização (inclusão, exclusão, modificação), menu da figura 5.8, e o local, menu da figura 5.9, que serão usados posteriormente para a determinação da chave de acesso da RI no Catálogo.

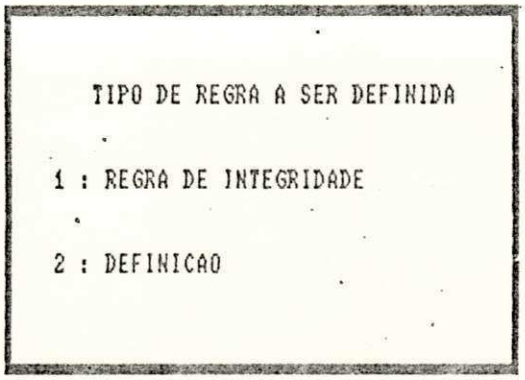

FIGURA 5.7 - MENU 7

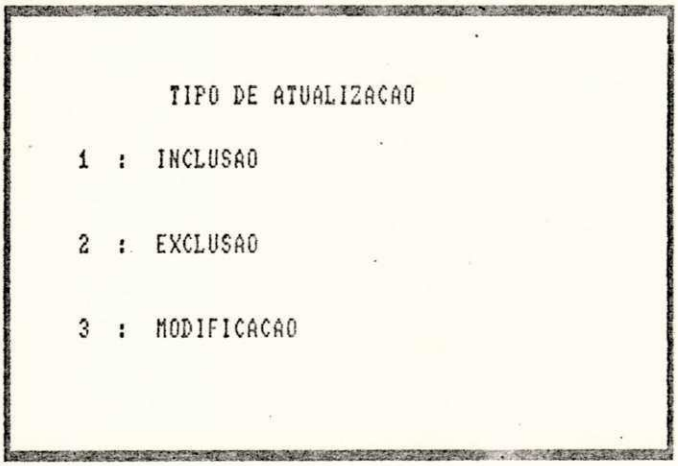

FIGURA 5.8 - MENU 8

FIGURA 5.9 - MENU 9

Se a opção for a 2, Definição, o sistema saberá que o usuário estará fornecendo uma regra, que deverá ser armazenada na tabela das definições.

O Catalogador terá diferentes comportamentos conforme a linguagem de definição utilizada.

5.1.1 CATALOGANDO UMA REGRA EM NRI

Neste caso,o Catalogador aciona um tradutor (Tradutor1), que terá como entrada a regra em NRI e como saida a mesma regra em LRI (figura 5.10). Tradutor1 é interativo : um diálogo é estabelecido sempre que ocorra um erro, ou ele não seja capaz de entender o que o usuário quis dizer. A partir do diálogo, o usuário pode corrigir ou clarear sua regra, que é novamente submetida ao Catalogador.

5.1.2 TRADUTOR1

Os três módulos que compõem Tradutor1 funcionam interdependentemente . A função do Analisador Léxico é fornecer ao Analisador Sintático os símbolos semanticamente corretos e as sua respectivas classes (categorias onde o simbolo está classificado). De posse do símbolo, fornecido pelo Analisador Léxico , o Analisador Sintático verifica a correção do mesmo quanto à sintaxe da NRI; caso não seja detectado um erro, o símbolo é então enviado ao Mapeador, que, de acordo com a classe do mesmo, gera uma proposição e/ou um relacionamento. O processo se repete até que toda a regra tenha sido analisada.

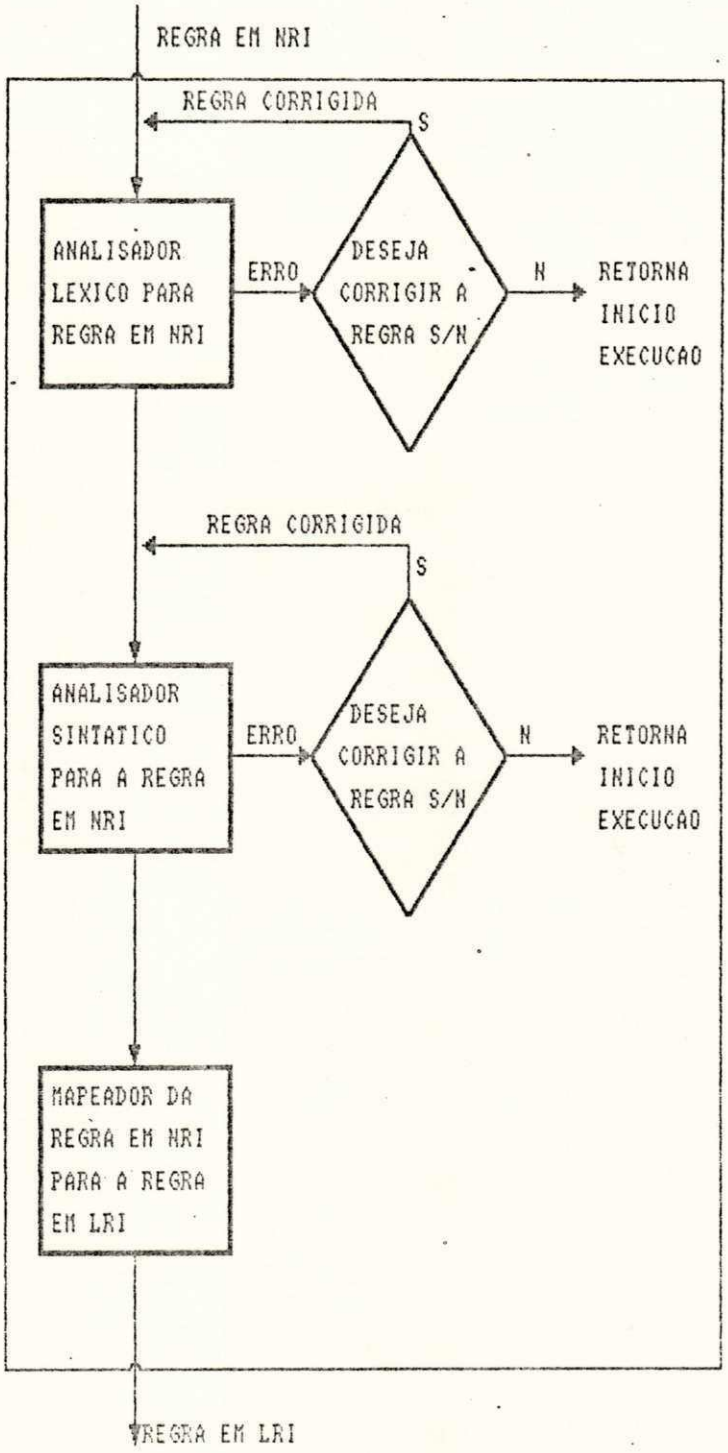

FIGURA 5.10 - TRADUTOR1

5.1.2.1 O ANALISADOR LÉXICO

O Analisador Léxico de posse da regra em NRI, fornecida pelo usuário, identifica a primeira sequência de caracteres diferentes de branco, da esquerda para a direita, na regra. O Analisador Léxico tenta classificar esta sequência nas diversas classes existentes. Se o símbolo não for de classificação imediata (nome de regra, valor, constante, símbolo especial) o Analisador Léxico tenta classificá - lo através das tabelas de símbolos terminais (figura 5.11).

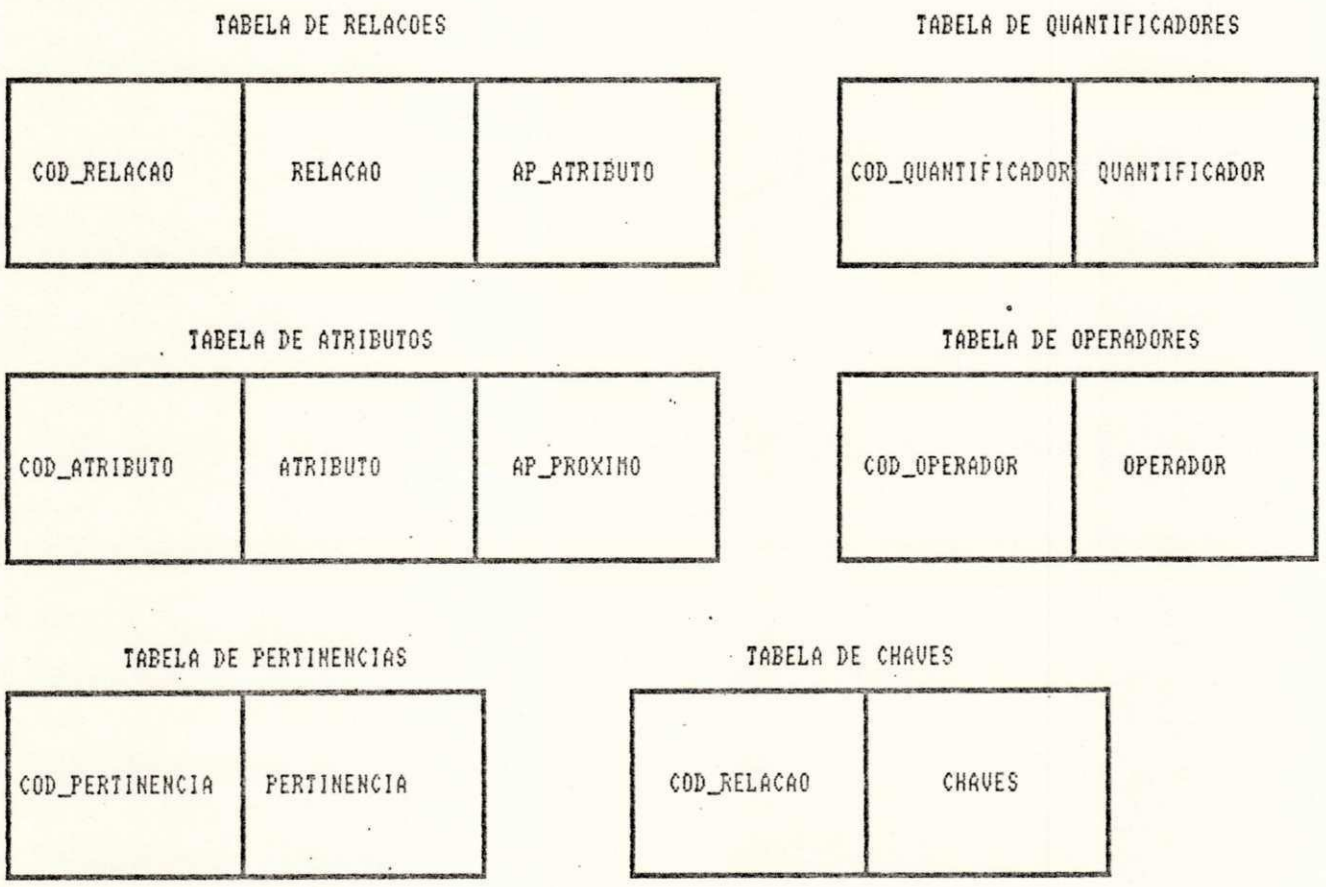

FIGURA 5.11 - TABELAS DE SIMBOLOS

Na tabela de relações cod_relação é um número que caracteriza cada relação do esquema do BD e ap_atributo é um apontador para o primeiro pertencente à relação referida. Se o símbolo não for igual a algum outro símbolo constante na tabela, o Analisador Léxico passa a fazer a classificação na tabela de sinônimos (figura 5.12).

FIGURA 5.12 - TABELA DE SINONIMOS

Na tabela de sinônimos cod_equivalente guarda o código da tabela que deve ser pesquisada para recuperar o símbolo original correspondente e ap_equivalente o número do registro, nesta tabela, que contém este simbolo. Se o símbolo é identificado como um sinônimo, o Analisador Léxico, através do cod_equivalente, aciona a tabela correspondente ao sinônimo, recuperando, via ap_equivalente, o símbolo que ele representa, que será o símbolo fornecido ao Analisador Sintático. Se, depois de todos estes procedimentos, o símbolo não foi ainda classificado e o seu comprimento for menor do que trinta (tamanho máximo da palavra correspondente ao símbolo), o Analisador Léxico recupera a próxima sequência de caracteres diferentes de branco da regra, faz a concatenação da sequência anterior com a atual e repete todos os procedimentos anteriores. Supondo que se queira classificar o símbolo não pode ser, inicialmente o Analisador

Léxico tenta classificar a sequência não e não consegue. Como o comprimento do símbolo que se tenta classificar (não) é menor do que trinta o Analisador Léxico recupera pode e faz a sua concatenação com a sequência anterior, o símbolo passa a ser então não pode . O Anlisador Léxico ainda não consegue classificar o simbolo (tamanho ainda menor que trinta), o procedimento acima é repetido, e finalmente não pode ser é classificado como uma pertinência.

Estes passos serão repetidos até que o símbolo seja classificado ou o tamanho do mesmo exceda o comprimento máximo permitido (30 caracteres). No segundo caso, o processamento é interrompido e uma mensagem de erro adequada é dada ao usuário, que pode optar por cancelar a régra ou tentar corrigí -la : neste caso, a regra corrigida é submetida novamente a Tradutor1.

5.1.2.2 O ANALISADOR SINTÁTICO

O Analisador Sintático verifica se a sintaxe da regra está correta, isto é, está de acordo com a gramática da NRI.

A metodologia utilizada para o reconhecimento foi a TOP_DOWN recursiva [7], devido, principalmente, ao seu uso anterior em outros trabalhos : o Analisador Sintático parte de um símbolo inicial e, através de procedimentos recursivos, representantes de cada símbolo não terminal (ver 4.1.1), tenta construir uma árvore sintática de reconhecimento, cujas folhas serão os símbolos terminais que compõem a regra.

5.1.2.3 O MAPEADOR

O mapeador tem como função mapear os símbolos enviados pelo Analisador Sintático para a LRI.

O código gerado pelo mapeador depende da classe dos símbolos que ele recebe do Analisador Sintático . Daremos a seguir o procedimento do Mapeador para cada classe de símbolos ou série de classes de símbolos. Nos exemplos, a parte em negrito da RI em NRI, corresponde à parte que será mapeada para LRI e ... significa que a RI em LRI está incompleta :

classe 1 : <relação> - Nenhum código é gerado.

classe 2 : <atributo> - O Mapeador gera uma proposição formada pela relação a qual pertence o símbolo, substituindo o atributo e chave da relação por variáveis temporárias (Xdd) e o restante dos atributos por - .

. Um chefe deve ser empregado.

. DEP(X01,-,X02), ...

classe 3 : <atributo> de [<quantificador>] <relação> - O Mapeador verifica se à relação, à qual pertence o atributo, é igual à relação recebida como símbolo. Se as relações forem iguais ele gera uma proposição como no caso anterior, senão o módulo de erro é ativado .

. Um gerente de projeto deve ser empregado.

. Projeto(X01,X02,-,-), ...

classe 4 : <atributo> de [<quantificador>] <atributo> - Temos dois casos a considerar :

comparação com os símbolos que serão reconhecidos, através do operador. Após o operador, o Mapeador pode receber qualquer das sequências de símbolos vistas anteriormente. Em todos os casos o Mapeador terá o mesmo comportamento visto em cada caso, apenas que será acrescentado, neste caso, um relacionamento envolvendo o atributo reconhecido antes do operador e o atributo semelhante reconhecido após o operador .

. Um chefe deve ter salário maior que que o salário de um gerente.

. DEP(X01,-,X02),EMP(X03,-,-,X04,-,-,-),PROJETO(X05,X06,-,-), EMP(X07,-,-,X08,-,-,-),X02=X03,X07=X06,X04)X08,X03( )X07.

classe 7 : *(identificador)* - O Mapeador apenas armazena o identificador como sendo o nome da definição que está sendo fornecida .

classe 8 : (definição) - O Mapeador recupera, na tabela de definições, a definição correspondente ao nome reconhecido, que corresponderá a proposição gerada :

Até agora temos considerado o papel do Mapeador apenas gerando proposições para símbolos reconhecidos, mas a sua função é mais ampla.

Nos casos vistos até então, não temos considerado a função exercida por duas classes de símbolos reconhecidos pelo AL : os quantificadores e as pertinências .

Como já foi dito anteriormente (4.1) o nosso sistema divide os quantificadores e pertinêncis em positivos e negativos. Uma regra tem, no mínimo, um quantificador e uma pertinência, que

serão responsáveis pelos sinais dos relacionamentos existentes na regra em LRI.

O Mapeador é que faz as considerações necessárias para a determinação destes sinais. Ele faz isto através de uma variável booleana, inicializada como true a cada fornecimento de regra . Todas as vezes que o Analisador Léxico reconhece um quantificador ou uma pertinência, o Mapeador determina a mudança ou não do valor desta variável de acordo com o sinal do símbolo reconhecido. Esta mudança é determinada usando as regras de sinal de multiplicação da matemática, onde true é considerado sinal positivo e false negativo. Quando é gerado um relacionamento, o Mapeador verifica o valor da variável e determina se deve haver uma troca do sinal do mesmo (variável é igual a false) ou não (variável é igual a true).

A tradução de uma regra em NRI para a LRI se dá em duas etapas : em um primeiro passo é reconhecido e mapeado para a LRI tudo que vem antes da pertinência, em um segundo passo repete-se o procedimento para o restante da regra (porção posterior à pertinência). Cabe ao Mapeador estabelecer um vínculo entre as proposições geradas nas duas etapas, através de um relacionamento entre as temporárias representantes dos atributos semelhantes nas proposições.

Apresentamos, a seguir, alguns exemplos de RIs, em NRI, e os passos dados por Tradutor1, para os seus mapeamentos em LRI.

Exemplo 1 : Na Inclusão de um empregado na relação EMP

Um departamento de empregado deve existir no arquivo de departamentos.

Passos executados por Tradutor1 :

- Reconhece um como um quantificador positivo;
- Reconhece departamento de empregado como um sinônimo do atributo depto na relação EMP ;
- Gera uma proposição referente ao atributo reconhecido : EMP(X01,-,-,-,-,X02,-) ;
- Reconhece deve existir como uma pertinência positiva ;
- Reconhece no arquivo de departamentos como uma locução equivalente à relação DEP ;
- Gera uma proposição referente à relação reconhecida : DEP(X03,-,-) ;
- Reconhece o "." como caractere finalizador da RI;
- Estabelece um relacionamento entre as proposições geradas, de acordo com os sinais dos quantificadores e pertinências reconhecidos : X02 = X03 ;
- Saida de Tradutor1 : RI em LRI, com a respectiva chave de acesso ao Catálogo :

  OP[I,EMP] <--- EMP(X01,-,-,-,-,X02,-),DEP(X03,-,-),
  X02 = X03.

Exemplo 2 : Na definição de um grande projeto

Um *grande_projeto* deve ter orçamento maior que "100 Milhões".

Passos executados por Tradutor1 :

- Reconhece um como um quantificador positivo;

. Reconhece *grande_projeto* como o nome da regra que está sendo definida;

. Reconhece deve ter como uma pertinência positiva;

. Reconhece orçamento como um atributo da relação PROJETO ;

. Gera uma proposição para o atributo reconhecido :

PROJETO(X01,-,X02,-)

. Reconhece maior que como um operador;

. Reconhece "100 Milhões" como uma constante;

. Reconhece "." como finalizador da definição

. Estabelece um relacionamento entre a proposição gerada o operador e a constante reconhecida, de acordo com o sinal do quantificador e pertinência : X02>"100 Milhões";

. Saida de Tradutor1 : RI, tipo definição em LRI :

GRANDE_PROJETO(X01) <--- PROJETO(X01,-,X02,-),
X02 > "100 Milhões".

Exemplo 3 : Na Inclusão de um projeto na relação Projeto

Um gerente de grande_projeto deve ter data_admissão menor que 1978.

Passos executados por Tradutor1 :

. Reconhece um como um quantificador positivo;

. Reconhece gerente de grande_projeto como um atributo da definição grande_projeto;

. Recupera a definição de grande_projeto e gera a proposição correspondente ao atributo reconhecido :

PROJETO(X01,X03,X02,-),X02 > "100M" ;

- Reconhece deve ter como uma pertinência positiva ;
- Reconhece data_admissão como um atributo da relação Empregado;
- Gera uma proposição correspondente ao atributo reconhecido : EMP(X04,-,-,-,-,-,X05) ;
- Reconhece maior que como um operador;
- Reconhece 1978 como um valor;
- Gera um relacionamento entre o atributo reconhecido anteriormente e a constante baseado no operador : X05 < "1978".
- Gera um relacionamento entre os atributos reconhecidos antes e depois da pertinência : X03 = X04 ;
- Saida de Tradutor1 : RI em LRI, com a respectiva chave de acesso no Catálogo :

OP[I,PROJETO] <--- PROJETO(X01,X03,X02,-),X02>"100M",
EMPREGADO(X04,-,-,-,-,X05,-),
X05 < "1978", X03 = X04.

Exemplo 4 : Na modificação e inclusão de um empregado na relação EMP.

Um empregado não deve ter salário maior que o salário de um gerente.

Passos executados por Tradutor1 :

- Reconhece um como um quantificador ;
- Reconhece empregado como sinônimo do atributo

matrícula na relação Empregado;

- Gera a proposição correspondente ao atributo reconhecido : EMP(X01,-,-,-,-,-,-) ;
- Reconhece a pertinência negativa não deve ser ;
- Reconhece o atributo salário ;
- Verifica se o atributo pertence à relação reconhecida anteriormente (EMP);como isto acontece assinala o atributo na proposição gerada anteriormente. A proposição ficará então : EMP(X01,-,-,X02,-,-,-)
- Reconhece maior que como um operador;
- Reconhece salário de um gerente como uma locução de dois atributos pertencentes a relações diferentes ;
- Gera uma proposição para cada um dos atributos reconhecidos anteriormente, estabelecendo um vínculo entre eles através de um relacionamento :
  EMP(X03,-,-,X04,-,-,-),PROJETO(X05,X06,-,-),X03=X06 ;
- Gera o relacionamento entre o atributo anterior ao operador e o atributo posterior ao operador :
  X02 < X04 (observe que devido à pertinência negativa o sinal do operador é invertido);
- Saida de Tradutor1 : RI em LRI, com duas chaves de acesso ao Catálogo :

```
OP[M,EMP] <--- EMP(X01,-,-,X02,-,-,-),
               EMP(X03,-,-,X04,-,-,-),
               PROJETO(X05,X06,-,-),X03 = X06,
               X02 < X04.

OP[I,EMP] <--- EMP(X01,-,-,X02,-,-,-),
```

EMP(X03,-,-,X04,-,-,-),

PROJETO(X05,X06,-,-),X03 = X06,

X02 < X04.

Exemplo 5 : Na inclusão de um departamento na relação DEP.

Um chefe de dep deve ser empregado.

Passos executados por Tradutor1 :

- Reconhece Um como um quantificador;
- Reconhece chefe de dep como uma locução equivalente ao atributo chefe;
- Gera uma proposição correspondente ao atributo reconhecido : DEP(X01,-,X02);
- Reconhece deve ser como uma pertinência positiva;
- Reconhece empregado como sinônimo do atributo matrícula da relação EMP ;
- Gera uma proposição relativa ao atributo reconhecido :

  EMP(X03,-,-,-,-,-,-) ;
- Estabelece uma relação entre as proposições geradas :

  X02 = X03 ;
- Saída de Tradutor 1 :

  OP[I,DEP] <--- DEP(X01,-,X02),EMP(X03,-,-,-,-,-,-),

  X02 = X03.

5.1.3 CATALOGANDO UMA REGRA EM LRI

No caso em que a linguagem de definição usada seja a LRI, antes da sua catalogação a regra passará por um Analisador (figura 5.13), para a verificação se a regra está de acordo com o léxico (a cargo do Analisador Léxico) e com a sintaxe (a cargo do

Analisador Sintático). Tanto o Analisador Léxico como a Analisador Sintático não trazem novidades em relação aos respectivos analisadores da NRI. O Analisador para a LRI é também interativo, permitindo a correção de regras mal formuladas.

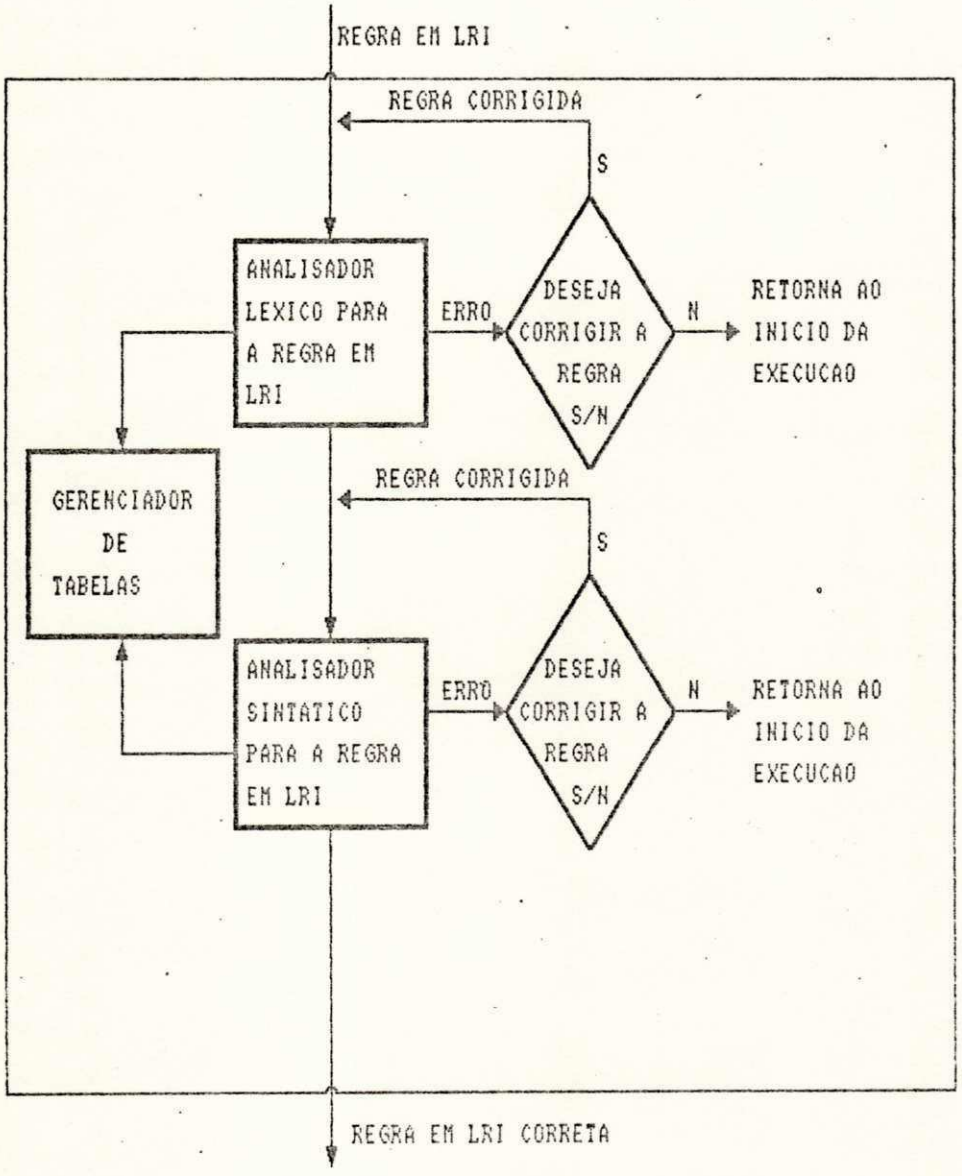

FIGURA 5.13 - O ANALISADOR PARA A REGRA EM LRI

Nesta fase do processamento da RI são criadas tabelas (figura 5.14) que armazenarão os dados da RI, as quais são utilizados posteriormente pelo Gerador, para a tradução da RI para a Álgebra Relacional (6.2). A função destas tabelas, bem como o seu detalhamento estão na seção 6.2.

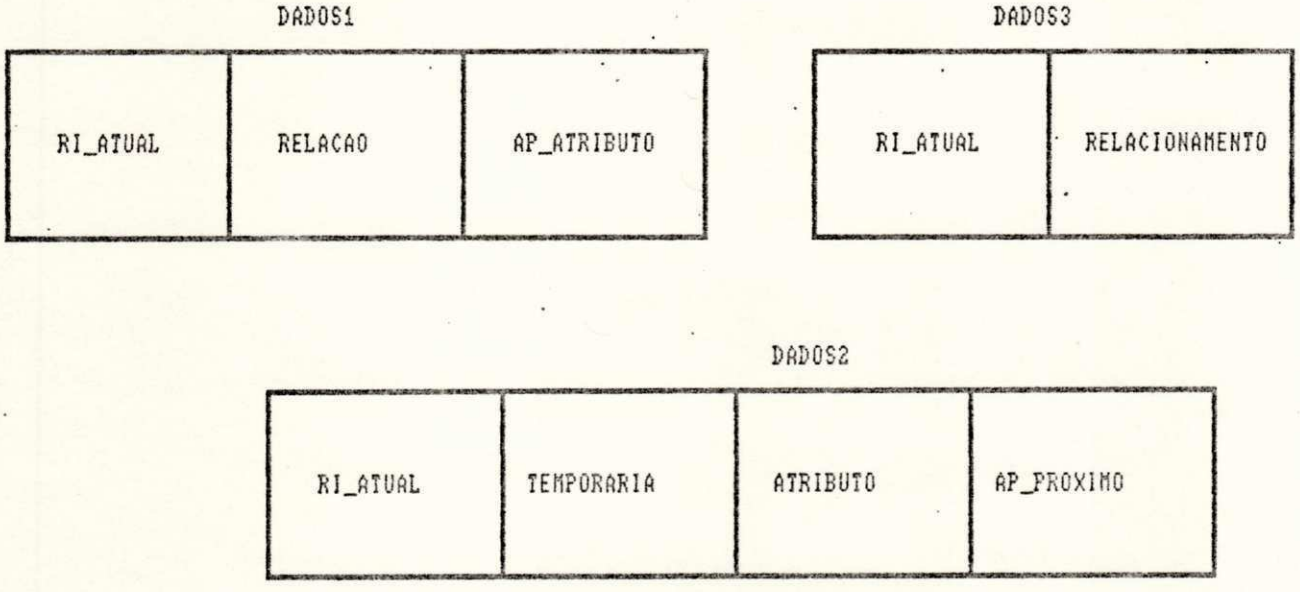

FIGURA 5.14 - TABELAS GERENCIADAS PELO ANALISADOR

5.2 O ARMAZENAMENTO DA REGRA

Como já foi dito anteriormente, as regras se dividem em dois tipos : as regras para atualizações (Regras de Integridade) e as definições. Estes dois tipos de regra serão armazenados separadamente, em estruturas de dados diferentes .

5.2.1 ARMAZENAMENTO DE REGRAS TIPO DEFINIÇÃO

As regras do tipo definição são armazenadas em uma tabela, formada por registros com cinco campos distintos (figura 5.15).

NOME DA REGRA	REGRA EM LRI	RELACAO ORIGINAL	ATRIBUTO DE COMPARACAO	MAIOR_TEMPORARIA

FIGURA 5.15 - ESTRUTURA DE DADOS DA TABELA DE DEFINICOES

Se a definição tiver sido fornecida em LRI, o usuário precisará fornecer ao sistema o nome da regra e a regra em LRI; se a linguagem utilizada foi a NRI, o nome da regra já foi armazenado quando do fornecimento da mesma e a própria definição, traduzida para LRI, é a saida do tradutor1.

O conteúdo do restante dos campos é determinado pelo próprio sistema : relação original é a relação que dá origem à regra ; o atributo de comparação é a chave da relação original ; maior_temp é o maior valor de temporária utilizada na definição em LRI ;

No exemplo 1 de definiçGrande-Projetoe, os valores dos campos seriam , respectivamente :

. Nome da Regra : Grande_Projeto ;

. Regra em LRI : PROJETO(X01,-,-,-,X02),X02 > "100 Milhões" ;

. Relação Original : Projeto ;

. Atributo de Comparação : X01 ;

. Maior_Temporária : X02 ;

Estes campos são importantes no mapeamento de RIs, em NRI, que utilizem definições. Vamos analisar o papel de cada um no mapeamento do exemplo 3 :

Para gerar a primeira proposição - Projeto(X01,X03,X02,-) , X02 > "100 Milhoes" - o Mapeador determina a relação à qual pertence o atributo gerente, compara esta relação com o campo relação_original, do registro da definição Grande_projeto, verificando se são iguais. Verificada a igualdade, o Mapeador verifica se a posição de atributo gerente, na relação Projeto (2) é ocupada, na definição, por temporárias. Ele faz isto verificando se a posição do atributo na relação original, determinada pela inspeção na tabela de atributos, está ocupada , na definição, por temporárias ou não. Ao verificar, que o atributo reconhecido não está assinalado na definição, o Mapeador gera uma temporária, a partir do valor do campo maior_temporária, e substitui o - relativo à posição 2 (posição do atributo gerente por esta temporária.

Ao gerar a proposição relativa à parte da RI posterior à pertinência , o Mapeador precisará do valor de maior_temporária, que, neste caso, será a última temporária gerada pelo sistema (X03); a proposição gerada será então : Emp(X04,-,-,-,-,X05),X05<"1978".

Para finalizar, o Mapeador precisa estabelecer uma ligação entre as duas proposições geradas. Ele precisará então do valor do campo atributo de comp. O fato de ter sido incluida uma nova temporária na definição, para gerar a primeira proposição, faz com que esta variável seja o valor adotado pelo Mapeador, para o exemplo 3, da temporária de comparação, logo o relacionamento gerado será : X03 = X04.

Quando o Catalogador termina o armazenamento de uma regra tipo definição o usuário ainda pode optar por armazenar uma versão em português para ela . A versão em português da definição em LRI será armazenada em um arquivo tipo texto e o propósito de facilitar a utilização das definições na formulação de novas RIs. Para a definição Grande-Projeto poderíamos armazenar a seguinte versão em português :

" Um Grande_Projeto é aquele que possui orçamento maior que cem (100) milhões de cruzados ."

5.2.2 ARMAZENAMENTO DE REGRAS PARA ATUALIZAÇÕES (RI)

No caso da regra ser uma RI, o seu armazenamento será feito em uma tabela organizada por tipo e local (Catálogo), como já foi visto. O acesso ao Catálogo é feito através de um índice, cuja chave de acesso é determinada pela concatenação do tipo e do local de uma RI. Cada entrada na tabela é uma cabeça de uma lista de RIs para um particular tipo i e local j (figura 5.16)

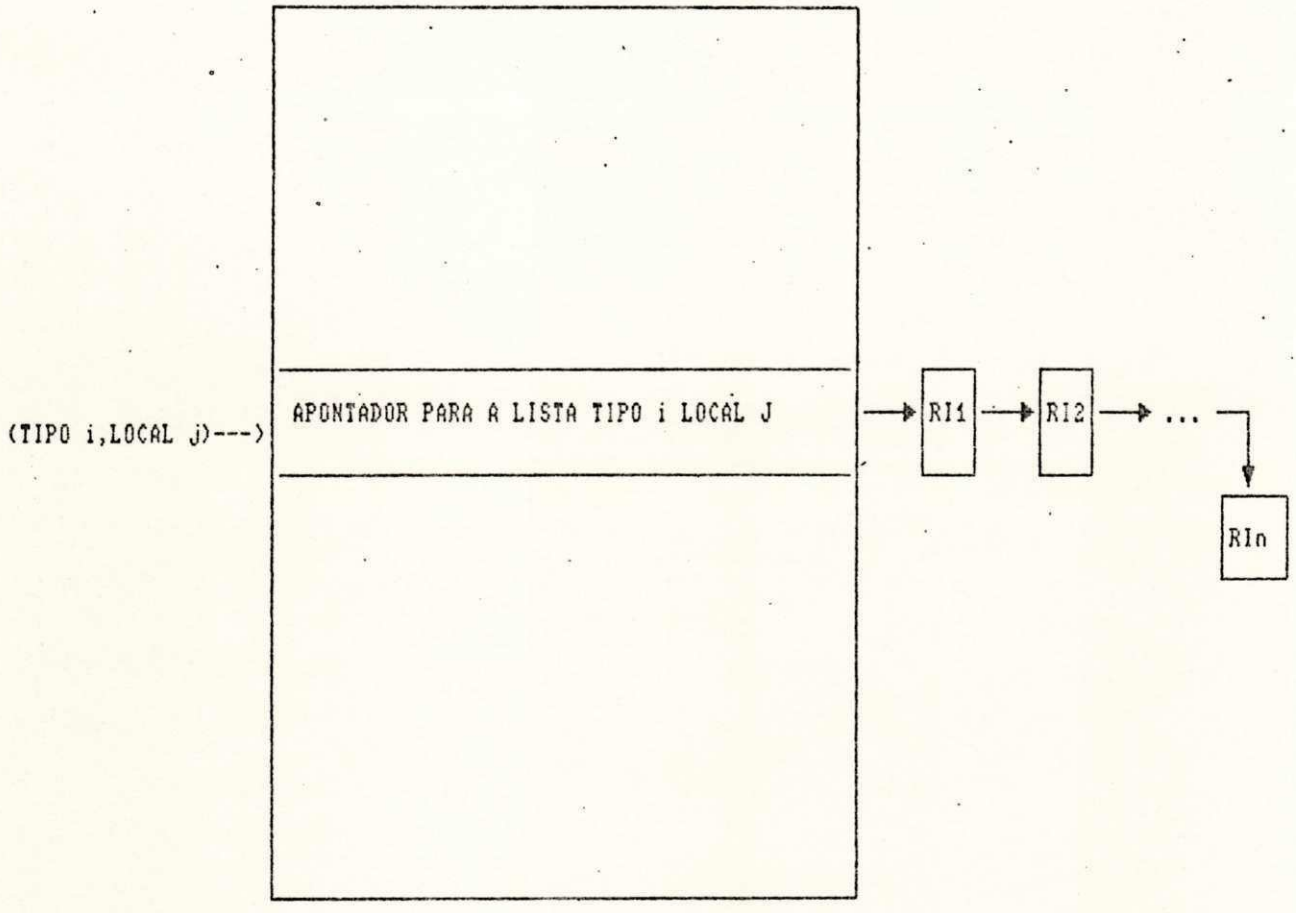

FIGURA 5.16 - CATALOGO DE RIs

Quando o Catalogador vai armazenar uma nova RI podem ocorrer dois casos distintos :

. Existe no Catálogo uma ou mais RIs pertencentes ao mesmo tipo e local da RI a ser armazenada - neste caso o catalogador insere a nova RI depois da última RI armazenada, a nova RI passa a ocupar a última posição na lista relativa à seu tipo e local;

. Não existem no Catálogo, RIs pertencentes ao tipo e local da RI a ser armazenada - neste caso o Catalogador insere no Catálogo um novo registro cuja chave de acesso é formada pelo tipo e o local da nova RI, que ocupará a primeira posição da

lista de RIs relativas à chave.

Uma RI pode ser armazenada para mais de uma chave de acesso, isto quer dizer que uma mesma RI deve ser satisfeita para operações diferentes em locais diferentes.

6. O GERADOR

A função do Gerador é escrever automaticamente um Programa de Crítica, a partir de um Catálogo de RIs e de um Esquema de um particular BD operacional (figura 6.1).

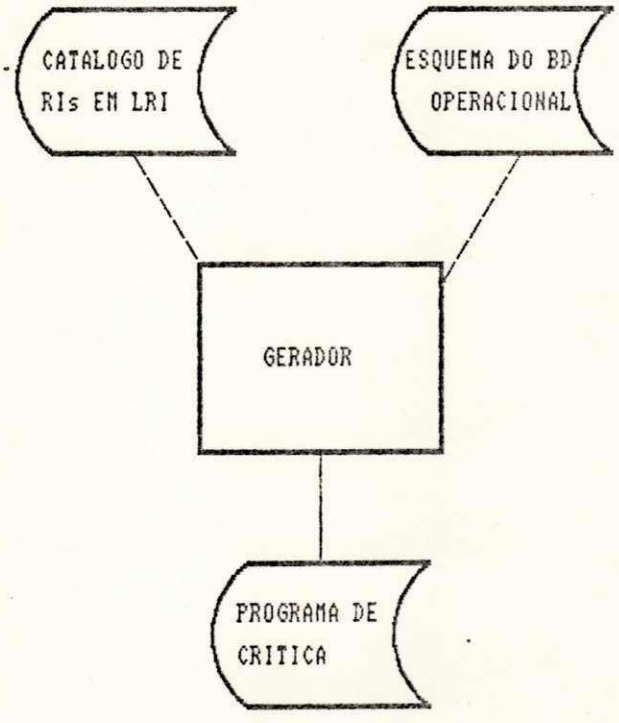

FIGURA 6.1 - O GERADOR

Para cada par (tipo, local) é gerada a rotina respectiva, contendo as instruções de consulta ao BD para a verificação das RIs correspondentes às instruções de decisão (decidem se atualizam ou não o BD) e à instrução de atualização correspondente.

6.1 MÓDULOS COMPONENTES DO GERADOR

O Gerador é composto dos seguintes módulos (figura 6.2) :

TRADUTOR2 : responsável pelo mapeamento de RIs em LRI para RIs em Álgebra Relacional .

TRADUTOR3 : faz o mapeamento de RIs em Álgebra Relacional para instruções de consulta, na linguagem do GBD utilizado. A existência de Tradutor3 permite que o nosso sistema seja adaptado para diferentes GBDs relacionais. No nosso caso, as instruções de consulta são geradas em SQL[2].

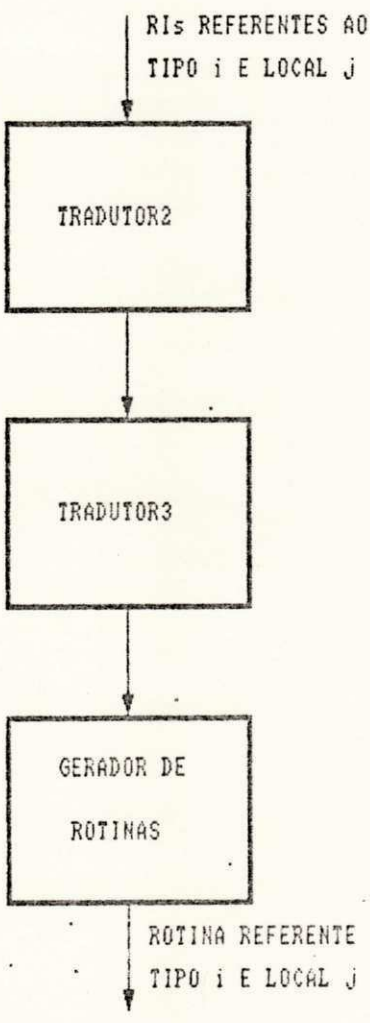

FIGURA 6.2 - MODULOS DO GERADOR

GERADOR DE ROTINAS : Gera cada rotina referente ao tipo e ao local de uma determinada atualização do BD.

O Gerador recebe do sistema o Catálogo de RIs e as Tabelas citadas na seção 5.1.3.

Para cada chave de acesso ao Catálogo deverá ser gerada uma rotina, composta de Instruções de Decisão, Instruções de Consulta e da própria transação a que ela se refere. Isto é feito pelo Gerador da seguinte maneira : para cada RI pertencente à lista de RIs referente à chave de acesso, o Gerador chama Tradutor2, para que ele faça o mapeamento da RI em LRI para a Álgebra Relacional. Tradutor2 devolve a RI, em Álgebra Relacional, para o Gerador, juntamente com as Tabelas (ver seção 6.3.1) que serão utilizadas para o mapeamento posterior, por Tradutor3. O Gerador chama Tradutor3, que terá como saída a RI em SQL e que será remetida para o Gerador de Rotinas, onde é finalmente gerada a rotina referente à RI em questão.

Os procedimentos descritos são repetidos, até que a última RI da lista referente à chave seja processada. O Gerador, de posse das rotinas, gera instruções de decisão que farão parte dos comandos referentes à chave de acesso em questão.

Todo o processo é repetido até que todas as chaves tenham sido analisadas. Ao final do processamento o Programa de Crítica estará gerado.

6.2 TRADUTOR2

Tradutor2 (figura 6.3) tem como entrada o Catálogo de RIs e as Tabelas Dados1, Dados2 e Dados3 (ver figura 5.15), e como saída, cada RI mapeada para Álgebra Relacional [5].

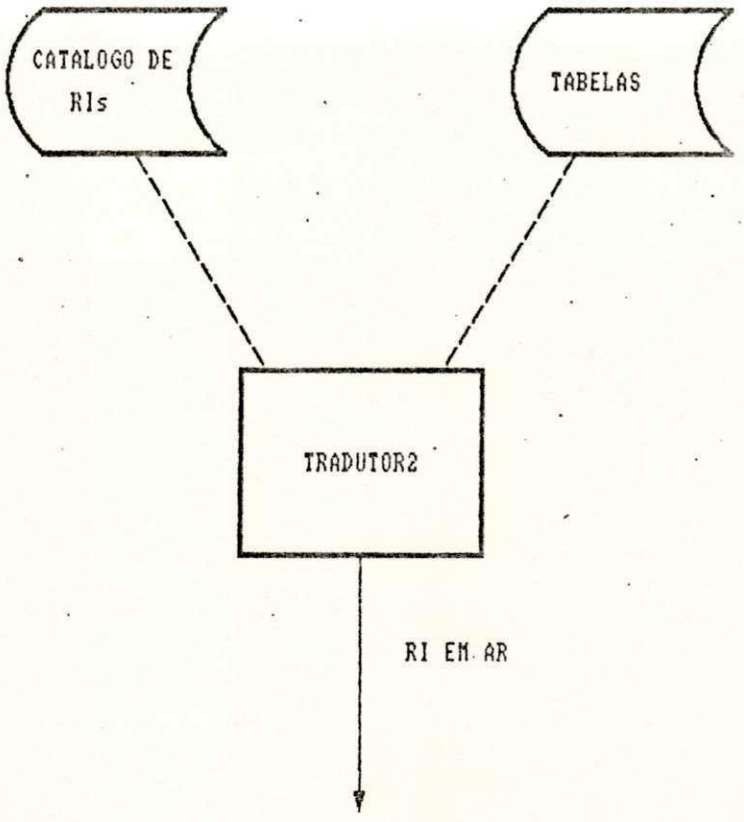

FIGURA 6.3 - TRADUTOR2

Dados1, Dados2 e Dados3 são fundamentais para o mapeamento da Lógica para a Álgebra Relacional.

Devemos observar que na representação dos atributos assinalados em uma relação, que faz parte de uma RI, substituimos o nome destes atributos por variáveis temporárias, que chamaremos temporárias. Esta substituição será importante quando de mapeamento Lógica ---> Álgebra Relacional.

Dados1 armazena as informações relativas às relações do BD que compõem determinada RI. O campo **ri_atual** armazena o número do registro, no Catálogo, que contém a RI; **relação** contém o nome da relação no Esquema do BD e **ap_atributo** é um apontador para o primeiro registro da tabela Dados2 que contém informações

relacionadas aos atributos marcados na relação.

Dados2 guarda informações referentes aos atributos assinalados em uma determinada relação componente de uma determinada RI : ri_atual guarda a mesma informação que o seu homônimo em Dados1 - a repetição se deve a necessidade de recuperações mais rápidas em Dados2 - nome_temporária armazena a temporária representativa do atributo na RI, atributo guarda o nome do atributo, representado pela temporária, no Esquema de BD e ap_proximo é um apontador para o próximo atributo pertencente à relação e assinalado na RI.

Dados3 contém informações relativas aos relacionamentos existentes entre os atributos assinalados em cada RI; ri_atual armazena o mesmo tipo de informações que os seus homônimos em Dados1 e Dados2 e o campo relacionamento guarda o relacionamento com os atributos ainda representados por temporárias.

6.2.1 OPERADORES RELACIONAIS

Para o mapeamento Lógica para a Álgebra Relacional utilizamos o seguinte conjunto básico de operadores relacionais [5] :

PR(r,y) - Projeção da relação r sobre y, onde y é um subconjunto dos atributos de r;

S1(r,f) - Seleção de um conjunto de tuplas da relação r que satisfazem à condição f, f tem como operandos atributos de r ;

U(r,s) - União de r com s, que é um conjunto de tuplas que estão em r, ou s ou nas duas relações, projetadas sobre os atributos comuns ou compatíveis.

r JN s - Junção das tuplas da relação r com as tuplas da relações, para formarem tuplas com valores do conjunto de

atributos R(r) U R(s), onde R(r) é o conjunto de atributos de r e R(s) é o conjunto de atributos de s e onde os atributos comuns de r e de s tem valores iguais;

O mapeamento de uma RI em LRI em um conjunto de seleções, uniões, projeções e/ou junções está resolvido em [5].

6.2.2 O MAPEADOR LÓGICA ---> ÁLGEBRA RELACIONAL

Antes de descrevermos como Tradutor2 faz o mapeamento das RIs em Linguagem em Lógica para a Álgebra Relacional, faremos algumas considerações.

A primeira diz respeito aos nomes das variáveis locais ao programa de crítica. Tais identificadores são formados pela concatenação do caractere $ com o nome do ítem de dado correspondente no esquema do BD (<nome-var>::=$(ítem-de-dado-do-Esquema)).

Outra consideração feita pelo sistema diz respeito à forma de apresentar a condição a ser satisfeita, representada pela RI. Optamos por representar a RI utilizada limitada por colchetes, representando um primeiro operando de uma expressão booleana. O operador desta expressão será o = ou o <>, de acordo com as condições a serèm satisfeitas , e o segundo operando será o Ø, equivalente a uma relação vazia.

Seja, por exemplo, a RI referente à unicidade do atributo matrícula na relação EMP :

a - RI em LRI : OP[I,EMP] (--- EMP(XØ1,-,-,-,-,-,-,-),
EMP(XØ2,-,-,-,-,-,-,-),XØ1 <> XØ2.

b - RI em AR : SL(PR(EMP,XØ2),$matrícula <> XØ2)

O fato de se fazer uma seleção do atributo matrícula na relação EMP e que seja diferente da matrícula do empregado sendo inserido não traduz exatamente o fato de que não podemos ter dois empregados com a mesma matrícula. Para que a RI em AR traduza exatamente a condição a ser satisfeita pela transação, mudamos o sinal do relacionamento e criamos a expressão booleana (Ítem c abaixo) na qual a RI em AR é o primeiro operando. O operador é determinado pela existência ou não de mudanças no operador original do relacionamento. No caso do exemplo como houve uma mudança o operador será igual a =.

c - [SL(PR(EMP,X02),$Matrícula = X02)] = ∅

O mapeamento Linguagem em Lógica ---> Álgebra Relacional passa por três etapas distintas : na primeira delas, é feita a classificação dos relacionamentos pertencentes à RI ; na etapa seguinte, é feito o mapeamento propriamente dito, a partir dos relacionamentos já classificados ; na terceira e última etapa, é gerada a condição, a ser satisfeita pela RI, através da expressão booleana citada anteriormente.

6.2.2.1 CLASSIFICANDO OS RELACIONAMENTOS

Nesta etapa, Tradutor2 pesquisa todos os relacionamentos pertencentes à RI que está sendo mapeada, classificando cada relacionamento em sua classe, de acordo com os tipos de operandos envolvidos.

Um operando pode ser de dois tipos : ou ele é uma temporária atributo ou uma constante. Caso seja uma temporária ele pode

ainda pertencer ou não ao local da transação, à qual a RI se refere.

De acordo com os tipos dos operandos envolvidos em um determinado relacionamento, este pode ser classificado em quatro classes diferentes:

Classe 0 :O relacionamento envolve um atributo pertencente ao local e uma constante;

Classe 1 : O relacionamento envolve dois atributos não pertencentes ao local;

Classe 2 : O relacionamento envolve um atributo não pertencente ao local e uma constante;

Classe 3 : O relacionamento envolve dois atributos, um pertencente ao local e o outro pertencente a uma relação diferente do local.

Seja, a título de exemplo, a seguinte RI em LRI :

```
OP[I,PROJETO] <--- PROJETO(X01,-,X02,-,X03),X02>"100Milhões",
                   EMP(X04,-,-,-,-,-,-,X05),X02 = X04 ,
                   X05 < "1978".
```

Os relacionamentos teriam a seguinte classificação :

X02 > "100Milhões" : classe 0

X02 = X04 : classe 3

X05 < "1978" : classe 2

6.2.2.2 O MAPEAMENTO LINGUAGEM EM LÓGICA ---> ÁLGEBRA RELACIONAL

Depois de classificar os relacionamentos, Tradutor2 inicia o mapeamento Linguagem em Lógica ---> Álgebra Relacional.

Inicialmente, Tradutor2 determina a quantidade de relacionamentos que a RI possui. Feito isto, ele começa o mapeamento a partir dos relacionamentos pertencentes à classe de número mais baixo até as de número mais alto.

Tradutor2 tem um comportamento diferente para cada classe, no exemplo, o relacionamento sendo analisado aparece em negrito e "..." indica que a regra ainda não está totalmente traduzida para a Álgebra Relacional . Procuramos ilustrar os casos de cada classe que nos pareceram mais difíceis de serem assimilados, o exemplo é único para toda explanação (exceto para relacionamentos da classe 0), referente, por exemplo, a modificação na relação departamento :

```
. OP[M,DEP] <--- DEP(X01,-,X02),EMP(X03,-,-,X04,-,-,-),
                 X02=X03,X04>100000,PROJETO(X05,X06,-,X07),
                 EMP(X08,-,-,X09,-,-,-),X06=X08,X01=X07,
                 X04>X09.
```

Que poderia ser traduzido por : Um chefe de departamento deve ter salário maior que cem mil cruzeiros e não deve ter existir nenhum gerente de projeto, subordinado ao departamento, com salário superior ao salário deste chefe.

Os relacionamentos do exemplo teriam a seguinte classificação :

```
X06 = X08        : Classe 1
X04 > X09        : Classe 1
X04 > 100000     : Classe 2
X02 = X03        : Classe 3
X01 = X07        : Classe 3
```

classe 0 : Podemos ter dois casos distintos :

caso 1 - O número de relacionamentos da RI é igual a um; neste caso a RI em AR será o próprio relacionamento, onde a temporária é substituida pelo atributo que ela representa, antecedido pelo caractere $.

. EMP(X01,-,-,-,-,X02,-,-),X02 > "10.000,00".

. $salário > "10.000,00.

caso 2 - O número de relacionamentos da RI é maior que um ; neste caso, uma variável booleana é setada como true,indicando ao Gerador que a RI só é válida se determinado atributo , pertencente ao local, obedece a uma determinada condição indicada pelo relacionamento. Quando recebe esta variável com o valor true o Gerador determina que um comando condicional IF, cuja condição é o relacionamento e cujo comando é a própria RI sem a parte referente à condição (RI), deverá ser gerada. O relacionamento X02 > "100 Milhões levará a criação de um comando na seguinte forma : IF $ORÇAMENTO > "100Milhões" THEN RI em Álgebra Relacional . Indicando que a expressão em Álgebra Relacional só é válida para projetos que tenham orçamentos maiores que cem milhões de cruzados.

classe 1 : Temos dois casos distintos :

caso 1 - O relacionamento da classe 1 é o primeiro relacionamento da RI sendo analisada ; Tradutor2 gera um SL(r,F), onde r é a resultante da operação (PR (r1,y1)) JN (PR(r2,y2)),r1 e r2 são respectivamente as relações que contém o primeiro e o segundo operandos do relacionamento e y1 e y2 são os conjuntos de atributos de r1 e r2 assinalados na RI. A RI será

então igual a :

SL((PR(r1,y1)JN(PR(r2,y2)),F) onde F é o relacionamento sendo analisado.

Relacionamento ---> X06 = X08

r1 ---> PROJETO y1 ---> X05,X06,X07

r2 ---> EMP y2 ---> X08,X09

F ---> X06 = X08

... SL(PR(PROJETO,(X05,X06,X07)) JN

PR(EMP,(X08,X09)),X06=X08) ...

caso 2 - O relacionamento da Classe 1 não é o primeiro relacionamento da RI sendo analisado; Se os relacionamentos analisados anteriormente forem da classe 0, o tradutor2 terá um comportamento semelhante ao do caso 1 visto nesta seção. Caso os relacionamentos analisados anteriormente forem da classe 1, teremos dois sub_casos a considerar :

. As duas relações , as quais pertencem os atributos do relacionamento, já foram foram reconhecidas anteriormente em outros relacionamentos da classe 1 ; neste caso é gerado um SL(r,F) onde r é igual ao código gerado anteriormente e F é o relacionamento em questão;

. Apenas uma das relações já foi reconhecida em outros relacionamentos da classe 1 pertencentes a RI; neste caso será gerado um SL(r,F) onde r será igual a (código anterior) JN (PR(r1,y1), sendo r1 a relação não reconhecida e y1 o seu conjunto de atributos assinalados e F é o relacionamento em questão;

Relacionamento ---> X04 > X09

r ---> SL(PR(PROJETO,(X05,X06,X07) JN PR(EMP,(X08,X09)),X06=X08)

r1 ---> EMP y1 ---> X08,X09

... SL(SL(PR(PROJETO,(X05,X06,X07)) JN

PR (EMP,(X08,X09)),X06=X08)) JN

(PR(EMP,(X03,X04))),X04>X09) ...

classe 2 : Temos dois casos distintos :

caso 1 - O relacionamento da classe 2 é o primeiro relacionamento da RI, sendo analisado, ou seja, a RI não possui relacionamentos da classe 0 ou classe 1 e nenhum relacionamento da classe 2 foi ainda analisado; neste caso um SL(r,F) será gerado, sendo r igual a PR(r1,y1) onde r1 é a relação não reconhecida e y1 o seu conjunto de atributos assinalados na RI, e F é igual ao relacionamento sendo analisado.

caso 2 - O relacionamento da classe 2 não é o primeiro relacionamento a ser analisado; neste caso podemos identificar dois sub_casos:

. A relação a qual pertence o atributo usado no relacionamento já foi reconhecida em relacionamentos da classe 1 ou da classe 2 anteriormente analisados ; neste caso é gerado um SL(r,F), onde r é igual ao código gerado anteriormente e F é o relacionamento em questão;

Relacionamento ---> X04 > 100000

r ---> SL(SL(PR(PROJETO,(X05,X06,X07)) JN PR(EMP,(X08,X09))

,X06=X08) JN (PR(EMP,(X03,X04)),X04>X09)

F ---> X04>100000

... SL(SL(SL(PR(PROJETO,(X05,X06,X07)) JN PR(EMP,(X08,X09)),

X06=X08) JN (PR(EMP,(X03,X04)),X04>X09))),X04>100000) ...

. A relação não foi reconhecida em relacionamentos analisados anteriormente; neste caso é gerado um SL(r,F), onde r será igual a (código anterior) JN PR(r1,y1) sendo r1 a relação em questão e y1 seu conjunto de atributos assinalados e F o relacionamento em questão.

classe 3 : Consideramos dois casos :

caso 1 - O relacionamento da classe 3 é o primeiro relacionamento sendo analisado, não existem relacionamentos das classe 1, classe 2 ou classe 3 anteriores. Neste caso é gerado um SL(r,F) onde r é igual a PR(r1,y1) tendo r1 e y1 o significado visto em casos anteriores e F é o relacionamento em questão;

caso 2 - O relacionamento não é o primeiro a ser analisado; podemos ter dois sub_casos:

. A relação já foi reconhecida anteriormente : neste caso é gerado um SL(r,F), onde r é igual ao código já gerado e F é o relacionamento em questão;

. A relação não foi reconhecida anteriormente : teremos SL(r,F) onde r será igual a PR(r1,y1) onde r1 e y1 tem o mesmo significado dos casos anteriores e F é o relacionamento em questão.

6.2.2.3 GERAÇÃO DA CONDIÇÃO

Na terceira e última etapa Tradutor2, através dos relacionamentos da RI, gera uma expressão booleana , com o significado de uma condição, em que a RI em Álgebra Relacional é um dos operandos (ver seção 6.2.2).

6.3 TRADUTOR3

Tradutor3 (figura 6.4) tem como entrada a RI em Álgebra Relacional e como saida instruções de consulta em SQL.

Tradutor3 também utiliza tabelas geradas durante o mapeamento Lógica ---> Álgebra Relacional por Tradutor2.

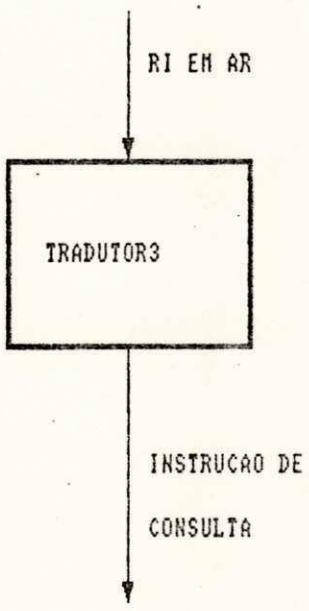

FIGURA 6.4 - TRADUTOR3

6.3.1 AS TABELAS FORNECIDAS POR TRADUTOR2

As tabelas fornecidas por Tradutor2 a Tradutor3 têm como objetivo facilitar o mapeamento da RI para instruções de consulta ao BD. As estruturas de dados destas tabelas são mostradas na figura 6.5.

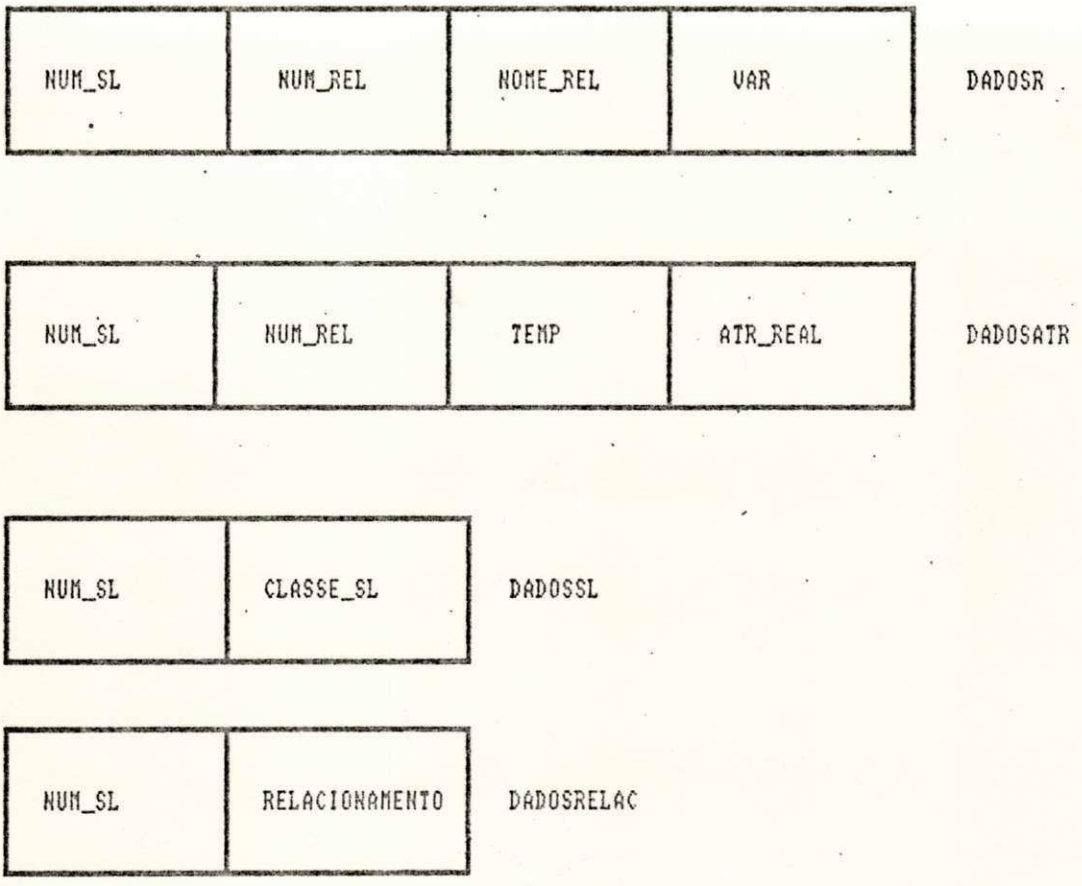

FIGURA 6.5 - AS TABELAS FORNECIDAS POR TRADUTOR2

DADOSR armazena os dados relativos à cada relação que aparece na RI em Álgebra Relacional, o SL específico em que elas são utilizadas (NUM_SL), a ordem em que aparecem (NUM_REL), o nome (NOME_REL) e a variável de consulta que percorrerá a relação nas instruções de consulta geradas (VAR).

DADOSSL armazena as informações relativas a cada operação de seleção; classe_sl contém o código do tipo da SL que depende da classe a qual pertence o relacionamento (seção 6.2.2.1) que dá origem ao SL. A ordem em que estes SLs são gerados pode ser determinada através do campo NUM_SL;

DADOSATR armazena os dados relativos aos atributos que são usados por cada operação de seleção NUM_SL contém o número do SL a que pertence o atributo; NUM_REL identifica a relação a qual pertence o atributo; TEMPORÁRIA e ATR_REAL armazenam respectivamente, a temporária e o nome correspondente do atributo.

DADOSRELAC contém os dados referentes aos relacionamentos da RI - NUM_SL guarda a mesma informação que o seu homônimo em DADOSATR, o campo relacionamento contém o próprio com suas temporárias substituidas pelos atributos do esquema a que representam.

6.3.2 O MAPEAMENTO PARA SQL

Uma consulta em SQL pode ser descrita como [8] :

SELECT <lista resultante>
INTO <relação resultante>
FROM <lista de relações>
WHERE <lista de relacionamentos>

Onde:

<lista resultante> é uma lista de elementos, separadas por vírgulas, da forma r.A, onde r é uma variável da consulta e A é um atributo da relação varrida por r;

<relação resultante> é um novo nome de relação que terá como atributos os listados em <lista resultante>;

<lista de relações> é uma lista de elementos separadas por vírgulas da forma Rr, onde R é um nome de relação do BD em questão e r é uma variável de consulta varrendo R;

⟨lista de relacionamentos⟩ são expressões booleanas sobre comparações da forma r.A ⟨op⟩ r.B / r.a ⟨op⟩ constante, onde op pertence ao conjunto de operadores (= , ⟨= , ⟩= , ⟩ , ⟨ , ⟨⟩).

Os parâmetros descritos acima são determinados por Tradutor3 através da RI em Álgebra Relacional das tabelas descritas em 6.3.1.

Os parâmetros da consulta são determinados da seguinte maneira :

⟨lista de relações⟩ : composta pelas relações referentes a RI, armazenadas na tabela DADOSR;

⟨lista_resultante⟩ : composta pelos atributos que estejam envolvidos nos relacionamentos pertencentes a RI;

⟨relacionamentos⟩ : Composta pelos relacionamentos existentes na tabela DADOS_RELAC, referentes à RI em questão.

Para o exemplo utilizado trĭamos os seguintes valores, determinados diretamente das tabelas :

⟨lista de relações⟩ ---⟩ EMP E1 , PROJETO P , EMP E2

⟨lista resultante⟩ ---⟩ matrĭcula E1, salário E1, cod P, gerente P, depto P, matrĭcula E2, salário E2

⟨relacionamentos⟩ ---⟩ P.gerente = E2.mat
E1.salário ⟩ E2.salário
E1.salário = "10000"
D.chefe = E1.mat
D.depto = P.depto

6.4 O GERADOR DE ROTINAS

O Gerador de Rotinas é responsável pela rotina referente à cada chave de acesso ao Catálogo de RIs. Para cada RI pertencente à lista encabeçada pela chave, é feito o mapeamento para uma instrução de consulta em SQL. Com estas instruções de consulta o Gerador de Rotinas monta a rotina referente à chave em questão.

Mostramos abaixo o aspecto que teria uma rotina específica para uma chave fictícia XX.

```
Case chave of
    .
    .
    .
    XX : BEGIN
            . Instrução de Consulta referente a RI1, com o
              resultado armazenado em R01
            IF R01 op 0  EXIT
            . Instrução de Consulta referente a RI2, com o
            resultado armazenado em R02

            IF R02 op 0 EXIT
            .
            .
            .
            . Instrução de Consulta referente a RIn, com o
            resultado armazenado em Rn
            IF RIn op 0 EXIT
            . TRANSAÇÃO DE ATUALIZAÇÃO
```

6.5 O PROGRAMA DE CRÍTICA

Quando o Gerador recebe o Catálogo de RIs, em LRI, do Catalogador, ele lê a chave de acesso à primeira lista de RIs, gerando a primeira entrada do comando CASE. O Gerador passa a ler as RIs pertencentes à lista acessada pela chave em questão. O Programa de Crítica é gerado à medida que as RIs vão sendo processadas.

A Instrução de Atualização só será efetivada se todas as condições, representadas pelas RIs mapeadas para instruções de Consulta em SQL, forem satisfeitas.

Consideremos como exemplo uma transação de Inclusão de um projeto na relação PROJETO, supondo a existência da definição Grande_Projeto (exemplo 2 do capítulo 5) na tabela de definições e as RIs abaixo :

```
OP[I,PROJETO] <--- PROJETO(X01,X03,X02,-),X02>"100Milhões",
                   EMPREGADO(X04,-,-,-,-,-,X05),
                   X05<"1978",X03=X04 .

OP[I,PROJETO] <--- PROJETO(X01,X02,-,-),
                   EMPREGADO(X03,-,-,-,-,-,-),X02=X03 .

OP[I,PROJETO] <--- PROJETO(X01,X02,-,-),
                   EMPREGADO(X03,-,-,X04,-,-,-),X02=X03,
                   EMPREGADO(X05,-,-,X06,-,-,-),X04>X06.
```

O trecho do Programa de Crítica relativo a chave de acesso OP[I,PROJETO] terá o seguinte aspecto :

```
BEGIN
  WHILE NOT(EOF(ARQ_TRANSAÇÃO)) DO
   BEGIN
     READ(ARQ_TRANSAÇÃO,AUX);
     CASE AUX.CHAVE OF
        .
        .
        .
        .
        .
       'IPROJETO' : BEGIN
                      IF  $orçamento > '100Milhões' THEN
                        BEGIN
                          SELECT  E.MAT
                          INTO R01
                          FROM EMPREGADO E
                          WHERE E.MAT = $gerente    AND
                                E.DATA_ADMISSÃO > '1978'
                          IF R01 = 0 THEN EXIT

                        END;

                      SELECT E.MAT
                      INTO R02
                      FROM EMPREGADO E
                      WHERE E.MAT = $gerente
                      IF R02 = 0 THEN EXIT

                      SELECT E2.MAT
                      INTO R03
                      FROM EMPREGADO E1, EMPREGADO E2
```

```
                    WHERE E1.MAT = $gerente   AND
                          E2.SALÁRIO > E1.SALÁRIO
                  IF R03 <> 0 THEN EXIT

                    INSERT INTO PROJETO
                    PROJ = $proj
                    GERENTE = $gerente
                    ORÇAMENTO = $orçamento
                END;
                .
                .
                .
      END; /*CASE*/
   END; /*WHILE*/
END./*PROGRAMA DE CRÍTICA*/
```

7. CONCLUSÕES

A necessidade premente de mais e mais ferramentas para o desenvolvimento automatizado de software é incontestável. Este é um dos caminhos para o enfrentamento da chamada crise do software : da boa resolução desta crise dependerá o próprio futuro da informática.

A ferramenta que estamos propondo libera os programadores da tediosa e longa tarefa de escrever programas de crítica, permitindo que concentrem seus esforços nas aplicações de banco de dados mais visíveis aos usuários, amenizando a insatisfação destes últimos pela longa espera (back_log).

Para desenvolver esta ferramenta precisamos antes de tudo de definir uma linguagem para especificação das regras de integridade, surgindo daí a LRI. Esta linguagem nos parece bastante potente (o mesmo poder de expressão da lógica de primeira ordem) ; não foram detectados casos nos quais não houve possibilidade da regra ser expressa por ela. O interesse em usar lógica foi grande porque, além de ser altamente expressiva, tínhamos conhecimento de um trabalho [5] em que o mapeamento de lógica para a álgebra relacional estava quase que totalmente resolvido. Da álgebra relacional para uma linguagem de um GBD relacional (objetivo final do trabalho) seria um passo trivial.

Um inconveniente da LRI é a sua compreensão por parte de pessoas não familiarizadas com a linguagem em lógica : daí surgiu a NRI. O objetivo principal do uso da NRI é dar condições a estas pessoas de poderem fornecer regras, mesmo que não tenham nenhum conhecimento de lógica.

Houve um preço a ser pago por esta facilidade ; a NRI, embora seja chamada por nós de "natural", apresenta-se , na realidade, como uma linguagem ainda bastante rígida, e somente regras bastante simples podem ser expressas por ela. O esforço desenvolvido para . fazer a sua tradução para a LRI foi grande, demandando um tempo de programação equivalente a 70% do tempo total gasto. Acreditamos que a utilização da NRI é viável e pode ser bastante satisfatória, desde que tenha o seu poder de expressão expandido,principalmente através de melhorias no seu repertório de locuções, sinônnimos, operadores, quantificadores e pertinências.

Quanto ao problema da sua pouca eficiência, devemos considerar que um programa de crítica é gerado e compilado uma única vez e só sofrerá alterações se houver mudanças no quadro de RIs do BD. Com isto, em um sistema como o nosso, a eficiência tem muito pouco peso.

Quanto ao programa de crítica gerado pelo sistema, não houve condições técnicas de submetê-lo à compilação. A verificação de sua correção foi feita manualmente.

A linguagem LRI apresenta também algumas falhas ; uma destas, é que o sistema , quando do . mapeamento LRI ---> AR, não considera relacionamentos em que os atributos_operandos pertençam a mesma relação. Supondo o acréscimo de um novo campo (DATA_PRM) na relação EMP, Este campo guardaria a data da última promoção de um empregado. Seja a' RI abaixo :

OP[M,EMP] <--- EMP(X01,-,-,-,-,X02,-,X03),X02(X03.

como os atributos operandos envolvidos no relacionamento pertencem à mesma relação, este relacionamento será ignorado e

devido a sua unicidade a própria RI não será considerada.

Outro aspecto negativo da LRI é a inexistência de um controle sobre o número de relacionamentos permitidos, principalmente porque este número não é determinado. Com isto, pode-se fornecer uma RI completamente sem sentido ao sistema, e este não detectar erro, desde que a sua sintaxe esteja correta. O único controle que o sistema possui sobre o número de relacionamentos de uma RI é que este número não pode ser igual a zero - neste caso o sistema ignora a RI dando uma mensagem para o usuário. Os atributos_operandos dos relacionamentos pertencentes a RIs fornecidas em LRI não sofrem qualquer tipo de crítica pelo sistema - com isto pode ocorrer que um relacionamento comparando atributos não semelhantes (o que teoricamente inviabilizaria a comparação) seja aceito como correto, gerando uma instrução de consulta incorreta. Este problema não ocorre quando a RI é fornecida em NRI, porque durante o processamento da mesma, é feito a verificação de compatibilidade entre os atributos-operandos.

Estes foram os problemas detectados por nós durante o processo de elaboração do sistema. Alguns, acreditamos, são de fácil solução. No caso da verificação da compatibilidade, por exemplo, poderíamos utilizar, com algumas modificações, o mesmo módulo que faz isto no caso da RI ser fornecida em NRI.

As sugestões de continuidade e aperfeiçoamentos do nosso trabalho se prendem, principalmente, à NRI. Acreditamos que podemos aumentar a sua flexibilidade através, principalmente, de um melhoramento substancial de sua de interação com o usuário. Acreditamos também que poderemos aumentar a potência da LRI,

através do uso de formas de fórmulas de primeira ordem mais potentes como sugerido em [9].

Outro desenvolvimento futuro por nós sugerido seria o uso de relações com chaves compostas, o que não foi considerado no nosso trabalho.

Outros trabalhos futuros que poderão ser desenvolvidos diz respeito as definições ; uma sugestão seria o uso de definições na definição de novas definições, o que não ocorre no nosso trabalho. Com relação a remoção de definiões o nosso sistema não remove automaticamente as RIs que utilizaram as mesmas na sua elaboração. Imaginamos que que o problema poderá ser resolvido criando apontadores para RIs que usem definições em sua elaboração, no caso de uma definição ser removida o sistema automaticamente remove estas RIs.

8.BIBLIOGRAFIA

[1] - DATE, C. J., An Introduction to Database Systems, Addison Wesley, vol 2, 1983.

[2] - CHAMBERLIN, D. D. et al, SEQUEL2 : A Unified Approach to Data Definition, Manipulation and Control, IBM Research Development, vol 20, No 6, novembro, 1977.

[3] - TSICHRITZTS, D. C. & KLUGS (eds), The ANSI/X3/SPARC DBMS Framework : Report of the Study Group on Data Base Management Systems, 3, 3, 1978.

[4] - CASANOVA, M. A. et al, Programação em Lógica, V Escola de Computação, 1986.

[5] - EMDE BOAS, C. D. et al, Storing and Evaluating Horn-Clause Rules in a Relational Database, IBM Research Development, vol 30, No 1, janeiro, 1986.

[6] - ULLMAN, J. D. , Principles of Database Systems, 2o edição, Computer Science Press, 1982.

[7] - AHO, A. V. et al, Principles of Compiler Design, Addison Wesley, Amman, 1978.

[8] - CASANOVA, M. A. et al, Princípios de Gerencia de Banco de Dados Distribuidos, Campus, 1985.

[9] - LING, T. W., Integrity Constraint Checking in Deductive Databases Using Prolog Not-Predicate, DATA & KNOWLEDGE ENGINEERING, 2:2, 1987, 145 - 148.

[10] - MENEZES, A. A. et al, Proposta de uma Linguagem de Restrição de Dados, Anais do 3o Simpósio Brasileiro de Banco de Dados, Recife, 1988.

[11] - DRUMOND, G. F. AND SAMPAIO, M. C., Uma Ferramenta Para a Definição e Verificação de Regras de Integridade em Banco de Dados, Anais do 3o Simpósio Brasileiro de Banco de Dados, Recife, 1988.

[12] - CASANOVA, M. A. et al, A Monitor Esforcing Referential Integrity, Anais do 3o Simpósio Brasileiro de Banco de Dados, Recife, 1988.