Universidade Federal da Paraíba

Centro de Ciências e Tecnologia

Coordenação de Pós-Graduação em Informática

NORMALIZADOR - Uma ferramenta para Geração de Esquemas Relacionais Normalizados

JORGE AUGUSTO LACERDA DE FARIAS

CAMPINA GRANDE

DEZEMBRO - 1992

JORGE AUGUSTO LACERDA DE FARIAS

NORMALIZADOR - Uma ferramenta para Geração de Esquemas Relacionais Normalizados

Dissertação apresentada ao Curso de MESTRADO EM INFORMÁTICA da Universidade Federal da Paraíba, em cumprimento às exigências para obtenção do Grau de Mestre.

ÁREA DE CONCENTRAÇÃO: CIÊNCIA DA COMPUTAÇÃO

Ulrich Schiel

(Orientador)

Pedro Sérgio Nicolletti

(Co-orientador)

Campina Grande

Dezembro - 1992

F226n Farias, Jorge Augusto Lacerda de
Normalizador : uma ferramenta para geracao de esquemas relacionais normalizados / Jorge Augusto Lacerda de Farias. - Campina Grande, 1992.
118 f. : il.

Dissertacao (Mestrado em Informatica) - Universidade Federal da Paraiba, Centro de Ciencias e Tecnologia.

1. Banco de Dados 2. Normalizador 3. Base de Dados - I. Schiel, Ulrich, Dr. II. Nicolletti, Pedro Sergio, M.Sc. III. Universidade Federal da Paraiba - Campina Grande (PB) IV. Título

CDU 004.65(043)

NORMALIZADOR - UMA FERRAMENTA PARA GERAÇÃO DE ESQUEMAS RELACIONAIS NORMALIZADOS

JORGE AUGUSTO LACERDA DE FARIAS

DISSERTAÇÃO APROVADA EM 11.12.1992

ULRICH SCHIEL, Ph.D.
Presidente

PEDRO SERGIO NICOLLETTI, M.Sc
Componente da Banca

DECIO FONSECA, Ph.D.
Componente da Banca

CAMPINA GRANDE, 11 de dezembro de 1992

Agradecimentos

A meu Deus, pela iluminação para chegar até aqui.

Aos meus familiares que sempre acreditaram em mim, Dona Terezinha (minha mãe), Sr. Zé Pequeno (meu pai), Lena, Kay, Jean, Lucita, Juci e Fábio (meus irmãos).

A Rozana (minha musa), que participou ao meu lado das alegrias e sofrimentos no decorrer deste trabalho.

A meus professores Ulrich Schiel, Pedro Sérgio Nicolletti, Marcus Sampaio e Joseluce Farias, pela colaboração e dedicação no desenvolvimento deste trabalho.

A meus companheiros do mestrado: André, Bezerra, Fernando (Mandi), Lauro (Japa), Luiz Maurício (Baiano), Lunguinho, Madalena, Vicente (Vick) e Vladimir (Delegado), pelo companheirismo e apoio durante este curso.

Aos meus amigos de longas caminhadas Olavo, Marlene, Flaubert, Rembrandt, Aldo e Ricardo, pela amizade.

Sinceramente, Obrigado!

Resumo

Obter um esquema relacional normalizado é um objetivo dos projetistas de banco de dados. Isto possibilita uma maior confiança e independência durante a atualização da base de dados.

Este trabalho apresenta uma ferramenta de apoio a projeto de banco de dados relacional, que automatiza o processo de normalizar relações. O projetista esquematiza os dados de sua aplicação segundo um modelo orientado à normalização [Smith 85]. A ferramenta encontra possíveis redundâncias na modelagem inicial e emite informações para sua eliminação [Bernstein 76] [Diederich-Milton 88]. Em seguida é fornecido um esquema relacional resultante normalizado em 5ª forma normal, contendo o projeto automático de chaves e identificação das junções naturais neste esquema.

Abstract

One of the main objectives of a database designer is to obtain normalized tables. This makes possible a greater reliability and independence during database updates.

This work presents a helpful tool on logical and physical projects of relational-databases, which makes the normalization process automatic. The designer organizes the data of his application using a normalization oriented model [Smith 85]. The tool finds possible redundancies in the first schema and gives information to its elimination [Bernstein 76] [Diederich-Milton 88]. After this, a set of fully normalized tables is derived containing the automatic key project and identification of joins in this set.

Sumário

Capítulo I

Capítulo II

Capítulo III

Capítulo VI

Lista de Figuras

Capítulo I

1 - Introdução

Motivação

Projetar um Banco de Dados Relacional (BDR), resumidamente significa decidir que relações são necessárias e quais deveriam ser seus atributos, de forma que represente o mundo real da aplicação. Em outras palavras, uma base de dados manipulada por uma aplicação, deve ser coerente com o mundo real inerente. Entretanto, nem sempre esta afirmação é verdadeira, podendo as relações pertencentes a um Banco de Dados (BD) ter características indesejáveis, tais como **redundâncias e inconsistências** que poderão levar a anomalias durante a manipulação dos dados. A técnica de projeto por Normalização tem por objetivo uma organização dos dados que evita estas características indesejáveis.

Para ilustrar os objetivos da normalização, apresentaremos um exemplo. Suponha uma relação denominada "DOAÇÕES" instanciada segundo a figura 1.1 . A princípio, perceba a redundância! As três primeiras tuplas repetem os atributos: nome, rua, número e bairro. Esta redundância poderá causar problemas futuros, por exemplo suponha que o doador cujo nome seja Manuel tenha mudado de endereço. Naturalmente deseja-se que seu endereço seja modificado na base de dados, porém suponha também que o aplicativo desenvolvido para realizar esta tarefa atualize apenas a primeira tupla. A partir deste ponto o BD torna-se inconsistente. Agora, imagine que alguém decida eliminar do BD toda doação cuja data seja anterior a junho de 1992. Neste caso seriam eliminados os doadores de nome Manuel e

Maria. Quando chegar a hora de solicitar mais donativos, dois excelentes doadores serão esquecidos.

Nome	Rua	Número	Bairro	Valor doado	Data
Manuel	rua X	10	A	5000,00	01/90
Manuel	rua X	10	A	6000,00	02/91
Manuel	rua X	10	A	3000,00	03/92
Maria	rua Y	20	B	4000,00	04/92
Luíza	rua Z	10	B	4000,00	08/92
Pedro	rua W	10	B	3000,00	09/92

Fig. 1.1 - Relação das doações com anomalias

Os problemas anteriores poderiam ser evitados se tivéssemos duas relações ao invés de apenas uma, como aparece na figura 1.2. Desta maneira, qualquer doação e alteração de dados das doações não afetaria os dados dos doadores, possibilitando uma maior confiança e independência dos dados no projeto do BD. O processo que garante isto chama-se normalização.

Nome	Rua	Número	Bairro
Manuel	rua X	10	A
Maria	rua Y	20	B
Luíza	rua Z	10	B
Pedro	rua W	10	B

Nome	Valor_doado	Data
Manuel	5000,00	01/90
Manuel	6000,00	02/91
Manuel	3000,00	03/92
Maria	4000,00	04/92
Luíza	4000,00	08/92
Pedro	3000,00	09/92

Fig. 1.2 - Relações Doadores e Doações

A teoria da normalização para o modelo relacional está baseada no conceito das formas normais. Codd inicialmente definiu a 1ª Forma Normal em um artigo sobre o modelo relacional [Codd 70], em seguida ele explicou minuciosamente a 1ª, 2ª e 3ª Formas Normais em [Codd 72]. Posteriormente Boyce e Codd apresentaram um aperfeiçoamento da 3ª Forma Normal denominada Forma Normal de Boyce-Codd em [Codd 74]. Graças a Fagin foram definidas posteriormente a 4ª e a 5ª Forma Normal, respectivamente em [Fagin 77] e [Fagin 79]. Uma determinada relação pertencente a um esquema relacional estará em uma dada Forma Normal se ela respeitar ao conjunto de restrições que definem esta Forma Normal.

Obter um esquema relacional normalizado é um objetivo dos projetistas de BD. Existem três métodos para projetar relações normalizadas: a partir de um **modelo conceitual**; utilizando o método de análise de relações existentes denominado "**decomposição de relações**" [Date 86] [Setzer 87] [Ullman 89]; o de **síntese de relações** a partir das regras de integridade dos dados. O objetivo deste trabalho é o desenvolvimento de uma ferramenta de apoio a projeto de BDR, que automatize o processo de normalizar relações, baseado no método de **síntese de relações**. Esta escolha é explicada mais adiante.

Algumas Propostas Existentes

Afirmar que, partindo de um modelo semântico, por exemplo o Modelo Entidade-Relacionamento de Chen [Chen 76] ou suas extensões, encontra-se sempre esquemas relacionais normalizados, é um pouco perigoso. Derivar um esquema semântico a partir da observação do mundo real é um processo não formal, manual e sujeito a erros do projetista (dois projetistas modelando uma mesma aplicação, sem se comunicarem, dificilmente obterão o mesmo esquema semântico). A riqueza deste modelo semântico, possibilita

várias maneiras de representação das regras de integridade do mundo real. Isto é um problema para o processo de normalização, pois este modelo não é direcionado à normalização como é o modelo de diagramas de dependências (explicado no capítulo II). Na prática, após o mapeamento do esquema conceitual para um esquema relacional (um processo com regras bem definidas), usa-se os conceitos de formas normais como uma técnica de validação do esquema semântico. Ao tentar verificar se as relações estão normalizadas ocorre uma mistura de níveis de semântica, pois o projetista é obrigado a sair do âmbito das relações (nível operacional) e tem que passar a pensar em regras de integridade do mundo real (nível conceitual) [Setzer 87].

Uma técnica baseada no modelo relacional de projetar relações normalizadas, é o método de **decomposição de relações**. Parte-se dos relatórios necessários à aplicação, extraindo-lhes os possíveis atributos necessários e considerando-os como uma relação universal, que será analisada e possivelmente particionada em diversas relações normalizadas. A utilização deste método é um processo pouco trivial e sujeito a erros. Há casos em que uma relação pode ser quebrada de diferentes maneiras, podendo ocorrer uma decomposição sem projeções independentes, o que levará à perda de informação do mundo real [Rissanen 77].

Um outro método sintetiza as relações normalizadas a partir do universo de propriedades (dependências) entre os atributos no sentido **conceitual**. Nas versões mais atualizadas deste método não existem os problemas de perda de informação e mistura de níveis de semântica comentados anteriormente, sendo por este motivo a linha de pesquisa adotada neste trabalho.

O primeiro trabalho nesta linha de pesquisa denominada de **síntese de relações** foi o de Wang e Wedekind [Wang-Wedekind 75], que posteriormente foi aperfeiçoado e corrigido por Bernstein [Bernstein 76]. Ele realizou um minucioso trabalho sobre este método determinando a complexidade de seus algoritmos.

Fagin posteriormente descreveu uma comparação entre o método da decomposição e o de síntese de relações [Fagin 77a], fornecendo exemplos deste último método. Porém, Fagin foi infeliz por não ter tratado de transitividade entre associações de atributos, em função disto, o problema de **perda de informações** reapareceu.

Este problema em função de transitividade foi resolvido por Raver e Hubbard [Raver-Hubbard 77] em um trabalho para projeto lógico automático de BD, voltado para o gerenciador IMS pertencente à IBM, que implementa o modelo operacional hierárquico com algumas características de rede. Este trabalho se baseia no uso de um grafo representando as associações de atributos, onde os nós do grafo representam os atributos e os arcos as associações entre eles. Desta maneira a transitividade da forma:

A -> B, B -> C e A -> C

(onde A, B, C são atributos e '->' é um arco) torna-se óbvia e visível, portanto o último arco é eliminado. Posteriormente Martin melhorou o trabalho de Raver-Hubbard, batizando o grafo de "diagrama de bolhas" e o aplicou para o modelo relacional [Martin 77]. Em ambos os trabalhos os diagramas de bolhas geram esquemas definitivos, denominado por Raver-Hubbard de "representações canônicas" e por Martin de "esquemas canônicos". Martin garante que seu "esquema canônico" está em 3ª Forma Normal e contém um número mínimo de relações.

É importante mencionar que os trabalhos de Raver-Hubbard e Martin se baseiam em relatórios necessários às aplicações. Em outras palavras, de relações disfarçadas de onde se extraem as regras de integridade do mundo real, portanto reaparecendo aqui a mistura de níveis de semântica. Graças ao trabalho de H. C. Smith, que organizou e aperfeiçoou as pesquisas de Raver-Hubbard e Martin, este problema foi eliminado [Smith 85]. Smith sugere a construção de diagramas de dados a partir de fatos extraídos do mundo real (de modelos descritivos ou conceituais). O diagrama dá origem a esquemas relacionais resultantes semelhantes aos de Martin, porém, as relações se encontram em 5ª Forma Normal. O trabalho de Smith, juntamente com várias extensões, serviu como base filosófica para a implementação de um modelo de dados orientado ao processo de normalização utilizado em nosso trabalho (ver capítulo II).

Porém, torna-se necessário comentar que, utilizando o diagrama de dependências de Smith para representar os dados de uma aplicação, redundâncias (de atributos e de associações entre eles) podem aparecer, como por exemplo transitividade entre associações de atributos e outras redundâncias indesejáveis. Para resolver este problema, nosso trabalho juntou os conceitos de **diagrama de dependências** aos conceitos de **cobertura mínima** inicialmente pesquisado por Bernstein em [Bernstein 76], posteriormente aperfeiçoados por Diederich e Milton em [Diederich-Milton 88] Esta junção de conceitos foi fundamental para que, a partir de um diagrama de dependências esquematizando uma aplicação, se obtenha esquemas relacionais resultantes normalizados e mínimos, tanto em relação à quantidade de relações quanto à de atributos pertencentes a cada relação (ver capítulo III).

NORMALIZADOR

Este trabalho apresenta uma ferramenta de apoio a projeto de Banco de Dados Relacional, que automatiza o problemático processo de normalizar relações. A ferramenta é intitulada **NORMALIZADOR** [Farias 92].

A grande maioria das metodologias de projeto ascendente de Banco de Dados Relacional segue um padrão: primeiro, o mundo real é esquematizado segundo um modelo semântico, em geral o Modelo Entidade-Relacionamento (E-R) de Chen [Chen 76] ou suas extensões; em seguida, o esquema semântico é mapeado para um esquema relacional (no mínimo, um conjunto de relações); finalmente, todas as relações são normalizadas como forma de validação do projeto semântico. Assumimos que o leitor está bem familiarizado com a teoria do modelo relacional definido por Codd [Codd 70] [Codd 72].

Para normalizar relações o projetista usa, quase sempre, o método de decomposição de relações, o qual pode acarretar perda de informação [Maier 83] [Fagin 77-a]. Além do mais, o projetista é sobrecarregado com dois níveis de semântica: a semântica das entidades (para o esquema E-R) e a semântica dos atributos das relações (para a normalização do esquema relacional).

Nosso objetivo é simplificar a vida do projetista, permitindo-lhe trabalhar com uma semântica uniforme e poupando-lhe da dispendiosa tarefa de normalizar relações. Para isto, foi necessário construir um modelo orientado ao processo de normalização (semântica de atributos); chamamos um esquema neste modelo de **diagrama de dependências** (funcionais e multivaloradas), inspirado em [Smith 85], com várias extensões (que são explicadas mais adiante). Uma ferramenta chamada **NORMALIZADOR** assiste ao usuário na tarefa de projetar seu Banco de Dados.

O projetista desenha na tela de seu computador um diagrama de dependências. **NORMALIZADOR** depura o diagrama eliminando redundâncias (de atributos e de associações entre eles) e fornece ao usuário o esquema relacional correspondente ao diagrama depurado, normalizado até a 5ª Forma Normal (incluindo BCNF).

A figura 1.3 mostra uma comparação entre a metodologia de projeto ascendente de relações normalizadas e a nossa proposta pelo uso da ferramenta **NORMALIZADOR**.

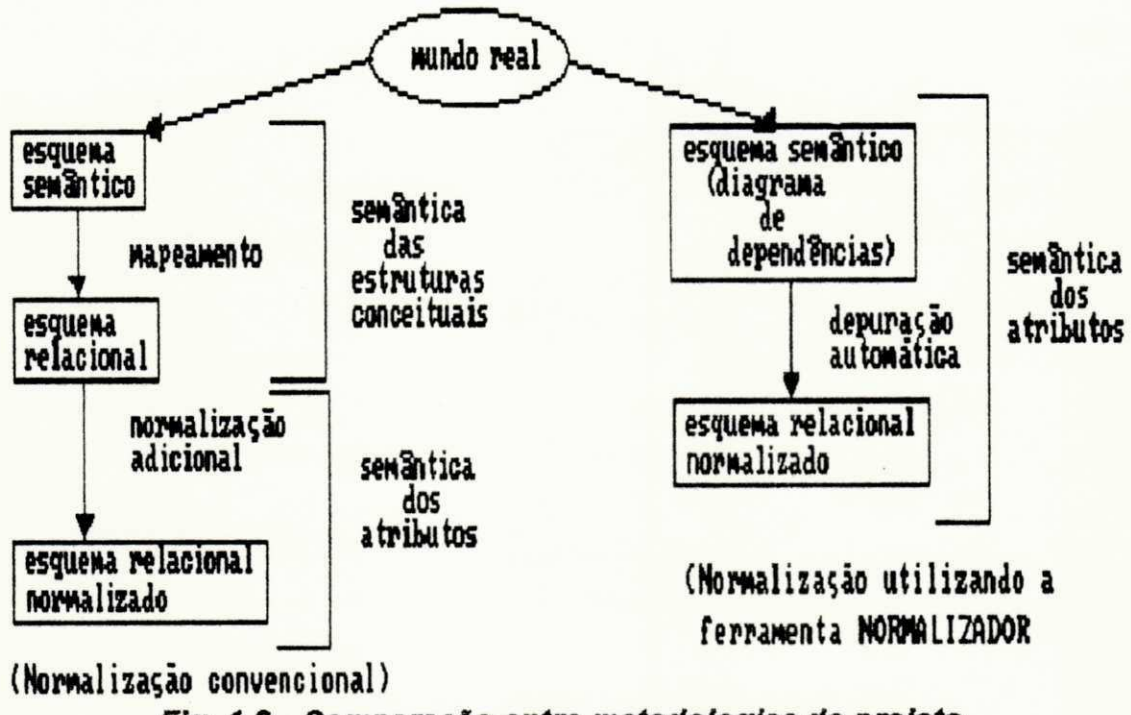


Fig. 1.3 - Comparação entre metodologias de projeto

Um outro aspecto importante de nossa proposta é que o modelo orientado à normalização que foi implementado, possibilita que o esquema relacional resultante determine de forma automática o projeto de chaves: chaves primárias, chaves alternativas e chaves estrangeiras ("foreign keys") das relações (informações preciosas para o bom desempenho do BD, pois influenciam a escolha de índices de acesso, por

exemplo), determinando também as possíveis junções naturais entre as relações para o esquema relacional gerado.

Tanto um usuário final quanto um projetista de BD se beneficiam dos serviços oferecidos por nossa ferramenta, pois sua interface é bastante amigável e totalmente interativa. Para usá-la o usuário representa sua aplicação segundo o modelo (bastante simples) orientado à atributos e dependências (ver capítulo II); a ferramenta auxilia a geração de um esquema relacional normalizado até a 5ª Forma Normal; além de encontrar as chaves para cada relação e identificar as junções naturais entre as mesmas. É importante lembrar que o usuário não precisa conhecer nem dominar o conceito das Formas Normais (1ª, 2ª e 3ª Formas Normais ; Forma Normal de Boyce-Codd; 4ª e a 5ª Forma Normal), nem tão pouco se preocupar com o projeto de chaves, pois o uso da ferramenta livra o usuário destas preocupações.

A figura 1.4, mostra um diagrama da ferramenta **NORMALIZADOR**, seguido da descrição de seus módulos básicos.

Fig. 1.4 - Hierarquia da ferramenta NORMALIZADOR

O módulo **Editor de diagramas** é um ambiente totalmente interativo, dispondo de características peculiares à edição gráfica que oferece condições ao projetista para representar suas aplicações. Em outras palavras, é um editor gráfico que possibilita o desenho de um diagrama de dependências para uma aplicação, representando de maneira gráfica sua semântica. O editor é totalmente direcionado ao modelo orientado ao processo de normalização, isto é, ele evita a construção de diagramas que fujam ao modelo. Exemplos práticos da utilização deste editor serão apresentados no capítulo V.

O módulo **Detector de redundâncias** é o responsável pela depuração de um diagrama de dependências, detectando quando existem redundâncias indesejáveis ao processo de normalização. Este módulo encontra automaticamente as possíveis redundâncias, emitindo explicações interativas informando ao usuário de que maneira o diagrama deve ser corrigido. Este detector oferece condições para geração de esquemas relacionais enxutos (sem redundâncias e portanto mínimos em relação ao número de relações e ao número de atributos para cada relação) (ver capítulo III).

O módulo **Gerador de esquemas relacionais**, é o responsável pela construção do esquema relacional normalizado apresentado ao usuário, bem como a determinação das chaves das relações: primárias, alternativas e estrangeiras, além de uma descrição de como poderá ser feita a navegação pelo esquema gerado através de junções naturais. De posse deste tipo de informação o usuário estará apto a utilizar uma linguagem de definição e manipulação de BD, para interagir com o módulo SGBDR (Sistema Gerenciador de Banco de Dados Relacional), no qual pode ser utilizado por exemplo o DB2 [Date 90], INGRES [Stonebraker 76], Oracle [Oracle 84], etc. (Ver capítulo IV).

Neste capítulo foram relatados os benefícios de se projetar relações normalizadas, acompanhados de um panorama global de pesquisa e escolha da estratégia ideal para o processo de normalização, que é o objetivo de nosso trabalho. Mostramos também uma visão geral da ferramenta **NORMALIZADOR**, usada para projetar relações normalizadas através do método de síntese de relações, bem como o papel dos módulos principais desta ferramenta.

No Capítulo II (Modelo de Dados) é apresentada uma avaliação mais profunda da estratégia escolhida para o processo de normalização, através do modelo de dados baseados nos diagramas de dependências, que servem para esquematizar uma análise do mundo real.

No Capítulo III (Detectando Redundâncias) é explicada como foi feita a união dos conceitos de diagrama de dependências aos conceitos de cobertura mínima (utilizados para detectar as possíveis redundâncias), mostrando uma nova maneira bem mais eficiente para encontrar uma cobertura mínima, graças aos aperfeiçoamentos realizados nos métodos iniciais de Bernstein.

No Capítulo IV (Gerando Esquemas Relacionais Normalizados) é mostrado como, a partir de um diagrama de dependências enxuto, se obtém um esquema relacional normalizado, acompanhado do projeto automático das chaves para as relações, identificação das junções entre as relações e finalmente uma prova informal de que as relações estão normalizadas.

O Capítulo V (Arquitetura do Sistema) descreve a estrutura lógica do sistema, seus principais módulos e funcionalidades, assim como a interface do sistema com o usuário, seguida de uma simulação de utilização da ferramenta.

No Capítulo VI (Estrutura de Dados) é apresentada a estrutura de dados utilizada para dar suporte a ferramenta.

O último Capítulo se refere às conclusões e contribuições obtidas com o trabalho. Para trabalhos futuros, são feitas algumas recomendações e sugestões relevantes para implementar melhorias no **NORMALIZADOR**.

2 - Modelo de Dados

Neste capítulo apresentaremos o modelo de dados orientado ao processo de normalização. Um esquema neste modelo denomina-se diagrama de dependências funcionais e multivaloradas entre os atributos do banco de dados. Um diagrama de dependências consiste nos dados de entrada para **NORMALIZADOR**.

2.1 - Um Novo Procedimento para Projeto de Banco de Dados

O projeto conceitual de um BD para uma aplicação, é feito em função da representação das dependências entre os atributos da aplicação. É de fundamental importância uma fiel representatividade destas regras em relação aos fatos verdadeiros do mundo real. Procedendo-se desta maneira evita-se os problemas de **redundâncias** e **inconsistências**, permitindo uma maior **confiança** e **independência** aos dados representados no projeto lógico (ver capítulo I).

A representação das regras de integridade do mundo real é realizada esquematizando-as segundo um modelo conceitual. O modelo de dados baseado nos diagramas de Smith (**método de síntese de relações**) foi o escolhido para o nosso trabalho, uma vez que, este modelo está livre dos problemas encontrados nas outras propostas (ver capítulo I). Esquematizando-se uma aplicação neste modelo através de um diagrama de dependências (regras do mundo real), evita-se a **mistura semântica** e a **perda de informações** do mundo real, pois o projetista trabalha com uma semântica uniforme

(apenas a semântica de atributos) para construir um diagrama e, a partir deste obtêm-se relações normalizadas, sem haver a necessidade de quebra de relações como ocorre no **método de decomposição**. Um outro aspecto importante é que a semântica deste modelo é bastante simples, não havendo várias formas de representação de uma mesma informação, além de ser totalmente direcionado ao processo de normalização. A baixa complexidade do modelo facilitou consideravelmente sua implementação e, principalmente, proporciona um fácil aprendizado para o usuário. É importante salientar que o usuário, ao desenhar um diagrama de dependências, não precisa conhecer nem dominar os conceitos de formas normais, pois o modelo esconde estes conceitos. A representação do modelo de Smith através de diagramas de dependências é mais uma vantagem, pois ela oferece ao usuário uma visão gráfica das regras de integridade do mundo real.

Representações gráficas tem sido utilizadas para ilustrar as restrições que definem as formas normais. Codd utiliza uma seta simples "-->", em diagramas ou texto, para mostrar uma dependência funcional entre conjuntos de atributos "A" e "B" de uma relação R ["R.A --> R.B"] [Codd72] [Codd 74]. Fagin usa uma seta dupla "-->>" para exemplificar a dependência multivalorada entre um empregado e seus filhos ["EMPREGADO -->> FILHOS"] [Fagin 77] [Fagin 79]. Date denomina os diagramas com setas simples, associando grupos de campos de dados, de "diagrama de dependência funcional" [Date 86]. Martin batizou de "diagramas de bolhas", os grafos contendo setas simples e duplas conectando campos dentro de círculos [Martin 77].

Embora todas estas representações citadas expliquem as dependências relevantes para uma forma normal, elas não são suficientes para gerar relações totalmente normalizadas. As regras utilizadas na construção destes diagramas não são rigorosas. Um novo procedimento de projeto lógico de BD

(expandindo as convenções de Codd, Fagin, Date e Martin) é mostrado neste capítulo. Este procedimento é baseado no uso do modelo orientado ao processo de normalização, com regras de construção suficientemente rigorosas para permitir a construção de um projeto lógico consistente, que será ponto de partida para gerar um projeto físico de relações normalizadas. Tomando como base este novo procedimento, dois conceitos fundamentais serão definidos: dependência funcional e dependência multivalorada.

2.2 - Dependência Funcional e Dependência Multivalorada

Considere que **A** e **B** são atributos simples ou conjunto de atributos. "Existe uma dependência funcional de **A** para **B** (**A** --> **B**) se, em qualquer ponto do tempo, um fato sobre **A**, determina um fato sobre B" [Smith 85]. Por exemplo, existe uma dependência funcional do numero de matrícula de um empregado, para seu nome e salário. Denotada da seguinte forma:

MATR_EMPR --> NOME, SALÁRIO

"Existe uma dependência multivalorada de **A** para **B** (**A** -->> **B**) se, em qualquer ponto do tempo, um fato sobre **A** determina um conjunto de fatos sobre **B**" [Smith 85] Por exemplo, existe uma dependência multivalorada do número de matrícula de um empregado, para as funções exercidas por ele no seu trabalho. Denotada da seguinte forma:

MATR_EMPR -->> FUNÇÃO

É importante frisar que estas definições dadas acima, possuem alguns aspectos diferentes das definições originais, feitas pelos autores citados na seção anterior. Entretanto, elas são derivadas e semelhantes daquelas definições, sendo

suficientes para atingir os nossos objetivos, através deste novo procedimento de projeto de BD.

Os conceitos definidos nesta seção serão fundamentais para o projetista identificar os requisitos de dados para uma aplicação e construir um modelo descritivo.

2.3 - Modelo Descritivo

A estrutura, bem como as consultas possíveis de serem realizadas em um BD, serão um retrato fiel da realidade se o projetista, durante a fase de projeto, entender e modelar todos os requisitos de dados para uma aplicação. Isto significa que o projetista precisa conhecer todos os campos de dados (atributos) necessários pela aplicação e identificar as associações entre os mesmos, utilizando o conceito de dependências funcionais e multivaloradas.

Esta informação inerente à aplicação é documentada na forma de um modelo descritivo. Para efeito de organização, cada regra de integridade documentada pode ser associada a um número. A seguir são dadas algumas normas para construir um modelo descritivo:

- Atribua um nome a cada campo de dado necessário à aplicação. Os nomes devem ser significativos e respeitar as regras de sintaxe do software que definirá e manipulará o BD.

- Descreva, entre parênteses, o nome do campo de dado escolhido, se for o caso deste não ser facilmente entendido (por exemplo, FUNÇÃO significa as funções exercidas por um funcionário em uma empresa, podendo ser: operador, digitador, programador, analista, administrador de BD). Esta norma pode ser bastante útil na construção de um dicionário de dados.

- Defina dependências funcionais e multivaloradas entre os campos de dados escolhidos. Por exemplo, uma MATR_EMPR (matrícula de empregado) está associada a um NOME (nome do empregado), a um SALÁRIO (salário do empregado) e têm mais de uma FUNÇÃO (as funções exercidas por um funcionário em uma empresa, podendo ser: operador, digitador, programador, analista, administrador de BD).

A figura 2.1 mostra um exemplo do modelo descritivo, fruto da análise de requisitos de dados, para um BD de controle acadêmico de uma universidade.

1) Cada ESTUDANTE tem um NOME e um CURSO.

2) Cada PROFESSOR tem um NOME e um SALÁRIO.

3) Cada DISCIPLINA tem um conjunto de TEXTOS.

4) Cada TURMA de uma DISCIPLINA tem vários DIA_AULA e, em um dado DIA_AULA (horário de uma aula), ela está alocada em uma SALA.

5) Uma DISCIPLINA é dividida em TURMAS, cada qual com um PROFESSOR e vários ESTUDANTES.

Fig. 2.1: Modelo descritivo de um BD de controle acadêmico

Por questão de organização apresentamos o modelo descritivo antes dos diagramas de dependências. Entretanto, na prática eles são preparados paralelamente. Sucessivas revisões poderão ser necessárias, à medida que dúvidas sobre os requisitos de dados apareçam e sejam esclarecidas.

Depois de aprender a construir um modelo descritivo passamos a explicar os diagramas de dependências.

2.4 - Diagrama de Dependências

Um diagrama de dependências é uma representação não ambígua das dependências entre campos de dados documentados em um modelo descritivo. Trata-se de um esquema do modelo de dados orientado ao processo de normalização mostrado nesta seção.

Os tipos primitivos do diagrama são os seguintes: bolhas, setas direcionadas simples e duplas, bolhas duplas, bolhas isoladas e indicadores de domínio comum.

Uma bolha é usada para representar um conjunto não vazio de atributos (contendo no mínimo um atributo), que devem ser atômicos, ou seja, que não se decompõe. Esta restrição atende as precauções de Codd em relação à 1ª Forma Normal. Por exemplo, um atributo denominado ENDEREÇO (significando rua e número), com a seguinte instância:

"rua K - 32", "rua Y - 30" e "rua Z - 10"

deve ser representado como RUA e NÚMERO separadamente, ao invés de ENDEREÇO isoladamente, pois este último atributo não é atômico. No exemplo da figura 2.2, temos uma representação do conjunto de atributos {NOME_PEÇA, TIPO_PEÇA}. Dentro de uma bolha, a ordem dos atributos é irrelevante, a menos que a bolha seja do tipo determinante (ver este conceito mais adiante), quando então a ordem indicada será tomada como tal pelo NORMALIZADOR.

Fig. 2.2: Representação de um conjunto de atributos

Uma seta direcionada simples serve para representar uma dependência funcional entre dois conjuntos de atributos X e Y. Na figura 2.3, temos a representação de uma dependência funcional (X --> Y). A bolha de onde parte a seta é chamada de **determinante**.

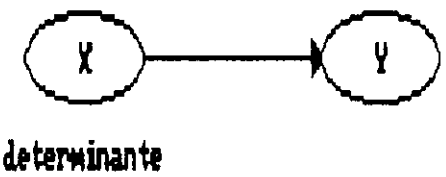


Fig. 2.3: Representação de uma dependência funcional

Uma bolha determinante será transformada em chave primária na fase de geração de relações (ver capítulo IV, Gerando Esquemas Relacionais Normalizados, seção 4.1). Em vista disto, é necessário que um valor do conjunto de atributos não seja nulo e que determine sem ambiguidade a entidade envolvida (isto evita que um atributo como "NOME" seja usado em uma bolha determinante, porque diferentes pessoas podem ter o mesmo nome). E ainda, nenhum atributo do conjunto de atributos determinante pode ter valor nulo.

Se uma bolha determinante é ligada a várias bolhas, o diagrama pode ser simplificado através da combinação dessas bolhas em uma única bolha. A figura 2.4 mostra esta situação. As duas representações são equivalentes.

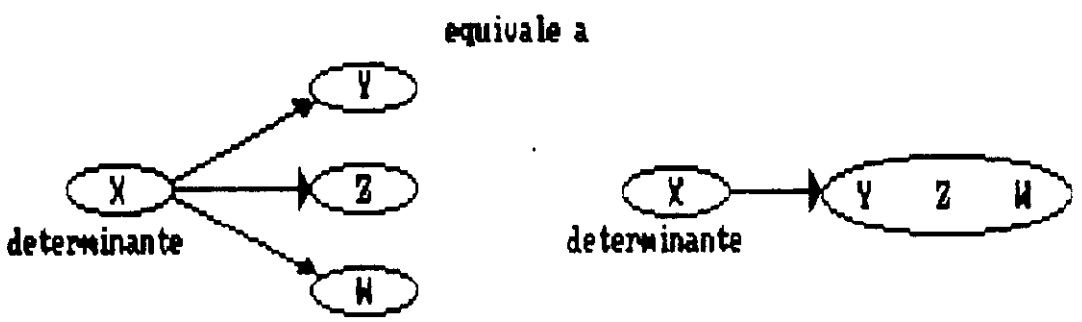


Fig. 2.4: Simplificação do diagrama

Para representarmos uma dependência multivalorada (X -->> Y) entre os conjuntos de atributos X e Y, utilizamos uma seta dupla de X para Y (ver figura 2.5). Neste caso, nem X e nem Y podem ter valor nulo (tampouco pode ter valor nulo qualquer atributo de X e Y) e a concatenação de X e Y (X + Y) deve ser única (ilustrando, dada DISCIPLINA -->> HORÁRIO, deve haver somente uma vez a instância "horario2" de HORÁRIO para a instância "disciplina1" de DISCIPLINA, por exemplo).

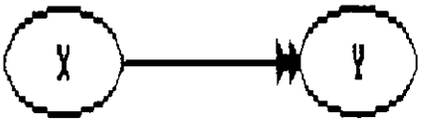


Fig. 2.5: Representação de uma dependência multivalorada

As dependências funcionais e multivaloradas podem ser combinadas formando cadeias de dependências (cadeia de dependências funcionais, multivaloradas ou mistas). Estas são utilizadas, quando existe uma interdependência entre as dependências que a formam, ou seja, quando uma dependência depende da outra.

Várias dependências funcionais podem formar uma cadeia (X --> Y --> ...). Nesta cadeia, pode ser inferido que ao menos Y (além de X, obviamente) é também um determinante. Deste modo, Y aparecerá em duas relações, como chave primária e como chave estrangeira, respectivamente (ver capítulo IV, Gerando Esquemas Relacionais Normalizados, seção 4.1).

Várias dependências multivaloradas podem formar uma cadeia (X -->> Y -->> ...). Aqui também a concatenação X + Y + ... deve ser única, pois esta será a chave primária da relação gerada correspondente a cadeia (ver capítulo IV).

Uma cadeia de dependências multivaloradas pode se ligar a uma cadeia de dependências funcionais, nesta ordem X -->> Y -->> Z --> A --> B --> ... (primeiro as dependências multivaloradas; depois as dependências funcionais, esta ordem é uma restrição deste modelo de dados). Chamêmo-la de cadeia mista. Quando em uma aplicação precisa-se modelar uma cadeia de dependências da forma (A --> B -->> C), que foge à regra de cadeias mistas, o usuário deve modelá-la da seguinte maneira (A --> B e A -->> C). Em futuras versões de **NORMALIZADOR**, está prevista a implementação automática desta modelagem alternativa. Finalmente uma bolha pode pertencer a mais de uma cadeia.

Bolhas múltiplas (duplas, triplas, etc.) são usadas para efeito de simplificação do diagrama. Evitam que uma bolha contendo um conjunto de atributos, necessite de ser repetida mais de uma vez no diagrama, quando este pertencer a mais de uma cadeia de dependências. Desta forma, as bolhas múltiplas servem para estabelecer os limites das cadeias de dependências representadas em um diagrama. Na figura 2.6 podemos observar uma bolha dupla contendo o atributo "NOME" que pertence a duas cadeias de dependências (#PROFESSOR --> NOME --> CPF e #ESTUDANTE --> NOME). Aqui, o uso da bolha dupla, contendo o atributo "NOME", estabeleceu o limite entre estas cadeias. A segunda cadeia começa na bolha contendo o atributo "#ESTUDANTE", terminando na bolha menor que contem o atributo "NOME". Enquanto que a primeira cadeia começa na bolha contendo o atributo "#PROFESSOR", passa pela bolha maior contendo o atributo "NOME" e termina na bolha contendo o atributo "CPF". Isto significa que o atributo "CPF" é apenas do professor (#PROFESSOR), não sendo também do estudante (#ESTUDANTE).

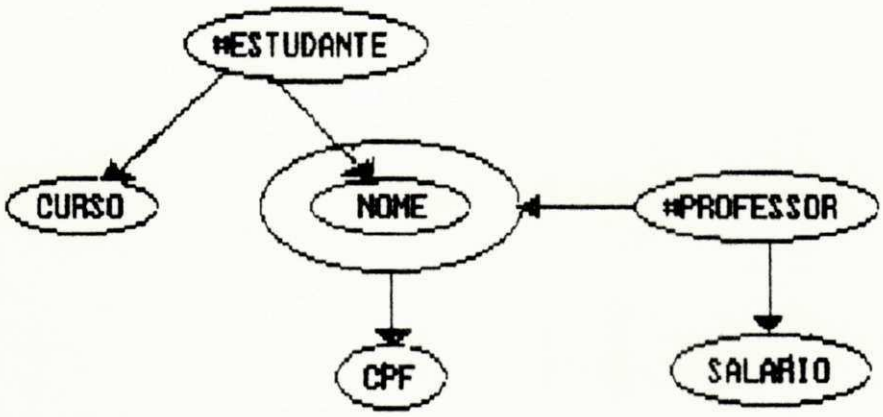


Fig. 2.6: Utilização de bolhas duplas

Ainda em relação a bolhas duplas, uma bolha pode circunscrever somente parte de outra (abrangendo geometricamente os atributos necessários, de um conjunto de atributos, pertencentes a bolha circunscrita).

Bolhas isoladas são utilizadas no diagrama para representar uma dependência multivalorada isolada como X -->> A, onde A não é determinado por nenhum outro conjunto de atributos e nem determina nenhum outro (ver figura 2.7). As duas representações são equivalentes pois geram uma mesma relação (ver capítulo IV).

Fatos representados como uma dependência multivalorada isolada

(a)

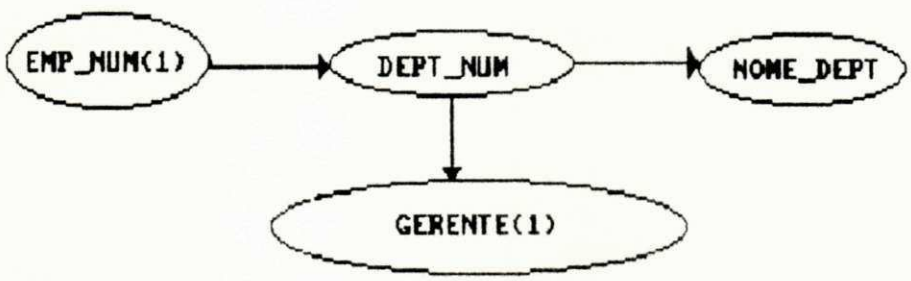
Fatos representados como uma bolha isolada.

(b)

Fig. 2.7: Utilização de bolhas isoladas

Um indicador de domínio comum "(n)", onde "n" é um inteiro, é utilizado quando vários atributos diferentes têm o mesmo domínio. Um exemplo de sua utilização é mostrado na figura 2.8, onde o nome do domínio comum é especificado no

canto inferior do diagrama, junto de seu indicador. Este último é colocado dentro das bolhas, ao lado dos atributos pertinentes. Se houver mais de um domínio comum, os indicadores serão discriminados por seus respectivos números.



(1)NÚMERO_EMPREGADO

Fig. 2.8: Utilização de indicadores de domínio comum

Acabamos de mostrar os conceitos básicos de um diagrama de dependências. A próxima seção mostra um exemplo prático de modelagem, construindo um diagrama que abrange os conceitos discutidos nesta seção.

2.5 - Construindo um Diagrama de Dependências

Nesta seção construiremos um diagrama de dependências, modelando as regras de integridade de um banco de dados para o controle acadêmico de uma universidade. Estas regras estão documentadas no modelo descritivo da figura 2.1.

A primeira regra de integridade afirma que um estudante possui **um único** nome e **um único** curso:

1) Cada **ESTUDANTE** tem um **NOME** e um **CURSO**.

Percebemos claramente a presença de três atributos (escritos em letras maiúsculas), são eles: ESTUDANTE, NOME e CURSO. Utilizando o conceitos de bolha e dependência funcional, podemos representar esta regra de acordo com a figura 2.9.

Fig. 2.9: Diagrama de dependências parcial

A regra de integridade número 2 representa uma dependência funcional entre os conjuntos de atributos {PROFESSOR} e {NOME, SALÁRIO}:

2) Cada **PROFESSOR** tem um **NOME** e um **SALÁRIO**.

Modelamos esta regra (ver figura 2.10) de forma semelhante ao diagrama anterior. Porém, este diagrama é mais simples, pois utilizamos a simplificação de bolhas alvo de uma bolha determinante. Se não tivéssemos feito isto, teríamos que representar no diagrama duas dependências (PROFESSOR --> NOME e PROFESSOR --> SALÁRIO) ao invés de apenas uma (PROFESSOR --> NOME, SALÁRIO). É importante lembrar que estas duas representações são equivalentes (ver figura 2.4).

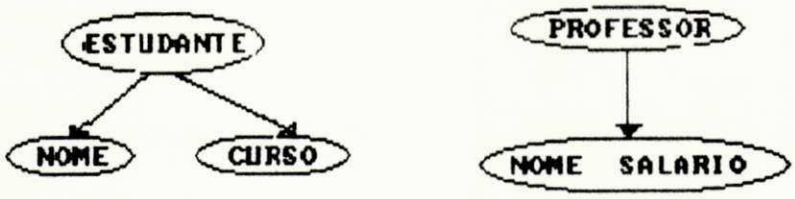


Fig. 2.10: Diagrama de dependências parcial (regras de integridade 1 e 2)

Para modelarmos a regra de integridade número 3, utilizamos o conceito de dependência multivalorada entre os conjuntos de atributos {DISCIPLINA} e {TEXTO}:

3) Cada **DISCIPLINA** tem um conjunto de **TEXTOS**.

Esta regra afirma que cada disciplina possui **um conjunto** de textos. Observe que, ao documentarmos esta regra de

integridade no modelo descritivo, escrevemos no plural o campo de dados "TEXTOS". Entretanto, ao modelarmos esta regra no diagrama, utilizamos o atributo no singular (TEXTO) (ver figura 2.11). É recomendável proceder desta maneira, pois a dependência multivalorada DISCIPLINA -->> TEXTO, implicitamente impõe a semântica de plural ao atributo destino ("TEXTO").

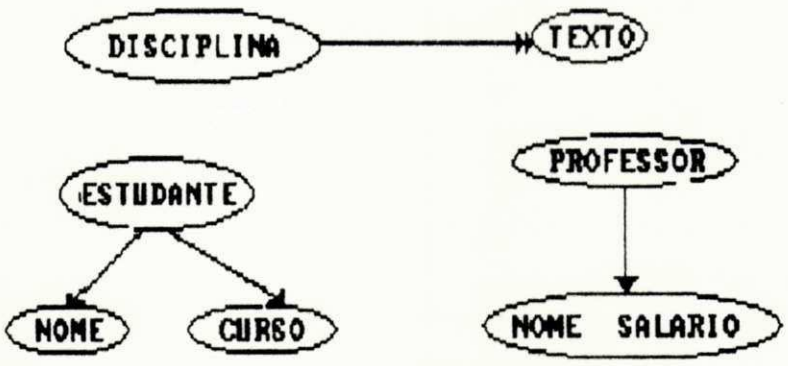


Fig. 2.11: Diagrama de dependências parcial (regras de integridade 1, 2 e 3)

Para modelarmos a regra de integridade número 4 precisamos do conceito de cadeias de dependências, especificamente as cadeias mistas:

4) Cada **TURMA** de uma **DISCIPLINA** tem vários **DIA_AULA** e, em um dado **DIA_AULA** (horário de uma aula), ela está alocada em uma **SALA**.

Inicialmente, identificamos as seguintes dependências:

DISCIPLINA -->> TURMA

TURMA -->> DIA_AULA

DIA_AULA --> SALA

Estas dependências são interdependentes, isto fica claro na maneira de se ler esta regra de integridade, pois na mesma uma dependência faz referência à outra. Em outras palavras,

uma depende da outra para representar um fato do mundo real. A figura 2.12 acrescenta ao diagrama a cadeia mista que modela esta regra de integridade. Observe que esta cadeia segue as regras de formação de cadeias mistas (primeiro as dependências multivaloradas e depois as funcionais).

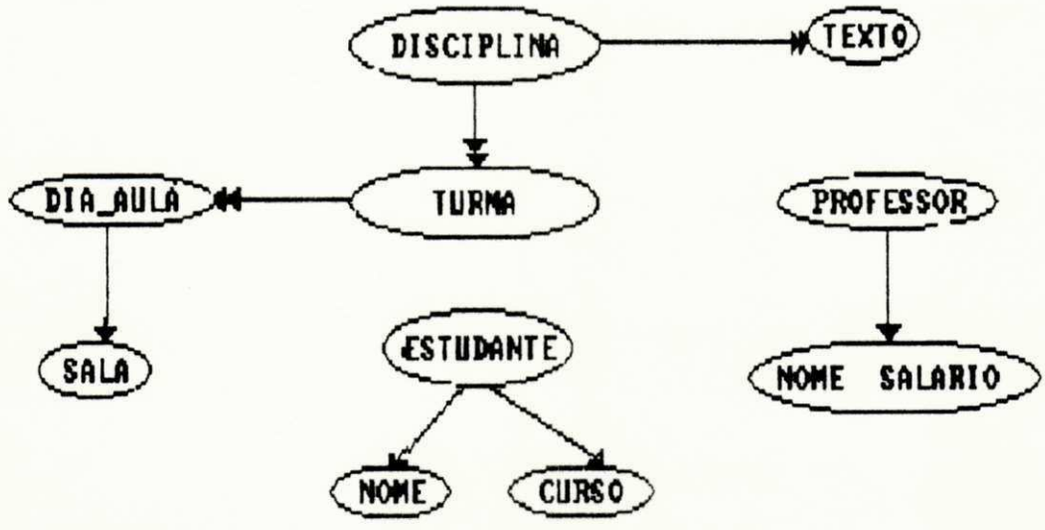


Fig. 2.12: Diagrama de dependências parcial (regras de integridade 1, 2, 3 e 4)

Utilizamos os conceitos de cadeia mista e cadeia multivalorada para modelar a regra de integridade número 5:

5) Uma **DISCIPLINA** é dividida em **TURMAS**, cada qual com um **PROFESSOR** e vários **ESTUDANTES**.

Porém, precisamos também do conceito de bolhas duplas, para estabelecer limites entre esta regra e as anteriormente representadas. Os seguintes limites foram estabelecidos: entre a cadeia mista (DISCIPLINA -->> TURMA --> PROFESSOR) e a dependência funcional (PROFESSOR --> NOME, SALÁRIO); entre a cadeia de dependências multivaloradas (DISCIPLINA -->> TURMA -->> ESTUDANTE) e as dependências funcionais (ESTUDANTE --> NOME e ESTUDANTE --> CURSO). Estes limites foram estabelecidos por não existir interdependência

entre as cadeias (mista e multivalorada) e as dependências funcionais. A figura 2.13 inclui a regra de integridade número 5 completando o diagrama de dependências, correspondente ao modelo descritivo (mostrado na figura 2.1) de um banco de dados para o controle acadêmico de uma universidade.

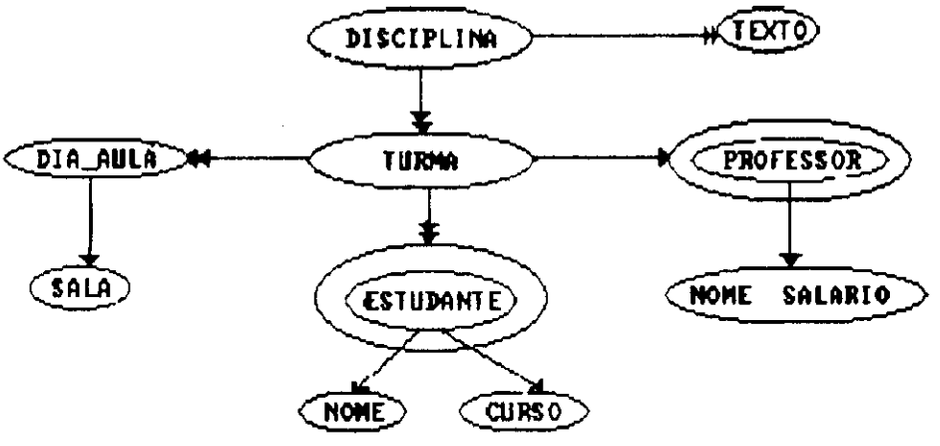


Fig. 2.13: Diagrama de dependências de um banco de dados de controle acadêmico

Acabamos de apresentar o modelo de dados orientado ao processo de normalização, através de seus diagramas de dependências. A ferramenta **NORMALIZADOR** precisa oferecer condições ao projetista de representar suas aplicações segundo este modelo. Assim sendo, um dos módulos de nossa ferramenta consiste de um editor gráfico, que possibilita o desenho de um diagrama de dependências para uma aplicação qualquer. Exemplos práticos da utilização deste editor serão apresentados no capítulo V (Arquitetura do Sistema, seção 5.2.5 - Simulando uma Utilização de NORMALIZADOR).

No próximo capítulo mostraremos a estratégia para detectar possíveis redundâncias em um diagrama de dependências.

3 - Detectando Redundâncias

Neste capítulo serão apresentados os possíveis problemas de modelagem via diagramas de dependências. Em seguida apresentamos nossa solução para os mesmos, através do cálculo de uma cobertura mínima para um diagrama de dependências. Mostraremos dois métodos para este cálculo: o primeiro baseado nos Algoritmos tradicionais de Bernstein; o segundo um aperfeiçoamento do primeiro método, com uma melhora de desempenho bastante significativa. Finalizaremos com uma comparação de desempenho, entre estes dois métodos utilizados por **NORMALIZADOR**.

3.1 - Redundâncias em um Diagrama de Dependências

Durante a fase de projeto lógico, o projetista precisa identificar completamente os requisitos de dados para uma aplicação. Porém, em algumas situações ele exagera, modelando situações redundantes. Estas redundâncias fogem às regras de definição das formas normais, além de impossibilitar a geração de esquemas relacionais mínimos (em relação ao número de relações geradas neste esquema e também à quantidade de atributos para cada relação).

Na figura 3.1 mostramos um exemplo de uma modelagem redundante. O diagrama mostra que: associada a cada número de fornecedor existe uma cidade (#FORNECEDOR --> CIDADE); existe um status associado a cada cidade (CIDADE --> STATUS); para cada número de fornecedor existe um status associado (#FORNECEDOR --> STATUS). Esta última dependência funcional é denominada por Codd, de dependência

transitiva (desreipeitando as restrições para a terceira forma normal [Codd 72]. Esta dependência é redundante e deve ser desprezada, pois ela é uma conseqüência lógica das outras duas. Conseqüentemente ela dá origem à uma relação desnecessária no esquema relacional resultante (ver capítulo IV, seção 4.2, figura 4.5).

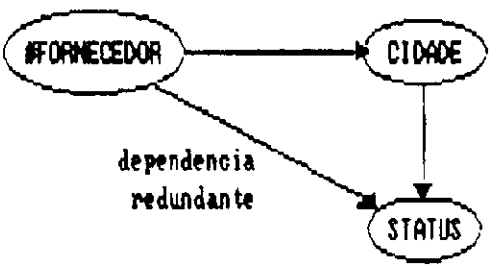


Fig. 3.1: Modelagem redundante

O exemplo da figura 3.1 é apenas um dentre muitos, que podem surgir ao construir um diagrama de dependências. Smith sugere eliminar todas as dependências transitivas, que por ventura surjam, ao desenhar um diagrama de dependências [Smith 85]. Para resolver este problema de redundâncias, juntamos os conceitos de diagrama de dependências aos de cobertura mínima. Oferecendo ao usuário um processo interativo para identificar e corrigir as possíveis redundâncias (ver capítulo V, seção 5.2.5.3.1).

3.2 - Cobertura Mínima

As redundâncias (de dependências funcionais e de atributos) de um diagrama de dependências são eliminadas depois que **NORMALIZADOR** acha automaticamente uma cobertura mínima para as dependências funcionais (DF's), de maneira transparente para o usuário.

Por definição um conjunto de DF's **CM** é dito ser uma **cobertura mínima** para uma lista de DF's F, se qualquer dependência funcional de F puder ser derivada de **CM** e se cada DF pertencente a **CM** possuir as seguintes propriedades [Ullman 89]:

- existência de apenas um atributo no lado direito;

- não existência de atributos desnecessários no lado esquerdo;

- **CM** não contém DF's redundantes, ou seja, nenhuma DF pertencente a **CM** pode ser derivada de outra.

Portanto, **NORMALIZADOR** têm que enxugar um diagrama de dependências, reconstruindo-o a partir do cálculo de **CM**.

Para efeito do cálculo de cobertura mínima, e só para isto, as cadeias de dependências multivaloradas são transformadas em pseudo-atributos, um para cada cadeia pois uma cobertura mínima é calculada para uma lista de DF's. A figura 3.2 mostra o efeito desta transformação: Na cadeia funcional (A --> B --> C) não tivemos nenhuma transformação, pois ela é composta apenas de DF's; porém, transformamos a cadeia multivalorada (A -->> B -->> C) no pseudo conjunto de atributos {A, B, C}, pois esta cadeia é composta de dependências multivaloradas e não fazem parte do escopo do cálculo de cobertura mínima; a principal transformação ocorreu na cadeia mista (A -->> B -->> C --> D), que é composta de dependências multivaloradas e funcionais. Estas últimas, por serem funcionais, precisam ser testadas no cálculo da cobertura mínima. A cadeia mista (A -->> B -->> C --> D) é equivalente a DF (A, B, C --> D), pois elas geram a mesma relação (ver capítulo IV, seção 4.1, figura 4.3).

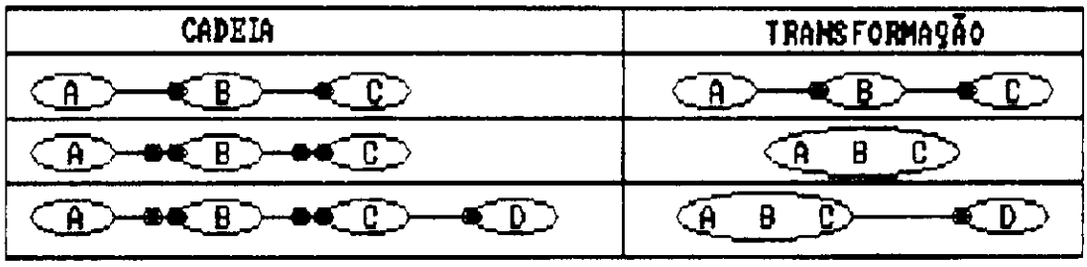


Fig. 3.2: Transformação das cadeias de dependências

Uma outra transformação em um diagrama de dependências é necessária, pois a definição de cobertura mínima exige que cada DF pertencente a uma lista de DF's, contenha apenas um atributo no lado direito. Assim sendo, reescrevemos uma lista de DF's utilizando o axioma de inferência da projeção, definido por Armstrong [Ullman 89], que afirma:

Se X --> Y, Z então X --> Y e X --> Z

Utilizando este axioma, as DF's que contiverem mais de um atributo no lado direito, serão **reduzidas a direita**. Por exemplo, a DF (NOME --> RUA, CIDADE, ESTADO), contendo três atributos no lado direito, é substituída pelas três DF's a seguir:

NOME --> RUA

NOME --> CIDADE

NOME --> ESTADO

Todas estas transformações, explicadas acima, são fundamentais para transformar um diagrama de dependências em uma lista de DF's (ponto inicial para realizar o cálculo de cobertura mínima). É importante frisar, que estas transformações representam a união dos conceitos de diagrama de dependências aos de cobertura mínima. Nas próximas seções apresentaremos os métodos computacionais para encontrar uma cobertura mínima.

3.3 - Método Tradicional de Cobertura Mínima

O algoritmo de Bernstein consiste de um método bastante simples para encontrar uma cobertura mínima. Este possui uma complexidade polinomial $O(n^2)$, podendo ser utilizado manualmente para uma lista de DF's pequena e programado para uma de tamanho maior [Salzberg 86] [Maier 83] [Ullman 89].

A idéia básica é tornar uma lista de DF's mínima, retirando dela as DF's e os atributos redundantes. Para tal, precisamos simplificar a lista de DF's da seguinte maneira:

- reduzir o lado direito de cada DF (utilizando o axioma de inferência da projeção);

- eliminar as DF's redundantes (aquelas que podem ser derivadas de outras);

- minimização do lado esquerdo das DF's (eliminado os atributos que podem ser derivados de outros).

Para tornar uma lista de DF's **F** mínima, precisamos do conceito de **fechamento**. Dado um conjunto de atributos **X**, o fechamento de **X** em relação a **F**, determina quais atributos são funcionalmente dependentes de **X**, usando as DF's de **F** e os axiomas de Armstrong [Maier 83] [Ullman 89]. O algoritmo mostrado a seguir implementa este conceito.

Inicialmente, o fechamento é igual ao conjunto de atributos **X**, em seguida são necessários um ou mais passos sobre o conjunto de DF's **F**, adicionando o lado direito de uma DF ao fechamento, se for o caso do seu lado esquerdo pertencer ao fechamento, e o lado direito não estiver contido no mesmo. No pior caso, este algoritmo possui a complexidade $O(a*n^2)$ [Diederich-Milton 88], onde **n** é o número de DF's pertencentes a **F** e **a** é o número de atributos.

Algoritmo 3.3.1 - Fechamento de X em relação à F

Entrada: X (um conjunto de atributos)

F (a lista de DF's)

Saída: fechamento (atributos funcionalmente dependentes de X)

Passo 1: faça fechamento = X;

Passo 2: repetir até que nenhum atributo seja adicionado ao fechamento

Passo 2.1: para cada DF (A --> B) em F, cujo lado esquerdo (A) estiver contido em fechamento, mas o lado direito (B) não estiver, adicione o lado direito (B) ao fechamento;

Vamos ilustrar o algoritmo 3.3.1, determinando os atributos funcionalmente dependentes do conjunto de atributos X = {#FORNECEDOR}. Considere a lista de DF's F, correspondente ao diagrama de dependências mostrado na figura 3.1, contendo as seguintes DF's:

> #FORNECEDOR --> CIDADE
>
> CIDADE --> STATUS
>
> #FORNECEDOR --> STATUS

Inicialmente o fechamento de **X** é igual a este conjunto de atributos, assim temos:

fechamento = {#FORNECEDOR}

A primeira DF (#FORNECEDOR --> CIDADE) possui o lado esquerdo contido no fechamento, mas o lado direito não. Portanto, obtemos um novo valor para fechamento inserindo o lado direito (CIDADE) em fechamento, da seguinte maneira:

fechamento = {#FORNECEDOR, CIDADE}

Analisando a segunda DF (CIDADE --> STATUS), analogamente obtemos um novo valor para fechamento:

fechamento = {#FORNECEDOR, CIDADE, STATUS}

A partir deste ponto, não existe nenhuma DF cujo lado esquerdo esteja contido no fechamento e o lado direito não esteja. Assim, o conjunto de atributos funcionalmente dependentes de **X** é o valor de fechamento calculado anteriormente.

3.3.1 - Eliminação das Dependências Funcionais Redundantes

Para conseguirmos eliminar as DF's redundantes, utilizaremos o algoritmo 3.3.2 , mostrado a seguir. Este chama o algoritmo 3.3.1 mostrado anteriormente, da seguinte maneira:

Algoritmo 3.3.2 - Eliminação de DF's redundantes

Entrada: F (uma lista de DF's reduzida a direita)

Saída: A lista F sem DF's redundantes

Passo 1: para cada DF (X --> Y) pertencente a F, faça:

> **Passo 1.1**: construa uma lista de DF's reduzida R, eliminando X --> Y da lista de DF's F;
>
> **Passo 1.2**: encontre o fechamento do lado esquerdo (X) em relação a lista de DF's reduzida R (algoritmo 3.3.1);
>
> **Passo 1.3**: Se o fechamento do lado esquerdo (X) em relação a R, contiver o lado direito (Y) da DF em questão então:
>
> > **Passo 1.3.1**: elimine a DF (X --> Y), pois ela é redundante na lista de DF's F;

Para ilustrar o algoritmo 3.3.2, considere novamente a lista de DF's F, correspondente ao diagrama de dependências da figura 3.1. Mostraremos que a DF (#FORNECEDOR --> STATUS) é redundante nesta lista. Esta contém as seguintes DF's reduzidas à direita:

#FORNECEDOR --> CIDADE

CIDADE --> STATUS

#FORNECEDOR --> STATUS

Inicialmente construímos a lista de DF's reduzida R. Esta possui as DF's da lista F menos a DF (#FORNECEDOR --> STATUS). Assim, a lista de DF's reduzida R contém as seguintes DF's:

#FORNECEDOR --> CIDADE

CIDADE --> STATUS

O fechamento do lado esquerdo (#FORNECEDOR) da DF testada, em relação a lista de DF's reduzidas R, contém os seguintes atributos

{#FORNECEDOR, CIDADE, STATUS}

Como este fechamento contém o lado direito (STATUS) da DF testada, então a DF em questão é redundante na lista de DF's F. Em outras palavras, ela é derivada das DF's pertencentes a lista de DF's R. Portanto, esta DF deve ser eliminada da lista de DF's F.

3.3.2 - Minimização do lado esquerdo das DF's

Após a eliminação das DF's redundantes, passamos a última tentativa para redução da lista de DF's. Vamos eliminar os possíveis atributos redundantes do lado esquerdo de cada DF. Para tanto, utilizamos o algoritmo 3.3.3 mostrado a seguir, este também utiliza o algoritmo 3.3.1 mostrado anteriormente. Após a execução do algoritmo 3.3.3 temos uma cobertura mínima para uma lista de DF's.

Algoritmo 3.3.3 - Eliminação dos atributos redundantes

Entrada: F (uma lista sem DF's redundantes)

Saída: uma cobertura mínima para F

Passo 1: para cada DF (X --> Y) pertencente a F, faça:

Passo 1.1: para cada atributo A pertencente ao lado esquerdo (X), faça:

Passo 1.1.1: elimine o atributo A do lado esquerdo (X) da DF em questão, obtendo uma redução R de X;

Passo 1.1.2: encontre o fechamento de R em relação a lista de DF's F (algoritmo 3.3.1);

Passo 1.1.3: Se o fechamento de R em relação a F contiver o lado direito (Y) da DF em questão então, o atributo A é redundante e deve ser eliminado desta DF;

Para ilustrar o algoritmo 3.3.3, considere a seguinte lista de DF's sem DF's redundantes:

```
SOBRENOME, MATR --> NOME
MATR --> SOBRENOME
```

Provaremos que o atributo (SOBRENOME) pertencente ao lado esquerdo da DF (SOBRENOME, MATR --> NOME) é redundante, vejamos:

A redução R do lado esquerdo da DF em questão possui o seguinte atributo:

R = {MATR}

O fechamento de R em relação a F, possui os seguintes atributos:

fechamento = { MATR, SOBRENOME, NOME}

Como este fechamento contém o lado direito (NOME) da DF em questão, então o atributo (SOBRENOME) é redundante nesta DF e deve ser eliminado.

Finalmente obtemos a cobertura mínima para a lista de DF's F, com as seguintes dependências:

```
MATR --> NOME
MATR --> SOBRENOME
```

Esta seção apresentou o método tradicional de cálculo de cobertura mínima, baseado nos algoritmos clássicos de Bernstein. Este é um método de fraco desempenho, em virtude do número de fechamentos calculados desnecessariamente. Na próxima seção apresentaremos um outro método bem mais eficiente de cobertura mínima.

3.4 - Método Eficiente de Cobertura Mínima

Para possibilitar um bom desempenho no cálculo de cobertura mínima, apresentamos novos métodos de cobertura mínima, baseados no trabalho de Diederich e Milton [Diederich-Milton 88]. Estes métodos são um aperfeiçoamento dos algoritmos tradicionais de Bernstein, possibilitando um aumento de desempenho bastante significativo.

3.4.1 - Definições Básicas e Convenções

Nesta seção apresentaremos as definições básicas e convenções, usadas nesta e nas próximas sub-seções, oferecendo o suporte teórico básico, para apresentação dos algoritmos que implementam este método de cobertura mínima.

As letras **X**, **Y** e **Z** são utilizadas para representar conjuntos de atributos, enquanto que **A**, **B** e **C** para representar atributos individuais (não há distinção entre um atributo **A** e um conjunto de atributos unitário {**A**}. A união dos conjuntos **X** e **Y** é denotada por **XY**. Uma DF entre **X** e **Y** é denotada por **X --> Y**. Freqüentemente usamos o termo **l-d** significando o lado direito de uma DF e **l-e**, o lado esquerdo. As letras **F**, **G** e **H** denotam listas de DF's. Letras

minúsculas são usadas para instanciar conjuntos de atributos. Uma DF (X --> A) é dita trivial se **A** pertence a **X**. Uma lista de DF's é dita simplificada se não contiver DF's duplicadas nem triviais.

O fechamento de um conjunto de atributos **X** em relação a uma lista de DF's **F** (definido na seção 3.3) é representado por X^+_F (Freqüentemente, o subscrito **F** é omitido quando **F** não possui DF's redundantes).

Dado uma lista de DF's **F**, um atributo **B** é dito redundante em uma DF (**X --> A**) pertencente a F, se:

X = ZB, X <> Z (X diferente de Z) e A pertencer a Z^+_F

Os atributos redundantes são classificados em atributos redundantes implícitos e explícitos. B é um atributo redundante implícito se:

X = ZB, X <> Z e B pertencer a Z^+_F

Se B é um atributo redundante, mas não implícito, então B é dito um atributo redundante explícito.

Para ilustrar o conceito de atributo redundante explícito mostramos a seguir um exemplo. Provaremos que **"n"** é um atributo redundante explícito, na DF (bn --> h) pertencente a lista de DF's **F**, mostrada a seguir:

```
F = { b --> a, b --> d, b --> e, b --> f, b --> g,
      b --> h, bc --> d, bc --> j, bc --> k, d --> a
      bn --> h }
```

Considere X --> A uma DF pertencente a **F**. Instanciando os valores X = bn; A = h, temos:

X = ZB => bn = ZB => Z = b, B = n ("=>" é o operador de decorrência) (cláusula verdadeira)

X = ZB => X = bn

X <> Z => bn <> b

$Z^+_F = b^+_F$ = { b, a, d, e, f, g, **h** }

A pertence a Z^+_F => h pertence a b^+_F

Portanto, o atributo **n** é redundante na DF bn --> h pertencente a F, pois:

X = ZB, X <> Z , A pertencer a Z^+_F (é verdadeiro)

Para classificar o atributo redundante **n**, vamos realizar o seguinte teste:

B pertence a Z^+_F => **n** pertence a b^+_F (cláusula falsa)

Como esta última cláusula é falsa, concluímos que **n** é um atributo redundante explícito, na DF (bn --> h) pertencente a F, como queríamos demonstrar.

Dada uma lista de DF's **F**, um conjunto de DF's denotado por **dep-para-L-E(X)**, representa o subconjunto não vazio de DF's pertencentes a **F**, cujo **l-e** é igual ao conjunto de atributos **X**. Chamamos o conjunto de todos os **l-e** das DF's de **F** de **L-E(F)**.

Quando um atributo **A** aparece apenas no **l-d** das DF's pertencentes a uma lista de DF's **F**, chamamos **A** de **atributo-d-a** (d-a significa direito apenas). Os atributos que não são classificados como **atributo-d-a** e, aparecem no l-e de no mínimo uma DF pertencente a lista de DF's **F**, podendo também aparecer no l-d de outras DF's nesta lista, são chamados de **atributos-e-d** (e-d significa esquerda e direita). Para sermos mais completos, definimos os conjuntos abaixo em função dos conceitos definidos anteriormente:

$D\text{-}A_X$ = {**A** / X --> A pertence a **F** e **A** é um **atributo-d-a**}

$E\text{-}D_X$ = {**A** / X --> A pertence a **F** e **A** é um **atributo-e-d**}

R_X = D-A_X U E-D_X ("U" representa o operador de união de conjuntos).

O subscrito **X** desaparece quando nenhuma ambiguidade surge ao se formar estes conjuntos.

3.4.2 - Fechamento-r

Observando o algoritmo 3.3.2, notamos que muitos fechamentos são recalculados a cada passo, para dependências com o mesmo **l-e X**. Com o objetivo de reduzir estas computações introduzimos o conceito de fechamento-r.

Um **fechamento-r** de X sobre uma lista de DF's **G**, denotado por $X^{\wedge}_G$, é o conjunto de todos os atributos **A** tal que Y --> **A** pertence a **G** e Y pertence a X^{+}_G. $X^{\wedge}_G$ é o conjunto de todos os **l-d** derivados de **X** em relação a **G**. Freqüentemente **G** será um subconjunto próprio de um conjunto de DF's **F** [Diederich-Milton 88].

Utilizando a notação de conjuntos temos:

$X^{\wedge}_G$ = {A / Y --> A pertence a G & Y pertence a X^{+}_G}

O algoritmo 3.4.1 mostrado a seguir, implementa o conceito de fechamento-r sobre uma lista de DF's.

Algoritmo 3.4.1 - Fechamento-r de X

Entrada: X (um conjunto de atributos)

G (uma lista de DF's)

(X^{+}_G)o fechamento padrão de X em relação a G

Saída: $X^{\wedge}_G$ (o fechamento-r de X em relação a G)

Passo 1: faça $X^{\wedge}_G$ = { } ;

Passo 2: para cada DF (Y --> A) pertencente a G, faça:

Passo 2.1: se (Y está contido em X^{+}_G) & (A não pertence a $X^{\wedge}_G$) então, faça: $X^{\wedge}_G$ = $X^{\wedge}_G$ + A

Vamos ilustrar o algoritmo 3.4.1 através de um exemplo. Inicialmente, considere a lista de DF's **G**, contendo as seguintes DF's:

```
#emp --> nome_emp, #emp --> endereço, #emp -->título,
#emp --> salário, #emp --> #dept, #emp --> gerente,
#dept --> nome_dept, #dept --> gerente,
#dept --> prédio, #dept --> #escritório,
nome_dept --> #dept
```

Queremos descobrir o conjunto de todos os **1-d** derivados do conjunto de atributos {#emp}, a partir da lista **G.**

Calculando $\{emp\#\}^{+}_{G}$ temos o seguinte conjunto de atributos:

$\{emp\#\}^{+}_{G}$ = { #emp, nome_emp, endereço, título, salário, #dept, gerente, nome_dept, prédio, escritório }

Já o cálculo de $\{emp\#\}^{\wedge}_{G}$ possui os seguintes atributos:

$\{emp\#\}^{\wedge}_{G}$ = { nome_emp, endereço, título, salário, #dept, gerente, nome_dept, prédio, escritório }

Note que #emp pertence a $\{emp\#\}^{+}_{G}$, mas não pertence a $\{emp\#\}^{\wedge}_{G}$, pois não existe nenhuma DF (Y --> #emp) pertencente a **G**, tal que Y pertença a $\{emp\#\}^{+}_{G}$.

É importante ressaltar que, para qualquer conjunto de atributos **X**, $X^{\wedge}_{G}$ está contido em X^{+}_{G}, e os únicos elementos de X^{+}_{G} retirados de $X^{\wedge}_{G}$ são elementos de **X**. Esta característica será útil na eliminação de atributos redundantes implícitos.

O conceito de r-fechamento é usado para detectar dependências redundantes (minimizando o número de fechamentos computados), através das propriedades P1 e P2 [Diederich-Milton 88] definidas a seguir. Considere **F** uma lista de DF's simplificada:

P1)X --> A é redundante em F, se A pertencer a: (X união $E-D_X$)^+_G interseção $D-A_X$).

P2)X --> A não é redundante em F, se A não pertencer a: (X união $E-D_X$)^+_G.

Em resumo, o r-fechamento do lado esquerdo de uma DF e seus atributos e-d, é a heurística que reduz o número de fechamentos computados Entretanto, em alguns casos é necessário usar o fechamento padrão (algoritmo 3.3.1), para checar uma DF como sendo redundante. Além do mais o fechamento-r não enxerga DF's duplicadas e triviais, porém isto não é um problema grave, pois ele é facilmente eliminado sem comprometimento do desempenho do fechamento-r.

3.4.3 - Algoritmos Eficientes de Cobertura Mínima

Os próximos algoritmos fazem parte do método mais refinado, para calcular uma cobertura mínima. Estes algoritmos também exigem que as DF's de uma lista de DF's contenham em seu l-d exatamente um atributo. Novamente o axioma da projeção de Armstrong será utilizado para tal finalidade.

A idéia básica deste método é, inicialmente eliminar todos os atributos redundantes implícitos de cada DF; em seguida, as dependências redundantes serão eliminadas. É importante ressaltar, que os possíveis atributos redundantes explícitos, necessariamente pertencem à DF's redundantes e serão eliminados junto com elas. Implementando esta idéia, nós evitamos muitas das computações executadas nos métodos tradicionais Atributos redundantes implícitos são primeiramente eliminados, permitindo a eliminação de um conjunto de DF's redundantes, ao invés de uma de cada vez como é feito no método tradicional. Além do mais, esta idéia reduz o número de atributos que necessitam ser testados como

redundantes, enquanto que no método tradicional todos os atributos são testados. As propriedades **P3** e **P4** [Diederich-Milton 88] apresentadas abaixo , são utilizadas para implementar esta idéia:

P3)Se (**C** pertence a **X**) & (**C** pertence a (**X** união $\mathbf{E-D_X}$)$\hat{}_G$) então **C** é um atributo redundante implícito em X --> A.

P4)Se (**C** pertence a **X**) & (**C** não pertence a (**X** união $\mathbf{E-D_X}$)$\hat{}_G$) então **C** não é um atributo redundante em X --> A ou X --> A é redundante em **F.**

O algoritmo 3.4.2, mostrado a seguir, elimina os atributos redundantes implícitos de uma lista de DF's F.

Algoritmo 3.4.2 - Eliminação de atributos redundantes implícitos

Entrada: F (uma lista simplificada de DF's)

Saída: H (sem atributos redundantes implícitos)

Passo 1: Para cada Dep-para-L-E(X) faça:

Passo 1.1: Se |X| > 1 então

Passo 1.1.1: fech-acumulativo = (X U $\mathbf{E-D_X}$)$\hat{}_F$

Passo 1.1.2: Enquanto |X| > 1, para cada B pertencente a X, faça:

Passo 1.1.2.1: Se B pertence a interseção do conjunto de atributos **X** e o conjunto de atributos fech-acumulativo, então

Passo 1.1.2.1.1: X' = X - B

Passo 1.1.2.1.2: compute $\mathbf{X'^+_F}$

Passo 1.1.2.1.3: se B pertence a $\mathbf{X'^+_F}$ então faça X = X'

Passo 1.1.3: Se o conteúdo de X foi trocado no passo 1.1.2.1.3, então troque X pelo seu novo valor em todas as DF's pertencentes a Dep-para-L-E(X)

Passo 2: Reconstitua os conjuntos Dep-Para-L-E(X), eliminando as DF's duplicadas.

Para ilustrar o algoritmo 3.4.2, considere a lista simplificada de DF's F mostrada abaixo. Este exemplo foi retirado do trabalho de Diederich e Milton:

```
a b --> d, a b --> e, a b --> f, a b --> g
d --> a
b n --> h
b --> a, b --> h, b --> g
a b c --> d, a b c --> j, a b c --> k
```

Para dep-para-L-E(a b), temos:

No Passo 1.1.1: fech-acumulativo = $\{a, b, d\}^{+}_{F}$ =

{a, h, g}

No Passo 1.1.2: apenas (a) é testado como sendo redundante e é eliminado.

No Passo 1.1.3: (a b) é trocado por (b) em todas as Df's pertencentes a Dep-para-L-E(a b).

Para dep-para-L-E(a b c), temos:

No Passo 1.1.1: fech-acumulativo = $\{a, b, c, d\}^{+}_{F}$ = {d, e, f, g, a, h}

No Passo 1.1.2: apenas (a) é testado como sendo redundante e é eliminado.

No Passo 1.1.3: (a b c) é trocado por (b c) em todas as Df's pertencentes a Dep-para-L-E(a b c).

Para dep-para-L-E(b n), temos:

No Passo 1.1.1: fech-acumulativo = $\{b, n\}^{+}_{F}$ =

{d, e, f, g, a, h}

No Passo 1.1.2: nenhum atributo é testado como sendo redundante (observe que n é um atributo redundante explícito).

No Passo 1.1.3: nenhuma mudança no l-e das DF's.

No Passo 2 novos conjuntos Dep-para-L-E(X) são formados, eliminando DF's duplicadas, obtendo a lista de DF's mostrada a seguir:

```
b --> a, b --> d, b --> e, b --> f, b --> g, b --> h
d --> a
b n --> h
b c --> d, b c --> j, b c --> k
```

Note que (b c --> d) possui um atributo redundante explícito (**c**), este será removido durante a eliminação de DF's redundantes, pois (d) está no fechamento-r de (X união E-D), onde X = {b, c} e E-D = {d}. Analogamente acontece com o atributo (**n**) pertencente a DF (b n --> h).

Finalmente, é importante realçar que são necessários computar 19 fechamentos, usando o algoritmo 3.3.3, enquanto que apenas 5 são suficientes com o algoritmo 3.4.2.

O algoritmo 3.4.3 apresentado a seguir, elimina as DF's redundantes juntamente com os possíveis atributos redundantes explícitos. Assumimos que atributos redundantes implícitos foram eliminados. Este algoritmo completa o cálculo de cobertura mínima.

Algoritmo 3.4.3 - Eliminação de DF's redundantes

Entrada: F (uma lista de DF's parcialmente reduzido à esquerda)

Saída: uma cobertura mínima para F

Passo 1: H = F

Passo 2: Para cada Dep-para-L-E(X) faça:

Passo 2.1: G = F - Dep-para-L-E(X)

Passo 2.2: fech-acumulativo = { }

Passo 2.3: atributo = X

Passo 2.4: Para cada DF X --> A pertencente a Dep-para-L-E(X) faça:

Passo 2.4.1: Se A pertence a fech-acumulativo então

Passo 2.4.1.1: elimine X --> A de H

Passo 2.4.2: Se A não pertence a fech-acumulativo então

Passo 2.4.2.1: Se A é um atributo-e-d então

Passo 2.4.2.1.1: atributo = atributo união A

Passo 2.4.2.1.2: atribua a fech-acumulativo o calculo do fechamemto-r. atributo $\hat{}_G$

Passo 2.5: Se atributo == { } então

/* nenhum fechamento-r */

/* foi computado */

Passo 2.5.1: fech-acumulativo = $X\hat{}_G$

Passo 2.6: Para cada DF X --> A, pertencente a interseção do conjunto Dep-para-L-E(X) e o conjunto H, faça:

Passo 2.6.1: Se A pertence a fech-acumulativo então

Passo 2.6.1.1: Se (A é um atributo-d-a) ou ((A é um atributo- e-d) & (A pertence a X^+ em relação a F - {X --> A})) então

Passo 2.6.1.1.1: elimine (X --> A) de H

Vamos ilustrar o algoritmo 3.4.3, com um outro exemplo retirado do trabalho de Diederich e Milton. Considere a lista simplificada e reduzida à esquerda de DF's F, mostrada abaixo. :

```
a --> b, a --> c, a --> d, a --> e, a --> f, a --> g,
a --> h,
d --> f, d --> b, d --> h
h --> d,
g --> k, g --> l, g --> m, g --> n
```

Passo 2.4

DF	tipo do l-d	teste/resultado	ações	fech-acum(a)
a --> b	d-a	b ∈ fech-acumulativo(a) ? não	--	{ }
a --> c	d-a	c ∈ fech-acumulativo(a) ? não	--	{ }
a --> d	e-d	d ∈ fech-aculumalivo(a) ? não	compute $\{a,d\}^{\wedge}_{G}$	{f,b,h,d}
a --> e	d-a	e ∈ fech-acumulativo(a) ? não	--	{f,b,h,d}
a --> f	d-a	f ∈ fech-acumulativo(a) ? sim	elimine a --> f	{f,b,h,d}
a --> g	e-d	g ∈ fech-acumulativo(a) ? não	compute $\{a,d,g\}^{\wedge}_{G}$	{f,b,h,d, k,l,m,n}
a --> h	e-d	h ∈ fech-acumulativo(a) ? sim	elimine a --> h	{f,b,h,d k,l,m,n}

O valor final de fech-acumulativo foi calculado no final do passo 2.4

Passo 2.6

a --> b	d-a	b ∈ fech-acumulativo(a) ? sim	elimine a --> b
a --> c	d-a	c ∈ fech-acumulativo(a) ? não	--
a --> d	e-d	d ∈ fech-acumulativo(a) ? sim	compute $\{a\}^{+}_{F'}$, onde F' = F - {a --> d} e elimine a --> d se d está neste fechamento
a --> e	d-a	e ∈ fech-acumulativo(a) ? não	--
a --> g	e-d	g ∈ fech-acumulativo(a) ? não	--

Fig. 3.3: Removendo DF's redundantes usando algoritmo 3.4.3

A figura 3.3 mostra as ações tomadas e o valor resultante para fech-acumulativo(a), em função da

manipulação de cada DF. Desde que cada conjunto Dep-para-L-E(X) será tratado isoladamente, inicialmente selecionamos Dep-para-L-E(a). Considere que, fech-acumulativo(a) representa o fechamento-r acumulativo em relação a DF tratada a cada momento. Inicialmente, faça fech-acumulativo(a) = { }, o conjunto vazio. Assuma que G = F - Dep-para-L-E(a).

Note que tanto (b) como (f) são atributos-d-a, mas no passo 2.4 apenas a --> f é classificada como redundante em F, pois o fechamento-r para o atributo-e-d (d) é manipulado depois de (b) e antes de (f). Conseqüentemente, o passo 2.6 é necessário para eliminar a DF redundante a --> b.

Este processo é repetido para Dep-para-L-E(X), onde X assume os seguintes valores: X = d, X = g, X = h. Um fechamento-r será calculado para X = d, nenhum quando X = g, e um quando X = h. No total serão necessários 5 fechamentos, incluindo fechamentos-r e fechamentos padrão. Enquanto que serão calculados 15 fechamentos padrão, se o algoritmo 3.3.1 for utilizado para a mesma lista de DF's F.

3.5 - Comparação de Desempenho

Os dois métodos de cálculo de uma cobertura mínima, apresentados nas seções anteriores, foram implementados em nosso trabalho. Testes realizados em seis conjuntos de DF's, retirados do trabalho de Diederich e Milton, foram realizados para comparação de desempenho desses dois métodos. Estes conjuntos genéricos de DF's são dados abaixo como A1, A2, ..., A6 (cada um possui de 40 a 50 DF's). Cinco destes arquivos possuem características, que esperamos encontrar em BD's típicos encontrados na prática. O conjunto A3 é um conjunto de DF's patológico encontrado na teoria, considerado como o pior caso para o algoritmo 3.3.1 [Maier 83], entretanto, paradoxalmente este conjunto representa o

melhor tempo no cálculo de uma cobertura mínima, utilizando o método tradicional.

A1) Conjunto Dependente (CD)

primeiro conjunto:

```
a1 --> b1, c1, d1, e1, f1, g1, h1
e1 --> j1, k1, m1, n1, p1, r1, s1
p1 --> d1, t1, u1, v1, h1, w1
```

Dependência de transição:

```
w1 --> a2
```

segundo conjunto, igual ao primeiro com cada sufixo 1 trocado por 2:

```
a2 --> b2, c2, d2, e2, f2, g2, h2
e2 --> j2, k2, m2, n2, p2, r2, s2
p2 --> d2, t2, u2, v2, h2, w2
```

Existe quatro DF's redundantes em A1, são elas: a1 --> d1, h1; a2 --> d2, h2

Este conjunto caracteriza grupos de entidades ligadas a um segundo grupo através de um atributo.

A2) Conjuntos Independentes (CI)

Este conjunto é semelhante a A1, porém a dependência de transição é eliminada Isto corresponde a conjuntos de entidades independentes.

A3) Linear Invertido (LI)

```
a1 --> a0,
a2 --> a1,
a3 --> a2,
  . . .
a50 --> a49
```

Este é o caso patológico, onde são necessários 50 passos sobre A3 para calcular o fechamento de a50.

A4) Linear Invertido Completo (LIC)

Este conjunto mantem alguns dos problemas de A3. Ele reflete um grupo de entidades depois da remoção de DF's

redundantes. Isto pode acontecer quando da adição de novas DF's a uma cobertura mínima existente, requerer um novo cálculo de cobertura mínima.

```
a0 --> a1, a2, a3, a4        b0 --> a0, b1, b2, b3, b4
c0 --> b0, c1, c2, c3, c4    d0 --> c0, d1, d2, d3, d4
. . .
h0 --> g0, h1, h2, h3, h4    i0 --> h0, i1, i2, i3, i4, i5
```

A5) Independente Completo (IC)

Este conjunto é semelhante a A4, entretanto não existe ligações entre a's, b's, c's, etc. A5 corresponde a entidades independentes em um BD.

A6) Independente Equivalente (IE)

```
a0 --> a1, a2, a3, a4, a5    a4 --> a1, a2, a3, a0, a5
b0 --> b1, b2, b3, b4, b5    b4 --> b1, b0, b3, b2, b5
c0 --> c1, c2, c3, c4, c5    c4 --> c0, c2, c3, c1, c5
d0 --> d1, d2, d3, d4, d5    d4 --> d1, d0, d3, d2, d5
```

Este conjunto corresponde a chaves equivalentes contendo muitas DF's redundantes.

A figura 3.4, mostra a comparação de desempenho entre os métodos mostrados anteriormente (tradicional e eficiente) através de "Benchmarks". Os testes foram realizados em um microcomputador compatível com o IBM AT. Contendo um processador central (UCP) Intel 80286 (16 MHz de clock), sem a ajuda de um co-processador aritmético. O Sistema Operacional utilizado foi o MS-DOS versão 5.0. É importante realçar, que os tempos dados para cada conjunto de DF's, são apenas de UCP, não foram acrescentados o tempo gasto com entrada e saída.

Métodos	"Benchmarks" (tempo em segundos)					
	CD	CI	LI	LIC	IC	IE
Bernstein	16,0	08,0	06,0	25,0	04,0	04,0
Diederich - Milton	04,0	02,0	62,0	08,0	01,0	02,0

Fig. 3.4: Comparação de Desempenho

Este capítulo apresentou o suporte teórico necessário a implementação do módulo **Detector de redundâncias** de **NORMALIZADOR**. Ele é responsável pela depuração de um diagrama de dependências, encontrando automaticamente as possíveis redundâncias, emitindo explicações interativas e informando ao usuário de que maneira o diagrama deve ser corrigido. No capítulo V (Arquitetura do Sistema) apresentaremos uma simulação deste módulo.

Depois que as redundâncias de um diagrama foram eliminadas, obtemos um esquema relacional resultante mínimo (em relação ao número de relações e ao número de atributos). O próximo capítulo mostra este objetivo.

4 - Gerando Esquemas Relacionais Normalizados

Neste capítulo mostraremos o mapeamento relacional para um diagrama de dependências, obtendo um esquema relacional resultante mínimo. Exibiremos como **NORMALIZADOR** encontra automaticamente as chaves para as relações geradas e identifica as possíveis navegações nesse esquema via junções naturais. Apresentaremos um exemplo prático de mapeamento de um diagrama de dependências. Finalizaremos com uma prova informal de que as relações geradas estão normalizadas.

4.1 - Mapeamento Relacional do Diagrama de Dependências

Esta seção mostra a composição de relações a partir de um diagrama de dependências. As regras que permitem compor um esquema relacional são bastante simples. Estas representam um mapeamento relacional, do modelo de dados orientado ao processo de normalização [Smith 85], apresentado em detalhes no capítulo II Esta seção mostra em detalhes como **NORMALIZADOR** realiza esta tarefa de forma automática.

Uma cadeia de dependências funcionais pode gerar várias relações, uma para cada dependência. Cada relação consiste dos dois conjuntos de atributos (doravante, simplesmente atributos) da dependência funcional correspondente, sendo que o primeiro atributo é chave primária e o segundo atributo é chave estrangeira (à exceção talvez da relação correspondente à última dependência da cadeia, que pode não ter chave estrangeira). Vale salientar, que esta regra

também é válida para uma dependência funcional que não pertença a uma cadeia. Neste caso, seria como se tivéssemos uma cadeia com apenas uma dependência funcional.

A figura 4.1, mostra a composição das relações a partir da cadeia de dependências funcionais (A --> B --> C --> D). Observe que três relações foram geradas, pois esta cadeia contem três dependências funcionais. As chaves primárias de cada relação, são os atributos determinantes de cada dependência funcional (ver restrição de bolha determinante no capítulo II, seção 2.4). Observe que o atributo **B** aparece em duas relações, como chave estrangeira da relação 1 e como chave primária da relação 2, isto porque uma cadeia de dependências funcionais é composta de mais de um determinante. Analogamente, o atributo C aparece nas duas relações 2 e 3, respectivamente como chave estrangeira e chave primária (ver definição de cadeias de dependências funcionais no capítulo II, seção 2.4). Finalmente, a relação 3 não contém chave estrangeira, pois esta corresponde a última dependência funcional da cadeia, que não se associa a nenhuma outra dependência pertencente ao diagrama.

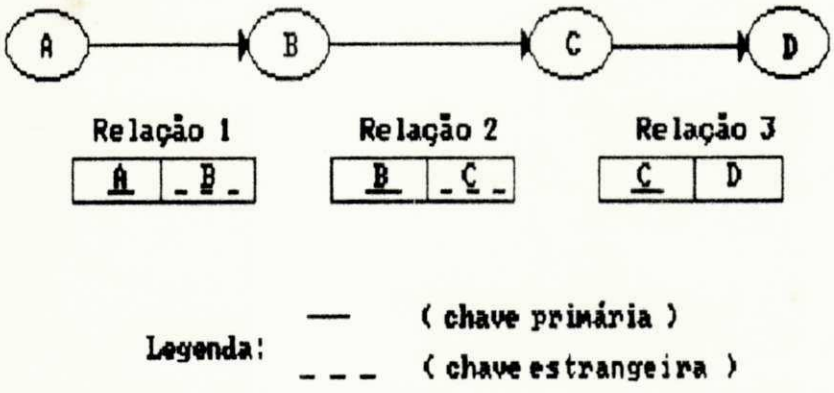
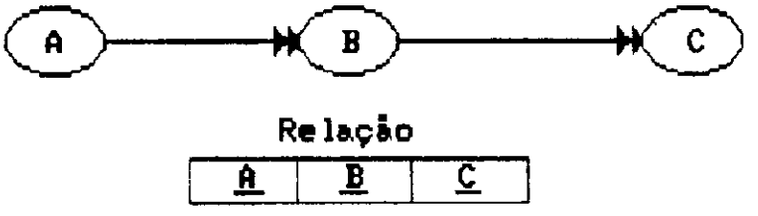


Fig. 4.1: Gerando relações de uma cadeia de dependências funcionais

Uma cadeia de dependências multivalorada é sintetizada como um única relação em que todos os atributos fazem parte

da chave primária. Vale frisar, que esta regra também é válida para uma dependência multivalorada isolada. Neste caso, seria como se tivéssemos uma cadeia com apenas uma dependência multivalorada. A figura 4.2 mostra a relação formada a partir da cadeia de dependências multivaloradas (A -->> B -->> C). Observe que os atributos desta cadeia fazem parte da chave primária desta relação.



Relação

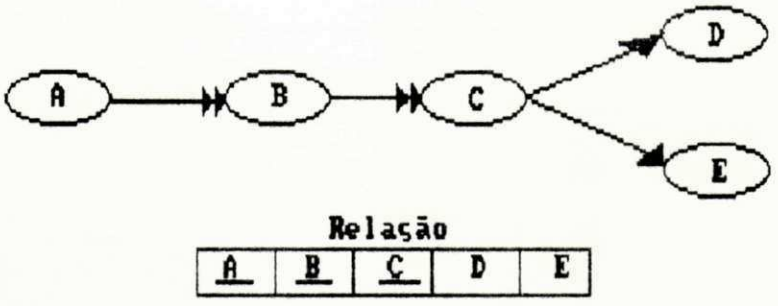
A	B	C

chave primária : A + B + C

Fig. 4.2: Gerando relações de uma cadeia de dependências multivaloradas

Uma cadeia mista é sintetizada como uma relação composta de chave primária (os atributos pertencentes à parte multivalorada desta cadeia) e atributos dependentes funcionalmente da chave (aqueles pertencentes à parte funcional da cadeia mista).

A figura 4.3 mostra uma relação sintetizada das cadeias mistas (A -->> B -->> C --> D) e (A -->> B -->> C --> E), lembrando que as bolhas com os atributos "D" e "E", respectivamente, podem ser fundidas em uma só bolha como "D E". A chave primária desta relação é composta dos atributos pertencentes a cadeia multivalorada comum as duas cadeias mistas. É importante perceber que a dependência funcional (A, B, C --> D, E), geraria a mesma relação gerada por estas cadeias mistas.



Relação

A	B	C	D	E

Chave Primária: A + B + C

Fig. 4.3: Sintetização de uma relação a partir de uma cadeia mista

Uma bolha isolada dá origem a uma relação, em que todos os atributos pertencentes a ela fazem parte da chave. A figura 4.4, mostra a relação formada a partir da bolha isolada contendo os atributos (A, B, C).

Relação

A	B	C

chave primária : A + B + C

Fig. 4.4: Gerando relação a partir de bolha isolada

4.2 - Esquemas Relacionais Mínimos

O cálculo de uma cobertura mínima para um diagrama de dependências, gera um diagrama mínimo, eliminando as dependências funcionais redundantes e os atributos redundantes em cada relação. Conseqüentemente, o esquema relacional gerado a partir deste diagrama de dependências é mínimo, pois ele só contém as relações necessárias e os atributos necessários em cada relação. A figura 4.5 ilustra um mapeamento dos dois diagramas de dependências "a" e "b". Para o diagrama "a" não foi calculada uma cobertura mínima, e por este motivo ele gera uma relação com um atributo a mais (relação 1.a). No diagrama "b" a dependência funcional A --> C (que é transitiva) foi identificada como redundante, graças ao cálculo de uma cobertura mínima. Portanto, o atributo redundante (C) não aparece na relação 1.b, sendo o esquema relacional correspondente mínimo.

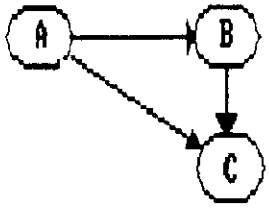


diagrama (a)

Esquema relacional (a)

Relação 1.a | A | B | C |

(atributo "C" redundante)

Relação 2.a | B | C |

diagrama mínimo (b)

Esquema Relacional Mínimo (b)

Relação 1.b | A | B |

Relação 2.b | B | C |

Legenda:
— (chave primária)
- - (chave estrangeira)

Fig. 4.5: Gerando esquemas relacionais mínimos

4.3 - Ligações entre Relações Geradas

Após gerar o esquema relacional e determinar as chaves primárias e estrangeiras de cada relação, **NORMALIZADOR** automaticamente determina como realizar a navegação neste esquema relacional, através de junções naturais.

Em uma relação gerada a partir de uma cadeia funcional, **NORMALIZADOR** estabelece a ligação entre a chave estrangeira e a chave primária, indicando como se pode fazer a junção natural de duas relações. A figura 4.6 mostra a ligação das três relações geradas, a partir da cadeia funcional A --> B --> C --> D.

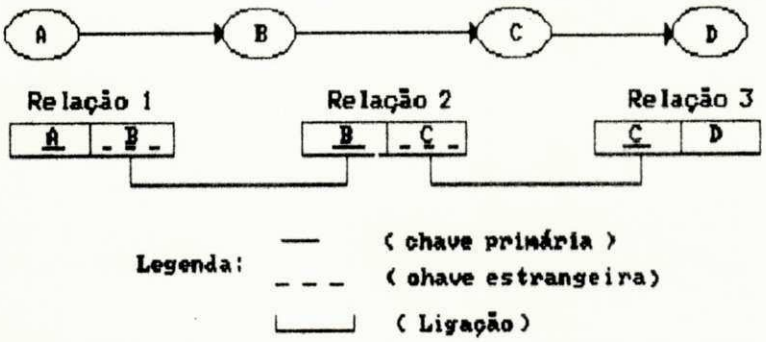


Fig. 4.6: Ligação de relações a partir de uma cadeia de DF's

NORMALIZADOR também estabelece ligações entre relações que têm atributos com o mesmo domínio (ver figura 4.7), e ainda entre relações que tem o mesmo atributo (ver figura 4.8).

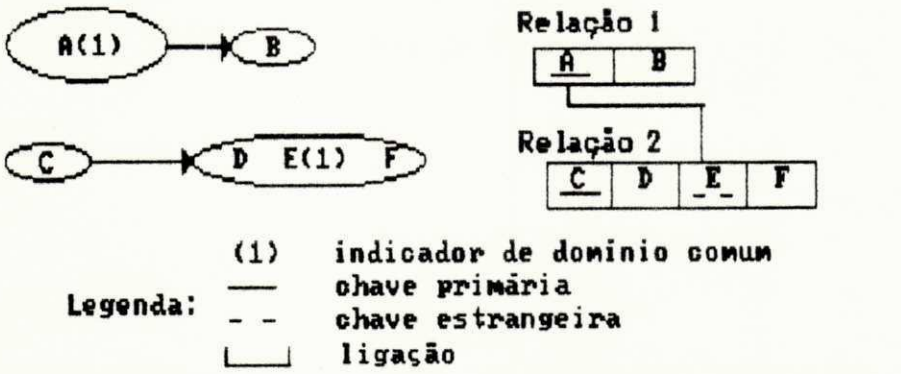


Fig. 4.7: Ligação de relações via atributos de mesmo domínio

4.4 - Um exemplo de Mapeamento Relacional

Nesta seção, vamos mostrar o esquema relacional gerado por **NORMALIZADOR**, correspondente ao diagrama de dependências de um banco de dados para o controle acadêmico de uma universidade (figura 2.13). Este diagrama foi construído no capítulo II, seção 2.5.

O esquema relacional da figura 4.8, mostra as relações geradas a partir do diagrama de dependências da figura 2.13. Retângulos representam as relações. Os componentes da chave primária de cada relação são sublinhados por uma linha horizontal cheia, enquanto que os pertencentes a chave estrangeira são sublinhados por uma linha horizontal pontilhada. As ligações entre as relações são representadas através de linhas verticais. Observe que a relação contendo os atributos: disciplina, turma e estudante se liga a relação contendo os atributos: estudante, curso e nome. Pois, estas relações contém o atributo comum (estudante).

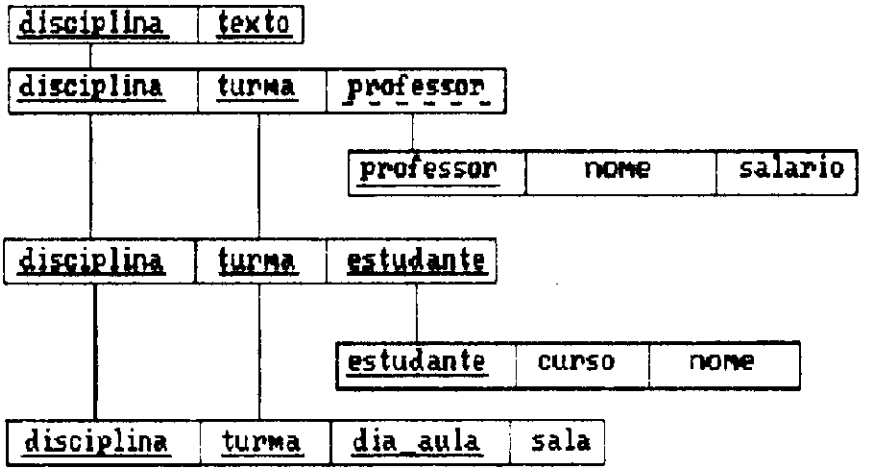


Fig. 4.8: Esquema Relacional de um Banco de Dados de controle acadêmico

4.5 - Prova da Normalização do Esquema Relacional

Provaremos agora, informalmente, que todo esquema relacional composto a partir de um diagrama de dependências correto está normalizado até a 5ª forma normal.

- 1ª forma normal (todo atributo de uma relação deve ser atômico): é satisfeita porque as regras do diagrama de dependências requerem esta condição (ver capítulo II, seção 2.4).

- 2ª forma normal (os atributos de uma relação em 1ª forma normal, que não são chave primária devem ser totalmente dependentes dela): é satisfeita porque, num diagrama de dependências, os atributos são dependentes funcionalmente ou de um atributo dentro de uma bolha determinante (transformado em chave primária), ou então de uma cadeia de dependências multivaloradas (todos os atributos da cadeia compondo uma chave primária). (ver figura 4.9).

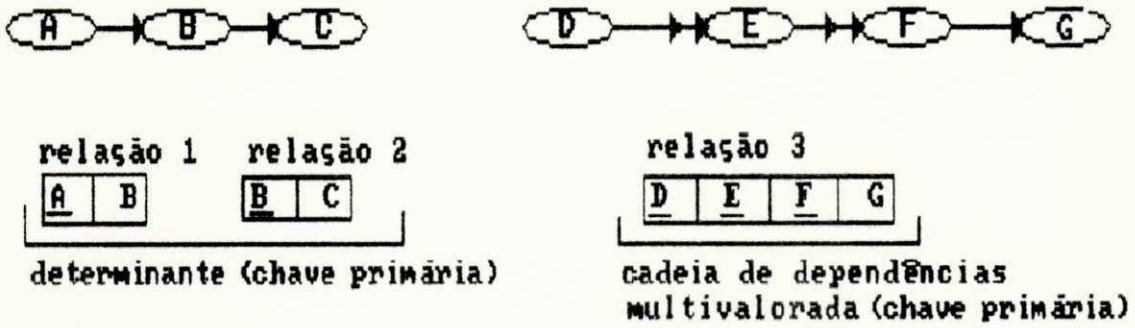


Fig. 4.9: Composição de relações em 2ª forma normal

- 3ª forma normal (uma relação R está em 3ª forma normal, se e somente se estiver na 2ª forma normal e todos os atributos não chave forem dependentes não transitivos da chave primária): é satisfeita pois, uma dependência transitiva é eliminada no cálculo de uma cobertura mínima, e não gera nenhuma relação (ver figura 4.10).

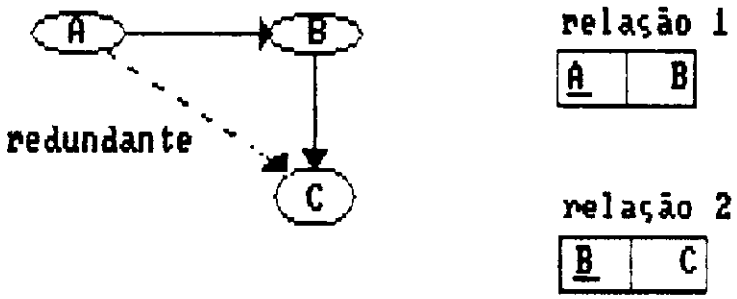


Fig. 4.10: Composição de relações em 3ª forma normal

- Forma normal de Boyce e Codd (BCNF) (requer que cada determinante funcional seja uma chave candidata): nos casos normais é satisfeita pois um determinante funcional vira chave primária (ver definição de bolha determinante no capítulo II, seção 2.4) ou chave alternativa. Existem situações em que a BCNF não pode ser alcançada sem perda de dependências [Beeri 78]. O caso clássico está ilustrado no diagrama 2 da figura 4.11. Na relação R2 (E, F, G) gerada, não há como garantir a dependência G --> F. Uma solução seria a imposição ao banco de dados de uma dependência de inclusão, no caso R2[F,G] = R3 [Casanova 82] [Schiel 84].

A figura 4.11 mostra a composição de relações a partir de bolhas determinantes. No primeiro diagrama, **NORMALIZADOR** gera a relação R1, com a chave primária contendo o determinante funcional (atributos A e B) e, uma chave alternativa contendo o outro determinante funcional (atributos B e C). O segundo diagrama mostra a solução para o caso clássico de perda de dependência citado anteriormente. No terceiro diagrama, a cadeia mista (H -->> I --> J) é equivalente a dependência funcional (H, I --> J), pois elas geram a mesma relação R4. Assim, os atributos pertencentes a parte multivalorada desta cadeia mista (H e I), funcionam como um determinante funcional, e viram a chave primária da relação R4.

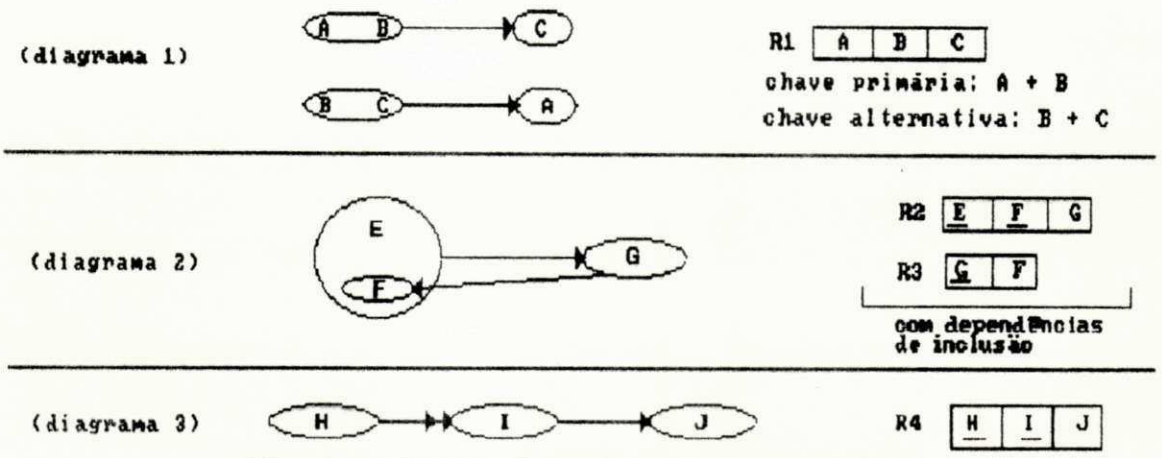


Fig. 4.11: Composição de relações em BCNF

- 4ª forma normal (requer que uma relação em 3ª forma normal não contenha duas ou mais dependências multivaloradas independentes): é satisfeita pela própria definição de cadeia de dependências multivaloradas, cujas dependências são interdependentes (ver capítulo II, seção 2.4). A figura 4.12 mostra dois exemplos de geração de relações a partir de cadeias multivaloradas. No primeiro, a relação X contém duas dependências (A -->> B) e (B -->> C), que não são, entretanto, independentes entre si. Já no segundo exemplo, (A -->> B) e (A -->> C) são independentes e, portanto, não ficam numa mesma relação.

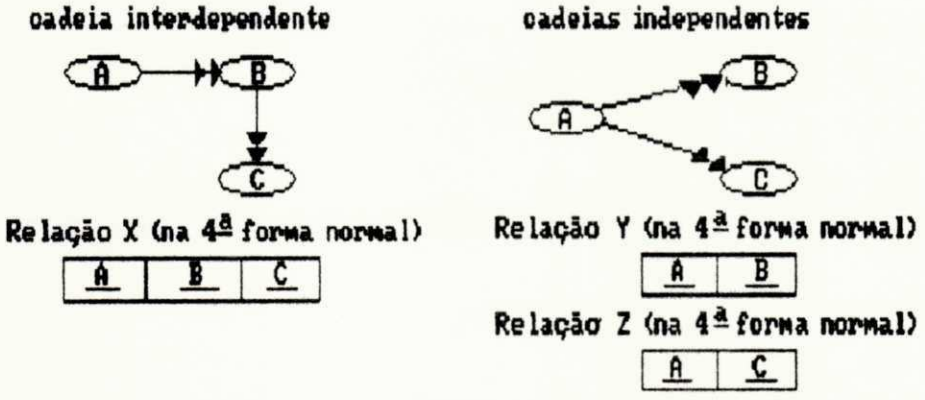


Fig. 4.12: Composição de relações em 4ª forma normal

- 5ª forma normal (uma relação em 4ª forma normal não pode ser decomposta, sem perda de informação, em três ou mais relações menores): novamente, a semântica de uma cadeia de dependências multivaloradas garante sua sintetização em uma relação na 5ª forma normal. Pelo fato dela ser interdependente, os valores de seus atributos devem ser conhecidos todos ao mesmo tempo. Assim, por exemplo, a relação 1 na figura 4.13 não pode ser decomposta em três relações, sem perdas. Entretanto, as outras relações (2, 3) separam os atributos "A", "B" e "C" em duas relações, pois o uso da bolha dupla no atributo "B" significa que não existe interdependência entre (A -->> B) e (B -->> C).

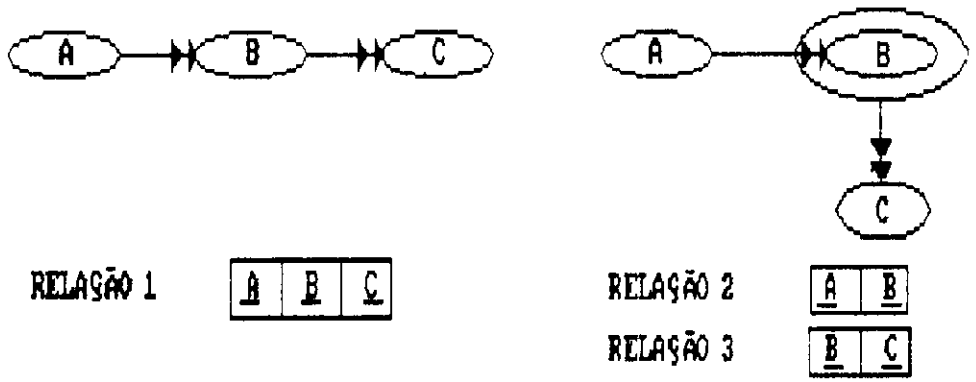


Fig. 4.13: Composição de relações em 5ª forma normal

Este capítulo completou a base teórica necessária a implementação de **NORMALIZADOR**. Nos próximos capítulos mostraremos como esta ferramenta foi implementada.

5 - Arquitetura do Sistema

Neste capítulo será apresentada a arquitetura do sistema juntamente com a funcionalidade de cada módulo componente da mesma. Também será dada uma descrição da interface de **NORMALIZADOR** com o usuário, seguida de uma simulação de utilização do sistema.

5.1 - Módulos do Sistema

O sistema **NORMALIZADOR** está dividido em 5 (cinco) módulos: Interface, Inicializar Estruturas de Dados, Detector de Redundâncias, Correção de Redundâncias e Gerador de Esquemas Relacionais. Os módulos da figura 5 representam a arquitetura de **NORMALIZADOR**. Em seguida será apresentada uma breve descrição da funcionalidade de cada módulo.

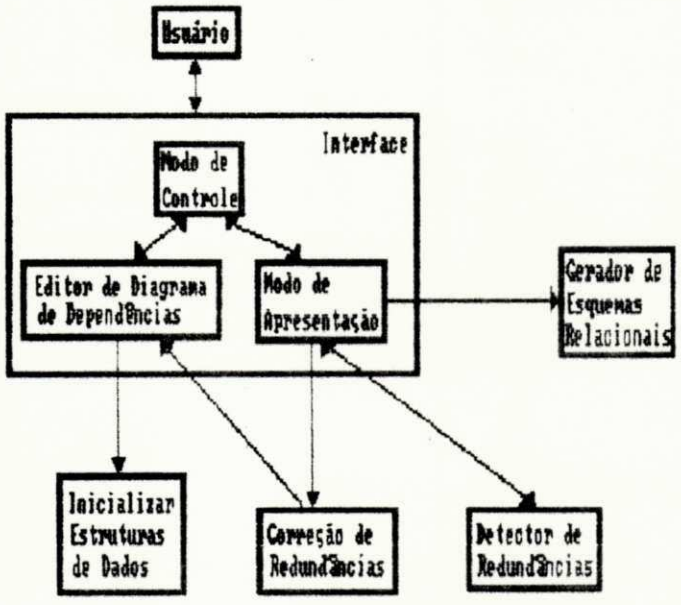


Fig. 5: Arquitetura do Sistema

5.1.1 - Interface

Este módulo é responsável pela comunicação com o usuário. A interface de **NORMALIZADOR** é dividida em três elementos básicos: Modo de Controle, Editor de Diagrama de Dependências e Modo de Apresentação. O projeto desta interface está detalhado na seção 5.2 (Interface com o usuário).

5.1.2 - Inicializar Estruturas de Dados

As estruturas de dados são inicializadas à medida em que está sendo editado um diagrama de dependências. Estas estruturas estão divididas em três blocos básicos:

- Estrutura de dados para edição gráfica.

- Estrutura de dados para cobertura mínima.

- Estrutura de dados para o mapeamento relacional.

Cada um destes blocos dá suporte respectivamente aos seguintes módulos do sistema:

- Editor de Diagrama de Dependências.

- Detector de Redundâncias e Correção de Redundâncias.

- Gerador de Esquemas Relacionais.

Estas estruturas de dados preenchidas, estão expostas detalhadamente no capítulo VI (Estrutura de Dados).

5.1.3 - Detector de Redundâncias

A detecção de redundâncias de um diagrama de dependências, é realizada segundo os algoritmos para encontrar uma cobertura mínima, explicados no capítulo III (Detectando Redundâncias). O Detector de Redundâncias implementa os dois métodos computacionais Tradicional e Eficiente, expostos no capítulo III (Detectando Redundâncias). A seção 5.2.5.3.1 (Encontrando e Corrigindo Redundâncias) apresenta de forma mais detalhada o uso deste detector.

5.1.4 - Correção de Redundâncias

Este módulo é responsável pela apresentação e correção das redundâncias, encontradas pelo módulo Detector de Redundâncias. Estas redundâncias são corrigidas através de um processo interativo, onde o usuário ré-edita o diagrama de dependências, eliminando cada uma das redundâncias expostas. A seção 5.2.5.3.1 (Encontrando e Corrigindo Redundâncias) apresenta mais detalhadamente este processo.

5.1.5 - Gerador de Esquemas Relacionais

Este módulo é o responsável pela geração do esquema relacional resultante a partir de um diagrama de dependências. Identificando também as ligações entre as relações neste esquema, baseado nas idéias mostradas no capítulo IV (Gerando Esquemas Relacionais Normalizados). O uso deste gerador está descrito detalhadamente na seção 5.2.5.3.2.

5.2 - Interface com o Usuário

Introdução

A interação entre o usuário e o computador tem sido uma das principais preocupações dos diversos produtos na área de informática lançados ultimamente. Cresce o número de usuários mais esclarecidos, com maior poder de contestação. Este fato, junto à grande concorrência entre os produtos, faz com que a preocupação na produção de software volte-se para o real objetivo, que é o de satisfazer plenamente o usuário final e não apenas a obtenção de lucro financeiro.

Descrição Funcional da Interface

Apresentaremos de forma breve a finalidade e o comportamento da ferramenta **NORMALIZADOR**, com o objetivo de dar maior clareza sobre as escolhas dos estilos de interface escolhidas.

Trata-se de uma ferramenta de apoio a projeto de Banco de Dados Relacional, que automatiza o processo de normalizar relações.

A interface é subdividida em três partes fundamentais: Modo de Controle, Modo editor e Modo de Apresentação.

O Modo de Controle gerencia os recursos oferecidos pela ferramenta, os quais são: construção de diagramas de dependências, eliminação de redundâncias de um diagrama de dependências e geração de esquemas relacionais normalizados.

O Modo editor oferece condições ao usuário para representar suas aplicações, segundo o modelo de dados orientado ao processo de normalização. Utilizando este editor, o usuário desenha suas aplicações através dos tipos

primitivos deste modelo, sendo advertido de forma interativa quando da construção de diagramas contendo erros semânticos e/ou sintáticos. O editor dispõe de facilidades peculiares à edição gráfica, tais como: rolamento de tela horizontal e vertical, manipulação de blocos de desenhos, gravação e recuperação de arquivos em disco, ajuda interativa, etc.

O Modo de Apresentação é responsável pelos resultados obtidos automaticamente a partir de um diagrama de dependências. Apresentando suas possíveis redundâncias e oferecendo condições para sua eliminação. Além de mostrar um esquema relacional normalizado, projeto de chaves e identificação das ligações entre as relações pertencentes a este esquema.

Características dos usuários

Um usuário desta ferramenta, não precisa ser necessariamente um projetista de banco de dados para poder utilizá-la. Bastando para isto, ter um conhecimento prévio da sua funcionalidade geral. Sua interface é bastante amigável e totalmente interativa Escondendo os conceitos de formas normais, de cobertura mínima, de projeto de chaves, etc Para suportar esta naturalidade de interação com a aplicação, foram utilizadas técnicas de interface, as mais variadas, como por exemplo:

- Ajuda "ON-LINE", na qual o usuário obtém informações imediatas sobre o que ele pode fazer a partir do contexto atual, no qual foi solicitada a ajuda.

- Utilização de menus e de teclas de funções.

5.2.1 - Descrição Detalhada da Interface

Inicialização da Ferramenta

Para dar início ao trabalho na ferramenta, devem ser realizados os seguintes procedimentos: Ativar a ferramenta através da linha de comando do sistema operacional MS-DOS ou compatíveis:

C> NORM <CR)

onde "C>" indica a unidade de disco atual; "NORM" é o nome da ferramenta a ser ativada e "<CR>" é a indicação do acionamento da tecla "RETURN" ou "ENTER", pelo usuário. A partir daí, o usuário terá à sua disposição a ferramenta ativada para que possa desenvolver seus projetos aplicativos.

Estilo de Diálogo

O estilo de diálogo escolhido foi o de seleção por menus, auxiliados pelas teclas de funções. O número de opções de cada menu não é muito grande, facilitando a percepção por parte dos usuários. Os itens do menu podem ser acionados pela tecla quente (letra em destaque contida em cada opção do menu), o que facilita uma busca lógica da opção desejada.

Uma opção de ajuda está contida em cada menu da ferramenta, descrevendo a funcionalidade das opções contidas no menu exposto, possibilitando a eliminação de eventuais dúvidas do usuário.

A navegação através das opções de um determinado menu, é realizada através das teclas de direcionamento do teclado (setas de direção). A escolha da opção é feita mediante o pressionamento da tecla "ENTER" sobre a opção atual. Isto

evita a escolha de uma opção inexistente no menu. Esta característica de navegação entre as opções de um menu, não está disponível para os menus contendo opções de edição. Pois, neste caso as setas de direção servem para navegação na tela editada.

Para que o usuário possa saber em que opção ele se encontra em um determinado instante, foi utilizado o recurso de vídeo reverso. Assim, a opção atual do menu recebe uma característica gráfica diferente das demais, facilitando a percepção por parte do usuário.

5.2.2 - Apresentação da Informação

O "layout" de telas é dividido em áreas bem definidas, dispostas na tela procurando o melhor efeito visual no espaço disponível. Estas áreas componentes possuem funções específicas e posições pré-determinadas, facilitando uma familiarização mais rápida com a ferramenta por parte do usuário, pois ele de antemão terá noção do que acontece a cada momento em função da informação exibida em cada uma das áreas.

Existem três tipos de telas: tela de controle, tela de edição gráfica e tela de apresentação. Estas contém áreas comuns e áreas específicas, o que as tornam funcionalmente diferentes. As figuras 5.1, 5.2 e 5.3 mostram o "layout" destes tipos de telas.

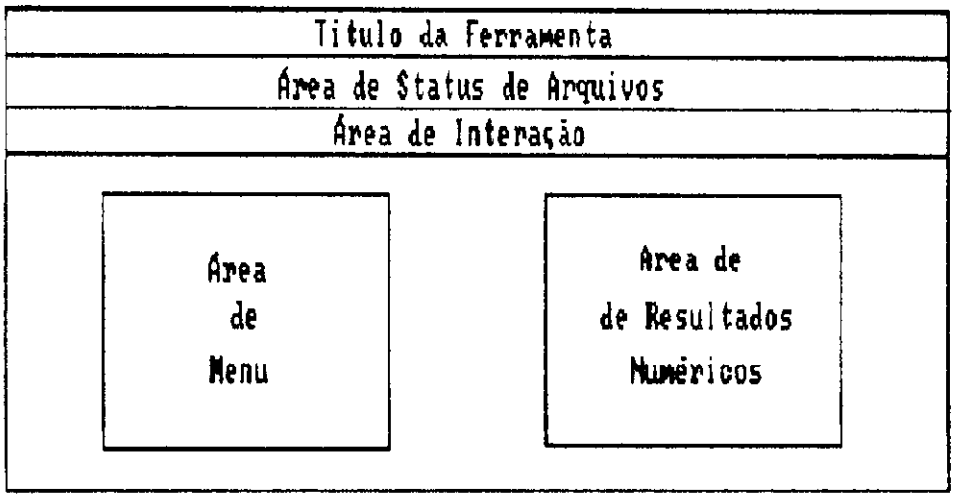


Fig. 5.1: Apresentação da Informação (Modo de Controle)

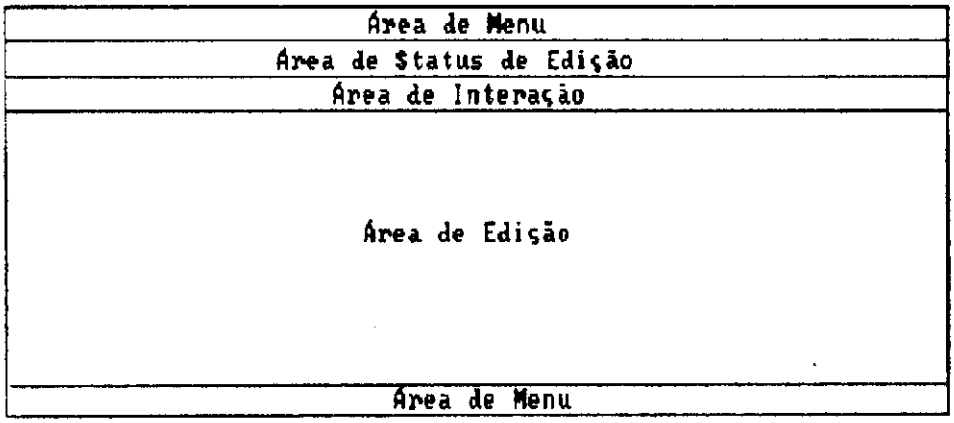


Fig. 5.2: Apresentação da Informação (Modo Editor)

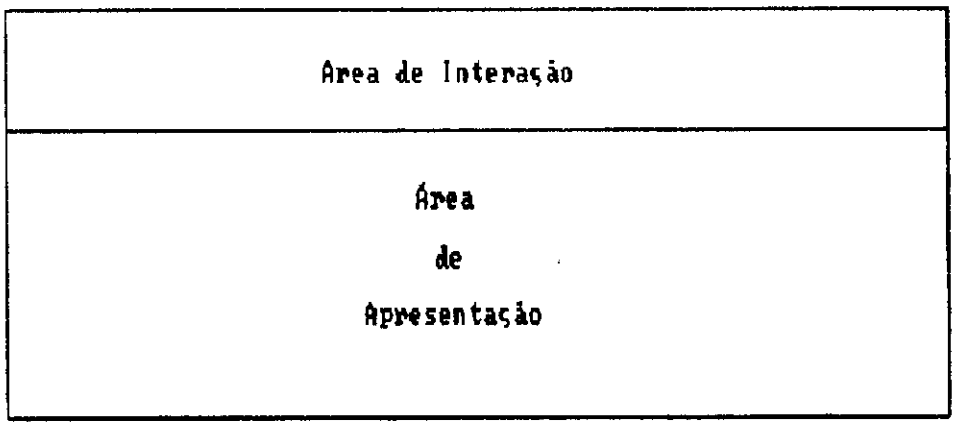


Fig. 5.3: Apresentação da Informação (Modo de Apresentação)

Área de Status de Arquivos

Esta área faz parte da tela de controle da ferramenta, expondo os arquivos de dados manipuladas pela mesma. São quatro tipos de arquivos contendo em seus nomes, os seguintes sufixos: ".FIG", ".COB", ".REL" e ".SAI".

"*.FIG" - é o diagrama de dependências atualmente editado, armazenado em disco.

"*.COB" - é o arquivo em disco fruto de uma cobertura mínima, para o diagrama atualmente editado. Neste arquivo são exibidos as possíveis redundâncias (de dependências funcionais e de atributos).

"*.REL" - é o arquivo em disco contendo o esquema relacional, correspondente ao diagrama atualmente editado. Aqui, são exibidos as relações formadas e o projeto de chaves para cada uma delas. Este arquivo também contem as possíveis ligações entre estas relações, através de junções naturais.

"*.SAI" - é o arquivo de saída contendo todos os serviços oferecidos por **NORMALIZADOR**. Este, representa um resumo final do projeto de banco de dados para uma aplicação. Aqui são listadas as dependências funcionais e/ou multivaloradas pertencentes ao diagrama atualmente editado, as redundâncias encontradas, juntamente com as relações, chaves e junções naturais. Este arquivo pode ser enviado para a impressora.

O caractere "*", escrito anteriormente, junto de um sufixo, significa qualquer nome.

Área de Interação

Esta área faz parte de todos os tipos de telas da ferramenta, ela é a responsável pela comunicação com o usuário. Fornece todas as mensagens explicativas sobre o contexto atual em processamento, mostrando o que deve ser feito pelo usuário. Quando o usuário comente algum erro de processamento, é exibida uma mensagem de erro acompanhada de um sinal sonoro. Em algumas situações, o usuário precisa entrar com alguns dados para realizar uma operação. Neste caso, os dados serão obtidos também nesta área (por exemplo obtenção do nome de um arquivo de dados).

Área de Resultados Numéricos

Esta área faz parte da tela de controle, sendo a responsável pela exibição dos resultados numéricos, para uma cobertura mínima e formação do esquema relacional, correspondente ao diagrama atualmente editado.

Durante o cálculo de uma cobertura mínima são exibidas nesta área os seguintes valores numéricos: dependências processadas, atributos redundantes e dependências redundantes encontradas, total de redundâncias encontradas e tempo em segundos para realizar este cálculo (sem contar entrada e saída em disco). A título de informação, são mostrados os nomes do diagrama de dependências atualmente editado e o nome do arquivo gerado, contendo os resultados da cobertura mínima (*.COB).

Durante a composição de um esquema relacional, correspondente ao diagrama de dependências atualmente editado, são exibidas nesta área o número de relações formadas e a quantidade de junções naturais identificadas, entre estas relações. A título de informação, são mostrados os nomes do diagrama de dependências e o nome do arquivo gerado, contendo os resultados desta operação (*.REL).

Área de Menu

Esta área está reservada para a apresentação de menus que serão expostos ao usuário, no modos de controle e editor. A descrição dos menus e seu comportamento foram relatadas no tópico Estilo de Diálogo da seção 5.2.1.

Área de Status de Edição

Esta área faz parte da tela de edição, sendo a responsável pela exibição de informações, inerentes à edição gráfica de um diagrama de dependências. Aqui, são mostradas os números da linha , coluna e página correspondentes a posição atual do cursor na tela. Além do número correspondente a velocidade do cursor na tela, que pode variar de 01 a 05 (a velocidade número 01 fornece uma melhor precisão para desenhar, enquanto que a de número 05 por ser a maior, é melhor para navegar na tela). Esta velocidade é alterada, selecionando a opção "F4-velocidade" do menu no canto inferior da tela de edição (ver figura 5.6). Como complemento, é exibido o nome do diagrama atualmente editado.

Área de Edição

Esta área faz parte da tela de edição, sendo reservada para desenhar diagramas de dependências. O cursor se movimenta aqui, através das teclas de direcionamento do teclado (PgUp, PgDn, Home, End e as setas de direção). Esta navegação, proporciona uma atualização da área de status de edição e um rolamento horizontal ou vertical, quando o cursor atinge os limites da área.

Área de Apresentação

Esta área faz parte da tela de apresentação, sendo reservada para a visualização dos resultados de processamento, das seguintes tarefas realizadas por **NORMALIZADOR**: Cobertura mínima (detectação e correção de redundâncias); composição do esquema relacional resultante; e visualização dos arquivos de dados manipulados ("*.COB", "*.REL" e "*.SAI"). Nesta área, é mostrada uma linha a cada instante destacada em vídeo reverso. A movimentação desta linha, é conseguida através das teclas de direcionamento do teclado (PgUp, PgDn, Home, End e as setas de direção). Esta navegação proporciona um rolamento horizontal ou vertical na tela, quando a linha em destaque atinge os limites desta área.

5.2.3 – Métodos de Ajuda

A ferramenta oferece acesso aos seus métodos de ajuda das seguintes formas:

- Através da tecla de função do teclado (F1).

- Na apresentação de mensagens explicativas e de erro, contidas na Área de Interação apresentada anteriormente.

- uma breve ajuda "off-line", que será apresentada posteriormente no tópico (Simulando uma utilização de **NORMALIZADOR**).

5.2.4 – Ambiente de Operação

A ferramenta trabalha basicamente em um ambiente PC-XT e compatíveis, uma unidade de disco rígido ou flexível, um monitor de vídeo (CGA, Hércules, HGA, VGA, etc.), teclado modelo XT/AT e 640K bytes de memória RAM.

Na sua versão atual, **NORMALIZADOR** possui aproximadamente 14.500 (quatorze mil e quinhentas) linhas de código fonte, programadas na linguagem de programação C (Turbo C - Versão 2.0).

O tamanho de um diagrama de dependências editado é a memória principal disponível, menos o tamanho do arquivo executável (267 Kbytes). Assim, para uma memória de 100 Kbytes podemos editar um diagrama contendo aproximadamente: 130 dependências, 260 atributos e 260 bolhas. Cada dependência ocupa aproximadamente 26 bytes, enquanto que cada atributo (considerando seu tamanho máximo que é 75 bytes), bolha, e outras estruturas de dados que suportam a ferramenta (grafo de cadeias de dependências, conjunto de dependências funcionais e lista de relações) mostradas no capítulo VI, ocupam respectivamente 89, 24 e 534 bytes.

5.2.5 - Simulando uma Utilização de NORMALIZADOR

Após executado o procedimento visto no tópico Inicialização da Ferramenta, a tela de controle aparece no vídeo conforme a figura 5.4:

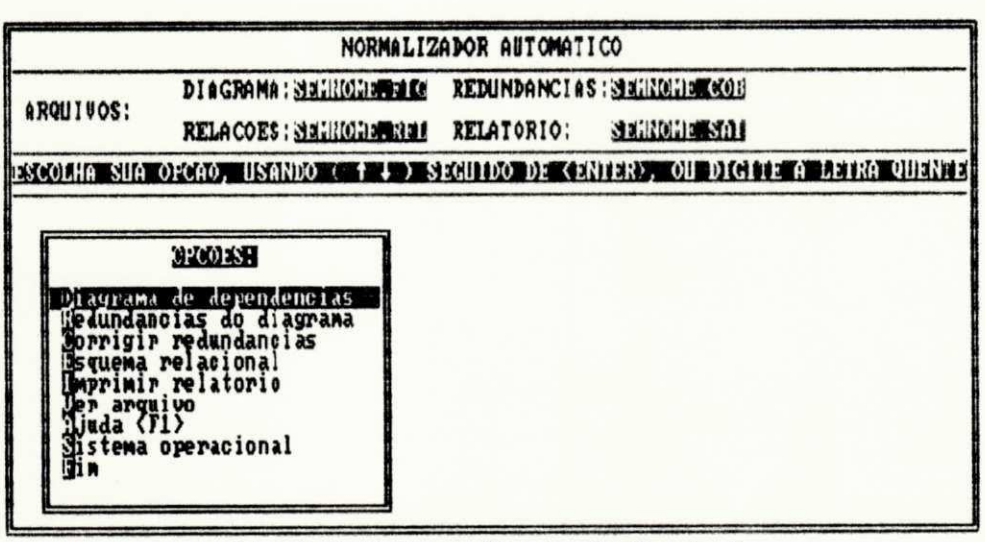


Fig. 5.4: Tela do Modo de Controle

5.2.5.1 – Ajuda (Tela de Controle)

Caso seja selecionada a opção de ajuda (ou pressionada a tecla <F1>) neste nível do sistema, é mostrada uma tela de ajuda, a qual contem breves explicações sobre as escolhas possíveis, apresentadas no menu da tela de controle da ferramenta. A tela de ajuda é apresentada conforme a figura 5.5

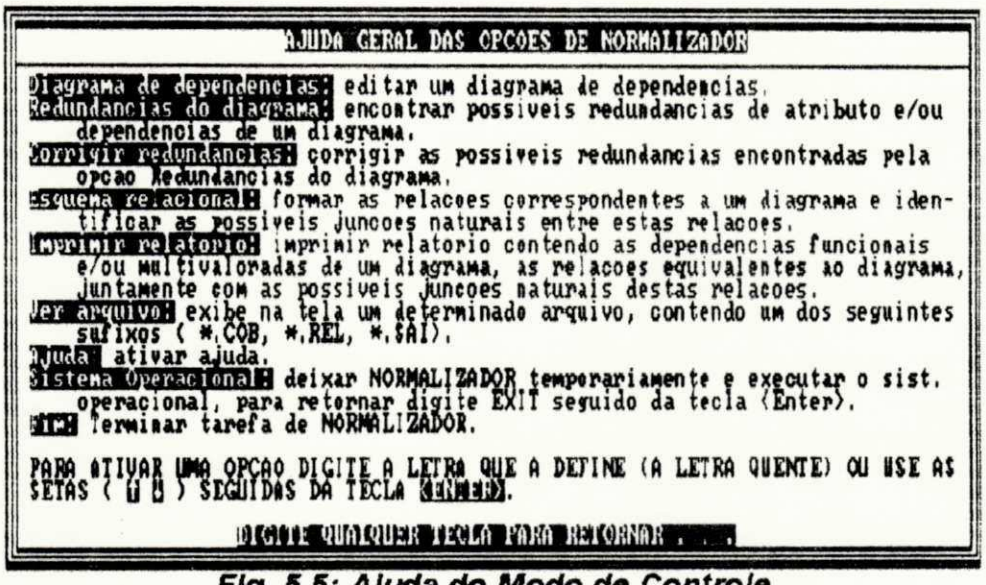


Fig. 5.5: Ajuda do Modo de Controle

Uma observação se faz necessária. As palavras que compõem as telas, que porventura possuem acentos gráficos, não foram acentuadas em virtude dos caracteres acentuados só aparecerem na tela em modo texto. Portanto, para manter um padrão não foram usados acentos nestas palavras.

5.2.5.2 – Modo Editor

Este nível é alcançado selecionando a opção "Diagrama de dependências", no menu da tela de controle. Trata-se de um editor gráfico, que possibilita o desenho de um diagrama de dependências, esquematizando uma aplicação no modelo de

dados orientado ao processo de normalização, apresentado no capítulo II.

Este editor possibilita o desenho dos elementos deste modelo (dependência funcional, dependência multivalorada, atributos, bolha, indicador de domínio comum). Estes elementos são desenhados respectivamente selecionando as opções do menu no canto superior da tela de edição (Funcional, Multivalorada, Atributo, Bolha e Domínio). A eliminação destes elementos é realizada selecionando a opção "<F6>-eliminar" do menu no canto inferior desta mesma tela.

O editor de diagramas de dependências é totalmente consistente com o modelo orientado à normalização. Ele exibe mensagens de erro, quando o usuário desrespeita uma restrição deste modelo (por exemplo, a formação de cadeias de dependências sem obediência à regra de cadeias de dependências funcionais, multivaloradas e mistas). Além de oferecer características peculiares à edição gráfica, tais como: rolamento de tela horizontal e vertical, gravação e recuperação de diagramas em disco, manipulação de bloco de desenhos na tela. Estas últimas três características citadas, são conseguidas selecionando as seguintes opções, respectivamente dispostas no menu inferior da tela de edição: "F2-salvar", "F3-recuperar" e "F5-bloco". A figura 5.6 mostra a tela do editor gráfico de **NORMALIZADOR**.

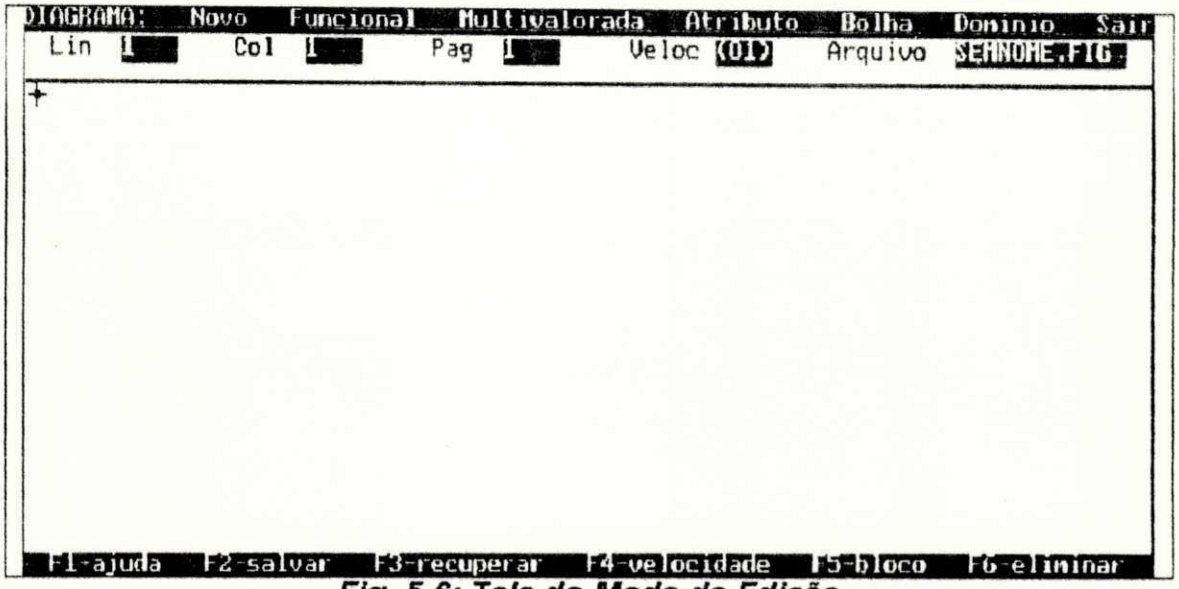


Fig. 5.6: Tela do Modo de Edição

Ajuda (Tela de Edição)

Caso seja pressionada a tecla <F1> no modo de edição, é mostrada uma tela de ajuda, a qual contém breves explicações sobre as escolhas possíveis, apresentadas nos menus superior e inferior da tela de edição. A tela de ajuda é apresentada conforme a figura 5.7

Fig. 5.7: Ajuda do Modo de Edição

Exemplo Prático do Editor de Normalizador

Utilizando o editor de **NORMALIZADOR**, construiremos um diagrama de dependências de um banco de dados, para o controle acadêmico de uma universidade. Este diagrama foi explicado no capítulo II (seção 2.5 - Construindo um Diagrama de Dependências).

Inicialmente vamos representar o conjunto de atributos {CURSO, NOME}. Navegue com o cursor até a posição na tela onde o atributo será editado (se necessário altere a velocidade do cursor pressionando a tecla <F4>). Selecione a opção "Atributo" pressionando a tecla <A>. Neste momento o cursor toma a forma de um quadrado, indicando que estamos em modo de edição de texto. As seguintes tarefas devem ser realizadas agora:

- Digite o nome dos atributos utilizando normalmente o teclado.

- Utilize a tecla de <Backspace> para concertar eventuais erros de digitação.

- Se a posição dos atributos na tela não for a desejada, posicione-os utilizando as setas de direcionamento do teclado.

- Para aceitar os atributos editados digite a tecla <Enter>, em caso contrário pressione <Esc> e recomece uma nova edição.

A figura 5.8 retrata a edição de atributos:

Fig. 5.8: Edição de Atributos

Para desenhar a bolha que conterá os atributos anteriormente definidos, inicialmente posicione o cursor na tela, de tal forma que ele fique bem próximo a eles. Agora, ative a opção "Bolha" digitando a tecla <B>. Neste momento, aparece na tela uma bolha e uma mensagem de orientação na área de interação. O usuário deve proceder da seguinte maneira:

- utilize as setas de direcionamento do teclado para determinar o tamanho da bolha,

- é fundamental que a bolha contenha totalmente os atributos desejados,

- Para aceitar a bolha desenhada digite <Enter>, em caso contrário pressione <Esc> e recomece a operação.

A figura 5.9 mostra a definição de uma bolha via o editor de **NORMALIZADOR**. Uma observação se faz necessária, a ordem utilizada neste exemplo foi definir o atributo e posteriormente a bolha contendo o mesmo. Esta ordem não é obrigatória, se o usuário preferir ele poderá definir

primeiramente a bolha e, em seguida editar os atributos dentro da mesma.

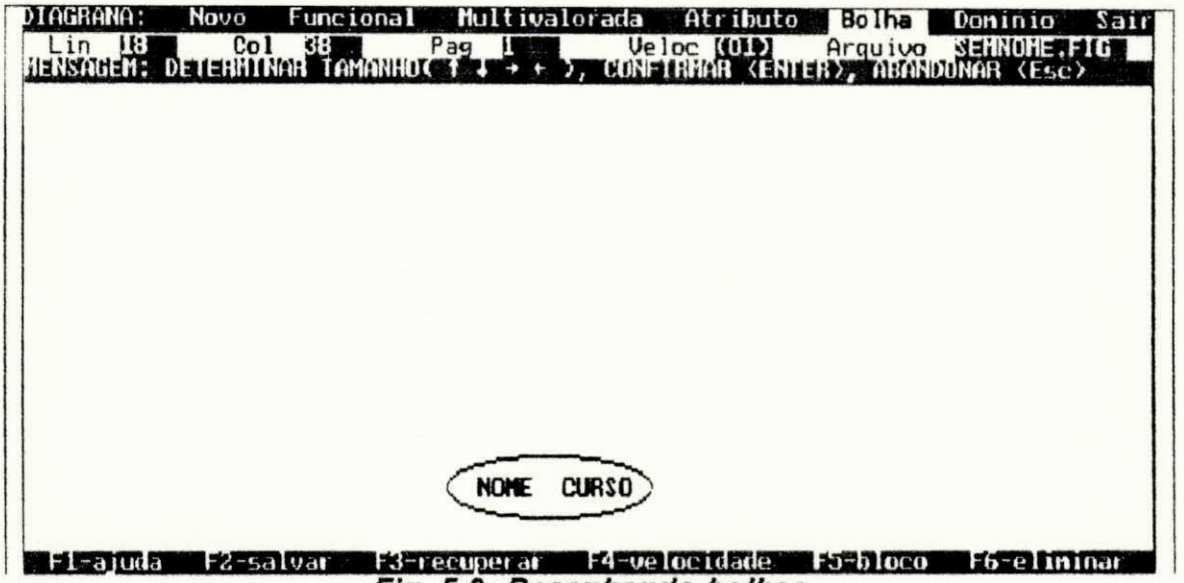


Fig. 5.9: Desenhando bolhas

Agora vamos definir a dependência funcional (ESTUDANTE --> CURSO, NOME). Primeiramente, desenhe da maneira explicada anteriormente uma bolha contendo o atributo "ESTUDANTE". Em seguida, selecione a opção "Funcional" através da tecla <F>. Neste momento, aparece a seguinte mensagem na área de interação:

"MARQUE A BOLHA ORIGEM COM <Enter> OU TECLE <Esc> PARA RETORNAR".

O usuário deve fazer as seguintes operações:

- localizar o cursor em qualquer parte interna a bolha, contendo o conjunto de atributos origem (lado esquerdo) da dependência funcional em questão (no nosso caso a bolha que contiver o atributo "ESTUDANTE").

- Para confirmar a escolha digitar a tecla <Enter>, em caso contrário pressionar a tecla <Esc> e recomeçe a operação.

Uma nova mensagem aparece na área de interação:

"MARQUE A BOLHA DESTINO COM <ENTER> OU TECLE <Esc> PARA RETORNAR".

O usuário deve proceder de forma análoga, à marcação da bolha origem, para marcar a bolha destino. A figura 5.10 retrata esta situação

Fig. 5.10: Definição de uma Dependência Funcional

Depois de marcar as bolhas origem e destino, representando respectivamente o lado esquerdo e direito de uma dependência funcional, o editor automaticamente desenha uma reta contendo uma bolinha cheia, próximo ao lado alvo (direito) desta dependência. Uma observação se faz necessária, o modelo de Smith utiliza uma seta simples e uma dupla para representar respectivamente uma dependência funcional e uma multivalorada. Entretanto, representamos a seta simples por uma bolinha cheia, enquanto que a dupla por duas bolinhas cheias, junto do lado alvo (direito) da dependência. Esta representação se deu por facilidade de implementação, pois o desenho é feito pelo editor e não pelo

usuário. A inclinação da reta pode variar mas a forma da bolinha mantem-se constante.

A figura 5.11 mostra a dependência funcional desenhada pelo editor. Uma outra observação convém ser dita: por questão de simplicidade utilizamos apenas uma bolha, contendo os dois atributos "NOME" e "CURSO", ligada à bolha determinante contendo o atributo "ESTUDANTE". Poderíamos ter desenhado uma bolha para cada atributo. Neste caso, teríamos que definir duas dependências funcionais ao invés de apenas uma. Esta alternativa é equivalente ao que foi feito (ver simplificação da bolha determinante comentada no capítulo II).

Fig. 5.11: Desenho de uma dependência funcional

Para tornar o diagrama de dependências editado permanente, selecionamos a opção "F2-salvar" digitando a tecla <F2>. Uma mensagem pedindo para ser digitado o nome do arquivo em disco que conterá o diagrama, é exibida na área de interação. O usuário deve digitar apenas o nome, pois o sufixo ".FIG" é acrescentado a este. Se houver erro durante a digitação, a tecla <Backspace> ajuda a corrigí-lo. A figura 5.12 mostra a gravação do diagrama atual com o nome

de "TESTE1.FIG". Quando o usuário desejar recuperá-lo, deve agir de forma análoga selecionado a opção "F3-recuperar".

Fig. 5.12: Salvando um diagrama em disco

Continuando nosso desenho, vamos agora desenhar as seguintes dependências :

PROFESSOR --> NOME, SALÁRIO

DISCIPLINA -->> TEXTO

A primeira delas é uma dependência funcional, que é desenhada da mesma forma como foi desenhada a dependência funcional (ESTUDANTE --> NOME, CURSO). A segunda que é multivalorada, é feita de maneira análoga. Entretanto, agora utilizamos a opção "Multivalorada", através da digitação da tecla <M>. A figura 5.13 mostra o desenho contendo as três dependências. Observe que a dependência multivalorada, possui junto ao seu lado alvo duas bolinhas cheias, ao invés de apenas uma contida na dependência funcional.

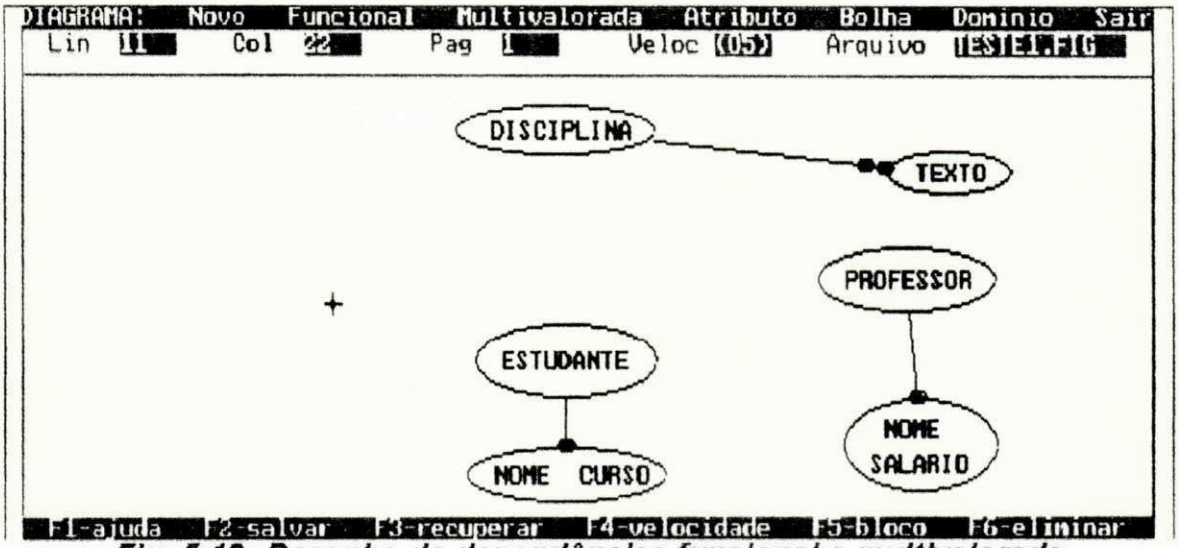


Fig. 5.13: Desenho de dependências funcional e multivalorada

Representaremos agora a cadeia de dependência mista (DISCIPLINA -->> TURMA -->> DIA_AULA --> SALA). Para desenhá-la utilizamos as operações já explicadas anteriormente. Primeiramente definimos a dependência multivalorada (DISCIPLINA -->> TURMA), porém, aproveitamos a bolha já desenhada contendo o atributo "DISCIPLINA". De maneira análoga, definimos as dependências multivalorada e funcional (TURMA -->> DIA_AULA) e (DIA_AULA --> SALA), completando assim a cadeia mista em questão.

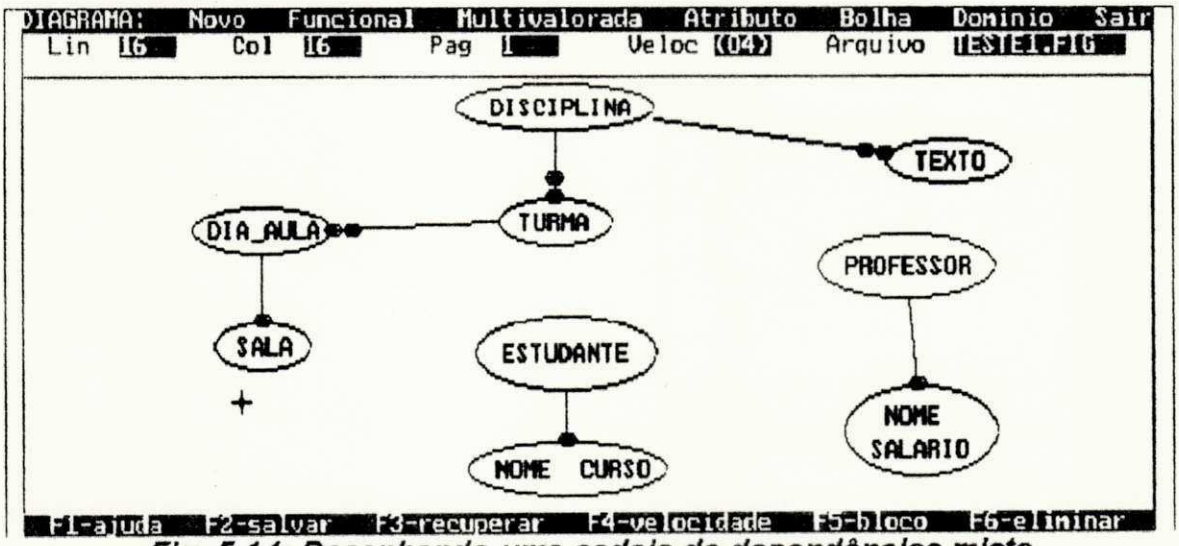


Fig. 5.14: Desenhando uma cadeia de dependências mista

Para finalizarmos nosso diagrama de dependências, desenharemos as cadeias de dependências mista e multivalorada:

DISCIPLINA -->> TURMA --> PROFESSOR

DISCIPLINA -->> TURMA -->> ESTUDANTE

Para desenhar a primeira delas, precisamos apenas definir a dependência funcional (TURMA --> PROFESSOR). A outra dependência multivalorada componente desta cadeia (DISCIPLINA -->> TURMA), já foi desenhada. Entretanto, precisamos desenhar uma bolha externa à bolha contendo o atributo professor, que já foi desenhada. Esta operação define uma bolha dupla contendo o atributo professor. Durante a marcação da dependência em questão (TURMA --> PROFESSOR), o usuário procede normalmente como já foi explicado. Porém, ele deve ter atenção de marcar a bolha externa desenhada, estabelecendo assim o limite entre a cadeia em questão e a dependência funcional (PROFESSOR --> NOME, SALÁRIO) representada anteriormente. Completando nosso desenho, definimos a segunda cadeia de dependências (DISCIPLINA -->> TURMA -->> ESTUDANTE), de maneira análoga a cadeia anterior (DISCIPLINA -->> TURMA --> PROFESSOR). Porém, desenhamos uma dependência multivalorada da bolha contendo o atributo "TURMA", que já foi desenhada, para uma bolha dupla que precisa ser definida, contendo o atributo "ESTUDANTE". A figura 5.15, mostra o desenho completo do diagrama de dependências de um banco de dados de controle acadêmico, construído com o editor de **NORMALIZADOR**.

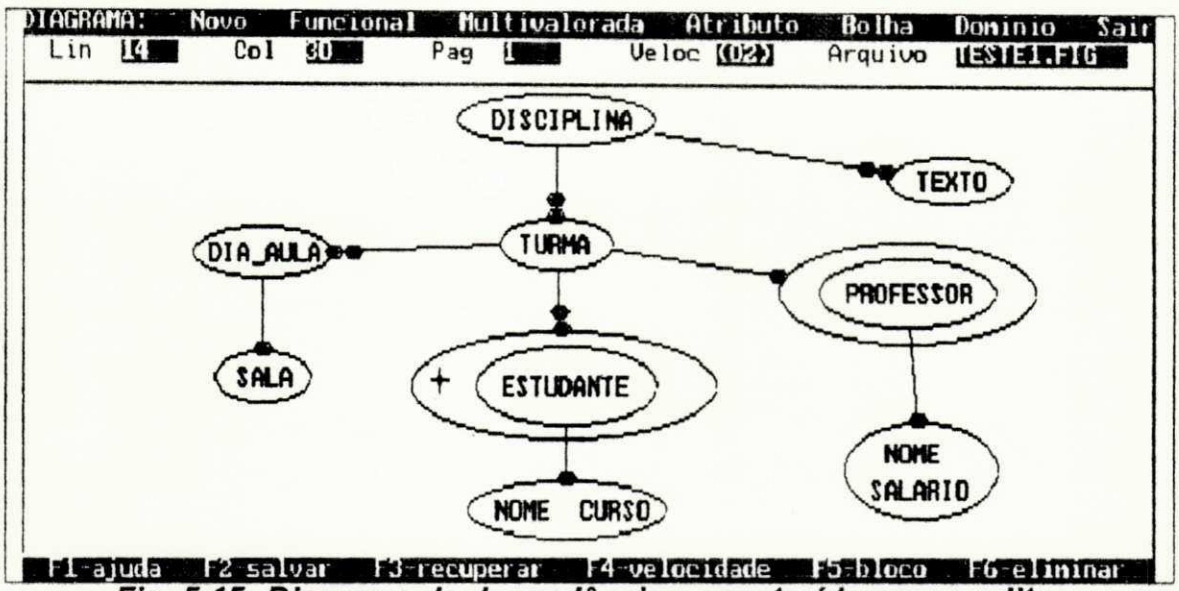


Fig. 5.15: Diagrama de dependências, construído com o editor gráfico de NORMALIZADOR

Ajuda (Tela da Opção Bloco)

A opção de bloco é utilizada para auxiliar na edição de um diagrama de dependências. O usuário pode apagar, copiar e mover blocos de desenhos na tela escolhidos por ele. Vale salientar, que os objetos (bolhas e atributos) devem estar incluídos por inteiro no bloco (pedaços destes objetos serão desprezados). As dependências vinculadas a uma bolha dentro do bloco serão reexaminadas automaticamente pelo editor, podendo ser redesenhadas ou apagadas (dependendo da operação). Esta opção é conseguida pressionando a tecla <F5> na tela de edição. Feito isto, aparece o menu da opção de bloco no canto inferior desta tela. Pressionando-se a tecla <F1> neste último menu, aparece uma tela de ajuda, contendo breves explicações sobre as escolhas possíveis. A figura 5.16 mostra esta tela de ajuda.

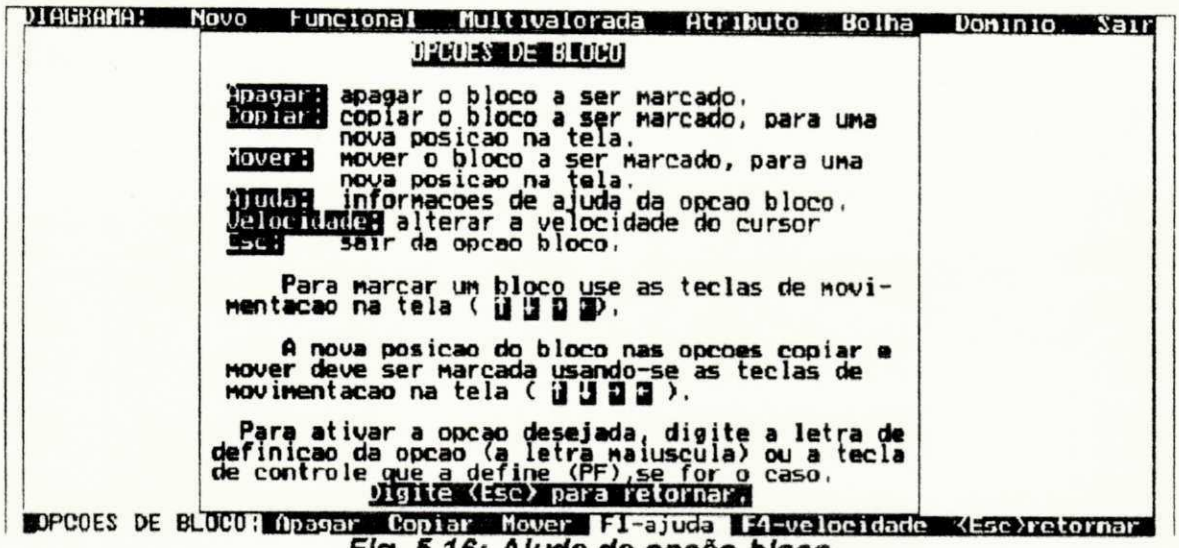


Fig. 5.16: Ajuda da opção bloco

Ajuda (Tela da Opção Eliminar)

A opção de eliminação também auxilia na edição de um diagrama de dependências. O usuário pode eliminar bolhas, atributos ou dependências. Aqui, as dependências vinculadas a uma bolha eliminada, serão eliminadas automaticamente pelo editor. Esta opção é conseguida pressionando a tecla <F6> na tela de edição, quando ativada é mostrado um menu no canto inferior da tela, contendo as opções possíveis. A figura 5.17 mostra a tela de ajuda da opção de eliminação conseguida pressionando-se a tecla <F1>.

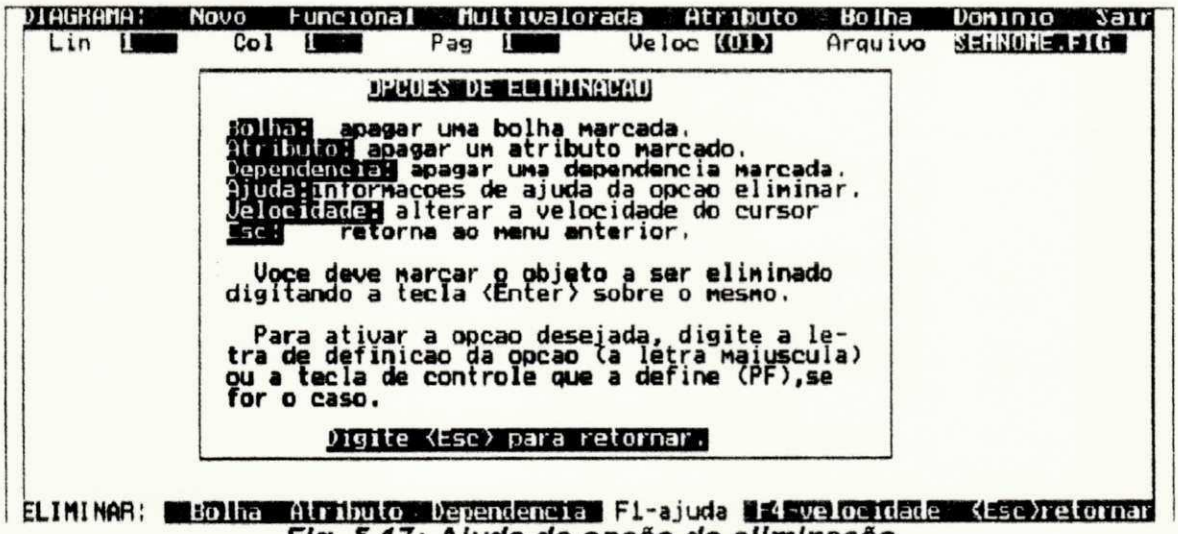


Fig. 5.17: Ajuda da opção de eliminação

5.2.5.3 - Modo de Apresentação

Este modo apresenta os resultados obtidos a partir de um diagrama de dependências editado, são eles:

- encontrar e corrigir redundâncias,
- formar o esquema relacional resultante,
- gerar relatório resultante,
- exibir os arquivos de dados do sistema.

5.2.5.3.1 - Encontrando e Corrigindo Redundâncias

Neste nível o usuário tem acesso ao detector de redundâncias de **NORMALIZADOR**, responsável pela depuração de um diagrama de dependências. Este detector oferece ao usuário os dois métodos computacionais para encontrar uma cobertura mínima (Método Tradicional e Método Eficiente) explicados no capítulo III (Detectando Redundâncias). Em ambos os casos, ele encontra automaticamente as possíveis redundâncias, emitindo explicações interativas informando ao usuário, de que maneira o diagrama de dependências deve ser corrigido.

Exemplo Prático do Detector de Redundâncias

Ilustraremos o uso do detector de redundâncias de **NORMALIZADOR**, encontrando as possíveis redundâncias do diagrama de dependências mostrado na figura 5.18. Este diagrama foi construído, segundo as orientações fornecidas no tópico "Exemplo Prático do Editor de **NORMALIZADOR**", mostrado na seção 5.2.5.2. As dependências funcionais contidas neste diagrama, são as que foram utilizadas nos exemplos das seções 3.3.1 e 3.3.2 do capítulo III (Detectando Redundâncias). Elas foram colocadas em um único

diagrama, para mostrar simultaneamente a detectação de redundâncias de dependências e de atributos.

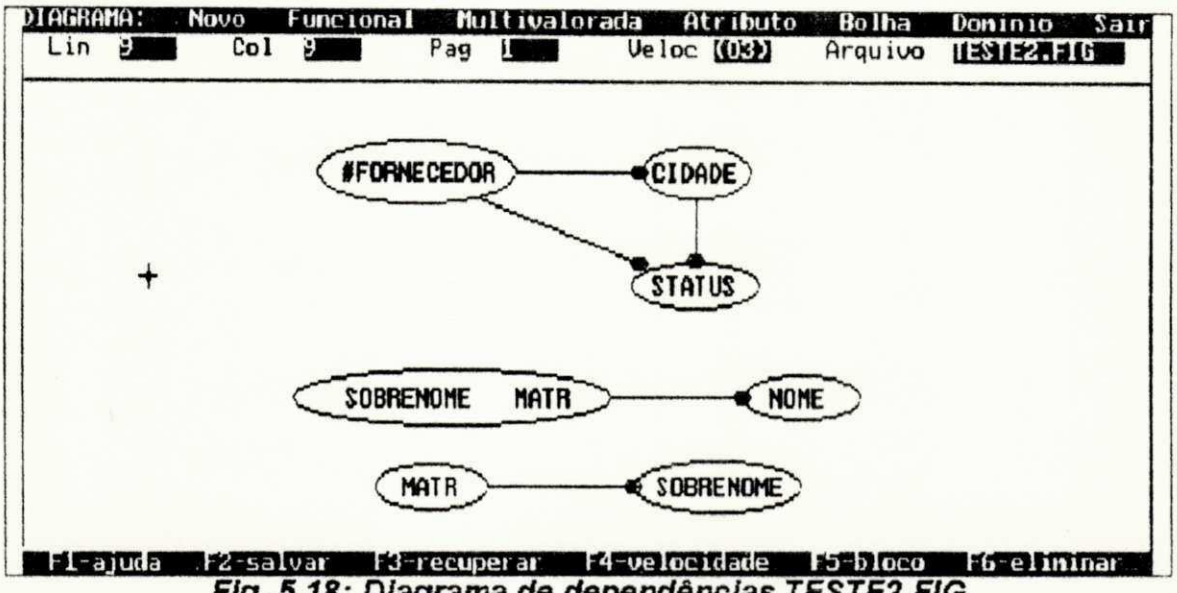


Fig. 5.18: Diagrama de dependências TESTE2.FIG

Depois de construído o diagrama de dependências "TESTE2.FIG", o usuário deve voltar ao menu de controle, selecionando a opção "Sair" da tela de edição. Prosseguindo, o usuário deve selecionar a opção "Redundâncias do diagrama", o que faz aparecer na área de interação a seguinte mensagem:

"DIGITE: <T> MÉTODO TRADICIONAL, <E> EFICIENTE OU <Esc> PARA SAIR"

O usuário deve escolher o método de cobertura mínima, que será utilizado pelo detector de redundâncias. Se ele digitar a tecla <T> será utilizada o Método Tradicional, caso seja pressionado a tecla <E> o detector utiliza o Método Eficiente. O acionamento da tecla <Esc> faz retornar ao menu de controle. A figura 5.19 mostra a tela de controle contendo na área de resultados numéricos, a exibição dos resultados da cobertura mínima para o diagrama "TESTE2.FIG".

utilizando o Método Eficiente. O tempo total de CPU gasto com este cálculo foi inferior a um segundo.

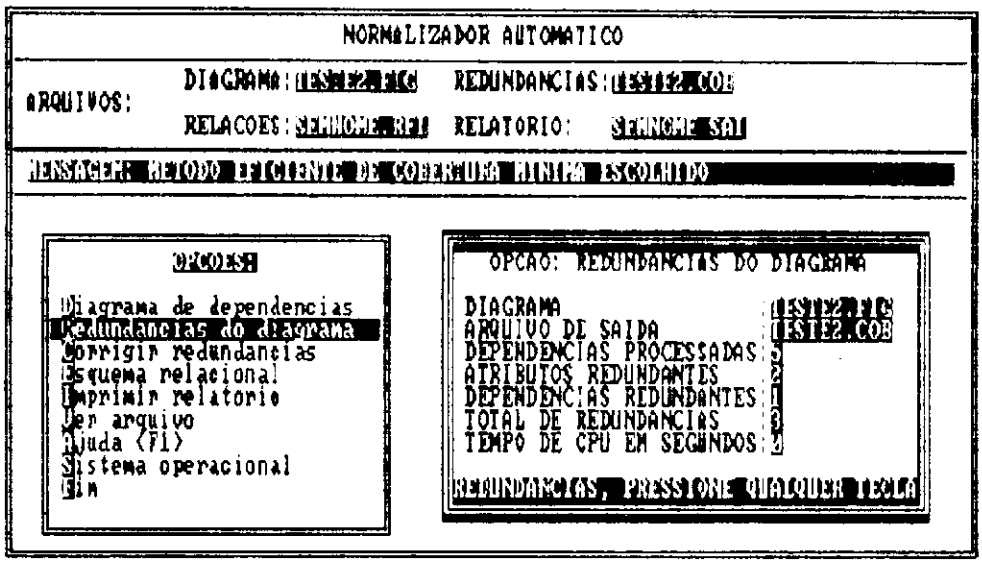


Fig. 5.19: Tela da opção Redundâncias do diagrama

Em seguida é mostrada a tela de apresentação, fruto dos resultados da cobertura mínima para o diagrama "TESTE2.FIG". Ela contém uma lista de redundâncias encontradas, expostas na área de apresentação. Já na área de interação, são exibidas as informações de orientação ao usuário.

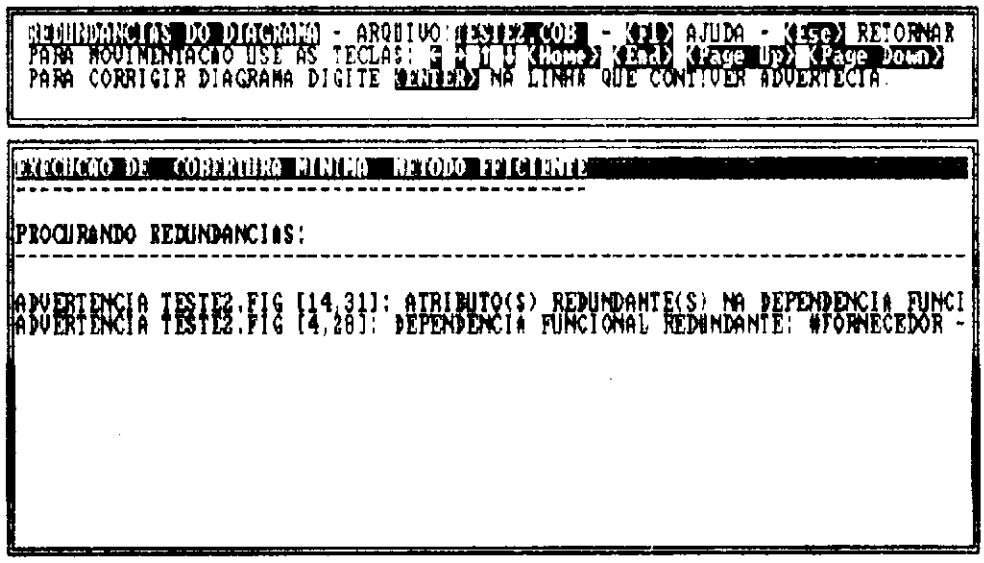


Fig. 5.20: Apresentação das redundâncias

Movimentando a linha da área de apresentação (conforme as indicações do tópico "Área de Apresentação", da seção 5.2.2), obtemos um rolamento horizontal para a esquerda, possibilitando a visualização da continuação da tela de apresentação. A figura 5.21 retrata esta situação:

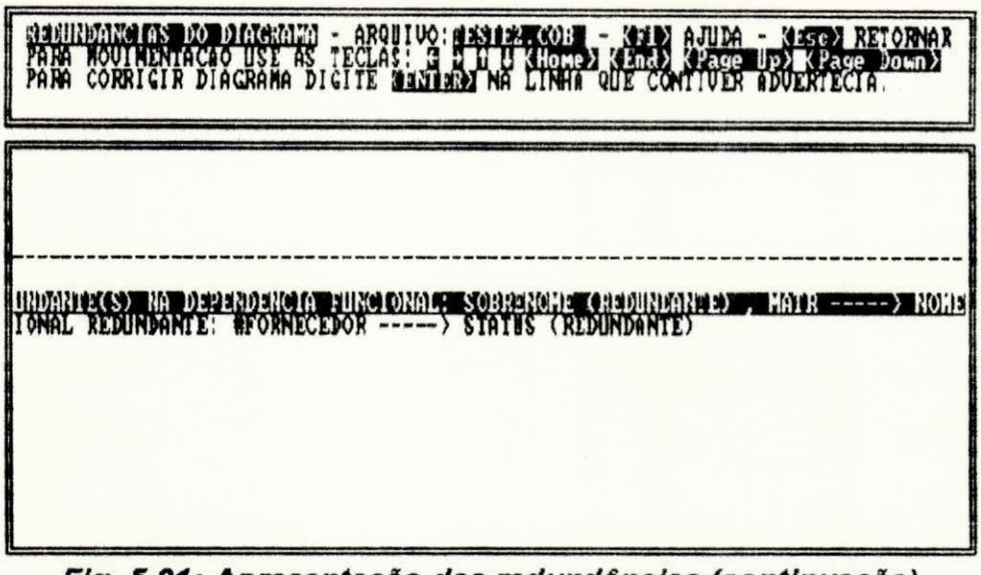


Fig. 5.21: Apresentação das redundâncias (continuação)

Para corrigir as redundâncias encontradas, o usuário deve editar o diagrama de dependências. Isto é conseguido através de um processo interativo, em que o usuário deve pressionar a tecla <Enter> sobre a linha, que contiver uma redundância. Esta, é identificada pela presença da palavra "ADVERTÊNCIA" no início da mesma. Após realizada esta operação, **NORMALIZADOR** passa automaticamente para o modo de edição, deslocando o cursor na tela para a linha e coluna onde a redundância foi encontrada. A figura 5.22 mostra o diagrama de dependências "TESTE2.FIG", contendo todas as redundâncias encontradas. Estas são facilmente identificadas, pois a dependência redundante está desenhada em linha pontilhada e os atributos redundantes destacadas em modo reverso. A mensagem exibida na área de interação da tela de edição, indica qual redundância deve ser corrigida neste instante. Neste nível a ativação da opção "F1-ajuda",

faz com que seja mostrada uma tela de ajuda, contendo as explicações necessárias para eliminação desta redundância (ver figura 5.23).

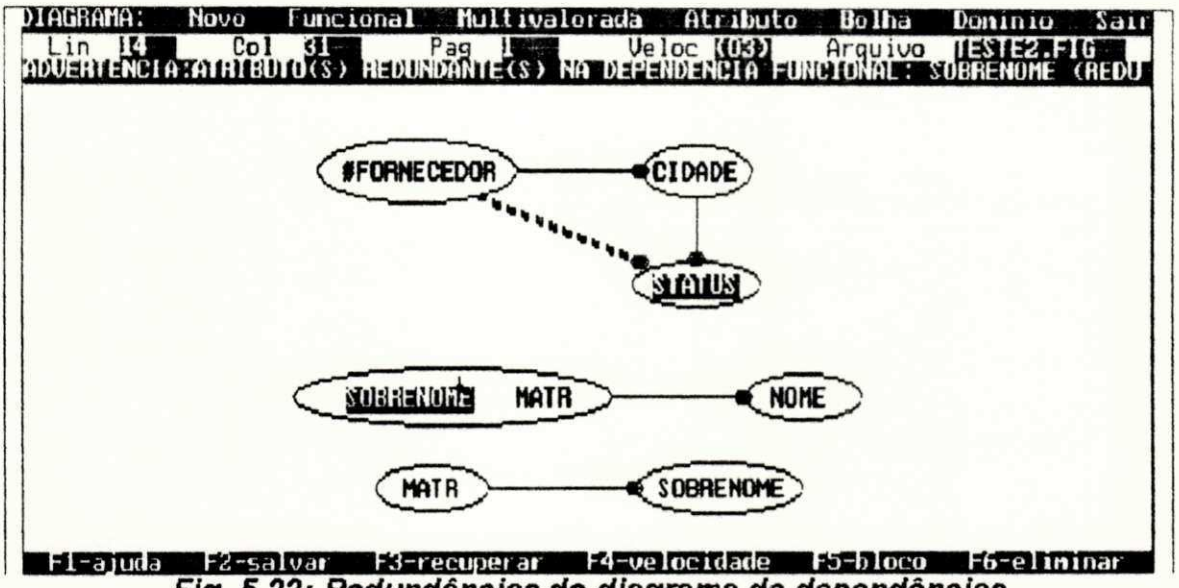


Fig. 5.22: Redundâncias do diagrama de dependências TESTE2.FIG

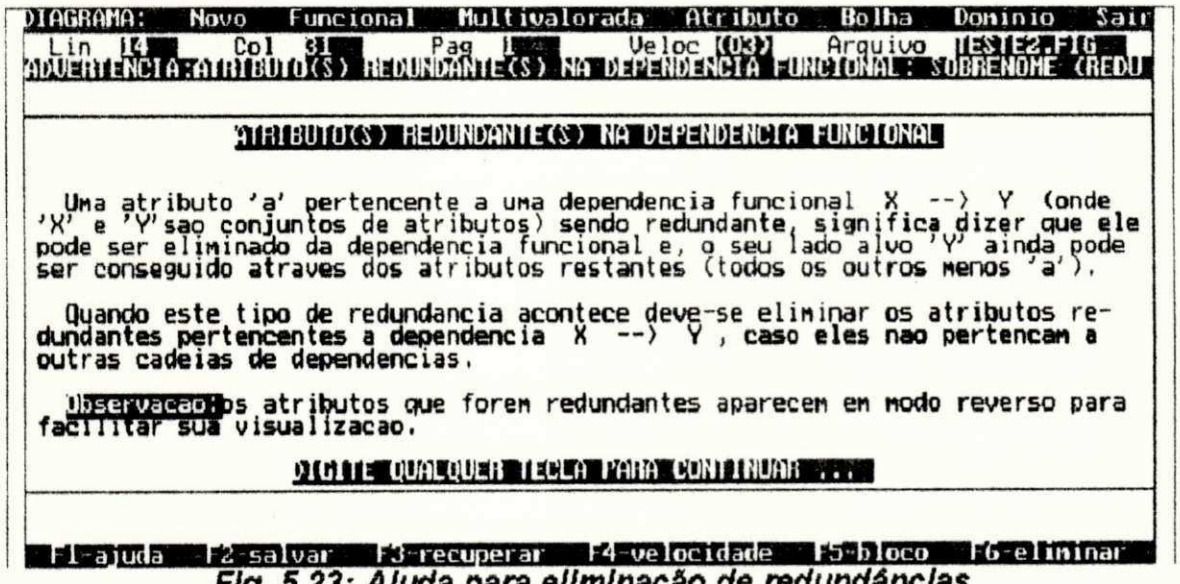


Fig. 5.23: Ajuda para eliminação de redundâncias

Após a eliminação de uma redundância o usuário deve voltar ao menu de controle, para recalcular uma nova cobertura mínima (opção "Redundâncias do Diagrama"). Isto é

necessário porque a eliminação de uma redundância pode eliminar automaticamente outra(s). Feito isto, o processo de eliminação explicado anteriormente, deve ser repetido até que todas as redundâncias tenham sido eliminadas. Isto é comprovado, pela ativação novamente da opção "Redundâncias do diagrama", fornecendo uma tela de controle contendo na área de resultados numéricos, a indicação de ausência de redundâncias.

Todas as orientações fornecidas nesta seção, valem também para a opção "Corrigir redundâncias" do menu da tela de controle. Porém, esta opção apenas corrige as redundâncias encontradas no arquivo em disco "*.COB", fruto da última cobertura mínima calculada. Sem invocar um novo cálculo de uma cobertura mínima.

5.2.5.3.2 - Esquema Relacional Resultante

Neste nível o usuário tem acesso ao gerador de esquemas relacionais, correspondente ao diagrama de dependências atualmente editado. Este, se baseia nos conceitos explicados no capítulo IV (Gerando Esquemas Relacionais Normalizados), para mapear um diagrama de dependências para o Modelo Relacional e determinar as ligações entre as relações geradas.

Exemplo Prático do Gerador de Esquemas Relacionais

Ilustraremos este serviço oferecido pelo **NORMALIZADOR**, formando o esquema relacional e identificando as junções naturais, correspondentes ao diagrama de dependências de um banco de dados para o controle acadêmico de uma universidade (figura 5.15), que foi construído na seção 5.2.5.2. O esquema relacional gerado, mostrado adiante, equivale ao exemplo de mapeamento relacional explicado no capítulo IV (seção 4.4).

Uma vez carregado o diagrama "TESTE1.FIG", o usuário deve verificar a existência de redundâncias neste diagrama, selecionando a opção "Redundâncias do diagrama" do menu da tela de controle. Ele deve proceder segundo as orientações fornecidas na seção 5.2.5.3.1. A figura 5.24 mostra a inexistência de redundâncias no diagrama "TESTE1.FIG":

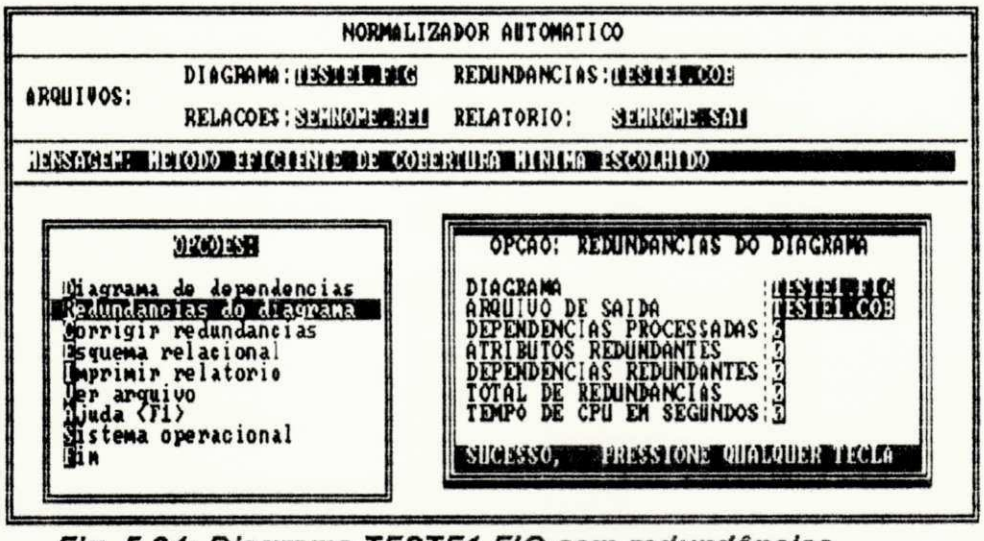


Fig. 5.24: Diagrama TESTE1.FIG sem redundâncias

Prosseguindo o usuário deve selecionar a opção "Esquema relacional", do menu da tela de controle. O que faz aparecer a seguinte área de resultados numéricos, da referida opção:

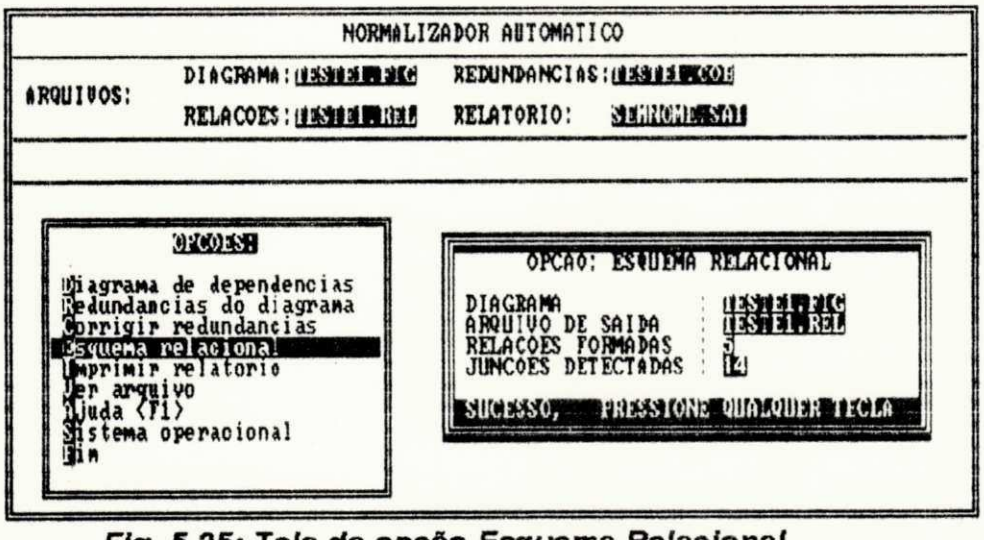


Fig. 5.25: Tela da opção Esquema Relacional

Em seguida é exposta a tela de apresentação, fruto da opção "Esquema relacional". Ela contém na sua área de apresentação: atributos componentes de cada relação; atributos componentes da chave de cada relação; junções naturais entre as relações geradas. Já na área de interação desta tela, são fornecidas as informações de orientação ao usuário. As figuras exibidas a seguir, mostram uma visualização da área de resultados desta tela. A qual é conseguida movimentando-se a linha desta área (conforme as dicas do tópico "Área de Apresentação", da seção 5.2.2). Foram mostradas apenas uma parte das junções identificadas, pois seriam necessárias muitas figuras para representar as quatorze junções detectadas.

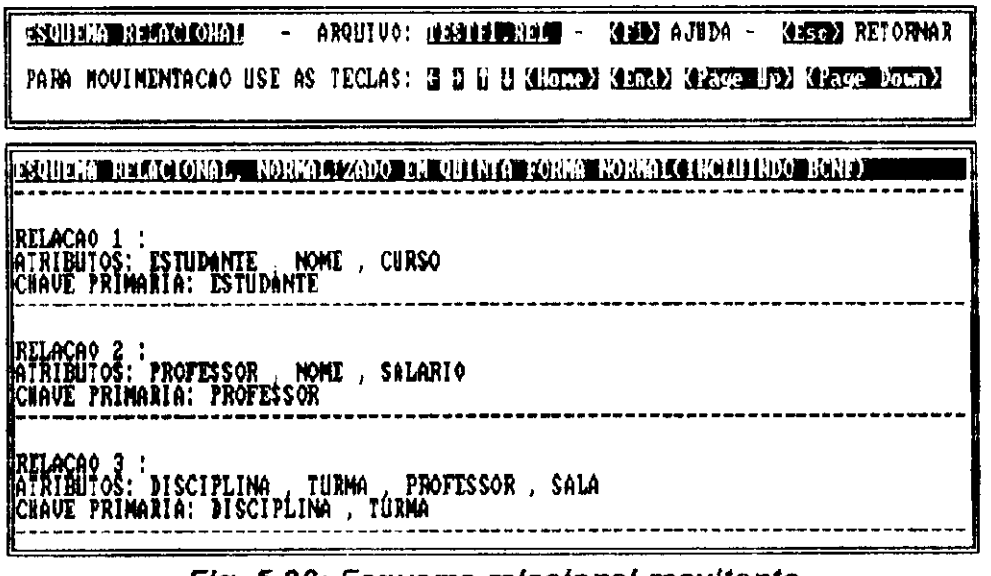


Fig. 5.26: Esquema relacional resultante

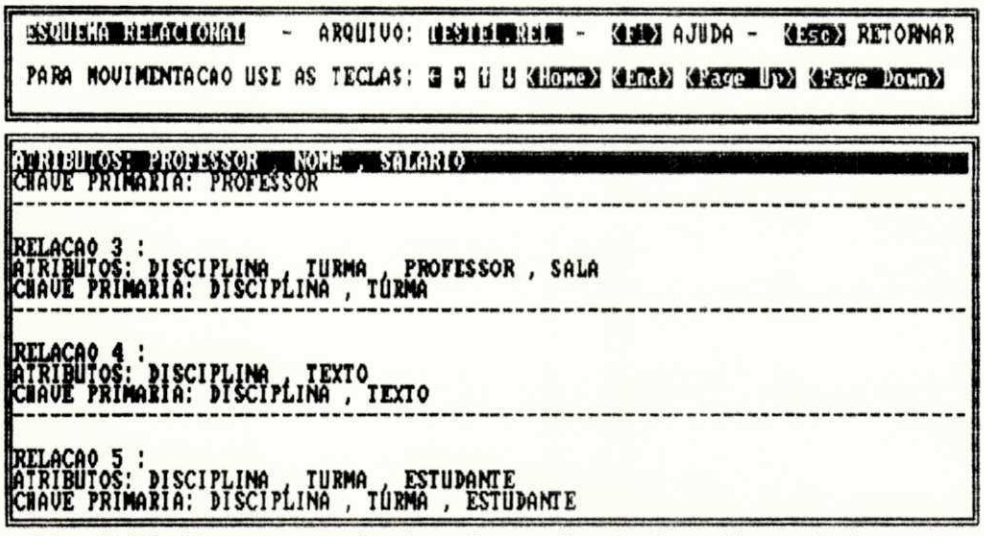


Fig. 5.27: Esquema relacional resultante (continuação)

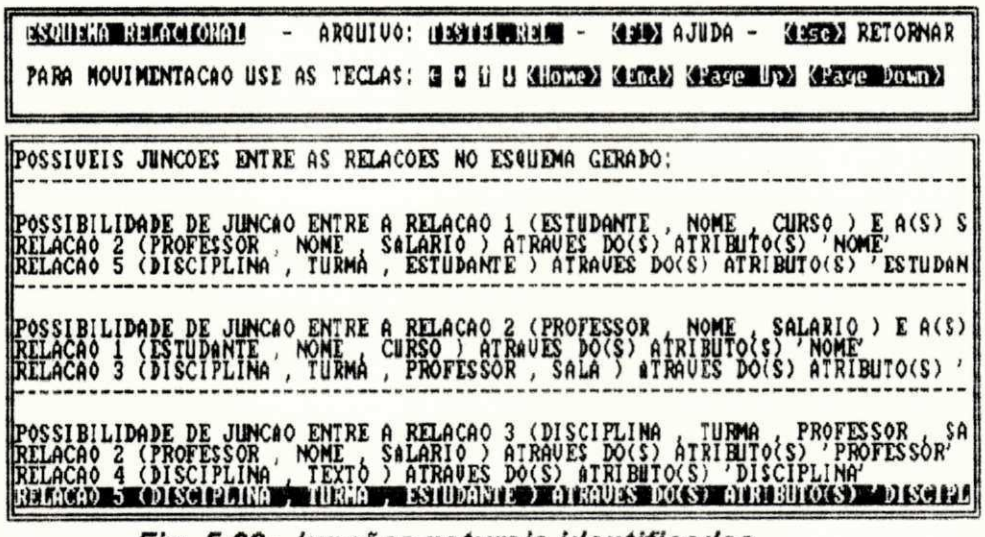


Fig. 5.28: Junções naturais identificadas

Fig. 5.29: Junções naturais identificadas (continuação)

As informações mostradas nas figuras 5.26, 5.27, 5.28 e 5.29, podem ser enviadas para a impressora em forma de relatório. Isto é conseguido, selecionando a opção "Imprimir relatório" do menu da tela de controle. Anexadas a estas informações, são fornecidas uma lista das dependências contidas no diagrama de dependências correspondente. Este relatório, mostra um resumo completo do projeto de banco de dados para uma aplicação. Caracterizando-se como um documento de orientação importante para o usuário.

No próximo capítulo será mostrada a estrutura de dados, utilizada para dar suporte a implementação de **NORMALIZADOR**.

6 - Estrutura de Dados

O projeto das estruturas de dados de **NORMALIZADOR** é composto de três blocos básicos: Estrutura de Dados para Edição Gráfica, Estrutura de Dados para Cobertura Mínima e Estrutura de Dados para o Mapeamento Relacional. Estes blocos dão suporte aos módulos componentes da arquitetura do sistema, explicados no capítulo V.

A figura 6.1, mostra os blocos básicos da estrutura de dados de **NORMALIZADOR**, com seus módulos componentes mais importantes. Em seguida é dada uma descrição de cada um destes blocos.

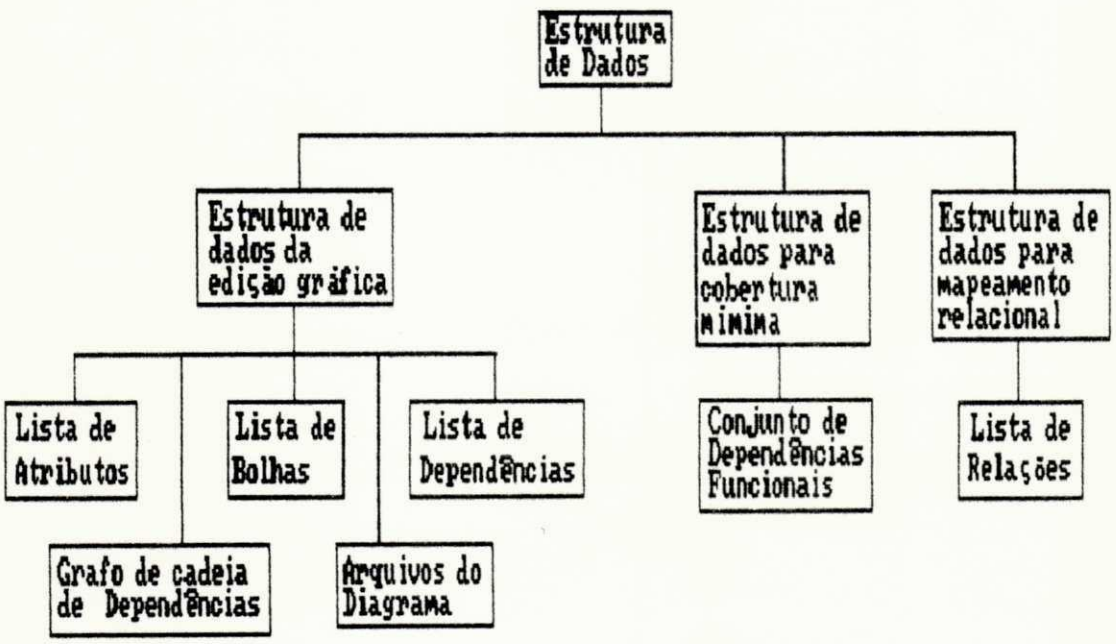


Fig. 6.1: Módulos da Estrutura de Dados

6.1 - Estrutura de Dados da Edição Gráfica

Os módulos componentes deste bloco de estrutura de dados, suportam a implementação do modelo de dados orientado à normalização, explicado no capítulo II. Este bloco é composto dos seguintes módulos: Lista de Atributos, Lista de Bolhas, Lista de Dependências, Grafo de Cadeias de Dependências e Arquivos do Diagrama. Estes módulos são a base de todas as estruturas de dados utilizadas por **NORMALIZADOR**. Em função do conteúdo deles, são montadas as estruturas de dados pertencentes aos outros blocos de estruturas de dados, explicados adiante. A seguir será mostrada a descrição de cada um dos módulos componentes deste bloco.

6.1.1 - Lista de Atributos

A estrutura de dados escolhida para armazenar os atributos de um diagrama de dependências, foi a de uma Lista Encadeada [Horowitz 87]. Escolheu-se uma estrutura de dados dinâmica, pois não se tem um controle do número de atributos definidos. A figura 6.2 mostra a Lista de Atributos.

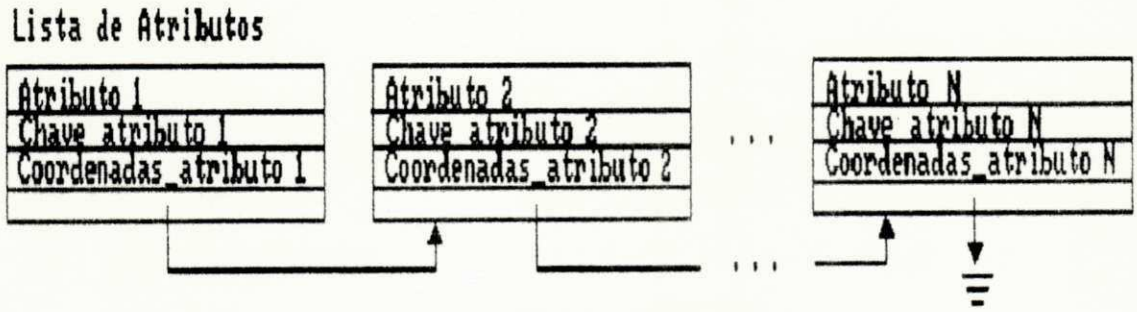


Fig. 6.2: Lista de Atributos

O campo Atributo serve para armazenar o nome de um atributo. Já o campo Chave_atributo é um número inteiro que identifica univocamente um atributo. Enquanto que o campo Coordenadas_atributo representa a posição (X,Y) na tela de

edição gráfica. Atributos repetidos na tela são armazenados na Lista de Atributos com coordenadas e chaves distintas. Como exemplo, observe o diagrama de dependências apresentado na figura 6.3. Em seguida atente para a figura 6.4, que mostra como a Lista de Atributos estaria preenchida, para representar os atributos definidos no diagrama da figura 6.3.

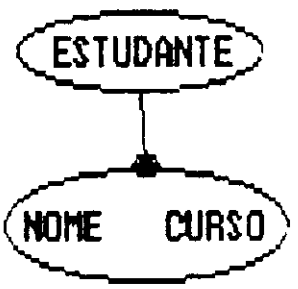


Fig. 6.3: Diagrama de Dependências - exemplo 1

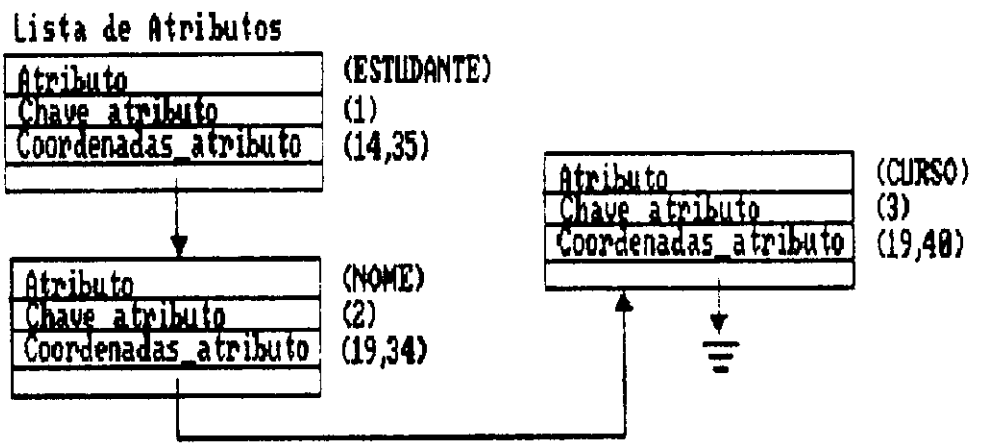


Fig. 6.4: Lista de Atributos preenchida

6.1.2 – Lista de Bolhas

As bolhas editadas em um diagrama de dependências, são armazenadas em uma Lista Encadeada, uma vez que o número de bolhas de um diagrama não é previamente determinado. A figura 6.5 mostra a Lista de Bolhas.

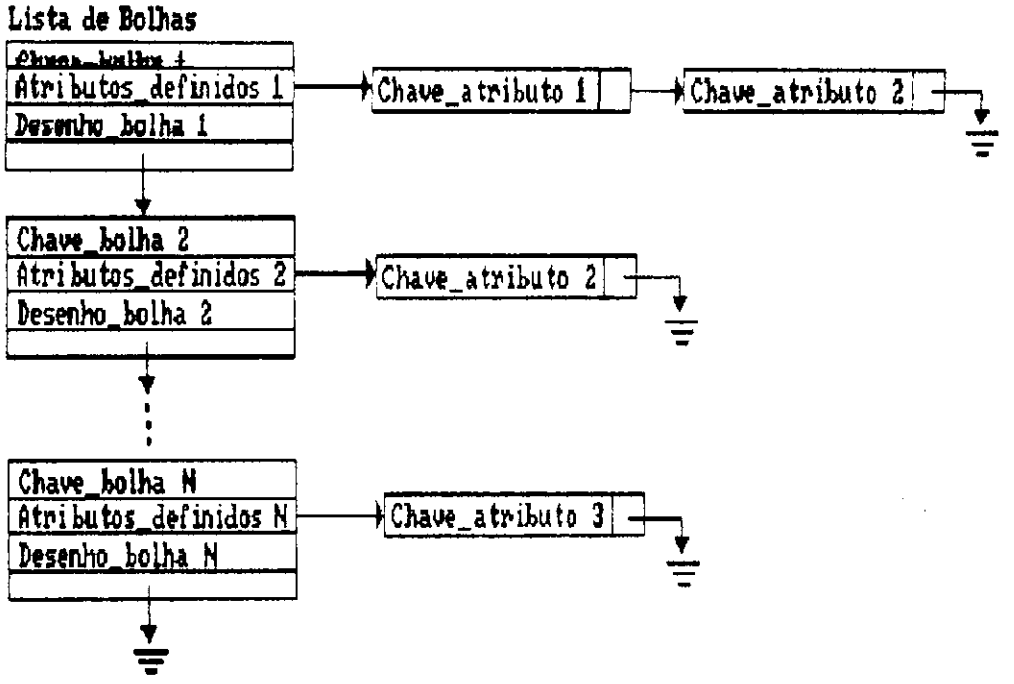


Fig. 6.5: Lista de Bolhas

O campo Chave_bolha é um número inteiro que identifica univocamente uma bolha. Já o campo Desenho_bolha, são os pontos gráficos que definem uma elipse no plano (coordenadas do centro e sub-eixos da elipse). O campo Atributos_definidos representa os atributos contidos em uma bolha. Trata-se de um apontador para uma Lista Encadeada, contendo a chave de identificação de cada um destes atributos. Observe que um mesmo atributo pode aparecer em mais de uma bolha. A figura 6.6 mostra a lista de bolhas preenchida para o diagrama de dependências da figura 6.3.

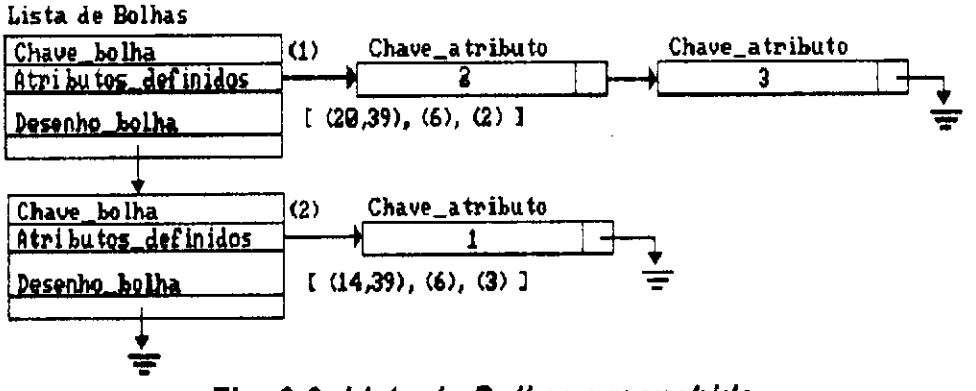


Fig. 6.6: Lista de Bolhas preenchida

6.1.3 – Lista de Dependências

As dependências funcionais e multivaloradas de um diagrama de dependências, são armazenadas em uma Lista Encadeada. Já que não sabemos a princípio, do número necessário de dependências que serão construídas. A figura 6.7 mostra a Lista de Dependências.

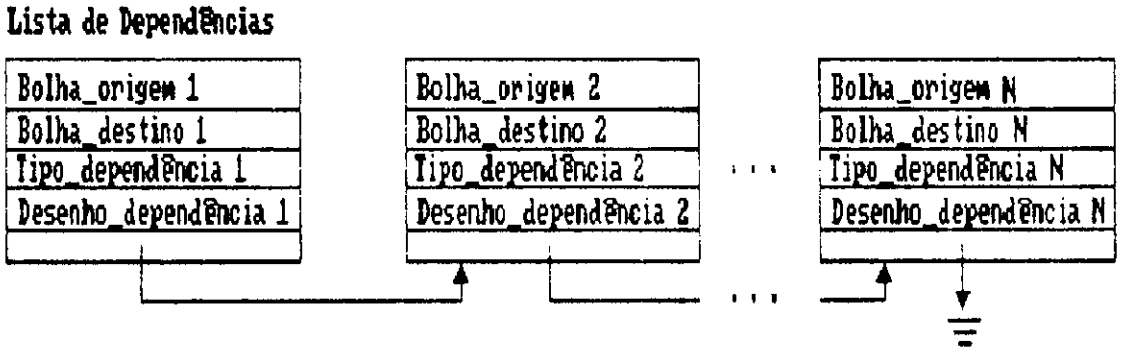


Fig. 6.7: Lista de Dependências

Os campos Bolha_origem e Bolha_destino representam as chaves de identificação das bolhas, que contêm os atributos do lado direito e esquerdo respectivamente, de uma dependência. O campo Tipo_dependência indica qual dependência esta armazenada (funcional ou multivalorada). Já no campo Desenho da dependência, são armazenados os pontos gráficos da reta e da(s) bolinha(s), que representa graficamente uma dependência. A figura 6.8 mostra a Lista de Dependências, caso fosse inicializada contendo a dependência funcional do diagrama de dependências da figura 6.3.

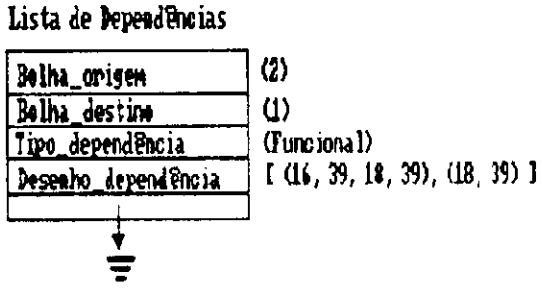


Fig. 6.8: Lista de Dependências preenchida

6.1.4 – Grafo de Cadeias de Dependências

A estrutura de dados escolhida para representar a ligação das dependências (armazenadas na Lista de Dependências), formando as cadeias de dependências: funcionais, multivaloradas e mistas, foi a de um Grafo Orientado [Horowitz 87]. Pois, a combinação destas cadeias (respeitando a regra de formação de cadeias, explicada no capítulo II - seção 2.4) naturalmente forma um grafo, onde os nós são as bolhas de um diagrama de dependências, enquanto que as setas são as dependências (funcionais ou multivaloradas) entre estes nós. A utilidade deste grafo é facilitar a formação a partir de cada caminho (contendo uma cadeia de dependências consistente com a regra de formação de cadeias), das estruturas de dados Conjunto de Dependências Funcionais e Lista de Relações, explicadas mais adiante.

O grafo de cadeias de dependências é acíclico, porque se assim não o fosse durante o mapeamento relacional, obteríamos relações contendo atributos repetidos. Esta restrição está prevista nos diagramas de dependências de Smith [Smith 85]. Porém, o uso de bolhas duplas proporciona a formação de cadeias de dependências cíclicas.

A figura 6.9 mostra o Grafo de Cadeias de Dependências, onde cada caminho é uma Lista Duplamente Encadeada [Horowitz 87], pois uma cadeia de dependências possui uma formação linear. A escolha deste tipo de estrutura de dados, motivou-se para facilitar o controle da regra de formação de cadeias de dependências. Foram escolhidas estruturas de dados dinâmicas, por não se ter um controle prévio da quantidade de cadeias formadas, durante a construção de um diagrama de dependências.

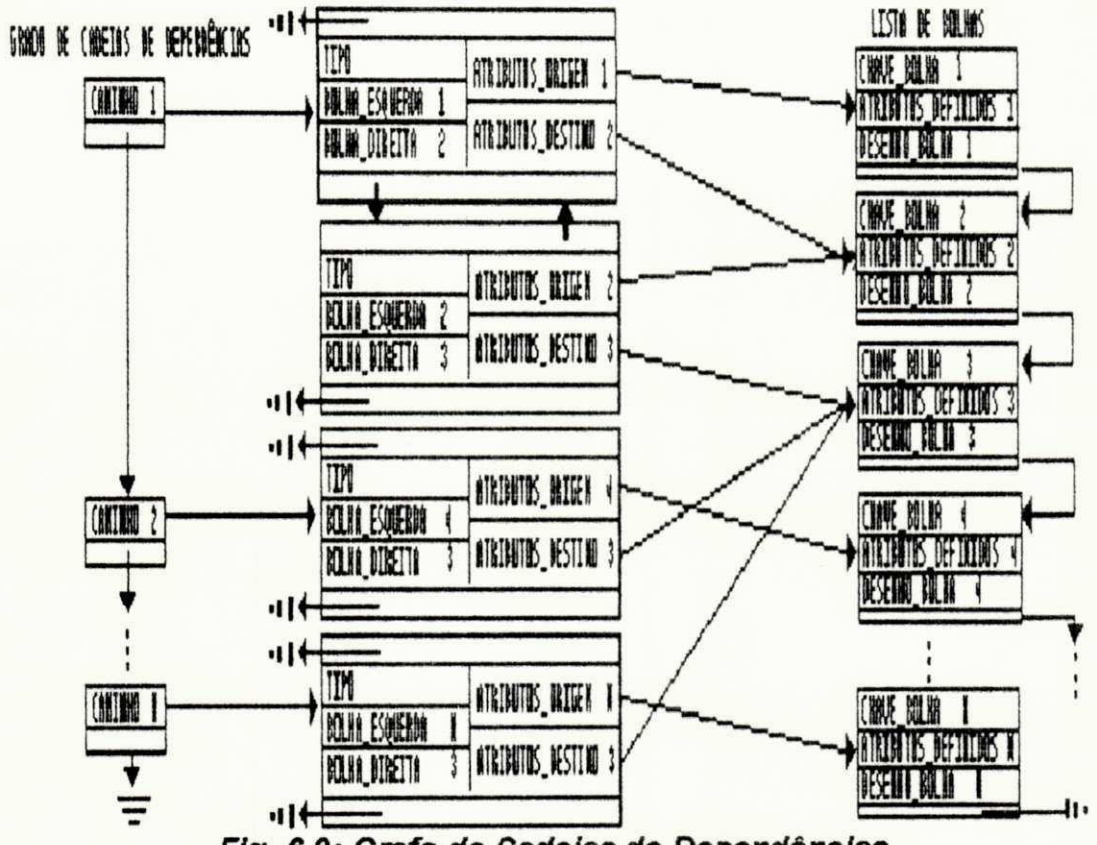


Fig. 6.9: Grafo de Cadeias de Dependências

O campo Caminho guarda uma cadeia de dependências do grafo. Uma cadeia é formada por dependências, que podem ser funcionais ou multivaloradas (indicado pelo campo tipo). As chaves de identificação das bolhas que compõem uma dependência (bolha origem e bolha destino), são armazenadas respectivamente nos campos Bolha_esquerda e Bolha_direita. Os campos Atributos_origem e Atributos_destino apontam para o campo Atributos_definidos da Lista de Bolhas. Estes campos representam respectivamente os lados esquerdo e direito de uma dependência. Eles são o ponto de partida para gerar as estruturas de dados, utilizadas no cálculo de uma cobertura mínima.

Como exemplo, observe o diagrama de dependências apresentado na figura 6.10. O preenchimento do Grafo de Cadeias de Dependências contidas neste diagrama, é mostrado

na figura 6.11, juntamente com a interligação com as estruturas de dados Lista de Atributos e Lista de Bolhas.

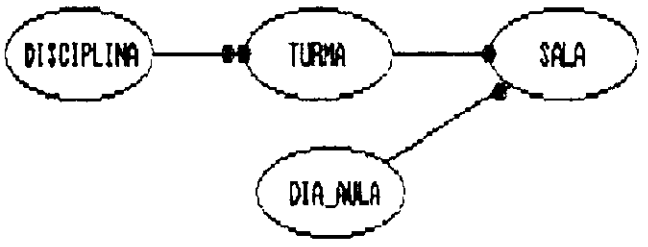


Fig. 6.10: Diagrama de Dependências - exemplo 2

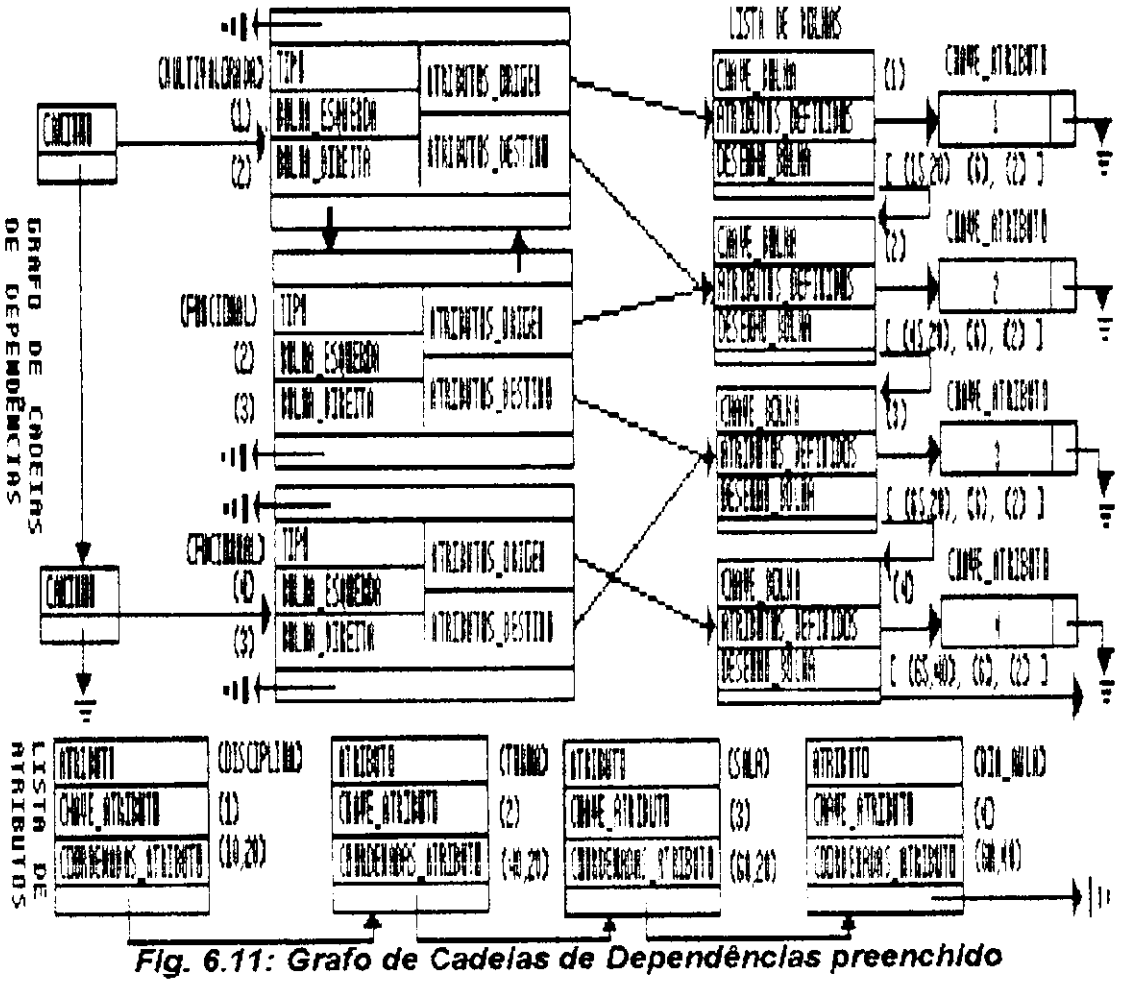


Fig. 6.11: Grafo de Cadeias de Dependências preenchido

6.1.5 - Arquivos do Diagrama

Objetivando conseguir o mínimo espaço em disco, durante a gravação de um diagrama de dependências editado, **NORMALIZADOR** salva em disco, apenas as estruturas de dados básicas: Lista de Atributos, Lista de Bolhas e Lista de Dependências. Os arquivos contendo o conteúdo destas estruturas de dados, é composto pelas informações que as definem (exceto aquelas que são apontadores de memória), juntamente com os pontos gráficos que são necessários para desenhar cada uma delas. Quando for necessário recuperar um diagrama de dependências previamente gravado, as informações nestes arquivos são suficientes para montar todas as estruturas de dados utilizadas por **NORMALIZADOR**.

6.2 - Estrutura de Dados para Cobertura Mínima

Este bloco de estrutura de dados é composto de um módulo básico, que dá suporte a implementação dos algoritmos de cobertura mínima apresentados no capítulo III, seções 3.3 e 3.4. Esses algoritmos são a base de implementação dos módulos Detector de Redundâncias e Correção de Redundâncias, pertencentes a arquitetura do sistema, explicada no capitulo V.

6.2.1 - Conjunto de Dependências Funcionais

Para calcular uma cobertura mínima de um diagrama de dependências, é necessário formar um Conjunto de Dependências Funcionais, transformando as cadeias de dependências multivaloradas de uma cadeia mista em pseudo-atributos, um para cada cadeia.

Uma outra transformação em um diagrama de dependências, é a utilização do axioma de inferência da projeção, definido por Armstrong. Isto faz com que cada dependência funcional

do conjunto seja reduzida à direita. Estas transformações foram detalhadas no capítulo III, seção 3.2.

Portanto, um Conjunto de Dependências Funcionais é montado a partir das estruturas de dados Grafo de Cadeias de Dependências e Lista de Atributos, aplicando-se as transformações descritas anteriormente.

A estrutura de dados escolhida para armazenar um Conjunto de Dependências Funcionais, foi a de uma Lista Encadeada. Pois, não se tem uma idéia prévia do número de dependências funcionais pertencentes a este conjunto. A figura 6.12 mostra o Conjunto de Dependências Funcionais.

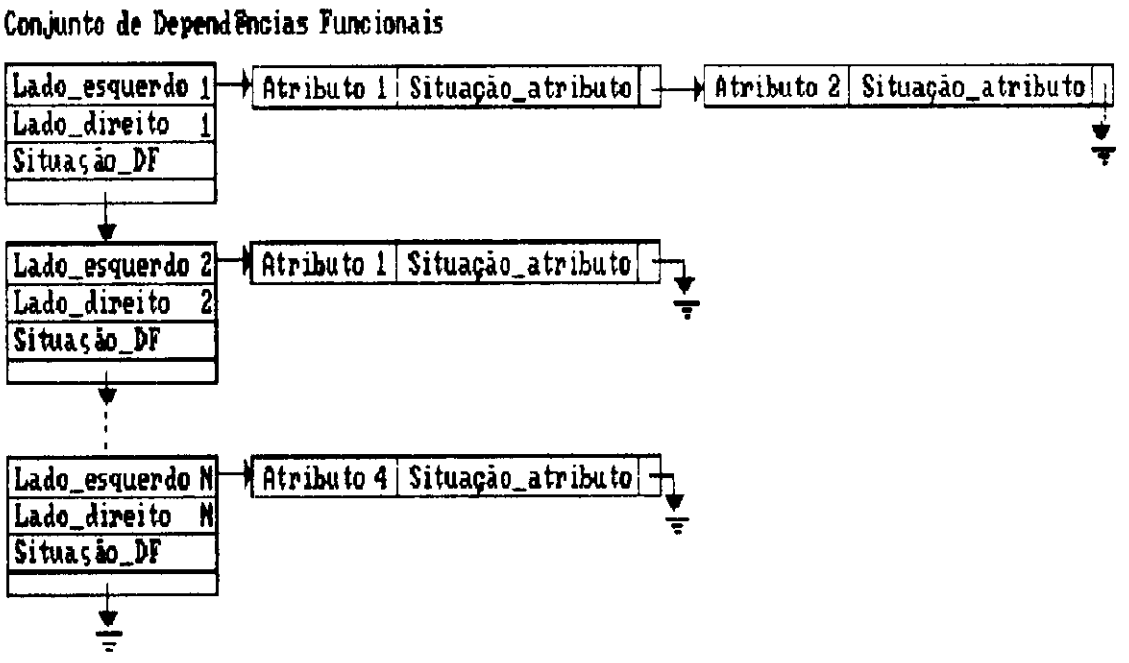


Fig. 6.12: Conjunto de Dependências Funcionais

O campo Lado_esquerdo aponta para uma Lista Encadeada de Atributos, pertencentes ao lado esquerdo de uma dependência funcional. Esta lista contem o nome de cada atributo, armazenado no campo Atributo, juntamente com sua situação (redundante ou não) armazenada no campo Situação_atributo. Já o campo Lado_direito, guarda o nome de

um único atributo, pertencente ao lado direito de uma dependência funcional reduzida à direita. O campo Situação_DF determina se uma dependência funcional é redundante ou não. A figura 6.13 mostra como o Conjunto de Dependências Funcionais, seria preenchido para o diagrama de dependências da figura 6.10.

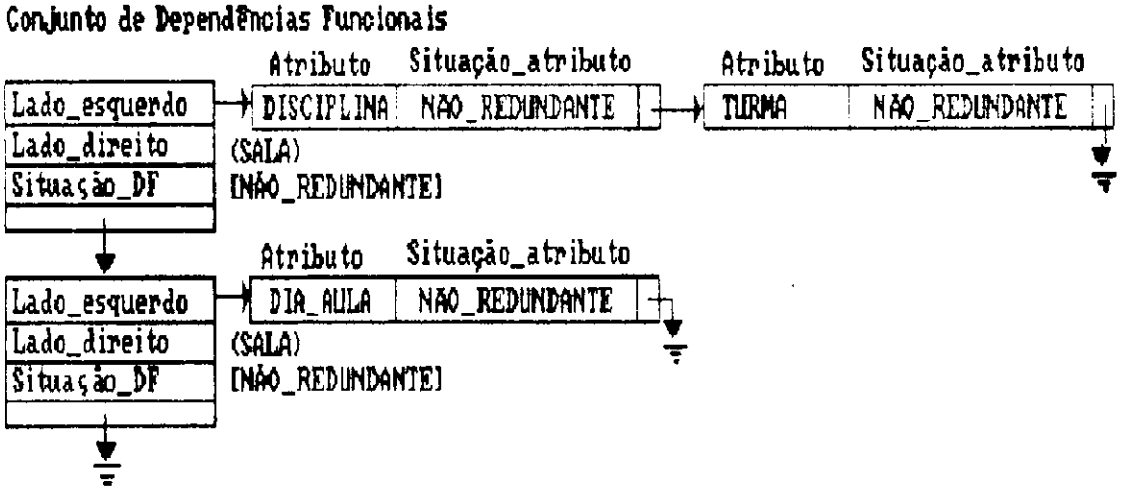


Fig. 6.13: Conjunto de Dependências Funcionais preenchido

6.3 - Estrutura de Dados para o Mapeamento Relacional

Este bloco de estrutura de dados é composto de um módulo base, que dá suporte a implementação do mapeamento relacional e determinação das ligações entre as relações geradas. Isto é feito a partir de um diagrama de dependências sem redundâncias, conforme os conceitos explicados no capítulo IV seções 4.1 e 4.3. Esses conceitos, são a base de implementação do módulo Gerador de Esquemas Relacionais, pertencente arquitetura de **NORMALIZADOR** explicada no capítulo V.

6.3.1 - Lista de Relações

Uma Lista de Relações é formada a partir das cadeias de dependências (funcionais, multivaloradas e mistas) e bolhas isoladas (ver capítulo II, seção 2.4). As redundâncias encontradas no cálculo de cobertura mínima, são desconsideradas durante a formação de relações, ou seja, as dependências funcionais e os atributos detectados como redundantes não geram relações. Portanto, uma Lista de Relações é formada a partir das estruturas de dados: Lista de Atributos, Lista de Bolhas, Grafo de Cadeias de Dependências e Conjunto de Dependências Funcionais.

A estrutura de dados escolhida para representar as relações, pertencentes a um esquema relacional resultante, foi uma Lista Encadeada, pois não se sabe a princípio quantas relações serão geradas. A figura 6.14 mostra a Lista de Relações.

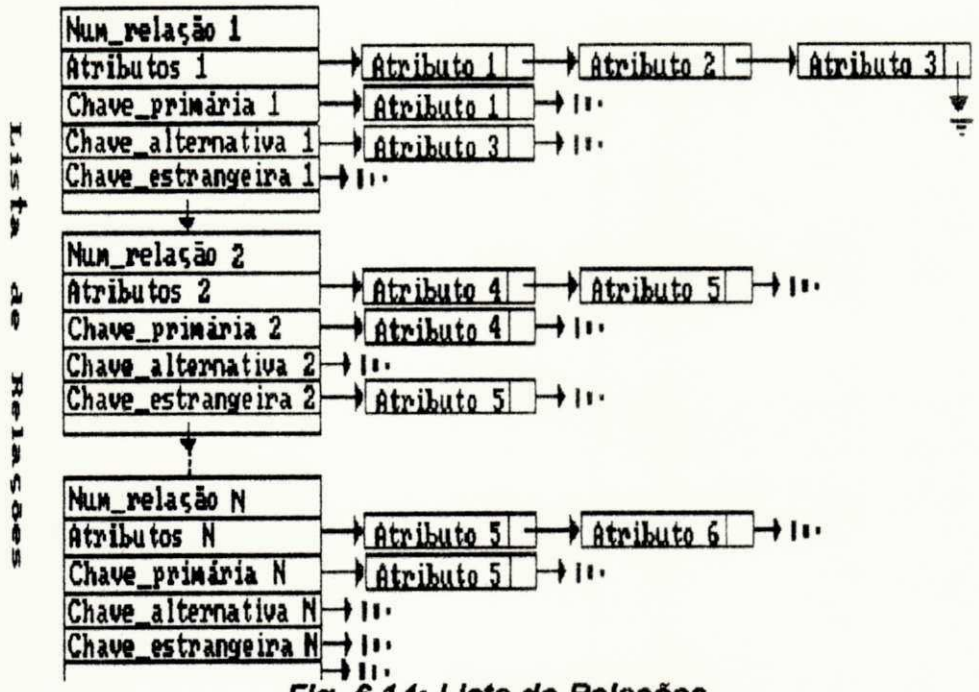


Fig. 6.14: Lista de Relações

O campo num_relação identifica univocamente uma relação. Os atributos componentes de uma relação, são armazenados em uma lista encadeada apontada pelo campo Atributos. Similarmente, os atributos pertencentes a chave primária e estrangeira são armazenadas nas Listas Encadeadas apontadas respectivamente pelos campos Chave_primária e Chave_estrangeira. O campo Chave_alternativa é uma lista encadeada contendo as possíveis chaves alternativas de uma relação. A figura 6.15 mostra a Lista de Relações formada a partir do diagrama de dependências, mostrado na figura 6.10.

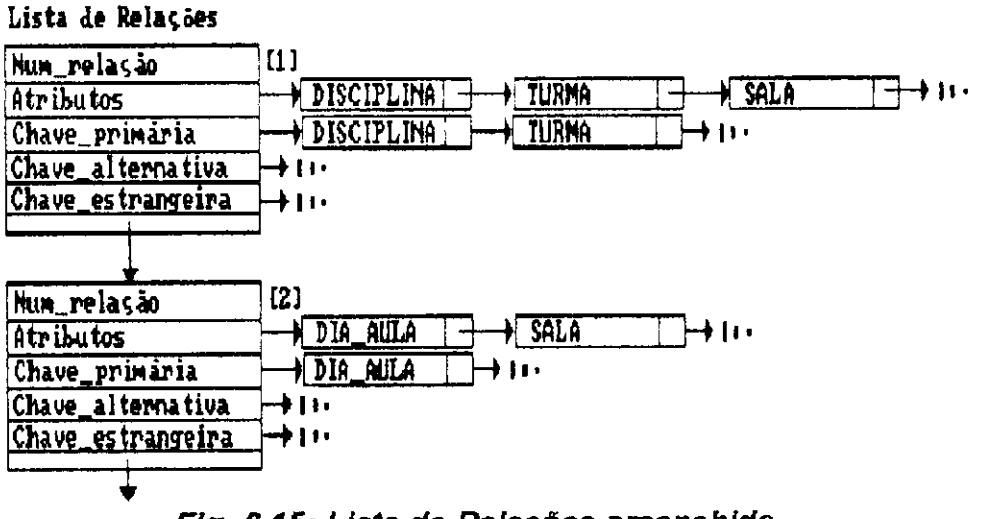


Fig. 6.15: Lista de Relações preenchida

Conclusão

Nossa ferramenta, **NORMALIZADOR**, atende aos objetivos de facilitar o trabalho dos projetistas de banco de dados relacional, livrando-os da tarefa de normalizar relações. Além disso tenta eliminar falhas de outros métodos similares.

A estratégia adotada foi a de utilizar um modelo de dados orientado ao processo de normalização aplicando-se o cálculo de cobertura mínima à um diagrama de dependências. Esta estratégia permite identificar as possíveis redundâncias representadas em um esquema semântico, possibilitando uma maior confiança e independência dos dados de um projeto de banco de dados.

O modelo de dados escolhido eliminou os problemas de mistura semântica e perda de informação do mundo real, contidos nas outras propostas pesquisadas. Ele oferece uma semântica bastante simples, o que facilita sua utilização por parte do usuário. Em contrapartida, o modelo é ele próprio limitado, não podendo ser comparado com modelos semanticamente mais ricos, tais como o Modelo Entidade-Relacionamento ou o Modelo Temporal Hierárquico (THM) [Schiel 83], que possuem diversas abstrações hierárquicas para modelar o mundo real.

NORMALIZADOR oferece ao usuário a vantagem de escolher entre dois métodos para o cálculo de uma cobertura mínima. O primeiro implementa os algoritmos de Bernstein. O segundo é um aperfeiçoamento do primeiro método, com uma melhora de desempenho bastante significativa. As redundâncias encontradas são eliminadas através de um processo interativo. Elas são exibidas e explicações são fornecidas pela ferramenta, orientando o usuário para eliminá-las.

Os esquemas relacionais normalizados obtidos de um diagrama de dependências, esquematizando uma aplicação, são mínimos tanto em relação a quantidade de relações quanto a de atributos pertencentes a cada uma delas. Além disto, para cada relação deste esquema é fornecido o projeto de chaves (primária, alternativa e estrangeira) juntamente com uma descrição das possíveis junções naturais, que podem ser feitas entre as relações geradas (facilitando a identificação das consultas inter-relacionais).

Uma outra vantagem oferecida por **NORMALIZADOR** é que um usuário não precisa ser necessariamente um especialista em banco de dados. Sua interface é bastante amigável, tornando transparente a teoria da normalização (conceito de formas normais), de cobertura mínima, de projeto de chaves, etc.

Trabalhos Futuros

Uma proposta interessante seria a implementação de um dicionário de dados automático, trabalhando em conjunto com o editor de diagramas de dependências. Este documentaria cada atributo representado em um diagrama de dependências, seu tipo de dados associado, indicador de domínio comum (ver capítulo II, seção 2.4), etc. Isto ajudaria na modelagem de banco de dados corporativos. As informações contidas neste dicionário proporcionariam a geração de esquemas relacionais, onde cada relação seria documentada de forma consistente com o dicionário.

A simplificação de um diagrama de dependências (explicada no capítulo II, figura 2.4), pode levar a perda de dependência funcional, durante a fase de geração de relações. Por exemplo, considere as seguintes dependências:

A --> B; A --> C; C --> A

Elas são simplificadas da seguinte maneira:

A --> B,C; C --> A

Esta simplificação faz com que sejam geradas as seguintes relações R1 e R2, com as respectivas chaves primárias "A" e "C".

R1(A, B, C)

R2(C, A)

Na relação R1 a dependência C --> A foi perdida. Para estes casos seria interessante, em trabalhos futuros, evitar esta simplificação.

Uma proposta interessante para futuras versões de **NORMALIZADOR**, seria a implementação automática da modelagem alternativa para cadeias de dependências mistas, que desrespeitarem a regra de formação das mesmas (ver capítulo II, seção 2.4).

A quantidade de atributos componentes de uma chave primária de uma relação, pode vir a prejudicar o desempenho de aplicativos que a manipulem. Se uma relação possui mais de três atributos em sua chave primária, o tempo de resposta de uma consulta pode se tornar longo. Uma proposta interessante seria o uso de chaves substitutas ("surrogate keys") quando necessário, reduzindo o número de campos de uma chave primária [Smith 85] [Date 83].

A geração de esquemas relacionais normalizados em 5ª forma normal, em certos casos pode provocar a queda de desempenho durante sua manipulação. Por exemplo, quando muitas operações relacionais (junções e projeções) precisam ser realizadas (pelo gerenciador de banco de dados) durante a realização de uma consulta, conseqüentemente o tempo de resposta é bastante alto. Uma proposta para trabalhos futuros, seria a predeterminação da forma normal ideal para

o esquema relacional resultante. Isto diminuiria o número de relações presentes no esquema resultante, diminuindo conseqüentemente o número de junções possíveis.

Uma outra proposta interessante é o cálculo de uma cobertura mínima abrangendo as dependências multivaloradas, pois as direções de pesquisa nesta área, tentam alcançar este objetivo [Diederich-Milton 88].

Quando escolhe-se o método tradicional para calcular uma cobertura mínima do caso patológico, apresentado no capítulo III (seção 3.5), o desempenho de **NORMALIZADOR** é bastante superior do que o mesmo cálculo utilizando o outro método de cobertura mínima. Sugerimos que em versões futuras este caso patológico seja identificado, para que o método adequado venha a ser utilizado automaticamente.

Outra proposta seria a migração deste trabalho para outros ambientes, pois este foi implementado na linguagem de programação C, o que poderia ser um bom começo. Também seria interessante acrescentar à ferramenta o recurso de "Mouse", o que proporcionaria maior facilidade em sua manipulação. Uma outra melhoria seria utilizar a memória estendida dos computadores, para que possa manipular diagramas de dependências maiores.

Bibliografia

[Beeri 78] C. Beeri, P.A. Bernstein e N. Goodman. "A Sophisticate's Introduction to Database Normalization Theory", Proc. 4th VLDB, Berlim, (1978) 113-124.

[Bernstein 76] Bernstein, P. A. "Synthesizing third normal form relations from funcional dependencies", ACM Transactions on Database Systems 1, 4. Dez. 1976, pp. 277-298.

[Casanova 82] M. Casanova, R. Fagin e C. Papadimitriou. "Inclusion Dependencies and their Interaction with Functional Dependencies", Proc. ACM SIGACT-SIGMOD Symposion on Principles of Database Systems, (1982) 171-176.

[Chen 76] Chen , P. P. -S. "The Entity-Relationship Model - Toward a Unified View of Data", ACM Transactions on Database Systems 1, 1. Mar. 1976, pp. 9-36.

[Codd 70] Codd, E.F. "A relational model of data for large shared data banks", Comm ACM 13, 6, Jun. 1970, pp. 377-387.

[Codd 72] Codd, E.F. "Further normalization of the data base relational model" in Rustin, R. (Ed.) Courant Computer Science Symposia 6, "Data Base Systems", Prentice-Hall, Englewood Cliffs 1972, pp. 33-64.

[Codd 74] Codd, E.F. "Recent Investigations into Relacional Data Base Systems", Proc. IFIP Congress 1974.

[Date 83] Date, C.J. "Relational Databases Seminar Workbook", Digital Consulting Associates. New York. 1983.

[Date 86] Date, C.J. "Introdução a Sistemas de Banco de Dados", Editora Campus, 1986.

[Date 90] Date, C.J. & Colin J. White. "A guide do DB2", Third Edition. Addison Wesley, 1990.

[Diederich-Milton 88] Diederich, J. , e Milton, J. , "New Methods and fast algoritims for database normalization", ACM Transactions on Database Systems 13, 3, Set. 1988, pp. 339-365.

[Fagin 77] Fagin, R. "Multivalue dependencies and a new normal form for relational databases", ACM Transactions on Database Systems 2, 3, Set. 1977, pp. 262-278.

[Fagin 77-a] Fagin, R. "The decomposition versus synthetic approach to relational databases", Proc. 3rd Int. Conference on Very Large Data Bases, 1977, pp. 441-446.

[Fagin 79] Fagin, R. "Normal forms and relational database operators", Proc. 1979 ACM SIGMOD Intl. Conf. on Management of Data, pp. 153-160.

[Farias 92] Farias, Jorge A.L., e Sampaio, M.C. "Normalizador: Uma Ferramenta para Geração de Esquemas Relacionais Totalmente Normalizados", 4ª Semana de Informática da UFBA, Salvador - BA, pp. 59 - 68, Abril 1992.

[Maier 83] Maier, D. "The Theory of Relational Databases", Computer Science Press, Rockville, Md., 1983.

[Martin 77] Martin, J. "Computer Data-Base Organization", (2nd. Ed.), Prentice-Hall, Englewood Cliffs 1977.

[Oracle 84] "Manuais de Referência", Oracle Corporation, Menlo Park, Calif., 1984.

[Raver-Hubbard 77] Raver, N. e Hubbard, G.U. "Automated logical data base design: concepts and applications", IBM System Journal 1977, 3, pp. 287-312.

[Rissanen 77] J. Rissanen, "Independent Components of Relations", ACM Transactions on Database Systems 2, 4, Dez. 1977.

[Salzberg 86] Salzberg, B. , "Third normal form made easy", Sigmod Record, vol. 15, Nº 4, December 1986, 2-18

[Schiel 83] U. Schiel, "An abstract introduction to the temporal-hierarchic data model (THM)", Proc. of the 9th Conference on Very Large Data Bases" (1983) 322-330.

[Schiel 84] U. Schiel, "Internal Schema Design and how to avoid the BCNF Problem", Report TR-DSC-002/84, Univ. Fed. da Paraíba.

[Setzer 87] Setzer Valdemar W. "Banco de Dados (Conceitos, Modelos, Gerenciadores, Projeto Lógico e Projeto Físico)", 2ª Edição, Editora Edgard Blücher Ltda, 1987.

[Smith 85] Smith, H. C. , "Database design: composing fully normalized tables from a rigorous dependency diagram", Communications of the ACM, August (1985), pp. 826-838.

[Stonebraker 76] Stonebraker, M., Wong, E. , Kreps, P., and Held, G. "The design and implementation of INGRESS", ACM TDS 1, 3 Sept/1976, 189-222.

[Ullman 89] Ullman, J. D. "Principles of database and knowledge - base systems" - Vol. II, Computer Science Press, Rokville, Maryland, 1989.

[Wang-Wedekind 75] Wang, C. P. e Wedekind, H. H. "Segment synthesis in logical data base design", IBM Journal Res. Develop. 19, 1, Jan. 1975, pp. 71-77.