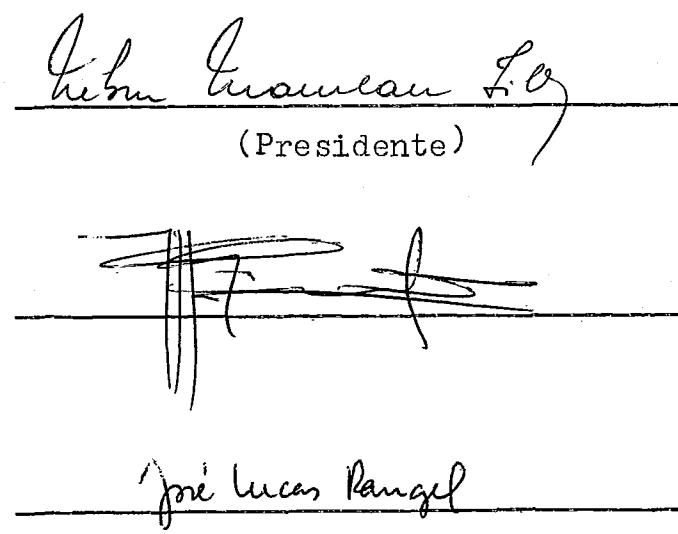
ESTUDO E APLICAÇÃO

DA TABELA DE DECISÃO

Walter Dominguez

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS - GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS (M.Sc.)

Aprovada por:



Rio de Janeiro, RJ - Brasil

DEZEMBRO DE 1977

Dominguez, Walter

Estudo e aplicação da Tabela de decisão. Rio de Janeiro, 1977.

255 p. ɓɓɓ 29,7 cm (COPPE - UFRJ, M.Sc, Sistemas, 1977)

Tese - Univ. Fed. Rio de Janeiro. Coppe

1. Tabela de Decisão. I.COPPE/UFRJ. II.Titulo. III.Série.

AGRADECIMENTOS

Em primeiro lugar agradeço a minha esposa, pelo apoio e sacrifício dedicados a mim, sem os quais, esse trabalho jamais poderia ter sido concluído.

Agradeço ao meu colega Carlos Flores pela valiosa contribuição técnica, a qual teve influência direta na viabilidade da realização desse trabalho.

Ao meu orientador Marcelo Pardo Brown pelo seu trabalho de orientação.

As bibliotecárias Marília Canetti e Sheila Gomes pela ajuda na pesquisa bibliográfica.

Aos meus colegas de trabalho pelas sujestões dadas.

E finalmente a Companhia de Telecomunicações do Rio de Janeiro (TELERJ) pelo apoio e contribuição material para o desenvolvimento do trabalho.

Resumo

Este trabalho apresenta de maneira informal, uma técnica para comunicação e documentação de sistemas em processamento de dados. Esta técnica é chamada de Tabela de Decisão.

Como existe muito pouca informação sobre esta ferramenta e a existente exige um conhecimento a priori, a preocupação do autor foi fazer um apanhado geral da literatura existente dando um enfoque bem prático de como preparar e utilizar as tabelas de decisões.

Serão apresentados também alguns métodos de otimização e conversão de tabelas para programas.

Finalmente será apresentado para estudos futuros a viabilidade de ser criada uma linguagem na forma de Tabela.

ABSTRACT

This work shows in an informal manner, a technic of comunication and documentation of systems in data process. This technic is called 'DECISION TABLES'.

As there is little information about this tool and it requires a prior knowledge, the preocupation of the author was to make a study about the literature that exists, taking a practical approach of how to prepare and to use the decision tables.

Some methods of optimization and conversions of tables into programs will also be presented.

Finally the viability creating a language in the form of tables will be presented to future study.

SUMÁRIO

CAPÍTULO I

OBJETIVO

CAP. I - OBJETIVO

Um dos maiores problemas em processamento de dados é a comunicação de homem para homem e homem para máquina, ou seja, uma vez tendo sido descoberta a solução de um determinado problema, qual a melhor maneira de representar esta solução, de tal forma que esta não seja modificada pela interpretação e obrigar a questionar a solução.

Um outro problema é forçar a documentação desta solução atualizada, pois feitas as alterações depois da implantação, na maioria dos casos, estas não são documentadas, tornando a documentação não confiável.

O objetivo desse trabalho é apresentar uma ferramenta para auxiliar tanto ao analista como ao programador na comunicação bem como na documentação.

Esta técnica é conhecida como 'Tabela de Decisão'.

Serão analizadas as características e o uso das tabelas de decisões em casos práticos, sendo introduzidos alguns melhoramentos, devido a experiência do autor no campo.

Finalmente serão dadas condições para a criação de uma linguagem na forma de tabelas de decisões.

CAPÍTULO II

INTRODUÇÃO

CAP. II - INTRODUÇÃO

POR QUE TABELAS DE DECISÕES?

1º) Seja o seguinte trecho de programa em PL/1:

```
IF (A≠B AND C≠D) OR (A=B AND C≠D)
    THEN CALL X,.
    ELSE CALL Y,.
```

Chamamos de C1 a espressão A=B e C2 a expressão C=D, então temos:

IF ($\overline{C1}$ AND $\overline{C2}$) OR (C1 AND $\overline{C2}$)

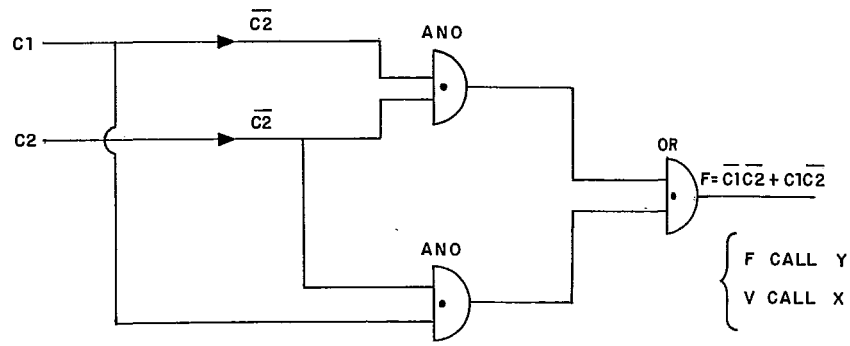
Representando este trecho de programa, por uma função booleana, temos:

$$f = \overline{C1}\ \overline{C2} + C1\ \overline{C2}$$

Representando essa função em uma tabela verdade, teríamos:

C1	C2	$\overline{C1} \wedge \overline{C2}$	$C1 \wedge \overline{C2}$	f	
F	F	V	V	V	CALL X
F	V	F	F	F	CALL Y
V	F	F	V	V	CALL X
V	V	F	F	F	CALL Y

Representando esta função, por circuito lógico, teríamos:



Para simplificar essa funcão, pode-se aplicar várias técnicas.

a) Aplicando álgebra booleana

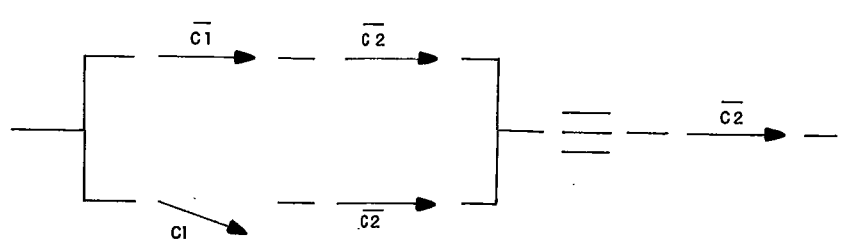
$= \overline{C1}\ \overline{C2} + C1\ \overline{C2}$

$= \overline{C2}\ (\overline{C1} + C1)$ $\qquad\qquad$ $\overline{C1} + C1$ = Teoria Morgan

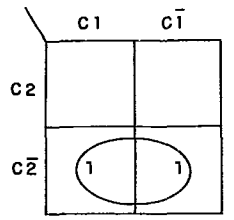
$= \overline{C2}$

$$\overline{C1}\ \overline{C2} + C1\ \overline{C2} = \overline{C2}$$

b) Representando p/ chaves teríamos:

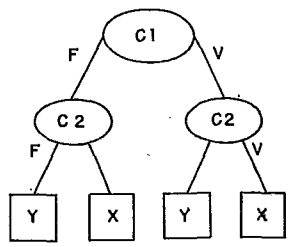


c) Aplicando a técnica de simplificação por MAPA DE KARNAUGH, teríamos:

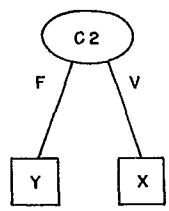


C1 irá variar de C1 para $\overline{C1}$, mas C2 permanecerá fixo, então $\overline{C1}\ \overline{C2} + C1\ \overline{C2} = \overline{C2}$

d) Fazendo um fluxograma de função, teríamos:



Como pode-se ver tanto faz C1 ser verdade como falso, que C2 será sempre testada. Tirando C1 teríamos:



Conclusão, o trecho do programa ficou simplificado, para:

IF $\overline{C2}$

THEN CALL X

ELSE CALL Y

Como podem ver, as mesmas técnicas utilizadas para simplificar circuitos, podem ser aplicadas para simplificar programas, bastando simplificar a tabela de decisão.

As tabelas de decisões complexas tem por objetivo expressar a lógica das decisões complexas de forma que a análise permita reduzir o problema à sua apresentação mais simples, ordenando e apresentando condições encontradas.

Tradicionalmente esta lógica de decisões é feita através de uma narrativa ou diagrama de fluxo.

As diferenças de enfoque sobre as técnicas serão imediatamente postas em evidência, através da ilustração feita a seguir:

INSTRUÇÕES PARA VENDA E RESERVA DE PASSAGENS

"Se o passageiro pedir um lugar de 1ª classe e se houver lugar disponível então lhe é entregue um bilhete de 1ª classe."

"Se todos os lugares de 1ª classe estão reservados e se o viajante está disposto a aceitar a 2ª classe e, caso haja lugar disponível nesta classe, então lhe é entregue o bilhete de 2ª classe.

"Caso o viajante não aceite trocar ou os lugares da 2ª classe já estão reservados, então é colocado na fila de espera para 1ª classe."

"Se o viajante pede um lugar na 2ª classe e há vaga, então lhe é entregue o bilhete de 2ª classe."

"Caso não tenha lugar e o viajante está disposto a aceitar na 1ª classe então lhe é entregue o bilhete de 1ª classe."

"Caso não aceite troca ou não tenha lugar na 1ª classe este será colocado na fila de espera para 2ª classe."

A 1ª vista o texto parece ser confuso e o procedimento complexo. Uma análise sistemática prova que não há nada disto:

A 1ª pergunta a se fazer é a seguinte:

O senhor x quer um lugar na 1ª classe ou um lugar na 2ª?

Depois verifica se tem lugar livre na classe pedida.
Se houver é entregue o bilhete correspondente, se não houver, pergunta se aceita troca. Se está de acordo e há lugar disponível na outra classe é entregue o bilhete correspondente, se não coloca na fila de espera da classe que inicialmente havia pedido.

Analisando agora o nosso problema por meio de um fluxograma, ou seja construir um grafo em forma de árvore, este toma a seguinte forma:

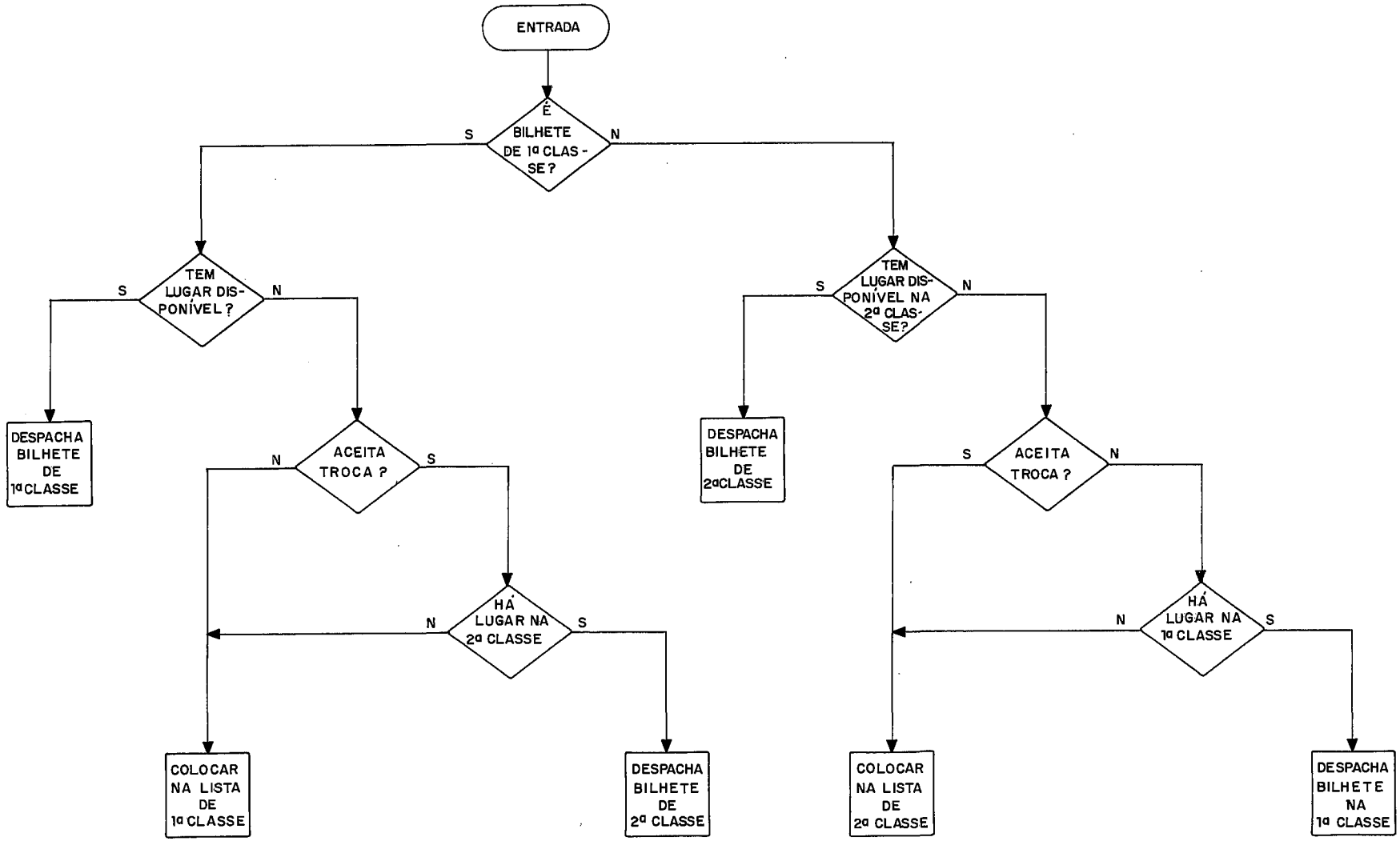

ENTRADA
É BILHETE DE 1ª CLASSE?
S
N
TEM LUGAR DISPONÍVEL?
S
N
DESPACHA BILHETE DE 1ª CLASSE
ACEITA TROCA?
N
S
HÁ LUGAR NA 2ª CLASSE
N
S
COLOCAR NA LISTA DE 1ª CLASSE
DESPACHA BILHETE DE 2ª CLASSE
TEM LUGAR DISPONÍVEL NA 2ª CLASSE?
S
N
DESPACHA BILHETE DE 2ªCLASSE
ACEITA TROCA?
S
N
HÁ LUGAR NA 1ª CLASSE
N
S
COLOCAR NA LISTA DE 2ª CLASSE
DESPACHA BILHETE NA 1ª CLASSE

fig. 2.1

Como pode-se ver quanto mais se avança no fluxograma, mais se complica o problema; é preciso em cada nível saber o que houve nos níveis anteriores. Como este grafo representa um problema limitado, é curto e simples, mas quando a análise é mais complexa, as dificuldades na compreensão do fluxograma serão maiores e difíceis de serem seguidas.

Analisando o enunciado do problema e resumindo tôdas as condições e ações temos:

CONDIÇÕES:

- Pedido de reserva é da 1ª classe?
- A 1ª classe está disponível?
- A 2ª classe está disponível?
- Aceita troca de classe?

AÇÕES:

- Despachar bilhete de 1ª classe.
- Despachar bilhete de 2ª classe.
- Colocar na lista de espera da 1ª classe.
- Colocar na lista de espera da 2ª classe.

A tabela de decisão irá permitir estabelecer uma correspondência entre o conjunto de condições possíveis e o conjunto de ações associadas ao conjunto de condições.

A tabela ficaria então:

	1	2	3	4	5	6	7	8
Pedido de reserva é 1ª classe.....	S	S	S	S	N	N	N	N
A 1ª classe está disponível.....	S	N	N	N	-	S	-	N
A 2ª classe está disponível.....	-	S	S	N	S	N	N	N
Aceita troca de classe.....	-	S	N	-	-	S	N	-
Despachar bilhete de 1ª classe.....	x					x		
Despachar bilhete de 2ª classe.....		x			x			
Colocar na lista de espera 1ª classe...			x	x				
Colocar na lista de espera 2ª classe...							x	x

O caráter 'S' quer dizer que aquela condição é verdadeira enquanto o 'N' quer dizer falsa. O traço '-' quer dizer que tanto faz, pode ser verdade ou falso. O 'x' indica as ações que serão executadas para aquêle grupo de condições.

Tôda a lógica está aqui resumida em oito linhas e colunas. O conjunto está perfeitamente definido e de fácil compreensão. O funcionamento é extremamente simples e um rápido exame do mecanismo é apresentado:

Digamos que se um viajante fizesse um pedido de reserva na 2ª classe e aceitasse troca de classe, estando a 2ª classe toda reservada e houvesse lugar na 1ª classe.

As condições da coluna 6 satisfazem esta solicitação, sendo então despachado um bilhete na 1ª classe.

Este método estabeleceu claramente a diferença de enfoque nos métodos de análise tanto na narrativa como no fluxograma.

A maior vantagem na utilização de tabelas aparece imediatamente: a tabela permite agrupar tôdas as combinações de condições e tôdas as possibilidades lógicas e controlar facilmente se não foi omitido nenhuma combinação; ao contrário do fluxograma, é difícil saber se foram considerados todos os casos.

CAPÍTULO III

CONCEITOS

CAP. III - CONCEITOS

III. 1 - HISTÓRICO

Em novembro de 1957 a General Elétric inicializou um estudo chamado 'Projeto de Sistemas Integrados'. Tornou-se aparente que os métodos utilizados de descrição de decisões - fluxograma, formulário, narrativa - foram inadequados para expressar lógica complexa encontrada no proceso em estudo. Por essa razão a equipe do projeto começou uma procura para um novo método de expressão que culminou no desenvolvimento da 'estrutura de tabelas de decisões' e num método de computarização para resolve-la.

Estas tabelas tem todas as características que nós conhecemos hoje como tabelas de decisões (T.D), porém tem um formato similar as tabelas "verdades" da qual elas originaram.

Exemplos de uma tabela verdade e uma tabela de decisão chamada TABSOL são mostrados nas figs.:3.1.1 e 3.1.2.

A	B	A OR B	A AND B
V	V	V	V
V	F	V	F
F	V	V	F
F	F	F	F

fig. 3.1.1

Tabela verdade

* V - verdade
  F - falso

ÍTEM-1	ÍTEM-2 EQ	ÍTEM-3 EQ	GOTO
EQ 4	3	05	TABLE - 2
EQ 5	4	10	TABLE - 2
EQ 7	5	15	TABLE - 3

fig. 3.1.2
Tabela TABSOL

Na figura 3.1.2 temos se o ÍTEM-1 é igual a 4 e se o ÍTEM-2 é igual a 3 e se o ÍTEM-3 é igual a 05 então vá para a TABLE-2.

Foram feitos processadores para resolver estas tabelas e operaram inicialmente em um IBM 702 e sucessivamente implementado em um IBM 305, 650 e 704.

Um processador nesta linguagem chamado TABSOL, foi implementado no GE 225 no começo de 1961.

Aproximadamente ao mesmo tempo e independente da GE, a Sutherland Company desenvolveu uma tabela de decisões de forma diferente porém de conceitos idênticos. Ao passo que a GE desenvolveu o conceito e soluções voltadas para o computador, a Sutherland desenvolveu suas tabelas voltada para a análise de sistemas e documentação, levando a solução da tabela para o programador.

Um exemplo de comunicação - orientada para tabela de decisões é mostrado na fig. 3.1.3

		CÓDIGO = 6		
		DESC = 8	DESC = 9	DESC = 10
CLASSE = 2	TIPO = 4	AÇÃO-1	AÇÃO-1	AÇÃO-2
	TIPO = 3	AÇÃO-3	AÇÃO-2	AÇÃO-1

Fig. 3.1.3 - Tabela de decisão usada para comunicação homem a homem.

Na fig. 3.1.3 temos: se o código = 6 e desc = 8 e classe = 2 e tipo = 4 então executa AÇÃO-1.

A Hurt Foods AND Industries, começou a usar as tabelas de decisões como ajuda a comunicação homem a homem em 1959.

Em maio de 1959 a Conference on Data Systems Languagens (CODASYL a organização que desenvolveu o Cobol) foi convocada com o obje - tivo de desenvolver uma técnica independente de uma linguagem.

Começaram então a estudar as tabelas de decisões durante 2 anos , resultando numa linguagem de tabela de decisões chamada DETAB-X (Decision Tables Experimental).

Em 1960 a GE cedeu seus trabalhos sobre tabelas de decisões para a Eastern Joint Computer Conference. Os 2 anos seguintes , não reportados em literatura, foram gastos no desenvolvimento de processadores de tabelas de decisões. A IBM comandou a implementação dos 3 últimos processadores de T.D, um no IBM 1401, outro no 7080 em cooperação com a Boing Company e outro no 7090 em conjunto com a Rand Corporation (FORTAB).

A Insure Company of North America produziu um processador de T.D chamado LOBOC, também no IBM 7080. A GE registrou a implementação do TABSOL no GE-225.

Em setembro de 1962 um Grupo de Sistemas da CODASYL, deu um seminário em Nova York para apresentar seus estudos sobre T.D ao público. O produto de seus esforços chamado DETAB-X , consistiu de uma linguagem interligada ao COBOL-61 para ser usada dentro da armação da T.D; o seminário destacou os últimos desenvolvimentos da: Sutherland, GE, Insure Company of North America e IBM . O objetivo era estimular o interesse e experiências sobre T.D e seus transladores.

Apesar do entusiasmo do Grupo de Sistemas e do conteúdo de informação do seminário, a troca de informação esperada nunca houve.

O pequeno grupo movido para outros projetos, levou como seu testamento o formato das T.D. agora aceito com standard. Um exemplo do seu formato é visto na fig. 3.1.4

Tabela 3	1	2	3		ELSE
Campo-1 = 3	Y	Y	N	N	-
Campo-2 =	3	4	10	15	-
Campo-3 =	ZERO	ZERO	POSITIVO	NEGATIVO	-
Move A-6 P/A-7	X	-	X	X	-
GO TO	TAB-4	TAB-4	TAB-5	TAB-6	TAB-9

fig. 3.1.4

Na figura 3.1.4 temos: Regra 2- Se campo-1 = 3 e campo-2 = 4 e campo-3 = ZERO então vá para TAB-4.

O período entre o seminário do DETAB-X até 1965 foi inativo. Poucos artigos foram publicados e pequena expansão do uso de T.D. por novos usuários foi observada.

Em julho de 1965, o Special Interest Group for Programming Languages (SIGPLAN) do Los Angeles Machinery designou um grupo de / trabalho para desenvolver um pré-processador de T.D. A fim de garantir a ampla distribuição, o processador foi escrito em subconjunto restrito do COBOL e aceitava T.D. codificada em COBOL e

para converter o código fonte em COBOL. O processador, chamado DETAB-65 foi distribuído gratuitamente através de Joint USERS Group.

Embora fosse implementado em um grande nº de computadores, incluindo CDC 1604, 3400 e 3600 e o IBM 7040, 7044 e 7094 as ineficiências dos algorítimos de conversão e a falta de manutenção levou ao eventual desaparecimento do processador.

Com excessão dos sistemas /360 da IBM o Decision Logic / Translator que processa T.D. codificados em FORTRAN, os outros geravam códigos em COBOL. Outro translator desenvolvido pela IBM foi o DECTAB que pode ser codificado tanto em COBOL como PL/1.

Em resumo, a história das T.D. pode ser vista como 4 eras:

1. A era inicial do desenvolvimento, 1957 - 1960
2. A 1ª era dos processadores, 1961 - 1962
3. A era do silêncio, 1963 - 1965
4. A 2ª era dos processadores, 1966- até hoje

As principais causas que desmotivaram a utilização das tabelas de decisões foram:

1 - Devido aos artigos serem de alto nível (o qual era necessário um conhecimento a priori) e de serem publicados apenas em jornais técnicos e não comerciais, faziam com que estes não fossem lidos.

Os próprios cursos de computação não costumam a ensinar tabelas de decisões e sim fluxogramas.

2 - A maneira de encarar um problema através de T.D. é diferente do fluxograma. A tabela requer uma análise geral das condições na solução. Enquanto o fluxograma é analisado de uma maneira sequencial . Com isto os analistas e programadores treinados em análise sequencial, relutavam em aprender a usar a T.D.

3 - Falta geral de processadores eficientes para a comunidade de processamento de dados. Como a otimização destas tabelas é um processo lento, torna-se necessário o uso de processadores que ao ler as tabelas não só otimizem como codifiquem uma linguagem para ser processada pelo computador. Experiências mostram que a ausência de um mecanismo translação resultará em uma rápida perda de interesse das T.D. pelos programadores.

4 - Com o advento da tabela, o trabalho do programador ficou reduzido a codificação, visto que as tabelas já vem prontas para serem transladas para uma linguagem processual pelo computador. Isto acarretou uma certa relutância por parte dos programadores pois seu trabalho passou de artesanato para codificação, desmotivando assim o programador.

Estas 4 condições podem ser contornadas através das seguintes medidas.

1 - Publicar artigos mais fáceis de serem compreendidos sôbre tabelas de decisões e introduzí-las em cursos de programação.

2 - Fazer transladores mais potentes, fazendo com que as tabelas de decisões já sejam a própria linguagem.

3 - Estruturar o centro de processamento de dados de tal forma que exista o elemento chamado 'codificador'.

Espera-se com isto aumento no nº de processadores assim como a utilização do mesmo.

III. 2 - TERMINOLOGIA E ESTRUTURA

III. 2.1 - Como trabalham as tabelas

Uma tabela de decisão é uma tabela que mostra as ações a serem tomadas para diferentes combinações de condições. Na preparação da tabela há 3 fatores importantes:

- condições
- ações
- regras

Uma condição é um evento ou fato que influi nas ações a serem tomadas. Em termos de fluxograma é equivalente ao que está escrito dentro do losango (símbolo decisão). Uma condição pode ter mais do que um valor, caso contrário o evento condição não teria qualquer relação material sobre as ações a serem tomadas. Uma condição pode ter somente dois valores, tais como SIM ou NÃO, FALSO ou VERDADEIRO e assim por diante ou muitos valores como por exemplo; a comparação entre as variáveis X e Y há 3 valores: igual, maior e menor (, ,). Na narrativa a condição está contida dentro da cláusula SE (IF em Ingles).

Uma ação é simplesmente qualquer operação (s) ou um processo (s) a ser (em) executado (s). Em processamento de dados pode ser manual ou uma operação de máquina, aplicando uma fórmula, arquivando um pedaço de papel, etc. Em termos de fluxograma corresponde ao símbolo retangular. Em narrativa é uma operação que está contida na cláusula ENTÃO (THEN em Ingles).

Uma regra (ou seja o relacionamento entre condições resultando em ações) pode ser vista considerando os valores para uma condição ou condições e partindo para as ações apropriadas. Em termos de fluxograma é equivalente a ligação das saídas de um símbolo de decisão ou a uma ação.

Ex.: Seja o seguinte texto:

"Em uma certa agencia de processamento de dados do governo é permitida visitação pública se: os computadores estiverem em operação, e não estiverem processando material confidencial ou quando o guia estiver disponível. Não será permitida a visita se: o guia não estiver disponível, os computadores não estiverem em operação ou material confidencial está sendo processado."

Analisando o texto temos:
As condições são:

C1 O computador está em operação?
C2 O material confidencial está sendo processado?
C3 O guia está disponível?

As ações são:

A1 A visita é possível.
A2 A visita não é possível; o computador não está operando
A3 A visita não é possível; material confidencial está sendo processado.
A4 A visita não é possível; o guia não está disponível.

O fluxograma seria da seguinte maneira:

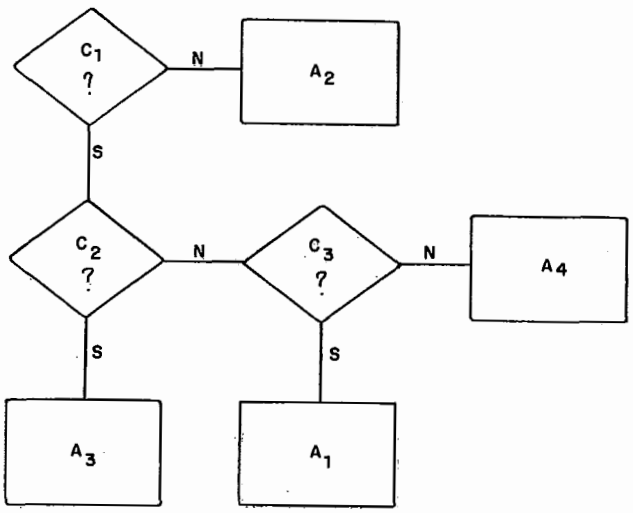


fig. 3.2.1-1

Analisando o fluxograma, existem 4 regras ou seja:

1) Se o computador está em operação e o material confidencial está sendo processado então a visita não é possível, pois o material classificado está sendo processado.

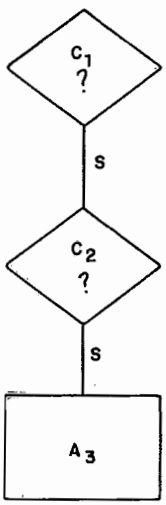


fig. 3.2.1-2

2) Se o computador está em operação e o material confidencial não está sendo processado e o guia está disponível então a visita é possível.

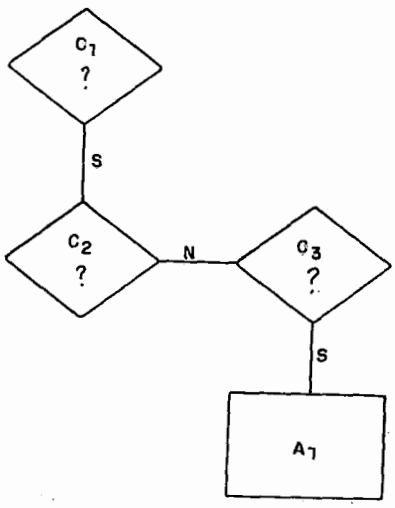


fig. 3.2.1-3

3) Se o computador está em operação, o material confidencial não está sendo processado e o guia não está disponível então não é possível a visita pois o guia não está disponível.

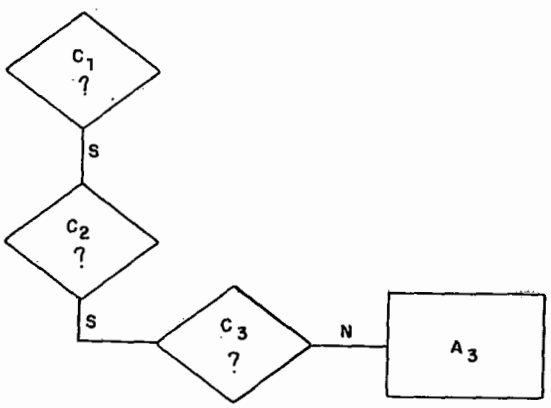


fig. 3.2.1-4

4) Se o computador não está em operação então não é possível a visita, pois o computador não está em operação.

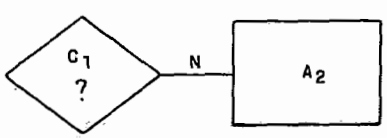


fig. 3.2.1-5

A fig. 3.2.1-6 mostra este mesmo processo em uma tabela de decisão.

	Regra 1	Regra 2	Regra 3	Regra 4
C1	S	S	S	N
C2	S	N	N	-
C3	-	S	N	-
A1		X		
A2			X	
A3	X			
A4				X

fig. 3.2.1-6

As condições estão no canto superior esquerdo, as ações estão no canto inferior esquerdo e as regras estão do lado direito. Os símbolos que aparecem nas regras são S, N, - ou seja SIM, NÃO e INDIFERENTE (don't care em Ingles, seja tanto faz , ser SIM ou NÃO que não afetam as ações a serem tomadas).

Ex.: Se o computador não está operando tanto faz se o material classificado está sendo processado ou não como também tanto faz que o guia esteja disponível ou não, que a visita não é possível pois o computador não está operando.

O símbolo INDIFERENTE poderá ter também outro significado , mas este assunto será visto mais tarde no capítulo IV item Ambiguidade.

Os x nos espaços do lado direito das ações indicam quais as ações que serão executadas.

Além desses 3 fatores básicos existem ainda outros fatores como: transição, peso, frequência e parâmetro que serão explicados no decorrer desse capítulo.

III. 2.2 - Formato

Uma tabela de decisões é uma representação tabular de informações e dados e nela estará contida toda a representação da lógica de um problema complexo, através de regras de decisões.

A forma básica aceita universalmente é vista na 3.2.1 .

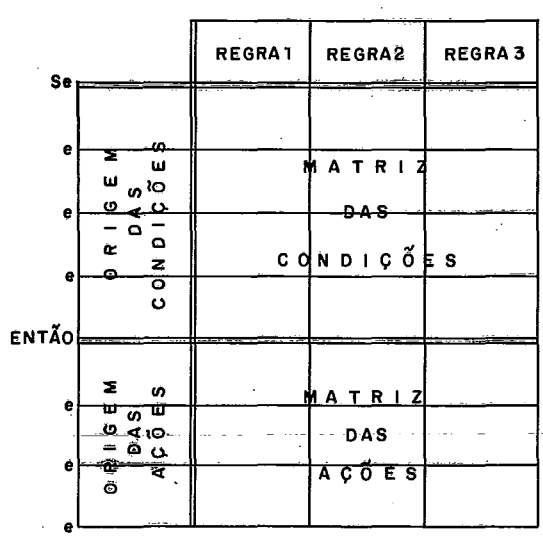


fig. 3.2.2.1 - T.D. standard

A tabela representa a lógica IF condição THEN ação.

Como podemos ver a tabela é composta de 4 quadrantes.

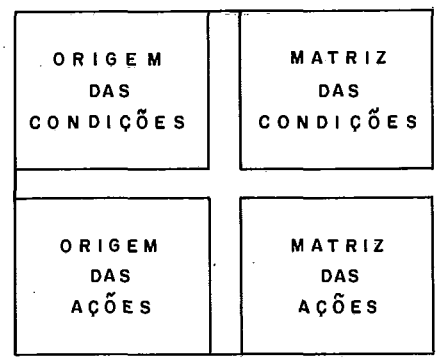


fig. 3.2.2-2 - ESTRUTURA DA T.D.

Chamamos de regra a uma determinada combinação na matriz de condições resultando numa determinada combinação na matriz de ações.

A origem das condições contém um conjunto condições que irão ser testadas (em ingles é chamada 'condition stub').

A matriz das condições contém símbolos para determinadas condições que irão compor uma regra. (em ingles é chamada 'entry condition'). Os símbolos podem ser 'S' (verdade), 'N' (falso) e '-' (tanto faz falso ou verdadeiro, em ingles é chamado de 'dash').

A origem das ações contém um conjunto de ações que irão ser executadas, resultante de determinadas condições. (em ingles é chamada de 'action stub').

A matriz de ações contém o símbolo 'x' indicando quais ações serão executadas através de uma combinação de condições, decorrente da aplicação de uma determinada regra. (em ingles é chamada de 'action entry').

Desta forma a tabela de decisão tem o seguinte significado: Se estas condições são satisfeitas então certas ações devem ser efetuadas. (fig. 3.2.2-3).

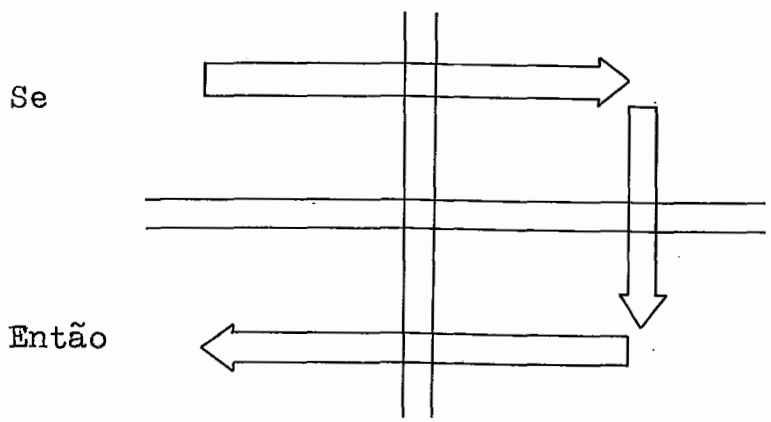


fig. 3.2.2-3

O conjunto destas regras de decisões devem cobrir um conjunto de casos possíveis sem omissão, sem redundancia(ações iguais para regras iguais) e sem contradição (ações diferentes para regras iguais).(vide cap. IV).

Em resumo, o funcionamento da tabela em cada passada por ela se utiliza de apenas uma coluna, existindo portanto somente uma regra de decisão a ser executada.

Crítica ao formato:

Observando-se a fig. 3.2.2-3 podemos tirar as seguintes conclusões:

- As condições e ações estão na vertical.
- As regras estão na horizontal.

Este formato apresenta, no meu entender uma desvantagem:

- O nº de regras possíveis é da ordem da 2n onde n é o número de condições (é 2 porque só existe 2 valores 'S' ou 'N' para cada condição). Visto isso a tendência de crescimento da tabela será para a direita (num formulário contínuo a limitação é no sentido horizontal, pois no sentido vertical não há limitações).

Ora, se fizermos as seguintes mudanças:

- As condições na horizontal.
- As regras e ações na vertical.

teremos a seguinte vantagem:

- A tendência de crescimento da tabela seria para baixo e não para a direita, podendo com isto utilizar o formulário contínuo.

Com essa alteração, o formato da tabela ficaria da seguinte forma:

	C1	C2	C3	C4		onde
						C - condições
R1					A1	A - ações
R2					A2, A1	R - regras
R3					A3	
R4					A4	
R5					A5, A3	

fig. 3.2.2-6

ou seja:

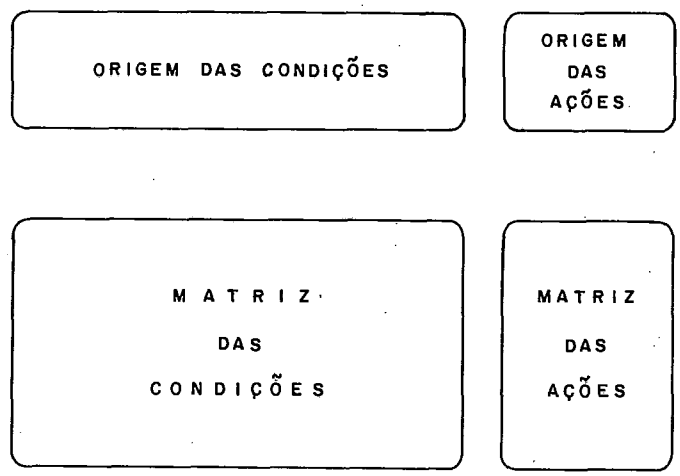


fig. 3.2.2-7

III. 2.3 - Tipos

Existem 3 tipos de tabela:

- Entrada limitada
- Entrada expandida ou semi- expandida
- Entrada mista.

Entrada limitada

A tabela de entrada limitada tem as seguintes características:

- As condições e ações estão inteiramente escritas nas próprias origens.

- Na matriz das condições só pode ter 'S', 'N' (sim ou não;ou Ø ou 1) ou (indiferente).

- Na matriz das ações só pode ter X (se a ação deve ser executada) ou branco (em caso contrário).

Exemplo:

Cálculo de percentagem de devolução, segundo situações de idade e familiares (fig. 3.2.3-1)

Idade 30	Casado		
S	S		Percentagem 1000 pts, redução 0,2%
S	N		Percentagem proporcional, " "
N	-		Redução 0,2%

fig. 3.2.3-1

As tabelas de entrada limitada são extremamente volumosas, se as condições do problema interferir nos valores discretos de uma variável. Como pode-se ver é necessário testar separadamente cada valor dessa variável o que resulta em escrever tantas colunas de condições, quantos valores diferentes haja.

Exemplo:

Cálculo de percentagem de desconto P aplicável a um pedido A.

Se A < 25 x calcula P = 0%
Se A < 40 x calcula P = 2%
Se A < 92 x calcula P = 5%
Em outro caso P = 10%

A tabela correspondente ficaria:

A < 25	A < 40	A < 92	
S	-	-	P = 0%
N	S	-	P = 2%
-	N	S	P = 5%
-	-	N	P = 10%

fig. 3.2.3-2

A observação feita quanto ao crescimento de condições é a mesma para o crescimento de ações, conforme o nº de valores diferentes atribuídos a P.

Entrada expandida ou semi-expandida:

A 1ª parte das condições continuam na origem, ou seja a 1ª parte da comparação fica na origem e a outra parte fica na matriz da regra (pode ser uma variável ou constante). Na entrada semi- ex-/ pandida só o operando pode figurar nas colunas.

Ex:

A =	T =	
6,5	-	GOTO TAB 22
B	110	Referência = 1010, escreva referência, GOTO TAB 30
B	220	Referência = 1021, escreva referência, GOTO TAB 30

fig. 3.2.3-3

A entrada expandida contém nas linhas o operando e o operador.

Ex:

Código	DA	
= 1	< 500	Calcula DA/N, GOTO TAB 12
= 1	⩾ 500	Erro + 0
= 2	< 1100	Calcula DA/N, GOTO TAB 12
= 2	⩾ 1100	Erro + 1
< 2		Ler registro, GOTO TAB 1

fig. 3.2.3-4

Entrada mista:

As tabelas de entrada mista contém tanto a limitada como expandida.

Ex: Cálculo de impostos

Renda Anual	Casado	
< 100.000	N	Cota = 2%, Q=1, GOTO TAB 3
< 100.000	S	Cota = 3%, Q=1, GOTO TAB 5
< 175.000	S	Cota = 1%, GOTO TAB 5
> 175.000	—	GOTO TAB 6

fig. 3.2.3-5

Uma tabela expandida pode conter ELSE , funções que aumentam mais as possibilidades de T.D.

Ex: Cálculo científico

ALFA	L >	Z	
<BETA	Y + C/D	= 1 2 3 4	ALFA = 5.6 , CALL SP
>GAMA (AB)	A (I)	≤COS (R)	$X/4^2$, CALL GAM (≠, 4)
ELSE	ELSE	ELSE	GO TO TAB 34

fig. 3.2.3- 6

Existe também a tabela incondicional em que as ações são executadas sistematicamente não havendo condições a serem executadas.

Exemplo:

LER;
GRAVAR;
ZERAR;
GOTO TAB 40

fig. 3.2.3-7

III.2.4 - Transição

Em qualquer caso onde há uma lógica complexa, o perfeito entendimento só é possível pela divisão do problema complexo em pequenos problemas. Se a lógica está sendo descrita na forma narrativa, por exemplo, um resumo geral deve ser dado para mostrar o problema como inteiro. As partes maiores do problema podem ser descritos em parágrafos. Cada parte do problema poderá ser relatada pelo uso de um resumo geral pela especificação de interligação em um parágrafo.

Em fluxograma a lógica do processamento total pode ser dividida em blocos maiores, sendo cada bloco descrito em uma página separada do fluxograma.

O mesmo acontece com as tabelas, podendo uma tabela ser subdividida em várias outras e interligadas.

Ex:

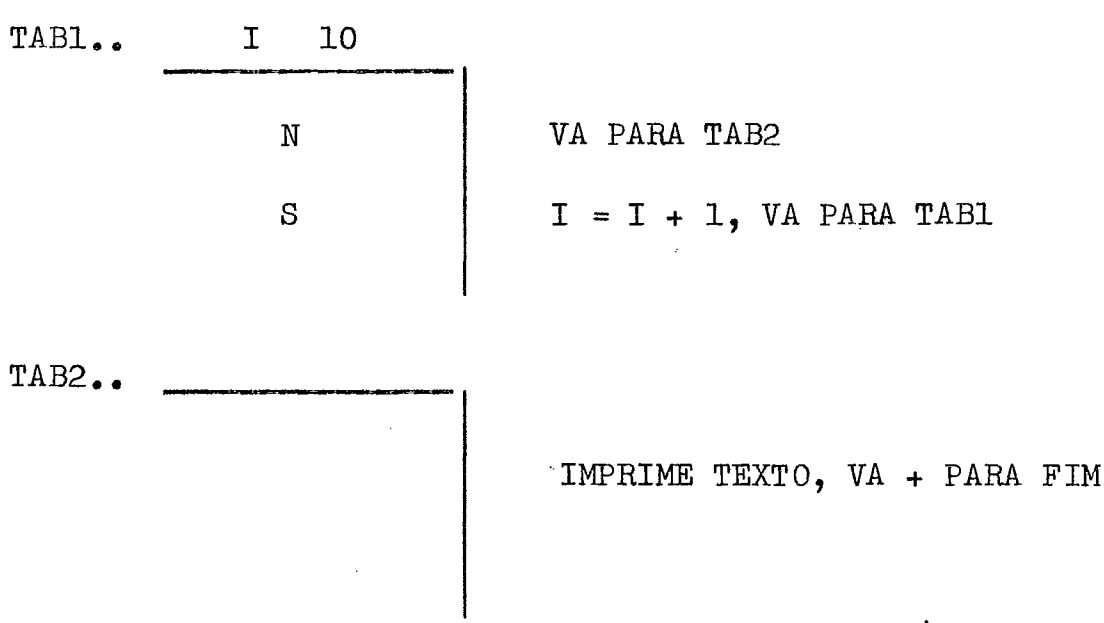


fig. 3.2.4-1

Observando o exemplo vemos 2 tabelas sendo uma condicional e a outra incondicional. As ações contidas na 1ª tabela contém desvios tanto de uma tabela para outra como para si própria. A esses desvios chamamos de 'TRANSIÇÃO'.

Para evitar estar repetindo vários VA PARA, convencionamos o seguinte formato da tabela:

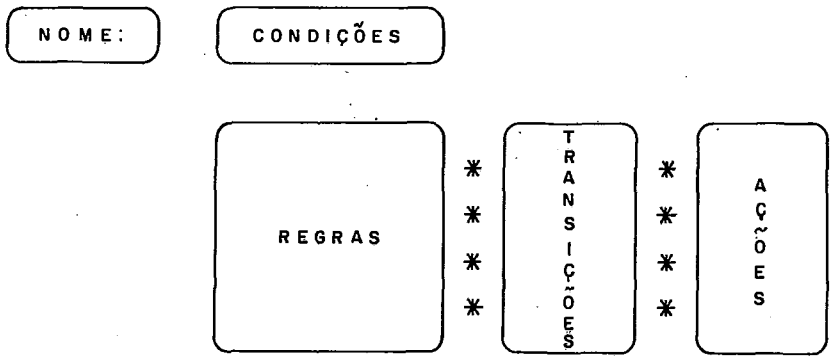


fig. 3.2.4-2

Aplicando este formato no exemplo temos:

```
TAB1..
        I ≤ 1Ø
          N        * TAB2 *
          S        * TAB1 *

TAB2..
                   * FIM * IMPRIME TEXTO
```

fig. 3.2.4-3

A transição foi colocada antes das ações, porque a transição é sempre o nome de uma tabela e tem um tamanho fixo. Já nas ações não acontece isso, pois o nº de ações pode ser variável.

Esta técnica de subdivisão de tabelas será visto posteriormente em 'Encadeamento de Tabelas' no capítulo IV.

III. 2.5 - Frequência e Peso

Os outros dois fatores a serem considerados são : a frequência de ocorrência de uma regra e o custo ou peso de um determinado teste ou condição. Estes dois fatores serão mostrados a seguir:

Seja a tabela:

C1	C2	C3	
S	-	-	A
N	S	S	B
N	N	-	C
ELSE			Z

fig. 3.2.5-1

Escolhemos aleatoriamente uma das 3 condições para ser testada primeiro. Digamos que a escolhida seja a condição C1. Representando graficamente temos:

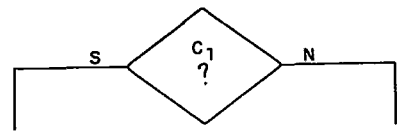


fig. 3.2.5-2

Para simplificar adotaremos as seguintes convenções:

- Colocar dentro do losangulo a condição.
- Colocar a ação dentro de um quadrado.
- Quando a condição for verdade colocar um 'S' do lado esquerdo do losangulo e quando for falso colocar 'N' do lado direito do losangulo.
- Quando uma condição na tabela tiver um hífem '-' a condição tanto pode ser 'S' como 'N' (Indiferente).

Voltando a tabela, vemos se a condição C1 for 'S', as condições C2 e C3 são indiferentes, neste caso não precisam ser testadas, resultando então na execução da ação A. Se C1 for 'N' ficaria uma parte da tabela a ser traduzida, ou seja representando no fluxograma ficaria:

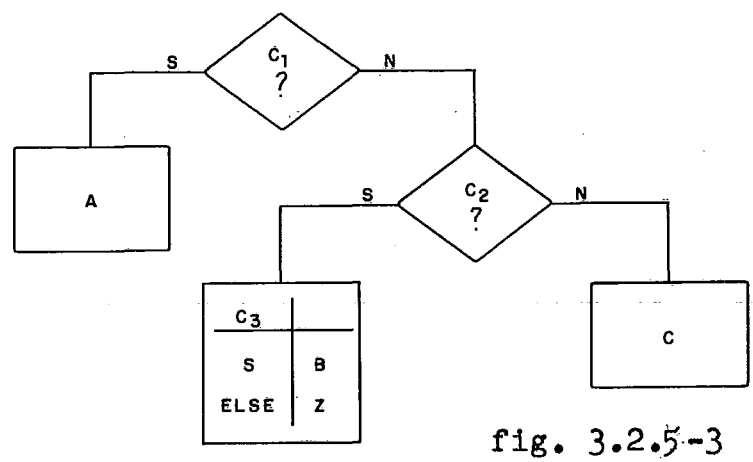


fig. 3.2.5-3

Como pode ser visto na fig. 3.2.5-3, sobraram as condições C2 e C3 para serem testadas. Digamos então que escolhemos aleatoriamente a condição C2. Se C2 for falso 'N', C3 é indiferente e não precisa / ser testado nesse caso, e será executada a ação C. Caso C2 seja verdadeiro então a tabela ficaria resúmida a 1 condição. O fluxograma ficaria:

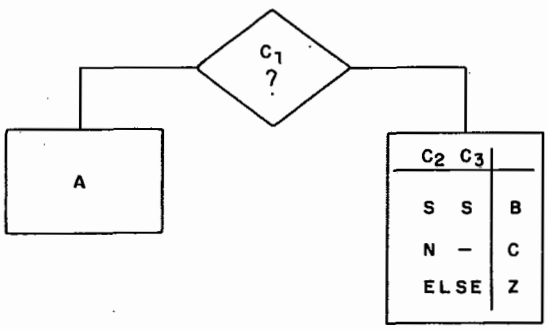


fig. 3.2.5-4

Se a condição C3 for verdadeira 'S' então será executada a ação B caso contrário será executada a ação Z. Finalmente o fluxograma ficaria:

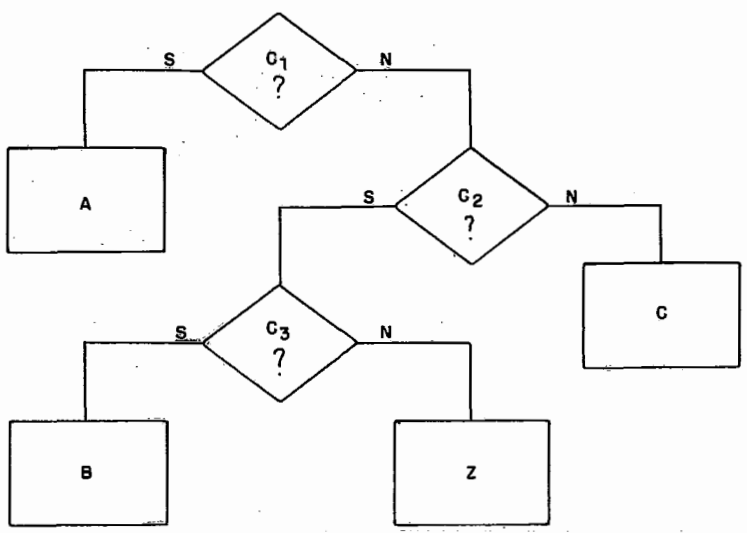


fig. 3.2.5-5

Caso a condição C2 tivesse sido escolhida primeiro, teria o seguinte desenvolvimento:

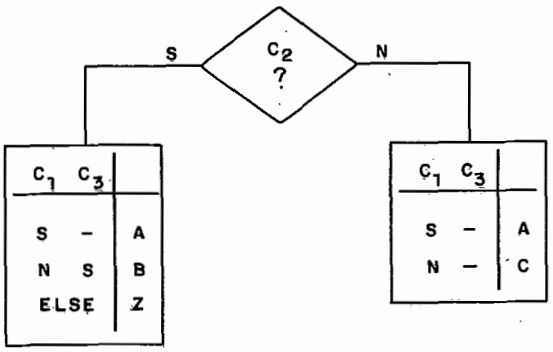


fig. 3.2.5-6

Escolhido arbtrariamente C1 para ser testada na fig. 3.2.5-6 teríamos o seguinte fluxograma:

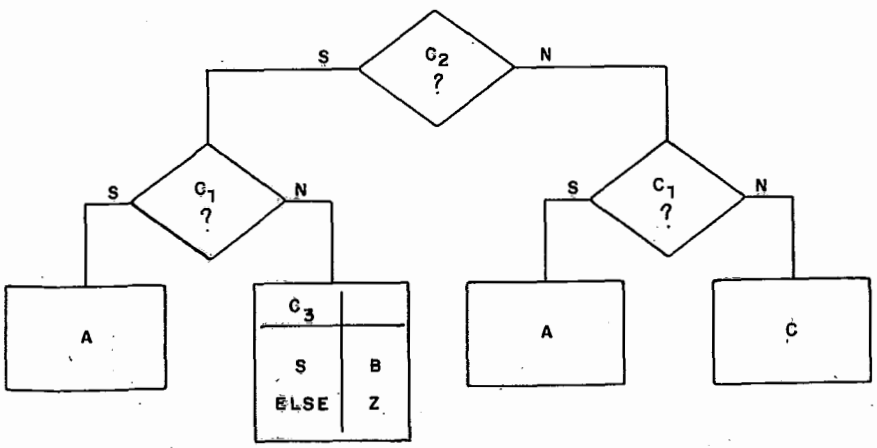


fig. 3.2.5-7

Finalmente teríamos o seguinte fluxograma convertido da fig. 3.2.5-8.

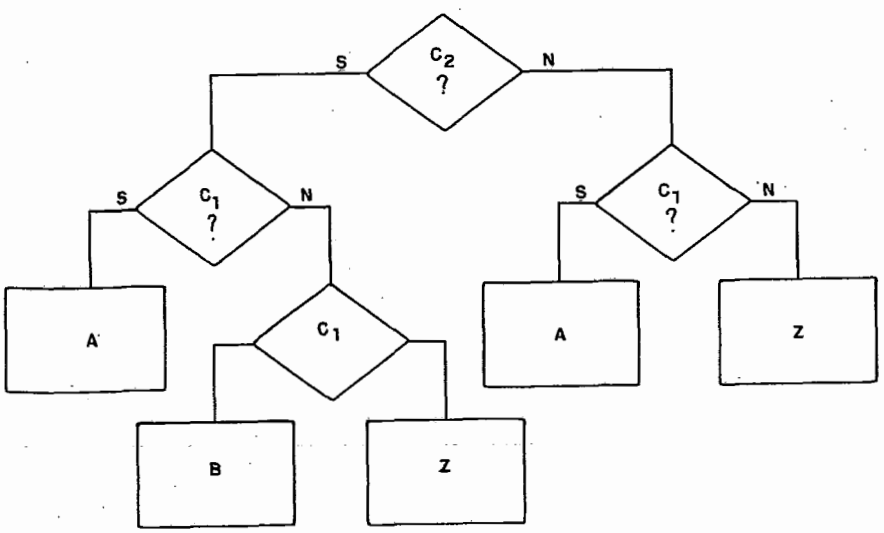


fig. 3.2.5-8

Como pode-se ver existem várias maneiras de se obter um fluxograma através de uma tabela de decisões. Como vimos nas figs. / 3.2.5-5 e 3.2.5-8 o mesmo total de teste pode ser diferente para uma mesma tabela.

A maneira ótima de se obter um nº total de testes mínimo será através da utilização tanto da frequência de ocorrencia de uma regra como o custo de uma determinada condição. Será visto posteriormente no capítulo IV em otimização, este assunto com mais detalhe.

O novo formato da tabela introduzindo a frequencia e o peso (ou custo) seria:

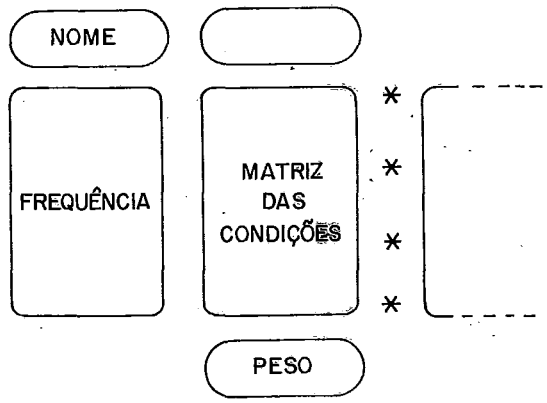


fig. 3.2.5-9

Ex:

```
TAB1..
                C1   C2
        70.0    S     N      * TAB1 *  I = I + 1
        30.0    N     S      * TAB2 *
                60    40
```

fig. 3.2.5-10

III. 2.6 - Parametros

Seja a seguinte caixa preta:

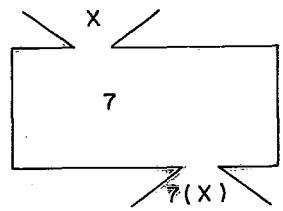


fig. 3.2.6-1

Ao entrar uma variável x na caixa preta saíra uma função desse valor que entrou. Essa função por exemplo pode ser o cálculo do CPF, etc.; A essa variável de entrada é que chamamos de parametro e pode tomar vários valores.

Uma tabela pode ter também uma entrada por parametro e o formato desta ficaria da seguinte maneira.

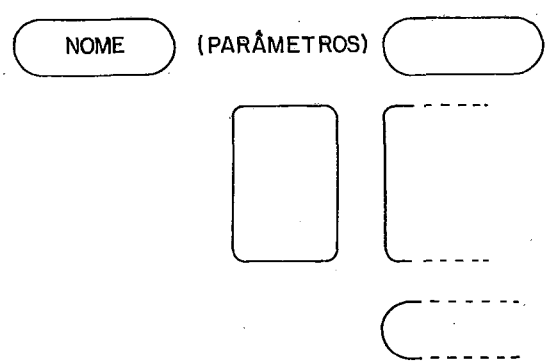


fig. 3.2.6-2

Ex:

```
TAB1..
        I = 1
          S       * TAB2 * J = 1, CALL TAB3 (J)
          N       * TAB1 *

TAB3..  (J)
                  * RETURN *  J = J + 1/J * + 2
```

III. 3 - Notação

A estrutura da tabela de decisão que será analizada daqui por diante, terá a seguinte forma:

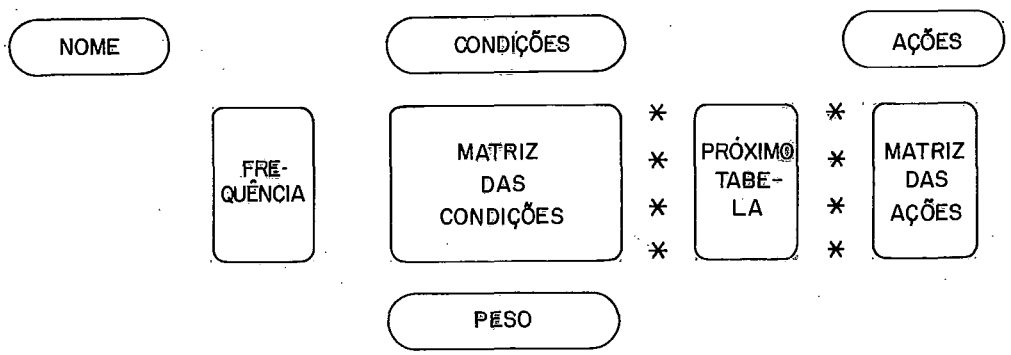


fig. 3.3-1

DESCRIÇÃO:

NOME: Nome da tabela

PARAMETRO: Valores de variáveis que podem ser assumidos dentro da tabela.

CONDIÇÕES: Conjunto de testes que terá a tabela.

FREQUÊNCIA: Nº de ocorrência de uma determinada regra(percentual).

PESO: Nº de uma determinada ocorrência para um determinado valorda condição.

MATRIZ DAS CONDIÇÕES: Combinação de condições resultando um conjunto de ações.

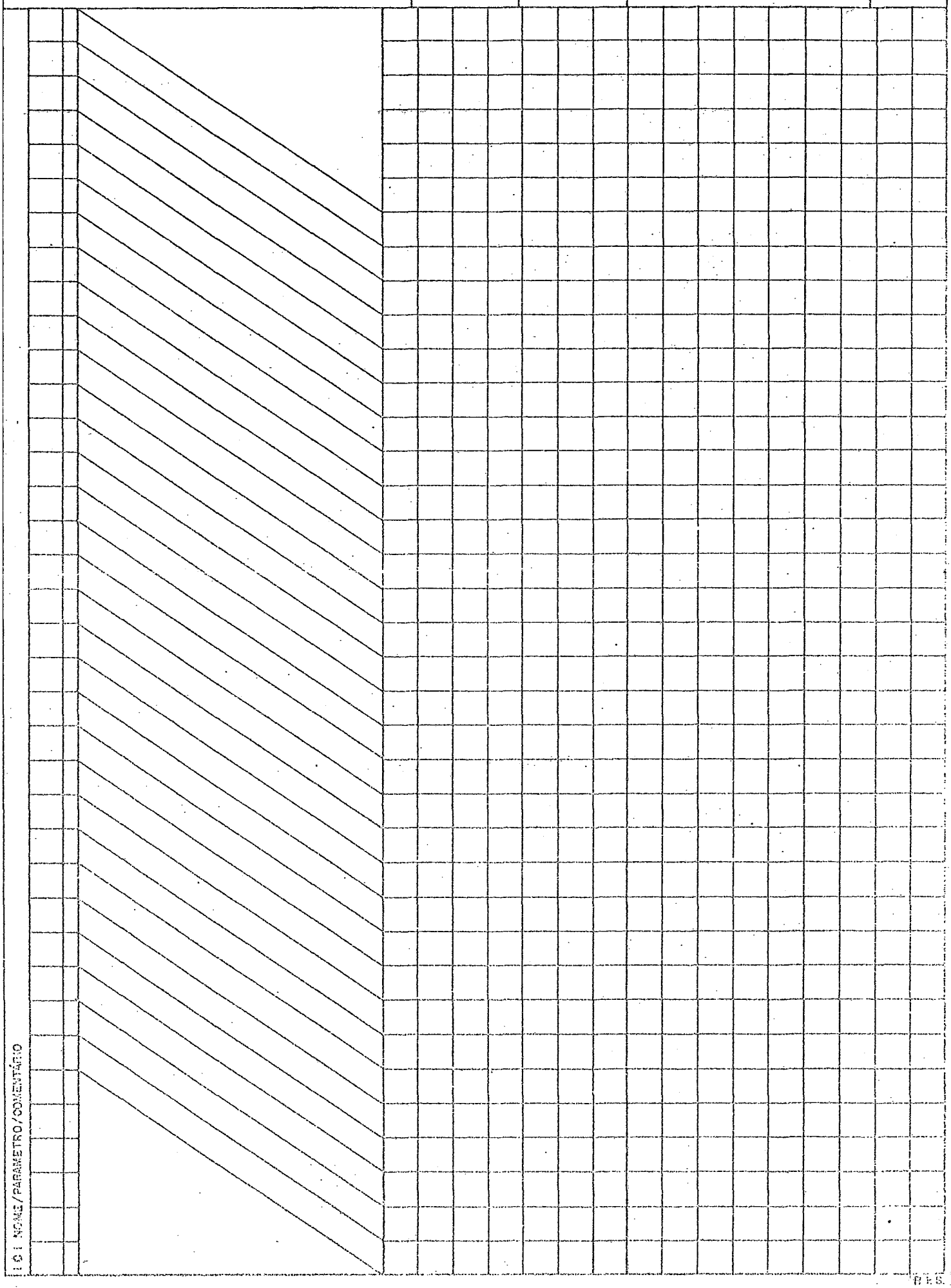
PRÓXIMA TABELA: Em vez de colocar um GOTO em AÇÕES, isto indicará qual a próxima tabela que deverá ir depois de ter executado um determinado conjunto de ações.

AÇÕES: Conjunto de ações que serão executadas sequencialmente para uma determinada conjunto de condições ou regra.

NOTA:

Para facilitar a utilização das tabelas, foi criado o formulário para preenchimento das tabelas, na fig. a seguir.

TÍTULO	SISTEMA	DOCUMENTO	NOME	FOLHA



CAPÍTULO IV

ANÁLISE DA CONSTRUÇÃO DA TABELA

CAP. IV - ANÁLISE DA CONSTRUÇÃO DA TABELA

IV.1 - Condições

IV.1.1 - Definições

As seguintes convenções serão adotadas:

- o símbolo 'n' representa o número de condições.
- a condição genérica será denominada como 'C'.
- o índice 'j' será usado para a sequência de condições.
- 'Cj' será uma determinada condição entre as n condições (portanto $1 \leq j \leq n$) ou seja C1, C2,... Cn

- Cada condição pode ser avaliada como verdadeira ou falsa, sendo aj = N se Cj é falsa ou aj = S se Cj é verdadeira. Isto será indicado por: V(Cj) = N ou V(Cj) = S.

- o estado do conjunto completo das n condições é representado pela função

$$F = (a_1, a_2, a_3, \ldots a_n)$$

- na j ésima de F ou seja aj ou, V(Cj) temos aj = N ou (V(Cj) = N) se Cj é falso e aj = S ou (V(Cj) = S) se Cj é verdade.

IV. 1.2 - Estrutura das condições

Cada condição Cj, consiste de 2 operandos ligados por um operando de relações. Pelo menos 1 operando é uma condição variável("data ítem" que pode assumir qualquer valor de um conjunto) e outro que pode ser uma condição variável ou uma constante. O operador relacional pode ser qualquer um dos seguintes símbolos: =, $\leq, \geq, <, >, \neq$ (igual, menor ou igual, maior ou igual, menor, maior e diferente).

A qualquer instante a determinação de um valor exato, V (Cj) é obtido de Cj. Cada componente aj da função F é igual a V (Cj) . Por exemplo, consideramos as 4 condições:

$C1 = (W < 3)$
$C2 = (X = 3)$
$C3 = (Y > 4)$
$C4 = (Z \leq 0)$

fig. 4.1.2-1

onde W, X, Y, Z, são condições variáveis.

A representação dos valores destas 4 variáveis erão mostradas a seguir junto com os valores exatos correspondentes das condições associadas.

J	Cj	VALOR ATUAL DA CONDIÇÃO VARIÁVEL	V(Cj)
1	W< 3	W = 2	S
2	X = 3	X = 5	N
3	Y > 4	Y = 3	N
4	Z ≤ 0	Z = -1	S

fig. 4.1.2-2

F= (a1,a2, a3, a4) = (S, N, N,S)

IV. 1.3 - Condições dependentes e independentes

Dependencia ou indenpendencia existe entre qualquer par de condições. Definimos duas condições. Ce e Cr, como sendo dependentes se ambas tem a mesma condição variável como operando. Duas condições são independentes se nenhuma das duas tem a mesma condição variável como operando.

Exemplo: Verifique fig. (4.1.3-3)

Há dois tipos de dependencia: mutuamente exclusiva e sobreposta.

Dependencia mutuamente exclusiva:

Esta dependencia ocorre para um par de condições, Cr e Ce, quando não tem um valor comum para as condições variáveis, talque ambos V(Cr) e V(Ce) nunca podem ser igual a S ao mesmo tempo.

Ex:

Cr	Ce	Valor de X	V(Cr)	V(Ce)
x=1	5=x	1	S	N
x=1	5=x	5	N	S
x=1	5=x	nem 5 nem 1	N	N

fig. 4.1.3-1

Se x=1 é verdade, 5=x não pode ser ao mesmo tempo verdade; enquanto se 5=x é verdade então x=1 não pode ser verdade. Note que ambos podem ser falso ao mesmo tempo, tal como x=3.

Dependencia Sobreposta

A dependencia sobreposta ocorre, para um par de condições, / quando pode existir pelo menos um valor, tal que, tanto Cr como Ce sejam verdadeiras. Pode haver também outras combinações.

Ex:

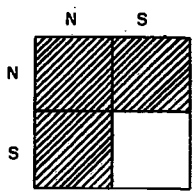
Cr	Ce	Valor de X	V(Cr)	V(Ce)
X > 0	X ≥ 1	1	S	N
X > 0	X ≥ 1	2	S	S
X > 0	X > 1	0	N	N

fig. 4.1.3-2

Diagrama de dependencias

O relacionamento possível entre V(Cr) e V(Ce) nos 2 tipos de dependencia é ilustrado na fig. 4.1.3-3.

V(Ce)



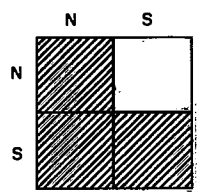
mutuamente exclusiva

Cr: x=a

Ce: x=b

(a≠b)

V(Ce)



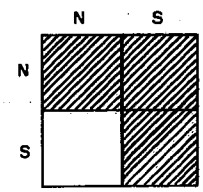
sobreposição

Cr: x > a

Ce: x > b

(a > b)

V(Ce)



sobreposição

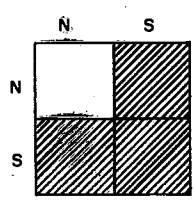
Cr: x < a

Ce: x < b

(a < b)

fig. 4.1.3-3

V(Ce)



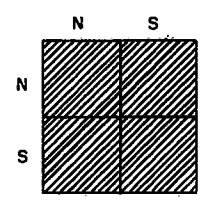
sobreposição

Cr: x > a

Ce: x < b

(a < b)

V(Ce)



condição independente

Como pode-se ver cada diagrama varia de acordo com a natureza de C_e e C_r; por exemplo no caso em que $C_r = (x > \emptyset)$ e $C_e = (x > 1)$ a combinação $V(C_r) = N$ e $V(C_e) = S$ nunca poderá acontecer ao mesmo tempo, pois se $x > 1$ então x precisa ser muito maior que $\emptyset$.

IV. 1.4 - Requisitos das condições

Ao se analisar um problema tôdas as condições relevantes deverão ser listadas. Se é uma tabela de entrada limitada, as condições serão encaradas com um questionário. As condições deverão ser bem precisas e claras - Assim como em fluxograma isso requer prática. Se qualquer abreviação comutária é usada, deverá ser explicada no topo de cada uma condição.

Ex: C1 Reserva para 1ª classe (RPC) está completo
C2 Pedido foi de 1ª classe

C1	C2
S	S
S	N
N	S
N	N

fig. 4.1.4-1

Expressões negativas deverão ser evitadas, uma expressão positiva geralmente é mais fácil de manipular quando uma tabela está sendo desenvolvida para o caso positivo.

Ex: C1 NÃO ESTOCAR?

S

N

fig. 4.1.4-2

A condição não estocar se for S significa ficar fora / do estoque e se for N significa estoque permitido.

Uma boa política também é evitar condições dependentes. Onde uma condição inclui outras condições, é característico dar confusões.

Seja o exemplo:

C1 $1 \leqslant X \leqslant 10$?

C2 $1 \leqslant X \leqslant 5$?

					A4 ERRO
C1	C2	A1	A2	A3	A4
S	S	X	X		
S	N	X	X		
N	N	X		X	
N	S				X

fig. 4.1.4-3

O que está tabela realmente representa é:

C1 $6 \leqslant X \leqslant 10$?

C2 $1 \leqslant X \leqslant 5$?

					A4 ERRO
C1	C2	A1	A2	A3	A4
N	S	X	X		
S	N	X	X		
N	N	X		X	
S	S				X

fig. 4.1.4-4

Note que ambas tabelas mostram uma lógica aparente de erro na regra 4; isto é devido a restrição das tabelas de entrada limitada. Entretanto na fig. 4.1.4-3 a condição C2 está incluída na condição C1; no 2º exemplo (fig. 4.1.4-4) as 2 condições são mutuamente exclusivas. Evitando o uso de condições dependentes, previnimos das confusões que surgiram na fig. 4.1.4-3.

Consideramos agora um outro exemplo de condições dependentes.

Ex: C1 X = 1 A 5?

C2 X = 5 A 10?

	C1	C2	A1	A2	A3	A4
Regra 1	S	S	X			
Regra 2	S	N	X		X	
Regra 3	N	N		X	X	
Regra 4	N	N				X

fig. 4.1.4-5

Neste caso temos uma interpretação grosseira e confusa do problema lógico. A regra 1 é aplicada somente para X = 5; a regra 2 é aplicada quando X = 1 A 4; a regra 3 é aplicada quando X = 6 A 10. Se nós ignoramos a interpretação sequencial das regras e aplicamos X = 5 para a regra 2, vemos um caso sem sentido. Se nós representarmos os 2 métodos no fluxograma dessa lógica temos:

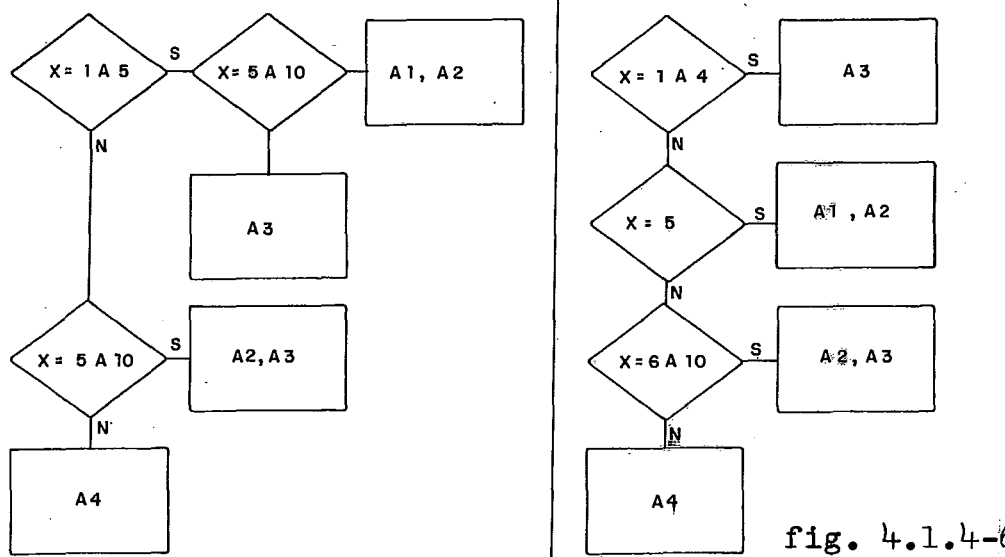


fig. 4.1.4-6

A solução da esquerda pode ter uma comunicação entre homem e máquina mais precisa, porém a do lado direito é certamente mais conveniente para comunicação entre homem e homem.

Por causa dos problemas causados pelas condições dependentes é preferível evitar a dependencia tanto quanto possível.

Logicamente a sequencia das tabelas não afeta a validade da norma, porém ela altera a facilidade de leitura e construção. Uma regra básica é reunir as condições na medida em que são identificadas, podendo ser revisada mais tarde para melhorar a compreensão.

Se o número de condições é grande, convém segmentar a tabela. Se possível não passar de 5 condições. Isto será discutido no ítem 'Encadeamento de tabelas', neste capítulo.

IV. 2 - Ações

Um determinado conjunto de condições irá selecionar uma determinada regra da tabela de decisão, que irá executar uma série de ações.

Ações são passos operacionais a serem executados em alguma sequencia específica, podendo ser variável de uma regra para outra.

Os passos operacionais são procedimentos imprencidíveis como:

```
        TAX = A * B
ou      A = B
ou      ARQUIVA RELATÓRIO
ou      LÊ ARQUIVO
ou      IMPRIME TEXTO
ou      CALL CABEÇÁRIO
```

fig. 4.2-1

Se qualquer abreviação e comentários são usados deverão ser explicados no topo de cada ação.

Ex: Seja uma tabela incondicional

TAB1..

A1 LÊ ARQUIVO
A2 IMPRIME TOTAIS
A3 IMPRIME CABEÇÁRIO

* TAB2 *	A1	A2	A3
	X	X	X

fig. 4.2-2

Em muitos casos a sequência de ações é de grande importância; como por exemplo 'Encadeamento'de tabelas, veremos posteriomnte, é recomendada então que as ações sejam mostradas na sequencia em que são chamadas, colocando um nº de sequencia após o 'x'.

Ex:

TAB1..

C1	C2	C3		A1	A2	A3
S	S	N	* TAB2 *	X1	X2	X3
N	S	S	* TAB3 *	X3	X1	X2

fig. 4.2-3

IV. 3 - Regras

IV. 3.1 - Regra Zero

A regra zero é uma regra incondicional ou seja não tem condições. Apenas terá ações e transações.

Ex:

TAB..		A1	A2	A3
	* TAB2 *	X1	X2	X3

fig. 4.3.1-1

IV - 3.2 - Regra Simples

A regra simples é uma regra que não tem o símbolo indiferente ('-'). A matriz das regras são agrupadas, para regra simples, em dois grupos: 'S' e 'N'. Esses dois grupos são inteiramente exclusivos. Assim uma entrada na matriz de regra não indiferente assume dois estados: VERDADE(S) e FALSO(N).

-Excluindo entrada expandida, se há n condições na tabela, existem 2^n regras simples independentemente possíveis para esta tabela. A base 2 nesta função (2^n) significa o nº de estados possíveis.

Ex: Se há 2 condições em uma tabela, não há mais que 4 ($=2^2$) regras simples independentes possíveis ou seja:

C1	C2	A1	A2
S	S	X	
S	N		X
N	S	X	X
N	N	X	

fig. 4.3.2-1

IV. 3.3 - Tabelas completas

As tabelas são ditas completas quando tôdas as regras representam tôdas as combinações possíveis das condições. O exemplo a seguir mostra uma tabela completa com todas as regras simples aplicadas a 4 condições.

Ex: TAB..

C1	C2	C3	C4
S	S	S	S
S	S	S	N
S	S	N	S
S	S	N	N
S	N	S	S
S	N	S	N
S	N	N	S
S	N	N	N
N	S	S	S
N	S	S	N

fig. 4.3.3-1

Ex: TAB..

(CONT.)

C1	C2	C3	C4
N	S	N	S
N	S	N	N
N	N	S	S
N	N	S	N
N	N	N	S
N	N	N	N

fig. 4.3.3-1

Há vários procedimentos que podem ser usados para formar uma tabela completa. Um dos procedimentos é o seguinte:

1º passo:

O nº total de regras é o nº de valores para a 1ª condição, vezes o nº de valores de enésima condição. No exmplo dado, cada condição tem dois valores SIM ou NÃO, porque a tabela era limitada. O nº total de regra simples é assim:

C1 (2 valores) x C2 (2 valores) x C3 (2 valores) x C4 (2 valores) = 16 regras simples.

Para uma tabela de entrada limitada, o nº de regras simples é igual a 2^4.

Para uma tabela de entrada expandida o procedimento de multiplicar o nº do valor da condição é o seguinte:

C1 compare X: 4

C2 tipo do registro

C3 idade

C1	C2	C3
>	1	=1
=	2	>1
<	3	<1
	4	

fig. 4.3.3-2

Aplicando o procedimento descrito a fig. 4.3.3-2 teremos:

(nº de valores de C1) x (nº de valores de C2) x (nº de valores de C3) = 3 x 4 x 3 = 36 regras simples.

2º passo:

Tendo determinado o nº total de regras o próximo passo é arrumar a matriz. O método recomendado aqui é o uso do fator de grupo. Este fator é obtido pela divisão do nº total de regras pelo nº de valores da 1ª condição, pegando o quociente e dividindo pelo nº de valores da 2ª condição e assim por diante.

Seguindo o exemplo da fig. 4.3.3-1, que tem 4 condições na entrada limitada, o fator de grupo é calculado da seguinte maneira.

Ex:

Total de nº de regras = 16

nº de valores de C1 = 2; fator de grupo C1 = 16/2 = 8

" " " " C2 = 2; " " " C2 = 8/2 = 4

" " " " C3 = 2; " " " C3 = 4/2 = 2

" " " " C4 = 2; " " " C4 = 2/2 = 1

fig. 4.3.3-3

O fator de grupo é então usado para arrumar os valores da matriz. Cada valor é escrito o nº de vezes indicado pelo fator de grupo até que o nº total de regra tenha completado. Pegando o exemplo anterior, o fator de grupo para C1 foi 8 com 2 valores (S/N). A 1ª condição será composta de 8 'S' e 8 'N'. A 2ª condição como o fator de grupo foi 4 então será composta de 4 'S'e 4 'N' até completar 16.

Ex:

C1	C2	C3	C4
S	S	S	S
S	S	S	N
S	S	N	S
S	S	N	N
S	N	S	S
S	N	S	N
S	N	N	S
S	N	N	N

fig. 4.3.3-4

Ex: (CONT.)

C1	C2	C3	C4
N	S	S	S
N	S	S	N
N	S	N	S
N	S	N	N
N	N	S	S
N	N	S	N
N	N	N	S
N	N	N	N

fig. 4.3.3-4

Depois de uma certa prática esta montagem é feita automaticamente, não precisando fazer cálculos.

Aplicando o mesmo procedimento a tabela expandida, teremos:

1 - Há 3 condições

C1 X: Y

C2 tipo do registro

C3 idade (em anos)

>	1	>1
=	2	=1
<	3	<1
-	4	-

fig. 4.3.3-5

2 - Nº total de regras simples

$$3 \times 4 \times 3 = 36$$

3 - Cálculo do fator de grupo para cada condição.

C1 fator de grupo = 36/3 = 12

C2 fator de grupo = 12/4 = 3

C3 fator de grupo = 3/3 = 1

4 - Construir a matriz

C1 - 12 vezes >, =, < cada um

C2 - 3 vezes 1, 2, 3, 4 cada um

C3 - 1 vez >1, =1, < 1 cada um

ou seja:

C1	C2	C3
>	1	>1
>	1	=1
>	1	<1
>	2	>1
>	2	=1
>	2	<1
>	3	>1
>	3	=1
>	3	
>	1	
>	1	
=	1	
=	2	
=	2	
=	2	
=	3	
=	3	
=	3	

fig.4.3.3-6

(Cont.)

=	1
=	2
=	3
=	1
<	2
<	3
<	.
<	.
<	.
<	.
<	.
<	.
<	.
<	.
<	.
<	.
<	.

Checando se uma tabela está completa:

Digamos que o nº total de regras simples de uma tabela completa seja Vc(= 2^n sendo n condições) e o nº de regras simples que está sendo analisado, seja V'c (= $\Sigma\, 2^r$ sendo 2^r de cada regra individual onde r é o nº de símbolo'-').

Como vimos anteriormente o 2 significa o nº de valores que pode ter uma condição. Para o caso de tabela de entrada limitada o nº de valores para uma condição vale 2. Uma condição que tenha 3 valores por exemplo <, =, > o símbolo de indiferente significa 3^r.

Ex: Seja a seguinte tabela. Verificar se está completa.

C1 A=1

C2 B=2

C3 C= 3

C4 D=4

C5 E=5

C1	C2	C3	C4	C5	Nº de '-' = R	Nº de regras simples p/regra = 2^r
S	-	-	S	N	2	$2^2 = 4$
N	S	S	-	-	2	$2^2 = 4$
-	-	N	N	-	3	$2^3 = 8$
-	S	N	S	S	1	$2^1 = 2$
N	N	N	S	S	0	$2^0 = 1$

Cálculo:

O nº de regras simples para uma tabela completa de 5 condições

$rc = 2^5 = 32$

O nº de regras simples da tabela acima; igual a $r'c = \Sigma 2^r = 19$

Nº de regras ausentes $(rc - r'c) = 32 - 19 = 13$

13 regras simples representada pelo ELSE.

Qualquer caso que não seja coberto pelas regras é coberto pe-
regra ELSE.

Comparando os valores de rc e r'c podemos ter as seguintes con-
clusões:

Se r'c < rc tem ELSE
Se r'c = rc está completa
Se r'c > rc há redundancia ou contradição.

IV. 3.4 - Ambiguidade

O símbolo indiferente '-' é usado na matriz das regras para indicar que o valor da condição não produz efeito nas ações a serem tomadas. Mas há circunstâncias onde esta interpretação pode causar erros - por exmplo, em uma regra poderá mostrar que um determinado valor da condição seja maior que, menor e igual a outro valor ao mesmo tempo. Esta situação será vista agora. Essencialmente será vista a diferença entre ambiguidade aparente, e ambiguidade real.

Consideramos o seguinte exemplo

Ex:

C1	C2	A1	A2	A3
S	-	X		
-	S		X	
N	N			X

fig. 4.3.4-1

Observando a tabela existe uma contradição entre regra 1 e 2, se C1 for S e C2 for S as ações a serem tomadas são diferentes.

Digamos que substituimos C1 e C2 por condições reais:

C1	C2
S	-
-	S
N	N

C1 20% de aumento?
C2 gosta do trabalho.

A1	A2	A3
X		
	X	
		X

A1 fica 1 ano
A2 fica 2 anos
A3 vai embora

fig. 4.3.4-2

Se uma pessoa tiver 20% de aumento de salário e gosta do trabalho, o que ela faz?
Por um lado a regra 1 diz que ela deverá ficar 1 ano. Já na regra 2 diz que ela deverá ficar 2 meses. Neste caso a tabela é de ambiguidade real, porque as 2 condições são independentes.

Se fosse feita a seguinte pergunta: o que ela faria se tanto recebesse o aumento de 20% como gostasse do trabalho, ao mesmo tempo?

Voltando a analisar a nossa tabela poderia ficar da seguinte maneira:

C1	C2
S	S
S	N
N	S
N	N

C1 20% de aumento?
C2 gosta do trabalho.

A1	A2	A3
X		
X		
	X	
		X

A1 fica 1 ano
A2 fica 2 anos
A3 vai embora

fig. 4.3.4-3

Nesse caso a tabela poderia ser simplificada pela introdução de uma regra complexa.

C1	C2	A1	A2	A3
S	-	X		
N	S		X	
N	N			X

fig. 4.3.4-4

Como pode-se observar a ambiguidade desaparece, bastando apenas reanalizar a tabela.

Usando o mesmo formato da tabela da fig. 4.3.4-1, apenas substituindo C1 e C2 por condições dependentes, teríamos:

C1	X = 5?
	C2 X = 10?

A1	Processa
A2	Rejeita
A3	Erro

C1	C2	A1	A2	A3
S	-	X		
-	S		X	
N	N			X

fig. 4.3.4-5

Se X = 5, então não é preciso testar se X = 10 e vice- versa; as condições são dependentes ou seja: se X = 5 então não pode ser / ao mesmo tempo ser igual a 10. Neste caso temos uma ambiguidade aparente.

Finalmente, consideramos outro exemplo com condições dependentes: uma companhia tem dois tipos de atividade: estocar, agrupar manufaturados em armazéns, e fábricar; a qual tem de ser feito sobre encomenda. O procedimento de checar o crédito é dependente se uma ordem é para estoque ou fabricar. Um analista de sistema vai ao gerente de venda e durante a entrevista, descobre a política de vendas e registra na tabela a seguir:

C1	Ordem de fabricação?	
	C2 Ordem de estoque?	
S	-	
-	S	fig. 4.3.3-6

Com algumas outras condições e ações apropriadas. Neste estágio/ o analista pode concluir que se uma ordem é recebida ou é um pedido de estoque ou é um pedido de fabricação, não podendo ser ambos.

Os pedidos são recebidos, pelo funcionário de checar crédito, em um formulário pré-impresso o qual tem as palavras estoque ou fabricação, impressas no topo. O funcionário adota então o procedimento apropriado. Quando o analista intrevistar o empregado que checa o crédito, ele volta à tabela de decisões:

C1 Marcado com ordem de fabricação?	C2 Marcado com ordem de estoque?
S	S
S	N
N	S
N	N

Neste caso, o analista achou que frequentemente as ordens são re marcadas por outros funcionários que processaram antes do fiscal de crédito ter recebido - algumas vezes ambos, estoque e fabricação foram marcados, algumas vezes não.

Neste caso, a ordem é referenciada a processamento especial de checagem. Ambas as tabelas representam de uma forma precisa o procedimento a ser seguido. Isto depende da realidade das duas situações facilitadas. Se o analista voltar de sua entrevista, / com o funcionário que checa a crédito, com a seguinte tabela:

C1 Marcada c/ordem de fabricação?	C2 Marcada c/ ordem de estoque?
S	-
-	S
N	N

então esta tabela terá contida uma ambiguidade real.

As vezes o significado exato de '-' (indiferença) pode ser confusa. Algumas regras podem ser dadas:

1 - Se as condições são realmente independentes, indiferença significa 'independencia dos valores das condições'. Se existe uma contradição ou redundancia será um erro real na construção e a lógica deverá ser revista.

2 - Se as condições são dependentes, a relidade da situação precisa ser examinada para determinar se indifernça significa 'tanto faz'; uma contradição ou redundancia pode ser somente uma ambiguidade aparente.

IV.3.5 - Combinação das regras

Em muitas tabelas de decisões, pelo menos uma condição não afetará materialmente a ação a ser tomada em algumas regras. A formação de regras complexas através de regras simples pode ser um procedimento totalmente mecanico, além de diminuir o tamanho da tabela. Visualmente isto pode ser feito apontando qualquer / condição que não apareça afetar as ações tomadas. Porém em grandes tabelas isto é difícil. Um erro de combinação pode produzir/ uma tabela logicamente errada.

A combinação deverá ser aplicada a grupos pequenos de regras; / por exemplo, pares de regras sobre uma tabela de entradda limitada.

Vários estágios de tabela podem ser produzidos durante a combinação; primeiro a formação de regras complexas através das regras simples existentes, seguida através das regras simples existentes, seguida de várias combinações de regras complexas para / formar outras regras complexas.

Embora o procedimento para combinar regras é o mesmo, tanto para tabelas de entrada limitada como expandida, é melhor descrever separadamente, porque as tabelas de entrada limitada são restritas a 2 valores para cada condição, facilitando a formalização das regras.

Na combinação de regras de uma tabela de entrada limitada é melhor considerar duas regras de cada vez. Assim para cada par devem ser feitas as seguintes perguntas:

- As ações são identicas?
- Todos os valores da condição são os mesmos exceto em um par? (isto é todos os valores são S/S, N/N ou -/- sendo um S/N ou N/S?)

Se a resposta a ambas questões for afirmativa, então as duas regras podem ser combinadas. As condições com o par diferente é marcada com o símbolo de **indiferença** (-).

Ex:

C1	C2	C3	C4	A1
S	S	S	S	X
S	S	S	N	X

⟹

C1	C2	C3	C4	A1
S	S	S	-	X

Pode ser combinado porque as ações são as mesmas, todas condições são S/S salvo a última. Esta condição pode ser substituída por um símbolo de indiferença '-' e as 2 regras combinadas.

EX:

C1	C2	C3	C4	C5	A1
S	N	S	N	S	X
S	N	N	N	S	

⟹

C1	C2	C3	C4	C5	A1
S	N	-	N	S	X

Pode também ser combinada, pois as ações são as mesmas e todos os pares de condição são S/N ou N/N com excessão da condição C3.

Ex:	C1	C2	C3	C4	C5	A1
	N	S	N	S	S	X
	S	S	N	N	N	X

Não pode ser combinada, embora as ações sejam as mesmas, há dois pares se condição N/S e S/N.

Ex:	C1	C2	C3	C4	C5	A1	A2
	S	S	S	S	S	X	
	S	S	S	S	N		X

Não pode ser combinada, porque as ações não são as mesmas.

Ex:	C1	C2	C3	C4	C5	A1
	S	N	S	S	-	X
	S	N	S	N	-	X

⇒

C1	C2	C3	C4	C5	A1
S	N	S	-	-	X

Pode ser combinada porque as ações são as mesmas e só há um par diferente S/N.

Ex:

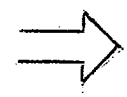
C1	C2	C3	C4	C5	A1
S	N	S	S	-	X
S	N	-	N	-	X

Não pode ser combinada porque há 2 pares diferentes S/- e S/N.

Na tabela de entrada expandida, as regras são as mesmas, porém a combinação de mais de 2 regras pode ocorrer.

Ex:

C1	C2	C3	A1
>	1	S	X
>	2	S	X
>	3	S	X



C1	C2	C3	A1
>	-	S	X

Pode ser combinada pois tem as mesmas ações e apenas uma condição há diferença de valores.

Ex:

C1	C2	C3	A1
=	2	N	X
=	3	S	X

Não pode ser combinada, pois há duas condições de valores diferentes.

Resumindo, as regras para serem combinadas, teremos:

C1 ações identicas?
C2 todas as condições tem o mesmo valor exceto 1 ?
C3 cond. c/ valores dif. rep. todos os valores p/ a-la condição?

A1 pode ser combinada
A2 não pode ser combinado.

C1	C2	C3	A1	A2
S	S	S	X	
S	S	N		X
S	N	S		X
S	N	N		X
N	S	S		X
N	S	N		X
N	N	S		X
N	N	N		X

IV.3.6 - Regra 'ELSE'

A regra ELSE cobre as seguintes situações logicas bem distintas.

- A regra ELSE recapitula todas as situações logicas possíveis que não foram explicitas nas regras precedentes.
- A regra ELSE se acrescenta as outras regras previstas para conservar o controle da continuação do programa nos casos teoricamente impossíveis (erros).

Por exemplo é preciso um codigo de valor ∅ ou 1, se é encontrado um codigo de valor 2, a regra ELSE pode enviar a uma tabela que indique erro.

Ex:

								A5 Erro
C1	C2	C3	C4	A1	A2	A3	A4	A5
S	S	S	S	X1		X2		
S	S	S	N		X1		X3	
S	N	S	S	X1			X2	
S	N	S	N	X1	X2		X3	
ELSE								X

fig. 4.3.6-1

IV. 4 - Encadeamento de tabelas

O encadeamento de várias tabelas de decisão é na verdade a maneira de representar as situações reais. Embora todo programa possa ser representado por apenas uma tabela, é praticamente impossível devido ao tamanho resultante da tabela após convertida.

Antes de utilizar as variáveis que figuram nas condições é/ preciso lê-las, o que se faz em geral por meio de uma tabela incondicional.

A análise de um problema pouco complexo conduz imediatamente, a distinguir um certo número de funções, sem relaciona-las. Não é necessário tratá-las na mesma tabela. Sempre é preferível subdividir uma tabela grande que contenha muitas condições e ações em várias tabelas pequenas de mais fácil compreensão.

Pode-se distinguir dois tipos de encadeamento tabela aberta e tabela fechada.

A tabela aberta se alcança por uma transferência incondicional (uma instrução desvio (transição) executada antes dela). Esta transferência não devolve o controle a tabela precedente.

Ex:

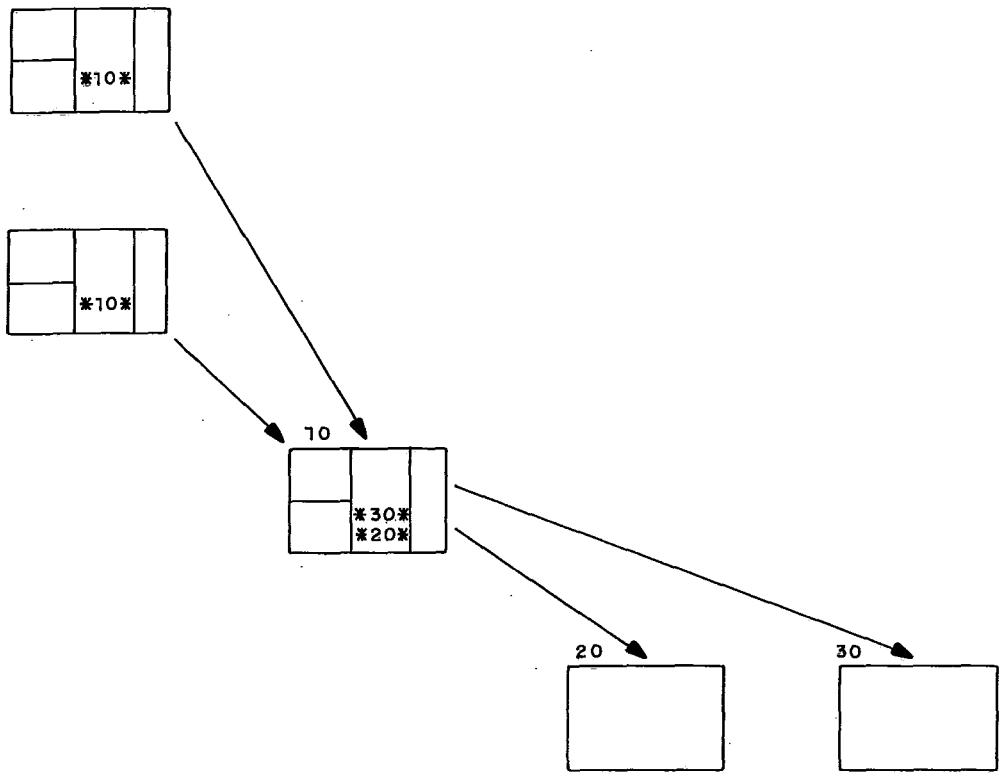


fig. 4.4-1

As tabelas 10, 20 e 30 são tabelas abertas na fig. 4.4-1

A tabela fechada, ao contrário, devolve o controle a tabela precedente, é alcançada pela instrução CALL. Ao final o controle é devolvido a tabela precedente por meio de uma instrução retorno:

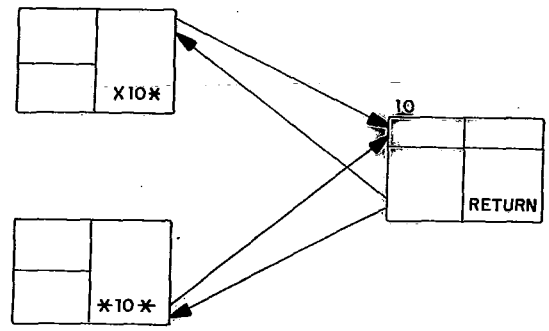


fig. 4.4-2

Para que o encadeamento de tabelas funcione, é preciso respeitar as seguintes convenções:

- Cada tabela é indentificada por um número que servirá de referencia. Não é possível executar somente uma parte de uma tabela, e sim executar toda a tabela.

- Cada regra de decisão deverá indicar como se continua a lógica. Quando em uma tabela pode-se levar a mais de uma tabela, faz-se então uma ramificação contendo vários desvios (transição)

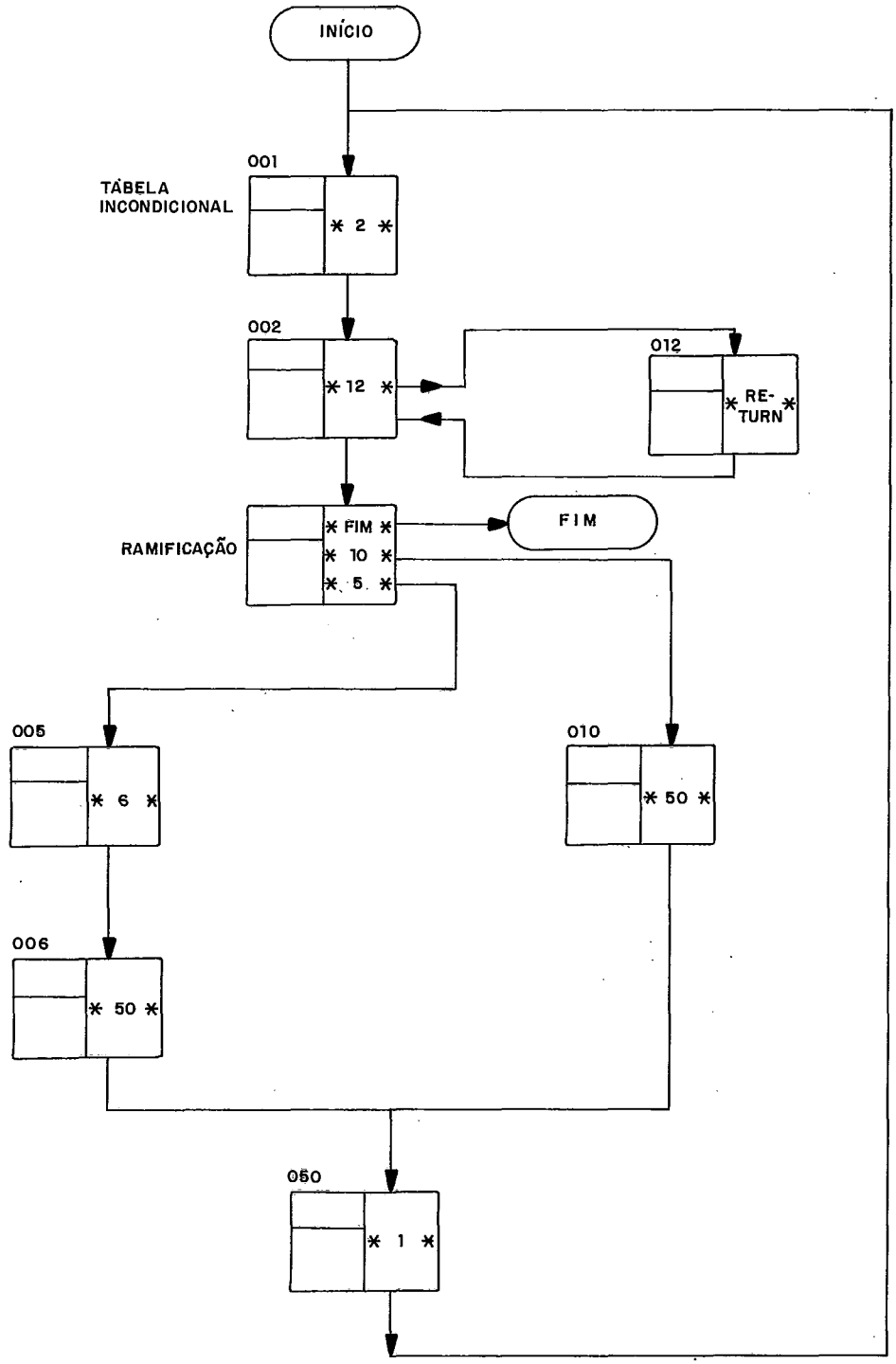

INÍCIO
001
TÁBELA
INCONDICIONAL
* 2 *
002
* 12 *
012
* RE-
TURN *
RAMIFICAÇÃO
* FIM *
* 10 *
* 5 *
FIM
005
* 6 *
010
* 50 *
006
* 50 *
050
* 1 *

fig. 4.4-3

Pode-se construir encadeamento de tabelas de forma tal que diminua o nº de testes efetuados sobre as condições possíveis. Esta segmentação logica de tabelas permite reduzir o tempo de execução do prógrama.

Seja a seguinte tabela

C1	C2	C3	A1	A2	A3	A4
S	S	S	X1		X2	
S	S	N	X1			X2
S	N	S	X1		X2	
S	N	N	X1			X2
N	S	S		X1		X2
N	S	N		X1	X2	
N	N	S		X1		X2
N	N	N		X1		X2

fig. 4.4-4

A ação A1 será executada quando for verdade e A2 será executada / quando C1 for falsa. Assim é possível reagrupar a condição C1 e as ações A1 e A2.

Pode-se observar ainda que a condição C2 é indiferente para as quatro primeiras regras. E nelas a ação A3 é executada quando C3 é verdade se C2 é verdadeira ou falsa. Inversamente será para A4.

Por último, as quatro últimas colunas (em que C1 é falso). A3 será executada para os outros 3 casos.

Construindo agora encadeamento da tabela anterior, ficaria:

TAB1

C1		A1	A2
S	* TAB2 *	X	
N	* TAB3 *		X

TAB2

C3		A3	A4
S		X	
N			X

TAB3

C2	C3		A3	A4
S	N		X	
ELSE				X

FIG. 4.4-5

Fazendo o fluxograma correspondente a 1ª tabela e a 2ª tabela, teríamos:

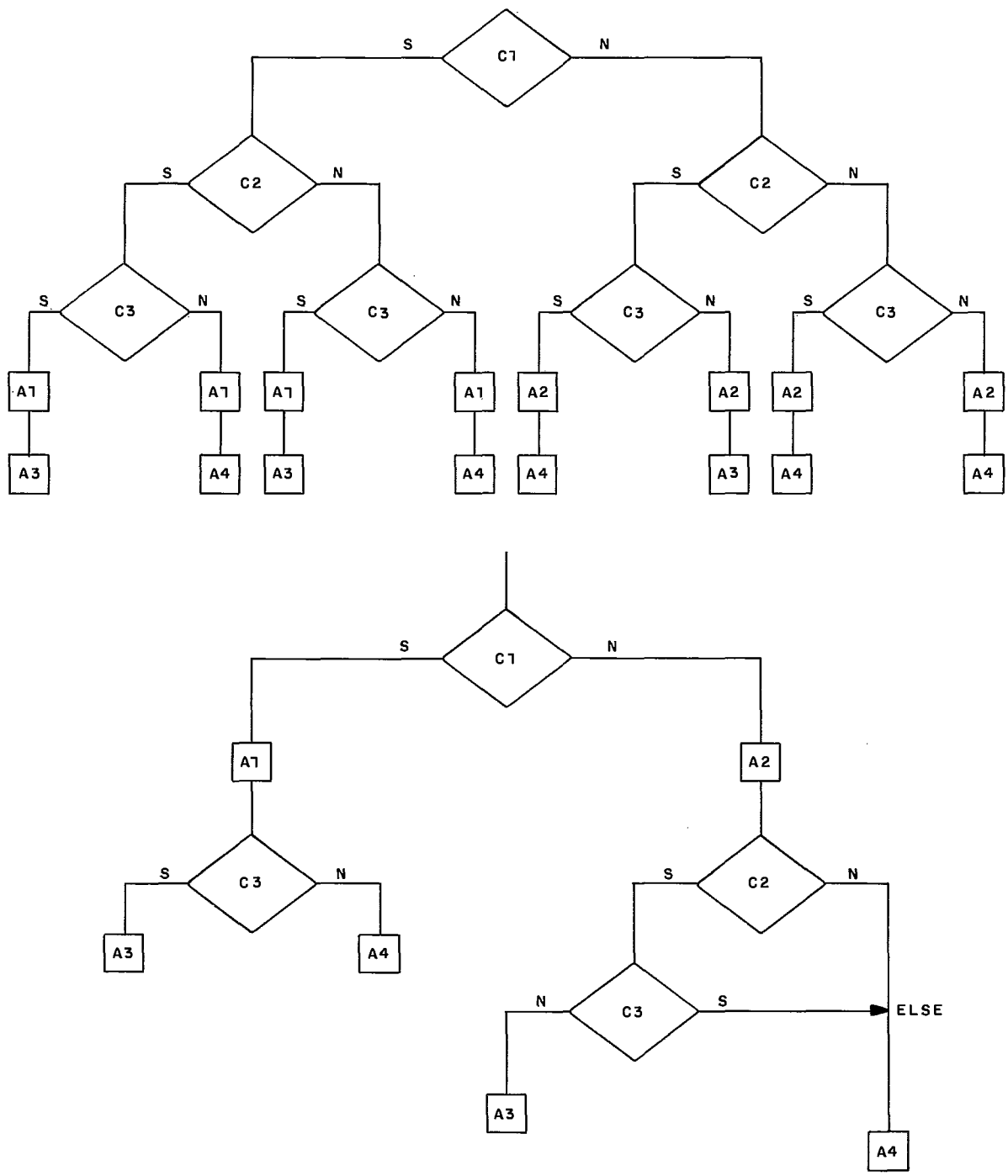


fig. 4.4-6

IV. 5 - Otimização

A palavra otimização denota o processo que resulta a utilização mínima de memória ou o mínimo de tempo de execução.

As técnicas de otimização de memória garantem que as regras são analisadas baseando-se nas condições expressas, para minimizar o nº de testes (ou seja o nº de IF).

Otimização de tempo é o menor caminho para executar uma determinada regra. Neste tipo de otimização, é considerada a frequencia relativa com que várias condições precisam ocorrer por um nº grande de transações. Quando um grande volume de transações harmoniza-se com esta distribuição de frequencia, o nº de testes executados é mínimo, reduzindo também o tempo de execução.

Exemplos de otimização

Ambos os tipos de otimização operam sobre os valores das condições. A estrutura logica representada pelas condições, as quais permitem a solução de regras e execução de ações associadas com aquelas regras selecionadas, pode ser representada de diversas maneiras.

Vejamos agora a seguinte tabela:

	C1	C2	C3
	Y = Ø?	X = 1?	X = 2?
Regra 1	-	N	S
Regra 2	S	S	N
Regra 3	N	S	N

fig. 4.5-1

Vamos representar aqui 3 maneiras diferentes de representar a tabela, em fluxograma:

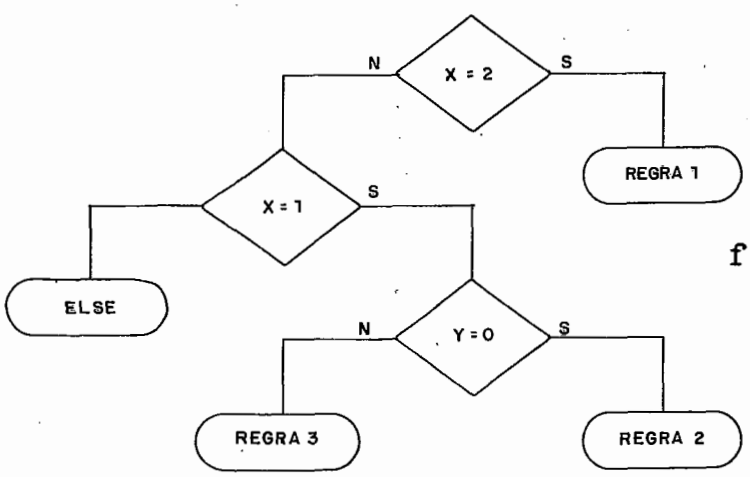


fig. 4.5-2

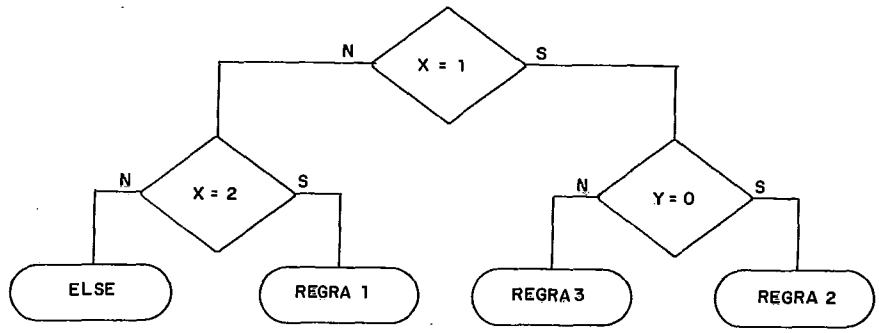


fig. 4.5-3

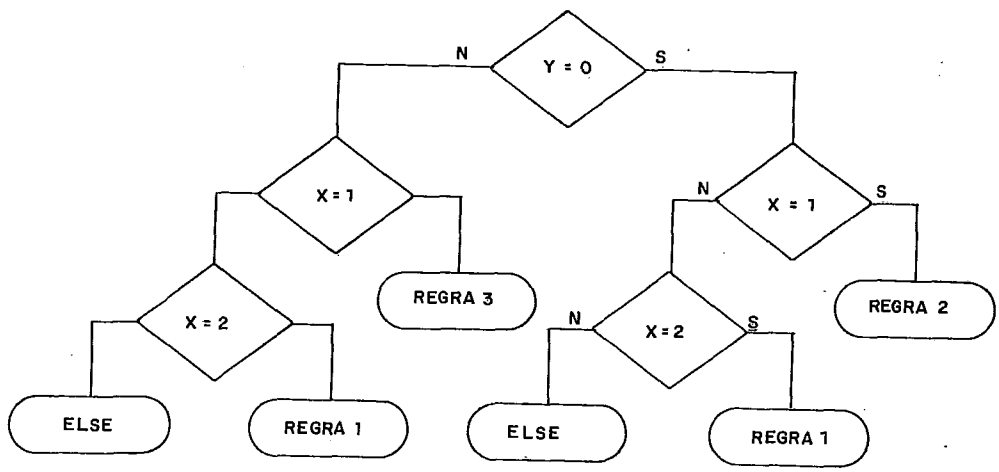


fig. 4.5-4

Façamos então a otimização de tempo. Para ilustrar, calcularemos o total de comparação necessária para identificar a regra 1 nas figs. 4.5-2, 4.5-3 e 4.5-4.

Na fig. 4.5-2 a regra 1, é selecionada sómente depois de um teste, enquanto na fig. 4.5-3 não é selecionada antes de 2 testes. Na fig. 4.5-4 há 3 maneiras de selecionar a regra 1, todos requerem 3 comparações.

Se houver 100 comparações, sendo 90 para a regra 1, 7 para a regra 2 e 3 para a regra 3, o resultado obtido é visto na fig. 4.5-5.

Fig.	Regra	Nº de comparações	% Frequência	Nº de comp. x Frequência
4.5-1	1	1	0.90	0.90
4.5-1	2	3	0.07	0.21
4.5-1	3	3	0.03	0.09
4.5-1	(total)			1.20 comparações
4.5-2	1	2	0.90	1.80
4.5-2	2	2	0.07	0.14
4.5-2	3	2	0.03	0.06
4.5-2	(total)			2.00 comparações
4.5-3	1	3	0.90	2.70
4.5-3	2	2	0.07	0.14
4.5-3	3	2	0.03	0.06
4.5-3	(total)			2. 90 comparações

fig. 4.5-5

Observando-se a fig. 4.5-4 chegamos a conclusão que a estrutura da fig. 4.5-1 tem menor custo, ou seja é a mais otimizada em relação as outras duas e a regra de maior ocorrencia é a regra 3

Agora se mudarmos a ocorrência da regra 1 de 0.90 para 0.07 e da regra 2 de 0.07 para 0.90, temos o seguinte quadro.

Fig.	Regra	Nº de comparações	% Frequencia	Nº de comp. x Frequencia
4.5-2	1	1	0.70	0.70
4.5-2	2	3	0.90	2.70
4.5-2	3	3	0.03	0.09
4.5-2	(total)			2.49
4.5-3	1	2	0.70	1.40
4.5-3	2	2	0.90	1.80
4.5-3	3	2	0.03	0.06
4.5-3	(total)			3.26
4.5-4	1	3	0.07	0.21
4.5-4	2	2	0.90	1.80
4.5-4	3	2	0.03	0.06
4.5-4	(total)			2.07

fig. 4.5-6

Com essa troca de frequencia vemos que a estrutura da fig. 4.5-3 tem o maior custo,sendo a regra 2 de maior ocorrencia.

A frequencia relativa torna-se portanto um fator importante para a otimização no tempo de execução.

A otimização de mémoria pode ser considerada como otimização do tempo de execução, dando igual probabilidade de sucesso para todas as regras. O exemplo da fig. 4.5-3 recai nesse caso, havendo otimização de memória.

O processo de otimização esta ligado diretamente a decomposição / de uma tabela de decisão e manipulada por elaborados algoritimos. O usuário não precisa se envolver com os mecanismos de otimização, caso exista pré-processadores de tabelas de decisões.

IV. 6 - Inicialização

Muitos procedimentos tem uma serie de ações as quais são chamadas, somente, no começo do procedimento manual, isto pode ser, apanhar arquivos, preencher parte da identificação de um impresso previamente para preencher uma lista, etc. Em programas de computadores as ações iniciais podem ser: abertura e fechamento de arquivos, limpar areas, etc. Estas inicializações de ações podem ser mostradas em tabela de decisões através da regra zero.

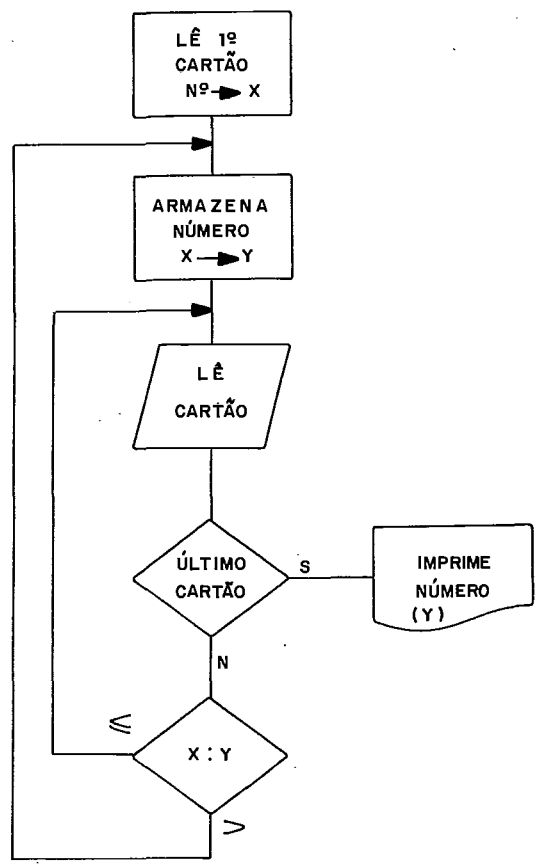
Ex:

TAB1..

	A1	open arq1		
		A2	A = ∅	
			A3	B = ∅
				A4 X(1) = 1
	A1	A2	A3	A4
* TAB2 *	X	X	X	X

IV. 7 - Repetição

Repetição é simplismente um looping condicional. O fluxograma da fig. 4.7-1 é um problema de logica simples.



O mesmo problema é mostrado na fig. 4.7-2

TAB1..

		A1	le cartão(x)
			A2 X → Y
		A1	A2
	* TAB2 *	X	X

TAB2..

C1	ultimo cartão?	A1	imprime Y	
	C2 X > Y?		A2 X → Y	
				A3 le cartão
C1	C2	A1	A2	A3
S	-	X		
N	S		X	X
N	N			X

fig. 4.7-3

IV. 8 - Problema

Um conjunto de transações são aplicadas a um arquivo mestre. As transações podem ser:

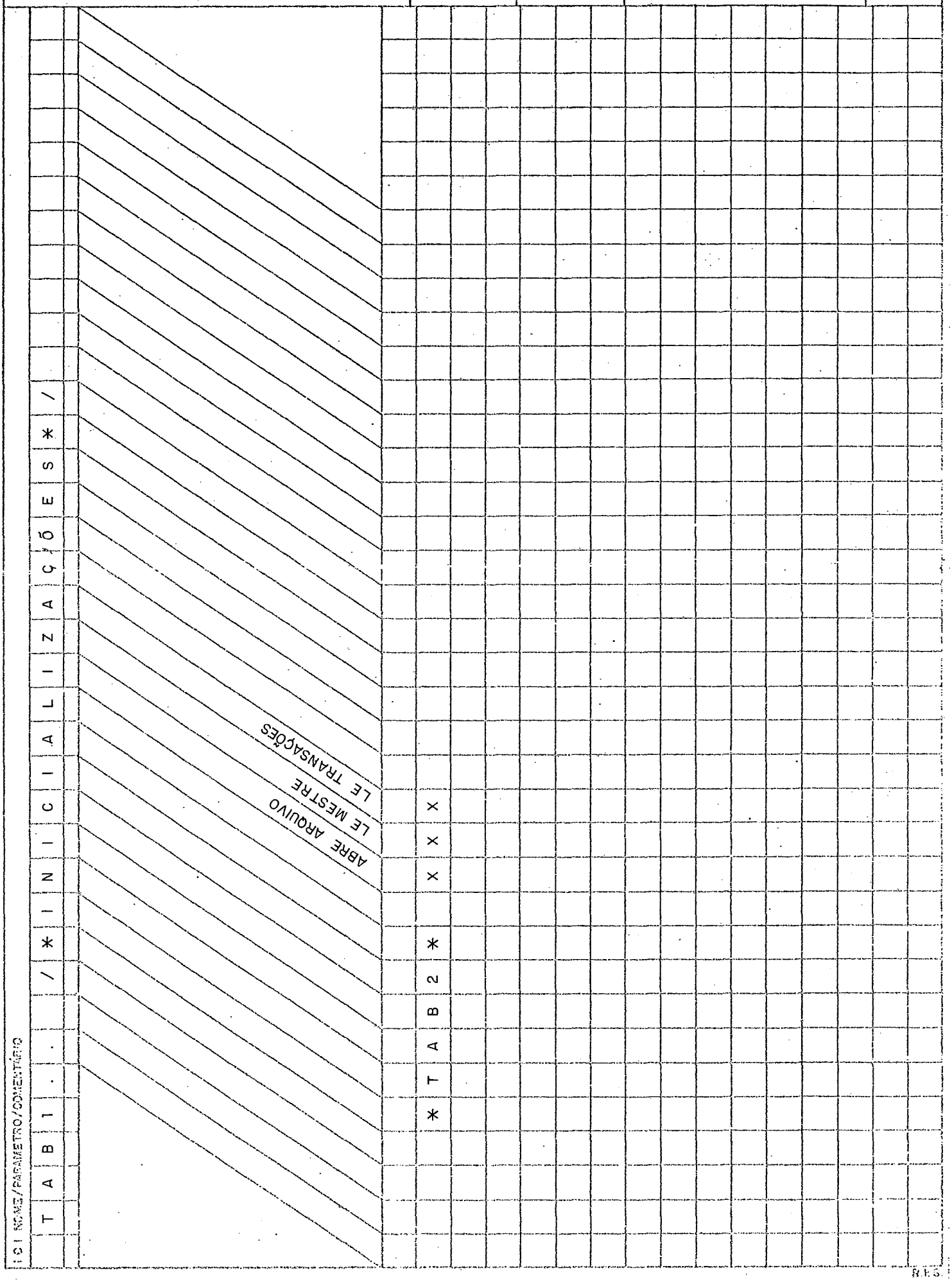
- alteração de campo no registro	código Ø1
- inclusão	código Ø2
- alteração no registro completo	código Ø3

As transações são classificadas pela chave do registro. O último registro no arquivo tem 6 asteriscos no campo da chave. O arquivo mestre está classificado pela chave do registro. O arquivo mestre é atualizado pelo processo serial(fita). Os dados para este problema são: chave, código de transação e data de origem, e sendo este último somente usado em alteração.

Para simplificar é assumido que todas as entradas foram unificadas em um processamento anterior tal que o registro mestre não há duplicata de chave.

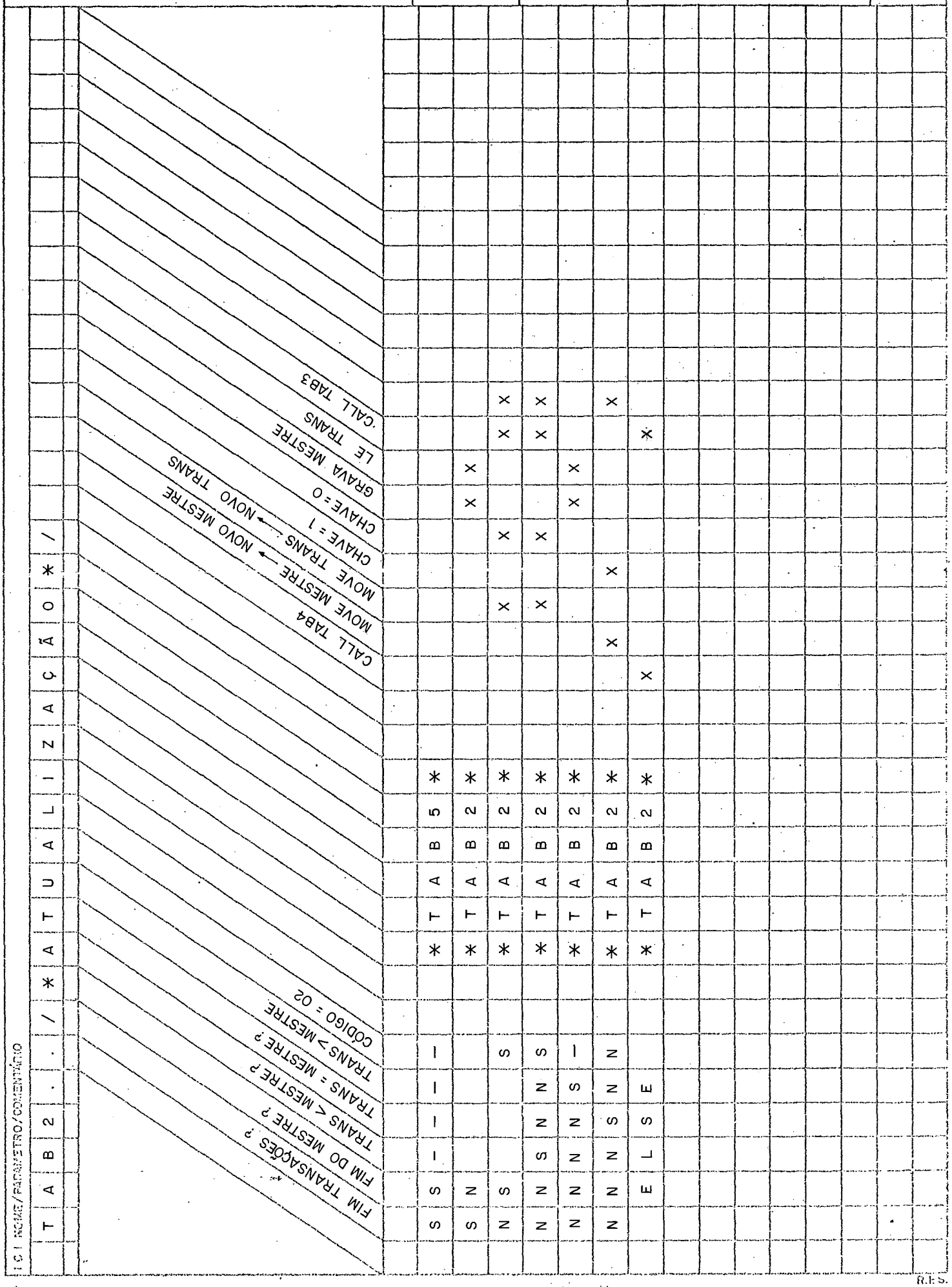
A lógica básica é especificada em uma tabela para um programador/ que irá detalhar a lógica do programa.

TÍTULO	SISTEMA	DOCUMENTO	NOME	FOLHA



R.F.S.

TÍTULO	SISTEMA	DOCUMENTO	NOME	FOLHA



R.E.S.

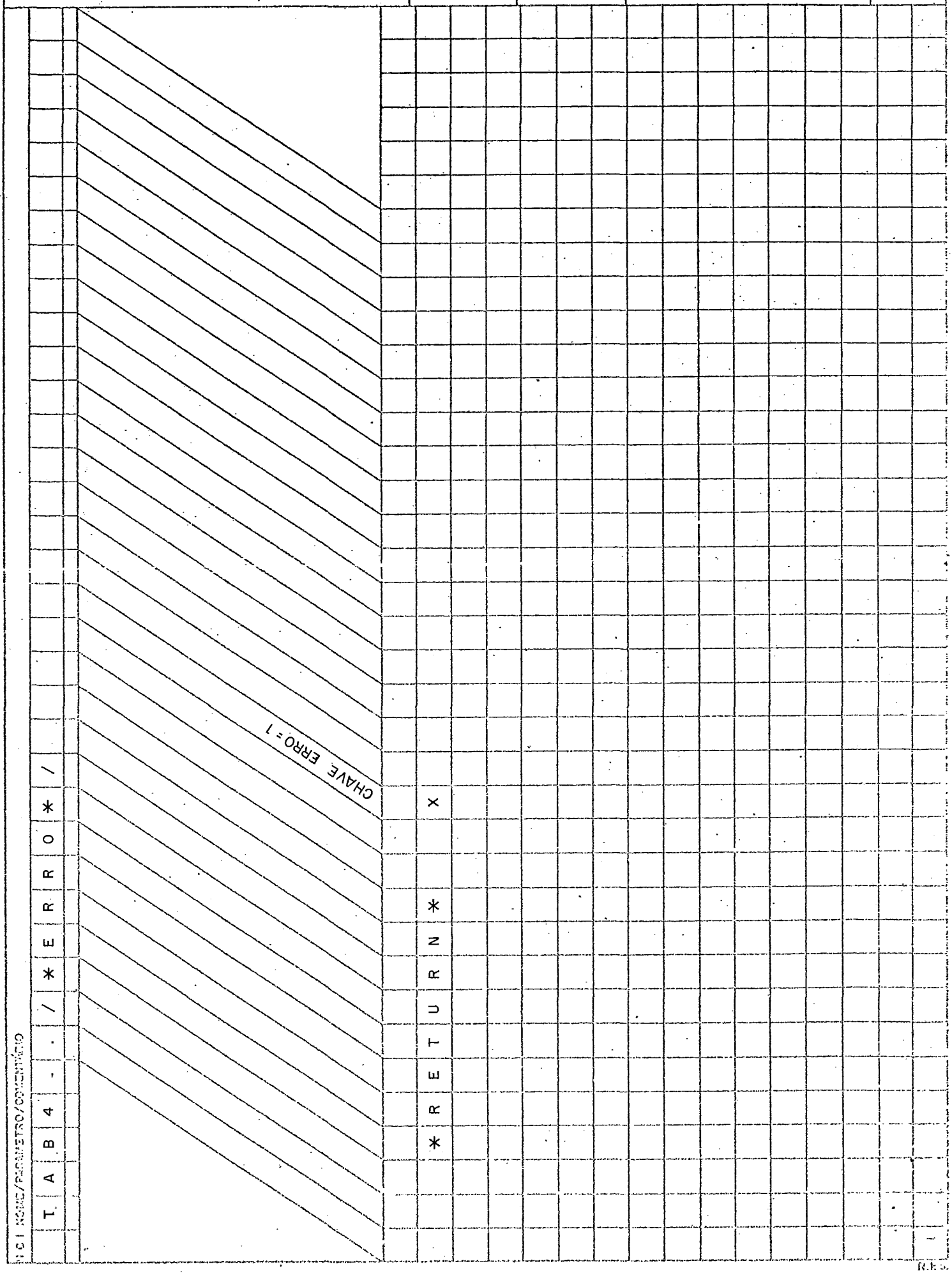
TÍTULO	SISTEMA	DOCUMENTO	NOME	FOLHA

NOME/PARAMETRO/COMENTÁRIO

TAB3.. /*ALTERAÇÃO*/

CHAVE = 0 ?	CHAVE = 1 ?	CÓDIGO = Ø1 ?	DATA TRANS DATA MESTRE	CÓDIGO = 03 ?		MOVE CAMPOS ALT. → NOVO MESTRE	MOVE TRANS → NOVO MESTRE	INCREMENTA CONTADOR TIPO Ø1	INCREMENTA CONTADOR TIPO Ø2	INCREMENTA CONTADOR TIPO Ø3	INCR. CONTADOR TIPO 4 - ERRO	CHAVE ERRO = 1	IMPRIME TRANS	IMPRIME MESTRE	IMP. NOVO MESTRE	GRAVA NOVO MESTRE	LE MESTRE	LE TRANS
S	N	—	—	—	*RETURN*				X								X	
N	S	S	S	N	*RETURN*	X		X					X	X	X	X	X	X
N	S	N	S	S	*RETURN*		X			X				X	X	X	X	X
N	S	S	N	N	*RETURN*											X	X	X
N	S	N	N	S	*RETURN*											X	X	X
	E	L	S	E	*RETURN*						X	X	X			X	X	X

TÍTULO	SISTEMA	DOCUMENTO	NOME	FOLHA



R.E.S.

TÍTULO

SISTEMA

DOCUMENTO

NOME

FOLHA

NOME/PARAMETRO/COMENTÁRIO

TAB5../*IMPRIME ERRO*/

FIM TRANS ?	FIM MESTRE ?	TRANS < MESTRE ?	TRANS = MESTRE ?	CÓDIGO = 02 ?		IMP. REG. ERRADO TRANS TIPO 01	INC. CONTADOR ERRO TIPO 01	IMP. REG ERRADO TRANS TIPO 02	INC. CONTADOR ERRO TIPO 02	IMP. REG. ERRADO TRANS TIPO 03	INC. CONTADOR ERRO TIPO 03	IMP. REG. MESTRE	PARE
N	S	—	—	N	*RETURN*	X	X						
N	N	S	N	N	*RETURN*			X	X				
N	N	N	S	S	*RETURN*					X	X	X	
	E	L	S	E	*RETURN*								X

TÍTULO	SISTEMA	DOCUMENTO	NOME	FOLHA

TAB 6	FIM DA TRANSAÇÃO ?	FIM DO MESTRE ?	CHAVE - ERRO = 1 ?		FECHA ARQUIVOS	IMP. CONTADOR ERRO	IMP. CONTADOR DE ATUALIZAÇÕES	PARE	DUMP	PARE - INCOMPLETO
	N	N	E					X	X	
	N	N	L					X		
	N	N	S		X					
	S	N	S		X	X				
			E	*	X					
				R						

ICI RCA/PARAMETRO/COMERCIAL

RES

CAPÍTULO V

COMO CONSTRUIR UMA TABELA

CAP. V - COMO CONSTRUIR UMA TABELA

V.1 - Técnicas básicas

V.1.1 - Clássica

A técnica clássica é baseada no desenvolvimento de uma matriz representando tôdas as regras simples para as condições. A matriz completa deverá então ser reduzida, se possível para poucas regras pela combinação de regras simples, para formar regras complexas. O procedimento é definido como nº distinto de passos / os quais podem ser seguidos mecanicamente. A tabela final produzida pelo método deverá estar logicamente correta e relativamente reduzida. O método requer 5 passos:

1 - Completar a origem das condições.

Listar todas as condições, quando identificadas.

As condições devem estar na sequencia da mais importante para a menos importante.

As condições devem ser sempre que possível, independente ou mutuamente exclusivas.

Evitar condições negativas.

As condições de tabelas de entrada limitada são perguntas, enquanto na de entrada expandida são declarações sem valor.

2 - Completar a origem das ações

Listar as ações na sequencia que elas aparecerem.

As ações devem ser declarações imperativas.

3 - Matriz das condições

Formar a matriz com todas as combinações possíveis das condições, conforme metodos vistos no capítulo anterior.

4 - Combinação das regras

Combinar as regras simples de acordo com as instruções da tabela da fig. 5.1.1-1

C1 Ações são identicas?

C2 Tôdas as cond. a excessão de uma tem o mesmo valor?

C3 Cond. c/ valores dif. rep. todos os valores p/ aquela condição?

A1 Pode ser combinado.

A2 Não pode ser combinado.

C1	C2	C3	A1	A2
S	S	S	X	
S	S	N		X
S	N	S		X
S	N	N		X
N	S	S		X
N	S	N		X
N	N	S		X
N	N	N		X

fig. 5.1.1-1

Combinar as regras em pequenos grupos.

5 - Verificação final da tabela

O cheque pode ser aplicado a duas categorias: conteúdo e estrutura.

O cheque do conteúdo deverá garantir que as ações associadas com cada regra simples estão corretas. O cheque da estrutura final da tabela, é uma tentativa para garantir que a tabela não contém ambiguidade, isto é, contradições ou redundancias. Se ela existe, identificar se é somente aparente ou real.

Depois é feito uma verificação ao conteúdo - independencia das condições, sequencia das condições e assim por diante.

Dois tipos de cheque podem ser feitos sobre a estrutura lógica: uma verificação aritmética e uma verificação de bifurcação.

Verificação aritmética

A verificação aritmética é um primeiro passo na verificação. Se o nº de regras simples representada pela tabela final não corresponde ao nº de regras simples, originalmente especificada na matriz de regras, significa que existe uma ambiguidade.

Ex:	C1	C2	C3	A1	A2
	S	S	S	X	
	S	S	N		X
	S	N	N		X
	S	N	-	X	
	N	-	-	X	

fig. 5.1.1-2

C1 contém 1 regra simples

C2 contém 2 regras simples (cada '-' vale 2)

C3 contém 4 regras simples

Total 7 regras simples

A matriz completa teria que ter 2^3 = 8 regras simples. A tabela então contem um erro, pois há uma contradição entre as regras 3 e 4, ou seja, duas regras idênticas dando ações diferentes.

	C1	C2	C3	A1	A2		C1	C2	C3	A1	A2
regra 3	S	N	N		X	⇨	S	N	N		X
regra 4	S	N	-	X			S	N	N	X	
							S	N	S	X	

fig. 5.1.1-3

Supondo as condições independentes a ambiguidade é real. Uma possível correção seria transformar a regra 4 em:

	C1	C2	C3	A1	A2
regra 4	S	N	S	X	

fig. 5.1.1-4

Observando outro exemplo, temos:

	C1	C2	C3	C4	A1	A2	A3
4 regras	S	-	-	S	X		X
2 regras	S	S	-	S	X	X	X
1 regra	S	S	N	S	X		X
1 regra	S	S	N	N		X	
2 regras	N	N	S	-	X	X	
2 regras	N	N	N	-		X	X
4 regras	N	S	-	-	X		X

16 regras

fig. 5.1.1-5

Embora o cheque aritmético não detetou erro, pois existem 16 regras simples para 4 condições, há uma ambiguidade entre as regras 1, 2, 3.

É possível ter compensação de erro como foi visto. Uma ambiguidade é assim determinada se há pelo menos 2 regras as quais não são mutuamente exclusivas. Um cheque aritmético é um bom indicador porém não deve ser feito exclusivamente.

Verificação bifurcação

O ponto de partida é detetar se uma regra está incluída em outra regra. Tendo detetado tal caso, as ações são examinadas para ver se é uma contradição ou uma redundancia. As condições que causaram ambiguidade deverão ser examinadas para ver se a ambiguidade é real ou aparente. O erro pode então ser corrigido voltando ao problema inicial. Em muitas tabelas (especialmente tabelas de entrada limitada) de tamanho considerável, um cheque de bifurcação pode apresentar problemas. Tais tabelas, entretanto, podem ser divididas em subtabelas, sendo cada subtabela definida por uma condição principal. Ocasionalmente, a resequenciação de condições pode mostrar alguns erros. A tabela mostrada na fig. 5.1.1-6, por exemplo, há numerosos erros. A maneira mais facil de checar tal tabela é resequenciar as condições, colocando as condições com menor nº de símbolos '-' a esquerda da tabela. Estas são as condições mais importantes, sendo que as menos importantes ficam a direita da tabela. A tabela pode ser então subdividida em subtabelas, sendo estas definidas pelas condições mais importantes.

Uso da técnica clássica

A técnica clássica consome tempo, porque cada passo é visto em detalhe como meticuloso cheque. Esta técnica será somente usada se o problema é uma descrição de um procedimento já estabelecido.

A vantagem da técnica clássica, está baseada no fato que é um procedimento passo a passo semimecanico. Atalhos geralmente levará a tabela final.

Em alguns casos, problema de lógica simples, não deve usar a técnica clássica pois o tempo gasto no passo a passo não vale a pena.

Tabelas grandes são muito difíceis de manipular pela técnica clássica.

C1	C2	C3	C4	C5	C6	C7	A1	A2	A3	A4	A5	A6
S	S	-	-	-	-	-	X		X		X	X
S	N	S	S	S	S	-	X		X	X	X	X
S	N	S	S	S	N	S	X	X		X	X	X
S	N	S	S	S	N	N	X	X		X	X	X
S	N	S	S	N	S	S	X		X		X	X
S	N	S	S	N	S	N	X		X		X	X
S	N	S	S	N	N	S	X	X		X	X	X
S	N	S	S	N	N	N	X	X		X	X	X
S	N	S	N	S	-	S	X	X	X		X	X
S	N	S	N	S	-	N	X	X		X	X	X
S	N	S	N	N	-	S	X	X	X	X	X	
S	N	S	N	N	-	N	X	X	X	X	X	X
S	N	N	-	S	S	S	X	X	X			X
S	N	N	-	S	S	N	X	X	X	X		X
S	N	N	-	S	N	-	X		X	X		X
S	N	N	-	N	S	S	X		X		X	
S	N	N	-	N	S	N	X	X	X	X	X	
S	N	N	-	N	N	-	X	X		X	X	
N	S	-	S	-	-	S	X	X		X	X	X
N	S	-	S	-	-	N	X	X			X	X
N	-	-	S	-	S	S	X	X		X	X	X
N	-	-	S	-	S	N	X	X			X	X
N	S	-	N	-	-	S	X	X	X	X		X
N	S	-	N	-	-	N	X	X	X			X
N	-	-	N	-	S	S	X	X	X	X		X
N	-	-	N	-	S	N	X	X	X			X
N	N	-	S	-	N	S	X	X	X	X	X	
N	N	-	S	-	N	N	X		X		X	
N	N	-	N	-	N	S	X	X	X	X	X	
N	N	-	N	-	N	N	X	X				X

fig. 5.1.1-6

Há outras técnicas para desenvolver tabelas de decisão, porém estas tendem a ser variações de dois métodos aqui descritos. Alguns melhoramentos podem ser acrescentados como a regra ELSE, tabelas encadeadas, inicialização. Estas técnicas já foram apresentadas / em capítulos anteriores.

V.1.2 - Desenvolvimento progressivo das regras

O desenvolvimento progressivo das regras é baseado na técnica de preparação de fluxograma. Enquanto a técnica clássica requer que tôdas as combinações de condições sejam definidas, a técnica de desenvolvimento progressivo de regras requer que as condições sejam escritas à medida que elas sejam identificadas. As regras para a construção da tabela através desta técnica, são as seguintes:

1. Considere uma condição verdadeira(s).
2. Considere as outras condições como a condição verdadeira até achar uma ação.
3. Registre as condições e ações.
4. Negue a última condição da sequencia.
5. Volte ao passo 2, por quanto a tabela estiver incompleta.

Ex: Seja o seguinte problema:

'Um determinado empregado de uma firma pedirá demissão se não conseguir promoção e se não conseguir um trabalho extra. Caso consiga promoção e o trabalho não seja interessante pedirá demissão' Montar uma tabela de decisão para a descrição acima.

1. Considere a condição: promovido (cargo mais elevado)
2. Considere o caso positivo:

fig. 4.1.2-1

3. Considere o caso positivo para todas as outras condições.

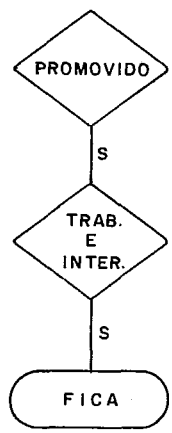


fig. 4.1.2-2

4. Negar a última condição e repetir o passo 2.

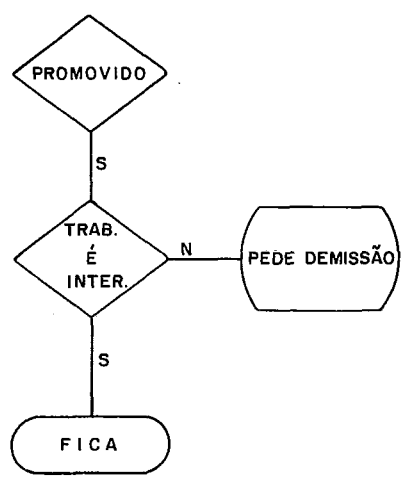


fig. 4.1.2-3

5. Negar a penúltima condição e repetir o passo 2.

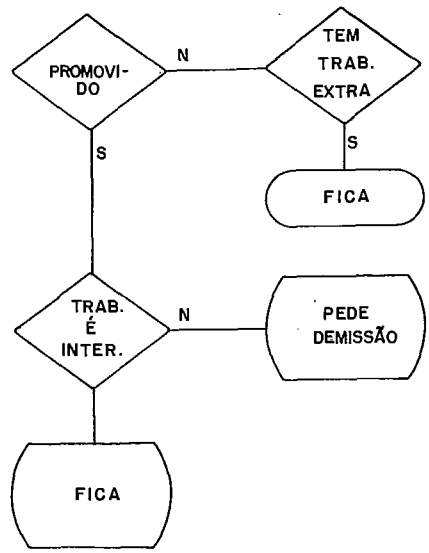


fig. 4.1.2-4

6. Considerar o caso negativo, para a condição que não foi testada negativamente e voltar ao passo 2.

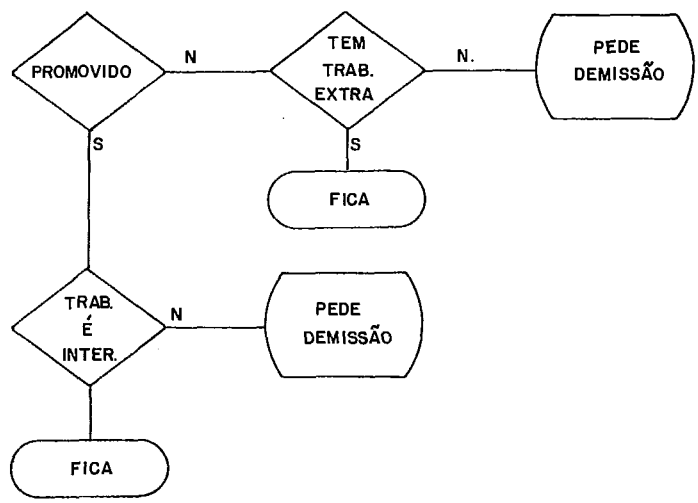


fig. 4.1.2-5

O mesmo processo é mostrado agora no desenvolvimento.

passo 1

C1	C2	FICA
S	S	X

passo 2

C1	C2	FICA	PEDE DEMISSÃO
S	S	X	
S	N		X

passo 3

C1	C2	FICA	PEDE DEMISSÃO
S	S	X	
S	N		X
N	S	X	

passo 4

C1	C2	FICA	PEDE DEMISSÃO
S	S	X	
S	N		X
N	S	X	
N	N		X

Ao terminar a tabela, faço uma verificação aritmética ou bifurcação, já descritos anteriormente.

cheque aritmético

regra 1	1
regra 2	1
regra 3	1
regra 4	1
total	4 regras simples

tabela completa: $2^2 = 4$ regras simples.

Uso do desenvolvimento progressivo da regra

É uma boa alternativa para a técnica clássica. A lógica é continuadamente revista quando a tabela é preparada. É muito utilizada para fazer entrevistas e checar o próprio entrevistador.

Se há uma ambiguidade na tabela, é mais provável que seja real do que aparente.

Uma quantidade maior de condições pode ser melhor manipulada/ do que na técnica clássica.

V.2 - PROBLEMAS

1. O texto abaixo descreve o procedimento de controle de crédito em uma companhia: Construa uma tabela de decisão do procedimento do empregado. A tabela deverá ser de entrada limitada e preparada pela técnica clássica.

'A COMPANHIA 'X' dá descontos a certos fregueses quando as faturas são preparadas. Um desconto de 5% é dado para todos os pedidos superiores ou iguais a Cr$ 1.000,00. Pedidos feitos por clientes comerciantes são concedidos um desconto de 10% independente do valor do pedido. Um cliente comerciante está também autorizado a um desconto adicional de 5% sobre todos os pedidos, desde que estes pertencem a associação de comerciantes 'X' . Todos os descontos são acumulativos.

A COMPANHIA X mantém uma lista negra de clientes que tem um registro de pagamento ruim ou cliente que são considerados perigosos para créditos. Entretanto, todos os descontos são aplicados, independente do estado do crédito do cliente. Se um cliente está na lista negra, uma fatura especial é preparada.'

Solução:

O 1º passo é localizar e identificar as condições. Há 4 condições:

C1 O cliente é comerciante?

C2 Pertence a associação?

C3 Valor do pedido ≥ Cr$ 1.000,00.

C4 O cliente está na lista negra.

O 2º passo é lista as ações.

A1 10% de desconto por ser comerciante.

A2 5% de desconto p/ pertencer a associação.

A3 5% de desconto do bruto.

A4 prepara uma fatura normal.

A5 prepara uma fatura especial.

O próximo passo é completar a matriz de condições. Como é uma tabela de entrada limitada há 2^4 = 16 regras possíveis. A matriz será então:

C1	C2	C3	C4
S	S	S	S
S	S	S	N
S	S	N	S
S	S	N	N
S	N	S	S
S	N	S	N
S	N	N	S
S	N	N	N
N	S	S	S
N	S	S	N
N	S	N	S
N	S	N	N
N	N	S	S
N	N	S	N
N	N	N	S
N	N	N	N

fig. 5.2-1

Cada regra é então examinada e as entradas das ações são marcadas. Baseado no texto, a tabela completa é mostrada na fig. 5.2-2 . A tabela não pode ser unificada. Examinando a tabela pode haver regras impossíveis. A tabela contém regras da seguinte forma:

C1 o cliente é comerciante?

C2 pertence a associação?

N S

C1	C2	C3	C4	A1	A2	A3	A4	A5
S	S	S	S	X	X	X		X
S	S	S	N	X	X	X	X	
S	S	N	S	X	X			X
S	S	N	N	X	X		X	
S	N	S	S	X		X		X
S	N	S	N	X		X	X	
S	N	N	S	X				X
S	N	N	N	X			X	
N	S	S	S		X	X		X
N	S	S	N		X	X	X	
N	S	N	S		X			X
N	S	N	N		X		X	
N	N	S	S			X		X
N	N	S	N			X	X	
N	N	N	S					X
N	N	N	N				X	

fig. 5.2-2

Este tipo de interpretação é errada pois para um cliente que pertence a associação tem que ser comerciante. Esta alternativa é mostrada na fig. 5.2-3 . (A6 - erro)

C1	C2	C3	C4	A1	A2	A3	A4	A5	A6
S	S	S	S	X	X	X		X	
S	S	S	N	X	X	X	X		
S	S	N	S	X	X			X	
S	S	N	N	X	X		X		
S	N	S	S	X		X		X	
S	N	S	N	X		X	X		
S	N	N	S	X				X	
S	N	N	N	X			X		
N	S	S	S						X
N	S	S	N						X
N	S	N	S						X
N	S	N	N						X
N	N	S	S			X		X	
N	N	S	N			X	X		
N	N	N	S					X	
N	N	N	N				X		

fig. 5.2-3

Note que agora a tabela pode ser unificada, resultando na seguinte tabela.

C1	C2	C3	C4	A1	A2	A3	A4	A5	A6
S	S	S	S	X	X	X		X	
S	S	S	N	X	X	X	X		
S	S	N	S	X	X			X	
S	S	N	N	X	X		X		
S	N	S	S	X		X		X	
S	N	S	N	X		X	X		
S	N	N	S	X				X	
S	N	N	N				X		
N	S	-	-						X
N	N	S	S			X		X	
N	N	S	N			X	X		
N	N	N	S					X	
N	N	N	N				X		

fig. 5.2-4

2º) OUTRO PROBLEMA:

Estude o texto a seguir e entreviste a si mesmo, preparando uma tabela de entrada limitada, pelo desenvolvimento da regra progressiva.

'A COMPANHIA Y é uma grande companhia que vende por atacado várias peças de veículos. Todos os pedidos vão para um empregado que processa-os da seguinte maneira:
Cada ítem, de cada pedido, é examinado para ver se há estoque disponível. Se há estoque disponível suficiente, para cobrir a quantidade pedida, o empregado ajusta o balanço do livro "razão" para a quantidade apropriada p/ ser despachada e envia detalhes para o departamento de despacho, tal que a mercadoria possa ser enviada ao cliente. Cada vez que o balanço do estoque é ajustado, o empregado examina o balanço e compara o nível de novos pedidos no livro razão de estoque.

Se há estoque, porém em quantidade insuficiente para satisfazer um pedido de um cliente, ele envia o que está disponível, ajusta o balanço, anota a quantidade pendente do pedido e guarda o pedido de encomenda em um arquivo auxiliar, para esperar o recebimen to de estoque. Um pedido de compra é também enviado se isto é necessário.

Se ele acha que um pedido de compra do livro de estoque tiver sido enviado previamente, ele prepara urgente uma carta postal especial ao fornecedor.

Se não há estoque disponível para atender ao pedido do cliente, então o pedido é guardado em um arquivo auxiliar. O empregado então telefona ao fornecedor para aconselhar uma entrega especial, indiferente se há um pedido de compra pendente ou não.

Solução:

C1 Há algum estoque?
- C2 Recebeu pedido de estoque?
  - C3 Balanço está abaixo do nível do estoque?
    - C4 Pedido de compra.

A1 Ajusta o balanço do estoque.
- A2 Envia instrução p/ despachar - pedido completo.
  - A3 Envia carta postal de entrega.
    - A4 Aumenta pedido de compra.
      - A5 Envia instrução p/ despacho.
        - A6 Arquivo auxiliar
        - A7 Aconselha p/ telefone.

C1	C2	C3	C4	A1	A2	A3	A4	A5	A6	A7
S	S	S	S	X	X	X				
S	S	S	N	X	X		X			
S	S	N	-	X	X					
S	N	-	S	X		X		X	X	
S	N	-	N	X			X	X	X	
N	-	-	-						X	
										X

ROL - Nível de novos pedidos.

fig. 5.2-5

3º) MAIS UM:

A fig. 5.2-6 contém multiplos erros de lógica. Use tanto o cheque aritmético como o de bifurcação para localizar os erros. Por ter a tabela sido simplificada pela omissão de condições e ações, não é possível corrigir estes erros. Como pode a tabela ser corrigida se as declarações não estão completas?

C1	C2	C3	C4	C5	C6	C7	A1	A2	A3	A4	A5	A6
S	S	-	-	-	-	-	X		X		X	X
S	N	S	S	S	S	-	X		X	X	X	X
S	N	S	S	S	N	S	X	X		X	X	X
S	N	S	S	S	N	N	X	X		X	X	X
S	N	S	S	N	S	S	X		X		X	X
S	N	S	S	N	S	N	X		X		X	X
S	N	S	S	N	N	S	X	X		X	X	X
S	N	S	S	N	N	N	X	X		X	X	X
S	N	S	N	S	-	S	X	X	X		X	X
S	N	S	N	S	-	N	X	X		X	X	X
S	N	S	N	N	-	S	X	X	X	X	X	
S	N	S	N	N	-	N	X	X	X	X	X	X
S	N	N	-	S	S	S	X	X	X			X
S	N	N	-	S	S	N	X	X	X	X		X
S	N	N	-	S	N	-	X		X	X		X
S	N	N	-	N	S	S	X		X		X	
S	N	N	-	N	S	N	X	X	X	X	X	
S	N	N	-	N	N	-	X	X		X	X	
N	S	-	S	-	-	S	X	X		X	X	
N	S	-	S	-	-	N	X	X			X	X
N	-	-	S	-	S	S	X	X		X	X	X
N	-	-	S	-	S	N	X	X			X	X
N	S	-	N	-	-	S	X	X	X	X		X
N	S	-	N	-	-	N	X	X	X			X
N	-	-	N	-	S	S	X	X	X	X		X
N	-	-	N	-	S	N	X	X	X			X
N	N	-	S	-	N	S	X	X	X	X	X	

fig. 5.2-6

continuação da fig. 5.2-6

C1	C2	C3	C4	C5	C6	C7	A1	A2	A3	A4	A5	A6
N	N	-	S	-	N	N	X		X		X	
N	N	-	N	-	N	S	X	X	X	X	X	
N	N	-	N	-	N	N	X	X				X

Solução:

Atacar este problema, por uma pesquisa da tabela completa é impossível, pois há muitas condições e ações. A melhor maneira para atacar o problema é segmentar e reorganizar tabela em 2 partes, baseado em C1; isto dá 2 tabelas baseadas em S e N, dividindo entre as regras 18 e 19.

Tomando a subtabela C1 = N.

C1	C2	C3	C4	C5	C6	C7
N	S	-	S	-	-	S
N	S	-	S	-	-	N
N	-	-	S	-	S	S
N	-	-	S	-	S	N
N	S	-	N	-	-	S
N	S	-	N	-	-	N

fig. 5.2-7

continuação da fig. 5.2-7

C1	C2	C3	C4	C5	C6	C7
N	-	-	N	-	S	S
N	-	-	N	-	S	N
N	N	-	S	-	N	S
N	N	-	S	-	N	N
N	N	-	N	-	N	S
N	N	-	N	-	N	N

-Resequenciando as condições tal que as condições com muitos símbolos "-" fiquem à direita teremos:

C1	C4	C7	C6	C2	C3	C5
N	S	S	-	S	-	-
N	S	N	-	S	-	-
N	S	S	S	-	-	-
N	S	N	S	-	-	-
N	N	S	-	S	-	-
N	N	N	-	S	-	-
N	N	S	S	-	-	-
N	N	N	S	-	-	-
N	S	S	N	N	-	-
N	S	N	N	N	-	-
N	N	S	N	N	-	-
N	N	N	N	N	-	-

fig. 5.2-7

O próximo passo é resequenciar as regras para produzir S e N. Vejamos o bloco C1 = N e C4 = S:

C1	C4	C7	C6	C2	C3	C5
N	S	S	-	S	-	-
N	S	N	-	S	-	-
N	S	S	S	-	-	-
N	S	N	S	-	-	-
N	S	S	N	N	-	-
N	S	N	N	N	-	-

fig. 5.2-8

Esta resequenciação de regras pode continuar até limpar o bloco S e N. Bifurcando o bloco das 6 regras 19, 20, 21, 22, 27, 28 vemos que há erro lógico. Examinando as ações das regras 19 e 21 há redundancia porque as ações destas duas regras são as mesmas. As regras 20 e 22 também são redundantes. Examinando o restante, de tabela temos:

regra 23 e 25 - redundancia
" 24 e 26 - "

O mesmo estudo pode ser usado no lado direito da subtabela C1=S Não há erros.

CAPÍTULO VI

TABELAS DE DECISÃO APLICADAS A ANÁLISE DE SISTEMAS

CAP. VI - T. D. EM ANÁLISE DE SISTEMAS

Quem está aprendendo as técnicas das tabelas de decisão, tende a visualizar um conjunto complexo de condições e ações,e utiliza a tabela para resolver seus problemas. Enquanto este uso individual é tanto válido como frequente, nós não podemos / perder a visão do fato de que, as tabelas de deciaão foram desenvolvidas primeiramente como um veículo de comunicação homem a homem. Esta área de comunicação entre pessoas é onde as tabelas de decisão, claramente mostram seus valores.

Este capítulo descreve o processo de análies de um sistema comercial, a fim de valorizar o uso das tabelas de decisões como ferramenta para a criação de um entendimento mútuo, de um problema entre pessoas.

Nós definiremos 'análise de sistemas' como o processo através do qual o analista consegue um entendimento para o desenvolvimento de um novo sistema é chamado projeto de sistemas.

VI. 1 - Funções chaves de sistema

O desenvolvimento de um sistema comercial é um processo complexo. Muitas pessoas são envolvidas - tanto pessoal do usuário como de processamento de dados. Com um projeto desta extensão é dificil resumir as muitas tarefas a serm executadas e documentá-las.

Há 4 pontos chaves no análise de um sistema:

1. Inicialização do projeto.
2. Definição das necessidades.
3. Implementação.
4. Supote após a implementação.

Estas fases tem sido quebradas em 11 passos, conforme anexo.

FASE	TAREFA	ENVOLVIMENTO PRINCIPAL	OBJETIVOS	DOCUMENTAÇÃO DE SAÍDA
A. Início do projeto.	1. Pedido do usuário.	Usuário, gerente de informações e o gerente senior de processamento de dados.	O usuário examina seus próprios objetivos e meios. Preferivelmente colocar diretamente na forma de pedido. Este pedido é examinado pelo gerente de informação em conjunto c/ o gerente de processamento de dados. Aponta os termos básicos de referencia p/ futuros trabalhos.	Pedido do usuário contém: (a) objetivos (b) limites e abrangencias (c) escala de tempos (d) principais relatórios (e) sistema e soluções seguidas pelo usuário (f) relacionamento c/ outros sistemas.
	2. Estudo de viabilidade.	Usuário e senior de sistema.	Técnicos seniors envestigam e revisam o ambiente e requisitos do usuário. Os técnicos seniors podem ser: analista, pessoas de O e M. Levantam soluções alternativas para o pedido do usuário.	Relatório do estudo de viabilidade: (a) objetivos, etc como visto no pedido do usuário, possivelmente modificados. (b) descrição básica do sistema atual.

				(c) solução que podem ser: não fazer, soluções O e M, soluções manuais, soluções em computador. (d) para cada solução custos de desenvolvimento e operacional, benefícios esperados, escala de tempos, etc.
	3. Exame do usuário.	Usuário, senior de sistemas.	Revisão do relatório de estudo de viabilidade e seleção das soluções as quais aparecem p/ encarar o problema apresentado no pedido do usuário. Emissão de termos formais de referencia p/ trabalhos posteriores.	Definição do sistema: declaração formal em termos de referencia para o desenvolvimento detalhado. Solução selecionada é extraída do relatório de flexibilidade, se necessária modificada.

B. Definição das necessidades.	1. Busca dos fatos.	Usuário, analista de sistemas.	Investigação do sistema atual em detalhe, através de entrevistas. Deverá ser feito por departamento: revisão das funções do trabalho, organização, procedimentos, fluxo de informação (incluindo volumes) crescimento, elementos de dados usados, tamanho de campos, relatórios de saúde, etc.	Organização dos dados: A documentação do resultado da busca dos fatos, inclui: (a) mapa da organização (b) mapa funcional (c) fluxograma (d) especificações (e) descrição do sistema (incluindo descrição dos procedimentos chaves).
	2. Análise	Usuário, analista de sistemas.	Examinar as saídas da organização de dados e identificar: - os processos essenciais/ (funções) - os problemas deste processo precisam ser resolvidos pelo novo sistema. - os dados essenciais.	Proposta do sistema para o usuário, mostrando todas informações levantadas e as conclusões para solucionar o problema.

C. Projeto Lógico.	1. Projeto lógico.	Usuário, analista de sistema e chefe dos programadores.	Baseado nas saídas da análise, o projetista define: (a) a estrutura de relacionamento (b) os documentos de entrada e saída. (c) sistemática funcional (d) fluxo de informação (e) rotina administrativa (f) dicionário de dados, relacionamento da informação.	Propostas do projeto lógico a ser examindo pelo usuário, contendo: (a) Desenho da estrutura/ relacional (b) Relatório de saída definitiva (c) Definição dos documentos de entrada - layout dos documentos. - preenchimentos. - crítica. (d) Desenho da sistemática funcional (e) Definição do fluxo de informação - fluxo p/ onde os relatórios irão circular - fluxo p/ onde os documentos irão circular (f) Definição das rotinas administrativas - intruções de envio e recebimento das informações.

D. Projeto Físico.	1. Projeto físico.	Usuário, analista de sistemas e chefe dos programadores,	Baseado nas saídas da análise e do projeto lógico; o projetista define: (a) Processamento do sistema. (b) Banco de dados lógico - Desenho do processamento do sistema. - Arquivos. (c) Programas (d) Rotinas de segurança e recuperação das informações (e) Dados para testes (f) Banco físico	Especificação do sistema: (a) Definição do sistema. (b) Desenho do processamento do sistema. (c) Descrição dos programas. (d) Plano de testes. (e) Schedule dos tempos e operacionalidade.

E. Implementação.	1. Programação.	Programadores e analista de sistema.	Produção de programas documentados.	
	2. Teste do sistema.	Usuário, analista de sistemas e programação.	Teste final do usuário e procedimento no computador e programas de acordo c/ o plano de teste p/ garantir a aceitabilidade do sistema.	Aprovado, acionado o plano de teste: casos a testar, resultados esperados, resultados obtidos, testes de procedimentos.
	3. Conversão.	Usuário, analista de sistemas,/ operação.	Conversão de arquivos, cheque da documentação final.	Vários procedimentos manuais, especialmente para treinamento do usuário.
			Treinamento. Aquisição de novos formulários.	
	4. Mudança geral.	Usuário, analista de sistemas,/ programadores e operadores	Para efetuar a mudança geral do velho para o novo sistema pode ser paralelo, gradualmente ou imediato.	Documentação final para o usuário é preParada.

F. Após a implementação.	1. Manutenção.	Usuário, analista de sistema, programadores.	Mudança no método de operação (procedimentos manuais de programas) para: - remover um erro. - incorporar mudanças em métodos organizacionais. - adicionar informações p/ o usuário. - melhorar a eficiencia do sistema ou permitir novo hardware ou software.	Atualização da documentação: toda a documentação atingida pela mudança é atualizada para registrar a mudança completa.
	2. Evolução.	Usuário, analista de sistema e programador.	Um estudo de revisão formal do sistema depois de um certo tempo em uso, p/ verificar se os objetivos estão sendo atendidos e operacionalmente estão eficiente.	Relatório de evolução: Repete as necessidades do usuário e estudo de viabilidade e compara com o sistema em uso. Comentar sobre o método de desenvolvimento, eficiencia do sistema e mudança da necessidade.

VI. 2- Entrevistas, exame ao usuário e entendimentos dos procedimentos.

A busca dos fatos aparece em 2 pontos do desenvolvimento do sistema. Inicialmente aparece quando o estudo de viabilidade é chamado. A busca dos fatos detalhda aparece depois do usuário ter aprovado o sistema, baseado no estudo de viabilidade. Neste estágio, a busca dos fatos é uma completa investigação detalhada do sistema presente junto com um exame das necessidades futuras. Esta investigação detalhada examina fatores como:

- necessidade do gerenciamento da informação
- funções do trabalho e organização
- habilidade e atitude de pessoas na operação do sistema
- papel e fluxo da informação
- procedimento na organização, processamento e uso da informação.

A busca do fato pode ser levada de diversas maneiras:

- exame da documentação existente (procedimentos manuais, etc)
- entrevistas
- observação
- questionários

A entrevista é em alto grau o método mais importante para a busca do fato; todas as outras técnicas tendem a se suplementar.

As entrevistas que procuram determinar o que é feito, envolve um exame de procedimento. Este exame pode ser muito difícil.

1. Independente de qualquer interrupção, divagação de entrevistas difícies, o analista precisa manter o controle total da entrevista.

2. Há uma tendencia para descobrir o que é feito no curso normal dos eventos. Entretanto é vital determinar o que acontece na excessão ou em caso de erro.

3. Para entender um procedimento, tôdas as condições que afetam as ações precisam ser examinadas.

Com prática, o desnvolvimento progressivo da regra pode ser usado para investigar um procedimento condicional complexo. O uso desta técnica pode acompanhar ao analista a executar os três objetivos vistos acima. Um procedimento pode ser examinado isoladamente pelo desenvolvimento progressivo da regra conforme o exemplo do capítulo 8.

A construção de uma tabela de decisão, durante um entrevista, pe lo desenvolvimento progressivo, da regra, exige disciplina porque as vantagens da técnica estão no deslocamento dos 'N' para o lado direito da tabela.

Outro uso da técnica é a entrevista a si próprio depois de uma entrevista, identificando questões suplementares e áreas vagas pa ra uma segunda entrevista.

Quando estiver examinando um procedimento, a técnica clássica po- de ser usada para testar o entendimento de um procedimento e com- pletar as informações.

VI. 3- Tabela de decisões e comunicação com o usuário.

É vital durante um exame ao usuário apresentar a ele o resulta- do dos procedimentos completos e sem ambiguidades. Uma fadiga na comunicação pode resultar num gasto em modificação e manuten- ção num estágio mais adiantado.

No exame de procedimentos com o usuário, tabelas de decisões po- dem ser muito uteis se são introduzidas para mecanizar a estrutu- ra deles. Tendo sido o básico explicado, as tabelas são fáceis pa ra ler e usar.

Similarmente, quando um sistema está em desenvolvimento, procedimento operacionais precisam ser especificados para todos os níveis. Se as tabelas de decisão são consideradas estranhas p/ o usuário, uma narrativa escrita pode ser preparada através de uma tabela. Narrativas são invariavelmente melhoradas se a lógica é inicialmente examinada na forma de tabela de decisão.

Quando as tabelas de decisão são preparadas para comunicação com o usuário (ou para revisão de procedimentos durante o desenvolvimento ou instrução de operação depois da implementação) é preciso que elas sejam simples e de fácil entendimento.

Os principais pontos quando da construção de tabelas para o exame com o usuário, são as seguintes:

1. Usar uma forma simples de visualização, não técnica - uma forma complexa pode assustar a um não técnico.

2. Não consolidar totalmente a tabela. Examinar a tabela final e ver se a resequenciação das condições pode tornar mais clara.

3. A tabela não deve ser muito grande para que não seja confusa nos detalhes.

4. Não usar a regra 'ELSE'. É usual para comunicação de técnico para técnico porém confunde o usuário. Além, pode cobrir regras / que sejam importantes.

5. Evitar uso de abreviações sempre que possível . Se for usá - las, definir antes todos os símbolos a serem utilizados.

6. Na 1ª vez que usar uma tabela c/ o usuário ser cuidadoso e paciente. Se ele reagir, escrever as tabelas na forma de narrativa.

VI. 4- Análise

Até aqui as tabelas de decisão tem sido consideradas do ponto de vista de união, entendimento e verificação de um sistema, que está sendo investigado. Como mostra a fig. 6.1.1-1, tendo registrado o sistema e as massidades futuras(e verificando os resultados c/ o usuário), ele está preparado para começar a análise.

Neste estágio, ele identifica o seguinte:

1. As funções essenciais as quais precisam ser chamadas no novo sistema.

2. Os problemas e deficiencias que o novo sistema precisa resolver e acertar.

3. Os dados disponíveis para o uso.

As tabelas de decisão provavelmente não serão usadas particularmente neste estágio - a nova documentação produzida pode consistir da matriz de análise de dados, "checklist" e outros similares.

VI. 5 - Projeto

Quando o analista termina a análise, está preparado para começar o projeto. Dependendo da complexidade do sistema, o projeto pode ser dividido em dois estágios:

No estágio 1, ele projeta o esboço e verifica este projeto com partes interessadas. Com o projeto aprovado, a princípio, ele pode refinar a idéia básica e preparar o projeto do sistema em detalhe - a nível de especificação de programa. Suas principais saídas é uma especificação de sistema, as quais definem todos os seus aspectos. O principal uso das tabelas de decisão está na especificação dos programas.

Cada especificação de programa conterá formato de dados de entrada, saída, arquivos, mensagens, etc e processamento lógico é que ocorre muitos problemas de comunicação.

A comunicação via especificação de progrma é inicial para o sucesso do seu desenvolvimento. É também nesta etapa que as tabelas de decisão podem ser extremamente útil, na definição da lógica condicional complexa. As técnicas utilizadas na programação será vista no próximo capítulo.

CAPÍTULO VII

TABELAS DE DECISÕES APLICADAS A PROGRAMAÇÃO

CAP. VII-TABELAS DE DECISÕES EM PROGRAMAÇÃO

VII.1 - Conversão de uma tabela para fluxograma

Se o programador deseja codificar uma tabela em alguma linguagem de alto nível, manualmente é necessário construir 1º um fluxograma inicial. Os métodos básicos para produzir uma lógica sequencial eficiente, a serem discutidos, são:

1. Teste sequencial das regras
2. Bifurcação
3. Análise gramatical
4. Ponderação da freq. relativa da regra
5. Método combinado
6. Método de comparação
7. Método de Máscara
8. Método de Muthukrishnan e Rajaraman
9. Adaptação do método de Muthukrishnan
10. Programas dirigidos p/ tabela
11. Matriz de transição

VII.1.1 - Teste sequencial das regras

Este método testa condição a condição.

Seja a seguinte tabela:

C1	C2	C3	C4	
S	S	S	-	* 1 *
S	S	N	S	* 2 *
S	N	S	-	* 3 *
S	N	N	-	* 4 *
N	S	-	-	* 5 *
N	N	-	-	* 6 *

fig. 7.1.1-1

aplicando o método ficaria:

```
IF C1 AND C2 AND C3
    THEN GOTO 1
IF C1 AND C2 AND NOT C3 AND C4
    THEN GOTO 2
IF C1 AND NOT C2 AND C3
    THEN GOTO 3
```

fig. 7.1.1-2

E assim por diante. O fluxograma ficaria como na fig. 7.1.1-3.

Este método tende a produzir um fluxograma mais preciso, porém é extremamente lento e enorme. Este certamente é o mais ineficiente de todos os métodos.

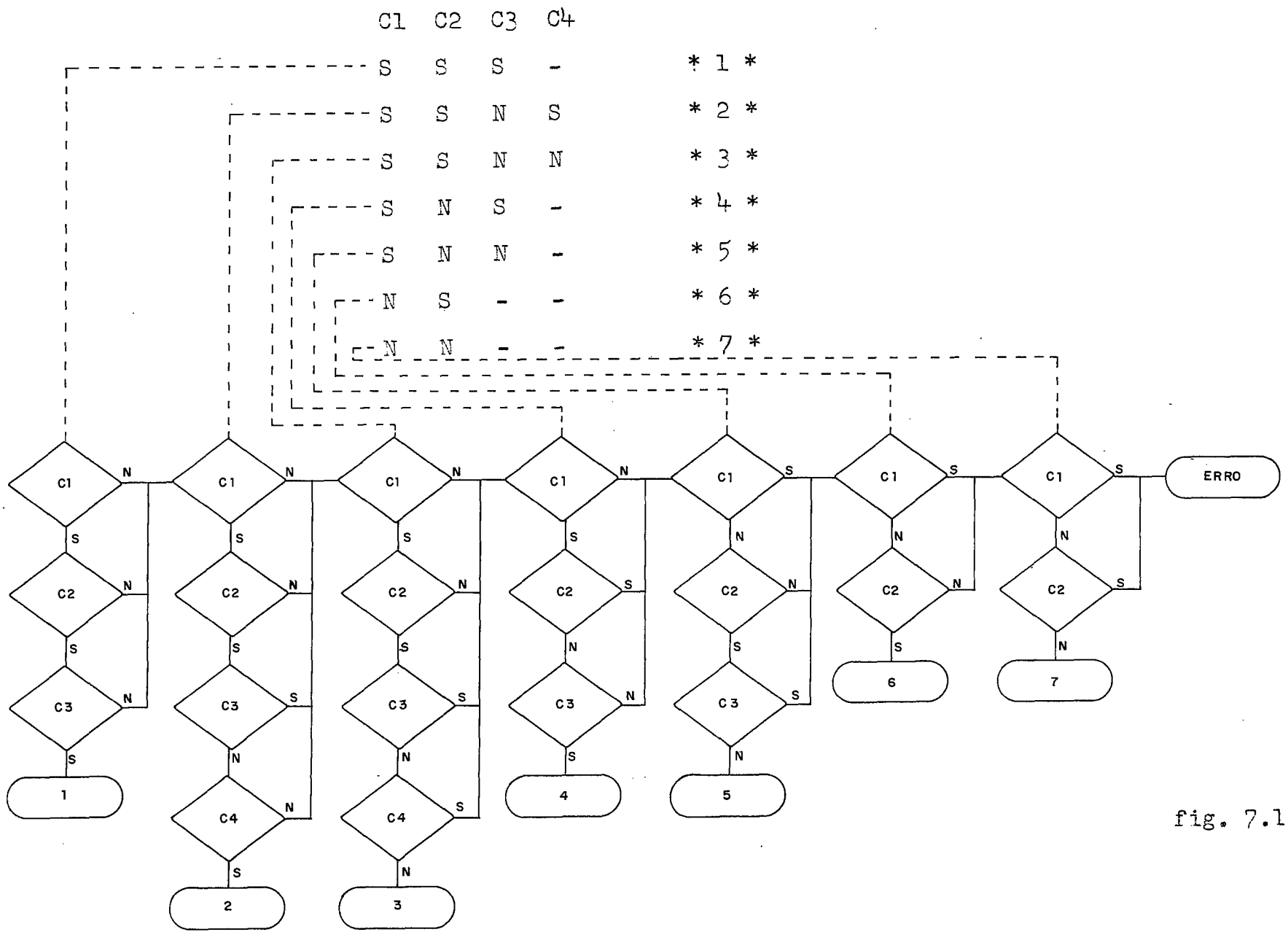

C1 C2 C3 C4
S S S -
S S N S
S S N N
S N S -
S N N -
N S - -
N N - -
* 1 *
* 2 *
* 3 *
* 4 *
* 5 *
* 6 *
* 7 *
C1
C2
C3
C4
N
S
1
2
3
4
5
6
7
ERRO

fig. 7.1.1-3

VII.1.2 - Bifurcação

Este método divide as partes S e N. O exemplo da fig. 7.1.2-1 mostra que a tabela é dividida na parte S e na parte N. A sequencia é a seguinte:

1. Divisão da condição C1.
   regras 1 a 5 = S (A abaixo)
   regras 6 a 7 = N (B abaixo)

A.2. Dentro das regras 1 a 5, divide C2.
   regra 1 a 3 = S
   regra 4 a 5 = N

3. Dentro das regras 1 a 3, divide C3.
   regra 1 = S
   regra 2 a 3 = N

4. Dentro das regras 2 a 3, divide C4.
   regra 2 = S
   regra 3 = N

B.5. Dentro das regras 6 a 7, divide C2.
   regra 6 = S
   regra 7 = N

Ex:

C1	C2	C3	C4	
S	S	S	-	* 1 *
S	S	N	S	* 2 *
S	S	N	N	* 3 *
S	N	S	-	* 4 *
S	N	N	-	* 5 *
N	S	-	-	* 6 *
N	N	-	-	* 7 *

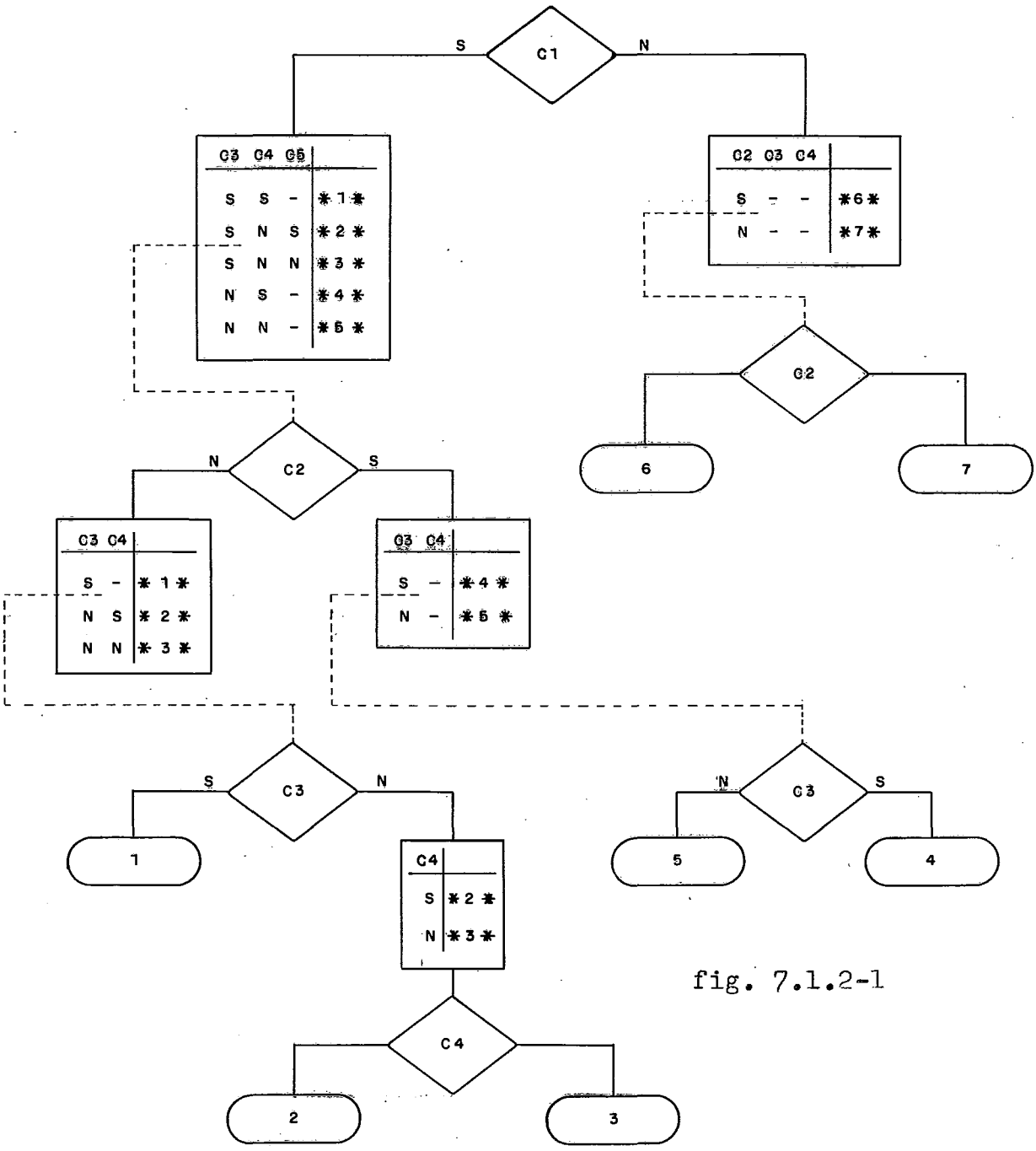


fig. 7.1.2-1

C1	C2	C3	C4	
S	S	S	-	* 1 *
S	S	N	S	* 2 *
S	S	N	N	* 3 *
S	N	S	-	* 4 *
S	N	N	-	* 5 *
N	S	-	-	* 6 *
N	N	-	-	* 7 *

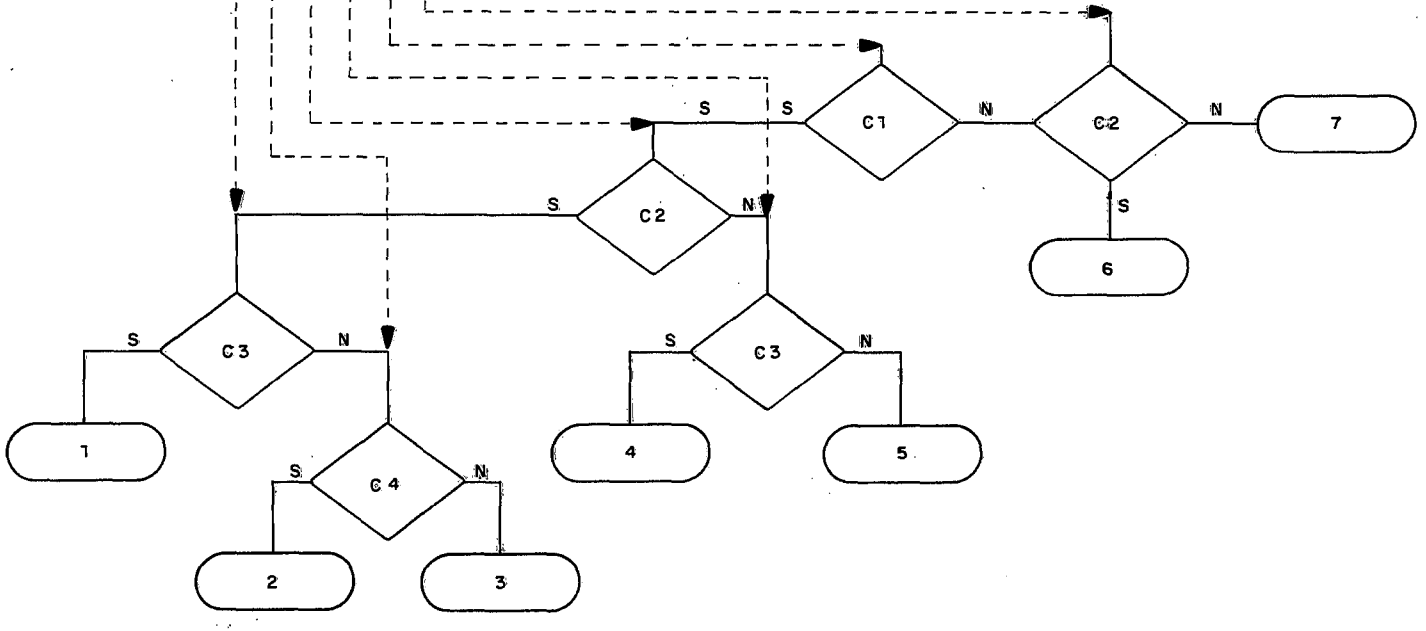


fig. 7.1.2-2

Note que é possível resequenciar as condições e usar condições 'dummy' para facilitar o método. Isto deve ser feito colocando as condições, com mais símbolos de indiferença, a direita da tabela.

Este método é deficiente, mas certamente é superior ao método do teste sequencial das regras, reduzindo assim o nº de testes e portanto, a memória utilizada.
Obviamente a eficiencia, isto é, o tempo de execução, depende da sequencia das condições e regras. A frequencia relativa de cada/ regra determina em grande parte o tempo de execução, contudo este fator não poderá ser considerado quando o método de conversão for a bifurcação.

VII.1.3 - Análise gramatical ("Parsing")

"Parsing" é um método que tenta minimizar a memória usada, porém não necessariamente reduz o tempo. "Parsing" tenta identificar a melhor sequencia de testes.

Consideremos o seguinte exemplo:

C1	C2	
S	N	* 1 *
-	S	* 2 *
N	N	* 3 *

fig. 7.1.3-1

Há duas maneiras de representar este problema em fluxograma:

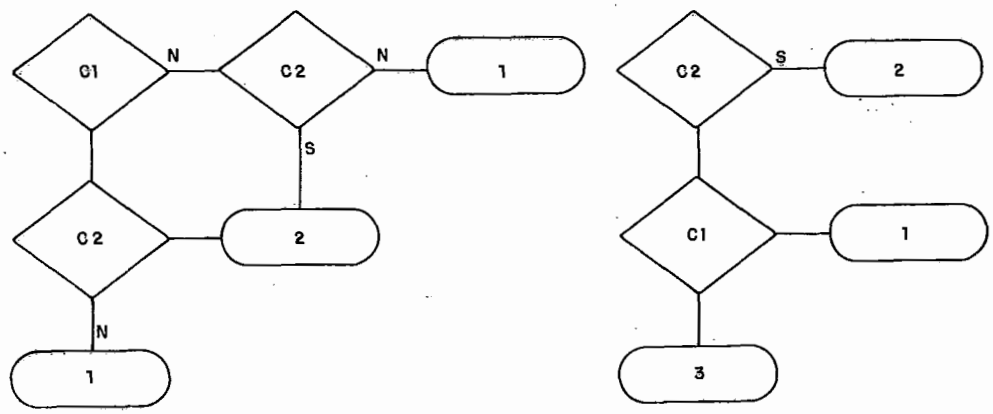


FIG. 7.1.3-2

Isto demonstra a 1ª regra de "parsing"

Regra 1: Comece com a condição que tenha menos símbolos de indiferença ('-').

Note que isto se refere a uma coluna de condição e não uma linha de regra. Assim no exemplo anterior, a coluna de C1 tem $\begin{cases} S \\ - \\ N \end{cases}$ enquanto que a coluna de C2 tem $\begin{cases} N \\ S \\ N \end{cases}$

Começamos então pela condição C2.

Seja outro exemplo:

C1	C2	C3	C4	
S	S	S	-	* 1 *
S	S	N	S	* 2 *
S	S	N	N	* 3 *
S	N	S	-	* 4 *
S	N	N	-	* 5 *
N	S	-	-	* 6 *
N	N	-	-	* 7 *

fig. 7.1.3-3

Utilizando o método de "parsing", temos a seguinte sequencia:

1º C1 nenhuma indiferença
C2 nenhuma indiferença

2º C3 2 indiferenças

3º C4 5 indiferenças

Quando não há nenhum símbolo de indiferença, ou há empate entre condições, será considerado outro critério.

Consideremos outro exemplo:

C1	C2	
S	S	* 1 *
N	S	* 2 *
ELSE		* 3 *

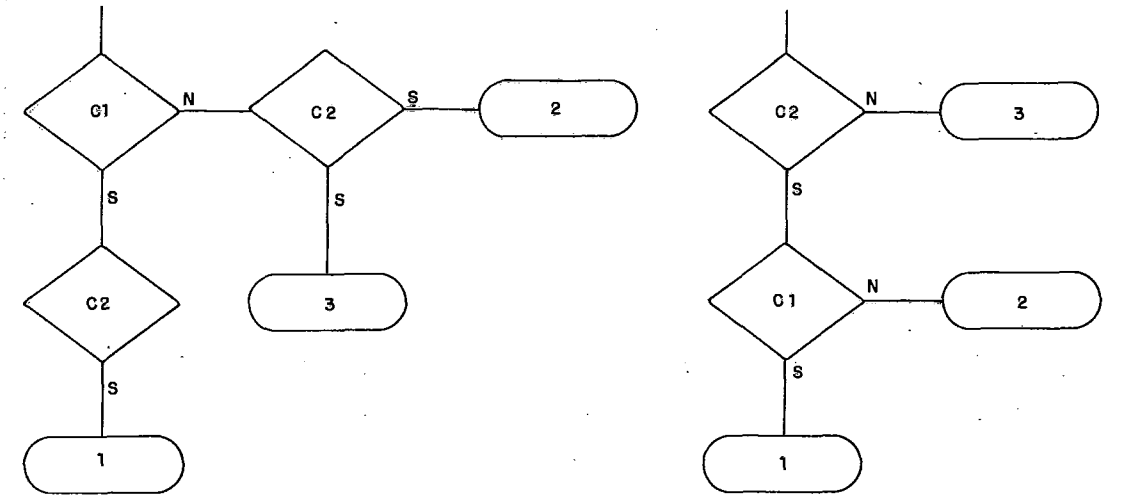


fig. 7.1.3-4

Isto nos leva a uma 2ª regra

Regra 2: Das condições isoladas pela regra 1, comemeçar com uma condição que só tenha 'S' ou 'N'.

Ex.:

	C1	C2	C3	C4	C5	
(6b)	S	S	S	S	S	* 1 *
(5)	S	S	S	S	N	* 2 *
(6a)	S	S	S	N	S	* 3 *
(4)	S	S	S	N	N	* 4 *
(6c)	S	S	N	-	S	* 5 *
(3)	S	S	N	-	N	* 6 *
(1)	S	N	-	-	-	* 7 *
(2)	N	-	-	-	S	* 8 *
	N	-	-	-	N	* 9 *

fig. 7.1.3-5

Segmentando em subtabelas

Convertendo para fluxograma temos a fig. 7.1.3-6
É possível construir novos requintes para estas regras, as quais resolveriam a situação do empate. Entretanto estes procedimentos requer aplicações matemáticas complexas o que não são aconselhaveis para o uso diário.

VII.1.4 - Ponderação da frequencia relativa das regras "Parsing" tenta reduzir a memória utilizada. Por outro lado a frequencia relativa de regras tenta reduzir o tempo execução. Pode ser usada para sequenciar as instruções em um teste de condição ou sequenciar as regras.

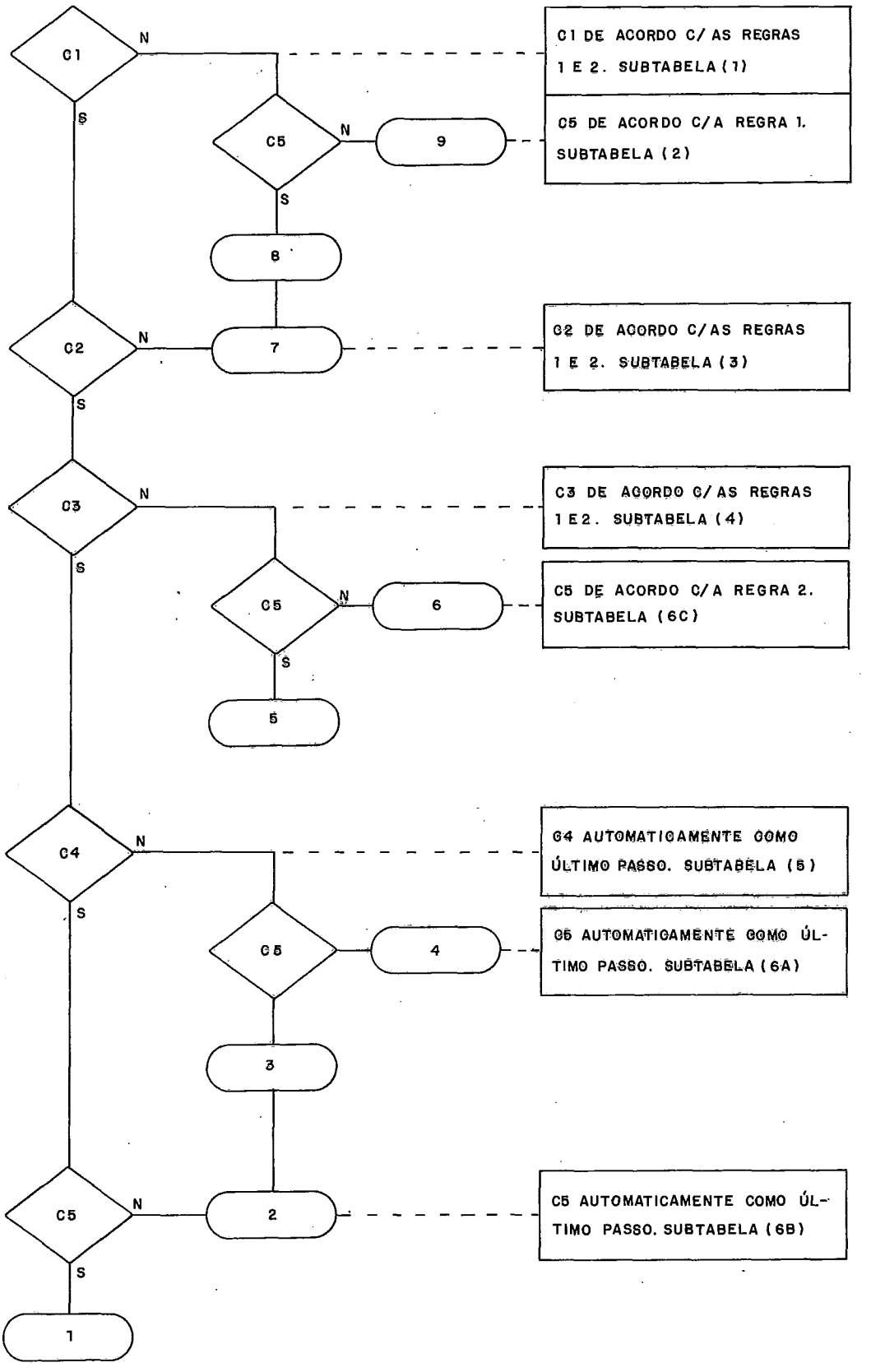

C1
N
S
C1 DE ACORDO C/ AS REGRAS 1 E 2. SUBTABELA (1)
C5
N
9
C5 DE ACORDO C/A REGRA 1. SUBTABELA (2)
S
8
C2
N
7
C2 DE ACORDO C/AS REGRAS 1 E 2. SUBTABELA (3)
S
C3
N
C3 DE ACORDO C/ AS REGRAS 1 E 2. SUBTABELA (4)
S
C5
N
6
C5 DE ACORDO C/A REGRA 2. SUBTABELA (6C)
S
5
C4
N
C4 AUTOMATICAMENTE COMO ÚLTIMO PASSO. SUBTABELA (5)
S
C5
4
C5 AUTOMATICAMENTE COMO ÚLTIMO PASSO. SUBTABELA (6A)
3
C5
N
2
C5 AUTOMATICAMENTE COMO ÚLTIMO PASSO. SUBTABELA (6B)
S
1

fig. 7.1.3-6

VII.1.5 - Método combinado (só para tabela de entrada limitada)

Este método tenta combinar os dois métodos principais: "parsing" e frequencia relativa das regras. As etapas seguidas pelo algoritimo estão assim descritas:

a. Calcula-se $Pj = a^P$, para cada regra; onde Pj é chamdo de potencia de regra j, P é a soma dos '-' contidos na regra x, e a o nº de valores possíveis de uma condição.

b. Calcula-se, para cada coluna, a soma Si = Pj * j, para cada regra contendo um '-', onde j é a frequencia absoluta da regra j e i o nº da condição a examinar.
Se a regra não contém '-' Si = ∅.

c. Escolhe-se a condição Cr que contém o menor somatório Si. Se existir somente um Cr executa-se a etapa.

d. Calcula-se o delta, i das condições, onde i = Si - Nj; onde Si = j * Pj, e calculado para toda regra contendo um 'S' e Nj = j x Pj, para toda regra contendo um 'N'.

e. Escolhe-se a condição Cr que contém o menor (delta). Se existir MAIS DE UMA CONDIÇÃO, escolhe-se a 1ª condição com um e passa-se a etapa seguinte.

f. Divide-se a tabela em duas subtabelas, de acordo com a condição Cr escolhida, de tal modo que as regras contendo um 'S' vão para subtabela, as regras contendo um 'N' vão para a outra; as regras contendo um'-' vão para as duas tabelas, sendo sua frequencia dividida por dois.

Observação: O processo é repetido e a partir da etapa b, até que se tenha isolado uma regra ou encontrado um redundancia ou contradição.

Ex.:

	C1	C2	C3	C4
5	N	N	N	N
8	N	S	-	N
4	-	S	S	S
6	S	-	N	S
9	S	S	S	N

fig.

a) Calculamos Pj para cada regra:

$P1 = 2^0 = 1$

$P2 = 2^1 = 2$

$P3 = 2^1 = 2$

$P4 = 2^1 = 2$

$P5 = 2^0 = 1$

b) Calculamos agora Si para cada coluna:

$S1 = 2 \times 4 = 8$

$S2 = 2 \times 6 = 12$

$S3 = 2 \times 8 = 16$

$S4 = \emptyset$

c) A coluna de menor Si é a 4ª coluna, portanto escolhe-se a condição C4 e temos:

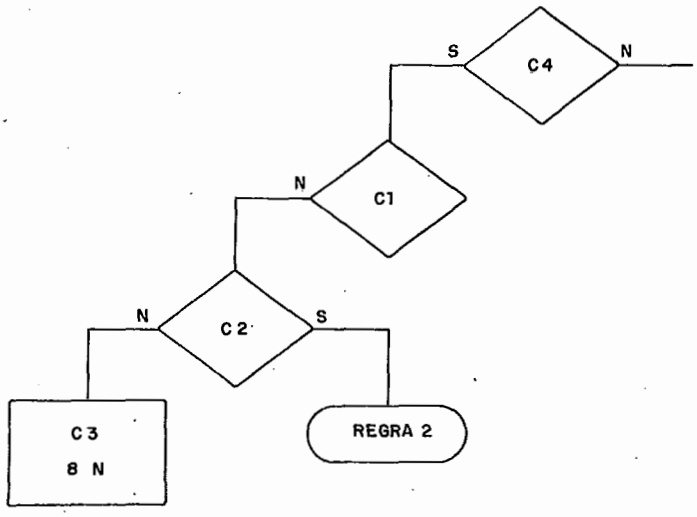


fig.

Analogamente para a tabela T2, teremos:

S1 = Ø

S2 = Ø

S3 = P2 x 2 = 2

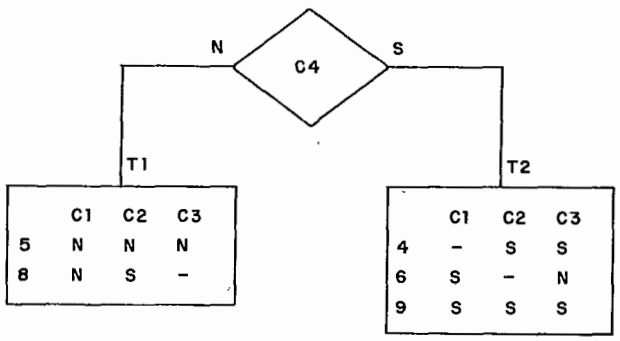


fig.

VII.1.6 - Método de comparação

Este método compara o vetor entrada (combinação de condições) com uma tabela, em que uma das regras poderá ser, ou não, igual ao vetor. O método é o seguinte:

1. A partir da combinação que se deseja, é criando um vetor linha (E) de tal forma que o 'N' seja Ø, o 'S' seja 1 e o '-' seja -1.

Ex.: S S N - S

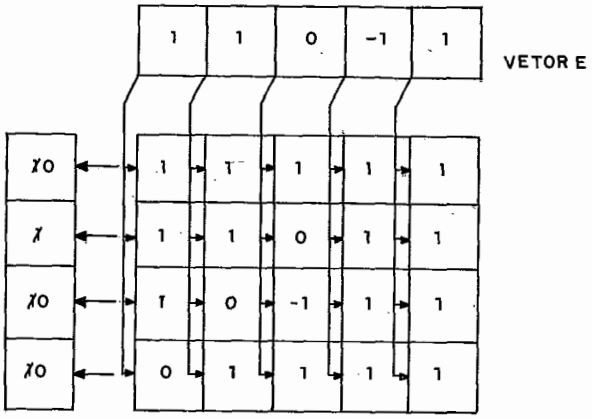
vetor E

2. Definir um vetor coluna (A), com um tamanho igual ao nº de regras da tabela e iniciar com 1.

Ex.: vetor A (digamos que existam 4 regras).

3. Para cada regra da tabela, comparar cada coluna do vetor E , com cada A correspondente da regra selecionada (fazendo uma multiplicação booleana AND) e o resultado colocar na linha correspondente do vetor A. Para cada valor que será colocado no vetor A será feito um AND, com o valor existente, ou seja basta ter um Ø que ficara sempre Ø. Quando tiver -1 não fazer a multiplicação.

Ex.:



4. A linha do vetor que contiver 1 será a regra selecionada. Se houver mais de uma linha com 1, a tabela contém um erro de ambiguidade.

5. A regra escolhida indicará a ação a ser executada.

VII.1.7 - Método da Máscara

Seja a seguinte tabela:

C1	C2	C3
S	-	-
N	S	-
N	N	S
N	N	N

O método é o seguinte:

1º) Converte 'S' para 1 e 'N' ou '-' para Ø e armazena uma matriz.

1	0	0
0	1	0
0	0	1
0	0	0

Tabela Matriz

2º) Converte 'S' ou 'N' para '1' e o '-' para Ø e armazena uma matriz

1	0	0
1	1	0
1	1	1
1	1	1

Matriz Máscara

3º) Convete a combinação que se deseja transformando 'S' para '1', 'N' e '-' para Ø.

EX.: N S S

0 1 1 vetor dado

4º) Multiplicar booleanamente o vetor dado com a tabela Máscara.

regra 1 0 1 1 x 1 0 0 = 0 0 0

regra 2 0 1 1 x 1 1 0 = 0 1 0

regra 3 0 1 1 x 1 1 1 = 0 1 1

regra 4 0 1 1 x 1 1 0 = 0 1 0

5º) Comparar o vetor dado com o resultado:

A regra 3 é igual ao vetor dado.

Se tivesse mais de uma regra possível e tabela conteria uma ambiguidade.

VII.1.8 - Método de MUTHUKRISHNAN e RAJARAMAN

Seja para uma tabela de entrada limitada

C1	C2	C3
S	-	N
-	N	N
N	S	-

1º) Converte S __ 10
N __ 01
- __ 11

10	11	01
11	01	01
01	10	11

MATRIZ A (TABELA)

2º) Converter também combinação que se deseja encontrar

N S S __ 01 10 10 VETOR E

3º) Fazer um AND entre a MATRIZ A e VETOR E e depois fazer um AND entre as colunas de matriz resultante

$$\begin{bmatrix} 10 & 11 & 01 \\ 11 & 01 & 01 \\ 01 & 10 & 11 \end{bmatrix}$$

AND $\begin{bmatrix} 01 & 10 & 10 \end{bmatrix}$

REGRA	1	0	1	0	AND	0
"	2	1	0	0	AND	0
"	3	1	1	1	AND	1

4º) A regra que contiver um 1 será a escolhida. Mais de um 1 a tabela está errada.

Seja agora uma tabela de entrada expandida

C1	C2	C3
=x	-	S
=z	180	-
-	180	S

1º)Transformar todos os '-' em 1 e o restante em Ø

$$M = \begin{matrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{matrix}$$

2º) Seja a seguinte combinação que queiram encontrar:

C1 = 200 x = 200 y = 150 C2 = 250 C3 =

Comparar a entrada com a tabela expandida e marcar com 1 o que for verdade e Ø o que for falso.

$$T = \begin{vmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

3º) Fazer um OR entre M e T e um AND em cada linha da matriz resultante

$$D = \begin{vmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{vmatrix} \text{ OR } \begin{vmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} = \begin{array}{ccccccc} 1 & 1 & 1 & \text{AND} & 1 & \text{regra} & 1 \\ 0 & 1 & 1 & \_\_\_\_ & 0 & " & 2 \\ 1 & 0 & 1 & \_\_\_\_ & 0 & " & 3 \end{array}$$

4º) A regra que contiver o 1 será escolhida

VII.1.9 - Adaptação do método de MUTHUKRISHNAN

Seja a seguinte tabela:

-	S	S
-	S	N
-	N	-
N	N	S
N	N	N

1º) Transformar 'S' em 1 'N' em Ø e '-' em -1

-1	1	1
-1	1	0
-1	0	-1
0	0	1
0	0	0

2º) Gerar duas matrizes sendo uma igual a original somente transformando os 'S' em '1', os 'N' em '0' e os '-' em '-1'.

S		
1	1	1
1	1	0
1	0	1
0	0	1
0	0	0

N		
1	0	0
1	0	1
1	1	1
1	1	0
1	1	0

3º) Transforme a combinação de entrada também em 2 vetores:

Seja d = S N N, então teremos:

d1 = 1 0 0 e d2 = 0 1 1

S N

4º) Fazer um AND entre vetores e matrizes correspondente.

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

AND $\begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 1 \end{bmatrix}$

$$\begin{matrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{matrix} \qquad \begin{matrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \\ 1 & 0 \end{matrix}$$

5º) Fazer um AND entre as colunas da matriz resultante

regra	1	1	0	0	OR	0
"	2	1	0	1	___	0
"	3	1	1	1	___	1
"	4	0	1	0	___	0
"	5	0	1	0	___	0

6º) A regra que contiver 1 será a escolhida.

VII. 2 - PROGRAMAS DIRIGIDOS POR TABELAS

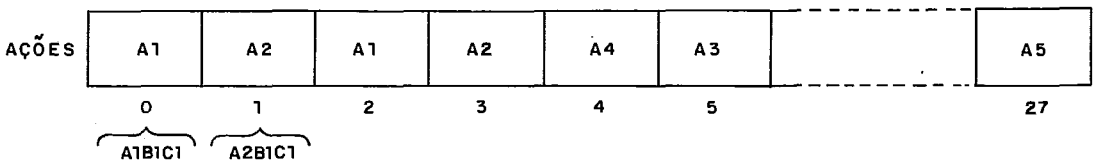
Este método é para ser aplicado para tabelas mistas e expandidas.

Começaremos definindo uma matriz que armazena em cada coluna os valores para uma determinada variável e definiremos, associado a cada valor, o coeficiente , que identificará este valor

Ex.: Seja 3 variáveis A, B, C e cada uma contenha 3 valores possíveis: A1, A2, A3; B1, B2, B3; C1, C2, C3

	A	B	C
valor	A1	B1	C1
coeficiente	0	0	0
	A2	B2	C2
	1	3	9
	A3	B3	C3
	2	6	18

Definiremos agora um vetor associação que conterá a ação/ou ações a serem executadas. O índice será a soma dos coeficientes da combinação de cada valor das variáveis.



A técnica consiste em somar os coeficientes de uma determinada combinação e o resultado indicará, através de um vetor associação, a ação a ser tomada.

A determinação é feita do seguinte modo

a) Cada variável é pesquisada na fila correspondente. Encontrada, acumula-se o coeficiente J.

b) A soma dos coeficientes associados determina um coeficiente que identifica a combinação.

Ex.: Digamos que ao ler um arquivo, tenhamos registros com tres campos cada. A combinação de um valor de campo entre os campos será executada uma determinada ação. Digamos que a tabela seja a seguinte.

A=	B=	C= 1
1	2	3
2	1	2
3	4	1

Montando a tabela com os coeficientes teríamos:

A	B	C
1	2	3
0	0	0
2	1	2
1	3	9
3	4	1
2	6	18

Digamos que o vetor associação seja:

AÇÕES	A1	A1	A2	A2	A3	A2		A1
COEFICIENTE	0	1	2	3	4	5		27

Digamos que quizessemos selecionar a combinação A = 1 / B2 = 1 C = 3

Somando os coeficientes achuriados na tabela teríamos: 0 + 3 - 0 = 3

O índice 3 no vetor associação, corresponde a executar a ação 2.

Os coeficientes são calculados da seguinte maneira:

	0	1	2		N
0	0	0	0		0
1	1	1 x M	1 x M^2		1 x M^{N-1}
2	2	2 x M	2 x M^2		2 x M^{N-1}
M	M- 1	(M-1) M	(M-1) M^2		(M-1) M^{N-1}

VII. 3 - MATRIZ DE TRANSIÇÃO

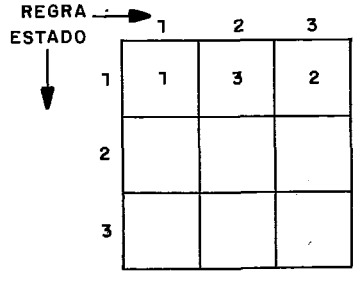
Vimos até agora métodos para transformar a matriz regra e ação em programa. Veremos agora como faremos os desvios de uma tabela para outra tabela ou seja como faremos o encadeamento das tabelas.

Seja a seguinte tabela

TAB1..

C1	C2	C3		A1	A2	A3
S	S	S	* TAB1 *	X		
S	S	N	* TAB3 *		X	
S	N	S	* TAB2 *			X

Montamos uma matriz de duas dimensões que se chamará matriz de transição. A 1ª dimensão indicará o estado, ou seja a tabela, em que se encontra e, a 2ª dimensão indicará a regra que será executada. O conteúdo conterá os desvios. Convertendo/ a tabela acima para uma matriz de transição teríamos:



Se estiverem na tabela 1 (estado = 1) e a regra escolhida foi a 2 então o desvio será para a tabela 3.

Se estiverem na tabela 1 (estado = 1) e a regra escolhida foi a 3 então o desvio será para a tabela 2.

Ex.:

```
TAB1..
        C1  C2
        S   S       * TAB2 *
        S   N       * TAB3 *
        N   N       * TAB1 *

TAB2..
        C1
        S           * TAB2 *
        N           * TAB3 *

TAB3..
        C1  C2  C3
        S   S   S   * TAB1 *
        S   N   S   * TAB2 *
        S   N   N   * TAB3 *
```

CAPÍTULO VIII

APLICAÇÕES

CAPÍTULO VIII - APLICAÇÕES

Será dado um exemplo de um sistema simples a ser desenvolvido, utilizando as técnicas até aqui descritas. A metodologia de desenvolvimento do sistema exemplo, foi baseada nos conceitos apresentados no capítulo VI. A preocupação do autor não foi de desenvolver um sistema ótimo, mas sim mostrar como as tabelas de decisão poderão ser úteis neste desenvolvimento. O exemplo irá até o nível de programação, aplicando as teorias do capítulo VII. Os documentos utilizados durante o exemplo foram adaptadas do 'MANUAL DE DOCUMENTAÇÃO DE SISTEMAS' DA NCC.

O projeto a ser desenvolvido visa atender as necessidades de uma companhia de aviação no setor de reservas de passagens.

A. INÍCIO DO PROJETO

1. PEDIDO DO USUÁRIO

OBJETIVOS: Desenvolver um sistema que acelere e controle o atendimento ao público, no pedido de reservas de passagens.

LIMITES E ABRANGENCIAS: Como a companhia está em fase de implantação, terá necessidade que esse projeto/ seja desenvolvido em 1 mes.
O sistema irá atender a divisão de atendimento; subordinada a diretoria administrativa.

ESCALAS DE TEMPOS: Como o sistema será utilizado p/ atendimento ao público, este deverá ter um tempo de resposta rápido.

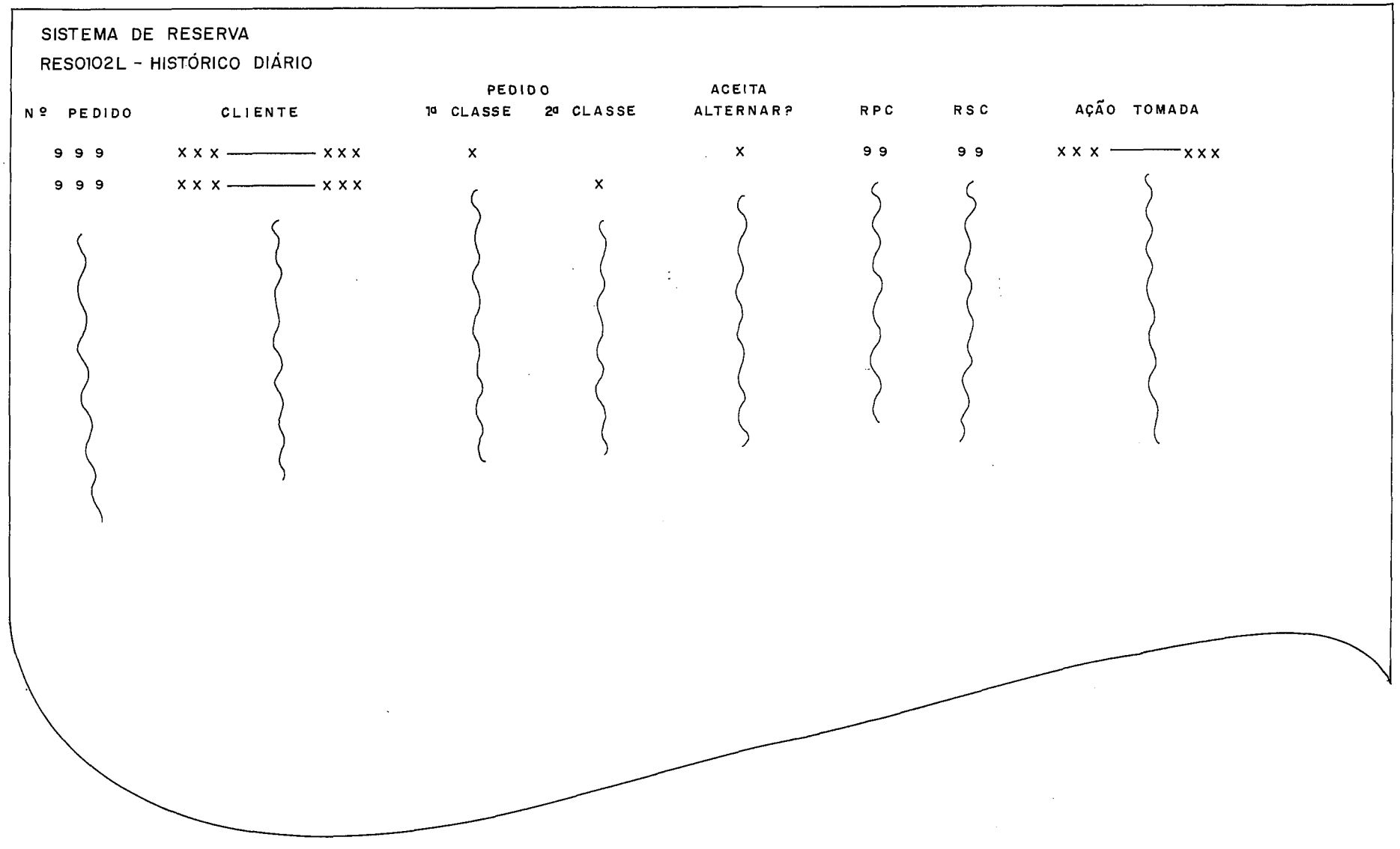
PRINCIPAIS RELATÓRIOS: Deverá ser emitido ao final do dia um relatório, que dê um histórico de todas as transações feitas durante o dia.

SISTEMAS E SOLUÇÕES SUGERIDAS PELO USUÁRIO: Recomenda-se que as transações sejam feitas, através de terminais e que as passagens sejam emitidas através de tele- impressoras.

SISTEMA DE RESERVA

RES0102L - HISTÓRICO DIÁRIO

Nº PEDIDO	CLIENTE	PEDIDO 1ª CLASSE	PEDIDO 2ª CLASSE	ACEITA ALTERNAR?	RPC	RSC	AÇÃO TOMADA
9 9 9	X X X ——— X X X	X		X	9 9	9 9	X X X ——— X X X
9 9 9	X X X ——— X X X		X				



RELACIONAMENTO C/
OUTROS SISTEMAS : A princípio não será necessário haver inligação c/ outros sistemas.

2. ESTUDO DE VIABILIDADE

OBJETIVO: Desenvolver um sistema que visa atender a pedidos de reservas de passagens, de uma maneira rápida e segura.

DESCRIÇÃO DO
SISTEMA : O pedido de reserva de passagem deverá ser enviado ao computador, e, dependendo da disponibilidade de lugar, deverá, ou não, ser feitas as reservas.
Caso seja feita uma reserva, os arquivos deverão ser atualizados e será emitida uma passagem.
Ao final do dia será emitido um relatório dando um histórico de todas as transações feitas durante o dia.

SOLUÇÃO: Para que a resposta seja rápida, o pedido deverá ser enviado via terminal, fazendo consultas e atualigações ON-LINE e conforme disponibilidade emitir passagem via tele- impressora de todas as transações feitas durante o dia.

CUSTO:

EQUIPAMENTO

Um terminal video - Cr$ 40.000,00

Uma tele- impressora - Cr$ 30.000,00

sub total 70.000,00

CUSTO: (continuação)

DESENVOLVIMENTO DO SISTEMA

EM 1 MES (com encargos)

1 ANALISTA	- G$ 20.000,00
1 PROGRAMADOR	- G$ 10.000,00
sub total	30.000,00

CUSTO OPERACIONAL MENSAL (com encargos)

1 PESSOA P/ ATENDIMENTO	- G$ 3.000,00
PESSOAL DO COMPUTADOR	- G$ 10.000,00
DIVERSOS	- G$ 3.000,00
sub total	G$ 16.000,00

3. EXAME DO USUÁRIO

O sistema apresentado satisfaz plenamente as nossas necessidades.

B. DEFINIÇÃO DAS NECESSIDADES

1. BUSCA DOS FATOS

Na busca dos fatos são feitos várias reuniões. Para não tomar muito tempo, será apresentada uma das entrevistas feitas com o chefe da divisão de atendimento (Sr. Marcus). Esta entrevista teve o objetivo de levantar os procedimentos para reservar uma passagem.

O analista de sistemas (A.S) irá mostrar uma tabela de decisão através da técnica DESENVOLVIMENTO PROGRESSIVO DA REGRA (Capítulo V).

Das entrevistas, o analista de sistemas aprendeu que existe dois tipos de passagens 1ª classe e 2ª classe. Troca de classes são feitas se o lugar não está disponível na classe requisitada. Tais trocas dependem se há lugar disponível na classe a ser trocada e se o passageiro aceita a troca.

A tabela de decisão é mostrada em vários estágios de desenvolvimento.

AS : Bem, Marcus, qual é a 1ª coisa que você considera quando é pedido uma reserva?

Marcus : Eu olho a classe para ver se o pedido é para 1ª ou 2ª classe.

AS : Suponha que o pedido seja para a 1ª classe. O que é feito?

C1 Reserva é 1 classe?

S

Marcus : Nós temos um mapa de reservas e eu checo se tanto para a 1ª classe como para 2ª classe tem lugar.

AS : O que voce faz se ambas as classes são requisitadas para um mesmo pedido? Rejeita, eu suponho?

C1 Reserva é 1ª classe?

C2 Reserva é 2ª classe?

C1	C2
S	S

Marcus : Está certo.

AS : Voce procede de alguma outra maneira, se isto acontece?

Marcus : Não - Eu mesmo recuso a reserva.

AS : Sim.

C1 Reserva é 1ª classe?

C2 Reserva é 2ª classe?

C1	C2	A1 Recuso
S	S	X

Agora se voce tem uma reserva e é só para 1ª classe. O que acontece?

Marcus : Eu checo as reservas de 1ª classe (RPC), voce sabe, para ver se há lugar disponível.

C1 Reserva é 1ª classe?
C2 Reserva é 2ª classe?
C3 RPC está disponível?

A1 Recuso

C1	C2	C3	A1
S	S	-	X
S	N	S	

AS : E se a 1ª está disponível?

Marcus : Eu emito uma passagem e atualizo a RPC.

AS : Voce pode me auxiliar enquanto eu anoto?

C1 Reserva é 1ª classe?
C2 Reserva é 2ª classe?
C3 RPC está disponível?

A1 Rejeito.
A2 Atualizo RPC.
A3 Emito passagem 1ª / classe.

C1	C2	C3	A1	A2	A3
S	S	-	X		
S	N	S		X	
S	N	N			X

Agora o que acontece se não há espaço disponível na RPC?

Marcus : Aqui é aonde eu tenho uma pequena dificuldade. Se são passageiros de 1ª classe é mais provável que aceitem voar na 2ª classe do que se forem de 2ª classe. Então eu checo se há espaço na reserva de 2ª classe (RSC). Se há espaço eu pergunto se aceita alternar.

AS : Eu vejo. Voce checa a 2ª classe e se está disponível voce checa se a alternativa é aceita.

Marcus : Esta certo. Então se a alternativa é aceita eu emito uma passagem e atualizo a RSC.

AS :

C1 Reserva é 1ª classe?
C2 Reserva é 2ª classe?
C3 RPC está disponível?
C4 RSC está disponível?
C5 Aceita alternativa?

A1 Rejeito.
A2 Atualizo RPC.
A3 Emito passagem 1ª classe.
A4 Atualizo RSC.
A5 Emito passagem 2ª classe.

C1	C2	C3	C4	C5	A1	A2	A3	A4	A5
S	S	-	-	-	X				
S	N	S	-	-		X	X		
S	N	N	S	S				X	X
S	N	N	S	N					

Diga-me, o que voce faz se a reserva é para 1ª classe, a RPC não está disponível, a RSC está disponível e o passageiro não aceita alternativa?

Marcus : Eu indico outro voo.

C1 Reserva é 1ª classe?
C2 Reserva é 2ª classe?
C3 RPC está disponível?
C4 RSC está disponível?
C5 Aceita alternativa?

A1 Rejeito.
A2 Atualizo RPC.
A3 Emito passagem 1ª classe.
A4 Atualizo RSC.
A5 Emito passagem 2ª classe.
A6 Indico outro voo.

C1	C2	C3	C4	C5	A1	A2	A3	A4	A5	A6
S	S	-	-	-	X					
S	N	S	-	-		X	X			
S	N	N	S	S				X	X	
S	N	N	S	N						X

AS : E se a 2ª classe não está disponível?

Marcus : Eu indico outro voo é claro.

C1	C2	C3	C4	C5	A1	A2	A3	A4	A5	A6
S	S	-	-	-	X					
S	N	S	-	-		X	X			
S	N	N	S	S				X	X	
S	N	N	S	N						X
S	N	N	N	-						X

AS : Digamos agora se a reserva não é para 1ª classe e sim para 2ª classe.

Marcus : Mais ou menos a mesma coisa. Eu checo a RSC. Se o lugar está disponível, atualizo a RSC e emito a passagem de 2ª classe.

AS : OK

C1	C2	C3	C4	C5	A1	A2	A3	A4	A5	A6
S	S	-	-	-	X					
S	N	S	-	-		X	X			
S	N	N	S	S				X	X	
S	N	N	S	N						X
S	N	N	N	-						X
N	S	-	S	-				X	X	
N	S	-	N							

Marcus : Eu suponho que voce deseja saber o que eu faço quando tem um pedido para 2ª classe e não há lugar nela.

AS : Por favor.

Marcus : Bem, devido a taxa alta para a 1ª classe, raramente um passageiro de 2ª classe aceita viajar na 1ª classe. Então eu primeiro pergunto se aceita troca de classe e se aceitar eu checo se RPC esta disponível

C1	C2	C3	C4	C5	C6	A1	A2	A3	A4	A5	A6
S	S	-	-	-	-	X					
S	N	S	-	-	-		X	X			
S	N	N	S	S	-				X	X	
S	N	N	S	N	-						X
S	N	N	N	-	-						X
N	S	-	S	-	-				X	X	
N	S	-	N	S	S						

Observe que para não quebrar o procedimento, foi repetida a condição C3, como condição dummy e C6.

AS : E se há um lugar na RPC.

Marcus : Eu atualizo a RPC e emito uma passagem de 2ª classe.

C1	C2	C3	C4	C5	C6	A1	A2	A3	A4	A5	A6
S	S	-	-	-	-	X					
S	N	S	-	-	-		X	X			
S	N	N	S	S	-				X	X	
S	N	N	S	N	-						X
S	N	N	N	-	-						X
N	S	-	S	-	-				X	X	
N	S	-	N	S	S		X	X			
N	S	-	N	S	N						

AS : Bem, eu suponho que voce indique outro voo se não há espaço disponível?

Marcus : Está certo.

As : É a mesma coisa se a troca não é aceita?

Marcus : Sim.

C1	C2	C3	C4	C5	C6	A1	A2	A3	A4	A5	A6
S	S	-	-	-	-	X					
S	N	S	-	-	-		X	X			
S	N	N	S	S	-				X	X	
S	N	N	S	N	-						X
S	N	N	N	-	-						X
N	S	-	S	-	-				X	X	
N	S	-	N	S	S		X	X			
N	S	-	N	S	N						X
N	S	-	N	N	-						X
N	N										

AS : Finalmente, há o caso onde o pedido de reserva não é nem 1ª classe nem 2ª classe. Eu suponho que seja um erro.

Marcus : Está certo.

AS : Nós cobrimos todos os pontos do procedimento básico.

Marcus : Sim.

AS : Nós podemos olhar agora os documentos que voce usa para isto, tudo...

A tabela final ficaria:

C1 Reserva é 1ª classe?
C2 Reserva é 2ª classe?
C3 RPC está disponível?
C4 RSC está disponível?
C5 Aceita alternativa?
C6 RPC está disponível?

A1 Rejeito
A2 Atualizo RPC
A3 Emito passagem de 1ª classe
A4 Atualizo RSC
A5 Emito passagem de 2ª classe
A6 Indico outro voo

C1	C2	C3	C4	C5	C6	A1	A2	A3	A4	A5	A6	contador de regras simples.
S	S	-	-	-	-	X						16
S	N	S	-	-	-		X	X				8
S	N	N	S	S	-				X	X		2
S	N	N	S	N	-						X	2
S	N	N	N	-	-						X	4
N	S	-	S	-	-				X	X		8
N	S	-	N	S	S		X	X				2
N	S	-	N	S	N						X	2
N	S	-	N	N	-						X	4
N	N	-	-	-	-	X						16
												64
												regra simples

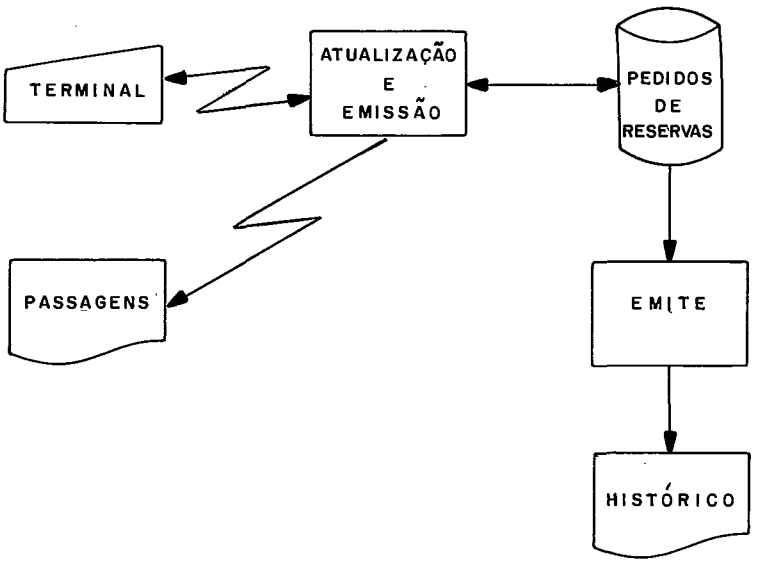
Teste: $2^6 = 64$ está OK

2. ANÁLISE

Com base no pedido do usuário, no estudo de viabilidade e na busca dos fatos, o analista envia uma proposta de sistema para ser aprovada pelo usuário.

3. PROJETO LÓGICO

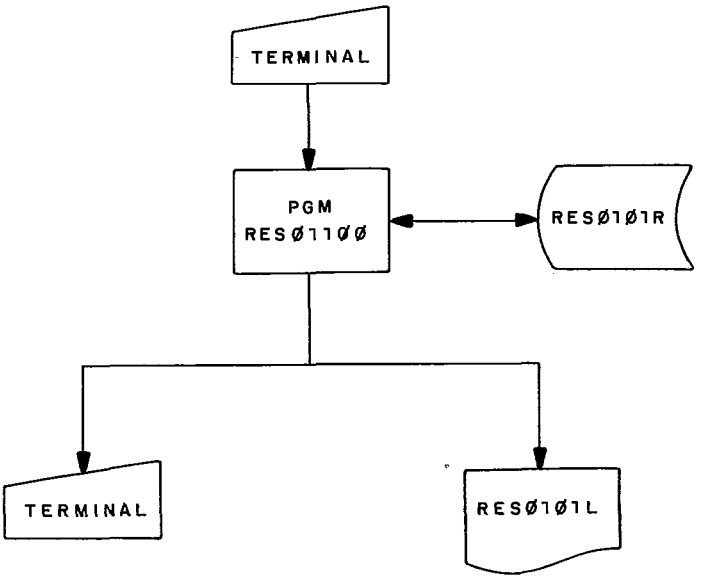
* As partes de consolidação das definições, consolidação das entradas, definição do fluxo de informação, definição da rotina administrativa e planejamento do projeto físico, não serão apresentados.



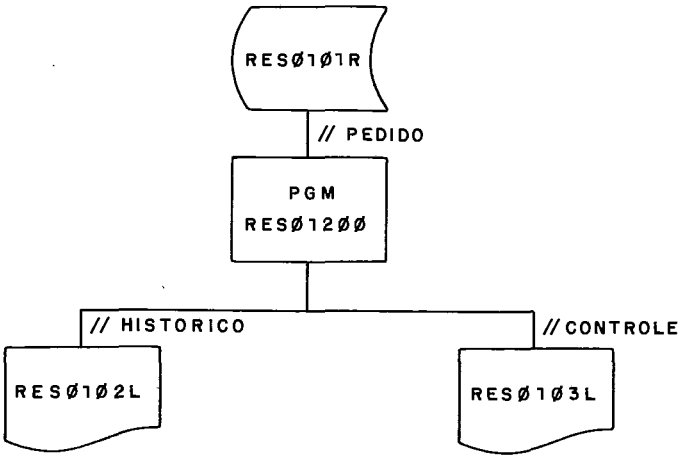
4. PROJETO FÍSICO

DESENHO DO SISTEMA

ON LINE:



BATCH:



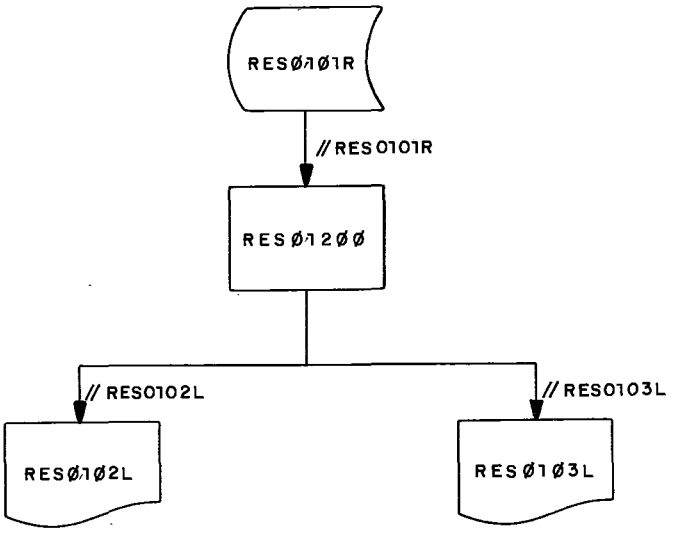
. Só será feito o projeto físico em batch

DEFINIÇÃO DO PGM RES012ØØ

I- NOME : HISTÓRICO DIÁRIO

II- OBJETIVO : Simular todas as transações feitas durante o dia, emitindo um relatório dessas transações.

III- DIAGRAMA:



IV - DESCRIÇÃO DO PROBLEMA

Existem 2 tipos de pedidos de reservas de passagens: 1ª classe e 2ª classe. Conforme a disponibilidade de lugar é emitida uma passagem e atualizado os mapas de controle (RPC - reserva de 1ª classe e RSC - reserva de 2ª classe). Caso não haja disponibilidade na classe pedido é feita troca de classe, caso o passageiro aceite a troca. No caso de não disponibilidade e não troca o passageiro será indicado para viajar em outro voo. Ver RES/3.1/TABRES que contém o procedimento completo. As transações deverão aparecer em relatório. RPC e RSC poderão conter no máximo 10 lugares.

V ANEXOS

- Descrição dos arquivos RES0101R e RES0102L
- Descrição dos registros RES0101R, RES0102L e RES0103L

- Lay-out dos relatórios RES0102L e RES0103L
- Includes - RES01B01 - estrutura do arquivo RES0101R
- Massa de teste (VOLUME = 000001, UNIT = 3330,
  DSN = TESTE, DCB = 80 x 800)

Uma vez enviada a definição do programa para o programador, este deverá devolver:

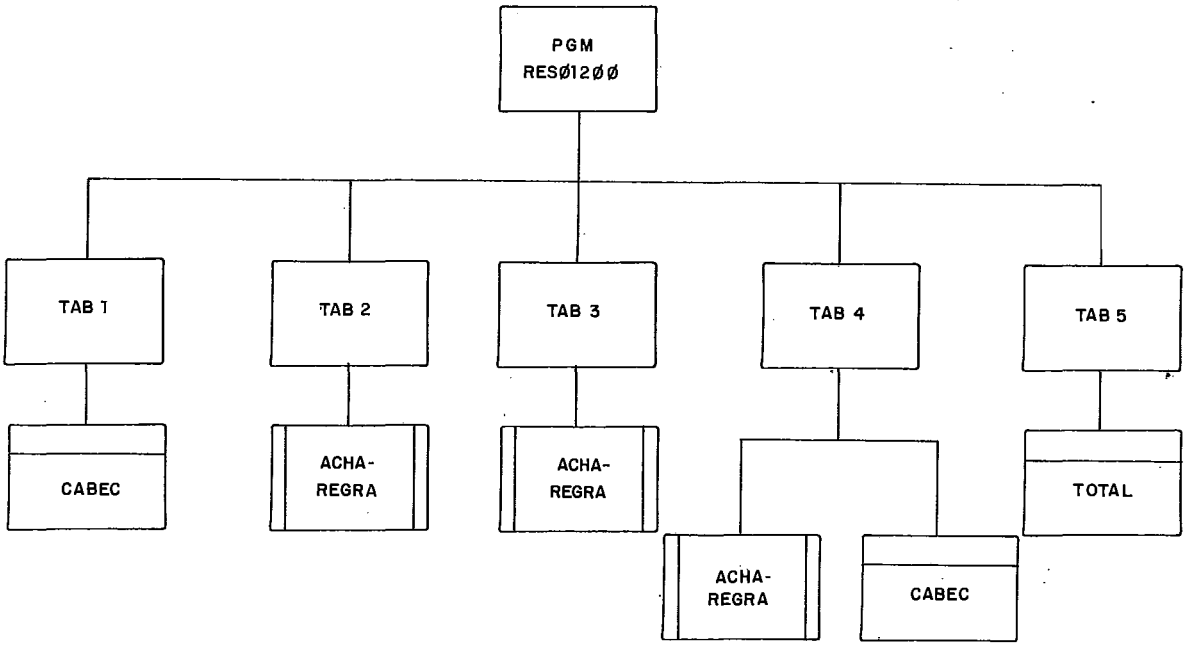
VI - Documentação do programa.

- Tabelas de Decisões.
- Interligação de módulos.

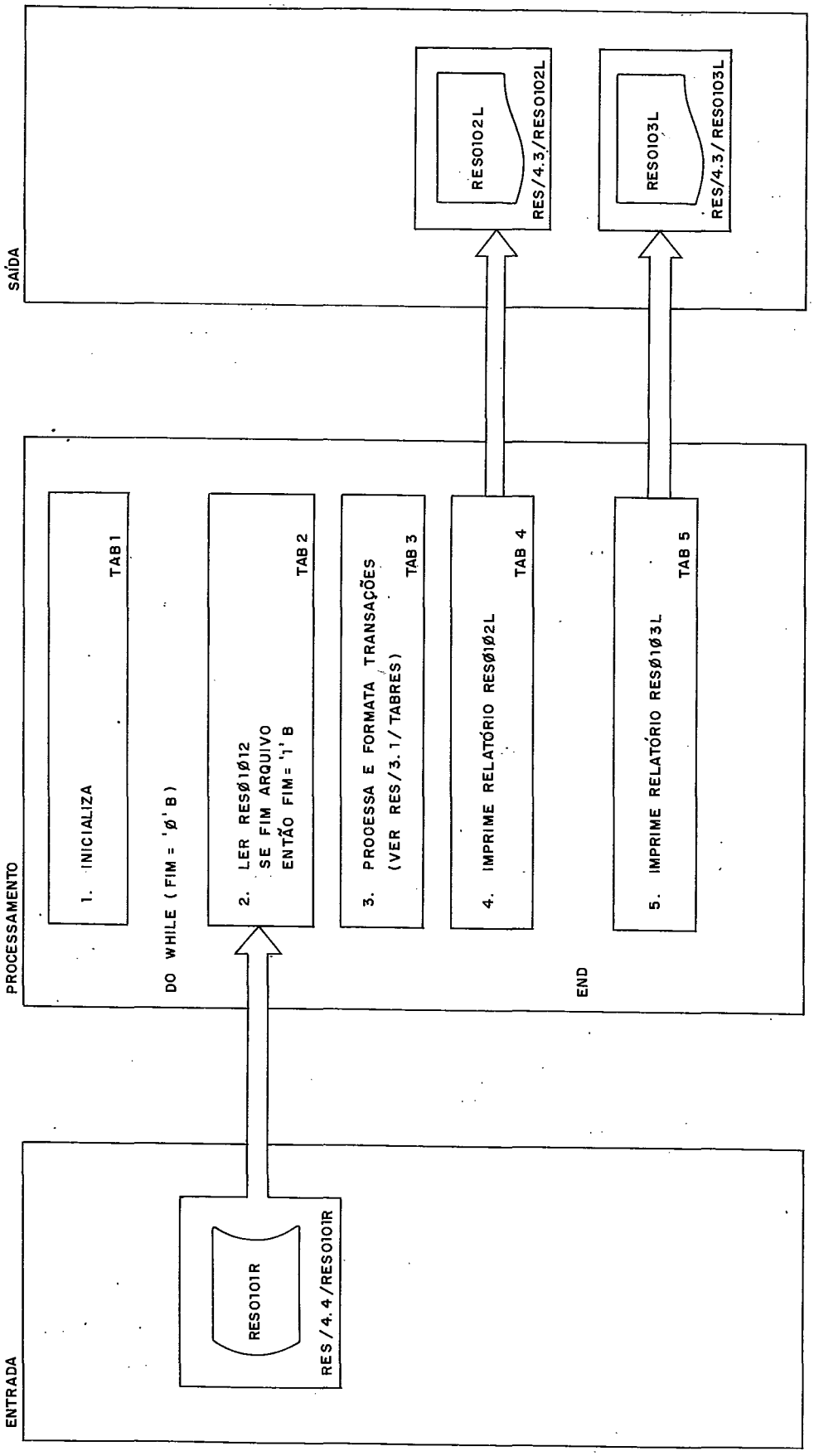
VII - Programa.

Teste c/ a massa de dados.

TITULO	SISTEMA	DOC.	NOME	FOLHA
DEFINIÇÃO DO PGM RESØ12ØØ	RES	3.1	RESØ12ØØ	



TÍTULO	SISTEMA	DOC.	NOME	FOLHA
DEFINIÇÃO DO PGM PESØ12ØØ	RES	3.1	RESØ12ØØ	2



TÍTULO	SISTEMA	DOC.	NOME	FOLHA
REUNIÃO DE LEVANTAMENTO DE INFORMAÇÃO	RES	2	REV 5	1

OBJETIVO

995

LEVANTAR PROCEDIMENTO PARA RESERVA DE PASSAGENS

DATA	LOCAL	DURAÇÃO	RESPONSÁVEL
17/10/77	DIVISÃO DE ATENDIMENTO	1 HORA	WALTER

PARTICIPANTES

MARCUS RODRIGO - DIVISÃO DE ATENDIMENTO

WALTER DOMINGUEZ - COMPUTAÇÃO

RESUMO

1. TABELA DE DECISÃO COM TODO O PROCEDIMENTO DE PEDIDO DE RESERVA

2. ABREVIAÇÃO

RPC - RESERVA DE 1º CLASSE

RSC - RESERVA DE 2º CLASSE

REFERENCIA

TÍTULO	SISTEMA	DOCUMENTO	NOME	FOLHA
TABELA DE DECISÃO DE PROCEDIMENTO PARA RESERVAS DE PASSAGENS	RES	3.1	TABRES	1 299

TABRES																			
RESERVA É 1a CLASSE?	S	S	S	S	S	S	N	N	N	N	N								
RESERVA É 2a CLASSE?	S	N	N	S	N	N	S	S	N	S	N								
RPC ESTÁ DISPONÍVEL?	N	N	N	S	N	N	S	N	N	N	N								
RSC ESTÁ DISPONÍVEL?	N	N	N	S	N	-	-	S	S	N	-								
ACEITA ALTERNATIVA?	-	-	-	-	-	-	-	S	N	-	-								
RPC ESTÁ DISPONÍVEL?																			
REJEITA PEDIDO	X	X									X								
ATUALIZA RPC			X					X											
EMITE PASSAGEM 1a CLASSE			X					X											
ATUALIZA RSC				X															
EMITE PASSAGEM 2a CLASSE					X	X			X	X									
INDICA OUTRO VOO																			

101 NOME/PARÂMETRO/COMENTÁRIO

PARAMETRO																	
NOME	DEFINIÇÃO	TAB 1		TAB 2		TAB 3		TAB 4		TAB 5		CABEÇALHO		TOTAL		ACHA. REGRA	
		E/S	ORD	E/S	ORD	E/S	ORD	E/S	ORD	E/S	ORD	E/S	ORD	E/S	ORD	E/S	ORD
RPC	RESERVA DE 1ª CLASSE	X				X											
RSC	RESERVA DE 2ª CLASSE	X				X											
NLINE	Nº DE LINHAS	X						X									
TULINE	TOTAL LINHAS	X						X						E			
NLIDOS	REG. LIDOS	X		X										E			
NPAG	Nº DE PAGINA	X						X						E			
ESTADO	TAB. ATUAL	X		X		X		X		X							
REGRA	REGRA ESCOLHIDA	X		X		X		X									
NREGRA	REGRA ACHADA															S	
TABELA	TAB. DE ENTRADA															E	
VETOR	COMB. POSSÍVEIS															E	
ENTRADA	VETOR DE COMBINAÇÃO			X		X		X									
FIM										X							
DATA														X			

TÍTULO: COMUNICAÇÃO ENTRE MODULOS

SISTEMA: RES

DOC.: 3.1

NOME: RESØ12ØØ

FOLHA: 7

DESCRIÇÃO DO ARQUIVO	SISTEMA	DOC.	NOME	FOLHA
PEDIDOS DE RESERVA	RES	4.4	RESØ1Ø1R	1

TIPO DO ARQUIVO — ENTRADA [ ] SAÍDA [ ] MESTRE [X] VARIÁVEL [ ] TEMPORÁRIO [ ]

ORGANIZAÇÃO DO ARQUIVO: SEQUENCIAL — 931

MEIO DE ARMAZENAMENTO — FITA MAG. [ ] DISCO [ ] SIMPLES [X] MÚLTIPLO [ ]

RETENÇÃO	Nº GERAÇÕES	Nº BACKUP
10 DIAS	1	1

PROCEDIMENTO DE RECUPERAÇÃO

CHAVE	CLASSIFICAÇÃO
Nº DO PEDIDO	Nº DO PEDIDO

LABEL	DSNAME

NÍVEL	NOME DO REG./ REFERÊNCIA	TAMANHO	UNIDADE	FORMATO	OCORRÊNCIA

BLOCO/BATCH		UNIDADE DE ARMAZ.	NÚMERO DE BLOCOS	
ATUAL, P/ TAM. FIXO	MÁXIMO P/ TAM. VAR.		MÉDIO	MÁXIMO
80		REGISTROS [ ] BYTES [X] CARACTERES [ ] CARTÕES [ ]		
TAMANHO DO ARQUIVO			TAXA DE CRESCIMENTO	
MÉDIO	MÁXIMO		1% - MES	

SÓ PARA FITA	TRILHAS	DENSIDADE	VELOCIDADE	TAMANHO

SÓ PARA ACESSO DIRETO	MÉTODO DE ACESSO	DENSIDADE PACK	FREQ. P/ REORGANIZAÇÃO

NÍVEL	TIPO DE OVERFLOW	TAMANHO DAS ÁREAS DE OVFL.

RELACIONAMENTO C/ OUTROS ARQUIVOS

VERSÃO: RESP:

DATA: OBS:

CAS

```
TAB1.. /* INICIALIZA VARIAVEIS */

       * TAB2 * RPC=10,. RSC=10 ,. NCOND=7,. NLINE=5,. NLIDOS=0,. NIMP=0,. NPAG=0,. CALL CABEC,.


TAB2.. /* LE ARQUIVO RESO101R */

 ON ENDFILE
 (ARQUIVO)

     S          * TAB5 *

     N          * TAB3 * READ FILE (RESO102R) INTO (PEDIDO),. NLIDOS=NLIDOS + 1,. SAIDA =     ,.


TAB3.. /* PROCESSA E FORMATA TRANSACOES */

 1 CLASSE= *                                    SAIDA=PEDIDO ,BY NAME
    2 CLASSE= *                                   ACAO TOMADA= PEDIDO REJEITADO
       RPC GT 0                                      RPC=RPC-1
           RCS GT 0                                     ACAO TOMADA= EMITE PASSAGEM 1 CLASSE
              ALTERNATIVA= *                               RSC=RSC-1
                 RPC GT 0                                      ACAO TOMADA= EMITE PASSAGEM 2 CLASSE
                                                                  ACAO TOMADA= INDICA OUTRO VOO

 C1 C2 C3 C4 C5 C6                              A1 A2 A3 A4 A5 A6 A7

 S  S  -  -  -  -        * TAB4 *              X  X
 S  N  S  -  -  -        * TAB4 *              X     X  X
 S  N  N  S  S  -        * TAB4 *              X           X  X
 S  N  N  S  N  -        * TAB4 *              X                 X
 S  N  N  N  -  -        * TAB4 *              X                 X
 N  S  -  S  -  -        * TAB4 *              X           X  X
 N  S  -  N  S  S        * TAB4 *              X     X  X
 N  S  -  N  S  N        * TAB4 *              X                 X
 N  S  -  N  N  -        * TAB4 *              X                 X
 N  N  -  -  -  -        * TAB4 *              X  X


TAB4.. /* IMPRIME RELATORIO RESO102L E VOLTA A LER ARQUIVO RESO101R */

 NLINE LT 50                                    NLINE=NLINE +1,. BYTE=0,.
                                                   TNLINE= TNLINE + NLINE ,. NLINE = 5,.BYT=1,.   NPAG = NPAG + 1,.CALL CABEC,.
                                                        WRITE FILE(RESO102L) FROM (SAIDA),.

 C1                                             A1 A2 A3

 S                       * TAB2 *               X

 N                       * TAB2 *                  X  X

TAB5.. /* IMPRIME RELATORIO RESO103L */

       * PARE * FIM =  1 B,. CALL TOTAL,.
```

SOURCE LISTING

```
STMT LEV NT

                /* PGM RES01200 -   VERSAO 1 DE 01/11/77          */
    1       0           RES01200.. PROC OPTIONS (MAIN),.
                /*********************************************************************
                 *                                                                   *
                 * TIPO - EMISSAO DE RELATORIO                                       *
                 *                                                                   *
                 * PARAMETROS (ENTRADA) - NAO TEM                                    *
                 *                                                                   *
                 * MODULOS EXTERNOS - NAO TEM                                        *
                 *                                                                   *
                 * ARQUIVOS - RES0101R - PEDIDOS DE RESERVAS                         *
                 *                       CLASSIFICADO P/ PEDIDOS                     *
                 *                       80 X 800  - SEQUENCIAL - INPUT              *
                 * RELATORIOS-RES0102L - HISTORICO DIARIO                            *
                 *                       CLASSIFICADO P/ PEDIDOS                     *
                 *                       133 X 133 - OUTPUT                          *
                 *            RES0103L - REDE DE CONTROLE                            *
                 *                       133 X 133 - OUTPUT                          *
                 *                                                                   *
                 * INCLUDES - RES01801                                               *
                 *                                                                   *
                 * REFERENCIA DE CONTROLE - RES0101R - CLASSE 1,CLASSE 2,ALTERNATIVA *
                 *                                                                   *
                 *********************************************************************/
```

```
STMT LEV NT

                /* TABELAS DE DESICOES DO PROGRAMA */

   2    1  0          DCL TABELAS (5) ENTRY VARIABLE INIT(TAB1,
                                                        TAB2,
                                                        TAB3,
                                                        TAB4,
                                                        TAB5),.

                /* TABELA DE TRANSICAO */

   3    1  0           DCL TRANSICAO (0..5,10) BIN FIXED INIT(
                  /*REGRA     1 2 3 4 5 6 7 8 9 10
                    ESTADO                          */
                  /*  0   */  1,1,1,1,1,1,1,1,1,1,
                  /*TAB 1 */  2,2,2,2,2,2,2,2,2,2,
                  /*TAB 2 */  5,3,0,0,0,0,0,0,0,0,
                  /*TAB 3 */  4,4,4,4,4,4,4,4,4,4,
                  /*TAB 4 */  2,2,2,2,2,2,2,2,2,2,
                  /*TAB 5 */  0,0,0,0,0,0,0,0,0,0 ) ,.

                /* VARIAVEIS DE CONTROLES */

   4    1  0          DCL ESTADO BIN FIXED INIT(0), /*TABELA ATUAL*/
                          REGRA  BIN FIXED INIT(1), /*REGRA SELECIONADADA */
                          FIM    BIT       INIT('0'B), /*.FIM DE PROGRAMA */
                          PROX TAB BIN FIXED,          /* INDICA PROXIMA TABELA */
                          ENTRADA CHAR(10),            /* CONCATENA CONDICOES   */
                          ENT(10) CHAR(1) DEF ENTRADA,
                          COL  BIN FIXED,           /* COL DA MATRIZ ACAO */
                          (DIM,DATE) BUILTIN,
                          DATA    CHAR(08) INIT ('01/11/77'),
                          DIME    BIN FIXED ,.         /* N CONDICOES DA TABELA */

                /* CONTADORES */

   5    1  0          DCL (NLINE,              /* N DE LINHAS P/ PAGINA  */
                          TNLINE,              /* TOTAL N DE LINHA       */
                          NPAG,                /* N DE PAGINAS           */
                          NLIDOS) BIN FIXED,.  /* N. REG. LIDOS          */

                /* ARQUIVOS */

   6    1  0          DCL RES0101R FILE RECORD INPUT,
                          RES0102L FILE RECORD OUTPUT,
                          RES0103L FILE PRINT ,.

                /* ESTRUTURAS */

   7    1  0          DCL 1 PEDIDOS,
                  /*01-03*/    2 N_PEDIDO     PIC '999',
                  /*04-23*/    2 NOME         CHAR(20),
                  /*24-24*/    2 CLASSE_1     CHAR(1),
                  /*25-25*/    2 CLASSE_2     CHAR(1),
                  /*26-26*/    2 ALTERNATIVA  CHAR(1),
```

```
STMT LEV NT

                /*27-80*/     2 RESTO         CHAR(54),.

  8   1  0          DCL 1 SAIDA,
                /*01-01*/     2 BYTE          CHAR(1),
                /*02-06*/     2 FIL1          CHAR(5) INIT(' '),
                /*07-09*/     2 N_PEDIDO      PIC'999',
                /*10-15*/     2 FIL2          CHAR(6) INIT(' '),
                /*16-35*/     2 NOME          CHAR(20),
                /*36-42*/     2 FIL3          CHAR(7) INIT(' '),
                /*43-43*/     2 CLASSE_1      CHAR(1),
                /*44-50*/     2 FIL4          CHAR(7) INIT(' '),
                /*51-51*/     2 CLASSE_2      CHAR(1),
                /*51-61*/     2 FIL5          CHAR(11) INIT(' '),
                /*62-62*/     2 ALTERNATIVA   CHAR(1),
                /*63-72*/     2 FIL6          CHAR(10) INIT(' '),
                /*73-74*/     2 RPC           PIC'99',
                /*75-81*/     2 FIL7          CHAR(6) INIT(' '),
                /*82-83*/     2 RSC           PIC '99',
                /*84-89*/     2 FIL8          CHAR(6) INIT(' '),
                /*90-115*/    2 ACAO_TOMADA   CHAR(26),
                /*116-133*/   2 RESTO         CHAR(18) INIT(' '),.

  9   1  0          DCL 1 CAB,
                /*01-01*/     2 BYT   CHAR(1),
                /*02-133*/    2 LINHA CHAR(132) INIT (' '),.
 10   1  0          DCL 1 LINHA7,
                /*01-01*/     2 BYTS  CHAR(1),
                /*02-05*/     2 FIL1  CHAR(4),
                /*06-13*/     2 NOM1  CHAR(8) INIT ('N.PEDIDO'),
                /*14-19*/     2 FIL2  CHAR(6) INIT (' '),
                /*20-32*/     2 NOM2  CHAR(13) INIT ('C L I E N T E'),
                /*33-40*/     2 FIL3  CHAR(8) INIT (' '),
                /*41-51*/     2 NOM3  CHAR(11) INIT('TIPO PEDIDO'),
                /*52-61*/     2 FIL4  CHAR(10) INIT (' '),
                /*62-67*/     2 NOM4  CHAR(6)  INIT ('ACEITA'),
                /*68-74*/     2 FIL5  CHAR(7)  INIT (' '),
                /*75-83*/     2 NOM5  CHAR(9)  INIT ('N       N'),
                /*84-95*/     2 FIL6  CHAR(12) INIT (' '),
                /*96-106*/    2 NOM6  CHAR(11) INIT ('ACAO TOMADA'),
                /*107-133*/   2 RESTO CHAR(27) INIT (' '),.
 11   1  0          DCL 1 LINHA8,
                /*01-01*/     2 BYT8 CHAR(1) INIT('0'),
                /*02-38*/     2 FIL1 CHAR(37) INIT(' '),
                /*39-69*/     2 NOM1 CHAR(31) INIT('CLASSE 1 CLASSE 2   ALTERNATIVA'),
                /*70-73*/     2 FIL2 CHAR(4) INIT(' '),
                /*74-84*/     2 NOM2 CHAR(11) INIT('RPC    RSC'),
                /*85-133*/    2 RESTO CHAR(49) INIT(' '),.
```

```
    STMT LEV NT

      12   1  0          DO WHILE (FIM='0'B),.
      13   1  1             PROX TAB = TRANSICAO (ESTADO,REGRA),.
      14   1  1             CALL TABELAS (PROX TAB),.
      15   1  1          END,.
```

```
STMT LEV NT

  16   1  0   TAB1.. PROC,.
              /*********************************************************************
              *                                                                    *
              * TIPO - INICIALIZA                                                  *
              *                                                                    *
              * PARAMETROS (ENTRADA) - NAO TEM                                     *
              *                                                                    *
              * PARAMETROS (SAIDA)   - NAO TEM                                     *
              *                                                                    *
              * VARIAVEIS ALTERADAS - RPC,RSC,NLINE,TNLINE,NLIDOS,NPAG,            *
              *                       ESTADO,REGRA                                 *
              *                                                                    *
              * MODULOS EXTERNOS - CABEC - ROTINA DE CABECARIO DE RES0102L         *
              *                                                                    *
              * REFERENCIA DE CONTROLE - NAO TEM                                   *
              *                                                                    *
              * ARQUIVOS - NAO TEM                                                 *
              *                                                                    *
              * RELATORIOS - NAO TEM                                               *
              *                                                                    *
              *********************************************************************/

  17   2  0           RPC,RSC=10,.
  18   2  0           NLINE = 5,.
  19   2  0           TNLINE,NLIDOS,NPAG=0,.
  20   2  0           CALL CABEC,.
  21   2  0           ESTADO,REGRA=1,.
  22   2  0   END TAB1,.
```

```
    STMT LEV NT

      23   1  0  TA82.. PROC,.
                 /*************************************************************************
                  *                                                                       *
                  * TIPO - LEITURA                                                        *
                  *                                                                       *
                  * PARAMETROS (ENTRADA) - NAO TEM                                        *
                  *                                                                       *
                  * PARAMETROS (SAIDA)   - NAO TEM                                        *
                  *                                                                       *
                  * VARIAVEIS ALTERADAS - NLIDOS,ESTADO,REGR                              *
                  *                                                                       *
                  * MODULOS EXTERNOS -  NAO TEM                                           *
                  *                                                                       *
                  * REFERENCIA DE CONTROLE - NAO TEM                                      *
                  *                                                                       *
                  * ARQUIVOS - RES0101R                                                   *
                  *                                                                       *
                  * RELATORIOS - NAO TEM                                                  *
                  *                                                                       *
                  *************************************************************************/

      24   2  0          DCL 1 TABELA2(2),
                            2 REGRA2(1) CHAR(1) INIT(
                                  'S',
                                  'N' ),.
      25   2  0          DCL EXECUTA2(2,1) ENTRY VARIABLE INIT(A1,
                                                            A2),.
      26   2  0          DIME= DIM(EXECUTA2,2),.
      27   2  0          ON ENDFILE (RES0101R)
                           BEGIN,.
      28   3  0             NLIDOS = NLIDOS - 1,.
      29   3  0             REGRA = 1,.
      30   3  0          END,.
      31   2  0          ENTRADA='0',.
      32   2  0          REGRA = ACHA REGRA(TABELA2,ENT),.
      33   2  0          DO COL = 1 TO DIME,.
      34   2  1             CALL EXECUTA2 (REGRA,COL),.
      35   2  1          END,.
      36   2  0          ESTADO = 2,.

      37   2  0  A1..PROC,.

      38   3  0  END A1,.

      39   2  0  A2..PROC,.
      40   3  0          READ FILE (RES0101R) INTO (PEDIDOS),.
      41   3  0          NLIDOS             = NLIDOS + 1,.
      42   3  0          SAIDA.N PEDIDO     = 0,.
      43   3  0          SAIDA.CLASSE 1     = ' ',.
      44   3  0          SAIDA.CLASSE 2     = ' ',.
      45   3  0          SAIDA.ALTERNATIVA  = ' ',.
      46   3  0          SAIDA.ACAO TOMADA  = ' ',.
      47   3  0          SAIDA.RESTO        = ' ',.
      48   3  0  END A2,.
```

```
    STMT LEV NT

      49   2  0       END TAB2,.
```

```
STMT LEV NT

 50   1  0  TAB3.. PROC,.
             /*****************************************************************
              *                                                               *
              * TIPO - PROCESSA E FORMATA SAIDA                               *
              *                                                               *
              * PARAMETROS (ENTRADA) - NAO TEM                                *
              *                                                               *
              * PARAMETROS (SAIDA)   - NAO TEM                                *
              *                                                               *
              * VARIAVEIS ALTERADAS - RPC,RSC,ESTADO,REGRA                    *
              *                                                               *
              * MODULOS EXTERNOS -  NAO TEM                                   *
              *                                                               *
              * REFERENCIA DE CONTROLE - RES0101R - CLASSE 1,CLASSE 2,ALTERNATIVA*
              *                                                               *
              * ARQUIVOS - NAO TEM                                            *
              *                                                               *
              * RELATORIOS - NAO TEM                                          *
              *                                                               *
              *****************************************************************/

 51   2  0          DCL 1 TABELA3(10),
                        2 REGRA3(6) CHAR(1) INIT(
                                  'S','S','-','-','-','-',
                                  'S','N','S','-','-','-',
                                  'S','N','N','S','S','-',
                                  'S','N','N','S','N','-',
                                  'S','N','N','N','-','-',
                                  'N','S','-','S','-','-',
                                  'N','S','-','N','S','S',
                                  'N','S','-','N','S','N',
                                  'N','S','-','N','N','-',
                                  'N','N','-','-','-','-' ),.
 52   2  0          DCL EXECUTA3 (0..10,3) ENTRY VARIABLE INIT(
                                   A,A,A,   /* ELSE */
                                  A1,A2,A,
                                  A1,A3,A4,
                                  A1,A5,A6,
                                  A1,A7,A,
                                  A1,A7,A,
                                  A1,A5,A6,
                                  A1,A3,A4,
                                  A1,A7,A,
                                  A1,A7,A,
                                  A1,A2,A ),.
 53   2  0          DIME = DIM(EXECUTA3,2),.
 54   2  0          ENTRADA = (PEDIDOS.CLASSE 1 = 'X') CAT (PEDIDOS.CLASSE 2 = 'X')
                     CAT (RPC GT 0) CAT (RSC GT 0) CAT
                     (PEDIDOS.ALTERNATIVA='X') CAT (RPC GT 0),.
 55   2  0          REGRA = ACHA REGRA (TABELA3,ENT),.
 56   2  0          DO COL = 1 TO DIME,.
 57   2  1             CALL EXECUTA3 (REGRA,COL),.
 58   2  1          END,.
 59   2  0          ESTADO = 3,.
```

```
STMT LEV NT

 60   2  0  A..PROC,.
             /***  ACAO NULA  ***/
 61   3  0  END A,.

 62   2  0  A0..PROC,.
 63   3  0          ACAO TOMADA = 'ERRO - NAO ACHOU REGRA',.
 64   3  0  END A0,.

 65   2  0  A1.. PROC,.
 66   3  0          SAIDA = PEDIDOS,BYNAME,.
 67   3  0  END A1,.

 68   2  0  A2.. PROC,.
 69   3  0          ACAO TOMADA = 'PEDIDO REJEITADO',.
 70   3  0  END A2,.

 71   2  0  A3.. PROC,.
 72   3  0          RPC = RPC - 1,.
 73   3  0  END A3,.

 74   2  0  A4.. PROC,.
 75   3  0          ACAO TOMADA = 'EMITE PASSAGEM DE 1 CLASSE',.
 76   3  0  END A4,.

 77   2  0  A5.. PROC,.
 78   3  0          RSC = PSC - 1,.
 79   3  0  END A5,.

 80   2  0  A6.. PROC,.
 81   3  0          ACAO TOMADA = 'EMITE PASSAGEM DE 2 CLASSE',.
 82   3  0  END A6,.

 83   2  0  A7.. PROC,.
 84   3  0          ACAO TOMADA = 'INDICA OUTRO VOO',.
 85   3  0  END A7,.
 86   2  0  END TAB3,.
```

```
STMT LEV NT

  87   1  0  TAB4.. PROC,.
              /*************************************************************************
               *                                                                       *
               * TIPO - EMISSAO                                                        *
               *                                                                       *
               * PARAMETROS (ENTRADA) - NAO TEM                                        *
               *                                                                       *
               * PARAMETROS (SAIDA)   - NAO TEM                                        *
               *                                                                       *
               * VARIAVEIS ALTERADAS - NLINE,NPAG,TNLINE,ESTADO,REGRA                  *
               *                                                                       *
               * MODULOS EXTERNOS - CABEC - ROTINA DE CABECARIO DE RES0102L            *
               *                                                                       *
               * REFERENCIA DE CONTROLE - NAO TEM                                      *
               *                                                                       *
               * ARQUIVOS - NAO TEM                                                    *
               *                                                                       *
               * RELATORIOS - RES0102L                                                 *
               *                                                                       *
               *************************************************************************/

  88   2  0          DCL 1 TABELA4(2),
                        2 REGRA4(1) CHAR(1) INIT(
                                'S',
                                'N' ),.

  89   2  0          DCL 1 EXECUTA4(2,2) ENTRY VARIABLE INIT(
                                A1,A3,
                                A2,A3),.
  90   2  0          DIME = DIM(EXECUTA4,2),.
  91   2  0          ENTRADA = (NLINE LE 58),.
  92   2  0          REGRA = ACHA_REGRA (TABELA4,ENT),.
  93   2  0          DO COL = 1 TO DIME,.
  94   2  1             CALL EXECUTA4(REGRA,COL),.
  95   2  1          END,.
  96   2  0          ESTADO = 4,.

  97   2  0  A1.. PROC,.
  98   3  0          NLINE = NLINE + 1,.
  99   3  0          BYTE = 0,.
 100   3  0  END A1,.

 101   2  0  A2.. PROC,.
 102   3  0          TNLINE = TNLINE + NLINE,.
 103   3  0          NLINE = 5,.
 104   3  0          BYT  = 1,.
 105   3  0          NPAG = NPAG + 1,.
 106   3  0          CALL CABEC,.
 107   3  0  END A2,.

 108   2  0  A3.. PROC,.
 109   3  0          WRITE FILE (RES0102L) FROM (SAIDA),.
 110   3  0  END A3,.
 111   2  0  END TAB4,.
```

```
    STMT LEV NT

     112   1  0  TAB5.. PROC,.
                 /*********************************************************************
                  *                                                                   *
                  * TIPO - FINALIZA                                                   *
                  *                                                                   *
                  * PARAMETROS (ENTRADA) - NAO TEM                                    *
                  *                                                                   *
                  * PARAMETROS (SAIDA)   - NAO TEM                                    *
                  *                                                                   *
                  * VARIAVEIS ALTERADAS - FIM,ESTADO,REGRA                            *
                  *                                                                   *
                  * MODULOS EXTERNOS - TOTAL - ROTINA DE TOTAL DE RES0103L            *
                  *                                                                   *
                  * REFERENCIA DE CONTROLE - NAO TEM                                  *
                  *                                                                   *
                  * ARQUIVOS - NAO TEM                                                *
                  *                                                                   *
                  * RELATORIOS - RES0103L                                             *
                  *                                                                   *
                  *********************************************************************/

     113   2  0           FIM = '1'B,.
     114   2  0           CALL TOTAL,.
     115   2  0           ESTADO = 5,.
     116   2  0  END TAB5,.
```

```
    STMT LEV NT

     117   1  0  CABEC.. PROC,.
                 /*********************************************************************
                  *                                                                   *
                  * TIPO - EMISSAO                                                    *
                  *                                                                   *
                  * PARAMETROS (ENTRADA) - NAO TEM                                    *
                  *                                                                   *
                  * PARAMETROS (SAIDA)   - NAO TEM                                    *
                  *                                                                   *
                  * VARIAVEIS ALTERADAS - NAO TEM                                     *
                  *                                                                   *
                  * MODULOS EXTERNOS -  NAO TEM                                       *
                  *                                                                   *
                  * REFERENCIA DE CONTROLE - NAO TEM                                  *
                  *                                                                   *
                  * ARQUIVOS - NAO TEM                                                *
                  *                                                                   *
                  * RELATORIOS - RES0102L                                             *
                  *                                                                   *
                  *********************************************************************
     118   2  0           BYT ='1',.
     119   2  0           LINHA = 'SISTEMA DE PEDIDO DE RESERVA',.
     120   2  0           WRITE FILE (RES0102L) FROM (CAB),.
     121   2  0           BYT ='0',.
     122   2  0           LINHA ='RES0102L - HISTORICO DAS TRANSACOES',.
     123   2  0           WRITE FILE (RES0102L) FROM (CAB),.
     124   2  0           LINHA = (132)' ',.
     125   2  0           WRITE FILE (RES0102L) FROM (CAB),.
     126   2  0           WRITE FILE (RES0102L) FROM (LINHA7),.
     127   2  0           WRITE FILE (RES0102L) FROM (LINHA8),.
     128   2  0           LINHA = (132)' ',.
     129   2  0           WRITE FILE (RES0102L) FROM (CAB),.
     130   2  0     END CABEC,.
```

```
STMT LEV NT

 131   1  0  TOTAL..PROC,.
            /*********************************************************************
             *                                                                   *
             * TIPO - EMISSAO                                                    *
             *                                                                   *
             * PARAMETROS (ENTRADA) - TNLINE,NPAG,NLIDOS,NLINE                   *
             *                                                                   *
             * PARAMETROS (SAIDA)   - NAO TEM                                    *
             *                                                                   *
             * VARIAVEIS ALTERADAS -  NAO TEM                                    *
             *                                                                   *
             * MODULOS EXTERNOS -  NAO TEM                                       *
             *                                                                   *
             * REFERENCIA DE CONTROLE - NAO TEM                                  *
             *                                                                   *
             * ARQUIVOS - NAO TEM                                                *
             *                                                                   *
             * RELATORIOS - RES0103L                                             *
             *                                                                   *
             *********************************************************************/

 132   2  0          TNLINE = TNLINE + NLINE,.
 133   2  0          PUT FILE (RES0103L) EDIT(
                        'SISTEMA DE PEDIDO DE RESERVA','PAGINA ',NPAG,
                        'RES0103L - REDE DE CONTROLE DO PGM RES01200 - ',
                        'VERSAO 1 DE ',DATA)
                        (COL(1),A,COL(110),A,A,COL(1),A,A,A),.
 134   2  0          PUT FILE (RES0103L) EDIT(
                        'ESTATISTICA DO PROCESSAMENTO',
                        'REG. LIDOS..... ',NLIDOS,
                        'LINHAS IMPRESSAS..... ',TNLINE,
                        'PAGINAS IMPRESSAS.... ',NPAG)
                        (LINE(29),COL(54),A,COL(55),A,F(5),COL(55),A,F(5),
                         COL(55),A,F(5)),.
 135   2  0    END TOTAL,.
```

```
STMT LEV NT

 136   1  0     ACHA REGRA.. PROC (TAB,VETOR),.
                /*********************************************************************
                *                                                                    *
                * TIPO - CALCULO                                                     *
                *                                                                    *
                * PARAMETROS (ENTRADA) - NAO TEM                                     *
                *                                                                    *
                * PARAMETROS (SAIDA)   - NREGRA                                      *
                *                                                                    *
                * VARIAVEIS ALTERADAS -  NAO TEM                                     *
                *                                                                    *
                * MODULOS EXTERNOS -  NAO TEM                                        *
                *                                                                    *
                * REFERENCIA DE CONTROLE - NAO TEM                                   *
                *                                                                    *
                * ARQUIVOS - NAO TEM                                                 *
                *                                                                    *
                * RELATORIOS -  NAO TEM                                              *
                *                                                                    *
                *********************************************************************/

 137   2  0          DCL 1 TAB (*),
                        2 TREGRA (*) CHAR(*),.
 138   2  0          DCL VETOR(*) CHAR(*) ,.
 139   2  0          DCL MATRIZ (0..1,*,*) BIT(1) CTL,.
 140   2  0          DCL COLUNA (*) BIT(1) CTL,. /* INDICA TREGRA */
 141   2  0          DCL NREGRA BIN FIXED INIT(0),.
 142   2  0          DCL (DIMV,DIMT,DIMR) BIN FIXED,.
 143   2  0          DIMV = DIM (VETOR,1),.
 144   2  0          DIMT = DIM (TREGRA,2),.
 145   2  0          DIMR = DIM (TREGRA,1),.
 146   2  0          ALLOCATE COLUNA (DIMR),.
 147   2  0          ALLOCATE MATRIZ(0..1,DIMR,DIMT),.
 148   2  0          MATRIZ = '1'B,.
 149   2  0          COLUNA = '1'B,.
 150   2  0          DO I = 1 TO DIMR ,.
 151   2  1             DO J = 1 TO DIMT,.
 152   2  2                SELECT (TREGRA(I,J)),.
 153   2  3                   WHEN ('N')    MATRIZ (1,I,J) = '0'B,.
 154   2  3                   WHEN ('S')    MATRIZ (0,I,J) = '0'B,.
 155   2  3                   OTHERWISE,.
 156   2  3                END,.
 157   2  2             END,.
 158   2  1          END,.
 159   2  0          DO M = 1 TO DIMR,.
 160   2  1             IF VETOR(M) NE '-' AND VETOR(M) NE ' '
                          THEN COLUNA = COLUNA AND MATRIZ (VETOR(M),*,M),.
 161   2  1           END,.
 162   2  0          IF ANY(COLUNA)
                       THEN DO I = 1 TO DIMR,.
 163   2  1                    IF COLUNA(I)
                               THEN DO,.
 164   2  2                         IF NREGRA NE 0
                                    THEN DO,.
```

```
    STMT LEV NT

     165   2  3                                    PUT SKIP EDIT('AMBIGUIDADE NAS ',
                                                   'REGRAS..',NREGRA,' E ',I)
                                                   (COL(5),A,A,F(2),A,F(2));.
     166   2  3                                    NREGRA = -1;.
     167   2  3                                  END;.
     168   2  2                                ELSE NREGRA = I;.
     169   2  2                             END;.
     170   2  1                          END;.
     171   2  0                    RETURN (NREGRA);.
     172   2  0                END ACHA REGRA;.

     173   1  0           END RES01200;.
```

SISTEMA DE PEDIDO DE RESERVA

RES0102L – HISTORICO DAS TRANSACOES

E	N.PEDIDO	C L I E N T E	TIPO PEDIDO		ACEITA	N	N	ACAO TOMADA
			CLASSE 1	CLASSE 2	ALTERNATIVA	RPC	RSC	
	001	WALTER DOMINGUEZ	X			09	10	EMITE PASSAGEM DE 1 CLASSE
	002	JOSE ANTONIO	X			08	10	EMITE PASSAGEM DE 1 CLASSE
	003	JOSE AUGUSTO	X			07	10	EMITE PASSAGEM DE 1 CLASSE
	004	FAISSOL	X			06	10	EMITE PASSAGEM DE 1 CLASSE
	005	CANABRAVA	X			05	10	EMITE PASSAGEM DE 1 CLASSE
	006	CICERO	X			04	10	EMITE PASSAGEM DE 1 CLASSE
	007	ADONIS	X			03	10	EMITE PASSAGEM DE 1 CLASSE
	008	FLAVIO	X			02	10	EMITE PASSAGEM DE 1 CLASSE
	009	WOLYN	X			01	10	EMITE PASSAGEM DE 1 CLASSE
	010	SILVIO	X			00	10	EMITE PASSAGEM DE 1 CLASSE
	011	CARLINDO	X			00	10	INDICA OUTRO VOO
	012	GUIEIRO	X		X	00	09	EMITE PASSAGEM DE 2 CLASSE
	013	ALVARO		X		00	08	EMITE PASSAGEM DE 2 CLASSE
	014	RONALDO	X		X	00	07	EMITE PASSAGEM DE 2 CLASSE
	015	FLORES		X		00	06	EMITE PASSAGEM DE 2 CLASSE
	016	SUSSANA		X		00	05	EMITE PASSAGEM DE 2 CLASSE
	017	MARCELO		X		00	04	EMITE PASSAGEM DE 2 CLASSE
	018	REGINA		X	X	00	03	EMITE PASSAGEM DE 2 CLASSE
	019	FILE		X		00	02	EMITE PASSAGEM DE 2 CLASSE
	020	MARILIA		X	X	00	01	EMITE PASSAGEM DE 2 CLASSE
	021	MARILHA		X		00	00	EMITE PASSAGEM DE 2 CLASSE
	022	MANOEL		X		00	00	INDICA OUTRO VOO
	023	OLIVARES		X	X	00	00	INDICA OUTRO VOO
	024	CABRAL		X		00	00	INDICA OUTRO VOO
	025	JOAO	X			00	00	INDICA OUTRO VOO
	026	FRANCISCO	X		X	00	00	INDICA OUTRO VOO
	027	HELENA	X			00	00	INDICA OUTRO VOO
	028	LUCY	X			00	00	INDICA OUTRO VOO
	029	GLORIA	X		X	00	00	INDICA OUTRO VOO
	030	JOSE AUGUSTO	X	X	X	00	00	PEDIDO REJEITADO

CAPÍTULO IX

CONCLUSÕES

CAPÍTULO IX - CONCLUSÕES

Como pode ser visto durante a esplanação da técnica, sempre as explicações das tabelas foram auxiliadas por um fluxograma. Embora muitos autores comparem tabelas de decisão com fluxograma, aqui não será feita esta comparação, pois pode-se usar as duas técnicas, para o raciocínio lógico. Faremos, sim, uma comparação entre tabelas de decisões e programa (. . linguagem).

As tabelas de decisão:

- São mais facilmente compreendidos, tanto p/ o pessoal de negócios como científico, pois a tabela poderá ser a própria linguagem fonte do computador.
- Tem uma visão dimensional no relacionamento lógico, enquanto programas tem uma visão unidimensional.
- Mais fácil de aprender e montar do que programas.
- Mantem sempre a documentação atualizada, pois qualquer alteração, já estaria alterando os procedimentos em computador.
- Forma resumida e compacta de definição e descrição do sistema para uso em análise, programação e documentação.

- São bastante compreensivos, pois todas as pessoas utilizam-se comumente de tabelas em seu cotidiano. Horários de trens, tabela de quilometragem em mapas, etc, são bons exemplos.

- Pode ter técnicas de verificação e otimização aplicadas a elas para obter precisão e perfeição.

- Garante que todas as possibilidades de ocorrencia de várias condições estejam testadas.

- Não é influenciada pela lógica do autor (maneira de pensar), pois cada um tem uma lógica.

GLOSSARIO

AÇÃO - É uma operação a ser realizada; pode ser uma operação manual, ou uma de máquina ou uma fórmula, um procedimento ou um ítem.

AMBIGUIDADE APARENTE- É uma situação que existe na tabela (geralmente por causa de condições dependentes) a qual é uma contradição ou redundancia, porém na realidade não pode ocorrer.

AMBIGUIDADE GENUINA - É um tipo de erro que pode causar contradição lógica ou redundancia. Quando o texto é real a ambiguidade é real no caso oposto é aparente.

VERIFICAÇÃO ARITMÉTICA - É a comparação entre contador de regras simples em uma tabela completa e o nº de regras simples possíveis que deverão existir em uma tabela.

VERIFICAÇÃO BIFURCAÇÃO - É o cheque feito pela expansão das regras complexas voltado para suas regras simples e pelo teste das contradições e redundancias.

COMBINAÇÃO - É o processo de combinar 2 ou mais regras, as quais tenham as mesmas ações, para formar uma regra complexa. As duas ou mais regras combinadas podem ser simples ou complexas.

CONDIÇÃO - É um fator variável que afeta as ações a serem tomadas em uma dada situação pela presença, ausencia, ou mudança de um valor. Pode ser por exemplo um evento, uma característica, um valor, uma dimensão, ou uma comparação.

CONDIÇÃO NULA - É a segunda ou subsequente repetição de uma condição em uma tabela. Uma repetição de uma condição ocorre comumente quando as tabelas são preparadas pelo desenvolvimento progressivo da regra.

CONDIÇÃO TESTE - É um método de codificar um programa através de uma decisão, onde cada uma regra na tabela é resultante de uma série de desvios de programa.

CONTADOR DE HÍFENS - É o nº total de regras que contem um símbolo de indiferença (-).

CONTADOR DE PESO DO HÍFEN - É a soma do produto da frequencia relativa de uma regra e o contador de regras, para toda regra contando um símbolo de indiferença.

CONTADOR DE REGRAS - É o nº de regras simples representada por uma regra complexa.

CONTRADIÇÃO - É um erro lógico em tabela de decisão na qual duas ou mais regras tem a mesma combinação de condições porém com ações diferentes. A ambiguidade causada por tais erros podem ser: real ou somente aparente.

DECLARAÇÃO DA AÇÃO - É a descrição da ação.

DECLARAÇÃO DAS CONDIÇÕES - É a descrição das condições.

DESENVOLVIMENTO CLÁSSICO - É uma técnica para preparar uma tabela de decisão a qual é mecanizada, na qual todas as condições possíveis são inicialmente definidas e especificadas.

DESENVOLVIMENTO DA REGRA PROGRESSIVA - É uma técnica para preparar uma tabela de decisão na qual as condições/ são agrupadas a medida que as regras são identificadas.

DETAB - É um processador de tabela de decisão desenvolvida pela CODASYL.

ENTRADA EXPANDIDA - É um tipo tabela de decisão, no qual os valores das condições são especificados na matriz de condições. Pode ser aplicada também as ações.

ENTRADA LIMITADA - É um tipo de tabela de decisão no qual todas as condições são declaradas como perguntas as quais tem uma resposta SIM ou NÃO. As entradas permitidas são S (SIM), N (NÃO), - (INDIFERENTE).

ENTRADA MISTA - É um tipo de tabela de decisão na qual tanto as entradas limitadas e entradas expandidas aparecem.

FREQUENCIA RELATIVA DE REGRA - É uma estimativa do nº de vezes que cada regra em uma tabela de decisão é satisfeita para uma média de valores.

INDIFERENÇA - Significa que o valor de uma condição não afeta a ação a ser tomada.

INICIALIZAÇÃO - É uma série de ações a serem chamadas apenas uma vez, no começo do procedimento.

LOOPING - Ver Repetição.

MATRIZ AÇÃO - Em uma tabela é o quadrante do lado direito inferior, onde a execução de uma ação é especificada (entrada expandida) ou marcada (entrada limitada).

MATRIZ CONDIÇÃO - Em uma tabela vertical é o quadrante do lado direito superior, onde o valor das variáveis são especificadas (entrada expandida) ou marcadas (entrada limitada). Na tabela horizontal é o quadrante do lado esquerdo inferior.

ORIGEM DAS AÇÕES - Em uma tabela vertical é o quadrante do lado esquerdo inferior ou em uma tabela horizontal é o quadrante do lado direito superior onde as ações são declaradas.

ORIGEM DAS CONDIÇÕES - Em uma tabela vertical ou horizontal é o quadrante do lado esquerdo superior ou uma tabela horizontal onde as condições são declaradas.

PARSING - É uma técnica para converter uma tabela de decisão em lógica sequencial para minimizar memória.

PREPROCESSADOR - É um programa especial o qual opera sobre um programa codificado em tabela de decisão para produzir um programa codificado. O código gerado é geralmente uma linguagem de alto nível, o qual será executada por um compilador para produzir um código objeto.

REPETIÇÃO - Um desvio condicional é feito, dependendo do valor da condição, onde o valor é marcado ou alterado por uma ação na tabela.

REDUNDANCIA - É um erro lógico na tabela de decisão no qual duas ou mais regras tem a mesma combinação de condições e com as mesmas ações especificadas. A ambiguidade causada pode ser real ou aparente.

REGRA - É uma combinação de valores de condições associadas a valores de ações. Na tabela vertical é uma coluna. Na tabela horizontal é uma linha.

REGRA COMPLEXA - É uma regra que contem pelo menos um símbolo de indiferença, ou seja é uma regra que representa mais de uma regra simples.

REGRA DA MÁSCARA - É um método de codificar de programa, usando tabela de decisão onde uma tabela é uma matriz.

REGRA ELSE - É uma regra que cobre todas as combinações que não forem especificadas.

REGRA SIMPLES - É uma regra na qual não contém símbolo de indiferença.

REGRA ZERO - É uma regra especial, a qual permite inicializações. As entradas de condições ficam em branco e as ações especificadas só são executadas uma vez quando a tabela é chamada.

ROTULO - É a parte da tabela de decisão que identifica a tabela, ou seja o nome da tabela.

TABELA ABERTA- É uma tabela que contem um ou mais desvio a outras tabelas. Especificamente é uma tabela que só tem desvios.

TABELA E (AND) - É a forma mais comum da tabela de decisão, na qual as condições são ligadas por um conector E.

TABELA FECHADA - É uma tabela a qual não contém ações e levam a outra tabela. Especificamente é uma tabela chamada por um CALL.

TABELA OU (OR) - É uma forma rara de tabela de decisão, na qual as condições são alternativas. Se pelo menos um dos valores fixos é verdade será então tomada uma ação.

TESTE SEQUENCIAL DE REGRAS - É uma técnica para converter tabela de decisões em lógica sequencial através de um fluxograma.

VALORES DA AÇÃO - É o conteúdo de uma ação.

VALORES DE UMA CONDIÇÃO - É o conteúdo de uma condição.

BIBLIOGRAFIA

. BAGLIN, G. & KLEE. J. Las tablas de decision, instrumento para el analise informatico. Bilbão, Deusto, 1973. 129 p. (Coleccion Informatico).

. BARNARD, T. J. A new rule mask technique for interpreting decision tables. The Computer Bulletin, 13 (5): 153-4 May 1969.

. BJORK, H. Decision tables in Algol 60. Bet, 8 : 147-53, 1968.

. BROWN, MARCELO PARDO. Programas dirigidos por tabelas. Informativo técnico 04/75. TELERJ

. CHAPMAN, A.E.& CALLAHAN, M.A. A description of the basic algorithm used in the DETAB 65 prepocenor. CACM, 10 (7): 441-6, July 1967.

. CANTRELL, H.N. et alu. Logic. structure tables CACM, 4 (6): 272-5, June 1961.

. CAVOURAS, J.C. On the conversion of programs to decision tables: method and objectives. CACM, 17 (8): 456 - 62. / Aug. 1974.

. CHAPIN, N. Parsing of decision tables. CACM, 10 (8): 507--12, Aug. 1967.

. CHENG, C-W & RABIN, J. Syntheses of decision rules. CACM, 19 (7): 404-6, July 1976.

. COHEN, A. Modular programs: defining the module. Datamation, 18 (1): 34-7 Jan. 1972.

. DANIELS, A. & YEATES, D. Formação básica em análise de sistemas. Rio de Janeiro, Livros Técnicos e Científicos. 1974 230 p. (Série LTC/LTD).

. DATHE, G. Conversion of decision tables by rule mask method without rule mask. CACM, 15 (10): 906- 9. Oct. 1972.

. DIAL, R.B. Decision table translation. CACM, 13 (9): 571-2 Sept. 1970.

. EGLER, J.F. A procedure for converting logic table conditions into on efficient sequence of test instruction. CACM, 6 (8): 510-4. Sept. 1963.

. FISHER, D.L. Data, documentation and decison tables. CACM, 9 (1): 26-31, Jan. 1966.

. FLOYD, R.W. Toward interactive design of correct programs. Stanford Universit, Computer Science Department, 1971. 12 p (Report nº CS - 235).

. GANAPATHY, S.& RAJARAMAN, V. Information theory applied to the conversion of decision tables to computer programs. CACM, 16 (9): 532-9, Sept. 1973.

. GORDON, R. M. Implicants of using modular programming Datamation, 20 (11): 29, Nov. 1974.

. GRAD,B. Tabular form in decision logic. Datamation,: 22-6, July 1961.

. HARTMAN, W. et alu. Management information systerus handbook. New York, Mc Grow. Hell, 1972.1 V.

. How to use decision tables. EDP Analyzer, 4 (5) May 1966.

. HUMBY, E. Programs from decision tables. London, Mc Donald, 1973. 91 p. (Computer Monographs,19).

. INGLIS, J. & KING, P. J. H. Flowchants and decision tables. The Computer Journal, 11 p. 117-8. 1968.

. KAVANAGH, R.F. TABSOL, the language for decision / making. Computers and Automation, 10 (9): 18-22, Sept. 1961.

. KING, P. J. H. Ambiguity in limited entry decision / tables. CACM, 11 (10): 680-4. Oct. 1968.

. _____ Conversion of decision tables to computer programs by the rule mask technique. CACM, 9 (11): 796-801. Nov. 1966.

. KING, P.J.H. Decision tables. The Computer Journal, 10 (2): 135-42, Aug. 1967.

. _____ The interpretation of limited entry decision table format and relatonship among conditions. The Computer Journal, 12 (4): 320-6, Nov. 1969.

. _____ Some comments on systematics. The Computer Journal, 10 (1): 116-9, May 1967.

. KING, P.J.H. & JOHNSON, R.G. Comments on the / algorithms of werhelst for the coversion of limited - entry decision tables to flowchart. CACM, 17 (1): 43--4, Jan. 1974.

. _____ The conversion of decision tables to sequiental listing procedures. The Computer Journal, 18 (4): 298-306, Nov. 1975.

. _____ Some comments on the use of ambigous decision/ tables and their conversion to computer programs. / CACM, 16 (5): 287-90, May 1973.

. KIRK, G.W. Use of decision tables in computer programming. CACM, 8 (1): 41-3, Jan. 1965.

. LOMBARDI, L. A general businen oriented languages based on decision expressions. CACM, 7 (2): 104-11, Febr. 1964.

. LONDON, K. Decision tables: a practical approach for data processing. Princeton, Anerboch, 1972.

. LOW, D.W. Programming by questionnaire: on effective way to use decision tables. CACM, 16 (5): 282-6, May 1973.

. MC DANIEL, H. Applications of decision tables. / Princeton, Brandon/Systems, 1970.

. _____ Decision table software. Princeton, Brandon/ Systems, 1970.

. _____ An introduction to decision logic table. New York, John Willy, 1968.

. MYERS, H.J. Compiling optimized code from decision / tables. IBM Journal of Research and Development, 16 (5): 489-503, Sept. 1972.

. MONTALBANO, M. Decision tables. Chicago, SRA, 1974. 191 p.

. _____ Tables, flowcharts, and program logic. IBM Systems Journal, 1 (3): 51-63, Sept. 1962.

. MUTHUKRISHNAN, C.R. & RAJARAMAN, V. On the conversion of decision tables to computer program. CACM, 13 (6): 347-51, June 1970.

. NATIONAL COMPUTING CENTRE. Systems documentation manual. 3. ed. Mancheter, 1973. 1 V.

. NICKERSON, R.C. An engineering application of logic structure tables. CACM, 4 (11): 516-20, Nov. 1961.

. ORILIA, L.S. et alu. Business data procening systems. New York, John Wiley, 1972. 310 p.

. PARNAS, D.L. On the interna to be used in decompositing systems into modules. CACM, 15 (12): Dec. 1972.

. POLLACK, S.L. CODASYL, CORD and DETAB-X. Datamation, 9 (2): 61, Febr. 1963.

. _____ Comments on the conversion of decision tables to computer programs. CACM, 14 (1): 52, Jan. 1971.

. _____ Conversion of limited entry decision tables to computer programs. CACM, 8 (11): 677-82, Nov. 1965.

. POLLACK, S.L. et alu. Decision tables: theory and practice. New York, Wiley- Interscience, 1971. 179 p. (Wiley Communigraph Series on Business Data Processing).

. POOCH, U.W. Translation of decision tables. Computing Surueys, 6 (2): 125-51, June 1974.

. PRESS, L.J. Conversion of decision tables to computer programs. CACM, 8 (6): 385-90, June 1965.

. REINWALD, L.T. & SOLAND, R. M. Conversion of limited entry decision tables to optmal computer programs - I: minimum overage processing time. JACM, 13 (3): 339-58, July 1967.

. _____. _____ II: minimum starage requirements. JACM, 14 (4): 742-55, Oct. 1967.

. RUBIN, M.L. Introduction to the system life cicle. Princeton, Brandon/Systems, 1970. V. 1 (Handbook of data Processing Management).

. SCHUMATCHER, H. & SEVCIK, K.C. The synthetic approach to decision table conversion. CACM, 19 (6): 343-51, June 1976.

. SHWAYDER, K. Combining decisin rules in a decision table CACM, 18 (8): 476-80, Aug. 1975.

. _____ Conversion of limited- entry decision tables to computer programs, a proposed modification to / Pollack's algorithm. CACM, 14 (2): 69-73, Febr. 1971.

. _____ Extending the information theory approach to converting limites - entry decision tables to / computer programs. CACM, 17 (9): 532-7, Sept. 1974.

. SPRAGUEIV, G. & POLLACK, S.L. ou starage of decision tables. CACM, 9 (5): 319, May 1966.

. STRUNZ, H. The development of decision tables via parsing of complex decision situations CACM, 16 (6): 366-9, June 1973.

. VEINOTT, C.G. Programing decision tables in Fortron, Cobol a Algol. CACM, 9 (1): 31-5, Jan. 1966.

. VERHELST, M. The conversion of limited-entry decision / tables to optmal and near-optimal flowcharts: two new algorithms. CACM, 15 (11): 974-80, Nov. 1972.

. WILLOUGHBY, T.C. & ARNOLD, A.D. Communicating with decision tables, flowcharts, and prose. Data Base.