ANTENOR TURAZI

SISTEMA DE EVENTOS E TRIGGERS APLICADO AO MODELO THM

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

ULRICH SCHIEL

Orientador

T929s Turazi, Antenor.
Sistemas de eventos e triggers aplicado ao modelo THM / Antenor Turazi. - Campina Grande, 1987.
73 f.

Dissertação (Mestrado em Sistemas e Computação) - Universidade Federal da Paraíba, Centro de Ciências e Tecnologia, 1987.
"Orientação : Prof. Dr. Ulrich Schiel".
Referências.

1. Banco de Dados. 2. Sistema de Eventos e Triggers. 3. Modelo THM. 4. Dissertação - Sistemas e Computação. I. Schiel, Ulrich. II. Universidade Federal da Paraíba - Campina Grande (PB). III. Título

CDU 004.65(043)

SISTEMA DE EVENTOS E TRIGGERS

APLICADO AO MODELO THM

ANTENOR TURAZI

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DO CURSO DE PÓS-GRADUAÇÃO EM SISTEMAS E COMPUTAÇÃO DA UNIVERSIDADE FEDERAL DA PARAÍBA COMO PARTE DOS REQUÍSITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS (M.Sc.).

Aprovado por:

ULRICH SCHIEL - Dr.

- Presidente -

MARCUS COSTA SAMPAIO

- Examinador -

ROBERT KALLEY C.DE MENEZES

- Examinador -

CAMPINA GRANDE - Pb

MARÇO - 1987

SISTEMA DE EVENTOS E TRIGGERS APLICADO AO MODELO THM

ANTENOR TURAZI

DISSERTAÇÃO APROVADA EM 25/03/87

ULRICH SCHIEL

Orientador

MARCOS COSTA SAMPAIO

Componente da Banca

ROBERT KALLEY C.DE MENEZES

Componente da Banca

SUMÁRIO

SUMÁRIO

AGRADECIMENTOS

Ao Professor Dr. ULRICH SCHIEL pela orientação.

Aos Professores do Departamento de Sistemas e Computação pelos ensinamentos.

A EMBRAPA pela ajuda financeira e apoio.

A LUIZA, ALEXIS E DEIGMA

O CONHECIMENTO EMPÍRICO, CIENTÍFICO, TECNOLÓGICO, FILOSÓFICO, FORTALECEM UM MESMO PROPÓSITO, CUJO OBJETIVO, É CHEGAR A VERDADE SOBRE O SER HUMANO.

RESUMO

Este trabalho, originou-se na proposta de disssertação de mestrado apresentada na UFPB como requisito parcial para a conclusão do Curso de Mestrado em Sistemas e Computação.

Propomos aqui um sistema de Eventos e Triggers (disparadores) como parte do modelo semântico de Bancos de Dados THM [SC82] de acordo com o projeto PROSEM desenvolvido na Universidade de Stuttgart na Alemanha.

O objetivo básico do trabalho, é fornecer uma ferramenta para automatização de Sistemas em tempo real.

ABSTRACT

This work originated from the proposal of master Dissertation apresented at Paraiba Federal University (UFPB) as part requirement for conclusion of master degree in computer science.

We propose here a system of Events and Triggers as part of the semantic model of Databases, THM, [SC 82] according to the Prosem project the University of Stuttgart in Germany.

The basic object of the work is to provide a tool for an automated systems in real time.

CAPÍTULO I

INTRODUÇÃO

Em geral, organizações funcionam com informações que podem ser usadas por vários setores e estes setores por sua vez sempre mantem os seus arquivos. Para evitar redundância de informação entre arquivos, ou seja, vários arquivos repetitivos é que surgiram Bancos de Dados, objetivando também rapidez na atualização, evitando assim a não atualização por algum setor, proporcionando agilidade e precisão para as tomadas de decisões com as informações integradas.

A especificação do conjunto de informações relevantes para a solução de um certo problema, assim como a especificação do significado dessas informações e a definição das operações que devem ser executadas para que sejam obtidos os resultados esperados, constituem um dos principais problemas que muitos pesquisadores da área de sistema de informação e de outras áreas, como Inteligência Artificial e Linguística Computacional, vem tentando resolver.

A solução para esse problema não é trivial porque ela deve satisfazer as pessoas que possuem diferentes níveis de conhecimento sobre o assunto e com diferentes interesses em relação ao sistema, ou seja, os projetistas e usuários do sistema.

Sendo que nesses últimos estão incluídos tanto os analistas e programadores de aplicação, quanto os usuários casuais que podem, ou não, ser programadores.

Vários modelos ja foram propostos, porém nenhum deles conseguiu satisfazer todas as exigências dos seus diferentes usuários. Dentre as exigências mais comuns citamos:

a) O modelo deve ter capacidade de captar a semântica das informações de modo que seja possível a interpretação dos valores que são armazenados no Banco de Dados;

b) As restrições de integridade devem ser representadas de maneira explícita;

c) Serve como modelo de Banco de Dados conceitual no Design e no processo de evolução, mais diretamente na captura de dados;

d) Pode ser usado como modelo de Banco de Dados para um novo tipo de sistema gerenciador de Banco de Dados, com uma grande capacidade funcional e características de interfaces usuárias.

As experiências em inteligência artificial mostraram que o uso de uma grande quantidade de conhecimento especializado para a solução de uma tarefa de inteligência é o enfoque mais promissor atualmente. A tendência para a solução de problemas nessa área é prover o sistema com grandes quantidades de conhe

cimentos sobre o problema, inclusive conhecimentos redundantes, de modo a limitar a pesquisa e a dedução.

Segundo esse enfoque, acreditamos que o modelo semântico oferece suporte bastante eficaz para o esquema conceitual de um Banco de Dados. O modelo semântico para especificação de um sistema de Banco de Dados nos níveis propostos pelo ANSI/SPARC correspondem ao esquema conceitual.

O esquema conceitual é a descrição do mundo real que será mapeado para o Banco de Dados. É um repertório de declarações verdadeiras sobre as entidades de interesse. Ele contém, não só a definição das entidades e dos relacionamentos entre elas, mas também as restrições de integridade que devem ser satisfeitas pelas entidades e pelos relacionamentos.

Os vários usuários do Banco de Dados operam em subconjuntos do modelo total, os quais são relevantes para suas necessidades particulares. Esses subconjuntos do modelo, como vistos pelos usuários, são chamados de esquemas externos. Um esquema externo contém a descrição do Banco de Dados da maneira como é considerada numa certa aplicação. É através do esquema externo apropriado que o analista e os usuários de aplicação obtem a descrição da parte do Banco de Dados que eles necessitam.

A descrição do Banco de Dados a nível de armazenamento físico é chamada de esquema interno. O esquema interno contém a descrição de como os dados são armazenados, ou seja as técnicas de implementação de registros e de campos, a sintaxe dos valores dos dados, etc...

Uma metodologia para descrição de sistemas de Banco de Dados denominada Prosem foi desenvolvida na Universidade de Stuttgart e nela se insere o modelo THM para projetar o esquema conceitual.

O THM (Temporal-Hierarchic Data Model) foi proposto por Ulrich Schiel em sua tese de doutorado na Universidade de Stuttgart. Para estabelecer a terminologia e o escopo da semântica de um Banco de Dados o THM dividiu o universo em três conceitos globais distintos:

a) **Conceito global do físico ou concreto**, objetos ou fatos como "uma pessoa", "meu cavalo",...

b) **Conceito global do metafísico ou abstrato**, que são coisas que só existem na mente humana, como "número 6", "a idade",...

Essas duas globais formam o mundo real.

c) **Modelo global**, são todos os objetos armazenados num sistema de computação.

Nesse trabalho o THM não será evidenciado em suas fases, já que existe trabalho sendo executado nesse sentido como dissertação de mestrado na Universidade Federal da Paraíba, razão pela qual concentramos esforços na sentido de implementar e incrementar o modelo de Eventos e Triggers, que faz parte do THM (SC84).

No Capítulo II abordamos o modelo de Eventos e Triggers, caracterizando os tipos de objetos, com ênfase para o objeto espacial evento e fornecemos uma função de tratamento do tempo, denominada Datahora. A seguir, no Capítulo III mostramos os problemas ainda em vias de solução por pesquisadores e o propósito deste trabalho de fornecer uma ferramenta para automatizar Eventos e acionar Triggers.

No Capítulo IV, apresentamos a sintaxe de Eventos e Triggers proposta pelo modelo THM e fazemos considerações dessa sintaxe. A metodologia de implementação é apresentada no Capítulo V onde mostramos as estruturas de Eventos e Triggers, as funções e os operadores por nós definidos nesse trabalho. No Capítulo VI estão as estruturas de dados para Eventos e para Triggers, assim como a estrutura de ações para Triggers e a validação de data e Hora. A seguir, no Capítulo seguinte apresentamos o sistema com suas funções e módulos com os respectivos algoritmos utilizados para a implementação. Finalmente, no Capítulo VIII tecemos considerações sobre o trabalho e fazemos as conclusões.

UFCG/BIBLIOTECA/BC

CAPÍTULO II

O MODELO DE EVENTOS E TRIGGERS

Para o universo em discussão nós distinguimos objetos no espaço e objetos de tempo. Objetos espaciais, como entidades, relacionamentos e Eventos são modelados por modelos de dados semânticos e objetos temporais, como instantes, intervalos ou períodos de tempo são descritos por modelos de tempo. Asserções de tempo podem ser precisas ou vagas, únicas ou periódicas. Na estrutura de um sistema de computação, espaço não é um espaço contínuo multidimensional mas um conjunto discreto de pontos (entidades), fatos (relacionamentos), e acontecimentos (Eventos) [SC 85]. Objetos de tempo podem ser instantes (pontos), ou intervalos (tempo). Podem ser expressados com valores absolutos, datas calendário, ou relativos para outros objetos de tempo ou Eventos.

II.1 - OBJETOS ESPACIAIS

Coisas básicas, como pessoas, idade, cargo,... são chamadas de entidades. Outro tipo de objeto espacial é o relacionamento, que é uma conexão entre duas entidades.

Outro tipo de objeto espacial é o Evento. É uma ocorrência no sistema de informação que envolve mudanças no Banco de Dados. Operações e transações num Banco de Dados são Eventos.

Os objetos espaciais tratados em maior detalhe nesse trabalho, são os Eventos.

Um Evento é composto por um predicado especificando em qual condição o Evento se torna verdadeiro chamado "On-Condition". Nesse caso, uma ou mais ações são disparadas por um Trigger que executam as transformações no sistema, correspondentes ao evento.

"ON-CONDITION" podem ser dos seguintes tipos:

1) ON DEMANDA

Meio pelo qual o Evento não tem nenhuma condição explícita de acontecer e pode ser acionado a qualquer tempo por um comando do usuário.

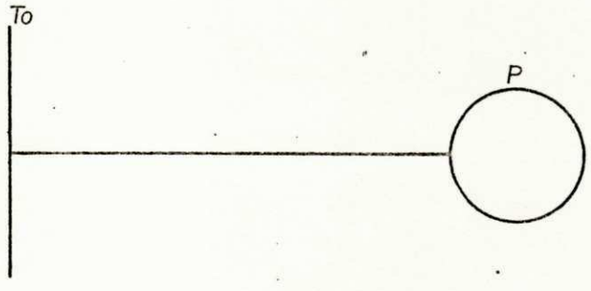

FIG. 1 – Evento on demanda

O Trigger to pode ser acionado a qualquer tempo, sem a necessidade de condição prévia. "P" é a situação do sistema, após a execução de to.

2) ON CONDIÇÃO-DE-TEMPO

Consiste de predicados sobre objetos temporais e referências a relógio. A ação será disparada automaticamente por um monitor de controle.

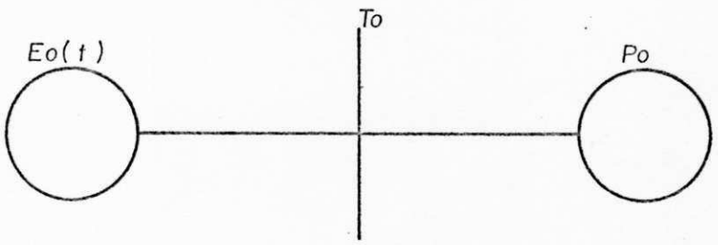

FIG. 2 – Evento on condição de tempo

Pode ser único ou periódico. O Trigger To estará sensibilizado com o Evento Eo marcado, devido a condição do relógio ou tempo P (T). Sendo único, executa o Trigger e elimina o evento e caso seja periódico, executa o Trigger no período correspondente.

3) ON CONDIÇÃO-DE-EVENTO

Um Evento que ocorre como consequência de um ou mais Eventos.

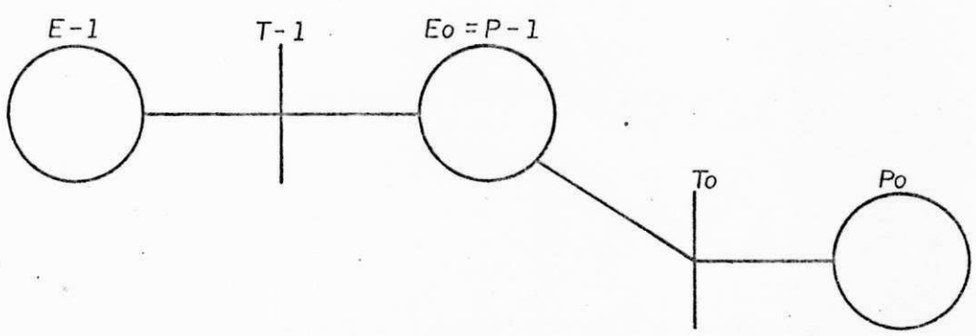

FIG. 3 - Evento on condição de evento

O Evento EO estando marcado pode ocasionar a execução do Trigger To que ocasionará a marcação do Evento PO.

4) ON CONDIÇÃO-DE-DATABASE

Se o Banco de Dados retorna uma condição prevista, o Evento retorna verdadeiro e um Trigger poderá ser disparado.

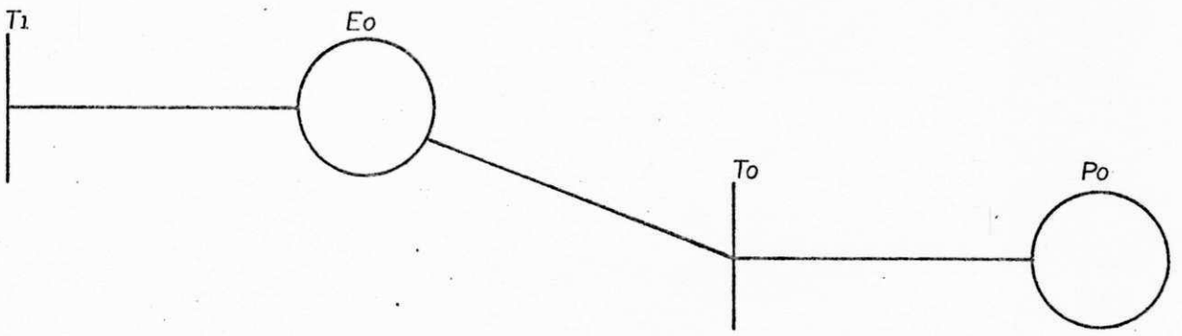

FIG. 4 – Evento on condição de database

Alguma mudança no Banco de Dados, Tl poderá causar a marcação de um Evento, que poderá ser a pré-condição de um Trigger To.

II.2 - OBJETOS TEMPORAIS

Existem duas formas de interpretação para tempo: absoluta e relativa. Na visão relativista, Eventos e entidades existem primeiro. Tempo somente pode ser expressado relativo a algum Evento ou para existência de alguma entidade. Não existem objetos de tempo puros, somente podem ser identificados via objetos espaciais.

Na visão absolutista, tempo é uma sequência linear de pontos, independente de objetos espaciais, subsequentemente, objetos espaciais são relatados ao tempo.

Considerando o tempo como uma linha real, o ponto zero pode ser interpretado como o agora, os números negativos, à

esquerda, são o passado e os números positivos, à direita, são o futuro.

O primeiro objeto temporal de interesse para um sistema de informação é o ponto de tempo, significando instante ou momento. Apesar do fato de um ponto ser conceitualmente infinitamente pequeno, como um ponto geométrico, pontos em sistemas de informação têm densidade chamada de granularidade. Dependendo do uso específico de ponto de tempo, a granularidade pode variar de frações de segundos a vários anos. Nosso sistema trabalha com data e hora para problemas temporais e granularidades diferentes não são tratadas nesse trabalho.

Visando uma solução para o tratamento de tempo nós optamos pela seguinte função:

DATAHORA (TO, T1, T2 - T3)
Onde
TO e T1 são pontos de tempo.
T2 - T3 intervalo de pontos de tempo.

é interpretada como:

a) Para pontos de tempo TO e T1:
   se somente se tempo = TO ou tempo = T1

b) Para intervalos de pontos T2 - T3:
   qualquer tempo pertencente ao intervalo T2 - T3.

Como pode ser visto em "A" a interpretação pode ser:

Durante TO an durante T1

E em "B":

Durante TN when TN > = T2 and < = T3

Exemplo:

DATAHORA (10:00:00, 10:01:00, 11:00:00, 11:05:00)

interpretada como:

Durante (10:00:00) and

Durante (10:01:00) and

Durante hora when hora > = 11:00:00 and

Hora < = 11:05:00

Nesse caso, com a granularidade sendo requisitada em segundos, a cada ciclo de relógio interno de máquina, a função se torna verdadeira se hora = 10:00:00, 10:01:00, 11:00:00 > = hora < = 11:05:00,

Supondo esta situação:

DATAHORA (1985 - 1986)

interpretada como:

Durante data when data < = 1986 anda data > = 1985 implicaria na veracidade de função durante todos os dias de 1985.

CAPÍTULO III

O PROBLEMA

A especificação do conjunto de informações para a solução de um problema, assim como a especificação do significado dessas informações e a definição de ações e operações, constituem problemas ainda em vias de solução por pesquisadores, no intuito de satisfazer pessoas com diferentes níveis de conhecimento e diferentes interesses em relação ao sistema, que são os projetistas e os usuários.

Na nossa visão, o esquema conceitual é o que deverá permanecer relativamente estável e por isso, é importante que seja projetado com cuidado, visando possíveis mudanças na empresa ou no ambiente computacional.

Para projetar um esquema conceitual, o administrador do Banco de Dados necessita de uma estrutura de conceitos e ferramentas para construir uma modelagem, objetivando uma aplicação.

Esses conceitos são obtidos pela semântica dos modelos de dados. No THM todos os objetos de algum tipo, os quais são de interesse para aplicação a ser modelada são apresentados como entidades ou membros de uma classe. No modelo global de um sistema

de informação tradicional, as alterações são efetuadas através de fitas magnéticas, console, terminal,... Sistemas em tempo real são diretos, independentes da manipulação do homem, projetados para aplicações especializadas. Para sistemas de informação, uma conexão de maior interesse com o propósito geral é o relógio e calendário e o THM não tem ferramenta implementada para isso. Praticamente as necessidades e operações a serem efetuadas são decorrentes de transações que ocorrem em determinadas horas ou dias de um período. Para evitar que deixem de ser efetuadas manualmente, o que facilmente pode ser esquecido, ou então, determinadas operações deixem de ser efetuadas e buscando com isso mais segurança nas aplicações, evitando que se corram riscos desnecessários é prático a automatização destas atividades.

Buscando a automatização de sistemas e sabendo que em aplicações de Bancos de Dados são efetuadas transações e operações que por sua vez executam Eventos ocorridos no sistema, é nosso objetivo dotar o modelo THM de uma ferramenta para:

Automatizar Eventos,

Acionar Triggers (disparadores).

CAPÍTULO IV

SINTAXE DE EVENTOS E TRIGGERS

EVENT <NOME-DO-EVENTO>
ON <OCORRÊNCIA>
CONDITION <PREDICADO-E>

TRIGGER <NOME-DO-TRIGGER>
PRÉ-CONDIÇÕES <PREDICADO-T>
CORPO <COMANDOS>
PÓS-CONDIÇÕES <EVENT>

onde

<OCORRÊNCIA>::= <TIME> | <DATE> | <IF> | <WHEN>
<ANTES> | <DURANTE> | <DEPOIS>
<PREDICADO-E>::= <EVENT> | <DATE> | <TIME> | <DEMAND>
<DATABASE> | <PERIODIC>
<PREDICADO-T>::= <EVENT>
<COMANDOS>::= <AÇÕES> | <NOME>
<AÇÕES>::= <INSERT> | <DELETE> | <FIND>
<REQUEST> | <UPDATE> | <CREATE>
<POSCOND>::= <EVENT>

Esta sintaxe e o seu tratamento estão mais detalhados na metodologia de implementação. ?pg Foram também observadas sintaxes propostas por [AT 82] e [BRO 84] donde colhemos alguns subsídios

para a implementação desta sintaxe. No INCOD-DTE, (interactive conceptual design of data, transactions and events) cuja estrutura geral é descrita por [AT 82] a sintaxe é a seguinte:

```
EVENT  <EVENT-NAME>
CONDITIONS  <CONDS>
ACTIONS     <ACTS>
ACTIVITY    <ACTIVITY-NAME>
```

Na nossa implementação, a ocorrência de um evento pode ser ligada a uma operação (evento exógeno), ou a uma condição de tempo (evento endógeno). A diferença fundamental entre Trigger e Operação é que Trigger deve conter pelo menos um evento de pré-condição, chamado de parte de evento, e se a pré-condição se torna verdadeira, o corpo deve ser executado imediatamente. Desde que a definição de eventos e triggers são independentes, pode ocorrer um evento seja pré-condição de vários triggers e nesse caso uma sequência de execução será estabelecida. O nome do trigger corrente é incluído nas pré-condições de um trigger aguardando, que permanece no papel de um evento adicional, o qual volta verdadeiro no final da execução.

Exemplos:

```
E
TESTE1
DATE 13/03/87
DEPOIS 12:00:00
```

```
T
ADICIONAL
TESTE1
INSERT CCE INTO DIN
END
TESTE1

E
TESTE2
DATE 13/13/87
ANTES 15:30:00

T
ACIONA2
TESTE2
CREATE DSC
END
TESTE2

E
TESTE3
DATE 13/03/87
TESTE1

T
ACIONA3
TESTE3
DELETE CCE FROM DIN
END
TESTE3
```

UFCG/BIBLIOTECA/BC

```
E
TESTE4
DURANTE 14:00:00  16:00:00
PERIODIC 00:01:00

T
ACIONA4
TESTE4
PGM00
END
TESTE4
```

CAPÍTULO V

METODOLOGIA DA IMPLEMENTAÇÃO

Dados os conceitos vistos até agora, um sistema de eventos e triggers dentro da Metodologia de desenvolvimento do modelo conceitual, automatiza os eventos previstos e pode dar soluções, fornecendo subsídios para formação de novos eventos, evitando que procedimentos corram o risco de não serem efetuados.

Optamos e acreditamos ser a estrutura de uma lista duplamente encadeada e apontadores, recomendável na montagem do sistema, objetivando velocidade no acesso devido a redução do tempo de pesquisa a eventos por nomes e de triggers por ponteiros, já que onde quer que se esteja posicionado nas listas num determinado momento, poderemos, através da varredura de uma delas, obter ponteiros correspondentes na outra de modo quase imediato.

V.1 - A composição de eventos está estruturada em um byte o qual indicará se o evento foi posicionado ou não. O encontro desse evento específico é feito através da varredura da lista de eventos, sendo que, em caso de inclusão o evento será posicionado ao final da lista, ou em algum lugar que esteja apontado como livre.

V.2 - Os testes de conditions para o evento foram organiza

dos em uma tabela de comandos interpretáveis e de eventos pré-definidos, como por exemplo, TIME e DATE, ou de eventos definidos a qualquer tempo através da palavra chave "EVENT". Eventos estes que não correm o risco de serem perdidos devido ao esquema de segurança do monitor, o qual recarrega a tabela de disco, ao início de cada ciclo pré-definido do relógio interno, para a memória, e a salva em disco ao final de cada ciclo.

Com relação ao ciclo interno de relógio ele é facilmente alterável através da modificação do software, caso não atenda com rapidez aos fins desejados.

V.3 - No tratamento de "ON" e "CONDITION" para o evento optamos pela criação de uma tabela especial de interpretação baseada nas seguintes palavras-chave:

V.3.1 - **IF** - Realiza o teste lógico que segue esta palavra sem envolver testes com valores de tempo.
Baseia-se na seguinte estrutura:

. IF ENTIDADE OPERADOR OPERANDO

Onde entidade indica que este parâmetro pode ser opcional, dependendo do teste a ser efetuado.

Operador para o sistema pode ser " > " (maior que), " < " (memor que), "=" (igual), "**NOT**" (negativo), "**IN**" (afirmação para condições simples. Para condições compostas), "**AND**" (operação e lógico significando que os resultados das operações simples que

antecedem e sucedem o operador devem ser verdadeiras para tornar o teste verdadeiro) , "OR" (operação ou lógico, significando que ao menos um dos resultados, subsequente ou antecedente ao operador deve ser verdadeiro para que toda condição seja avaliada como verdadeira), "OREX" (operação ou exclusivo, significando que ou a condição antecedente ao operador, oú a condição subsequente ao operador deverá ser verdadeira, porém não ambas, para que a condição seja satisfeita.

Operando pode ser um valor, um evento já definido pelo usuário, ou uma expressão lógica a ser avaliada a tempo de teste do "ON" ou "CONDITION".

V.3.2 - FUNÇÕES - Objetivando o tratamento de tempo foram definidas as seguintes funções:

V.3.2.1 - DATE - Fornecendo a data no formato DD/MM/AA

V.3.2.2 - HOUR - Fornecendo a hora no formato HH:MM:SS

Estas duas funções permitem o tratamento do tempo tanto de maneira absoluta, por exemplo, um evento que tivesse de ser posicionado em uma data e hora específicas, como de maneira relativa, por exemplo, um evento posicionado sempre depois de alguma data ou hora.

As funções "CICLO" e "RESTART" são tratadas

automaticamente pelo sistema:

Ciclo - Contendo o byte associado, posicionado se início de ciclo de relógio.

Restart - Contendo indicação de reinício da execução do sistema.

Exemplo:

Seja a condição:

IF RESTART=TRUE AND RELATÓRIO= TRUE AND NOT IMPRESSORA

Envolve os seguintes testes e posicionamentos:

Caso RESTART=TRUE então EV.INTERNO1 = 1

Senão EV.INTERNO1 = 0.

O IF se transforma em:

IF EV.INTERNO1 AND RELATÓRIO = TRUE AND NOT IMPRESSORA

Como foi encontrado um and a expressão após o mesmo é avaliada e depois checada com o evento cujo nome está antes do AND. Caso RELATÓRIO = TRUE ENTÃO: EV.INTERNO2 = 1

SENÃO EV.INTERNO2 = 0

Supondo RELATÓRIO = TRUE fica assim o IF:

IF EV.INTERNO1 AND EV.INTERNO2 AND NOT IMPRESSORA. Como EV.INTERNO1 e EV.INTERNO2 são verdadeiros o resultado do AND entre dois será verdadeiro, posicionando então EV.INTERNO3 como verdadeiro, I. é = 1.

As operações AND, OR, OREX são feitas através da álgebra boolena, podendo fornecer os seguintes resultados:

	EV1	EV2	RESULT
	0	0	0
AND	1	0	0
	0	1	0
	1	1	1
	0	0	0
OR	1	0	1
	0	1	1
	1	1	1
	0	0	0
OREX	1	0	1
	0	1	1
	1	1	0

Continuando, agora temos:

IF EV.INTERNO3 AND NOT IMPRESSORA

Será avaliado o evento impressora. Caso IMPRESSORA = FALSE então EV.INTERNO4 = 1 e a expressão se resume a IF EV.INTERNO3 AND EV.INTERNO4 que são verdadeiros, permitindo o posicionamento do evento para o qual a CONDITION foi satisfeita.

V.3.3 **WHEN** - Efetua teste lógico envolvendo tempo e data.

Tem a seguinte estrutura:

WHEN DATE/TIME operador operando

Neste caso, o evento será comparado via operador com a hora ou data do sistema, fornecendo um valor falso ou verdadeiro, que sendo satisfeito ocasionará o posicionamento do evento.

Como no IF, o WHEN também pode ser composto, através de AND, OR, OREX e o processo de avaliação é o mesmo. Uma particularidade no WHEN é que um campo contendo ":" será associado a hora e um campo contendo "/" será associado a data e um campo inteiro para ano.

WHEN DATE = 31/06/86

Pode ser reescrito como WHEN 31/06/86. Do mesmo modo

WHEN HOUR = 10:00:00 pode ser WHEN 10:00:00.

A avaliação é da esquerda para a direita, como no IF de modo que:

WHEN 1985 AND DATAHORA(10:00:00 - 12:00:00)

implica nos passos:

SE ANO = 1985 posiciona EV.INTERNO1

SE HORA ENTRE 10:00:00 e 12:00:00 posiciona EV.INTERNO2

Se o resultado do AND ENTRE EV.INTERNO1 E EV.INTERNO2 for verdadeiro, o evento será posicionado, podendo satisfazer a condição de um trigger aguardando. Isto significa, que durante todo o ano de 1985, sempre que a hora estiver no intervalo de 10:00:00 a 12:00:00 Hs o evento será posicionado, permitindo a ativação de algum trigger. Isto significa que antes de 10:00:00 e após 12:00:00 o status do evento volta a ser zero isto é, o evento volta a ser desposicionado.

V.4 - **OPERADORES** - Os operadores reconhecidos pelo interpretador são os seguintes:

V.4.1 - **AND** - Efetua a operação "E" lógica, entre os operadores. Isto é, o resultado da operação será considerado verdadeiro, se um operador a esquerda do "AND" for verdadeiro, assim como o operador da direita também seja verdadeiro.

V.4.2 - **OR** - Efetua a operação "OU" lógica entre os operadores, ou seja, o resultado da operação será considerado verdadeiro se o operador a esquerda do "OR" for verdadeiro, ou se o operador a sua direita for verdadeiro, ou ambos forem verdadeiros.

V.4.3 - **OREX** - Efetua a operação lógica "OU EXCLUSIVA". Define-se como a operação anterior, exceto pelo fato de que ambos os operadores não podem ser verdadeiros. A tabela verdade para estas operações (AND, OR, OREX) foram vistas no IF.

V.4.4 - **<** - Efetua a comparação lógica entre dois operandos retornando a condição verdadeira se o primeiro for menor que o segundo.

V.4.5 - **>** - Efetua a comparação lógica entre dois operandos retornando a condição verdadeira se o segundo for menor que o primeiro.

V.4.6 - **=** - Efetua a comparação lógica entre dois operandos retornando a condição verdadeira se ambos contiverem o mesmo valor.

V.4.7 - **NOT** - Torna o valor da expressão avaliada, expressão a sua direita, o oposto do avaliado, isto é

se a avaliação for verdadeira ele a tornará falsa e caso seja falsa, ele a tornará verdadeira.

"NOT" primeiro verifica a veracidade da expressão, em seguida posiciona ou desposiciona o evento temporário.

Supondo NOT IMPRESSORA

O procedimento será:

a) Verificar o status do evento IMPRESSORA;

b) se status = 0 modifica para 1
   se status = 1 modifica para 0

Deste modo, supondo o evento impressora desativada, isto é, status = 0 "NOT" IMPRESSORA será verdadeiro porque status mudará para 1 que corresponde a condição verdade.

V.4.8 - **TRUE** - Verifica se uma expressão proposta num IF é verdadeira.

"TRUE" apesar de redundante numa expressão, pode torna-la de mais fácil compreensão numa leitura. Na verdade, TRUE, irá verificar se o evento está posicionado ou não, retornando valor 1 para um evento temporário, caso o evento testado esteja posicionado.

V.4.9 - **FALSE** - verifica se uma expressão proposta num IF é falsa. Tem a mesma finalidade de TRUE.

Por exemplo:

```
NOT IMPRESSORA
NOT IMPRESSORA = TRUE
IMPRESSORA = FALSE
```

São equivalentes.

V.4.10 - EL - Indica que o evento a sua esquerda será posicionado caso o da direita também esteja. "EL" praticamente significa correspondência. No caso, não pode ser substituido pelo operador "=" porque este implica em teste, e "EL" implica em assinalação, sem teste algum.

Assim:

```
DEFEITO.EL.NOT.IMPRESSORA
```

Significa:

Se não há impressora posicione defeito. Note, que se NOT impressora for falsa, defeito não será posicionado de acordo com a definição.

CAPÍTULO VI

ESTRUTURA DOS DADOS

VI.1 - Estrutura de Eventos

Os eventos foram estruturados numa lista duplamente encadeada, visando economia de memória, assim como maior velocidade de acesso e segurança. A estrutura da lista de eventos é a seguinte:

NOME DO EVENTO	ON	CONDITION	PONTEIROS PARA TRIGGERS	STATUS	PONTEIRO ELEMENTO ANTERIOR	PONTEIRO PROXIMO ELEMENTO

FIG. 5 – Estrutura de Eventos

Onde:

NOME DO EVENTO	20 bytes	- Contém nome do evento.
ON	60 bytes	- Contém o ON com valores de tempo, hora e data. Sendo verdadeiros levam ao teste da CONDITION.
CONDITION	60 bytes	- Contém a CONDITION do evento para que seja posicionado. Pode-se utilizar nome de evento, relógio, calendário.
PONTEIROS P/TRIGGER	48 bytes	- Contém até doze ponteiros para trigger com quatro bytes cada. Julgamos que doze triggers apontando para um evento uma quantidade de razoável. Contudo, este número poderá ser alterado facilmente com a recompilação do programa.
STATUS DO EVENTO	1 byte	- Indica se o evento foi ou não posicionado.
PONTEIRO ELEM. ANTER.	4 bytes	- Apontador para elemento anterior na lista ou zeros no caso de primeiro elemento.

PONTEIRO PROX. ELEM. 4 bytes - apontador para próximo elemento livre na lista, ou zeros no caso de último elemento.

V.I.2 - Estrutura de Ações para Triggers

Os triggers podem ser programas residentes em disco sendo ativados quando os eventos a eles relacionados são todos verdadeiros. Também podem ser completamente definidos com o conjunto de funções seguintes, definidas como ações.

As funções a seguir são tratadas como ações dentro do corpo de um trigger. Para assegurar seu funcionamento, é necessária a existência de uma área na memória que servirá de interface entre o sistema de eventos e trigger e o database. Esta área terá o formato genérico:

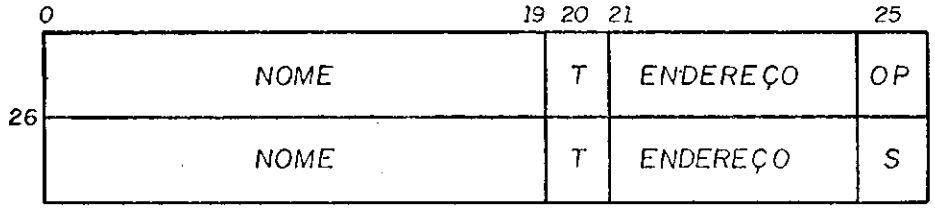

FIG. 6 - Interface entre o sistema de eventos e o database

Onde:

NOME - pode ser o nome de uma entidade ou de uma classe. Como este campo ocorre duas vezes temos a possibilidade de enviar ao DB o nome de uma entidade e de uma classe.

T - significa o tipo de elemento que qualifica o nome como entidade (E) ou classe (C).

ENDEREÇO - é preenchido pelo DB com o endereço da lista na memória, ou zeros.

OP - é o tipo da ação desejada podendo assumir os valores:

I - Insert;

D - Delete;

U - Update;

C - Create;

F - Find;

R - Request.

S - é o devolvido pelo DB indicando a firmação do término de uma operação (0) ou não ($\neq$0).

Cada operação de detabase é qualquer alteração ou operação de recuperação sobre instâncias simples de classes de objetos [BRO 84]. Na nossa implementação a síntaxe das operações é:

INSERT - inserção de entidade em uma classe.

INSERT e **INTO** X

onde E = ENTIDADE, X = CLASSE

DELETE - remoção de uma entidade ou de um objeto de uma entidade. Tem os formatos

DELETE e **FROM** X

onde E = ENTIDADE e X = CLASSE

DELETE E

onde E = ENTIDADE

UPDATE - realiza alteração de um valor numa classe.

UPDATE X **INC** C **BY** Y

onde X = VALOR, C = CLASSE, Y = NOVO VALOR

CREATE - cria uma classe.

CREATE C

onde C = CLASSE a ser criada.

FIND - verifica a existência de uma entidade em uma CLASSE, possuindo o formato

FIND E **IN** C

onde E = ENTIDADE C = CLASSE.

REQUEST - devolve interativamente o conteúdo de uma CLASSE, ou uma ENTIDADE (objeto) específico. Tem os formatos:

REQUEST E

Onde E = ENTIDADE

REQUEST E **IN** C

onde E = ENTIDADE e C = CLASSE.

VI.3 - Estrutura de Triggers

Os triggers estão também estruturados numa lista du

plamente encadeada, pela mesma razão da lista de eventos. A lista de triggers tem o seguinte formato:

NOME TRIGGER	PRECOND	AÇÕES	POSCOND	PONTEIRO PRÓXIMO ELEMENTO	PONTEIRO ELEMENTO ANTERIOR

FIG. 7 - Estrutura de Triggers

Onde:

NOME-TRIGGER — 8 bytes - contendo nome do trigger.

PRECOND — 20 Bytes - contendo os ponteiros dos eventos que são pré-condition para o trigger, que viabiliza ou não o disparo da ação.

AÇÕES — 160 bytes - contendo as ações que poderão ou não residir em disco, e serão chamadas conforme definido no corpo de trigger.

PONTEIRO-PRÓXIMO — 4 bytes - contendo o próximo endereço de memória que contém uma descrição de trigger.

PONTEIRO-ANTERIOR — 4 bytes - contendo o endereço anterior na memória, o qual contém uma descrição de trigger.

VI.4 - Validação de Data e Hora

Data e Hora serão tratadas pelo ON do EVENT. A verificação da validade de relógio e calendário fornecido no ON será efetuada através do algoritmo:

```
SE MÊS < 3
      ENTÃO
          N=365*ANO+1+31*(MÊS-1)+INT((ANO-1)/4)-
          INT(3/4*INT((ANO-1)/100)+1)
      SENÃO
          N=365*ANO+1+31*(MÊS-1)-INT(.4*MÊS+2.3)+
          INT (ANO/4)-INT(3/4*(INT(ANO/100)+1))
      FIMSE
      DIA-DA-SEMANA = N-INT(N/7)*7
      SE DIA-DA-SEMANA = 0
          ENTÃO DIA-DA-SEMANA = 7
      FIMSE
      (DIA DA SEMANA ESTÁ ENTRE 1 E 7, SENDO QUE
      1 = DOMINGO, 2 = SEGUNDA,..., 7 = SÁBADO)
      SE MÊS = 2
          ENTÃO Z = ANO/4-INT(ANO/4)
          SE Z = 0
```

```
            ENTÃO
                NÚMERO-DIAS = 29
            SENÃO
                NÚMERO-DIAS = 28
            FIMSE
        SENÃO
        SE MÊS = 4 OU 6 OU 9 OU 11
            ENTÃO
                NÚMERO-DIAS = 30
            SENÃO
                NÚMERO-DIAS = 31
        FIMSE
```

O algoritmo acima serve tanto para calcular dias da semana, quanto para se descobrir a quantidade de dias num mês, o que serve para validar a data, assim como o dia da semana.

A validação da hora é mais simples, não necessitando de algoritmo especial. A simples verificação de minuto maior que 59 e hora maior que 24, invalida a hora entrada no ON do EVENT.

CAPÍTULO VII

O SISTEMA

Neste Capítulo mostraremos todas as funções e algorítmos que foram definidos e utilizados para a implementação do Sistema de Eventos e Triggers.

VII.1 - Função Gerenciar Tempo

Posicionar intervalo de tempo para ciclo interno de verificação das listas de carga/salvamento dos eventos e triggers. Ao ser encontrada a função fim de intervalo ocorrerá o salvamento das listas de eventos e de triggers para disco e logo em seguida seu recarregamento. Caso alguma situação anormal ocorra durante o tratamento com leitura e gravação em disco, a integridade dos dados não será afetada, devido ao método de acesso que se está empregando. Após a carga e salvamento dos eventos e triggers, o intervalo será reposicionado para novo ciclo. Caso o ciclo não tenha terminado, ou tenha sido reposto, o sistema passa o controle para a função gerenciar eventos.

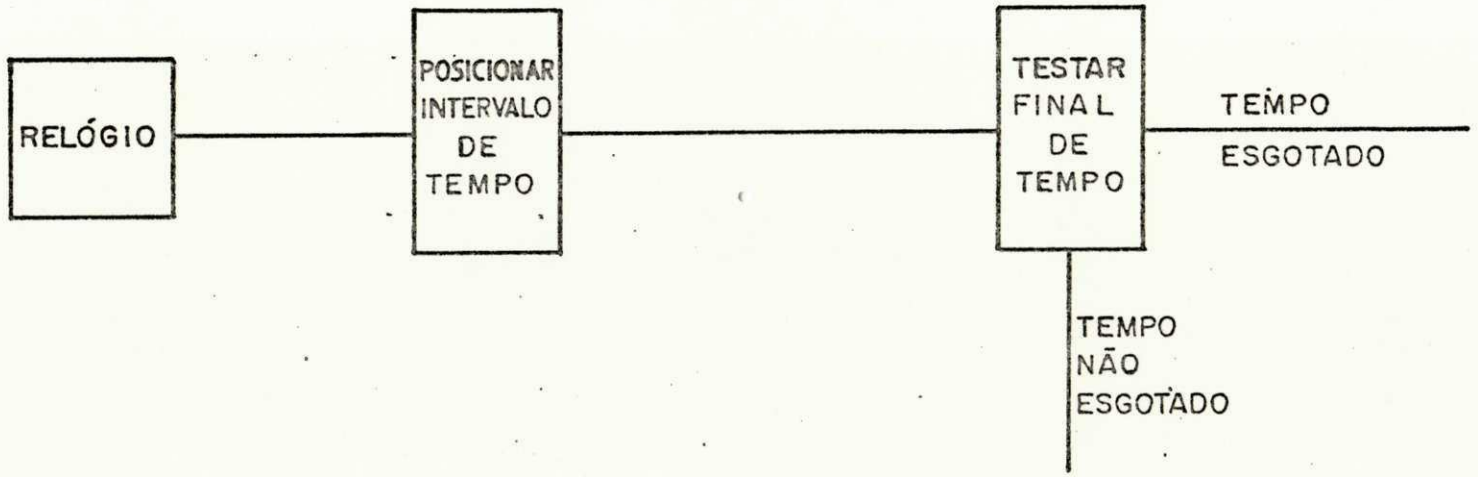

FIG. 8 – Visão da função gerenciar tempo.

```
Algoritmo
        TEMPO = 01:00
        WHILE TEMPO > 0 REPEAT
                CALL SBRTEC;
                CALL SBRANA;
        END;
        IF TEMPO = 0
                REPEAT
                    FIMGRAVA:=  ∂EVENTO ;
                    GRAVA EVENTO ;
                    ∂EVENTO:= ∂PRÓXIMO EVENTO ;
                UNTIL FIMGRAVA:= 0 ;
                REPEAT
                    FIMGRAVA:= ∂TRIGGER ;
                    GRAVA TRIGGER ;
                    ∂TRIGGER:= ∂PRÓXIMO TRIGGER ;
                UNTIL FIMGRAVA:= 0 ;
                REPEAT
                    FIMLER:= ∂EVENTO ;
                    LER EVENTO ;
                    ∂EVENTO:= ∂PRÓXIMO EVENTO ;
                UNTIL FIMLER:= 0 ;
                REPEAT
                    FIMLER:= ∂TRIGGER ;
                    LER TRIGGER ;
                    ∂TRIGGER:= ∂PRÓXIMO TRIGGER ;
                    UNTIL FIMLER:= 0 ;
                TEMPO:= 01:00 ;
                ENDIF
```

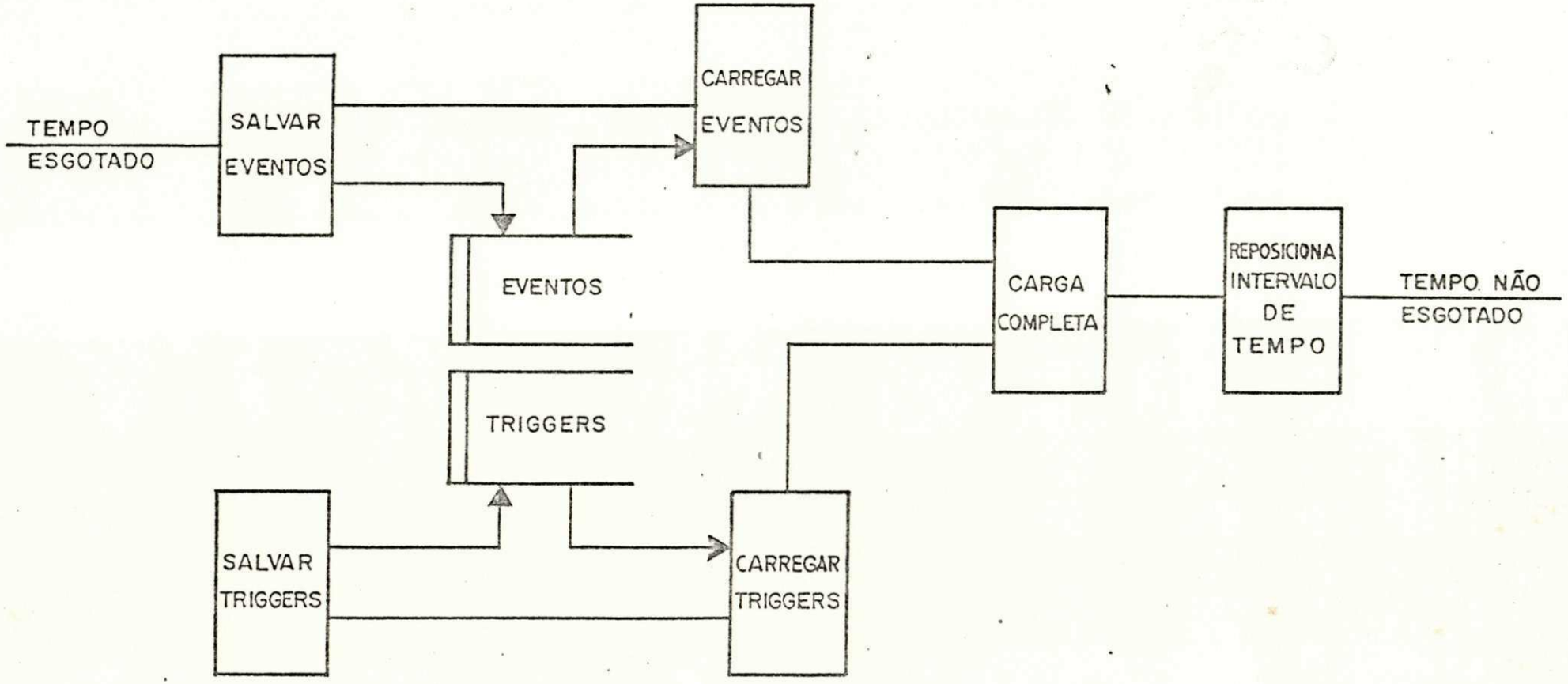

FIG. 9 – Visão da função de carga e de salvamento.

VII.2 - Módulo Posicionar Início Lista de Eventos

Efetua a carga do ponteiro do 1º evento na lista.

VII.3 - Função Gerenciar Eventos Posicionados

Verifica na lista de eventos os que estão posicionados, ou seja, STATUS 1 e ativa os triggers que dependem destes eventos. Ao retorno desta ativação, a lista é novamente varrida, verificando ONS e CONDITIONS as quais, sendo satisfeitas, posicionam o evento e tornam a verificar a lista.
Na primeira passagem verifica eventos posicionados.
Na segunda, verifica de acordo com ONS e CONDITIONS.

Algoritmo

```
∂EVENTO = ∂INÍCIO LISTA EVENTO;
WHILE EVENTO ≠ 0
   IF STATUS EVENTO=1
      CALL SBRPRE;
      LOAD TRIGGER EVENTO;
      START TRIGGER EVENTO;
   ELSE
       EVENTO:=  PRÓXIMO EVENTO;
      ENDIF
   END
   ∂EVENTO:= INÍCIO-LISTA-EVENTO;
    WHILE EVENTO:= 0;
      CALL SBRON;
      CALL SBRCON;
      ∂EVENTO:= ∂PRÓXIMO EVENTO
```

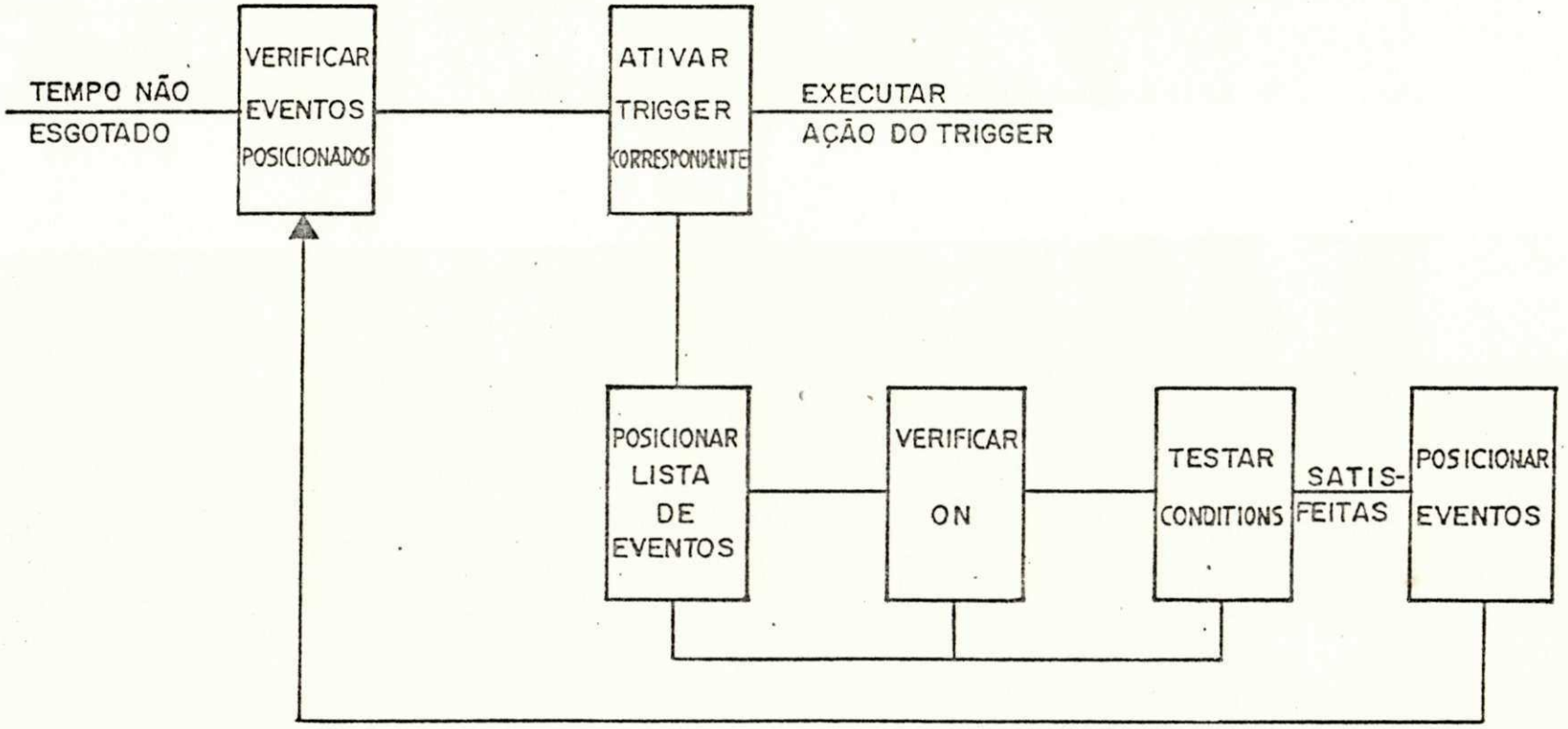

FIG. 10 - Visão da função gerenciar eventos.

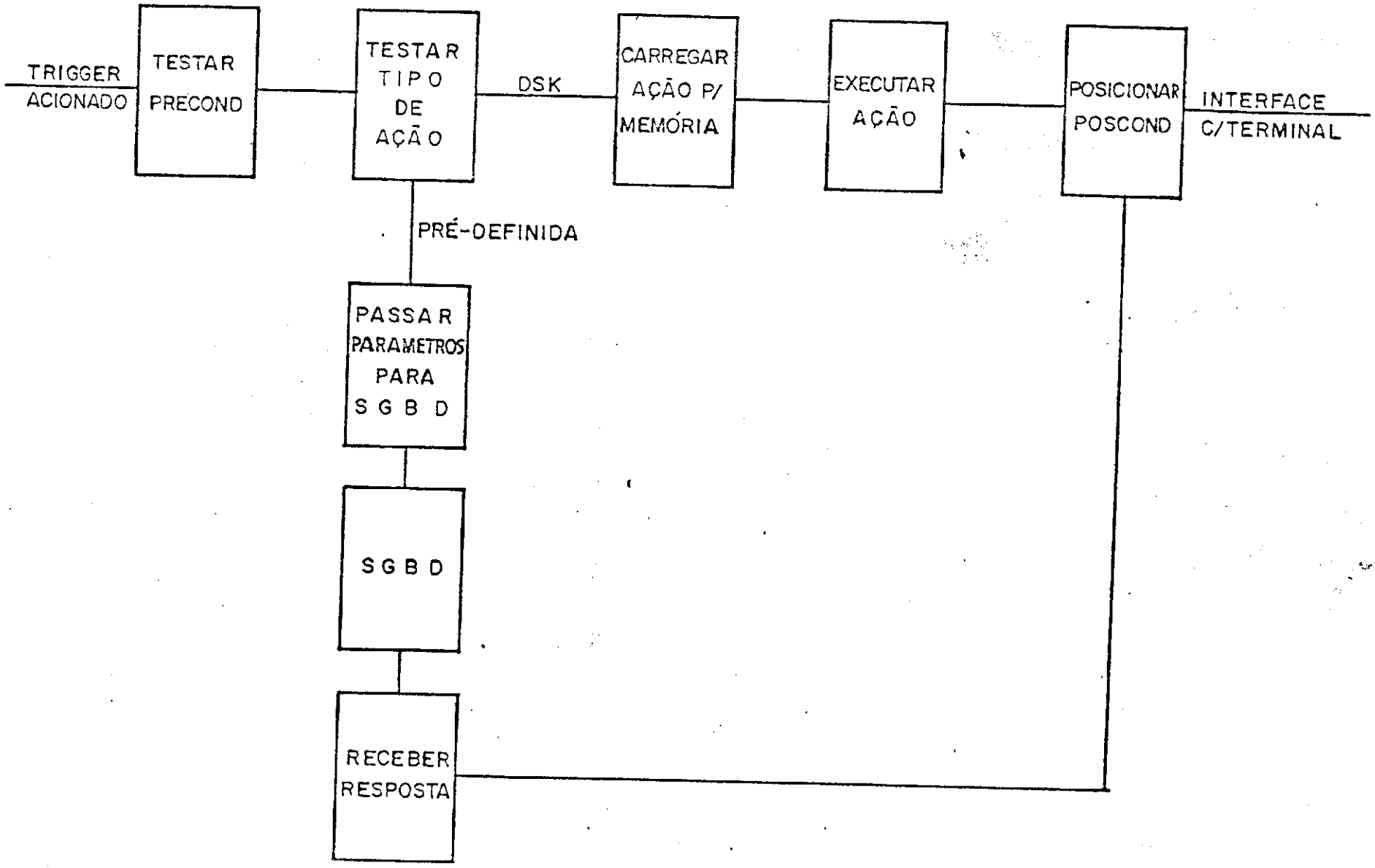

FIG. 11 - Visão da função gerenciar ações.

VII.4 - Função Gerenciar Ações de Trigger

Verifica as pré-condições para trigger e o tipo de ação a executar. Caso seja uma ação pré-definida, ocorrerá interface com o DB, passando-lhe e recebendo os parâmetros inerentes a cada ação. De acordo com a resposta do DB, a pós-condição será ou naõ posicionada. Caso a ação resida em disco, isto é, não seja pré-definida, a ação será carregada com LOAD e receberá o controle com **CALL**. A pós-condição será posicionada e o sistema passará o controle para a próxima função.

Algoritmo

```
IF PRECONDITION SATISFEITA
    REPEAT
        AÇÃO:= AÇÃO TRIGGER;
        IF AÇÃO ≠ INSERT, FIND.....
            LOAD AÇÃO;
            START AÇÃO;
            STATUS POSCOND:= 1;
        ELSE
            CARREGAR PARMDB COM CLASSE, ENTIDADE
            WHILE AÇÃO = DELETE
                AÇÃODB:= D;
            END
            WHILE AÇÃO = FIND
                AÇÃODB:= F;
            END
            WHILE AÇÃO = INSERT
                AÇÃODB:= I;
```

```
                    END
                    WHILE ......
                          .
                          .
                    END
                    CALL DATBASE(PARMDB)
                    IF RETORNDB = 0
                         STATUS POSCOND:= 1;
                    ENDIF
               ENDIF
               AÇÃO:= PRÓXIMA AÇÃO TRIGGER;
          UNTIL AÇÃO:= 0;
     ENDIF
```

VII.5 - Função Definição de Eventos e Triggers

Verifica a cada ciclo se o usuário quer definir alguma coisa e não o desejando, retorna o controle para o gerenciador de tempo. Na hipótese de o usuário desejar definir alguma coisa, terá quatro opções:

Definir trigger - inclui trigger na lista.
Definir evento - inclui evento na lista.
Deletar trigger - retira trigger da lista,
Deletar evento - retira evento da lista.

Ao final da definição do usuário, o controle passa ao gerenciador de tempo novamente.

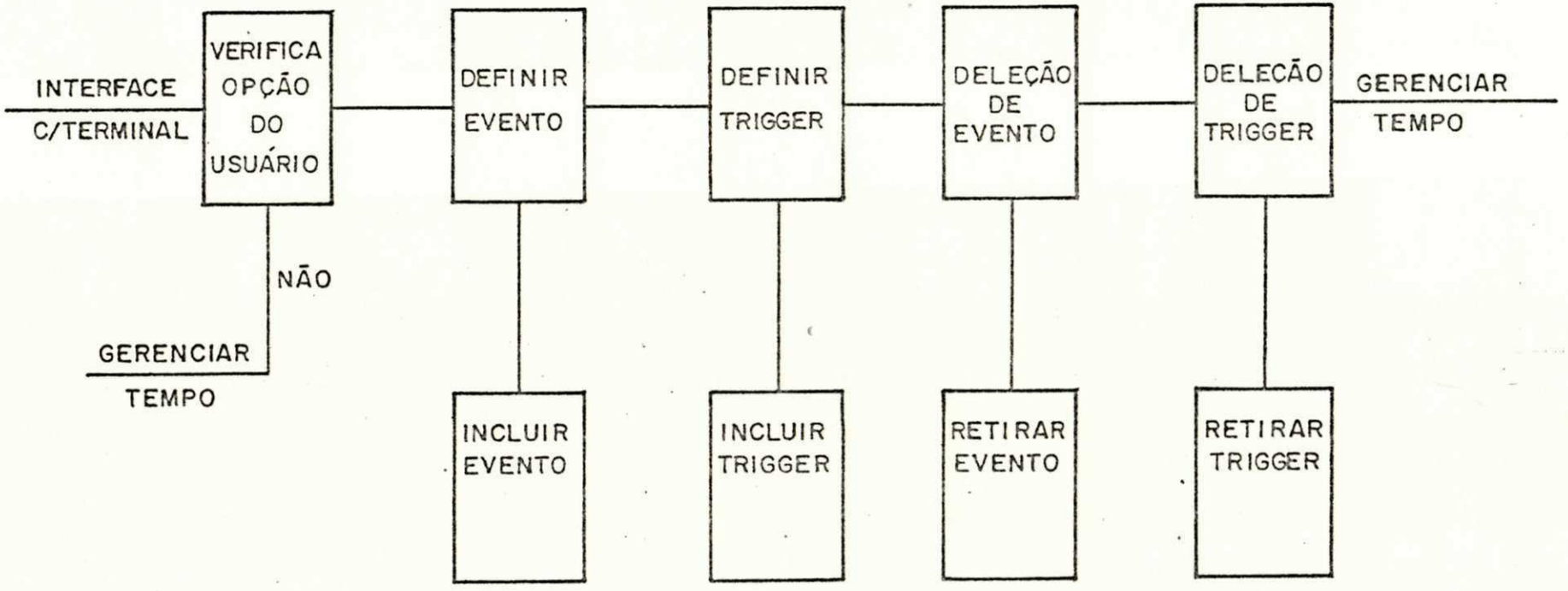

FIG. 12 - Visão da interface com o usuário.

```
Algoritmo

        IF TEMPO > 0
            DISPLAY VAI DEFINIR EVENTO OU TRIGGER
                ACCEPT AÇÃO;
            WHILE OPÇÃO = E
                ACCEPT EVENTO FROM TERMINAL
                IF ENTIDADE ≠ " "
                   CALL INCLUI EVENTO;
                ELSE
                   CALL EXCLUI-EVENTO;
                ENDIF
            END
            WHILE OPÇÃO = T
                ACCEP TRIGGER FROM TERMINAL
                IF PRECOND ≠ " "
                   CALL INCLUI-TRIGGER;
                ELSE
                   CALL EXCLUI-TRIGGER
                ENDIF
            END
        ENDIF
```

VII.6 - Módulo Posicionar Intervalo de Tempo

Este módulo adquire data e hora para mensagem de informação ao operador de monitor ativo. Esta data não servirá como referência caso o sistema encontre mais adiante alguma data no "ON". Ela simplesmente documenta o momento em que o sistema foi

ativado.

Em seguida, o intervalo de tempo para o ciclo de execução é carregado. Inicialmente este intervalo é tratado como sendo de 1 minuto, podendo ser alterado pela recompilação do módulo caso seja necessário. Esta operação é feita pela macro do supervisor OS/VS1 STIMER.

VII.7 - Módulo Teste de Final de Intervalo de Tempo

Através da macro TTIMER, do supervisor OS/VS1 é feita a checagem para verificação se final de intervalo. Caso tenha ocorrido, as tabelas de eventos e de triggers são gravadas em disco e recuperadas logo em seguida. Os arquivos são tratados de modo que se ocorrer uma queda no sistema, tanto durante o salvamento ou durante a carga, não irá afetar a integridade dos arquivos, embora se parca somente o que estiver na memória virtual.

Ao final da recarga dos arquivos para a memória, o intervalo é novamente posicionado, tendo então, início de um novo ciclo. Caso o ciclo não tenha terminado, o controle passará a função VERIFICAR EVENTOS POSICIONADOS.

VII.8 - Módulo Salvar Eventos

Efetua a descarga da lista de eventos para disco através da macro do DATA MANAGEMENT **WRITE**, tendo sido o arquivo aberto com **OUTIN**. Com isto assegurou-se que os dados em disco não

serão perdidos em caso de alguma anormalidade durante a operação.

VII.9 - Módulo Salvar Triggers

Módulo idêntico ao anterior, com a diferença de que a tabela (lista) tratada é a de triggers. Os mesmos cuidados com a integridade dos arquivos são tomados aqui.

VII.10 - Módulo Carregar Eventos

Operação de carga do arquivo de eventos para a memória através da macro READ, com arquivo aberto com INOUT, objetivando integridade em caso de situação anormal.

VII.11 - Módulo Carregar Trigger

Opera a carga da lista de triggers para a memória efetuada da mesma maneira já descrita para eventos, tomando-se os mesmos cuidados.

VII.12 - Módulo Carga Completa

Apenas faz a checagem de final da operação de salvamento e de recuperação das listas, para que se possa passar para o próximo módulo, através do parâmetro EODAD da DCB(OS/VS1).

VII.13 - Módulo Reposicionar Intervalo de Tempo

Recebe o controle ao final de cada ciclo, logo após

os quatro módulos anteriores. Sua função é posicionar o intervalo de tempo, com o objetivo de permitir novo ciclo no sistema. Feito através da macro STIMER (OS/VS1).

VII.14 - Módulo Verificar ON

Uma vez que toda a lista tenha sido varrida e nenhum evento tenha sido posicionado, ou todos os triggers correspondentes tenham sido executados, a lista de eventos é varrida para posicionamento de eventos satisfeitos, sendo que este módulo verifica a veracidade do ON para cada evento. Sendo o ON satisfeito o próximo módulo recebe o controle, caso contrário, o próximo elemento na lista é checado.

Algoritmo

```
WHILE  C(ON) ≠  " "
    IF ON ≠ PALAVRAS-CHAVE THEN
       EVINTERNO1:= STATUS-EVENTO NO ON
       INCREMENTAR ON PARA SALTAR NOME-EVENTO
     ELSE
    IF ON = AND OU OREX OU OR OU EL OU NOT
          OPCODE:= ON;
          INCREMENTA ON PARA SALTAR PALAVRA-CHAVE
          EVINTERNO2:= ON;
          STATUS-EVENTO:= EVINTERNO1(OPCODE)EVINTERNO2
    ENDIF
    IF ON = IF OU WHEN OU TIME OU DATE
```

```
        INCREMENTA ON PARA SALTAR PALAVRA-CHAVE
ENDIF
IF ON = ANTES OU ON = <
        SALTAR PALAVRA-CHAVE NO ON
        IF ON+2 = ":"
        ADQUIRIR HORA;
        MOVER PARA WORKAREA;
    ELSE
        ADQUIRIR DATA;
        MOVER PARA WORKAREA;
    ENDIF
    IF ON <  WORKAREA
         POSICIONAR STATUS = 1
    ENDIF
ENDIF
IF ON = DEPOIS OU ON =  ">"
      SALTAR PALAVRA CHAVE NO ON
      IF ON+2 = ":"
         ADQUIRIR HORA;
         MOVER PARA WORKAREA;
      ELSE
         ADQUIRIR DATA;
         MOVER PARA WORKAREA;
      ENDIF
      IF ON > WORKAREA
          POSICIONA STATUS = 1
      ENDIF
ENDIF
IF ON = DURANTE OU ON = "="
```

```
                SALTAR PALAVRA CHAVE NO ON
                IF ON+2 = "/"
                   ADQUIRIR DATA;
                   MOVER PARA WORKAREA;
                ELSE
                   ADQUIRIR HORA;
                   MOVER PARA WORKAREA;
                ENDIF
                    WORK1:= STRING(ON,8)
                    WORK2:= STRING(ON + 8,8)
                    IF WORK1 >= WORKAREA AND
                       WORK2 <= WORKAREA
                          POSICIONA STATUS = 1
                    ENDIF
                ENDIF
                INCREMENTAR ON
                IF ON = TRUE
                    IF EVINTERNO1 = 1
                        STATUS:= 1;
                    ENDIF
                ENDIF
                IF ON = FALSE
                   IF EVINTERNO = 0
                       STATUS:= 1;
                    ENDIF
                ENDIF
            END
```

VII.15 - Módulo Testar CONDITION

Recebe o controle caso o ON para o evento esteja satisfeito. Caso a CONDITION seja avaliada como verdadeira o próximo módulo recebe o controle, caso contrário o próximo elemento é verificado.

Algoritmo

```
WHILE  C(CONDITION)  ≠ " "
     IF CONDITION ≠ PALAVRAS-CHAVE THEN
        EVINTERNO1:= STATUS-EVENTO NO CONDITION
        INCREMENTAR CONDITION PARA SALTAR NOME-EVENTO
      ELSE
     IF CONDITION = AND OU OREX OU OR OU EL
          OPCODE:= CONDITION;
          INCREMENTA CONDITION P/ SALTAR PALAVRA-CHAVE;
          EVINTERNO2:= CONDITION;
          STATUS EVENTO:= EVINTERNO1(OPCODE)EVINTERNO2;
     ENDIF
     IF CONDITION = IF OU WHEN OU TIME OU DATE
           INCREMENTA CONDITION P/ SALTAR PALAVRA-CHAVE
     ENDIF
     IF CONDITION = ANTES OU CONDITION = <
           SALTAR PALAVRA CHAVE NO CONDITION
           IF CONDITION+2 = ":"
              ADQUIRIR HORA;
              MOVER PARA WORKAREA;
             ELSE
```

```
            ADQUIRIR DATA;
            MOVER PARA WORKAREA;
         ENDIF
         IF CONDITION < WORKAREA
               POSICIONAR STATUS = 1
         ENDIF
      ENDIF
      IF CONDITION = DEPOIS OU CONDITION = >
            SALTAR PALAVRA-CHAVE NO CONDITION
            IF CONDITION+2 = ":"
               ADQUIRIR HORA;
               MOVER PARA WORKAREA;
            ELSE
               ADQUIRIR DATA;
               MOVER PARA WORKAREA;
            ENDIF
            IF CONDITION > WORKAREA
                POSICIONA STATUS = 1
            ENDIF
      ENDIF
      IF CONDITION = DURANTE OU CONDITION = "="
            SALTAR PALAVRA-CHAVE NO CONDITION
            IF CONDITION+2 = "/"
               ADQUIRIR DATA;
               MOVER PARA WORKAREA;
              ELSE
                ADQUIRIR HORA;
                MOVER PARA WORKAREA;
```

```
                ENDIF

                WORK1 := STRING(CONDITION,8);
                WORK2 := STRING(CONDITION+8,8)
                IF WORK1  >= WORKAREA AND
                   WORK2  <= WORKAREA
                       POSICIONA STATUS = 1
                ENDIF
            ENDIF
            INCREMENTAR CONDITION
            IF CONDITION = TRUE
                IF EVINTERNO1 = 1
                    STATUS = 1
                ENDIF
                ENDIF
                IF CONDITION = FALSE
                    IF EVINTERNO1 = 0
                        STATUS = 1
                    ENDIF
                ENDIF
            ENDIF
```

VII.16 - Módulo Posicionar Evento

Sendo o ON e a CONDITION verdadeiros para um evento este módulo é ativado, posicionando seu status para verdadeiro, ou seja, = 1. Este módulo é interno ao ON e ao CONDITION.

VII.17 - **Módulo Ativar Correspondentes**

Gerenciar a execução da função gerenciar ações de triggers. Na verdade, é uma bandeira (FLAG) para que se passe a esta função.

VII.18 - **Módulo Testar PRECOND**

O teste de um PRECOND é feito de maneira imediata pela carga do endereço de evento e teste de seu status. Para o caso de condições compostas, o status de cada evento é guardado e a expressão é avaliada da esquerda para a direita. Sendo o PRECOND avaliado como verdadeiro, o prósimo módulo recebe o controle. Não sendo verdadeiro, volta ao MÓDULO VERIFICAR EVENTO POSICIONADO.

Algoritmo

```
WHILE C(PRECOND) ≠ " "
    IF PRECOND = PALAVRAS-CHAVE  THEN
       EVINTERNO1:= STATUS-EVENTO NO PRECOND
       INCREMENTAR PRECOND PARA SALTAR NOME-EVENTO
    ELSE
    IF PRECOND = AND OU OREX OU OR OU EL NOT
          OPCODE:= PRECOND;
          INCREMENTA PRECOND P/ SALTAR PALAVRA-CHAVE
          EVINTERNO2:= PRECOND;
          STATUS EVENTO:= EVINTERNO1(OPCODE)EVINTERNO2
    ENDIF
    IF PRECOND = IF OU WHEN OU TIME OU DATE
```

```
        INCREMENTA PRECOND P/ SALTAR PALAVRA CHAVE
ENDIF
IF PRECOND = ANTES  OU PRECOND =  <
      SALTAR PALAVRA-CHAVE NO PRECOND
      IF PRECOND+2 = ":"
          ADQUIRIR HORA;
          MOVER PARA WORKAREA;
         ELSE
          ADQUIRIR DATA;
          MOVER PARA WORKAREA;
        ENDIF
        IF PRECOND < WORKAREA
            POSICIONAR STATUS = 1
        ENDIF
    ENDIF
    IF PRECOND = DEPOIS OU PRECOND = >
        SALTAR PALAVRA-CHAVE NO PRECOND
        IF PRECOND+2 =":"
          ADQUIRIR HORA;
          MOVER PARA WORKAREA;
         ELSE
           ADQUIRIR DATA;
           MOVER PARA WORKAREA;
         ENDIF
         IF PRECOND > WORKAREA
             POSICIONA STATUS = 1
         ENDIF
     ENDIF
     IF PRECOND = DURANTE OU PRECOND = "="
```

```
                    SALTAR PALAVRA CHAVE NO PRECOND
                    IF PRECOND+2 = "/"
                      ADQUIRIR DATA;
                      MOVER PARA WORKAREA;
                     ELSE
                       ADQUIRIR HORA
                       MOVER PARA WORKAREA;
                    ENDIF
                    WORK1:= STRING(PRECOND,8);
                    WORK2:= STRING(PRECOND,8);
                    IF WORK1 >= WORKAREA AND
                       WORK2 <= WORKAREA
                            POSICIONA SATATUS = 1
                     ENDIF
               ENDIF
               INCREMENTA PRECOND
               IF PRECOND = TRUE
                    IF EVINTERNO1 = 1
                         STATUS = 1
               ENDIF
          ENDIF
          IF PRECOND = FALSE
             IF EVINTERNO1 = 0
                  STATUS = 1
             ENDIF
        ENDIF
END
```

VII.19 - Módulo Testar Tipo de Ação

Módulo necessário para verificar tipo de ação deseja da no trigger. O campo de ação é varrido e encontrando-se um dos comandos FIND, REQUEST, INSERT, DELETE, UPDATE, CREATE o controle é passado ao próximo módulo. Caso contrário, o controle é passado ao MÓDULO CARREGAR AÇÃO PARA MEMÓRIA.

Este módulo é interno ao MÓDULO GERENCIAR AÇÕES.

VII.20 - Módulo Passar Parâmetros para SGBD

De acordo com a ação o campo de parâmetros é preenchi do e um ATTACH é feito para o DB, esperando-seo fim da operação com um WAIT. Isto significa que o ciclo de tempo continuará sendo contabilizado, enquanto o DB providencia a operação da ação requi sitada. Módulo Interno ao Módulo Gerenciar Ações.

VII.21 - Módulo SGBD

É a interface entre o DB e o sistema de Eventos e triggers, sendo ativado com este nome, o qual pode ser modificado pela recompilação do módulo. O DB deve prever as operações FIND, REQUEST, CREATE, DELETE, INSERT, UPDATE, bem como a interface via zona de parâmetros. Seu retorno deve ser feito via um BR 14, RE TURN, GOBACK ou STOP RUN, pois em qualquer destas situações o sis tema de eventos e triggers reassumirá o controle.

VII.22 - **Módulo Receber Resposta**

Dependendo da operação de DB requisitada, pode testar o campo S da área de parâmetros para verificar se tudo ocorreu bem, ou ler a extensão da área de parâmetros a partir do byte 52 (caso de um REQUEST). Caso a situação retornada pelo DB esteja de acordo com o requisitado passa-se ao módulo posicionar POSCOND. Caso contrário volta-se ao MÓDULO VERIFICAR EVENTO POSICIONADO.

VII.23 - **Módulo Carregar Ação para Memória**

Ativado quando a ação não é uma operação de DB, podendo ser um programa, uma procedure. Efetua a carga da ação, pelo nome, com a macro LOAD.

VII.24 - **Módulo Executar Ação**

Efetua um CALL para a ação carregada anteriormente executando-a. Ao seu término, a ação é eliminada da memória com a macro DELETE.

VII.25 - **Módulo Posicionar POSCOND**

Sendo a ação do trigger bem sucedida, este módulo endereçará o evento indicado em pós-condition e o posicionará como verdadeiro, ou seja, =1. Volta a verificar evento posicionado, e ao final da lista de eventos passa-se ao módulo seguinte.

VII.26 - Módulo Verificar se Usuário

Efetua via vetor, uma indagação ao operador se vai ocorrer nova definição. Não havendo, retorna ao teste de fim de intervalo de tempo. Havendo a intensão de movimentação passa ao próximo módulo.

VII.27 - Módulo Definir Trigger

Se o operador/usuário, seleciona definição de trigger o sistema interativamente pede-lhe os campos necessários, sendo que as pré-conditions e as pós-conditions são checadas quanto a sua existência, e em caso de erro, toda a definição é reiniciada. Não havendo erros o próximo módulo é ativado. Neste módulo, todos os nomes de eventos na pré-condition e na pós-condition são transformados para apontadores com o objetivo de acelerar as fases da pesquisa. Módulo interno a função definição de eventos e triggers.

VII.28 - Módulo Inclui Trigger

Aqui o ponteiro do trigger é incluído no campo correspondente de seus eventos associados na pré-condition, e o próprio trigger é incluído na lista. Atualiza ponteiro de trigger anterior e posterior.

Algoritmo

```
INCLUI TRIGGER
        ADQUIRE MEMÓRIA PARA TRIGGER;
        MOVER TRIGGER PARA MEMÓRIA;
```

```
∂TRIGGER:= ∂PROX TRIGGER DO TRIGGER ANTERIOR
∂TRIGGER ANTERIOR:= ∂TRIGGER ANTERIOR NO.
                    TRIGGER ATUAL
RETURN;
```

VII.29 - Módulo Definir Evento

Interativamente pede seus campos ao operador/usuário e passa o controle ao próximo módulo.

VII.30 - Módulo Incluir Evento

Inclui o evento na lista de eventos e atualiza os ponteiros de evento anterior e posterior.

Algoritmo

```
INCLUI EVENTO
    ADQUIRIR MEMÓRIA PARA EVENTO
    MOVER EVENTO PARA MEMÓRIA;
    ∂EVENTO:= ∂PROX EVENTO NO EVENTO ANTERIOR;
    ∂EVENTO ANTERIOR:= ∂EVENTO ANTERIOR NO  EVENTO
                       ATUAL;
    RETURN
```

VII.31 - Módulo Deletar Trigger

Se o operador/usuário entrar com um nome de trigger sem os demais componentes, o sistema interpretará como sendo soli

citada uma operação de exclusão de trigger, que ativa o próximo módulo.

VII.32 - **Módulo Retirar Trigger da Lista**

Atualiza os ponteiros anterior e posterior do trigger seguinte e antecedente, respectivamente, anulando a definição do trigger selecionado.

Algoritmo

```
EXCLUI TRIGGER
      LOCALIZA TRIGGER PELO NOME;
      LIBERA MEMÓRIA DE ALOCAÇÃO DO TRIGGER;
      ∂TRIGGER ANTERIOR:= ∂TRIGGER ANTERIOR;
      ∂TRIGGER POSTERIOR:= ∂TRIGGER POSTERIOR;
      ∂PROX TRIGGER NO TRIGGER ANTERIOR:= ∂TRIGGER AN
                                          TERIOR;
      ∂TRIGGER ANTERIOR NO PRÓXIMO TRIGGER:= ∂TRIGGER
                                             PRÓXIMO;
      RETURN;
```

VII.33 - **Módulo Deletar Evento**

Atualiza ponteiros anterior e posterior dos eventos sucedente e antecedente, respectivamente, anulando uma definição de evento. O controle retorna ao TESTE DE FIM DE INTERVALO DE TEMPO.

Algoritmo

```
EXCLUI EVENTO
      LOCALIZA EVENTO PELO NOME;
      LIBERA MEMÓRIA DE ALOCAÇÃO DO EVENTO;
      ∂EVENTO ANTERIOR:= ∂EVENTO ANTERIOR;
      ∂EVENTO POSTERIOR:= ∂EVENTO POSTERIOR;
      ∂PROX EVENTO NO EVENTO ANTERIOR:= ∂EVENTO  ANTE
                                                RIOR;
      ∂EVENTO ANTERIOR NO PRÓXIMO EVENTO:= ∂EVENTO
                                                PRÓXIMO;
      RETURN;
```

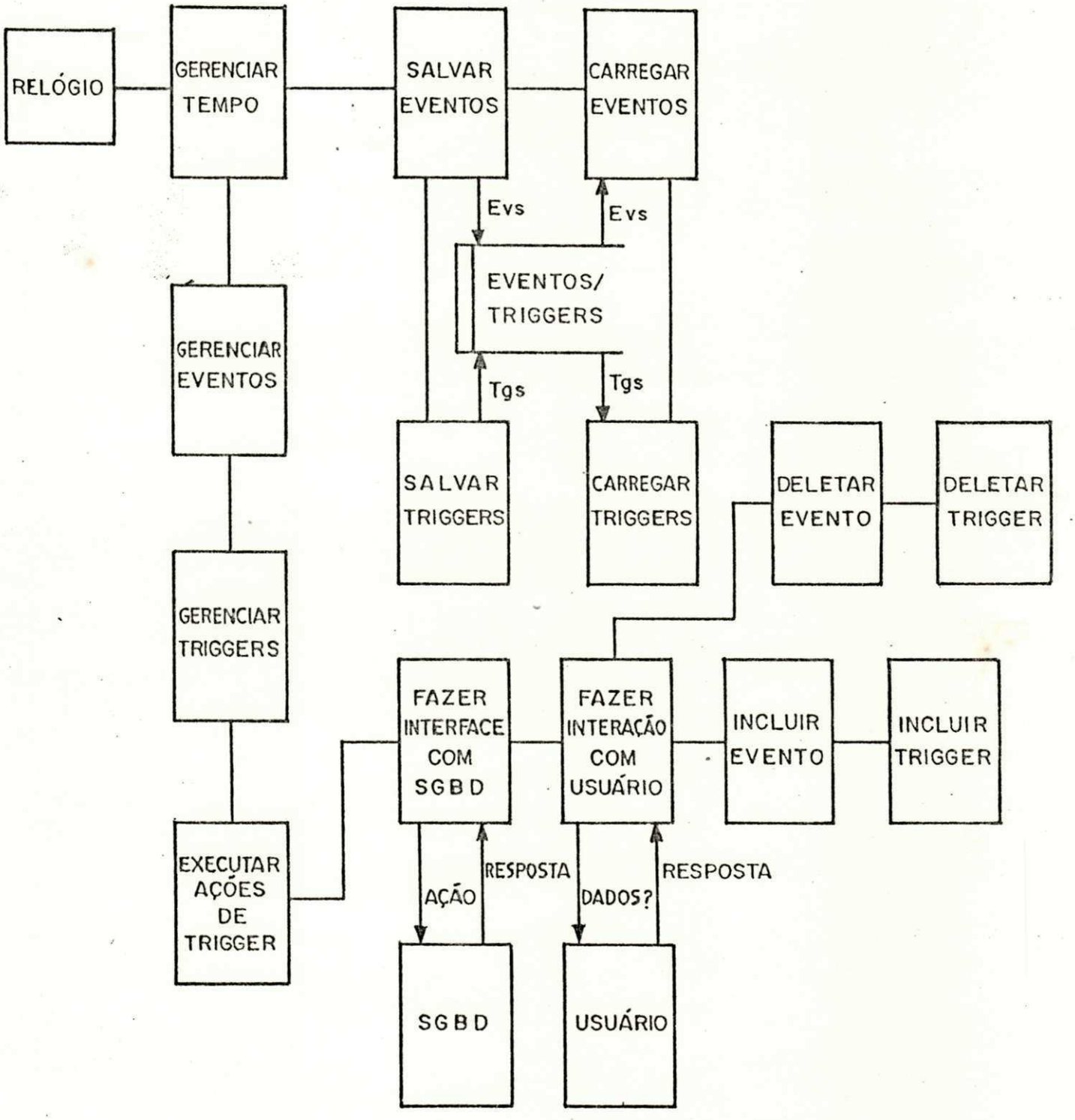

FIG. 13 - O sistema de eventos e triggers.

CAPÍTULO VIII

CONCLUSÕES

Como pode ser observado, é perfeitamente viável a definição de um sistema de eventos e triggers para sistema de informação em Bancos de Dados semânticos e particularmente no Modelo THM [SC 82]. O Esquema Conceitual contém a descrição das entidades e classes, assim como as restrições de integridade que devem ser satisfeitas. A definição do Esquema Interno não é aqui por nós considerada, mas nos parece que o Modelo Relacional dispõe de ferramentas que suportam este mapeamento.

O objetivo principal deste trabalho foi calcado no fornecimento de uma ferramenta para facilitar a execução automática de serviços, através da definição de eventos. Acreditamos ser também o tempo fator fundamental na vida de um sistema de informação e implementamos calendário e relógio, com a finalidade de atender requisitos dessa natureza.

Dificuldades foram sentidas para efetivar ciclos de tempo dentro do sistema de eventos e triggers, assim como para tratamento da granularidade do tempo. A falta de uma realidade para atestar com mais firmeza a potência da ferramenta, foi também um fator importante, já que a abstração e simulação demandam mais tempo na continuidade do trabalho.

No cômputo geral, na nossa opinião, as facilidades projetadas para o Modelo THM [SC 82] , superam as dificuldades, os problemas encontrados não nos parecem tão complexos, a ponto de afetar a boa impressão que tivemos do modelo.

A sugestão de um trabalho que cuide de efeitos colaterais, em face das alterações e eventos que venham a ocorrer num determinado sistema, é por nós colocada, objetivando atender requisitos de integridade. Acreditamos também, que com algumas modificações, o sistema de eventos e triggers serve como ponto de partida para uma ferramenta de controle de processos.

BIBLIOGRAFIA

[AT82] ATZENI, PAOLO; et all - "A Computer Aidet Tool For Conceptual Databse Design. In Automated Tools For Information Systems Design", North Holland Publishing Company, IFIP, 1982.

[BRO84] BRODIE, MICHAEL L. - "On The Design And Specification of Database Transactions", In Michael L.B., John M. and Joachim W.S. (eds), On Conceptual Modelling, Springer-Verlag, 1984.

[CHE83] CHEESEMAN, PETER - "A Represetation of Time for Planning", Technical Note 278 In SRI International, 1983.

[HOR85] HORNDASCH, ANGELIKA; et all - "An Aproach to Conceptual Schema Design of Information Systems", In Sernadas et all (eds.), PROC. IFIP WG 8.1 - Working Conference on Theoretical and Formal Aspects of Information Systems, North Holland Publishing Company, 1985.

[RIC85] RICHTER, G. - "Clocks and Their Use For Time Modelling", In Sernadas et all (eds.), Proc. IFIP WG 8.1, Working Conference on Theoretical and Formal Aspects of Information Systems, North Holland Publishing Company, 1985.

[KR84] KING, ROGER e MCLEOD, DENIS - "A Unified Model And Methodology For Conceptual Database Desing", In Michael L.B., John M., and Joachim W.S. (eds.), On Conceptual Modelling, Spring-Verlag, 1984.

[SC82] SCHIEL, ULRICH - "The Temporal-Hierarchic Data Model (THM)", Institut Für Informatic, Universitat, Stutgart, Bericht, 10/1982.

[SC84] SCHIEL, ULRICH - "A Semantic Data Model For Conceptual Schemas and Their Mapping to Internal Relational Schemas University of Stuttgart, Doctoral Thesis, 1984.

[SC85] SCHIEL, ULRICH - "Sistemas com Expressões Temporais e Informações Imprecisas", Reporte Técnico, Universidade Federal da Paraíba, 1985.

[SMI82] SMITH, J.M. e SMITH, D.C.P. - "Principles of Database Conceptual Design", In G. Goos, J. Hartmanis (eds.), lectures Notes in Computer Science, Springer-Verlag, 1982.

[STU84] STUDER, RUDI - "Modelling office Information Systems By Using Timed THM-Nets", Tecnical Report 8/84, University of Stuttgart, 1984.

[WAL83] WALTER, BERND - "Timed Petri-Nets For Modellinf And Analyzing Protocols With Real Time Caracteristics", In H. Rudin, C. West (eds.), Protocol Specilication, Testing Verifications, North Holland Publishing Company, 1983.

APÊNDICE

Estudo de Caso

SUPONDO A DEFINIÇÂO DO SEGUINTE EVENTO E TRIGGER

EVENTO

NOME	TESTE1
ON	DURANTE 15/08/86 15/09/86
CONDITION	TERMINAR-CURSO

TRIGGER

NOME	FINAL DE CURSO
PRECONDITION	TESTE1
AÇÃO	IMPREL
POSCONDITION	TESTE1 FALSE

IMPREL = NOME DE AÇÃO RESIDENTE EM DISCO, QUE SERÁ ACIONADA (EXECUTADA) SE A PRECONDITION FOR SATISFEITA.

TESTE1 FALSE = PARA DESPOSICIONAR O EVENTO E NÃO ATIVAR O TRIGGER.

COMPORTAMENTO DE SISTEMA

```
SETAR INTERVALO DE TEMPO PARA 1 MIN.
REPETIR
ENQUANTO INTERVALO NÃO ACABAR
  LER TECLADO
  SE DEFINIÇÃO DE EVENTO PELO OPERADOR
```

```
        LER NOME DO EVENTO
        LER ON
        LER CONDITION
        INCLUIR EVENTO
            ADQUIRIR MEMÓRIA PARA EVENTO
            MOVER EVENTO PARA MEMÓRIA
            MOVER MEMÓRIA PARA FINAL LISTA EVENTOS
            MOVER MEMÓRIA P/ CAMPO PROX.EVENTO  NO EVENTO ANTERIOR
            MOVER O PARA CAMPO  PRO.EVENTO NO EVENTO ATUAL
            MOVER EVENTO ANTERIOR PARA EVENTO ATUAL
        FIM INCLUIR EVENTO
FIMSE
SE DEFINIÇÃO DE TRIGGER
   LER NOME DE TRIGGER
   LER PRECONDITION
   ANALISAR PRECONDITION
        SE PRECONDITION = NOME DE EVENTO
            VARRER LISTA DE EVENTOS ATÉ ENCONTRAR
            MOVER EVENTO PARA CAMPO PRECONDITION
        FIMSE
   FIM ANALISAR PRECONDITION
   LER AÇÃO
   LER POSCONDITION
   ANALISAR POSCONDITION
        SE POSCONDITION = NOME DE EVENTO
            VARRER LISTA DE EVENTOS ATÉ ENCONTRAR
```

```
            MOVER EVENTO PARA CAMPO POSCONDITION
        FIMSE
    FIM ANALISAR POSCONDITION
    INCLUIR TRIGGER
        ADQUIRIR MEMÓRIA PARA TRIGGER
        MOVER TRIGGER PARA MEMÓRIA
        MOVER MEMÓRIA P/ FINAL LISTA DE TRIGGERS
        MOVER MEMÓRIA P/ CAMPO PROX.TRIGGER NO TRIGGER ANTERIOR.
        MOVER O PARA CAMPO PROX. TRIGGER NO TRIGGER ATUAL
        MOVER TRIGGER ANTERIOR PARA TRIGGER ATUAL
    FIM INCLUIR TRIGGER
FIMSE
POSICIONA LISTA DE EVENTOS NO INÍCIO
ENQUANTO NÃO ACABA LISTA DE EVENTOS
        ANALISAR ON
        SE  ON  SATISFEITA
            POSICIONAR STATUS ON
        FIMSE
        ANALISAR CONDITION
        SE CONDITION SATISFEITA
            POSICIONAR STATUS CONDITION
        FIMSE
        STATUS EVENTO = STATUS ON  AND  STATUS CONDITION
        CARREGAR ENDEREÇO DO PRÓXIMO EVENTO NA LISTA
    FIM ENQUANTO
    POSICIONA LISTA DE TRIGGERS NO INÍCIO
    ENQUANTO NÃO ACABAR LISTA DE TRIGGERS
```

```
        SE PRECONDITION POSICIONADO
            EXECUTAR AÇÕES
            POSICIONAR POSCONDITION
        FIMSE
        CARREGAR ENDEREÇO PRÓXIMO TRIGGER NA LISTA
    FIM ENQUANTO
    FIM ENQUANTO INTERVALO
    SE INTERVALO ACABOU
        SALVAR EVENTOS EM DISCO
        SALVAR TRIGGERS EM DISCO
        CARREGAR EVENTOS PARA MEMÓRIA
        CARREGAR TRIGGERS PARA MEMÓRIA
        POSICIONAR INTERVALO DE TEMPO PARA 1 MIN.
    FIMSE
```