SANDRAMARA SCANDELARI KUSANO DE PAULA SOARES

MaTrigs – UM AMBIENTE DE INTERFACE PARA MAPEAMENTO DE TRIGGERS EM MODELO ERC+

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre, Curso de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.

Orientadora: Prof.ª Dr.ª Laura Sánchez García

CURITIBA

2003

Ministério da Educação
Universidade Federal do Paraná
Mestrado em Informática

PARECER

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática, da aluna *Sandramara Scandelari Kusano de Paula Soares*, avaliamos o trabalho intitulado, "*MaTrigs – Um ambiente de interface para mapeamento de triggers em modelo ERC+*", cuja defesa foi realizada no dia 29 de setembro de 2003, às quatorze horas, no Auditório da Informática da Universidade Federal do Paraná. Após a avaliação, decidimos pela aprovação da candidata.

Curitiba, 29 de setembro de 2003.

Profª. Dra. Laura Sanchez Garcia
DINF/UFPR - Orientadora

Prof. Dr. Luiz Ernesto Merkle
CEFET/PR – Membro Externo

Prof. Dr. Marcos Sfair Sunyé
DINF/UFPR – Membro Interno

À minha filha Ana Harumi que
acompanhou todo o desenvolvimento
deste trabalho, antes mesmo de nascer.

AGRADECIMENTOS

À professora e orientadora Laura Sánches García pelo incentivo, acompanhamento e esclarecimentos indispensáveis para a conclusão deste trabalho.

Ao meu marido Mario, que sempre me incentivou e me deu todo o apoio para que eu pudesse realizar este trabalho, soube compreender as dificuldades e teve muita, mas muita paciência. É muito bom ter você ao meu lado.

À minha mãe, Ana Maria, pois tudo o que sou devo a ela. Obrigada.

A todos os meus familiares, em especial à Vó Ângela, Sueli, Anderson e Gandy, pela força e carinho.

À direção, professores, funcionários e alunos da Escola Técnica da UFPR pelo apoio recebido para a realização deste trabalho.

SUMÁRIO

SUMÁRIO IV

LISTA DE FIGURAS VI

LISTA DE TABELAS VIII

LISTA DE ABREVIAÇÕES E SIGLAS IX

RESUMO X

ABSTRACT XI

1 INTRODUÇÃO 12

1.1 PROBLEMA 13

1.2 MODELO ERC+, INTEGRAÇÃO DE ESQUEMAS E AMBIENTE J-SCHEMAS INTEGRATOR 14

1.3 JUSTIFICATIVA 14

1.4 OBJETIVO 16

1.5 ORGANIZAÇÃO DA DISSERTAÇÃO 16

2 O MODELO ERC+ 18

2.1 A ÁLGEBRA DO MODELO 21

2.2 REPRESENTAÇÃO VISUAL 22

3 A INTEGRAÇÃO DE ESQUEMAS, A FERRAMENTA J-SCHEMAS INTEGRATOR, A LINGUAGEM SQL E OS TRIGGERS 24

3.1 INTEGRAÇÃO DE ESQUEMAS 24

3.2 O AMBIENTE J-SCHEMAS INTEGRATOR 26

3.3 STRUCTURED QUERY LANGUAGE 28

3.3.1 A LINGUAGEM 28

3.3.2 TRIGGERS 30

3.3.2.1 TRIGGERS NO ORACLE8 33

4 INTERAÇÃO SER HUMANO-COMPUTADOR 36

4.1 ABORDAGENS COGNITIVAS 37

4.1.1 ENGENHARIA COGNITIVA 37

4.1.2 MODELO GOMS - GOALS, OPERATIONS, METHODS AND SELECTION RULES 40

4.2 ENGENHARIA SEMIÓTICA 41

4.2.1 LINGUAGEM DE ESPECIFICAÇÃO DA MENSAGEM DO DESIGNER 42

4.3 UM PROCESSO DE DESIGN DE INTERFACES 44

5 O MÓDULO DE VISUALIZAÇÃO DE TRIGGERS 47

5.1 O PROCESSO DE DESIGN DE INTERFACES NA FERRAMENTA MATRIGS 47

5.2 FUNCIONAMENTO DO SISTEMA DE MAPEAMENTO DE TRIGGERS 52

5.2.1 IDENTIFICAÇÃO DAS ENTIDADES DO BANCO DE DADOS 54

5.2.2 IDENTIFICAÇÃO DOS TRIGGERS 55

5.2.3 ANÁLISE DA SEMÂNTICA DOS TRIGGERS 55

5.2.4 GERAÇÃO DAS ASSOCIAÇÕES ENTRE AS ENTIDADES DO TRIGGER .. 56

5.2.5 REPRESENTAÇÃO GRÁFICA DOS TRIGGERS NO MODELO ERC+ 56

5.3 UMA SESSÃO DA FERRAMENTA PROPOSTA 58

6 CONCLUSÕES E TRABALHOS FUTUROS 67

REFERÊNCIAS BIBLIOGRÁFICAS 69

APÊNDICE I - MODELO GOMS 73

APÊNDICE II - LINGUAGEM DE ESPECIFICAÇÃO DA MENSAGEM DO DESIGNER 77

APÊNDICE III - CORRELAÇÃO MENSAGEM-WIDGETS DA INTERFACE 82

APÊNDICE IV - TELAS DO SISTEMA 85

ANEXO I - DIAGRAMAS DE SINTAXE DE TRIGGERS EM ORACLE 92

LISTA DE FIGURAS

FIGURA 2.1 EXEMPLO DE UM DIAGRAMA ERC+ (PARENT92) 19

FIGURA 2.2 COMPARAÇÃO DE NOTAÇÃO ENTRE DIAGRAMA DE VENN E DIAGRAMA ERC+ (SPACCAPIETRA95) 20

FIGURA 2.3 REPRESENTAÇÃO VISUAL DO ERC+, ADAPTADO DE SPACCAPIETRA ET. AL (SPACCAPIETRA92) 23

FIGURA 3.1 PROCESSO GLOBAL DE INTEGRAÇÃO, ADAPTADO DE SPACCAPIETRA ET AL. (PARENT98). 25

FIGURA 4.1 MODELO ENVOLVIDOS NA ENGENHARIA COGNITIVA, ADAPTADO DE NORMAN (NORMAN&DRAPER86) 38

FIGURA 4.2 GOLFOS DE AVALIAÇÃO E EXECUÇÃO, ADAPTADO DE HUTCHINS, HOLLAN AND NORMAN (HUTCHINS86). 40

FIGURA 5.1 TELAS DO MS-SQL PARA GERAR SCRIPT DO BANCO DE DADOS 52

FIGURA 5.2 TRECHO DO SCRIPT DE UMA BASE DE DADOS ONDE PODE-SE VISUALIZAR OS OBJETOS E SEUS RESPECTIVOS ATRIBUTOS 53

FIGURA 5.3 TELA DA FERRAMENTA J-SCHEMAS INTEGRATOR COM BOTÃO PARA ACESSAR O AMBIENTE MATRIGS..... 53

FIGURA 5.4 REPRESENTAÇÃO DE TRIGGERS NO MODELO ERC+ 57

FIGURA 5.5 COMPARAÇÃO DA REPRESENTAÇÃO DE TRIGGERS EM MODELO ERC+ EM RELACAO À UTILIZAÇÃO DE CORES..... 58

FIGURA 5.6 TELA PARA FORNECER AS INFORMAÇÕES DA BASE DE DADOS NOTA FISCAL 60

FIGURA 5.7 TELA COM O MAPEAMENTO DE TRIGGERS DA BASE DE DADOS NOTA FISCAL 61

FIGURA 5.8 DETALHE DA TELA DE MAPEAMENTO DO TRIGGER ITEMNOTAFISCAL_INSERT – COMANDO SQL EM FORMATO TEXTO 62

FIGURA 5.9 DETALHE DA TELA DE MAPEAMENTO DO TRIGGER ITEMNOTAFISCAL_INSERT – COMANDO SQL EM FORMATO DE GRAFO 62

FIGURA 5.10 DETALHE DA TELA DE MAPEAMENTO DO TRIGGER ITEMNOTAFISCAL_INSERT - VISUALIZAR..... 63

FIGURA 5.11 TELA DE MAPEAMENTO DE TRIGGERS COM A OPÇÃO DEPENDÊNCIA DINÂMICA - ALTERA 64

FIGURA 5.12 TELA DE MAPEAMENTO DE TRIGGERS COM A OPÇÃO DEPENDÊNCIA DINÂMICA – ALTERADO POR..... 65

FIGURA 5.13 TELA VISUALIZANDO TODOS OS TRIGGERS DA BASE DE DADOS NOTA FISCAL 66

FIGURA A1 TELA PARA VISUALIZAÇÃO E INFORMAÇÕES DO MAPEAMENTO DE TRIGGERS..... 85

FIGURA A2 TELA DE ENTRADA DE DADOS DAS INFORMAÇÕES DA BASE DE DADOS A SER MAPEADA..... 86

FIGURA A3 DETALHE DA TELA DE MAPEAMENTO DE TRIGGERS PARA VISUALIZAR O COMANDO SQL EM FORMATO TEXTO..... 86

FIGURA A4 DETALHE DA TELA DE MAPEAMENTO DE TRIGGERS PARA VISUALIZAR O COMANDO SQL EM FORMATO DE GRAFO..... 87

FIGURA A5 DETALHE DA TELA DE MAPEAMENTO DE TRIGGERS PARA VISUALIZAR UM TRIGGER SELECIONADO GRAFICAMENTE ... 87

FIGURA A6 DETALHE DA TELA DE MAPEAMENTO DE TRIGGERS PARA LISTAR E VISUALIZAR GRAFICAMENTE AS ENTIDADES QUE SÃO ALTERADAS PELA ENTIDADE SELECIONADA..... 88

FIGURA A7 DETALHE DA TELA DE MAPEAMENTO DE TRIGGERS PARA LISTAR E VISUALIZAR GRAFICAMENTE AS ENTIDADES QUE ALTERAM UMA ENTIDADE SELECIONADA..... 88

FIGURA A8 MENSAGEM DE PREENCHIMENTO INCORRETO DO CAMPO SCRIPT NA TELA FORNECENDO O SCRIPT..... 88

FIGURA A9 MENSAGEM DE PREENCHIMENTO INCORRETO DO CAMPO BASE DE DADOS NA TELA FORNECENDO O SCRIPT..... 89

FIGURA A10 MENSAGEM DE FALTA DE PREENCHIMENTO DO CAMPO SCRIPT NA TELA FORNECENDO O SCRIPT..... 89

FIGURA A11 MENSAGEM DE FALTA DE PREENCHIMENTO DO CAMPO BASE DE DADOS NA TELA FORNECENDO O SCRIPT..... 89

FIGURA A12 TELA PARA VISUALIZAÇÃO GRÁFICA DE TODOS OS TRIGGERS QUE COMPÕE UMA BASE DE DADOS 90

FIGURA A13 DIAGRAMA DE SEQÜÊNCIA DE TELAS DO SISTEMA DE MAPEAMENTO DE TRIGGERS..... 91

LISTA DE TABELAS

TABELA 2.1 MENSAGENS DA UTILIZAÇÃO DA LEMD PARA A ENGENHARIA SEMIÓTICA DE INTERFACES DE USUÁRIO 43

TABELA 5.1 ESTRUTURA HIERÁRQUICA DE TAREFAS 51

TABELA 5.2 ESTRUTURA DA ENTIDADE PRODUTO 59

TABELA 5.3 ESTRUTURA DA ENTIDADE ITEMNOTAFISCAL 59

LISTA DE ABREVIAÇÕES E SIGLAS

ANSI	*American National Standards Institute*
BD	Banco de Dados
ER	Entidade Relacionamento
ERC	Entidade Relacionamento Complexo
GOMS	*Goals,Operations, Methods and Selection Rules*
IHC	Interação Ser Humano-Computador
ISO	*International Standards Organizations*
LEMD	Linguagem de Especificação da Mensagem do *Designer*
PL/SQL	*Procedural Language extensions to SQL*
SGDB	Sistema Gerenciador de Base de Dados
SQL	*Structured Query Language*
VIQUEN	*Visual Query Environment*

RESUMO

O presente trabalho consiste em desenvolver um projeto de interfaces para visualizar *triggers* no modelo ERC+, Entidade Relacionamento Complexo Estendido, utilizando conceitos da Interação Ser Humano-Computador, como a Engenharia Cognitiva e a Engenharia Semiótica.

No conceito de integração de esquemas de Banco de Dados, quanto mais informações a respeito das bases de dados a serem integradas estiverem disponíveis, mais o processo será facilitado. Atualmente, somente os aspectos estáticos são considerados para integração de bases de dados, deixando de lado os aspectos dinâmicos, em particular os *triggers*, que podem ser cruciais para identificar novas correspondências de dados ou até descartá-las.

A contribuição deste trabalho se dá, não somente para o campo da integração de esquemas, mas também como ferramenta para consultar e visualizar os *triggers* em bases de dados isoladas, facilitando sua manipulação e gerenciamento.

Palavras-chave: Integração de esquemas, bancos de dados visuais, *trigger*.

ABSTRACT

The present work consists in developing an interface project to visualize triggers in the model ERC+, Entity Relationship Complex Extend, using Human-Computer Interaction concepts, as Cognitive and Semiotic Engineering. In the concept of integration of schemes for databases, the more information on databases to be integrated is avaliable the more the process will be facilitated. Currently, only the static aspects, in particular the triggers which can be crucial to identify new correspondences of data or even discard them. In the environments of schema integration which envolve manu representations and cognitively complex human decisions the quality of the interface has a crucial importance. This work contributes, not only in the field of schema integration, but also as a tool to consult and visualize the triggers in isolated databases, facilitating its manipulation and management.

Keywords: Schema integration, visual databases, trigger

1 INTRODUÇÃO

A evolução do gerenciamento de banco de dados constitui no componente central num ambiente de tecnologia da informação. De uma forma geral, os bancos de dados são heterogêneos, além de estarem muitas vezes em locais diferentes, dificultando a sua integração. O objetivo geral da integração de esquemas é facilitar o acesso a um recurso organizacional de informação.

Por outro lado, muitos sistemas computacionais que realizam suas tarefas de forma satisfatória têm sua comunicação com o usuário resolvida de forma inadequada. A frustração e a ansiedade fazem parte da vida de muitos usuários destes sistemas de informação informatizados, que ficam intimidados por usarem os sistemas de computação face às interfaces complexas, comandos de linguagem inconsistentes, seqüências de operações confusas, instruções incompletas e procedimentos trabalhosos de recuperação de erro (Carol85).

Pesquisadores têm mostrado que um bom projeto de interface usuário-sistema pode fazer uma diferença substancial no tempo de aprendizado, desempenho, minimização das taxas de erros e satisfação dos usuários. Cientistas da Informação e da Computação, entre outros, têm testado alternativas de projeto que visam melhorar o desempenho humano. Gerentes comerciais reconhecem que sistemas mais fáceis de usar são mais competitivos. Programadores e gerentes de desenvolvimento estão cada vez mais atentos e conscientes de que o projeto deve garantir uma interface de alto nível para o usuário. Enfim, os aspectos humanos deixam de ser pano de fundo para se tornar a estrutura em cima da qual os sistemas são construídos (Shneiderman92).

A solução desses problemas não se dará apenas por meio da argumentação sobre o que são interfaces mais "amigáveis" e naturais, mas, sim, a partir de uma abordagem disciplinada, iterativa e empírica do estudo do desempenho humano no

uso de sistemas interativos. Com o progresso no atendimento às necessidades dos usuários, estes se sentirão mais seguros e competentes ao usarem o sistema, e poderão facilmente corrigir erros e acompanhar suas tarefas, passando então uma mensagem de qualidade do sistema para outras pessoas (Shneiderman92).

Nos ambientes de integração de esquemas, que envolvem muitas representações e decisões humanas cognitivamente complexas, a qualidade da interface adquire importância crucial.

1.1 PROBLEMA

O processo de integração de bancos de dados pode ser definido como uma atividade que, a partir de uma entrada de um conjunto de esquemas de banco de dados, produz como saída uma descrição unificada dos esquemas iniciais, chamada de "esquema integrado" e a informação de mapeamento entre o esquema integrado e os esquemas iniciais. Neste contexto, Scopim (Scopim03) desenvolveu uma ferramenta de apoio para a integração de esquemas de banco de dados, cujo objetivo é auxiliar e facilitar a tarefa de geração de esquema integrado e da informação de mapeamento existente entre este esquema e os esquemas das bases de dados iniciais.

Na ferramenta proposta por Scopim para integração de bases de dados que foi tomada como ponto de partida para o desenvolvimento deste trabalho, os *triggers* das bases de dados iniciais não foram considerados. Adicionalmente, a interface usuário-sistema não se constituiu no foco principal. O presente trabalho se propõe a desenvolver um ambiente de interface-usuário para a identificação de *triggers*, chamado MaTrigs, com a finalidade de auxiliar na integração de esquemas na fase de identificação de correspondências semânticas.

1.2 MODELO ERC+, INTEGRAÇÃO DE ESQUEMAS E AMBIENTE J-SCHEMAS INTEGRATOR

O ERC+ – Entidade Relacionamento Complexo Estendido – é um modelo de dados que expressa de uma forma mais completa a relação entre objetos com o mesmo significado no mundo real e sua representação no banco de dados. Este modelo, que é uma extensão do modelo Entidade Relacionamento (ER) proposto por Chen (Chen90), possui diversas características que contribuem no processo de integração de bancos de dados.

O processo de integração de esquemas de bases de dados é uma atividade que, através da entrada de um conjunto de esquemas de bancos de dados, já representados em um mesmo modelo de dados, produz como saídas uma descrição unificada dos esquemas iniciais, chamada de esquema integrado, e a informação de mapeamento entre o esquema integrado e os esquemas iniciais.

A ferramenta J-Schemas Integrator desenvolvida por Scopim (Scopim03) tem como objetivo servir de apoio para visualização e participação no processo de integração de esquemas de bancos de dados heterogêneos. Possibilita ao usuário integrador visualizar os esquemas de bancos de dados, identificar os conflitos existentes entre os esquemas a serem integrados, gerar o esquema integrado a partir dos esquemas informados e dos conflitos apontados pelo usuário na ferramenta, e gerar a informação de mapeamento entre o esquema gerado e os esquemas iniciais num formato padrão.

1.3 JUSTIFICATIVA

A tecnologia de informação tem possibilitado o acesso à informação de maneira rápida e flexível. A disponibilidade de dados e o acesso à informação com resultados satisfatórios têm se tornado fatores críticos para ganho de vantagem competitiva. Desta forma, vem à tona a necessidade da integração de esquemas de base de dados, a fim de fornecer compatibilidade e acesso transparente aos

recursos da informação sem envolver investimentos substanciais no replanejamento do sistema.

As ferramentas existentes para a documentação de banco de dados abrangem somente os aspectos estáticos, como diagramas de classes e diagramas entidade-relacionamento. Aspectos dinâmicos envolvendo a manipulação de dados de forma automática, tais como os *triggers*[1], não são documentados de forma satisfatória para a visualização da dinâmica da base de dados. Normalmente é utilizada apenas uma listagem destas rotinas. A apresentação dos *triggers* se justifica pois eles podem tanto auxiliar na identificação de uma correspondência quanto levar à sua eliminação.

Não foram encontrados trabalhos que mostrassem especificamente a preocupação em identificar *triggers* num modelo entidade-relacionamento. O que existe são pesquisas voltadas à utilização de recursos visuais e de interfaces gráficas e multimodais, para facilitar a percepção e a interpretação dos aspectos estáticos de modelos entidade-relacionamento.

Dentre estas contribuições podem ser citados trabalhos como de Catarci et al (Catarci96) pesquisa cujo objetivo é descrever um método para acessar uma base de dados com representações visuais diferentes. A ferramenta SUPER (Dennebouy95) é um ambiente que permite a edição do modelo e formulação de consultas utilizando o modelo ERC+ para exibir o esquema completo da base e para interagir com o usuário, e tem a preocupação com o nível de conhecimento do usuário. O problema desta ferramenta é que não há acesso a SGDB's relacionais, nem para traduzir esquemas relacionais em esquemas semânticos, nem para a extração de dados, ou seja, não há acesso aos dados da base de dados original. Gueiber desenvolveu na UFPR uma ferramenta de consulta visual chamada VIQUEN (Gueiber01) que consiste num método para extrair um esquema conceitual

[1] *Triggers* são programados diretamente na base de dados visando manter a integridade referencial, a atualização de tabelas ou, ainda, a consistência mecânica dos dados.

a partir de uma base de dados relacional, fazendo o mapeamento dos operadores algébricos do modelo conceitual para SQL e utilizando a representação gráfica do modelo conceitual em todas as fases da formulação da consulta.

A simples consideração dos *triggers* ainda não é suficiente para garantir a qualidade da interface de auxílio à identificação de correspondências. É necessário também utilizar conceitos e metodologias de Interação Ser Humano-Computador (IHC) no desenvolvimento da interface-usuário da ferramenta para integração de esquemas, principalmente na fase de identificação de correspondências, onde somados aos aspectos estáticos da documentação da base de dados, têm-se os aspectos dinâmicos – dentre os quais *triggers*, aumentando a quantidade e a complexidade da informação de insumo ao processo de análise humana.

1.4 OBJETIVO

O objetivo do presente trabalho consiste em projetar um ambiente de interface-usuário, chamado MaTrigs, para identificação e mapeamento de *triggers* de uma base de dados, utilizando como modelo de dados o ERC+ – modelo de dados Entidade Relacionamento Complexo Estendido (Spaccapietra95), a fim de auxiliar no processo de integração de esquemas, tendo seu foco voltado principalmente à fase de identificação de correspondências.

1.5 ORGANIZAÇÃO DA DISSERTAÇÃO

A presente dissertação está organizada em seis capítulos, quatro apêndices e um anexo.

O capítulo 2 descreve resumidamente o modelo Entidade Relacionamento Complexo Estendido - ERC+, abordando a álgebra do modelo e sua representação visual, que servirá de base para o mapeamento dos *triggers* da base de dados.

No capítulo 3 são apresentados os conceitos referentes à integração de esquemas e são descritas suas diversas fases. Também é apresentada a ferramenta proposta por Scopim, a J-Schemas Integrator. Ainda são caracterizados a linguagem SQL e os *triggers*, e discutidas sua importância para a integração de esquemas.

No capítulo 4 é feita uma explanação sobre abordagens Cognitivas, como a Engenharia Cognitiva e o modelo *Goals, Operations, Methods and Selection Rules* - GOMS, Engenharia Semiótica, focando a Linguagem de Especificação da Mensagem do *Designer* – LEMD, e a relação destes abordagens de Interação Ser Humano-Computador com o ambiente proposto, mostrando em que medida o trabalho proposto irá atender aos requisitos determinados por estas duas vertentes. Neste capítulo também é descrito um processo de *design* de interfaces.

O capítulo 5 descreve o módulo de visualização de *triggers*. Numa primeira fase, é descrito o processo de *design* de interfaces na ferramenta MaTrigs. Em seguida é demonstrado o funcionamento do sistema de mapeamento de *triggers*. E por último, é apresentada uma sessão de utilização da ferramenta proposta.

O capítulo 6 discute as contribuições de trabalho e as possibilidades de pesquisa futura.

No Apêndice I tem-se a especificação GOMS da interface proposta. No Apêndice II a especificação do programa em Linguagem de Especificação da Mensagem do *Designer* (LEMD). No Apêndice III, é mostrada a estrutura hierárquica de tarefas requeridas pelo usuário. E finalmente, no Apêndice IV, são apresentadas as telas do sistema.

O Anexo I apresenta os diagramas de sintaxe de *triggers* em Oracle.

2 O MODELO ERC+

O modelo ERC+ - Entidade Relacionamento Complexo Estendido - é um modelo projetado para dar suporte à modelagem de objetos complexos. O desenvolvimento de ERC+ como um novo modelo de dados iniciou-se em 1983, como parte de um projeto num sistema de base de dados heterogêneos distribuídos (Spaccapietra83). Utiliza diagramas Entidade Relacionamento (ER) para representar o esquema conceitual, que é independente do gerenciador de bases de dados utilizado, e é uma extensão do Modelo Entidade Relacionamento.

O Modelo ER foi originalmente proposto por Chen (Chen90) como uma forma de facilitar o mapeamento de bases de dados, e foi suficiente para a representação da maioria das aplicações administrativas convencionais. Porém, o crescimento da aplicação da Informática às diversas áreas do conhecimento trouxe consigo a necessidade de melhoria do poder semântico, visando a representação de situações mais complexas. O modelo ERC+ possui as mesmas características do modelo ER agregando os conceitos de especialização, generalização e composição, introduzidos por Spaccapietra et al. (Spaccapietra95).

Seu objetivo é representar com mais fidelidade objetos do mundo real. Sua utilização se baseia nos conceitos de tipo **entidade** para descrever objetos e tipo **relacionamento** para representar as relações entre os objetos. Uma identificação de objeto – **oid** – é associado a cada entidade. Um objeto do mundo real, independentemente de sua complexidade, pode ser modelado como uma simples ocorrência do tipo **entidade**. Tipo **entidade** pode ser composto de qualquer número de atributos, os quais podem, por sua vez consistir de outros atributos. A estrutura de um tipo **entidade** pode ser considerado como uma árvore cuja raiz e nós internos são respectivamente representados pelo tipo **entidade** e seus atributos, como mostrado na Figura 2.1 (Parent92).

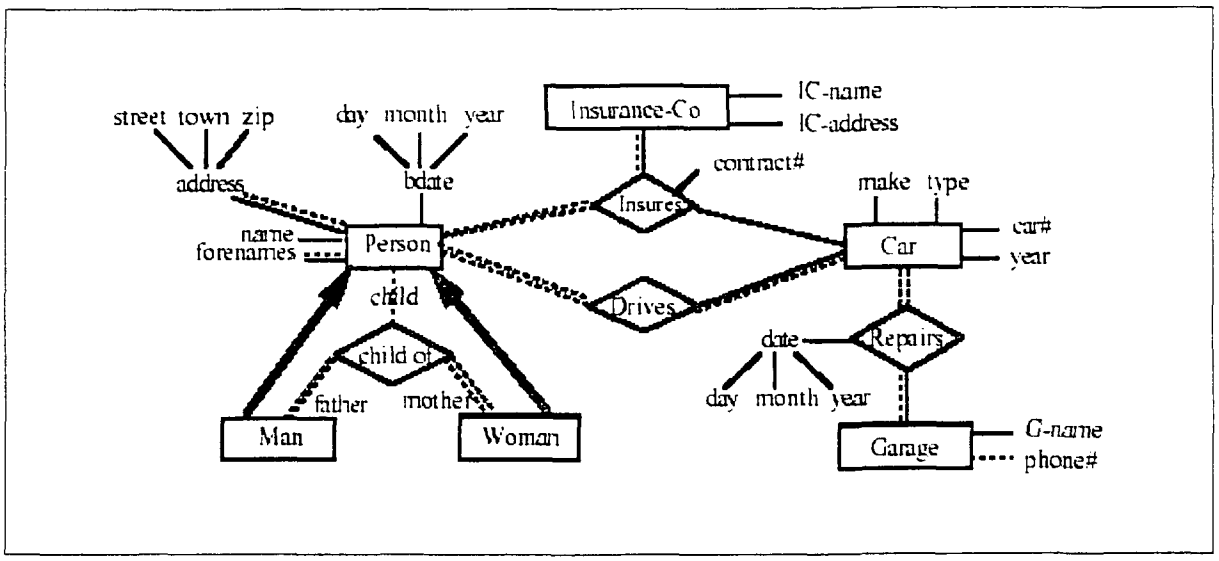


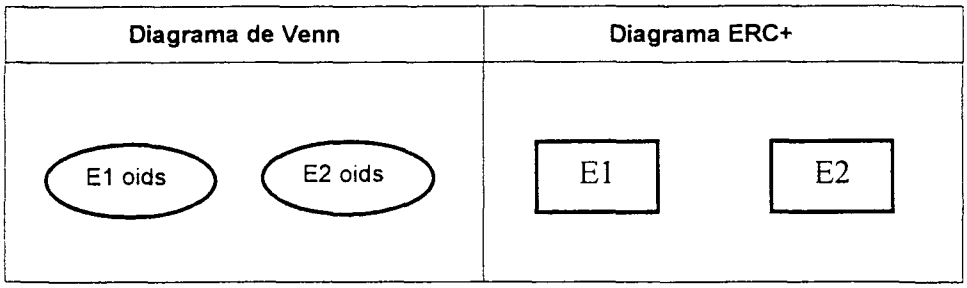
FIGURA 2.1: EXEMPLO DE UM DIAGRAMA ERC+ (PARENT92)

ERC+ admite o conceito de tipo **relacionamento**. Vários (dois ou mais) tipos **entidade** podem ser ligados por um tipo **relacionamento**. Como tipo **entidade**, um tipo **relacionamento** pode ter atributos. Tipos **relacionamento** podem ser cíclicos, ou seja, um tipo **relacionamento** pode ligar duas vezes – ou várias vezes – o mesmo tipo **entidade**, cada vez com um papel diferente.

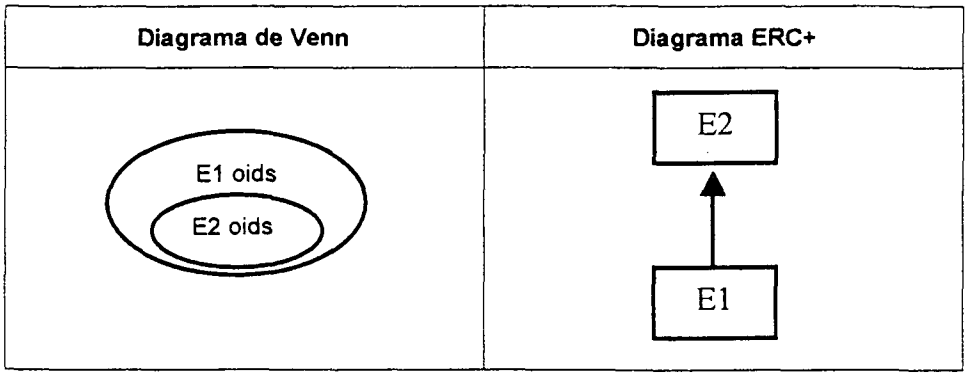
O modelo ERC+ permite ainda dois outros tipos de *links* entre entidades: *link is-a* (é-um) e *link may-be-a* (pode-ser-um) (Spaccapietra95). O *link is-a,* também chamado de *link* de generalização, especifica que a população do tipo de entidade ES é uma subclasse (especialização) de um outro tipo de entidade EG (generalização) representado no diagrama ERC+ por uma flecha: ES → EG. Todo oid contido em ES é também um oid de EG. Ciclos não são permitidos usando este tipo de *link*. O *link may-be-a* ou *link* de conjunção, entre dois tipos de entidades E1 e E2 é usado para especificar que algumas entidades de E1 descrevem o que os mesmos objetos do mundo real nas entidades E2 fazem, e vice-versa, isto é, a população de E1 e E2 podem compartilhar oids. O *link may-be-a* é geralmente representado por uma linha tracejada entre E1 e E2: E1- - - E2.

Os links *is-a* e *may-be-a* permitem modelar as três seguintes situações que podem surgir entre o conjunto de oids de dois tipos **entidade** E1 e E2, como mostra a Figura 2.2.

a) o conjunto de oids são disjuntos: não há ligação entre E1 e E2



b) o conjunto de oids de um tipo entidade E2 está incluído no outro conjunto E1: isto é um *link is-a*.



c) E1 e E2 compartilham alguns oids comuns: isto é um *link may-be-a*.

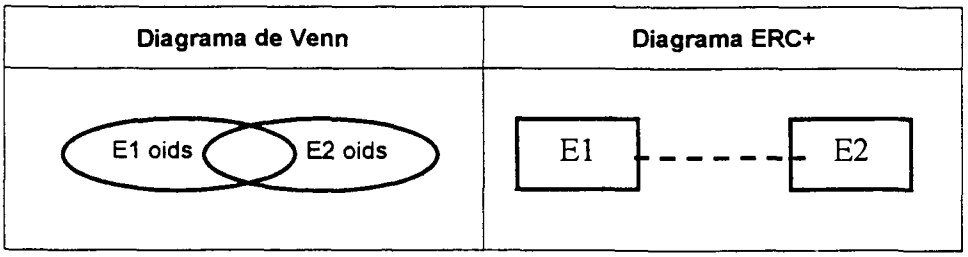


FIGURA 2.2: COMPARAÇÃO DE NOTAÇÃO ENTRE DIAGRAMA DE VENN E DIAGRAMA ERC+ (SPACCAPIETRA95)

Tanto o *link* **is-a** quanto o **may-be-a** são usados para relacionar dois tipo **entidade** que tenham alguma representação estendida do mesmo objeto do mundo real, cada *link* com um diferente ponto de visão.

2.1 A ÁLGEBRA DO MODELO

A álgebra ERC+ foi desenvolvida como linguagem para manipulação de dados do modelo ERC+, dando suporte ao modelo, pois outras linguagens não contemplam operadores para manipular objetos complexos.

Em Spaccapietra et al. (Spaccapietra95) são apresentadas as definições formais do modelo ERC+. O conceito de **domínio** define um conjunto de todos os valores possíveis de um atributo, uma entidade ou um relacionamento. Valores podem ser tanto atômicos, como "Maria" ou "1991", ou complexo, isto é, composto de outros valores. Um valor complexo é um conjunto de pares <nomeatributo, v> onde **v** é tanto um valor ou um conjunto de valores. O conceito de **estruturas** é necessário para descrever objetos complexos, pois convenciona as características de um atributo (nome e cardinalidade, composição e domínio) independente de sua associação com o objeto descrito. O tipo **entidade** é uma estrutura que contém o nome da entidade, seu esquema (conjunto de estruturas de seus atributos, seus tipos de entidades genéricas, o conjunto de tipo **entidade** com o qual está em conjunção), suas populações (conjunto de ocorrências), identificadores de objetos e valores. O tipo **relacionamento** é definido pelo nome do relacionamento, conjunto de tipo **entidade** ao qual está conectado (com a descrição das características dos *links* – papel e cardinalidade), conjunto de estruturas de seus atributos (que constituem seu esquema) e o conjunto de suas ocorrências. O **atributo** é definido pelo objeto ao qual está relacionado, sua estrutura e seus valores para cada ocorrência do seu objeto.

A álgebra do modelo ERC+ inclui diversos operadores, entre os quais podemos citar o **selection**, que faz a seleção de objetos sob um dado predicado; **projection**, que faz a projeção de objetos sob um conjunto de seus atributos; **reduction**, que elimina dentre os valores de um atributo aqueles que não satisfazem um dado predicado; **relationship-join** *(r-join)*, que constrói um novo objeto complexo dentro de uma estrutura hierárquica, a partir de uma rede de tipos objetos; o operador **identity-join** *(i-join)*, que permite a junção de objetos participantes em uma ligação multi-instanciação; **renaming**, que possibilita a trocar o nome de objetos; e **simplification**, para eliminar complexidades desnecessárias, que podem ocorrer como resultado de outros operadores (Spaccapietra95).

2.2 REPRESENTAÇÃO VISUAL

No modelo ERC+, a entidade é representada por um retângulo com o nome da entidade no seu interior. Da mesma forma, o relacionamento é representado por um losango com sua identificação no seu interior. Setas representam generalização entre tipos entidade.

A cardinalidade é representada da seguinte maneira: uma linha simples e contínua identifica uma relação obrigatória e monovalorada (1:1); uma linha simples e tracejada representa uma relação opcional e monovalorada (0:1); uma linha dupla e tracejada representa uma relação opcional e multivalorada (0:n); e uma linha tracejada juntamente com uma linha contínua representa uma relação obrigatória e multivalorada (1:n) (Spaccapietra92). A representação visual da entidade, relacionamento e cardinalidade pode ser vista na Figura 2.3.

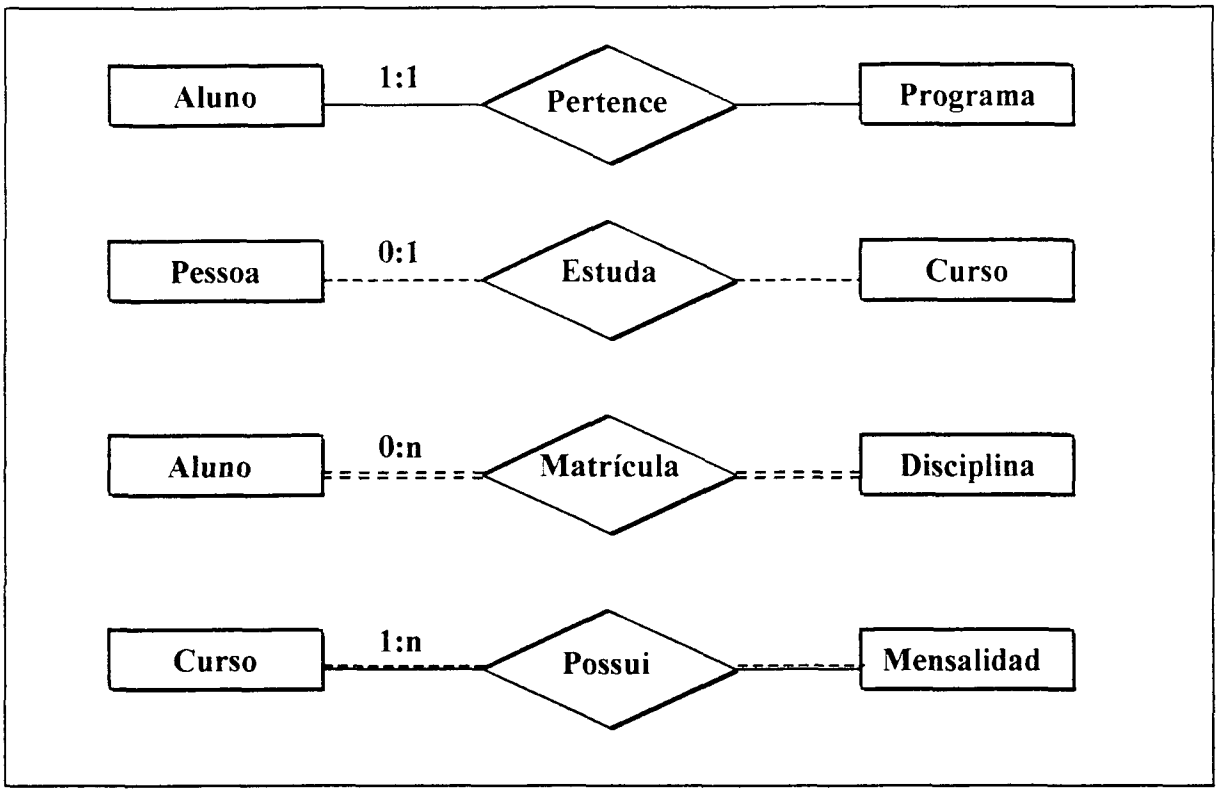


FIGURA 2.3: REPRESENTAÇÃO VISUAL DO ERC+, ADAPTADO DE SPACCAPIETRA ET. AL (SPACCAPIETRA92)

3 A INTEGRAÇÃO DE ESQUEMAS, A FERRAMENTA J-SCHEMAS INTEGRATOR, A LINGUAGEM SQL E OS *TRIGGERS*

A integração da base de dados é o processo que tem como entrada um conjunto de bases de dados e produz como saída uma única descrição unificada dos esquemas de entradas – o esquema integrado – e o mapeamento de informações associadas que permitem acesso integrado aos dados existentes através do esquema integrado. A integração da base de dados é usada também no processo de reengenharia de sistemas legados. Além da caracterização do modelo de dados utilizado para o desenvolvimento de uma ferramenta de integração de esquemas, apresentado no Capítulo 2, faz-se necessária a caracterização do ambiente, incluindo a linguagem de banco de dados a ser utilizada, já que o objetivo proposto é identificar e mapear os *triggers* de uma base de dados visando um módulo de interface-usuário na fase de identificação de correspondências semânticas.

3.1 INTEGRAÇÃO DE ESQUEMAS

O objetivo geral da integração de esquemas consiste em integrar uma organização com diferentes propósitos ou sistemas de base de dados e percepções do usuário que facilitam, deste modo, o acesso global a um recurso organizacional integrado da informação.

Segundo Parent et al. (Parent98), para desenvolver um esquema integrado têm-se as seguintes fases (Figura 3.1):

- **Pré-integração** onde os esquemas de entrada são transformados para se tornarem mais homogêneos – sintática e semanticamente;

- **Identificação de correspondências** dirigida à identificação e à descrição de relacionamentos inter-esquemas;
- **Integração**, a etapa final, que resolve conflitos inter-esquemas e unifica itens correspondentes gerando um esquema integrado.

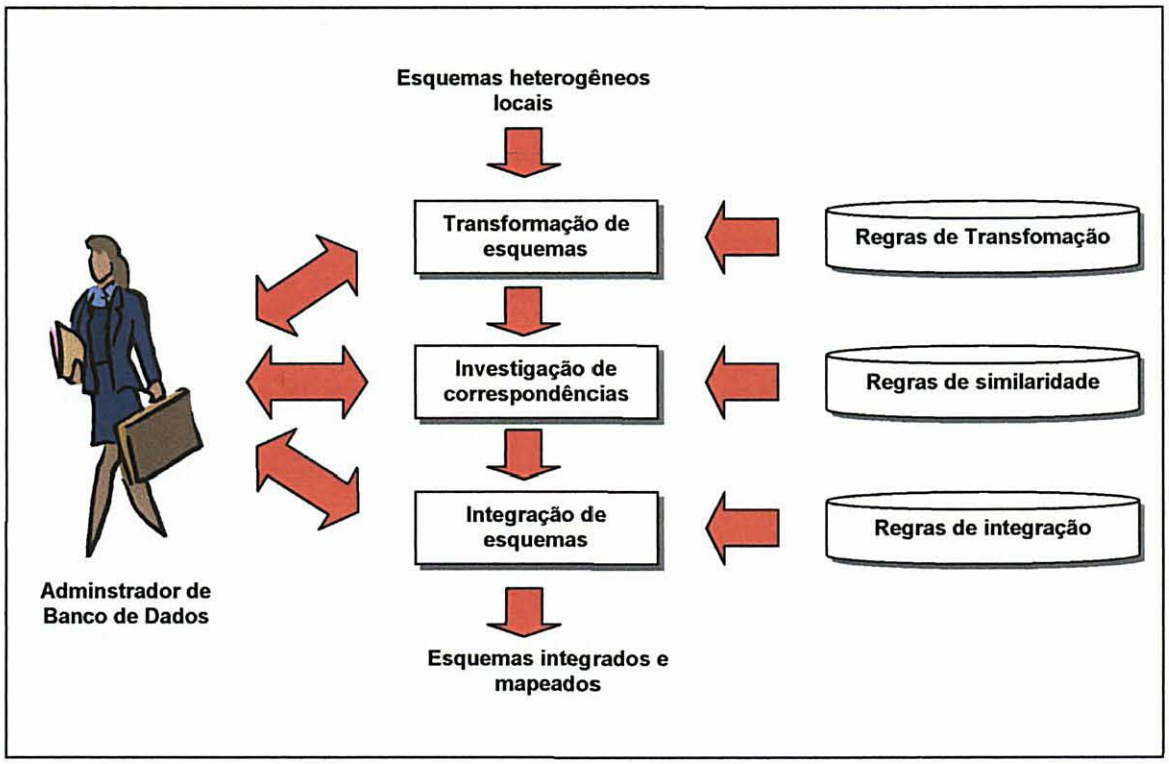


FIGURA 3.1 – PROCESSO GLOBAL DE INTEGRAÇÃO, ADAPTADO DE SPACCAPIETRA ET AL. (PARENT98).

Na fase de pré-integração, esquemas que correspondem às bases de dados individuais a serem integradas devem ser traduzidos para um esquema usando um modelo de dados comum. O modelo ERC+ é indicado por reunir requisitos de aplicações de banco de dados combinando características da semântica tradicional do modelo de dados com capacidades de orientação a objetos, tais como orientação estrutural dos objetos, herança e identidade dos objetos, como descrito no Capítulo 2.

Na fase de identificação das correspondências entre os objetos dos esquemas, o objetivo é identificar objetos que possam estar semanticamente relacionados, isto é, entidades, atributos e relações que estejam em conflito, e classificar os relacionamentos entre estes objetos. Isto é feito examinando-se a semântica dos objetos nos diferentes bancos de dados e identificando os relacionamentos entre eles. A principal meta desta fase é gerar um conjunto de relacionamentos confiáveis entre os objetos dos bancos de dados. É importante que esses relacionamentos sejam precisos, pois eles serão usados como entrada na fase de geração do esquema integrado.

O objetivo deste trabalho é desenvolver uma interface para acrescentar ao modelo ERC+ aspectos dinâmicos, mais especificamente *triggers*, a fim de obter um mapeamento completo dos dados para que estes possam ser integrados, já que um *trigger* pode tanto ajudar a identificar quanto levar à eliminação de correspondências.

Na fase de integração ocorre o processo de geração de um ou mais esquemas integrados a partir dos esquemas existentes. Estes representam a semântica dos bancos de dados a serem integrados e são usados como entrada para o processo de integração. A saída do processo consiste em um ou mais esquemas integrados representando a semântica dos esquemas iniciais. Esses esquemas integrados poderão ser usados para receber consultas que são remetidas para os bancos de dados iniciais.

3.2 O AMBIENTE J-SCHEMAS INTEGRATOR

A ferramenta J-Schemas Integrator desenvolvida por Scopim (Scopim03) consiste de um repositório de esquemas e informações de transformações entre

esquemas. Este repositório armazena informações sobre os esquemas iniciais e os integrados e informações de mapeamento entre esses esquemas. As informações contidas neste repositório podem, num momento seguinte, serem utilizadas para realização de consultas em qualquer esquema pertencente a esse repositório.

Entre suas principais funcionalidades, a ferramenta possibilita a visualização em uma interface gráfica de esquemas de banco de dados pelo modelo ERC+, onde o usuário pode clicar/selecionar os objetos dos esquemas (tipo entidade, tipo relacionamento e atributos) e pode identificar o tipo de conflito existente entre esses objetos. Para cada tipo de conflito identificado entre os esquemas deve existir uma associação com os *templates* de integração que soluciona os conflitos existentes. Também foi implementada na ferramenta J-Schemas Integrator um módulo para importação de *templates* de integração[2], permitindo que à medida que novos *templates* forem criados, eles possam ser ligados na arquitetura da ferramenta e possam ser utilizados para solucionar algum tipo de conflito particular. Um *template* de integração recebe como entrada de seu processamento objetos que estão em conflitos nos esquemas iniciais; gera como saída de seu processamento o objeto integrado que deve ser gerado no esquema integrado e a informação de mapeamento entre este objeto integrado e os objetos nos esquemas iniciais. À medida que o usuário for clicando sobre os objetos dos esquemas iniciais e for categorizando os conflitos existentes entre eles, o usuário pode visualizar o esquema integrado sendo gerado e a informação de mapeamento existente entre os esquemas. Portanto, existem três frames de visualização na interface gráfica do editor, uma para cada um dos esquemas a serem integrados e uma para o esquema que está sendo gerado pelo processo de integração. Todos os três esquemas, esquemas iniciais e o esquema integrado são representados no modelo ERC+. As informações de mapeamento geradas entre o esquema integrado e os esquemas

[2] *Templates* de integração é uma implementação de uma regra de integração. Todo *template* de integração implementa uma determinada interface utilizada pela ferramenta, para solucionar determinados tipos de conflitos.

iniciais são armazenadas no repositório de esquemas e transformações. O esquema integrado gerado é armazenado no mesmo formato de arquivo dos esquemas de dados dos bancos de dados iniciais. Isto é necessário, pois este mesmo esquema integrado pode ser reutilizado para a integração com um outro esquema qualquer.

3.3 *STRUCTURED QUERY LANGUAGE*

Nas sub-seções seguintes são descritos conceitos da linguagem SQL (*Structured Query Language*) e de *triggers*. Também é apresentado o funcionamento de *triggers* no SGBD (Sistema Gerencial de Banco de Dados) *Oracle*.

3.3.1 A Linguagem

A linguagem SQL tem representado um padrão para linguagens de banco de dados relacionais. A versão original, chamada Sequel, foi desenvolvida pela IBM no início dos anos 70. Desde então, a linguagem Sequel evoluiu e seu nome foi mudado para SQL – *Structured Query Language*. Atualmente no mercado, inúmeros produtos dão suporte para a linguagem SQL. Em 1986, o *American National Standards Institute* – ANSI - e *a International Standards Organizations* - ISO – publicaram os padrões para a SQL, chamada SQL-86. A IBM publicou seus próprios padrões para a SQL, chamada SAA-SQL. Em 1989 foi publicada uma extensão para a SQL, na qual um sistema de banco de dados típico atualmente dá suporte ao menos aos recursos dispostos nesta linguagem, chamada SQL-89 (Silberschatz01).

A versão em uso do padrão ANSI/ISO SQL é o padrão SQL-92. Segundo Silberschatz et al (Silberschatz01), a linguagem SQL se apóia em diversos conceitos, entre os quais podemos citar:

- **Linguagem de definição de dados** (Data Definition Language – DDL): proporciona comandos para a definição de esquemas de relações, exclusão de relações, criação de índices e modificação nos esquemas de relações.
- **Linguagem interativa de manipulação de dados** (Data Manipulation Language – DML): abrange uma linguagem de consulta baseada tanto na álgebra relacional quanto no cálculo relacional de tuplas[3]. Engloba também comandos para inserção, exclusão e modificação de tuplas no banco de dados
- **Incorporação DML** (Embedded DML): a forma de comandos SQL incorporados foi projetada para aplicação em linguagens de programação de uso geral, como PL/I, Cobol, Pascal, Fortran e C.
- **Definição de visões[4]**: a SQL DLL possui comandos para definição de visões.
- **Autorização**: a SQL DLL engloba comandos para especificação de direitos de acesso a relações e visões.
- **Integridade**: a SQL DLL possui comandos para especificação de regras de integridade que os dados serão armazenados no banco de dados devem satisfazer. Atualizações que violarem as regras de integridade serão desprezadas.

[3] Uma tupla é um conjunto de pares (atributo-valor), que define uma linha da tabela, ou seja, uma ocorrência de uma entidade ou relacionamento.

[4] Visão é qualquer relação que não faça parte do modelo lógico, mas é visível para o usuário como uma relação virtual.

- **Controle de transações**: a SQL inclui comandos para a especificação de iniciação e finalização de transações. Algumas implementações também permitem explicitar bloqueios de dados para controle de concorrência.

A estrutura básica de uma expressão em SQL para consulta consiste de três cláusulas: **select, from** e **where.** A cláusula **select** é usada para relacionar os atributos desejados no resultado de uma consulta; **from** associa as relações que serão pesquisadas durante a evolução de uma expressão; e **where** consiste em um predicado envolvendo atributos da relação que aparece na cláusula **from**.

Para modificação de informações no banco de dados, a SQL consiste basicamente de três cláusulas: **delete, insert** e **update.** A cláusula **delete** é usada para remover tuplas inteiras; **insert** é usada para inserir tuplas ao banco de dados; e **update** é usada para modificar alguns valores das tuplas.

3.3.2 *Triggers*

Trigger é um bloco de comandos presente na linguagem SQL que é automaticamente executado quando um comando para modificação de base de dados - **Insert, Delete** ou **Update** – é executado em uma tabela. A principal aplicação de um *trigger* é a criação de consistências e restrições de acesso ao banco de dados, como rotinas de segurança (Silberschatz01).

Triggers não foram incluídos no padrão SQL-92, apesar de estarem disponíveis há bastante tempo na maioria dos Sistemas Gerenciadores de Banco de Dados (SGBDs) comerciais. Eles estão especificados no padrão emergente SQL3 com sintaxe bastante próxima à adotada por SGBDs como o Oracle, por exemplo.

Ramalho (Ramalho99) cita que entre outras aplicações de um *trigger,* podem ser destacadas a geração do conteúdo de uma tupla derivada de outras tuplas, geração de mecanismos de validação que envolvam pesquisas em múltiplas tabelas, geração de *logs* para registrar a utilização de uma tabela, atualização de outras tabelas em função de inclusão ou alteração da tabela atual.

Segundo Silberschatz et al. (Silberschatz01), duas exigências devem ser satisfeitas para a projeção de um mecanismo de *trigger*, a especificação das condições sob as quais o *trigger* deve ser executado, e a especificação das ações que serão tomadas quando o *trigger* for disparado.

Os *triggers* são armazenados no banco de dados de modo a mantê-los persistentes e acessíveis a todas as operações do banco de dados. Uma vez que o *trigger* é introduzido no banco de dados, cabe ao sistema de banco de dados a responsabilidade de executar a ação sempre que o evento especificado ocorrer e as condições correspondentes forem satisfeitas (Silberschatz01).

A visualização e/ou documentação dos *triggers* nos bancos de dados se dá por meio de listagens, ou seja, uma relação dos nomes dos *triggers* existentes e seus respectivos blocos de comandos.

A sintaxe para *triggers* em SQL3 é a seguinte:

<SQL3 *trigger*> ::=
CREATE *TRIGGER* <*trigger* name>
{BEFORE | AFTER | INSTEAD OF} <*trigger* event>
ON <table name>
[ORDER <order value>]
[REFERENCING <references>]
WHEN (<condition>)
<SQL procedure statements>
[FOR EACH {ROW | STATEMENT}]

<*trigger* event> ::=

INSERT | DELETE | UPDATE [OF <column names>]

<reference> ::=

OLD AS <old value tuple name> |

NEW AS <new value tuple name> |

OLD_TABLE AS <old value table name> |

NEW_TABLE AS <new value table name>

Onde:

CREATE *TRIGGER* – Gera um novo *trigger*

BEFORE | AFTER | INSTEAD OF – Determina se o *trigger* deve ser executado antes, depois ou ao invés da operação que o aciona;

ON – Define a tabela específica sobre a qual o *trigger* vai atuar;

ORDER – É utilizada para especificar prioridades;

REFERENCING – É utilizada para associar nomes com os valores anteriores e posteriores à modificação feita pela operação;

WHEN – Especifica a condição do *trigger;*

FOR EACH ROW – Especifica que um *trigger* é executado uma vez para cada tupla modificada;

FOR EACH STATEMENT – Especifica que um *trigger* é executado uma vez para todo um comando SQL.;

OLD / NEW - Quando linhas são atualizadas por uma declaração *trigger,* o *alias* **old** permite acessar as linhas antes da atualização da tabela e o *alias* **new** permite acessar as linhas com os valores atualizados; **old** e **new**

podem ser interpretados como *alias* para duas tabelas temporárias contendo as linhas citadas e só acessíveis durante a execução do *trigger*.

No processo de integração de esquemas a interpretação da realidade subjacente é complexa e demorada. Qualquer contribuição de auxílio ao processo se constitui em contribuição relevante. Neste sentido, o tratamento de triggers representa uma primeira fase, a visualização, na melhoria deste tipo de ferramenta.

3.3.2.1 *Triggers* no *Oracle8*

O *Oracle Server* ou *Oracle Data Server* é um sistema de gestão de bases de dados relacionais. Com a versão 8 do *Oracle Server*, foi também introduzido o conceito de *Object Relational*, ou seja, um sistema que permite emular a existência de objetos, por meio do modelo relacional, mais concretamente com o *Object View*. O sistema de gestão de base de dados *Oracle Server* passou a denominar-se *Object Relational DataBase Management Sytem* (ORDBMS). Com a introdução da máquina virtual JAVA dentro do ORDBMS em 1999, através do Oracle8i, os objetos foram definitivamente incorporados pelo *Oracle Server*.

Com o servidor Oracle, grandes volumes de dados podem ser armazenados e ficar acessíveis a um grande número de usuários. Ele adminte todos os tipos de dados, como numéricos, texto, imagens, sons e vídeos. Um *trigger* no Oracle possui três partes básicas: Evento, Restrição e Ação. Diagramas da sintaxe de *triggers* em Oracle podem ser consultados no anexo I.

O Evento pode ser um dos três comandos de modificação de dados (**Delete**, **Insert** e **Update**), um dos três comandos de definição de dados (**Create**, **Alter** e **Drop**) ou um estado particular do banco de dados (*ServerError, LogOn, LogOff, StarUp* e *ShutDown*). A Restrição especifica a condição do *trigger*, que é uma

condição SQL que deve ser satisfeita para que o *trigger* seja executado. Tanto em SQL3 quanto no Oracle, a restrição é precedida da palavra **when**. A Ação do *trigger* é o procedimento[5] que contém comandos SQL e código PL/SQL (*Procedural Language extensions to SQL*) para serem executados quando o evento associado ao *trigger* ocorrer, e sua condição for avaliada como verdadeira.

O Oracle, da mesma forma que o SQL3, possui diferentes tipos de *triggers*: **Row**[6] e **Statement**[7], **Before** e **After**, e **Instead Of**[8].

Os *triggers* **Row** e **Statement** especificam o número de vezes que a ação deve ser executada, quando um *trigger* é definido. O *trigger* **Row** é executado toda vez que a tabela é afetada por uma operação do *trigger*, e o *trigger* **Statemet** é executado uma única vez para uma operação do *trigger*. Quando um *trigger* é definido, ele pode especificar o momento em que será executado. **Before** leva à execução da ação do *trigger* antes do evento do *trigger* a ser executado, e **After** leva à execução da ação do *trigger* depois da execução do evento do *trigger*. **Instead Of** proporciona um mecanismo transparente para modificar visões que não podem ser modificadas diretamente por meio de comandos de manipulação de dados (**Insert**, **Delete** e **Update**).

O Oracle possui um modelo de execução de *trigger* que assegura a seqüência de acionamento de múltiplos *triggers* e a checagem de integridade:

1. Execute todos os *triggers* do tipo **before statement** que se aplicam ao comando;
2. Para cada linha afetada pelo comando SQL:
    a. Execute todos os *triggers* **before row** que se aplicam ao comando;

---

[5] bloco em PL/SQL

[6] *For each row* é o comando equivalente em SQL3

[7] *For each statement* é o comando equivalente em SQL3

[8] *Before / After / Instead of* são os mesmos comandos tanto para o Oracle quanto para o SQL3

b. Execute um *lock,* modifique a linha, e cheque restrições de integridade;

c. Execute todos os *triggers* **after row** que se aplicam ao comando;

3. Cheque restrições de integridade a nível de comando;
4. Execute todos os *triggers* **after statement** que se aplicam ao comando.

No modelo de execução há mecanismos para verificar se a execução de comandos SQL foram bem formulados. Se uma execução é lançada dentro de um *trigger* e não é explicitamente tratada, todas as ações executadas como resultado do comando SQL original, incluindo todas as ações já executadas, são desfeitas.

É possível criar múltiplos *triggers* do mesmo tipo, que sejam acionados para o mesmo evento e definidos na mesma tabela. A ordem em que o Oracle ativa esses *triggers* é indeterminada. Logo, se uma aplicação requer que um *trigger* seja acionado necessariamente antes de outro, os dois *triggers* devem ser combinados em um único *trigger,* cuja ação execute as ações dos *triggers* originais na ordem desejada.

4 INTERAÇÃO SER HUMANO-COMPUTADOR

Para os sistemas de hoje, a comunicação com os usuários se tornou tão importante quanto a própria Computação. É neste contexto que surge a Interação Ser Humano-Computador (IHC), uma área de pesquisa cujo objetivo é estudar e melhorar as formas de as pessoas interagirem com os computadores e se manterem atualizados nos avanços tecnológicos, assegurando que estes sejam aproveitados para maximizar o benefício humano. O estudo da IHC é um campo destinado a responder à questão de como melhorar a ainda árdua interação usuário-sistema. Moran (Moran81) definiu a interface-usuário como "aqueles aspectos do sistema com os quais o usuário entra em contato físico, perceptiva e conceitualmente", caracterizando a interface de usuário como tendo um componente físico, onde o usuário percebe e manipula, e outro conceitual, onde o usuário interpreta, processa e raciocina, chamado de "modelo conceitual do usuário".

A IHC inclui o estudo de interfaces de hardware e software, fatores humanos, estudos empíricos, metodologias, técnicas e ferramentas. O objetivo dos trabalhos em IHC tem sido fornecer ao usuário "explicações e previsões para fenômenos de interação usuário-sistema e resultados práticos para o design da interface de usuário" (ACM92), provendo o usuário com o mais alto nível de usabilidade. Atualmente, novos fatores foram inseridos no uso de computadores, como preocupação com treinamento, práticas de trabalho, aspectos de gerenciamento e organizacionais.

Dentre as teorias que dão sustento à IHC podem ser citadas, entre outras, a Engenharia Cognitiva e a Engenharia Semiótica, que a complementa. Estes alicerces conceituais são apresentados a seguir.

4.1 ABORDAGENS COGNITIVAS

Nas sub-seções seguintes são descritos conceitos de duas abordagens cognitivas: a Engenharia Cognitiva e o modelo GOMS – *Goals, Operations, Methods and Selection Rules.*

4.1.1 Engenharia Cognitiva

A Engenharia Cognitiva (Norman&Draper86) consiste na aplicação de resultados das Ciências Cognitivas ao projeto e à construção de máquinas. Trabalha com aspectos psicológicos e físicos no projeto de interfaces homem-máquina que maximizam a interação do homem com o computador, tornando esta relação uma atividade "agradável" e produtiva.

Os objetivos da Engenharia Cognitiva são determinar quais são os fenômenos críticos envolvidos na interação homem-máquina, entender os princípios fundamentais por trás da ação humana e desempenhos que são relevantes para o desenvolvimento dos princípios da Engenharia de Projeto, projetar sistemas visando não somente a eficiência, a facilidade e a eficácia, mas também que sejam agradáveis de usar e eventualmente divertidos (Norman&Draper86).

O principal objetivo da cognição associada à IHC tem sido compreender e representar a forma como as pessoas interagem com computadores em termos de como o conhecimento é transmitido entre os dois. Modelos cognitivos, que descrevem os processos e estruturas mentais, como recordação, interpretação, planejamento e aprendizado, podem indicar para os projetistas quais as propriedades que os modelos de interação devem ter de maneira que a interação possa ser desempenhada mais facilmente pelos usuários (Leite98). Como o foco da abordagem cognitiva é o usuário, o projeto feito com base nela é chamado *Design*

Centrado no Usuário - *User-Centered Design* (Norman&Draper86), que é a base da Engenharia Cognitiva, sendo Norman seu precursor e principal pesquisador.

Segundo Norman, o projetista do sistema deve proporcionar ao projeto um bom modelo do sistema. O usuário constrói seus modelos mentais através da experiência, treinamento ou instrução na utilização de sistemas, ou seja, o modelo mental do usuário é formado pela interpretação das ações percebidas por ele e de sua estrutura visível. O modelo do sistema é baseado nos perfis de usuários e de suas tarefas desempenhadas. A implementação do modelo do sistema é a imagem do sistema. Se bem projetado, o modelo mental deve antecipar o resultado das ações e avaliar se os efeitos são de fato os esperados. A Figura 4.1 mostra a interação entre os modelos da Engenharia Cognitiva.

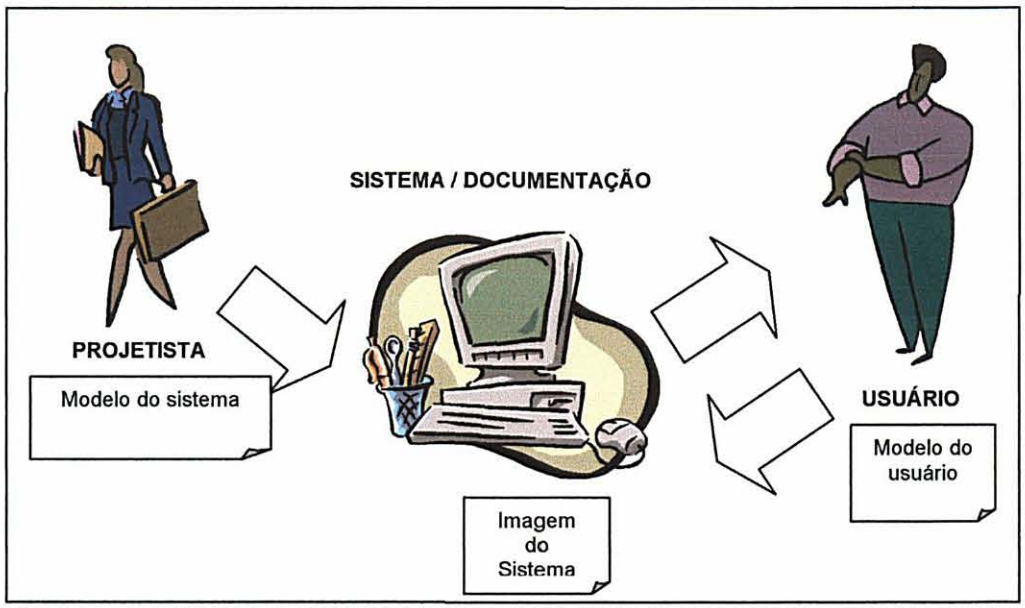


FIGURA 4.1 – MODELO ENVOLVIDOS NA ENGENHARIA COGNITIVA, ADAPTADO DE NORMAN (NORMAN&DRAPER86)

Na Engenharia Cognitiva, o objetivo do projetista ao desenvolver um modelo do sistema, é permitir ao usuário desenvolver seu modelo mental consistente com o que foi projetado e implementado. Segundo Norman, o projetista precisa entender o processo pelo qual o usuário interage com a interface do sistema. Para tanto,

propõe uma aproximação da teoria da ação, definindo que a interação usuário-sistema é desempenhada em dois "golfos" a serem atravessados entre o objetivo que o usuário deseja atingir e o sistema projetado: golfo de execução e golfo de avaliação (Norman86).

O golfo de execução representa a dificuldade do usuário em executar as ações do sistema e este atende às suas necessidades. Envolve as etapas de formulação da meta de acordo com o objetivo desejado e as opções de funções disponíveis no sistema, especificação da seqüência de ações para que possa atingir sua meta, e atividade física de execução, que é a concretização do que foi mentalizado pelas duas fases anteriores através de uma ação física (figura 4.2).

O golfo de avaliação se inicia após a execução pelo sistema da ação definida pelo usuário e refere-se à facilidade de avaliação de até onde o sistema cumpriu com o objetivo inicial do usuário. Envolve as etapas de percepção do usuário do novo estado em que o sistema se encontra após a execução, interpretação deste novo estado e avaliação da resposta do sistema de acordo com a meta estabelecida inicialmente pelo usuário (figura 4.2).

Além de ter cunhado os conceitos de ***golfos de execução*** e ***de avaliação***, que denotam, respectivamente, os esforços necessários para o enunciado e a interpretação das saídas, Norman (Hutchins86) responde ainda pela caracterização das **Distâncias Semânticas** e **Articulatória**, que se referem às discrepâncias de conteúdo e forma entre a maneira como o usuário enxerga o problema em questão e a ótica do projetista, que a interface revela. A Engenharia Cognitiva fundamenta, do ponto de vista teórico, as diretrizes experimentalmente levantadas.

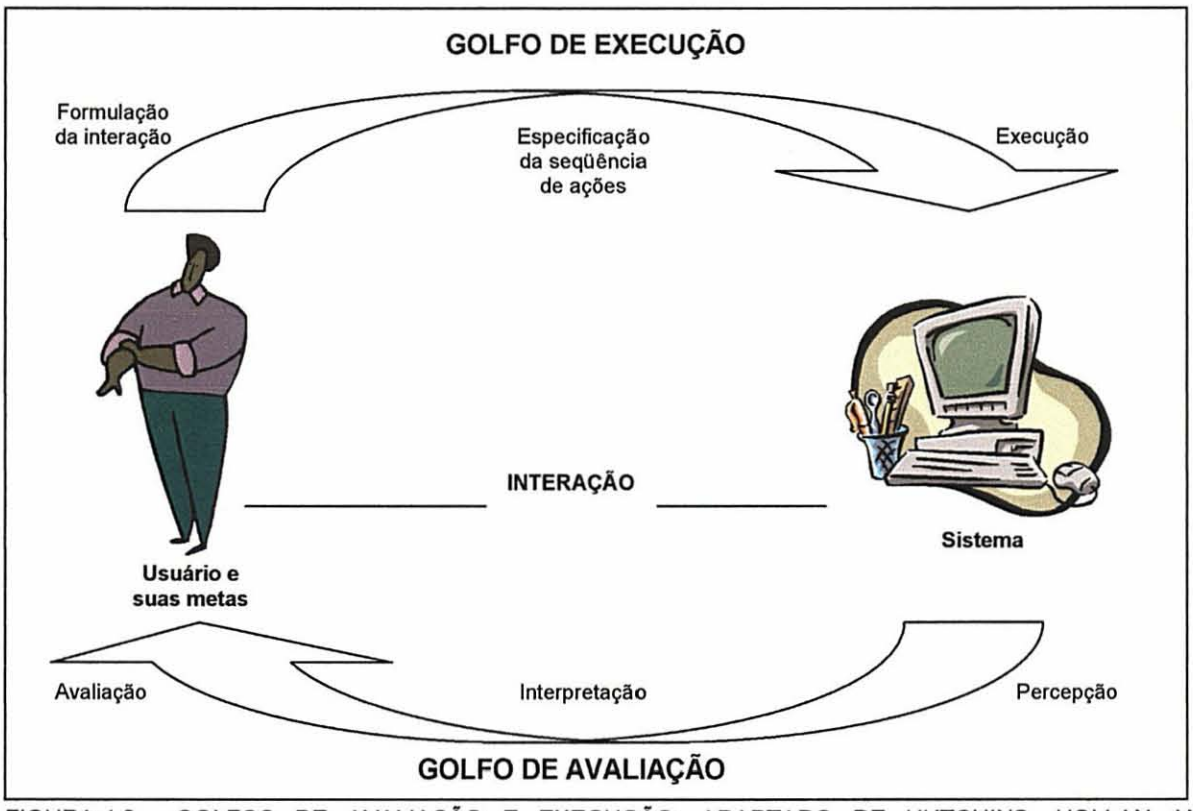


FIGURA 4.2 – GOLFOS DE AVALIAÇÃO E EXECUÇÃO, ADAPTADO DE HUTCHINS, HOLLAN AND NORMAN (HUTCHINS86).

4.1.2 Modelo GOMS - *Goals, Operations, Methods and Selection Rules*

Segundo Card et al (Card83), o modelo GOMS é uma teoria das habilidades cognitivas envolvidas nas tarefas homem-computador. Está baseada em uma estrutura de processamento de informações que assume um número de diferentes estágios ou tipos de memória, como memória sensorial ou de trabalho, com processamento perceptivo, motor e cognitivo separado.

De acordo com o modelo GOMS, a estrutura consiste de componentes em termos de objetivos (objetivos que os usuários tem em mente ao realizar uma tarefa, podendo estes serem divididos em sub-objetivos), operadores (ações pertencentes a um repertório de habilidades do usuário), métodos (seqüências de sub-objetivos e operadores geralmente concluídos de forma automática para obtenção de

habilidades), e regras de seleção (para escolher entre diferentes possíveis métodos para alcançar um objetivo particular).

Para uma dada tarefa, uma estrutura particular GOMS pode ser constituída e usada para mostrar o tempo necessário para completar a tarefa. Além disso, o modelo pode ser usado para identificar os efeitos dos erros de desempenho da tarefa.

4.2 ENGENHARIA SEMIÓTICA

As abordagens semióticas fundamentam-se no estudo da Semiótica, que é a disciplina que estuda os signos, os sistemas semióticos e de comunicação, bem como os processos envolvidos na produção e interpretação de signos. A Engenharia Semiótica está fundamentada em diversas teorias semióticas, como por exemplo as propostas por *Charles Peirce* (Peirce77), que promove a importância na utilização de signos para interpretação do raciocínio do ser humano, e *Louis Hjelmslev* (Hjelmslev63), que desenvolveu uma teoria que permite uma análise mais específica, detalhada e formal das estruturas dos signos lingüísticos.

De Souza (DeSouza93) propõe uma abordagem fundamentada na teoria semiótica, na qual o sistema computacional é visto como um artefato de *metacomunicação*, onde as interfaces comunicam mensagens enviadas dos projetistas para os usuários e estas mensagens, por sua vez, enviam e recebem mensagens entre os usuários. A mensagem entre o projetista do sistema e o usuário deve descrever a funcionalidade do sistema, ou seja, quais os tipos de problemas que o sistema pode solucionar, e como estes problemas podem ser resolvidos por meio do modelo de interação.

A abordagem da Engenharia Semiótica não se aplica somente a interfaces gráficas, pois seu foco está na comunicação interpessoal entre o projetista do sistema e os usuários. Segundo Leite (Leite98), o projetista tem um papel comunicativo ao se utilizar da interface para explicitar suas mensagens ao usuário, e a funcionalidade da aplicação pode ser ensinada por meio da coleção de signos da interface. Por outro lado, o usuário ao utilizar uma interface, realiza um esforço de interpretação e compreensão a respeito do significado do que o projetista quis transmitir.

4.2.1 Linguagem de Especificação da Mensagem do *Designer*

Do trabalho desenvolvido por Leite (Leite98), a presente dissertação aproveita como auxílio ao *design* do ambiente a LEMD (Linguagem de Especificação da Mensagem do *Designer*), que tem como objetivo apoiar a formulação da mensagem sobre o modelo de usabilidade. Ela permite especificar os signos do domínio, as funções da aplicação e os comandos de função como sendo mensagens enviadas pelo projetista do sistema. As sentenças da linguagem descrevem as mensagens de interação que compõem a mensagem global. Estas mensagens são descritas de maneira abstrata independentemente de quais signos de interfaces serão utilizados.

A LEMD possui vários tipos de mensagens de acordo com seu significado, como por exemplo, mensagens que comunicam ao usuário as tarefas que ele pode utilizar, o processo de navegação nas telas de interface, e informações para auxiliar o usuário na interação com o sistema. A diferenciação entre os signos de interface é fundamental para o usuário utilizar o sistema, e a organização destes signos em mensagens é que irá auxiliar o usuário a aprender sobre a realização de tarefas do sistema. As mensagens descritas na Tabela 2.1 foram extraídas do artigo de Leite a

respeito da utilização da LEMD para a Engenharia Semiótica de interfaces de usuário (Leite99).

TABELA 2.1 – MENSAGENS DA UTILIZAÇÃO DA LEMD PARA A ENGENHARIA SEMIÓTICA DE INTERFACES DE USUÁRIO

MENSAGEM	DESCRIÇÃO
Estados de signos do domínio	- Revelam o estado do sistema - Permitem ao usuário avaliar se sua meta foi atingida
Funções da aplicação	- Revelam o estado operacional da aplicação - Revelam o que o usuário deve fazer para controlar a aplicação
Estrutura sintática dos comandos	- Revelam a estrutura e a articulação das interações que o usuário precisa desempenhar
Interações básicas	- Indicam ao usuário a interação a ser realizada, como por exemplo *Enter* para fornecer informação ou *Select* para selecionar informação
Metacomunicação de assistência a tarefas	- Auxiliam ao usuário a realizar tarefas compostas por mais de um comando
Metacomunicação para apresentação e controle de leitura da mensagem	- Comunicam como o usuário deve "ler" a própria mensagem do projetista de interface. A navegação entre telas, a ação de mover, aumentar e diminuir janelas são exemplos de ações que o usuário faz para "ler" a interface, como quem folheia um livro
Metacomunicação direta	- Permitem ao projetista de interface enviar uma mensagem diretamente ao usuário para se referir a qualquer outro elemento da interface, inclusive à própria mensagem.

Concluída a especificação da mensagem, o projetista deve relacionar as sentenças da linguagem em signos da interface gráfica, chamados de *widgets*, utilizando as regras de mapeamento semântico. Estas regras mostram, por exemplo, que *Activate* pode ser mapeado em botões de acionamento, permitindo a construção de protótipos da interface que podem ser avaliados por meio de testes de usabilidade (Leite98).

Na Engenharia Semiótica o projetista de interface pode ser apoiado pela LEMD, que oferece a possibilidade de construir os diversos tipos de mensagens que possam ser conduzidas pela interface de usuário. A LEMD funciona como sistema semântico que permite descrever estes componentes do conteúdo da mensagem. Isto significa que cada elemento do modelo de usabilidade que tenha sido concebido e especificado deve estar associado por regras de correlação semântica a um signo de interface.

4.3 UM PROCESSO DE *DESIGN* DE INTERFACES

No pressuposto de que o *design* de interfaces situa-se melhor na arquitetura estrela, tendo no centro a avaliação, tem-se que não existe, portanto, um único processo de *design*, podendo-se, apenas, propor caminhos alternativos (Hartson89). Nesta abordagem, o presente trabalho valeu-se de uma linguagem para orientar o projetista a planejar e projetar as interfaces do sistema utilizando os conceitos de Engenharia Semiótica (DeSouza93). Este modelo inicia-se com a análise e modelagem de usuários e tarefas, para identificar o perfil e características do usuário e as tarefas que ele deve executar no sistema. A seguir desenvolve-se o *design* da comunicação projetista-usuário, para determinar as mensagens que o projetista de interface deve transmitir ao usuário e a relação destas mensagens com os signos da aplicação. Com todos estes requisitos levantados, desenha-se um esboço da interface do sistema, chamado de *storyboarding*. Por último, faz-se a revisão do projeto (avaliação), e sendo necessário, corrige-se as inconsistências levantadas (DeSouza93).

A análise e modelagem de tarefas têm por objetivo identificar as tarefas requeridas do sistema segundo a visão dos seus usuários potenciais e a formalização destas tarefas. Primeiramente são determinados os cenários da aplicação, supostamente por meio de interação direta com o usuário potencial e

questionamento sistemático, identificando situações de insucesso (erros) e de *breakdowns* possíveis. Em seguida define-se a estrutura hierárquica de tarefas e desenvolve-se uma especificação GOMS (*Goals, Operations, Methods and Selection Rules)* (Card83) da aplicação, tentando captar as regularidades da linguagem de interface.

O objetivo da fase de análise e modelagem da comunicação *designer*-usuário é definir o que o *designer* tem que transmitir ao usuário e como deverá fazê-lo, de forma a revelar aos usuários as respostas às perguntas: (a) qual a interpretação do *designer* sobre o problema em questão?; e (b) como o usuário tem que interagir com a aplicação para resolver este problema?(DeSouza93)

Primeiramente é necessário determinar as mensagens que o *designer* deverá passar ao usuário, incluindo os diversos casos de interação (mensagens de orientação sobre as tarefas disponíveis num dado instante e mensagens de apoio à solicitação das tarefas – golfo de execução, mensagens de sucesso, de fracasso e de *breakdowns*[9] – golfo de avaliação), correspondentes às diversas tarefas antes identificadas.

O objetivo da fase *storyboarding*, que é um esboço da interface, é verificar se os requisitos definidos no início do processo estão sendo atendidos pela interface. Esta técnica está relacionada com a utilização de cenários, com a diferença de que estes utilizam narrativa para descrever os casos de uso, enquanto que o *storyboarding* utiliza desenhos e ilustrações, tanto em papel quanto em computador.

A avaliação do *storyboarding* deve verificar se todas as tarefas definidas estão sendo facilitadas pela interface, tanto na ida (execução – solicitação de

[9] *Breakdowns* ocorre quando a interface apresenta funções que causam dificuldades ao usuário, impedindo ações e operações em determinados sujeitos e objetos (BØdker91). A mudança imprevista nas condições materiais de uma atividade causa um *breakdown* que leva a atenção do usuário para a utilização da interface que a princípio deveria ser apenas um meio para a atividade.

tarefas) quanto na volta (avaliação do resultado), buscando a minimização das distâncias semântica (significado – conteúdo, estruturação, granularidade, expressão de toda a natureza de informação necessária) e articulatória (forma – expressão natural para o usuário, seja ela textual, gráfica, etc., explorando recursos de comunicação visual). Deve avaliar também se a interface está consistente, segundo os eixos de consistência sintática (regras de especificação de tarefas, seja esta especificação via comandos ou gráfica), consistência sintático-semântica (articulação do significado na expressão das tarefas), consistência semântica (completude funcional), e consistência lexical (relação significado-forma não arbitrária), lembrando que a consistência é uma propriedade de configuração, isto é, que ter que ser analisada no todo da interface. O *storyboarding* avalia ainda se a interface está verificando as diretrizes e critérios gerais do *design* de interfaces. Esta fase é apenas de verificação, já que estas diretrizes devem ser decorrentes do processo recém descrito. E por fim, avalia se a interface está explorando todo o potencial da tecnologia, isto é, se a interface está se apropriando do poder da tecnologia computacional (e da interface em particular) para introduzir melhorias significativas na execução das tarefas, e atuando como espaço de inovação e não apenas de automação.

5 O MÓDULO DE VISUALIZAÇÃO DE *TRIGGERS*

O objetivo do presente trabalho consistiu no desenvolvimento de um ambiente de interface que permitirá ao usuário visualizar e comparar os *triggers* existentes nas bases de dados iniciais, utilizando como modelo de relacionamento o ERC+, como auxílio à compreensão e à identificação de correspondências existentes, a partir do aproveitamento de conceitos e metodologias de IHC, tais como a Engenharia Cognitiva e a Engenharia Semiótica.

Nas seções a seguir são descritos o processo de *design* utilizado, o funcionamento do ambiente e um exemplo de aplicação.

5.1 O PROCESSO DE *DESIGN* DE INTERFACES NA FERRAMENTA MATRIGS

A análise e modelagem de usuários tem por objetivo levantar e caracterizar os usuários potenciais da aplicação, de forma a identificar os requisitos mais importantes do sistema e instrumentar a tomada de decisões de projeto.

- Papéis: administrador de banco de dados, analista de sistemas
- Familiaridade com computadores: administrador de banco de dados e analista de sistemas - experiente
- Nível de conhecimento do domínio da aplicação:
  - administrador: como já possui domínio nas ferramentas de banco de dados, deve ser especialista, pois a aplicação a ser desenvolvida deverá ser similar com estas ferramentas de trabalho.
  - analista de sistemas: inicia como médio, pois o analista pode não ter intimidade com as ferramentas de banco de dados, mas tem

conhecimento conceitual de como deve funcionar. Deve evoluir para especialista por meio da interação com a aplicação.

Tanto o administrador quanto o analista podem não possuir nenhum conhecimento do domínio do banco de dados, como por exemplo, o acervo de uma biblioteca universitária ou de um museu. Neste caso, começam como iniciante, devendo evoluir para médio e especialista por meio da interação com a aplicação.

- Freqüência de uso: administrador e analista de sistemas – freqüente.
- Fatores sócio-culturais: Em relação à ferramentas de banco de dados, a linguagem utilizada é conhecida por administradores e analistas de sistemas. Em relação à área na qual pertence o banco de dados, pode não haver nenhuma familiaridade.

O modelo GOMS contribuiu para a resolução do problema permitindo antecipar as dificuldades cognitivas do usuário no uso do sistema. A especificação GOMS para o problema em questão encontra-se no Apêndice I.

São dois os cenários determinados para as tarefas do usuário, sendo o primeiro referente ao mapeamento dos *triggers* de bases de dados já existentes no sistema, e o segundo referente ao mapeamento dos *triggers* de novas bases de dados, devendo o usuário, neste caso, fornecer o arquivo *script* da base de dados original e identificar a base de dados no sistema.

CENÁRIO 1: Mapeamento dos *triggers* de base de dados já existentes

O usuário deve selecionar a base de dados, suas entidades e atributos que deseja visualizar detalhadamente.

O usuário deve poder consultar a qualquer momento as informações detalhadas de cada um dos objetos, como o comando SQL da base de dados de

origem, tanto em formato texto quanto em forma de grafo (diagrama de sintaxe de *triggers* – anexo I), a visualização gráfica dos *triggers*, individualmente, quando estes forem selecionados, e a relação da dependência dinâmica entre as entidades, ou seja, a partir de uma entidade selecionada deve ser possível listar e visualizar graficamente todas as entidades que são alteradas por ela e todas as entidades que a alteram.

O sistema deve permitir ao usuário, além de navegar entre as entidades, visualizar graficamente todos os *triggers* pertencentes à base de dados selecionada, informar um novo *script* ou sair do sistema.

O sistema deve permitir ao usuário a visualização tanto de todos os *triggers* da base de dados, quanto individualmente, sob demanda específica do usuário.

CENÁRIO 2: Mapear os *triggers* de uma nova base de dados

O usuário deve solicitar ao sistema que deseja entrar com novo arquivo *script*.

O usuário deve informar a localização e nome do arquivo *script* da base de dados que deseja mapear. Caso ele não saiba a localização precisa, deve ser dada a possibilidade de busca no sistema de arquivos.

O usuário deve informar um nome qualquer para identificar a base de dados que está mapeando.

O sistema deve permitir ao usuário limpar todos os campos já digitados para nova entrada de dados, cancelar a entrada de novo arquivo *script* voltando para a tela de mapeamento ou sair do sistema.

Uma vez confirmada a entrada dos dados, o sistema deve exibir como resposta a relação de objetos com seus respectivos atributos e *triggers*, mapeando a base de dados focando os *triggers*.

Dentre os *breakdowns* previstos, podem ser citados os seguintes:

- O arquivo *script* informado pelo usuário pode não existir. O sistema deve orientar o usuário a realizar novas tentativas de busca, auxiliando na localização do arquivo.
- O nome da base de dados pode existir. O sistema deve orientar o usuário e permitir que ele possa entrar com a informação correta.
- O campo informando o arquivo *script* pode estar em branco. O sistema deve orientar o usuário e permitir que ele possa entrar com a informação correta.
- O campo informando o nome da base de dados pode estar em branco. O sistema deve orientar o usuário e permitir que ele possa entrar com a informação correta.

De acordo com a especificação do presente trabalho, identifica-se o mapeamento da base de dados tendo como foco seus *triggers,* como sendo a principal tarefa a ser desempenhada pelo usuário no ambiente do sistema. A partir daí, pode-se relacionar a estrutura hierárquica das tarefas (Tabela 5.1).

A utilização da LEMD viabiliza o planejamento e a adequada realização da comunicação *designer*-usuário, resultando numa interface mais auto-explicativa. A especificação em LEMD da comunicação *designer*-usuário é apresentado no Apêndice II. Uma vez especificado o programa LEMD deve-se determinar as regras de correlação mensagem-*widgets* da interface que irão determinar a forma final da aplicação. No Apêndice III são relacionadas a mensagem do sistema (ação) com o correspondente *widget* da interface proposta, além do significado para cada mensagem.

TABELA 5.1 – ESTRUTURA HIERÁRQUICA DE TAREFAS

Tarefa	Sub-Tarefas	Ações	Operações
Mapear a Base de Dados em relação aos *triggers*	Fornecer nome do arquivo *script*	*Entrar com informações do arquivo script*	Clicar botão "Novo Script" (tela Mapeamento)
			Digitar caminho (path) e nome do arquivo *script* da base de dados a ser mapeada
	Buscar arquivo *script*	Localizar caminho e arquivo *script*	Clicar botão "Localizar..."
			Navegar através das pastas e arquivos
			Selecionar arquivo desejado
			Clicar botão "Confirmar"
	Fornecer nome da Base de Dados	Entrar com nome da base de dados	Digitar o nome da base de dados para identificação
	Mapear nova base de dados	Confirmar entrada de dados para novo mapeamento	Clicar botão "Confirmar" na tela Fornecer o *script*
	Mapear base de dados existente	Selecionar componente da base de dados	Clicar sobre o componente da base de dados (seleciona) no quadro à esquerda
		Visualizar comando SQL	Selecionar pasta "comando SQL"
			Selecionar pasta "texto" ou
			Selecionar pasta "grafo"
		Visualizar *trigger*	Selecionar pasta "visualizar"
			Clicar botão ativar condições
			Clicar botão desativar condições
		Visualizar Denpendência Dinâmica	Selecionar pasta "Dependência Dinâmica"
			Selecionar pasta "Altera" ou
			Selecionar pasta "Alterado por"
		Visualizar todos os *triggers*	Selecionar modo visualizar todos os *triggers*
			Clicar botão mais zoom
			Clicar botão menos zoom
			Clicar botão ativar condições
			Clicar botão desativar condições

A fase da avaliação parcial ficou restrita à avaliação pelo especialista, permanecendo a necessidade de avaliação pelo usuário final, atividade a ser desenvolvida em trabalhos futuros.

As telas da interface resultante do processo de *design* são apresentadas no Apêndice IV. Um exemplo do ambiente é proporcionado na seção 5.3.

5.2 FUNCIONAMENTO DO SISTEMA DE MAPEAMENTO DE *TRIGGERS*

Como entrada para o módulo do sistema de mapeamento de *triggers* é necessário um mapeamento completo do banco de dados inicial, ou seja, um extrato em formato texto. Com isso é possível identificar e localizar os dados estáticos e dinâmicos do banco de dados, como objetos, chaves e *triggers*. Este mapeamento é chamado *script* e é gerado automaticamente pelos sistemas gerenciadores de banco de dados, como Oracle e MS-SQL (Figura 5.1).

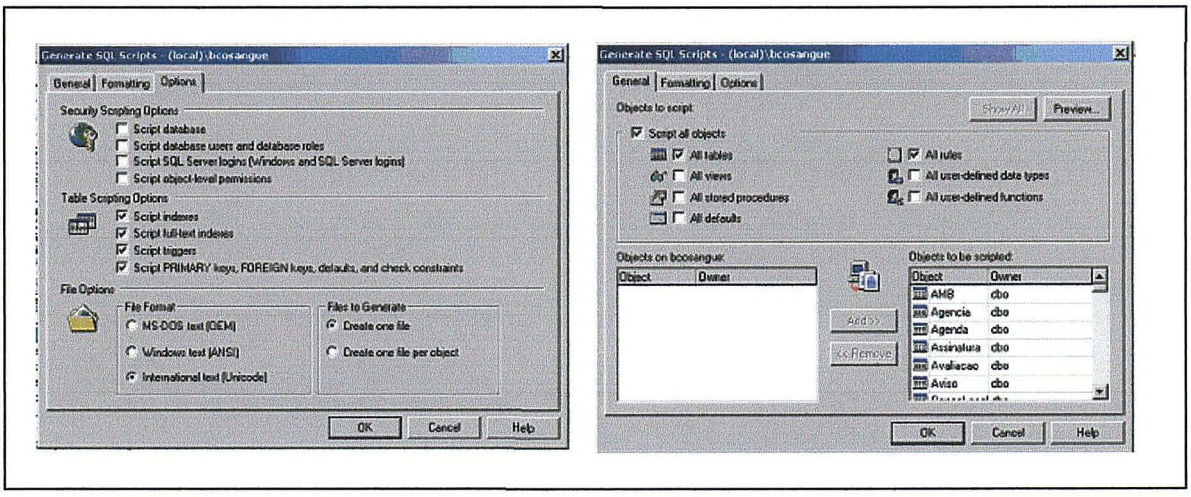


FIGURA 5.1 – TELAS DO MS-SQL PARA GERAR *SCRIPT* DO BANCO DE DADOS

Na Figura 5.2 é mostrado um exemplo de um *script* gerado pelo MS-SQL e visualizado num editor de texto.

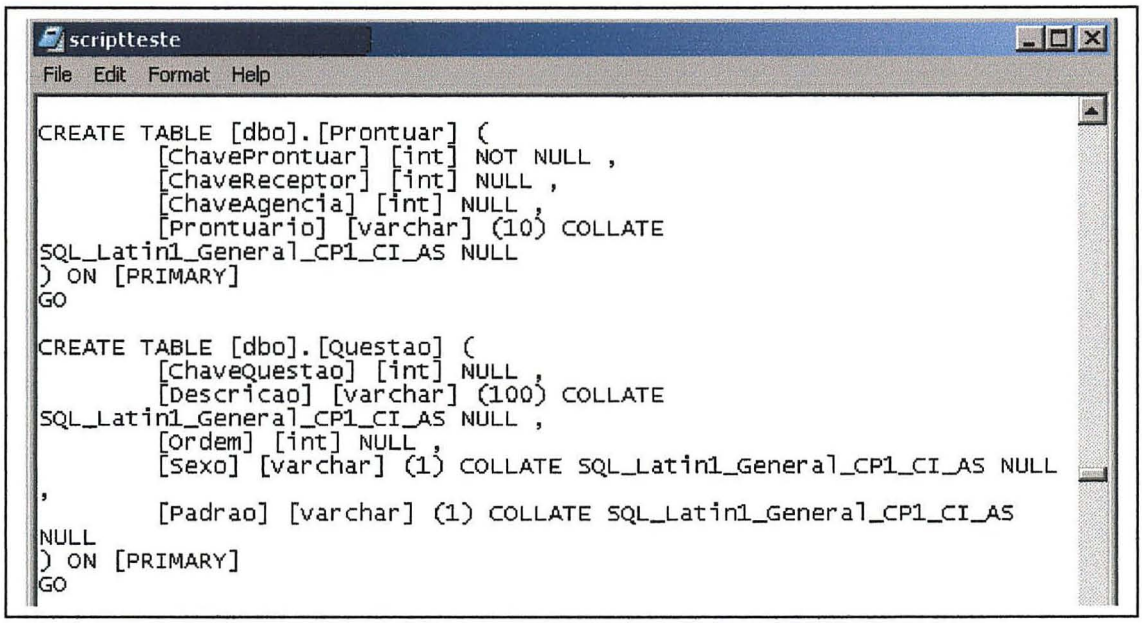


FIGURA 5.2 – TRECHO DO *SCRIPT* DE UMA BASE DE DADOS ONDE PODE-SE VISUALIZAR OS OBJETOS E SEUS RESPECTIVOS ATRIBUTOS

O ambiente MaTrigs inicia-se na tela de mapeamento de *triggers* (Figura A1, Apêndice IV), onde estão disponíveis as bases de dados já mapeadas anteriormente através do fornecimento do arquivo *script* (Figura A2, Apêndice IV). Na ferramenta desenvolvida por Scopim (Scopim03), a J-Schemas Integrator, pode-se acessar MaTrigs através de um botão  na paleta de botões correspondentes às regras de integração implementadas em cada frame (Figura 5.3).

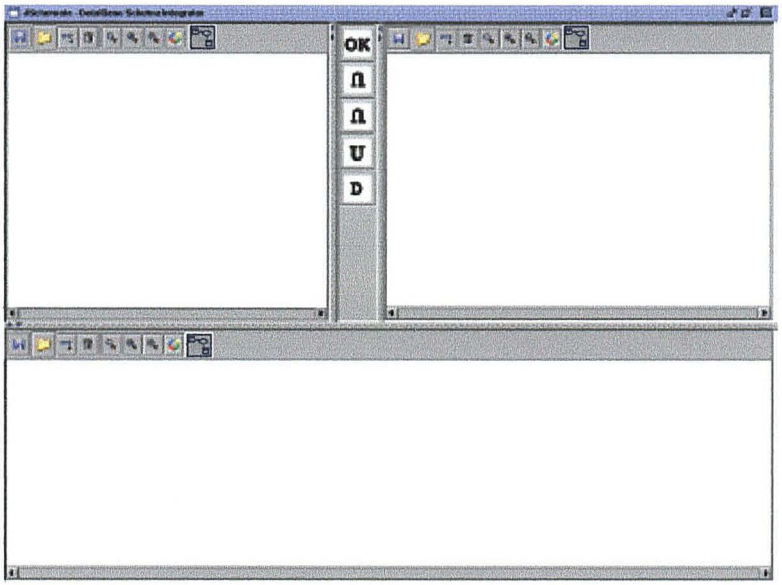


FIGURA 5.3 – TELA DA FERRAMENTA J-SCHEMAS INTEGRATOR COM BOTÃO PARA ACESSAR O AMBIENTE MATRIGS.

O fornecimento de dados para mapeamento da base de dados se dará através de uma interface, conforme Figura A2 no Apêndice IV, onde o usuário deve fornecer o nome do arquivo *script*, inclusive sua localização, e um nome para identificar a base de dados no sistema, definido pelo usuário. O usuário pode a qualquer momento cancelar as tarefas e voltar para o estado anterior (mapeamento das bases já existentes, Figura A1 - Apêndice IV), sair do sistema ou continuar com o processo.

Uma vez fornecidos os dados para o mapeamento dos *triggers*, o processo será composto dos seguintes passos:

1. Identificação de todas as entidades do banco de dados, por meio do *script* fornecido;
2. Identificação dos *triggers*;
3. Análise semântica dos *triggers*, localizando, a partir de palavras reservadas, as entidades, condições e ações que estão contidas em cada *trigger*;
4. Geração das associações entre objetos contidos nos *triggers* (somente os objetos que participam do *trigger*);
5. Representação gráfica dos *triggers*.

5.2.1 Identificação das entidades do banco de dados

Como visto no Capítulo 4, as linguagens SQL dos bancos de dados atualmente disponíveis no mercado possuem uma sintaxe que varia conforme seu fabricante. A partir de palavras reservadas pertencentes às linguagens, é possível obter informações importantes para o mapeamento do diagrama entidade-relacionamento da base de dados. Por exemplo, **CREATE TABLE** é um comando para criar tabelas em SQL3 e Oracle, podendo a partir dele identificar todas as

tabelas e seus respectivos atributos do banco de dados. Na Figura 5.2 são mostrados trechos de comandos para criar tabelas em MS-SQL.

A identificação de todos os objetos do banco de dados - tabelas e seus atributos – será fundamental para mapear os *triggers*, pois como revisado no Capítulo 4, os *triggers* são comandos acionados por tabelas, e suas condições e ações advém dos atributos das mesmas. Não é necessário identificar o relacionamento entre os objetos, pois esta informação não é utilizada para *triggers*, nem interfere no seu funcionamento.

5.2.2 Identificação dos *triggers*

Da mesma forma como são identificados os objetos da base de dados por meio de palavras reservadas, pode-se encontrar os *triggers*. Por exemplo, **CREATE TRIGGER** é o comando para criar *triggers* em SQL-3 e Oracle.

O objetivo desta fase é identificar a tabela à qual o *trigger* pertence.

5.2.3 Análise da semântica dos *triggers*

Identificados os *triggers* e sua entidade de origem, devem ser identificadas as condições e ações. A condição ou restrição é o conjunto de atributos (um ou mais atributos pertencentes a uma ou mais tabelas) que combinados servem para acionar (ou não) um comando *trigger*. A ação do *trigger* é o comando (ou bloco de comandos) que devem ocorrer se a condição for avaliada como verdadeira. A ação do *trigger* está sempre direcionada a um conjunto de atributos de uma (ou mais) tabelas.

5.2.4 Geração das associações entre as entidades do *trigger*

Concluída a análise da semântica do *trigger*, é possível identificar todas as entidades e atributos utilizados pelo *trigger*, tanto para avaliar a condição quanto para executar sua ação, bem como a entidade que ativa o *trigger* (onde está armazenado).

Desta forma é possível fazer a associação entre as entidades e atributos que fazem parte de um determinado *trigger*, respondendo às seguintes perguntas: a) Qual entidade dispara o *trigger*? b) Qual(is) entidade(s) e atributo(s) são utilizados pelo *trigger* para gerar a condição de ativação? c) Qual(is) entidade(s) e atributo(s) são utilizados pelo *trigger* para executar a ação caso a condição seja verdadeira?

Como conseqüência tem-se a extração de outras informações de suma relevância para o entendimento e mapeamento da base de dados, como por exemplo, a possibilidade de determinar quais as entidades que são alteradas e que *alteram uma* determinada entidade. No Apêndice IV, pode-se visualizar as telas, sem as entidades, que demonstram a dependência dinâmica das entidades que são alteradas e que alteram, Figuras A.6 e A.5, respectivamente.

5.2.5 Representação gráfica dos *triggers* no modelo ERC+

A representação gráfica do modelo ERC+, como visto no Capítulo 4, inclui retângulos (entidade), losangos (tipo relacionamento), linhas para identificar a cardinalidade e setas para generalização. Para a representação de *triggers* no modelo ERC+ propõe-se acrescentar as representações abaixo descritas, e que estão especificadas no Apêndice III através da correlação mensagem-widgets da interface:

- **Círculo (elipse)** para indicar a existência de *trigger* no modelo;

- **Seta cheia preta** para indicar qual entidade aciona o *trigger*. Esta seta se inicia da entidade à qual o *trigger* pertence, indicando sua origem. As setas já são utilizadas no modelo ERC+ para indicar generalização, porém a generalização ocorre somente entre entidades (retângulos), e em *triggers* somente entre uma entidade (retângulo) e seu respectivo *trigger* (círculo);
- **Seta cheia verde** que ligará o *trigger* (círculo) ao atributo de uma entidade que modifica com a execução do *trigger* (ação). Se houver mais de um atributo envolvido na ação, todos estarão ligados até o *trigger;*
- **Linha cheia azul** que ligará o *trigger* (círculo) ao atributo que indica condição. Se houver mais de um atributo envolvido na condição, todos estarão ligados até o *trigger.*

Na figura 5.4 pode-se visualizar a representação gráfica proposta para mapear *triggers* no ambiente MaTrigs.

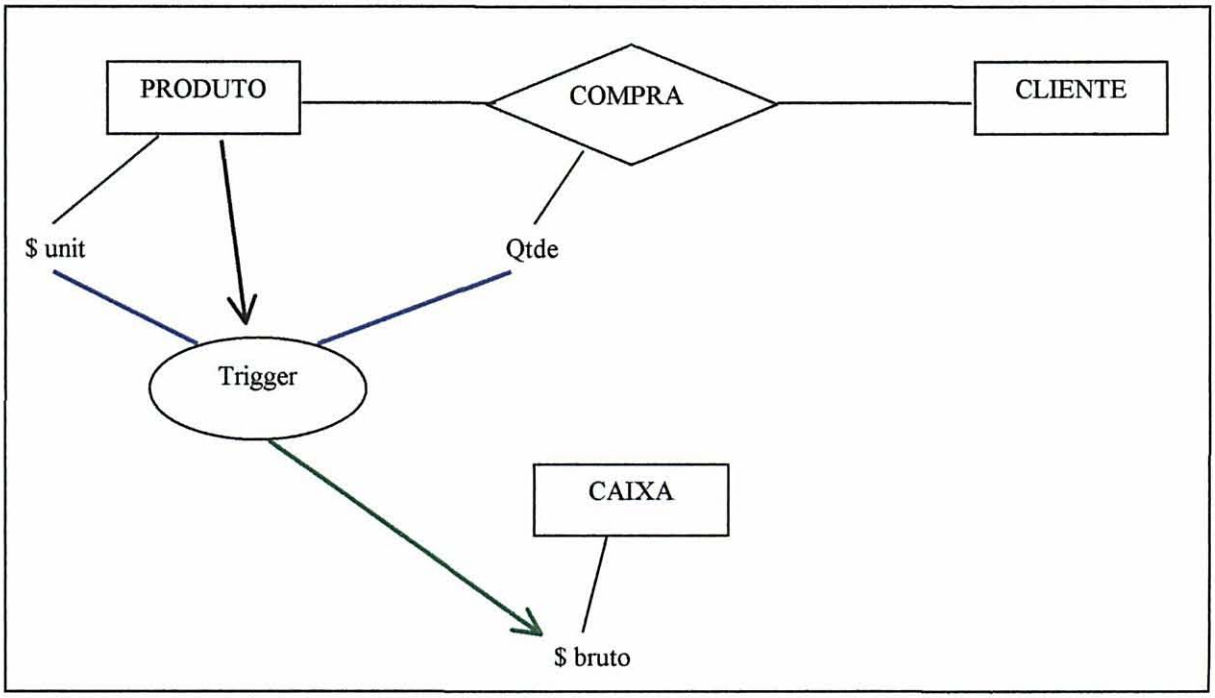


FIGURA 5.4 – REPRESENTAÇÃO DE TRIGGERS NO MODELO ERC+

A escolha das cores azul e verde não está relacionada ao significado expresso pela Psicologia no que se refere ao uso das cores. Foi uma escolha ao acaso, com o intuito de auxiliar o usuário na identificação dos *triggers*, já que a utilização de cores no modelo de dados destaca a existência de *triggers* e difere os atributos entre os que participam como condição e/ou ação. A solução apresentada para demonstrar *triggers* no modelo ERC+ independe do uso de cores, isto é, caso não seja possível diferenciar as cores, por problemas de hardware ou do próprio usuário (daltonismo), a representação gráfica é única, pois a ligação ocorre sempre entre o *trigger* e a(s) entidade(s) envolvidas. Na Figura 5.5 é mostrada, a título de comparação, a mesma representação em formato colorido e monocromático. Em ambos os casos, a utilização das cores não modificam o significado da representação de *triggers*.

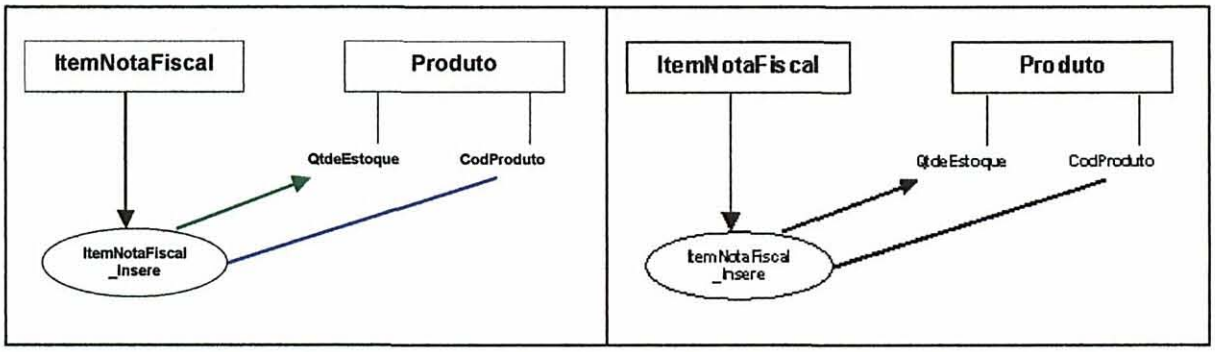


FIGURA 5.5 – COMPARAÇÃO DA REPRESENTAÇÃO DE TRIGGERS EM MODELO ERC+ EM RELACAO A UTILIZAÇÃO DE CORES

5.3 UMA SESSÃO DA FERRAMENTA PROPOSTA

Para mostrar o potencial do ambiente, será apresentado um exemplo de utilização da ferramenta proposta nesta dissertação. Para tanto foi utilizado como exemplo parte de um sistema fictício de nota fiscal com movimentação de estoque

após uma venda. Na Tabela 5.2 tem-se a estrutura da entidade Produto, onde estão armazenadas informações a respeito dos produtos comercializados. Na Tabela 5.3 tem-se a estrutura da entidade **ItemNotaFiscal**, onde estão armazenadas os itens pertencentes a uma nota fiscal; para cada item, pode-se basicamente incluir um novo produto, alterar a quantidade de um item, ou exclui-lo da nota fiscal.

TABELA 5.2 – ESTRUTURA DA ENTIDADE PRODUTO

Produto			
CodProduto	DescProduto	QtdeEstoque	ValorUnitario
12345	Caneta	990	2,00
12346	Lápis	1000	1,00
12347	Borracha	2000	1,00
12348	Caderno	492	5,00

TABELA 5.3 – ESTRUTURA DA ENTIDADE ITEMNOTAFISCAL

ItemNotaFiscal				
CodItemNotaFiscal	NumNotaFiscal	CodProduto	Qtde	ValorItem
1	1	12345	10	20,00
2	1	12348	8	40,00

Supondo que a entidade **ItemNotaFiscal** dispare um *trigger* para inserir um novo item de produto na nota fiscal ou para alterar a quantidade de um produto de um item da nota fiscal ou para excluir um item de um produto da nota fiscal, temos os seguintes comandos, em linguagem SQL do SGBD Oracle, que estão disponíveis no arquivo *script* gerado pela base de dados utilizada:

```
CREATE OR REPLACE TRIGGER itemnotafiscal_insert
BEFORE INSERT OR UPDATE OR DELETE
ON itemnotafiscal
REFERENCING NEW AS NEW OLD AS OLD
FOR EACH ROW
BEGIN
 IF Inserting THEN
   UPDATE produto SET qtdeestoque = qtdeestoque + :new.qtde
            WHERE (produto.codproduto = :new.codproduto);

 END IF;
```

```
IF Updating THEN
   UPDATE produto SET qtdeestoque = qtdeestoque - :old.qtde + :new.qtde
            WHERE (produto.codproduto = :new.codproduto);
END IF;

IF Deleting THEN
   UPDATE produto SET qtdeestoque = qtdeestoque - :old.qtde
            WHERE (produto.codproduto = :new.codproduto)
END IF;
END;
```

Para visualizar os *triggers* do sistema de nota fiscal, o usuário deve inicialmente clicar no botão **Nova Base de Dados** [Nova Base de Dados] na tela de mapeamento de *triggers*. Abre-se a tela "Fornecendo o *Script*" para que o usuário forneça o nome e localização do arquivo *script* e o nome da base de dados (Figura 5.6). Uma vez fornecido os dados de entrada do sistema, o usuário deve clicar no botão **Confirmar** [Confirmar]. O sistema volta para a tela de mapeamento de *triggers* e insere no quadro à esquerda a base de dados recém criada.

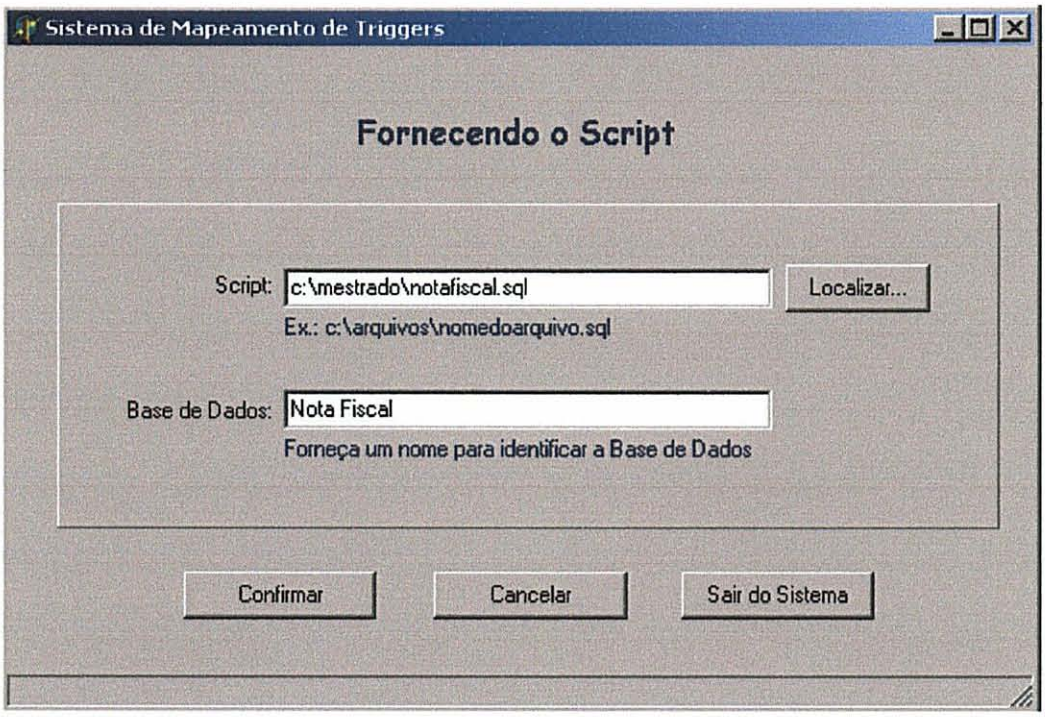


FIGURA 5.6 – TELA PARA FORNECER AS INFORMAÇÕES DA BASE DE DADOS NOTA FISCAL

Neste ponto, o usuário já pode consultar e visualizar as informações relativas a *triggers* da base de dados selecionada. Para tanto, no quadro à esquerda da tela (Figura 5.7), deve clicar sobre a base de dados ou no sinal [+][10] imediatamente à esquerda do nome da base de dados ou de um componente. Se selecionar um *trigger,* são habilitadas as abas **Comando SQL** e **Visualizar**. Se selecionar uma entidade, é habilitada somente a aba **Dependência Dinâmica**.

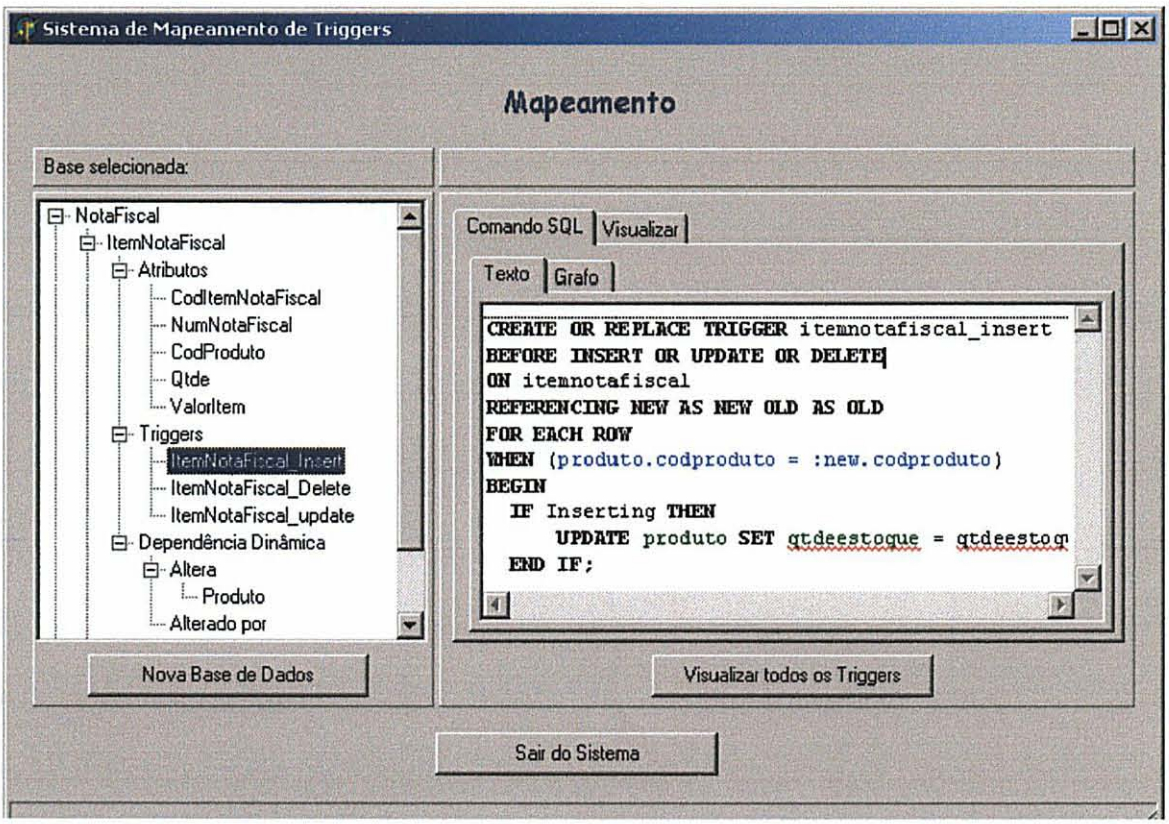


FIGURA 5.7 – TELA COM O MAPEAMENTO DE *TRIGGERS* DA BASE DE DADOS NOTA FISCAL

O usuário seleciona o *trigger* **itemnotafiscal_insert**. Na aba **Comando SQL** são oferecidas duas possibilidades de consulta do comando, tanto em formato texto (Figura 5.8) quanto em formato de grafo (diagrama de sintaxe, Anexo I) (Figura

[10] Este sinal indica a existência de uma sub-estrutura. É um padrão já estabelecido nas interfaces gráficas.

5.9) utilizando mensagens do sistema para o usuário conforme Apêndice III. Selecionando a aba **Texto** o usuário pode visualizar o comando SQL para inserir um item na nota fiscal (**ItemNotaFiscal_insert**) por extenso.

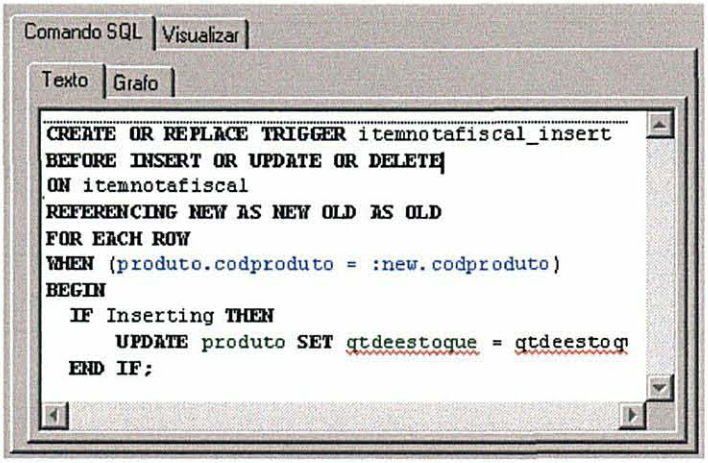


FIGURA 5.8 – DETALHE DA TELA DE MAPEAMENTO DO *TRIGGER* **ITEMNOTAFISCAL_INSERT** – COMANDO SQL EM FORMATO TEXTO

Selecionando a aba **Grafo** o usuário pode visualizar o comando SQL da base de dados num diagrama de sintaxe (Anexo I).

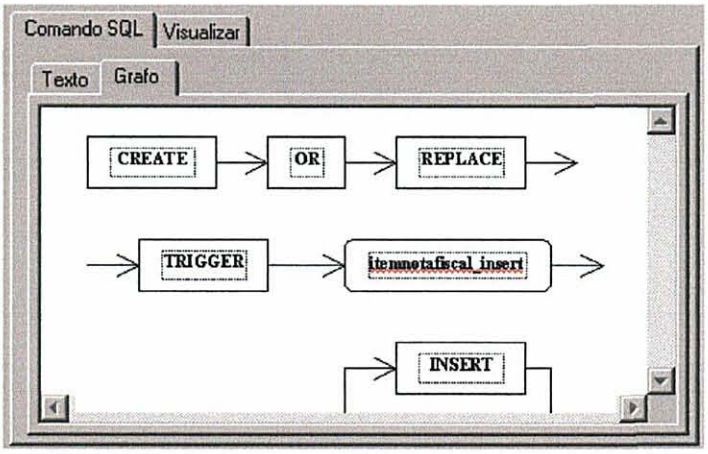


FIGURA 5.9 – DETALHE DA TELA DE MAPEAMENTO DO *TRIGGER* **ITEMNOTAFISCAL_INSERT** – COMANDO SQL EM FORMATO DE GRAFO

Na aba **Visualizar** é apresentada a representação gráfica do *trigger* **itemnotafiscal_insert** (Figura 5.10) seguindo as mensagens do sistema para o usuário conforme listado no Apêndice III.

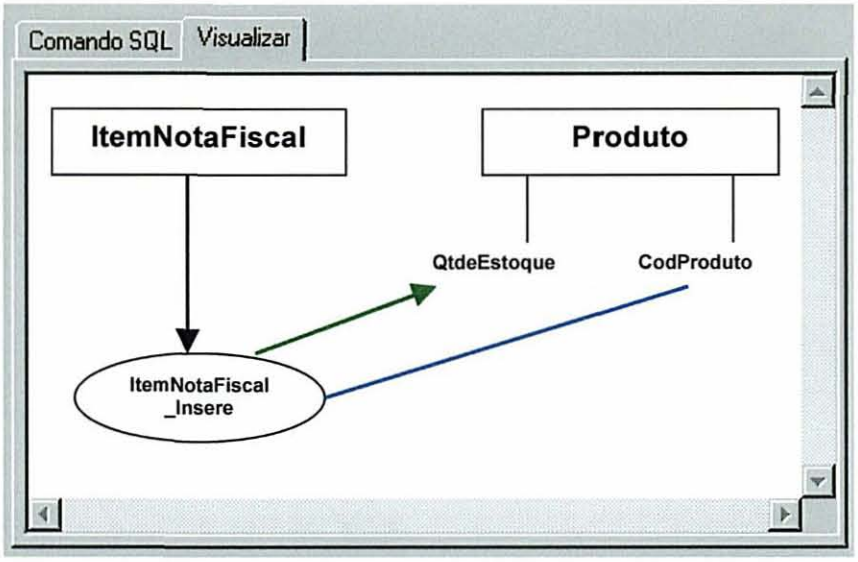


FIGURA 5.10 – DETALHE DA TELA DE MAPEAMENTO DO *TRIGGER* **ITEMNOTAFISCAL_INSERT** - VISUALIZAR

Se o usuário selecionar uma entidade no quadro onde estão listados a base de dados e seus componentes, como por exemplo a entidade **ItemNotaFiscal**, a aba **Dependência Dinâmica** é habilitada (figura 5.11). O usuário tem duas possibilidades de consulta. A primeira, chamada **altera**, visa a(s) entidade(s) que é(são) alterada(s) pela entidade selecionada. No exemplo, o usuário seleciona a entidade **ItemNotaFiscal** e o sistema apresenta uma lista de entidades que sofrem alteração (no exemplo, Produto) e a visualização gráfica de todos os *triggers* que são acionados pela entidade selecionada (no exemplo, os *triggers* itemnotafiscal_insert, itemnotafiscal_delete, itemnotafiscal_update são acionados pela entidade ItemNotaFiscal).

Na segunda, chamada **alterada por**, mostra a(s) entidade(s) que acionam *trigger(s)* que alteram a entidade selecionada. No exemplo mostrado na figura 5.12 o usuário seleciona a entidade **Produto** e o sistema apresenta uma lista de entidades que a alteram (no exemplo, ItemNotaFiscal) e a visualização gráfica de todos os *triggers que* alteram a entidade selecionada (no exemplo, os *triggers* itemnotafiscal_insert, itemnotafiscal_delete, itemnotafiscal_update alteram a entidade Produto).

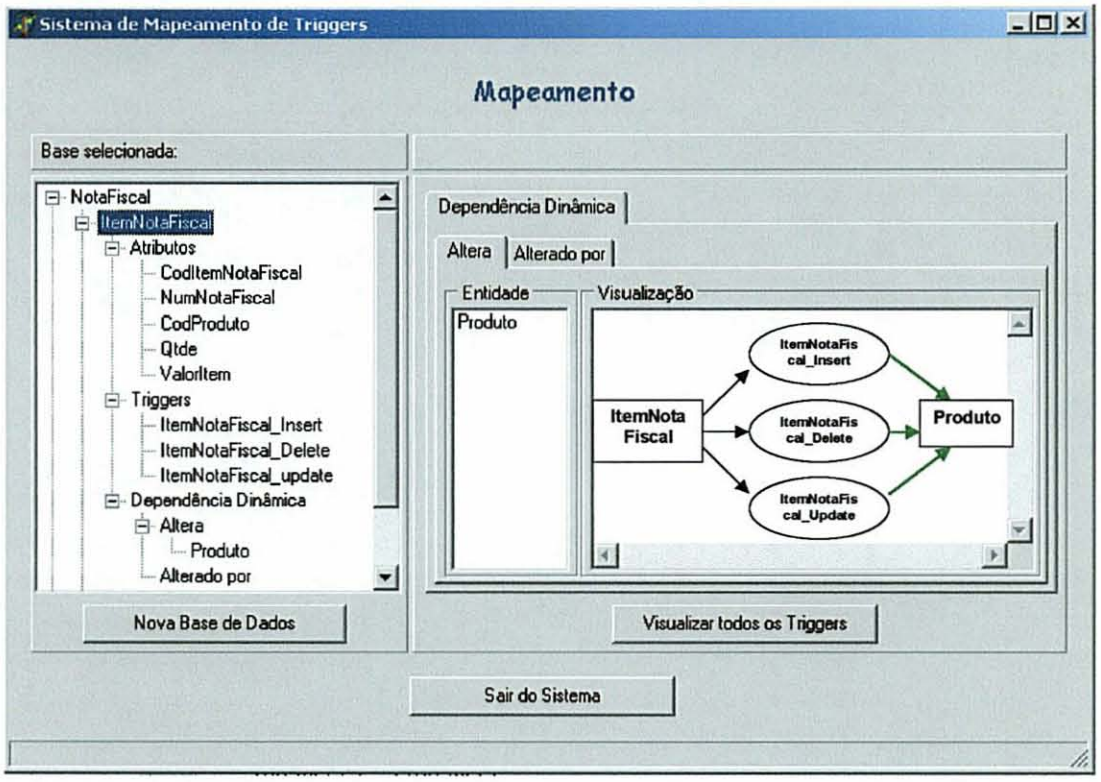


FIGURA 5.11 – TELA DE MAPEAMENTO DE *TRIGGERS* COM A OPÇÃO DEPENDÊNCIA DINÂMICA - ALTERA

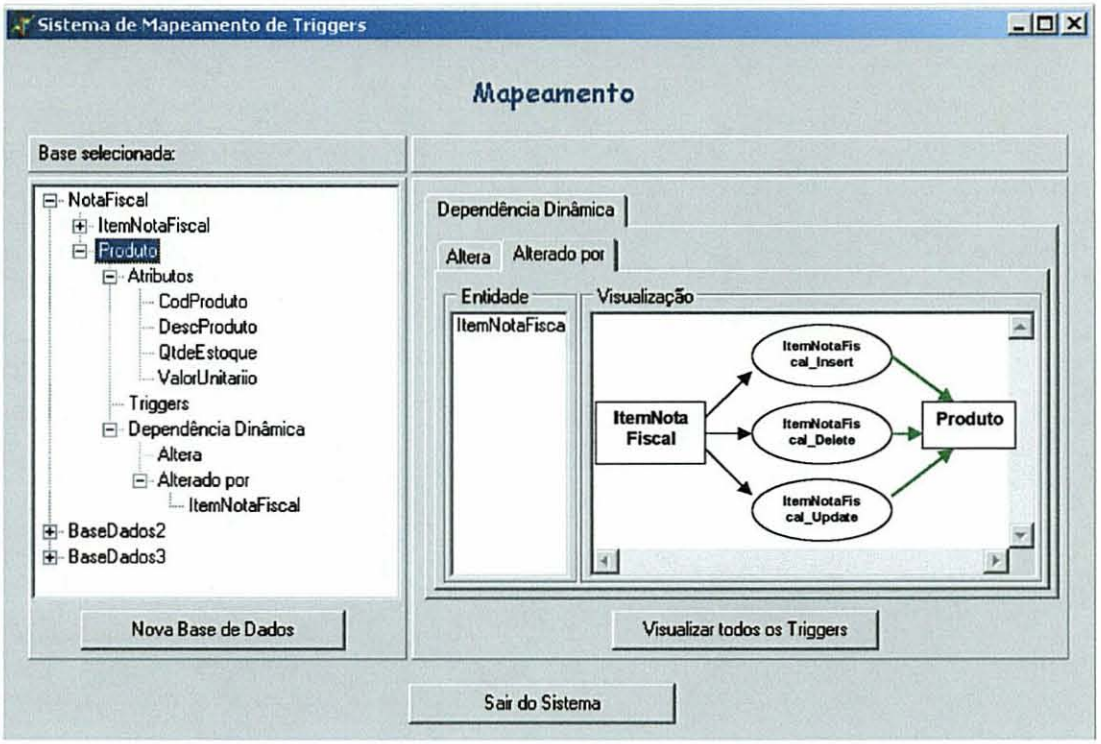


FIGURA 5.12 - TELA DE MAPEAMENTO DE *TRIGGERS* COM A OPÇÃO DEPENDÊNCIA DINÂMICA – ALTERADO POR

A qualquer momento, o usuário tem a possibilidade de visualizar todos os *triggers* da base de dados, clicando no botão **Visualizar todos os *Triggers*** 

. No exemplo, abre-se uma tela com a relação das entidades e *triggers* que compõe a base de dados nota fiscal[11] (figura 5.13).

[11] Mostra somente as entidades que participam de um ou mais *triggers*

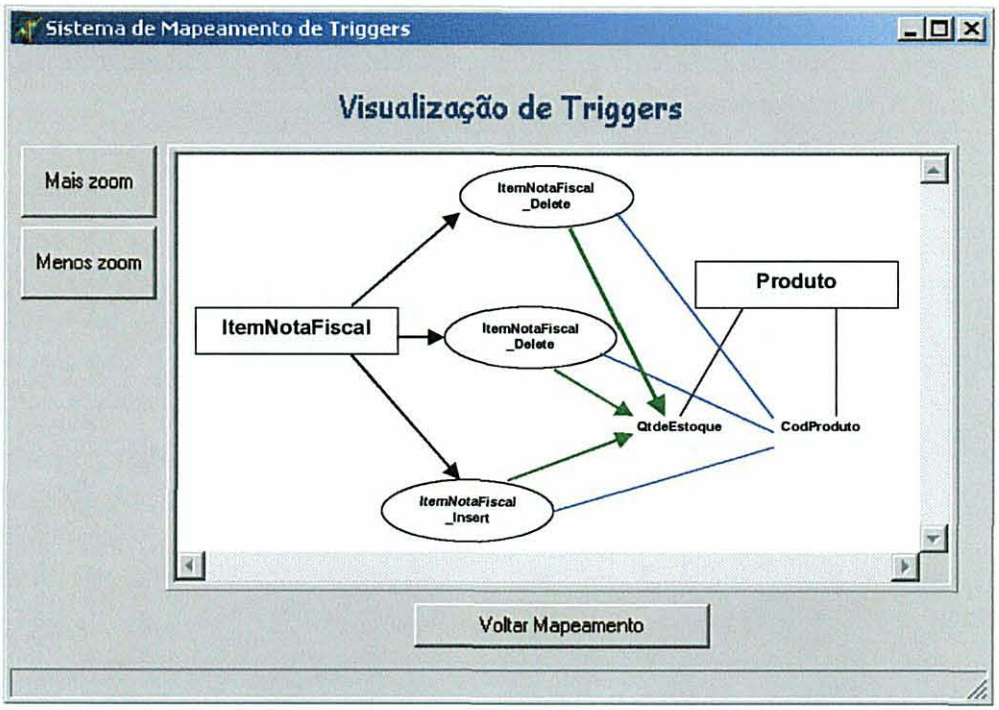
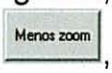


FIGURA 5.13 – TELA VISUALIZANDO TODOS OS *TRIGGERS* DA BASE DE DADOS NOTA FISCAL

Nesta tela o usuário tem a possibilidade de ampliar ou reduzir o diagrama, através da utilização dos botões **mais zoom** 

e **menos zoom** 

, respectivamente. Outra opção de visualização é o botão **Ligar / Desligar condições**, que denota a possibilidade de visualizar as condições que participam do *trigger* ou ocultar estas condições.

6 CONCLUSÕES E TRABALHOS FUTUROS

De maneira geral, a existência de uma ferramenta para mapeamento de *triggers* de uma base de dados, onde o usuário pode visualizar as entidades envolvidas no processo, seja como repositório do *trigger* (entidade que o aciona), como entidade que possui atributos utilizados como condição de ativação do *trigger*, ou como entidade que terá seus atributos modificados pela execução do *trigger*, contribui com o processo para manipulação de bases de dados.

No entanto, este trabalho de pesquisa se insere no contexto de um ambiente de auxílio à integração de esquemas. Com base em algumas abordagens subjacentes à área de Interação Ser Humano-Computador (a Engenharia Cognitiva, o modelo *Goals, Operators, Methods and Selection Rules*, a Engenharia Semiótica e um processo de *design* de interfaces envolvendo adicionalmente cenários e *storyboarding*) produziu um protótipo de ambiente de interface para a visualização destes aspectos dinâmicos até o momento desconsiderados neste tipo de ferramenta. A contribuição deste recurso apóia-se no fato de a existência de um *trigger*, elemento semântico forte do banco de dados, poder ter um papel determinante tanto na identificação quanto na eliminação de aparentes correspondências entre os esquemas.

Na integração de esquemas a interpretação da realidade subjacente é complexa e demorada por se apoiar em decisões humanas prévias desconexas. Neste contexto, a visualização de triggers se apresenta como uma primeira melhoria substancial no auxílio à tarefa de identificação de correspondências.

A apresentação desta ferramenta gráfica para a integração de esquemas reforça a convicção da dependência da área de IHC na resolução de problemas complexos de BD.

Como desdobramento da presente dissertação podem ser vislumbrados pelo menos três trabalhos de pesquisa. O primeiro consiste na integração do ambiente MaTrigs ao J-Schemas Integrator, de forma a, juntamente com o módulo de identificação de correspondências, em andamento como trabalho paralelo, potencializar o auxílio ao usuário no processo de integração de esquemas. O segundo trabalho é a realização de um experimento de avaliação do ambiente de interface-usuário da ferramenta J-Schemas Integrator completa, que envolva a participação ativa do usuário. O terceiro trabalho, desdobramento conceitual principal, consiste na realização de análises sintática e semântica aprofundadas dos triggers, com o intuito de permitir a comparação entre os triggers de duas bases em integração.

REFERÊNCIAS BIBLIOGRÁFICAS

[ACM92] ACM SIGCHI. **ACM SIGCHI Curricula for Human-Computer Interaction**. ACM Special Interest Group on Computer-Human Interaction Curriculum Development Group, 1992.

[BØdker91] BØDKER, S. Through the Interface: **A human activity approach to user interface design**. Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1991.

[Card83] CARD, S.; MORAN, T.; NEWELL, A. **The Psychology of Human-Computer Interaction**. Lawrence Erlbaum, 1983.

[Carol85] CAROL, J. M.; ROSSON, M. B. Advances in human-computer interaction, volume 1, H. R. HARTSON. **Usability Specifications as a tool in Iterative Development** 1985.

[Catarci96] CATARCI, T.; CHANG, S.; LEVIALDI, S.; SANTUCCI, G. **A Graph-Based Framework for Multiparadigmatic Visual Access to Databases.** IEEE Transactions on Knowledge and Data Engineering, Vol 8, n. 3, June 1996.

[Chen90] CHEN, P. **Modelagem de dados: a abordagem entidade-relacionamento para projeto lógico**. Tradução de Cecília Camargo Bartalotti. São Paulo: McGraw-Hill: Makron, 1990.

[Dennebouy95] DENNEBOUY, Y.; ANDERSSON, M.; AUDDINO, A.; DUPONT, Y.; FONTANA, E.; GENTILE, M.; SPACCAPIETRA, S.. **SUPER: Visual Interfaces for Object+Relationship Data Models.** Journal of Visual Languages and Computing, 1995.

[DeSouza93] DE SOUZA, C.S. **The Semiotic Engineering of User Interface Languages**. International Journal of Man-Machine Studies Academic Press, 1993.

[Gueiber01] GUEIBER, E. **VIQUEN – Um ambiente interativo para consulta visual e extração de esquemas.** Dissertação de mestrado, Programa de Pós- Graduação em Informática, UFPR, 2001.

[Hartson89] HARTSON, H.; HIX, D. **Human-Computer Interface Development: Concepts and Systems for its Management**. ACM Computing Surveys, v.21. 1989.

[Hutchins86] HUTCHINS, E. L.; HOLLAN, J. D.; NORMAN, D. A. **Direct Manipulation Interfaces**. In Norman, D. A. and DRAPER, S. W. *User Centered System Design*, Hillsdale: New Jersey. Lawrence Erlbaum Associates, 1986.

[Leite98] LEITE, J. **Modelos e Formalismos para a Engenharia Semiótica de Interfaces de Usuário.** Tese de Doutorado. Departamento de Informática, PUC-RJ, 1998.

[Leite99] LEITE, J.; DE SOUZA, C. **Uma Linguagem de Especificação para a Engenharia Semiótica de Interfaces de Usuário.** II Workshop sobre Fatores Humanos em Sistemas Computacionais. Campinas – SP, 1999.

[Moran81] MORAN, T. **The Command Language Grammars: a representation for the user interface of interactive computer systems.** International Journal of Man-Machine Studies Academic Press, 1981.

[Norman86] NORMAN, D. A. **Cognitive Engineering**. In Norman, D. A. and DRAPER, S. W. *User Centered System Design*, Hillsdale: New Jersey. Lawrence Erlbaum Associates, 1986.

[Norman&Draper86] NORMAN, D. A.; DRAPER, S. W. **User Centered System Design.** Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1986.

[Oracle2000] Oracle Documentation. PL/SQL User's Guide and Reference. Release 8.1.6. Cap.11 – Language Elements.

[Parent92] SPACCAPIETRA, S.; PARENT, C. **ERC+: an object based entity relationshp approach.** In Conceptual Modelling, Databases and CASE: An Integrated View of Information Systems Development. Peri Loucopoulos and Roberto Zicari Eds. John Wiley, 1992.

[Parent98] PARENT, C.; SPACCAPIETRA, S. **Issues and Approaches of Database Integration**. Communication of the ACM, v.41, n. 5, 1998.

[Peirce77] PEIRCE, C.S. **Semiótica**. São Paulo: Ed. Perspectiva, 1977.

[Ramalho99] RAMALHO, J. A. **SQL Server – Iniciação e Referência**. São Paulo: Makron Books, 1999.

[Scopim03] SCOPIM, K. **J-Schemas Integrator – Uma ferramenta para integração de banco de dados heterogêneos distribuídos.** Dissertação de mestrado, Programa de Pós-Graduação em Informática, UFPR, 2003.

[Shneiderman92] SHNEIDERMAN, B. **Designing the user interface: strategies for effective human-computer interaction**, Addinson-Wesley, 2nd edition, 1992.

[Silberschatz01] SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Database System Concepts**. McGraw-Hill, 4 edition, 2001.

[Spaccapietra83] SPACCAPIETRA, S.; DEMO, B.; PARENT, C. **SCOOP: a system for integration existing heterogeneous distributed data bases and application programs**. IEEE INFOCOM Conference, San Diego, 1983.

[Spaccapietra92] SPACCAPIETRA, S.; PARENT, S.; DUPONT, Y. **Model independent assertions for integration of heterogeneous schemas.** The International Journal on Very Large Data Bases J.1 (1), 1992.

[Spaccapietra95] SPACCAPIETRA, S.; PARENT, C.; SUNYE, M.; YETONGNON, K.; LEVA, A. D. **ERC+: An Object+Relationship Paradigm for Database Applications**, IEEE Computers Science Press, 1995.

APÊNDICE I

MODELO GOMS

Meta: Mapear-base-de-dados

Seleção

Meta: Lista-base-de-dados

Mostra lista de todos os componentes da base de dados, hierarquicamente

Selecione um dos componentes da base de dados

Seleção

Meta: Lista-comando-SQL-da-base-de-dados

Meta: Lista-comando-SQL-texto

Mostra em formato texto o comando SQL da base de dados original (legado) do componente selecionado na lista-base-de-dados

Fim-meta-Lista-comando-SQL-texto

Meta: Lista-comando-SQL-grafo

Mostra em forma de grafo o comando SQL da base de dados original (legado) do componente selecionado na lista-base-de-dados

Fim-meta-Lista-comando-SQL-grafo

Fim-meta-Lista-comando-SQL-da-base-de-dados

Meta: Visualiza-trigger-da-base-de-dados

Mostra graficamente o trigger, com os respectivos objetos que são modificados e que modificam o trigger selecionado na lista-base-de-dados

Fim-meta-Visualiza-trigger-da-base-de-dados

Meta: Lista-Dependência-dinâmica

Meta: Lista-Altera

Mostra a lista e a visualização gráfica no modelo ERC+ de todas as entidades que são alteradas por triggers disparados pela entidade selecionada (saída).

Fim-meta-Altera

Meta: Lista-Alterado-por

Mostra a lista e a visualização gráfica no modelo ERC+ de todas as entidades que, através de triggers disparados por elas, alteram a entidade selecionada (entrada).

Fim-meta-Lista-Alterado-por

Fim-meta-Lista-Dependência-dinâmica

Meta: Visualiza-todos-triggers-da-base-de-dados

Executa meta Mapeamento-todos-triggers

Fim-meta-Visualiza-todos-triggers-da-base-de-dados

Fim-Seleção

Fim-meta-Lista-base-de-dados

Meta: Optar-novo-script

Clique no botão Nova Base de Dados

Seleção:

Meta: Entrar-nome-*Script*

Forneça o nome do caminho e do arquivo *script*

Fim-meta-Entrar-nome-*Script*

Meta: Buscar-*Script*

Procure o caminho por meio da navegação de arquivos (estrutura de árvore)

Selecione o arquivo *script* desejado

Fim-meta-Buscar-*Script*

Fim-Seleção

Meta: Entrar-nome-base

Forneça o nome da base de dados

Fim-meta-Entrar-nome-base

Meta: Confirmar

Clique botão Confirmar

Mostre mensagem para aguardar enquanto cria base-de-dados

Mostre mensagem para aguardar enquanto identifica objetos

Mostre mensagem para aguardar enquanto identifica triggers

Feche janela "Fornecer o Script"

Fim-meta-Confirmar

Fim-meta-Optar-novo-script

Meta: Sair-do-sistema

Clique no botão Sair Sistema

Fim-meta-sair-do-sistema

Fim-Seleção

Meta: Mapeamento-todos-triggers

Mostra graficamente todos os trigger da base de dados selecionada na lista-base-de-dados, com os respectivos objetos que são modificados e que modificam cada um dos triggers.

Seleção

Meta: Mais zoom

Clique no botão 'Mais zoom'

Fim-meta-Mais-zoom

Meta: Menos zoom

Clique no botão 'Menos zoom'

Fim-meta-Menos-zoom

Meta: Voltar-mapeamento

Clique no botão 'Voltar Mapeamento'

Feche janela 'Mapeamento-todos-triggers'

Fim-meta-Voltar-mapeamento

Fim-seleção

Fim-meta-mapeamento-todos-triggers

Fim-meta-Mapear-base-de-dados

APENDICE II

LINGUAGEM DE ESPECIFICAÇÃO DA MENSAGEM DO *DESIGNER*

Task-Message *Fornecendo o Script* (execução)

Sequence

View (título) "Fornecendo o Script"

Join

Select

Enter information-of Script

Activate Start for Application-Function Localizar...

View (orientação) "Ex.: c:\arquivos\nomedoarquivo.sql"

Enter information-of Base de Dados

View (orientação) "Forneça um nome para identificar a Base de Dados"

Join

Desactivate Start for Application-Function Confirmar

Activate Start for Application-Function Cancelar

Activate Stop for Application-Function Sair do Sistema

Task-Message *Fornecendo o Script (avaliação)*

Sequence

Select

Resposta sucesso

Activate Start for Application-Function Confirmar

Resposta fracasso

Select

Join

View (título) "Atenção"

View (mensagem fracasso) "Não foi possível localizar o arquivo script informado. Entre com novo arquivo script ou utilize o botão 'Localizar...' para encontrar o arquivo script desejado."

Activate Start for Application-Function Ok

Join

View (título) "Atenção"

View (mensagem fracasso) "Já existe uma base de dados com o nome informado. Escolha outro nome para identificar a base de dados."

Activate Start for Application-Function Ok

Join

View (título) "Atenção"

View (mensagem fracasso) "O preenchimento do nome do arquivo script é obrigatório. Entre com o nome do arquivo script desejado ou utilize o botão 'Localizar...' para encontrar o arquivo desejado."

Activate Start for Application-Function Ok

Join

View (título) "Atenção"

View (mensagem fracasso) "O preenchimento do nome da base de dados é obrigatório. Entre com um nome para identificar a base de dados."

Activate Start for Application-Function Ok

Command-Message *Comando SQL* **for Application-function** *Texto*

Join

Select information-of Texto

Command-Message *Comando SQL* **for Application-function** *Grafo*

Join

Select information-of Grafo

Command-Message *Mapeamento* **for Application-function** *Comando SQL*

Join

Select

Command-Message Texto

Command-Message Grafo

Command-Message *Mapeamento* **for Application-function** *Visualizar*

Join

Select information-of Visualizar

Command-Message *Dependência Dinâmica* **for Application-function** *Altera*

Join

Select information-of Entidade

Select information-of Visualização

Command-Message *Dependência Dinâmica* **for Application-function** *Alterado por*

Join

Select information-of Entidade

Select information-of Visualização

Command-Message *Mapeamento* **for Application-function** *Dependência Dinâmica*

Join

Select

Command-Message Altera

Command-Message Alterado por

Task-Message *Mapeamento* (execução)

Sequence

View (título) "Mapeamento"

Join

View(orientação) "Base selecionada: "

Select information-of Base selecionada

Select information-of Base de Dados

Join

Select

Command-Message Comando SQL

Command-Message Visualizar

Command-Message Dependência Dinâmica

Activate Start for Application-Function Visualizar todos os Triggers

Activate Start for Application-Function Nova Base de Dados

Activate Stop for Application-Function Sair do Sistema

Task-Message *Visualização de Triggers* (execução)

Sequence

View (título) "Visualização de Triggers"

Join

Select information-of Visualização de Triggers

Activate Start for Application-Function Voltar Mapeamento

Activate Start for Application-Function Mais zoom

Activate Start for Application-Function Menos zoom

APÊNDICE III

CORRELAÇÃO MENSAGEM-*WIDGETS* DA INTERFACE

Ação	*Widget*	Exemplo
Activate	Botão	Botão disponível para acionamento de uma determinada função
Command-Message	Pastas com abas	Agrupamento visual
Desactivate	Botão	Botão inibido para acionamento de uma determinada função
Enter	Campos de texto, número ou outros caracteres na cor branca	Campos através dos quais o usuário fornece informações para o sistema (entrada de dados)
Join	Caixa de diálogo, painel de controle, formulário	Agrupamento visual dos *widgets*
Select	Campos para entrada de dados, botões, pastas com abas, lista de seleção contidas em Caixa de diálogo, painel de controle, formulário.	Agrupamento visual dos *widgets* para seleção de interações ou informações
Select information-of	*Labels*, lista de seleção	Visualização das informações de domínio somente para leitura
Sequence	Caixa de diálogo, painel de controle	Indica seqüência de execução das tarefas
Task-Message	Janela, Tela, Formulário	Especificação de uma mensagem de tarefa

Trigger (visualização)	Elipse com o nome do *trigger* em seu interior	Indica a existência de *triggers* no modelo ERC+
Trigger (comando SQL - texto)	Texto em letra minúscula na cor preta	Indica o nome do *trigger* na aba "comando SQL" no formato texto
Trigger (comando SQL – grafo)	Para palavras reservadas utilizar retângulo; pontuação utilizar círculo; outras informações utilizar retângulo com cantos arredondados.	Visualiza o comando SQL em formato de grafo (diagrama de sintaxe, anexo I)
Trigger – condição (visualização)	Linha cheia azul	Relaciona o *trigger* com a entidade que possui condição(es) para sua execução
Trigger – condição (comando SQL – texto)	Texto em letra minúscula na cor azul	Indica os atributos que participam como condição a serem satisfeitas para execução do *trigger*
Trigger – condição (comando SQL – grafo)	Retângulo azul com cantos arredondados, com a condição no interior em texto na cor preta	Visualiza o comando SQL em formato de grafo (diagrama de sintaxe, anexo I)
Trigger – origem (visualizar)	Seta na cor preta apontando para o *trigger*	Entidade que dispara o *trigger*
Trigger – ação (visualizar)	Seta na cor verde apontando para a entidade que sofre modificação	Relaciona o *trigger* com a entidade que sofre modificação
Trigger – ação (comando SQL - texto)	Texto em letra minuscula na cor verde	Indica os atributos que são modificados pela execução do *trigger*

Trigger – ação (comando SQL – grafo)	Retângulo verde com cantos arredondados, com o bloco de comandos (ação) no interior em texto na cor preta	Visualiza o comando SQL em formato de grafo (diagrama de sintaxe, anexo I)
View (mensagem fracasso)	Texto	Texto destacando insucesso. Está associado a uma nova janela, tela ou formulário com a mensagem de fracasso.
View (orientação)	Texto	Texto destacando um *label* ou uma informação para auxiliar na entrada de dados
View (título)	Texto centralizado na parte superior do formulário, na cor azul	Texto indicando o título de cada janela, tela ou formulário

APENDICE IV

TELAS DO SISTEMA

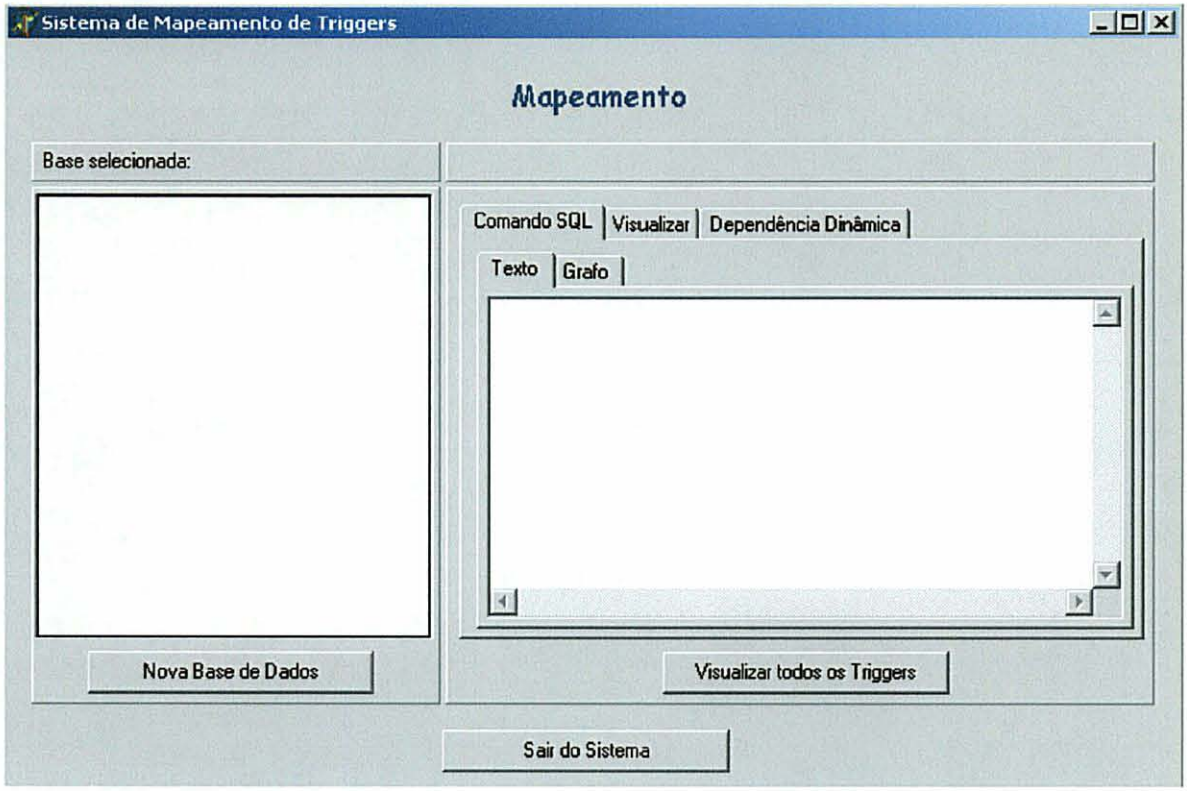


FIGURA A1 – TELA PARA VISUALIZAÇÃO E INFORMAÇÕES DO MAPEAMENTO DE *TRIGGERS*

FIGURA A2 – TELA DE ENTRADA DE DADOS DAS INFORMAÇÕES DA BASE DE DADOS A SER MAPEADA

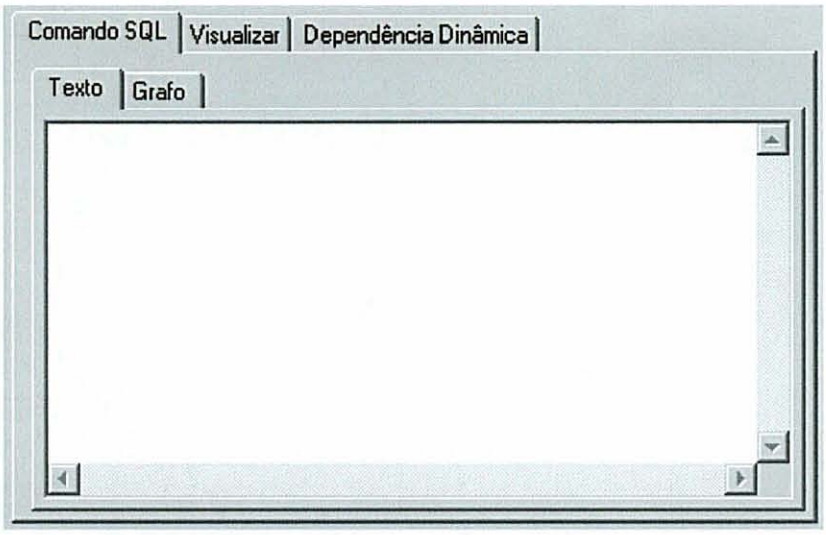


FIGURA A3 – DETALHE DA TELA DE MAPEAMENTO DE *TRIGGERS* PARA VISUALIZAR O COMANDO SQL EM FORMATO TEXTO

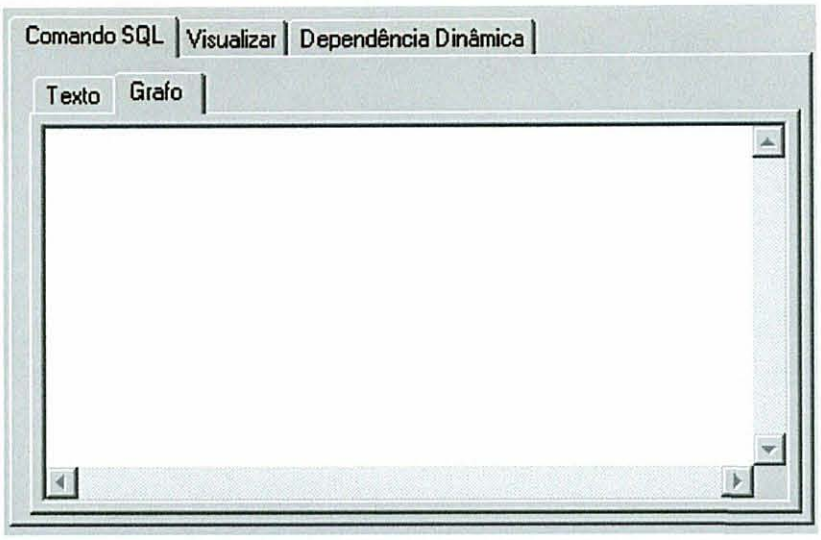


FIGURA A4 – DETALHE DA TELA DE MAPEAMENTO DE *TRIGGERS* PARA VISUALIZAR O COMANDO SQL EM FORMATO DE GRAFO

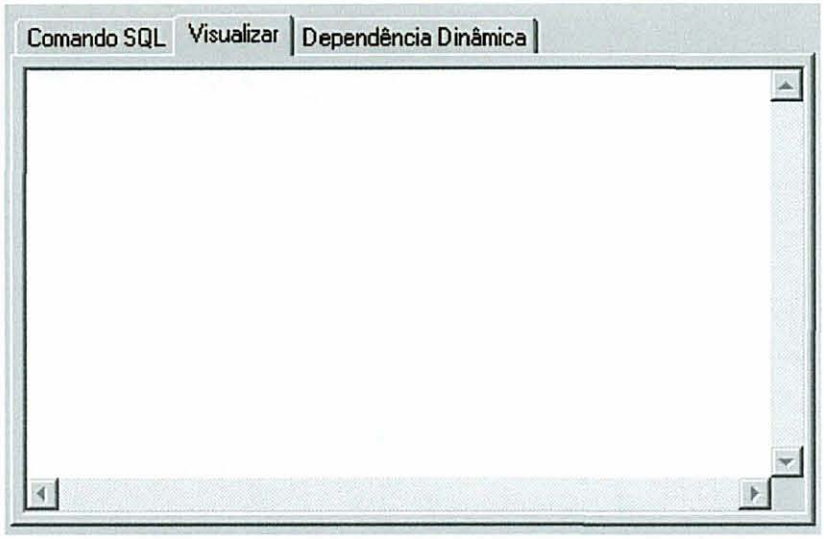


FIGURA A5 – DETALHE DA TELA DE MAPEAMENTO DE *TRIGGERS* PARA VISUALIZAR UM *TRIGGER* SELECIONADO GRAFICAMENTE

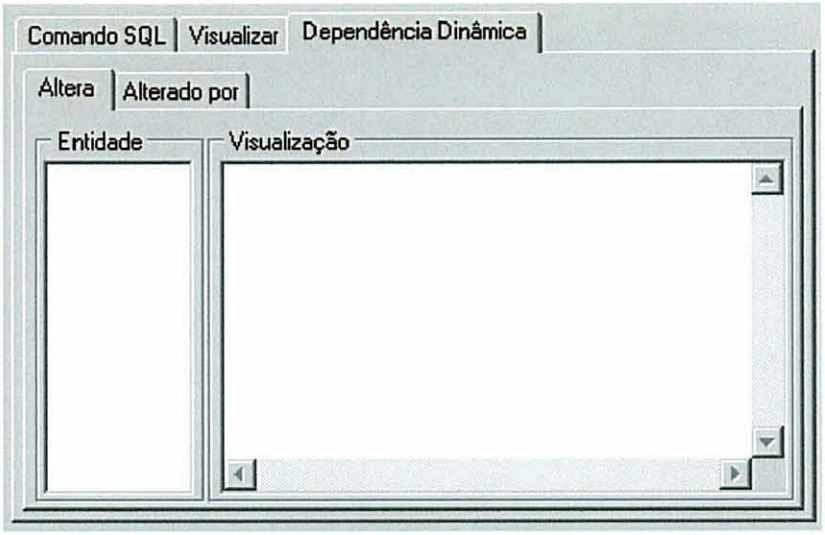


FIGURA A6 - DETALHE DA TELA DE MAPEAMENTO DE TRIGGERS PARA LISTAR E VISUALIZAR GRAFICAMENTE AS ENTIDADES QUE SÃO ALTERADAS PELA ENTIDADE SELECIONADA

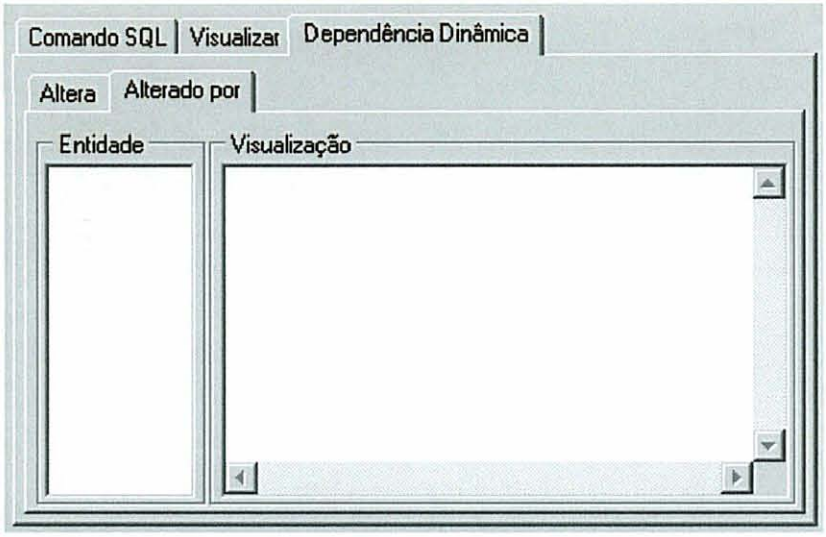


FIGURA A7 - DETALHE DA TELA DE MAPEAMENTO DE TRIGGERS PARA LISTAR E VISUALIZAR GRAFICAMENTE AS ENTIDADES QUE ALTERAM UMA ENTIDADE SELECIONADA

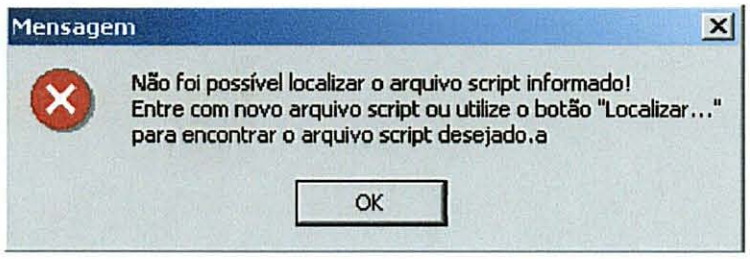


FIGURA A8 – MENSAGEM DE PREENCHIMENTO INCORRETO DO CAMPO *SCRIPT* NA TELA FORNECENDO O SCRIPT

FIGURA A9 – MENSAGEM DE PREENCHIMENTO INCORRETO DO CAMPO BASE DE DADOS NA TELA FORNECENDO O SCRIPT

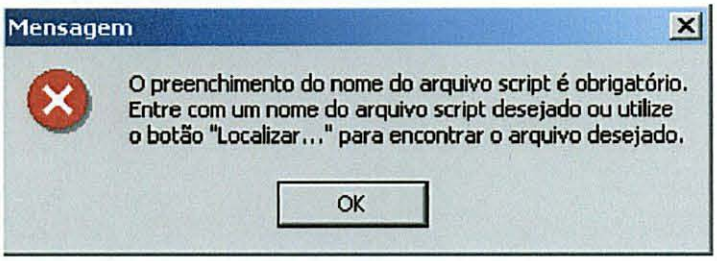


FIGURA A10 – MENSAGEM DE FALTA DE PREENCHIMENTO DO CAMPO *SCRIPT* NA TELA FORNECENDO O SCRIPT

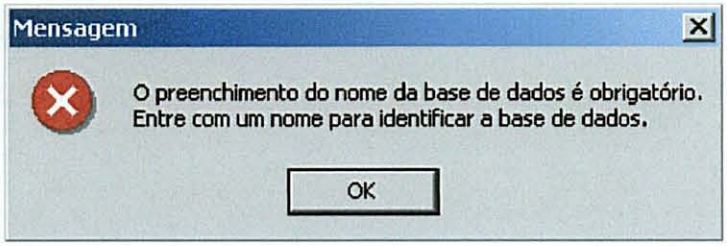


FIGURA A11 – MENSAGEM DE FALTA DE PREENCHIMENTO DO CAMPO BASE DE DADOS NA TELA FORNECENDO O SCRIPT

FIGURA A12 – TELA PARA VISUALIZAÇÃO GRÁFICA DE TODOS OS TRIGGERS QUE COMPÕE UMA BASE DE DADOS

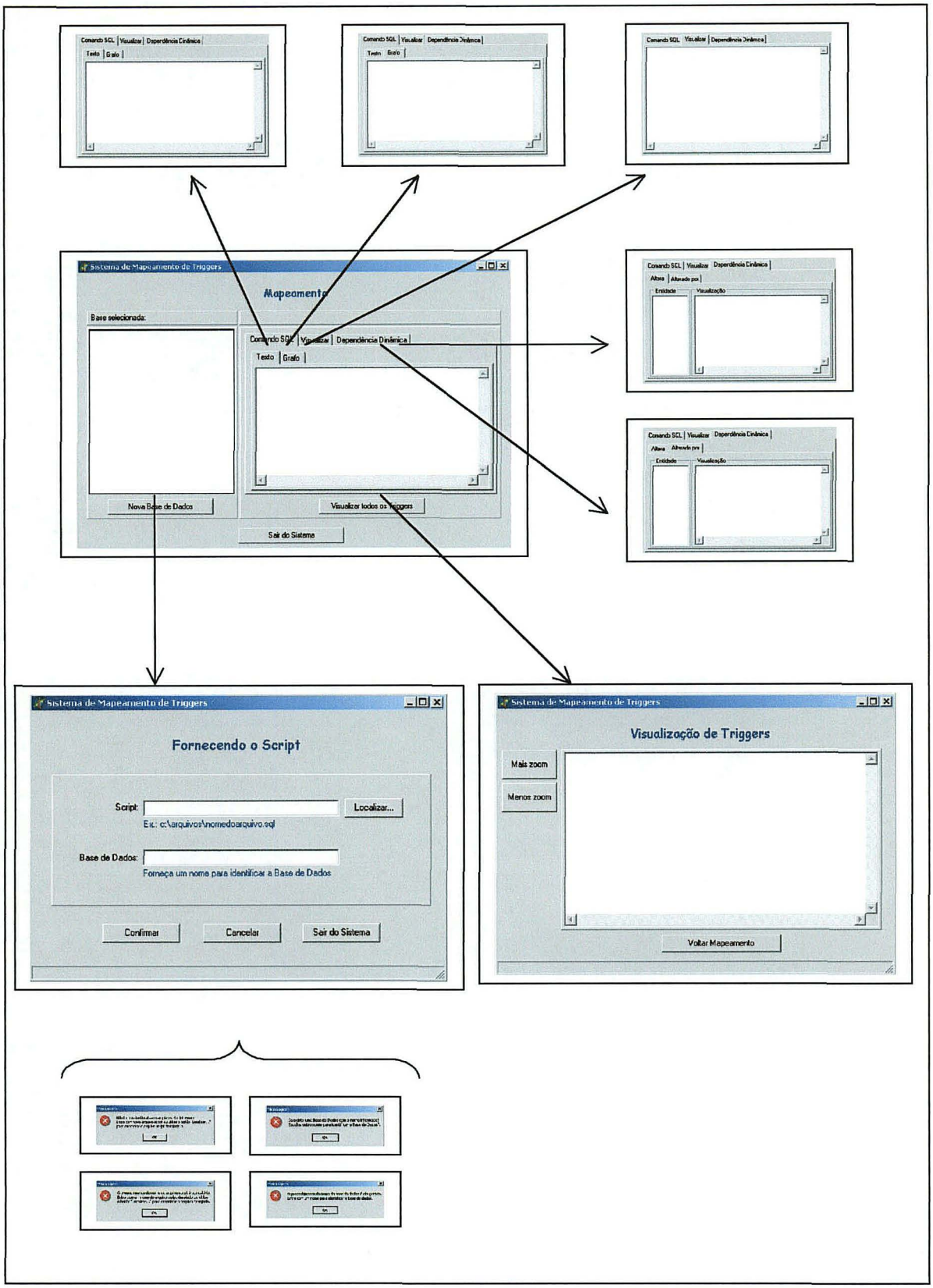


FIGURA A13 – DIAGRAMA DE SEQÜÊNCIA DE TELAS DO SISTEMA DE MAPEAMENTO DE *TRIGGERS*

ANEXO I

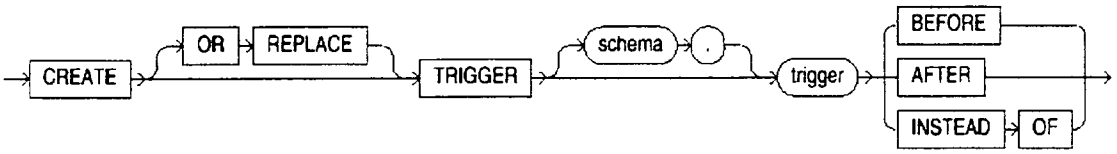
DIAGRAMAS DE SINTAXE DE *TRIGGERS* EM ORACLE

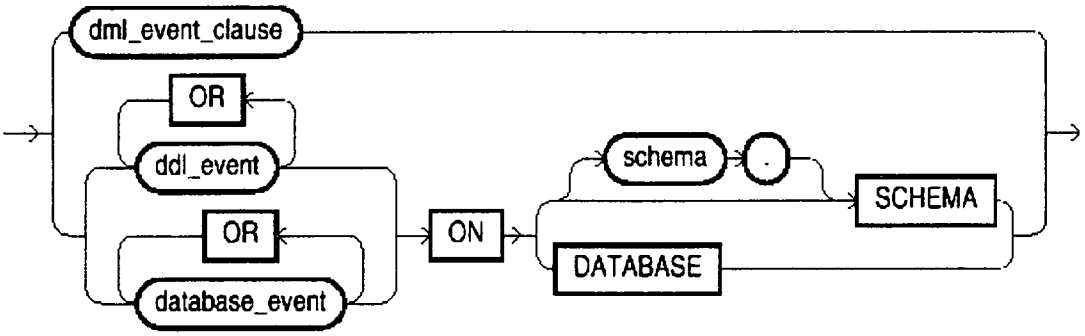
Fonte: Oracle Documentation (Oracle2000)

Os diagramas de sintaxe tem por objetivo auxiliar a construir ou verificar instrução PL/SQL. Nos diagramas, palavras reservadas são delimitadas por caixas (retângulos), delimitadores por círculos, e identificadores por elipses. As instruções devem ser seguidas passo-a-passo no diagrama de sintaxe, sempre da esquerda para a direita, de cima para baixo, seguindo a direção das setas. Os diagramas de sintaxe permitem ainda o uso de *loops*.

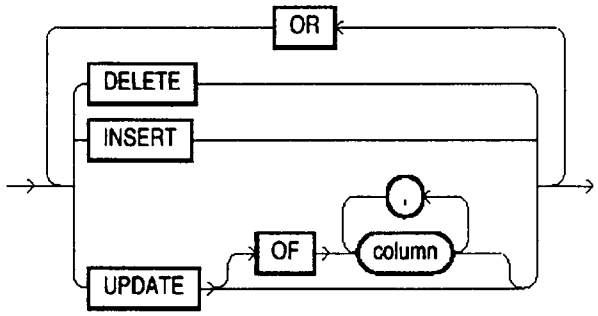
A seguir são apresentados os diagramas de sintaxe para criar, ativar, desativar, compilar e remover *triggers* em Oracle (Oracle2000), o detalhamento dos elementos de cada diagrama, e exemplos de utilização de instruções SQL em Oracle, utilizando *triggers*.

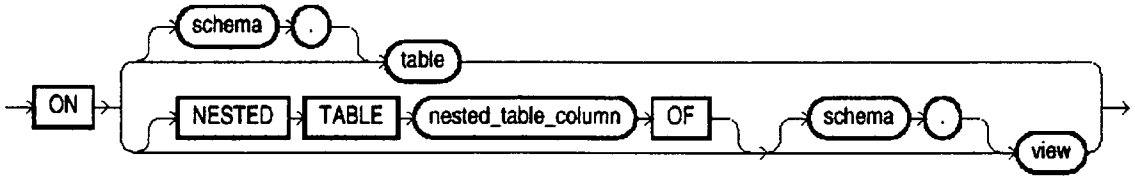
Sintaxe para criar *triggers em Oracle*:



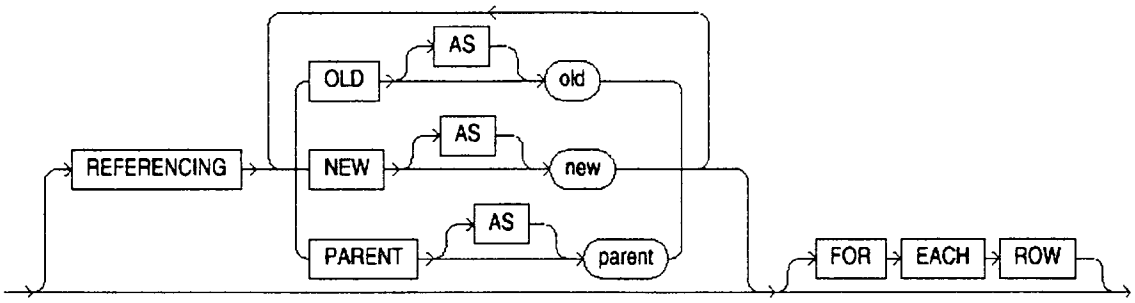


dml_event_clause::=





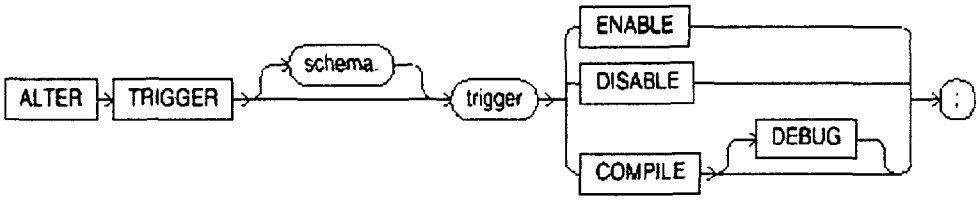
referencing_clause::=



CREATE	Cria um novo *trigger*
OR REPLACE	Recria um *trigger* sem destruir os privilégios associados
Schema	É o esquema que irá conter o *trigger*. Se for omitido, o Oracle cria o *trigger* no esquema do usuário atual
BEFORE	Faz com que o Oracle dispare o *trigger* antes de executar o evento do *trigger*. Não é possível especificar um *trigger* desse tipo em uma *view* ou *object view*.
AFTER	Faz com que o Oracle dispare o *trigger* depois de executar o evento do *trigger*. Não é possível especificar um *trigger* desse tipo em uma *view* ou *object view*.
INSTEAD OF	Faz com que o Oracle dispare o *trigger* ao invés de executar o evento do *trigger*. Essa cláusula só é válida para visões. Não é permitido especificar um *trigger instead of* em uma tabela
dml_event_clause	Especifica um dos três comandos DML que podem disparar um *trigger*: *DELETE, INSERT* e *UPDATE*. A cláusula *update* pode ser seguida da palavra *of* e de uma lista de colunas, especificando que o *trigger* deve ser disparado se uma das colunas for modificada. Não é permitido especificar *of* com *update* para um *trigger* do tipo *instead of*.
ddl_event	Especifica um dos três comandos DDL que podem disparar um *trigger*. Os *triggers* para esses eventos só podem ser definidos em um esquema ou banco de dados.

database_event	Descreve um estado particular do banco de dados que pode causar o disparo do *trigger*. Os *triggers* para esses eventos só podem ser definidos em um esquema ou banco de dados. *SERVERERROR* *LOGON* *LOGOFF* *STARTUP* *SHUTDOWN* Os *triggers AFTER STARTUP* e *BEFORE SHUTDOWN* aplicam-se somente a banco de dados
ON	Determina o objeto do banco de dados no qual o *trigger* deve ser definido. O objeto pode ser uma tabela, uma *view*, uma *nested table*, um esquema ou o próprio banco de dados
referencing_clause	Especifica nomes de correlações: *NEW, OLD* e *PARENT*
FOR EACH ROW	Faz com que o *trigger* seja do tipo *row trigger*
WHEN	Especifica a restrição ou condição do *trigger*. Esta cláusula só é permitida para *triggers* do tipo *row triggers*. Não é possível especificar restrições para *triggers* do tipo *INSTEAD OF*.
pl / sql_block	Especifica a ação do *trigger* como um bloco de código em PL/SQL
call_procedure_ statement	Uma chamada a uma stored procedure.

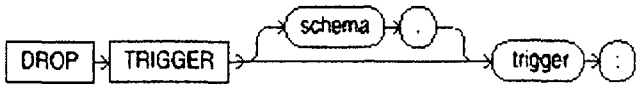
Sintaxe para ativar, desativar e compilar *trigger em Oracle*:



O comando *ALTER TRIGGER* é utilizado para ativar, desativar e compilar um *trigger*. Este comando não muda a declaração ou definição de um *trigger* existente. Para redefinir um *trigger*, utiliza-se o comando *CREATE TRIGGER* com *OR REPLACE*.

schema	É o esquema que contém o *trigger*. Se for omitido, o Oracle assume que o *trigger* está no esquema do usuário atual
trigger	É o nome do *trigger* a ser alterado
ENABLE	Ativa o *trigger*. Para ativar todos os *triggers* de uma tabela, pode-se usar a cláusula EN*ABLE ALL TRIGGERS* do comando *ALTER TABLE*
DISABLE	Desativa o *trigger*. Para desativar todos os *triggers* de uma tabela, pode-se usar a cláusula *DISABLE ALL TRIGGERS* do comando *ALTER TABLE*
COMPILE	Compila explicitamente o *trigger*. A compilação explícita elimina a necessidade de compilação implícita em tempo de execução e previne erros em tempo de execução e overhead de performance.
DEBUG	Instrui o compilador PL/SQL a gerar e guardar o código para ser usado pelo *debugger*.

Sintaxe para remover *trigger em Oracle*:



O comando *DROP TRIGGER* é utilizado para remover um *trigger* do banco de dados.

Schema	É o esquema que contém o *trigger*. Se for omitido, o Oracle assume que o *trigger* está no esquema do usuário atual
Trigger	É o nome do *trigger* a ser removido

Exemplos:

a) Criar um *trigger* do tipo *before* statement cujo nome é *emp_permit_changes* no esquema *scott*

```
CREATE TRIGGER scott.emp_permit_changes
        BEFORE
        DELETE OR INSERT OR UPDATE
        ON scott.emp
        pl / sql_block
```

b) Criar um *trigger* do tipo *before row*. O Oracle dispara esse *trigger* quando um dos seguintes comandos é emitido:

- *Insert* que insere linhas na tabela *emp*
- *Update* que modifica valores nas colunas *sal* ou *job* da tabela *emp.*

```
CREATE TRIGGER scott.salary_check
        BEFORE
        INSERT OR UPDATE OF sal, job
        ON scott.emp
        FOR EACH ROW
        WHEN (new.job <> 'PRESIDENT')
                pl / sql_block
```

c) Criar um *trigger* que chama uma *stored procedure* ao invés de especificar a ação em um bloco de código PL/SQL.

```
CREATE TRIGGER scott.salary_check
        BEFORE
        INSERT OR UPDATE OF sal, job
        ON scott.emp
        FOR EACH ROW
        CALL  check_sal (:new.job, :new.sal, :new.ename);
```

d) Criar um *trigger* para um evento do banco de dados

```
CREATE TRIGGER log_errors AFTER SERVERERROR ON DATABASE
        BEGIN
                IF (IS_SERVERERROR (1017)) THEN
                        <special processing of logon error>
                ELSE
                        <log error number>
                END IF;
        END;
```

e) Criar um *trigger* do tipo INSTEAD OF

```
CREATE TRIGGER customers_sj
        (cust NUMBER(6),
         address VARCHAR2(50),
         credit NUMBER(9,2) );
CREATE TRIGGER customers_pa
        (cust NUMBER(6),
         address VARCHAR2(50),
         credit NUMBER(9,2) );
```

```
CREATE TYPE customers_t AS OBJECT
        (cust NUMBER(6),
         address VARCHAR2(50),
         credit NUMBER(9,2),
         location VARCHAR2(20) );
CREATE VIEW all_customers (cust)
        AS SELECT customers_t (cust, address, credit, 'SAN JOSE')
        FROM customers_sj
        UNION ALL
        SELECT customers_t (cust, address, credit, 'PALO_ALTO')
        FROM customers_pa;
CREATE TRIGGER instrig INSTEAD OF INSERT ON all_customers
        FOR EACH ROW
        BEGIN
                IF (:new.cust.location = 'SAN JOSE') THEN
                        INSERT INTO customers_sj
                        VALUES (:new.cust.cust, :new.cust.address, :new.cust.credit);
                ELSE
                        INSERT INTO customers_pa
                        VALUES (:new.cust.cust, :new.cust.address, :new.cust.credit);
                END IF;
        END;
```