UM SISTEMA DE INTERFACE COOPERATIVO PARA MODELAGEM EM BANCO DE DADOS

Luís Eduardo Saraiva Câmara Décio Fonseca
Departamento de Informática - UFPE
Caixa Postal 7851, 50739 Recife - PE

RESUMO

Este artigo apresenta um sistema de interfaces cooperativo e inteligente desenvolvido para facilitar o processo de modelagem de aplicações de Banco de Dados utilizando o modelo de dados E/D [1]. A ênfase maior está no tratamento de erros que possibilita o uso do modelo por um grande número de usuários de diferentes níveis de conhecimento

1 - INTRODUÇÃO

A necessidade de interfaces inteligentes para ambientes de concepção de Banco de Dados englobando ferramentas diversas e utilizados por usuários de diferentes níveis de conhecimento para permitir o uso simplificado e adaptável dos ambientes, se torna a cada dia mais vital [8, 12, 14, 15]. Pesquisadores de áreas diversas como Inteligência Artificial, Banco de Dados, Engenharia de Software, Psicologia Cognitiva e Lingüística Computacional, concordam que cada vez mais se torna necessário a existência de interfaces cooperativas que se ajustem ao conhecimento das pessoas que utilizam o ambiente [5, 9, 16]. Existe, também, uma corrente de pensamento que defende o desenvolvimento de tais interfaces através de uma cooperação interdisciplinar.

Muitos dos sistemas desenvolvidos no passado e que são utilizados atualmente, apresentam o problema da adaptabilidade das interfaces ao nível de compreensão do usuário em relação às funções do sistema. Nesses sistemas, o tratamento de erros é muito precário. Não existe a preocupação de interpretar o que foi solicitado por um comando não totalmente correto. A indicação do erro, geralmente, se restringe a mensagens e códigos de retorno cujo significado, muitas vezes, não é compreensível à maioria dos usuários.

Um método bastante comum para obter informações e/ou realizar operações, é a utilização de uma Interface de Linguagem

de Comandos. Tal interface é obtida através de uma linguagem formal que permite ao usuário expressar de forma não ambígua suas requisições ao sistema. O grande problema está no fato que este tipo de interface implica em uma grande familiaridade com a sintaxe da linguagem de comandos e com as funções que o sistema oferece. Para usuários leigos ou com pouca experiência, as interfaces tradicionais de Linguagem de comandos demonstram ser bastantes ineficientes.

Com a intenção de resolver estes problemas, diversos tipos de interfaces para usuários de menor grau de experiência têm sido desenvolvidos. Interfaces Gráficas, Interfaces baseadas em menus, Interfaces de linguagem Semi-natural eliminam ou suavizam as dificuldades existentes na comunicação entre os usuários leigos e o sistema. Estas interfaces requerem pouca sofisticação computacional e todo comando é solicitado pelo usuário, através de opções ou do preenchimento de respostas pré-definidas. A necessidade de um conhecimento prévio das funções do sistema é eliminada, uma vez que elas são apresentadas sempre que necessário. Os erros são diminuídos, visto que tais interfaces são projetadas de maneira a guiar a interação e sua indicação é feita de forma mais amigável.

Entretanto, à medida que usuários leigos adquirem experiência sobre o sistema através de sua utilização, torna-se claro que comandos relativamente simples exigem um longo processo de escolhas e respostas para serem realizados. Para estes usuários com uma nova visão sobre o sistema bem como para aqueles que já são especialistas, tais interfaces são consideradas monótonas e requerem muito tempo para a execução de comandos.

Na tentativa de se obter uma interface que sirva de maneira satisfatória tanto para usuários leigos como para usuários especialistas, pesquisas têm sido direcionadas para a interface de Linguagem natural. Claramente uma linguagem de comandos é uma sublinguagem da linguagem natural, e desta forma Interfaces de Linguagem Natural não apresentam inconvenientes para os especialistas do sistema. Para os usuários leigos, o fato de poder dialogar com o sistema da mesma forma em que se expressam no dia-a-dia, torna bem mais amigável sua comunicação com o

sistema. Embora interface de linguagem natural seja uma boa alternativa para resolver o problema existente quando um sistema possui uma variedade de tipos de usuário, é reconhecido que o processamento de linguagem natural, na sua forma geral (principalmente, quando expressa por usuários leigos) requer um grande investimento de pesquisa, além do fato da língua natural ser, ela própria, ambígua.

O nosso trabalho constituiu-se na definição e implementação de um Sistema de interfaces interativo e convergente para a concepção de aplicações de Banco de Dados utilizando o Modelo Dinâmico e Estático (Modelo E/D). Dentro deste processo convergente, está embutida uma metodologia para a concepção interativa, evolutiva e cooperativa de Banco de Dados. A grande ênfase, nesta direção, está no tratamento de erros. Acreditamos que o tratamento inteligente e cooperativo de erros permite, principalmente, que :

- O ambiente seja utilizado por usuários de diferentes níveis de conhecimento;
- Usuários com experiência em outros modelos conceituais de dados, trabalhem satisfatoriamente com o modelo E/D;
- O manuseio do modelo E/D constitua-se, ele próprio, em um processo gradativo de aprendizado

O trabalho está dividido em 4 seções, das quais esta é a primeira. Na próxima seção são descritas as principais características do modelo Estático e Dinâmico. Na seção 3 é apresentado o Sistema de Interface com suas características cooperativas e convergentes. Finalmente, na última seção, são descritas algumas conclusões sobre o trabalho e aspectos necessários para a sua continuação.

2 - O MODELO ESTÁTICO DINÂMICO (E/D)

O modelo Estático e Dinâmico (E/D) é uma extensão do modelo Entidade / Relacionamento [7] com características de orientação a objetos. O modelo E/D foi desenvolvido na UFPE e integra o projeto POESIS cuja meta é a construção de um protótipo de ambiente de desenvolvimento de Banco de Dados multimídia [6]. O modelo E/D tem como objetivo principal tratar e especificar, de forma integrada, as propriedades estáticas, relativas aos estados (aspecto estrutural), as propriedades dinâmicas relativas às

transações de estado (aspecto comportamental) e o conjunto de estados válidos (restrições de integridade) dos sistemas de informação [1, 13].

O modelo E/D trata todos os seus conceitos como objetos. Cada objeto identificado no processo de modelagem é especificado através de uma Linguagem de Definição de Objetos (LDO). Os conceitos podem ser agrupados em tres grupos: Representação estrutural do modelo (Entidade, Relacionamento, Atributo, Domínio e respectivas restrições de integridade); Propriedades Dinâmicas (Ação e Regra de Comportamento) e propriedades de Controle (Transação e Meta-regra).

ENTIDADE é o objeto básico do modelo E/D. Suas propriedades estruturais são representadas pelos atributos e as comportamentais por ações e regras de comportamento. Uma classe entidade tanto pode herdar propriedades de outras classes entidades (herança) quanto ser definida a partir de outras classes (Generalização Especialização). O modelo comporta herança múltipla e herança seletiva [2]. Uma classe entidade deve possuir, no mínimo, uma propriedade que a identifique (atributo determinante). Sua especificação compreende: Identificação, Descrição, Atributos, Restrições de Integridade, Ações, Regras Comportamentais, Generalização e Especificação.

ATRIBUTO representa as propriedades relativas às classes entidades e às associações entre elas (relacionamento). É um objeto que está associado a um conjunto de valores (instâncias). A especificação de um objeto atributo compreende: Identificação, Descrição, Tipo (simples ou composto), Multivaloração, Domínio, Sinônimo, Edição, Restrição e Determinante.

DOMÍNIO representa um contexto ou uma estrutura sob a qual um atributo está definido. Pode ser do tipo mais básico (inteiro, caracter, boleano) como de estruturas mais complexas (voz, imagem, registro). A especificação de cada objeto domínio compreende: Identificação, Descrição e Tipo.

RELACIONAMENTO é uma estrutura que indica as associações entre instâncias de uma classe entidade com outra, ou ainda, com ela mesma (auto-relacionamento). É um objeto dependente, ou seja, sua existência só faz sentido se for para representar associações entre classes entidades. O relacionamento é um objeto que, embora

possa ter propriedades próprias, não pode herdar propriedades de classes entidades ou de outros relacionamentos. Sua especificação compreende: Identificação, Descrição, Participantes, Atributos próprios, Restrições de Integridade, Ações, Regras de Comportamento, Forma de associação (Total ou Parcial).

RESTRIÇÕES DE INTEGRIDADE evitam que os objetos fiquem em um estado incorreto ou inconsistente. O modelo E/D trata as restrições de integridade como um conjunto de objetos. Através deste conjunto, asserções são definidas sobre os objetos entidade, atributo e relacionamento. As restrições de integridade são classificadas em: Pré-definidas (Dependência Funcional, Unicidade, Não Nulidade, Dependência de Inclusão Excludente), Verificação e Cardinalidade.

AÇÃO permite descrever os fatos que modificam os estados dos objetos entidade e relacionamento através de um conjunto de operações básicas (inserção, Modificação, Recuperação, etc). Sua especificação integrada às propriedades estruturais aumenta a capacidade de representação do modelo. O objeto Ação é composto de: Identificação, Descrição, Entidade e/ou Relacionamento e a Ação.

REGRA DE COMPORTAMENTO representa as propriedades comportamentais dos objetos entidade e relacionamento. Este objeto permite exprimir uma ação condicional composta de uma regra que envolve uma condição para que o fato/ação ocorra e a ação propriamente dita. Sua especificação compreende: Identificação, descrição, Entidade e/ou Relacionamento e a regra.

TRANSAÇÃO caracteriza um procedimento atômico envolvendo duas ou mais ações e/ou regras de comportamento que modificam os objetos entidade e relacionamento. Sua especificação compreende: Identificação, Descrição, Entidade e/ou Relacionamento e Conjunto de Procedimentos.

META-REGRA representa o mecanismo de controle na aplicação das regras de comportamento. O objeto meta-regra permite definir prioridades de execução entre as regras, controlar a execução de regras mutuamente excludentes e controlar a dependência de uma ou mais regras em relação a outras. Sua especificação compreende: Identificação, Descrição e Conjunto de Controle.

3 - O SISTEMA DE INTERFACES

O Sistema de Interface cooperativo, cuja arquitetura é apresentada na figura 1, foi desenvolvido utilizando a linguagem PROLOG e compreende, basicamente, cinco módulos: Meta-base de Conhecimento, Modelo do Usuário, Analisador Sintático, Aquisição de Conhecimento, Explicativo. Estes módulos, atuando em conjunto, são capazes de fornecer ao usuário do modelo E/D as seguintes facilidades:

- Adaptação da interface ao usuário;
- Definição e evolução do perfil do usuário;
- Explicações sempre que requisitado;
- Tratamento amigável de erros;
- Aquisição de conhecimento.

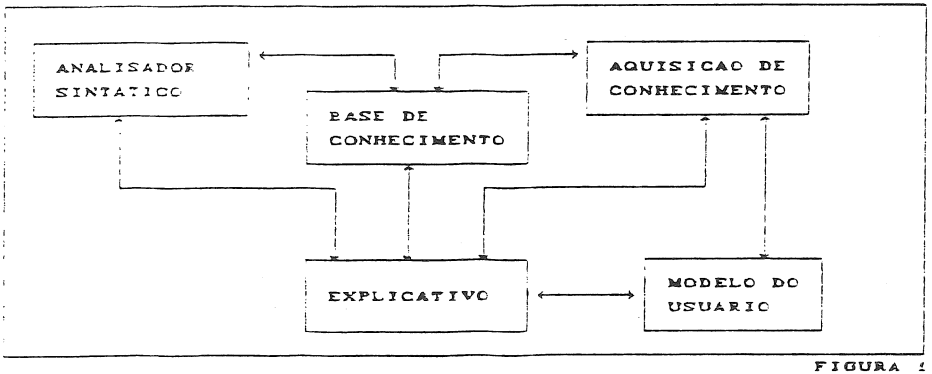

FIGURA 1

Antes de descrever com maiores detalhes as funções do Sistema de Interface, será apresentada a estrutura da Meta-base de conhecimento, uma vez que ela serve como fonte de alimentação para a maioria das funções.

3.1 - META-BASE DE CONHECIMENTO

A meta-base de conhecimento é uma base de regras que visa representar o conhecimento necessário ao Sistema de Interfaces cooperativo para a modelagem de Banco de Dados (modelo E/D, usuários e aplicações). Os objetos da meta-base são modelados a partir do próprio modelo E/D.

A meta-base de conhecimento é composta pelos objetos: entidade, relacionamento, atributo Estas informações torna

possível o funcionamento do tratamento de erros, definição do perfil, explicações e serve como alimentação para o analisador sintático. Na figura 2 são apresentadas as definições dos principais objetos que serão necessários no contexto do trabalho.

O objeto entidade COMANDO é composto de quatro atributos. O atributo *COMANDO* pode ser dividido em dois subatributos. Isto ocorre em virtude de alguns comandos E/D necessitarem de duas palavras chaves para serem identificados. O atributo *MAX-PARA* estabelece o número máximo de parâmetros do comando. O atributo *QTD-PRE* indica os códigos dos comandos que são pré-condições e que devem ser satisfeitas para a execução do comando. É um atributo multivalorado.

O objeto-relacionamento PARAMETRO é responsável pela formação completa do comando. A partir de seus dados, o Analisador Sintático é capaz de, através de casamento de padrões, assegurar a corretude de um comando enviado. Sua concepção é baseada na Forma Normal de Backus - BNF em que cada expressão é quebrada em subexpressões para uma melhor representação.

```
OBJETO-ENTIDADE   : COMANDO
DESCRICAO         :/* Entidade representando os comando do
                  modelo E/D
ATRIBUTOS         : COMANDO, CODIGO, MAX-PARA, QTD-PRE
FIM-OBJETO-ENTIDADE

OBJETO-RELACIONAMENTO : PARAMETRO
DESCRICAO             :/* Autorelacionamento, descrevendo os
                      parametros de cada comando E/D
PARTICIPANTES         : CLASSE-COMANDO(CODIGO)
ATRIBUTOS             : SEQUENC, OPCIONAL, PAL-RES, STRING,
                        OCOR-MUL, CARC-ESP
RESTRICAO             : OBJETO-CARDINALIDADE(UM-PARA-MUITOS)
RELACIONAMENTO-TOTAL  : NAO
FIM-OBJETO-RELACIONAMENTO

OBJETO-RELACIONAMENTO : SINONIMO
DESCRICAO             :/* Relacionamento, associando cada
                      comando E/D a um sinonimo
PARTICIPANTES         : CLASSE-COMANDO(CODIGO)
ATRIBUTOS             : SINONIMO
RESTRICAO             : OBJETO-CARDINALIDADE(UM-PARA-UMO
RELACIONAMENTO-TOTAL  : SIM(CLASSE COMANDO)
FIM-OBJETO-RELACIONAMENTO
```

FIGURA 2

A conjunção *CODIGO* e *SEQUENC* identifica qual a posição do parâmetro em relação ao comando. O atributo *OPCIONAL* indica se o

parâmetro é ou não opcional. O atributo *PAL-RES* é preenchido se o parâmetro for uma palavra reservada da sintaxe do comando, ou seja, não é uma informação fornecida pelo usuário. De forma inversa, o atributo *STRING* informa qual a categoria da informação dada. Entre as categorias válidas estão: atributo, entidade, relacionamento, índice, variável ou código do subcomando, caso este parâmetro corresponda a uma expressão. Expressões são tratadas da mesma forma que comandos. A indicação se o parâmetro é de ocorrência múltipla ou não, é feita pelo objeto-atributo *OCOR-MUL*. Dependendo da BNF do comando, existem alguns parâmetros que são caracteres especiais. O conjunto de caracteres especiais é representado pelo objeto-atributo *CARC-ESP*. A figura 3 traz a representação gráfica do comando DEFINE DOMINIO e sua respectiva representação na meta-base.

O objeto-relacionamento SINÔMINO, associa um comando E/D a um comando estrangeiro. Comandos estrangeiros são comandos associados a um modelo de dados qualquer, e que possui um correspondente no modelo E/D. O Sistema de Interface tem cadastrado para cada comando E/D, os comandos mais conhecidos em outros modelos de dados no objeto-relacionamento SINÔNIMO público. Cada usuário, entretanto, pode requerer ao sistema seus próprios sinônimos. Seu cadastramento é concretizado através do diálogo com cada usuário e são armazenados no seu objeto relacionamento SINÓNIMO privado. Na aquisição de novos sinônimos, uma checagem simples de verificação torna-se necessária para manter consistente e coerente a base de conhecimento.

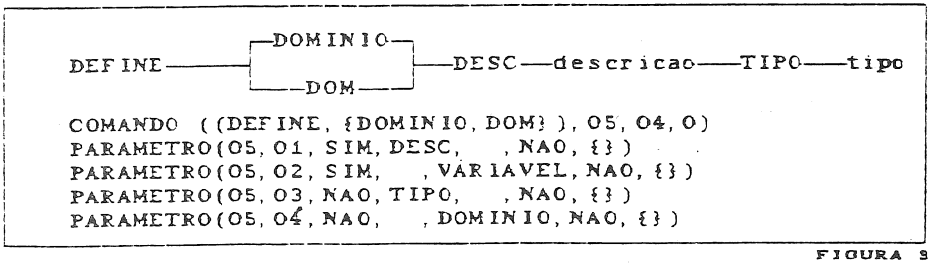

FIGURA 3

3.2 - ADAPTAÇÃO DA INTERFACE, DEFINIÇÃO E EVOLUÇÃO DO PERFIL DO USUÁRIO

A função de adaptação da interface é realizada usando as

informações existentes no módulo MODELO DO USUÁRIO. Iniciada uma seção, o sistema de interface recupera o perfil do usuário que possui todos os dados referentes ao seu grau de conhecimento do modelo E/D. Estes dados ajudam na escolha da forma de comunicação mais adequada para que os objetivos do usuário sejam alcançados.

Para novos usuários do sistema, uma pseudo-entrevista é estabelecida. Neste processo, as informações necessárias para a composição do perfil podem ser obtidas de uma forma amigável e descontraída. A pseudo-entrevista é baseada na abordagem de estereotipos [11] que consiste, fundamentalmente, em um conjunto de características que coexistem nos usuários do ambiente e que podem, a partir de um número pequeno de observações (no nosso caso através de perguntas), ser atribuídas ao entrevistado. No momento atual, o Sistema de Interface dispõe de duas formas de comunicação: Interface baseada em menus e a Interface de linguagem de comandos. Associada a cada tipo estão, respectivamente, os usuários leigos e os não leigos. A possibilidade de uma classe tão geral como a dos não leigos é resultado do bom tratamento de erros que possibilita usuários de perfis diferentes fazerem uso da mesma interface, embora com grau de assistência e forma de cooperação, diferenciados.

A evolução do perfil do usuário é essencial para o nosso sistema de interfaces, visto que a aquisição de conhecimento é um processo contínuo. À medida que o usuário se torna familiar com as funções existentes no sistema e a ocorrência de erros atinja a um patamar desprezível, é necessário que novas formas mais avançadas de interação com o ambiente sejam oferecidas a ele.

O processo de aprendizado e evolução do conhecimento têm demonstrado ser uma tarefa bastante complexa e difícil e requerem, dentro do contexto deste trabalho, uma forte ligação entre Informática, Educação e Psicologia Cognitiva. Este trabalho está sendo iniciado juntamente com os Departamentos de Letras, Eduacação e Psicologia [3]

Em [4] é retratado um sistema assistente de projeto físico de Banco de Dados com tratamento diferenciado para duas classes de usuário: leigo e especialista. Neste trabalho, mesmo tratando com grupos tão distintos, a detecção da evolução do conhecimento continuou sendo bastante difícil. Foi constatado que a mudança de

um perfil para outro sempre requer a necessidade da definição de um perfil intermediário. O critério adotado consistiu em atribuir pesos a cada tipo de solicitação de explicação e tipo de erro cometido. Após um número pré-estabelecido de interações, o usuário, através de seus pontos, era ou não transferido para outro perfil. Apesar de sua importância, a abordagem da evolução do perfil, em razão das dificuldades encontradas, segue, neste trabalho a mesma linha de [4].

3.3 - SISTEMA DE APOIO AO USUÁRIO

A função que permite ao sistema fornecer explicações à medida em que for solicitado é executada pelo módulo EXPLICATIVO utilizando as informações a respeito do perfil do usuário e da base de conhecimento. O usuário tem a possibilidade de questionar o sistema a qualquer momento, independente do seu grau de conhecimento e da interface utilizada no momento. Neste contexto, a explicação consiste no resultado de uma solicitação de informações adicionais a respeito do estado atual ou do comando a ser executado.

Estando a função de explicação disponível a qualquer usuário, uma mesma requisição pode ser respondida de várias maneiras diferentes. O perfil do usuário determina o formato (sentenças em linguagem natural, seqüência de comandos na linguagem de comandos, gráficos, etc) e a complexidade (detalhista, generalizada, superficial) da explicação. A seguir, cada forma de explicação e sua forma de implementação são detalhadas.

3.3.1 - EXPLICAÇÃO DO ESTADO

Este tipo de explicação encontra-se disponível tanto na interface baseada em menus como na de linguagem de comandos. Sua ativação é feita através de uma tecla especial (F1). Considerando que a interface baseada em menu é destinada a usuários completamente leigos, e que o processo de escolhas pode implicar na passagem de diversos submenus, a importância da explicação compreende:

- Situar o usuário em seu trabalho, explicando porque foi necessário a passagem de outros estados para se atingir o estado atual;
- Caso o estado atual não corresponda a um estado final,

fornecer detalhes sobre o significado dos estados finais que podem ser atingidos do ponto atual.

Em relação à interface de linguagem de comandos, a explicação abrange apenas o último comando enviado aceito e inclui informações funcionais.

3.3.2 - EXPLICAÇÃO DO COMANDO

Explicação de comandos está diretamente associada a interface de linguagem de comandos, e por conseguinte, a usuários especialistas. Por especialista, considera-se aqueles que possuem conhecimento do modelo E/D e aqueles que têm trabalhado com Sistemas de Banco de Dados, porém não, necessariamente, com o modelo E/D.

Para evitar que a segunda classe de especialistas seja classificada como usuário leigo e tenha que modelar suas aplicações através de menus, foi criado um mecanismo que permite a correspondência entre comandos E/D e comandos estrangeiros. Enviado um comando estrangeiro ao sistema, o módulo Explicativo retorna a sintaxe completa do comando E/D correspondente e sua explicação funcional.

Toda cadeia de caracteres precedida por uma interrogação é considerada uma requisição de explicação de comando. Solicitada a explicação, o Sistema pesquisa o comando na seguinte ordem: Objeto entidade COMANDO, objeto relacionamento SINÔNIMO público e objeto relacionamento SINÔNIMO privado. Caso a pesquisa não tenha sucesso, o módulo Explicativo é transferido para o módulo Aquisição de Conhecimento para a inclusão de um novo sinônimo.

Estabelecido qual comando E/D ou seu correspondente foi requerido, o Analisador Sintático gera a sintaxe completa do comando. Neste ponto do diálogo, o usuário é capaz de prosseguir sua modelagem. Na figura 4 é apresentado, parcialmente, o objeto-relacionamento SINÔNIMO público referente ao comando 01, DEFINE ENTIDADE.

```
SINONIMO(DEFINE ENTIDADE, {CREATE TABLE,  DEFINIR,  ENTIDADE,
                           CRIACAO,  CRIAR TABELA,  GERACAO})
```

FIGURA 4

3.4 - TRATAMENTO AMIGÁVEL DE ERROS

A função de tratamento de erros é realizada em conjunto pelos módulos meta-base de conhecimento e analisador sintático. O tratamento amigável de erros não é aplicável a todo tipo de interface. [10] afirma que certas formas de comunicação homem-máquina como interfaces gráficas ou baseadas em menus, são projetadas de forma a interagir com o usuário, isentas de erros. Não existe, nestes tipos de interação, a possibilidade de se requisitar um comando cujas pré-condições não tenham ainda sido satisfeitas. Em nosso trabalho, a análise de erros está restrita às interfaces de linguagem de comando.

O tratamento amigável de erros corresponde a não se rejeitar completamente o comando quando algum tipo de erro estiver associado a ele. De acordo com o tipo do erro, o sistema pode: corrigí-lo sem a interferência do usuário, requisitar informações adicionais e não um novo envio do comando reformulado ou alertar o usuário que alguns comandos devem ser executados anteriormente ao comando enviado.

Durante uma interação com o ambiente utilizando uma linguagem de comandos, diversos tipos de erros estão propensos a ocorrer. Os erros podem ser categorizados em: expressão, sintaxe, associação e preparação.

3.4.1 - ERROS DE EXPRESSÃO - Correspondem aqueles surgidos na digitação do comando. Em muitos sistemas, erros de expressão implicam na resubmissão do comando, mesmo que estes se encontrem no final de um comando de muitos parâmetros. O Sistema de Interface detecta estes tipos de erros de forma a agilizar a utilização do modelo E/D. Como existe uma ordem de precedência de cada parâmetro de um comando, sempre que ocorre um problema de reconhecimento, por exemplo a inexistência de um comando ou de um objeto-entidade, o analisador enquadra este não reconhecimento como erro de expressão. Como a maioria dos erros de digitação são de duplicação e troca de caracteres, duas soluções são tentadas para validar o parâmetro com erro de expressão: Eliminação de caracteres adjuntos repetidos e varredura exaustiva do parametro desconhecido, onde a cada processamento, um caracter é desconsiderado na pesquisa.

3.4.2 - ERROS DE SINTAXE - Ocorrem quando o usuário sabe que tipo de comando deve ser executado, porém uma parte ou toda a sintaxe do comando não é lembrada. O analisador sintático tenta estabelecer um casamento de padrões entre o comando enviado e a sintaxe armazenada. Havendo discordância, apenas parametros *STRING* obrigatórios omitidos são solicitados para completar o comando. Sempre que um parâmetro *PAL-RES* que delimita dois tipos *STRING*, e um destes string forem omitidos, é possível que o Sistema de Interface entre em conflito, pois diversas instanciações do comando são geradas. Neste caso, o usuário deve intervir e informar qual das interpretações corresponde à sua solicitação.

O tratamento de erros de sintaxe permite que a linguagem de comandos possa ser manuseada por usuários não leigos de diferentes perfis, pois a necessidade do conhecimento prévio da sintaxe do comando é eliminada. Aqueles com pouca familiaridade com o modelo E/R, necessitam apenas enviar o nome principal do comando, seja ele E/R ou estrangeiro. O sistema de interface pede os parâmetros necessários através de um diálogo amistoso. Quanto mais especialista for o usuário, mais completo são seus comandos e menor é a intervenção do sistema. Através de seus erros e omissões, o usuário avança no aprendizado da linguagem.

3.4.3 - ERROS DE ASSOCIAÇÃO - São erros de origem semântica e que correspondem a relacionamentos errôneos entre objeto-atributo e respectivos objeto-entidade ou objeto-relacionamento. Quando de uma discordância, o Sistema de Interface tenta estabelecer, usando sua meta-base, que outras associações entre atributo, entidade e relacionamento, são possíveis:

- Objetos entidade ou objetos relacionamento que tenham como propriedade o objeto atributo informado;
- Objetos atributo associados ao objeto entidade ou objeto relacionamento informado, possuidores do mesmo domínio semântico do objeto atributo informado.

3.4.4 - ERROS DE PREPARAÇÃO - Ocorrem quando o comando é considerado compreensível ao sistema, porém não é possível executá-lo, pois algumas pré-condições devem ser satisfeitas. O sistema informa que o comando não pode ainda ser processado e

indica que comandos necessitam ser enviados antes do seu processamento. Está sendo projetado, embora não implementado, um módulo de PLANOS/OBJETIVOS, que possibilitará ao sistema deduzir, através das solicitações do usuário, qual o seu objetivo, ou seja, que estado se deseja alcançar. Estabelecido o objetivo, é deduzido o plano escolhido para esta tarefa. Este módulo possibilitará que comandos, incompatíveis com a situação atual de interação, sejam detectados e haja um bom tratamento para esta classe de erros.

4 - CONCLUSÃO

O sistema de Interfaces foi desenvolvido como uma conseqüência natural do Sistema Assistente para Projeto Físico de Banco de Dados [4]. Seu objetivo inicial foi facilitar o processo de modelagem através do modelo E/D. No momento atual, o Sistema de Interface está sendo incorporado ao ambiente POESIS e embora ainda existam algumas restrições, o Sistema de Interfaces tem conseguido ser utilizado eficazmente por pessoas de diferentes níveis e ser um instrumento de auxílio e aprendizado.

O Sistema de Interface deverá ser estendido para abranger outros módulos do ambiente POESIS, como o Integrador de Visões e o Especificador Formal. Para tanto, a Meta-base de conhecimento necessitará de outros objetos Entidade e Relacionamento para representar as novas informações.

Acreditando que o tratamento de erros é o grande responsável pela possibilidade de diversos tipos de usuários usarem satisfatoriamente o ambiente, novas formas de cooperação, principalmente para erros de associação e preparação, devem ser investigadas. Um estudo mais aprofundado da evolução do conhecimento do usuário torna-se, também, necessário. Este estudo possibilitará a existência de outros perfis de usuário e formas de interface.

5 - REFERÊNCIAS BIBLIOGRÁFICAS

[1] M. Bandeira. Modelagem Estática/Dinâmica de Sistemas de Informações. Tese de mestrado, UFPE, 1989.

[2] J. Banerjee, W. Kim, K. Kim. Queries in Objects-Oriented Databases. Object-Oriented Database I, IEEE, 1988.

[3] F.A Barros. Aspectos Teóricos e Práticos no Reconhecimento

Automático de Fenômenos Linguísticos. Tese de Mestrado, UFPE, 1990

[4] L.E.S. Câmara. Um Sistema Assistente para Projeto Físico de Banco de Dados. Relatório Técnico. Departamento de Informática, UFPE, 1990.

[5] J.M. Carroll, J.Mckendree. Interface design issues for advice giving expert systems. Communications of the ACM. 30(1). 1987.

[6] P. Chen. The Entity-Relationship Model: Toward an unified new date. ACM Transaction on Database System. 1(1). 1976.

[7] D. Fonseca. Relatório do Projeto APOIO. Relatório técnico. Departaento de Informática, UFPE. 1990

[8] H.R. Hartson, Hix D. Human-Computer Interface Development : Concepts and Systems for its Management. ACM Computing Surveys. 21(1). 1989.

[9] R. Hirschheim. H.K. Klein. Four paradigms of Information Systems development. Communication of the ACM. 32(10). 1989.

[10] E. Kantorowitz. O. Surarsky. The adaptable user interface. Communications of the ACM. 32(11). 1989.

[11] A. Kobsa. W.Wahlster. User Model in Dialog Systems. Springer-Verlag. 1989.

[12] A. Motro. FLEX: A Tolerant and Cooperative User Interface to Databases. IEEE Transactions on Knowledge and Data Engineering. 2(2). 1990.

[13] R. Motz. Especificação Formal para o Modelo E/D. Tese de mestrado. UFPE. 1990.

[14] J.M. Pager, D.M. Lamberti, D.L. Gardner, S.R. Balzac. REASON An intelligent user assistant for interactive environments. IBM Systems Journal. 29(1). 1990

[15] E. Schoen, R.G. Smith, B.G. Buclanam. Design of Knowledge-Based systems with a Knowledge-Based assistant. IEEE Transactions on Software Engineering. 14(12). 1988.

[16] R. Zwicker, N. Reinhard. Interfaces Inteligentes: Perspectivas para novas formas de aprendizado e uso de sistemas. Revista Brasileira de Computação. 5(3). 1990.