EZEQUIEL GUEIBER

VIQUEN – UM AMBIENTE INTERATIVO PARA CONSULTA VISUAL E EXTRAÇÃO DE ESQUEMAS

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.

Orientador: Marcos Sfair Sunye

CURITIBA

2001

UFPR

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 do aluno ***Ezequiel Gueiber***, avaliamos o trabalho intitulado "***VIQUEN - Um Ambiente Interativo para Consulta Visual e Extração de Esquemas***", cuja defesa foi realizada no dia 23 de fevereiro de 2001. Após a avaliação, decidimos pela aprovação do candidato.

Curitiba, 23 de fevereiro de 2001.

Prof. Dr. Marcos Sfair Sunye
Presidente - Orientador

Profª. Dra. Maria Salete Marcon Gomes Vaz
Membro Externo - UEPG

Profª. Dra. Laura Sanchez Garcia
DINF/UFPR

MINISTÉRIO DA EDUCAÇÃO - UNIVERSIDADE FEDERAL DO PARANÁ - Setor de Ciências Exatas - Coordenação de Pós-Graduação em Informática

AGRADECIMENTOS

A Deus

Pela saúde e tudo que me permitiu realizar.

A minha esposa

Pela paciência, apoio e cuidado que dedicou a mim e minhas filhas, é muito bom ter você ao meu lado.

Ao meu orientador

Pela motivação e esclarecimentos indispensáveis para a conclusão deste trabalho.

Aos meus pais, sogros, irmãos e cunhados

Formamos uma família de verdade e, isto é o que nos motiva a enfrentar cada novo dia, especialmente ao meus pais que sempre me motivaram a estudar.

Aos meu amigos

Aos presentes e ausentes, obrigado pelo auxílio ou simples incentivo, ambos são igualmente importantes e nunca faltaram.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E LISTA DE SIGLAS

ERC	-	Entidade Relacionamento Complexo.
LDD	-	Linguagem de Definição de Dados.
LMD	-	Linguagem de Manipulação de Dados.
ODMG	-	Object Data Management Group.
OQL	-	Object Query Language.
SGBD	-	Sistema Gerenciador de Base de Dados.

RESUMO

As aplicações de consulta à bases de dados têm dado muito atenção ao poder de expressão dos modelos semânticos e a SGBDs baseados em objeto, entretanto, o grande legado de informação ainda reside sob bases relacionais. Objetivando preencher esta lacuna, o resultado deste trabalho é uma aplicação de consulta visual que opera sobre esquemas relacionais e que utiliza um modelo semântico objeto-relacional para interagir com o usuário. A aplicação tem como principais características: um método para extrair um esquema conceitual a partir de uma base de dados relacional; o mapeamento dos operadores algébricos do modelo conceitual para SQL; o uso da representação gráfica do modelo conceitual em todas as fases da formulação da consulta. A tradução está baseada exclusivamente nas informações sobre o esquema relacional, mantida no dicionário de dados pelo SGBD relacional, tais como chaves e restrições de unicidade, sobre as quais são aplicadas regras de refinamento para que se atinja a semântica do modelo conceitual. Os operadores da linguagem de manipulação de dados do modelo conceitual são mapeados para SQL, preservando o poder de expressão do modelo e, por outro lado, permitindo acesso ao vasto volume de informações relacionais existente. Sob a abordagem da representação visual e manipulação direta, a linguagem visual do modelo conceitual é utilizada desde a seleção do subesquema escolhido para a consulta até a formação do predicado, dispensando o usuário de construir ou interpretar qualquer expressão em linguagem de manipulação de dados. A população resultante da consulta à base relacional é obtida de acordo com a demanda, sendo que nenhuma informação adicional é mantida ou gerada sobre os dados do esquema relacional. Dessa forma a resposta é exibida preservando a estrutura de objetos do modelo, isto é, a mesma estrutura de objetos selecionada na consulta é vista na resposta, ao invés da forma planar, e que tem se mostrado difícil de ser interpretada por usuários inexperientes e é tipicamente utilizada como forma de exibição de resposta por aplicações de consulta a bases relacionais.

Palavras-chave: Visualização de dados, exploração interativa de dados, linguagem de consulta visual, interface com o usuário.

ABSTRACT

The user interfaces applications for databases have been given special attention to the expression power of the semantic models and about databases object oriented, however, the great legacy of information is still under relational databases. Aiming at filling in this gap, the result of this work is an application of visual query that operates on relational schemas and uses an object- relationship semantic model to interact with the user. The application has as the main features: a method to extract a conceptual schema from a relational database; the mapping of the algebraic operators from the conceptual model to SQL; the use of the graphical representation of the conceptual model during all the phases of the query construction. The translation is based on the relational schema information, kept in the data dictionary by relational SGBD, such as keys and unicity restrictions, in which the refinement rules are applied so that if it reaches the biggest semantics of the conceptual model. The data manipulation language operators of the conceptual model are mapping for SQL, preserving the expression power of the model and, on the other hand, allowing access to the great volume of relational information existing. Under the visual representation and direct manipulation focus, the conceptual model visual language is used since the selection of sub schema chosen for the query until the predicate composition, excusing the user to construct or to interpret any expression in language of manipulation of data. The query resultant population, from the relationship base, is recover according to demand, so no additional information is kept or generated on the data of the relational schema. On this way the answer is shown preserving the model object structure, that is, the same object structure selected in the building query, instead of the plan form, that been shown difficult of being interpreted by no experts users and typically is used as a way of exhibition by interfaces applications for databases.

Keywords: Data visualization, interactive data exploration, visual query language, user interfaces, automatic presentation systems.

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

Nos últimos anos, conduzidas pelo número crescente de usuários de Sistemas Gerenciadores de Bases de Dados (SGBDs), especialmente de usuários finais de sistemas de informação, pesquisas em interfaces para bases de dados têm motivado muitas conferências e oficinas[1]. Vários protótipos têm sido apresentados com a proposta de facilitar o acesso de usuários aos SGBDs[2]. O convencional aprendizado de linguagens de consulta e prévio conhecimento do esquema de base de dados está sendo substituído pela abordagem de manipulação gráfica direta para formulação de consultas, que envolve paradigmas de representação visual, tais como formulários, diagramas e ícones acrescidos da ação do usuário sobre esses, isto é, operações sobre as representações para interagir com a base de dados (CATARCI et al., 1996).

O suporte necessário à representação visual e às tarefas feitas pelo usuário é dado pelo modelo de dados que deve ser capaz de capturar os conceitos da base de dados sobre o qual a interface atua. Portanto, as aplicações de bases de dados devem utilizar modelos de dados capazes de capturar os conceitos do mundo real percebidos pelo usuário, ou seja, modelos de dados de grande poder semântico (CATARCI et al., 1996). Por outro lado, é evidente o domínio de SGBDs baseados em modelos tradicionais pobres semanticamente, tal como o relacional. Assim forma-se uma lacuna entre o requisito de se usar modelos de ricos semanticamente em aplicações de interface e os modelos de dados tradicionais, sob os quais temos armazenado um vasto legado de informação.

---

[1] Apresentado por KNUTH; WEGNER, 1992. e P. SWAYER, 1994.

2 Destacam-se alguns protótipos como KALEYDOQUERY (MURRAY; NORMAN, 1998b), SUPER (Y. Dennebouy, 1995) e Pasta-3's (KUNTZ; MELCHERT, 1989).

1.2 PROBLEMÁTICA E CONTRIBUIÇÃO

Com o objetivo de suprir a deficiência semântica do modelos tradicionais, várias extensões a estes têm sido propostas, entre elas está o modelo Entidade-Relacionamento-Complexo, abreviado por ERC+ (SPACCAPIETRA et al., 1995), que é um modelo Entidade - Relacionamento (P. CHEN, 1990) estendido, o qual adiciona ao paradigma Entidade - Relacionamento conceitos do modelo comportamental da orientação a objetos e outros conceitos semânticos, como relacionamento genérico, discutidos posteriormente.

Associado ao modelo ERC+, também foi construída a álgebra ERC+, utilizada como linguagem para manipulação de dados do modelo ERC+, e que dá suporte ao modelo. Assim, como o modelo suporta objetos complexos, os operadores algébricos constróem seus tipos de entidades resultantes como objetos complexos. Essa propriedade traz uma importante contribuição, qualquer expressão construída, obrigatoriamente, tem como resultado os mesmos tipos dos operandos envolvidos, ou seja, também objetos complexos. Isso se torna uma vantagem na construção da resposta para o usuário pois, se são utilizados objetos complexos na composição da consulta, a resposta igualmente é composta de objetos complexos. Ressalta-se que apesar da disponibilidade de alguns SGBDs que suportam o modelo [3]objeto-relacional, os mesmos não foram acompanhadas pelo incremento das respectivas linguagens de manipulação de dados (LMDs) para suportar seus conceitos, ou seja, a linguagem não contempla os respectivos operadores para manipular objetos complexos. Assim, as estruturas complexas têm que ser utilizadas como estruturas planares[4] pela LMD, da mesma forma que o modelo relacional as manipula. O resultado de tal abordagem impõe tanto anomalias e restrições na manipulação de dados, quanto na interpretação dos resultados das consultas, visto que a estruturação da informação na forma de objetos são perdidos ao se transformar objetos novamente em tabelas.

---

[3] O legado existente sob estes SGBDs ainda é pequeno quando comparado aos SGBDs relacionais.

[4] denominadas *flat*.

Apesar dos benefícios da abordagem ERC+ ainda existe uma lacuna entre este modelo conceitual e SGBDs comerciais. As aplicações para base de dados precisam de modelos de maior poder de expressão, tal como o modelo ERC+. Entretanto, a grande maioria do legado de sistemas de informação está sob SGBDs que mantêm esquemas baseados no modelo relacional, de menor poder de expressão, ou ainda não implementam integralmente os conceitos dos modelos ricos semanticamente, especialmente a categoria de modelos objeto-relacional, tal como o modelo ERC+. A conexão entre esses modelos envolve tanto para extrair a partir do modelo relacional o modelo conceitual ERC+, quanto efetuar o mapeamento entre as Linguagens de Manipulação de Dados (LMD), isto é, da álgebra ERC+ para SQL, suportada pela maioria dos SGBDs comerciais.

Para extrair um esquema ERC+ a partir de um esquema relacional há uma proposta que utiliza a análise de declarações de dados, mais precisamente instruções em SQL, existentes no código de aplicações que utilizam SGBDs relacionais (M. ANDERSSON, 1995). A proposta deste trabalho, por outro lado, usa informações do dicionário de dados dos SGBDs para perfazer essa tarefa. Esta abordagem, ao não utilizar o código fonte de aplicativos, mostra-se vantajosa, pois é capaz de recuperar todas as associações e tabelas existentes, as quais poderiam, casualmente, ficar omissas pela primeira abordagem devido à comum indisponibilidade do código fonte de aplicações adquiridas de terceiros e, também, nem sempre a camada de código de acesso a dados está separada da camada de regras de negócios, o que pode dificultar a extração daqueles.

Com relação à tarefa de mapeamento entre as LMDs dos modelos ERC+ e aquelas suportadas pelos SGBDs comerciais, temos o protótipo SUPER (DENNEBOUY et al., 1995), um protótipo de interface visual para bases de dados, descrito na seção 3, onde foi construído um interpretador para a álgebra ERC+. Também no protótipo SUPER, pretende-se implementar versões[5] para serem

[5] Atualmente não há acesso a SGBDs relacionais.

executadas no topo de SGBDs tradicionais existentes, como INGRES. Entretanto, este recurso ainda não foi implementado e também, na mesma referência, se descreve o possível uso de um interpretador algébrico ERC+ como uma camada de software intermediária que, se por um lado permite a independência quanto ao SGBD que mantém o legado de informações, por outro, pode levar à degradação de desempenho quando comparado ao acesso nativo da LMD suportada pelo SGBD. Sobre esta característica o protótipo desenvolvido nesse trabalho faz acesso nativo a um dos principais SGBDs comerciais de mercado, ORACLE 8i™ e de suas novas implementações da LMD, com a qual fazemos as equivalência dos operadores da álgebra ERC+, aplicados a formulação de consultas.

1.2.1 Objetivos

Os objetivos do presente trabalho são:

- um método para extrair esquemas ERC+ a partir de esquemas relacionais;
- o mapeamento dos operadores da álgebra ERC+, que dá suporte ao modelo, para SQL;
- a implementação de uma aplicação de consulta à bases de dados, denominada VIQUEN (Visual Query Environment), a qual contenha a extração de esquemas e o mapeamento dos operadores propostos. Ainda a aplicação deve permitir ao usuário: visualizar esquemas sobre um modelo de dados semanticamente mais rico - o modelo ERC+, recuperando a estrutura dos esquemas que residem em bases de dados relacionais e traduzindo-os para um esquema sob esse modelo; expressar suas requisições de consulta visualmente sobre o diagrama ERC+; submeter a consulta e recuperar a informação de esquemas sobre o modelo relacional, utilizando a LMD do SGBD correspondente; finalmente, exibir o resultado de consultas preservando a forma de objetos complexos.

Sendo assim, para descrever esse trabalho esta dissertação está organizada em mais seis capítulos e um apêndice, a seguir especificados.

O capítulo 2 discorre sobre os principais modelos de dados, confrontando o modelo de dados ERC+ (escolhido como sustentação para este trabalho), com os demais modelos, destacando a relevância semântica e, consequentemente, a sua adequação a interface de consulta visual às bases de dados.

O capítulo 3 destaca as principais aplicações de consulta visual e a contribuição pretendida neste trabalho comparativamente a estas aplicações.

O capítulo 4 apresenta um método para tradução de esquemas relacionais em esquemas ERC+ em um abordagem que utiliza somente informação do dicionário de dados de um SGBD relacional.

O capítulo 5 descreve os mapeamentos de operadores algébricos ERC+ para SQL.

O capítulo 6 apresenta a aplicação implementada neste trabalho.

O capítulo 7 descreve os resultados obtidos com este trabalho e os trabalhos futuros pretendidos.

O Apêndice A apresenta a criação das visões que formam a estrutura base do dicionário de dados do modelo ERC+, utilizada pela aplicação de consulta.

2 MODELOS DE DADOS

As interfaces visuais para bases de dados objetivam aproximar os conceitos do mundo real, percebidos pelo usuário, dos esquemas que descrevem os dados dos sistemas de informação (MURRAY; NORMAN; GLOBE, 1998). A fim de expressar adequadamente esses conceitos, o sistema de consulta deve dispor de um modelo de dados com capacidade semântica suficiente para descrevê-los (CATARCI et al., 1996). Sendo assim o modelo de dados escolhido deve ser expresso em termos de representações para dar suporte a interação visual da aplicação. Dessa forma, o modelo de dados é a sustentação da aplicação.

Pretende-se neste capítulo descrever as principais contribuições e deficiências de alguns destes modelos.

Um modelo de dados típico é composto de: uma parte que descreve os conceitos inerentes aos componentes da base de dados, segundo os quais a base de dados pode ser construída, sendo eles: objetos, ligações e propriedades; uma parte de manipulação de dados que define as operações que podem ser feitas sobre os dados, atualização, consulta e inclusão de dados, ou seja, o comportamento dinâmico.

Há vários modelos de dados, tais como relacional, relacional estendido, orientado a objetos, entre outros. É o tipo do modelo de dados quem determina os componentes do ambiente que podem ser visualizados e as operações disponíveis. Assim, para cada interface será o modelo de dados subjacente que especificará como os conceitos serão utilizados. A interface da base de dados não precisa usar todos os conceitos do modelo de dados, ela pode prover apenas um subconjunto da capacidade operacional do modelo. Por exemplo, ao exibir o esquema da base de dados é possível exibir apenas o nome do atributo, ao invés de todas as propriedades deste (MURRAY; NORMAN, 1998a).

A subseção seguinte aborda os principais modelos e seus aspectos relevantes para as aplicações de interfaces para bases de dados.

2.1 PRINCIPAIS MODELOS DE DADOS

Os requisitos impostos pelas aplicações de interface para bases de dados tornaram modelos tradicionais deficitários. Analisando o modelo relacional, observa-se que sua propriedade elementar de valor atômico para atributos o torna incapaz de capturar a complexidade da informação envolvida nos objetos do mundo real (DENNEBOUY et al., 1995). Várias extensões foram propostas, como o modelo relacional aninhado, cuja estrutura puramente hierárquica e acíclica também limitou seu poder de expressão, ou seja, a estrutura hierárquica impede múltiplas percepções de objetos e a acíclica não permite que um componente de um objeto seja do mesmo tipo do objeto que ele compõe. Também esse modelo, bem como a abordagem entidade-relacionamento (outro modelo tradicional), não tem mecanismos apropriados para descrever o comportamento dinâmico dos objetos do mundo real (SPACCAPIETRA et al., 1995).

Um poder de expressão maior foi obtido com o advento do modelo orientado a objetos para base de dados capaz de descrever ambos, estruturas complexas e o comportamento dinâmico dos objetos. A seguir são apresentados os principais conceitos deste modelo:

- a estrutura do objeto: um conjunto de variáveis que contém a informação acerca do objeto e métodos implementados para as mensagens ao qual o objeto responde, isto é, seu comportamento dinâmico(SILBERCHATZ; KORTH; SUDARSHAN, 1999);
- classes de objetos: objetos similares, que respondem às mesmas mensagens e com o mesmo conjunto de variáveis, são agrupados em classes.
- herança: classes que possuem os mesmos atributos e mesmo comportamento dinâmico mas, adicionalmente, possuem outros atributos ou respondem a outras mensagens podem ser definidas a partir daquelas. Tal definição é feita através de uma hierarquia de especialização. Para

exemplificar, pode-se definir estudante como uma especialização de pessoa. Esta forma de ligação é dita [6]*é-um*. Também é possível definir uma classe a partir de duas ou mais classes, quando esta reúne as características de ambas, denominada de herança múltipla;

- identidade de objeto: um objeto tem sua identidade preservada, mesmo que suas variáveis ou métodos sejam alterados. Conceitualmente, é atribuído a cada objeto, um identificador quando o mesmo é criado;
- objetos compostos: objetos podem fazer referencia a outros objetos. Considere uma bicicleta que contém rodas, um quadro, freios e marchas. As rodas, por sua vez, contêm um aro, um conjunto de raios e um pneu. Cada um destes componentes pode ser modelado como um objeto componente que é parte da bicicleta. Este é o conceito utilizado pela orientação a objetos para fornecer múltiplas percepções do mesmo objeto.

Com tais características, o modelo orientado a objetos tem poder semântico suficiente para descrever a complexidade dos objetos do mundo real. Entretanto, o poder de expressão para descrever associações entre os objetos é limitado, pois nas duas formas de descrever associações disponibilizadas por este modelo, composição de objetos e herança, ou seja, as associações são tratadas de forma direcional pelo modelo, isto é, ou um objeto é componente ou subclasse de outro (SPACCAPIETRA et al., 1995).

Essa abordagem, freqüentemente, não corresponde à associação percebida pelo usuário no mundo real. Considere um exemplo envolvendo conceitos de *Livro* e *Autor*, em um suposto esquema de base de dados *Editora*, certamente há percepções do usuário em que, nem *Livro* é componente de *Autor*, nem *Autor* é componente de *Livro* e, sim, ambos são vistos como objetos independentes e com possíveis

[6] um objeto é descrito como sendo um tipo de outro, ou seja especialização deste.

associações entre si, de forma não direcionada. Todavia, uma implementação sob um modelo de objetos obrigaria a escolha de uma composição direcionada entre os mesmos, ou seja, *Autor* como componente de *Livro* ou vice-versa.

A obrigatoriedade de associações direcionadas imposta pelo modelo de objetos se contrapõe à visão de objetos independentes percebida pelo usuário, os quais simplesmente se relacionam. Consequentemente, a utilização de modelos de objetos como interface com o usuário em aplicações de consulta retrata essa deficiência semântica, tornando-se esdrúxula e de difícil percepção pelo usuário não especialista. Outra característica desse modelo é que o mesmo ignora descrição de diferentes papéis em associações. Ainda no mesmo exemplo de *Autor* e *Livro,* pode-se ter uma associação entre os mesmos que descreva o papel do *Autor* principal do *Livro* e outra associação, também entre estes, para descrever os co-autores do *Livro*, este exemplo é exibido na Figura 1, utilizando a notação no Modelo Entidade-Relacionamento..

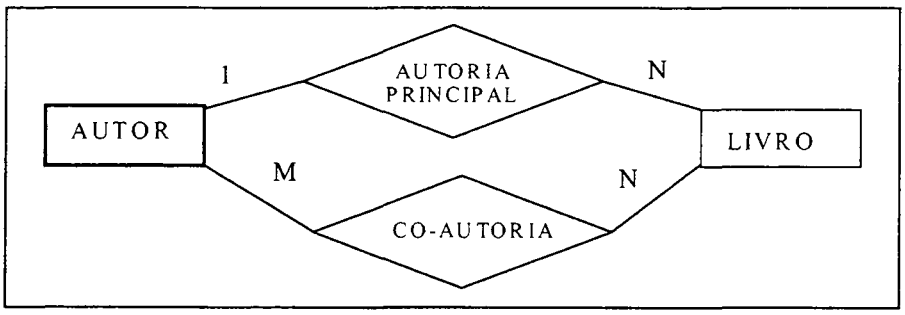


FIGURA 1: MÚLTIPLOS PAPÉIS E ASSOCIAÇÕES GENÉRICAS

A ausência de um modo para descrever diferentes papéis reduz o poder de expressão do modelo. Considerando a Figura 1, observa-se que o usuário pode desejar uma visão sobre o ponto de vista de qualquer um dos papéis, a fim de obter as diferentes informações retratadas por estes, bem como em qualquer um dos sentidos da associação, de *Livro* para *Autor* ou vice-versa. A composição direcionada disponibilizada pelo modelo de objetos não atende esse requisito de associação genérica, sendo que a alternativa dada por este modelo para expressar este tipo de associação está no uso de atributos que implementam referências cruzadas entre ambas

as classes. Porém além do fator de desempenho envolvido em uma eventual implementação, a informação semântica sobre a descrição do papel não pode ser incluída (SPACCAPIETRA et al., 1995). Outrossim, o modelo de objetos não permite que se especifique atributos em associações, o que causa anomalias de definição. Como por exemplo a associação genérica entre *Aluno* e *Disciplina* descrevendo um papel *Matrícula* e a respectiva data desta. O atributo data faz parte da associação e não de nenhuma das classes envolvidas na associação.

Para suprir as deficiências semânticas em representar associações genéricas foram propostas algumas extensões ao modelo orientado a objetos. Esta abordagem é denominada modelo de dados objeto-relacional. Nesse modelo as associações entre objetos não são representadas por mecanismos de agregação, mas através de relacionamentos *n-ários* entre objetos, que compõem novas classes que podem ter atributos adicionais, próprios do relacionamento (ALBANO; GHELLI; ORSINI, 1991).

Abordando também relacionamentos, J. RUMBAUGH (1987) definiu três formas destes: generalização, agregação e associações genéricas, entretanto limitadas a associações binárias (SPACCAPIETRA et al., 1995). Também ALBANO, GHELLI e ORSINI (1991) propõem uma linguagem de programação orientada a objetos que tem relacionamentos genéricos, capaz de suportar relacionamentos *n-ários*, e que possui restrições de integridade para assegurar a integridade referencial entre objetos. A desvantagem desta abordagem é que as implementações que complementam o modelo orientado a objetos estão embutidas em uma linguagem e não em um modelo semântico, pois aplicações de interface para bases de dados baseadas em grafos utilizam justamente modelos na interação com o usuário.

Elaborado como uma extensão para o, amplamente utilizado, modelo entidade-relacionamento, temos o modelo ERC+, que em um nível conceitual incorpora relacionamentos genéricos *n-ários*, estruturas de objetos complexos/compostos e duas linguagem formais, as quais dão suporte ao modelo.

Os principais conceitos que caracterizam este modelo e que o diferenciam de modelos tradicionais, segundo SPACCAPIETRA et al. (1995), são descritos a seguir:

- escolha entre generalização e especialização: o usuário determina a visão desejada naquela interação com a base de dados. Dado, por exemplo, um esquema *Pessoa* e *Estudante*, assumindo estes como generalização e especialização respectivamente, a partir de uma interface de consulta, o usuário pode escolher a visualização de *Pessoa* ou *Estudante*;

- dois tipos de relacionamentos: o relacionamento tradicional de generalização "*is a*"[7] para descrever associações subclasse e classe e associações "*may-be-a*"[8] para descrever conjunções, isto é, conjuntos de objetos podem compartilhar objetos entre si. Assim conjuntos distintos de objetos de duas classes podem ser: disjuntos, um conjunto pode conter parte de outro ou ainda um pode conter todo o outro conjunto;

- objetos do tipo relacionamento com identificador: relacionamentos são descritos com identificadores próprios, diferente dos identificadores dos objetos que relacionam. Cabe lembrar do modelo Entidade-Relacionamento, proposto por P. CHEN (1990), onde os relacionamentos são identificados pelo uso de identificadores das entidades envolvidas no relacionamento. O conceito de identificador próprio em relacionamentos foi introduzido para permitir relacionamentos duplicados entre mesmas entidades, bem como eventuais alterações. Considere o exemplo de um cupom fiscal, em que o mesmo produto pode aparecer na mesma quantidade mais de uma vez, tipicamente não se altera a quantidade e, sim, um novo item é registrado. Assim a associação entre o *Cupom* e *Produto*, pode se repetir. Da mesma forma, um dos itens pode ser excluído sem que seja necessário excluir todos os itens onde aparece aquele dado *Produto* no *Cupom*[9]. O uso de um identificador próprio resolve ambos os problemas de inclusão e alterações (inclusão ou exclusão).

---

[7] é um.

[8] pode ser um.

[9] Pode-se excluir um relacionamento, sem que se tenha que excluir outros iguais a este.

O incremento semântico pertinente ao modelo ERC+ em relação ao demais, obtido pela possibilidade de múltiplas visões e associações genéricas, levou a escolhe-lo como suporte ao protótipo desenvolvido neste trabalho.

Na subseção a seguir, esse modelo e a respectiva álgebra apresentam-se descritos formalmente.

2.2 MODELO ERC+

Os dois próximos subitens descrevem o modelo e a álgebra ERC+. Este modelo foi utilizado na aplicação como suporte as interações do usuário, ou seja, é através dos conceitos deste modelo que o usuário formulará suas consultas. Também as expressões em SQL são construídas pelo suporte da linguagem visual dado por este modelo, com equivalência aos operadores da álgebra ERC+[10].

2.2.1 Definição formal do modelo ERC+

Este modelo é uma extensão do modelo entidade-relacionamento, projetada para suportar objetos complexos. Sua meta é aproximar a representação de objetos do mundo real, sendo que esse modelo usa o conceito de tipo entidade para descrever objetos, e o de tipo relacionamento para representar relacionamentos entre objetos. De acordo com SPACCAPIETRA et al. (1995), segue a definição formal:

> *Domínios*. Define o conjunto de todos os valores possíveis para um tipo atributo, entidade ou relacionamento. Valores podem ser atômicos ou complexos, isto é, formado de outros valores. Um valor complexo é um conjunto de pares <nome atributo nome, v>, onde v é um valor simples ou multivalorado. Considere *ED* o conjunto de domínios de valores atômicos, chamados domínios elementares. Estes são definidos:
> Domínio Elementar: *ed*, $ed \in ED \Leftrightarrow ed = (\text{name}, V, R)$
> - name(ed) $\in$ NAMES (onde name(ed) é o nome do domínio e NAMES é o conjunto de todos os nomes)
> - $V(ed)$ é o conjunto de valores elementares[11] no domínio
> - $R(ed)$ é o conjunto de operadores relacionais definidos sobre $(V(ed)^2)$.

[10] Aplicados a formulação de consultas.

[11] Atômicos.

Considere *CD* ser o conjunto de domínios complexos. Um domínio complexo é um conjunto de todos os valores complexos os quais tem o mesmo formato. Há também um domínio complexo especial cujo propósito é representar a ausência de valor de relacionamentos e entidades que não tem atributo. Domínios complexos são definidos:
Domínio complexo: *cd*, $cd \in CD \Leftrightarrow cd = (\text{name}, V)$ tal que: name(*cd*) ∈ NAMES é o nome do domínio complexo[12], *V*(cd) = {∅} é o domínio vazio de entidades e relacionamentos sem atributos ou $\exists I = \{1,2,\ldots,n\}$ n ∈ **N*** $\exists F = \{ (A_i, d_i, min_i, max_i) / i \in I)$, tal que $\forall i \in I$, $A_i \in$ NAMES $\wedge$ $d_i \in (ED \cup CD) \wedge min_i \in N \wedge max_i \in N^* \wedge min_i \leq max_i \wedge \forall (k,j) \in N^2$ $A_k = A_j \Rightarrow (A_k, d_k, min_k, max_k) = (A_j, d_j, min_j, max_j)$ e V(cd) = { {($A_{i,i}$)/i ∈ *I*} / $\forall i \in \exists (A_i, d_i, min_i, max_i) \in F \wedge i \in p^{min\ i:max\ i} (V(d_i,))) \wedge \forall (k,j) \in N^2$ $A_k = A_j \Rightarrow$ ($A_{k,k}$) = ($A_{j,j}$) }, onde *P*(*V*) significa o subconjunto de *V*, estendido para multiconjuntos, denominado multiconjunto potencial de *V*;
e $P^{m:n}$ (*V*) denominado multiconjunto potencial de *V*, gerando multiconjuntos contendo pelo menos *m* elementos e no máximo *n* elementos (incluindo duplicados).

Estruturas. O conceito de estrutura fornece a recursividade necessária para descrever objetos complexos. Este conceito dá as características de uma atributo, isto é, nome, cardinalidade, composição e domínio, independentemente de suas associações e dos objetos que ele descreve. Tal combinação permite que atributos diferentes compartilhem as mesmas estruturas, o que simplifica a definição formal dos operadores algébricos, cujas ações freqüentemente incluem criar um novo atributo com a mesma estrutura dos atributos que ele derivou.
Considere o conjunto de estruturas. Aqui definidas:
S ∈ S ⇔ S = (name, min, max, comp, d) tal que:

- name(S) ∈ NAMES é o nome da estrutura (os atributos associados a estrutura terão este nome)
- min(S) ∈ **N**, max (S) ∈ **N***, min(S ≤ max(S) são a cardinalidade mínima e o máxima da estrutura. Estes números limitam o número de valores, incluído duplicados, associados aos atributos que podem ter uma instância do objeto ao atributo relacionado.

Se min(*S*) = 0, S define um atributo opcional;
Se min(*S*) ≤ 1, S define um atributo obrigatório;
Se max(*S*) =1, S define um atributo monovalorado;
Se max(*S*) >=2, S define um atributo multivalorado;

- comp(s) = { $S_i / i \in I \wedge S_i \in S$}, *I* = {1,2,...,n}, n ∈ **N*** ou *I* = ∅ é a composição da estrutura. Se comp(*S*) = ∅, a estrutura define um atributo atômico, caso contrário, comp(S) é o conjunto de estruturas dos atributos componentes.
- *d*(*S*) é o domínio subjacente da estrutura: se comp(*S*) = ∅ então *d*(*S*) ∈ *ED* (*d*(*S*) é um domínio elementar) senão *d*(*S*) ∈ *CD* (*d*(*S*) é um domínio complexo)

O conjunto de valores complexos de *d*(*S*) é uma informação derivada: $V(d(S))$= { $(\text{name}(S_i)_{,i})/i \in I\} / \forall i \in I)_{,i} \in p^{min(S_i):max(S_i)} (V(d(S_i)))$ }

Tipos Entidades. Um tipo entidade é definido pelo seu nome, esquema, formado pela estrutura de seus atributos, o conjunto de tipos entidades com os quais ele está em um conjunção e suas ocorrências, compostas de identificadores e valores. Considere *E* o conjunto de tipos entidades, definido por: *E* ∈ E⇔=(name, sch, gen, muid, pop) tal que:

- name (*E*) ∈ NAMES é o nome do tipo entidade
- sh(*E*) = { $S_i / i \in I \wedge S_i \in S$}, *I* = {1,2,...,n}, n ∈ **N*** ou *I* = ∅ é o esquema do tipo entidade. Ele é um conjunto de estruturas (pode ser vazio).
- gen (*E*) = { (EG_j)/ $EG_j \in E$}, é o possível conjunto vazio de tipos entidade genérica de *E*.

[12] Domínios complexos não vazios são conjuntos de todos os valores complexos os quais tem o mesmo formato. Este formato especifica, para cada componente do valor complexo, seu nome, domínio e cardinalidade.

Considere *EG* ser qualquer tipo entidade genérico de *E*, então para cada ocorrência de *E* corresponde uma ocorrência de *EG* que descreve o mesmo objeto do mundo real: soid(*E*) soid(*EG*), deve ser satisfeito a qualquer tempo; soid(*E*) é o conjunto de oids de *E* e é formalmente definido a seguir.

- muid(*E*) = { (E_j) / $E_j \in d(E)$} é o conjunto de tipos entidades ligados a *E* por conjunção, o conjunto pode ser vazio.

- pop(*E*) = { (oid, val)/val $\in d(E)$} é a população do tipo entidade. É o conjunto de entidades. Cada entidade é um par feito do tipo identidade (oid) e seu valor. O valor é um elemento do domínio do tipo entidade, *d*(*E*), o qual é uma informação derivada: $d(E) \in CD$, $V(d(E)) = \{ (\text{name}(S_i),_i)/i \in I\} / \forall\ i \in I)$, $S_i \in \text{sch(E)} \wedge {}_i \in p^{\text{mi}_i::\text{ma}_i})\ (V(d(S_i)))\ \}$, onde: $\text{mi}_i = \min(S_i)$ e $\text{ma}_i = \max(S_i)$. O conjunto de oids de *E* é chamado soid(E), e é formalmente definido por: soid(*E*) = { $e / \exists\ v, (e,v) \in \text{pop}(E)$}

Tipo Relacionamento: Um tipo relacionamento é definido pelo seu nome, o conjunto de tipos entidades que ele liga, com a descrição das características da ligação, i.e., papel, nome e cardinalidade, também o conjunto de estruturas de seus atributos e o conjunto de suas ocorrências.

$R \in R \Leftrightarrow R$ = (name, pet, sch, pop) tal que:

- name (*R*) $\in$ NAMES é o nome do tipo relacionamento
- pet(R) é o conjunto de tipos entidade participantes no tipo relacionamento. Para cada tipo entidade, seus papel e as cardinalidades mínimas e máximas de sua ligação no tipo relacionamento são especificadas:
  $\text{pet}(R) = \{ (E_j, \text{role}_j, \min_j, \max_j) / j \in J \wedge E_j \in E \wedge \text{role}_j \in \text{NAMES} \wedge \min_j \in \mathbf{N} \wedge \max_j \in \mathbf{N}^* \wedge \min_j \leq \max_j \}$, $J = \{1,2,\ldots, p\}$, $p \in \mathbf{N}^*$, $p > 1$, dentro do tipo relacionamento nomes de papel são únicos: $\forall\ ((E_1, \text{role}_1, \min_1, \max_1), (E_2, \text{role}_2, \min_2, \max_2)) \in (\text{pet}(R))^2$, $\text{role}_1 = \text{role}_2 \Rightarrow (E_1, \text{role}_1, \min_1, \max_1) =), (E_2, \text{role}_2, \min_2, \max_2)$
- sch(*R*) = { $S_1, i \in I \wedge S_1 \in S\}$, $I = \{1, 2, \ldots, n\}$, $n \in \mathbf{N}^*$ ou $I = \varnothing$ é o esquema do tipo relacionamento. Ele é conjunto de estruturas de seus atributos.
- pop(*R*) = { (oid, poc, val) } é a população do tipo relacionamento. Cada relacionamento é uma tupla composta do identificador do relacionamento (oid), o conjunto de entidades ligadas (poc) e o valor do relacionamento (val).
  $\forall\ r \in R$, $\text{poc}(r) = \{\{ (E_j, \text{role}_j, e_j) / (E_j, \text{role}_j, \min_j, \max_j) \in \text{pet}(R) / \forall j \in J, e_j) \in \text{soid}(E_j)\}$
  $\forall j \in J, \forall\ e_j \in \text{soid}(E_j)$, $\min_j \leq \text{card}(\{r / r \in \text{pop}(R) \wedge (E_j, \text{role}_j, e_j) \in \text{poc}(R)(r)\}) \leq \max_j$.

O valor de um relacionamento é um elemento do domínio do tipo relacionamento, *d*(*R*), o qual é uma informação derivada: $d(R) \in CD\ V(d(R)) = \{\{(\text{name}(S_i), {}_i)/i \in I) / \forall i \in I, S_i \in \text{sch}(R) \wedge {}_i \in p^{\text{mi}_i::\text{ma}_i})\ (V(d(S_i)))\}$, onde:

$\text{mi}_i = \min(S_i)$ e $\text{ma}_i = \max(S_i)$. O conjunto de oids de *R* é chamado soid(*R*), e é formalmente definido por: soid(*R*) = { $e / \exists$ role , (*e*, role, *v*) $\in$ pop(*R*)}

Atributos. Um atributo é definido pelo objeto ao qual ele está associado, sua estrutura e seus valores para cada ocorrência do objeto.

Considere *A* o conjunto de atributos. Definido por:

$A \in \text{A} \Leftrightarrow$ = (obj, str, inst) tal que:

- *obj*(*A*) $\in$ ($E \approx R \approx A$) é o objeto (tipo entidade, tipo relacionamento ou atributo complexo) ao qual o atributo está associado.
- str(*A*) (name(*A*) min(*A*), max(*A*), comp(*A*), *d*(*A*)) é a estrutura associada ao atributo,

$\text{str}(A) \in S$

se ($\text{obj}(A) = E_i, \wedge E_i \in E$) *então:* $\text{str}(A) \in \text{sch}(E_i)$

se ($\text{obj}(A) = R_j \wedge R_j \in R$) *então:* $\text{str}(A) \in \text{sch}(R_j)$

se ($\text{obj}(A) = A_k \wedge A_k \in A$) *então:* $\text{str}(A) \in \text{comp}(A_k)$

- inst(A) é a instanciação do atributo. É a função completa associando cada valor do objeto para com o atributo correspondente, o valor (que pode ser um multiconjunto) componente do atributo.

inst(A) é definido por:

inst(A): V(d(obj(A))) $\rightarrow$ $P^{min(A):max(A)}$ ($V(d(A))$), tal que: $\forall$ **v** $\in$ $V(d(\text{obj}(A)))$, inst(A)(**v**) = $\text{proj}_{\text{name}(A)}$(**v**).

Axioma de Identidade. Considere gen*(E) ser o conjunto de ancestrais de um tipo entidade E no grafo de generalização.

Considere gen+-(E) ser o conjunto de tipos entidade para os quais um tipo entidade E está conectado através de um caminho no grafo de generalização:

1. Não há ciclos no grafo de generalização: $\forall E \in E \; E \notin \text{gen}^*(E)$
2. Dois tipos entidade ligados por um caminho no grafo de generalização podem compartilhar objetos. Definir uma ligação de conjunção entre elas é desnecessário: $\forall E_i \in \text{gen+-}(E_2) \; E_1 \notin \text{muid}(E_2)$
3. Dois tipos entidade, E_1 e E_2, os quais não estão envolvidos nem por ligação de generalização nem por um ligação de conjunção não podem ter nenhum oid comum: $\forall E_i \in E \; \forall E_2 \in E \; (E_1 \notin \text{gen+-}(E_2) \wedge (E_1) \notin \text{muid}(E_2)) \Rightarrow \text{soid}(E_1) \_ \text{soid}(E_2) = \varnothing$ (SPACCAPIETRA et al., 1995, p. 188-190).

Os conceitos apresentados aqui serão utilizados posteriormente para formação das regras de tradução e refinamento de esquemas, ambas apresentadas no capítulo 4. A semântica deste modelo também será utilizada por meio de uma linguagem visual na especificação da consulta, que será apresentada no capítulo 6.

2.2.2 ÁLGEBRA ERC+

Um dos principais problemas das linguagens de manipulação de dados desenvolvidas para trabalhar com modelos objeto-relacional e extensões do modelo entidade relacionamento é não dispor de operadores para trabalhar com todos estes conceitos, especialmente operadores para manipular diretamente relacionamentos genéricos. A álgebra ERC+ foi construída para manipular todos os conceitos do modelo ERC+. Desta forma, ela será utilizada como origem do mapeamento para SQL, visto que o SGBD subjacente, adotado para o protótipo VIQUEN, tem como alvo esquemas relacionais.

Neste capítulo estão descritos os operadores de consulta associados com o modelo ERC+, conforme apresentado em SPACCAPIETRA et al. (1995). Tais conceitos serão utilizados para especificar o mapeamento dos operadores de consultas

para LMD do SGBD utilizado nesse protótipo. O exemplo referenciado durante essa descrição é exibido na figura seguinte:

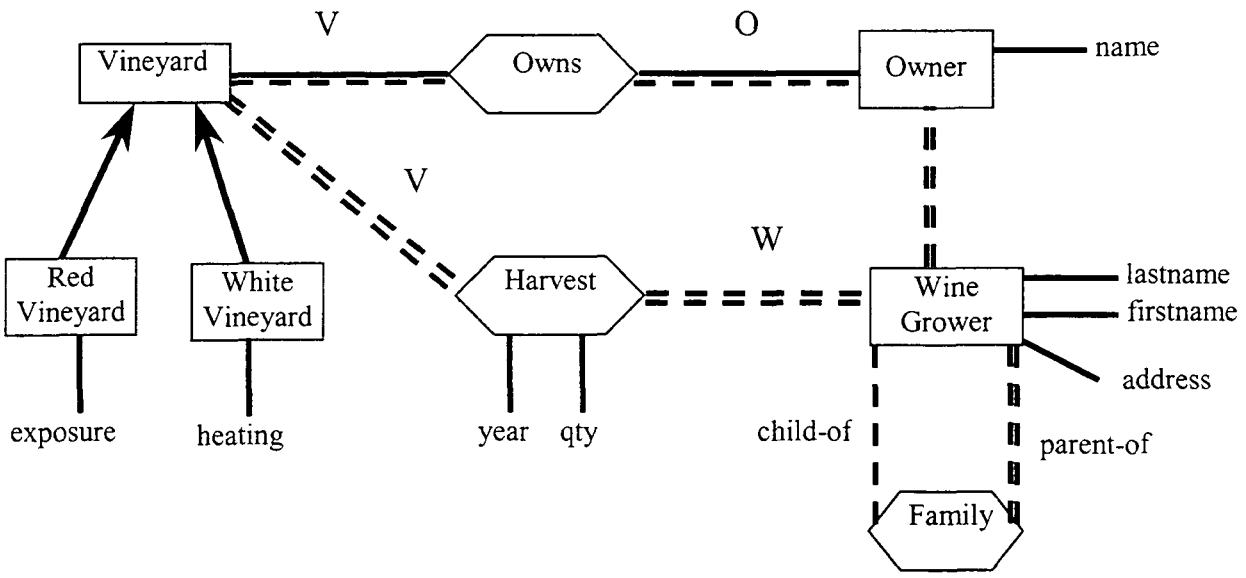


FIGURA 2: EXEMPLO DE UM ESQUEMA ERC+

A álgebra ERC+ é um conjunto de operadores primitivos que podem ser combinados em qualquer ordem dentro de expressões, de tal forma que qualquer consulta sobre um base de dados ERC+ pode ser satisfeita por uma expressão algébrica apropriada.
A propriedade de fechamento[13] e o poder de expressão da álgebra permitem cinco pressupostos básicos:

- operandos e resultados são do tipo entidade. Cada operador é projetado para manipular um (ou mais) tipo(s) entidade e para construir o resultado como um tipo entidade derivado. Ainda o resultado pode servir como um operando para um operador subsequente.
- tipos entidade resultantes são complementados com tipos relacionamentos, generalização e ligação de conjunção derivados dos operandos. Este procedimento permite que o resultado das expressões sejam manipulados normalmente como tipos entidades. Também expressões algébricas de qualquer complexidade podem ser definida. A álgebra ERC+ é completa.
- a álgebra preserva objeto. Cada ocorrência do resultado é derivada dos operandos, que são entidades, do seguinte modo:
  - seu identificador de objeto (oid[14]) é idêntico ao oid da entidade operando do qual ele veio. Por exemplo, os oids no resultado desta seleção: *select* [firstname="Paul"] *Winer-Grower* são os oids de Paul *winer-grower*(s);

[13] A álgebra suporta a semântica do modelo (PARENT C., 1987).

- seu valor é derivado dos valores do(s) operando(s). Por exemplo, o valor da seleção acima é o valor dos correspondentes *wine grower*. O valor de uma ocorrência desta projeção *project*[lastname] *WineGrower* é o valor da correspondente *wine grower*, restrito ao atributo último nome;
- seus relacionamentos são derivados daqueles que ligam os operandos;
- suas ligações de generalização e conjunção são derivadas daquelas que envolvem os operandos;

Devido ao modelo suportar objetos complexos os operadores constróem seus tipos de entidades resultantes como objetos complexos. Em particular os operadores *product*[15], *relationship-join*[16] e *identity-join*[17] (usados para transformar diferentes tipos de entidades dentro de um simples objeto) usam estruturas de atributos complexo para inserir a informação dos outros operandos dentro de um operando principal, do tipo entidade. *Product*, *relationship-join* são operadores que retornam estruturas não planares[18]. A ocorrência dos outros operandos são transformadas dentro de um atributo complexo multivalorado. Por exemplo a junção relacional de Owner e Vineyard através do relacionamento Owns:

O := Owner r-join (Owns, Vineyard) cria um novo tipo entidade *O*, cujo esquema compreende o esquema de Owner acrescido de um atributo complexo, agrupando todos os atributos de Vineyard.

Esta particularidade da álgebra ERC+ tem duas principais vantagens:

- o preenchimento de todos os requisitos do usuário para produzir estruturas não planares que mostrem para cada ocorrência da entidade principal todas as ocorrências de entidades secundárias ligadas a ela.
- operadores binários com um operando principal (produto e junção-relacional) preservam os objetos: os oids dos resultados são derivados (são iguais) daqueles do operando principal. Com estruturas planares esta condição não seria possível.
- herança através de ligações *is-a*[19] e *may-be-a*[20] é efetuada de duas maneiras.

A álgebra oferece um operador, o *identity-join*, o qual permite ao usuário agrupar dois tipos entidades através de ligações *is-a* ou *may-be-a* dentro de um tipo entidade simples. Todos os demais operadores usam somente suas próprias propriedades (não herdam) de seus operandos.

A álgebra inclui nove operadores primitivos. Operadores derivados podem também ser definidos para facilitar a elaboração de consultas envolvendo vários operadores. A seguir serão apresentados brevemente os operadores.

- O operador *selection*[21] é similar ao da álgebra relacional. A operação:

  E = *select* [predicado] E_1,

  onde E_1 é um tipo entidade, E é um tipo entidade derivado, cuja população é o subconjunto da população de E_1 para o qual o predicado é verdadeiro. O esquema, os relacionamentos, generalizações e ligações de conjunção de E são derivados daqueles de E_1.

---

[14] É a abreviatura para o termo em inglês object identificator - identificador do objeto. Os identificadores de objetos são únicos, isto é, cada objeto tem um identificador e não há dois objetos com o mesmo identificador. Uma entidade mantém sua identidade independentemente de alterações do valor de suas propriedades (SILBERSCHATZ; KORTH & SUDARSHAN, páginas 258-259).

[15] Produto.

[16] Junção-relacional.

[17] Junção identidade.

[18] Estruturas não planares ou estruturas que não estão na primeira forma normal, são aquelas que permitem atributos compostos em sua estrutura.

[19] "é um".

[20] "pode-ser-um".

[21] Seleção.

O predicado pode envolver qualquer atributo ou atributo componente de E. Para cada atributo multivalorado uma variável associada com um quantificador ($\exists$ ou $\forall$) deve ser definida.

Exemplo:

Relacionar os *vineyards* que tiveram o melhores anos em 1983 e 1990.

select [$\exists b_1 \in$ bestyears, $\exists b_2 \in$ bestyears ($b_1 = 1990 \wedge b_2 = 1983$)] Vineyard

- operador [22] *projection* é similar ao operador da álgebra relacional.

A operação E = *project* [lista-atributos] E_1, onde E_1, é um tipo entidade e lista-atributos é uma lista de atributos de E_1 e/ou subestruturas de atributos de E_1 que criam um tipo entidade derivado E, cujos oids são os mesmos daqueles de E_1. Os relacionamentos, generalizações e ligações de conjunção E são derivados daqueles de E_1. O esquema de E é constituído dos atributos ou subatributos especificados na lista de atributos. As subestruturas da lista de atributos são definidas usando a notação convencional de ponto. O valor de cada ocorrência E é derivada dos valores das ocorrências correspondentes E_1, restringindo conforme a lista de atributos.

Exemplo:

Para cada vinha (Vineyard) listar sua identificação (label) e seus anos de plantio (planting years).

project[label, planting.year] Vineyard.

- operador *reduction*[23] é um operador novo.

Este operador complementa as funcionalidades oferecida pelos operadores *selection* e *projection* com respeito a meta de selecionar a informação desejada de um tipo entidade. Através da seleção e projeção permite ao usuário descartar ocorrências ou atributos que não lhe interessem, a redução lhes permite eliminar valores de atributos que não satisfazem um dado predicado.

A operação:

E = reduce [A / predicate] E_1, onde E_1 é um tipo entidade e A é um atributo de E_1, criado como um tipo derivado de E, cujos oids são os mesmos daqueles de E_1. O esquema, relacionamentos, generalizações e ligações de conjunção E são derivados daqueles de E_1. O predicado deve envolver A ou seus atributos componentes. Ele deve também envolver outros atributos de E_1. Ele é definido como uma predicado de seleção. O valor de cada ocorrência de E é derivado dos valores das ocorrências correspondentes de E, mantendo em A somente os valores que satisfazem o predicado; o valor de outros atributos não é alterado.

Exemplo:

Listar vinhas (vineyards), exibindo os melhores anos da vinha anteriores a 1960.

reduce [bestyears / bestyears < 1960]

Vineyard

- o operador *union*[24] .

Diferente do operador da álgebra relacional, o qual é baseado em valor, mescla a população de duas entidades de acordo com seus oids. Ele é um operador binário com dois operandos principais. O ambiente do resultado (relacionamentos, generalizações e conjunções) é derivado de ambos os operandos. A operação: $E = E_1$ *union* E_2 onde E_1 e E_2 são dois tipos entidades, E é um tipo entidade derivado, cujos oids são os mesmos daqueles de E_1 acrescidos dos oids de E_2. O esquema de E é derivado da união-fusão dos esquema E_1 e E_2. Cada atributo A_i de E_1 (e de E_2) é também um atributo de E. Se E_2 não tem atributo com o mesmo nome, o atributo A_i de E é idêntico ao atributo A_i de E_1 (a única diferença é que A_i se torna opcional). Se E_2 também tem um atributo A_i, o atributo A_i de E é a união de ambos os atributos: seu

[22] Projeção.

[23] Redução.

[24] União.

domínio é a união de ambos os domínios, seu valor é a união multi-conjunto dos dois valores. Do mesmo modo, o relacionamento, generalização e ligações de conjunção E são derivados daqueles existentes em E_1 e E_2 pela união-fusão (fusão dos nomes idênticos e ligação dos mesmos tipos entidades).
Exemplo: Considere que a base de dados vinha contém, complementarmente vinhas vermelhas e brancas, verdes e outras.

Listar as vermelhas e brancas.

RedVineyard *union* WhiteVineyard.

O resultado desta consulta é uma subclasse de Vineyard, que está implicitamente ligada por um ligação de conjunção com WhiteVineyeard e RedVineyard, as quais tem dois atributos opcionais, *exposure*[25] e *heating*[26].

- operador *r-join*[27].

É um operador n-ário com um operando principal. Ele é usado para transformar um rede de tipos entidades dentro de uma estrutura hierárquica (uma entidade simples). Assim este operador agrupa dentro de uma entidade simples a informação dispersa sobre as entidades ligadas através de um relacionamento. De um ponto de vista orientado a objeto, um *r-join* pode recompor como um simples objeto um objeto complexo que foi desmembrado dentro de componentes objeto.
A operação: $E = E_1$ ü role_1 *r-join* ($A : R, E_2$ ü role_2 .... E_n ü role_n) onde role_i é o papel executado pelo tipo entidade E_1 no tipo relacionamento(a especificação dos papéis é somente obrigatório em relacionamento cíclicos) e A é um nome para o novo atributo complexo que será criado, E é um tipo entidade derivado, cujos oids são os mesmo daqueles de E_1 (E_1 é o operando principal). Os relacionamentos, generalizações e ligações de conjunção de E são derivados dos pertencentes a E_1. O esquema de E é constituído de atributos de E_1 mais um novo atributo multivalorado denominado A, que é derivado dos esquemas de E_1, ..., E_n e R. O valor de uma ocorrência de E é feita sobre os valores das ocorrências de E_1 mais um multiconjunto de E_1, ..., E_n e ocorrências de valores de R que estão ligados a E_1 (se existir algum).

Exemplo: listar cada plantador com sua respectivas vinhas.

WineGrower *r-join* (V : Harvest, Vineyard)

O resultado contém uma ocorrência para cada plantador.

- operador *i-join* [28]

É um operador binário com um operando principal. Ele é usado para agrupar dentro de uma entidade simples a informação dispersa sobre duas entidades delimitadas por uma ligação generalização ou conjunção. Permite ao usuário mesclar dois pontos de vista sobre um mesmo objeto do mundo real.
A operação: $E = E_1$ *i-join* ($A : E_2$), onde E_1 e E_2 são dois tipos entidades envolvidos por uma ligação *is-a* ou *may-be-a*, e A é uma nome para o novo atributo complexo que será criado, E é um tipo entidade derivado, cujos oids são os mesmo dos de E_1. Os relacionamentos, generalizações e ligações de conjunção de E são derivados daqueles de E_1. O esquema de E é feito dos atributos de E_1 mais um novo atributo complexo monovalorado, denominado A, o qual é derivado dos esquemas de E_2: ele agrupa todos os atributos de E_2. O valor de uma ocorrência de E é feita sobre os valores das ocorrências correspondentes de E_1 adicionado do valor das ocorrências de E_2 (se existir).

Exemplo:

Listar todas as informações sobre vinhas vermelhas.

RedVineyard *i-join* (V: Vineyard)

---

[25] Exposição.
[26] Aquecimento.
[27] Join-relacional.
[28] Junção-identidade.

O resultado contém uma ocorrência para cada vinha vermelha, com o atributo *exposure* adicionado a todos os atributos de *Vineyard*.

O *i-join* oposto:

Vineyard *i-join* (*R*: RedVineyard) deveria conter uma ocorrência para cada vinha, vermelha ou não (neste caso o atributo *exposure* no resultado é opcional).

- operador *product*.

É usado para colocar tipos entidades não relacionados dentro de uma entidade simples. Este resultado é similar ao operador de produto relacional NF2[29], porque cada entidade do primeiro operando está associado com todas as entidades do segundo operando. Ele é um operador binário com um operando principal (o primeiro). O produto é necessário para permitir ao usuário estabelecer dinamicamente ligações não previstas (não expressa por tipos relacionamento no esquema) entre tipos entidades não relacionados.

A operação: $E = E_1\ (A : E_2)$, onde E_1 e E_2 são dois tipos entidade, e A é um nome para o novo atributo complexo que será criado, E é um tipo entidade derivado, cujos oids são os mesmo dos de E_1 (E_1 é o operando principal). Os relacionamentos, generalizações e ligações de conjunção de E são derivados daqueles de E_1. O esquema de E é feito dos atributos de E_1 mais um novo atributo complexo multivalorado, denominado A, o qual é derivado do esquemas de E_2. O valor de uma ocorrência de E é feita sobre os valores das ocorrências correspondentes de E_1 adicionado do valor das ocorrências do multiconjunto de todas as ocorrências de E_2.

A álgebra ERC+ também contém dois operadores sintáticos, *renaming*[30] e *simplification*[31], o qual permite adequar o esquema de uma entidade com as regras do modelo ou da álgebra:

- operador *renaming* muda o nome de um atributo para prepar a fusão de atributos similiares durante uma união.
- operador *simplification* exclui complexidade desnecessária na estrutura que pode ser construída por outros operadores, a projeção em particular. A simplificação exclui um nível na estrutura complexa quando um atributo complexo tem somente um atributo componente (exceto se são ambos multivalorados).

Operadores derivados podem ser definidos para auxiliar o usuário a escrever consultas mais curtas.

O operador *difference*[32] (-) pode ser definido através da composição apropriada do produto, seleção e projeção. Uma *intersection*[33] pode também ser definida através de duas diferenças. Comumente a semântica destes operadores é formar uma nova população correspondente aquelas ocorrências dos primeiros operandos para os quais não há ocorrências no segundo operando com mesmo oid (diferença) ou para aquelas ocorrências dos primeiros operandos para os quais há uma ocorrência no segundo operando com o mesmo oids (interseção). Este dois operandos tem um operando principal, o primeiro.

O *r-join* é equivalente ao *outer join*[34] do modelo relacional: o resultado tem uma ocorrência para cada ocorrência do operando principal mesmo que ele não esteja presente no relacionamento. Um operador derivado útil é *simplification* que exclui do resultado todas as ocorrências não ligadas através do relacionamento. O *sel-r-*

---

[29] Modelo relacional aninhado que descreve estruturas que não estão na primeira forma normal.

[30] Operador para troca de nome.

[31] Simplificação.

[32] Diferença.

[33] Interseção.

[34] Junção externa.

join é equivalente a uma expressão ERC+ feito sobre um *r-join* seguido por uma seleção.

Exemplo:

Listar todos os plantadores de vinha que também são filhos de plantadores.

WineGrower / child-of sel-r-join (*F* : Family, WineGrower/parent-of)

Do mesmo modo, outro operador derivado, *v-join*[35], é uma junção teta NF2: ela agrupa dentro de um tipo entidade simples a informação dispersa sobre dois tipos entidades ligados por um predicado sobre seus atributos. É equivalente para uma expressão ERC+ feito sobre um produto seguido por uma redução e uma seleção.

Exemplo:

Listar os plantadores de vinha que tem o mesmo nome que o nome (identificador) de vinha.

WineGrower *v-join* [name=label] Vineyard

O resultado contém uma ocorrência para cada plantador de vinha que satisfaça a condição.

ERC+ incorpora duas linguagens básicas: a álgebra apresentada acima, e um cálculo. Estas linguagens básicas são as fundamentações básicas sobre as quais linguagens amigáveis ao usuário tais como SQL e linguagens gráficas são construídas (SPACCAPIETRA et al., 1995, p. 191-194).

Destaca-se na álgebra ERC+ o operador *r-join* pela sua importância em aplicações de consulta, pois a possibilidade de reduzir um conjunto de entidades, ligadas por tipos relacionamentos a uma única entidade permite uma grande flexibilidade em consultas, visto que se pode construir múltiplas visões hierárquicas sobre um mesmo esquema ERC+ exibido ao usuário e principalmente preservando a estrutura não planar obtida na resposta, portanto mantendo o conceito de objeto complexo do modelo. Este operador associado ao operador de redução dão especial suporte à aplicações de consulta e serão especialmente explorados pela aplicação de consulta implementada nesse trabalho.

O próximo capítulo apresenta algumas aplicações de interface para base de dados, baseadas em grafos, as quais utilizam modelos de dados orientados a objetos e

[35] Junção por valor.

relacionais, na interface com o usuário. Sobre estas aplicações será discutida a contribuição do protótipo VIQUEN, desenvolvido nesse trabalho.

3 INTERFACES PARA MANIPULAÇÃO DIRETA DE DADOS

A manipulação direta de dados, representados visualmente, é uma técnica poderosa para explorar dados operacionais e um grande número de ambientes de consulta para bases de dados têm sido propostos com esta finalidade. Esta seção apresenta breves descrições das abordagens de algumas aplicações correlatas a este trabalho, ressaltando e avaliando apenas suas características relativas ao SGBD utilizado, construção de consultas por manipulação direta e apresentação de resultados.

3.1 KALEIDOSCAPE

É uma implementação tridimensional de uma linguagem de consulta visual orientada a fluxo de dados, cujas consultas são traduzidas para linguagem de consulta de objetos[36] e que fornece suporte a qualquer SGBD que esteja em conformidade com o padrão *Object Data Management Group* - ODMG (MURRAY; NORMAN, 1998b). Essa é uma aplicação que objetiva preencher a lacuna de utilização formada entre linguagens textuais e ambientes de consulta e utiliza como sustentação uma linguagem de consulta visual denominada *Kaleidoquery* (MURRAY; NORMAN, 1998a).

O ambiente de interface acessa SGBDs através de um módulo independente, trazendo, a vantagem de que, quando um diferente SGBD for utilizado, somente o módulo que o acessa necessitaria ser modificado. Porém somente SGBDs orientados a objetos são suportados, pois as consultas formuladas no ambiente são traduzidas para *Object Query Language* (OQL).

O esquema apresentado ao usuário é composto de um conjunto de classes e suas extensões, que são representadas por ícones e uma descrição textual. É através destas representações que o usuário navega para compor suas consultas. Para impor restrições à população de uma classe o usuário seleciona um atributo desta e escolhe

[36] Object Query Language (OQL).

um operador dentre uma lista disponibilizada. Predicados envolvendo expressões lógicas são visualizadas usando uma representação gráfica de filtro e fluxo.

O resultado de uma consulta pode ser visto no ambiente tridimensional ou alternativamente mediante uma apresentação textual. Para ter acesso a este resultado o usuário deve navegar até a instância que deseja ver.

3.2 QUIVER

É uma implementação baseada em grafos onde cada nodo, arco e vértice tem uma representação visual associada e denota um construtor de consulta na aplicação, assim, por exemplo nodos representados por círculos sombreados denotam objetos (CHAVDA; WOOD, 1997).

Quanto à especificação de predicados, o usuário navega até o nodo e estabelece a restrição desejada, sendo que não são descritas representações para consultas envolvendo expressões lógicas.

Essa aplicação traduz consultas para OQL e fornece suporte a um SGBD orientado a objeto. A visualização da resposta é exibida como um subgrafo do grafo primário apresentado ao usuário.

3.3 QBD*

Através de uma interface gráfica para formulação de consulta esta aplicação usa o modelo entidade-relacionamento (P. CHEN, 1990) como modelo conceitual para interagir com o usuário. Um conjunto de primitivas gráficas são disponibilizadas para que, a partir do esquema conceitual, o usuário selecione o esquema envolvido na consulta, rotulado como *esquema de interesse*. A atividade de seleção do esquema de interesse tem como método mais comum a seleção dos objetos desejados, denominada *extração direta* (ANGELACCIO; CATARCI; SANTUCCI, 1990).

Após a seleção do esquema de interesse o usuário pode trabalhar na interpretação através de primitivas de navegação que formam a linguagem gráfica de consulta, que é uma linguagem visual aplicada sobre o esquema entidade-relacionamento. A idéia básica é decompor uma consulta dentro de passos elementares, expressos por um conjunto limitado de operações gráficas, tais como ícones, seleção de conceitos e navegação. É definida uma correspondência *um-para-um* entre uma operação gráfica e o construtor sintático da linguagem de consulta textual, a fim de expressar formalmente a sintaxe e a semântica da linguagem.

A estrutura geral de uma consulta está baseada na locação de um conceito distinto, chamado *conceito principal*, composto por uma entidade ou relacionamento, que pode ser visto como ponto de entrada para uma ou mais subconsultas, sendo que estas podem ser combinadas através dos operadores de união e interseção. A seleção das instâncias desejadas, ou seja, a formação do predicado é feita em um janela separada que contém todos os elementos envolvidos na comparação, isto é, constantes, atributos e operadores.

Finalmente, as consultas são traduzidas em expressões da álgebra relacional e, mediante um módulo de interface para SGBDs relacionais, a informação é extraída. Em ANGELACCIO, CATARCI e SANTUCCI (1990) não há descrição da forma, nem de interface gráfica utilizada para exibição dos dados que compõem a resposta à consulta.

3.4 SUPER - UMA INTERFACE PARA O MODELO ERC+

Projetado para o modelo ERC+, o protótipo SUPER é um ambiente que permite edição do modelo e formulação de consultas. Conforme abordado por DENNEBOUY et al. (1995). Suas principais características são:

- a interface para formulação da consulta utiliza o modelo ERC+ para exibir o esquema completo da base e para interagir com usuário;

- formulação de consulta projetada de forma que dispensa o usuário de conhecer a linguagem subjacente de manipulação de dados pois, as consultas são formuladas através do modelo conceitual;
- alta flexibilidade na implementação das tarefas, como por exemplo o usuário pode desfazer[37] parte de sua interação na consulta ou complementa-la;
- possibilidade de seleção entre diferentes níveis de interação de acordo com o perfil do usuário.

A formulação de consulta no SUPER consiste em:

- selecionar tipos objetos e relacionamentos, contidos no esquema completo, apresentado ao usuário. Os elementos selecionados são arranjados em um grafo que inicialmente pode ser cíclico, e que, posteriormente, é transformado em uma árvore. A escolha da raiz e os ciclos são removidos pelo usuário para evitar interpretações ambíguas entre diferentes pontos de vista possíveis;
- após a estruturação na forma de uma árvore o usuário escolhe quais atributos deseja na resposta;

FINALMENTE, O USUÁRIO ESTABELECE AS RESTRIÇÕES, ISTO É., OS PREDICADOS DESEJADOS. A FORMA DE UTILIZAÇÃO E REPRESENTAÇÃO DO PREDICADO DEPENDE DO TIPO DO OBJETO ENVOLVIDO. SE O PREDICADO É IMPOSTO SOBRE UM ESQUEMA EM PRIMEIRA FORMA NORMAL, ELE É APLICADO ÀS INSTÂNCIAS DO TIPO ENTIDADE ENVOLVIDO. PORÉM, QUANDO O PREDICADO É ESTABELECIDO SOBRE UM ATRIBUTO COMPLEXO, SOMENTE É APLICADO ÀS OCORRÊNCIAS DO ATRIBUTO E NÃO ÀS DAS INSTÂNCIAS DO TIPO ENTIDADE A QUE ELE PERTENCE, OU SEJA, NESTE CASO APLICA-SE O OPERADOR DE REDUÇÃO[38] DA ÁLGEBRA ERC+. EM AMBOS OS CASOS, PRIMEIRA FORMA NORMAL

---

[37] Operação *undo*.

[38] Veja operador de redução na subseção 2.2.2.

OU ATRIBUTO COMPLEXO, A REPRESENTAÇÃO DE UM ÚNICO PREDICADO É FEITA DIRETAMENTE UTILIZANDO ESTES OBJETOS, OS QUAIS APARECEM NO ESQUEMA SELECIONADO PELO USUÁRIO. TODAVIA, QUANDO MAIS DE UM PREDICADO É INSERIDO, A VISUALIZAÇÃO SE DÁ DE FORMA DIFERENCIADA: PARA ESQUEMAS EM PRIMEIRA FORMA NORMAL O MESMO OBJETO PODE SER USADO DUAS VEZES; JÁ, PARA REPRESENTAR O USO DE UMA SEGUNDA VARIÁVEL, NO CASO DE ATRIBUTOS COMPOSTOS, O ATRIBUTO DEVE SER DUPLICADO PARA QUE SE ATRIBUA UM NOVO PREDICADO A ELE, CONFORME FIGURA 3 E

- Figura 4, respectivamente;
- as consultas são submetidas a uma máquina abstrata ERC+ que interpreta a linguagem de consulta, a álgebra ERC+, e extrai os dados.

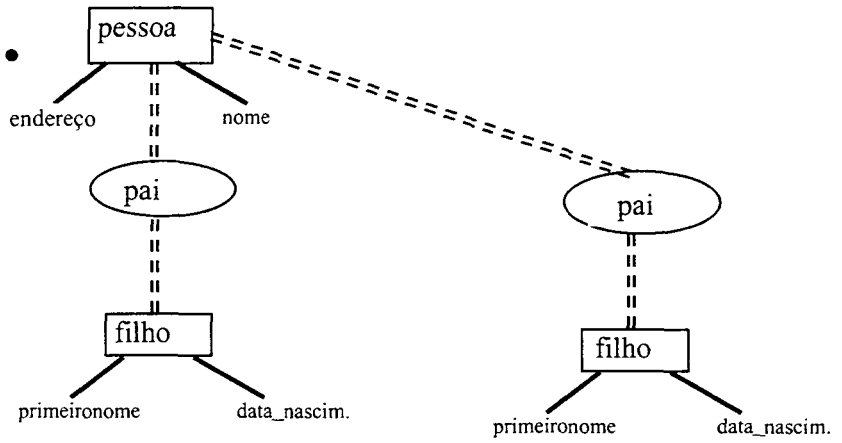


FIGURA 3: ESQUEMA EM 1ª FORMA NORMAL.

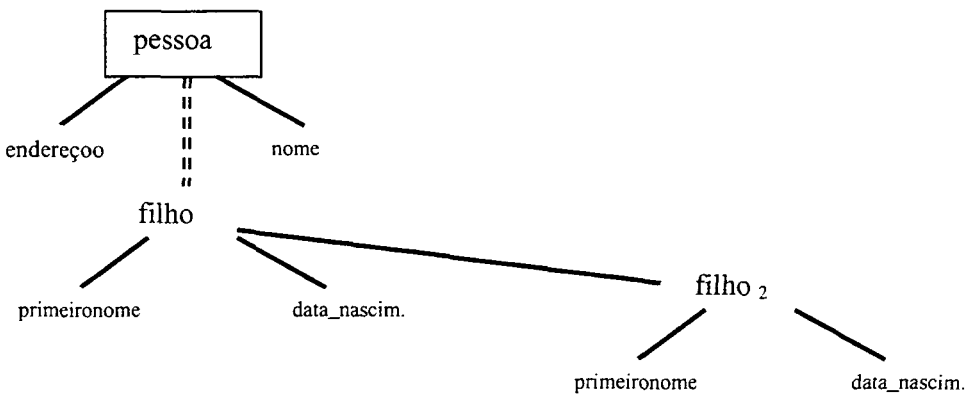


FIGURA 4: ESQUEMA COM ATRIBUTO COMPLEXO *FILHO*.

3.5 ANÁLISE CRÍTICA

Tal como diversas outras aplicações que utilizam modelos orientados a objetos como modelo conceitual para interação com o usuário, as aplicações KALEIDOSCAPE e QUIVER traduzem suas consultas somente para linguagens de manipulação de dados orientadas a objetos e, portanto, possuem interface exclusivamente com SGBDs orientados a objetos. É uma característica que exclui o acesso à grande base relacional existente atualmente.

Tipicamente o grupo de aplicações de consulta para acesso a SGBDs relacionais, ao qual QBD* pertence, dedicam-se somente à formulação de consultas, desconsiderando a exibição de respostas. Essas aplicações apresentam respostas planares para que o usuário as interprete ou formate sua exibição em ferramentas de relatórios, exigindo conhecimentos razoáveis de ordenação por múltiplos atributos, quebras em relatórios, etc. Ainda com relação a QBD*, esta emprega o modelo entidade relacionamento como modelo conceitual, o qual não suporte objetos complexos em sua semântica, característica que o torna deficitário em relação a aplicações de consulta que se valem de modelos orientados a objetos/objeto-relacionais[39].

O protótipo SUPER fornece suporte completo ao modelo ERC+ e um importe conjunto de características desejáveis em aplicações de consulta. Entretanto, observamos que o mesmo utiliza uma máquina abstrata ERC+ para interpretar a linguagem de consulta - a álgebra ERC+, que constitui um camada adicional de software e não há acesso a SGBDs relacionais, nem para traduzir esquemas relacionais

[39] Veja comparação entre modelos no capítulo 2.

em esquemas semânticos, nem para extração de dados. Assim, não há acesso ao vasto legado relacional.

A abordagem do protótipo, que será apresentado nas próximas seções deste trabalho, propõe: utilizar um modelo conceitual para exibir esquemas com os quais o usuário vai interagir, extraindo-o de esquemas relacionais ao invés de SGBDs orientados a objeto; e efetuar acesso nativo a LMD suportada pelo SGBD, em nosso caso o ORACLE, exibindo respostas de forma a preservar a visão de objetos dada pelo modelo conceitual. Assim, o protótipo permitirá acesso, mediante um esquema conceitual, ao vasto legado de informações relacionais, detido por um dos principais SGBDs comerciais do mercado.

4 ENGENHARIA REVERSA

Considerando a hegemonia do modelo e dos SGBDs relacionais nas últimas décadas, temos um vasto legado de informação construído sob este modelo. No entanto, a conversão destes esquemas para esquemas baseados em modelos ricos semanticamente, emergentes atualmente, como modelo de objetos e objeto relacional, se torna honerosa, tendo em vista que tanto os esquemas como as aplicações dos sistemas de informação devem ser alterados para uma destas abordagens. O propósito dessa engenharia reversa está em obter um esquema construído sob o modelo relacional, armazenado em um SGBD , e traduzi-lo para um esquema sob o modelo ERC+ para que seja utilizado pela aplicação de consulta, disponibilizando a semântica do modelo de objetos e preservando o investimento feito sobre bases relacionais.

A engenharia reversa aqui proposta utiliza-se das definições de regras de integridade sobre esquemas relacionais, sob os quais seus projetos foram construídos e que são suportadas pelos dicionários de dados dos SGBDs relacionais. Adicionalmente o processo de conversão acrescenta conceitos para efetuar o mapeamento para o modelo ERC+ e, assim, disponibilizar uma interface para consulta, mediante um modelo de rico semanticamente. Essa abordagem para tradução define regras de tradução e refinamentos aplicadas sobre as informações existentes no dicionário de dados de um SGBD relacional[40] .

Entende-se que o dicionário de dados mantém, conforme descrito por SILBERCHATZ, KORTH e SUDARSHAN (1999), pelo menos as seguintes informações:

- nomes das relações;
- nomes dos atributos de cada relação;
- domínios dos atributos;

[40] Os metadados dos esquemas relacional e objeto relacional são mantidos de forma distinta no dicionário de dados, assim nossa proposta atinge somente esquemas relacionais.

- regras de integridade, tais como restrições de domínio e integridade referencial.

Há uma correspondência natural entre os conceitos do modelos relacional e do de objetos (KLAS, 1994):

- relação → entidade;
- atributo → atributo[41];
- tupla → instância.

Os conceitos de simples equivalência podem, em muitos casos, produzir um esquema ERC+ inadequado ao maior poder de expressão deste modelo, comparativamente ao modelo relacional. Objetivando um esquema ERC+ mais próximo do mundo real percebido pelo usuário, o mapeamento aqui proposto, considera a simples equivalência pela transformação sintática e o posterior refinamento para atingir um enriquecimento semântico na tradução. Esse métodos serão abordados nas subseções seguintes, sob as notações da definição formal do modelo ERC+, descritas na seção 2.2.

Os exemplos aqui apresentados foram gerados através do protótipo VIQUEN, desenvolvido nesse trabalho. Os mapeamentos em SQL, nesta seção apresentados como exemplo, serão discutidos detalhadamente na seção 5.

A fim de expor o processo de engenharia reversa empregado, será utilizado o exemplo de uma implementação relacional típica, construída sob o sistema de informação de uma propriedade agrícola, o qual será apresentado parcialmente nesta seção à medida em que as regras forem formuladas.

4.1 MAPEAMENTO POR SIMPLES EQUIVALÊNCIA

Este mapeamento será introduzido mediante o exemplo de uma relação, denominada *Propriedade*, pertencente a um esquema relacional.

---

[41] No modelo ERC+ um atributo pode ser atômico ou complexo.

Seja a relação *Propriedade*, descrita como segue: *Propriedade*(NrInscricao, Nome, Local). Esta pode ser traduzida para uma entidade no modelo ERC+ utilizando a correspondência natural de relação para entidade e entre atributos. Assim, dando origem a um tipo entidade *PROPRIEDADE*, no modelo ERC+, composto pelos atributos atômicos *Nome* e *Local*. Ainda pode-se efetuar a tradução da chave primária da relação *Propriedade*, aqui formada pelo atributo *NrInscricao*, como identificador de entidade. Para isto se deve assumir a correspondência direta entre tupla e instância. No entanto observa-se que o identificador de entidade é único somente dentro da população do tipo entidade *PROPRIEDADE*.

Sobre estas considerações podemos definir uma regra geral de tradução do modelo relacional para o modelo ERC+, conforme segue:

Seja um esquema relacional $M_R=\{R_1,..,R_N\}$; $R_i(a_{i,1},a_{i,2},...,a_{i,k})$ onde $R_i \in M_R$, $a_{i,k}$ é um atributo atômico; PK_i é a relação de atributos que define a chave primária de R_i e $1 \leq i \leq N$; RN_i o nome da relação R_i. Utilizando a simples equivalência temos a seguinte regra para traduzir R_i em um esquema ERC+:

> *Regra de tradução direta:-*
>
> Cada relação R_i define um tipo entidade E_i = (RN_i, S_i, gen_i, $muid_i$, P_i); onde RN_i define o nome do tipo entidade E_i ; S_i é o esquema $sch(E_i)=\{ e_{ij} \}$, $1 \leq j \leq k_i$, em que e_{ij} é um atributo atômico correspondente ao atributo atômico $a_{i,k}$, sendo o domínio $d(e_{ij}) \in ED$ um domínio elementar igual ao domínio $d(a_{i,k})$; $gen_i=\varnothing$[42]; $muid_i=\varnothing$[43]; P_i é a população do tipo entidade $P_i=\{(t[PK_i], t[R_i])\}$, cada par formado pelas tuplas $t[PK_i]$ e $t[R_i]$ correspondem

[42] Sem esquema de generalização.
[43] Sem esquema de ligação por conjunção.

respectivamente a um identificador de entidade[44] e ao valor de cada entidade $E \in E_i$.

Nesta regra considerou-se:

- a tradução direta de relação para entidade e entre atributos atômicos;
- a transformação de uma ocorrência de chave primária para definir o identificador de um tipo entidade. O conceito de identificador de objeto é importante para o modelo ERC+, e portanto, uma eventual função de mapeamento $f(t[PK_i])$pode ser considerada para gerar um identificador de objeto para cada nova tupla $t[PK_i]$. Porém, para o processo de engenharia reversa, neste trabalho foi utilizado somente o conceito de identificador de entidade, traduzido pelo valor da chave primária a fim de atingir as implementações relacionais implementadas sem identificador de tupla[45], onde se encontra a maioria do legado de esquemas relacionais. Essa consideração implica em uma nova execução do processo de engenharia reversa para refletir eventuais alterações dos componentes da chave primária para o modelo traduzido, entretanto, isto é irrelevante para os propósitos deste trabalho;
- ligação generalização/especialização e ligação por conjunção são desconsiderados nesta regra.

A tradução direta é suficiente para traduzir atributos atômicos do modelo relacional para atributos atômicos do modelo ERC+, mas não o é para traduzir relações para entidades, pois não contribui para o enriquecimento semântico. Ao simplesmente serem traduzidas relações como entidades está sendo desconsiderado a maior semântica do modelo ERC+, que ainda contempla em sua estrutura ligação por generalização ou conjunção, tipos relacionamento e atributo complexo. Esse tipo de

---

[44] Identificador de uma entidade $E \in E_i$.

[45] Equivalente diretamente ao identificador de objeto empregado por modelos de objeto e objeto-relacionais.

tradução pode produzir um modelo ERC+ inadequado. Considere-se, por exemplo, a subdivisão de uma propriedade em áreas, denominadas *glebas*, expressa pela seguinte relação *GLEBA*(NrInscricao, Número, Area). A tradução pela regra de tradução direta levaria a uma entidade *GLEBA*, ao passo que uma modelagem em ERC+ mais adequada a semântica do modelo traria *GLEBA* como um atributo complexo de *PROPRIEDADE*.

Análises como esta propiciam um enriquecimento semântico levando a conceitos coerentes com o poder de expressão do modelo ERC+. Serão abordadas, na próxima subseção, regras de refinamento para atingir este objetivo.

4.2 REFINAMENTO DO MAPEAMENTO

4.2.1 Atributo complexo

O primeiro refinamento que será observado é o mapeamento de relações para atributo complexo. Considere-se a subdivisão de uma propriedade em glebas e desta em lavouras, a qual mantém as culturas de diferentes safras e anos. A Tabela 1 mostra a descrição relacional deste subesquema.

TABELA 1: SUBESQUEMA PROPRIEDADE EM UMA PROPRIEDADE AGRÍCOLA

Relação	Atributos	Chave Primária	Chave Estrangeira
Propriedade	NrInscricao, Nome, Local	NrInscricao	
Gleba	NrInscricao, Numero, Area	NrInscricao, Numero	Propriedade→ NrInscricao
Lavoura	NrInscricao, Numero, IdCultura, Ano, Safra, QtdeProduzida	NrInscricao, Numero, IdCultura	Gleba→ NrInscricao, Numero

Dentro deste contexto, observa-se, quanto à relação *GLEBA*:

- a formação de sua chave primária é composta de um atributo próprio *Numero* e do atributo *NrInscricao* pertencente a relação *PROPRIEDADE*, i.e., *GLEBA* tem dependência de identificador[46];
- há somente uma chave estrangeira em *GLEBA*, composta pelo atributo *NrInscricao*, que liga a relação *GLEBA* à relação *PROPRIEDADE*, e os atributos da chave estrangeira de *GLEBA* são os mesmos da chave primária de *PROPRIEDADE*;
- a relação *PROPRIEDADE* à qual *GLEBA* está ligada é uma relação que não possui nenhuma chave estrangeira.
- a relação *LAVOURA*, que tem dependência de identificador da relação *GLEBA*, contém somente uma chave estrangeira, justamente a que as relaciona.

Quanto às características da relação *LAVOURA*:

- a formação de sua chave primária é composta igualmente de um atributo próprio que identifica a cultura, *IdCultura*, e dos atributos *NrInscricao* e *Numero*, pertencentes à relação *GLEBA*, isto é, *LAVOURA* também tem dependência de identificador;
- há somente uma chave estrangeira em *LAVOURA*, composta pelos atributos *NrInscricao* e *Numero* que a ligam com a relação *GLEBA*;
- a relação *GLEBA* à qual *LAVOURA* está ligada, possui uma única chave estrangeira, sendo que sua chave primária contém todos os componentes da chave estrangeira de *LAVOURA*;
- não há nenhuma outra relação com dependência de identificação de *LAVOURA*.

---

[46] Este conceito é utilizado também no modelo Entidade-Relacionamento para denotar aquelas entidades que não podem ser identificadas inequivocamente por seus próprios atributos[7].

A similaridade entre as características das relações *GLEBA* e *LAVOURA* permite se estabelecer um refinamento para traduzir em atributo complexo uma relação que:

- na composição de sua chave primária contenha, pelo menos, um atributo próprio e um ou mais atributos de outra relação;
- contenha somente uma chave estrangeira, que pode ser simples ou composta;
- esteja ligada a outra relação[47], cuja chave primária seja a mesma de sua chave estrangeira, e que aquela, no máximo, contenha uma chave estrangeira;
- se houver alguma outra relação com independência de identificação direta ou indiretamente a esta[48], aquela só deve possuir uma única chave estrangeira, ou seja, unicamente a chave estrangeira que as liga[49].

Assumindo a regra de tradução direta da relação *PROPRIEDADE* e este refinamento a relação *GLEBA* é traduzida para um atributo complexo de *PROPRIEDADE* e a relação *LAVOURA* para um atributo complexo de *GLEBA*, representado graficamente, conforme notação do modelo, na Figura 5.

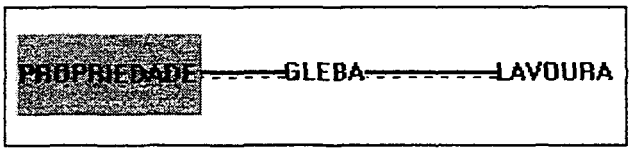


FIGURA 5: ENTIDADE PROPRIEDADE

Para obter-se a população da entidade *PROPRIEDADE* foi utilizado um mapeamento em SQL[50] que emprega um operador multiconjunto *cursor*[51], através do

---

[47] Através de sua única chave estrangeira.

[48] Indiretamente se a dependência é dada hierarquicamente por uma terceira relação que contém mais de uma chave estrangeira.

[49] Caso contrário a relação deve ser traduzida como um tipo entidade pela regra de tradução direta.

[50] A descrição completa deste mapeamento pode ser vista na subseção 5.2.1.

qual foram obtidas as ocorrências dos atributos complexos. Este mapeamento é exibido a seguir e a Figura 6 apresenta uma instância de sua população.

```
select    cursor( select NOME, LOCAL ,
                  cursor( select NUMERO, AREA ,
                        cursor( select ANO, SAFRA, IDCULTURA, QTDEPRODUZIDA
                              from LAVOURA LAVOURA
                               where GLEBA.NRINSCRICAO=LAVOURA.NRINSCRICAO
                                    and GLEBA.NUMERO=LAVOURA.NUMERO  ) LAVOURA
                        from GLEBA GLEBA
                         where PROPRIEDADE.NRINSCRICAO=GLEBA.NRINSCRICAO  ) GLEBA
                     from PROPRIEDADE  ) PROPRIEDADE from dual
```

Face ao apresentado até aqui, pode-se estabelecer uma regra geral para este refinamento, conforme segue:

Seja o esquema relacional $M_R = \{R_1,..,R_N\}$; $R_i(\underline{A}, B, ...)$ $R_i \in M_R$, $1 \leq i \leq N$; $R_j(\underline{A} \subseteq D)$, $R_j \in M_R$, $1 \leq j \leq N$ e $i \neq j$; RN_j o nome da relação R_j; O_i um objeto (tipo entidade ou atributo complexo), obtido pelo mapeamento de R_i; o conjunto de atributos $A = \{a_1,...,a_k\}$ chave primária em R_i e a única chave estrangeira em R_j ; $F_j(A)$ a cardinalidade máxima[52] permitida a chave estrangeira A em R_j; A C a chave primária de R_j em que $C=\{c_1,...,c_k\}$; $B=\{b_1,...,b_k\}$ e $D=\{d_1,...,d_k\}$ são conjuntos de atributos não componentes de chave primária, com a possibilidade de conjunto vazio. Utilizando a definição de atributo complexo do modelo ERC+ temos a seguinte regra de refinamento:

Regra de refinamento 1:

Cada relação R_j define um atributo complexo $Z_j=(O_i, str_j, inst_j)$, onde:

- O_i é o objeto ao qual Z_j está associado;

[51] Este operador faz parte da linguagem de manipulação de dados do SGBD objeto relacional Oracle.

- str_j=(RN_j, 0[53], $F_j(A)$, $comp_j(S)$, d_j (S)) é a estrutura de Z_j; RN_j define o nome da estrutura; se $F_j(A)$=1 define um atributo monovalorado caso contrário um atributo multivalorado; $comp_j(S)$ = $(C\ D\ )$ é a composição da estrutura em que $C\ D$ são conjuntos de atributos atômicos[54] traduzidos pela *regra de tradução direta*; $d_j(S)$ é o domínio de $comp_j(S)$ traduzidos pela *regra de tradução direta*;
- $inst_j(Z_j)$ é a instância do atributo gerada por uma função que associa cada ocorrência de O_i ao(s) valor(es) do atributo; $inst_j(Z_j)$ é definido por $inst_j(Z_j)$: $t[R_i]$ $M_{set}(\pi_{Rj(C\ D)}\ (R_i\ |X|\ R_j))$, em que M_{set} é um operador multiconjunto que associa a cada tupla $t[R_i]$ a instância do atributo Z_j [55]

Tal que não há nenhuma relação $R_k \in M_{R,}$ onde $1 \leq k \leq N$, $i \neq j \neq k$ e $i \neq k$, com dependência de identificação direta ou transitivamente à R_j com mais de uma chave estrangeira.

---

[52] A cardinalidade máxima pode ser estabelecida no dicionário de dados através de restrição de unicidade para o atributo se A é atômico ou mediante um índice com restrição de unicidade se A é composto.

[53] Não há informação no dicionário para determinar qualquer outra cardinalidade mínima, assim assumimos zero.

[54] Conforme observamos na subseção 4.1 os atributos atômicos são obtidos por tradução direta.

[55] Apesar de o operador ser de multiconjunto quando aplicado a engenharia reversa sempre gera um conjunto para as ocorrências de R_j e nunca um multiconjunto, pois a restrição de chave primária imposta a esta relação, pelo modelo relacional, inclui o atributo C o que o faz único para cada tupla $t[R_i]$.

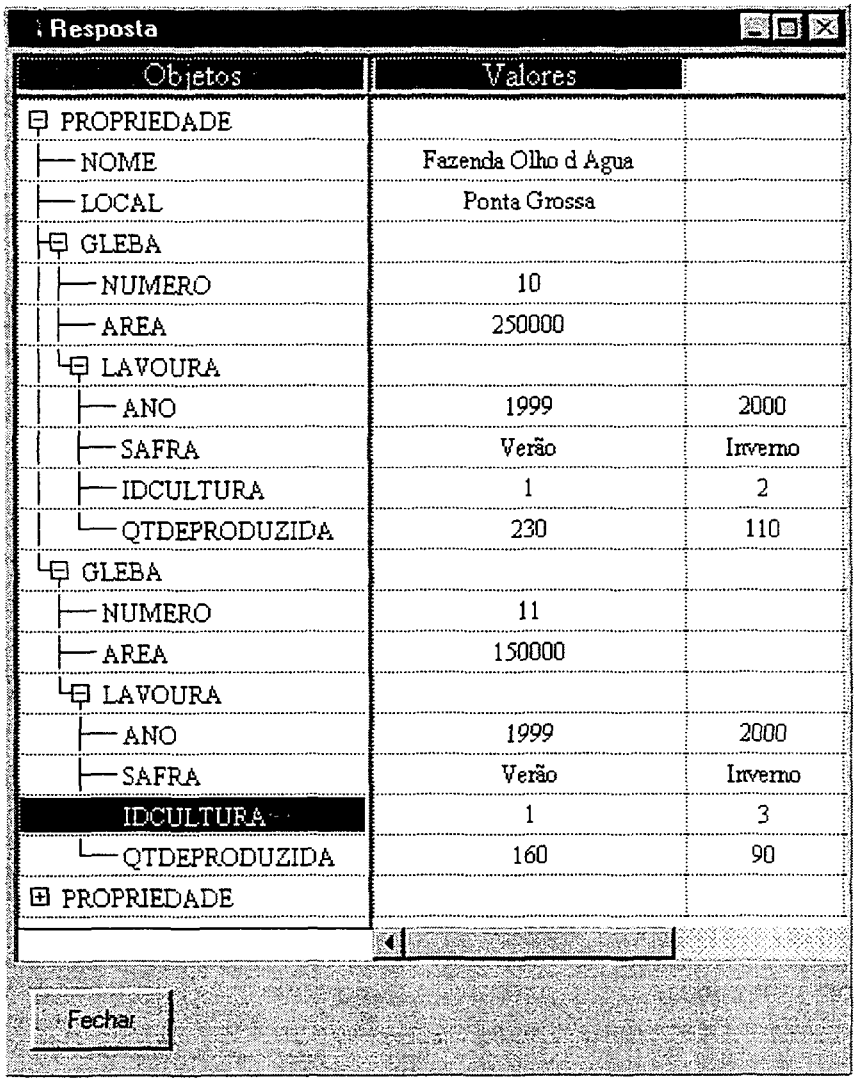


FIGURA 6: UMA INSTÂNCIA DO TIPO ENTIDADE PROPRIEDADE

4.2.2 Herança

O modelo ERC+, conforme apresentamos na seção 2.2.2, disponibiliza herança através de ligações do tipo *is-a*[56] ou *may-be-a*[57] , a seguir definiremos regras de refinamento para ambos.

[56] Especialização/Generalização.
[57] Ligação através de conjunção.

4.2.2.1 Ligação de generalização/especialização

Considerando-se outro subesquema de uma propriedade agrícola, podemos ter proprietários para as propriedades agrícolas e possíveis especializações deste como pessoas físicas e jurídicas, implementados no modelo relacional conforme a Tabela 2.

TABELA 2: SUBESQUEMA PROPRIETÁRIO E SUAS ESPECIALIZAÇÕES.

Relação	Atributos	Chave Primária	Chave Estrangeira
Proprietario	Identificador, Nome, Cidade,UF	Identificador	
Fisica	Identificador, CPF, Data_Nascimento, RG	Identificador	Proprietario →Identificador
Juridica	Identificador, CGC	Identificador	Proprietario →Identificador

As características das relações *FISICA* e *JURIDICA*:

- nenhuma das duas têm atributo(s) próprio(s) entre os componente(s) de sua chave primária, somente o atributo *Identificador* referente à relação *PROPRIETARIO*;
- há uma chave estrangeira com a mesma composição da chave primária, formada pelo atributo *Identificador* da relação *PROPRIETARIO* , a qual tem como chave primária também o atributo *Identificador*.

Em conformidade a estas propriedades segue um refinamento para traduzir como especialização uma relação que:

- não possua atributos próprios na composição de sua chave primária;
- possua uma chave estrangeira com a mesma formação de sua chave primária e a relação referenciada por esta chave estrangeira contenha os mesmos componentes em sua chave primária.

Este refinamento nos permite traduzir as relações *FISICA* e *JURIDICA* como especializações de *PROPRIETÁRIO*. A Figura 7 mostra a representação gráfica deste subesquema ERC+.

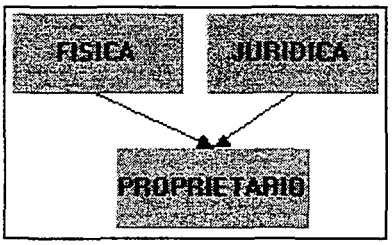


FIGURA 7: SUBESQUEMA PROPRIETÁRIO E SUAS ESPECIALIZAÇÕES

A regra geral para traduzir uma relação para entidade ligada por uma generalização/especialização é dada a seguir:

Seja o esquema relacional $M_R = \{R_1,..,R_N\}$; $R_i(\underline{A}, B, ...)$ $R_i \in M_R$, $1 \leq i \leq N$; $\underline{A}$ é chave primária em R_i; E_{Ri} um tipo entidade resultante do mapeamento de R_i; T'_A o conjunto de ocorrências de A em R_i; $R_j(\underline{A}\ C)$ $R_j \in M_R$, $1 \leq j \leq N$ e $i \neq j$; A é chave primária em R_j e também chave estrangeira que referencia a relação R_i; E_{Rj} um tipo entidade resultante do mapeamento de R_j; T''_A o conjunto de ocorrências de A em R_j, onde $A = \{a_1,...,a_n\}$:

> *Regra de refinamento 2:*
>
> Existe uma ligação de generalização $E_{Ri} \in gen(E_{Rj})$ determinada pela dependência de inclusão de T'_A em relação a T''_A, i.e., $T'_A \subseteq T''_A$[58].

A dependência de inclusão T'_A, especificada no *refinamento 2*, é garantida pela existência, na mesma relação, de uma chave estrangeira com composição igual a da chave primária, visto que a chave primária restringe sua ocorrência a valores não nulos e a chave estrangeira a ocorrência do valor em T''_A antes da inclusão em T'_A.

[58] Podemos obter a mesma regra pela declaração de A como chave estrangeira de R_i.

4.2.2.2 Ligação por conjunção (*May-be-a*)

Uma implementação relacional típica para este tipo de ligação deste tipo é colocada a seguir:

TABELA 3: SUBESQUEMA RELACIONAL DE UMA LIGAÇÃO POR CONJUNÇÃO

Relação	Atributos	Chave Primária	Chave Estrangeira
Proprietario	Identificador, Nome, Endereco, Cidade, UF	Identificador	
Produtor	RegistroRural, Nome, Endereco, Identificador	RegistroRural	Proprietario → Identificador

Adicionalmente ao esquema acima, considera-se a existência de restrição de unicidade e permissão de valores nulos sobre o atributo que compõe, em *PRODUTOR*, a chave estrangeira *Identificador*.

A implementação convencional[59] acima não permite diferenciar ligações do tipo *um-para-um* (*ou zero*) de ligações do tipo *may-be-a*. Tomando um outro exemplo, onde ocorre uma ligação de casamento entre duas relações hipotéticas *Homem* e *Mulher*, em um regime monogâmico, poderíamos ter uma implementação similar à apresentada acima, entretanto sem a mesma semântica, pois, no caso de proprietário e produtor, o que se deseja expressar é que um produtor pode, eventualmente também ser proprietário e vice-versa e, no último exemplo uma eventual associação entre *Homem* e *Mulher*. Todavia, nossa implementação de engenharia reversa se propõe a utilizar, também, implementações típicas como a apresentada na Tabela 3 e usar exclusivamente informações do dicionário de dados dos SGBDs para efetuar a tradução das mesmas. Assim, para casos iguais a esse, assume-se uma ligação por

[59] Uma implementação não convencional poderia se utilizar de identificadores de objetos ou tuplas o que permitiria a identificação de uma ligação *maybe*, semanticamente mais adequada.

conjunção[60], e deixa-se para o usuário modificar esta interpretação para um tipo relacionamento, quando for o caso.

Sob essas considerações o subesquema da Tabela 3 é traduzido como segue:

- pela regra de tradução direta estabelece as relações *PROPRIETARIO* e *PRODUTOR* como tipos entidades;
- uma ligação por conjunção entre elas ("*may-be-a*") entre elas, sendo a representação de cardinalidade do modelo ERC+ definida pela restrição de cardinalidade e nulos impostos sobre a chave estrangeira, *Identificador*, no dicionário de dados. Neste caso *um-para-um ou zero.*

A Figura 8 exibe a representação gráfica desta tradução.

FIGURA 8: LIGAÇÃO *MAY-BE-A* ENTRE *PROPRIETARIO* E *PRODUTOR*

Enfatiza-se que para este caso não temos uma regra inequívoca para ligações por conjunção e sim uma regra para determinar uma ligação *um-para-um ou zero* entre entidades , a qual assumiu-se como uma ligação *may-be-a*. Segue a regra geral:

Seja o esquema relacional $M_R = \{R_1,..,R_N\}$; $R_i(\underline{A}, B, ...)$ $R_i \in M_R$, $1 \leq i \leq N$; $\underline{A}$ é chave primária em R_i; E_{Ri} um tipo entidade resultante do mapeamento de R_i; $R_j(\underline{C}\ D\ A)$ $R_j \in M_R$, $1 \leq j \leq N$ e $i \neq j$; C é chave primária em R_j e A é chave estrangeira R_j que referencia a relação R_i; $F_j(A)$ a cardinalidade da chave estrangeira A em R_j; E_{Rj} um tipo entidade resultante do mapeamento de R_j; onde $A = \{a_1,...,a_n\}$:

[60] *"may-be-a".*

Regra de refinamento 3:

Existe uma ligação *um-para-um ou zero* entre E_{Ri} e E_{Rj} se, e somente se, $0 \leq F_j(A) \leq 1$.

A possibilidade de valores nulos sobre $F_j(A)$, é assegurada no dicionário de dados, pela ausência de restrições a valores nulos. A restrição de unicidade sobre $F_j(A)$ pode ser assegurada de duas formas: se A é composto por um conjunto de atributos, através de um índice com restrição de unicidade sob o conjunto de atributos de A que formam a chave estrangeira em R_j; se A é formado por um único atributo, através de uma restrição de unicidade sobre o atributo em R_j.

4.2.3 Tipo Relacionamento

Considere, na Tabela 4, outro subesquema de uma propriedade agrícola.

TABELA 4:RELACIONAMENTO ENTRE PROPRIETARIO E PROPRIEDADE

Relação	Atributos	Chave Primária	Chave Estrangeira
Proprietario	Identificador, Nome, Endereco, Cidade, UF	Identificador	
Possui	Identificador, NrInscricao, DataAquisicao	Identificador, NrInscricao	Proprietario → Identificador Propriedade → NrInscricao
Propriedade	NrInscricao, Nome, Local	NrInscricao	

Observa-se como característica deste subesquema:

- no relacionamento da Tabela 4 a relação *POSSUI* tem a sua chave primária composta dos atributos *Identificador* e *NrInscricao* respectivamente, das relações *PROPRIETARIO* e *POSSUI*; portanto, tem dependência de identificador sob estas duas relações;

- as duas chaves estrangeiras juntas compõem a chave primária da relação *POSSUI*;

Podemos definir um refinamento para traduzir para um tipo relacionamento uma relação que:

- na composição de sua chave primária referencie duas ou mais chaves estrangeiras.

A Figura 9 exibe esta tradução para o modelo ERC+.

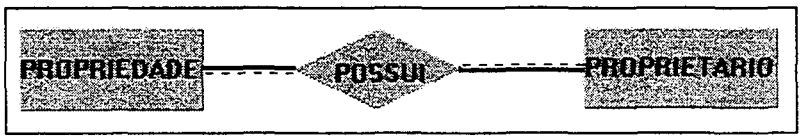


FIGURA 9: TIPO RELACIONAMENTO *POSSUI*

A regra geral para este refinamento é dada a seguir.

Seja o esquema relacional S_R[61]$=\{R_1,..,R_N\}$; $R_i(\underline{A}\ B)$ $R_i \in M_R$ e $1 \leq i \leq N$; E_{Ri} um tipo entidade resultante do mapeamento de R_i; $R_j(\underline{C}\ D)$ $R_j \in M_R$ e $1 \leq j \leq N$; E_{Rj}, um tipo entidade resultante do mapeamento de R_j; $R_k(\underline{A}\ \underline{C}\ F)$ $R_k \in M_R$ e $1 \leq k \leq N$; PK_k é a relação de atributos que define a chave primária de R_k; RN_k o nome da relação R_k; $\underline{A}$ chave primária de R_i; $\underline{C}$ chave primária de R_j; $\underline{A}\ \underline{C}$ compõem a chave primária de R_k [62]; $i \neq k$ e $j \neq k$. Temos a seguinte regra para traduzir R_k para um tipo relacionamento:

Regra de refinamento 4:

Cada relação R_k é traduzida para um tipo relacionamento $P_k=(\ RN_k, pet_k, sch_k, pop_k\ )$, onde:

- RN_k é o nome do tipo relacionamento;
- pet_k o conjunto de tipos entidade participantes, definido por $pet_k\ (P_k)=\{(\ E_{Ri}, E_{Rj})\}$;

[61] Utilizamos S_R para denotar um esquema relacional.

[62] A chave primária pode conter também atributos próprios desta relação.

- sch_k é o esquema definido por $\text{sch}_k(P_k)=\{F\}$, em que F é um conjunto de atributos atômicos traduzidos pela *regra de tradução direta*.
- $pop_k(P_k)=\{(\ t[PK_k],\ \text{poc}_k,\ t[R_k])\}$, em que cada par formado pelas tuplas $t[PK_k]$ e $t[R_k]$ correspondem respectivamente ao identificador e ao valor do relacionamento; $poc_k=\{E',\ E''\ \}$é o conjunto de entidades ligadas em que $E' \in E_{Ri}$ e $E'' \in E_{Rj}$.

A cardinalidade máxima das ligações entre os tipos entidades E_{Ri} , E_{Rj} e o tipo relacionamento P_k é determinada pela restrição de unicidade imposta a cada uma das chaves estrangeiras de R_j, se houver restrição de unicidade a cardinalidade máxima é um, caso contrário *n*. A cardinalidade mínima é um, pois as chaves estrangeiras são componentes da chave primária, logo não podem ser nulas em R_j.

4.2.4 Tipo Entidade Fraca

Considere as relações da Tabela 5.

TABELA 5: IMPLEMENTAÇÃO RELACIONAL DE UMA ENTIDADE FRACA.

Relação	Atributos	Chave Primária	Chave Estrangeira
Possui	Identificador, NrInscricao, DataAquisicao	Identificador, NrInscricao	Proprietario → Identificador Propriedade → NrInscricao
Registro_Imovel	IdContrato, Identificador, NrInscricao	IdContrato, Identificador, NrInscricao	Possui → Identificador, NrInscricao

Se compararmos as características da relação *REGISTRO_IMOVEL*, quanto a dependência de chave estrangeira e identificação, com as das relações *GLEBA* e

LAVOURA, elas são bastante similares. Assim, em uma primeira análise, a relação *REGISTRO_IMOVEL* satisfaz os requisitos do refinamento que traduz uma relação para atributo complexo. Entretanto a relação *REGISTRO_IMOVEL* tem dependência de identificador da relação *POSSUI*, a qual pela regra de refinamento apresentada na subseção 4.2.1, tem como tradução um tipo relacionamento, ao passo que *GLEBA* e *LAVOURA* tem dependência de identificação da relação *PROPRIEDADE,* traduzida pela *regra de tradução geral* para um tipo entidade. Essa diferenciação permite-se estabelecer um novo refinamento, para determinar como um *tipo entidade fraca* uma relação que tenha as mesmas características apresentadas na regra de refinamento de atributo complexo, porém, que tenha dependência de identificador de uma relação que foi traduzida para um *tipo relacionamento*. A Figura 10 exibe esta tradução.

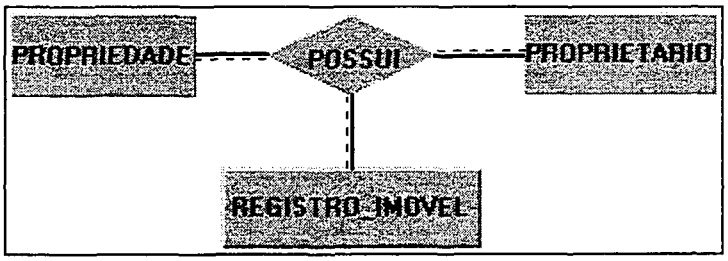


FIGURA 10: TIPO ENTIDADE FRACA *REGISTRO_IMOVEL*

Sob as mesmas definições da *regra de refinamento 1*, temos:

> *Regra de refinamento 5:*
> Cada relação R_j , com dependência de identificação de R_i define um entidade fraca E_j, se a tradução de R_i implica um tipo relacionamento O_i.

Nesse capítulo foi descrita uma regra de tradução direta e vários refinamentos para traduzir um esquema relacional em um esquema ERC+. Sob essas regras, o apêndice A contém a implementação de um conjunto de visões que são

criadas para formar os objetos do esquema ERC+, a partir do dicionário relacional mantido pelo SGBD. Na próxima seção serão apresentados os mapeamentos dos principais operadores da álgebra ERC+ para SQL.

5 MAPEAMENTO ENTRE ÁLGEBRA ERC+ E SQL

Esta seção apresenta a equivalência dos operadores da álgebra ERC+, mostrados na subseção 2.2.2 para SQL. Através do protótipo VIQUEN, foi gerado um exemplo para o mapeamento de cada operador. Um operador adicional de multiconjunto[63] é considerado como extensão a SQL relacional, o qual existe na linguagem de manipulação do SGBD que dá suporte a este protótipo.

5.1 VISÃO GERAL

Os mapeamentos dos objetos ERC+ são construídos através de consultas e subconsultas em SQL. Entretanto estes objetos, tal como visto anteriormente na descrição formal deste modelo, podem ser compostos de atributos complexos e /ou multivalorados, portanto de relações fora da primeira forma normal. Para que se consiga esta equivalência é necessário permitir a projeção de subconjuntos. Esta projeção é obtida mediante o uso de um operador multiconjunto, aplicado sobre subconsultas, onde o resultado de cada subconsulta eqüivale a um atributo complexo.

Este operador é empregado tanto no mapeamento de objetos ERC+, como no mapeamento de operadores, uma vez que o resultado de qualquer operação ERC+ também é um objeto ERC+.

5.2 OPERADORES

A seguir serão analisados cada um destes operadores e suas respectivas correspondências em SQL.

---

[63] Em nossa implementação este operador é o operador CURSOR, existe na LMD do SGBD escolhido para implementação.

5.2.1 R-join

O operador *R-join* tem como operandos tipos entidades e, como resultado, constrói um novo objeto complexo, formando uma estrutura hierárquica. O operando, escolhido como *operando principal*, deve ser de um tipo entidade, e dará origem a um novo tipo entidade, onde, os demais operandos, serão projetados como seus atributos complexos.

Observa-se que esse operador fornece uma resposta diferente do resultado produzido pela projeção[64] em SQL, uma vez que esta envolve a junção de várias relações, e produz como resultado uma única relação planar.

No mapeamento deste operador para SQL será considerado um operador multiconjunto, a fim de que se obtenha o mesmo resultado da álgebra ERC+. A introdução deste operador será feita através do mapeamento de dois tipos entidade e, posteriormente, será aplicado o operador *r-join* sobre os mesmas.

A seguir, é exibido o mapeamento para SQL da entidade *PROPRIEDADE* e seus atributos complexos *GLEBA* e *LAVOURA*. A representação gráfica deste tipo entidade pode ser vista na Figura 5:

```
1. select   cursor( select NOME, LOCAL ,
2.                  cursor( select NUMERO, AREA ,
3.                          cursor( select ANO, SAFRA, IDCULTURA, QTDEPRODUZIDA
4.                                  from LAVOURA LAVOURA
5.                                  where GLEBA.NRINSCRICAO=LAVOURA.NRINSCRICAO
6.                                        and GLEBA.NUMERO=LAVOURA.NUMERO  ) LAVOURA
7.                          from GLEBA GLEBA
8.                          where PROPRIEDADE.NRINSCRICAO=GLEBA.NRINSCRICAO  ) GLEBA
9.                  from PROPRIEDADE  ) PROPRIEDADE
10.         from dual
```

Sobre o mapeamento anterior se pode observar que:

---

[64] SELECT.

- na linha 1 se tem a projeção dos atributos *NOME* e *LOCAL*, pertencentes à relação *PROPRIEDADE*, sendo que suas ocorrências são projetadas através do operador multiconjunto *cursor* e a referência à relação é feita na cláusula *from* da linha 9, fechando, portanto, o complemento sintático necessário para projeção dos atributos atômicos desta relação;
- visto que a relação *GLEBA* está mapeada como atributo complexo de *PROPRIEDADE*, toda a subconsulta que a mapeia (linhas 2 à 8) é projetada como um multiconjunto através do operador *cursor* da linha 2[65]. A junção da linha 8 é utilizada, a fim de associar a projeção de cada multiconjunto de *GLEBA* a ocorrência corresponde da relação *PROPRIEDADE*;
- de maneira similar, a relação *LAVOURA* é projetada como atributo complexo de *GLEBA* pelo operador cursor da linha 3, aplicado à subconsulta apresentada entre as linhas 3 e 6. Cada multiconjunto da relação *LAVOURA* é associado a *GLEBA* pela junção dessas (subconsulta entre as linhas 5 e 6).

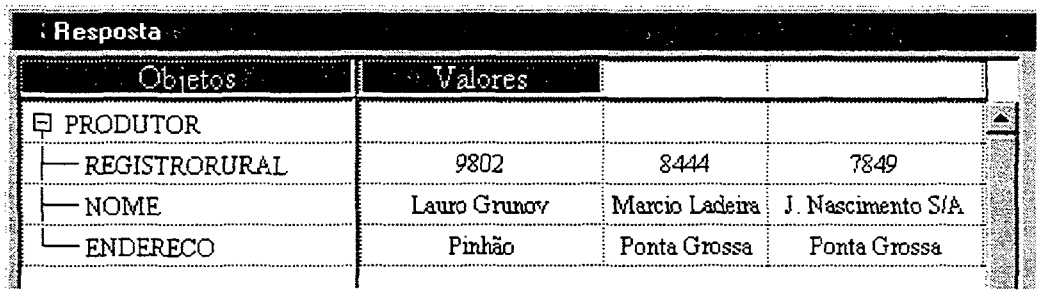
Finalmente, a linha 10 faz o fechamento da consulta que constrói o mapeamento da entidade *PROPRIEDADE*, onde a relação *DUAL* é utilizada apenas para o complemento sintático da expressão SQL. Essa relação contém uma única linha e não afeta o resultado das subconsultas aqui apresentadas. A Figura 6 exibe uma ocorrência deste tipo entidade.

A seguir é mostrado o mapeamento da entidade *PRODUTOR*:

```
select    cursor( select REGISTRORURAL, NOME, ENDERECO
                  from PRODUTOR  ) PRODUTOR
from dual
```

[65] Observa-se que essa subconsulta faz parte da consulta que projeta a relação *PROPRIEDADE*.

Como esta entidade tem apenas atributos atômicos, apenas faz-se necessário projetar o multiconjunto *PRODUTOR*. A Figura 11 exibe algumas ocorrências desta entidade.



Resposta

Objetos	Valores		
PRODUTOR			
REGISTRORURAL	9802	8444	7849
NOME	Lauro Grunov	Marcio Ladeira	J. Nascimento S/A
ENDERECO	Pinhão	Ponta Grossa	Ponta Grossa

FIGURA 11: EXEMPLO DE OCORRÊNCIAS DA ENTIDADE *PRODUTOR*.

Após a exibição destes tipos entidades, e escolhendo como ligação entre essas o tipo relacionamento *CULTIVA*, será demonstrado o operador *r-join*. Assim, para a expressão em álgebra ERC+ *PRODUTOR r-join* (*CULTIVA, PROPRIEDADE*) obtém-se o seguinte mapeamento:

```
1. select    cursor( select  REGISTRORURAL, NOME, ENDERECO,
2.                   cursor( select  ANO,
3.                         cursor( select  NOME, LOCAL,
4.                               cursor( select  NUMERO, AREA,
5.                                     cursor( select  ANO, SAFRA, IDCULTURA, QTDEPRODUZIDA
6.                                           from LAVOURA LAVOURA
7.                                           where GLEBA.NRINSCRICAO=LAVOURA.NRINSCRICAO
8.                                                 and GLEBA.NUMERO=LAVOURA.NUMERO)
9.                                           LAVOURA
10.                                    from GLEBA GLEBA
11.                                    where PROPRIEDADE.NRINSCRICAO=GLEBA.NRINSCRICAO  )
12.                                    GLEBA
13.                              from PROPRIEDADE PROPRIEDADE
14.                              where CULTIVA.NRINSCRICAO=PROPRIEDADE.NRINSCRICAO)
15.                              PROPRIEDADE
16.                  from CULTIVA CULTIVA
```

```
17.             where PRODUTOR.REGISTRORURAL=CULTIVA.REGISTRORURAL  ) CULTIVA
18.     from PRODUTOR )
19.     PRODUTOR
20.from dual
```

No mapeamento acima:

- entre as linhas 3 e 15, está o mesmo mapeamento anteriormente apresentado para a entidade *PROPRIEDADE*, incluindo seus atributos complexos, executando-se a linha 14, que faz a junção desta entidade com o tipo relacionamento *CULTIVA*. Finalmente, a linha 17 faz a junção que liga o tipo relacionamento *CULTIVA* a entidade *PRODUTOR*;
- imposta pelo aninhamento das subconsultas projetadas pelo operador SQL *cursor*, a hierarquia *PRODUTOR*, *CULTIVA*, *PROPRIEDADE*, é resultante da escolha da entidade *PRODUTOR* como o *operando principal*, dentro da sintaxe do operador *r-join*, sendo que os mapeamentos dos demais operandos, *CULTIVA* e *PROPRIEDADE*, serão projetados como atributos complexos da nova entidade;
- o nome da nova entidade gerada por este operador é aquele indicado na linha 19 do mapeamento, sendo que não há dependência deste com o nome da relação no esquema relacional.

Assim, a equivalência do operador *r-join* é obtida pelo acréscimo de ligações de junção entre os mapeamentos dos operandos envolvidos e a projeção de *multiconjuntos* desses, de forma hierarquia. Na Figura 12 pode-se observar, graphicamente, a resposta desta expressão SQL e que a estrutura dos objetos é preservada, tal qual o resultado do operador algébrico ERC+, *r-join.*

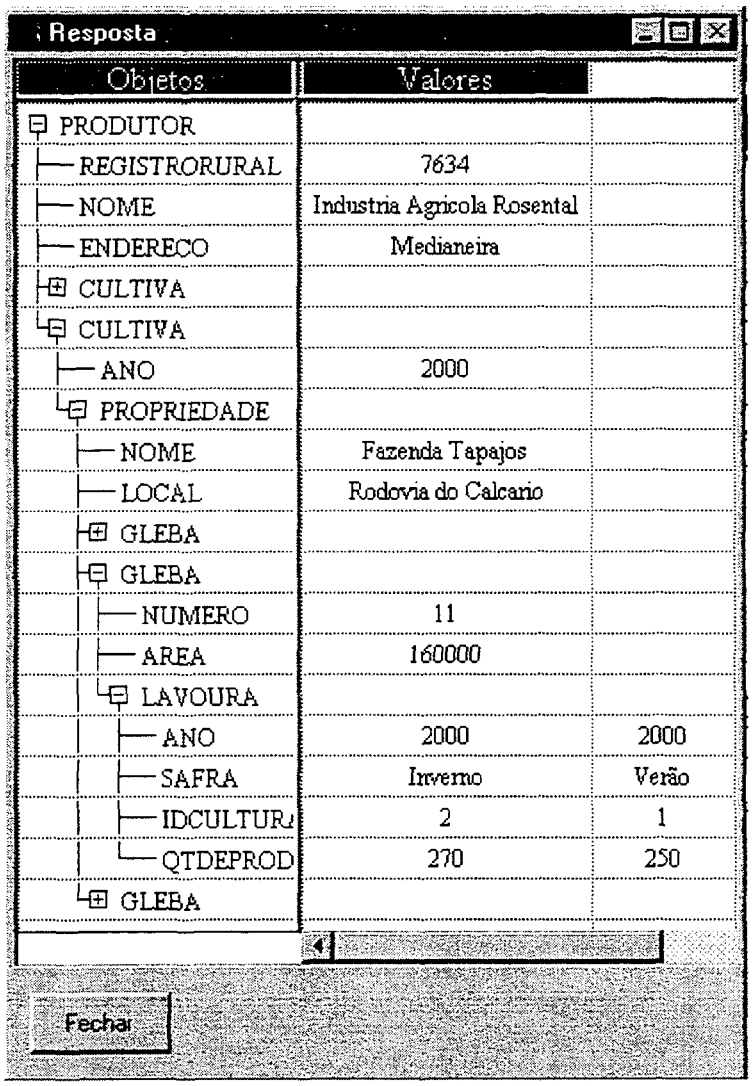


FIGURA 12: RESULTADO DO OPERADOR *R-JOIN*.

A regra geral para este operador é apresentada na próxima expressão SQL. Onde: R_1,..,R_N, são relações; $1 \leq i \leq N$; $1 \leq k \leq N$; R_1 é o operando principal; J é uma operação de junção por igualdade entre duas relações R_1 e R_2, com base em um domínio comum; Z é o nome do novo objeto criado, formado pelos atributos de R_1, acrescido de atributos complexos multivalorados gerados pelo operador multiconjunto *cursor* sobre as relações R_2, ..., R_N, que são os operandos adicionais de *R-join*:

select cursor (select R_1.*,

cursor(select R_2.* , ... , cursor(select R_i.* from R_i where $J(R_{i-1},R_i))R_i$

from R_2 where $J(R_1, R_2)$) R_2
, ... , cursor(select R_k.* from R_k where $J(R_1,R_k)$) R_k
from R_1) Z

Assim, a expressão anterior, define o operador equivalente a *R-join* através do mapeamento de um conjunto de relações R_1,..,R_N para um tipo entidade E.

A mesma definição pode ser obtida sob o ponto de vista de um conjunto de consultas Q_1,..,Q_N, que mapeiam entidades e/ou resultado de outros operadores:

select cursor (Q_1,
cursor(Q_2.* , ... , cursor(Q_i)Q_i
) Q_2
, ... , cursor(Q_k.) Q_k
) E

Onde E é uma nova entidade e Q_i inclui junção para Q_{i-1}, Q_2 para Q_1 e Q_k para Q_1.

5.2.2 Selection

Apesar da similaridade deste operador com o operador correspondente de predicado do modelo relacional e, consequentemente similar à cláusula *where* em SQL, o operador *selection* tem como operando uma entidade que constitui-se de um mapeamento em SQL e, portanto, pode envolver várias subconsultas, sendo assim, o predicado tem um escopo maior dentro deste mapeamento.

Sobre o tipo entidade *PROPRIEDADE*, considere-se o acréscimo de um predicado, o qual restringe as suas ocorrências àquelas que tiveram *LAVOURA* no ano de 1999. A expressão SQL que demonstra este novo mapeamento é dado a seguir:

```
1.select cursor( select NOME, LOCAL ,
2.              cursor( select NUMERO, AREA,
3.                      cursor( select ANO, SAFRA, IDCULTURA, QTDEPRODUZIDA
4.                              from LAVOURA LAVOURA
```

```
5.                          where GLEBA.NRINSCRICAO=LAVOURA.NRINSCRICAO
6.                              and GLEBA.NUMERO=LAVOURA.NUMERO
7.                              and (ANO = 1999) )
8.                          LAVOURA
9.                  from GLEBA GLEBA
10.                 where PROPRIEDADE.NRINSCRICAO=GLEBA.NRINSCRICAO
11.                     and EXISTS(select * from LAVOURA LAVOURA
12.                         where GLEBA.NRINSCRICAO=LAVOURA.NRINSCRICAO
13.                                 and GLEBA.NUMERO=LAVOURA.NUMERO
14.                                 and (ANO = 1999)) )
15.                 GLEBA
16.         from PROPRIEDADE
17.         where EXISTS(select * from GLEBA GLEBA
18.                     where PROPRIEDADE.NRINSCRICAO=GLEBA.NRINSCRICAO
19.                         and EXISTS(select * from LAVOURA LAVOURA
20.                                 where GLEBA.NRINSCRICAO=
21.                                     LAVOURA.NRINSCRICAO
22.                                                                 and
GLEBA.NUMERO=LAVOURA.NUMERO
23.                                             and (ANO = 1999)) )  )
24.         PROPRIEDADE
25.from dual
```

No mapeamento acima, a restrição ao atributo complexo *LAVOURA* aparece pela primeira vez na linha 7, dentro da subconsulta que projeta seus valores (entre as linhas 3 e 8). Entretanto, para que se tenha equivalência ao operador de seleção da álgebra ERC+, o predicado desta subconsulta deve ser aplicado também para as consultas que o antecedem hierarquicamente dentro do mapeamento da entidade, mais precisamente nas subconsultas que efetuam o mapeamento de *GLEBA* e *PROPRIEDADE*.

Para submeter o predicado à subconsulta que mapeia o atributo complexo *GLEBA*, o operador padrão SQL *exists*[66] foi utilizado, dentro da subconsulta do atributo complexo *LAVOURA*. Ainda, esta subconsulta também inclui a junção entre essas mesmas entidades, bem como a restrição imposta ao ano, pelo acréscimo do predicado (linhas 11 a 14).

De forma semelhante, a subconsultas de *GLEBA* e *LAVOURA* devem ser incluídas na condição de existência do predicado da entidade *LAVOURA* (linhas 17 a 23). Assim, mediante o operador SQL *exists*, a seleção imposta ao atributo complexo *LAVOURA* foi aplicada também as ocorrências da entidade *PROPRIEDADE*. A Figura 13 exibe a resposta deste operador.

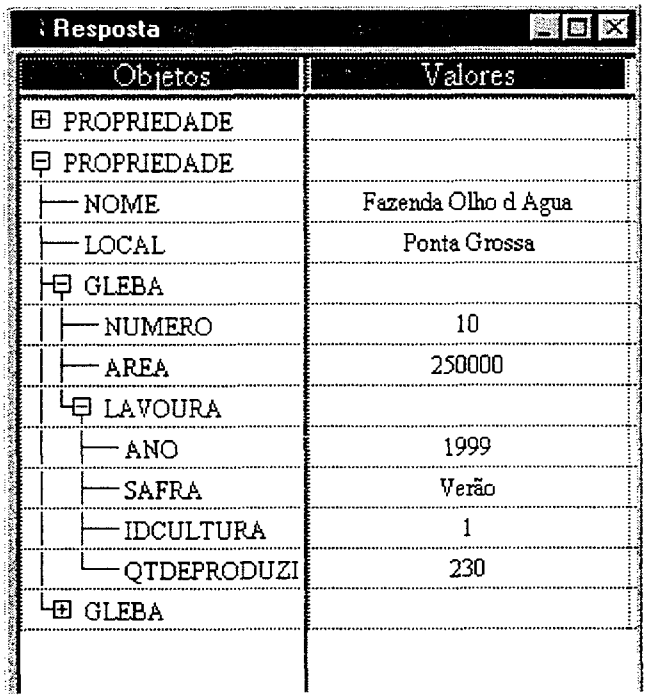


FIGURA 13: RESULTADO DO OPERADOR *SELECTION*.

A regra geral para este operador é dada pela seguinte expressão:

select cursor (select R_1.*,

[66] Expressões tais como NOT NULL LAVOURA ou EXISTS(LAVOURA)não são suportadas pela LMD.

cursor(select R_2.* , ... , cursor(select R_i.* from R_i where $P(R_i)$)R'_i

from R_2 where $P(R_2)$ ... and exists $SQ(R'_i)$)) R'_2

, ... , cursor(select R_k.* from R_k where $P(R_k)$) R'_k

from R_1 where $P(R_1)$ and exists $SQ(R'_2)$ ... and exists $SQ(R'_k)$) R'_1

onde: $1 \leq i \leq N$; $1 \leq k \leq N$; $P(R)$ é um predicado para uma relação R; R' é uma relação derivada de R; $SQ(R')$ a subconsulta que projeta o multiconjunto R'; a população de R' é um subconjunto da formada por R.

O operador de existência *exists* aplica a uma consulta R_1 o predicado de suas subconsultas $SQ(R_2)...SQ(R_k,)$ através da restrição de existência imposta pela população das relações derivadas R'_2... R'_k. O predicado $P(R)$ para cursores aninhados inclui junção com o nível anterior, tal como apresentado para o operador *r-join*.

Sob o ponto de vista de um conjunto de consultas, a mesma definição pode ser obtida por :

select cursor (Q_1,
cursor(Q_2.* , ... , cursor(Q_i)Q_i
) Q_2
, ... , cursor(Q_k. and exists (Q_i.)) Q_k
where exists (Q_2) ... and exists (Q_k.)) E

Onde: Q_1,..,Q_N, são consultas que fazem o mapeamento de entidades e/ou resultados de outros operadores; E é uma nova entidade; os predicados impostos às consultas Q_2, Q_i e Q_k. são propagados, nos níveis de aninhamento que os antecedem, pelo operador de existência dos mesmos, que filtra as ocorrências destas consultas através da dependência de ligação com cada uma de suas subconsultas; tal qual no mapeamento do operador *r-join,* também o predicado de Q_i inclui junção para Q_{i-1}, Q_2 para Q_1 e Q_k para Q_1.

5.2.3 Reduction

O operador *reduction*, assim como ocorre com o operador *selection*, pode ter como operandos entidades cujos mapeamentos para SQL envolvam várias subconsultas e, assim, a sua implementação é similar. Porém, o operador *reduction* da álgebra ERC+ difere do operador *selection* ao permitir que se estabeleça um predicado exclusivamente sobre a população de um atributo sem afetar as ocorrências da entidade e, portanto assemelhando-se mais a junção externa da álgebra relacional, com a exceção de que esta produz um resultado planar, ao passo que o operador *reduction* preserva a estrutura de objetos de seus operandos na resposta.

No mapeamento da subseção 5.2.2, a restrição imposta sobre o atributo complexo *LAVOURA*, da entidade *PROPRIEDADE*, descartou as ocorrências das entidades que não possuíam lavouras no ano de 1999. Se, ao invés do operador *selection*, for aplicado o operador de redução com esta mesma restrição, as propriedades e glebas que não tenham lavouras em 1999 serão exibidas, pois o predicado se aplica exclusivamente ao atributo complexo LAVOURA. O mapeamento para este predicado é exibido a seguir.

```
1. select cursor( select NOME, LOCAL,
2.                 cursor( select NUMERO, AREA,
3.                         cursor( select ANO, SAFRA, IDCULTURA, QTDEPRODUZIDA
4.                                 from LAVOURA LAVOURA
5.                                 where GLEBA.NRINSCRICAO=LAVOURA.NRINSCRICAO
6.                                       and GLEBA.NUMERO=LAVOURA.NUMERO
7.                                       and (ANO = 1999) )
8.                                 LAVOURA
9.                         from GLEBA GLEBA
10.                        where PROPRIEDADE.NRINSCRICAO=GLEBA.NRINSCRICAO )
11.                        GLEBA
12..                from PROPRIEDADE )
13.                PROPRIEDADE
14. from dual
```

No mapeamento acima, observa-se que a restrição às ocorrências de lavoura foi adicionada exclusivamente na subconsulta que a mapeia (linhas 3 a 8). As demais subconsultas que projetam as ocorrências do atributo complexo *GLEBA* e dos atributos atômicos da entidade *PROPRIEDADE* não são afetadas por este predicado. Para enfatizar-se o emprego deste operador, a Figura 14 exibe uma ocorrência de *PROPRIEDADE* e de duas glebas desta que não possuem lavouras no ano de 1999 e mesmo assim são projetadas, devido ao operador *reduction* empregado.

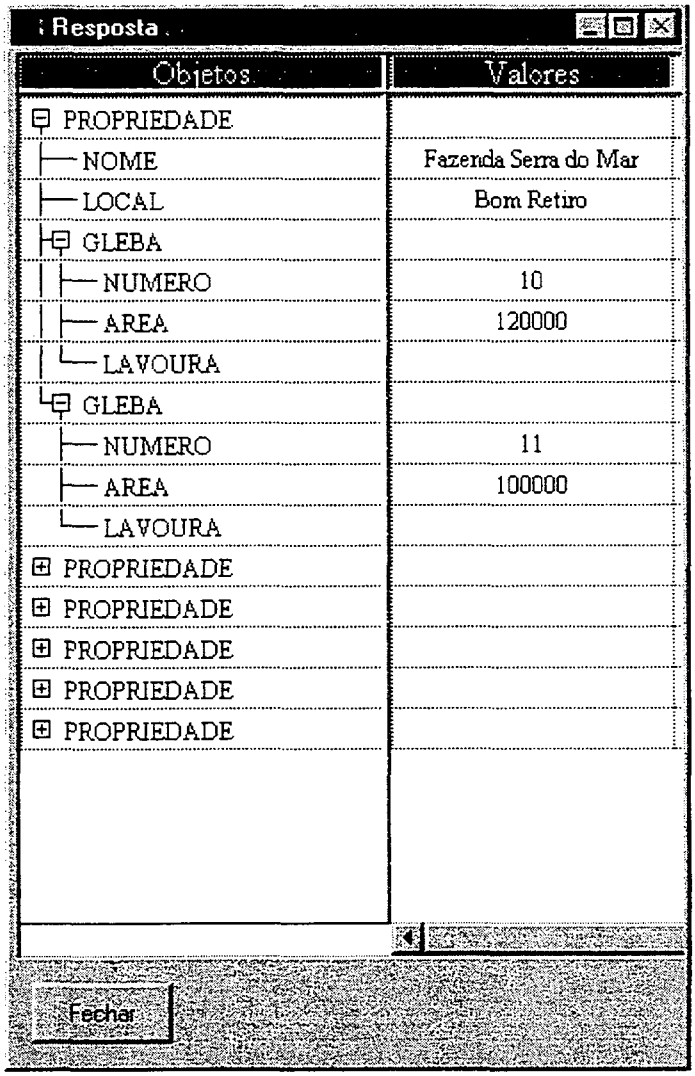


FIGURA 14: RESPOSTA DO OPERADOR *REDUCTION*.

Temos a seguir a regra geral para este operador.

select cursor (select R_1.*,

cursor(select R_2.* , ... , cursor(select R_i.* from R_i where $P(R_i)$)R'_i

from R_2 where $P(R_2)$) R'_2

, ... , cursor(select R_k.* from R_k where $P(R_k)$) R'_k

from R_1) R_1

onde: $1 \leq i \leq N$; $1 \leq k \leq N$; $P(R)$ é um predicado para uma relação R; R' é uma relação derivada de R; a população de R' é um subconjunto de R; p predicado $P(R)$ é aplicado exclusivamente às subconsultas de R_1 e subjacentes[67].

A mesma definição pode ser obtida sob o ponto de vista de um conjunto de consultas Q_1,..,Q_N, que mapeiam entidades e/ou resultado de outros operadores:

select cursor (Q_1,

cursor(Q_2.* , ... , cursor(Q_i where $J(Q_{i-1},Q_i)$)Q_i

where $J(Q_1, Q_2)$) Q_2

, ... , cursor(Q_k. where $J(Q_1,Q_k)$) Q_k

) E

Onde Q_i inclui junção para Q_{i-1}, Q_2 para Q_1 e Q_k para Q_1. Na nova entidade E, diferentemente do operador *selection*, os predicados existentes em Q_2, Q_i e Q_k são desconsiderados em níveis de aninhamento superiores.

[67] Através da junção com as subqueries de R_1.

5.2.4 I-join

Este operador permite agrupar, dentro de uma única entidade, o resultado de duas entidades relacionadas entre si por uma ligação de generalização ou conjunção. Para este operador um operando principal deve ser estabelecido.

A expressão em álgebra ERC+ *FISICA i-join PROPRIETARIO*, construída sob o subesquema de especialização da Figura 7, resulta no seguinte mapeamento:

```
1. select    cursor( select  CPF, DATA_NASCIMENTO, RG  ,
2.                         cursor( select  NOME, ENDERECO, CIDADE, UF
3.                                 from PROPRIETARIO PROPRIETARIO
4.                                 where FISICA.IDENTIFICADOR=PROPRIETARIO.IDENTIFICADOR)
5.                          PROPRIETARIO
6.                 from FISICA
7.                 where  EXISTS(select * from PROPRIETARIO PROPRIETARIO
8.                                  where FISICA.IDENTIFICADOR=PROPRIETARIO.IDENTIFICADOR ) )
9.           FISICA
10.from dual
```

O mapeamento desse operador é tal qual o operador *selection*, exceto que o segundo operando é mapeado para um atributo complexo monovalorado, sendo isto o resultado natural obtido pela cardinalidade do relacionamento entre as entidades envolvidas, ou seja, *um-para-um*.

Nessa operação a ordem dos operandos é importante, visto que a entidade *FISICA* é uma especialização da entidade *PROPRIETARIO*. Na ordem *FISICA/PRORIETARIO,* demonstrada neste mapeamento, há uma ocorrência de *PROPRIETARIO* para toda ocorrência de *FISICA*, entretanto, se trocarmos a ordem dos operandos haveriam somente ocorrências de *PROPRIETARIO* que fossem também pessoas físicas 4.2.2.1. A Figura 15 mostra o resultado deste operador.

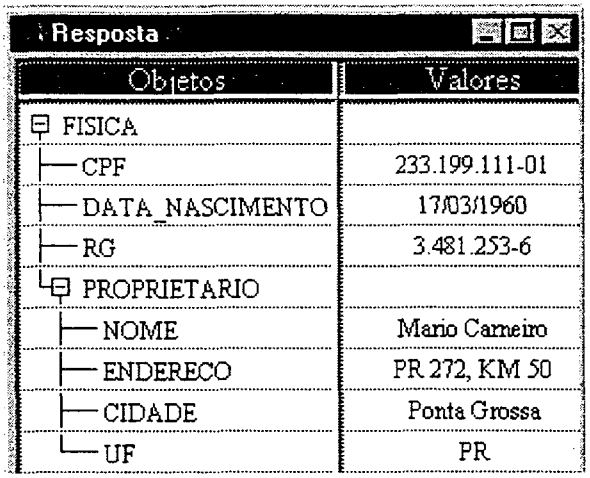


FIGURA 15: RESPOSTA DO OPERADOR *I-JOIN*.

A seguir apresentaremos a regra geral para o mapeamento deste operador. Considerando-se as relações R_1 e R_2 temos:

```
select cursor (select R1.*,
                    cursor(select R2.*  from R2 where J(R1, R2)) Z
                from R1
                where exists SQ(Z)
                ) R'1
from dual
```

Onde: R_1 é o operando principal; J é uma operação de junção por igualdade entre duas relações R_1 e R_2 entre as quais há um ligação por generalização ou conjunção, como definido na subseção 0; Z é o nome do novo atributo complexo criado, com cardinalidade zero ou um; R'_1 é uma relação derivada de R_1.

O operador de existência *exists* aplica à subconsulta de R'_1 o predicado de junção entre R_1 e R_2, expresso pela subconsulta $SQ(Z)$que projeta Z.

O mesmo mapeamento pode ser obtido através de consultas que mapeiam entidades e/ou resultado de outros operadores:

```
select cursor (Q1,
            cursor(Q2) Z
            where exists( Q2)
        ) E
from dual
```

Onde: E é uma nova entidade e Q_2 inclui junção para Q_1.

5.2.5 Projection

Este operador permite projetar atributos de uma entidade. Como as entidades são compostas de mapeamentos, deve-se remover atributos atômicos e/ou mapeamentos de atributos complexos indesejados na resposta.

Considerando a Figura 15 e que somente o nome, RG e data de nascimento sejam escolhidos para compor a resposta, o mapeamento é reduzido a:

```
select    cursor( select  DATA_NASCIMENTO, RG  ,
                        cursor( select  NOME
                                from PROPRIETARIO PROPRIETARIO
                                where FISICA.IDENTIFICADOR=PROPRIETARIO.IDENTIFICADOR)
                         PROPRIETARIO
                  from FISICA
                  where EXISTS(select * from PROPRIETARIO PROPRIETARIO
                                where FISICA.IDENTIFICADOR=PROPRIETARIO.IDENTIFICADOR ))
          FISICA
from dual
```

Resposta

Objetos	Valores
FISICA	
DATA_NASCIMENT	17/03/1960
RG	3.481.253-6
PROPRIETARIO	
NOME	Mario Carneiro

FIGURA 16: RESULTADO DO OPERADOR *PROJECTION* SOBRE *I-JOIN*.

O resultado desta consulta aparece na Figura 16. Em outro exemplo, tome-se novamente a entidade *PROPRIEDADE*, apresentada na Figura 5, e que, além da

informação sobre as propriedades, sejam necessários somente os dados sobre lavouras. Deste modo, nenhuma informação sobre *GLEBA* deve estar contida na resposta. O mapeamento desta projeção é construído da seguinte forma:

```
1. select    cursor( select  NOME, LOCAL,
2.                         cursor(select
3.                                    cursor( select  ANO, SAFRA, IDCULTURA, QTDEPRODUZIDA
4.                                           from LAVOURA LAVOURA
5.                                           where GLEBA.NRINSCRICAO=LAVOURA.NRINSCRICAO
6.                                                 and GLEBA.NUMERO=LAVOURA.NUMERO)
7.                                     LAVOURA
8.                                from GLEBA GLEBA
9.                                where PROPRIEDADE.NRINSCRICAO=GLEBA.NRINSCRICAO  )
10.                        GLEBA  from PROPRIEDADE  )
11.                PROPRIEDADE
12.from dual
```

Na linha 2 não há projeção de nenhum atributo atômico de *GLEBA*, entretanto, sua subconsulta foi projetada para ligar o atributo complexo *LAVOURA* à entidade *PROPRIEDADE*. As linhas 3 a 7 da subconsulta mapeiam o atributo complexo *LAVOURA*. O resultado desta consulta pode ser visto na Figura 17.

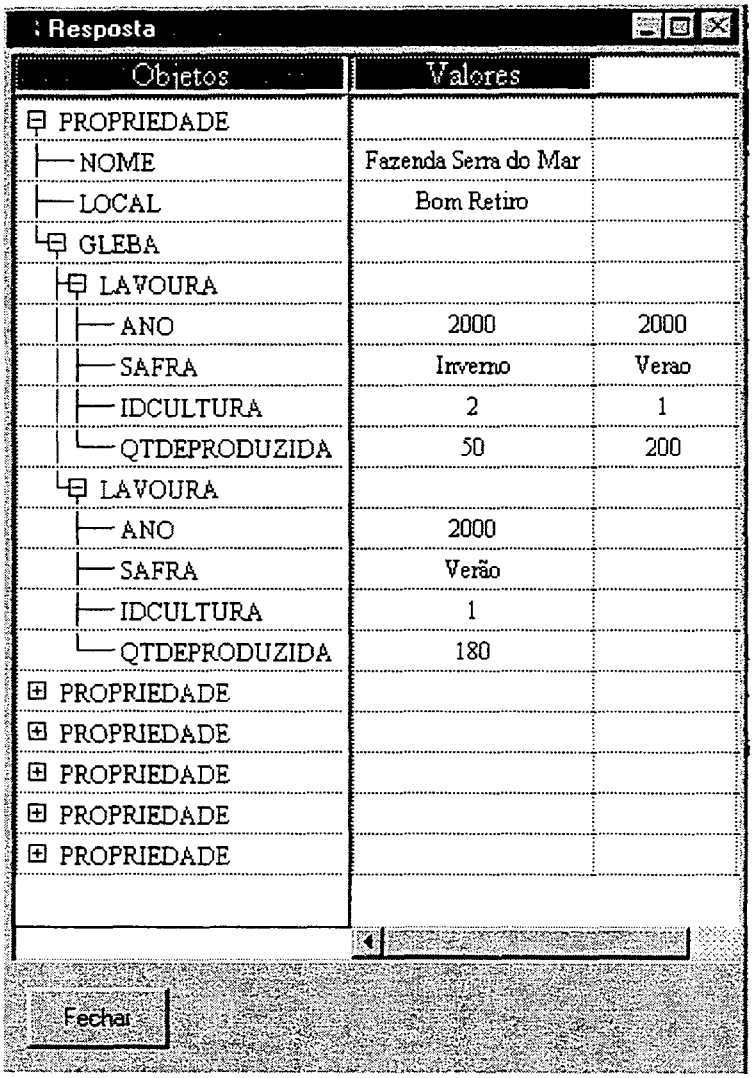


FIGURA 17: RESPOSTA DO OPERADOR *PROJECTION*.

Genericamente este operador é traduzido para um tipo entidade E por:

select cursor(select $x_1,..,x_i$, cursor(SQ_1), ..., cursor(SQ_j) from R) E

Onde: R é uma relação; X o conjunto de atributos atômicas da relação R; M o conjunto de subconsultas que mapeiam atributos complexos para E, tal que $x_i \in X$ e $SQ_i \in M$;

5.2.6 Simplification

Este operador é um operador sintático da álgebra ERC+, que elimina complexidades desnecessárias, as quais podem ser ocasionadas por outros operadores, excluindo um nível em uma estrutura complexa, desde que o atributo complexo removido contenha somente um atributo, e este seja monovalorado.

Na subseção 5.2.5, o exemplo apresentado na Figura 16 tem, como resultado da projeção de *PROPRIETARIO*, um atributo complexo com um único atributo, o atributo *nome*. Tendo em vista que há um relacionamento *um-para-um* entre este atributo e cada ocorrência do tipo entidade *FISICA*, esse atributo também é monovalorado. Assim esta projeção pode ser simplificada, tal como dado pelo mapeamento seguinte:

```
1. select   cursor( select DATA_NASCIMENTO, RG ,
2.                      ( select NOME
3.                              from PROPRIETARIO PROPRIETARIO
4.                              where FISICA.IDENTIFICADOR=PROPRIETARIO.IDENTIFICADOR )
5.                      PROPRIETARIO
6.                  from FISICA)
7.          FISICA
8. from dual
```

Na linha 2, o operador multiconjunto *cursor* foi removido e o atributo atômico *nome* será projetado no mesmo nível dos demais atributos atômicos do tipo entidade *FISICA*, como pode ser visto na Figura 18.

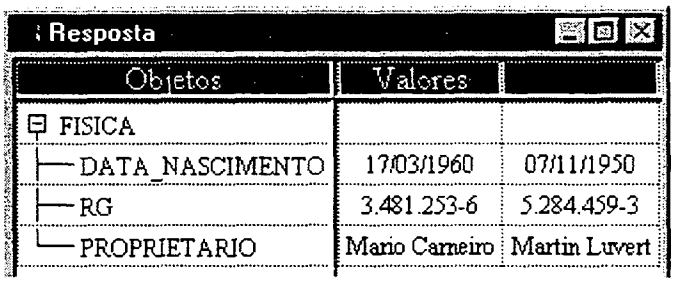


FIGURA 18: RESPOSTA DO OPERADOR *SIMPLIFICATION.*

A simplificação em nosso mapeamento consiste em elilminar o aninhamento de uma subconsulta, deste que esta contenha um único atributo complexo ou atômico monovalorado. Como regra geral, dada uma consulta Q:

select cursor (select $R_1.a_{11}$,.., $R_1.a_{1k}$,
cursor(SQ_1) Z_1
from R_1
) O_1
from dual

onde: a_{11},..,a_{1k} são atributos atômicos e monovalorados de R_1; SQ_1 é uma subconsulta contendo um único atributo monovalorado; Z_1 é o atributo complexo mapeado pela subconsulta SQ_1; O_1 é o objeto resultante da consulta Q.

Como simplificação de Z_1 temos:

select cursor (select $R_1.a_{11}$,.., $R_1.a_{1k}$,
(SQ_1) Z_1
from R_1
) O'_1

Onde: O'_1 é uma relação resultante da projeção dos atributos atômicos e monovalorados $a_{11},..,a_{1k}$; Z_1 é um atributo monovalorado mapeado pela subconsulta SQ_1.

5.2.7 Product

É um operador binário da álgebra ERC+ em que, para cada ocorrência de um operando principal, há um multiconjunto formado por todas as ocorrências do segundo operando.

A seguir, é exibido o mapeamento para o operador *product* entre *PROPRIEDADE* e *PRODUTOR*.

```
1.select    cursor( select  NOME, LOCAL  ,
2.                      cursor( select  NUMERO, AREA  ,
3.                          cursor( select  ANO, SAFRA, IDCULTURA, QTDEPRODUZIDA
4.                                  from LAVOURA LAVOURA
5.                                   where GLEBA.NRINSCRICAO=LAVOURA.NRINSCRICAO
6.                                       and GLEBA.NUMERO=LAVOURA.NUMERO  ) LAVOURA
7.                           from GLEBA GLEBA
8.                           where PROPRIEDADE.NRINSCRICAO=GLEBA.NRINSCRICAO  ) GLEBA,
9.                      cursor( select  REGISTRORURAL, NOME, ENDERECO
10.                          from PRODUTOR  ) PRODUTOR
11.                from PROPRIEDADE  )
12.          PROPRIEDADE_X_PRODUTOR
13.from dual
```

As linhas 9 e 10 descrevem o mapeamento do tipo entidade *PRODUTOR*, sendo que não há nenhuma junção com a subconsulta que mapeia o tipo entidade *PROPRIEDADE*. Assim, para cada ocorrência de *PROPRIEDADE*, temos um multiconjunto com todas as ocorrências do tipo entidade *PRODUTOR*, conforme o resultado exibido na Figura 19.

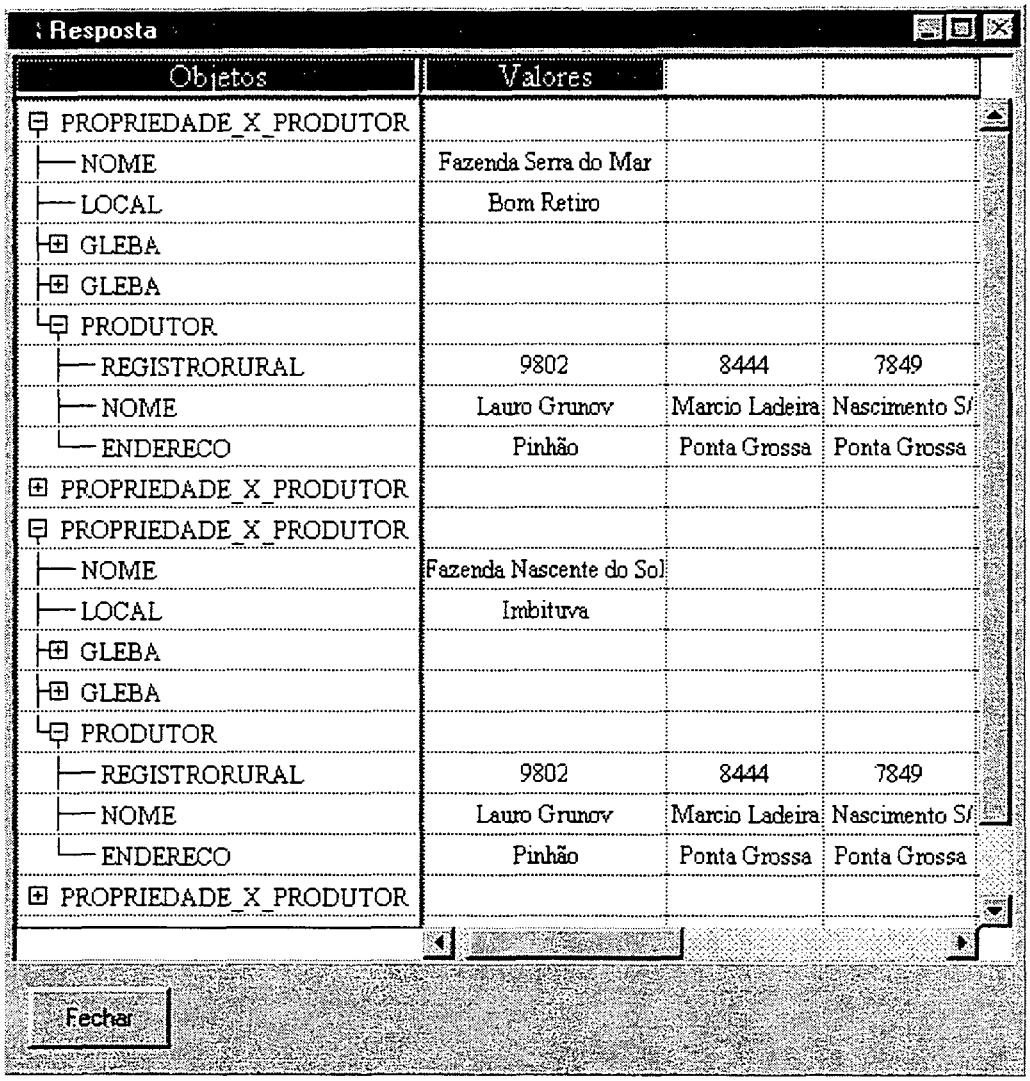


FIGURA 19: RESPOSTA DO OPERADOR *PRODUCT*.

Segue a regra geral para este operador:

- Q_1=select cursor(cursor(SQ_1), ..., cursor(SQ_i) from R) E_1 from dual ;
- Q_3=select cursor(select cursor(SQ_1), ...,cursor(SQ_i), cursor(Q_2) A from R) E from dual;

Onde: R é uma relação; Q_1 é uma consulta que mapeia um tipo entidade E_1; SQ_1,..,SQ_i são subconsultas de Q_1; Q_3 é um consulta que mapeia o operador *product* sobre os operandos Q_1 e Q_2, em que Q_1 é operando principal; Q_2 é um consulta que mapeia um tipo entidade E_2, e não há junção entre Q_2 e Q_1 em Q_3; A é o nome de um

novo atributo complexo de E, o qual é o novo tipo entidade criado pelo produto de E_1 e E_2. O valor de uma ocorrência E é formada por uma ocorrência de E_1 mais um multiconjunto de todas as ocorrências geradas pela consulta Q_2.

5.2.8 Renaming

É um operador sintático que troca o nome de um atributo. Para obtermos a operação de *renaming* dentro do mapeamento temos que considerar, distintamente, atributo complexo e/ou multivalorado de atributo atômico e monovalorado, pois as duas primeiras categorias constituem-se em relações projetadas por um operador multiconjunto, e um atributo atômico monovalorado não é projetado por um operador *cursor*. Por exemplo, para trocarmos o nome do atributo complexo *GLEBA* para *AREA* temos o seguinte mapeamento:

```
1. select    cursor( select NOME, LOCAL ,
2.                   cursor( select NUMERO, AREA ,
3.                           cursor( select ANO, SAFRA, IDCULTURA, QTDEPRODUZIDA
4.                                   from LAVOURA LAVOURA
5.                                   where GLEBA.NRINSCRICAO=LAVOURA.NRINSCRICAO
6.                                         and GLEBA.NUMERO=LAVOURA.NUMERO  ) LAVOURA
7.                           from GLEBA GLEBA
8.                           where PROPRIEDADE.NRINSCRICAO=GLEBA.NRINSCRICAO  )
9.                   AREA
10.                  from PROPRIEDADE  ) PROPRIEDADE
11.          from dual
```

Na linha 9, o novo nome é indicado, apenas trocando o nome do multiconjunto gerado. Por outro lado a troca de nome de um atributo atômico monovalorado é disponibilizada diretamente em SQL, tal como segue:

```
select NOME NOMECOMPLETO, ENDERECO from PRODUTOR
```

Nesta expressão SQL, o atributo *nome* foi renomeado para *nomecompleto*.

Genericamente: Seja $Q(Z)$ uma consulta que mapeia um atributo complexo e/ou multivalorado Z, temos o operador equivalente a *renaming* por:

select cursor(Q) A from dual

Onde A é o novo nome do atributo complexo e/ou multivalorado e Q uma subconsulta que mapeia um atributo complexo.

Para atributos atômicos e monovalorados o mapeamento é dado por:

select z a from R,

Onde z é o nome anterior do atributo; a é o novo nome para o qual z foi renomeado; e R uma relação.

5.2.9 Union

A álgebra ERC+ permite a união entre duas entidades de duas formas:

- baseada por igualdade de identificadores de objeto, isto é, dois objetos podem ser unidos se os mesmos possuem o mesmo identificador[68], independentemente de seus valores;
- pela igualdade de valores dos objetos envolvidos.

Diferentemente do operador relacional *union*, o primeiro tipo de união mescla as populações de dois tipos entidades criando um tipo entidade derivada de ambos. Na operação $E=E_1$ *union* E_2 todos os atributos de E_1 e E_2 constam em E, e cada atributo existente em E_1 e E_2 é unido de modo que o novo domínio é a união de ambos os domínios e seu valor é um multiconjunto dos dois valores.

A abordagem deste trabalho não gera identificadores de objetos para as tuplas recuperadas em consulta. São utilizados somente identificadores de tuplas, que

[68] OID.

são únicos apenas dentro de um tipo entidade ou tipo relacionamento, obtidos através da chave primária. Entretanto, a fim de mapear esse operador, é disponibilizada a união entre duas entidades através da junção sob um domínio comum, sendo esse domínio equivalente ao identificador de objeto, limitado, entretanto, ao escopo das duas entidades envolvidas nesta operação. Ficando a critério do usuário a escolha das entidades sob as quais haja validade semântica[69] em uni-las.

A seguir será apresentado o mapeamento do operador ERC+ *union,* aplicado sobre as entidades apresentadas na Figura 8, onde ocorre uma ligação por conjunção entre *PROPRIETARIO* e *PRODUTOR* e cuja implementação relacional, conforme apresentado na subseção 4.2.2.2, é feita através da chave estrangeira de *PRODUTOR*:

```
1.select cursor( select  p.identificador,p.registrorural,
2.                       cursor(select nome from proprietario where proprietario.identificador=p.identificador
3.                              union all
4.                              select nome from produtor where produtor.registrorural=p.registrorural
5.                             )
6.                       nome,
7.                       cursor(select endereco
8.                              from proprietario where proprietario.identificador=p.identificador
9.                              union all
10.                             select endereco from produtor where  produtor.registrorural=p.registrorural
11.                              )
12.                      endereco,
13.                      cursor(select registrorural
14.                             from produtor where produtor.registrorural=p.registrorural)
15.                      produtor ,
16.                      cursor(select cidade,uf from
17.                             proprietario where proprietario.identificador=p.identificador)
18.                      proprietario
19.              from
20.                    ( select pr.identificador, pt.registrorural from proprietario pr, produtor pt
21.                      where pr.identificador=pt.identificador (+)
```

[69] Como a união utiliza junção, é possível unir quaisquer relações com algum domínio em comum, entretanto este domínio pode não ser válido semanticamente, tal como a união de proprietário e propriedade pela chave primária de

```
22.                union
23.                select pr.identificador, pt.registrorural from proprietario pr, produtor pt
24.                where pt.identificador=pr.identificador (+) ) p )
25.      E
26.from dual
```

Neste mapeamento:

- os identificadores das entidades são projetados na linha 1, sendo que pode existir ocorrência de ambos ou de apenas um destes, visto que há uma junção externa total[70] entre as entidades envolvidas na união, sob o domínio comum do atributo *identificador*(linhas 20 e 24);
- os atributos comuns *nome* e *endereço* são projetados pelo operador multiconjunto *cursor* , sendo que o operador *union all* é empregado para projetar eventuais valores repetidos existentes nos domínios destes;
- atributos distintos entre as entidades são projetados através de ligações de conjunção com a nova entidade gerada *E*, tal como *cidade* e *uf* pertencentes, exclusivamente a entidade *PROPRIETARIO* (linhas 16 a 18).

Uma ocorrência desta união é exibida na Figura 20.

Para formulação da regra geral deste operador considere a união ERC+ E_1 *union* E_2 e duas consultas Q_1 e Q_2 que as mapeiam; também PK_1 e PK_2 como identificadores respectivamente de Q_1 e Q_2, onde PK_1 e PK_2 são conjuntos[71] de atributos atômicos. A próximo expressão SQL constitui-se a regra geral deste mapeamento:

select cursor(select $P.\mathrm{PK}_1$, $P.\mathrm{PK}_2$,
cursor(select b_{11} from Q_1 where $J(Q_1, P.\ \mathrm{PK}_1)$

---

ambos, o domínio da chave de ambos é numérico e inteiro, entretanto semanticamente esta união é inválida.

[70] Junção externa à esquerda e à direita.

```
                union all
                select b2l from Q2  where J(Q2, P.PK2)) zl
        ,...,
        cursor(select b1s from Q1  where J(Q1, P. PK1)
                union all
                select b2s from Q2  where J(Q2, P.PK2)) zs,
         cursor(select cursor(d1l) from Q1  where J(Q1, P. PK1),
                        cursor(d2l) from Q2  where J(Q2, P.PK2)
                 )dl
         ,... ,
         cursor( select cursor(d1t) from Q1  where J(Q1, P. PK1),
                        cursor(d2t) from Q2  where J(Q2, P.PK2)
                 ) dt,
         cursor(select Q1.a11,.., Q1.a1k ,
                        cursor(c11),...,cursor(c1w)
                 from Q1 where J(Q1, P.PK1))
         E1,
         cursor(select Q2.a21,.., Q2.a2k ,
                        cursor(c21),...,cursor(c2u)
                 from Q2 where J(Q2, P.PK2))
         E2
    from (select Q1.PK1, Q2.PK2 from Q1, Q2.
          where JE(Q1, Q2) ) P
) E from dual
```

Onde: sejam S_1 e S_2 conjuntos de atributos atômicos de Q_1 e Q_2; $B = S_1 \cap S_2$ o conjunto de atributos comuns a Q_1 e Q_2, em que $b_{1m} \in B$, $b_{2m} \in B$ e $1 \leq m \leq s$; $A = (S_1 - S_2 \cup S_2 - S_1)$ o conjunto de atributos distintos entre Q_1 e Q_2 e $a_{1i} \in A$, $a_{2n} \in A$ e $1 \leq i \leq k$, $1 \leq n \leq v$ e $i \neq n$; P é a relação resultante da união dos identificados das entidades Q_1 e Q_2 que formaram os identificadores da nova entidade E; $z_1,.., z_s$ são os atributos resultantes da união multiconjunto de cada atributo comum entre Q_1 e Q_2; sejam S_3 e S_4 os conjuntos de todos os atributos complexos e/ou multivalorados de E_1 e

[71] Genericamente o identificador pode também ser composto e assim todos os identificadores das entidades envolvidas no mapeamento de união devem ser projetados para permitir que o resultado deste operador seja entrada de outro operador, atingindo o fechamento, tal qual o da álgebra ERC+.

E_2; C= (S_3 - S_4 $\cup$ S_4 - S_3) o conjunto de subconsultas que mapeiam atributos complexos e/ou multivalorados distintos entre E_1 e E_2, tal que $c_{1o} \in C$, $1 \leq o \leq w$, e $c_{2h} \in C$, $1 \leq h \leq u$ e $o \neq h$; $D = S_3 \cap S_4$ o conjunto de subconsultas que mapeia atributos complexos e/ou multivalorados comuns entre E_1 e E_2, tal que $d_{1x} \in D$, $d_{2x} \in D$ e $1 \leq x \leq t$; *JE* é um operador de junção externa [72], utilizado para produzir P, sob Q_1 e Q_2; J é um operador de junção.

Por esta regra, cada atributo comum é mapeado como um operador de E, e os atributos distintos são mapeados para atributos complexos, através de uma ligação de conjunção com E, que recebem o nome de suas respectivas entidades.

A união na álgebra ERC+ pode também ser feita *por valor*. Considerando-se o mesmo exemplo de união por identificador, observa-se que os valores são distintos entre as instâncias destes tipos entidade. Desta forma, a união dos mesmos gera uma nova entidade com dois atributos complexos, com os respectivos nomes de seus tipos entidade. O mapeamento deste operador é mostrado a seguir:

```
select cursor(select
                    cursor( select IDENTIFICADOR, NOME, ENDERECO, CIDADE, UF
                            from PROPRIETARIO)
                    PROPRIETARIO,
                    cursor( select REGISTRORURAL, NOME, ENDERECO from PRODUTOR )
                     PRODUTOR
               from dual )
      E
from dual
```

A união destes tipos entidade produz um novo tipo entidade contendo dois novos atributos complexos com todas as entidades envolvidas. Este resultado é exibido na Figura 21.

[72] Junção externa total.

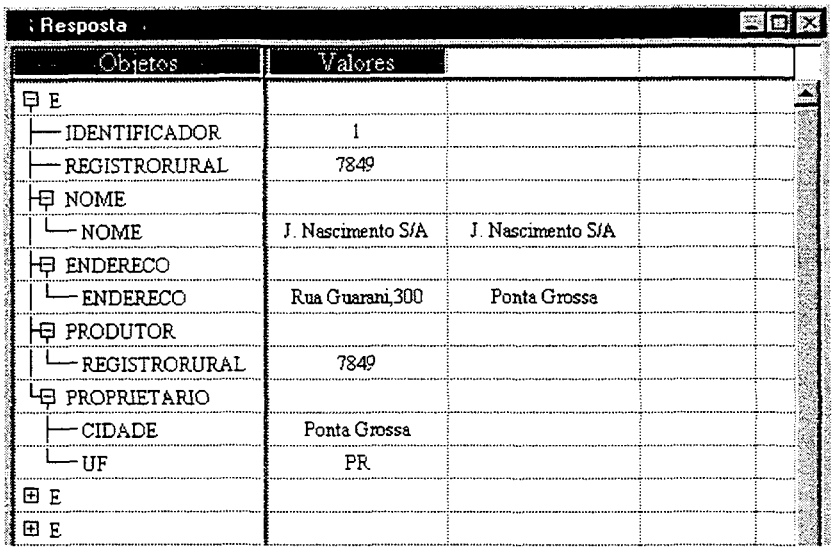


FIGURA 20: OPERADOR *UNION*, SOB A ABORDAGEM DE IDENTIFICADOR

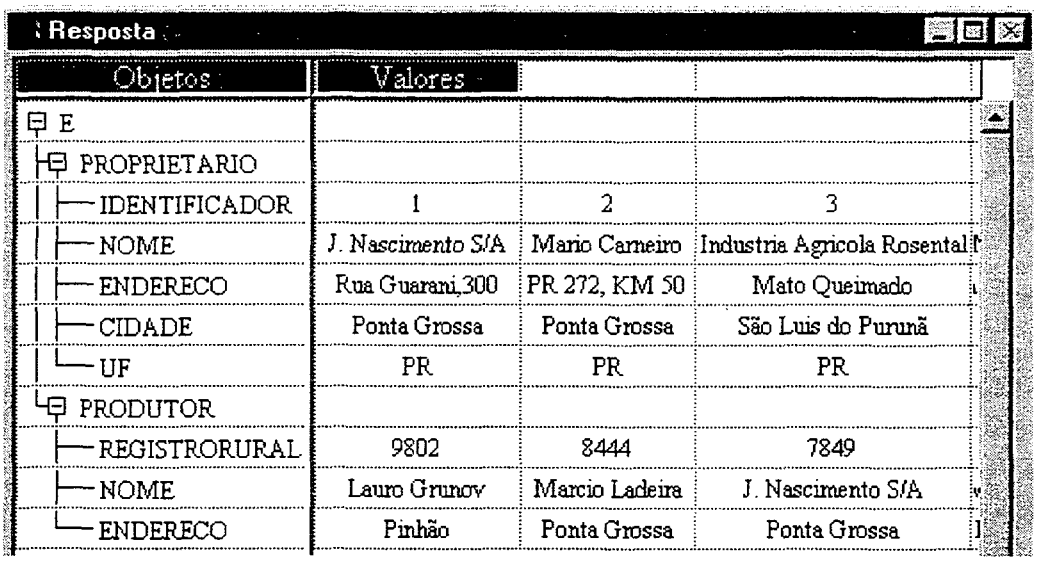


FIGURA 21: RESPOSTA DO OPERADOR *UNION*, SOB A ABORDAGEM POR VALOR

Aqui foram descritos os mapeamentos em SQL, para cada operador da álgebra ERC+. Na próxima seção apresentaremos o protótipo desenvolvido, o qual implementa os principais mapeamentos e o modo como os mesmos são apresentados ao usuário.

6 VIQUEN - UM AMBIENTE INTERATIVO PARA CONSULTA VISUAL E EXTRAÇÃO DE ESQUEMAS

VIQUEN (*Visual Query Environment*), é um protótipo que foi construído para atingir a categoria de usuários comuns, os quais buscam freqüentemente informações armazenadas em sistemas de banco de dados, mas que desconhecem as sintaxes de linguagens de consulta, construídas tipicamente para uso por especialistas. Esta aplicação pertence à categoria de editores gráficos tal qual QBD*(ANGELACCIO; CATARCI; SANTUCCI, 1990) e gql/ER (ZHANG; MENDELZON, 1983) e SUPER (DENNEBOUY et al., 1995), que fazem uso da linguagem visual com a qual o usuário interage para formular suas consultas, sem ser necessário que ele conheça a linguagem textual de consulta do modelo ERC+ ou a linguagem de manipulação de dados utilizada pelo SGBD. As operações estão implícitas nas ações tomadas pelo usuário sobre o próprio esquema apresentado durante a formulação das consultas, ou seja, a consulta é construída através manipulação direta de objetos do modelo e mediante a linguagem visual aplicada sobre ele.

Este capítulo descreve o funcionamento do protótipo VIQUEN, a partir das etapas apresentadas na interface com o usuário, que são: a engenharia reversa, que traduz uma base relacional para o correspondente modelo ERC+; a seleção do subesquema desejado na consulta; a especificação do predicado; a visualização da resposta. Também, no final deste capítulo, são abordados detalhes de implementação.

6.1 RECUPERAÇÃO DO ESQUEMA

Se, por um lado, há um número elevado de usuários não especialistas que necessitam de acesso a dados, por outro, temos um grande domínio de SGBDs relacionais que, por conseqüência, detêm um vasto legado de informações. Este contexto torna importante que as aplicações de consulta disponibilizem acesso a estas

bases de dados. Sob este prisma, a primeira característica desta aplicação é recuperar um esquema construído sobre o modelo relacional, de pouco poder de expressão[73], e traduzi-lo para o modelo ERC+.

A seguir apresentaremos um modelo relacional e a tradução corresponde para o modelo ERC+. A Tabela 6 exibe a implementação relacional.

TABELA 6: IMPLEMENTAÇÃO RELACIONAL DE UMA PROPRIEDADE AGRÍCOLA.

Nome Relação	Lista de Atributos	Chave Primária	Chave(s) Estrangeira(s)
Proprietario	Identificador, Nome, Endereco, Cidade, UF	Identificador	
Fisica	Identificador, CPF, Data_Nascimento, RG	Identificador	Proprietario →Identificador
Juridica	Identificador, CGC	Identificador	Proprietario →Identificador
Possui	Identificador, NrInscricao, DataAquisicao	Identificador, NrInscricao	Proprietario → Identificador Propriedade → NrInscricao
Propriedade	NrInscricao, Nome, Local	NrInscricao	
Gleba	NrInscricao, Numero, Area	NrInscricao, Numero	Propriedade→ NrInscricao
Lavoura	NrInscricao, Numero, IdCultura, Ano, Safra, QtdeProduzida	NrInscricao, Numero, IdCultura	Gleba→ NrInscricao, Numero
Cultiva	IdCultiva, RegistroRural, NrInscricao, Ano	IdCultiva, RegistroRural, NrInscricao	Propriedade→ NrInscricao Produtor→ RegistroRural
Arrenda	RegistroRural, NrInscricao, DataInicio, Duracao	RegistroRural, NrInscricao	Propriedade→ NrInscricao Produtor→ RegistroRural
Produtor	RegistroRural, Nome, Endereco, Identificador (restrição de unicidade[74])	RegistroRural	Proprietario → Identificador
Registro_Imovel	IdContrato, Identificador, NrInscricao	IdContrato, Identificador, NrInscricao	Possui → Identificador, NrInscricao

[73] Quando comparado a modelo de objetos e objeto-relacionais, conforme descrito na seção 2.

[74] Há restrição no dicionário de dados sobre este atributo que não permite repetição de valor para o mesmo.

Para traduzir um esquema relacional, o usuário deve informar o nome do usuário/esquema[75] e senha de acesso, conforme as janelas de diálogo da Figura 22.

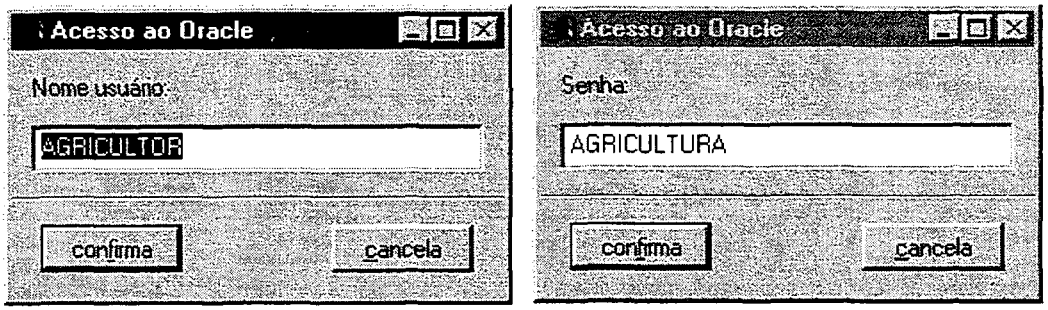


FIGURA 22: CONEXÃO COM SGBD

O esquema contendo todas as tabelas e restrições pertencentes ao usuário/esquema informado é traduzido automaticamente pela aplicação para um esquema sob o modelo ERC+. A Figura 23 exibe este esquema.

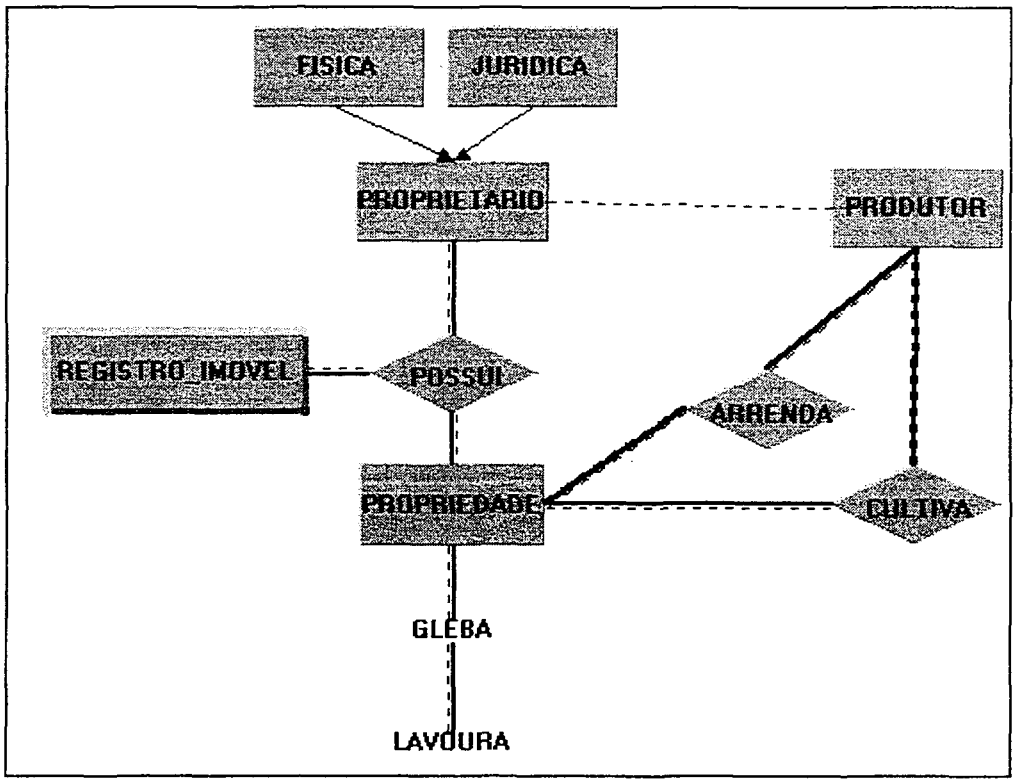


FIGURA 23: ESQUEMA DE UMA PROPRIEDADE. AGRÍCOLA SOB O MODELO ERC+

[75] No SGBD Oracle toda tabela tem um proprietário, denominado *owner*.

Tipos entidade, relacionamento e atributos complexos são mostrados graficamente e atributos atômicos são visualizados em uma janela distinta sob uma caixa de diálogo ativada pelo usuário através de um *duplo click* sobre um objeto. Essa opção de não exibir atributos atômicos no grafo foi adotada tendo em vista a possibilidade de um número elevado de atributos deste tipo dentro de esquemas, o que pode saturar a visualização e dificultar a manipulação do modelo exibido.

O esquema apresentado na Figura 23 foi obtido através das regras de tradução e refinamento descritas na seção 4. A associação entre os elementos relacionais e as respectivas regras adotadas para traduzi-los são descritas na Tabela 7.

TABELA 7:TRADUÇÃO DO ESQUEMA RELACIONAL PARA O ESQUEMA ERC+

Elemento Relacional	**Regra de Tradução/ Refinamento**	**Elemento ERC+**	**Observação**
Tabela Proprietario	Regra de tradução direta	Tipo Entidade Proprietario	Entidade com independência de identificação
Tabela Propriedade	Regra de tradução direta	Tipo Entidade Propriedade	Entidade com independência de identificação
Tabela Gleba	Regra de refinamento 1	Atributo Complexo Gleba	Atributo complexo de Propriedade
Tabela Lavoura	Regra de refinamento 1	Atributo Complexo Lavoura	Atributo complexo de Gleba
Tabela Produtor	Regra de tradução direta	Tipo Entidade Produtor	Entidade com independência de identificação
Tabela Fisica	Regra de Refinamento 2	Tipo Entidade Fisica com ligação Especialização/ Generalização	Entidade especialização de Proprietário
Tabela Juridica	Regra de Refinamento 2	Tipo Entidade Jurifica com ligação Especialização/ Generalização	Entidade especialização de Proprietário

TABELA 7:TRADUÇÃO DO ESQUEMA RELACIONAL PARA UM ESQUEMA ERC+

Conclusão

Elemento Relacional	Regra de Tradução/ Refinamento	Elemento ERC+	Observação
Possui	Regra de Refinamento 4	Tipo Relacionamento Possui	A cardinalidade das ligações com os tipos entidades Proprietario e Propriedade são mínimo um e máximo *n*[76].
Arrenda	Regra de Refinamento 4	Tipo Relacionamento Arrenda	A cardinalidade das ligações com os tipos entidades Produtor e Propriedade são mínimo um e máximo *n*
Cultiva	Regra de Refinamento 4	Tipo Relacionamento	A cardinalidade das ligações com os tipos entidades Produtor e Propriedade são mínimo um e máximo *n*
Registro_Imovel	Regra de refinamento 5	Tipo Entidade Fraca Registro_Imovel	
Tabela Produtor: Chave Estrangeira Identificador em Produtor	Regra de refinamento 3	Ligação de conjunção ou ligação *um-para-um*[77]	Há restrição de unicidade e permissão de valores nulos sobre o atributo *Identificador*.

Sobre o esquema resultante apresentado na Figura 23, cabe ao usuário definir a ligação entre *PROPRIETARIO* e *PRODUTOR* como *um-para-um* ou por conjunção, a qual, a princípio, é apresentada como ligação por conjunção. As demais traduções são determinadas exclusivamente pela aplicação.

O esquema recuperado pode ser salvo para uso posterior sem a necessidade de nova tradução, exceto se houver alterações do esquema relacional.

[76] A cardinalidade mínima um é determinada pelas chaves estrangeiras comporem também a chave primária o que as torna obrigatórias, a cardinalidade máxima *n* é assumida pois não há indicação de restrição de unicidade sobre nenhuma das chaves estrangeira.

[77] Fica a critério do usuário determinar se a ligação é de conjunção ou *um-paraum*.

Como próximo passo, o usuário deve selecionar dentre o esquema quais objetos deseja envolver na consulta e também a visão desejada. Esta etapa é descrita na próxima subseção.

6.2 SELEÇÃO DOS OBJETOS DE CONSULTA

A partir do esquema recuperado, o usuário pode selecionar[78] os objetos desejados na formação da consulta. Esta seleção deve incluir tanto os objetos a serem desejados na resposta como aqueles necessários na formação do predicado. Também é possível selecionar apenas tipos relacionamentos, pois a aplicação, automaticamente, selecionará os objetos e papéis envolvidos. Os atributos atômicos e monovalorados devem ser selecionados de uma caixa de diálogo, que é ativada por um *duplo click* sobre o objeto[79] que os contém. Os atributos atômicos selecionados podem ser salvos junto com o esquema completo, apresentado na fase anterior, facilitando a reutilização em novas consultas, porém, a qualquer tempo os mesmos podem ser desmarcados e novos atributos selecionados. A Figura 24 mostra um exemplo da seleção de objetos para consulta e a Figura 25 o subesquema selecionado pelo usuário, o qual é exibido em uma nova janela do protótipo.

Tal qual o protótipo SUPER apresentado na subseção 3.4, uma das principais características desta aplicação é exibir o resultado de qualquer consulta preservando a estrutura de objetos do modelo ERC+.

Para submeter uma consulta o usuário deve determinar qual visão deseja, escolhendo uma raiz para a consulta e também eliminar eventuais ciclos que hajam entre os objetos da consulta.

---

[78] Para selecionar mais de um objeto o usuário deve manter a tecla *shift* pressionada e selecionar os objetos com o mouse.

[79] Tipo entidade ou relacionamento.

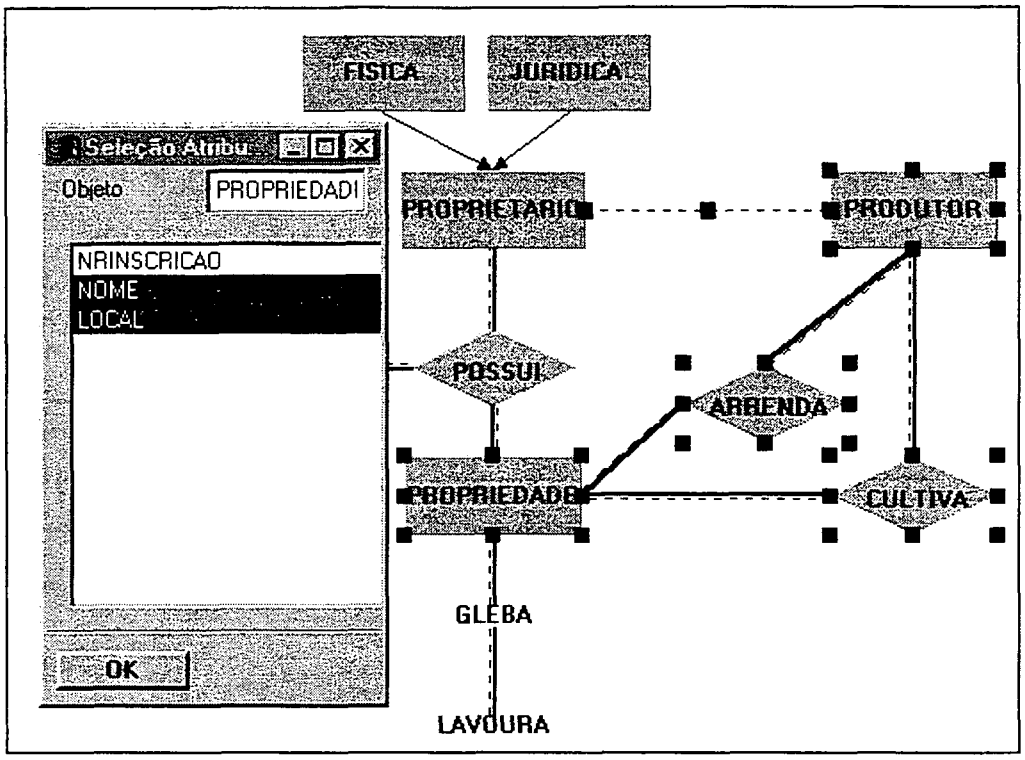


FIGURA 24: SELEÇÃO DE OBJETOS

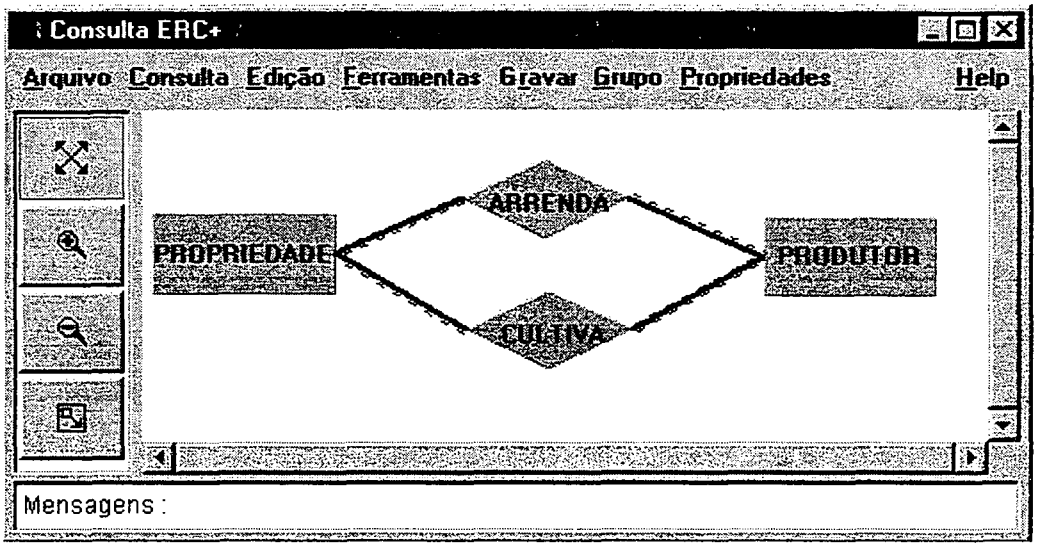


FIGURA 25: SUBESQUEMA SELECIONADO PARA CONSULTA

Por exemplo, na Figura 25 a raiz deve ser determinada para estabelecer se o usuário deseja uma visão de produtor e quais propriedades ele cultiva, ou da propriedade e quem as cultiva, pois, há dois pontos de vista possíveis, dependendo do

sentido da leitura feita sobre o relacionamento. A exclusão de ciclos para formular a consulta é necessária, pois o processo de construção de uma consulta é uma expressão algébrica que não poder incluir recursão[80] nem fechamento transitivo (DENNEBOUY et al., 1995).

Analisando-se estas tarefas sobre um exemplo em que o usuário deseje "selecionar os produtores que cultivam e arrendam propriedades ". O esquema selecionada para consulta é dado pelo gráfico cíclico da Figura 25. Entretanto para que essa consulta possa ser submetida, deve-se determinar como raiz o tipo entidade *PRODUTOR* e romper o ciclo, desconectando, por exemplo, a ligação entre *CULTIVA* e *PROPRIEDADE*. Este resultado pode ser visto na Figura 27.

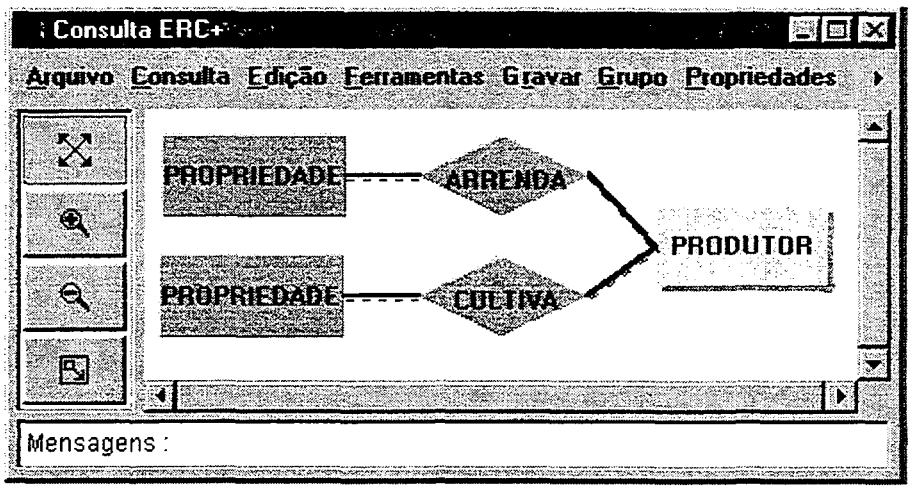


FIGURA 27: EXEMPLO DE UMA CONSULTA REPRESENTADA POR UM GRAFO ACÍCLICO E COM RAIZ ESCOLHIDA

A desconexão é uma operação em que o usuário seleciona a ligação[81] com o mouse e então escolhe a opção, do *menu*, *desconecta* e, então, a aplicação automaticamente duplica o objeto comum da ligação, eliminando o ciclo.

---

[80] Porém expressões cíclicas ainda são permitidas mediante eqüijoins, definidos pelo usuário na especificação do predicado.

[81] Aresta.

Para determinar a raiz o usuário deve selecionar o objeto através do mouse e escolher a opção *raiz*, a partir do *menu consulta*.

Ainda, nessa etapa de consulta, é disponibilizado um recurso de flexibilidade de bastante importância. A fim de complementar consultas, o usuário pode adicionar subesquemas a ela. Considere que o usuário deseja acrescentar o subesquema de proprietário à consulta, ele deve voltar ao esquema original, selecionar os objetos desejados e utilizar a opção *adiciona à consulta*, do *menu consulta*, que permite adicionar objetos a uma consulta e, em seguida, voltar à janela de consulta a fim de unificar os subesquemas, onde o usuário deve selecionar o objeto comum aos dois subesquemas, pois pode haver mais de um objeto comum entre eles. Estas operações são demonstradas pelas Figura 28 e Figura 29.

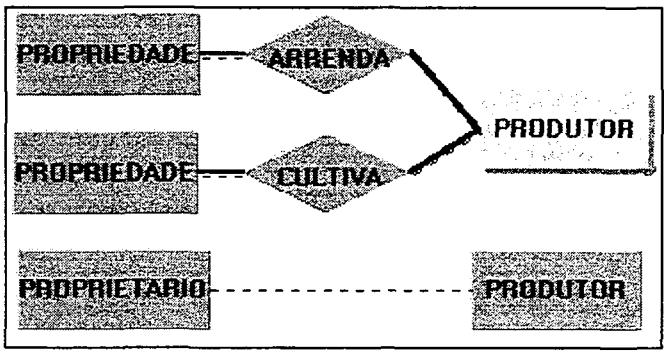


FIGURA 28: SUBESQUEMA PROPRIETÁRIO ADICIONADA À CONSULTA

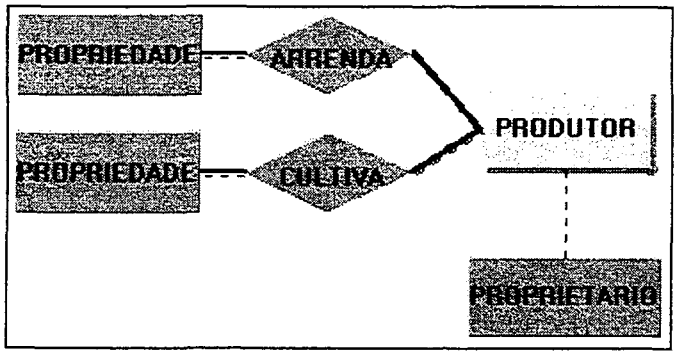


FIGURA 29: CONSULTA APÓS A UNIFICAÇÃO DOS SUBESQUEMAS

A próxima etapa da consulta é a especificação de predicados. Para isso o usuário deve, ainda na janela de consulta, selecionar a opção, do *menu, predicado*, então a aplicação gera uma nova janela de interação contendo os elementos selecionados na etapa de consulta. A apresentação da construção de predicados é mostrada na subseção seguinte.

6.3 ESPECIFICAÇÃO DE PREDICADOS

O público alvo pretendido por esta aplicação são usuários não especialistas, por isto, um ponto crítico para ela e outras aplicações de consulta para bases de dados de sua categoria é a especificação de predicados, por se tratar de uma tarefa que pode atingir um elevado nível de complexidade. Algumas ferramentas tentam prover uma representação visual das condições lógicas estabelecidas no predicado, entretanto, tipicamente são ambíguas e difíceis de ler (DENNEBOUY et al., 1995).

No protótipo VIQUEN, para que se alcançasse a abordagem em dispensar o usuário de conhecer a linguagem de manipulação do modelo, a grande maioria dos operadores e operações estão implícitos dentro da representação gráfica do modelo, ou seja, a linguagem visual para especificação do predicado também utiliza os próprios conceitos do modelo, apenas algumas operações exigem complementação pelo usuário.

Sob uma visão preliminar, a fim de demonstrar a especificação de predicado sobre o exemplo a Figura 29, assuma-se que o usuário escolheu como raiz a entidade *PRODUTOR*. A hierarquia dos objetos participantes da consulta, determinada pela escolha do usuário, conduz ao seguinte predicado: "Selecione os produtores que também sejam proprietários e que cultivam e arrendam propriedades". O operador à ser utilizado, sob a visão da álgebra ERC+, é o operador *sel-r-join*[82], uma vez que a exibição de cada ocorrência do tipo entidade *PRODUTOR*, na resposta, depende da

[82] E a combinação dos operadores primários, *selection* e *r-join*.

existência dos objetos a que ele está ligado, e a representação gráfica determina uma hierarquia de tipos objetos. O resultado desta consulta é um único tipo entidade *PRODUTOR* com os demais objetos participando como seus atributos complexos.

Assim, o usuário se utilizou do operador e impôs a ordem dos operandos, apenas pela representação visual, determinada pela escolha da raiz e quebra de ciclos da etapa anterior, sem nenhum conhecimento da linguagem de manipulação de dados. A representação gráfica do predicado é similar a exibida para a consulta e é apresentada na Figura 30.

A representação visual das ligações envolvendo os objetos do predicado é distinta. Sob a representação em forma de árvore do grafo, se na ligação, no esquema original[83], o *nodo filho* é do tipo atributo complexo, então a ligação é representada por uma linha dupla, tal qual ocorre na representação deste objeto do modelo ERC+ e, caso contrário, por uma linha simples. Esta abordagem serve para distinguir se o tipo de operador que será aplicado sobre o objeto. Para atributo complexo *r-join*, caso contrário *sel-r-join*.

Após a exibição gráfica inicial, o predicado pode ser alterado e complementado. Por exemplo, supondo-se que o usuário deseje modificar a consulta para exibir somente aquele produtor, o qual também seja um proprietário e tenha cultivado propriedades no ano de 1999 e, caso este produtor também tenha arrendado esta propriedade, exibir os dados do arrendamento. Observa-se que não há mais dependência em arrendar e cultivar propriedades. Estas alterações, vistas sobre o prisma da álgebra ERC+, levam a substituição do operador *selection* pelo operador *reduction*, sobre o tipo relacionamento *ARRENDA*, e também no acréscimo de seleção, sobre o tipo relacionamento *CULTIVA*.

---

[83] Resultante da engenharia reversa.

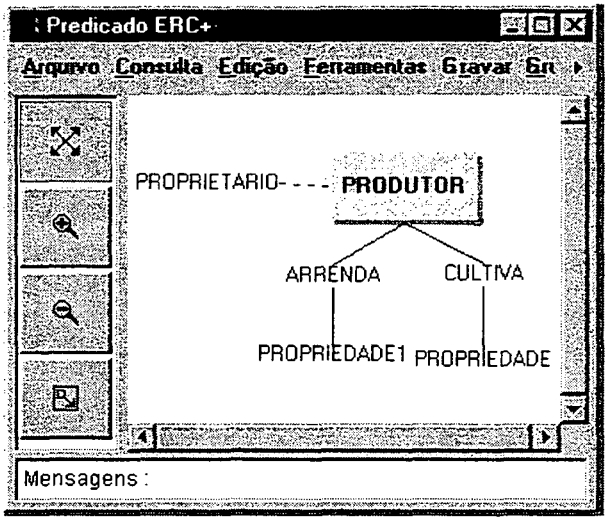


FIGURA 30: REPRESENTAÇÃO GRÁFICA DO PREDICADO SOBRE A ENTIDADE *PRODUTOR.*

Para fazer estas tarefas, o usuário deve primeiramente selecionar o objeto *ARRENDA*, em seguida ativar a janela de complementação de predicado através do botão direito do *mouse*, e nesta janela selecionar a opção redução.

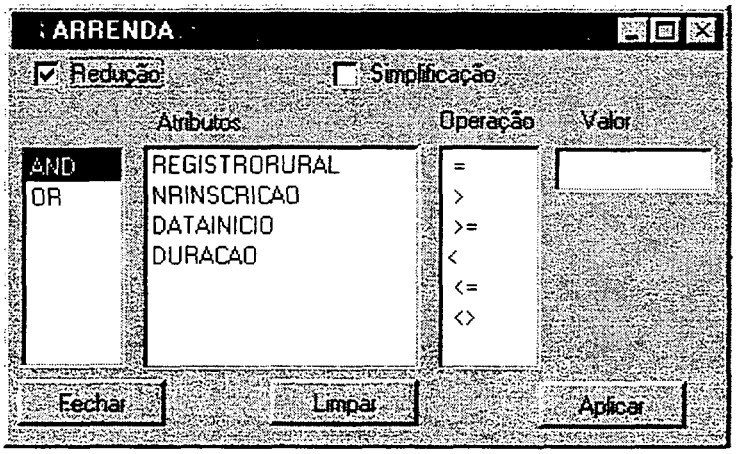


FIGURA 31: OPERAÇÃO DE REDUÇÃO SOBRE O TIPO RELACIONAMENTO *ARRENDA*

Repetindo os mesmos passos para o tipo relacionamento *CULTIVA*, ao invés de selecionar a opção de redução, através de uma janela de diálogo, o usuário deve selecionar o atributo *ano* e estabelecer o valor de restrição 1999. Também, podem ser

combinados operadores lógicos, para se construir expressões lógicas sobre os atributos atômicos. A Figura 31 e a Figura 32 exibem estas duas operações.

A Figura 33 exibe os predicados estabelecidos pelo usuário, nas folhas de uma árvore com raiz, e também a instrução SQL, que foi gerada automaticamente pela aplicação.

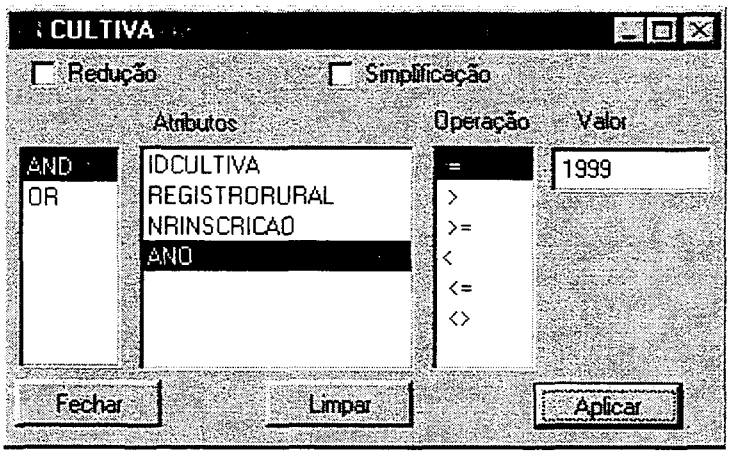


FIGURA 32: RESTRIÇÃO SOBRE UM ATRIBUTO ATÔMICO

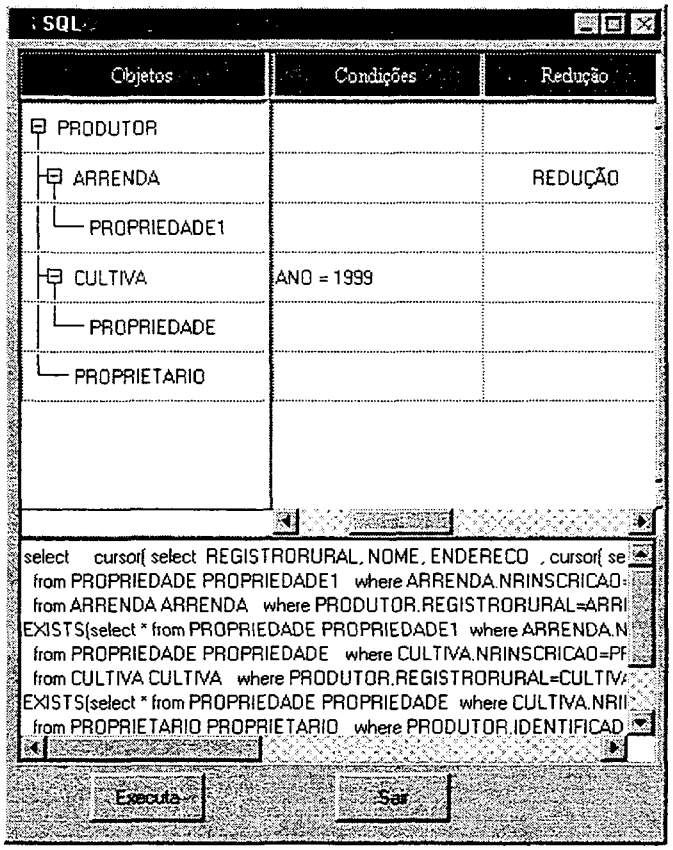


FIGURA 33: PREDICADO E EXPRESSÃO EM SQL

6.4 VISUALIZAÇÃO DA RESPOSTA

A última etapa do uso da aplicação é a visualização da resposta, onde os objetos selecionados durante a consulta permanecem igualmente vistos como objetos e não como estruturas planares, comumente apresentadas pela maioria das ferramentas de consultas a bases relacionais e de difícil entendimento pelo usuário final. Este resultado pode ser visto na Figura 34.

FIGURA 34: RESPOSTA DE CONSULTA MANTENDO A ESTRUTURA DE OBJETOS.

Na resposta, cada ocorrência do objeto escolhido como raiz da consulta pode ser expandida para a estrutura hierárquica determinada anteriormente na formulação da consulta e ,assim, ter seus valores avaliados sobre este ponto de vista.

6.5 DETALHES DA IMPLEMENTAÇÃO

Este protótipo foi construído nas seguintes camadas: interface gráfica, acesso a dados e regras de negócio, utilizando a linguagem C++. Para a camada de interface gráfica foram utilizadas as bibliotecas gráficas ILOG Views. O SGBD utilizado foi Oracle 8i™, entretanto, os esquemas objeto da engenharia reversa e, também da consulta. foram exclusivamente relacionais. Atualmente essa aplicação utiliza o sistema operacional Windows 98, contudo há versões das bibliotecas e SGBD utilizados para o sistema operacional Linux™, igualmente o código fonte foi construído dentro de padrões que permitem esta portabilidade.

A camada de interface do usuário está construída sob uma classe que implementa edição de grafos, sendo que existem três instâncias dessa sendo utilizadas para disponibilizar as janelas exibidas nas etapas de formulação de consultas: janela que exibe o esquema proveniente da tradução relacional/ERC+; janela que exibe o subesquema desejado para consulta e a janela em que o usuário especifica o predicado. Ainda na interface são exibidas outras duas janelas principais, a que mostra os predicados e a que exibe a resposta, ambas na forma de uma árvore.

As principais rotinas da camada de regras de negócio são:

- consistência do grafo. Conforme apresentado na subseção 6.2 para submeter uma consulta o usuário deve romper ciclos, determinar uma raiz e manter o grafo conexo, esta rotina verifica o cumprimento destes requisitos;
- construção de predicados/objetos de consulta. Esta rotina constrói um grafo com os elementos presentes na consulta e na especificação do predicado, e atribui às folhas do grafo os valores do predicado, estabelecidos pelo usuário;
- construção da expressão SQL. A partir do grafo que mantém o predicado e os objetos de consulta, a rotina constrói a sentença SQL, Ainda, a rotina

avalia o emprego dos operadores *reduction* e simplification, indicados pelo usuário, a fim de produzir a expressão SQL correta.

As principais rotinas da camada de acesso a dados são:

- construção do conjunto de *views*[84] que formam o dicionário do esquema ERC+: de acordo com as regras apresentadas na seção 4, um conjunto de visões é criado pelo *script* apresentado no apêndice A. São estas visões que dão suporte a todo o protótipo. Este *script* deve ser executado pelo administrador do SGBD antes da execução da aplicação;
- execução da expressão SQL: esta rotina resolve, recursivamente, todas as consultas/subconsultas que implementam o mapeamento dos operadores algébricos ERC+, retornando um ponteiro para a raiz da árvore que descreve cada ocorrência dos tipos objetos submetidos à consulta.

[84] Visões.

7 CONCLUSÃO

7.1 QUANTO AOS OBJETIVOS

Os objetivos propostos nesse trabalho dividem-se em três grupos e foram atingidos da seguinte forma:

- método para extrair esquemas ERC+: Através da elaboração de regras para extrair esquemas ERC+ a partir de esquemas relacionais, baseadas exclusivamente no conteúdo de dicionário de dados relacionais, chegou-se a um método que permitiu o enriquecimento semântico da tradução, quando comparado a um método de tradução direta entre os conceitos básicos dos modelos envolvidos. Todos os tipos de associações existentes em esquemas relacionais foram abordados.
- mapeamento dos operadores algébricos ERC+: todos os operadores algébricos primários e o principal operador derivado, *sel-r-join*, foram mapeados. Apesar dos esquemas e dicionário de dados, alvo de consulta e tradução, serem puramente relacionais, foi utilizado nesses mapeamentos um operador da linguagem objeto-relacional proprietária do SGBD Oracle. Esta característica limita sua implementação e restringe o uso a este SGBD. Entretanto, este protótipo pode acessar esquemas relacionais implementados (ou importados), tanto diretamente nesse SGBD, quanto indiretamente, na grande maioria de outros SGBDs, através de *links*, implementados pelo SGBD em questão, conhecidos como *database links*.
- protótipo de ambiente de consulta visual: foi implementado, nesse protótipo, o método para extração do esquema ERC+ e os principais mapeamentos da álgebra desse modelo. O protótipo VIQUEN permitiu que o esquema relacional, adotado no estudo de caso, fosse traduzido de relacional para ERC+, com o enriquecimento semântico esperado. Também foi possível disponibilizar ao usuário a elaboração de consultas sem nenhum conhecimento sintático da linguagem de manipulação de dados, apenas com os conceitos do modelo e da diferenciação dos operadores *r-join* e *selection*. Diversas consultas foram elaboradas e, uma das características

mais desejada foi alcançada, a resposta da consulta preservou a característica de objetos do modelo.

- esse trabalho não abordou aspectos de qualidade de interface em si mesmo, exceto na viabilização do uso de esquemas ERC+ na interface com o usuário. Outrossim, os aspectos de interface são objeto de outras dissertações associadas, em andamento.

7.2 TRABALHOS FUTUROS

Entre futuras implementações previstas para esse protótipo estão:

- funcionalidades de um editor de esquemas ERC+, a fim de permitir ao usuário criar novos esquemas conceituais;
- substituição da atual janela de diálogo, que permite comutar entre os operadores *r-join* e *sel-r-join*, por manipulação direta do esquema ERC+;
- inclusão das operações, atualmente, disponíveis somente através de *menu* e/ou botão direito do *mouse*, também por meio de ícones;
- elaboração de esquemas relacionais a partir de esquemas ERC+, permitindo também o processo inverso ao da reengenharia;
- edição de múltiplos esquemas, com o acréscimo de funções para integração de esquemas federados, utilizando ERC+ como modelo canônico, tendo como objeto bases relacionais, tendo em vista as facilidades já implementadas de acesso a essas bases;
- reutilização de consultas previamente gravadas como objetos em novas consultas;
- visualização de respostas através de formulários mais adequados às versões impressas;
- uso alternativo de técnicas de inteligência artificial para elaboração de predicados, como linguagem natural;

- inclusão de rotinas de acesso a outros SGBDs relacionais, bem como à nova geração, emergente, de SGBDs que implementam esquemas objeto-relacionais;
- construção de um método para visualizar esquemas com elevado número de elementos;
- submissão desse protótipo a comparação com aplicações de consulta de mercado, mediante testes práticos de uso.

8 REFERÊNCIAS BIBLIOGRÁFICAS

ALBANO, A.; GHELLI, G.; ORSINI, R. A Relationship Mechanism for a Strongly Typed Object-Oriented Database Programming Language. In: 17th INTERNATIONAL CONFERENCE ON VERY LARGE DATABASES, 1991, Barcelona. *Annals-proceedings.* Barcelona, 1991. p. 565-575. ALBAN

ANDERSSON, M. Extracting na Entity Relationship Schema from a Relation Database Through Reverse Engineering. *International Journal of Cooperative Information Systems*, 1995. v. 4, n. 2, p. 259-285. M. AND

ANGELACCIO, M.; CATARCI, T.; SANTUCCI, G. Query by Diagram*: A Full Visual Query System. *Journal of Visual Languages and Computing*, 1990. v. 1, p. 255-273. ANGEL 199(

CHAVDA, M.; WOOD, P. T. Towards an ODMG-Compliant Visual Object Query Language. In: 23rd VLDB CONFERENCE ATHENS, 1997, Grécia. *Annals-proceeding.* Grécia, 1997. p. 456-465. CHAVD

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

DENNEBOUY, Y.; ANDERSSON, M.; AUDDINO, A.; YANN, D.; FONTANA, E.; GENTILE, M.; SPACCAPIETRA, S. Super: Visual Interfaces for Object + Relationship Data Models. *Journal of Visual Languages and Computing 5, Special Issue on Visual Query Language*, 1995. p. 73-99. DENNE

DENNEBOUY, Y. Flexibility of Visual Languages for Data Manipulation. In: IFIP WG2.6 3rd WORKING CONFERENCE ON VISUAL DATABASE SYSTEMS, 1995. *Annals-proceedings*. mar. p. 84-102. Y. Denn

KLAS, W.; FISCHER, G., ABERER, K. Integrating Relational and Object-Oriented Database Systems Using a Metaclass Concept. *Journal of Systems Integration*, 1994. vol. 4, n. 4, Kluwer Academic Publischer, 1994. KLAS, I

KNUTH E.; WEGNER L. M. *Visual Database Systems II.* North-Holland, 1992. KNUTH

KUNTZ, M.; MELCHERT, R. Pasta-3's Graphical Query Language: Direct Manipulation, Cooperative Queries, Full Expressive Power. In: 15th INTERNACIONAL CONFERENCE ON VERY LARGE DATA BASES, 1989. *Annals-proceedings*, p. 97-105. KUNTZ

MURRAY, N.; NORMAN, P.; GOBLE, C. *A Visual Query Language for Object Databases.* Manchester, 1998. Disponível em: < http://www.cs.man.ac.uk/~murrayn> Acesso em: 15 abr. 2000. MURRA

_____. *A 3D Environment for Querying ODMG Compliant Databases.* Manchester, 1998. Disponível em: <http://www.cs.man.ac.uk/~murrayn> Acesso em: 15 abr. 2000. MURRA

_____. *A Framework for Describing Visual Interfaces to Databases.* Manchester, 1998. Disponível em: <http://www.cs.man.ac.uk/~murrayn> Acesso em: 15 abr. 2000. MURRA

PAPANTONAKIS, A.; KING, P. J. H. Gql: A Declarative Graphical Query Language Base on the Funcional Data Model. *Annals-proceedings* Advanced Visual Interfaces, 1994. p. 113-122. PAPAN

PARENT, C. *L'approache ERC: un modèle de données et une algèbre de type entité-relation.* PAREN
thèse d'état. Université Pierre et Marie Curie (Paris VI), Paris, 1987.

RUMBAUGH, J. Relations as Semantic Constructs in an Object-Oriented. LANGUAGE J. RUM
OOPSLA CONFERENCE, 1987, Orlando - Flórida. *Annals Proceedings*. Orlando, Flo.,
October 4-8, 1987. p. 466-481.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. *Sistemas de banco de dados*. 3ª ed. SILBER
São Paulo: MAKRON Books, 1999. 199

SPACCAPIETRA, S.; PARENT, C.; SUNYE, M.; YETONGNON, K.; LEVA, A. D. *An* SPACC
Object + Relationship Paradigm for Database Applications. Readings in Object-Oriented
Systems, D. Rine (Ed.), IEEE Press, 1995.

SWAYER P. Proc. 2nd Int. Workshop on Interfaces to Database Sytems, 1994, Springer- P. SWA
Verlag, 1994. [12] [Swa94]

CATARCI, T.; CHANG, S. K.; COSTABILE, M. F.; LEVIALDI, S.; SANTUCCI, G. A CATAR
Graph-Based Framework for Multiparadigmatic Visual Access to Databases. *IEE Transaction
on Knowledge and Data Engineering*, v. 8, n. 3, jun. 1996. p. 455-473. [3][CCC+96]

ZHANG Z.Q.; MENDELZON A. O. A Graphical Query Language for Entity-Relationship ZHANG
Databases. In: DAVIS et al. *Entity-Relationship Approach to Software Engineering*. North-
Holland, 1983. p. 441-448.

APÊNDICE A

Nesse apêndice está descrito a implementação das regras e refinamentos utilizados na tradução de um esquema relacional para ERC+ visto neste trabalho. O mesmo gera o conjunto de visões (*views*) que são utilizados pelo protótipo VIQUEN para extrair e exibir o esquema ERC+. Esta rotina (*script*) está baseada no refinamento de visões. Inicialmente são criadas visões dos tipos básicos do modelo entidade - relacionamento e sobre essas novas visões são geradas até atingir-se a semântica dos objetos ERC+. Assim, o conjunto de visões resultantes constitui-se no dicionário de dados do modelo ERC+. Um usuário, denominado ERC_REL (ERC+ sob relacionamento), detém a criação do esquema, sendo que sinônimos públicos são criados para que qualquer proprietário de esquema possa efetuar a engenharia reversa, sendo que os próprios direitos de acesso sobre os objetos relacionais delimitam o esquema alvo da tradução.

A tarefa de execução desta rotina cabe ao DBA, o qual deve alterar usuário e senhas de acordo com os privilégios necessários para executá-la. Segue este *script*.

```
-------------------------------------------------------------------------
-- Script para gerar as views, a partir de qualquer base RELACIONAL,
-- que formam os elementos do modelo ERC+
-------------------------------------------------------------------------
connect system/manager;
DROP public synonym user_tab_columns_primarias
/
DROP public synonym  user_tab_columns_dependente
/
DROP public synonym user_tables_dependente
/
DROP  public synonym user_entidade
/
DROP public synonym user_entidade_dependente
/
DROP public synonym user_tab_relacionamentos
/
DROP public synonym user_tipo_relacionamento
/
DROP public synonym user_entidade_multi_instancia
/
DROP public synonym user_tipo_multivalorado
/
DROP public synonym  user_entidade_fraca
```

```
/
DROP USER ERC_REL CASCADE
/
---- executar script para criar tablespace apenas uma vez
CREATE TABLESPACE TBS_USER_DATA_ERC_R DATAFILE
'DF_USER_DATA_ERC_REL' SIZE 1M AUTOEXTEND ON NEXT 1M
/
CREATE USER "ERC_REL" IDENTIFIED BY "ERC" DEFAULT TABLESPACE
"TBS_USER_DATA_ERC_R"
 PROFILE DEFAULT QUOTA UNLIMITED ON "SYSTEM" QUOTA UNLIMITED ON
"TBS_USER_DATA_ERC_R"
 ACCOUNT UNLOCK
/
connect system/manager;
GRANT "CONNECT" TO "ERC_REL"
/
GRANT CREATE PUBLIC SYNONYM TO ERC_REL
/
GRANT SELECT ANY TABLE TO ERC_REL
/
--ALTER USER "ERC_REL" DEFAULT ROLE ALL
/
CONNECT ERC_REL/ERC
/
--1. Análise estrutural
--Determina o tipo das entidades em: Entidade-Indenpendente, Entidade-
Dependente, Tipo relacionamento
--- todas as colunas primárias de tabelas, i.e., que não estão em chave-
estrangeira
 create view user_tab_columns_primarias as
 (
 (select c4.table_name,c4.column_name from user_tab_columns c4)
 minus
  (select c5.table_name, c5.column_name from user_constraints c4,
user_cons_columns c5
              where                c4.constraint_type='R'                and
c5.constraint_name=c4.constraint_name )
 )
/
create public synonym user_tab_columns_primarias for
user_tab_columns_primarias
/
--
----------------------------
---tabelas dependentes: aquelas cujos componentes de suas respectivas
chaves primárias estejam:
---opcao a)contidos entre as colunas primárias de outras tabelas
--    select distinct c2.table_name from user_constraints c1,
user_cons_columns c2
--where c1.constraint_type='P' and c2.constraint_name=c1.constraint_name
-- and
-- exists
-- (select c3.table_name from user_tab_columns_primarias c3
--               where     c3.table_name<>c2.table_name
--                                 and  c3.column_name=c2.column_name)
--
---opcao b)a mesma relacao acima pode ser obtida pela interseção entre os
componentes da chave primária
--e os da chave estrangeira, i.e., se um componente de  chave primária
também participa como
```

```
--chave estrangeira esta tabela é dependente. Esta opcao é melhor por não
restringir a identificacao pela
--igualdade de nomes (sinonimos) de atributos em diferentes relações
create view user_tab_columns_dependente as
(
( select distinct c2.table_name,c2.column_name from user_constraints c1,
user_cons_columns c2
 where c1.constraint_type='P' and c2.constraint_name=c1.constraint_name)
 INTERSECT
 (select distinct c2.table_name,c2.column_name from user_constraints c1,
user_cons_columns c2
 where c1.constraint_type='R' and c2.constraint_name=c1.constraint_name))
/
create public synonym user_tab_columns_dependente FOR
user_tab_columns_dependente
/
create view user_tables_dependente as
(select distinct table_name from user_tab_columns_dependente)
/
create public synonym user_tables_dependente FOR user_tables_dependente
/
----------------------------
--tipos entidade-independente, i.e., aquelas que tem independência de
identificador
create view user_entidade as
( (select table_name from user_tables)
  minus
  (select table_name from user_tables_dependente)
)
/
create public synonym user_entidade for user_entidade
/
----------------------------
---relacionameto entre tabelas
create view user_tab_relacionamentos as
(select c1.table_name, c1.constraint_name, c2.table_name table_ref from
user_constraints c1, user_constraints c2
where c1.constraint_type='R' and c1.r_constraint_name=c2.constraint_name)
/
create public synonym user_tab_relacionamentos for user_tab_relacionamentos
/
----------------------------
--tipos relacionamento: constraints FK de tabelas com mais de dois
relacionamentos e com dependencia de identificador
create view user_tipo_relacionamento as
(           select distinct c1.* from user_tab_relacionamentos c1,
                          (select c2.table_name, count(*) nr_relac from
user_tab_relacionamentos c2,

user_tables_dependente c3
                            where c2.table_name=c3.table_name
                            group by c2.table_name) c4
                       where  c1.table_name=c4.table_name
                        and  c4.nr_relac > 1
)
/
create public synonym user_tipo_relacionamento for user_tipo_relacionamento
/
----------------------------
```

```
--tipos entidade-dependente. Aquelas que possuem constraints FK de tabelas
com um único relacionamento e com
--dependencia de identificador. Relaciona o nome das tabelas dependentes
com os dados das constraints
--de chave estrangeira ( user_tab_relacionamentos)
create view user_entidade_dependente as
(select c1.* from user_tab_relacionamentos  c1,
                      (--obtem o nome das tabelas dependentes
                        select table_name from  user_tab_columns_dependente
                          minus
                        select table_name from user_tipo_relacionamento) c2
where
c1.table_name=c2.table_name)
/
create public synonym user_entidade_dependente for user_entidade_dependente
/
-----------
--2.Tipo de ligação
--2.1. Multi-instanciação: se uma entidade dependente e uma independente
têm a mesma chave-primaria (candidata) então há uma ligação do
--tipo multi-instanciação.
-- ->Entre duas entidades E1 dependente, E2 independente relacionadas por
uma chave estrangeira fk, dado os conjunto PK1 e PK2 contendo as colunas
que compõe
-- as respectivas chaves primárias de E1 e E2 temos que se pk1-pk2 = 0
(conjunto vazio) há uma ligação do tipo multi-instanciação entre elas.

create view user_entidade_multi_instancia as (
select e1.* from user_entidade_dependente e1, user_entidade e2
where
 e1.table_ref=e2.table_name
and
not exists
    ( (select  c2.column_name from  user_constraints c1, user_cons_columns
c2
                      where c1.constraint_name=c2.constraint_name and
                                  e1.table_name=c1.table_name           and
c1.constraint_type='P')
       minus
      (select column_name from  user_constraints c1, user_cons_columns c2
                      where c1.constraint_name=c2.constraint_name and
                      e2.table_name=c1.table_name and
                      c1.constraint_type='P'
      )
   )
)
/
create      public      synonym      user_entidade_multi_instancia      for
user_entidade_multi_instancia
/
--2.1.1.Relacionamentos isa: (conferir este aqui para diferenciar de maybe)
create view user_entidade_isa as (
select e1.* from user_entidade_multi_instancia e1, user_entidade e2
where
 e1.table_ref=e2.table_name
and
not exists
    ( (select  c2.column_name from  user_constraints c1, user_cons_columns
c2
                      where c1.constraint_name=c2.constraint_name and
```

```
                                    e1.table_name=c1.table_name            and
c1.constraint_type='P')
       minus
      (select column_name from  user_constraints c1, user_cons_columns c2
                      where c1.constraint_name=c2.constraint_name and
                      e2.table_name=c1.table_name and
                      c1.constraint_type='P'
      )
   )
)
/
create public synonym user_entidade_isa for user_entidade_isa
/
-------------------------------------
--2.1.2.Relacionamento maybe: falta fazer e diferenciar de isa.
----------------------
--2.1.3. Tipo atributo multivalorado. Aquelas entidades dependentes que não
são multi-instancias e não estão ligadas
--através de um tipo relacionamento
create view user_tipo_multivalorado as (
            select t1.* from ( select * from user_entidade_dependente
                                    minus
                                    select            *            from
user_entidade_multi_instancia
                                  ) t1
             where
              t1.table_ref
              not in
              (select table_name from user_tipo_relacionamento)
)
/
create public synonym user_tipo_multivalorado for user_tipo_multivalorado
/
-------------------------------------
----------------------
--2.1.4. Tipo entidade-fraca. Aquelas entidades dependentes   que estão
ligadas
--através de um tipo relacionamento ou de um tipo multi-instanciação
create view user_entidade_fraca as
 select t1.* from ( select * from user_entidade_dependente
                                    minus
                                    select            *            from
user_entidade_multi_instancia
                                  ) t1
             where
              t1.table_ref
               in
              (select table_name from user_tipo_relacionamento
                union
               select table_name from user_entidade_multi_instancia)
/
create public synonym  user_entidade_fraca FOR  user_entidade_fraca
/
--2.2 Entidade e seus tipos
--Cria uma view para indicar a Entidade e seu tipo
create view user_entidade_tipo as (
  select table_name, 'INDEPENDENTE' Tipo from user_entidade
  union
  select table_name, 'ISA' Tipo from user_entidade_isa
  union
```

```
  select table_name, 'RELACIONAMENTO' Tipo from user_tipo_relacionamento
  union
  select table_name, 'MULTIVALORADO' Tipo from user_tipo_multivalorado
)
/
create public synonym  user_entidade_tipo FOR  user_entidade_tipo
/
--3. Cardinalidade
--3.1A cardinalidade maxima das ligações é determinada pelas restrições de
integridade presentes no dicionário de
--dados impostas sobre chaves estrangeiras, i.e., se há uma restrição de
unicidade imposta sobre o conjunto de
--atributos que formam a chave estrangeira então fica determinado a
cardinalidade máxima de uma ocorrência
--desta ligação, caso contrário cardinalidade n.
--A implementacao abaixo obtem a cardinalidade maxima. Cada conjunto A
contendo as colunas de uma
--chave estrangeira são comparados com cada conjuntos B desta tabela
contendo restricoes de unicidade
--se forem iguais a cardinalidade maxima é um, i.e.,  A-B  U  B-A = {}.
--obs: para obter as ligacoes com cardinalidade maior que deve ser feito um
not in na view abaixo,
--substituir not exists por exists nao é correto pelo fato de que a regra
para cardinalidade maxima um
--esta em que a existencia de um conjunto que contenha restricoes de
unicidade com os mesmos ele-
--mentos da chave é suficiente, entretanto para cardinalide maxima maior
que um nao basta trocar not
--exists por exists, isto traria apenas aquelas ligacoes cujo conjunto de
elementos de chave estrangeira
--nao esta contido em todos os demais conjunto de unicidade, o que ocorre
sempre que mais de uma defini
--cao de conjuntos de unicidade existir.
create view user_cardinalidade_maxima_um as (
 select  t1.constraint_name,  t1.table_name,  t4.table_name  table_ref  from
user_constraints t1, user_constraints t3,
   user_constraints t4
 where
 t1.r_constraint_name=t4.constraint_name  and
 t1.constraint_type='R'     and              t3.constraint_type='U'     and
t1.table_name=t3.table_name and
 not exists
(
 (  select t2.column_name from user_cons_columns t2
     where
     t1.constraint_name=t2.constraint_name
     minus
     select t2.column_name from  user_cons_columns t2
       where
     t3.constraint_name=t2.constraint_name
   )
 union
  ( select t2.column_name from  user_cons_columns t2
     where
     t3.constraint_name=t2.constraint_name
     minus
    select t2.column_name from user_cons_columns t2
     where
     t1.constraint_name=t2.constraint_name
    )
```