UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE CIÊNCIAS E TECNOLOGIA
CURSO DE MESTRADO EM INFORMÁTICA

SQL-FÁCIL :
UMA INTERFACE AMIGÁVEL PARA ACESSO A BANCO DE DADOS RELACIONAL

ED PÔRTO BEZERRA

CAMPINA GRANDE
NOVEMBRO - 1989

ED PORTO BEZERRA

SQL-FÁCIL :

UMA INTERFACE AMIGÁVEL PARA ACESSO A

BANCO DE DADOS RELACIONAL

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

ÁREA DE CONCENTRAÇÃO : BANCO DE DADOS

MARCUS COSTA SAMPAIO

Orientador

MARIA DE FÁTIMA Q. V. TURNELL

Co-orientadora

CAMPINA GRANDE

NOVEMBRO - 1989

DIGITALIZAÇÃO:

SISTEMOTECA - UFCG

SQL-FACIL :

UMA INTERFACE AMIGAVEL PARA ACESSO A

BANCO DE DADOS RELACIONAL

ED PORTO BEZERRA

DISSERTAÇÃO APROVADA EM 10/11/89

MARCUS COSTA SAMPAIO - M. Sc.

Presidente

MARIA DE FATIMA Q. V. TURNELL - Ph. D.

Examinadora

PEDRO SERGIO NICOLLETTI - M. Sc.

Examinador

RAIMUNDO HAROLDO DO CARMO CATUNDA - M. Sc.

Examinador

CAMPINA GRANDE

NOVEMBRO - 1989

AGRADECIMENTOS

Não fossem vocês - Orientador, Co-orientadora, colegas mestrandos e graduandos, professores e funcionários do DSC, companheiras e companheiros, familiares, paixões e, especialmente, amores passado e presente - teria sido muito mais difícil a realização deste trabalho. Vocês são tantos que a relação de nomes seria imensa e ao mesmo tempo injusta, pois merecem mais do que serem citados; merecem, cada um, sinceros discursos sobre o quanto representaram e representam para mim. Com certeza estarão sempre bem vivos e bem vindos na minha lembrança.

SQL-FÁCIL :

UMA INTERFACE AMIGÁVEL PARA ACESSO A

BANCO DE DADOS RELACIONAL

DISSERTAÇÃO APROVADA EM 10/11/89

MARCUS COSTA SAMPAIO
Orientador

MARIA DE FÁTIMA Q. V. TURNELL
Co-orientadora

PEDRO SÉRGIO NICOLLETTI
Componente da Banca

RAIMUNDO HAROLDO DO CARMO CATUNDA
Componente da Banca

CAMPINA GRANDE

NOVEMBRO - 1989

RESUMO

SQL-FÁCIL é uma interface amigável para acesso a banco de dados relacional baseada no conceito de Relação Universal. Esta interface suporta inferência automática de navegação lógica, livrando os seus usuários da preocupação sobre a organização lógica dos dados do banco de dados. A independência de navegação lógica não é provida pela interface padrão SQL. Os usuários de SQL-FÁCIL expressam suas consultas por meio de atributos, condições de seleção sobre atributos e nomes de relacionamentos da "janela" da Relação Universal; não há explicitação da operação de junção de relações e nem utilização de variável-tupla nas consultas.

ABSTRACT

SQL-FÁCIL is a user-friendly interface to access relational database based upon the concept of Universal Relation. This interface supports automatic inference for logical navigation, freeing the users from the preoccupation about the logic organization of data in the database. The independence from logic navigation is not provided by the standard SQL interface. The users of SQL-FÁCIL express their queries using attributes, selective conditions on attributes and relations' names from the Universal Relation window; there is no explicit statement on the join of relations nor the utilization of tuples in the statement of the queries.

SUMÁRIO Página

ÍNDICE DE FIGURAS

1 - INTRODUÇÃO

Este trabalho descreve uma interface amigável para acesso a banco de dados relacional, que chamamos de SQL-FÁCIL, a qual livra os usuários da responsabilidade da "navegação" lógica entre as relações do banco de dados. Por navegação, devemos entender a necessidade de especificar de que maneira as informações pretendidas se inter-relacionam (geralmente através da operação de "junção" de relações [MAIER 83]) - ela é lógica porque independe de como as relações estão armazenadas.

A interface relacional padrão SQL remove a necessidade da navegação física, mas não provê independência de navegação lógica. Por exemplo, consideremos um banco de dados que tem as relações **Empregados (Empregado, Departamento) e Departamentos (Departamento, Gerente)**. Se o usuário está interessado no relacionamento entre **Empregado** e **Gerente** através de **Departamento**, ele deve especificar a junção natural (Empregados.Departamento = Departamentos.Departamento) das relações **Empregados** e **Departamentos**. Esta junção é uma especificação de navegação lógica, sem a qual a resposta à consulta não poderia ser computada. Logo, o usuário deve conhecer **quais** relações estão no banco de dados (BD) e **como** elas podem ser unidas, ou seja, o usuário deve conhecer a organização lógica dos dados do BD para consultá-lo, o que, convenhamos, não é nada amigável.

Existem várias interfaces relacionais ditas amigáveis, além de SQL, como por exemplo QUEL [MAIER 83], QBE [MAIER 83], etc., nas quais a navegação lógica deve ser especificada pelo usuário. Propomos uma interface baseada no conceito de **Relação Universal** [ULLMAN 83], que suporta inferência automática de navegação lógica. Esta interface dispensa do usuário a preocupação sobre a organização lógica dos dados, permitindo que este consulte o BD sem qualquer conhecimento das estruturas física e lógica deste.

O tratamento de ambiguidade na interpretação de consultas é uma característica comum a todas as interfaces baseadas no conceito de Relação Universal. Entendamos por ambiguidade a possibilidade de obtenção de mais de uma resposta para uma consulta. Destacaremos a simplicidade de SQL-FÁCIL na resolução de ambiguidade, comparativamente com a solução adotada por dois outros sistemas chamados System/U e DURST.

No CAPÍTULO 2, apresentamos o conceito de Relação Universal, descrevemos e analisamos a interface chamada System/U, a qual utiliza este conceito, e comentamos resumidamente três outros sistemas afins chamados DURST, PIQUE e FIDL.

No CAPÍTULO 3, apresentamos a interface SQL-FÁCIL destacando sua interação amigável com o usuário e apresentamos um novo conceito por nós criado que se chama Janela da Relação Universal. Também mostramos exemplos comentados, abordando características diversas, de consultas em SQL-FÁCIL e suas respectivas consultas em SQL.

No CAPÍTULO 4, conceituamos o pré-processador da interface SQL-FÁCIL e detalhamos suas principais estruturas de dados implementadas (com explicações e exemplos de cada uma delas). Também comentamos os analisadores léxico e sintático e descrevemos um procedimento de permutação de elementos de um vetor, destacando sua importância à implementação da interface SQL-FÁCIL.

No CAPÍTULO 5, resumimos as principais conclusões tiradas no desenvolvimento da interface SQL-FÁCIL, descrevendo comentários comparativos com as outras interfaces que utilizam o conceito de Relação Universal; e sugerimos melhorias na versão atual da interface SQL-FÁCIL a título de trabalhos futuros.

2 - A RELAÇAO UNIVERSAL

Neste capítulo, apresentamos o conceito de Relação Universal, descrevemos e analisamos criticamente o sistema experimental chamado System/U onde este conceito é aplicado. Também comentamos outros três sistemas em desenvolvimento, chamados DURST, PIQUE e FIDL.

2.1. Conceito de Relação Universal.

Vamos imaginar que os dados de um BD sejam colocados numa só relação, chamada Relação Universal (RU), cujo esquema consista de todos os atributos em quaisquer dos esquemas das relações originais de um BD. Uma RU não existe de fato (ela é virtual). Assumamos que atributos representando a mesma coisa em diferentes relações tenham o mesmo nome, e atributos representando coisas diferentes tenham nomes diferentes. Esta suposição básica, chamada suposição do esquema da RU, incorpora parte da semântica de um BD nos atributos da RU. Os atributos na RU assumem ter um papel único. Por exemplo, um atributo como **nome** não pode significar nome de empregado, nome de cliente e nome de fornecedor, ao mesmo tempo - somente um dos três significados, ou um outro, tem que ser determinado. A grande vantagem da RU é que o usuário é poupado de memorizar detalhes sobre quais atributos estão agrupados para formar relações. Resumidamente, podemos conceituar uma RU como um conjunto de atributos sobre o qual o

usuário formula suas consultas. Estas são expressas por meio de nomes de atributos e condições de seleção sobre atributos desse conjunto.

Suponhamos um BD com os atributos **Empregado, Departamento** e **Gerente** cujo esquema da RU é **(Empregado, Departamento, Gerente)**. Um usuário poderia expressar-se da seguinte maneira :

```
SELECT Departamento
WHERE  Empregado = "Jose"
```

sem preocupar-se se no BD há uma só relação com esquema (Empregado, Departamento, Gerente), ou duas relações (Empregado, Departamento) e (Departamento, Gerente), ou ainda (Empregado, Gerente) e (Departamento, Gerente). A resposta, desconsiderando-se a estrutura do BD, deve ser a mesma : o departamento de José.

Não vamos inferir deste exemplo que nenhum conhecimento semântico do BD é requerido do usuário. Achamos que é mais fácil para o usuário entender conceitos como Empregado e Departamento e seu relacionamento do que entender o que significa uma relação (Empregado, Departamento). Nossa proposta, ao usar o conceito de RU, é ir um passo além do Modelo Relacional [CODD 70], por remover do usuário não apenas a preocupação sobre a organização física dos dados, mas também sobre a organização lógica. É razoável pensar que todas as consultas tornam-se mais

fáceis de expressar em sistemas que suportam a RU, comparativamente com aqueles que não utilizam tal conceito.

2.2. Sistemas que utilizam o conceito de Relação Universal.

2.2.1. System/U.

O System/U (Universal System) foi desenvolvido na Universidade de Stanford nos Estados Unidos. Seu modelo de dados é baseado nos conceitos de **atributos**, **objetos** e **objetos maximais** [KORTH 84]. Objetos são conjuntos mínimos de atributos que têm significado coletivo. Cada objeto está contido em um esquema de uma relação. Falaremos mais adiante sobre os objetos maximais.

O System/U está interagindo com uma interface para o sistema chamado ERIS (Experimental Relational Information System) [REIS 83]. Uma consulta feita no System/U é processada e traduzida para a linguagem ERIS a qual recupera a informação; então o System/U a exibe e a armazena num arquivo. Ambos os sistemas são escritos em C e rodam sob o sistema operacional Unix.

O System/U permite mudança de nomes de atributos dos objetos de modo que a mesma relação possa ser usada por muitos objetos que são efetivamente idênticos, como veremos no seguinte exemplo : suponhamos uma genealogia baseada numa só relação (Filho, Pai). O projetista do BD declara os atributos **Pessoa, Pai, Avô e Bisavô** e os objetos **Pessoa-Pai, Pai-Avô e Avô-Bisavô**; cada um destes definido sobre a relação (Filho, Pai), com a correspondência óbvia de atributos. A seguinte consulta, na linguagem de consulta do System/U (detalhada adiante), resultaria nos bisavôs de Maria :

```
retrieve (Bisavô)
where    Pessoa = "Maria"
```

O System/U é operacionalizado através de duas ferramentas : a Linguagem de Definição de Dados e a Linguagem de Consulta.

- Linguagem de Definição de Dados.

A Linguagem de Definição de Dados é usada para definir o esquema do BD. O projetista ou administrador do BD declara os atributos e seus tipos de dados, nomes das relações e seus esquemas (conjuntos de atributos), objetos e, finalmente, as dependências funcionais. O projetista deve especificar de qual relação o objeto é formado. Vejamos o BD Banco-Comercial, definido como segue :

```
integer emprestimo conta;
float total saldo;
char [20] banco;
char [25] cliente;
char [50] endereco;
relation rcliente = cliente, endereco;
relation remprestimo = cliente, banco, emprestimo, total;
relation rconta = cliente, banco, conta, saldo;
object ocliente in rcliente = cliente, endereco;
object oemprestclient in remprestimo = cliente, emprestimo;
object oemprestbanco in remprestimo = banco, emprestimo;
object oempresttotal in remprestimo = emprestimo, total;
object ocontacliente in rconta = cliente, conta;
object ocontabanco in rconta = conta, banco;
object ocontasaldo in rconta = conta, saldo;
conta -> banco;
conta -> saldo;
emprestimo -> banco;
emprestimo -> total;
cliente -> endereco;
```

Após a definição dos dados, é feita a computação dos objetos maximais.

A figura 1 mostra um hipergrafo definindo a estrutura do BD Banco-Comercial onde os nós são os atributos e os arcos são os objetos. Cada objeto maximal é representado por um círculo envolvendo seus objetos.

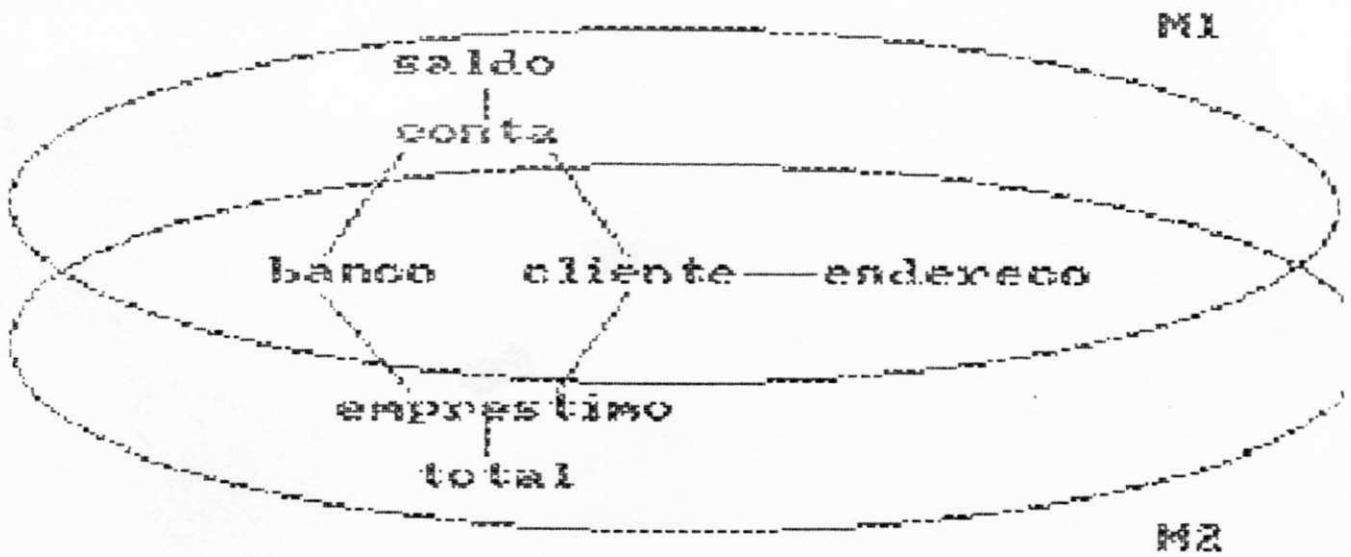


Figura 1 - Hipergrafo representando o BD Banco-Comercial.

Objetos maximais são conjuntos de objetos definidos pelo System/U a partir dos objetos. Os objetos maximais são abrangidos pelas variáveis-tupla [MAIER 83] de uma consulta. A criação de objetos maximais objetiva a resolução de ambiguidades, as quais estão presentes em BD cíclico [ULLLMAN 83]. Entendamos por BD cíclico aquele que possui atributos ambíguos (atributos com mais de um significado) como por exemplo os atributos **banco** e **cliente** da figura 1.

Vejamos, de uma maneira geral, como foi construído o objeto maximal M2 da figura 1. Seja j a junção natural dos objetos **ocliente**, **oemprestcliente**, **oempresttotal** e **oemprestbanco**. Seja Y = {banco} e Z = {cliente}, onde Z é disjunto de Y. Para que o objeto maximal M2 seja computado, é necessário provar que a dependência multivalorada Y ->-> Z segue logicamente de j. Isto é verdade se, e somente se, Z continua pertencendo a j quando Y for excluído de j. Logo, podemos afirmar que Z continua pertencendo aos objetos **ocliente** e **oemprestclient** quando Y for deletado do objeto **oemprestbanco**, ou seja, podemos afirmar que existe a dependência multivalorada **banco ->-> cliente** e consequentemente M2 é um objeto maximal. Para uma melhor compreensão de como é feita a computação de objetos maximais, sugerimos a leitura de [ULLMAN 83].

- Linguagem de Consulta.

O algoritmo de interpretação da consulta do System/U tem como entrada uma consulta, juntamente com os esquemas das relações, objetos e objetos maximais do BD. A saída é uma expressão algébrica, representando a interpretação da consulta. A expressão é a união de termos (produtos de objetos maximais), sendo que cada um deles será otimizado por um algoritmo otimizador.

A Linguagem de Consulta do System/U é essencialmente QUEL, com a seguinte diferença : não é preciso a declaração de variáveis-tupla, pois elas abrangem apenas a RU.

Uma consulta no System/U é constituída das seguintes cláusulas :

retrieve (lista-de-atributos) : Cada atributo da lista é da forma variável-tupla.atributo.

where (predicado) : O parâmetro (predicado) é uma combinação booleana de condições de seleção sobre atributos.

O System/U supõe que há uma variável-tupla "invisível", chamada **branco**. Logo, um atributo A sozinho significa branco.A.

- A conexão sobre o conjunto de atributos de uma consulta.

Seja t uma variável-tupla numa consulta e seja X o conjunto de atributos A tal que t.A aparece nesta consulta. A conexão sobre X, denotada [X], é o relacionamento entre os atributos que é inferido pelas relações do BD. Este não é necessariamente o único relacionamento possível sobre X, mas é o único calculado pelo System/U e considerado como sendo

intuitivamente o mais básico.

O algoritmo usado no System/U para encontrar [X] é o seguinte :

A - Encontre todos os objetos maximais contendo todos os atributos em X.

B - Caso :

- não existam objetos maximais então não há conexão natural entre os atributos em X, e [X] é vazio.

- exista 1 (um) só objeto maximal então [X] é a projeção da junção natural de todos os objetos que compõem o objeto maximal.

- existam vários objetos maximais então faz-se a junção natural de todos os objetos para cada objeto maximal, projeta-se nos atributos em X e toma-se a união dos resultados.

Comentemos o algoritmo, usando a seguinte consulta feita sobre o BD representado na figura 1 :

```
retrieve (banco)
where cliente = "Jose"
```

Entre os possíveis significados para esta consulta estão os seguintes :

a) Todos os bancos nos quais José tem uma conta e um empréstimo.

b) Todos os bancos nos quais José tem uma conta ou um empréstimo.

A resposta que o System/U dará é a b), pois "julga" que esta é a resposta pretendida pelo usuário. Se o usuário quiser a resposta a) ou tiver qualquer outro significado na mente, ele pode ainda obrigar o System/U a dar a resposta. Para obrigar a resposta a), o usuário formularia a consulta abaixo.

```
retrieve (banco)
where t.cliente = "Jose" and v.cliente = "Jose"
```

Notemos que duas variáveis-tupla (t e v) são necessárias à consulta; uma para cada objeto maximal.

Particularmente, achamos que o System/U não devia determinar um relacionamento sobre [X] como sendo o mais básico ou o que representa a intuição do usuário. Tal mecanismo atribui um só significado a uma consulta, assumindo que este é o significado pretendido pelo usuário. Julgamos que esta assunção é inconveniente, pois caso o usuário queira expressar um outro significado em sua consulta ele terá de formular uma nova

consulta, possivelmente usando variável-tupla e, para tanto, necessitando conhecer os objetos maximais. Cremos que isto foge ao objetivo da RU.

O System/U é incapaz de atribuir um significado a certas consultas, pelo fato dos atributos das mesmas não estarem contidos em nenhum dos objetos maximais. Neste caso, a consulta não é respondida. Vejamos a seguinte consulta sobre o BD da figura 1 para a qual não há resposta :

retrieve (cliente, total)

where saldo > 1.000,00

A variável-tupla branco é a única que aparece. Uma vez que nenhum objeto maximal contêm todos estes atributos (cliente, total e saldo), nenhum objeto maximal é abrangido pela variável-tupla branco e o System/U não responde esta consulta por considerá-la sem significado.

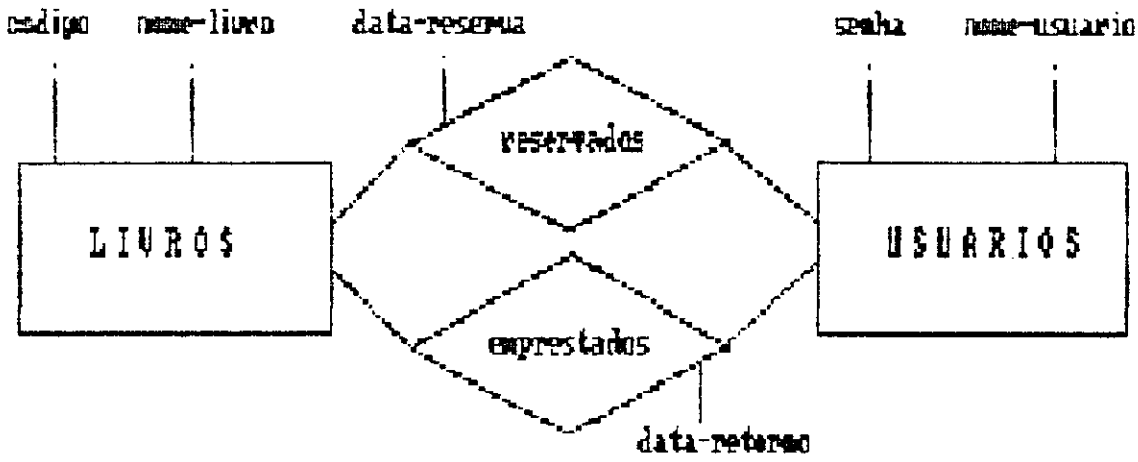
2.2.2. DURST.

O DURST (Datenbank mit Universalrelation-Schnittstelle) [BISKUP & BRUGGUEMANN 83], em desenvolvimento na Universidade de Dortmund na Alemanha Ocidental, é um sistema que suporta uma abordagem da RU diferente da abordagem seguida pelo System/U.

A linguagem de consulta do DURST é basicamente QUEL, mas sem a declaração de variáveis-tupla, pois elas abrangem apenas a RU.

Um requisito fundamental no DURST é que o esquema do BD não seja ambíguo. Entendamos por esquema sem ambiguidade, aquele onde devem existir atributos característicos que são atributos que podem servir como substituto do nome da relação e, portanto, podem ser usados para remover ambiguidades na interpretação de consultas. O conceito de atributos característicos é básico para projetar esquema de BD sem ambiguidade. Se o esquema do BD é ambíguo (cíclico), e não há atributos característicos nas relações (podendo causar ambiguidades), estes podem ser criados objetivando exclusivamente a remoção de ambiguidades. Julgamos que a criação de atributos característicos para este fim desvirtua a função de um atributo, que deve ser a de ter uma significação natural na relação onde este atributo está inserido. Também achamos que a especificação

de tais atributos na cláusula WHERE torna a condição de uma consulta sem significação lógica. Vejamos a consulta "Quais os nomes de usuários que possuem livros reservados ?", feita na linguagem de consulta do DURST sobre o BD representado abaixo segundo o Modelo de Entidades e Relacionamentos (MER) [CHEN 76] :




```
retrieve nome-usuario
where data-reserva = data-reserva
```

Notemos que a condição, sem significação lógica, foi formulada para referir-se aos usuários que possuem livros reservados.

Salientamos que os atributos característicos data-reserva e data-retorno não foram criados objetivando remover ambiguidades, mas são usados para removê-las.

2.2.3. PIQUE.

PIQUE (PIts QUEry language) [MAIER 83] é uma interface experimental para o sistema PITS (Pie In The Sky) que está sendo desenvolvida na Universidade do Estado de New York e no Oregon Graduate Center. PIQUE é baseado nos conceitos de Associação e Objetos, os quais estão relacionados respectivamente aos conceitos de objetos e objetos maximais do System/U.

2.2.4. FIDL.

FIDL (Flexible Interrogation and Declaration Language) [VARDI 88] é um sistema que usa o conceito da RU de uma maneira diferente de todos os outros sistemas. Enquanto a maioria dos sistemas usa uma RU virtual, FIDL está baseado na implementação física da RU. A RU, chamada JNF (Joint Normal Form), é armazenada num hardware especializado chamado CAFS (Content-Addressable File Store). A abordagem do FIDL é igual a do DURST, exceto na implementação física da RU.

3 - A INTERFACE SQL-FÁCIL

Neste capítulo apresentamos a interface SQL-FÁCIL, enfatizando os aspectos mais relevantes na interação com o usuário.

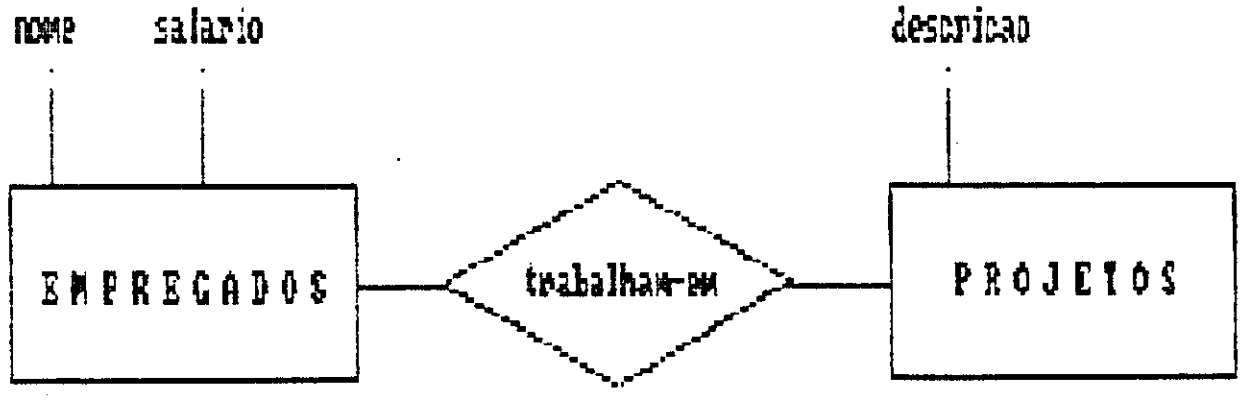
SQL-FÁCIL é uma interface ao estilo da interface SQL, mas verdadeiramente não-algoritmica. Sendo mais específicos, SQL só oferece independência física de dados : o usuário precisa especificar a navegação lógica no BD (materializada pelas junções naturais das relações do BD e pelos nomes de relações expressos na cláusula FROM) em adição às informações que ele necessita. Em SQL-FÁCIL as consultas são expressas por meio de atributos e condições de seleção sobre atributos; não há explicitação da operação de junção e nem utilização de variáveis-tupla.

A interface SQL-FÁCIL foi criada objetivando ter como usuário qualquer pessoa sem formação na área de informática ou conhecimento prévio de SQL (ao contrário, SQL tem sido adotada quase que exclusivamente por programadores profissionais). Para tanto criamos o conceito de janela da RU e projetamos a interface com uma sintaxe simples e com métodos de ajuda. Cremos que o usuário seja capaz de utilizar a interface SQL-FÁCIL logo após uma breve orientação, dada a priori pelo projetista do BD, sobre o que é a janela da RU e como formular consultas.

3.1. Conceito de Janela da Relação Universal.

O usuário da SQL-FÁCIL faz consultas utilizando-se de janelas da RU. Uma janela da RU é uma percepção dos relacionamentos conceituais entre conjuntos de atributos universais (objetos) sobre os quais as consultas são feitas. Cada usuário percebe o BD através de uma ou mais janelas da RU que lhe são fornecidas pelo projetista ou administrador do BD.

O modelo utilizado para descrever uma janela da RU é o Modelo de Entidades e Relacionamentos (MER). Escolhemos o MER pela sua ampla utilização na área de BD, pela sua simplicidade na representação das estruturas de informações no nível conceitual e consequente facilidade de assimilação pelo usuário. Para exemplificar, examinemos a janela da RU a seguir :



A leitura desta janela é feita da seguinte maneira : "empregados (ou um conjunto de atributos que descrevem um empregado) trabalham em projetos (cada projeto é descrito por um conjunto de atributos); inversamente, projetos têm empregados trabalhando neles " - fica claro que tem sentido uma pergunta do tipo "Quais as descrições dos projetos que têm empregados trabalhando neles e cujos salários são superiores a NCz$ 5.000,00?".

3.2. Aspectos importantes na interação com o usuário.

Alguns aspectos importantes abordados no projeto da interface são descritos a seguir :

- Método de acesso.

A entrada na interface é feita pela digitação dos caracteres **sqlfacil** seguidos pela tecla <Enter>. Logo após, a interface é carregada e executada exibindo inicialmente a tela de apresentação, mostrada na figura 2, a qual contém informações sobre o que é, e como utilizar a interface.

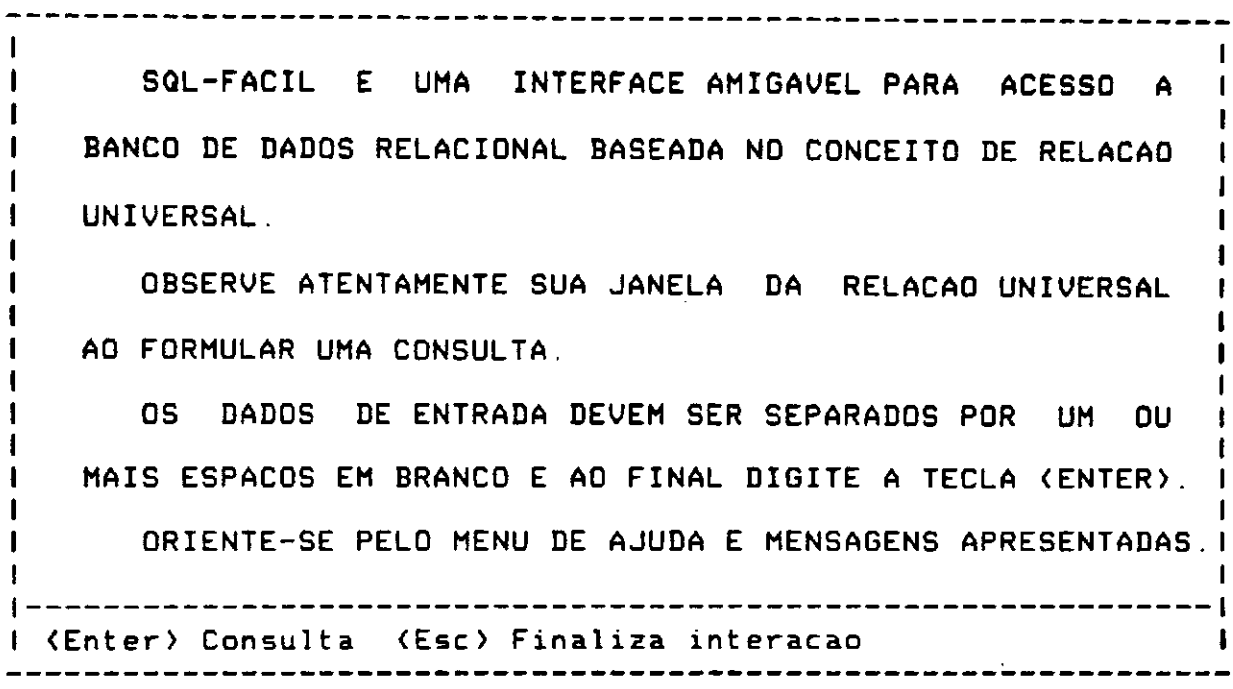


Figura 2 - Tela de apresentação da interface SQL-FÁCIL.

Não foi necessário criarmos nenhum mecanismo de segurança no acesso à interface, mas vale salientar que existe o mecanismo de segurança do BD (provido pelos Gerenciadores). Ao determinar quais os dados que cada usuário pode consultar, quando da criação da janela da RU para cada usuário, o projetista do BD também está sendo o responsável pela segurança do BD e cada usuário está sendo o responsável pela segurança de sua(s) janela(s) da RU. Podemos imaginar que as janelas da RU assemelham-se às senhas dos sistemas convencionais.

- Lay-out das telas.

A figura 3 destaca todas as áreas da tela-padrão projetada para a interface. Estas áreas (de trabalho, de mensagens e de menu) têm suas funções descritas ao longo deste capítulo.

Figura 3 - Lay-out das telas da interface SQL-FÁCIL.

- Sintaxe da consulta em SQL-FÁCIL.

Uma consulta em SQL-FÁCIL é constituída de três cláusulas, das quais apenas a formulação da primeira é obrigatória. As palavras SELECT, WHERE e USING são exibidas pela interface na área de trabalho e os parâmetros de cada cláusula são entrados, via teclado, pelo usuário. A tecla ⟨Enter⟩ finaliza a entrada de dados de cada parâmetro, quando então é feita a crítica aos dados. As cláusulas são, respectivamente, as seguintes :

a) **SELECT** ⟨lista-de-atributos⟩ - O parâmetro ⟨lista-de-atributos⟩ é uma lista de atributos universais e expressa o significado dos dados obtidos na consulta. Os atributos da lista devem ser separados por um ou mais espaços em branco.

b) **WHERE** ⟨predicado⟩ - O parâmetro ⟨predicado⟩ é uma combinação booleana de condições de seleção feitas sobre atributos universais.

c) **USING** ⟨lista-de-relacionamentos⟩ - O parâmetro ⟨lista-de-relacionamentos⟩ é uma lista de nomes de relacionamentos da janela da RU do usuário. Esta cláusula é usada para remover ambiguidades ou para determinar um relacionamento qualquer, objetivando especificar a semântica correta da consulta. Os nomes de relacionamentos devem ser separados por um ou mais espaços em branco.

A escolha do espaço em branco como separador de dados nas cláusulas deu-se apenas por este separador ser mais natural do que outro qualquer.

A digitação da tecla <Enter> na cláusula USING indica o fim da consulta em SQL-FÁCIL.

- **Métodos de ajuda**.

Os métodos de ajuda ao usuário projetados na interface são o menu de teclas de função, as mensagens de erro e as mensagens de orientação, descritos a seguir :

- **Menu de teclas de função**.

Em cada tela exibida é mostrado um menu contextual de teclas de função (válidas naquele instante da interação).

TECLA	FUNÇÃO
End	- Finalizar a execução da interface.
Esc	- Cancelar uma consulta em SQL-FÁCIL antes de sua execução.
->	- Exibir a tela da consulta em SQL.
<-	- Exibir tela da consulta em SQL-FÁCIL.

Enter - Formular uma nova consulta.

F1 - Exibir a janela da RU do usuário.

F2 - Imprimir a janela da RU do usuário.

- Mensagens de erro.

São mensagens que especificam o tipo de erro cometido e como se recuperar dele. Estas mensagens são exibidas após a finalização de uma cláusula onde houve uma entrada inválida de dados.

Exemplo : FALTA ASPA EM CONSTANTE LITERAL.

- **Mensagens de orientação.**

São mensagens orientadoras das ações a serem executadas pelo usuário no momento em que se processa a entrada de dados.

Exemplo : DIGITE ATRIBUTO(S) UNIVERSAL(IS).

As mensagens de erro e de orientação são exibidas em letras maiúsculas para melhor destaque, uma vez que os dados de entrada são em letras minúsculas.

- **Método de detecção e recuperação de erros.**

Quando há erros (de sintaxe ou de semântica) uma mensagem de erro é exibida.

- Ambiente para execução.

A configuração atual necessária para a execução da interface SQL-FÁCIL é a seguinte :

- 1 micro-computador compatível com IBM-PC, com 640 KBytes de memória RAM. Este valor pode aumentar em função dos tamanhos das tabelas auxiliares (detalhadas no próximo capítulo). O sistema operacional é o D.O.S.

- 1 monitor de vídeo.

- 1 teclado simples.

- 1 disk-drive.

- 1 impressora.

- Documentação.

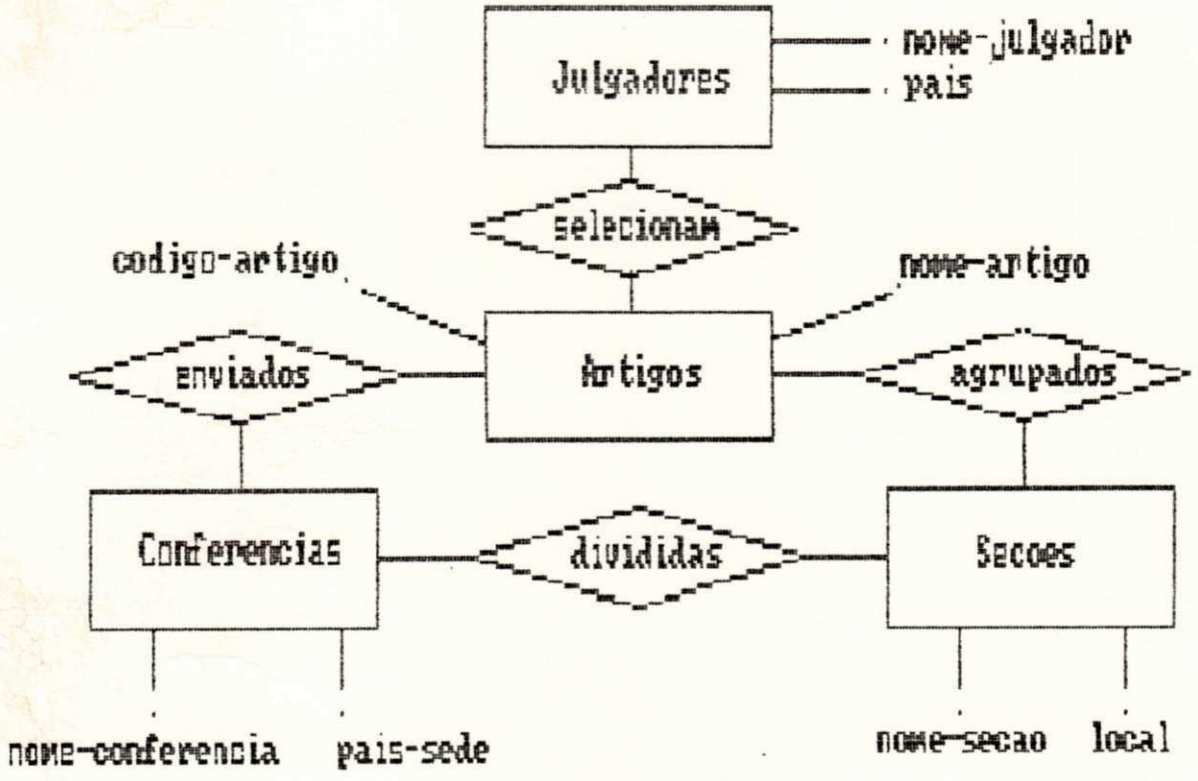
Julgamos ser dispensável a elaboração de uma documentação off-line (manuais), pois a interface foi projetada de modo que seja auto-explicativa. Entretanto, criamos o guia do usuário. Este guia, o qual objetiva complementar a informação dada pelo projetista do BD a um usuário iniciante, contém informações básicas sobre o que é uma janela da RU e exemplos de como especificar consultas em SQL-FÁCIL.

3.3. Exemplos de consultas em SQL-FÁCIL.

Vejamos, a seguir, alguns exemplos comentados de consultas em SQL-FÁCIL com suas respectivas transformações em consultas SQL.

Exemplo-1 : "Quais os artigos da conferência A e da seção Banco de Dados que foram julgados por João ?".

JANELA DA RU :



Consulta em SQL-FÁCIL :

```
SELECT nome-artigo
WHERE  nome-conferencia = "A"
  AND  nome-secao = "Banco de Dados"
  AND  nome-julgador = "Joao"
```

Consulta em SQL :

```
SELECT artigos.titulo
FROM  julgadores, artigos, selecionam, secoes, conferencias
 WHERE conferencias.nome = "A"
    AND secoes.nome = "Banco de Dados"
    AND julgadores.nome = "Joao"
    AND julgadores.nome = selecionam.nome-julgador
    AND selecionam.codigo = artigos.codigo
    AND artigos.nome-secao = secoes.nome-secao
    AND secoes.conferencia = conferencias.nome
```

correspondendo ao seguinte esquema relacional : Julgadores (nome, pais); Selecionam (nome, codigo); Artigos (codigo, titulo, nome-secao, nome-conferencia); Conferencias (nome, pais) e Secoes (nome-secao, nome-conferencia, presidente).

COMENTÁRIO :

O pré-processador da interface SQL-FÁCIL (detalhado no próximo capítulo) supõe a necessidade do menor número de junções para responder uma consulta. Devido a essa suposição, criamos as matrizes de distâncias mínimas e caminhos mínimos (detalhadas nas seções 4.2.3 e 4.2.4 do próximo capítulo).

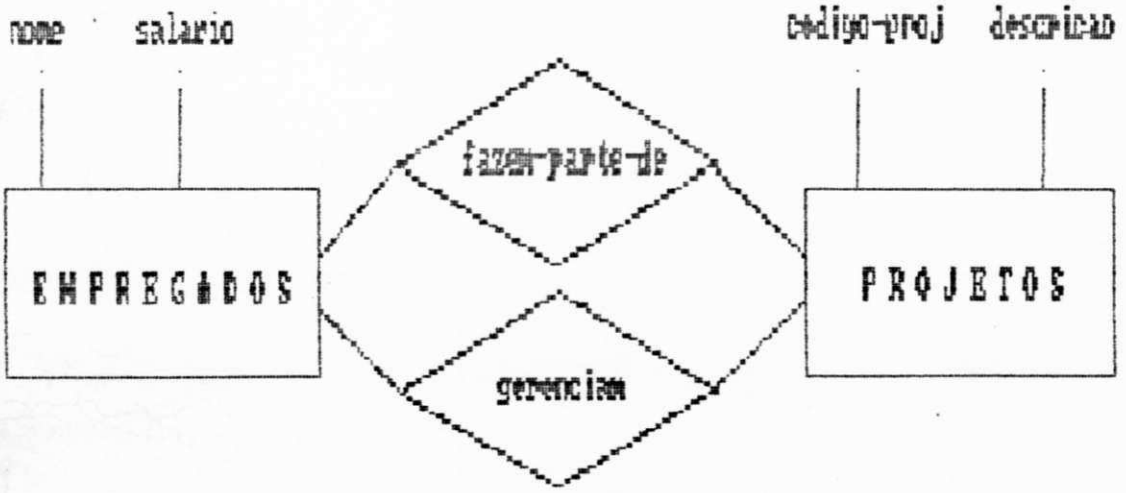
Neste exemplo ressaltamos o problema da ambiguidade entre caminhos mínimos. Há três caminhos mínimos (cada um com quatro junções naturais) que satisfazem a consulta. No primeiro, faríamos as junções das relações **Julgadores, Selecionam, Artigos, Secoes e Conferencias**, nesta ordem. Logo, resultaria nos nomes de artigos selecionados por João e agrupados pela seção Banco de Dados da conferência A. Isto é notório pela junção natural das relações **Artigos** e **Secoes**. No segundo caminho mínimo, faríamos as junções das relações **Julgadores, Selecionam, Artigos, Conferencias e Secoes**, nesta ordem. Logo, resultaria nos nomes de artigos selecionados por Joao e enviados para a conferência A cuja seção é Banco de Dados. Isto é notório pela junção das relações **Artigos** e **Conferencias**. No terceiro, e último, caminho mínimo, faríamos as junções das relaçoes **Julgadores, Selecionam, Artigos e Secoes**, nesta ordem, e a junção das relações **Artigos e Conferencias**. Logo, resultaria nos nomes de artigos selecionados por João e enviados para a conferência A e nos nomes de artigos selecionados por João e agrupados na seção Banco de Dados. Portanto, temos três significados (ambiguidade) para esta

consulta tal qual foi formulada. Como o usuário não especificou nenhum nome de relacionamento na cláusula USING, o pré-processador escolhe qualquer uma das três respostas que satisfazem a esta consulta. Para obter os nomes de artigos enviados para a conferência A, o usuário deve especificar o nome de relacionamento **enviados** na cláusula USING. Para obter os nomes de artigos agrupados na seção Banco de Dados, o usuário deve especificar o nome de relacionamento **agrupados** na cláusula USING. Para obter os nomes de artigos enviados para a conferência A e os nomes de artigos agrupados pela seção Banco de Dados, o usuário deve especificar os nomes dos relacionamentos enviados e agrupados na cláusula USING.

Neste exemplo, também ressaltamos a simplicidade na formulação da consulta. Se esta consulta fosse em SQL, o usuário além de informar o que de fato lhe é necessário, teria que formular as quatro junções naturais e especificar o nome de todas as relações envolvidas na consulta o que, convenhamos, não é nada amigável. Também correria o risco de errar a tão extensa formulação da consulta. Observemos o processo de navegação lógica que seria feito pelo usuário da SQL e a facilidade de expressão dada ao usuário da SQL-FÁCIL.

Exemplo-2 : "Quais os nomes de empregados que fazem parte do projeto CNPq ?"

JANELA DA RU :



Consulta em SQL-FÁCIL :

```
SELECT nome
WHERE  descricao = "CNPq"
USING  fazem-parte-de
```

Consulta em SQL :

```
SELECT Empregados.nome
FROM   Empregados, fazem-parte-de, Projetos
WHERE  Projetos.nome = "CNPq"
  AND  Empregados.codigo = fazem-parte-de.cod-e
  AND  fazem-parte-de.cod-p = Projetos.projeto
```

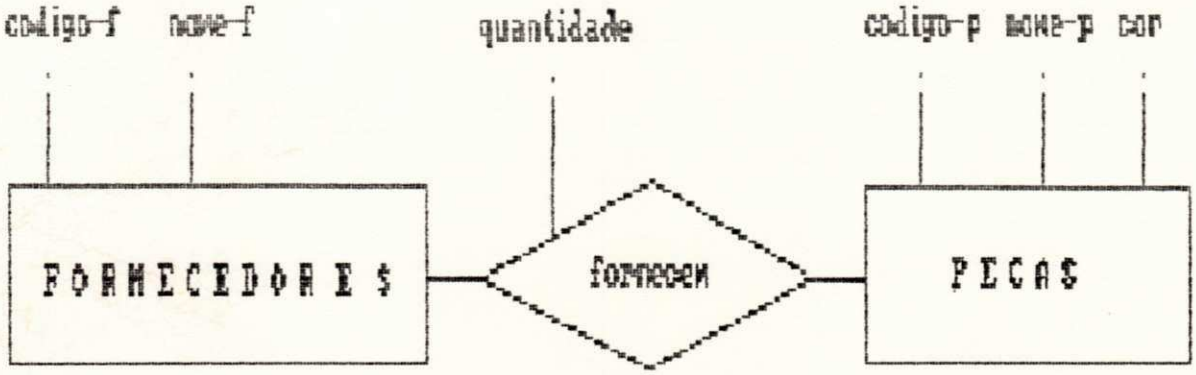
correspondendo ao seguinte esquema relacional : Empregados (codigo, nome, endereco, salario, projeto); fazem-parte-de (cod-e, cod-p); e Projetos (projeto, nome, local, gerente).

COMENTÁRIO :

Este é mais um exemplo que trata do problema da ambiguidade. Há dois relacionamentos entre os objetos Empregados e Projetos. O usuário escolheu, baseado numa prévia observação de sua janela da RU, o relacionamento **fazem-parte-de** que de fato expressa a semântica da consulta. Caso nenhum dos relacionamentos fosse escolhido pelo usuário, o pré-processador selecionaria qualquer um deles, podendo assim expressar um significado não desejado pelo usuário (se o relacionamento selecionado fosse **gerenciam**).

Exemplo-3 : "Quais os nomes de fornecedores que atualmente estão fornecendo peças ?".

JANELA DA RU :



Consulta em SQL-FÁCIL :

```
SELECT nome-f
USING fornecem
```

Consulta em SQL :

```
SELECT Fornecedores.nome-fornecedor
FROM   Fornecedores, Fornecem, Pecas
WHERE  Fornecedores.codigo = Fornecem.cod-f
  AND      Pecas.codigo = Fornecem.cod-p
```

COMENTÁRIO :

Este exemplo mostra a necessidade da cláusula USING para expressarmos a semântica correta da consulta para a qual não há ambiguidade. Caso o usuário não especificasse o relacionamento fornecem, a semântica seria "Quais os nomes de fornecedores ?", a qual não seria a pretendida pelo usuário, pois resultaria em todos os fornecedores cadastrados no BD, dentre os quais aqueles que não fornecem peças atualmente.

A consulta em SQL obtém a resposta correta, porém de uma forma menos eficiente uma vez que o pré-processador gera uma junção desnecessária (Pecas.codigo = Fornecem.cod-p) aumentando assim o tempo de resposta. Isto acontece porque quando um relacionamento (expresso na cláusula USING) envolve duas junções, ambas são utilizadas pelo pré-processador.

Exemplo-4 : "Quais os nomes das peças de cor azul fornecidas pelo fornecedor de código 99 ou código 88 ?".

JANELA DA RU : É a mesma do exemplo anterior.

Consulta em SQL-FÁCIL :

```
SELECT nome-p
WHERE  cor = "azul"
  AND  (codigo-f = "99" OR codigo-f = "88")
```

Consulta em SQL :

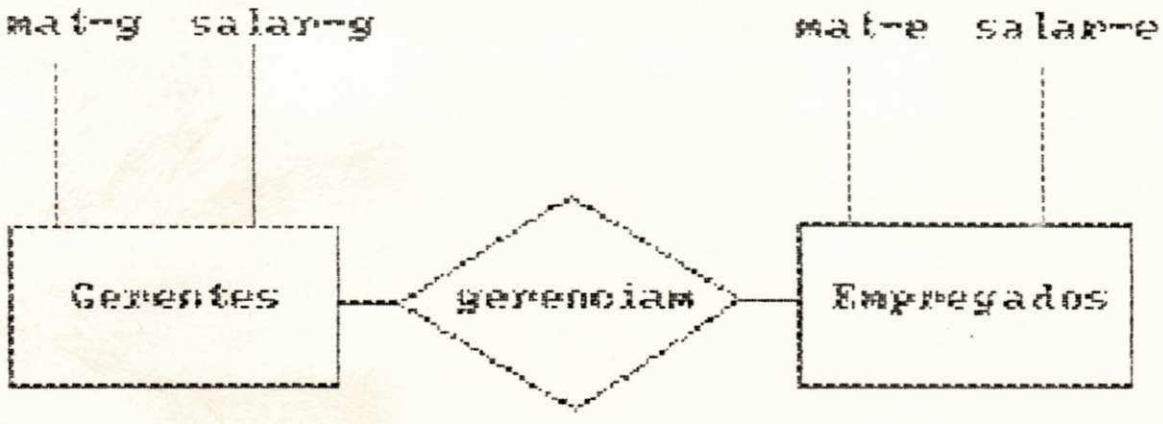
```
SELECT Pecas.nome-peca
FROM   Fornecedores, Fornecem, Pecas
WHERE  Pecas.cor = "azul"
  AND  (Fornecedores.codigo = "99"
   OR   Fornecedores.codigo = "88")
  AND   Fornecedores.codigo = Fornecem.cod-f
  AND   Peças.codigo = Fornecem.cod-p
```

COMENTÁRIO :

Este exemplo destaca o uso dos parênteses na precedência de operações e o uso do operador booleano OR.

Exemplo-5 : "Quais os empregados que ganham mais do que seus gerentes ?".

JANELA DA RU :



Consulta em SQL-FÁCIL :

```
SELECT mat-e
WHERE  salar-e > salar-g
```

Consulta em SQL :

```
SELECT Y.matricula
FROM   Empregados X, Empregados Y
WHERE  X.matricula = Y.gerente
  AND  Y.salario > X.salario
```

lembrando que o BD relacional possui apenas a relação Empregados (matricula, nome, salario, gerente) e por isso é necessário usar as variáveis-tupla X e Y.

4. O PRÉ-PROCESSADOR DA INTERFACE SQL-FÁCIL

4.1. Operacionalização do pré-processador.

O pré-processador da interface SQL-FÁCIL é um frontend para um Gerenciador de Banco de Dados Relacional (GBDR) que processe consultas em SQL como, por exemplo, o ORACLE da Oracle Corporation [ORACLE 84] e o DB2 da IBM [DATE 86]. Seu objetivo é transformar uma consulta expressa na interface SQL-FÁCIL numa consulta correspondente em SQL.

O pré-processador calcula a menor distância entre todos os pares de relações do BD e calcula também as relações intermediárias do caminho mínimo entre todos os pares de relações do BD. A menor distância entre as relações referenciadas numa consulta corresponde ao menor número de junções necessárias para responder esta consulta. Assumimos a necessidade do menor número de junções, objetivando eficiência no processamento da consulta gerada (quanto menor o número de junções, menor será o tempo de resposta).

A linguagem usada na implementação do pré-processador é o Turbo Pascal 4.0 [PASCAL 87]. Entre as linguagens apropriadas para a implementação (Pascal, C, PL/1, etc.) do pré-processador da interface SQL-FÁCIL escolhemos a linguagem de programação

Pascal pelo fato de termos um maior domínio desta linguagem e consequentemente economia do tempo que seria empregado no aprendizado de outra linguagem.

Uma visão geral do funcionamento do pré-processador é dada pela figura 4.

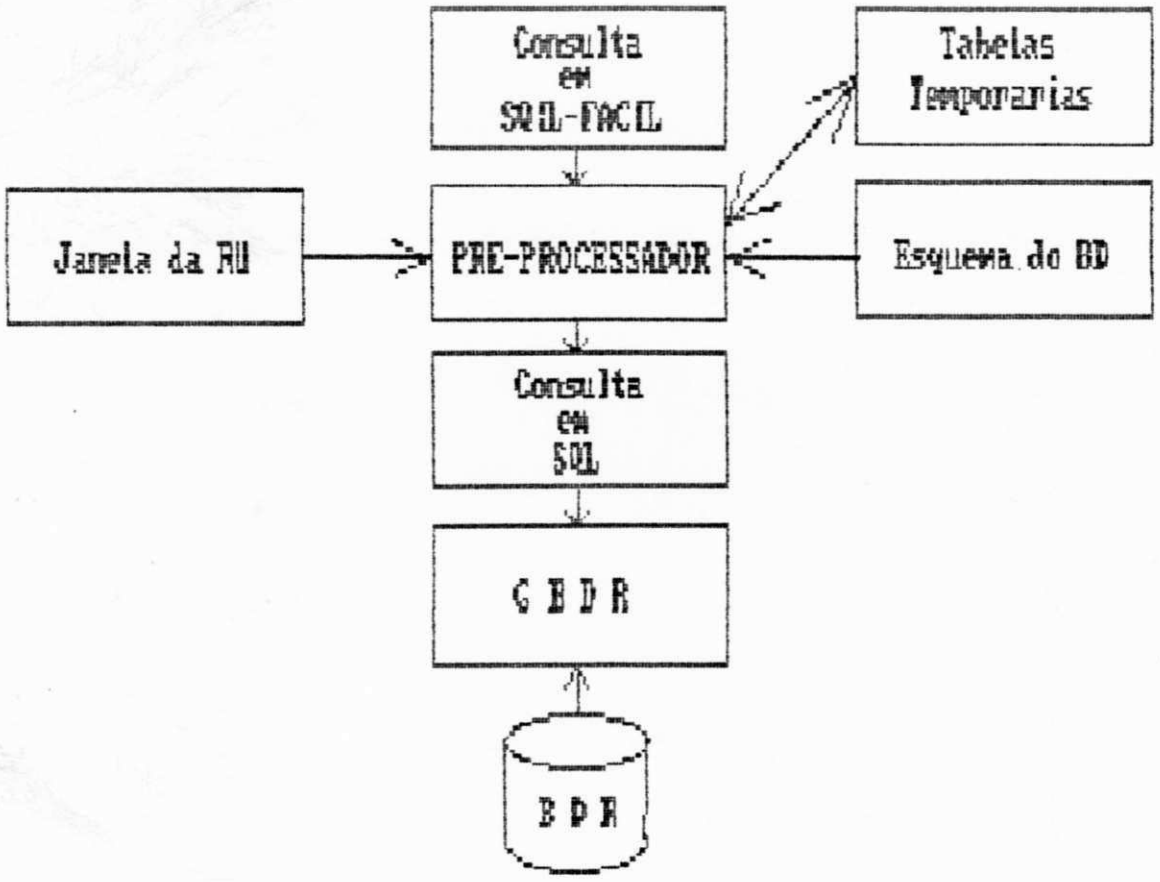


Figura 4 - Funcionamento do pré-processador da interface SQL-FÁCIL.

4.2. Principais estruturas de dados implementadas.

4.2.1. Tabelas auxiliares.

Estas tabelas definem o esquema do BD e as janelas da RU, além de características particulares de cada consulta. As tabelas auxiliares foram concebidas objetivando eficiência no processamento de informações pelo pré-processador (é mais eficiente a manipulação de endereços de tabelas do que de seus conteúdos). A tabela da relação universal, a tabela de relacionamentos I e a tabela de relações são criadas e mantidas pelo projetista ou administrador do BD utilizando-se qualquer editor de textos. A tabela de relacionamentos II e as tabelas de constantes são criadas pelo pré-processador e atualizadas a cada nova consulta.

- Tabela da Relação Universal.

Esta tabela contém a semântica da RU. É usada na crítica dos atributos universais especificados na consulta. Na verdade, a tabela da relação universal poderia ser vista como a incorporação de duas tabelas : a tabela de atributos da relação universal (vetor) e a tabela que define o esquema do BD, a qual conteria os nomes das relações e os nomes dos atributos do BD. A seguir, mostramos a tabela da relação universal construída para o exemplo-3 do capítulo anterior.

atributoU	relacao	atributoBD
codigo-f	fornecedores	codigo
nome-f	fornecedores	nome
	fornecem	cod-f
	fornecem	cod-p
quantidade	fornecem	quantidade
codigo-p	pecas	codigo
nome-p	pecas	nome
cor	pecas	cor

O campo **atributoU** contém um nome de atributo universal. Quando o campo atributoU estiver vazio, significa que o atributoBD faz parte da chave de um relacionamento (o campo atributoU não aparece na janela da RU do usuário).

O campo **relacao** contém o nome de uma relação do banco de dados que é referenciada pelo atributoU.

O campo **atributoBD** contém o nome de um atributo do banco de dados pertencente à relação referenciada no campo relação.

Todos os campos ocupam 20 posições na tabela.

- **Tabela de relacionamentos I.**

Esta tabela contém os nomes dos relacionamentos do BD, os atributos da junção natural e as duas respectivas relações envolvidas em cada relacionamento. É usada na crítica à ⟨lista-de-relacionamento⟩ da cláusula USING, na criação da matriz de distâncias mínimas (com listas de atributos da junção - detalhadas na seção 4.2.3) e na criação da tabela de relacionamentos II (definida na próxima sub-seção).

A tabela de relacionamentos I é representada pela janela da RU da figura 4. A tabela da relação universal é representada pelo esquema do BD da figura 4 . A seguir mostramos a tabela de relacionamentos construída para o exemplo-3 do capítulo anterior.

rel1	rel2	atr1	atr2	rela
fornecedores	fornecem	codigo	cod-f	fornecem
fornecem	pecas	cod-p	codigo	fornecem

Os campos **rel1** e **rel2** contêm os nomes das duas relações envolvidas no relacionamento.

Os campos **atr1** e **atr2** contêm os nomes dos atributos da junção natural das relações referenciadas nos campos rel1 e rel2.

O campo **rela** contém o nome do relacionamento entre as relações referenciadas nos campos rel1 e rel2.

Todos estes campos ocupam 20 posições na tabela.

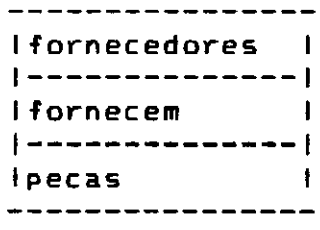
- Tabela de relacionamentos II.

Esta tabela contém os nomes dos relacionamentos do BD e os códigos das duas relações envolvidas neste relacionamento. Estes códigos correspondem aos endereços na tabela de relações. A seguir mostramos a tabela de relacionamentos II construída para o exemplo-3 do capítulo anterior.

fornecem	1	2
fornecem	2	3

- Tabela de relações.

Esta tabela contém os nomes das relações do BD. A seguir mostramos a tabela de relações construida para o exemplo-3 do capítulo anterior.



- Tabelas de constantes (inteiras, reais e literais).

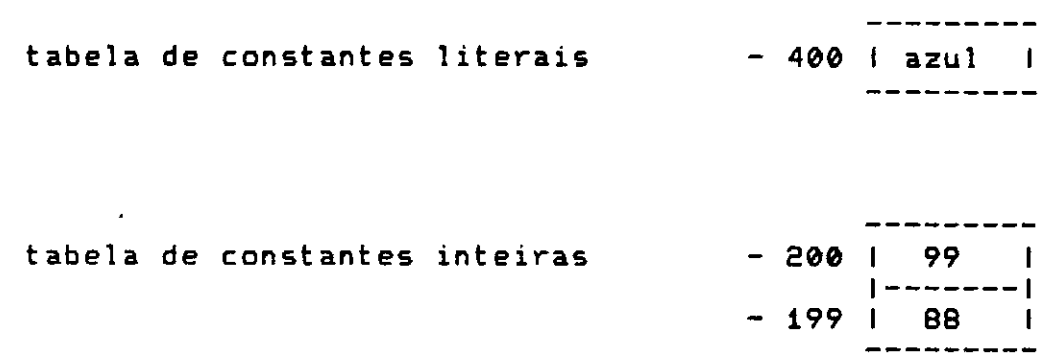
Estas tabelas são criadas a cada nova consulta e contêm as constantes especificadas na consulta. Estas tabelas são as tabelas temporárias da figura 4.

A tabela de constantes inteiras é um vetor de endereços -200 a -101.

A tabela de constantes reais é um vetor de endereços - 300 a -201.

A tabela de constantes literais é um vetor de endereços -400 a - 301.

A seguir mostramos a tabela de constantes literais e a tabela de constantes inteiras construída para o exemplo-4 do capítulo anterior.



4.2.2. Vetores.

Nestes vetores são armazenadas todas as informações estruturais da consulta em SQL que está sendo gerada.

- Vetor de relações.

É o vetor onde são armazenados, sem repetição, os códigos das relações referenciadas pelos nomes de atributos e/ou pelos nomes de relacionamentos da consulta em SQL-FÁCIL. Estes códigos correspondem aos endereços das relações na tabela de relações. Os elementos do vetor de relações (VR) são permutados objetivando uma sequência que servirá para encontrarmos o caminho mínimo envolvendo todas as relações participantes da consulta em SQL. Na seção 4.3 deste capítulo descrevemos o procedimento de permutação.

- Vetor geral.

É o vetor que contém os códigos gerados pelos analisadores léxico e sintático (comentados na seção 4.4 deste capítulo), a partir da consulta em SQL-FÁCIL, podendo conter também códigos gerados pelo pré-processador referentes às junções necessárias à consulta. Os códigos gerados pelos analisadores são os seguintes : os endereços da tabela da relação universal correspondentes aos atributos da consulta, os endereços das tabelas de constantes onde foram armazenadas as constantes da consulta, os códigos dos operadores (aritméticos, relacionais e booleanos) e dos parênteses da consulta. Caso sejam necessárias operações de junção na consulta SQL, o vetor geral conterá também os códigos gerados pelo pré-processador referentes a estas junções. Estes códigos são os seguintes : os endereços da tabela da relação universal correspondentes aos atributos da junção, o código do operador relacional de igualdade, o código do operador booleano and; e o código especial 0 (zero) que é usado para separar as cláusulas SELECT e WHERE.

As informações destes vetores são usadas para montar a consulta em SQL da seguinte maneira : As informações do vetor geral são usadas na montagem das cláusulas SELECT e WHERE, e as informações do VR na montagem da cláusula FROM.

Vejamos exemplos dos vetores construídos a partir da seguinte consulta em SQL-FÁCIL feita sobre a janela da RU do exemplo-3 do capítulo anterior :

```
SELECT nome-f
WHERE  quantidade > 10 AND quantidade < 20
```

O VR contém os elementos 1 e 2 que correspondem aos endereços da tabela de relações onde estão armazenados respectivamente as relações fornecedores e fornecem as quais são referenciadas pelos atributos **nome-f** e **quantidade**.

O vetor geral contém respectivamente os seguintes elementos :

-57 0 -56 17 -200 1 -56 15 -199 1 -55 18 -54

onde :

-57 : é o endereço do atributo universal **nome-f** na tabela da relação universal.

0 : é o código especial separador das cláusulas SELECT e WHERE.

-56 : é o endereço do atributo universal **quantidade** na tabela da relação universal.

17 : é o código do operador relacional > (maior).

-200 : é o endereço na tabela de constantes inteiras onde foi armazenada a constante inteira 10.

1 : é o código do operador booleano AND.

-56 : é o endereço do atributo universal **quantidade** na tabela da relação universal.

15 : é o código do operador relacional < (menor).

-199 : é o endereço na tabela de constantes inteiras onde foi armazenada a constante inteira 20.

1 : é o código do operador booleano AND.

-55 : é o endereço do atributo de junção **cod-f** na tabela da relação universal.

18 : é o código do operador relacional de igualdade.

-54 : é o endereço do atributo de junção codigo-f na tabela da relação universal.

Os endereços das tabelas auxiliares são propositadamente números inteiros negativos apenas para diferenciação com os códigos dos símbolos usados na interface.

A consulta em SQL gerada a partir destes vetores é a seguinte :

```
SELECT fornecedores.nome
FROM   fornecedores, fornecem
WHERE  fornecem.quantidade > 10
  AND  fornecem.quantidade < 20
  AND  fornecedores.codigo = fornecem.cod-f
```

4.2.3. Matriz de distâncias mínimas (com listas de atributos da junção).

A matriz de distâncias mínimas (D) contém as distâncias mínimas entre todos os pares de relações do BD.

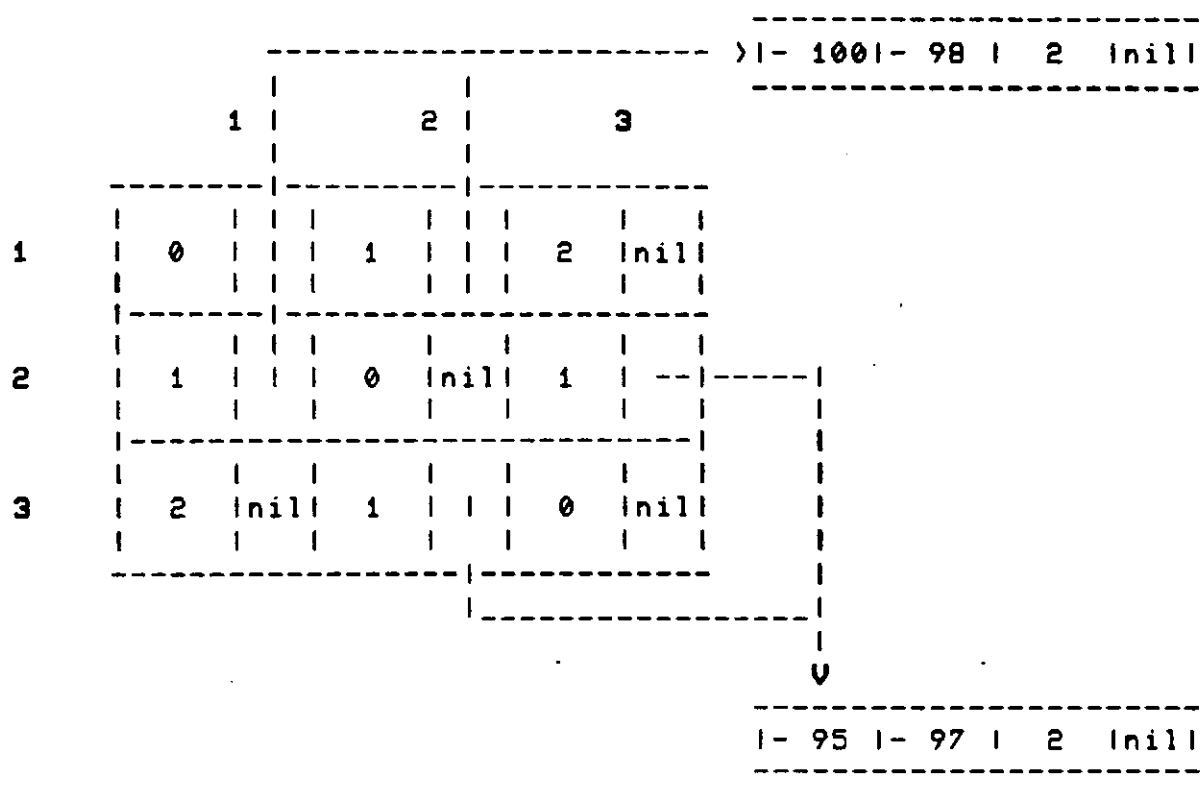
Se há uma junção natural entre duas relações, digamos i e j, assumimos que a distância tem valor 1 (um) e os endereços da tabela da relação universal (onde estão armazenados os atributos da junção das respectivas relações) são gravados numa lista apontada pelos elementos D[i,j] e D[j,i]. Cada lista contém

os endereços dos atributos da junção natural na tabela da relação universal; e o endereço do relacionamento na tabela de relacionamentos II. Quando não há junção natural entre as relações i e j, D é atualizada com valores correspondentes a soma das distâncias das relações pertencentes ao menor caminho entre i e j e, obviamente, D[i,j] e D[j,i] não apontam para nenhuma lista.

D é inicializada com 0 (zero) quando i = j e, caso contrário, com um valor inteiro representando o infinito da matemática. D é atualizada a partir da tabela de relacionamentos I.

O esquema conceitual de um BD relacional é representado pela matriz de distâncias mínimas (D) com respectivas listas simplesmente encadeadas de atributos da junção e pela matriz de caminhos mínimos (C).

Vejamos como construir D, com as listas, para as tabelas de relações, de relacionamentos II e da relação universal (definidas na seção 4.2.1 deste capítulo) :



Notemos que a distância entre as relações 1 e 2 é 1, pois D(1,2) = 1. Logo, devemos concluir que há uma junção natural entre estas relações e os elementos D(1,2) e D(2,1) apontam para a lista de atributos da junção. A distância entre as relações 1 e 3 é 2, pois D(1,2) = 1 somado com D(2,3) = 1 é 2. Donde concluímos que não há junção natural entre as relações 1 e 3; e o elemento D(1,3) não aponta para nenhuma lista.

4.2.4. Matriz de caminhos mínimos.

A matriz de caminhos mínimos (C) contém as relações intermediárias do caminho mínimo entre as relações i e j.

C é inicializada em todos os seus elementos com o valor de j e é atualizada, simultaneamente à atualização de D, com o índice da primeira relação intermediária entre i e j.

Para pesquisarmos as relações intermediárias às relações i e j, primeiro pesquisamos C(i,j) = k (significa que k é a primeira relação intermediária). A seguir, pesquisamos C(k,j) para encontrarmos a próxima relação intermediária. Este processo é repetido até acharmos C(x,j) = j. Vejamos como C é construída para o exemplo-3 do capítulo anterior.

	1	2	3
1	1	2	2
2	1	2	3
3	2	2	3

Se quisermos as relações do caminho mínimo entre as relações 1 e 3, primeiro pesquisamos C(1,3) = 2. A relação 2 é a primeira relação intermediária. Depois pesquisamos C(2,3) = 3 indicando o final do processo. Logo, concluímos que o caminho mínimo é formado pelas relações 1, 2 e 3 respectivamente.

A concepção e implementação dos algoritmos para criação e atualização de D e C foram inspiradas nas principais idéias do algoritmo de Floyd [CHISTOFIDES 75]. Este algoritmo calcula os caminhos mais curtos entre todos os pares de vértices de um grafo e foi primeiro descrito por Floyd [FLOYD 62] e depois melhorado por Murchland [MURCHLAND 65]. Alternativamente um mecanismo sugerido por Hu [HU 70] foi inserido neste algoritmo para gravar informações sobre os vértices do caminho mínimo entre dois vértices.

Assumimos que o nosso grafo, definido pela tabela de relacionamentos I, é representado por D e C.

Mostramos abaixo o algoritmo de atualização de D e C implementado no pré-processador, onde n é o número de relações do BD, k é a relação intermediária entre as relações i e j, e alfa corresponde ao valor infinito.

```
para k = 1 até n faça
   para i = 1 até n faça
      para j = 1 até n faça
         se (i é diferente de k) e
            (j é diferente de k) e
            (D[i,k] é diferente de alfa) e
            (D[k,j] é diferente de alfa)
         então
            se D[i,j] é maior do que D[i,k] + D[k,j]
            então
               D[i,j] <--- D[i,k] + D[k,j]
               C[i,j] <--- C[i,k]
```

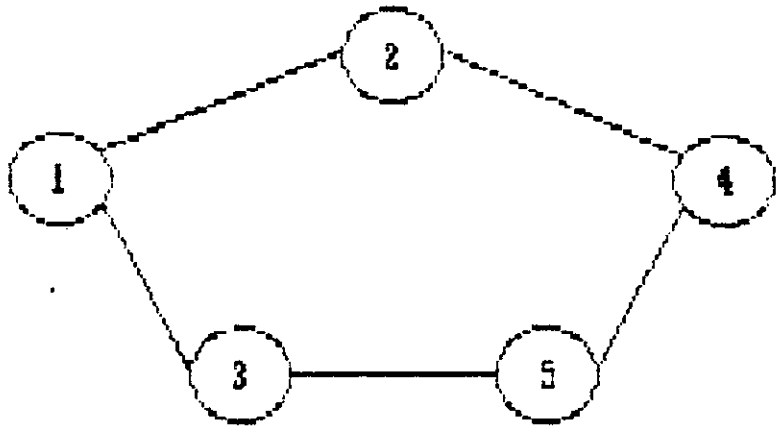
Este algoritmo é baseado numa sequência de n x n x n transformações (iterações) das matrizes D e C já inicializadas, de modo que na k-ésima iteração as matrizes representem as

distâncias mínimas e caminhos mínimos entre todos os pares de relações do BD. Notemos que o total de computações é de n x n x n adições e comparações. Ressaltamos que esta complexidade não afeta a eficiência do algoritmo uma vez que o número de relações do BD (n) geralmente não é muito grande.

4.3. Procedimento de permutação.

Este procedimento permuta os elementos do VR para encontrar a sequência que nos dá o caminho mínimo envolvendo todas as relações do VR. Esta sequência é encontrada pesquisando-se C, para cada par de elementos do VR, e ao mesmo tempo montando-se a tabela de junções. A tabela de junções é uma tabela auxiliar que contém todos os pares de relações pertencentes ao caminho mínimo. Os elementos desta tabela são exclusivos, ou seja, não se repetem. Um par de elementos do VR pode gerar vários elementos na tabela de junções, ou apenas um, caso este par possua uma junção natural. A melhor sequência será aquela que gerar menos elementos na tabela de junções (menor número de junções).

Suponhamos um VR contendo os elementos 4, 1 e 5 nesta ordem, e o BD representado pelo grafo abaixo onde os nós representam as relações e cada arco significa uma junção natural das relações envolvidas.



Tomando o primeiro par de elementos do VR (o par 4 e 1) e pesquisando C(4,1) = 2 (indicando que 2 é a primeira relação intermediária entre as relações 4 e 1), geramos o elemento 4, 2 na tabela de junções (indicando que há uma junção natural das relações 4 e 2). Depois, pesquisando C(2,1) = 1 (indicando o fim da pesquisa em C para o par 4 e 1), geramos o elemento 2, 1 na tabela de junções. A seguir, tomando o próximo par de elementos do VR (o par 1 e 5) e pesquisando C(1,5) = 3, geramos o elemento 1, 3 na tabela de junções. Agora pesquisando C(3,5) = 5, geramos o elemento 3, 5 na tabela de junções. A tabela de junções gerada contém os elementos 4, 2; 2, 1; 1, 3 e 3, 5; indicando que o caminho mínimo envolvendo os elementos 4, 1 e 5 do VR possui quatro junções naturais. Façamos agora a permutação do VR indicando a sequência 1, 4 e 5 de seus elementos. Tomando o

primeiro par de elementos do VR (o par 1 e 4) e pesquisando C(1,4) = 2, geramos o elemento 1, 2 na tabela de junções (indicando que 2 é a primeira relação intermediária entre as relações 1 e 4, havendo portanto uma junção natural das relações 1 e 2). Continuando, pesquisando C(2,4) = 4 (indicando o fim da pesquisa em C para o par 1 e 4), geramos o elemento 2, 4 na tabela de junções. A seguir, tomando o próximo par de elementos do VR (o par 4 e 5) e pesquisando C(4,5) = 5, geramos o elemento 4, na tabela de junções. A tabela de junções gerada contém os elementos 1, 2; 2, 4 e 4, 5; indicando que o caminho mínimo envolvendo os elementos 1, 4 e 5 do VR possui três junções naturais. Logo, esta última sequência do VR é melhor que a primeira, pois nos dá o menor número de junções necessárias à consulta.

Há uma situação onde temos ambiguidade entre caminhos mínimos (existe mais de uma permutação do VR que satisfaz a uma consulta). Caso o usuário tenha especificado nomes de relacionamentos da sua janela da RU na cláusula USING, será escolhida a permutação que envolva todos estes relacionamentos. Caso contrário, qualquer uma das permutações é escolhida.

As idéias básicas do algoritmo de Johnson e Trotter [EVEN 73] foram implementadas. Este algoritmo é para a geração de todas as permutações por transposições adjacentes. A diferença

entre este e o nosso é que calculamos apenas a metade das permutações, pois um VR com os elementos 5, 4 e 1 nos dá o mesmo caminho mínimo que um VR com os elementos 1, 4 e 5.

4.4. Analisadores léxico e sintático.

O Analisador Léxico reconhece e codifica 18 símbolos da interface SQL-FÁCIL.

O Analisador Sintático usado é o Analisador Sintático Descendente Recursivo [GRIES 71] que foi escolhido pela sua flexibilidade e simplicidade de implementação. A geração de código dos analisadores resulta no vetor geral.

5 - CONCLUSÕES E TRABALHOS FUTUROS

Atualmente, várias interfaces estão sendo desenvolvidas utilizando o conceito de RU. Tal conceito tenta ir um passo além do Modelo Relacional, por remover do usuário destas interfaces, não apenas a preocupação sobre a organização física dos dados, mas também sobre a organização lógica. É evidente que algumas interfaces comentadas neste trabalho substituem a preocupação sobre a organização lógica dos dados pela preocupação sobre o conhecimento de suas estruturas básicas (objetos maximais, etc.) para que seja possível consultar o BD. Entretanto, é notável que todas as consultas são mais fáceis de expressar nas interfaces que utilizam o conceito de RU do que naquelas que não o utilizam. Vale salientar que o pretendido não é a substituição do Modelo Relacional, mas o incremento do conceito de RU nas interfaces a banco de dados. Esta tem sido uma forte linha de pesquisa e desenvolvimento na área de BD.

Cremos que o objetivo principal a que nos propomos (criar uma interface amigável para acesso a BD relacional utilizando o conceito de RU) foi alcançado. Tal certeza é adquirida pelos muitos testes realizados, nos quais diversos tipos de consultas foram formulados e mapeados correta e instantaneamente. Também pela observação de que outros sistemas,

que se propõem ao mesmo objetivo, o atingem de maneira menos bem sucedida, principalmente quanto à simplicidade no tratamento de ambiguidades. Julgamos nossa abordagem, a qual utiliza-se da cláusula USING para resolver ambiguidades, bem mais simples do que a do DURST, que usa atributos característicos. Com respeito ao System/U, o mesmo impõe a utilização de variável-tupla abrangendo os objetos maximais, o que depõe contra a sua pretendida simplicidade.

Lembramos que com a interface SQL-FÁCIL nunca é preciso fazer a declaração e nem a utilização de variável-tupla, o que é uma característica importante de sua interação com o usuário. Tanto o System/U quanto o DURST não exigem a declaração de variável-tupla, mas permitem, e em certos casos até obrigam, a sua utilização pelo usuário.

Ressaltamos a infelicidade do System/U ao dar uma significação intuitiva a cada consulta. O usuário teria de refazer a consulta, caso tenha uma outra significação que não aquela dada pelo System/U, possivelmente usando variáveis-tupla e, para tanto, precisando conhecer os objetos maximais existentes. Também destacamos que o System/U não responde algumas consultas enquanto que SQL-FÁCIL sempre responde às consultas, pois o usuário é quem lhes dá significação.

É importante salientar que o poder da interface SQL-FÁCIL está vinculado à definição das janelas da RU e do esquema do BD, feitas pelo projetista do BD. Se forem bem definidas, então a interface será bem sucedida. Logo, há uma grande responsabilidade do projetista na definição da semântica do BD e consequente desempenho da interface SQL-FÁCIL, assim como nas outras interfaces comentadas neste trabalho.

Na sua versão atual, o pré-processador só gera consultas em SQL do tipo "join queries". Isto significa que o poder de expressão da SQL-FÁCIL está limitado ao poder de expressão de "join queries" expressas em SQL. Sugerimos que as próximas versões contemplem novas formas de código SQL, as quais possam utilizar funções embutidas, sub-consultas e todos os tipos de operadores e declarações existentes em SQL. Em consequência, o próprio poder de expressão da SQL-FÁCIL poderia ser melhorado, com a introdução, na sua sintaxe, de funções embutidas, do operador de pertinência de um elemento num conjunto, entre outros exemplos. Além do mais, o pré-processador poderia gerar código SQL adequado para resolver consultas que levam a comparações entre tuplas de uma mesma relação (variáveis-tupla).

Outros pontos importantes a serem tratados numa próxima versão é a criação de uma linguagem de definição e validação de janelas da RU; e a resolução do problema da junção desnecessária (visto no exemplo-3 do capítulo 3).

Consideremos também a possibilidade de criação de um procedimento genérico para acessar os catálogos de Gerenciadores de Bancos de Dados, uma vez que tais catálogos já contêm a descrição (esquema) do BD.

Quanto a melhorias na interação da interface com o usuário, sugerimos a criação de algum mecanismo para avisá-lo sobre as respostas para as quais o pré-processador resolveu uma ambiguidade não prevista pelo usuário; e também mecanismos que permitam a exibição/impressão das janelas da RU de cada usuário (teclas de função F1 e F2). Sugerimos também a exibição (opcional) da janela da RU, em paralelo com a consulta gerada, com uma indicação do caminho seguido pelo pré-processador para respondê-la. A escolha (feita no menu de atributos universais) de atributos universais da janela exibida ao invés da digitação de nomes dos atributos universais, é mais uma sugestão que certamente tornará SQL-FÁCIL mais amigável. A implementação de facilidades para a edição dos dados de entrada também é importante.

6 - REFERÊNCIAS BIBLIOGRÁFICAS

[BISKUP & BRUGUEMANN 83] - BISKUP, J. & BRUGGUEMANN, H. H. Universal Relations Views : A Pragmatic Approach, Proc. Ninth Int' I Conf. Very Large Databases, William Kaufmann, Los Altos, Calif., 1983, pp 172-185.

[CHEN 76] - CHEN, P. P. The Entity-Relationship Model : Toward A Verified View Of Data, ACM Trans. Database Syst. 1, 1 (1986). pp 9-36.

[CHRISTOFIDES 75] - CHRISTOFIDES, N. Graph Theory - An Algorithmic Approach, Academic Press, 1975. pp 163 - 165.

[CODD 70] - CODD, E. F. A Relational Model For Large Shared Data Banks, Commun. ACM 13,6, 1970. pp 377 - 387.

[DATE 86] - DATE, C. J. Relational Database : Selected Writings, Addison-Wesley Publishing Company, 1986.

[EVEN 73] - EVEN, S. Algorithmic Combinatorics, The Macmillan Company, 1983. pp 1-6.

[FLOYD 62] - FLOYD, R. W. Algorithm 97 - Shortest Path, Commun. of the ACM, 5, 1962. p. 345.

[GRIES 71] - GRIES, D. Compiler Construction For Digital Computers, John Wiley & Lons, Inc., 1971, pp 97-100.

[HU 70] - HU, T. C. Integer Programming And Network Flows, Addison-Wesley Publishing Company, second printing, 1970. pp 156-161.

[KORTH 84] - KORTH, H. F. et alii. System/U : A Database System On The Universal Relation Assumption. ACM Transactions on Database Systems, Sept. 1984, v. 9, number 3.

[MAIER 83] - Maier, D. The Theory Of Relational Databases, Computer Science Press, 1983.

[MURCHLAND 65] - MURCHLAND, J. D. A New Method For Finding All Elementary Paths In A Complete Directed Graph, London School of Economics, Report LSE - TNT - 22, 1965.

[REISS 83] - REISS, S. P. Eris : The Design And Implementation Of An Experimental Relational Information System. CS-83-02, Dept. of Computer Science, Brown Univ, Jan. 1983.

[ORACLE 84] - Manuais de Referência. Oracle Corporation. Menlo Park, Calif., 1984.

[PASCAL 87] - Turbo Pascal : Owner's Handbook. Version 4.0. Borland International, 1987. 327 p.

[ULLMAN 83] - ULLMAN, J. D. Principles of Database Systems, Computer Science Press, Rockville, 1983.

[VARDI 88] - VARDI, M. Y. The Universal-Relation Data Model For Logical Independence, IEEE Software, Mar. 1988. pp 80-85.

APÊNDICE

- Sintaxe em BNF -

```
SELECT ⟨lista-de-atributos⟩
WHERE ⟨predicado⟩
USING ⟨lista-de-relacionamentos⟩

⟨lista-de-atributos⟩ ::= atributo ⟨mais-atributos⟩

⟨mais-atributos⟩ ::= atributo ⟨mais-atributos⟩
                   | &

⟨predicado⟩ ::= ⟨expr-bool⟩

⟨expr-bool⟩ ::= ⟨expr-bool⟩ or ⟨expr-e⟩
              | ⟨expr-e⟩

⟨expr-e⟩ ::= ⟨expr-e⟩ and ⟨expr-rel⟩
           | ⟨expr-rel⟩

⟨expr-rel⟩ ::= ⟨expr⟩ ⟨oprel⟩ ⟨expr⟩

⟨oprel⟩ ::= ⟨
          | ⟩
          | ⟨=
          | ⟩=
          | =
          | ⟨⟩
```

```
⟨expr⟩ ::= ⟨expr⟩ ⟨opadicao⟩ ⟨termo⟩
         | ⟨termo⟩

⟨opadicao⟩ ::= +
             | -

⟨termo⟩ ::= ⟨termo⟩ ⟨opmult⟩ ⟨fator⟩
           | ⟨fator⟩

⟨opmult⟩ ::= *
           | /

⟨fator⟩ ::= atributo
           | ⟨constante⟩
           | "(" ⟨expr-bool⟩ ")"

⟨constante⟩ ::= inteira
              | real
              | cadeia

⟨lista-de-relacionamentos⟩ ::= relacionamento ⟨lista-de-relacionamentos
                             | &
```

OBSERVAÇÕES :

- As palavras-chave SELECT, WHERE e USING são automaticamente exibidas pelo pré-processador, nesta ordem.

- **atributo** é um atributo universal.

- **inteiro** é um número inteiro.

- **real** é um número real.

- **cadeia** é qualquer cadeia de caracteres entre aspas.

- **relacionamento** é um nome de relacionamento da janela da RU.