Uma linguagem visual de consulta a banco de dados utilizando o paradigma de fluxo de dados

Ana Paula Appel

Uma linguagem visual de consulta a banco de dados utilizando o paradigma de fluxo de dados

Ana Paula Appel

Orientador:
Prof. Dr. Caetano Traina Júnior

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC - USP, como parte dos requisitos para obtenção do título de Mestre em Ciências de Computação e Matemática Computacional.

"Versão Revisada Após Defesa"

Data da Defesa:02/04/2003

Visto do Orientador: _____

USP - São Carlos
Abril de 2003

Agradecimentos

A DEUS, por mais uma vez, ter permitido vencer uma importante etapa da minha vida!

Aos meus pais, Anna e José, pelo amor, força, dedicação e preocupação. Sem vocês nada seria possível! Vocês são tudo pra mim!

Aos meus irmãos, Ana Beatris e Antônio José e a minha querida cunhada, amiga e irmã Solimar obrigada pelo amor e pela força de vocês! O meu obrigado à vocês!

Aos meus sobrinhos Bruna, Matheu e Pedro obrigado pelo amor e momentos de alegria. Vocês são muito importante pra mim!

Ao Prof. Dr. Caetano, não só pela excelente orientação, mas por ter acreditado na minha capacidade de lutar e vencer. Obrigada pela confiança, oportunidade, paciência, ajuda nas horas de dúvidas e principalmente pela amizade.

Às minhas grandes amigas Regiane, Daniela e Veronica, que compartilharam todos os bons e maus momentos durante esses dois anos de mestrado. Obrigada pela amizade, carinho, força, paciência e compreensão de vocês!

À Prof. Dra. Agma que sempre foi uma verdeira mãe durante todo esse tempo o meu muito obrigado!

A todos os meus amigos do GBDI pela força durante este mestrado! Conviver com vocês é muito bom e importante pra mim! Humberto, Enzo e Josiel valeu pela grande ajuda nas idéias, no latex e no Builder!

A todos os amigos que conheci nesses dois anos, pela amizade e pelos bons momentos compartilhados! A Elaine o meu obrigada pela amizade, carinho e compreensão.

Aos funcionários do ICMC pelo apoio técnico e administrativo.

Ao CNPQ pelo apoio financeiro.

Como não poderia esquecer também ao meu cachorrinho, Sherlock que me fez companhia nos finais de semana de trabalho.

"Deus, dai-me a serenidade para aceitar as coisas que eu não posso mudar,

coragem para mudar as coisas que eu possa,

e sabedoria para que eu saiba a diferença:

vivendo um dia a cada vez, aproveitando um momento de cada vez;

aceitando as dificuldades como um caminho para a paz;

indagando, como fez Jesus, a este mundo pecador, não como eu teria feito;

aceitando que Você tornaria tudo correto se eu me submetesse à sua vontade

para que eu seja razoavelmente feliz nesta vida

e extremamente feliz com você para sempre no futuro."

Sumário

Lista de Figuras

Lista de Tabelas

APPEL, A. P. *Uma linguagem visual de consulta a banco de dados utilizando o paradigma de fluxo de dados.* São Carlos, 2003. 49 p. Dissertação de Mestrado - Instituto de Ciências Matemáticas e de Computação - ICMC, USP.

Resumo

Apesar de muito trabalho ter sido dispendido sobre linguagens de consulta a Sistemas de Gerenciamento de Bancos de Dados Relacionais, existem somente dois paradigmas básicos para essas linguagens, que são representados pela *Structured Query Language – SQL* e pela *Query by Example – QBE*.

Apesar dessas linguagens de consultas serem computacionalmente completas, elas tem a desvantagem de não permitir ao usuário nenhuma interação gráfica com a informação contida na base de dados. Um dos principais desenvolvimentos na área de base de dados diz respeito às ferramentas que proveêm aos usuários um entendimento simples da base de dados e uma extração amigável da informação. A linguagem descrita neste trabalho possibilita que usuários criem consultas graficamente por meio de diagramas de fluxo de dados. Além da linguagem de consulta gráfica, este trabalho mostra também a ferramenta de apoio *Data Flow Query Language* - DFQL, que é um editor/executor de consultas construído para suportar essa linguagem, através de um conjunto de operadores representados graficamente, e a execução desses diagramas, analisando a rede e gerando os comandos correspondentes em SQL para realização da consulta. Esses comandos são submetidos ao sistema de gerenciamento de banco de dados e o resultado é mostrado/gravado conforme a consulta feita.

Abstract

In spite of many works done on query languages, all existing languages are direct extensions of Structured Query Language – SQL and Query-By-Example – QBE. These two languages were developed in the beginning of the Relational Database Management Systems – RDBMS development.

Althoug these languages are computationally complete, they take the disadvantage of not supporting graphical interaction with data. One of the the main developments in the database area concerns tools to provide users a simple understand of database content, and friendly extraction of the information. The language described in this work enables users to create graphical queries using data flow diagrams.

Besides the graphical query language, this work also shows the Data Flow Query Language - DFQL tool. This tool is a query editor/executer that supports this language, using a set of operators represented graphicaly, and the diagram execution is done by analising the network and producing the respective commands in SQL to realize the query. This commands are sent to the DBMS and the result is shown/recorded according to the query.

Capítulo

1

Introdução

1.1 Motivação

Embora muito trabalho tenha sido feito na elaboração de linguagens de consulta a Sistemas de gerenciamento de Bancos de Dados - SGBD, exceto as linguagens de propósitos específicos, todas as linguagens existentes são extensões diretas de apenas duas linguagens básicas: a *Structured Query Language - SQL*, ou a *Query by Example - QBE*.

Ambas as linguagens foram desenvolvidas quase ao mesmo tempo, nos primórdios do desenvolvimento do modelo e dos sistemas relacionais, e graças a isso a linguagem SQL é o padrão de fato para o acesso a Sistemas de Gerenciamento de Banco de Dados Relacionais - SGBDRs, e a própria QBE esta presente na maioria dos produtos dos fabricantes de SGBD, como parte de ferramentas auxiliares de acesso à base. Apesar de diferenças nos diversos sistemas quanto a interface que as ferramentas em QBE apresentam para o usuário, elas seguem sempre basicamente o mesmo estilo.

Além do fato de ter se tornado o padrão para o acesso a SGBDR, tanto a linguagem SQL quanto a QBE apresentam a grande vantagem de serem linguagens declarativas, ou seja, ambas permitem ao usuário descrever o que ele quer consultar, e não o como a busca deve ser executada. Essa característica permite que o SGBDR possa analisar a consulta, e decidir "o como" de maneira a otimizar a resposta, de acordo com parâmetros do estado atual do gerenciador e dos próprios dados, tirando do programador a responsabilidade de escrever consultas "eficientes". Na realidade, desenvolvendo sempre melhores algoritmos de busca e de otimização dessas consultas, a comunidade de banco de dados foi capaz de sobrepujar problemas que, a princípio, pareciam impedir que um sistema baseado num SGBDR fosse tão eficiente quanto um sistema baseado nos modelos de dados anteriormente existentes, para tornar os SGBDRs uma excelente opção de implementação, mesmo

quando restrições de desempenho fossem severas.

A independência de como os comandos de consulta podem ser executados é conseguido pela possibilidade de representar os comandos expressos tanto em SQL quanto em QBE numa forma algébrica. A notação algébrica é fundamental para a otimização de uma consulta, pois permite que sejam geradas diversas alternativas para a execução, as quais podem ter seu custo pré-avaliado, permitindo a escolha daquela que apresenta o custo mínimo. A notação algébrica também tem a vantagem de se constituir num conjunto de operadores que atuam sempre sobre operandos de um mesmo tipo (no caso tabelas), o que confere à linguagem uma expressividade muito grande.

Um aspecto, que em determinadas aplicações pode ser negativo, é que a linguagem SQL é puramente textual, e não existe uma maneira natural de estendê-la para permitir consultas envolvendo dados que não tenham uma descrição textual adequada. Por exemplo, consultas que envolvam referência a imagens somente podem ser escritas representando-se uma imagem através de um texto que as identifique, como por exemplo, o nome de um arquivo do sistema operacional que as armazena. Embora isso seja perfeitamente factível e adequado a uma linguagem de comando embutida entre um aplicativo e o SGBD, a quebra do paradigma da linguagem para uso como uma linguagem de consulta casual envolvendo usuários humanos é no mínimo desconfortável. Numa representação gráfica, a integração com elementos gráficos numa linguagem de consulta envolvendo imagens seria mais natural. Da mesma maneira, diagramas de fluxo de dados são bastante utilizados para a representação de processos em geral, em vários campos de conhecimento. As atividades de *data mining* e processamento analítico de dados (*On Line Analitical Processing* - OLAP) são também consideradas processos, e, portanto também poderiam se beneficiar da possibilidade de notação em fluxo de dados, incluindo-se aí símbolos para representar cada uma de suas tarefas, tais como classificação, agrupamento (*clustering*), projeções, etc.

Este projeto visa criar uma nova forma de representação de consultas a SGBDRs, baseada na noção de fluxo de dados, com uma interface interativa, que represente graficamente comandos de consulta como diagramas de fluxo de dados. Essa representação irá se constituir em uma terceira forma de expressão de comandos relacionais, que mantém o mesmo poder de representação das linguagens SQL e QBE, uma vez que também pode, igualmente, ser convertido para a mesma representação algébrica. No entanto, por ter uma expressão gráfica, deverá ser mais adequada para representar consultas que envolvem elementos gráficos.

1.2 Objetivos do Projeto

Uma das caracteristicas mais importantes de um SGBD é que diversos tipos de usuários, experientes ou não, são capacitados a recuperar informações da base de dados. Esta facilidade de acesso ao dados requer uma grande quantidade de trabalho antes dos usuários

finais fazerem suas consultas.

Os usuários finais desse projeto também podem ser chamados de usuários casuais por terem alguma facilidade com a linguagem SQL, que é o padrão para linguagem de consulta em um banco de dados relacional. Podemos então caracterizar o usuário casual de um SGBD como:

1. Interage com o SGBD ocasionalmente;
2. As consultas que faz não são repetitivas;
3. Pode experimentar a consulta, até obter o resultado desejado;
4. Embora conheça os conceitos da linguagem, não tem fluência em construções elaboradas da linguagem.

Uma forma de ajudar essa classe de usuários a construir consultas mais eficientes, é oferecer um ambiente que permite ao usuário construir suas consultas de uma forma mais intuitiva.

Assim esse projeto tem por objetivo gerar a definição de uma linguagem de consulta a bancos de dados relacionais utilizando o paradigma de fluxo de dados, e implementar uma ferramenta que permita ao usuário a criação de diagramas de consulta e a execução das mesmas.

A capacidade de representação seja através de fluxo de dados, de SQL ou QBE, deve ser idêntica (provido que operadores adequados sejam sempre disponibilizados), pois qualquer consulta expressa numa das formas pode ser mapeada para as demais. Com isso pretende-se ter uma nova forma de acesso a bases de dados relacionais: na realidade uma forma tão flexível quanto as tradicionais linguagens SQL e QBE, ao mesmo tempo em que permite uma integração mais natural entre gerenciadores relacionais e ambientes que incluem a manipulação de imagens, e/ou incluem processos de análise de dados, como processos de *data mining e OLAP*.

1.3 Organização do Trabalho

Para que o trabalho possa ser melhor compreendido é necessário apresentar uma visão geral dos assuntos a ele relacionados. Assim, este trabalho está organizado da seguinte maneira.

O capítulo 2 explora os conceitos teóricos sobre Álgebra Relacional.

O capítulo 3 é uma abordagem relativa ao estado da arte das Linguagens de Consulta.

O capítulo 4 apresenta como representar uma consulta através de Fluxo de Dados.

O capítulo 5 apresenta a ferramenta desenvolvida para se criar uma consulta baseado no paradigma de Fluxo de Dados.

O capítulo 6 apresenta as conclusões e as contribuições deste trabalho.

Capítulo

2

Conceitos

Neste capítulo são apresentados a conceituação teórica sobre álgebra relacional, seus operadores, transformações e árvores de comando.

2.1 Introdução

O modelo relacional foi concebido com um forte embasamento na álgebra relacional. Isso permitiu que poderosas ferramentas de otimização levassem à construção de ferramentas de recuperação de dados muito eficientes. Os principais conceitos da álgebra relacional são a coleção de operadores definidos, e o conjunto de elementos em que a álgebra se apoia, os quais serão sempre relações.

2.2 Operadores

A álgebra relacional consiste em um conjunto de operações sobre um conjunto de relações as quais são obtidas pelo produto cartesiano de domínios de valores de atributos. Cada operando tem como entrada uma ou duas relações e produz como resultado uma nova relação.

As operações da álgebra relacional são divididas em dois grupos. Um grupo de operadores teóricos fazem uso do fato das tabelas serem essencialmente um conjunto de colunas. Os operadores desse grupo são chamados operadores de conjuntos, e são: **união, interseção, diferença e produto cartesiano**. Vale ressaltar que para que duas relações possam ser operadas por uma operação sobre conjunto, é necessário que ambas sejam "Compatíveis em Domínio". Isto acontece quando além de ter o mesmo número de atributos, cada par de atributos correspondentes tem

o mesmo domínio. O segundo grupo consiste em operações desenvolvidas especificamente para bancos de dados relacionais e são chamados operadores relacionais; estes incluem as operações relacionais unárias (que envolve apenas uma tabela) **seleção e projeção**, e as operações relacionais binárias (que envolvem duas tabelas) **junção e divisão**. Esses operadores são descritos nas subseções a seguir[Date, 2000] [Elmasri and Navathe, 1999][Korth and Silberschatz, 1999][O'Neil and O'Neil, 2001].

1. *Produto Cartesiano:*

   O operador Produto Cartesiano retorna uma relação contendo todas as tuplas possíveis que são uma combinação de duas tuplas, uma tupla pertencente a cada uma das duas relações especificadas. O operador Produto Cartesiano é representado por "TIMES" ou $\times$.

   R TIMES S

   $R \times S$

2. *União:*

   Retorna uma relação contendo todas as tuplas que pertencem a ambas ou cada uma de duas relações especificadas, porém tuplas que estão em ambas aparecem apenas uma vez. O operador de união é representado pela palavra "UNION" ou $\cup$.

   R UNION S

   $R \cup S$

3. *Interseção:*

   Retorna uma relação contendo todas as tuplas que pertencem às duas relações especificadas ao mesmo tempo. O operador de interseção é representado pela palavra "INTERSECT" ou $\cap$.

   R INTERSECT S

   $R \cap S$

4. *Diferença:*

   Retorna uma relação contendo todas as tuplas que pertencem a primeira e não à segunda entre duas relações especificadas. O operador de divisão é representado por "MINUS" ou $-$.

   R MINUS S

   $R - S$

5. *Seleção (Select - σ)*

   Retorna uma relação contendo todas as tuplas de uma dada relação que satisfazem a uma condição especificada. A seleção é denotada por:

S WHERE C ou $\sigma_{(c)}S$

Onde S é uma tabela e c é a condição de seleção.

A condição de seleção pode envolver a comparação entre dois atributos ou entre um atributo e uma constante. Em geral podem ser usadas comparações do tipo $=$, $\leqslant$, $<$, $>$ e $\geqslant$ na condição de seleção. Pode-se também combinar várias tipos de comparações em uma mesma condição de seleção através de conectores AND, OR e NOT.

6. *Projeção(Project - π)*

   Retorna uma relação contendo todas as tuplas que permanecem em uma relação após terem sido removidos os atributos especificados. A projeção é denotada por:

   $R\,[Ai1, ..., Aik]$ ou $\pi_{(Ai1,...,Aik)}R$

   Onde R é a relação e $Ai1...Aik$ são os atributos a serem selecionados.

7. *Divisão:*

   O operador de divisão toma duas relações unárias e uma relação binária e retorna uma relação contendo todas as tuplas de uma única relação unária que aparecem na relação binária, coincidindo com todas as tuplas de outra relação unária. A divisão é denotada por DIVEDEBY OU $\div$. Sendo R e S duas relações, a divisão pode ser representada como:

   R DIVEDEBY S

   $R \div S$

8. *Junção (Join - $\boxtimes$)*

   Existem diversas variações sobre a operação de Junção, todas elas definidas em razão de sua ampla utilização, e levando-se em conta que se refinando sua definição em operadores específicos, cada um viabiliza uma implementação mais eficiente do que seria possível pela execução dos operadores elementares que teoricamente os implementam.

   Cada operador de junção tem uma definição própria de como são tratados os atributos envolvidos na comparação, e como são tratadas tuplas onde os atributos envolvidos na comparação têm valor nulo.

   - *Junção Externa (Outer Join)*

     O operador de junção externa é uma extensão do operador de junção para tratar informações omitidas. Este operador pode ser usado para evitar perda de informações. Existem de fato, três formas de usar este operador: junção externa à esquerda, junção externa à direita e junção externa total. Todas os três computam a junção e adicionam tuplas extras a seu resultado.

- *Junção Externa à Esquerda:*
  toma todas as tuplas da relação à esquerda que não encontraram par entre as tuplas da relação à direita, preenche a tupla com valores nulos para todos os outros atributos da relação à direita e a adiciona ao resultado da junção natural. Todas as informações da relação à esquerda são apresentadas no resultado da junção externa à esquerda. A junção externa à esquerda é denotada por:
  $R \boxtimes_c S$
  R LEFT OUTER JOIN S WHERE C
- *Junção Externa à Direita:*
  simétrica à junção externa à esquerda. As tuplas da relação à direita que não encontram par na relação da esquerda são preenchidas com nulos e adicionadas ao resultado da junção natural. Assim, todas as informações da relação da direita estão presentes no resultado da junção externa à direita. A junção externa à direita é denotada por:
  $R \boxtimes_c S$
  R RIGHT OUTER JOIN S WHERE C
- *Junção Externa Total:*
  faz ambas as operações, preenche as tuplas da relação da esquerda que não encontraram par na relação da direita, assim como preenche as tuplas da relação da direita que não encontraram par na relação da esquerda, adicionando-as ao resultado da junção. A junção externa total é denotada por:
  $R \bowtie_c S$
  R OUTER JOIN S WHERE C

- *Junção Teta (θ-Join)*

  Um operador de junção denominado Junção - Teta, onde pode ser usado qualquer operador θ válido no domínio dos atributos comparados, é uma das mais genéricas. No Operador Junção-Teta, os atributos envolvidos na comparação aparecem ambos na relação resultado, e tuplas com valores nulos nos atributos envolvidos na comparação não aparecem no resultado. A junção - Teta é representada pela palavra JN ou $\boxtimes$. Considerando que R e S são as relações, a junção-θ pode ser representada como:

  $R \boxtimes_c S$

  R THETA JOIN S WHERE C

- *Equi-Junção (Equi-Join)*

  É comum o operador θ de comparação ser a igualdade, o que é interessante, por simplificar o algoritmo de comparação. Portanto é interessante dispor de um operador de junção equivalente à Junção-Teta que compare todos os atributos

envolvidos com o operador igual (=). Esse operador é chamado Equi-Junção (ou Equi-Join).

Como no Operador Junção-Teta, os atributos envolvidos na comparação aparecem ambos na relação resultado, resultando em pares de atributos com valores idênticos na relação resultado. Tuplas com valores nulos nos atributos envolvidos na comparação não aparecem no resultado.

A Equi - Junção é representada pela palavra EQUI-JOIN ou $\boxtimes$. Considerando que R e S são as relações, a Equi - Junção pode ser representada como:

R EQUI-JOIN S WHERE $< atribR >=< atribS >$

$R \boxtimes_{<AtribR>=<AtribS>}$ S

- *Junção Natural (Natural Join)*

  Como os atributos comparados num operador de Equi-Junção aparecem em pares com valores idênticos na relação resultado, um de cada par pode ser eliminado. A operação de Junção Natural é semelhante à operação de Equi-Junção, porém dos atributos comparados (pela operação de igualdade), apenas os originários de uma das relações operadas aparecem na relação resultado.

  No operador de Junção Natural $R * S$, os atributos envolvidos na comparação aparecem apenas os originários da relação R na relação resultado. Tuplas com valores nulos nos atributos envolvidos na comparação também não aparecem no resultado.

  A junção natural é representada pela palavra JOIN ou $*$. Considerando que R e S são as relações, a junção natural pode ser representada como:

  R JOIN S WHERE $< atribR >=< atribS >$

  $R *_{<AtribR>=<AtribS>} S$

2.3 Compilações de Consultas

A primeira ação que o sistema tem de executar ao responder uma consulta é traduzir a consulta para sua forma interna, que para sistemas de banco de dados relacionais é baseada na álgebra relacional. No processo de geração da forma interna da consulta, o analisador sintático verifica a sintaxe da consulta do usuário e verifica se os nomes das relações que aparecem na consulta são nomes de relações no banco de dados. Uma representação da consulta em uma árvore sintática é gerada, e então traduzida para uma expressão algébrica relacional.

Existem muitas regras para transformações de operações em álgebra relacional em equivalentes. Por isso, se duas representações têm o mesmo grupo de atributos em ordem diferente, mas as duas relações representam as mesmas informações, a representação é considerada equivalente. Cada representação gerada é chamada de plano de avaliação. Planos alternativos para cada expressão podem ser gerados por meio de regras semelhantes

e o plano com a execução mais barata entre todos é o escolhido. Algumas propriedades utilizadas para a geração de alternativas são exemplificadas abaixo:

1. Cascata de seleção (σ):

   Uma condição de seleção conjuntiva pode ser quebrada em uma cascata(seqüência) de operações s individuais:

   $\sigma_{(c1ANDc2AND...ANDcn)}(R) \equiv \sigma_{(c1)}(\sigma_{(c2)}(...(\sigma_{(cn)}(R)...))$

2. Operações de seleção são comutativas:

   $\sigma_{(c1)}(\sigma_{(c2)}(R)) \equiv \sigma_{(c2)}(\sigma_{(c1)}(R))$

3. Apenas as operações finais em uma seqüência de operações de projeção são necessárias, as outras podem ser omitidas. Essas transformações podem ser chamada de cascata de π:

   $\pi_{(List1)}(\pi_{(List2)}(...(\pi_{(Listn)}(R)...)) \equiv \pi_{(list1)}(R)$

4. Comutativo de σ com π:

   Se a condição de seleção envolve apenas os atributos $A1, ..., An$ em uma lista de projeções, os dois operadores podem ser comutados.

   $\pi_{(A1,A2,...,An)}(\sigma_{(c)}(R)) \equiv \sigma_{(c)}(\pi_{(A1,A2,...,An)}(R))$

5. Comutativa de $\times$ (e $\boxtimes$):

   Os operadores $\boxtimes$e $\times$ são comutativos:

   $R \boxtimes_c S = S \boxtimes_c R$

   $R \times S = S \times R$

   Note que a ordem dos atributos pode não ser a mesma na relação resultante dos dois *joins*, o significado é o mesmo por que a ordem dos atributos não é importante na definição alternativa da representação que é usada aqui. O operador de junção natural é simplesmente um caso especial do operador de junção teta; conseqüentemente a junção natural também é comutativa.

6. Comutativa de σ com $\boxtimes$(ou $\times$):

   Se todos os atributos da condição de seleção c envolve apenas os atributos de uma das relações que esta sendo "juntada", as duas operações podem ser comutadas como a seguir:

   $\sigma_c(R \boxtimes S) \equiv (\sigma_c(R)) \boxtimes S)$

   Alternativamente, se a condição de seleção c pode ser escrita como ($c1$ e $c2$), onde a condição $c1$ envolve apenas os atributos de R e a condição $c2$ envolve apenas os atributos de S, a operação de comutatividade é a seguinte:

   $\sigma_c(R \boxtimes S) \equiv (\sigma_{c1}(R)) \boxtimes (\sigma_{c2}(S))$

A mesma regra é aplicada se o operador ⊠for substituido pelo operador $\times$.

7. Comutativa de π com ⊠(ou $\times$):

   Supondo que a lista de atributos da projeção é $L = \{A1, ..., An, B1, ..., Bm\}$, onde $A1, ..., An$ são atributos de R e $B1, ..., Bm$ são atributos de S. Se a condição de junção c envolve apenas atributos em L, as duas operações podem ser comutadas da seguinte forma:

   $\pi_L(R \boxtimes_c S) \equiv (\pi_{(A2,...,An)}(R)) \boxtimes_c (\pi_{(B1,...,Bm)}(S))$

   Se a condição de junção c contém atributos adicionais que não estão em L, estes devem ser adicionados à lista de projeção, e uma projeção final é necessária. Por exemplo, se os atributos $L_R = \{An + 1, ..., An + k\}$ de R e $L_S = Bm + 1, ..., Bm + p$ de S são envolvidos na condição de junção c, mas não na lista de projeção L, a operação comutativa fica da seguinte forma:

   $\pi_L(R \boxtimes_c S) \equiv \pi_L((\pi_{(A1,...,An,An+1,...,An+k)}(R)) \boxtimes_c (\pi_{(B1,...,Bm,Bm+1,...,Bm+p)}(S)))$

   Para $\times$, nunca é expressa nenhuma condição c, então a primeira regra de transformação sempre aplica a substituição $\boxtimes_c$ com $\times$.

8. Comutativa de operadores de conjunto:

   Os operadores de conjunto $\cup$ e $\cap$ são comutativos mas o operador "$-$" não é.

9. Associatividade de ⊠, $\times$, $\cup$ e $\cap$:

   Estes quatro operadores são individualmente associativos, isto é sendo θ qualquer um destes quatro operadores por toda expressão, tem-se:

   $(R\,\theta\,S)\,\theta\,T \;\equiv\; R\,\theta\,(S\,\theta\,T)$

10. Comutativa de σ com operadores de conjunto:

    A operação σ é comutativa com as operações $\cup$, $\cap$ e $-$. Sendo θ qualquer uma dessas três últimas operações, têm-se:

    $\sigma_c(R\,\theta\,S) \equiv (\sigma_c(R))\,\theta\,(\sigma_c(S))$

11. A operação π é comutativa com $\cup$:

    $\pi_L\,(R \cup S) \equiv (\pi_L(R)) \cup (\pi_L(S))$

12. Convertendo uma sequência de σ, $\times$ em ⊠:

    Se a condição c de uma operação σ que é seguida de um $\times$ corresponde a uma condição de *join*, converter a sequência σ, $\times$ em ⊠como a seguir:

    $(\sigma_c(R \times S)) \equiv (R \boxtimes_c S)$

Existem outras transformações possíveis envolvendo condições, seja de junção ou de seleção chamadas transformações lógicas. Por exemplo, uma seleção ou condição

de junção c pode ser convertido em uma condição equivalente pelo uso das seguintes regras:

C = NOT (c1 AND c2) = (NOT c1) OR (NOT c2)

C = NOT (c1 OR c2) = (NOT c1) AND (NOT c2)

2.4 Árvore de Comando

Uma consulta expressa em uma linguagem de consulta de alto nível tal como SQL deve ser primeiro verificada, analisada e validada. A varredura identifica os componentes da linguagem (*tokens*) no texto da consulta, enquanto a análise (*parser*) verifica a sintaxe da consulta para determinar se ela é formulada de acordo com as regras de sintaxe (regras de gramática) da linguagem de consulta. A consulta deve ser validada, verificando-se que todos os atributos e relação de nomes são válidos e semanticamente significativos. Uma representação interna da consulta é então criada, usualmente como uma árvore ou grafo, a qual é chamada uma árvore ou grafo de consulta. O SGBD deve planejar uma estratégia de execução para recuperar o resultado da consulta de arquivos internos da base de dados.

Uma árvore de consulta é uma estrutura em árvore que corresponde a uma expressão da álgebra relacional representando as relações como nós folhas da árvore e os operadoes da álgebra relacional como nós internos. Uma execução da árvore de consulta consiste da execução de uma operação do nó interno se seus operandos estão disponíveis e então o nó interno é substituído pelo resultado da relação da execução do operador. A execução termina quando o nó raiz é executado e produz a relação resultante.

Como por exemplo, dada a consulta expressa em álgebra relacional:

$$\sigma_{(M.Nota \geqslant 6.0)} Aluno \boxtimes_{(A.NumAluno = M.NumAluno)} AulaMinistrada$$

A árvore de consulta desta expressão é mostrada na Figura 2.1.

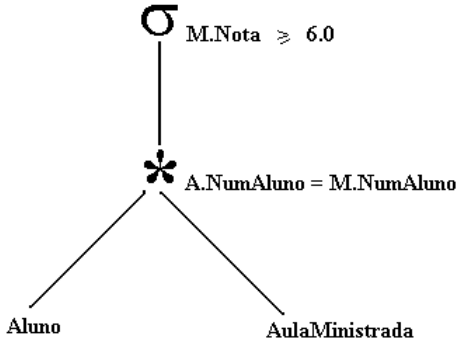


Figura 2.1: Árvore de Consulta

Em geral, muitas expressões diferentes em álgebra relacional, e portanto muitas árvores de consulta diferentes, podem ser equivalentes; isto é, elas podem corresponder à mesma

consulta e por isso gerar o mesmo resultado. A consulta pode ser iniciada de vários caminhos em uma linguagem de consulta de alto nível como SQL. A análise da consulta geralmente irá gerar uma árvore de consulta canônica que corresponde à consulta SQL, sem fazer nenhuma otimização. Esta é então transformada, gerando-se várias alternativas que são submetidas a um processo de estimativa de custo, para que a mais barata seja escolhida para a execução.

Como visto, a árvore de consulta é uma forma padrão simples de expressar consultas. É trabalho do otimizador de consultas transformar a árvore de consulta inicial em uma árvore de consulta final que possa ser mais eficientemente executada. O otimizador deve incluir regras de equivalência entre expressões em álgebra relacional que podem ser aplicadas na árvore inicial, guiada pelas regras do otimizador de consultas, para produzir a árvore de consulta final otimizada.

2.5 Conclusão

Neste capítulo foram abordados os conceitos de álgebra relacional, transformações e otimizações de consultas expressas em álgebra e árvores de comando. Esses são conceitos fundamentais, desenvolvidos no início da criação dos SGBDs relacionais, que tornam possível a sua evolução, e sobre os quais apoiam-se hoje toda a tecnologia dos Sistemas de Gerenciamento de Bancos de Dados em uso.

Capítulo

3

Estado da Arte

Neste capítulo será apresentado o estado da arte das linguagens de consulta a banco de dados.

3.1 Introdução

A década de 60 marcou a área de processamento de dados pela criação e utilização em ambiente comercial dos primeiros programas especializados no gerenciamento de banco de dados. Baseados nos modelos hierárquico e de rede, estes produtos careciam de versatilidade e facilidades para recuperação das informações, e fundamentalmente, não possuíam uma base teórica que permitisse a implementação de modelos matemáticos da realidade. Além disto, a tecnologia de hardware e redes impedia que se obtivesse capacidade de processamento adequada nas estações de trabalho que não passavam de terminais orientados a caracteres. Já no início dos anos 80, com o surgimento do PC e dos servidores de arquivo conectados através de redes, o modelo centralizado dá espaço para um modelo mais flexível de arquitetura de hardware e software, e a capacidade de processamento das estações de trabalho é gradualmente ampliada.

Ainda nos anos 70 surgiu o modelo relacional como uma nova técnica para modelagem de banco de dados e que, a partir dos anos 80, viria a determinar a evolução do mercado de SGBDs, com uma sólida base teórica definida por Codd [Codd, 1970], na qual sugeriu que todos os dados em uma base de dados poderiam ser representados como uma estrutura tabular (tabelas com colunas e linhas, as quais ele chamou relações) e na qual estas relações poderiam ser acessadas usando uma linguagem de alto nível e não procedural (declarativa). Esta linguagem é usada para ganhar acesso sobre as relações, na qual o grupo de dados desejado e o programador não tem que escrever algoritmos para a navegação. Pelo uso

desta proposta a implementação física da base de dados é escondida, de modo que o programador não tem que conhecer a implementação física para ser capaz de acessar os dados. Como conseqüência, em 1974 Chamberlain [Chamberlin and Boyce, 1974] e outros pesquisadores da IBM propuseram a linguagem SEQUEL como a primeira linguagem de banco de dados de alto nível não procedural, a qual eventualmente evoluiu para linguagem hoje conhecida como SQL.

3.2 Linguagem de Consulta

O sucesso de um sistema de banco de dados está bastante relacionado com a facilidade de recuperação das informações armazenadas. É através da linguagem de consulta do banco de dados que o usuário é capaz de expressar um conjunto de restrições e selecionar as informações que deseja.

Uma linguagem de consulta é definida como um linguagem computacional de alto nível para recuperação e modificação de dados em uma base de dados ou em um arquivo, usualmente interativa, *on-line*, e capaz de suportar consultas que são *ad hoc* [Jarke and Vassiliou, 1985]. Ela freqüentemente assume que os principais usuários das linguagens tenham uma boa habilidade técnica. Sendo assim, a interface do usuário tende a ter complexidade limitada, pois assume que o usuário pode suprir as informações necessárias e o tempo de exibição dos resultados é relativamente curto. Uma linguagem de consulta relacional pode ser entendida como um conjunto formalmente bem definido de operadores que podem ser combinados para expressar consultas em banco de dados [Batini et al., 1991a] [Catarci et al., 1997]. O processo de formulação da consulta é realizado em três etapas segundo [Batini et al., 1991b]:

- **localização** - o usuário seleciona o conjunto de informações de seu interesse, ou seja, os objetos que farão parte do seu conjunto de resposta ou que serão utilizados para restringir o conjunto de valores da resposta;
- **definição de requisitos** - sobre os objetos selecionados na fase anterior, são aplicadas algumas restrições que determinam as condições a serem satisfeitas para que os mesmos possam fazer parte do resultado da consulta;
- **visualização do resultado** - o resultado da consulta é apresentado ao usuário. Um mesmo conjunto de informações pode ser apresentado de diferentes formas, mas a escolha da representação adequada possibilita uma melhor interpretação dos resultados por parte do usuário.

A consulta pode ser realizada através de uma linguagem textual ou de uma linguagem visual. A linguagem de consulta textual exige que o usuário tenha um grande conhecimento da sintaxe e da forma como as informações estão organizadas no banco de dados. A linguagem de consulta visual apresenta um ambiente mais amigável para recuperação

de informações permitindo que usuários não familiarizados com a sintaxe da linguagem de consulta textual formulem suas consultas de forma simples e intuitiva.

3.3 Linguagem de Consulta Textual

Desenvolvida para o SYSTEM R [Astrahan et al., 1976], um dos primeiros protótipos de um sistema relacional implementado para sistemas UNIX, a linguagem SEQUEL(Structured English Query Language)[Chamberlin and Boyce, 1974] foi desenvolvida no início da década de 70. O propósito dessa linguagem era ajudar uma nova classe de usuários que estavam surgindo nesta época que não eram especialistas em computação e queriam interagir com o computador com uma linguagem de alto nível e não procedural.

A linguagem SEQUEL foi baseada em uma outra linguagem existente na época considerada de uso fácil, chamada SQUARE, mas que adotava um número muito grande de notações matemáticas. A linguagem SEQUEL apoia-se no uso de palavras da língua inglesa. SEQUEL apresentou ao usuário uma forma consistente para construção de consultas simples. O usuário deveria especificar as colunas que ele gostaria de selecionar (SELECT), a tabela onde (FROM) ele tinha escolhido as colunas e a condição que restringia (WHERE) cada registro a ser retornado. O bloco SELECT-FROM-WHERE foi o componente básico da linguagem, como continua sendo até hoje. Outros conceitos importantes das linguagens de consultas também tiveram sua definição na linguagem SEQUEL a qual mais tarde, devido a problemas legais, foi renomeada para SQL. Em 1986 o comitê ANSI X3H2 aceitou SQL como um padrão ANSI. Uma nova versão estendida foi aprovada em 1992, a qual passou a ser chamada SQL2 ou SQL-92 [ANSI, 1992].

A linguagem SQL tem sido um padrão industrial que atingiu mais de 95% do mercado para gerenciadores de dados de um total que movimentou mais de 15,6 bilhões de dólares no último ano [Leavitt, 2000].

No entanto as limitações dos SGBDRs tradicionais na implementação de aplicações que manipulem tipos complexos de informação como imagens, sons, vídeos, textos entre outros, são inquestionáveis [Leavitt, 2000] [Darwen and Date, 1995]. Apesar da tecnologia de orientação a objeto ser largamente aplicada no mercado de software desde a década de 80, foi na década de 90 que alguns produtos para gerenciamento de bancos de dados orientados a objetos surgiram no mercado tentando solucionar duas das limitações dos bancos de dados relacionais: suporte para dados e aplicações multimídia e a modelagem de dados mais próxima do mundo real. Estes produtos incorporam mecanismos poderosos para manipulação de objetos e dados não estruturados ou complexos. Apesar disto não chegaram a um grau de maturidade aceitável para o desenvolvimento de sistemas que gerenciem grandes volumes de dados ou processem um elevado número de transações[Atkinson et al., 1989].

Dentro do processo de evolução dos SGBDR's está a incorporação de recursos para

armazenamento e manipulação de objetos, baseados no modelo relacional. Da extensão da capacidade de manipulação de dados complexos surgiu o termo Sistema de Gerenciamendo de Banco de Dados Objeto - Relacional (SGBDOR)[Stonebraker and Rowe, 1986].

Estes produtos aproveitam a versatilidade, robustez e tecnologia consagrada dos SGBDR incorporando a flexibilidade e potencialidade da orientação a objetos num único produto, permitindo a recuperação de objetos complexos através da extensão do endereçamento do conteúdo relacional. Hoje praticamente todas as grandes empresas de software de gerenciamento de banco de dados do mercado oferecem produtos seguindo esta linha de evolução, integrados a seus produtos, ou através de pacotes adicionais. Isto tem ocorrido de fato por que muitas empresas que usam SGBDR's não precisam dos recursos do modelo orientado a objeto em todas as suas aplicações, não existindo portanto um interesse unânime numa revolução tecnológica mas sim em uma evolução dos atuais produtos. Além das funções tradicionais dos SGBDRs o SGBDOR deve suportar objetos complexos e regras, acesso de dados através de linguagem não procedural SQL.

A linguagem SQL utilizada para manipulação de bancos de dados relacionais, originalmente foi definida com três tipos fundamentais de dados: Numérico (Inteiro e Decimal), Caractere e Temporal (Data e Hora). Entretanto as aplicações atuais exigem que se guardem tipos complexos de dados que não podem ser diretamente manipulados pela SQL. Para oferecer às ferramentas a capacidade de definição e manipulação de outros tipos de dados, foram integradas à linguagem SQL padrão (DDL e DML), componentes das linguagens ODL e OQL equivalentes àquelas no mundo OO, na forma de uma nova linguagem conhecida como SQL estendida ou SQL00, que inclui as extensões para recuperação de conteúdo de tipos de dados complexos. Ainda fazem parte destes produtos o suporte às características fundamentais da orientação a objetos como herança, polimorfismo e tipificação. Essa nova versão foi aprovada pela ANSI em 2001, tendo sido designada SQL3 [Eisenberg and Melton, 1999].

Alguns trabalhos estendem as linguagens não apenas para suportar outros recursos, mas desenvolvem recursos teóricos/conceituais importantes, para suportar essas extensões. Por exemplo, Sheng em [Sheng et al., 1999] apresenta uma extensão à linguagem OQL, baseada numa álgebra denominada O-Álgebra, para permitir a representação gráfica dos caminhos de acesso e subconjuntos recuperados por uma consulta, feitas sobre os objetos de uma base de dados para aplicações multimídia com suporte temporal. Caminho semelhante é seguido no sistema Delaunay [Cruz et al., 1997], o qual suporta visualização de uma base de dados orientada a objetos especificada pelo usuário como uma linguagem de consulta baseada em constantes visuais.

Em razão disso, quando uma determinada classe de aplicações precisa de recursos que a linguagem SQL não oferece, uma das opções mais procuradas é a extensão dessa linguagem para que ela possa vir a suportar os recursos necessários. Essas extensões são variadas e incluem, por exemplo, extensões para suportar dados temporais [Chen and Zaniolo, 1999] [Snodgrass, 1995], classificação [Chaudhuri et al., 1999],

lógica nebulosa [Dubois et al., 2001], dados multimídia [Nepal and Ramakrishna, 1999] [Lee et al., 1999], Internet [Jones et al., 2001] [Melton et al., 2001] [Zhang et al., 2000], ontologias e bases de conhecimento [Andrade and Saltz, 2000].

Atualmente estão em desenvolvimento propostas para extensões no SQL para suportar dados geográficos e operações de *data mining* [Lin and Huang, 2001] [Netz et al., 2001] [Adler, 2001]. Além disso, devido ao estilo textual de interação de SQL, modalidades alternativas de linguagens de consultas têm sido propostas na literatura, sendo freqüentemente baseadas no uso de representação visual e mecanismos de interação de manipulação direta.

3.4 Linguagem de Consulta Visual

Os sistemas visuais de consulta (*Visual Query Systems* - VQS) utilizam uma linguagem para expressar as consultas em um formato visual e uma variedade de funcionalidades para facilitar a interação do usuário com o sistema [Batini et al., 1991a]. Os VQS podem ser vistos como uma extensão das linguagens de consulta para SGBD (Sistema de Gerenciamento de Bancos de Dados), permitindo que as pesquisas ao banco de dados sejam feitas por usuários menos experientes.

Um VQS pode ser dividido em duas partes: ambiente para interação com o usuário e ambiente de implementação [Batini et al., 1991a]. A primeira define a forma pela qual o usuário irá visualizar e manipular as informações do esquema e a segunda determina como estas informações serão armazenadas e manipuladas internamente. O ambiente para interação com o usuário deve apresentar um modelo de dados com um grande poder de expressão e uma linguagem de consulta bastante amigável. Já o ambiente de implementação depende do banco de dados utilizado e da linguagem de consulta do mesmo. A possibilidade de utilização de modelos diferentes permite que o ambiente para interação com o usuário seja definido sem considerar os aspectos de implementação.

De forma similar, o modelo de consulta externo é representado pela linguagem de consulta utilizada pelo usuário e o modelo de consulta interno é manipulado através da linguagem de consulta definida pelo ambiente de implementação. Por exemplo, o modelo de consulta interno utiliza a linguagem de consulta SQL e modelo de consulta externo, a linguagem gráfica definida para o ambiente. O usuário elabora sua consulta utilizando a linguagem gráfica e esta consulta é mapeada para a linguagem SQL para que possa ser executada no banco de dados. Para permitir a tradução entre os modelos interno e externo deve existir um módulo de mapeamento que faça o relacionamento entre as operações e representações de cada um dos modelos.

Os VQS podem ser classificados em mais de uma categoria de acordo com os elementos visuais definidos para elaboração da consulta e a forma de interação com os mesmos.

Os VQS podem ser classificados de acordo com os elementos visuais definidos para elaboração da consulta e a forma de interação com os mesmos. Levando em conta esta in-

teração, o formalismo visual pode ser classificado segundo quatro paradigmas: Linguagem baseada em formulários, linguagem baseada em diagramas, linguagem baseada em ícones e linguagem híbrida. Cada um deles é descrito nas subseções a seguir.

3.4.1 Linguagem Baseada em Formulários

Um formulário é uma coleção de objetos que tem a mesma estrutura. Ela é a primeira tentativa de deixar o espaço textual monodimensional, explorando a bi-dimensionalidade da tela. Ele facilita que usuários não experientes possam aproveitar a tendência natural das pessoas a usarem estruturas regulares e/ou organizar dados em tabelas. A principal caracteristica de formulários computacionais é que eles são representações estruturada de uma abstração de formulários em papel convencional. Um formulário pode ser visto como uma grade retangular cujos componentes podem ser combinações de células e/ou grupos de células (subformulário). O formulário é uma generalização da tabela, pois os componentes das tabelas são geralmente células elementares, não sendo permitido o agrupamento das mesmas. No formulário, a célula é a menor unidade de dado que pode ser referenciada. A representação dos relacionamentos entre formulários pode ser feita através de uma célula ou de um conjunto de células [Catarci et al., 1997]. A seguir serão apresentadas alguns exemplos de linguagens de consulta baseadas em formulários.

A Linguagem QBE

A linguagem QBE desenvolvida pela IBM no início da década de 70 [Zloof, 1975], foi o primeira linguagem a adotar uma representação visual: as consultas lembram tabelas e o usuário não precisa especificar a estrutura da consulta, porque ela é formulada através da seleção de esquemas de relações que são mostradas na tela. Também não é necessário lembrar os nomes de atributos ou relações porque estes são parte dos esquemas mostrados, diferentemente da linguagem SQL.

A linguagem QBE utiliza um paradigma tabular onde a interface gráfica é utilizada para a representação da porção do esquema da base a ser usada na consulta, mas não utilizam de fato uma representação gráfica para a própria consulta.

As ferramentas que se baseiam em QBE são desenvolvidas para uso específico e todas mantêm o uso da representação tabular. Dentre as aplicações específicas existem por exemplo a linguagem Pictorial Query-By-Example(PQBE)[Papadias and Sellis, 1994], que provê uma interface para uso em aplicações geográficas; e a interface de consulta visual para gerenciamento de grandes volumes de mensagens eletrônicas Mail-by-example(MBE) [Becker and Cardoso, 2000]. O MBE é uma interface visual integrada ao ambiente de e-mail do produto Lotus Notes, que permite aos usuários definirem consultas *ad-hoc* para recuperação de mensagens, pastas ou informações sobre elas.

3.4.2 Linguagem Baseado em Diagramas

Uma linguagem baseada em diagramas é o formalismo visual mais utilizado pelos VQS. Esta linguagem utiliza o paradigma diagramático com um conjunto limitado de símbolos, geralmente correspondendo a figuras geométricas (quadrados, círculos, retângulos, etc...), cada um associado a um tipo conceitual. Além dos símbolos geométricos podem existir um conjunto de ligações para expressar os relacionamentos entre os elementos conceituais. Estes elementos gráficos definidos para a linguagem de consulta constituem o conjunto de primitivas gráficas da linguagem. Os símbolos e suas ligações são geralmente utilizados para visualização do esquema do banco de dados, sendo as consultas realizadas navegando-se pelo esquema e aplicando-se restrições aos elementos do banco de dados. Não é necessário especificar o relacionamento entre os elementos da consulta, pois o mesmo já é apresentado graficamente. Em algumas outras propostas, é permitido aos usuários definirem suas visualizações, deste modo costurando os dados mostrados como em DOODLE [Cruz, 1992] a visualização definida pelo usuário pode ser usada tanto na consulta a base de dados quanto na definição de novas visualizações.

O sistema QBD*

O sistema QBD**(Query by Example)* [Catarci and Santucci, 1994] [Angelaccio et al., 1990], mostrado na Figura 3.1 é um sistema de consulta visual baseado na representação diagramática do esquema entidade-relacionamento como mostrado na Figura 3.1, segundo os autores descrevendo o fundamento (relacional) da base de dados. Ele balanceia alto poder expressivo com facilidade de uso. Além disso, ele inclui uma classe significativa de consultas recursivas e variáveis estendidas de um grupo de operadores da álgebra relacional. Através destes operadores é possível computar o grupo de operadores para qualquer par de tabelas relacionais compartilhando o mesmo identificador.

A estrutura geral do QBD* é baseada na localização entre elementos distintos, do chamado conceito principal (entidade ou relacionamento), que pode ser visto como o ponto de entrada de uma ou mais consultas; estas sub-consultas expressam navegações possíveis do conceito principal para outros conceitos no esquema. Os atributos pertencentes a cada sub-consulta são determinados pela seguinte estratégia: os atributos do conceito principal são automaticamente considerados (a menos que sejam removidos explicitamente pelo usuário) enquanto os outros são mostrados no resultado apenas se requisitados pelo usuário. A presença de um conceito principal associa um tipo a cada sub-consulta; por exemplo, se a entidade principal é a entidade 'pessoa' o resultado da consulta será um conjunto de pessoas.

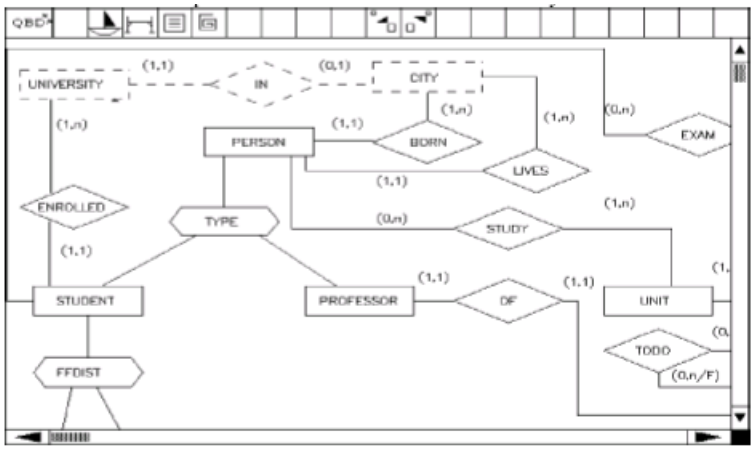


Figura 3.1: Esquema de navegação da linguagem QBD* através de caminhos existentes

3.4.3 Linguagem Baseada em Ícones

Um ícone pode ser definido na computação como um segmento, uma imagem estilizada. Segmentação de uma imagem implica na extração de um componente simples de um fundo enquanto a estilização se referencia a uma representação feita por um pequeno número de linhas significantes que podem constituir junto um ícone. Os VQS precisam representar não apenas imagens de objetos reais, mas também conceitos, ações ou processo. Os ícones podem ser uma analogia, um minemônico ou uma convenção, na qual, em princípio, são os símbolos. Sendo assim tem-se a seguinte definição para ícone segundo Catarci em [Catarci et al., 1997] "o ícone é um objeto visual segmentado no qual informa-se a visão sobre uma mensagem ou informação interior (conceito, função, estado ou modo) atribuída por um projetista".

O objetivo de um ícone é representar (ou lembrar) um certo conceito, enquanto diagramas favorecem a visualização de relações entre conceitos. Além disso, nos diagramas a descrição de relações não usam simbolos mas ligações entre simbolos, e a descrição da ligação é provida sem ambiguidade. O ícone tem um significado metafórico poderoso, isto é, ele pode ser interpretado como uma metáfora visual. Em contraste, o único poder metafórico de um diagrama está nas legendas (*labels*).

Os VQS baseados em ícones usam um grupo de icones na qual denotam as funções do sistema. A consulta é expressa primeiramente pela combinação de ícones de acordo com alguma sintaxe espacial. O sistema baseado em ícones representa uma melhoria na amigabilidade de sua manipulação. Geralmente nestes sistemas o esquema da base de dados não é mostrado. De fato, estes VQS são principalmente endereçados para usuários que não são familiares com os conceitos de modelos de dados e podem encontrar dificuldades para interpretar um diagrama E-R. Quando um sistema icônico é projetado, um problema crucial é como construir ícones que podem expressar um significado expressivo para os seres humanos, muito frequentemente a mesma imagem carrega diferrentes significados para pessoas diferentes.

O sistema QBI

QBI (*Query By Icon*)[Massari et al., 1995]é um sistema que permite usuários consultar e entender a informação contida em uma base de dados pela manipulação de ícones. Ele é especialmente enderaçado para usuários que tenham habilidades limitadas e exigências imprevisiveis em suas requisições. Ainda que QBI seja uma interface de consulta de propósito geral, ele tem suas origens em uma base de dados distribuída de imagens radiológicas.

O sistema mantém um esquema interno da estrutura da base de dados feita de acordo com o modelo semântico denominado *Graph Model*. Este esquema, não visivel para o usuários, é um representação gráfica tanto da informação estruturada (por exemplo, classes e relacionamentos) quanto das restrições de conscistência da base de dados. A visão externa é baseada no conceito de Atributo Generalizado pelo significado da qual a base de dados aparece estruturada. Também há a possibilidade de reuso de consultas na especificação de outras mais complexas, aumentanto assim a usabilidade do sistema.

QBI traz as vantagens de ícones metafóricos para a visualização tanto da informação estrutural quanto das restrições.

3.4.4 Linguagem Híbrida ou Visual

A representação híbrida usa uma combinação arbitrária de dois ou mais formalismos visuais, oferecendo assim ao usuário várias alternativas para a representação da base de dados e consultas, ou combinando diferentes formalismos em uma única representação. Muito dos VQS adotam mais de um formalismo, mas freqüentemente um deles é predominante.

Os sistemas que apresentam uma linguagem híbrida combinam qualquer uma das linguagens de consulta apresentadas anteriormente para formulação de consultas.

Uma linguagem de consulta híbrida também pode ser uma linguagem de consulta visual. Este termo denota um caráter mais abrangente de interação onde a comunicação com o usuário é feita através de símbolos que carregam a semântica das operações.

3.5 Conclusão

O sucesso de um sistema de banco de dados está bastante relacionado com a facilidade de recuperação das informações armazenadas. Apesar das linguagens de consultas textuais como SQL serem computacionalmente completas, elas têm a desvantagem de não permitir ao usuário nenhuma interação gráfica com a informação contida na base de dados. O sistema de consulta visual usa todo o poder de novas tecnologias, como duas/três represetações dimensionais, cores e múltiplas janela, deste modo estendendo a comunicação homem-máquina em várias direções. Ao mesmo tempo, a consulta é representada de uma maneira mais natural, diminuindo o caminho de percepção que o usuário deve seguir para

reconhecer a realidade de interesse. Além disso, linguagens visuais são mais flexiveis que as tradicionais linguagens tal como SQL, onde usualmente o aprendizado de toda linguagem (ou da parte significante) é necessário para ser capaz de expressar consultas simples ou complexas.

Capítulo

4

Fluxo de Dados

Neste capítulo inicialmente será apresentado como expressar consultas através de fluxo de dados, as vantagens de se usar este tipo de diagrama e como as consultas são executadas.

4.1 Introdução

Um dos principais desenvolvimentos na área de base de dados diz respeito às ferramentas que proveêm aos usuários um entendimento simples da base de dados e uma extração amigável da informação.

A criação de uma consulta usando uma linguagem textual obriga o usuário a ser capaz de modelar a informação contida na base de dados e transformar essa informação em comando para recuperar a informação desejada. As linguagens visuais desenvolvidas nos últimos anos permitem aos usuários menos experientes acessarem os dados de uma forma mais intuitiva, mas não permitem que usuário especialistas façam consultas complexas, tornando-se assim linguagens sem um grande poder expressivo. A ferramenta descrita neste trabalho possibilita que usuários especialista criem consultas complexas ou não, graficamente por meio de diagramas de fluxo de dados.

4.2 Expressando Consultas através de Fluxo de Dados

Diagramas de fluxo de dados são adequados para a representação de como uma coleção de processos se interconectam para realizar uma atividade complexa.

As operações de recuperação de dados em SGBDRs podem ser consideradas processos

e as consultas podem ser consideradas atividades expressas através de diagramas de fluxo de dados. Essa forma de representação carrega consigo as vantagens típicas desses diagramas, principalmente a representação visual e a intuitividade das atividades (no caso, das consultas) e a interatividade na elaboração e execução das mesmas.

Atualmente tem sido intensamente pesquisado o suporte que SGBDRs podem dar a tipos de dados complexos como imagens e vídeo, nesses casos, os diagramas de fluxo de dados apresentam ainda a vantagem de permitir uma redução na descontinuidade semântica entre a representação das consultas, que passam a poder ser feitas também graficamente.

Na representação de consultas como fluxos de dados, representam-se os operadores da álgebra relacional como processos que podem ser conectados em um diagrama. Cada operador é representado por um ícone. Além dos operadores tradicionais da álgebra relacional foram incorporados operadores para entrada e saida dos dados(tabelas).

Os operadores tradicionais da álgebra relacional também sofreram algumas modificações conceituais para aumentar sua flexibilidade de utilização. Os operadores de junção e o operador de seleção da álgebra relacional incorporam a especificação de suas respectivas condições de junção e de seleção como parte do operador. Na versão para compor os diagramas de fluxo de dados, esses operadores foram alterados para que a especificação dessas condições seja feita externamente. O operador de projeção também teve a sua lista da atributos especificada externamente. Assim, enquanto na versão da álgebra relacional todas as variáveis são sempre tabelas, na versão para fluxo de dados existem outros tipos de dados, que são "conduzidos" por um duto(ligação entre dois operadores no diagrama) de dados específico para esse tipo de dado.

Verificou-se que são necessários quatro tipos de dutos(fluxos/conexões) para expressar uma consulta em um diagrama de fluxo de dados, que são:

- **Fluxo de Tuplas (ou de Dados) (D)** - Esses dutos correspondem aos dados (propriamente ditos), que são processados desde as tabelas armazenadas na base até que se obtenham as respostas à consulta;

- **Fluxo de Condições (C)** - Esses dutos levam as condições que são utilizadas pelos operadores de seleção e pelos vários operadores de junção;

- **Fluxo de Atributos (A)** - Esses dutos especificam conjuntos de atributos que são utilizados, por exemplo, pelos operadores de projeção;

- **Fluxo de Funções (F)** - Esses dutos especificam funções que são utilizadas, por exemplo, em operadores de cálculo de agregados (embora ainda não tenham sido implementados operadores de agrupamento e cálculo de agregados, os dutos necessários já estão operacionais).

Cada operador possui um ou mais tipo de fluxo(duto) sendo que cada duto pode ser ainda de entrada ou de saída de dados. O operador que possui um certo tipo de duto de

saida só pode ser conectado a um operador que possui o mesmo tipo de duto de entrada. Cada fluxo é representado por uma cor diferente, sendo assim temos azul para fluxo de tuplas ou dados, vermelho para o fluxo de condições, amarelo para o fluxo de atributos e verde para o fluxo de condições.

Após receber os dados dos fluxos de entrada, estes são processados segundo a operação representada pelo ícone e enviados ao próximo operador através do fluxo de saída. Sendo assim os operadores definidos nessa linguagem têm a seguinte especificação:

Tabela - Este operador representa a entrada dos dados, na qual é feita a escolha e leitura da tabela no diagrama, possui um duto **Saída do Fluxo de Tuplas**.

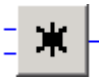
Condição - Este operador representa a escolha da condição da seleção ou da junção, dependendo do operador que esteja conectado com ele. Possui um duto de **Saída do Fluxo de Condições**.

Lista de Atributos - Este operador representa a escolha dos atributos da projeção. Possui um duto de **Saída do Fluxo de Atributos**.

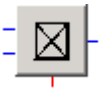
Seleção - Este operador representa a operação de seleção. Possui três tipos de dutos: um duto de **Entrada do Fluxo de Tuplas**, um de **Entrada do Fluxo de Condições** e outro de **Saída do Fluxo de Tuplas**.

Projeção - Este operador representa a operação de projeção. Possui três tipos de dutos: um duto de **Entrada do Fluxo de Tuplas**, um de **Entrada do Fluxo de Atributos** e outro de **Saída do Fluxo de Tuplas**.

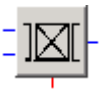
Produto Cartesiano - Este operador representa a operação de produto cartesiano. Possui dois tipos de dutos: dois duto de **Entrada do Fluxo de Tuplas** e um duto **Saída do Fluxo de Tuplas**.



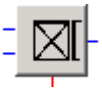
Junção Natural - Este operador representa a operação de junção natural. Possui três tipos de dutos: dois dutos de **Entrada do Fluxo**, **de Tuplas** e um duto de **Entrada do Fluxo de Condição** e outro de **Saída do Fluxo de Tuplas**.



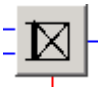
Junção Externa total - Este operador representa a operção de junção externa total. Possui três tipos de dutos: dois dutos de **Entrada do Fluxo de Tuplas**, um duto de **Entrada do Fluxo de Condição** e outro de **Saída do Fluxo de Tuplas**.



Junção Externa à esquerda - Este operador representa a operação de junção externa à esquerda. Possui três tipos de dutos: dois dutos de **Entrada do Fluxo de Tuplas**, um duto de **Entrada do Fluxo de Condição** e outro de **Saída do Fluxo de Tuplas**.



Junção Externa à direita - Este operador representa a operação de junção externa à direita. Possui três tipos de dutos: dois dutos de **Entrada do Fluxo de Tuplas**, um duto de **Entrada do Fluxo de Condição** e outro de **Saída do Fluxo de Tuplas**.



Saída dos Resultados - Este operador representa a tabela resultante da consulta, possuí apenas o duto de **Entrada do Fluxo de Tuplas**.

A Tabela 4.1 apresenta a sumarização da descrição acima com a indicação dos tipos de dutos que podem ser criados em cada operador através das conexões. Assim, indica-se o tipo de conexão (D, C, A ou F), e se ela é de Entrada (E) ou Saída (S). Como exemplo, a junção natural recebe dois dutos (conexões) de tuplas (2 ED) e um duto de condição (EC), e gera um duto de tuplas (SD).

Operador	Significado	Conexão
π	Projeção	ED, SD, EA
σ	Seleção	ED, SD, EC
$\times$	Produto Cartesiano	2ED, SD
$\boxtimes$	Junção	2ED, SD, EC
$\boxtimes$	Junção Externa à Esquerda	2ED, SD, EC
$\boxtimes$	Junção Externa à Direita	2ED, SD, EC
$\bowtie$	Junção Externa Completa	2ED, SD, EC
$\cup$	União	2ED, SD
✎	Leitura de tabela armazenada na base de dados	SD
?	Condição Manual	SC
{}	Lista de Atributos Manual	SA
▙	Saída de Resultados	ED

Tabela 4.1: Significados e tipos de conexões dos operadores tradicionais da álgebra relacional.

A definição dos múltiplos tipos de fluxo de dados permite aumentar o poder representativo da nova linguagem para suportar novos operadores, não previstos na álgebra relacional. Dentre esses operadores, surgem como aplicação futura, os processos de data mining. Por exemplo, um processo de descoberta de agrupamentos gera uma coleção de condições para definir cada classe que pode ser entrada para os operadores que recebem o fluxo de condições. Da mesma forma, um processo de redução de dimensionalidade [Traina et al., 2000] gera uma lista de atributos, que corresponde às dimensões escolhidas no processo e que pode ser entrada para operadores que recebem o fluxo de atributos. Outra aplicação futura é o suporte à representação de imagens, onde tanto operadores que geram condições quanto os que geram listas de atributos e de funções podem ser parametrizados por janelas específicas com suporte a imagens.

Utilizando esse conceito de fluxo de dados, foi desenvolvida uma interface de consulta a um SGBD Relacional. Essa interface é realizada através de um editor de consultas, que é formado basicamente por uma coleção de operadores e um painel onde o usuário pode montar um diagrama de fluxo que representa sua consulta (instanciando e parametrizando os operadores). Embora o projeto desta ferramenta inclua a possibilidade de suportar futuramente processos especiais para determinadas aplicações (como por exemplo os processos de data mining e operadores para processamento de imagens), os operadores implementados e descritos neste trabalho são os operadores usuais da álgebra relacional.

A interface é suportada por uma ferramenta que permite a edição de diagramas de fluxo de dados e a execução desses diagramas, isto é da consulta. Durante a fase de edição,

o usuário pode escolher os operadores que irão compor a sua consulta e "carimbá-los" no painel. Cada operador é representado por um ícone, que indica a operação que ele realiza. Ao mesmo tempo o usuário vai definindo as conexões entre os operadores instanciados e seus parâmetros.

Essa ferramenta, que pode ser descrita como um editor de Consultas por Diagramas de Fluxo de Dados (DFQL - Data Flow Query Language), tem sua tela principal mostrada na Figura 4.1, onde estão indicadas as áreas principais de interação com o usuário.

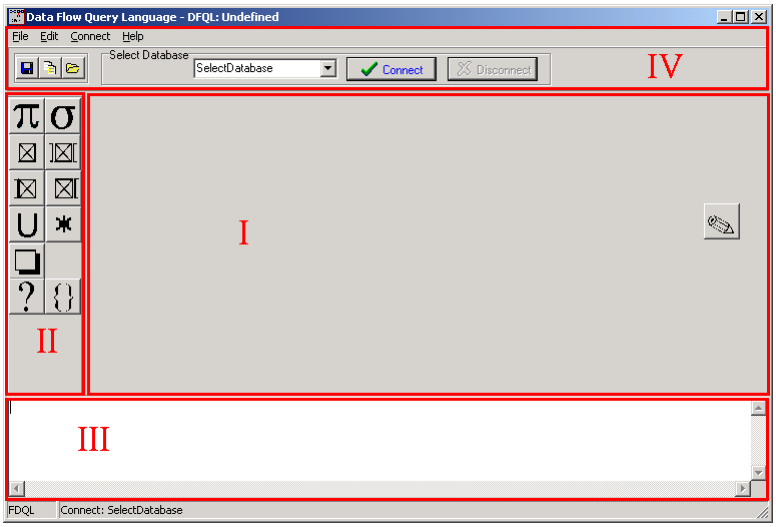


Figura 4.1: Tela principal da ferramenta DFQL

A área I de acordo com a Figura 4.1, corresponde ao painel onde a consulta é construída, "carimbando-se" os operadores que compõem a consulta. A área II é a coleção de operadores disponíveis, onde o usuário escolhe o operador para ser carimbado no painel. A área III é usada para mensagens da ferramenta, por exemplo, nessa área são mostradas as consultas em SQL montadas automaticamente quando as mesmas são submetidas ao SGBD. A área IV mostra os menus e ferramentas para a operação da ferramenta, como por exemplo a ferramenta de conexão com a base de dados.

4.3 Representando Consultas na Ferramenta DFQL

A representação de uma consulta típica em DFQL pode ser planejada e representada da mesma maneira que na álgebra relacional. Por exemplo, considerando-se uma base de dados de matrículas que possui uma tabela de alunos e uma tabela das matrículas em disciplinas feitas pelo alunos, com as respectivas notas obtidas, a consulta: **"Quais são as notas que cada aluno obteve em cada disciplina em que foi aprovado?"** pode ser expressa em SQL como:

```
SELECT *
FROM Aluno A, AulaMinistrada M
WHERE A.NumAluno = M.NumAluno AND M.Nota >= 6.0;
```

Uma possível expressão dessa consulta em álgebra relacional é a seguinte:

$$\sigma_{(M.Nota \geqslant 6.0)} Aluno \boxtimes_{(A.NumAluno = M.NumAluno)} AulaMinistrada$$

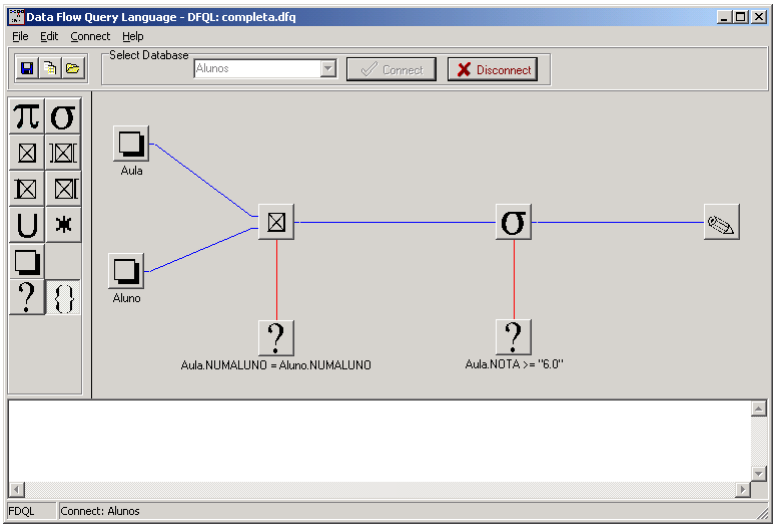


Figura 4.2: Representação da consulta anterior

Utilizando a representação pelo paradigma de fluxo de dados, o mesmo exemplo pode ser representado como mostrado no painel da Figura 4.2. Verifica-se portanto que a representação em fluxo de dados de uma consulta típica é a própria representação gráfica da notação algébrica da consulta. A representação gráfica preserva também todas as possibilidades de otimização que a notação algébrica propicia.

Enquanto "monta" a sua consulta, o usuário deve também parametrizar os operadores utilizados. Isso corresponde a definir a tabela que deve ser lida em cada operador de leitura de tabela, definir as condições nos operadores de condição manual e os atributos nos operadores de listas de atributos. A escolha da tabela a ser lida num operador de leitura (através de um duplo toque no ícone do operador de leitura em questão) é mostrada na Figura 4.3. Para que o usuário possa escolher a tabela dentre as disponíveis na base de dados, como mostrado nessa figura, é necessário que o usuário já tenha realizado a conexão com uma base de dados usando a ferramenta de conexão da área IV da ferramenta conforme apresentado na Figura 4.1.

4.4 Executando Consultas na Ferramenta DFQL

Após serem explicitados os parâmetros que definem cada um dos operadores que compõem o diagrama de fluxo de dados, o usuário pode então executar as consultas criadas. Existem duas maneiras de solicitar a execução de uma consulta representada em um diagrama de fluxo de dados na ferramenta DFQL: execução completa e visualização de resultados intermediários.

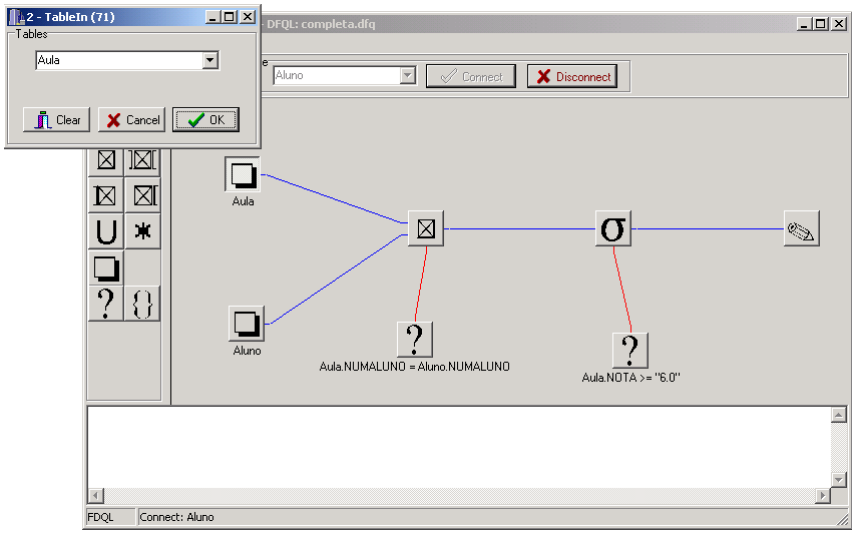


Figura 4.3: Escolha da tabela

A visualização de resultados intermediários pode ser solicitada em cada operador da rede de fluxo de dados. Para isso o usuário deve indicar o operador no qual se pretende verificar o resultado. A Figura 4.4 ilustra esse modo de consulta, na qual foi indicada a visualização do operador de leitura da tabela Aluno (parametrizada anteriormente como mostrado na Figura 4.3). Para isso, o diagrama é rastreado a partir desse módulo voltando-se até aos operadores de leitura das tabelas e compondo-se a consulta que deve ser efetuada para obter-se os dados desse módulo.

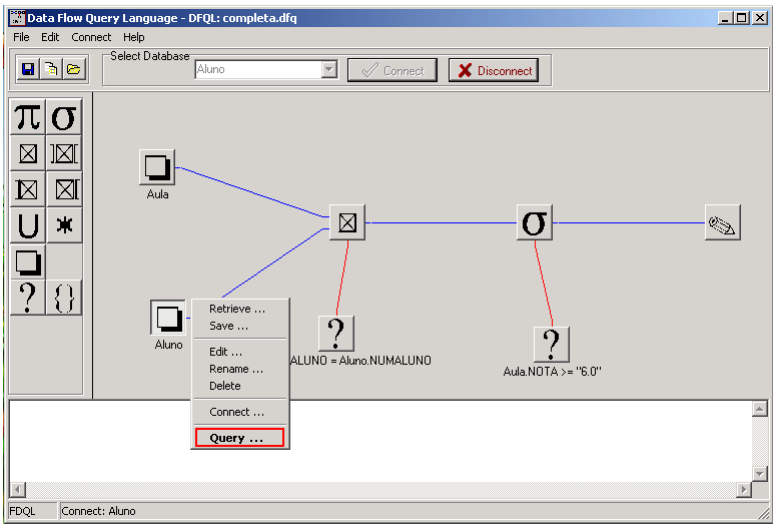


Figura 4.4: Executando uma consulta na tabela

Quando um pedido de consulta é feito como mostrado na Figura 4.4, é chamada uma função recursiva, que recebe como parâmetro o índice do objeto, o tipo (tabela, projeção, seleção, etc..) e o conector de saída, com exceção do símbolo de saída dos dados que é passado ao conector de entrada. Essa função recursiva, varre a rede a começar do símbolo onde foi feito o pedido de consulta e vai atráves dos fluxos (conexões) existentes varrendo a rede e montando uma pilha com os dados da consulta. Esse procedimento pode

ser descrito como exemplificado na Figura 4.5.Quando é solicitada a execução completa da consulta, toda rede é avaliada, partindo-se do operador resultado que no diagrama é indicado pelo operador ✎.

$$
\begin{aligned}
R_1 &\leftarrow (\sigma_{(Nome=\text{“}Lab.BasedeDados)}(Discip) \boxtimes Turma)_{(Sigla=Sigla)} \\
R_2 &\leftarrow (R_1 \boxtimes \sigma_{(Nota \geq 5.0)}(Matricula))_{(Codigo=CodigoTurma)} \\
R_3 &\leftarrow (R_2 \boxtimes \sigma_{(Idade>22)}(Aluno))_{(NUSP=NUSP)} \\
Resultado &\leftarrow \pi_{(Aluno.Nome,Turma.Codigo,Matricula.Nota)}(R_3) \qquad (1)
\end{aligned}
$$

Figura 4.5: Procedimento de execução da consulta

A função recursiva vai passando de um operador para o outro até encontrar um operador “Tabela de Entrada”. Quando esse operador é encontrado, a tabela indicada no operador é lida e então é criada pela função uma StringList que conterá, atributos, nome das tabelas, condições e indicação de join e união quando existente. Depois de lida a tabela e montada a StringList, a função vai retornando aos outro operadores do diagrama, verificando a existência de outros tipos de fluxos (condições e atributos). A figura 4.5 mostra os passos seguidos pela função para transformar o diagrama na representação interna. Ao final dessa função a StringList é lida por uma outra função na qual a desmonta e identifica cada string, através de um caracter existente na frente de cada string que indica se é um atributo, uma tabela ou uma condição, criando assim a representação interna, isto é transformando o diagrama em uma consulta escrita em SQL.

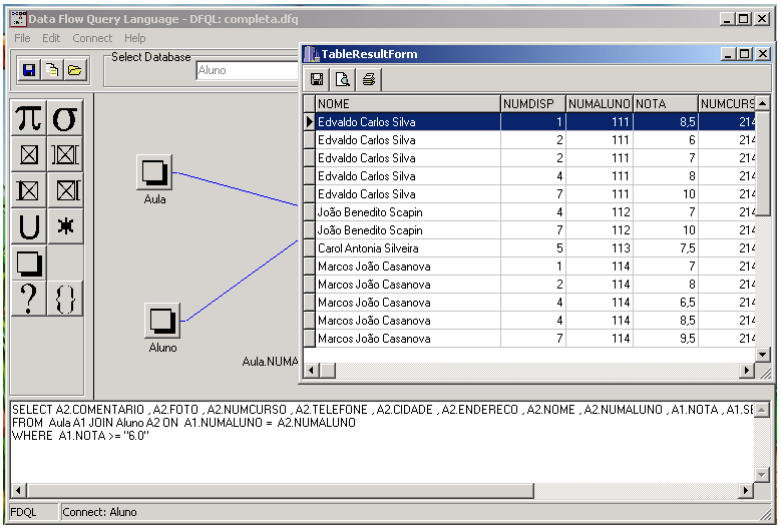


Figura 4.6: Visualizando o resultado da consulta inteira

Depois de montada a consulta em SQL, ela é submetida à base e a resposta é mostrada em uma janela temporária, como mostrado na Figura 4.6. Esse resultado pode ser guardado em um arquivo ou impresso.

A visualização de resultados intermediários é realizada de maneira semelhante a consulta, porém tendo como módulo de partida o operador escolhido, e podendo não apenas visualizar o resultado, mas também gravá-lo em um arquivo externo, de acordo com os

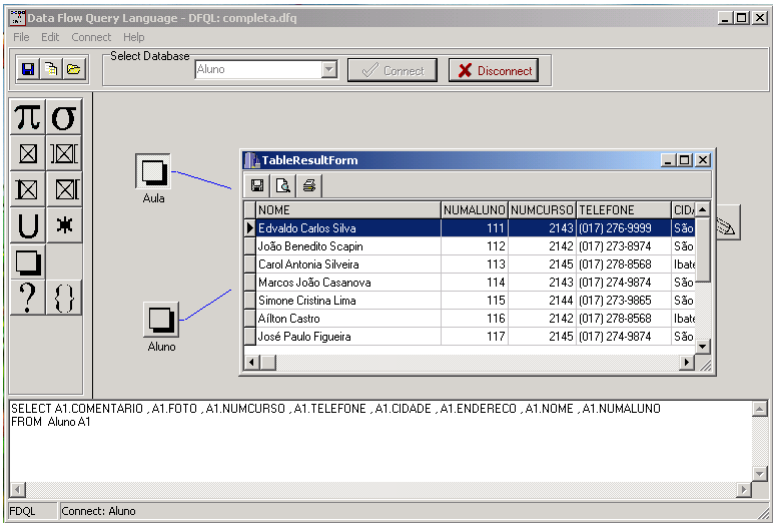


Figura 4.7: Visualizando o resultado da consulta na tabela

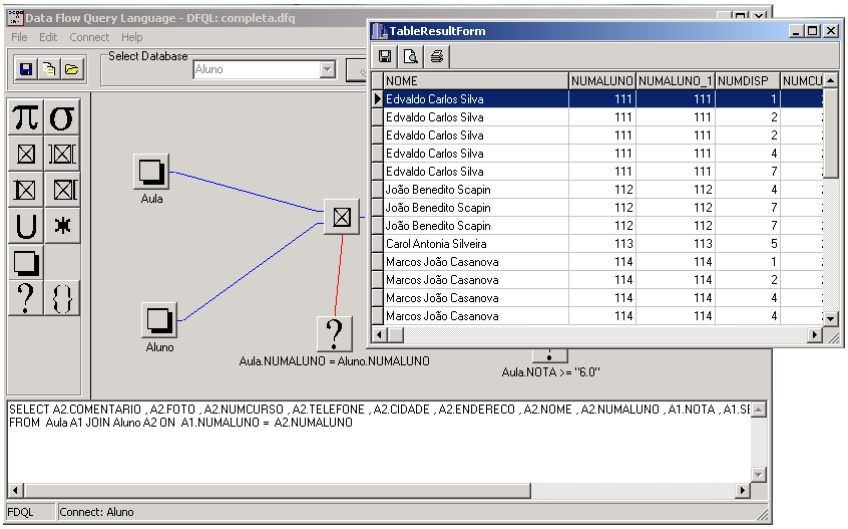


Figura 4.8: Visualizando o resultado da consulta na junção

parâmetros de configuração do operador resultado. A Figura 4.7 mostra o resultado da execução após o operador de tabela de entrada e a Figura 4.8 após o operdor de junção.

Nota-se que, em qualquer visualização, (figuras 4.6, 4.7 e 4.8) a área III da ferramenta como apresentado na Figura 4.1 mostra a consulta em comandos SQL equivalente à consulta efetuada para a execução do diagrama.

4.5 Implementação: Integrada Vs. Independente

Se essa interface fosse implementada atrelada a um interpretador de consultas completo, essa representação interna corresponderia à árvore de comandos, equivalente à que é normalmente gerada durante a fase de interpretação de um comando em SQL em um servidor relacional. Embora computacionalmente mais eficiente, essa abordagem requer uma forte integração com algum gerenciador de banco de dados relacional e tornaria a ferramenta (aqui descrita) restrita à operação desse gerenciador específico.

Para evitar essa dependência, a ferramenta foi implementada representando a consulta extraída da rede de fluxo de dados em um ou mais comandos em SQL. Exceto de uma menor eficiência na interpretação dos comandos, essa abordagem apresenta os mesmos benefícios que seriam alcançados se a integração fosse feita internamente a um interpretador de consultas. Tais benefícios seriam:

- facilidade de uso;
- poder expressivo;
- adequação para suportar consultas equivalentes sobre dados gráficos;
- essa abordagem não fica "presa" a nenhum produto em particular.

4.6 Conclusão

Neste capítulo foi apresentado o projeto desenvolvido no mestrado isto é, a linguagem gráfica e a ferramenta desenvolvida baseada no paradigma de fluxo de dados.

Este trabalho explora a representação visual de uma consulta utilizando o paradigma de fluxo de dados como uma nova forma de acesso a bases de dados relacionais: na realidade uma forma tão flexível quanto às tradicionais linguagens SQL e QBE, ao mesmo tempo em que permite uma integração mais natural entre gerenciadores relacionais e com ambientes, como aqueles que incluem a manipulação de imagens, e/ou incluem processos de análise de dados, como processos de data mining e OLAP. Esta interação é conseguida através da definição de uma linguagem de consulta a bancos de dados relacionais que utiliza o paradigma de fluxo de dados, e uma ferramenta (DFQL) que permite ao usuário a criação de diagramas de consulta e a execução das mesmas.

Capítulo

5

Construindo uma Consulta com a Ferramenta Data Flow Query Language - DFQL

Neste capítulo será mostrado como se construir uma consulta com a ferramenta DFQL. A construção de uma consulta na ferramenta corresponde à execução dos seguintes passos:

1. Escolha da Base de Dados;
2. Escolha dos Operadores;
3. Conexão dos Operadores;
4. Parametrização dos Operadores;
5. Execução da Consulta;
6. Conclusão.

5.1 Escolha da Base de Dados

Antes de usar a ferramenta DFQL, é necessário armazenar os dados a serem visualizados em uma relação de uma base de dados que deve estar registrada no BDE. Para se conectar pela primeira vez a uma base de dados na ferramenta, é preciso criar uma conexão BDE (veja "BDE Administrator" no Painel de Controle do Windows) ou ODBC (veja também o "Adminsitrador de fonte de dados ODBC") como mostrado nas Figuras 5.1, 5.2 respectivamente.

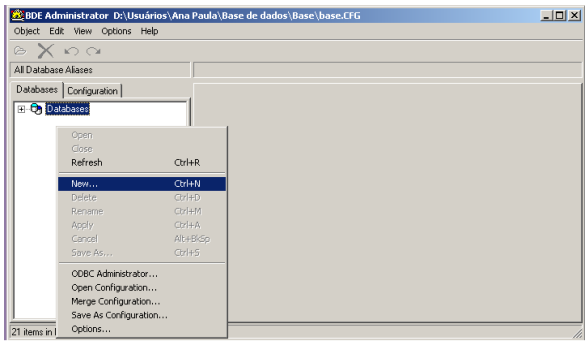


Figura 5.1: *Borland Database Engine* - BDE

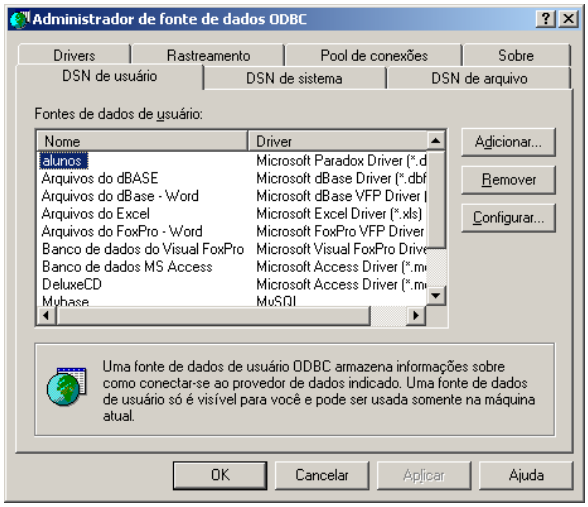


Figura 5.2: *Open Database Connectivity* - ODBC

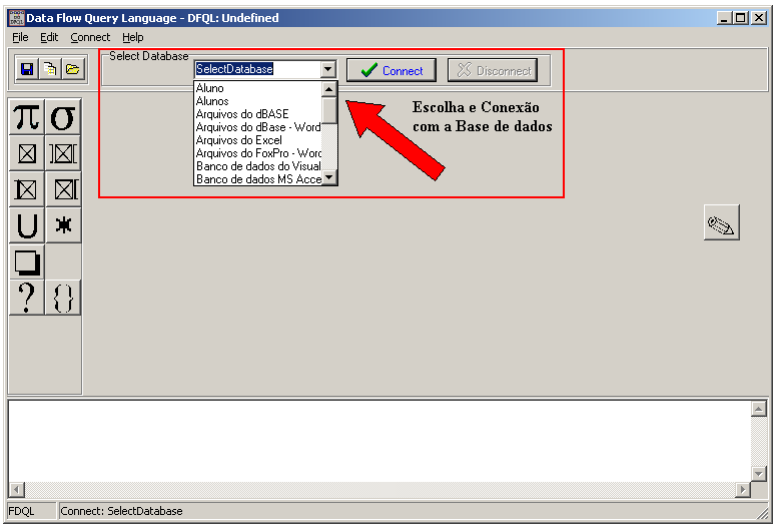


Figura 5.3: Seleção e Conexão com a Base de dados

O usuário pode escolher entre se conectar com a base logo que entra na ferramenta (Figura 5.3), ou se conectar quando for parametrizar os operadores (veja mais detalhes na seção 5.3). Para fazer a escolha da base de dados, o usuário deve selecionar um dos itens da caixa de seleção *"Select Database"* e depois clicar no botão *"Connect"* como mostrado na Figura 5.3. Se houver algum controle de acesso na base selecionada, o sistema pede a identificação e a senha do usuário.

A barra de *status* mostrada na área III da Figura 5.4 indica com que base de dados o usuário se conectou. Para trocar de base ou apenas se desconectar o usuário deve clicar no botão *"Disconnect"* mostrado na Figura 5.4 área I ou no menu *"Connect"* mostrado também na Figura 5.4 área II.

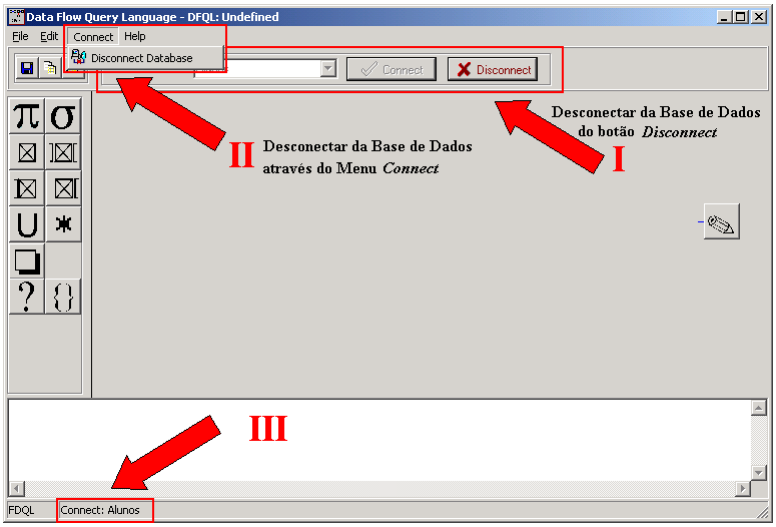


Figura 5.4: Seleção e Conexão com a Base de dados

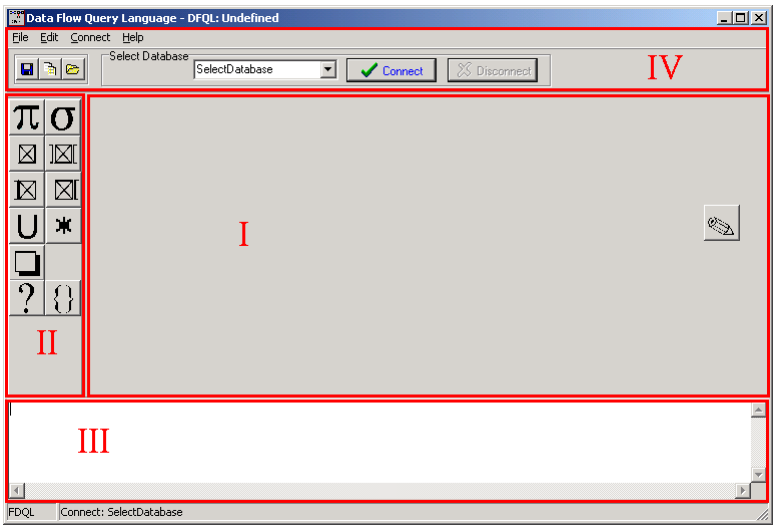


Figura 5.5: Tela Principal da Ferramenta DFQL

5.2 Escolha dos Operadores

Para carimbar os operadores no editor de consultas é necessario antes o usuário clicar sobre o símbolo do operador desejado na paleta mostrada na área II da Figura 5.5 e depois no painel representado pela área I da Figura 5.5.

Dessa forma o usuário vai instanciando os operadores que farão parte da consulta que está sendo construída como mostra a Figura 5.6. Para deletar um operador colocado erroneamente no painel, o usuário deve clicar com o botão direito sobre o ícone que se deseja deletar e escolher a opção delete no menu *pop-up* que aparecerá.

5.3 Conexão dos Operadores

Depois de ter os operadores escolhidos carimbados no painel, esses operdores devem ser conectados de forma a montar o diagrama que representará a consulta. Para criar uma

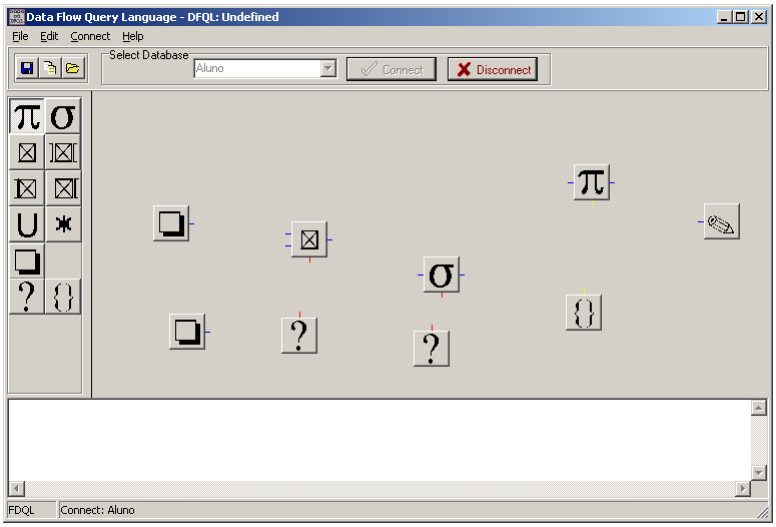


Figura 5.6: Operadores carimbados no painel

conexão(fluxo) entre dois operadores basta clicar com o botão direito do mouse sobre um dos operadores a serem ligados e escolher a opção *"Conectar"* que aparecerá no menu *pop-up* como mostra o área I da Figura 5.7 e em seguida clicar sobre o outro operador representado pela área II da Figura 5.7. Com isso é aberta uma nova janela mostrada Figura 5.7 área III chamada *"ConnectForm"*.

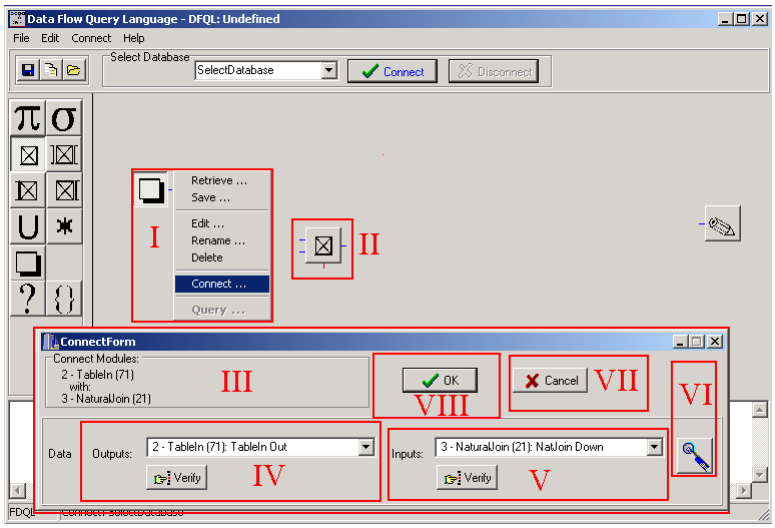


Figura 5.7: Passos para a criação da conexão entre dois operadores

Esse nova janela aberta contém a ferramenta para fazer a criação do fluxo entre os dois operadores escolhido. Para efetuar essa criação o usuário deve conferir se o nome do operador de entrada indicado pelo área V na Figura 5.7 está correto, caso não esteja o nome correto deve ser selecionado na caixa de seleção *"INPUTS"*. O mesmo procedimento deve ser feito com o nome do operador de saída na caixa de seleção *"OUTPUTS"* Figura 5.7 área IV.

Depois de escolher os operadores que se deseja conectar o usuário deve clicar no botão de criação do fluxo indicado pela Figura 5.7 passo VI para fazer a ligação entre os operadores. A criação da conexão entre os operador é confirmada clicando-se no botão "OK"

mostrado na área VII da Figura 5.7 que fecha essa tela de conexão e retorna para a ferramenta que agora exibe o fluxo criado entre o operador Tabela de Entrada e Junção Natural como mostrado na Figura 5.8.

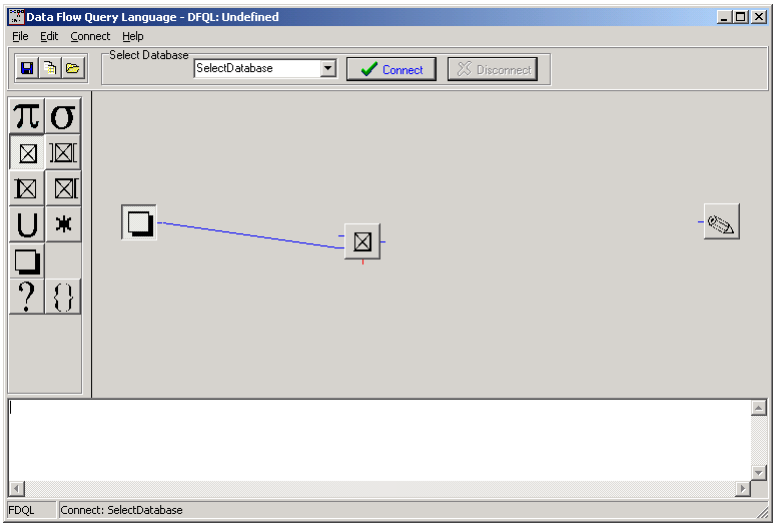


Figura 5.8: Dois operadores conectados por um fluxo de Dados

Vale lembrar que esse processo de conexão dos operadores deve ser executado em todos os operadores existentes no paínel construindo assim o diagrama de fluxo de dados.

5.4 Parametrização dos Operadores

A parametrização dos operadores pode ser feita em qualquer parte do processo de criação do diagrama de fluxo de dados, para isso basta que a ferramenta já esteja conectada com uma base de dados. Uma vez colocados no painel, cada operador pode ser parametrizado em uma janela específica, acessível através de um toque duplo sobre seu ícone no painel. Não são todos os operadores que podem ser paramentrizados, apenas o operador Tabela de Entrada, de Condição e Lista de Atributos Manuais. Os outros operadores apenas processam as informações vinda desses operadores segundo a operação que representam.

Para o operador Tabela de entrada, a janela de parametrização exibida contém uma caixa de seleção com todas as tabelas existentes na base selecionada como mostra a Figura 5.9.

O operador de "Condição" deve ser paramentrizado para uma condição de seleção ou de junção escolhendo respectivamente a opção *"Value"* ou *"Atribut"*. Para a condição de seleção o usuário deve selecionar o nome da tabela e o atributo da condição, operador e atribuir um valor para a condição como mostrado na Figura 5.10. Já para a condição de junção, o usuário deverá escolher as duas tabelas e em seguida o atributo de cada tabela que será comparado formando a condição como mostrado na Figura 5.11.

No operador Lista de Atributos Manuais a parametrização é feita escolhendo os atributos pertencente dentre as diversas tabelas contidas na base de dados. Essa escolha é feita se escolhendo uma tabela por vez e selecionado o(s) atributo(s) desejado na tabela

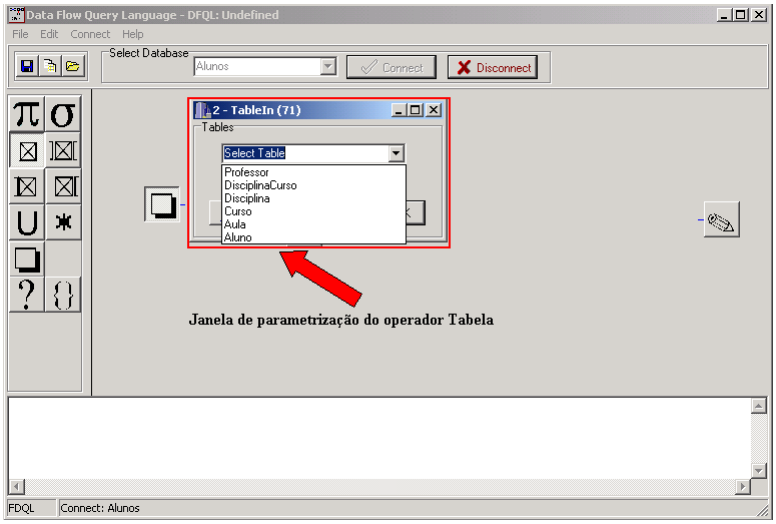


Figura 5.9: Parametrização do operador Tabela

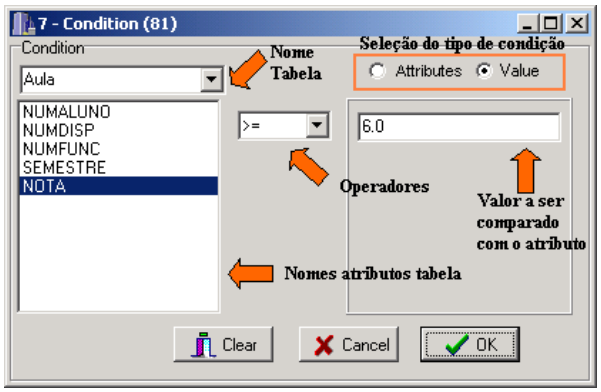


Figura 5.10: Parametrização do operador de condição de seleção

Figura 5.11: Parametrização do operador de condição de junção

e direcionando esse atributo para a lista de atributos escolhidos(*"Selected Attributes"*) através dos botões de "Seleção de Atributos" como mostrado pela Figura 5.12.

Depois do operador ser parametrizado é exibida uma legenda com o conteúdo do operador embaixo do ícone que o representa. Como por exemplo o nome da tabela escolhido no operador Tabela de Entrada como mostrada na Figura 5.13.

O processo de parametrização deve ser feito em todos os operadores de Tabela de Entrada, Condição e Lista de Atributos Manuais existentes no diagrama de fluxo de dados.

5.5 Execução da Consulta

Após o diagrama estar construído como mostrado na 5.14, isto é todos os operadores estão conectados e parametrizados a opção de executar a consulta é habilitada. O pedido de execução de uma consulta é feito através de um clique sobre um operador e escolha da opção *query* no menu exibido conforme visto na Figura 5.15.

Assim, a fase de execução de um diagrama em DFQL gera comandos em SQL que

Figura 5.12: Parametrização do operador Lista de Atributos Manuais

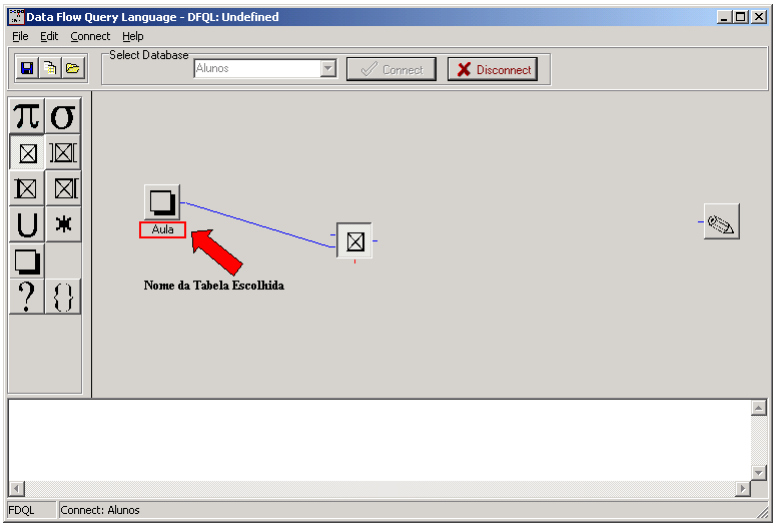


Figura 5.13: Operador Tabela já parametrizado

representam a consulta a ser emitida para cada operação de visualização de resultados e/ou materialização de tabelas intermediárias. Esses comandos são submetidos ao gerenciador de banco de dados e o resultado é mostrado ou gravado de acordo com os parâmetros do operador resultado ou do comando de visualização solicitado como visto na Figura 5.15.

O diagrama de fluxo de dados que representa a consulta também pode ser gravado, isso é feito através da opção *Save/Save as* do menu *File* ou do ícone  mostrado na ferramenta.

5.6 Conclusão

Neste capítulo foi apresentada as funcionalidades da Ferramenta DFQL desenvolvida no projeto de mestrado para apoiar a linguagem de consulta baseada no paradigma de fluxo de dados também desenvolvida no mestrado. Possibilitando assim mostrar o funcionamento da ferramenta.

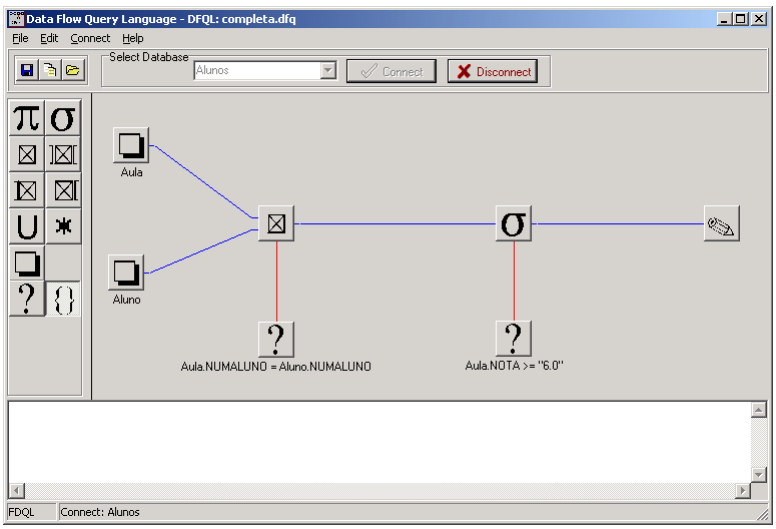


Figura 5.14: Diagrama completo da consulta

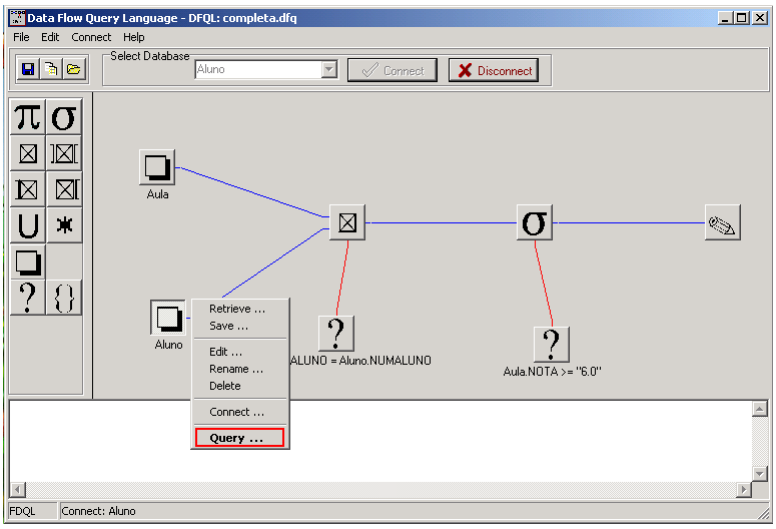


Figura 5.15: Execução completa da consulta

Capítulo

6

Conclusão

Este trabalho desenvolveu uma linguagem de consulta a bancos de dados relacionais utilizando o paradigma de fluxo de dados, e mostra a construção de uma ferramenta que a implementa, permitindo o acesso a uma base de dados relacional.

A utilização do paradigma de fluxo de dados para o acesso aos dados de um gerenciador relacional nunca foi explorada anteriormente, pois o acesso normalmente é feito utilizando o próprio paradigma relacional através da linguagem SQL(e suas variantes), ou através do paradigma de preenchimento de formulários, através da linguagem QBE. Esses dois paradigmas são extremamente adequados para o tratamento dos tipos de dados tradicionalmente suportados pelos gerenciadores relacionais, que são representados em formatos numéricos ou de textos curtos.

No entanto, recentemente tem havido muito esforço de pesquisa e de implementação de recursos a serem adicionados aos gerenciadores relacionais para o suporte a tipos de dados mais complexos, como imagens, audio, séries temporais, dados de estruturas genéticas, etc. A manipulação desses dados em linguagens de acesso centradas puramente em textos e/ou preenchimento de formulários baseados em textos, como é o caso das linguagens SQL e QBE, já se revelou inadequada, mas por enquanto não existem estudos que viabilizem outras linguagens com igual poder expressivo em outros paradigmas.

Este trabalho desenvolveu uma linguagem para acesso a dados armazenados em gerenciadores relacionais que constitui-se em uma terceira alternativa para a expressão das consultas, que tem o mesmo poder expressivo de SQL, baseada no paradigma de fluxo de dados. Essa nova possibilidade de representação de consultas em uma linguagem mais interativa, e mais "visual" pode ser conveniente em algumas aplicações, pois permite uma representação gráfica da consulta enquanto mantém o mesmo poder de representação da álgebra relacional. Além da linguagem de consulta gráfica, este trabalho mostra também

a ferramenta de apoio DFQL, que é um editor/executor de consultas construído para suportar essa linguagem.

6.1 Contribuições do Trabalho

Este trabalho apresenta duas contribuições importantes e inéditas na área de representação de consultas a bases de dados relacionais:

1. Uma linguagem de consulta baseada no paradigma de fluxo de dados;
2. Uma ferramenta que interpreta e executa comandos expressos segundo essa linguagem.

Como parte da primeira contribuição, este trabalho mostrou que qualquer consulta pode ser expressa em diagramas de fluxo que integram quatro tipos de fluxo, como mostrado no Capítulo 4.

Embora esses quatro tipos não sejam necessários para manter a total compatibilidade com a linguagem SQL, o desmembramento de todos os tipos de dados que fluem entre os blocos de processamento de dados nesses quatro tipos permitem que a linguagem possa ser adotada para a representação de outros tipos de processo, tal como processos de data mining. Como um exemplo dessa afirmação, pode-se considerar que o tipo fluxo de atributos poderia ser desnecessário agrupando-se o operador Projeção com o operador Lista de Atributos.

No entanto, um processo de data mining de redução de dimensionalidade por seleção de atributos somente pode ser expresso num diagrama de fluxo de dados que suporte o tipo fluxo de atributos. Assim, a definição dos operadores, tal como apresentada no Capítulo 4, e a definição dos quatro tipos de fluxo de dados, constituem-se também como contribuições deste trabalho.

Como parte da segunda contribuição, existe o fato da ferramenta prover um ambiente interessante para o emprego da DFQL como ferramenta de apoio ao ensino de álgebra relacional em disciplinas básicas de Banco de Dados na graduação.

6.2 Sugestões de Trabalho Futuro

6.2.1 Suporte às operações de agregação

Neste trabalho, a linguagem DFQL foi desenvolvida com os recursos necessários para suportar as operações fundamentais da álgebra relacional, de maneira a constituir-se numa terceira alternativa, junto com SQL e QBE, para a representação de consultas a bases de dados relacionais, dentro do escopo da álgebra relacional. No entanto, tais operações

podem ser estendidas, seguindo o paradigma de fluxo de dados, para suportar as operações de agrupamento e funções de agregados, tal como é suportado pela linguagem SQL atualmente.

Na realidade, embora não tenham sido desenvolvidos os operadores para suportar esse recurso, já existente em SQL, um tipo de fluxo de dados específico para isso já foi previsto na linguagem, que é o Fluxo de Funções. Assim, esta sugestão consiste em desenvolver os operadores adequados para o suporte às operações de agrupamento e funções de agregados, bem como o suporte correspondente na ferramenta.

6.2.2 Suporte a processos de descoberta de conhecimento em bases de dados e mineração de dados

Muitas das ferramentas comerciais de para auxílio à descoberta de conhecimento em bases de dados(KDD) apresentam como forma de representação dos processos de descoberta de conhecimento os diagramas de fluxo de dados. Essas ferramentas partem do pressuposto que as atividades são sempre executadas por processos(de mineração de dados, de limpeza, de preparo de validação de resultados, etc.), e tais como processos em geral, estes podem ser representados por diagramas de fluxo, e no caso, de dados. No entanto, em nenhuma dessas ferramentas existe a preocupação de caracterizar os fluxos de dados em tipos. Dessa maneira, cada processo pode ser ligado a princípio com qualquer dos demais, e cada processo é responsável por validar os dados recebidos, gerando mensagens de erro quando forem solicitados a executar dados recebidos de maneira impropria.

A caracterização dos fluxos em tipos, tal como feito neste trabalho, permite uma avaliação sintática de cada diagrama durante seu projeto, sem a necessidade de execução do fluxo para sua validação. Assim, propõe-se que os processos de KDD sejam analisados como os operadores da álgebra relacional o foram, e que estes sejam definidos como operadores que se somam aos já existentes. Dessa maneira, a linguagem de consulta/descoberta de conhecimento resultante pode se tornar muito mais consistente, do ponto de vista conceitual e, supõe-se, mais poderosa e natural para expressar processos de consulta/descoberta de conhecimento em bases de dados relacionais.

6.2.3 Suporte a atributos de tipo imagem

O desenvolvimento deste trabalho teve como objetivo imediato criar uma nova forma de representação de consultas a bancos de dados relacionais, que mantivesse o mesmo poder expressivo da álgebra relacional, par a par com as linguagens SQL e QBE. A motivação para isso foi a constatação que, embora SQL e QBE sejam adequadas para consultas a dados em formato textual/numérico, a formulação nessas linguagens de consultas envolvendo tipos de atributos mais complexos, principalmente do tipo imagens, envolve uma descontinuidade semântica, pois o uso de imagens como parte dos comandos de consultas obriga a uma quebra do paradigma de representação. Assim, tendo sido atingido o

objetivo de ser possível a representação de consultas numa linguagem que pode ser representada graficamente com o mesmo poder expressivo da álgebra relacional, o próximo passo imediato é a extensão da linguagem para a representação de consultas envolvendo busca por conteúdo de imagens, e particularmente de busca por similaridade e junção por similaridade. Na realidade, essa sugestão é o próximo passo que deve ser seguido para disponibilizar a nova forma de consulta em todo o seu poder de expressão de consultas.

Referências Bibliográficas

[Adler, 2001] Adler, D. W. (2001). Db2 spatial extender - spatial data within the rdbms. In Apers, P. M. G., Atzeni, P., Ceri, S., Paraboschi, S., Ramamohanarao, K., and Snodgrass, R. T., editors, *27th International Conference on Very Large Data Bases*, pages 687–690, Roma, Italy. Morgan Kaufmann.

[Andrade and Saltz, 2000] Andrade, H. C. M. and Saltz, J. (2000). Query optimization in kess 0 an ontology-based kbms. In Becker, K., Souza, A. A. d., Fernandes, D. Y. d. S., and Batista, D. C. F., editors, *XV Brazilian Symposium on Databases*, pages 35–48, Joao Pessoa, PB.

[Angelaccio et al., 1990] Angelaccio, M., Catarci, T., and Santucci, G. (1990). QBD∗: A graphical query language with recursion. *IEEE Transactions on Software Engineering*, 16(10):1150–1163.

[ANSI, 1992] ANSI (1992). American national standard for information systems: database language — SQL: ANSI X3.135-1992. pages xiv + 580. pub-ANSI.

[Astrahan et al., 1976] Astrahan, M. M., Blasgen, M. W., Chamberlin, D. D., Eswaran, K. P., Gray, J., Griffiths, P. P., King, W. F., Lorie, R. A., McJones, P. R., Mehl, J. W., Putzolu, G. R., Traiger, I. L., Wade, B. W., and Watson, V. (1976). System r: Relational approach to database management. *ACM TODS*, 1(2):97–137.

[Atkinson et al., 1989] Atkinson, M., Bancilhon, F., DeWitt, D., Dittrich, K., Maier, D., and Zdonik, S. (1989). The object-oriented database system manifesto. *First International Conference on Deductive and Object-Oriented Databases, Kyoto, Japan.*

[Batini et al., 1991a] Batini, C., Catarci, T., Costabile, M. F., and Levialdi, S. (1991a). Visual query systems: A taxonomy. In *VDB*, pages 153–168.

[Batini et al., 1991b] Batini, C., Catarci, T., Costabile, M. F., and Levialdi, S. (1991b). Visual strategies for querying databases. In *Proceedings of the 1991 IEEE Workshop on Visual Languages*, pages 183–189.

[Becker and Cardoso, 2000] Becker, K. and Cardoso, M. d. O. (2000). Mail-by-example: a visual query interface for managing large volume od electronic messages. In Becker, K., Souza, A. A. d., Fernandes, D. Y. d. S., and Batista, D. C. F., editors, *XV Brazilian Symposium on Databases*, pages 20–34, Joao Pessoa, PB.

[Catarci et al., 1997] Catarci, T., Costabile, M. F., Levialdi, S., and Batini, C. (1997). Visual query systems for databases: A survey. *J. Visual Languages and Computing*, 8(2):215–260.

[Catarci and Santucci, 1994] Catarci, T. and Santucci, G. (1994). Query by diagram: A graphical environment for querying databases - prototype description. In *ACM SIGMOD Conference on Management of Data*, volume 23(2), pages 515–515.

[Chamberlin and Boyce, 1974] Chamberlin, D. D. and Boyce, R. F. (1974). Sequel: A structured english query language. In Rustin, R., editor, *ACM-SIGMOD Workshop on Data Description, Access and Control*, volume 1, pages 249–264, Ann Arbor, MI. ACM Press.

[Chaudhuri et al., 1999] Chaudhuri, S., Fayyad, U. M., and Bernhardt, J. (1999). Scalable classification over sql databases. In *15th IEEE International Conference on Data Engineering - ICDE*, pages 470–479, Sydney, Australia. IEEE Computer Society.

[Chen and Zaniolo, 1999] Chen, C. X. and Zaniolo, C. (1999). Universal temporal extensions for database languages. In *Intl. Conf. on Data Engineering (ICDE)*, pages 428–437, Sydney, Australia. IEEE Computer Society.

[Codd, 1970] Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387.

[Cruz, 1992] Cruz, I. F. (1992). Doodle: A visual language for object-oriented databases. In Stonebraker, M., editor, *Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data, San Diego, California, June 2-5, 1992*, pages 71–80. ACM Press.

[Cruz et al., 1997] Cruz, I. F., Averbuch, M., Lucas, W. T., Radzyminski, M., and Zhang, K. (1997). Delaunay: A database visualization system. In Peckham, J., editor, *ACM Int'l Conference on Data Management (SIGMOD)*, pages 510–513, Tucson, Arizona, USA. ACM Press.

[Darwen and Date, 1995] Darwen, H. and Date, C. J. (1995). The third manifesto. *SIGMOD Record*, 24(1):39–49.

[Date, 2000] Date, C. J. (2000). *Introdução a Sistemas de Banco de Dados - Tradução da 7ª Edição*. Editora Campus, 7 edition.

[Dubois et al., 2001] Dubois, D., Prade, H., and Sédes, F. (2001). Fuzzy logic techniques in multimedia database querying: A preliminary investigation of the potentials. *IEEE Transactions on Knowledge and Data Engineering*, 13(3):383–392.

[Eisenberg and Melton, 1999] Eisenberg, A. and Melton, J. (1999). Sql-1999, formerly known as sql3. *SIGMOD Record*, 28(1):131–138.

[Elmasri and Navathe, 1999] Elmasri, R. and Navathe, S. B. (1999). *Fundamentals of Database Systems*. Benjamin Cummings, third edition.

[Jarke and Vassiliou, 1985] Jarke, M. and Vassiliou, Y. (1985). A framework for choosing a database query language. *ACM Computing Surveys*, 17(3):313–340.

[Jones et al., 2001] Jones, M. B., Berkley, C., Bojilova, J., and Schildhauer, M. (2001). Managing scientific metadata. *IEEE INTERNET COMPUTING*, 5(5):59–68.

[Korth and Silberschatz, 1999] Korth, H. and Silberschatz, A. (1999). *Sistema de Bancos de Dados (3º Edição)*. Makron Books, 3 edition.

[Leavitt, 2000] Leavitt, N. (2000). Whatever happened to object-oriented databases ? *IEEE Computer*, 33(8):16–19.

[Lee et al., 1999] Lee, T., Sheng, L., Bozkaya, T., Balkir, N. H., Özsoyoglu, Z. M., and Özsoyoglu, G. (1999). Querying multimedia presentations based on content. *IEEE Transactions on Knowledge and Data Engineering*, 11(3):361–385.

[Lin and Huang, 2001] Lin, H. and Huang, B. (2001). Sql/sda: A query language for supporting spatial data analysis and its web-based implementation. *IEEE Trans. on Knowledge and Data Engineering*, 13(4):671–682.

[Massari et al., 1995] Massari, A., Pavani, S., Saladini, L., and Chrysanthis, P. K. (1995). Qbi: Query by icons. In Carey, M. J. and Schneider, D. A., editors, *ACM SIGMOD International Conference on Management of Data*, page 477, San Jose, California. ACM Press.

[Melton et al., 2001] Melton, J., Michels, J.-E., Josifovski, V., Kulkarni, K. G., Schwarz, P. M., and Zeidenstein, K. (2001). Sql and management of external data. *SIGMOD Record*, 30(1):70–77.

[Nepal and Ramakrishna, 1999] Nepal, S. and Ramakrishna, M. (1999). Query processing issues in image(multimedia) databases. In *15th IEEE International Conference on Data Engineering - ICDE*, pages 22–29, Sydney, Australia. IEEE Computer Society.

[Netz et al., 2001] Netz, A., Chaudhuri, S., Fayyad, U. M., and Bernhardt, J. (2001). Integrating data mining with sql databases: Ole db for data mining. In *Intl. Conf. on Data Engineering (ICDE)*, pages 379–387, Heidelberg, Germany. IEEE Computer Society.

[O'Neil and O'Neil, 2001] O'Neil, P. and O'Neil, E. (2001). *Database - Principles, Programming and Performance*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers, San Francisco, CA, 2nd edition.

[Papadias and Sellis, 1994] Papadias, D. and Sellis, T. (1994). A pictorial language for the retrieval of spatial relations from image databases. *Sixth International Symposium on Spatial Data Handling*, 1:476–488. Edinburgh, Scotland.

[Sheng et al., 1999] Sheng, L., Özsoyoglu, Z. M., and Özsoyoglu, G. (1999). A graph query language and its query processing. In *15th IEEE International Conference on Data Engineering - ICDE*, pages 572–581, Sydney, Australia. IEEE Computer Society.

[Snodgrass, 1995] Snodgrass, R. T. (1995). *The TSQL2 Temporal Query Language*. Kluwer Academic Publishers.

[Stonebraker and Rowe, 1986] Stonebraker, M. and Rowe, L. A. (1986). The design of postgres. *SIGMOD Conference*, pages 340–355.

[Traina et al., 2000] Traina, Caetano, J., Traina, A. J. M., Wu, L., and Faloutsos, C. (2000). Fast feature selection using fractal dimension. In Medeiros, C. B. and Becker, K., editors, *XV Brazilian Database Symposium*, pages 158–171, João Pessoa - PA - Brazil.

[Zhang et al., 2000] Zhang, C., Meng, W., Wu, Z., and Zhang, Z. (2000). Webssql - a query language for multimedia web documents. In Papazoglou, M. P. and Sheth, A., editors, *IEEE Advances in Digital Libraries 2000 - ADL2000*, page 10p, Washington, D.C. IEEE Computer Society.

[Zloof, 1975] Zloof, M. M. (1975). Query by example. In Press, A., editor, *AFIPS National Computer Conference*, volume 44, pages 431–438, Anaheim, CA.

Este documento foi preparado com o formatador de textos LaTeX no Laboratório do Grupo de Bases de Dados e Imagens - GBDI. O sistema de citações bibliográficas utiliza o padrão *Apalike* do sistema BibTeX.