UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

DEPARTAMENTO DE ENGENHARIA DE SISTEMAS

Sistema de Consultas para a Análise de Dados Cadastrais de Usinas Hidrelétricas.

Por: Ieda Geriberto Hidalgo

Banca Examinadora:

Secundino Soares Filho (orientador)
Cássio Dener Noronha Vinhal
Takaaki Ohishi
Paulo Morelato França

Tese submetida à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, para preenchimento dos pré-requisitos necessários a obtenção do Título de Mestre em Engenharia Elétrica.

Julho 2004

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

Hidalgo, Ieda Geriberto
H53s Sistema de consultas para a análise de dados cadastrais de usinas hidrelétricas / Ieda Geriberto Hidalgo. -- Campinas, SP: [s.n.], 2004.

Orientador: Secundino Soares Filho.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Engenharia de sistemas. 2. Usinas hidrelétricas. 3. Linguagem de consulta (Banco de dados). 4. Banco de dados. 5. Banco de dados relacionais. 6. C++ (Linguagem de programação de computador). 7. Sistemas de consulta e respostas. I. Soares Filho, Secundino. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Resumo

Esse trabalho apresenta o desenvolvimento de uma ferramenta para a criação de consultas avançadas a uma base de dados cadastrais do Sistema Hidrelétrico Brasileiro, denominada HydroData.

Para realizar essas consultas foi desenvolvido um módulo, chamado HydroConsulta, integrado à interface do programa HydroData.

O HydroConsulta pode ser entendido como um agente gerenciador que conecta o usuário à base de dados do sistema HydroData e possibilita a criação de consultas, deixando os detalhes de sintaxe invisíveis para o usuário. O mecanismo interno de execução dessas consultas baseia-se numa linguagem denominada Structured Query Language (SQL).

O projeto de software e a implementação computacional do HydroConsulta utilizaram-se do paradigma da orientação a objetos e da linguagem C++, respectivamente, o que assegura ao sistema uma estruturação moderna e eficiente.

Abstract

This work presents the development of a new toolbox for making advanced search procedures inside a registered data base of the Brazilian Hydroelectric System, denominated HydroData.

To implement such search procedures, it was developed a software module called HydroConsulta, which is integrated with the interface of the HydroData program.

The HydroConsulta program can be understood as a management agent which connects the user with data base content of the HydroData and makes it possible for the user to build advanced searches leaving syntax requisites invisible for the end user. The internal execution mechanism of the search procedure was based on a query language called Structured Query Language (SQL).

The software project of HydroConsulta and its computational implementation use the object-oriented paradigm and C++ language respectively, which assures to the system a modern and efficient structure.

Aos meus pais,

que mantiveram-se firmes, presentes e unidos na educação de seus quatro filhos.

A minha filha Nathalia, pelo carinho e compreensão.

Ao Julio, pelo incentivo de sempre.

"*Não há saber mais ou saber menos: Há saberes diferentes*".

(*Paulo Freire*)

Agradecimentos

Ao meu orientador, Prof. Secundino, pela extrema competência, pelo profissionalismo e por me aceitar como orientanda.

Ao Marcelo, uma pessoa motivadora, competente, de boa índole, querida por todos e dotada de uma didática impressionante, que me auxiliou em absolutamente todos os momentos.

A um grande amigo Tiago Dias, por quem tenho enorme admiração e de quem eu quero amizade eterna.

A Maria de Lourdes (D. Nena), que sempre me ajudou.

Aos meus amigos Nídia, Jéssia, Eliana, Daniela, Dimas, Anivaldo, Hércules, Aldie, Salete, Rebello, Claudete, Márcio, Feital e Henrique (Kiko), que muito bem me orientaram em decisões importantes.

Aos colegas do COSE e DENSIS, sempre solícitos.

Este trabalho teve o apoio da Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES).

Sumário

Capítulo 1

Introdução

Esse capítulo está dividido em cinco etapas. Nas três primeiras, algumas características do sistema elétrico brasileiro de geração são apresentadas com o objetivo de ilustrar o cenário em que esse trabalho está inserido. Na quarta etapa, apresenta-se o programa HydroData cuja interface gráfica e organização interna iniciaram o processo de busca pela qualidade dos dados cadastrais das usinas hidrelétricas. Na quinta etapa, explicita-se a importância do desenvolvimento de uma ferramenta que assegure a confiabilidade desses mesmos dados e apresenta-se a proposta de solução para o problema.

1.1 O Sistema Elétrico Brasileiro

De forma simples, um sistema de energia elétrica pode ser dividido em *meios de produção*, *meios de transporte* e *meios de consumo* da energia elétrica, cuja organização está ilustrada na Figura 1.1.

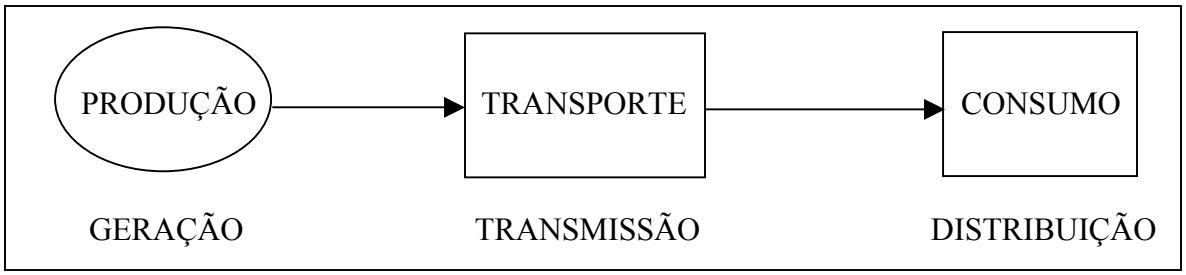


Figura 1.1. Representação simplificada de um sistema de energia elétrica.

Os meios de produção correspondem aos diferentes tipos de equipamentos necessários para a geração de energia elétrica em escala industrial. Dentre esses, são particularmente relevantes as usinas hidrelétricas, onde a energia elétrica é obtida a partir da transformação da energia potencial dos cursos d'água, e as usinas termelétricas, onde a eletricidade resulta da transformação da energia cinética de gases de vapores em expansão, aquecidos pela queima de combustíveis.

Os meios de transporte correspondem às linhas de transmissão e subestações utilizadas para transportar aos consumidores a energia produzida nas usinas.

Os meios de consumo da energia elétrica correspondem ao conjunto das cargas (equipamentos, instalações, etc.) dos consumidores que recebem e utilizam a energia.

Em relação ao Brasil, a capacidade de geração de energia elétrica é predominantemente hidrelétrica, como mostra a Tabela 1.1.

Fonte: Eletrobrás & ANEEL

Tipo	1970	1980	1990	1998	2003	2004
Hidrelétrica	8,7	27,0	44,9*	56,0*	66,3*	66,9*
Termelétrica	1,7	3,7	4,1	5,3	15,8	20,1
Total	10,4	30,7	49,0	61,3	82,1	87,0

(*) Considera somente 50% da capacidade de Itaipu.

Tabela 1.1. Evolução da capacidade instalada [GW] do sistema brasileiro.

Para os dados de potência instalada no sistema brasileiro cabe uma observação. Alguns desses dados consideram apenas 50% da capacidade de Itaipu, sendo essa uma

empresa binacional entre Brasil e Paraguai. Porém, deve-se considerar que o Brasil compra a energia gerada por oito, dentre as nove máquinas paraguaias.

A concentração na produção de eletricidade por fontes hidrelétricas distingue o país dentre as demais nações do mundo, como pode ser verificado na Figura 1.2.

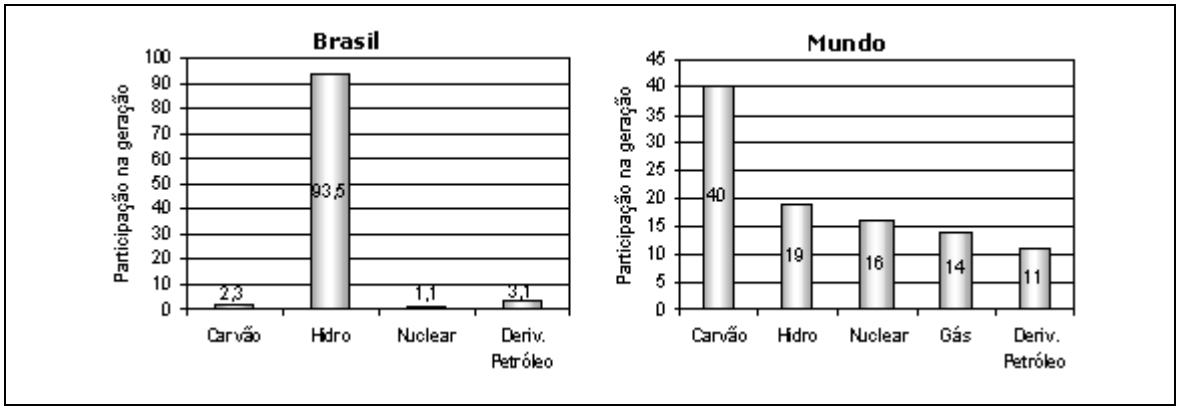


Figura 1.2. Fontes de geração de energia elétrica (Fonte: Eletrobrás, 1998).

De fato, com um grande potencial hidrelétrico, com reservas insuficientes em termos de petróleo e com reservas de carvão mal distribuídas na sua área, o país tem investido intensivamente na geração hidrelétrica. Se por um lado os empreendimentos requerem investimentos de vulto, o custo de geração resultante tem sido compensador em relação às demais alternativas.

A seguir, serão discutidas algumas características do problema de planejamento de um sistema hidrotérmico visando identificar a complexidade de seus componentes.

1.2 A Complexidade do Problema

A exemplo de outros produtos, como a água e gás doméstico, a energia elétrica demanda um intrincado sistema de componentes para fazer chegar aos diferentes consumidores, onde quer que eles estejam e no momento em que o desejarem, o bem produzido nas diferentes fontes de geração. A energia elétrica deve chegar aos consumidores dentro de determinados padrões de quantidade e qualidade de suprimentos, obtidos às custas de um certo investimento no sistema de geração, transmissão e distribuição de energia. Se, por um lado, investimentos insuficientes implicarão na perda de qualidade do produto, por outro lado, o excesso de investimentos resultará em um produto final com custo muito elevado, o que irá desestimular o consumo. Se forem considerados, ainda, os danos causados ao meio ambiente, provocados tanto pelos efluentes tóxicos de usinas termelétricas quanto pelas áreas alagadas de reservatórios de usinas hidrelétricas, conclui-se ser extremamente necessário o estudo de metodologias de planejamento da operação de sistemas de energia elétrica que retirem o máximo de benefícios das unidades geradoras existentes (Fortunato e outros, 1990).

A diminuição do custo de geração de energia envolve a substituição de geração termelétrica por hidrelétrica. Como os recursos hídricos, representados pela água armazenada nos reservatórios são limitados, deve haver um compromisso entre o presente e o futuro. Essas características tornam o problema *dinâmico*, ou seja, o estado de armazenamento dos reservatórios no presente depende de decisões anteriores; e as decisões, no presente, influenciam a geração futura.

O elevado número de aproveitamentos com reservatórios de acumulação e as características de regularização plurianual do sistema brasileiro implicam a adoção de longos períodos de estudo para o planejamento da operação, caracterizando o problema como de *grande porte*.

Outro fator que dificulta o planejamento da operação é o acoplamento operativo entre as usinas pertencentes a uma mesma bacia hidrográfica. Ao contrário do parque termelétrico, no qual todas as usinas são independentes entre si, numa mesma bacia hidrográfica, as hidrelétricas constituem um sistema *interconectado* de geração. O

sistema brasileiro tem como característica grandes bacias com um grande número de usinas.

Além da interdependência operativa entre usinas da mesma bacia hidrográfica, a operação do sistema deve respeitar restrições de uso da água para navegação, irrigação, controle de cheias e saneamento.

As vazões afluentes que chegam aos reservatórios do sistema, dependentes das condições climáticas, são desconhecidas, dando ao problema um caráter *estocástico*.

As funções de geração hidráulica, que descrevem a transformação de água armazenada em energia, e de custo da complementação térmica são representadas por funções *não lineares*.

Além das características acima destacadas do problema de geração, o problema de transmissão da energia gerada nas usinas coloca-se como um aspecto adicional de complexidade do problema de planejamento. A representação do sistema de transmissão é fundamental para assegurar a viabilidade da operação do sistema elétrico. Esta representação, através de modelos de fluxo de potência ótimo AC ou DC, introduz o acoplamento espacial das decisões operativas.

O problema de planejamento da operação é, portanto, um problema de otimização dinâmico, de grande porte, interconectado, estocástico e não linear.

1.3 A Cadeia de Planejamento e Programação

Devido à sua complexidade, o problema de planejamento de um sistema hidrotérmico requer a criação de etapas de planejamento, constituindo o que se chama de *cadeia de planejamento e programação*. Essa divisão do problema em etapas é feita segundo a classificação de características comuns encontradas nas tarefas necessárias à determinação do planejamento global. O principal critério para classificação de etapas baseia-se no tamanho do horizonte de planejamento.

Para exemplificar esse tipo de dependência do problema de planejamento com o horizonte de estudo, pode-se citar o impacto da aleatoriedade das vazões na operação do sistema de grandes reservatórios (sazonalidade). Em se tratando de horizontes anuais, é grande o impacto que a incerteza das vazões tem na operação dos reservatórios, uma vez que a magnitude das vazões define se os reservatórios estarão mais cheios ou vazios ao final do horizonte de planejamento. Já em horizontes de pequena duração, como, por exemplo, uma semana, a aleatoriedade das vazões pode ser deixada de lado, pois é pequeno o impacto das vazões no estado final dos reservatórios.

Apresenta-se a seguir, as etapas da cadeia de planejamento e programação adotadas pelo setor elétrico brasileiro.

- **Planejamento da Operação**: com horizontes de até cinco anos, o objetivo desta etapa é estabelecer o comportamento do sistema para um horizonte de operação de alguns anos à frente. Esta etapa deve promover o aproveitamento racional dos recursos, garantindo-se a qualidade e segurança no atendimento à demanda e a factibilidade das restrições operativas do sistema hidrotérmico.
- **Programação da Operação**: o objetivo principal da programação da operação de curto prazo do sistema hidrotérmico é compatibilizar a operação do sistema hidráulico e elétrico ao longo de horizontes de curto prazo (de alguns dias até uma semana), respeitando as metas energéticas estabelecidas pelo planejamento da operação realizada em horizonte de médio prazo. Devido à sua proximidade da operação em tempo real, na programação da operação requer-se uma representação das restrições elétricas, as quais são ignoradas no planejamento da operação em horizonte de médio ou longo prazo.

Essas etapas da cadeia de planejamento e programação foram implementadas pelo grupo de pesquisa do Laboratório de Sistemas Hidrotérmicos do Departamento de Engenharia de Sistemas (LSH/DENSIS) da Faculdade de Engenharia Elétrica e de Computação da UNICAMP. Na Figura 1.3, faz-se uma representação esquemática da estrutura do sistema de suporte à decisão, denominado HydroLab, que implementa a cadeia de planejamento e programação do grupo. Esse desenho esquemático destaca os modelos constituintes das etapas de planejamento e programação da operação de sistemas hidrotérmicos de geração. O sistema de suporte à decisão HydroLab pode ser entendido como uma reengenharia e extensão do trabalho desenvolvido por Vinhal (1998).

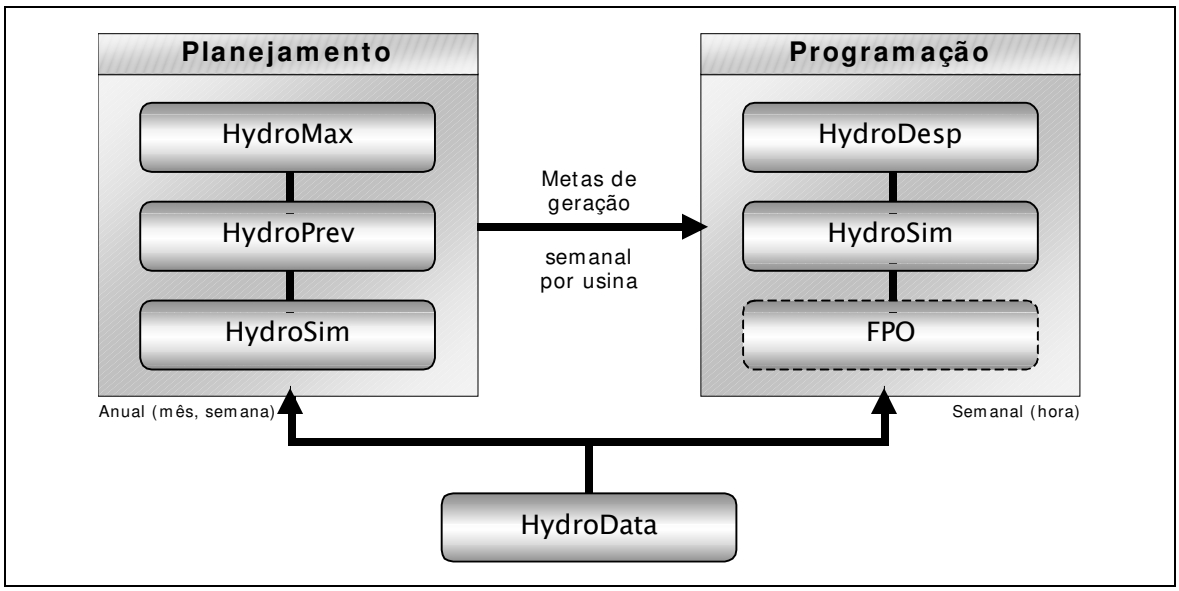


Figura 1.3. Representação esquemática do sistema de suporte à decisão HydroLab.

Como pode ser visto na figura acima, o sistema agrupa modelos em duas categorias principais: planejamento e programação. Estas duas categorias são suportadas por uma base de dados, denominada HydroData, que será detalhada no próximo item. Maiores detalhes sobre o sistema HydroLab podem ser vistos em Cicogna (2003).

1.4 A Base HydroData

Em Cicogna (2003), o autor descreve a implementação e os recursos de um programa computacional gerenciador dos dados cadastrais de usinas hidrelétricas. Esse programa, chamado HydroData, é um aplicativo para computadores que permite a consulta, organização, padronização e gerenciamento dos dados cadastrais das principais usinas hidrelétricas brasileiras.

Os dados cadastrais do HydroData contam atualmente com um total de 104 usinas, englobando o conjunto de usinas sob responsabilidade de operação do Operador Nacional do Sistema Elétrico (ONS) brasileiro.

Apenas para ilustração do sistema, na Figura 1.4 a seguir, apresenta-se a usina hidrelétrica (UHE) de Itaipu selecionada com seus dados gerais visíveis.

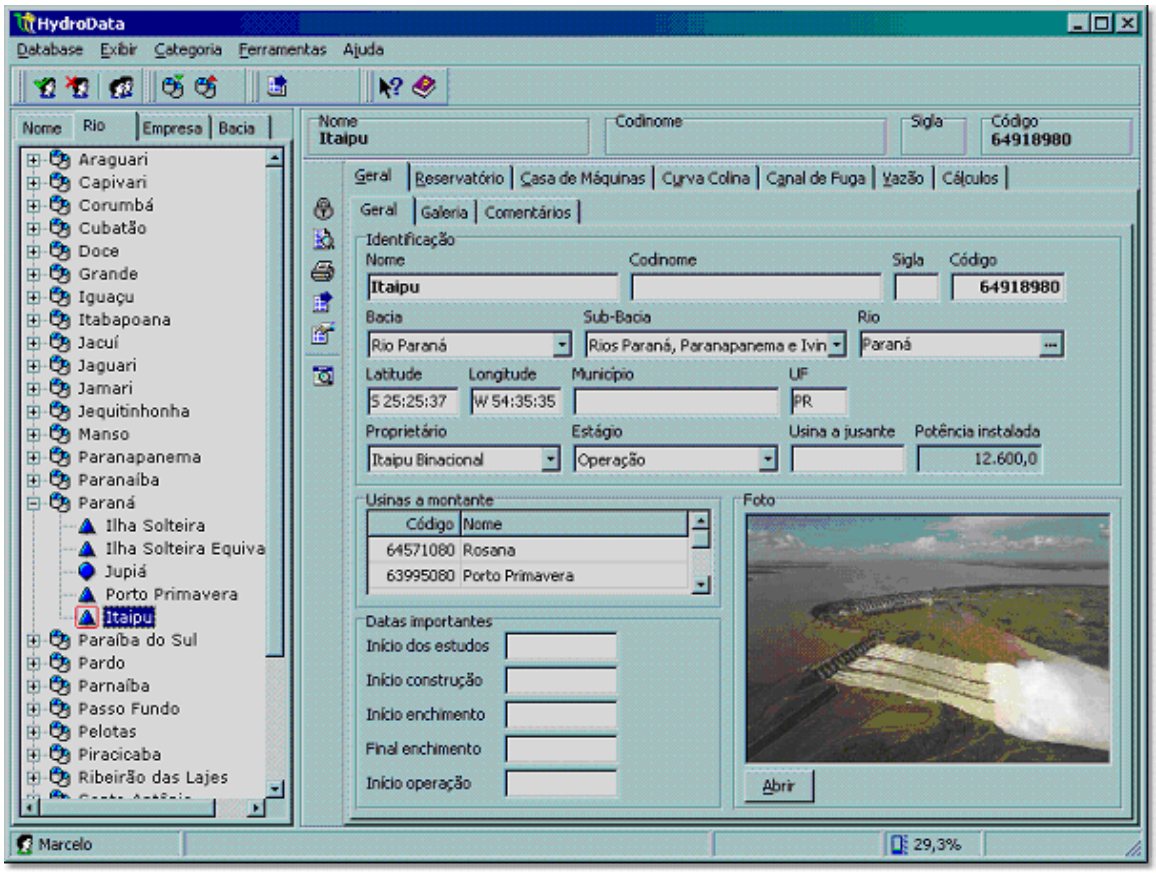


Figura 1.4. A janela principal do programa HydroData.

A janela principal do programa HydroData possui três quadros principais. O *primeiro quadro*, localizado à esquerda, apresenta opções para a organização das usinas.

O usuário pode escolher entre organizar as usinas por nome, rio, empresa ou bacia hidrográfica. Para cada uma dessas categorias, as usinas do banco de dados são apresentadas na forma de uma árvore. No exemplo da Figura 1.4, pode-se ver que os nós principais identificam os nomes dos rios encontrados no banco de dados. Para cada um desses nós, existe um conjunto de nós subordinados. Esses nós indicam o nome das usinas que se localizam no rio em questão.

Como o banco de dados é capaz de armazenar usinas que ainda não estão em operação (pertencentes ao conjunto de empreendimentos estudados no planejamento da expansão do setor elétrico), a legenda para os símbolos colocados no quadro de usinas possui a seguinte descrição:

Símbolo	Descrição
	Usina com reservatório de acumulação. Estágio classificado como em operação.
	Usina com reservatório a fio d'água. Estágio classificado como em operação.
	Usina com reservatório de acumulação. Estágio classificado como em construção ou em fase de projeto básico.
	Usina com reservatório a fio d'água. Estágio classificado como em construção ou em fase de projeto básico.

Tabela 1.2. Legenda para os tipos de usinas no quadro de usinas.

O *segundo grande quadro*, à direita, apresenta os dados da usina atualmente selecionada. Os dados são organizados em categorias, utilizando um recurso visual que lembra um fichário. O usuário pode facilmente mudar de uma categoria para outra, escolhendo uma das abas do fichário (Geral, Reservatório, Casa de Máquinas, Curva Colina, Canal de Fuga, Vazão e Cálculos).

O *terceiro e último quadro* possui o menu principal do programa, uma barra de ferramentas e um cabeçalho preenchido com o nome, codinome, sigla e código da usina atualmente selecionada. Esses dados ajudam a rápida identificação da usina, quando se muda a opção de organização no quadro de dados.

Existe uma barra de ferramentas entre o quadro de usinas e o quadro de dados. Essa barra de ferramentas fornece opções ao usuário, como, por exemplo, relatórios e exportação de dados. Em função da categoria de dados selecionada, a barra de ferramentas modifica-se para atender às necessidades de cada uma das categorias de

dados. Esse comportamento dinâmico da barra de ferramentas é bastante útil para a organização das opções apresentadas ao usuário.

A maioria desses recursos gráficos permite ao usuário do banco de dados uma visão crítica sobre a qualidade das informações nele contidas. Por exemplo, pode-se citar o caso de análise de acuidade dos polinômios de quarto grau que descrevem a relação entre o volume armazenado e a cota do nível d'água do reservatório. Nesse caso, o HydroData apresenta um gráfico com o polinômio grafado entre os limites mínimo e máximo do reservatório, fornecendo ao usuário a possibilidade de aferir a qualidade do perfil do polinômio dentro desses limites. Essa análise gráfica é o recurso mais avançado que o HydroData possui e depende do usuário vasculhar toda a base.

Para o grupo de pesquisa do LSH/DENSIS, o HydroData tem outro papel importante que é servir de base de dados para os modelos de otimização, simulação e previsão de vazões desenvolvidos pelos pesquisadores e gerenciados pelo HydroLab.

1.5 Objetivo

Sistema de Consultas

É sabido, que toda metodologia de planejamento da expansão ou operação de sistemas hidrotérmicos tem forte dependência da qualidade dos dados cadastrais que alimentam modelos matemáticos.

Durante décadas, o Setor Elétrico Brasileiro, em sua divisão do planejamento nas etapas de expansão, operação e programação, replicou dados cadastrais das usinas em várias superintendências de seu órgão gestor, a Eletrobrás. Na presente data, verifica-se uma multiplicação de dados cadastrais, em função de seu uso nas etapas de planejamento, desde a expansão do sistema gerador (longo e médio prazo), até à programação diária da operação de usinas (curto prazo). Essa duplicação de dados, ao longo de anos, ativou a inconsistência dos mesmos, levando à proliferação de informações distintas e dúbias para a descrição de um mesmo dado físico. Em alguns casos nota-se uma forte deterioração da confiabilidade dessas informações cadastrais, como por exemplo, as informações de características físicas de turbinas hidrelétricas.

O cenário de qualidade de informações no setor elétrico possui um recente agravante motivado pelo fato das empresas do setor elétrico não possuírem um procedimento claro e estruturado de informar (via Agência Nacional de Energia Elétrica – ANEEL) ao ONS os dados de suas usinas. Como conseqüência, vê-se que os modelos de planejamento e programação utilizados pelo ONS possuem informações incompatíveis com os dados utilizados pelas empresas.

Os efeitos indesejáveis dessa inconsistência de dados cadastrais são explicitados em implementações computacionais de modelos de otimização, simulação e previsão de vazões utilizados no planejamento e operação do sistema elétrico brasileiro. Por várias vezes, o que explica o comportamento inesperado ou o mau funcionamento desses modelos é a baixa qualidade dos dados fornecidos aos mesmos. Frente a essa realidade, é imperativo que se traga os dados cadastrais das usinas hidrelétricas brasileiras às luzes da consolidação de informações.

O sistema HydroData teve por objetivo resolver o problema de duplicação e gerenciamento dos dados cadastrais das usinas brasileiras. A interface gráfica e a organização interna aumentaram a qualidade dos dados armazenados, favorecendo os pesquisadores do LSH/DENSIS na análise de desempenho de seus modelos. No entanto, novas ferramentas são necessárias para a perfeita consolidação dos dados cadastrais.

A complexidade e dimensão do problema de organização dos dados cadastrais das usinas brasileiras levaram à necessidade de recorrer-se a um tratamento especial do modelo de banco de dados adotado no desenvolvimento do HydroData. Por fim, recorreu-se ao estudo e implementação da teoria de banco de dados relacionais (Elmasri e Navathe, 2000).

A estrutura relacional do banco de dados HydroData pode ser explorada por um sistema de consultas que baseie seus procedimentos em comandos de uma linguagem especialmente desenvolvida para esse fim, denominada "Structured Query Language" – SQL (Bowman, 1996).

Com essa ferramenta de consulta, pode-se determinar respostas a questões importantes para os modelos de otimização e simulação, questões essas diretamente relacionadas à precisão e desempenho de tais modelos. São exemplos dessas consultas:

- Quais usinas brasileiras possuem valores de rendimento médio das turbinas superiores a 90 % no processo de conversão de energia potencial em energia elétrica?
- Quais usinas brasileiras não dispõem de informações sobre a elevação do canal de fuga em função da vazão defluente da usina?

O sistema de consultas desenvolvido nesse trabalho encontra-se integrado ao sistema HydroData, gerenciando e armazenando as consultas na própria base de dados cadastrais das usinas hidrelétricas. Esse sistema possui uma interface gráfica amigável com o objetivo de facilitar ao usuário abstrair as regras e formatos impostos pela linguagem SQL.

A linguagem SQL será detalhada no Capítulo 2. Nesse capítulo o foco principal é dado ao comando SELECT, pois ele é o responsável por compor consultas em um banco de dados relacional.

No Capítulo 3, serão apresentadas as demais ferramentas e conceitos utilizados no desenvolvimento do sistema de consultas, tais como o paradigma da programação orientada a objetos, a linguagem técnica de programação C++ e o ambiente de desenvolvimento C++ Builder.

O modelo relacional e a interface gráfica do sistema de consultas, denominado HydroConsulta, serão descritos no Capítulo 4.

Por fim, no Capítulo 5, faz-se alguns estudos de caso para aferir o desempenho do HydroConsulta na análise dos dados cadastrais das usinas hidrelétricas brasileiras.

Capítulo 2

A Linguagem Estruturada SQL

Embora também seja uma ferramenta utilizada no desenvolvimento do sistema de consultas, devendo fazer parte do Capítulo 3, a SQL merece destaque em um capítulo à parte devido a sua importância nesse trabalho. Inicialmente, são apresentados os conceitos de Banco de Dados e Banco de Dados Relacional. Em seguida são definidos o padrão ANSI SQL e as categorias básicas de comandos utilizados em SQL. O foco principal é dado ao comando SELECT, pois ele é o responsável por compor consultas em um banco de dados relacional (que é o objetivo desse trabalho). Por fim, são detalhados os operadores e as funções que complementam esse comando.

2.1 Banco de Dados

De forma simplificada, um *banco de dados* é um conjunto de informações organizadas disponíveis à manutenção e acesso. É possível pensar em um banco de dados como um mecanismo organizado que tem a capacidade de armazenar informações, que podem ser recuperadas por um usuário de maneira eficiente (Elmasri e Navathe,2000).

Em geral, as pessoas utilizam banco de dados todos os dias. Uma agenda de telefones é um banco de dados. Os dados contidos consistem em nomes, endereços e números de telefones dos indivíduos. As listas são alfabéticas, o que permite ao usuário referenciar um residente local com facilidade.

O método ou mecanismo organizado usado para manter os dados é referido como um sistema de gerenciamento de banco de dados (*Database Management System* – DBMS). Um sistema gerenciador de banco de dados nada mais é do que uma ferramenta computacional que, entre outras tarefas, automatiza a construção e o gerenciamento de um banco de dados a partir de seu modelo. Tais sistemas têm como características principais a criação, alimentação, alteração e manutenção de banco de dados.

Banco de Dados Relacional

Um banco de dados relacional permite dividir os dados em unidades lógicas chamadas tabelas, oferecendo mais fácil manutenção e fornecendo melhor desempenho do banco de dados de acordo com o nível de organização.

Para banco de dados relacionais, o método ou mecanismo organizado usado para manter os dados é referido como um sistema de gerenciamento de banco de dados relacional (*Relational Database Management System* – RDBMS).

Os cabeçalhos das colunas das tabelas de um banco de dados são referenciados como *campos*. Cada campo de uma tabela possui uma informação específica. As linhas das tabelas de um banco e dados são denominadas *registros*, conforme ilustra a Figura 2.1.

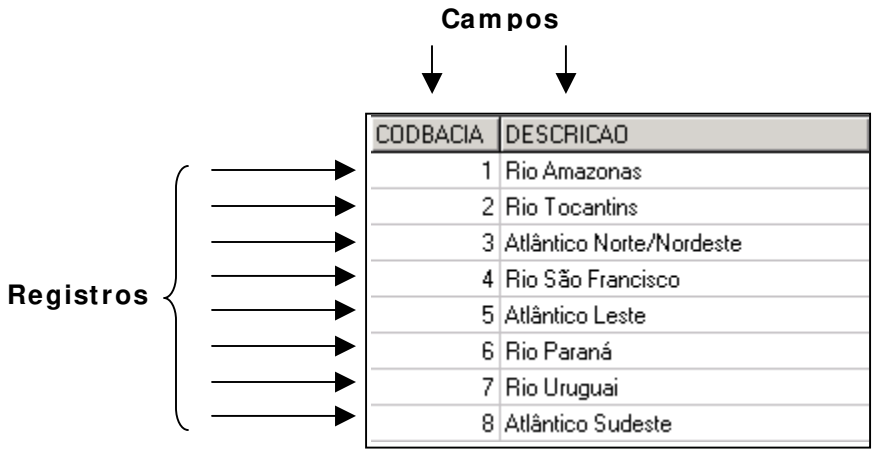


Figura 2.1. Campos e registros em uma tabela.

Em um banco de dados relacional as tabelas são relacionadas entre si por uma chave comum. Tendo-se chaves, ou campos, comuns entre tabelas de banco de dados relacional, pode-se unir dados de múltiplas tabelas para formar um grande conjunto de resultados.

Na Figura 2.2, os campos "CodBacia da tabela Bacia" e "CodSubBacia da tabela SubBacia", pertencentes ao HydroData, *são chaves primárias.* Chave primária é o termo utilizado para identificar um ou mais campos que tornam único um registro. Na mesma figura, o campo "CodBacia da tabela SubBacia" representa uma *chave estrangeira,* pois é utilizado para referenciar uma coluna definida como uma chave primária em outra tabela.

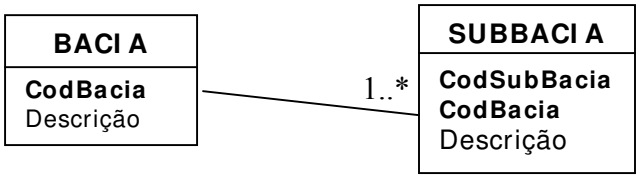


Figura 2.2. Relacionamento entre as tabelas "Bacia" e "SubBacia" do sistema HydroData.

O símbolo "**1..***" na linha que une as duas tabelas da Figura 2.2 identifica o tipo de relacionamento entre elas. Na Tabela 2.1, são apresentados três tipos de relacionamentos.

Tipo	Símbolo	Descrição
Um para um	1..1	Cada registro em uma tabela combina com apenas um registro em outra tabela, e vice-versa.
Um para muitos	1..*	Cada registro em uma tabela combina com um ou mais registros em uma segunda tabela, mas cada registro na segunda tabela combina com apenas um registro da primeira tabela.
Muitos para muitos	*..*	Cada registro em uma tabela combina com vários registros em outra tabela, e vice-versa.

Tabela 2.1. Tipos de relacionamento entre tabelas.

Então, de acordo com a Figura 2.2, cada registro da tabela "Bacia" pode estar relacionado com um ou mais registros da tabela "SubBacia", mas cada registro na tabela "SubBacia"combina com apenas um registro da tabela "Bacia".

Ainda em relação a banco de dados relacionais, existe o conceito de *restrições de integridade referencial*. Estas restrições são utilizadas para assegurar a consistência entre os dados do banco. Por exemplo, não tem sentido excluir uma bacia para a qual existam sub-bacias relacionadas. Deve-se primeiro excluir todas as sub-bacias relacionadas à bacia que será excluída, caso contrário, os dados do banco tornam-se inconsistentes.

2.2 O Padrão ANSI SQL

Structured Query Language (SQL), é a linguagem padrão utilizada para comunicar-se com um banco de dados relacional.

O *American National Standards Institute* (ANSI) é uma organização que aprova certos padrões ao setor industrial mundial. A SQL foi considerada a linguagem padrão na comunicação de banco de dados relacional, originalmente aprovada em 1986 com base na implementação da IBM. Em 1987, o padrão ANSI SQL foi aceito como padrão internacional pela *International Standards Organization* (ISO). O padrão foi revisado novamente em 1992 e chamado SQL/92. O padrão agora é denominado SQL3 ou as vezes referido como SQL/99 (Plew e Stephens, 2000).

Embora exista um padrão ANSI SQL, um fornecedor de banco de dados pode aprimorar o padrão existente adicionando, por exemplo, elementos a este padrão.

O padrão SQL3 tem cinco documentos relacionados:

- **SQL/Framework** – Especifica os requisitos gerais para conformidade e define os conceitos fundamentais de SQL;
- **SQL/Foundation** – Define a sintaxe e as operações de SQL;
- **SQL/Call-Level Interface** – Define a interface para programação de aplicativo para SQL;
- **SQL/Persistent Stored Modules** – Define as estruturas de controle que então definem as rotinas de SQL. (Esta parte também define os módulos que contém as rotinas de SQL);
- **SQL/Host Language Bindings** – Define como embutir instruções de SQL em programas de aplicativo que são escritos em uma linguagem de programação padrão.

Com qualquer padrão vem numerosas vantagens óbvias. Um padrão conduz os fornecedores na direção correta para desenvolvimento da indústria; no caso da SQL, isso fornece um esqueleto básico de fundamentos necessários que, como um resultado final, permite a consistência entre várias implementações e melhor atende à necessidade de portabilidade. O padrão SQL3 exige recursos que devem estar disponíveis em qualquer

implementação de SQL completa e destaca conceitos básicos que não apenas forçam a consistência entre todas as implementações competitivas de SQL, mas também aumentam o valor de um programador de SQL ou usuário de banco de dados relacional no mercado de banco de dados atual.

2.3 Categorias Básicas de Comandos Utilizados em SQL

A seguir serão apresentadas as categorias básicas de comandos utilizados em SQL para realizar várias funções, tais como, construir e reestruturar um banco de dados, preencher tabelas, atualizar os dados existentes em tabelas, excluir dados, realizar consultas de banco de dados, controlar o acesso a dados e fazer a administração geral do banco.

Essas categorias são denominadas *DDL (Data Definition Language)*, *DML (Data Manipulation Language)*, *DQL (Data Query Language)*, *DCL (Data Control Language)*, *Comandos de administração de dados* e *Comandos de controle transacional,* detalhadas a seguir.

2.3.1 DDL (Data Definition Language)

Essa é a categoria da linguagem SQL que permite a um usuário criar e reestruturar um banco de dados. Os principais comandos de DDL são:

- CREATE TABLE (cria uma tabela)
- ALTER TABLE (modifica uma tabela)
- DROP TABLE (exclui uma tabela)
- CREATE INDEX (cria um índice)
- ALTER INDEX (modifica um índice)
- DROP INDEX (exclui um índice)

2.3.2 DML (Data Manipulation Language)

Essa categoria da linguagem SQL é utilizada para manipular dados de tabelas de um banco de dados relacional. Entende-se por manipular dados a ação de inserir, atualizar ou excluir registros de tabelas. Seus comandos básicos são:

- INSERT (insere dados em uma tabela)
- UPDATE (atualiza os dados de uma tabela)
- DELETE (exclui dados de uma tabela)

2.3.3 DQL (Data Query Language)

Embora compreendido de apenas um comando, a Data Query Language (DQL), é o principal foco da linguagem SQL para usuários de banco de dados relacional. Esse comando, acompanhado por várias opções e cláusulas, é utilizado para compor consultas para esse tipo de banco de dados. Portanto, esse comando será detalhado logo adiante.

- SELECT (constrói consultas em banco de dados relacionais)

2.3.4 DCL (Data Control Language)

Os comandos de controle de dados em SQL permitem o controle do acesso a dados dentro do banco de dados. Esses comandos de DCL são normalmente utilizados para criar objetos relacionados com acesso de usuário e também para controlar a distribuição de privilégios entre os mesmos. Alguns comandos de controle de dados são:

- ALTER PASSWORD (altera senha)
- GRANT (atribui permissões)
- REVOKE (remove permissões)
- CREATE SYNONYM (reduz nomes longos através de sinônimos)
- DROP SYNONYM (exclui sinônimos)

2.3.5 Comandos de Administração de Dados

Os comandos de administração de dados permitem ao usuário realizar auditorias e análises em operações dentro do banco de dados. Também podem ser utilizados para ajudar a analisar o desempenho do sistema. A seguir, são apresentados dois comandos gerais de administração de dados:

- START AUDIT (inicia auditoria)
- STOP AUDIT (interrompe auditoria)

2.3.6 Comandos de Controle Transacional

Além das categorias de comandos previamente apresentadas, há comandos que permitem ao usuário gerenciar *transações* de banco de dados. Uma transação é, por exemplo, caracterizada por uma alteração qualquer nos dados do banco. Seus comandos são:

- COMMIT (utilizado para salvar transações de banco de dados)
- ROLLBACK (utilizado para desfazer transações de banco de dados)
- SAVEPOINT (cria pontos dentro de grupos de transações ate que eles desfaçam transações com rollback)
- SET TRANSATION (atribui um nome a uma transação)

A Figura 2.3 ilustra como as alterações são aplicadas em um banco de dados relacional.

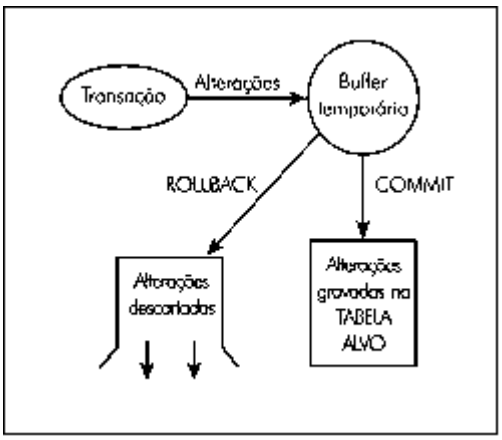


Figura 2.3. Comandos de controle transacional.

2.4 Consultas

Uma consulta é utilizada para extrair dados do banco de dados em um formato legível de acordo com a solicitação do usuário. A instrução utilizada para construir consultas de banco de dados é a SELECT. Ela não é uma instrução independente, o que significa que cláusulas são requeridas.

Há quatro palavras chave, ou cláusulas, que são parte valiosa de uma instrução SELECT. Essas palavras-chave são as seguintes:

```
SELECT (elemento requerido)
FROM (elemento requerido)
WHERE (cláusula opcional)
ORDER BY (cláusula opcional)
```

A sintaxe completa para a instrução SELECT é a que segue:

```
SELECT [ALL | * | DISTINCT coluna1, colunaN] [alias-name]
FROM tabela1 [,tabela2, tabelaN]
WHERE [condição1 | expressão1] [AND condiçãoN | expressãoN]
ORDER BY [coluna1, colunaN | inteiro] [ASC | DESC]
```

A explicação de cada uma das palavras-chave será ilustrada com exemplos aplicados à tabela denominada “Bacia” da base de dados HydroData. O objetivo destes exemplos é facilitar o entendimento de toda a teoria dessa seção. Portanto, são exemplos didáticos e não soluções para problemas reais.

2.4.1 SELECT / FROM

A instrução SELECT é utilizada em conjunção com a cláusula FROM. A palavra-chave SELECT, em uma consulta, é seguida por uma lista de campos os quais serão

exibidos no resultado da consulta. A palavra-chave FROM é seguida por uma lista de tabelas que contem os campos que farão parte da saída da consulta. O asterisco (*) é utilizado para denotar que todas as colunas em uma tabela devem ser exibidas como parte da saída da consulta. A opção ALL é utilizada para exibir todos os valores para uma coluna, incluindo duplicatas. Esta opção é padrão, portanto, não precisa ser especificada. A opção DISTINCT é utilizada para eliminar linhas duplicadas. Os campos que se seguem ao comando SELECT são separados por vírgulas, assim como o é a lista de tabelas que se seguem ao FROM.

- **Listar todos os campos da tabela "Bacia".**

```
SELECT
  BACIA.CODBACIA, BACIA.DESCRICAO
FROM
  BACIA

------------------------- O U --------------------------

SELECT
   *
FROM
  BACIA
```

	CODBACIA	DESCRICAO
▶	1	Rio Amazonas
	2	Rio Tocantins
	3	Atlântico Norte/Nordeste
	4	Rio São Francisco
	5	Atlântico Leste
	6	Rio Paraná
	7	Rio Uruguai
	8	Atlântico Sudeste
	9	Outras

- **Listar apenas o campo "Descrição" da tabela "Bacia".**

```
SELECT
  BACIA.DESCRICAO
FROM
  BACIA
```

	DESCRICAO
▶	Rio Amazonas
	Rio Tocantins
	Atlântico Norte/Nordeste
	Rio São Francisco
	Atlântico Leste
	Rio Paraná
	Rio Uruguai
	Atlântico Sudeste
	Outras

2.4.2 Aliases de Colunas

Os *aliases* de coluna são utilizados para personalizar nomes de cabeçalhos de colunas. Quando uma coluna é renomeada em uma instrução SELECT, o nome não é uma alteração permanente. A alteração aplica-se apenas a essa instrução em particular.

- **Listar todos os campos da tabela "Bacia" alterando o nome das colunas "CodBacia" e "Descrição" para "Bacia" e "Nome" respectivamente.**

```
SELECT
  BACIA.CODBACIA Bacia , BACIA.DESCRICAO Nome
FROM
  BACIA
```

	BACIA	NOME
▶	1	Rio Amazonas
	2	Rio Tocantins
	3	Atlântico Norte/Nordeste
	4	Rio São Francisco
	5	Atlântico Leste
	6	Rio Paraná
	7	Rio Uruguai
	8	Atlântico Sudeste
	9	Outras

2.4.3 WHERE

A cláusula WHERE é utilizada para impor condições a uma consulta eliminando algumas linhas que normalmente seriam retornadas por uma consulta sem restrições.

Pode haver mais de uma condição na cláusula WHERE. Se houver mais de uma condição, elas são conectadas pelos operadores AND e OR.

- **Listar todos os campos da tabela "Bacia" alterando o nome das colunas "CodBacia" e "Descrição" para "Bacia" e "Nome" respectivamente. Aplicar, também, o filtro campo "CodBacia" maior que dois e menor que oito.**

```
SELECT
  BACIA.CODBACIA Bacia , BACIA.DESCRICAO Nome
FROM
  BACIA
WHERE
  BACIA.CODBACIA > 2  AND
  BACIA.CODBACIA < 8
```

	BACIA	NOME
▶	3	Atlântico Norte/Nordeste
	4	Rio São Francisco
	5	Atlântico Leste
	6	Rio Paraná
	7	Rio Uruguai

2.4.4 ORDER BY

Em geral, é interessante que a saída da consulta tenha algum tipo de ordem. Os dados podem ser classificados utilizando-se a cláusula ORDER BY. Esta cláusula organiza os resultados de uma consulta em um formato de listagem especificado pelo usuário. A ordem padrão da cláusula ORDER BY é uma ordem crescente. Para ordens decrescentes deve-se acrescentar a palavra DESC. Uma coluna listada na cláusula ORDER BY pode ser abreviada com um inteiro. O inteiro é uma substituição para o nome real de coluna, identificando a posição da coluna depois da palavra-chave SELECT. Este recurso de substituição do nome da coluna pelo valor inteiro não funciona com SELECT *.

- **Listar todos os campos da tabela "Bacia" alterando o nome das colunas "CodBacia" e "Descrição" para "Bacia" e "Nome" respectivamente. Aplicar, também, o filtro campo "CodBacia" maior que dois e menor que oito e organizar os registros em ordem ascendente do campo "Descrição".**

```
SELECT
  BACIA.CODBACIA Bacia , BACIA.DESCRICAO Nome
FROM
  BACIA
WHERE
  BACIA.CODBACIA > 2  AND
  BACIA.CODBACIA < 8
ORDER BY
  BACIA.DESCRICAO ASC
```

---------- O U ----------

```
SELECT
  BACIA.CODBACIA Bacia , BACIA.DESCRICAO Nome
FROM
  BACIA
WHERE
  BACIA.CODBACIA > 2  AND
  BACIA.CODBACIA < 8
ORDER BY
   2 ASC
```

BACIA	NOME
5	Atlântico Leste
3	Atlântico Norte/Nordeste
6	Rio Paraná
4	Rio São Francisco
7	Rio Uruguai

Em geral, para as palavras-chave da linguagem SQL não é feito distinção entre maiúsculas e minúsculas. O mesmo acontece para nomes de tabelas e campos do banco de dados. No entanto, em relação aos dados do banco é necessário utilizar em consultas cadeias de caracteres que correspondam à maneira como as informações foram armazenadas. Diante disso, conclui-se que os dados não são consistentes se forem arbitrariamente inseridos utilizando letras maiúsculas e minúsculas de modo aleatório:

- SOUZA
- Souza
- souza

É importante ressaltar que embora as colunas seguintes à palavra SELECT, WHERE ou ORDER BY possam ser referenciadas apenas pelo nome do campo, optou-se no trabalho por compô-las pelo nome da tabela agregado ao nome do campo ambos separados por um ponto final. As duas formas são aceitas pelo padrão ANSI SQL3, mas com a segunda consegue-se maior confiabilidade na saída produzida pelo comando.

2.5. Operadores

Um operador é uma palavra reservada ou um caractere utilizado principalmente na cláusula WHERE de uma instrução SQL, embora, também possa aparecer na cláusula SELECT. Os operadores podem ser de *comparação*, *lógico*, *conjuntivo* ou *aritmético*.

2.5.1 Operadores de Comparação

Os operadores de comparação são símbolos utilizados para testar valores únicos na cláusula WHERE de uma instrução SQL. Na Tabela 2.2, são listados esses operadores.

Símbolo	Descrição
=	igualdade
<>	não igualdade
<	valores menores que
>	valores maiores que
<=	valores menores que ou igual
>=	valores maiores que ou igual

Tabela 2.2. Operadores de Comparação.

2.5.2 Operadores Lógicos

Os operadores lógicos são aqueles que fazem comparações, também na cláusula WHERE, através de palavras-chave da linguagem SQL. Essas palavras-chave são: *ISNULL*, *BETWEEN*, *IN* e *LIKE*, descritas na Tabela 2.3.

Palavra-chave	Descrição
IS NULL	Compara um valor com um valor NULL
BETWEEN	Procura valores que estão dentro de um conjunto de valores, dado o valor mínimo e o valor máximo. Between é inclusivo, portanto inclui os valores mínimos e máximos na consulta
IN	Compara um valor com uma lista de valores literais que foi especificada
LIKE	Compara um valor com valores semelhantes utilizando operadores curinga

Tabela 2.3. Operadores Lógicos.

Há dois operadores curingas utilizados em conjunção com o operador LIKE:

- % (o sinal de porcentagem). Representa zero, um ou múltiplos caracteres.
- _ (o sublinhado). Representa um único número ou caractere.

Abaixo são listados sete exemplos usando os operadores curingas "%" (sinal de porcentagem) e "_" (sublinhado):

LIKE '200%'	(localiza quaisquer valores que iniciam com 200).
LIKE '%200%'	(localiza quaisquer valores que tem 200 em qualquer posição).
LIKE '_00%'	(localiza quaisquer valores que têm "0" na segunda e terceira posições).
LIKE '2_%_%'	(localiza quaisquer valores que iniciam com 2 e tem pelo menos três caracteres de comprimento) .
LIKE '%2'	(localiza quaisquer valores que terminam com 2).
LIKE '_2%3'	(localiza quaisquer valores que têm um 2 na segunda posição e terminam com 3).
LIKE '2___3'	(localiza quaisquer valores em um número de cinco dígitos que iniciam com 2 e terminam com 3).

Negando Condições com o Operador NOT

O operador NOT inverte o significado do operador lógico com quem ele é utilizado. O NOT pode ser utilizado com os seguintes operadores nos seguintes métodos:

- IS NOT NULL
- NOT BETWEEN
- NOT IN
- NOT LIKE

2.5.3 Operadores Conjuntivos

Através dos operadores conjuntivos torna-se possível combinar duas ou mais condições. Esses operadores são *And* e *Or*, apresentados na Tabela 2.4.

Operador	Descrição
AND	Todas as condições devem ser verdadeiras
OR	Pelo menos uma das condições deve ser verdadeira

Tabela 2.4. Operadores Conjuntivos.

2.5.4 Operadores Aritméticos

Os operadores aritméticos são utilizados para realizar funções matemáticas em instruções SQL. Podem ser usados nas cláusulas SELECT e/ou WHERE. Há quatro operadores convencionais para funções matemáticas: *adição*, *subtração*, *multiplicação* ou *divisão*, descritos na Tabela 2.5.

Os operadores matemáticos podem ser utilizados em combinações entre eles. Como nas regras de precedência da matemática básica, as operações de multiplicação e divisão são realizadas primeiro, e depois as operações de adição e subtração. A única maneira de o usuário controlar a ordem das operações matemáticas é por meio do uso de parênteses. Os parênteses que envolvem uma expressão fazem com que essa expressão seja avaliada como um bloco.

Símbolo	Descrição
+	adição
-	subtração
*	multiplicação
/	divisão

Tabela 2.5. Operadores Aritméticos.

2.6 Funções

Uma função é utilizada para fornecer informações de resumo para uma instrução SQL, como somas, médias, valores máximos e mínimos. As funções que podem ser usados no comando SQL são: *COUNT*, *SUM*, *AVG*, *MAX* e *MIN*.

Nos exemplos a seguir, para o cálculo de potência instalada a unidade de medida será kw, conforme armazenadas no HydroData.

2.6.1 COUNT

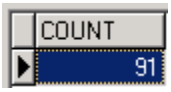
A função COUNT é utilizada para contar linhas ou valores de uma coluna. O retorno desta função é um valor numérico. Ela pode ser usada com o asterisco (*). COUNT, quando utilizado com asterisco, conta todas as linhas de uma tabela incluindo duplicatas e valores nulos. Quando a função COUNT é utilizada com o comando DISTINCT, apenas as linhas distintas são contadas. ALL (oposta de DISTINCT) é o padrão, não é necessário incluí-la na sintaxe. Como a função COUNT conta as linhas, os tipos de dados não entram em cena. As linhas podem conter uma coluna com qualquer tipo de dado.

A sintaxe para a função COUNT é como segue:
COUNT [(*) | (DISTINCT | ALL)](nome_da_coluna)

A seguir são listadas três maneiras diferentes de usar a função COUNT.

- **Contar todas as linhas da tabela “Usina” que contenham um valor para o campo “CodJus”.**

```
SELECT
  COUNT (USINA.CODJUS)
FROM
  USINA
```



- **Contar somente as linhas distintas do campo “CodJus” da tabela “Usina”.**

```
SELECT
  COUNT(DISTINCT USINA.CODJUS)
FROM
  USINA
```

COUNT
78

- **Contar todas as linhas da tabela “Usina” incluindo duplicatas e valores nulos.**

```
SELECT
  COUNT (*)
FROM
  USINA
```

COUNT
103

2.6.2 SUM

A função SUM é utilizada para retornar o total dos valores de uma coluna. Esta função também pode ser utilizada em conjunção com DISTINCT. Quando SUM é utilizada com DISTINCT, somente as linhas distintas são somadas, o que pode não ter muito sentido. Esta função só pode ser utilizada em colunas que armazenem valores numéricos.

A sintaxe para a função SUM é como segue: SUM ([DISTINCI] nome_da_coluna)

Abaixo, um exemplo da função SUM.

- **Somar todos os valores da potência instalada.**

```
SELECT
  SUM(CONJTG.PEFE* CONJTG.NTURB)
FROM
  CONJTG
```

SUM
100306405

2.6.3 AVG

A função AVG é utilizada para retornar a média dos valores de uma coluna. Quando utilizada com o comando DISTINCT, a função AVG retorna a média das linhas distintas. O valor do argumento deve ser numérico para a função AVG trabalhar.

A sintaxe para a função AVG é como segue:

AVG ([DISTINCT] nome_da_coluna)

Abaixo, um exemplo da função AVG.

- **Calcular a média aritmética dos valores da potência instalada.**

```
SELECT
  AVG(CONJTG.PEFE* CONJTG.NTURB)
FROM
  CONJTG
```

2.6.4 MAX

A função MAX é utilizada para retornar o valor máximo para os valores de uma coluna. Os valores NULL são ignorados. O comando DISTINCT é uma opção. Entretanto, como o valor máximo para todas as linhas é o mesmo que o valor máximo distinto, ele é inútil. Esta função é valida para campos que contenham valores numéricos, data ou caracter.

A sintaxe é como segue:

MAX ([DISTINCT] nome_da_coluna)

Abaixo, um exemplo da função MAX.

- **Apresentar o maior valor da potência instalada.**

```
SELECT
  MAX(CONJTG.PEFE* CONJTG.NTURB)
FROM
  CONJTG
```

2.6.5 MIN

A função MIN retorna o valor mínimo de uma coluna. Os valores NULL são ignorados. O comando DISTINCT é uma opção. Entretanto, como o valor mínimo para todas as linhas é o mesmo que o valor mínimo distinto, ele é inútil. Esta função é valida para campos que contenham valores numéricos, data ou caracter.

A sintaxe é como segue: MIN ([DISTINCT] nome_da_coluna)

Abaixo, um exemplo da função MIN.

- **Apresentar o menor valor da potência instalada.**

```
SELECT
  MIN(CONJTG.PEFE* CONJTG.NTURB)
FROM
  CONJTG
```

Capítulo 3

Desenvolvimento do Sistema de Consultas

O objetivo desse capítulo é apresentar ao leitor as ferramentas e os conceitos utilizados no desenvolvimento do sistema de consultas. Inicialmente, será explorada a teoria do paradigma da orientação a objetos. A fase seguinte apresenta a linguagem técnica de programação C++ que é uma das mais completas para o tratamento de problemas de engenharia. Por fim, serão demonstrados os fundamentos do ambiente C++ Builder que possui um grande número de recursos e facilidades para a implementação de aplicativos, principalmente quando se visa seguir o padrão do sistema operacional *Windows*.

3.1 O Paradigma da Programação Orientada por Objetos

A programação orientada por objetos é uma técnica de programação, ou seja, um estilo para a escrita de programas que tem por finalidade a solução de um conjunto de problemas. Paradigma é o conjunto de regras e idéias que norteiam o pensamento das pessoas durante uma determinada época. Paradigmas da programação podem ser entendidos como maneiras de se pensar e representar a solução computacional de um problema. O termo "linguagem de programação orientada por objetos" especifica uma linguagem de programação que fornece mecanismos para o suporte do estilo de programação que utiliza objetos na solução de um problema (Cicogna, 1999).

É dito que uma linguagem suporta um estilo de programação se essa fornece facilidades para tornar esse estilo conveniente (fácil de usar, seguro e eficiente), como também, diz-se que uma linguagem não suporta uma técnica de programação se a aplicação do estilo resultar em um grande esforço na implementação dos programas. Por exemplo, é possível escrever programas orientados por objetos utilizando-se a linguagem C, mas o esforço para representar tais paradigmas nestas linguagens é grande o suficiente para desencorajar o mais dedicado dos programadores.

A principal diferença entre a Programação Procedural e a Programação Orientada a Objetos está na forma pela qual os dados e procedimentos intercomunicam-se (Schildt, 2002).

Na Análise Procedural, a principal ênfase é dada aos procedimentos. Esses são implementados em blocos estruturados e a comunicação entre os mesmos se dá pela passagem de dados. Em outras palavras, os dados são processados dentro de blocos e migram de um para outro. Um programa pertencente a este paradigma, quando em execução, é caracterizado pelo acionamento de procedimentos cuja tarefa é a manipulação de dados.

Na Análise Orientada a Objetos, os dados e os procedimentos fazem parte de um só elemento, os objetos. Esses elementos, ao estabelecerem comunicação entre si, caracterizam a execução do programa. Assim, ao contrário da filosofia procedural, onde dados e procedimentos são entidades dissociadas, no paradigma da orientação a objetos os dados e procedimentos estão encapsulados em um só elemento.

A Figura 3.1 mostra esquematicamente as diferenças entre os paradigmas procedural e orientado a objetos.

Figura 3.1. Paradigma procedural x Orientado a objetos

Os pontos principais sobre os quais fundamentam-se as vantagens da programação orientada a objetos sobre a programação procedural são: aplicabilidade encontrada em uma variedade muito grande de problemas, o maior índice de reaproveitamento de código, a maior facilidade de manutenção de sistemas e, em geral, o menor código gerado pelos programas orientados a objetos, quando comparados aos programas estruturados.

3.1.1 Mecanismos da Análise Orientada a Objetos

O conceito fundamental de "o*bjeto"* significa uma entidade cujas informações podem incluir desde suas características até os procedimentos para manipulação dessas características. A um objeto estarão sempre associados seu *estado*, *comportamento* e *identidade*. O estado é definido pelas propriedades do objeto e pelos valores que elas possuem. O comportamento define como o objeto age e reage em relação a mudanças em

seu estado e à comunicação com outros objetos. A identidade de um objeto é a propriedade pela qual ele se distingue dos demais.

Quando objetos possuem características semelhantes, pode-se agrupá-los em uma "*classe*". A classe para a qual são definidos os dados e procedimentos associados a todos os objetos da classe denomina-se c*lasse_objeto*. Algumas classes são generalizações e não possuem objetos sendo denominadas apenas como c*lasse*.

Atributos e serviços associados a uma classe ou a uma classe_objeto são chamados de *membros da classe*. O primeiro tipo de membro de uma classe são os atributos (também chamado de dados, propriedades ou características). Eles descrevem as propriedades do objeto. A manipulação dos atributos é efetivada pelos membros denominados serviços, as vezes referenciados como métodos, procedimentos, funções ou operações, associados à classe.

Quando o atributo deve ser distinto para cada objeto, ele caracteriza um a*tributo de objeto*. Estes podem ter, até mesmo, valores idênticos, mas são entidades separadas no sistema. Além destes, uma classe pode possuir atributos cujo valor deve ser compartilhado entre todos os seus objetos. A estes, dá-se o nome de a*tributos de classe*. Quando um objeto modifica o valor de um atributo de classe, todos os demais objetos da classe vêem o valor modificado.

Ao definir uma classe, é necessário que se especifique o tipo de acesso aos seus membros. *Acesso público* permite que todas as outras entidades do sistema (funções e demais classes) acessem este membro. *Acesso protegido* permite que as classes na hierarquia da classe em questão possam acessar os membros sob essa especificação. Finalmente, os membros que só devem ser acessados pelos serviços da classe possuem *acesso privado*.

Quando um programa orientado a objetos está sendo executado, ocorre um processo de comunicação entre os objetos, através do envio de *mensagens* (chamadas de funções ou procedimentos). Estas mensagens definem o comportamento do objeto receptor. Para alcançar este objetivo, o objeto aciona aqueles serviços relacionados à mensagem enviada. Todo o processamento é definido pelo envio, pela interpretação e pelas respostas às mensagens entre objetos.

3.1.2 Conceitos-Chave da Análise Orientada a Objetos

Dois conceitos, essenciais no paradigma da orientação a objetos, foram utilizados no desenvolvimento do sistema de consultas. São eles: *encapsulamento* e *herança*.

Encapsulamento

Uma vez abstraídos as classes e objetos do problema, tanto ao nível de atributos como ao nível de serviços necessários, há a necessidade de protegê-los na sua forma em uma mesma entidade. À propriedade de se implementar dados e procedimentos correlacionados em uma mesma entidade (objeto) dá-se o nome de encapsulamento. Encapsulamento é, então, um mecanismo de programação que une atributos e serviços e que mantém ambos seguros contra uso indevido e alteração externa.

Herança

Apenas a definição de entidades (classes), que modelam tipos semelhantes (objetos), e a implementação destas não abrange o potencial mais significativo da orientação a objetos. É necessário considerar as relações entre classes, de modo a permitir o compartilhamento de dados e procedimentos semelhantes. Isto é conseguido através da herança. Esta é fruto de um mecanismo de hierarquia entre classes, onde uma classe mais especializada (classe-filha) herda as propriedades da classe mais geral (classe-mãe), a que ela está imediatamente subordinada na hierarquia. A classe mais geral é denominada superclasse e a classe mais especializada é denominada subclasse.

Três classes desenvolvidas nesse trabalho foram utilizadas com frequência no sistema de consultas. São elas: TFrameAddCampos, TFrameTreeCampos e TFrameListCampos. Estas classes podem ser visualizadas em tempo de projeto na Figura 3.2.

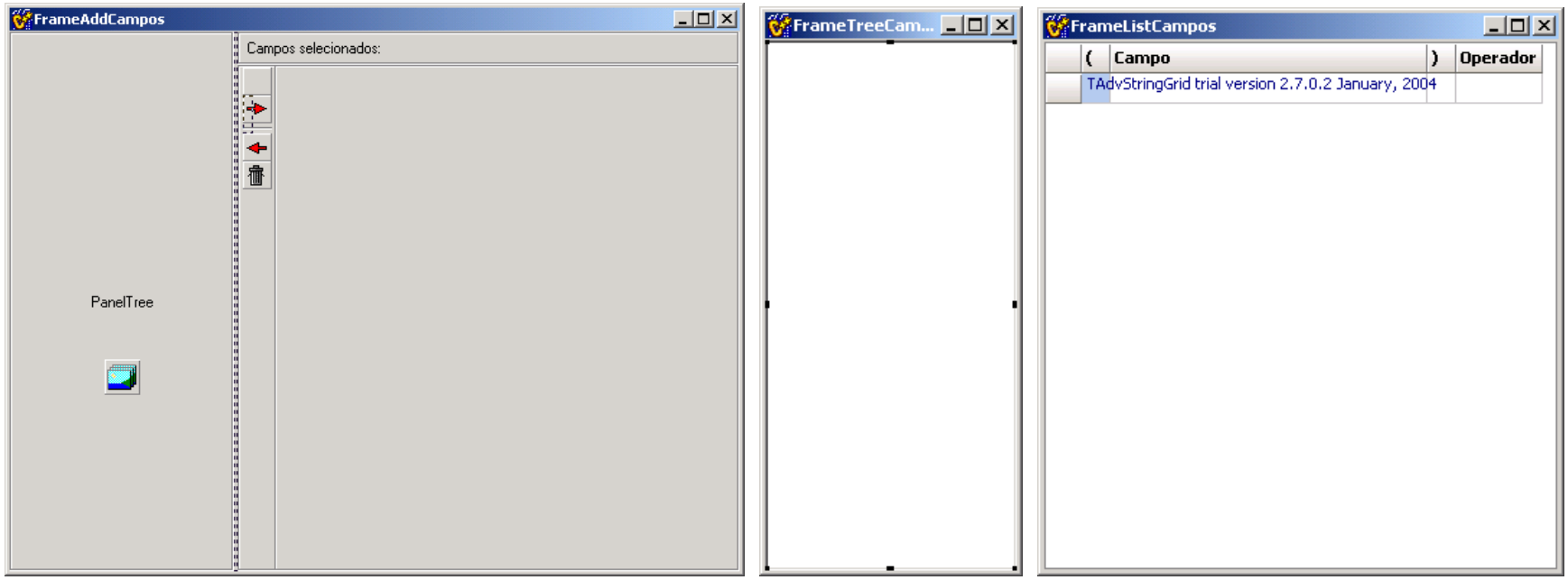


Figura 3.2. Frames utilizados no desenvolvimento do sistema de consultas (em tempo de projeto).

Em tempo de execução, a combinação dessas janelas (chamadas *frames*) aparece ao usuário como um elemento que permite adicionar/remover campos das tabelas do HydroData, conforme mostra a Figura 3.3.

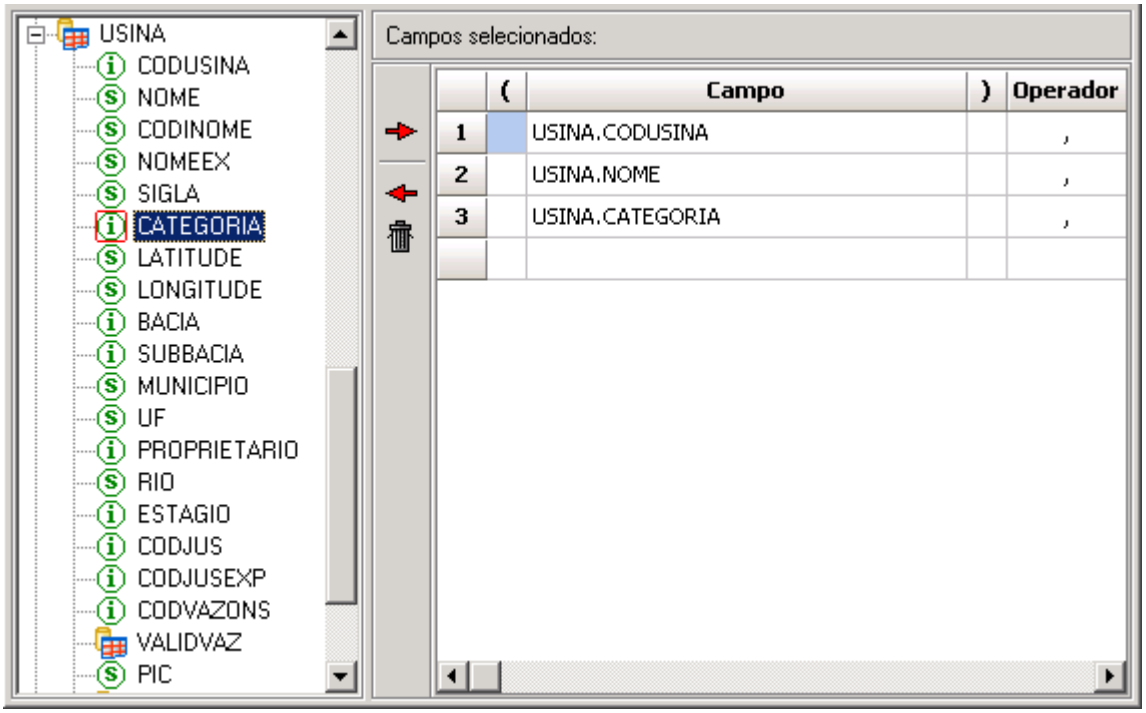


Figura 3.3. Frames utilizados no desenvolvimento do sistema de consultas (em tempo de execução).

Os recursos do paradigma da programação orientada são suportados pela *linguagem de programação C++* apresentada no próximo item.

3.2 A Linguagem Técnica de Programação C++

Historicamente, a metodologia de desenvolvimento de programas iniciou-se na década de 40 (Montenegro e Pacheco, 1994). Nessa época não existiam linguagens de programação e todos os procedimentos eram implementados diretamente ao nível de máquina, através da alteração física de circuitos eletrônicos.

Com o objetivo de se aproximar cada vez mais do problema a ser resolvido e se distanciando dos detalhes de hardware as linguagens de programação foram evoluindo. Assim, as chamadas linguagens de *primeira geração*, surgiram na segunda metade da década de 50. Nessa fase, os computadores eram programados diretamente em linguagem de máquina através de cadeias de 0`s e 1`s. A Figura 3.4, apresenta um exemplo de código escrito nessa geração.

```
00000100    10000001    /* Carrega 128
00000000    10000001    /* Soma 121
00000101    00001001    /* Armazena 128
00001111    00000000    /* Interrompe
```

Figura 3.4. Exemplo de código escrito numa linguagem da primeira geração.

Nas chamadas linguagens da *segunda geração*, incluem-se as linguagens FORTRAN e ASSEMBLY. Nessa geração, era permitido o uso de abreviações para denotar as operações de interesse. Observa-se, na Figura 3.5 o mesmo exemplo da Figura 3.4 agora codificado numa linguagem da segunda geração.

```
LOD 128     /* Carrega 128
ADD 121     /* Soma 121
STO 128     /* Armazena 128
HLT         /* Interrompe
```

Figura 3.5. Código da Figura 3.4 escrito numa linguagem da segunda geração.

Seguindo a evolução das linguagens, surgiram, durante a década de 60, as chamadas linguagens de *terceira geração*. Entre elas incluem-se linguagens como C e PASCAL. Esta última, desenvolvida por Niklaus Wirth teve grande impacto no meio acadêmico pelo seu caráter didático e estruturado. As linguagens de programação desta

geração são chamadas de *linguagens de alto nível*. Elas eliminam a necessidade dos programadores entenderem detalhes de como os computadores processam os dados. A Figura 3.6, mostra o mesmo exemplo da Figura 3.4 agora codificado numa linguagem de terceira geração.

```
X = X + 121
```

Figura 3.6. Código da Figura 3.4 escrito numa linguagem da terceira geração.

Apesar dessa evolução nas linguagens computacionais, ao final da década de 60 a indústria de software amargou profunda crise, causada pelos baixíssimos níveis de produtividade. O índice de reaproveitamento de código era ínfimo. A principal causa era a falta de uma metodologia que formalizasse o desenvolvimento de sistemas.

Até então, as linguagens eram *procedurais*, ou seja, os sistemas eram compostos por subprogramas. Essa decomposição dos programas força o desenvolvedor a fixar a atenção nos procedimentos, e não nos dados. É justamente esse desequilíbrio de importância uma das principais fontes de crítica a esse tipo de linguagem.

Surgem então as "mães" das linguagens orientadas a objetos, a linguagem *Simula* (ainda na década de 60) e a linguagem *Smalltalk* (na década de 70). O despertar do interesse da comunidade da Computação pela programação orientada a objetos deve-se, em parte, à aceitação destas duas linguagens. Ambas demonstraram o poder da modelagem baseada em classes, onde dados e procedimentos estão formalizados em uma só entidade. Houve o reconhecimento, principalmente no meio acadêmico, de que se poderia poupar esforço de implementação e de que objetos poderiam ser pré-programados e reaproveitados. No caso de *Smalltalk*, a filosofia do ambiente *Windows* que a linguagem proporcionava, indicava o potencial deste tipo de interface.

Apesar disso, durante a década de 80 pode-se dizer que a adoção dos princípios da programação orientada a objetos pela indústria ficou abaixo do esperado. Vários fatores contribuíram para que isto ocorresse. As linguagens: *Simula* e *Smalltalk* ficaram praticamente confinadas aos meios acadêmicos e aos grupos que as criaram. Além desse confinamento, em algumas das utilizações, a programação orientada a objetos com

Smalltalk era encarada mais como técnica de programação por janelas do que como um novo paradigma de programação.

Na primeira metade dos anos 80, a posição da linguagem C (procedural) firmou-se no mercado. Esta linguagem é o resultado de um processo de desenvolvimento que começou com uma linguagem mais antiga, chamada de BCPL, desenvolvida por Martin Richards, que influenciou uma linguagem chamada B, inventada por Ken Thompson. Uma das grandes virtudes de C é a portabilidade dos códigos-fonte escritos. Isso significa que um programa escrito em C, para um determinado equipamento ou sistema operacional, pode ser compilado num equipamento diferente, rodando outro sistema operacional, sem necessidade de muitas alterações. Esta linguagem unia outras vantagens, era muito poderosa, flexível, seus códigos eram legíveis, compactos e de execução rápida .

A linguagem C++ é uma evolução natural da linguagem C e foi inicialmente chamado de *C com classes*. Durante a primeira metade dos anos 80, Bjarne Stroustrup, nos laboratórios Bell (AT & T), desenvolvia o C++ com o objetivo de integrar os poderes da programação orientada a objetos presentes em Simula ao C, ou em suas palavras: "C++ foi projetado como uma imitação deliberada e confessa da abordagem de Simula usando a mais popular linguagem de programação de sistemas, C, como base" (Bjarne Stroustrup, 1992).

Atualmente, o poder de C++ como linguagem profissional de implementação de programas orientados a objetos pode ser medido, por exemplo, pelos investimentos de empresas importantes (exemplos: *Borland* e *Microsoft*) no sentido de dominarem o mercado da programação C++.

A linguagem C++ foi, então, escolhida para implementar o sistema de consultas desse trabalho. Ela fornece mecanismos modernos e muito eficientes para o suporte da orientação a objetos descrito no próximo item.

Apenas para ilustração, na Figura 3.7, pode-se visualizar uma parte do código de programação escrito na linguagem C++ extraído do sistema de consultas desenvolvido nesse trabalho. Esse código monta a instrução SQL responsável pela execução efetiva da consulta. Ele faz parte das mais de duas mil linhas de código necessárias ao

funcionamento do sistema em questão. Além dessas duas mil linhas implementadas, muitas outras foram geradas automaticamente pela linguagem.

```
AnsiString
TDModSearchMan::MontaSQL(int CodConsulta)
{
    LocateConsulta(CodConsulta);
    AnsiString Campos = ListaPorExtenso(QuerySelectCodLista->AsInteger);
    //  ---------------
    VTabelas.clear();
    TabelasDistinct(QuerySelectCodLista->AsInteger);
    LocateCondicoes(CodConsulta);
    QueryCondicoes->First();
    while (!QueryCondicoes->Eof)
    {
        TabelasDistinct(QueryCondicoesCodLista->AsInteger);
        QueryCondicoes->Next();
    }
    AnsiString Tabelas;
    for (int i = 0; i < VTabelas.size(); i++)
        Tabelas += VTabelas[i] + ", ";
    Tabelas = Tabelas.SubString(1,Tabelas.Length()-2);
    //  ---------------
    AnsiString Condicoes = CondicoesPorExtenso(CodConsulta);
    AnsiString Ordem = OrdemPorExtenso();
    AnsiString SQL;
    SQL = "SELECT\r\n  " + Campos + "\r\n";
    SQL = SQL + "FROM\r\n  " + Tabelas;
    if (Condicoes !="")
        SQL = SQL + "\r\nWHERE\r\n  " + Condicoes;
    if (Ordem !="")
        SQL = SQL + "\r\nORDER BY\r\n  " + Ordem;
    return (SQL);
}
//---------------------------------------------------------------------
```

Figura 3.7. Código escrito na linguagem C++, responsável por montar a instrução SQL do sistema de consulta.

O sistema de consultas desenvolvido nesse trabalho foi implementado num ambiente integrado de desenvolvimento chamado *C++ Builder* que será descrito no próximo item.

3.3 O Ambiente de Desenvolvimento C++ Builder

O ambiente de desenvolvimento *C++ Builder* foi criado pela *Borland*. Ela ingressou no mercado de compiladores C em meados da década de 80, com o lançamento do *Turbo C*. O Turbo C possuía um ambiente de desenvolvimento integrado, a exemplo do *Turbo Pascal*, que na época encontrava-se na versão 4.0.

Em 1990 a Borland dá mais um passo lançando o *Turbo C++*. A partir de então, o surgimento do *C++ Builder* seria apenas questão de tempo.

O C++ Builder é uma ferramenta para desenvolvimento de aplicações que se baseia no conceito denominado *RAD* ("Rapid Application Development" – Desenvolvimento Rápido de Aplicação). Isso significa que é possível criar toda a interface de um aplicativo de forma visual, apenas adicionando objetos gráficos aos formulários, como botões, caixas de edição, legendas, caixas de combinação, etc (Alves, 2002).

Após a definição da interface, torna-se necessária, a inclusão de códigos de programação responsáveis pela lógica e processamento das informações do usuário.

Sendo o C++ Builder um ambiente que possibilita o desenvolvimento de sistemas orientado por objetos, é possível criar códigos que serão reutilizados posteriormente em varias aplicações. Por exemplo, suponha-se que tenha sido desenvolvida uma rotina de impressão de relatórios. Essa rotina pode ser transformada num componente e incorporada à paleta do C++ Builder. Quando precisar de uma rotina de impressão num aplicativo, basta inserir o componente que executa esta tarefa.

Pode-se encontrar uma vasta quantidade de bibliotecas de componentes que expandem os recursos existentes no C++ Builder, muitas delas a partir da Internet. Desta maneira, foi encontrado um componente chamado *TAdvStringGrid* utilizado no sistema de consultas desenvolvido nesse trabalho. Os detalhes a respeito desse componente serão vistos no Capítulo 4.

A Figura 3.8 mostra o ambiente de desenvolvimento do C++ Builder 5.

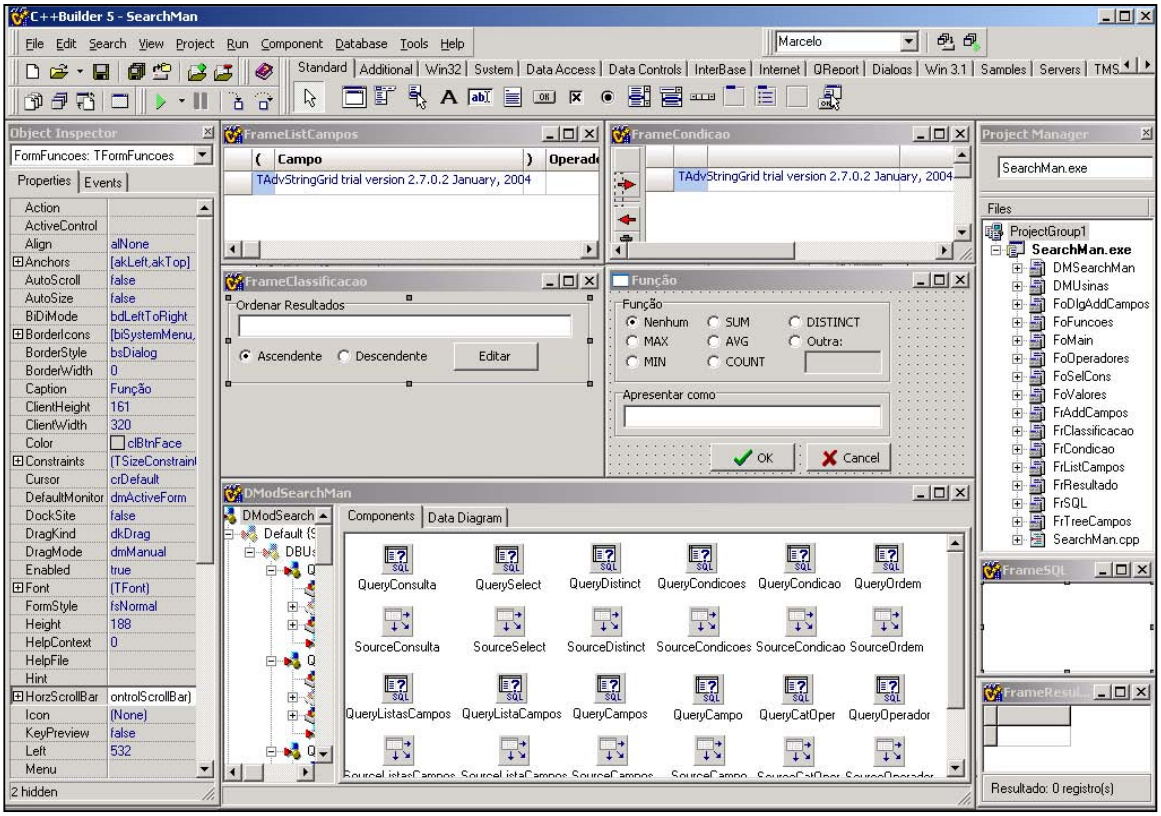


Figura 3.8. Ambiente C++ Builder 5.

Uma aplicação em C++ Builder é construída em torno de um projeto que contém diversos formulários, e estes, por sua vez, estão ligados a unidades de códigos que são arquivos-fonte com extensão ".cpp". Sendo assim, formulários e seus componentes são a essência e a parte visível de uma aplicação, responsáveis pela interface com o usuário final.

Os componentes de um formulário podem ser objetos gráficos visíveis (como um botão, uma caixa de lista, um menu, etc) que fazem parte da interface, ou invisíveis (como o temporizador, controle de acesso à base de dados, etc). Todo componente C++ Builder possui *propriedades* e *eventos*.

As propriedades permitem que o componente tenha sua aparência modificada, como a cor de um botão. Já os eventos são utilizados na construção de rotinas que devem ser executadas de acordo com as ações do usuário. Por exemplo, um botão deve ter um código a ser executado quando o usuário clicar nele.

Cada formulário criado em C++ Builder é formado por dois arquivos que trabalham juntos: um arquivo com a definição da aparência do formulário e dos objetos inseridos nele, gravado com a extensão ".dfm"; e um arquivo de unidade de código com o mesmo nome do formulário, mas com a extensão ".cpp". Essa unidade contém eventos e funções necessários à execução das tarefas a que o aplicativo se destina.

Durante o desenvolvimento de uma aplicação, vários arquivos são gerados. Cada um tem sua parcela de importância. A Tabela 3.1 relaciona o significado de cada arquivo.

Extensão do arquivo	Descrição
.BPR	Arquivo de projeto do C++ Builder. Nele estão referenciados todos os formulários, unidades de código de formulários e unidades de código independentes.
.DFM	Arquivo de formulário. Esse arquivo contém, na forma binária ou textual, as informações referentes ao visual do formulário (controles e suas propriedades).
.CPP	Arquivo de unidade. Esse arquivo contém as funções implementadas pelo programador em C++. Cada arquivo ".dfm" contém um arquivo ".cpp" associado. Se forem criadas unidades independentes de formulários, elas também receberão essa extensão.
.OBJ	Arquivo-objeto de unidades compiladas (formulários e unidades de código).
.H	Arquivo de cabeçalho, que são unidades de códigos com definições especiais.
.DSK	Arquivo de configurações de opções do ambiente de trabalho do C++ Builder, ajustadas pelo menu Tools/Enviroment Options.
.RES	Arquivo de recursos, que contém ícones, imagens bitmap, cursores, etc. associados ao projeto.
.~DFM, .~CPP, .~H	Arquivos de reserva gerados quando os originais sofrem alterações e são regravados.

Tabela 3.1. Tipos de arquivos que fazem parte de um aplicativo desenvolvido em C++.

Quando um projeto é compilado, o C++ Builder gera um arquivo-objeto a partir dos formulários e das unidades de código. Se o projeto fizer uso de uma biblioteca externa ou de outros arquivos ".cpp", eles também serão incluídos na compilação.

A Figura 3.9 mostra todo o processo envolvido na geração de um arquivo executável. Todos os formulários e unidades de códigos são primeiramente compilados para gerar um arquivo com o mesmo nome e extensão ".obj". O segundo passo resume-se à linkedição (ligação) de todos esses arquivos com a biblioteca do C++ Builder, outros

arquivos-objeto ou unidades criadas pelo programador e que sejam necessários. Isso resulta num arquivo executável com o mesmo nome do arquivo de projeto.

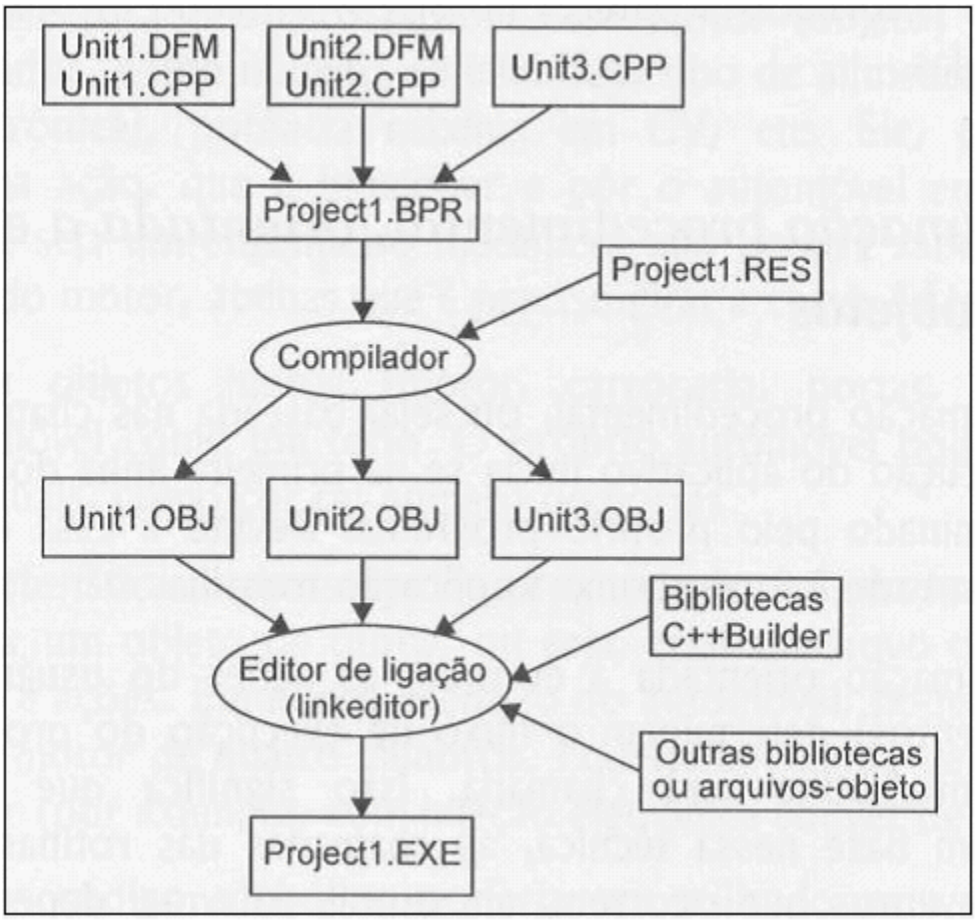


Figura 3.9. Processo envolvido na geração de um arquivo executável.

Capítulo 4

O HydroConsulta

Nesse capítulo descreve-se o sistema computacional desenvolvido para a consulta dos dados hidrelétricos brasileiros, em um nível mais avançado em relação à versão atual do sistema HydroData. Esse programa é chamado HydroConsulta e será apresentado em duas etapas. Na primeira, explora-se o modelo relacional criado para seus dados. Na segunda etapa, pretende-se que o leitor conheça as abas que o compõem. É importante ressaltar que sua interface segue o padrão do *Windows* isso caracteriza-o como um sistema fácil de usar.

4.1 Modelo Relacional

Embora a função do HydroConsulta seja consultar uma base de dados existente (o HydroData), seus dados também devem permanecer armazenados num banco. Dessa maneira, torna-se possível restaurar as informações de uma consulta sem que haja necessidade de especificá-las novamente.

Para criar um modelo de banco de dados para o HydroConsulta foi analisado o comando SELECT e suas cláusulas FROM, WHERE e ORDER BY para uma consulta completa.

Para assegurar ao banco uma estrutura eficiente, um bom desempenho e facilidade de acesso aos dados optou-se pelo modelo relacional, apresentado no Capítulo 2.

A teoria de banco de dados relacional aplicada diretamente aos resultados da fase de levantamento de informações necessárias a uma consulta, resultou num modelo relacional para os dados do HydroConsulta que pode ser visualizado em um diagrama de entidades (tabelas) e relacionamentos, conforme a Figura 4.1. As entidades são desenhadas como quadros, tendo em seu conteúdo uma descrição dos elementos que compõem sua estrutura. Essa descrição apresenta os tipos de dados que cada propriedade possui.

Para conseguir que tanto os dados relacionados às usinas quanto os dados relacionados às consultas ficassem num único banco, optou-se por adicionar o modelo relacional do HydroConsulta ao banco de dados do HydroData. Esse procedimento facilita o transporte de informações.

Para distinguir, no banco de dados, as tabelas do HydroData das tabelas do HydroConsulta, essas últimas iniciam seus nomes com a sigla SM (*SearchMan*) isso pode ser observado também na Figura 4.1.

De acordo com o modelo relacional da Figura 4.1, percebe-se que a tabela SMConsulta armazena dados relacionados à consulta e se relaciona com outras três tabelas: SMSelect, SMCondicao e SMOrdem. Cada uma destas três tabelas está preparada para receber informações referentes às cláusulas SELECT, WHERE e ORDER BY, respectivamente. A tabela SMConsulta se relaciona também com a tabela SMListaCampos. Esta tabela guarda todas as listas de campos que se seguem aos

comandos SELECT, WHERE e ORDER BY para todas as consultas. Para identificar qual lista pertence a qual comando, cada uma das três tabelas relacionadas à SMConsulta possui um campo chamado CodLista. A tabela SMListaCampos obtém as informações dos campos da lista através da tabela SMCampos.

Existem ainda outras três tabelas identificadas neste modelo. A tabela SMPrimaryKey que apenas gera a chave primária de cada uma das tabelas citadas acima e as tabelas SMCatOper e SMOperador. Essas duas últimas têm conteúdo fixo. Elas armazenam os operadores de comparação, lógicos, conjuntivos e aritméticos que podem ser utilizados nas cláusulas SELECT e WHERE de um comando SQL.

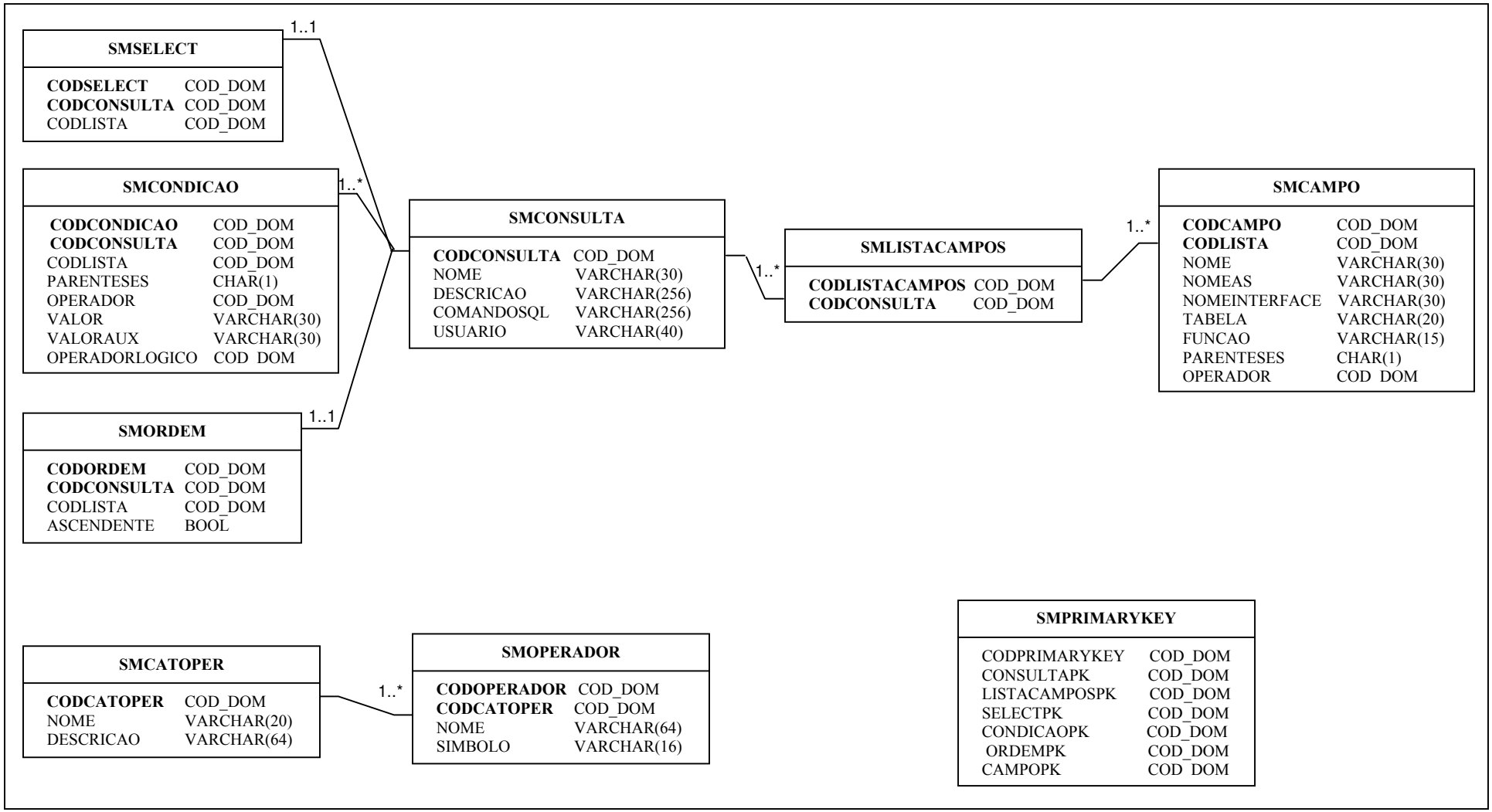


Figura 4.1. Modelo relacional do banco de dados do HydroConsulta.

4.1.1 Implementação do Modelo Relacional

Para implementar o modelo relacional do HydroConsulta utilizou-se o mesmo sistema de gerenciamento de banco de dados relacional do HydroData. O RDBMS utilizado foi o *Borland Interbase*. Esse sistema encontra-se na classe de sistemas gerenciadores com recursos *multi-plataforma* e *multi-usuário*. Isso significa que ele permite criar um banco de dados que seja acessado em sistemas operacionais diferentes e por vários usuários ao mesmo tempo. O sistema permite o gerenciamento dos usuários e seus privilégios de acesso.

Na implementação do modelo relacional é necessária a criação do arquivo que conterá todo o modelo das entidades e relacionamentos. Por tratar-se de uma linguagem estruturada, é possível criar tipos de dados definidos pelo usuário (domínios). Foram aproveitados os tipos criados, por motivos de simplificação, pelo HydroData, conforme Figura 4.2.

```
/* Domínios */
CREATE DOMAIN COD_DOM AS INTEGER;
CREATE DOMAIN WORD AS SMALLINT;
CREATE DOMAIN BOOL AS SMALLINT DEFAULT 0 CHECK (VALUE IN (0,1));
```

Figura 4.2. Exemplo da criação de domínios, via comandos SQL.

Na figura a seguir, vê-se um exemplo onde é criada a tabela SMCampo. Esses comandos criam colunas nas tabelas utilizando os nomes das propriedades listadas no modelo, bem como os tipos, aceitos pelo sistema RDBMS. As relações entre as entidades são criadas com as especificações de chaves primárias e chaves estrangeiras, apresentadas no Capítulo 2.

```
CREATE TABLE SMCAMPO (
  CODCAMPO            COD_DOM NOT NULL,
  CODLISTA            COD_DOM NOT NULL,
  NOME                VARCHAR(30),
  NOMEAS              VARCHAR(30),
  NOMEINTERFACE       VARCHAR(30),
  TABELA              VARCHAR(20),
  FUNCAO              VARCHAR(15),
  PARENTESES          CHAR(1),
  OPERADOR            COD_DOM NOT NULL,
  POSICAO             COD_DOM NOT NULL
  PRIMARY KEY (CODCAMPO),
  FOREIGN KEY (CODLISTA) REFERENCES SMLISTACAMPOS(CODLISTACAMPOS)
)
```

Figura 4.3. Exemplo da criação da tabela SMCAMPO, via comandos SQL.

Esse tipo de construção foi repetido para todas as tabelas identificadas no modelo. O resultado é um arquivo texto com instruções SQL capazes de criar a estrutura relacional da Figura 4.1. Uma vez que a estrutura é detalhada a partir de comandos SQL, dentro do padrão ANSI, essas instruções podem ser processadas por outros sistemas RDBMS, dando ao trabalho a característica de portabilidade.

Algumas tabelas têm conteúdo fixo, e por isso podem ser preenchidas no momento em que são criadas. Embora os dados possam ser fornecidos às tabelas através de uma interface gráfica do Interbase, essa tarefa foi executada de outra maneira, pois a linguagem SQL permite adicionar dados na forma de comandos. Essa técnica foi utilizada no preenchimento das tabelas SMCatOper e SMOperadores, conforme as Figuras 4.4 e 4.5.

```
/* Tabela SMCatOper (Categorias de Operadores) ---------*/
INSERT INTO SMCatOper (CodCatOper, Nome, Descricao) values
(0, 'Vírgula', 'Separador de lista');
INSERT INTO SMCatOper (CodCatOper, Nome, Descricao) values
(1, 'Aritimético', 'Operações matemáticas de aritimética');
INSERT INTO SMCatOper (CodCatOper, Nome, Descricao) values
(2, 'Comparação', 'Operações de comparação');
INSERT INTO SMCatOper (CodCatOper, Nome, Descricao) values
(3, 'Lógico', 'Operações lógicas');
INSERT INTO SMCatOper (CodCatOper, Nome, Descricao) values
(4, 'Conjuntivo', 'Operações conjuntivas');
```

Figura 4.4. Exemplo de preenchimento dos dados da tabela SMCatOper.

```
/* Tabela SMOperador (Operadores) ----------------------------------*/
INSERT INTO SMOperador (CodOperador, CodCatOper, Nome, Simbolo) values
(0, 0, 'Vígula', ',');
INSERT INTO SMOperador (CodOperador, CodCatOper, Nome, Simbolo) values
(1, 1, 'Soma', '+');
INSERT INTO SMOperador (CodOperador, CodCatOper, Nome, Simbolo) values
(2, 1, 'Subtração', '-');
INSERT INTO SMOperador (CodOperador, CodCatOper, Nome, Simbolo) values
(3, 1, 'Multiplicação', '*');
INSERT INTO SMOperador (CodOperador, CodCatOper, Nome, Simbolo) values
(4, 1, 'Divisão', '/');
INSERT INTO SMOperador (CodOperador, CodCatOper, Nome, Simbolo) values
(5, 2, 'Igual', '=');
INSERT INTO SMOperador (CodOperador, CodCatOper, Nome, Simbolo) values
(6, 2, 'Maior', '>');
INSERT INTO SMOperador (CodOperador, CodCatOper, Nome, Simbolo) values
(7, 2, 'Menor', '<');
INSERT INTO SMOperador (CodOperador, CodCatOper, Nome, Simbolo) values
(8, 2, 'Maior ou igual', '>=');
INSERT INTO SMOperador (CodOperador, CodCatOper, Nome, Simbolo) values
(9, 2, 'Menor ou igual', '<=');
INSERT INTO SMOperador (CodOperador, CodCatOper, Nome, Simbolo) values
(10, 2, 'Diferente', '<>');
INSERT INTO SMOperador (CodOperador, CodCatOper, Nome, Simbolo) values
(11, 2, 'Não menor', '!<');
INSERT INTO SMOperador (CodOperador, CodCatOper, Nome, Simbolo) values
(12, 2, 'Não maior', '!>');
INSERT INTO SMOperador (CodOperador, CodCatOper, Nome, Simbolo) values
(13, 3, 'Entre', 'Between');
INSERT INTO SMOperador (CodOperador, CodCatOper, Nome, Simbolo) values
(14, 3, 'Não entre', 'Not Between');
INSERT INTO SMOperador (CodOperador, CodCatOper, Nome, Simbolo) values
(15, 3, 'Pertence a lista', 'In');
INSERT INTO SMOperador (CodOperador, CodCatOper, Nome, Simbolo) values
(16, 3, 'Não pertence a lista', 'Not In');
INSERT INTO SMOperador (CodOperador, CodCatOper, Nome, Simbolo) values
(17, 3, 'Contém na string', 'Like');
INSERT INTO SMOperador (CodOperador, CodCatOper, Nome, Simbolo) values
(18, 3, 'Não contém na string', 'Not Like');
INSERT INTO SMOperador (CodOperador, CodCatOper, Nome, Simbolo) values
(19, 3, 'É nulo', 'Is Null');
INSERT INTO SMOperador (CodOperador, CodCatOper, Nome, Simbolo) values
(20, 3, 'É diferente de nulo', 'Is Not Null');
INSERT INTO SMOperador (CodOperador, CodCatOper, Nome, Simbolo) values
(21, 4, 'E', 'And');
INSERT INTO SMOperador (CodOperador, CodCatOper, Nome, Simbolo) values
(22, 4, 'Ou', 'Or');
```

Figura 4.5. Exemplo de preenchimento dos dados da tabela SMOperadores.

4.2 Interface

Nesse item serão apresentadas todas as telas que compõem o HydroConsulta. Esse sistema de consultas divide-se, basicamente, em duas abas: "*Dados*" e "*Resultado*".

4.2.1 Aba "Dados"

Nessa categoria é possível especificar todas as características da consulta e visualizar/alterar o comando SQL gerado. Utilizando-se de um recurso visual que lembra um fichário, os dados dessa categoria foram agrupados em quatro sub-abas. São elas: "*Selecionar*", "*Condições*", "*Classificação*" e "*Comando SQL*".

Sub-Aba "Selecionar"

Através dessa aba o usuário é capaz de selecionar o conjunto de campos que serão apresentados no resultado da consulta, ou seja, aqueles que farão parte da cláusula SELECT. Podem ser campos de uma ou mais tabelas. A sua interface pode ser visualizada na Figura 4.6.

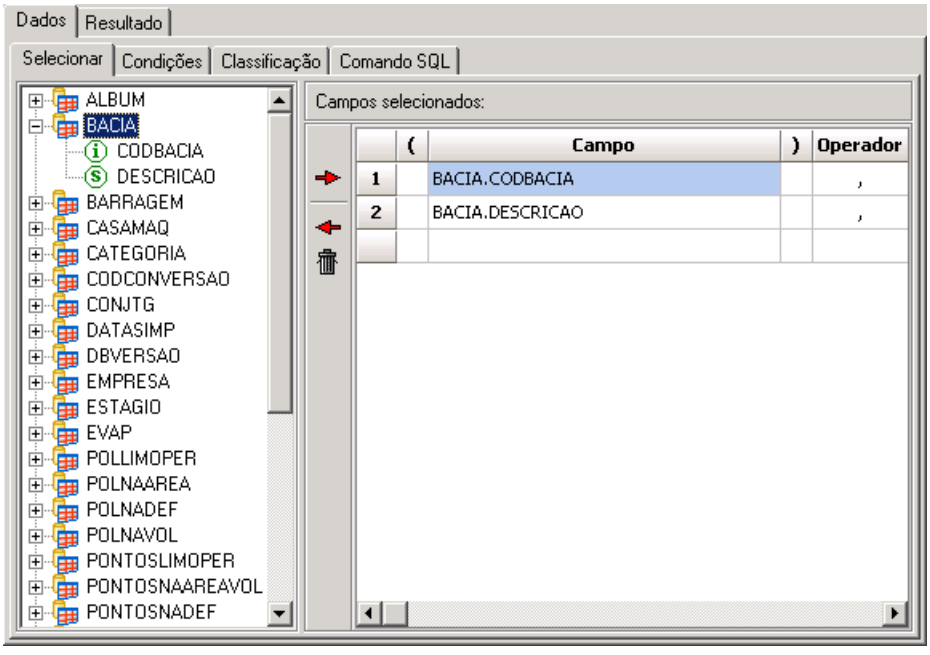


Figura 4.6. Aba "Selecionar".

No quadro à esquerda dessa aba, são exibidas todas as tabelas e campos da base de dados HydroData na forma de uma árvore de expansão. Esse recurso gráfico foi

explorado na tentativa de apresentar várias tabelas em um pequeno espaço da interface com o usuário. Pode-se ver que os nós principais identificam as tabelas. Para cada um desses nós, existe um conjunto de nós subordinados. Esses nós indicam os campos pertencentes à tabela em questão.

Tanto as tabelas quanto os campos, nesse quadro, possuem um ícone à esquerda. O ícone de representação das tabelas é único, porém o ícone dos campos varia de acordo com o seu tipo, conforme ilustra a Tabela 4.1.

Símbolo	Descrição
	ícone de tabela.
	ícone de campo do tipo Smallint, Integer ou Word.
	ícone de campo selecionado do tipo Smallint, Integer ou Word.
	ícone de campo do tipo Float.
	ícone de campo selecionado do tipo Float.
	ícone de campo do tipo String.
	ícone de campo selecionado do tipo String.

Tabela 4.1. Representação dos ícones das tabelas e campos do Hidrodata

Um campo do tipo *Smallint*, *Integer* ou *Word* pode ser entendido como um campo que armazena valores numéricos inteiros, um campo do tipo *Float* armazena números reais e um campo do tipo *String* armazena cadeia de caracteres. A função desses ícones é informar ao usuário o tipo de dado do conteúdo do campo.

No quadro à direita da Figura 4.6, para apresentar os campos selecionados, foi utilizado um componente chamado *TAdvStringGrid*. Esse componente permite que seja disponibilizado um botão nas células pertencentes a uma determinada coluna. Ao editar um campo pertencente a uma coluna programada para explorar esse recurso, um botão é exibido. Um clique nesse botão permite a edição do conteúdo da célula através, por exemplo, de uma nova janela.

Na aba "Selecionar", o recurso acima citado, foi disponibilizado para as colunas "*Campo*" e "*Operador*", conforme a Figura 4.7.

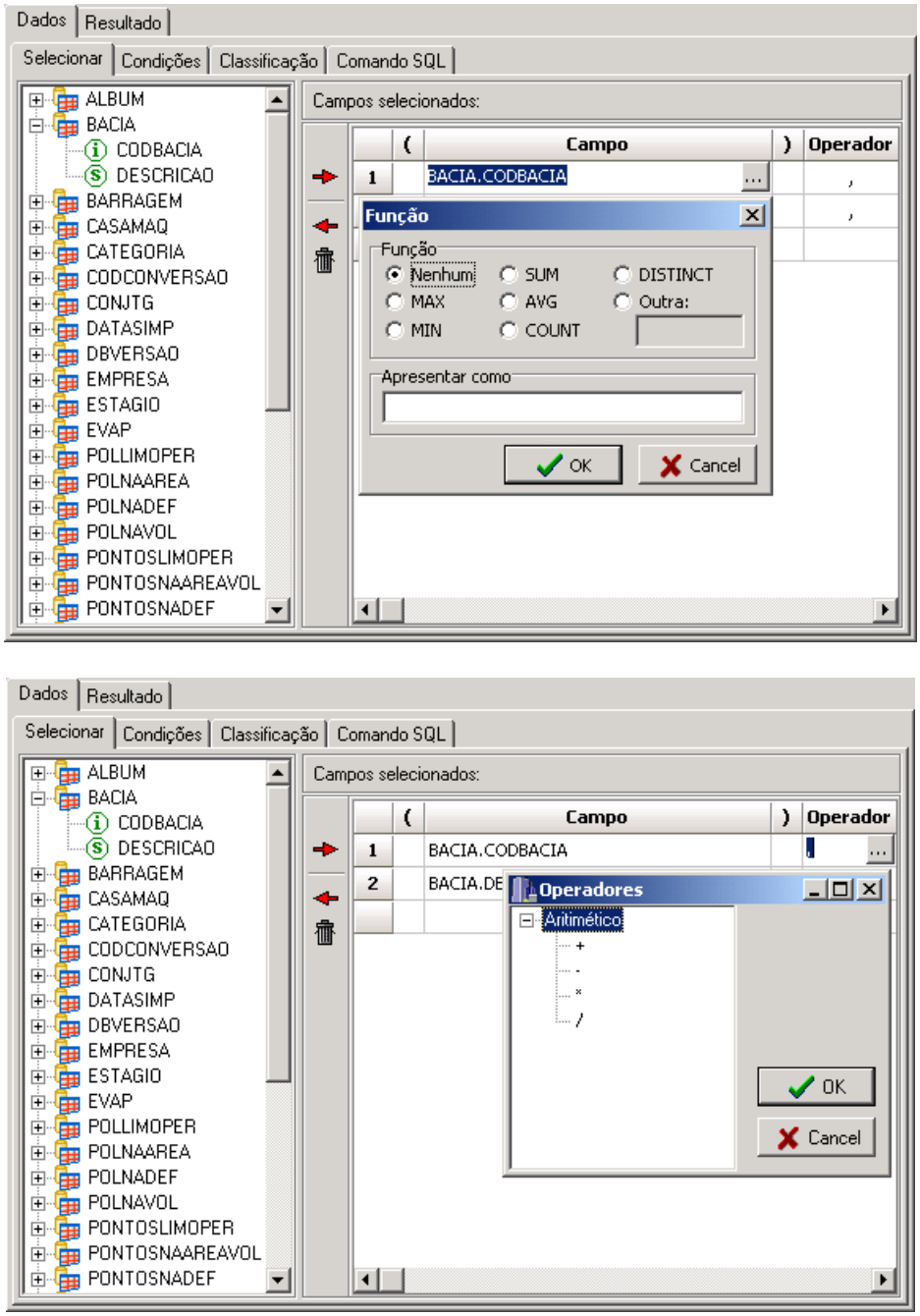


Figura 4.7. Botões do componente TAdvStringGrid na aba "Selecionar".

Botão na coluna "Campo"

Através desse botão é permitido que se associe ao campo selecionado uma função específica. Para isso, faz-se uso do f*ormulário de funções* ilustrado na Figura 4.8. Esse formulário é apresentado após o clique do *mouse* no botão em questão. Ele disponibiliza

uma caixa de edição com o título "*Apresentar como*" cujo conteúdo será exibido na consulta em troca do nome oficial do campo.

Figura 4.8. Formulário de funções

Botão na coluna "Operador"

Muitas vezes é interessante exibir, no resultado da consulta, duas ou mais colunas, agregadas por um operador aritmético. Isso é conseguido através do formulário apresentado na Figura 4.9 e do uso dos parênteses. Os parênteses que envolvem uma expressão fazem com que essa expressão seja avaliada como um bloco.

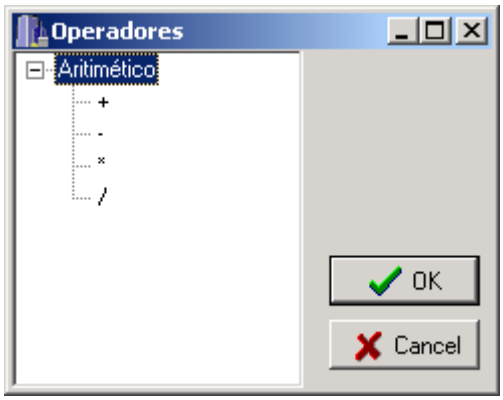


Figura 4.9. Operadores aritméticos.

Assim como o formulário de funções, os o*peradores aritméticos* são exibidos após o clique do mouse no botão da célula ativa.

Sub-Aba "Condições"

Nessa aba é definida a cláusula WHERE do comando SQL. O componente *TAdvStringGrid* dessa aba disponibiliza botões para as colunas "Campo", "Oper" (operadores de comparação e lógico), "Valor" e "Oper" (operadores conjuntivo).

Botão na coluna "Campo"

Um clique nesse botão apresenta um *diálogo padrão*, muito parecido com a aba "Selecionar". Nesse diálogo são escolhidos os campos para os quais será aplicado um filtro. Esse diálogo padrão é ilustrado na Figura 4.10.

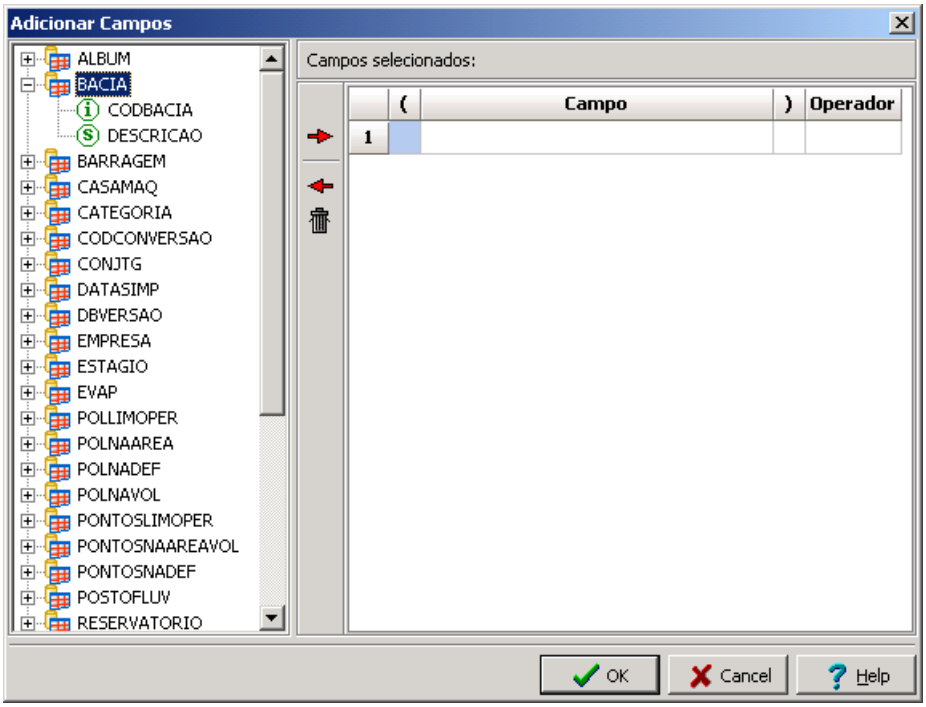


Figura 4.10. Diálogo "Adicionar campos".

Botão na coluna "Oper" (operadores de comparação e lógico)

Uma vez definido o campo ou a lista de campos para a qual será aplicado o filtro, torna-se necessária, a escolha de um operador que associe esse campo ou essa lista a um valor. Esses operadores são exibidos nas Figuras 4.11 e 4.12.

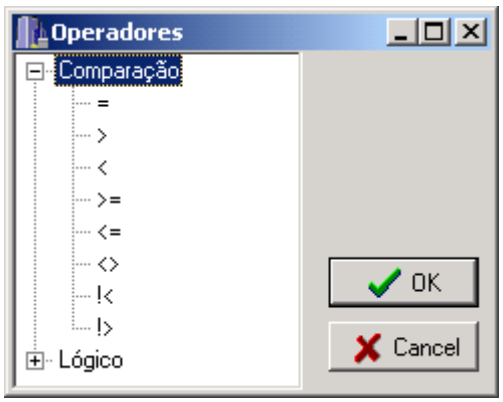


Figura 4.11. Operadores de comparação.

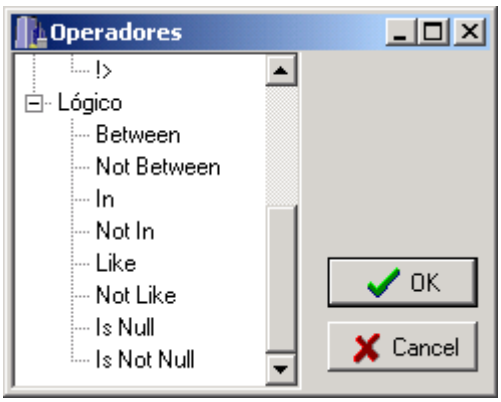


Figura 4.12. Operadores lógicos.

Botão na coluna "Valor"

O valor ao qual estará associado o operador de comparação ou lógico pode ser digitado diretamente na coluna "Valor" ou através do diálogo mostrado na Figura 4.13.

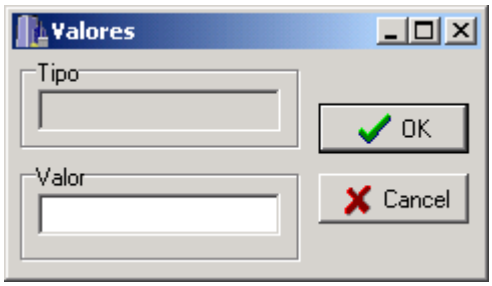


Figura 4.13. Diálogo "Valores".

Botão na coluna "Oper" (operadores conjuntivos)

Duas ou mais condições podem ser associadas com o auxílio dos operadores conjuntivos ilustrados na Figura 4.14. Pode-se também fazer uso dos parênteses, definindo assim a precedência, ou seja, a ordem em que as expressões serão avaliadas.

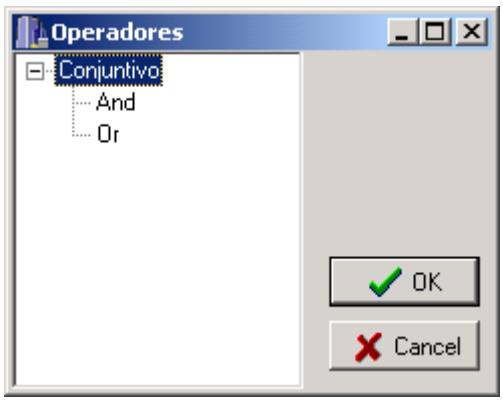
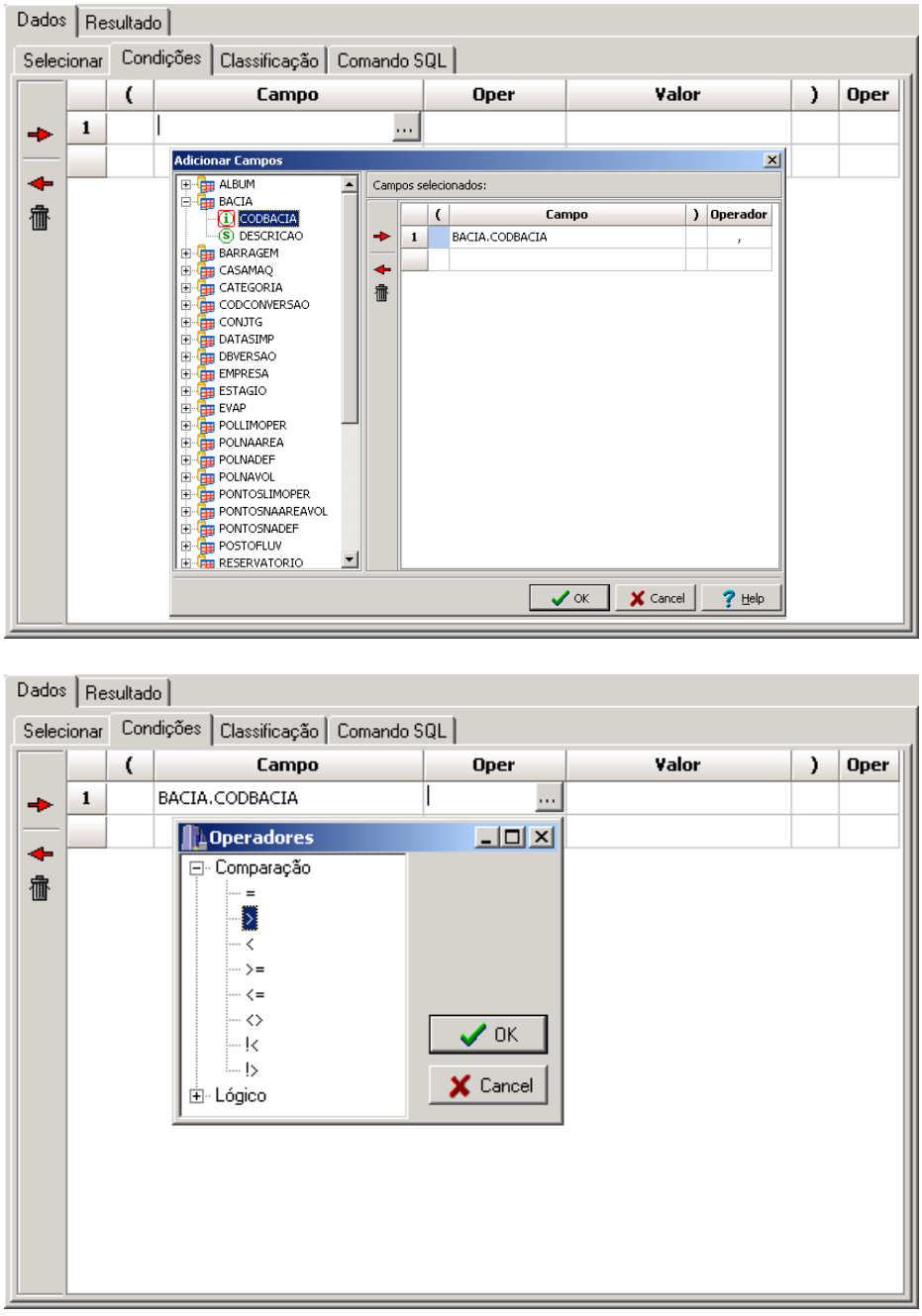


Figura 4.14. Operadores conjuntivos.

A seqüência acima, necessária para o estabelecimento de um filtro na cláusula WHERE, pode ser melhor observada na Figura 4.15 composta por uma seqüência de quatro telas.



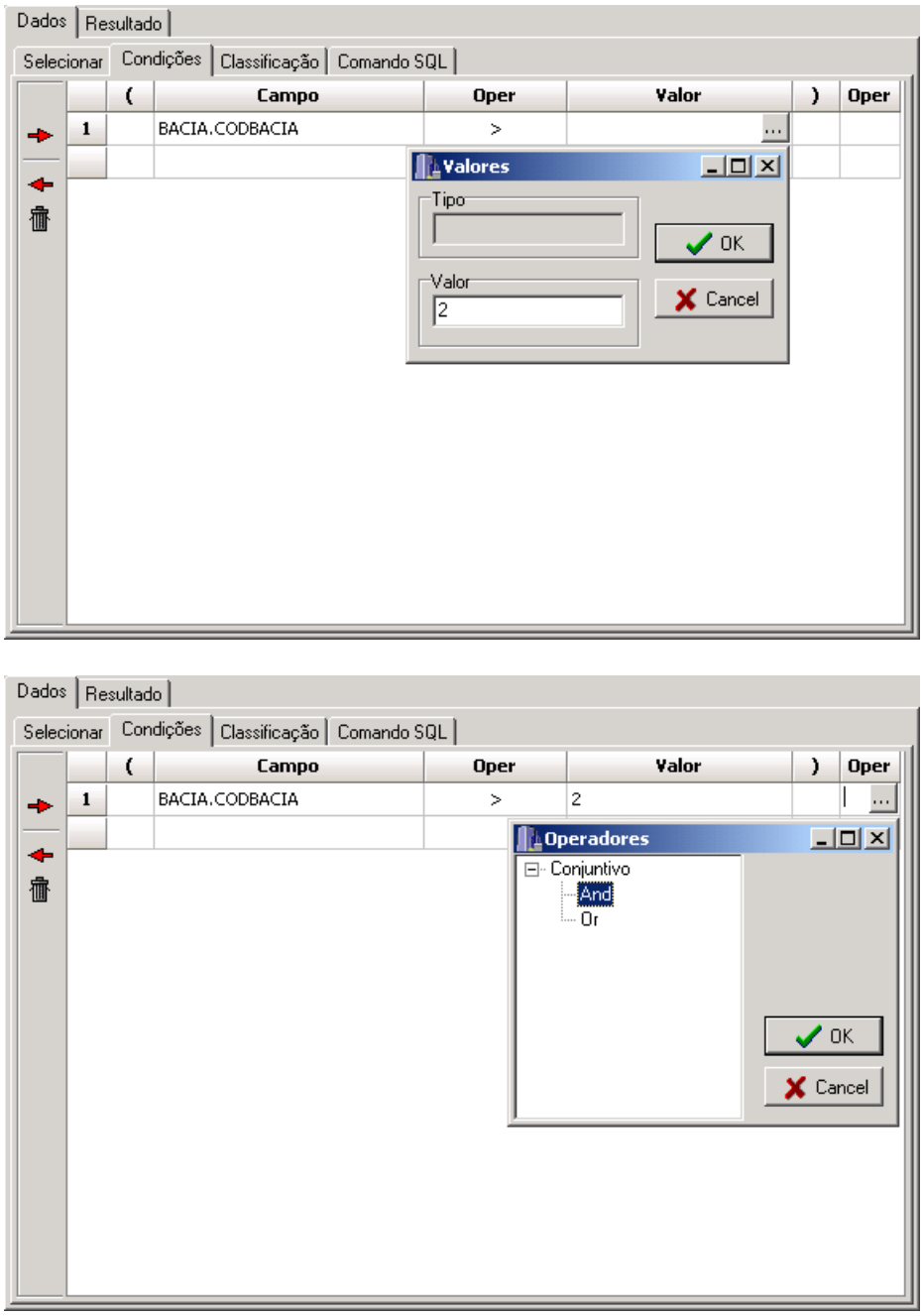


Figura 4.15. Botões do componente TAdvStringGrid na aba "Condições".

A Figura 4.16 mostra o conteúdo da aba "Condições" com três filtros. Nessa aba também é permitido o uso dos parênteses para que uma expressão seja avaliada como um bloco.

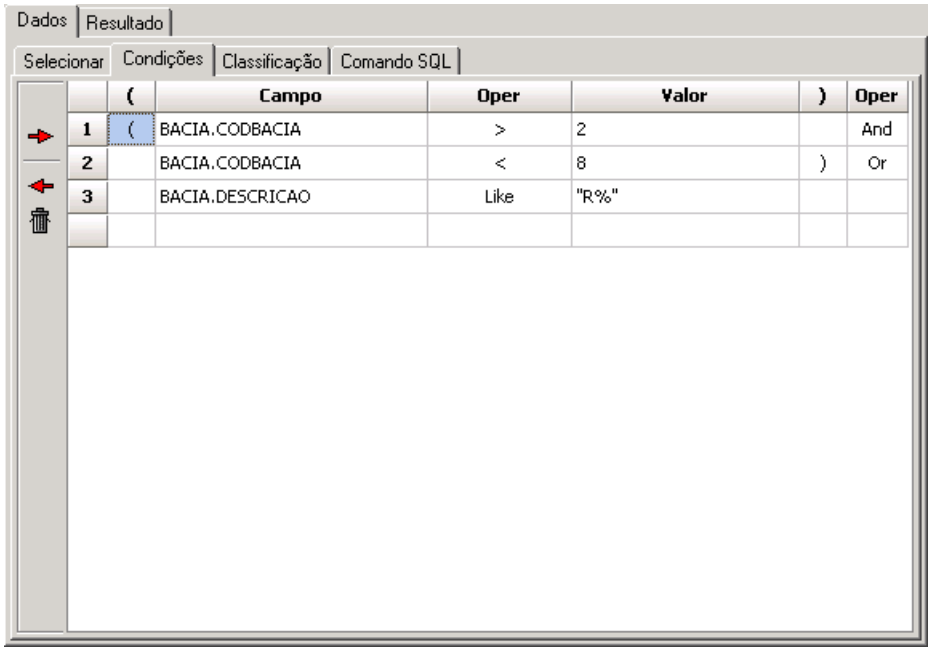


Figura 4.16. Aba "Condições"

Todas as tabelas envolvidas na consulta devem estar relacionadas na aba "Condições". Dessa forma, mantém-se a integridade referencial dos dados citada no Capítulo 2. Exemplos desse tipo de filtro podem ser consultados no Capítulo 5.

Sub-Aba "Classificação"

Nessa aba é definida a cláusula ORDER BY do comando SQL. Através do mesmo diálogo padrão, citado em "Condições" (Figura 4.12), são selecionados os campos a serem classificados de maneira ascendente ou descendente. Sua interface, extremamente simples, é mostrada na Figura 4.17.

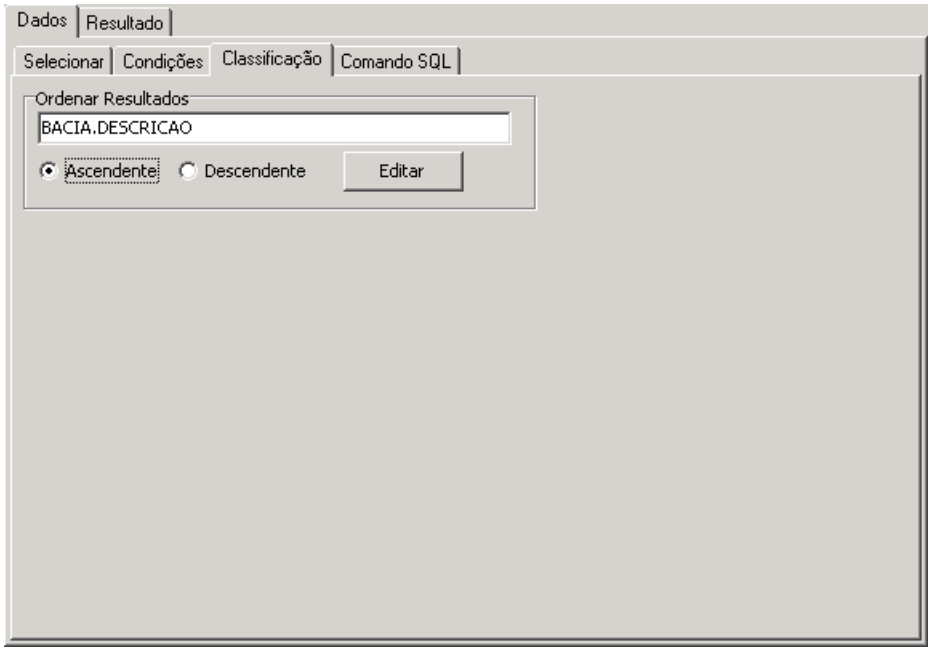


Figura 4.17. Aba "Classificação".

Sub-Aba "Comando SQL"

Nessa aba é exibido o comando SQL gerado de acordo com as especificações do usuário nas abas "Selecionar", "Condições" e "Classificação". A edição do comando é permitida, embora para este caso seja necessário o conhecimento específico da linguagem SQL por parte do usuário do sistema. Na Figura 4.18, visualiza-se a distribuição das cláusulas SELECT, FROM, WHERE e ORDER BY no objeto *Memo* associado a aba em questão.

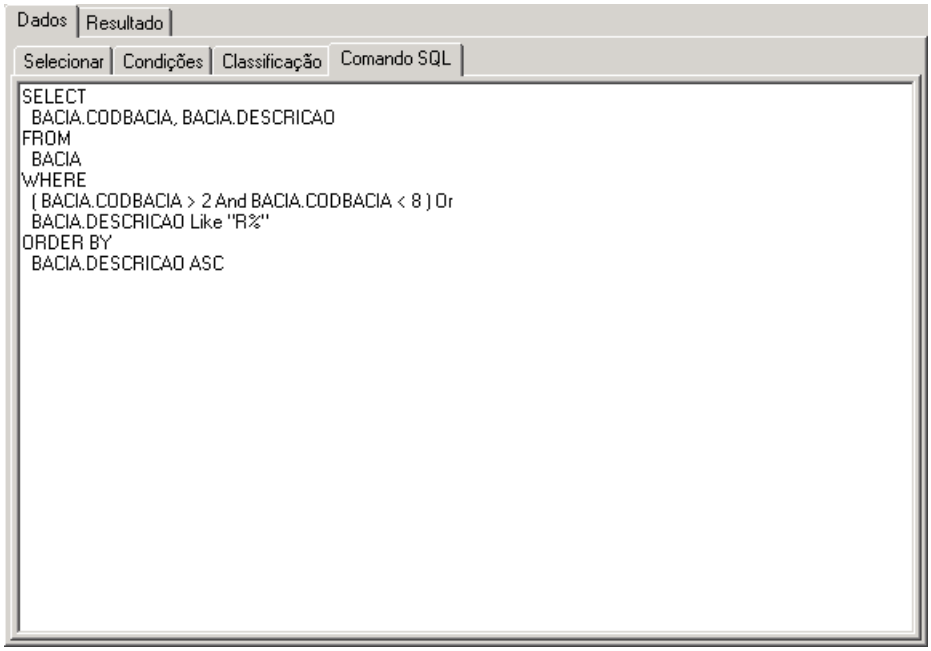


Figura 4.18. Aba "Comando SQL".

4.2.2 Aba "Resultado"

É exibido, num objeto chamado *DBGrid*, os campos escolhidos na aba "Selecionar", com o conteúdo dos registros que respeitaram o filtro da aba "Condições", na ordem estabelecida pela aba "Classificação". Isso pode ser observado na Figura 4.19.

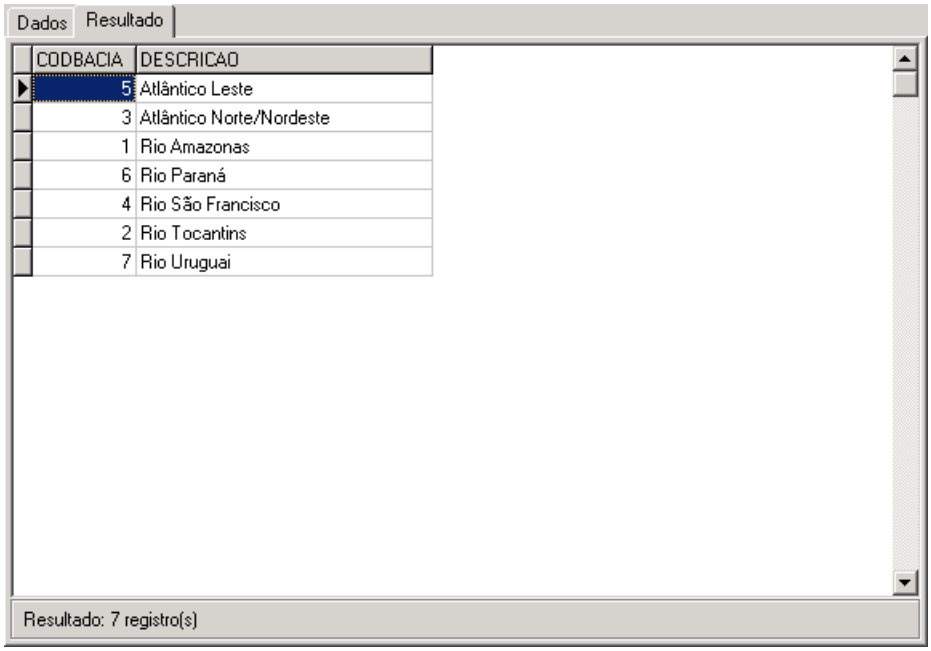


Figura 4.19. Aba "Resultado".

Capítulo 5

Estudos de Caso

Esse capítulo explora a aplicação do módulo HydroConsulta à base de dados HydroData. Para cada caso, aqui analisado, será justificada a importância da consulta e serão apresentadas imagens do sistema HydroConsulta preenchidas de maneira a produzir o resultado desejado.

5.1 Estudo de Caso Introdução

O objetivo desse estudo é apenas localizar o código da bacia "Rio Paraná" que será utilizado no estudo seguinte. Para isso, são exibidos todos os dados da tabela "Bacia". Estes dados descrevem as bacias hidrográficas brasileiras e são gerenciados pelo sistema HydroData.

Na Figura 5.1 são listados os dois únicos campos da tabela "Bacia".

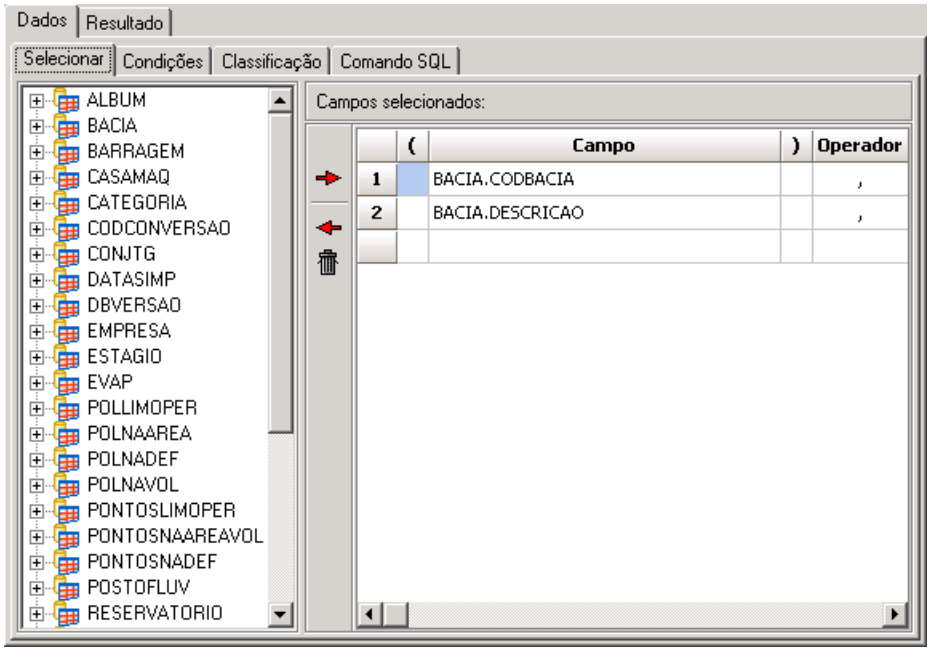


Figura 5.1. Estudo de Caso Introdução – Aba "Selecionar".

Para essa consulta não é necessário filtro e nem mesmo classificação.

Na Figura 5.2, pode-se verificar o comando SQL resultante da seleção de campos feita anteriormente.

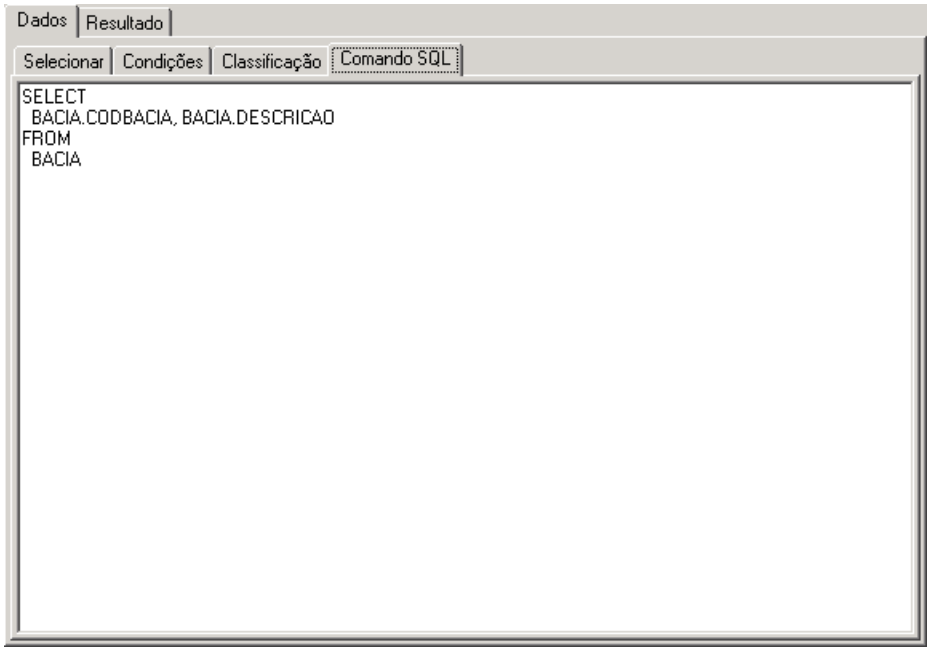


Figura 5.2. Estudo de Caso Introdução – Aba "Comando SQL".

Na Figura 5.3, visualiza-se o resultado da consulta.

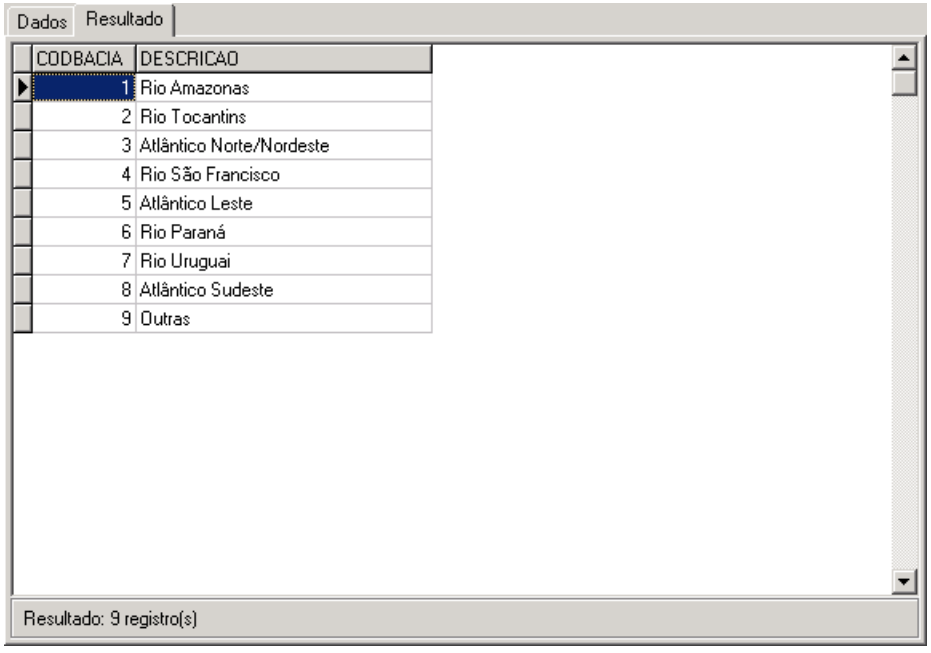


Figura 5.3. Estudo de Caso Introdução – Aba "Resultado".

5.2 Seleção Avançada de Campos

Nesse estudo será exibido o volume útil do reservatório das usinas da bacia do Rio Paraná, cujo código foi identificado na consulta anterior. O volume útil é aquele armazenado entre as cotas máxima e mínima normal operativa.

Na Figura 5.4, são listados dos dados que farão parte do resultado da consulta. Observa-se, pelo uso dos parênteses, que os campos "VolMaxOper" e " VolMinOper" da tabela "Reservatório" serão apresentados numa única coluna obtida pela diferença entre eles. Campos obtidos por meio de uma operação aritmética são chamados de c*ampos calculados*.

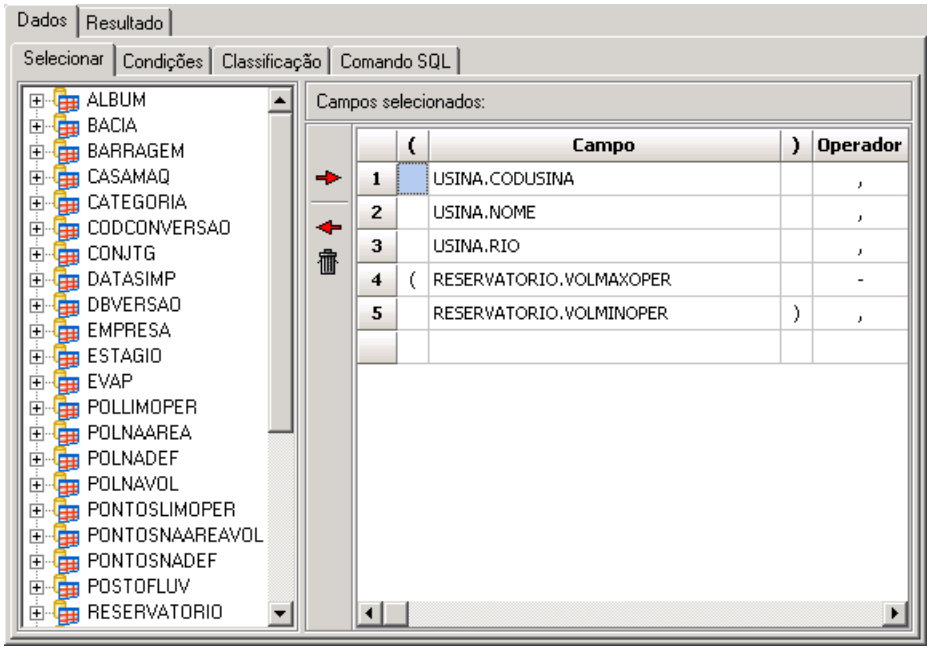


Figura 5.4. Seleção Avançada de Campos – Aba "Selecionar".

Como os campos, selecionados para esta consulta, pertencem a duas tabelas distintas (tabela "Usina" e tabela "Reservatório"), é necessário estabelecer o relacionamento entre elas, mantendo assim a integridade referencial dos dados, apresentada no Capítulo 2. Isso é feito na primeira condição da Figura 5.5 (para conhecer o modelo relacional do HydroData, deve-se consultar o Apêndice A). Na linha seguinte, da mesma figura, é aplicado um filtro para o campo "Bacia" que deverá ser igual a "6". Desta maneira, apenas as usinas da bacia do Rio Paraná serão exibidas.

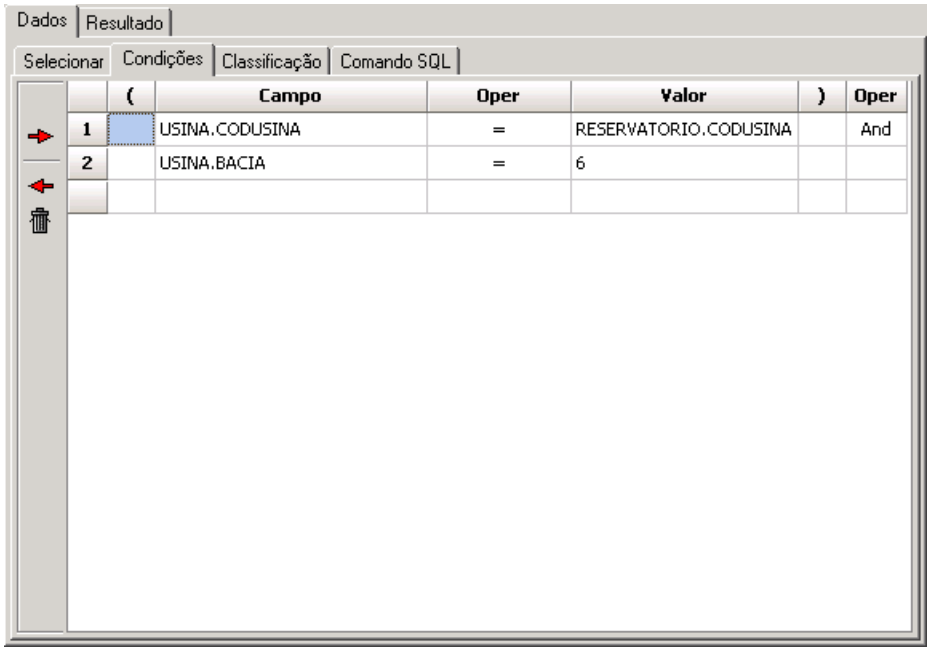


Figura 5.5. Seleção Avançada de Campos – Aba "Condições".

A classificação da consulta em ordem ascendente do campo "Nome" da tabela "Usina" pode ser visualizada na Figura 5.6.

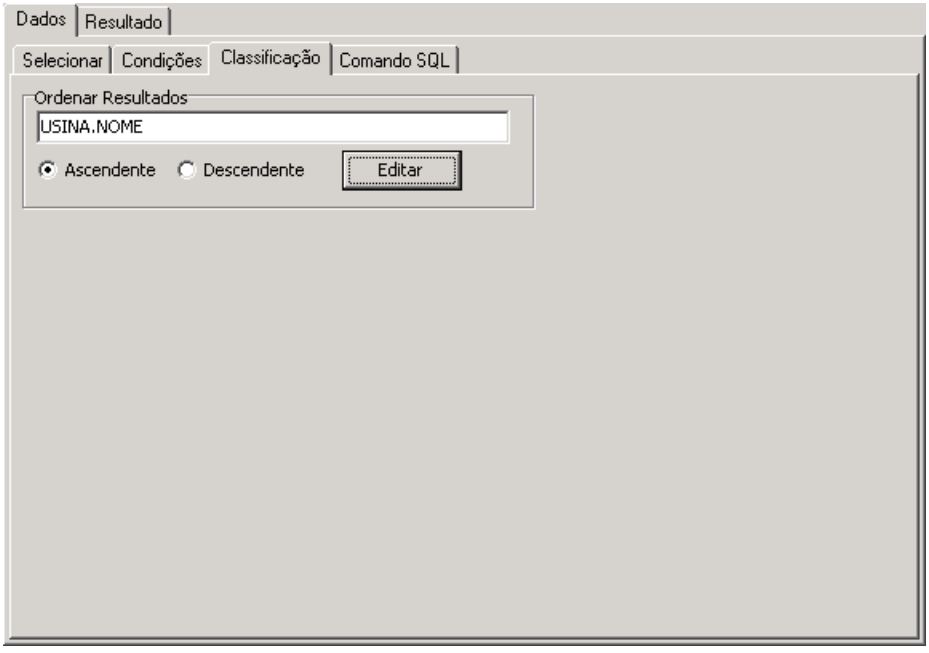


Figura 5.6. Seleção Avançada de Campos – Aba "Classificação".

A Figura 5.7 exibe o comando SQL responsável por esta consulta. Nota-se, que após a diferença entre os campos "VolMaxOper" e " VolMinOper" aparece a palavra

"VolUtil". Essa palavra foi acrescentada manualmente e será o alias (apelido) para a coluna que apresentará a diferença entre os campos. Um alias, conforme o Capítulo 2, personaliza o nome do cabeçalho de uma coluna. Campos calculados, para os quais não são especificados aliases, são exibidos com um cabeçalho padrão: COLUMN seguido do número da coluna. Para este exemplo, o cabeçalho padrão do campo calculado seria COLUMN4.

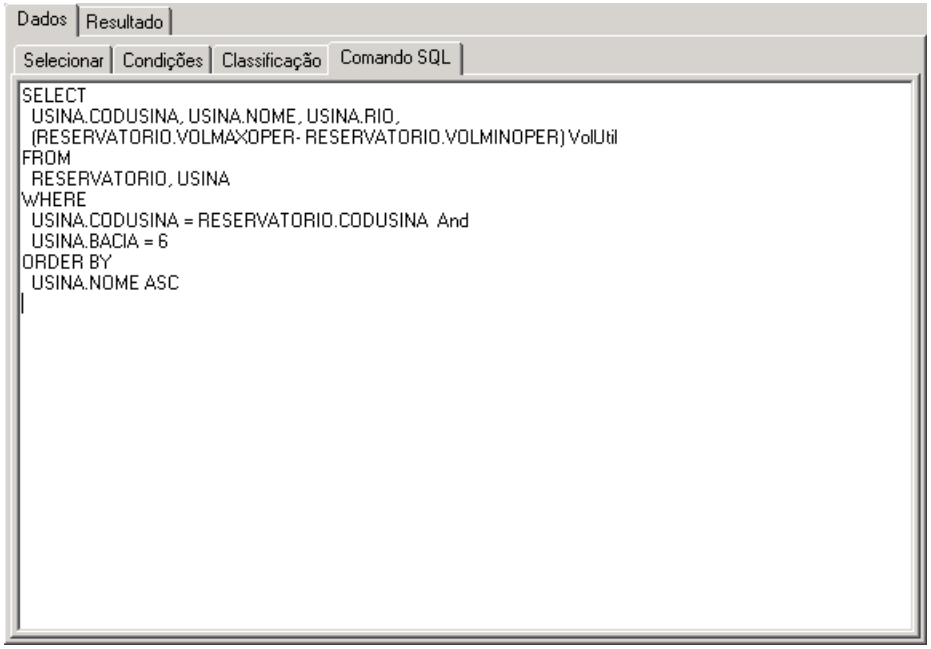


Figura 5.7. Seleção Avançada de Campos – Aba "Comando SQL".

As Figuras 5.8 e 5.9 mostram o resultado dessa consulta.

Dados | Resultado

CODUSINA	NOME	RIO	VOLUTIL
62744080	Álvaro Souza Lima	Tietê	58,9000244140625
64215080	Armando Avellanal Laydner	Paranapanema	3165
61819080	Armando Salles de Oliveira	Pardo	17,3799996376038
62729080	Barra Bonita	Tietê	2566,90002441406
60625080	Cachoeira Dourada	Paranaíba	0
61811080	Caconde	Pardo	504,050018310547
61061080	Camargos	Grande	672
64345080	Canoas I	Paranapanema	30,6299896240234
64345075	Canoas II	Paranapanema	25,1600036621094
60360080	Capim Branco I	Araguari	0
60360085	Capim Branco II	Araguari	0
64270080	Chavantes	Paranapanema	3041
60460000	Corumbá I	Corumbá	1024
60444000	Corumbá IV	Corumbá	784
63995080	Eng. Sérgio Motta	Paraná	4252
63007080	Eng. Souza Dias	Paraná	903,780029296875
64516080	Escola Engenharia Mackenzie	Paranapanema	5724,51953125
64535080	Escola Politécnica	Paranapanema	138,130004882813
61818080	Euclides da Cunha	Pardo	5,22999954223633
61146080	Funil Grande	Grande	0
61661000	Furnas	Grande	17217
65774403	Gov. Bento Munhoz da Rocha Netto	Iguaçu	3805
65805010	Gov. Ney Aminthas de Barros Braga	Iguaçu	388
62790080	Ibitinga	Tietê	56,2999877929688
61740080	Igarapava	Grande	0
62020080	Ilha Solteira	Paraná	5515,5009765625
62020081	Ilha Solteira Equivalente	Paraná	8965
64918980	Itaipu	Paraná	5970
60610080	Itumbiara	Paranaíba	12454
61065080	Itutinga	Grande	0
61734080	Jaguara	Grande	0
61998080	José Ermírio de Moraes	Grande	5169
64332080	Lucas Nogueira Garcez	Paranapanema	29,3699979782104
61731080	Luiz Carlos Barreto de Carvalho	Grande	0

Dados | Resultado

CODUSINA	NOME	RIO	VOLUTIL
61146080	Funil Grande	Grande	0
61661000	Furnas	Grande	17217
65774403	Gov. Bento Munhoz da Rocha Netto	Iguaçu	3805
65805010	Gov. Ney Aminthas de Barros Braga	Iguaçu	388
62790080	Ibitinga	Tietê	56,2999877929688
61740080	Igarapava	Grande	0
62020080	Ilha Solteira	Paraná	5515,5009765625
62020081	Ilha Solteira Equivalente	Paraná	8965
64918980	Itaipu	Paraná	5970
60610080	Itumbiara	Paranaíba	12454
61065080	Itutinga	Grande	0
61734080	Jaguara	Grande	0
61998080	José Ermírio de Moraes	Grande	5169
64332080	Lucas Nogueira Garcez	Paranapanema	29,3699979782104
61731080	Luiz Carlos Barreto de Carvalho	Grande	0
66240080	Manso	Manso	3119
61730080	Marechal Mascarenhas de Moraes	Grande	2500
61941080	Marimbondo	Grande	5260
62820080	Mário Lopes Leão	Tietê	2127,39990234375
60351080	Miranda	Araguari	146
60330080	Nova Ponte	Araguari	10375
64278080	Ourinhos	Paranapanema	0
64219080	Pirajú	Paranapanema	0
61796080	Porto Colômbia	Grande	0
64575003	Porto São José	Paraná	0
64571080	Rosana	Paranapanema	407,52001953125
62829580	Rui Barbosa	Tietê	398
65973500	Salto Caxias	Iguaçu	0
65894991	Salto Osório	Iguaçu	0
65883051	Salto Santiago	Iguaçu	4113
60877080	São Simão	Paranaíba	5540
60160080	Theodomiro C. Santiago	Paranaíba	13056
62900080	Três Irmãos	Tietê	3448
61760080	Volta Grande	Grande	0

Figuras 5.8 e 5.9. Seleção Avançada de Campos – Aba "Resultado".

5.3 Canal de Fuga Constante

A cota do canal de fuga de uma usina é uma função da vazão defluente, ou seja, das vazões turbinadas e vertidas que são descarregadas no canal de fuga e retornam para o leito natural do rio onde localiza-se a usina (Cicogna, 1999). Essa função é, geralmente, representada por polinômios que podem possuir até quatro graus. A obtenção desses polinômios é feita a partir de estudos topográficos e hidrológicos do perfil do canal de fuga.

A finalidade deste estudo é listar o nome das usinas cujo campo "Coef1" da tabela "PolNaDef" não apresenta conteúdo, ou seja, usinas que ao invés de possuírem funções não lineares para representação da cota do canal de fuga, possuem uma função constante.

Na Figura 5.10, são listados os campos que farão parte da cláusula SELECT.

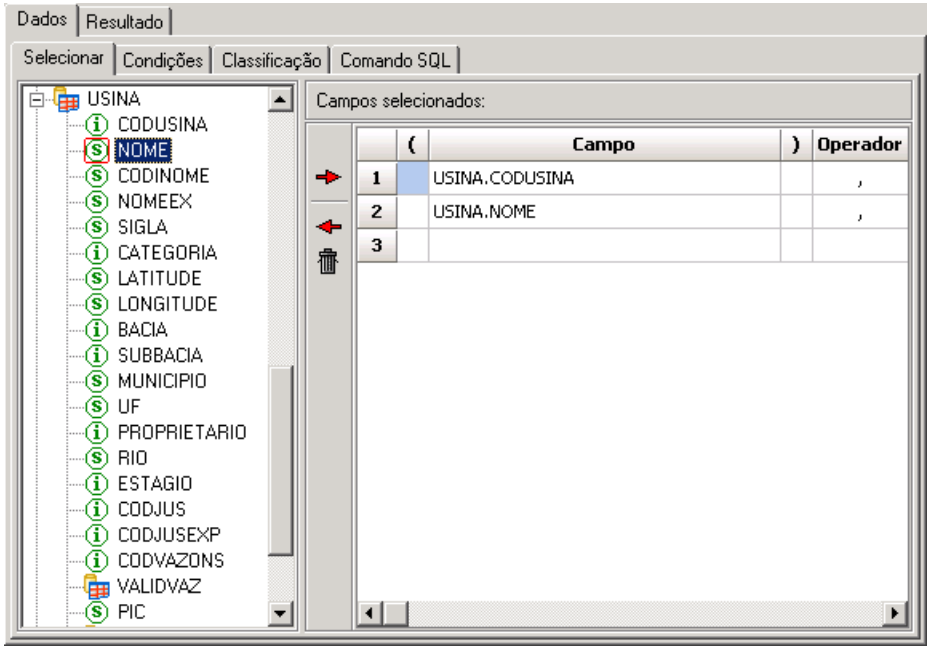


Figura 5.10. Canal de Fuga Constante – Aba "Selecionar".

Como o campo "Coef1" pertence à tabela "PolNaDef", é necessário estabelecer os relacionamentos que ligam essa tabela a tabela de usinas (Apêndice A). Isso é feito na primeira e na segunda condição da Figura 5.11. Na linha seguinte, da mesma figura, é aplicado o filtro para o campo "Coef1" da tabela "PolNaDef". Dessa maneira, são exibidas apenas as usinas que não contém valor para o campo "Coef1".

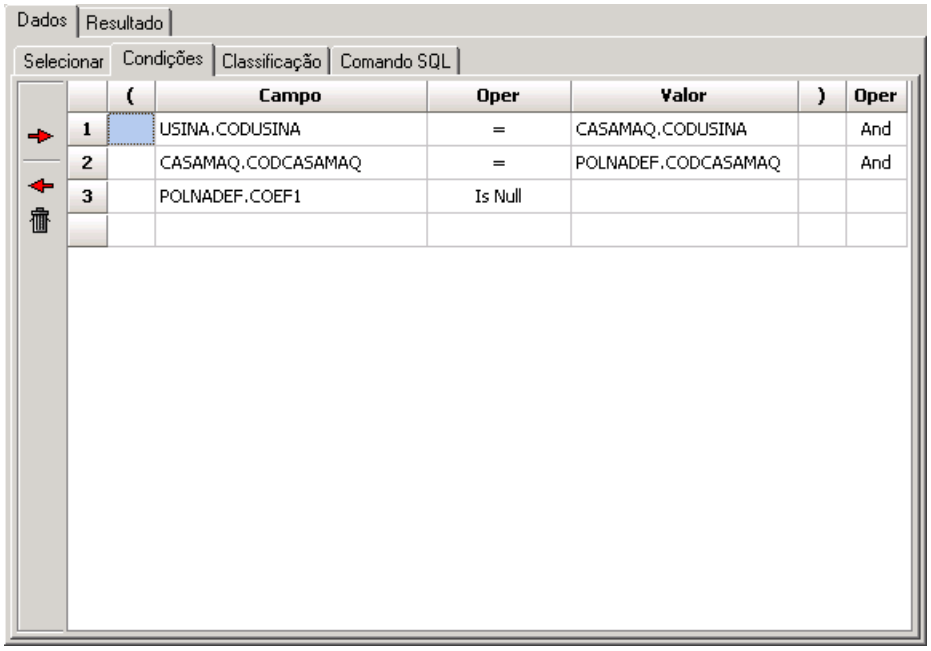


Figura 5.11. Canal de Fuga Constante – Aba "Condições".

A classificação da consulta em ordem ascendente do campo "Nome" da tabela "Usina" pode ser visualizada na Figura 5.12.

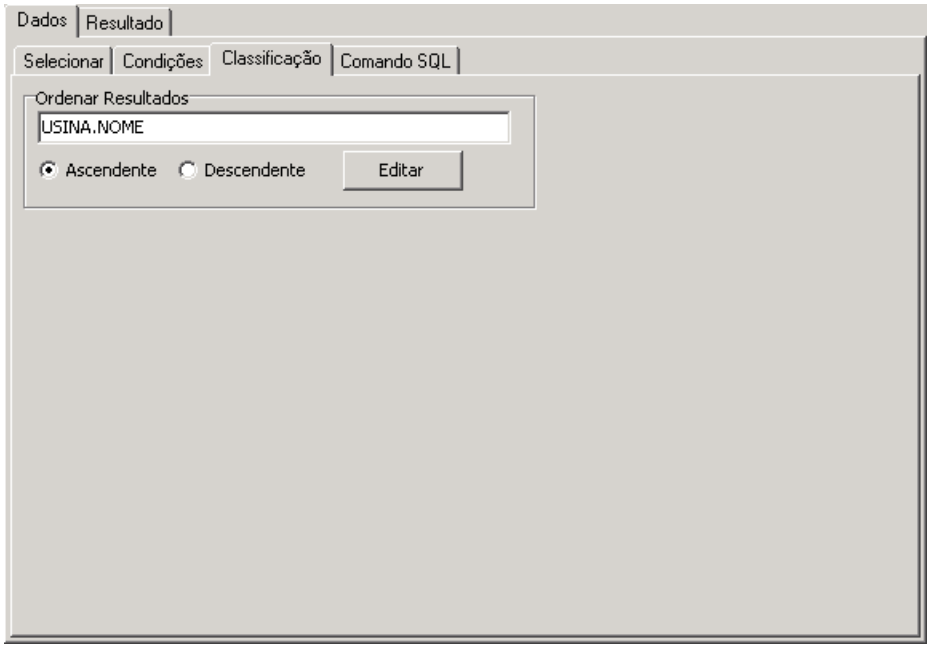


Figura 5.12. Canal de Fuga Constante – Aba "Classificação".

Na Figura 5.13, é exibido o comando SQL responsável pela consulta em questão.

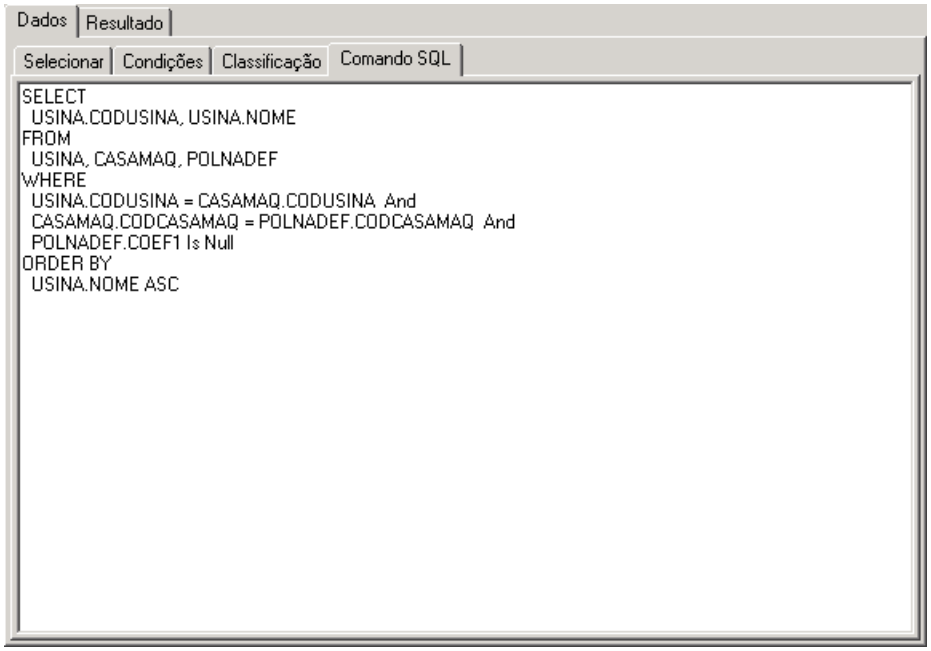


Figura 5.13. Canal de Fuga Constante – Aba "Comando SQL".

Na Figura 5.14 são listadas as usinas que não apresentam valores para o campo "Coef1" da tabela "PolNaDef". Essas usinas possuem função constante para representação da cota do canal de fuga.

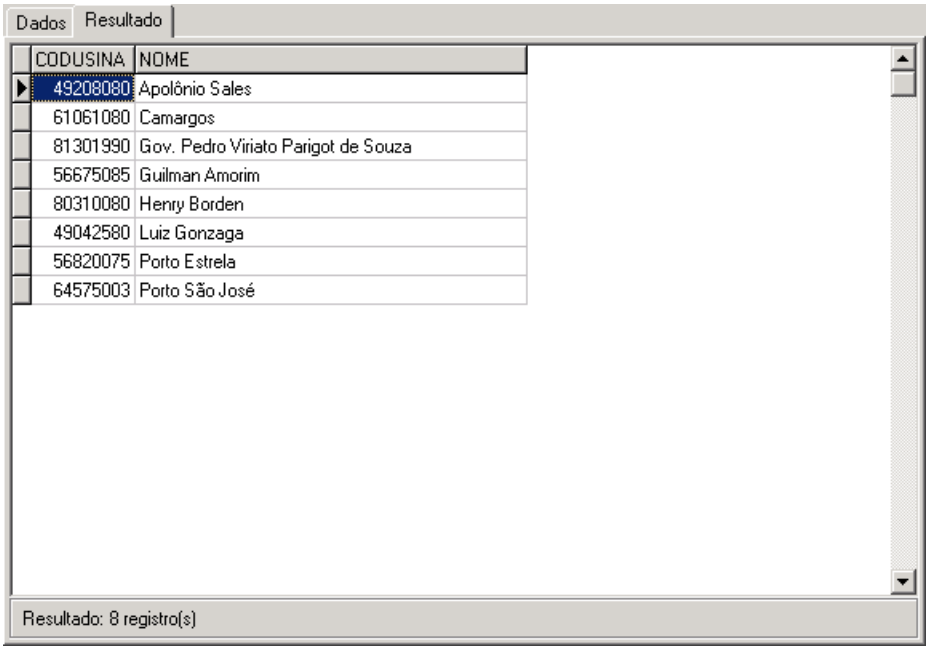


Figura 5.14. Canal de Fuga Constante – Aba "Resultado".

5.4 Rendimento Médio Inconsistente

O objetivo deste estudo é listar as usinas que apresentam valores de rendimento médio das turbinas superiores a 90% no processo de conversão de energia potencial em energia elétrica. Para representar o rendimento médio da turbina divide-se o campo "Produtib" da tabela "CasaMaq" por 0.0098 (Cicogna, 1999).

Na Figura 5.15, são listados os campos que farão parte da saída da consulta.

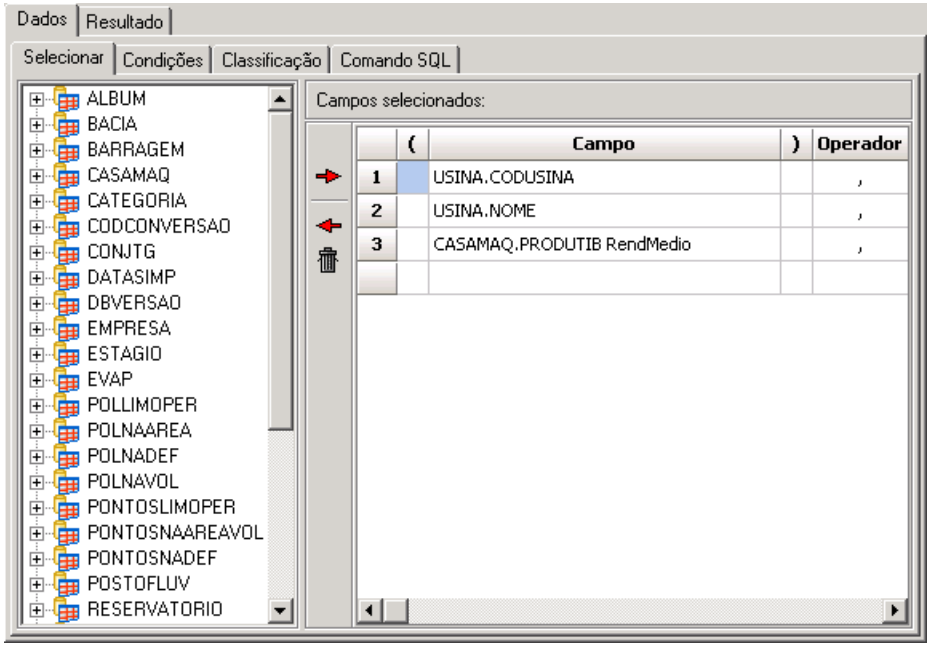


Figura 5.15. Rendimento Médio Inconsistente – Aba "Selecionar".

Como o campo "Produtib" pertence à tabela "CasaMaq" é necessário estabelecer o relacionamento entre essa tabela e a tabela de usinas (Apêndice A). Isso é feito na primeira condição da Figura 5.16. Na linha seguinte, da mesma figura, é aplicado o filtro específico da consulta.

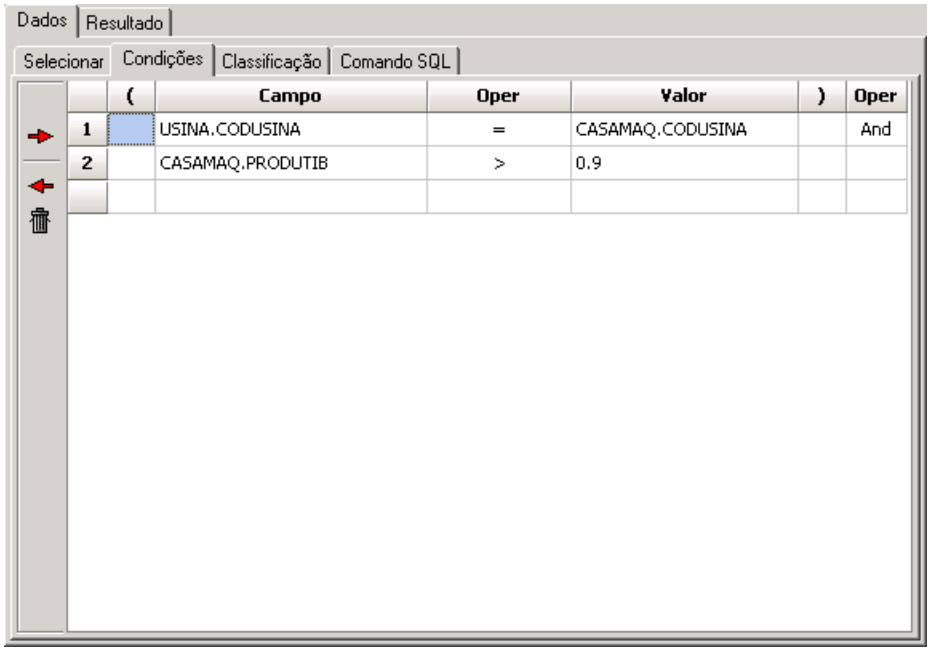


Figura 5.16. Rendimento Médio Inconsistente – Aba "Condições".

Para facilitar a visualização do resultado da consulta, este será classificado em ordem ascendente do campo "Nome" da tabela "Usina", conforme a Figura 5.17.

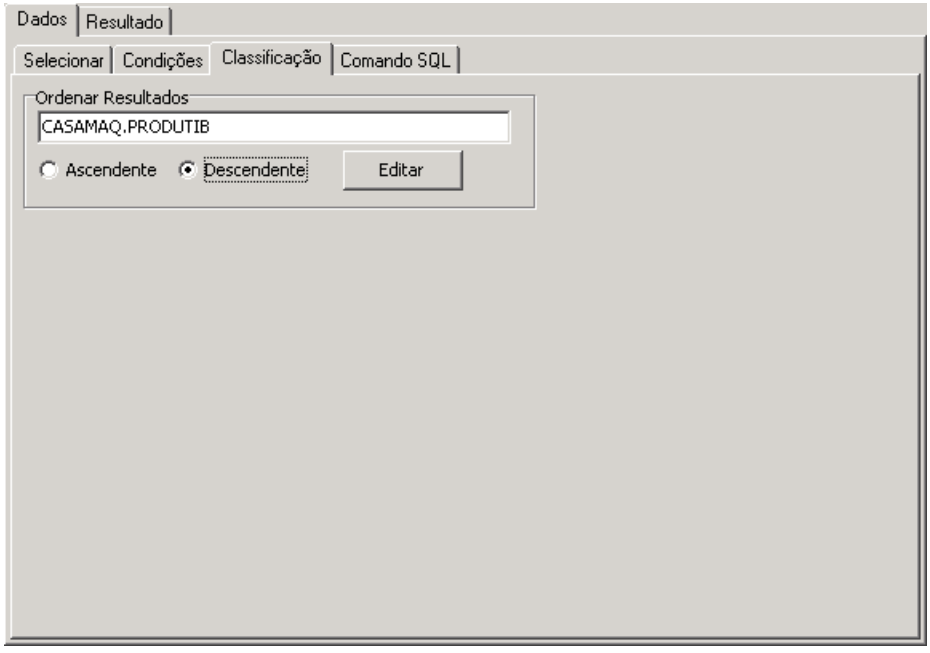


Figura 5.17. Rendimento Médio Inconsistente – Aba "Classificação".

O comando SQL, gerado automaticamente pelo sistema, foi manualmente alterado. O campo “Produtib” da tabela “CasaMaq” foi dividido por 0.0098, na cláusula SELECT e WHERE, para representar o rendimento médio da turbina.

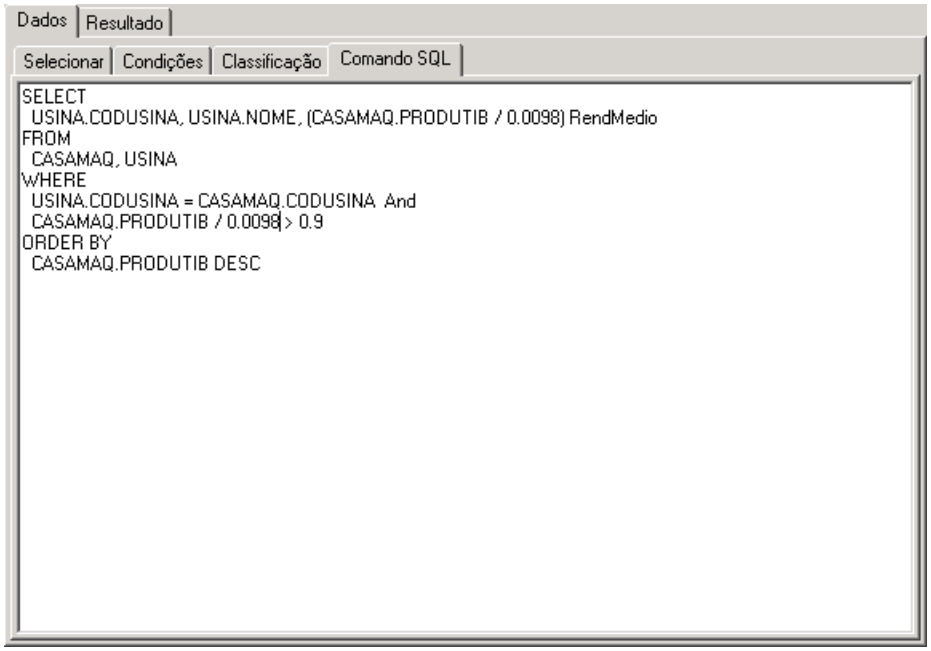


Figura 5.18. Rendimento Médio Inconsistente – Aba “Comando SQL”.

A saída dessa consulta pode ser visualizada nas Figuras 5.19 e 5.20.

Dados | Resultado

CODUSINA	NOME	RENDMEDIO
85260001	Passo Real	0,940918333220239
85365000	Itaúba	0,940918333220239
85398000	Dona Francisca	0,930918399624679
64345080	Canoas I	0,929897936175064
60444000	Corumbá IV	0,929795965856435
64345075	Canoas II	0,928265270682014
49208080	Apolônio Sales	0,924897969377284
49210084	Paulo Afonso 4	0,921938739412901
70840080	Barra Grande	0,920938803073095
61740080	Igarapava	0,920918370996203
49340080	Xingó	0,920918370996203
65774403	Gov. Bento Munhoz da Rocha Netto	0,920918370996203
65805010	Gov. Ney Aminthas de Barros Braga	0,920918370996203
47750080	Sobradinho	0,920918370996203
61760080	Volta Grande	0,920918370996203
64918980	Itaipu	0,920918370996203
54960080	Itapebi	0,920918370996203
60877080	São Simão	0,920918370996203
64571080	Rosana	0,916734736944948
22041080	Peixe Angical	0,915918404198423
64535080	Escola Politécnica	0,914795875397264
18849080	Altamira	0,910928605922631
16070980	Balbina	0,910918342367727
61734080	Jaguara	0,910918342367727
72690081	Machadinho	0,910918342367727
73420080	Passo Fundo	0,910918342367727
18900080	Belo Monte	0,910918342367727
60330080	Nova Ponte	0,910918342367727
73200080	Itá	0,910918342367727
49042580	Luiz Gonzaga	0,910918342367727
64516080	Escola Engenharia Mackenzie	0,903469377330371
23700080	Estreito	0,902959145605564

Resultado: 58 registro(s)

Dados | Resultado

CODUSINA	NOME	RENDMEDIO
18900080	Belo Monte	0,910918342367727
60330080	Nova Ponte	0,910918342367727
73200080	Itá	0,910918342367727
49042580	Luiz Gonzaga	0,910918342367727
64516080	Escola Engenharia Mackenzie	0,903469377330371
23700080	Estreito	0,902959145605564
21360000	São Salvador	0,902959145605564
61731080	Luiz Carlos Barreto de Carvalho	0,901020379090796
60360080	Capim Branco I	0,900918408772167
60360085	Capim Branco II	0,900918408772167
66240080	Manso	0,900918408772167
15459080	Samuel	0,900918408772167
28544080	Santa Isabel	0,900918408772167
65894991	Salto Osório	0,900918408772167
65883051	Salto Santiago	0,900918408772167
62790080	Ibitinga	0,900918408772167
56990777	Aimorés	0,900918408772167
60351080	Miranda	0,900918408772167
64219080	Pirajú	0,900918408772167
62020080	Ilha Solteira	0,900918408772167
62020081	Ilha Solteira Equivalente	0,900918408772167
63995080	Eng. Sérgio Motta	0,900918408772167
81301990	Gov. Pedro Viriato Parigot de Souza	0,900918408772167
61998080	José Ermírio de Moraes	0,900918408772167
56820075	Porto Estrela	0,900918408772167
62900080	Três Irmãos	0,900918408772167
60610080	Itumbiara	0,900918408772167
61146080	Funil Grande	0,900918408772167
56675085	Guilman Amorim	0,900918408772167
64278080	Ourinhos	0,900918408772167
62829580	Rui Barbosa	0,900918408772167
62820080	Mário Lopes Leão	0,900918408772167

Resultado: 58 registro(s)

Figuras 5.19 e 5.20. Rendimento Médio Inconsistente – Aba "Resultado".

5.5 Cláusula Like

Tão simples quanto o Estudo de Caso Introdução, a finalidade desse estudo também é apenas localizar um código. Neste caso, o código a ser consultado é o da empresa "*Duke Energy International, Geração Paranapanema SA*" que será utilizado no estudo seguinte. Para isso são exibidos os campos "CodEmpresa" e "Nome" da tabela "Empresa", conforme a Figura 5.21.

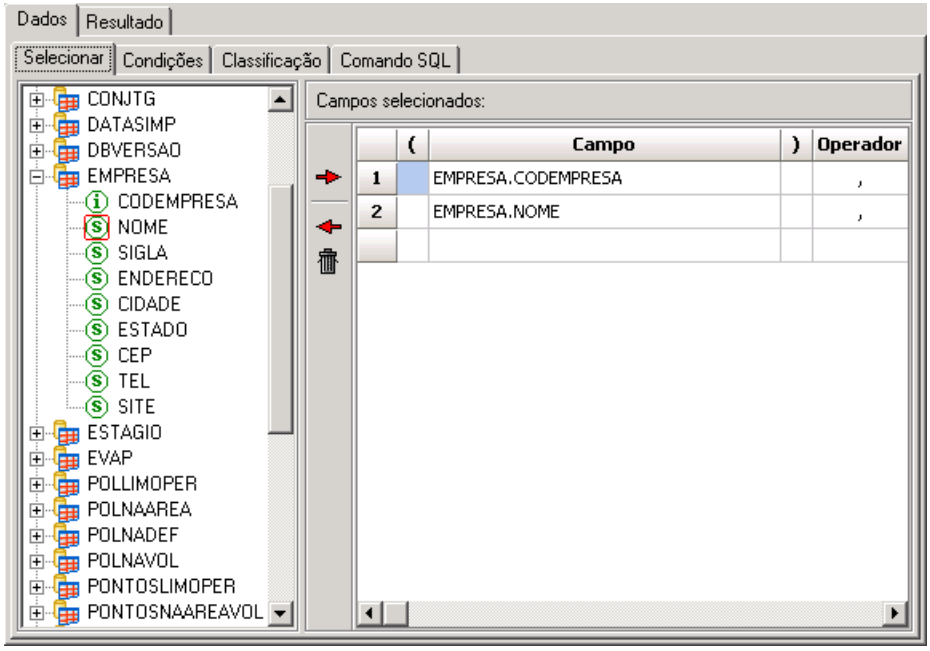


Figura 5.21. Cláusula Like – Aba "Selecionar".

Embora esta tabela não apresente mais que 20 registros, optou-se por aplicar um filtro na consulta (Figura 5.22). Isso facilita a localização do código da empresa "Duke".

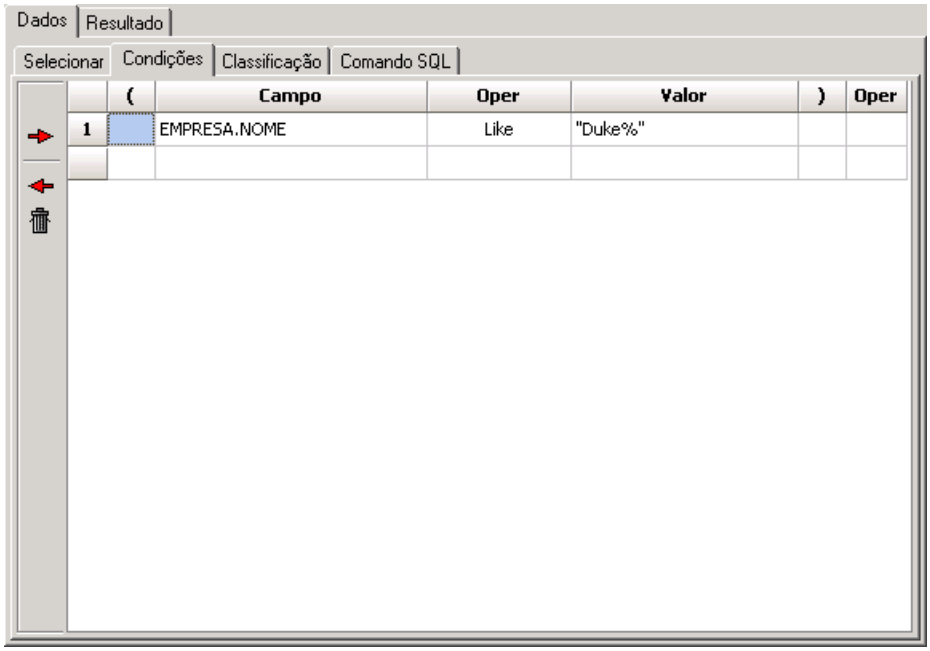


Figura 5.22. Cláusula Like – Aba "Condições".

A Figura 5.23 , apresenta o comando SQL para essa consulta.

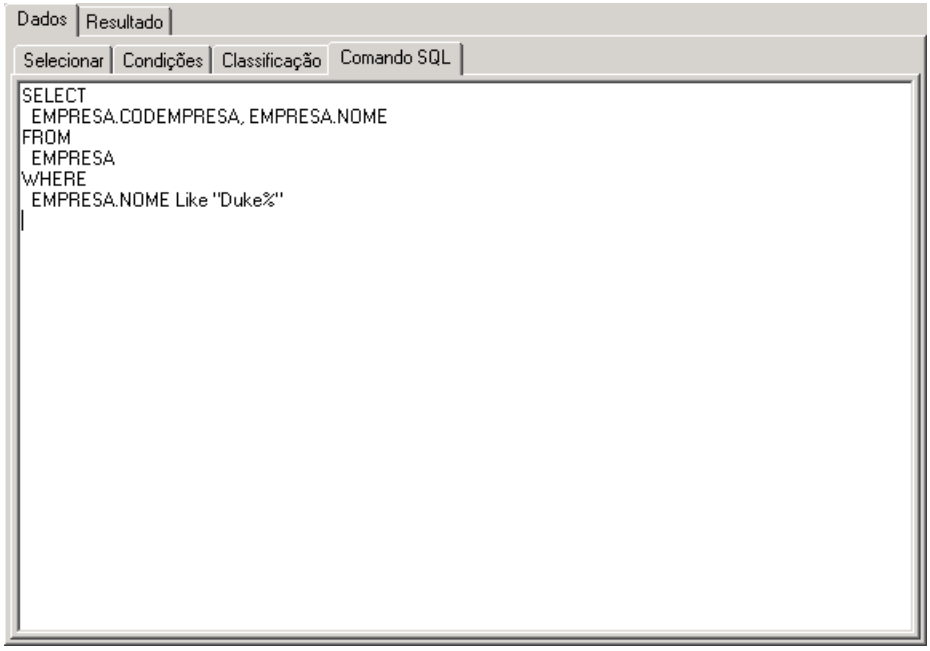


Figura 5.23. Cláusula Like – Aba "Comando SQL".

A Figura 5.24, apresenta o código procurado.

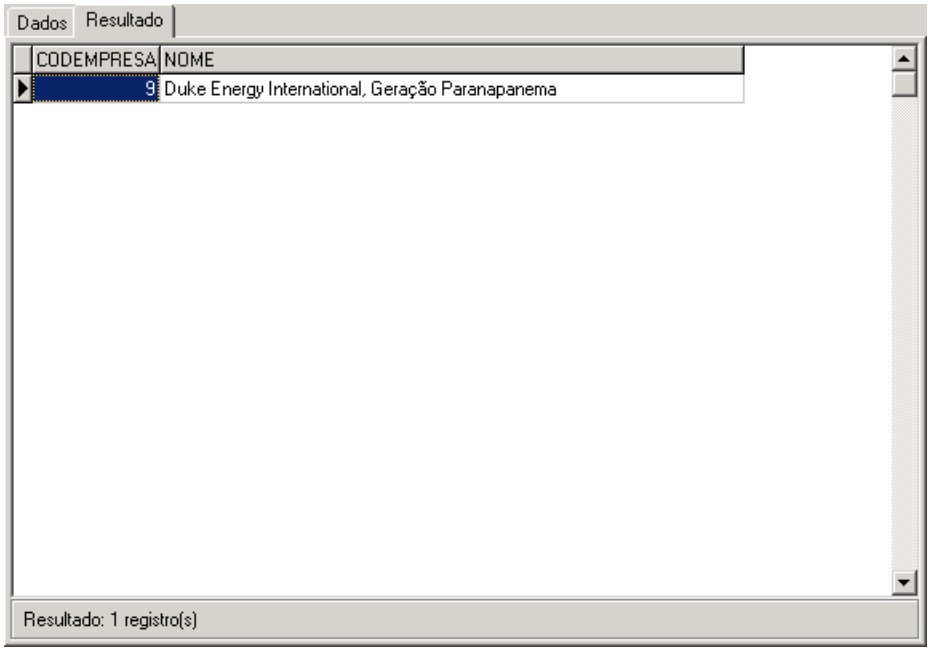


Figura 5.24. Cláusula Like – Aba "Resultado".

5.6 Relatórios Avançados

Esse estudo exibe alguns dados solicitados pela *ANELL (Agência Nacional de Energia Elétrica)* ao presidente da empresa *Duke Energy Internacional, Geração Paranapanema S.A,* conforme Apêndice B. Trata-se de um relatório necessário ao procedimento de atualização periódica dos dados utilizados para o cálculo dos valores a serem recolhidos a título de compensação financeira pela utilização de recursos hídricos para fins de geração de energia elétrica.

Nas Figuras 5.25 e 5.26, a seguir, são listados os campos de quatro tabelas do Hydrodata que contém os dados solicitados pela ANEEL à empresa Duke.

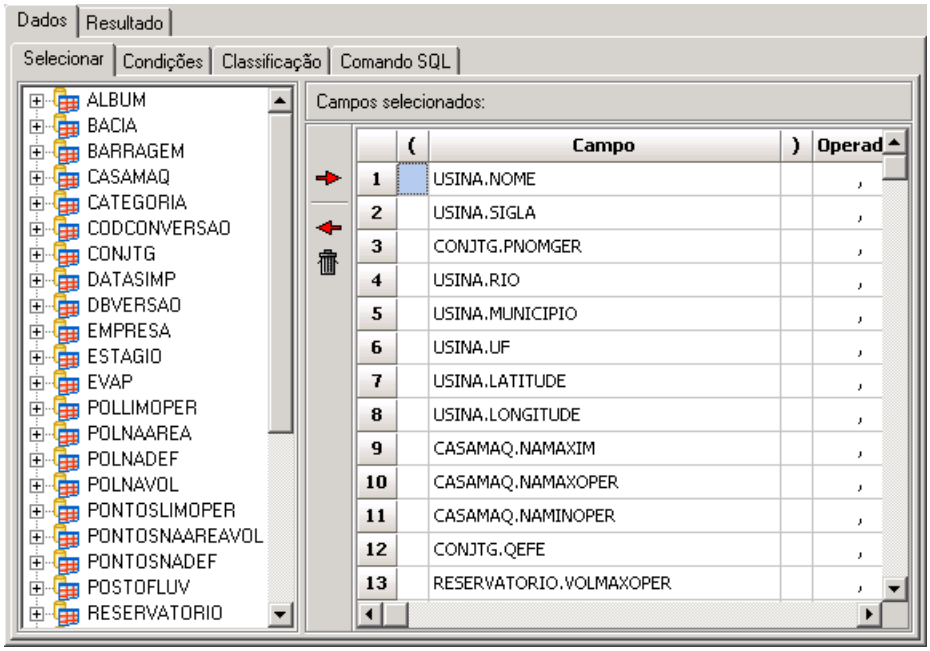


Figura 5.25. Relatórios Avançados – Aba "Selecionar1".

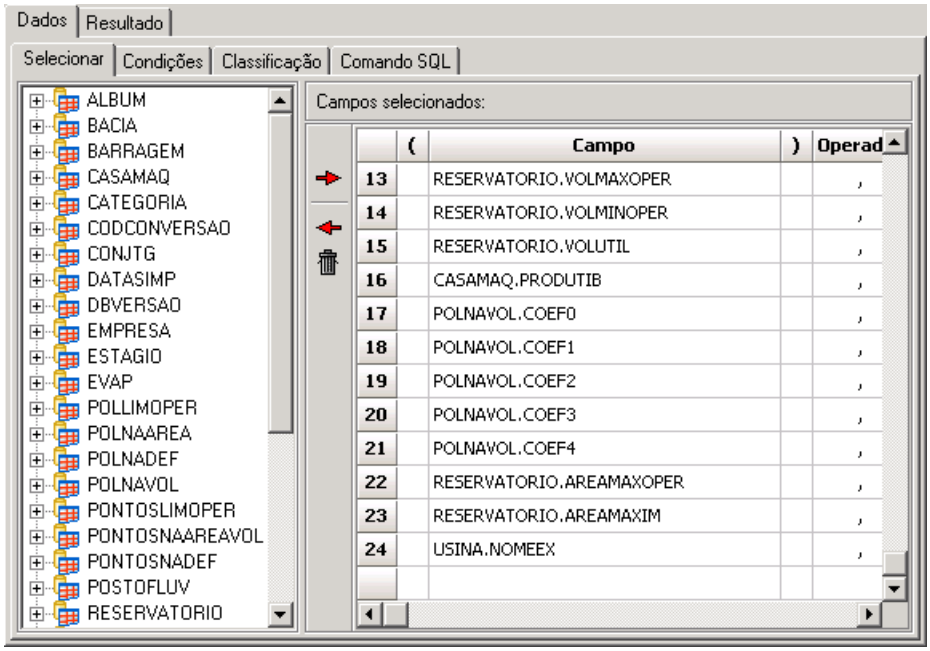


Figura 5.26. Relatórios Avançados – Aba "Selecionar2".

Embora nenhuma restrição tenha sido imposta pela ANEEL, as quatro primeiras condições, da Figura 5.27, são necessárias para o relacionamento das tabelas do HydroData envolvidas nessa consulta (Apêndice A). O último filtro estabelece que sejam exibidas apenas as usinas da empresa "Duke".

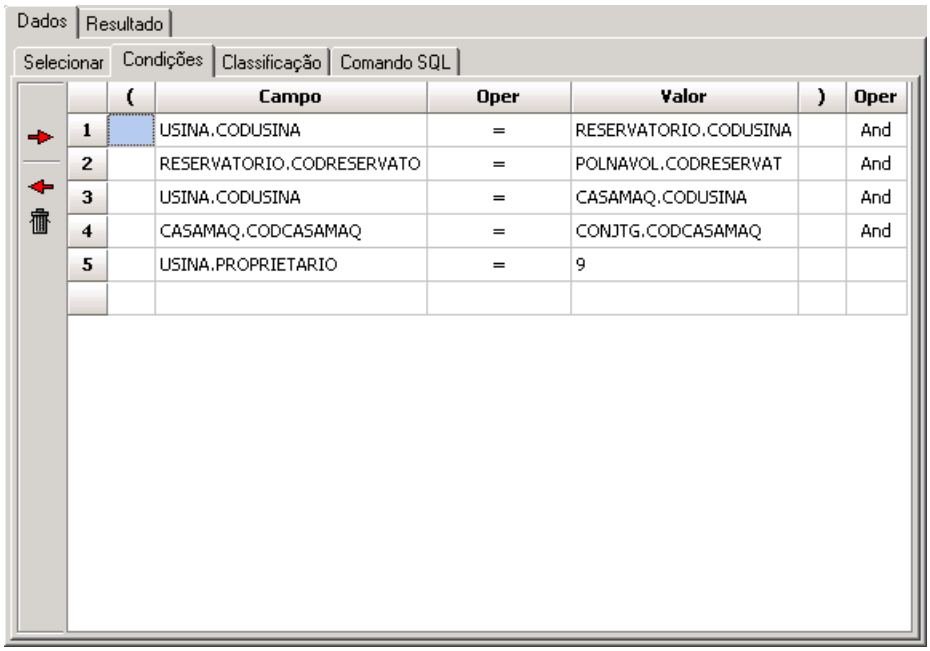


Figura 5.27. Relatórios Avançados – Aba "Condições".

Para facilitar a visualização dos dados esta consulta foi classificada em ordem ascendente do campo “Nome” da tabela “Usina”, mostrada na Figura 5.28.

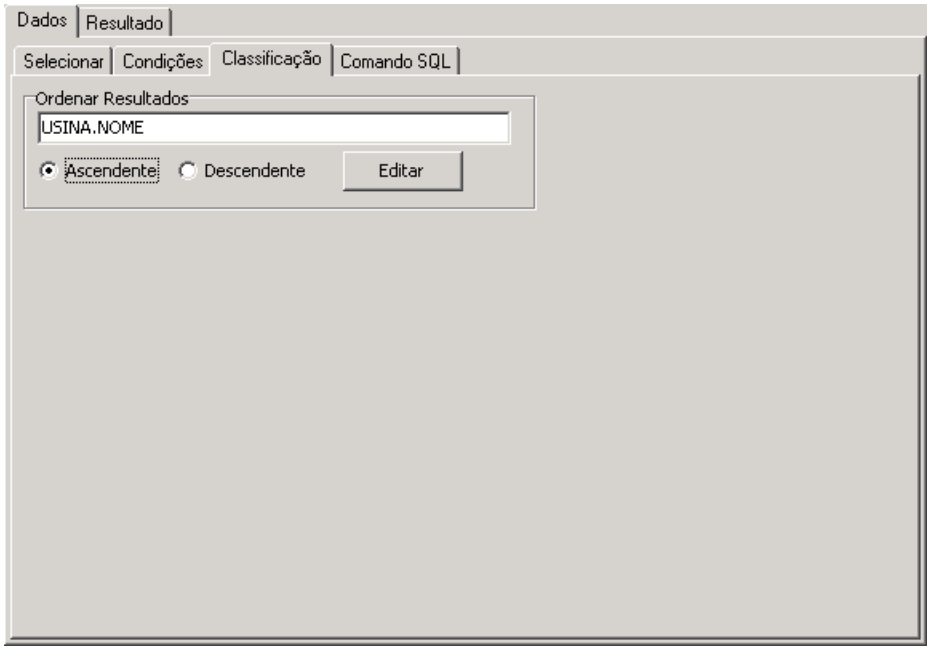


Figura 5.28. Relatórios Avançados – Aba “Classificação”.

O Comando SQL, gerado automaticamente de acordo com as especificações acima e responsável pela execução da consulta, é apresentado na Figura 5.29.

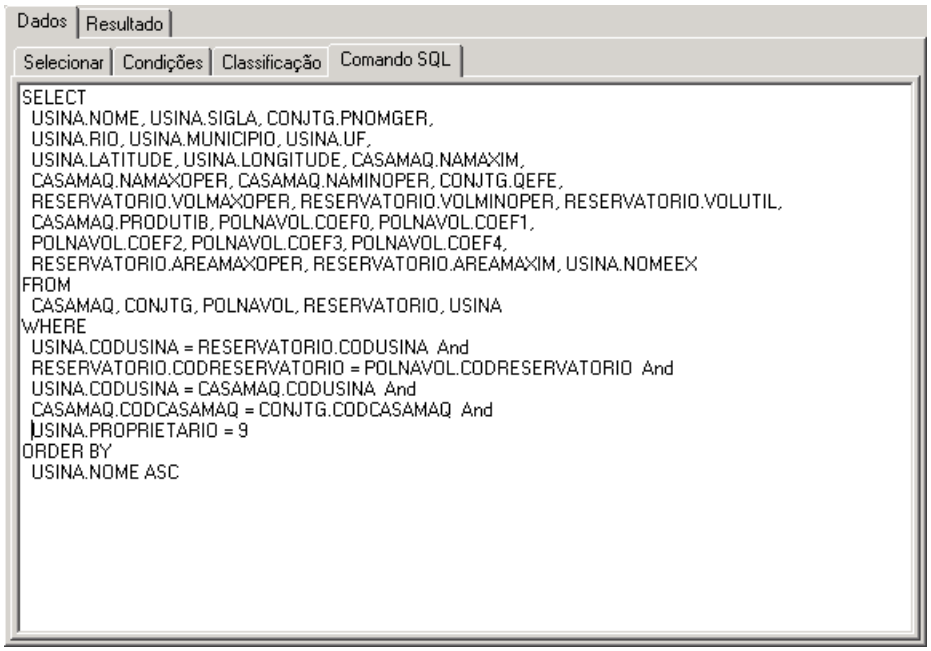


Figura 5.29. Relatórios Avançados – Aba “Comando SQL”.

Nas Figuras 5.30, 5.31 e 5.32, visualiza-se o resultado da consulta.

Dados | Resultado

NOME	SIGLA	PNOMGER	RIO	MUNICIPIO	UF	LATITUDE	LONGITUDE	NAMAXIM	NAMAXOPER
Armando Avellanal Laydner	JUR	48875	Paranapanema	Cerqueira César	SP	S 23:13:00	W 49:14:00	537,599975585938	536,5
Canoas I	CN1	28100	Paranapanema	Cândido Mota	SP/PR	S 22:56:00	W 50:31:00	337,899993896484	334,200012207031
Canoas II	CN2	28100	Paranapanema	Palmital	SP/PR	S 22:56:00	W 50:15:00	356,700012207031	351,350006103516
Chavantes	CHV	103500	Paranapanema	Chavantes	SP/PR	S 23:07:00	W 49:44:00	403,5	398,670013427734
Escola Engenharia Mackenzie	CAP	152000	Paranapanema	Taciba	SP/PR	S 22:39:00	W 51:20:00	295,5	285,200012207031
Escola Politécnica	TAQ	110800	Paranapanema	Sandovalina	SP/PR	S 22:33:00	W 52:00:00	267,399993896484	260,049987792969
Lucas Nogueira Garcez	SAG	17595	Paranapanema	Salto Grande	SP/PR	S 22:54:00	W 50:00:00	374,670013427734	366,600006103516
Lucas Nogueira Garcez	SAG	21000	Paranapanema	Salto Grande	SP/PR	S 22:54:00	W 50:00:00	374,670013427734	366,600006103516
Rosana	ROS	93000	Paranapanema	Rosana	SP/PR	S 22:36:00	W 52:52:00	247,300003051758	241

Resultado: 9 registro(s)

Figura 5.30. Relatórios Avançados – Aba "Resultado".

Dados | Resultado

NAMINOPER	QEFE	VOLMAXOPER	VOLMINOPER	VOLUTIL	PRODUTIB	COEF0
531	187,5	7008	3843	3165	0,00862099975347519	546,200988769531
333,200012207031	227	209,899993896484	179,270004272461	30,6299896240234	0,00911299977451563	341,256500244141
351,100006103516	232	147,970001220703	122,809997558594	25,1600036621094	0,00909699965268373	358,501892089844
397,220001220703	176	8795	5754	3041	0,00877899955958128	441,808502197266
283,600006103516	369	10540,349609375	4815,830078125	5724,51953125	0,00885399989783764	301,627410888672
256,100006103516	540,900024414063	676,830017089844	538,700012207031	138,130004882813	0,00896499957889318	271,579711914063
364,869995117188	140	44,5099983215332	15,1400003433228	29,3699989318848	0,00848099961876869	378,016998291016
364,869995117188	147	44,5099983215332	15,1400003433228	29,3699989318848	0,00848099961876869	378,016998291016
235,5	693	1909	1501,47998046875	407,52001953125	0,00898400042206049	246,444107055664

Resultado: 9 registro(s)

Figura 5.31. Relatórios Avançados – Aba "Resultado".

Dados | Resultado

COEF1	COEF2	COEF3	COEF4	AREAMAXOPER	AREAMAXIM	NOMEEX
0,00400110986083746	-1,26800998145882E-7			449	462,399993896484	Jurumirim
0,0597485415637493	-6,12199291936122E-5			29,3899993896484	29,9500007629395	Canoas I
0,0604750290513039	-6,15878525422886E-5			23,6700000762939	23,9799995422363	Canoas II
0,00484512187540531	-1,34501703996648E-7			400	418,899993896484	Chavantes
0,005178595893085	-2,73339111345194E-7	6,95687301169312E-12		576	623,700012207031	Capivara
0,0232159309089184	-7,18820183465141E-6			74,5800018310547	80,0999984741211	Taquaruçu
0,251794397830963	-0,00313107995316386	1,87262194231153E-5		12	13,5299997329712	Salto Grande
0,251794397830963	-0,00313107995316386	1,87262194231153E-5		12	13,5299997329712	Salto Grande
0,00750992493703961	-7,62998297432205E-7			217,639999389648	219,029998779297	Rosana

Resultado: 9 registro(s)

Figura 5.32. Relatórios Avançados – Aba "Resultado".

Capítulo 6

Conclusões

A utilização da energia elétrica em diversas atividades da sociedade requer a implantação e operação dos sistemas elétricos de potência. No sentido de assegurar um suprimento confiável e de mínimo custo, é de grande importância um adequado planejamento e programação da operação, que considerem aspectos de longo e curto prazo da operação. Essa cadeia de planejamento implementada pelo grupo LSH/DENSIS e gerenciada pelo HydroLab tem forte dependência da qualidade dos dados fornecidos pelo HydroData. É essencial que esses dados estejam corretos para que os modelos de otimização, simulação e previsão de vazões apresentem um comportamento esperado.

O sistema de consultas implementado nesse trabalho mostrou-se eficiente na análise dos dados cadastrais das usinas hidrelétricas brasileiras. Ele pode ser utilizado basicamente em duas situações: para obter informações quaisquer relacionadas às usinas cadastradas no sistema HydroData ou para localizar inconsistências nos dados dessas mesmas usinas.

Para exemplificar a primeira situação, pode-se citar o estudo de caso "Relatórios Avançados" que tinha por objetivo apenas apresentar diversas informações a respeito das usinas da empresa *Duke*. Para exemplificar a segunda situação, pode-se citar o estudo de caso "Canal de Fuga Constante" cuja finalidade foi listar as usinas que apresentavam valores duvidosos para o polinômio que representa a função da defluência versus a cota do canal de fuga.

Uma vez identificada uma inconsistência nos dados de determinada usina, estes podem ser consolidados através do sistema que os gerencia: o HydroData. Dessa maneira, é possível afirma que o HydroConsulta não só analisa dados cadastrais das principais usinas hidrelétricas brasileiras como também facilita a consolidação dos mesmos através do HydroData. Essa consolidação dos dados é importante para os modelos matemáticos embutidos nos algoritmos desenvolvidos pelo grupo LSH/DENSIS.

Outra facilidade trazida pelo HydroConsulta deriva do fato deste encontrar-se integrado ao HydroData, gerenciando e armazenando as consultas na própria base de dados cadastrais das usinas hidrelétricas. Esse fato faz com que as consultas prontas tornem-se disponíveis também para os usuários do HydroLab. Esses usuários, ao invés de selecionarem o conjunto de usinas sobre as quais um determinado modelo vai operar,

podem simplesmente restaurar os dados da consulta que contem os valores desejados. Para isso, basta que a consulta seja selecionada na aba “Consultas” do sistema HydroData.

A implementação do HydroConsulta utilizando-se do paradigma da orientação a objetos prepara-o para sofrer alterações em seus componentes, sem que essa tarefa exija grandes esforços por parte do programador.

O conceito de encapsulamento utilizado nos objetos do HydroConsulta torna possível a alteração de um componente específico, sem a necessidade de compreensão dos demais componentes do programa.

Uma sugestão para a continuidade desse trabalho seria a de tornar invisível, ao usuário do sistema, o trabalho de estabelecer o relacionamento entre as tabelas do HydroData envolvidas na consulta. Com isso, reduz-se a necessidade do usuário apresentar um conhecimento prévio do modelo de relacional do HydroData.

É importante salientar que embora tenha sido desenvolvido com o objetivo de solucionar um problema específico, o HydroConsulta pode acessar e consultar a base de dados de vários outros sistemas, bem como transformar-se, sem grandes esforços, em um gerador de relatórios.

Capítulo 7

Referências Bibliográficas

7 Referências Bibliográficas

[1] **Calvert's, C.** Borland C++ Builder – Sams Publishing, 1997.

[2] **Fortunato, L. A. M. e outros.** Introdução ao Planejamento da Expansão e Operação de Sistemas de Produção de Energia Elétrica – EDUFF / Eletrobrás, 1990.

[3] **Teixeira, S. e Pacheco, X.** Delphi 5 Developer's Guide. Sams Publishing, 2000.

[4] **Cicogna, M. A.** Modelo de Planejamento da Operação Energética de Sistemas Hidrotérmicos a Usinas Individualizadas Orientado por Objetos. Dissertação de Mestrado, FEEC/UNICAMP, 1999.

[5] **Hollingworth, J.**, et al. C++ Builder 5 Developer's Guide. Sams Publishing, 2001.

[6] **Cicogna M. A.** Sistema de Suporte à Decisão para o Planejamento e a Programação da Operação de Sistemas de Energia Elétrica. Tese de Doutorado. FEEC/UNICAMP, dezembro, 2003.

[7] **Bowman J. S., Emerson S.L., Darnovsky M.** The Practical SQL Handbook – Using Structured Query Language. Third Edition. Addison Wesley, 1996.

[8] **Plew, R. R. e Stephens, R. K.** Aprenda em 24 horas SQL. -Segunda edição. Editora Campus, 2000.

[9] **Horstmann, C. S.** Practical Object-Oriented Development in C++ and Java. John Wiley e Sons, 1997.

[10] **Mizrahi, V. V.** Treinamento em Linguagem C – Módulo 1. Editora Makron Books, 1990.

[11] **Lippman, S. B. e Lajoie, J.** C++ Primer, Third Edition. Addison Wesley, 1998.

[12] **Cardoso, C.** UML na prática do problema ao sistema. Editora Ciência Moderna, 2003.

[13] **Breymann, U.** Designing Components with the C++ STL – A New Approach to Programming. Addison-Wesley, 1998.

[14] **Montenegro, F. e Pacheco, R.** Orientação a objetos em C++. Editora Ciência Moderna, 1994.

[15] **Mckay, E. N.** Developing User Interfaces for Microsoft Windows – Practical and effective methods for improving the user experience. Microsoft Press, 1999.

[16] **Ascencio, A. F. G. e Campos, E. A. V.** Fundamentos da Programação de Computadores. Prentice Hall, 2002.

[17] **Elmasri e Navathe.** Fundamentals of Database Systems. Third Edition. Addison Wesley, 2000.

[18] **Alves, W. P.** C++ Builder 6 – Desenvolva Aplicações para Windows. Editora Érica, 2002.

[19] **Cantú, M.** Mastering Delphi 5 – SYBEX, 1999.

[20] **Schildt, H.** C++ guia para iniciantes. Editora Ciência Moderna, 2002.

[21] **Josuttis, N. M.** The C++ Standard Library – A Tutorial and Reference. Addison Wesley, 1999.

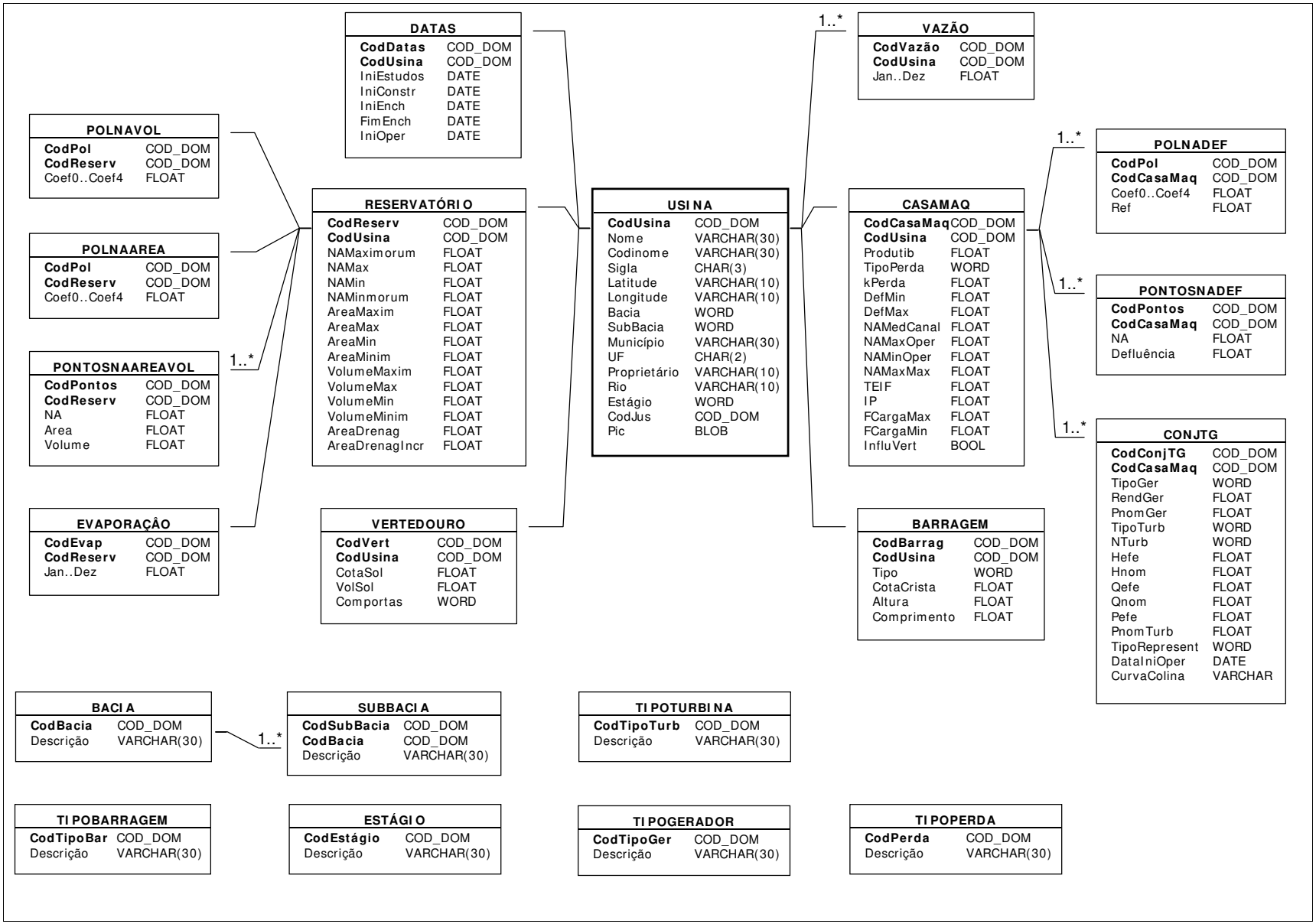
[22] **Reisdorph, K.** et al. Borland C++ Builder 4 Unleashed. Sams Publishing, Borland Press, 1999.

[23] **Vinhal, C. D. N**. Sistemas de apoio à decisão para o planejamento da operação energética de Sistemas de Energia Elétrica. Tese de Doutorado, FEEC/UNICAMP, 1998.

Apêndice A

Modelo Relacional do HydroData

Nesse apêndice apresenta-se o modelo relacional do banco de dados das usinas hidrelétricas brasileiras. A consulta a esse modelo torna-se necessária na especificação dos relacionamentos entre as tabelas envolvidas numa consulta. Esses relacionamentos são feitos na Sub-Aba “Condições” do HydroConsulta.


DATAS
CodDatas COD_DOM
CodUsina COD_DOM
IniEstudos DATE
IniConstr DATE
IniEnch DATE
FimEnch DATE
IniOper DATE
VAZÃO
CodVazão COD_DOM
CodUsina COD_DOM
Jan..Dez FLOAT
1..*
POLNAVOL
CodPol COD_DOM
CodReserv COD_DOM
Coef0..Coef4 FLOAT
POLNAAREA
CodPol COD_DOM
CodReserv COD_DOM
Coef0..Coef4 FLOAT
PONTOSNAAREAVOL
CodPontos COD_DOM
CodReserv COD_DOM
NA FLOAT
Area FLOAT
Volume FLOAT
1..*
EVAPORAÇÃO
CodEvap COD_DOM
CodReserv COD_DOM
Jan..Dez FLOAT
RESERVATÓRIO
CodReserv COD_DOM
CodUsina COD_DOM
NAMaximorum FLOAT
NAMax FLOAT
NAMin FLOAT
NAMinmorum FLOAT
AreaMaxim FLOAT
AreaMax FLOAT
AreaMin FLOAT
AreaMinim FLOAT
VolumeMaxim FLOAT
VolumeMax FLOAT
VolumeMin FLOAT
VolumeMinim FLOAT
AreaDrenag FLOAT
AreaDrenagIncr FLOAT
VERTEDOURO
CodVert COD_DOM
CodUsina COD_DOM
CotaSol FLOAT
VolSol FLOAT
Comportas WORD
USINA
CodUsina COD_DOM
Nome VARCHAR(30)
Codinome VARCHAR(30)
Sigla CHAR(3)
Latitude VARCHAR(10)
Longitude VARCHAR(10)
Bacia WORD
SubBacia WORD
Município VARCHAR(30)
UF CHAR(2)
Proprietário VARCHAR(10)
Rio VARCHAR(10)
Estágio WORD
CodJus COD_DOM
Pic BLOB
CASAMAQ
CodCasaMaq COD_DOM
CodUsina COD_DOM
Produtib FLOAT
TipoPerda WORD
kPerda FLOAT
DefMin FLOAT
DefMax FLOAT
NAMedCanal FLOAT
NAMaxOper FLOAT
NAMinOper FLOAT
NAMaxMax FLOAT
TEIF FLOAT
IP FLOAT
FCargaMax FLOAT
FCargaMin FLOAT
InfluVert BOOL
1..*
POLNADEF
CodPol COD_DOM
CodCasaMaq COD_DOM
Coef0..Coef4 FLOAT
Ref FLOAT
1..*
PONTOSNADEF
CodPontos COD_DOM
CodCasaMaq COD_DOM
NA FLOAT
Defluência FLOAT
1..*
CONJTG
CodConjTG COD_DOM
CodCasaMaq COD_DOM
TipoGer WORD
RendGer FLOAT
PnomGer FLOAT
TipoTurb WORD
NTurb WORD
Hefe FLOAT
Hnom FLOAT
Qefe FLOAT
Qnom FLOAT
Pefe FLOAT
PnomTurb FLOAT
TipoRepresent WORD
DataIniOper DATE
CurvaColina VARCHAR
BARRAGEM
CodBarrag COD_DOM
CodUsina COD_DOM
Tipo WORD
CotaCrista FLOAT
Altura FLOAT
Comprimento FLOAT
BACIA
CodBacia COD_DOM
Descrição VARCHAR(30)
1..*
SUBBACIA
CodSubBacia COD_DOM
CodBacia COD_DOM
Descrição VARCHAR(30)
TIPOTURBINA
CodTipoTurb COD_DOM
Descrição VARCHAR(30)
TIPOBARRAGEM
CodTipoBar COD_DOM
Descrição VARCHAR(30)
ESTÁGIO
CodEstágio COD_DOM
Descrição VARCHAR(30)
TIPOGERADOR
CodTipoGer COD_DOM
Descrição VARCHAR(30)
TIPOPERDA
CodPerda COD_DOM
Descrição VARCHAR(30)

Apêndice B

Relatório Aneel

Nesse apêndice encontra-se o ofício enviado pela ANEEL (Agência Nacional de Energia Elétrica) ao presidente da empresa Duke.

AGÊNCIA NACIONAL DE ENERGIA ELÉTRICA

Ofício nº 449 /2003-SIH/ANEEL

Brasília, 13 de novembro de 2003.

A Sua Senhoria o Senhor
Michael Lawrense Dulaney
Presidente
Duke Energy Internacional, Geração Paranapanema S. A
São Paulo - SP

Assunto: Atualização do Banco de Dados da Compensação Financeira-indicação de Técnico.

Senhor Presidente,

A Agência Nacional de Energia Elétrica - ANEEL tem como procedimento atualizar periodicamente os dados utilizados para o cálculo dos valores a serem recolhidos à título de Compensação Financeira pela Utilização de Recursos Hídricos para fins de Geração de Energia Elétrica, através da validação das informações disponíveis em nosso banco de dados. Visando tornar este procedimento mais ágil, está sendo disponibilizado um sistema de coleta de dados no site da ANEEL.

Desta forma, solicitamos a indicação de um técnico que será credenciado para acessar esse sistema e terá a responsabilidade, quando o sistema estiver operacional, pela validação das informações relacionadas abaixo:

- **Nome da Usina**
- **Código da Usina**: Código do SIPOT
- **Potência em KW**: Potência nominal total dos geradores
- **Nome do Rio**:
- **Município**: onde encontra-se a casa de máquinas
- **Unidade da Federação**: onde encontra-se a casa de máquinas
- **Latitude da casa de máquinas**: Latitude expressa em graus, minutos e segundos. O valor deve ser digitado sem caracteres separadores.
- **Longitude da casa de máquinas**: Longitude expressa em graus, minutos e segundos. O valor deve ser digitado sem caracteres separadores.

SGAN - Guadra 603 /Módulos "I" e "J"
CEP 70830-030 - Brasília - DF - Brasil
Tel. 55 (61) 426 5600
Ouvidoria 0800 612010
www.anell.gov.br

(Fls. 02 do Ofício nº 449 /2003-SIH/ANEEL, de 13/11/2003)

- **Nível D'Água Máximo Maximorum**: Deve ser expresso em metros
- **Nível D'Água Máximo Operacional**: Deve ser expresso em metros
- **Nível D'Água Mínimo Operacional**: Deve ser expresso em metros
- **Queda Bruta**: Deve ser expresso em metros
- **Volume Máximo Operacional**: Volume armazenado no reservatório da usina, expresso em hm3, correspondente ao Nível Máximo Operacional.
- **Volume Mínimo Operacional**: Volume armazenado no reservatório da usina, expresso em hm3, correspondente ao Nível Mínimo Operacional.
- **Volume Útil**: Volume armazenado no reservatório da usina, expresso em hm, entre as cotas do Nível Máximo Operacional e a cota do Nível Mínimo Operacional.
- **Produtibilidade Especifica [MW/m3/s/m]**: Produtibilidade da Usina
- **Curva Cota [m] x Volume [hm3]**: Coeficientes do polinômio da Curva Cota x Volume do reservatório da usina.
- **Área do reservatório**: Área da superfície do espelho d'água do reservatório da UHE, expressa em Km2, no nível d'água Máximo Maximorum, incluindo a calha do rio.
- **Área Inundada dos municípios**: Área da superfície do município afetada diretamente pelo espelho d'água do reservatório do empreendimento hidrelétrico, expressa em Km2, no nível d'água Máximo Maximorum, incluindo a calha do rio.

Atenciosamente,

HÉLVIO NEVES GUERRA
Superintendente de Estudos e Informações Hidrológicas

SGAN - Guadra 603 /Módulos "I" e "J"
CEP 70830-030 - Brasília - DF - Brasil
Tel. 55 (61) 426 5600
Ouvidoria 0800 612010
www.anell.gov.br