

Publicação Especial

Nº 09
2000

Um Banco de Dados Para a Cosmologia
Observacional

João Luiz Kohl Moreira

Tese de Doutorado
Rio de Janeiro, Setembro de 2000

**Ministério da Ciência e Tecnologia
Observatório Nacional**

**Um Banco de Dados para a Cosmologia
Observacional**

João Luiz Kohl Moreira

Orientador: Dr. Reinaldo R. de Carvalho

Banca: Dr. Reinaldo R de Carvalho (ON/MCT, presidente)
Dr. Sayd Codina Landaberry (ON/MCT)
Dr. Ramachrisna Teixeira (IAG/USP)
Dr. Marcos Dias (IAG/USP)
Dr. Albert Bruch (LNA/MCT)

**Tese apresentada para a obtenção do grau de
Doutor em Astrofísica
pelo Observatório Nacional**

o6

Rio de Janeiro – Setembro de 2000

Agradecimentos

Ao colega e amigo Reinaldo R. de Carvalho, que me propôs esse tema e orientou esse trabalho: agradeço pela idéia que me conduziu a um tema extraordinário e desconhecido para mim. Reconheço, igualmente, sua paciência e dedicação no trabalho de orientação. Sua constante preocupação com a excelência e o estado da arte serviu de referência em todos os momentos de desenvolvimento dessa tese.

A minha esposa Sonia e meu filho Guido pela paciente convivência e confiança.

A Sonia devo o minucioso trabalho de revisão do texto. Sem ele, essa tese não seria legível.

A minha mãe Vera cujo carinho e força de caráter se mostraram nos momentos mais difíceis e serviram de inspiração nessa jornada.

Aos colegas do Departamento de Astrofísica e muitos do ON cujo apoio e torcida serviram de suporte e não me deixaram esmorecer.

Homenagem àqueles que marcaram minha vida e já se foram:

meu tio Nelson (*de los indispensables*),
meu pai, Mauricy, cuja vida se apagou,
minha irmã Sônia (um sonho interrompido),
minha sobrinha Natascha (pedaço arrancado de mim), e
meus amigos Airton Barbosa, Luiz B. F. Clauzet, Luiz C. Martins
e Raimundo T. Mendes.

Resumo

Neste trabalho, apresentamos um banco de dados, denominado SKICCOSMO (*Sky Integrated Catalogs for Cosmology*), com aplicação em Cosmologia Observacional. Em função de sua concepção, o SKICCOSMO presta-se também para outros campos da astronomia. Sua constituição básica tem por origem os dados do DPOSSII (Reid, 1991, [48]). Os dados das placas digitalizadas são reduzidos, os objetos são identificados e classificados no contexto do SKICAT (Weir, 1994, [66]) e, em seguida, transferidos para o SKICCOSMO. A principal característica do SKICCOSMO é a sua condição integralmente relacional e em condições de avançar no sentido da estrutura objeto-relacional. Ao contrário do SKICAT, que exige que as operações sejam feitas localmente, o SKICCOSMO oferece condição de operar via página *WEB*, pela Internet, sendo assim um banco de dados em sintonia com a tendência atual: ser disponível para a comunidade astronômica mundial. O SKICCOSMO permite, também, o acréscimo de outros catálogos e bancos de dados astronômicos à sua estrutura, bem como permite sua integração a redes de bancos de dados globais. No sentido de facilitar a interação com o usuário, foi desenvolvido um utilitário que “traduz” uma consulta construída com instruções simples numa consulta em SQL, linguagem universal dos bancos de dados relacionais.

Apresentamos também resultados preliminares do programa que está sendo realizado em colaboração com pesquisadores de Caltech na procura de QSOs localizados em altos desvios para o vermelho. Este programa faz extensivo uso do programa desenvolvido para *match* os campos do POSS-II. Como discutido neste trabalho a otimização do processo de *match* tornou possível tornar a busca destes objetos um programa competitivo dentro do estudo destes sistemas. Este projeto está não somente gerando dados sobre a época inicial do Universo mas também permitindo o estudo dos absorvedores localizados na linha de visada do QSO.

Abstract

We present a new database oriented for Observational Cosmology called SKICCOSMO (Sky Integrated Catalogs for Cosmology). The new design proposed here, makes its use in other branches of astronomy feasible. We use the DPOSS catalogs as our basic source of information to generate SKICCOSMO (Reid 1991, [48]). The plate data are reduced using a specifically designed packaged called SKICAT (Weir 1994, [66]) and subsequently stored in SKICCOSMO, with proper calibration and classifications steps done beforehand. It is a relational database structured in a such a way that can be easily modified to be also object oriented. SKICCOSMO, as oposed to SKICAT, allows interaction via WEB, following the footsteps of the modern tendency: being available to the general astronomical community. This database incorporates the facility of not only interacting but also exchanging data with other DBs. In order to make easy for the general user to access SKICOSMO, we developed a special language which allows the interface between the way the user wants to retrieve some information from the DB and the actual SQL instruction.

We also present some preliminary results of an ongoing program searching for QSOs at high redshift ($z > 4$), in collaboration with a group from Caltech. This project makes extensive use of the matching program presented in this thesis. The optimization of the matching process was an essential step in making this study possible and competitive, generating data about the early epoch of the universe, and allowing the study of the absorbers along the line of sight of the QSOs.

Sumário

Agradecimentos	iii
Resumo	v
Abstract	vii
1 Bancos de Dados Relacionais e a Astronomia	1
1.1 Introdução	1
1.2 A Evolução dos Bancos de Dados	4
1.2.1 Os diferentes sistemas de gerenciamento dos dados	4
1.2.2 Banco de Dados Sequencial	4
1.2.3 Banco de Dados Hierárquico	6
1.2.4 Banco de Dados Relacional	8
1.2.5 Banco de Dados Orientado Objeto	9
1.2.6 Banco de Dados Objeto-Relacional	10
1.2.7 Qual é a vantagem dos DBMS's?	11
1.3 O Conceito de "Objetividade"	12
1.3.1 Introdução	12
1.3.2 A Programação Sequencial	13
1.3.3 A Programação Estruturada	15
1.3.4 A Programação Objeto (POO)	20
1.3.5 A Extensão da "SQL" à Objetividade	24
1.4 Fundamentos do Banco de Dados Relacional	26
1.4.1 A terminologia matemática e seu correspondente nos "DBMS"s	26
1.4.2 Definições	26
1.4.2.1 Elementos do Esquema Conceitual	27
1.4.2.2 Relacionamento	29

1.4.2.3	Modelagem dos Dados	31
1.4.2.4	Detalhamento das Entidades	32
1.4.2.5	Diagrama da Visão do Usuário	32
1.4.2.6	A Chave Primária e a Chave Estrangeira	33
1.4.2.7	Ponto de Partida para a Construção do Modelo	34
1.4.2.8	O Estágio da Normalização	35
1.4.2.9	As Formas Normais	35
1.4.2.10	As Regras da Normalização	36
1.4.2.11	Integridade e Segurança	38
1.4.2.12	A Linguagem Estruturada de Consulta (SQL)	39
1.4.2.13	Os Índices	41
1.4.3	A Organização de um Banco de Dados orientado para a Astronomia	43
1.5	O Banco de Dados SKICAT	44
2	Os Bancos de Dados na Astronomia	51
2.1	O que é e por que queremos um Banco de Dados?	51
2.1.1	O que é um banco de dados?	51
2.1.2	Por que queremos um banco de dados?	52
2.2	Os Tipos de Banco de Dados na Astronomia	53
2.3	Alguns Bancos de Dados "On line"	54
2.3.1	SIMBAD	55
2.3.2	IUE	55
2.3.3	STARVIEW	57
2.3.4	IPAC/NED	59
2.3.5	ADF	61
2.3.5.1	AMASE	61
2.3.5.2	IMPREcSS	62
2.3.6	Conjunto de ADC	62
2.3.7	O Serviço HEASARC	64
2.3.8	COBE	66
2.3.9	Conjunto MAST	68
2.3.10	IRSA	68
2.3.11	DaBIAS/LNA	71
2.3.12	GSC	71
2.4	Características dos Banco de Dados na Astronomia	73

3	Transferindo os dados	77
3.1	O SKICAT	77
3.1.1	A Estrutura dos Catálogos do SKICAT	77
3.1.1.1	Os Catálogos registrados e os Catálogos <i>on-line</i>	77
3.1.1.2	As Tabelas de Características (<i>Features</i>)	78
3.1.1.3	As Tabelas de Cabeçalho (<i>Header</i>)	78
3.1.1.4	A Tabela das Características <i>Matched</i> (<i>MatchedFeatures Table</i>)	79
3.1.1.5	A Tabela dos Catálogos <i>Matched</i> (<i>MatchedCatalogs</i>)	79
3.1.1.6	A Tabela de Contagem de <i>Matching</i> (<i>MatchCount</i>)	79
3.1.1.7	Tabela de Objetos (<i>Objects</i>)	79
3.1.1.8	A Tabela dos Catálogos (<i>Catalogs</i>)	79
3.1.1.9	A Tabela “ <i>CatVersions</i> ”	80
3.1.1.10	A Tabela “ <i>CatTypes</i> ”	80
3.1.1.11	A Tabela “Placas” (<i>Plates</i>) e CCDs	80
3.1.1.12	A Tabela “ <i>MatchProc</i> ”	81
3.1.1.13	A Tabela “ <i>MatchColumn</i> ”	81
3.1.1.14	A Operação com o SKICAT	81
3.1.2	Os Aplicativos disponíveis no SKICAT	84
3.1.2.1	“ <i>Stored Procedures</i> ”	84
3.1.2.2	Utilitários	84
3.2	Modelo do SKICCOSMO	85
3.2.1	Legado de Catálogos	86
3.2.2	Descrição das Diferentes Entidades no Banco de Dados	86
3.2.2.1	A Tabela <i>Objects</i>	86
3.2.2.2	POSS_II: Tabelas de <i>features</i>	87
3.2.2.3	As tabelas <i>Devices</i> e <i>Filters</i>	91
3.2.2.4	A tabela <i>Classifications</i>	92
3.2.2.5	A tabela <i>Cat_techniques</i>	93
3.2.2.6	A tabela DPOSSII_Headers	93
3.2.2.7	As Tabelas DPOSSII_xxx_Head	96
3.2.2.8	A Tabela de Comentários DPOSSII	99
3.3	A “Vista” (<i>view</i>) <i>Features</i>	100
3.4	Tabelas de auxílio à administração	101

3.5	A Chave Primária	102
3.5.1	Como definir a chave primária?	102
3.5.2	A imprecisão astrométrica	102
3.5.2.1	Os métodos de <i>matching</i> presentes na literatura	103
3.5.2.2	O algoritmo de <i>matching</i> desenvolvido no interior do SKICAT'104	104
3.5.3	Procurando Alternativas	105
3.5.4	O problema da identificação de um campo para o astrônomo obser- vational	105
3.5.5	A transposição do problema de identificação para o algoritmo	106
3.5.6	O estimador de <i>Matching</i>	106
3.5.7	Termos para uma relação biunívoca	109
3.5.8	A estimativa de distância e sua variância	110
3.5.9	A identificação dos objetos	111
3.5.10	Ambigüidade na identificação	111
3.5.11	Identificação ponto a ponto	111
3.5.12	O histograma das diferenças ponto a ponto	112
3.5.12.1	Critério de significância de <i>matching</i>	113
3.5.12.2	Falha no processo de <i>Matching</i>	115
3.5.12.3	<i>Matching</i> em campos densamente populados	115
3.5.12.4	Limites do método	120
3.5.13	Aplicações em casos reais	123
3.5.13.1	Catálogos FK5 e BSC5	123
3.5.14	Tempo de Cálculo	126
3.5.15	O Processo de <i>matching</i> em duas dimensões	128
3.5.15.1	Algoritmo	128
3.5.15.2	Critério para o intervalo de confiança	131
3.5.16	Comparação entre Métodos	131
3.6	A Estratégia de Transferência	131
4	Operando com o SKICCOSMO	135
4.1	O Ambiente de Consulta	135
4.1.1	Introdução	135
4.1.2	Estratégia Operacional	136
4.1.3	A Página <i>Web</i>	137
4.1.4	A Página de Construção de Consultas	141

4.1.5	As Janelas para Construção de Consultas	146
4.2	A Linguagem de Consulta	147
4.2.1	Convenção	147
4.2.2	Conceitos	148
4.2.3	A Definição dos Atributos	149
4.2.3.1	Indexação	153
4.2.3.2	Campos com múltipla indexação	154
4.2.3.3	Indexação Indireta	154
4.2.3.4	Indexação na Classificação do Objeto	155
4.2.3.5	Indexações esparsas	155
4.2.3.6	Funções do Compilador	156
4.2.3.7	Funções Agregadas	157
4.2.3.8	Nomes de Variáveis	158
4.2.3.9	Atributos Expostos e Escondidos	158
4.2.4	As Condições	159
4.2.4.1	Operador <i>or</i>	160
4.2.4.2	Condições Assumidas por Falta (<i>default</i>)	160
4.2.4.3	Inclusão e Exclusão	161
4.2.5	Erros e Limitações	162
4.3	Exemplos	162
4.3.1	Exemplo n. 1: <i>Total Mag F J N near equinox</i>	164
4.3.2	Exemplo n. 2: <i>Finding chart for Field F 836</i>	167
4.3.3	Exemplo n. 3: <i>Color - Color Diagram for stars</i>	168
4.3.4	Exemplo n. 4: <i>Choose galaxies in J and N but not in F</i>	169
4.3.5	Exemplo n. 5: <i>Finding Chart for stars where $DEX(F-N) > 10$</i>	170
4.3.6	Exemplo n. 6: <i>Stars & galaxies within a small area (N-band)</i>	171
4.3.7	Exemplo n. 7: <i>Round Galaxies from F 861</i>	172
4.3.8	Exemplo n. 8: <i>QSO at high redshift s</i>	174
5	Utilizando o DPOSS na Procura de QSOs a $z > 4$	177
5.1	Introdução	177
5.2	O Levantamento Fotométrico	179
5.3	O Levantamento Espectroscópico	183
5.4	Discussão	183

6	Conclusões e Perspectivas	187
6.1	Bancos de Dados	188
6.2	BDR, POO e BDOR	188
6.3	Comparação e <i>matching</i> de Campos	189
6.3.1	Campos Populosos e Densamente Povoados	190
6.4	A Disponibilização dos Dados	190
6.4.1	A Recuperação dos Dados	190
6.4.2	O Observatório Virtual	191
6.5	O que há a fazer no SKICCOSMO	194
6.5.1	O Problema astrométrico no SKICCOSMO	195
6.5.2	Novas Ferramentas de Consulta	195
6.5.3	Ferramentas Gráficas	196
6.5.4	Integração de Catálogos do Usuário com o SKICCOSMO	196
6.5.5	Integração do SKICCOSMO ao Observatório Virtual	198
	Referências Bibliográficas	199
	Apêndices	205
A	PRESKI	205
B	CAROL (<i>Carry On-Line</i>)	221

Lista de Tabelas

1.1	Termos matemáticos e usuais.	26
1.2	Tabela de Modelo de Relacionamento entre Entidades	31
2.1	Resumo das características de alguns bancos de dados na internet.	74
3.1	Valores da estatística Kolmogorov-Smirnov para alguns campos caracterís- ticos.	126
3.2	Regiões no céu cobrindo números aproximadamente iguais de objetos.	127
4.1	Funções do tipo “macro”.	156
4.2	Funções agregadas do Informix	157
4.3	Resumo do SkiccQL	163
5.1	Campos estudados para os candidatos a QSOs.	181

Lista de Figuras

1.1	Esquema dos diferentes tipos de banco de dados.	3
1.2	Quadro de evolução dos bancos de dados, destacando-se cinco paradigmas.	12
1.3	Estrutura e ponteiro.	18
1.4	Programação estruturada.	19
1.5	Direções possíveis que o analista segue ao desenvolver um programa.	22
1.6	Tipos de entidades em um banco de dados na astronomia.	29
1.7	Exemplo de modelo de Relacionamento de Entidades (<i>ER-Model</i>)	32
1.8	Atributos relevantes à entidade Catálogo	33
1.9	Exemplo de “vista” para a astrometria.	33
1.10	Decomposição de entidades.	36
1.11	Atributos indexados.	37
1.12	Árvore binária.	42
2.1	Janela de consulta do SIMBAD.	56
2.2	Janela de consulta no IUE.	58
2.3	Página do IPAC/NED.	60
2.4	Estrutura do AMASE na parte dos objetos astronômicos (Cheung, 1995, [11])	63
2.5	Uma das janelas de consulta do AMASE.	64
2.6	Consulta no IMPREeSS.	65
2.7	A janela de consulta por palavra chave do HEASARC	67
2.8	Página de introdução do MAST e as alternativas de consulta.	69
2.9	Consulta no IRSA.	70
2.10	Janela para consulta por objetos ou coordenadas do DaBIAS/LNA.	72
3.1	Diagrama E-R para o SKICAT	82
3.2	Atributos das tabelas do SKICCOSMO.	88
3.3	Tabelas auxiliares à administração do SKICCOSMO	101

3.4	Histogramas das diferenças para os campos J823 e F823 no <i>footprint</i> 1,1.	116
3.5	Faixas de <i>matching</i>	117
3.6	Distribuição uniforme.	121
3.7	Histograma das diferenças do campo uniforme.	124
3.8	Histogramas de distribuição e das diferenças.	125
3.9	Histograma das diferenças do <i>matching</i> entre o BSC5 e FK5 para a região ($6^h \dots 9^h, -30^\circ \dots 30^\circ$).	127
3.10	Rede em três pontos para a carga do SKICCOSMO	133
4.1	Esquema da distribuição em três camadas	137
4.2	Esquema do SKICCOSMO no Observatório Nacional	138
4.3	Apresentação da página <i>Web</i> do SKICCOSMO	139
4.4	Página após o <i>e-mail</i> ser reconhecido pelo SKICCOSMO	140
4.5	Visão da 1a. e 2a. parte da página de construção de consultas	142
4.6	Visão da 3ª parte da página de construção de consultas	143
4.7	Histogramas da magnitude total nas bandas F , N e J de um campo do POSS-II.	145
4.8	Aspecto da consulta <i>Total Mag F J N near equinox</i>	164
4.9	Diagrama Cor - Cor para o campo 826 do POSS-II.	175
5.1	Diagrama (g-r) versus (r-i)	180
5.2	Espectros de candidatos a QSO para $4.0 < z < 5.0$	184

Capítulo 1

Bancos de Dados Relacionais e a Astronomia

1.1 Introdução

A técnica de preparar a informação para uso de um certo público, que dela se serve para tomar uma decisão, evoluiu na mesma velocidade que propriamente a informática (Chapnick, 1997, [10]). A ciência, assim como todas as atividades baseadas em processos investigativos, para estar em condições de promover a associação das diferentes grandezas, precisa passar pelo estágio da classificação dos objetos sob seu estudo (Wightman, 1950, [71]). A organização dos dados segundo um critério coerente, portanto, é de fundamental importância para o cientista chegar a um termo em seus estudos. Não é diferente na astronomia.

Não estaremos totalmente enganados se dissermos que o primeiro ato científico na astronomia foi a catalogação dos objetos celestes visíveis. Quando Hiparcos classificou os objetos celestes segundo sua magnitude, ele estava construindo um catálogo (Jeans, 1948, [32]). Os mapas celestes que foram sendo construídos no decorrer da história podem, e devem, ser considerados catálogos. Como resultado, catálogos com alto grau de precisão e coerência como os da série **FK**, ou com grande grau de completeza como os da série **SAO**, estão à disposição da comunidade científica há muito tempo antes da informatização. O advento dos computadores cria em consequência, a idéia de armazenar a informação em unidades de memória de massa. O “banco de dados”, como diz o nome, refere-se à informação armazenada em um “local”, recuperável com a intervenção de ferramentas computacionais. Na astronomia, foram sendo desenvolvidos tanto os dados obtidos de programas específicos (*surveys*, ex.: POSS-II e ESO), como iniciativas de **compilação** de

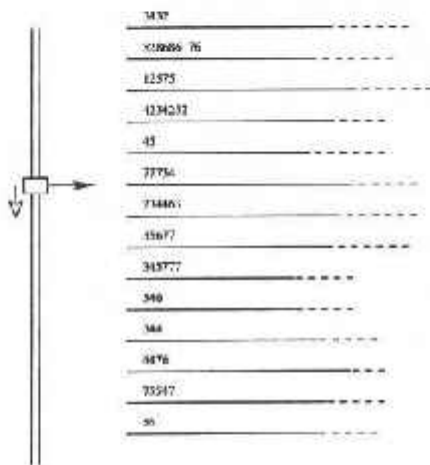
catálogos e publicações (ex.: catálogos integrados do ADS-NASA, Weiss & Good, 1991[69]; o SIMBAD, Egret *et al*, 1991, [19]; NED, Helou *et al*, 1991, [26], etc), e hoje temos uma verdadeira miríade de bancos de dados disponíveis, cada um oferecendo um universo de informações específicas provenientes, tanto de técnicas observacionais, quanto de um critério de classificação ou mesmo de uma sonda ou telescópio espacial.

Os primeiros bancos de dados na astronomia foram construídos com base nos desenvolvimentos efetuados nos laboratórios de computação dos centros que trataram de sua criação. Assim, a julgar pela leitura de Albrecht & Egret (1991) [1], a maioria dos bancos de dados disponibilizados foram construídos segundo uma arquitetura e linguagens de consulta próprias. Com a evolução dos gerenciadores de bancos de dados comerciais e a universalização dos procedimentos de construção de consultas, a tendência é a adequação dos bancos de dados da astronomia às estruturas dos produtos encontrados no mercado (Vansteenberg & Green, 1991, [64]). A evolução dos bancos de dados encontrados no mercado passa, assim, a ter interesse na astronomia. Isso é o que está exposto brevemente em 1.2.

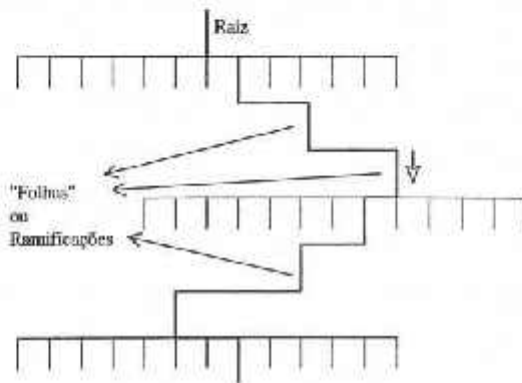
Existem diversas formas de construção de um banco de dados aplicado à astronomia. O ponto de partida e a motivação podem ser (Vansteenberg & Green, 1991, [64]):

- O conjunto dos dados obtidos de um programa observacional (*survey*) em um telescópio ou técnica observacional (POSS-II, ESO, IUE, COBE, etc);
- O conjunto dos dados obtidos por uma comunidade específica num esforço combinado por diversas instituições e grupos de pesquisa de um objeto específico ou classe de objetos (*International Halley Watch*, SN 1987a, etc);
- O centro de compilação de dados a partir de publicações e/ou observações de uma classe de objetos (SIMBAD, NED, ADS, etc);
- A disponibilização ao público do conjunto de observações levadas a cabo em um observatório ou telescópio pela comunidade (Hubble, LNA, etc).

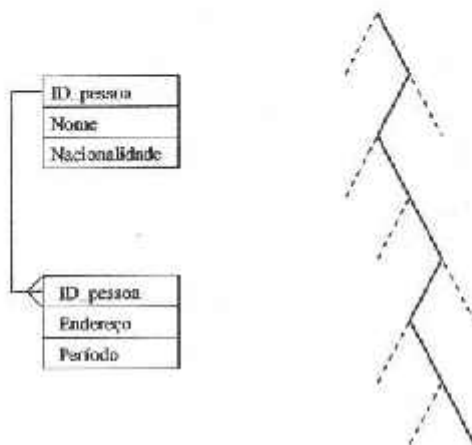
Por sua construção, esses bancos de dados possuem características quanto à *integridade de domínio* que merecem comentário. Essa é uma discussão que se colocará após estudarmos as propriedades e diretrizes dos bancos de dados modernos. Portanto, a esse assunto retornaremos em 2.4.



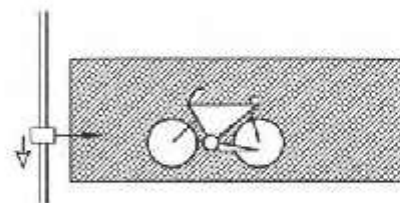
a) Banco de Dados Seqüencial



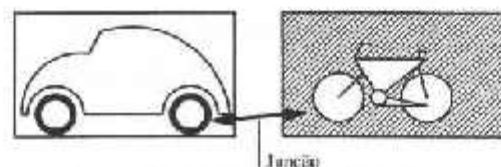
b) Banco de Dados Hierárquico



c) Banco de Dados Relacional e a Árvore Binária



d) Banco de Dados Orientado Objeto



e) Banco de Dados Objeto-Relacional

Figura 1.1: Esquema dos diferentes tipos de banco de dados.

a) Banco de dados seqüencial; b) Banco de dados hierárquico; c) Banco de dados relacional; d) Banco de dados orientado objeto; e) Banco de dados objeto-relacional.

1.2 A Evolução dos Bancos de Dados

O desenvolvimento da técnica de armazenagem passou por vários estágios caracterizados por conjuntos de conceitos que usualmente são chamados **paradigmas** (Chapnick, 1997, [10]). A ciência dos bancos de dados, desde a década de 60, conheceu cinco grandes paradigmas. São eles: o banco de dados seqüencial, o hierárquico, o relacional, o orientado objeto e o objeto relacional (Chapnick, 1997, [10]). Cada um deles representa uma evolução e, em certo sentido, contém os conceitos do anterior. Podemos estudar o conceito de um banco de dados como sendo uma extensão do banco de dados imediatamente anterior a ele. O primeiro banco de dados pode ser assim concebido como a mera extensão de um fichário físico simples, como o que encontramos numa biblioteca. A descrição de cada um será apresentada a seguir.

1.2.1 Os diferentes sistemas de gerenciamento dos dados

No mercado de informática, dá-se o nome de “Gerenciador de Banco de Dados” ou (*DBMS* do inglês *Data Base Manager System*) ao conjunto de utilitários que auxiliam na tarefa do administrador e do usuário tratarem com a massa de dados armazenada. Nos primórdios dos computadores, os sistemas operacionais confundiam-se com os “DBMS” (Chapnick, 1997, [10]). Com o desenvolvimento da ciência da computação, os DBMS’s se separam dos sistemas operacionais caracterizando-se pela especialização em armazenar e recuperar informação estruturada, enquanto que os sistemas operacionais, tais como DOS, VMS, UNIX são dedicados a aplicações gerais.

A Figura 1.1 dá uma ilustração dos diferentes paradigmas dos bancos de dados que veremos a seguir.

1.2.2 Banco de Dados Seqüencial

Banco de dados seqüencial é aquele que mantém os dados em seqüência na ordem em que esses dados foram registrados.

A origem do banco de dados seqüencial confunde-se com a origem dos dispositivos de armazenagem de massa de dados. Assim, o primeiro banco de dados seqüencial surgiu no mesmo momento em que registrou-se dados em cartão perfurado. No entanto, a sistematização da ciência da informação surgiu quando a IBM lançou a linguagem **COBOL** de aplicação voltada para a área de contabilidade (Mammana, 1973, [43]). A primeira

aplicação da tecnologia da informação é o controle das transações bancárias e contabilidade, portanto o COBOL apresentou-se como a linguagem natural para esse fim (Chapnick, 1997, [10]).

O COBOL - linguagem aceita até os dias de hoje (a exemplo do FORTRAN na área científica) - muito evoluiu, cooperando, igualmente, para o aperfeiçoamento das técnicas de banco de dados. Apesar disso, o COBOL é uma linguagem pouco intuitiva exigindo uma "adaptação" do usuário aos seus conceitos e limitações sobretudo se a aplicação não está orientada a transações contábeis (Butzen & Forbes, 1997, [8]). Dessa maneira, o COBOL teve pouco ou quase nulo interesse para os cientistas das áreas exatas e biomédicas.

O aparecimento do "disco" magnético abriu caminho para os rudimentos dos bancos de dados seqüenciais de aplicação geral (Madnick & Donovan, 1981, [56]). Inicialmente, esses dispositivos eram usados apenas para administração do sistema e seu controle e, em última análise, podiam ser encarados como um grande e único arquivo. A única ferramenta de controle era o "*scan and find*", que pode ser entendida por "rastrear e encontrar" (Wetherbe, 1979, [70]). Esse processo está na raiz de todas as ferramentas e suas tecnologias associadas que vieram após esse período. Uma característica desse tipo de organização de dados é a necessidade de se começar a pesquisa sempre do primeiro dado do arquivo. Em outras palavras, não há qualquer ferramenta de otimização de pesquisa. A Figura 1.1 mostra (painel "a") um esquema do banco de dados seqüencial: um ponteiro percorre de cima para baixo cada "linha" da tabela até encontrar uma seqüência de dados que se ajusta à consulta (*query*).

O usuário típico, excluindo-se os analistas das grandes corporações, traziam seus dados em enormes quantidades de cartão perfurado. Sabendo-se como funcionava a técnica de leitura de cartões perfurados (ou fita perfurada ou magnética), vê-se facilmente, que a tecnologia "*scan and find*" era a única concebível. A procura iniciava-se a partir do primeiro cartão, ou "bloco", no caso de fita perfurada ou magnética, e prosseguia até se achar a seqüência de caracteres desejada. Havendo qualquer problema no procedimento era necessário iniciar o processo novamente.

Um problema clássico, originário dessa época, é o "ordenamento" (*sorting*) dos registros. Não é difícil conceber que uma das aplicações imediatas do banco de dados é o conhecimento do encadeamento segundo um certo critério. Para tal, tornou-se necessário o aumento da capacidade da memória do computador, pela necessidade de se ter área de "troca" (*swap*)¹

¹Área *swap*: Região da memória utilizada como tampão para troca de valores na memória de forma a dispô-los de maneira ordenada. Uma ilustração da área *swap* é o jogo conhecido por "Torres de Hanoi" usado para mostrar as capacidades da linguagem Pascal ou Algol. Nesse jogo o desafio é transferir uma

([5]), e sobretudo para diminuir a “carga” na assim chamada “paginação”². Desenvolveu-se, então, o conceito de “ponteiro”³ e concomitantemente o do “índice”⁴ ([14]). Um vetor, ou *array*, contém a ordem em que uma seqüência de registros deve ter, a ordem é determinada pela seqüência que o índice indica. O índice é um conceito fundamental na passagem para o próximo paradigma que é o banco de dados hierárquico, como veremos a seguir.

1.2.3 Banco de Dados Hierárquico

Banco de dados hierárquico é aquele que organiza os dados em “hierarquias” que são definidas previamente. Os dados são reunidos em “temas” e “sub-temas”, por exemplo, classificação de filmes: “ação”, “drama”, “comédia”, etc.

Com o desenvolvimento do UNIX nos laboratórios *Bell* (subsidiária da AT&T) (Arthur, 1990, [2]), aparece o conceito de árvore de diretórios, o que, em outras palavras, introduz no mercado o conceito de “hierarquia” na organização dos dados. Uma dada informação está armazenada em uma certa ramificação dentro de uma *árvore* lógica. Conhecendo-se a posição dessa informação é possível acessá-la imediatamente. A literatura na língua inglesa consagrou os termos “raiz” (*root*) e “folha” (*leaf*)⁵. A Figura 1.1 mostra, no painel “b)”, o esquema da arquitetura desse tipo de banco de dados: segue a estrutura de diretórios em um sistema de arquivos (*file system*, Madnick & Donovan, 1981, [56]). A pesquisa torna-se rápida na direção de cima para baixo (*top-down*, Chapnick, 1997, [10]).

A década de 70 é a época da consolidação das ciências computacionais. Computadores como o IBM/360M, Borroughs 9000, CDC se firmam como referências para centrais de computação (*mainframes*). Os mini computadores também se tornam populares, tais como o Nova-3, HP-1000 e PDP-11. Todos eles continham discos magnéticos como os que conhecemos hoje, porém, as técnicas de armazenagem, tanto do ponto de vista tecnológico quanto conceitual, nos parecem rudimentares. Na época o UNIX ainda era praticamente desconhecido do mercado. Estava em fase de desenvolvimento nos laboratórios do MIT (Arthur, 1990, [2]). Enquanto isso, cada fabricante tinha sua própria forma de armaze-

pilha de moedas dispostas em ordem crescente de valor para outra pilha mantendo essa mesma ordem. No desenvolvimento da solução é preciso utilizar uma pilha auxiliar (*swap*) para não se quebrar a ordem pretendida.

²Ato do computador colocar dados temporários em certo espaço da memória de massa, que àquela época não passava de uma unidade de fita magnética auxiliar.

³**Ponteiro:** Variável contendo o endereço de memória de uma certa grandeza.

⁴**Índice:** Matriz contendo os endereços da memória com os valores de uma grandeza.

⁵Na estrutura do antigo PC-DOS, raiz é designada por C:\ enquanto que uma folha seria, por exemplo, C:\TC\LIB. Nessa hierarquia TC se coloca como um ramo intermediário entre a raiz C:\ e a folha LIB.

nar os dados em disco, segundo suas próprias disponibilidades e conveniências. Antes do UNIX ganhar o mundo, ainda tínhamos o reinado do "VMS" que equipava os "VAX's" da DIGITAL Research, verdadeiro paradigma n^o "*modus operandi*" dos profissionais ligados às ciências computacionais ([14]).

Uma das características do banco de dados hierárquico é o posicionamento estático dos endereços das "folhas", isto é, dos membros do banco de dados, como vimos acima. Uma área do disco é especialmente reservada para guardar os endereços. Essa característica faz com que a estrutura do banco de dados contenha "páginas" de tamanho estaticamente definido, independente do conteúdo dos dados⁶.

Um exemplo emblemático do banco de dados hierárquico é o sistema "ISIS", distribuído gratuitamente pela UNESCO para administração de bibliotecas. Pela maneira com que foi concebido, de forma hierárquica, um banco de dados gerenciado pelo "ISIS" permitia uma consulta rápida e fácil por ordem de autor ou título. Se, no entanto, quiséssemos fazer uma pesquisa dos títulos que utilizam uma certa palavra, por exemplo, a consulta se tornava difícil e demorada. Era usado o conceito de "inversão", da seara do administrador, que criava uma tabela de índices para essa pesquisa ser possível. O procedimento de inversão era longo e oneroso. Além disso, a cada atualização do banco de dados, essa inversão deveria ser refeita. A linguagem utilizada para a construção de consultas desse tipo de estrutura de banco de dados é conhecida como QBE (*Query By Example*). Suas diretrizes baseiam-se em comandos do tipo "list" ou "browse" sobre colunas de uma tabela, seguidos de definições de processos de filtragem. Conexões com tabelas eventualmente relacionadas são estabelecidas em instruções em nível de ambiente⁷. A verificação de integridade referencial ou construção de índices é feita nas chamadas "inversões" ou "arquivos invertidos" que devem ser aplicadas freqüentemente (ver em [61] e Jones, 1987, [34]).

Ainda nos dias de hoje, os bancos de dados hierárquicos são utilizados, sobretudo em tarefas específicas, tais como sistemas de controle de autenticação de acesso em redes de computadores ([58]). Essa estrutura caracteriza-se por ser razoavelmente ágil, diante de certas tarefas, no entanto são inadequadas para outras. A necessidade de se procurar um sistema que satisfizesse a maioria das necessidades na administração dos bancos de dados fez aparecer um novo paradigma na ciência de armazenagem e recuperação da informação. Trata-se do banco de dados relacional, que veremos a seguir.

⁶Ver capítulo 6 de Madnick & Donovan (1981, [56]).

⁷Um banco de dados muito usado em PC's era o DBASE-III ([34]). Sua linguagem de consulta era conhecida como *Clipper*. Para conectar tabelas relacionadas no DBASE-III era necessário lançar a instrução "relation to" para que os comandos subseqüentes fossem interpretados como "relacionados".

1.2.4 Banco de Dados Relacional

Banco de dados relacional é aquele que permite relacionamentos entre tabelas⁸. O conceito geral de “relação” é definido por Codd (1990, [13]) como:

“... qualquer conexão, correspondência, ou associação, que pode ser concebida como naturalmente existindo entre coisas”.

Na prática, um relacionamento entre tabelas é mantido através de atributos comuns, os quais podem ser matematicamente comparados.

E.F. Codd, matemático da IBM em San Jose, California, apresentou um trabalho em 1970 ([12]) onde propõe um banco de dados baseado nos conceitos de “conjuntos relacionais”, a teoria do Banco de Dados Relacional (RDB). O desenvolvimento de todos os conceitos desse banco de dados representa uma aplicação da teoria dos conjuntos. Essa é a principal propriedade do trabalho de Codd. A Figura 1.1 - "c)" mostra o esquema de um banco de dados relacional e sua relação com a árvore binária, forma de indexação que permitiu um grande avanço nessa área. O banco de dados relacional aparece concomitante com o sistema de índices em árvore binária (Chapnick, 1997, [10]). O “ponteiro” sempre faz um teste entre duas opções, descendo até o ponto da seqüência desejada. Dessa forma podem-se construir tantas estruturas do tipo em (b) quanto se queira (Butzen & Forbes, 1997, [8]).

Na esteira dos trabalhos de Codd, Michael Stonebraker e seu grupo desenvolveram o “Ingres” com a linguagem “QUEL” (*Query Language*) enquanto que a IBM lançou as bases do “DB2” e a linguagem “SQL” (*Structure Query Language*)⁹, que acabou dominando o mercado (Butzen & Forbes, 1997, [8]).

Os trabalhos subsequentes de Codd (Frank, 1988, [21]) estendem o modelo relacional, estabelecendo os parâmetros para o desenvolvimento de DBMS's (*Data Base Manager System*) pela indústria. Finalmente a ANSI (*American National Standards Institut*) codificou, em 1987, um padrão para o SQL, e a ISO (*International Standards Organization*) lançou, em conjunto com a ANSI, em 1992, a extensão SQL2 (Butzen & Forbes, 1997, [8]).

Atualmente, praticamente todos os DBMS's são baseados nos fundamentos da RDB. Entre os mais famosos, encontram-se o *Oracle*, *MS-SQL Server*, *Sybase*, e *Informix*, no

⁸O conceito de “tabela” é generalizado para o conceito de “entidade”.

⁹O SQL é tido como linguagem de quarta geração a partir da proposta de padrão chamado CODASYL (*Conference On Data Systems Language*), mantido por um comitê do governo americano e adotado a partir de 1978 (Frank, 1988, [21]).

campo dos produtos comerciais. Entre os *sharewares*, encontramos o *PostgreSQL*, *MySQL*, e *mSQL*, sendo distribuídos na base Linux, esses DBMS são excelentes ferramentas de testes em modelos de pequenas bases de dados (Butzen & Forbes, 1997, [8]).

1.2.5 Banco de Dados Orientado Objeto

Um banco de dados orientado objeto é aquele que mantém dados complexos, e não somente tabelas, que podem ser recuperados através de diretrizes pré-definidas.

Para entender o Banco de Dados Orientado Objeto é preciso conhecer o papel dos BLOBs (*Binary Large Objects*, Jennings, 1997, [33]). A maioria dos DBMSs oferecem a oportunidade de se criar áreas de memória dedicadas a "objetos binários", isto é, seqüências de *bytes* cujo significado é revelado através de uma aplicação: imagens, sons, sinais criptografados, etc.

Para dar a um BLOB algum significado é preciso recuperar todo o conteúdo do objeto para interpretá-lo com um aplicativo fora do banco de dados. Essa operação representa o mesmo que trazer seqüencialmente toda uma tabela do banco de dados, para, em seguida, fazer a procura dos dados com algum aplicativo. Em última instância, essa operação é o mesmo que usar o banco de dados como uma espécie de sistema de arquivos (*file system*) sem se servir da capacidade de otimização do DBMS. Como consequência os fabricantes de DBMS passaram a desenvolver aplicativos para serem chamados no interior do banco de dados e não fora dele, para aproveitar suas propriedades de otimização¹⁰. A orientação objeto vem do fato que os módulos tratando dos BLOBs são construídos com base na programação objeto (ver em 1.3).

Com o advento das linguagens orientadas objeto, tais como o C++, MSVB (Microsoft Visual Basic), Borland Delphi, PowerBuilder, IBM SmallTalk, etc, lançou-se alguns produtos no mercado trazendo o conceito de "objetividade" a bancos de dados. No entanto, uma das grandes vantagens do banco de dados relacional foi desprezada nesse novo tipo de banco de dados: a propriedade relacional. De certa forma, retornou-se aos antigos bancos de dados seqüenciais caracterizados pelo baixo índice de normalização (Jennings, 1997, [33]).

Um problema comum entre as linguagens orientadas ao objeto, e por extensão, dos

¹⁰A forma de otimização varia. Alguns DBMS, como o MS SQL-server mantêm ferramentas especiais de desenvolvimento como o OLE (*Object Linking and Embedding*, [33]) em que é possível visualizar imagens em diversos formatos, por exemplo. Outros DBMSs permitem chamar aplicativos internos e outros externos, por exemplo, o Adobe Photoshop em BLOBs para serem executados quando se recupera alguma imagem de um BLOB.

bancos de dados derivados delas, é o dilema: facilidade de desenvolvimento - apoio de biblioteca (*library*). Enquanto linguagens de fácil transposição como o C++ exigem que o programador desenvolva suas próprias ferramentas para desenvolver seus aplicativos, outras, como o SmallTalk, fornecem um ferramental excessivamente grande em sua biblioteca, a ponto de criarem dificuldades para conhecermos todas as ferramentas, como são utilizadas e o que elas retornam.

Os bancos de dados orientados objeto não alcançaram o sucesso esperado. Isso devido à grande solução que se apresentou em seguida, que são os bancos de dados objeto - relacional, que veremos abaixo.

A Figura 1.1 - "d)" mostra uma ilustração do banco de dados orientado objeto: o "ponteiro" agora é capaz de "apontar" para estruturas complexas.

1.2.6 Banco de Dados Objeto-Relacional

Banco de dados objeto-relacional é o banco de dados que é capaz de manter relacionamento (nos termos definidos por Codd, 1990, [13]) entre objetos complexos.

Poder-se-ia dizer que o banco de dados objeto-relacional (BDOR) seria a "grande unificação" dos bancos de dados. Em síntese, os BDORs unem todas as capacidades dos bancos de dados relacionais com o que se conhece de mais avançado na programação objeto. Para resumir, poderíamos falar em relacionamento entre objetos complexos (Figura 1.1). Há de se estabelecer padrões para que esse relacionamento seja possível¹¹. A Informix S.A. saiu na frente através de seus *datablades* (Simmons, 1997, [54]). Os concorrentes também começaram a procurar suas soluções (McNabb, 1997, [44]). No atual estágio da ciência dos bancos de dados, estamos ainda nos passos preliminares e longe de uma padronização.

A idéia para o desenvolvimento das ferramentas para banco de dados objeto - relacional aparece na extensão das ferramentas dos "BLOB's" (*Binary Large Objects*) que virtualmente todos os fabricantes disponibilizaram aos seus clientes. Desde a sua concepção, os BLOB's mereceram tratamento distinto, sendo a eles dedicadas áreas exclusivas em "páginas", "chunks" e outras unidades da estrutura interna dos DBMS (ver 1.4.2 abaixo).

A Figura 1.1 - "e)" mostra uma ilustração de um relacionamento possível entre entidades em um banco de dados objeto - relacional. Objetos complexos podem ser indexados e relacionados ao modo dos bancos de dados relacionais. Trata-se de uma representação pictórica daquilo que é possível nesse tipo de banco de dados, pois esse relacionamento

¹¹Tabelas poderiam ser entendidas como objetos. A padronização da interpretação do texto permite o relacionamento entre elas.

depende do programador. Desde que os objetos são criados a nível de programação, o relacionamento entre entidades também é feito nesse nível ou no chamado "esquema conceitual" - deixando de ser em nível de "back end" também conhecido por "esquema interno" - (ver mais à frente em 1.4.2).

Alguns centros acadêmicos, como Berkeley, divulgaram seus DBMS's munidos de ferramentas para programação orientada objeto tais como o "PostgreSQL" ([59]). A Informix foi o primeiro fabricante comercial a desenvolver um ferramental poderoso no tratamento de objetos complexos, que constitui a base de seus produtos conhecidos como **Datablade** (ver Simmons, 1997, [54]). A Informix disponibilizou para o Observatório Nacional o **Web Datablade** dada a orientação voltada para a disponibilização via *Web* de nosso banco de dados.

A Figura 1.2 mostra um interessante quadro de evolução dos diferentes paradigmas em bancos de dados. A "complexidade" cresce da esquerda para a direita, acompanhando o tempo, e o poder de consulta de baixo para cima. Ela foi apresentada em um seminário promovido pela "SUN" para apresentação de parcerias para o desenvolvimento de ferramentas em Java. Essa é a ilustração de como a ciência dos bancos de dados evolui através de cinco paradigmas. As curvas indicam a evolução da aceitação pelo mercado no tempo. Note-se a efêmera evolução dos bancos de dados orientados ao objeto, enquanto que a expectativa se mantém crescente para os bancos de dados objecto - relacionais. A razão da curva de evolução dos BDOR's partir do início da década de 70 vem das primeiras pesquisas com os "BLOB's".

1.2.7 Qual é a vantagem dos DBMS's?

Uma outra pergunta decorre desta: vantagem sobre o quê? A comparação é com o "sistema de arquivos" (*file system*). A base do sistema de arquivos, depois do lançamento do UNIX (Arthur, 1990, [2]), é hierárquica (ver 1.2.3) e, portanto, otimizada na procura de "cima para baixo"¹². Quanto a essa propriedade não há muito o que um DBMS pode acrescentar. É preciso introduzir, portanto, as propriedades relacionais e a operação com os índices. Esse ponto cada fabricante tem desenvolvido segundo seus próprios modelos. Alguns DBMS, tais como o MySQL, MsqI, PostgreSQL, geram um novo diretório a cada banco de dados criado pelo usuário. Debaxo desse diretório esses DBMSs criam diferentes arquivos como

¹²Quem trabalha com UNIX, provavelmente já usou o comando *find* que promove a procura de arquivos com uma determinada propriedade. Esse comando faz a procura percorrendo, sempre, a árvore de diretórios "abaixo" do diretório em que se está, nunca acima.

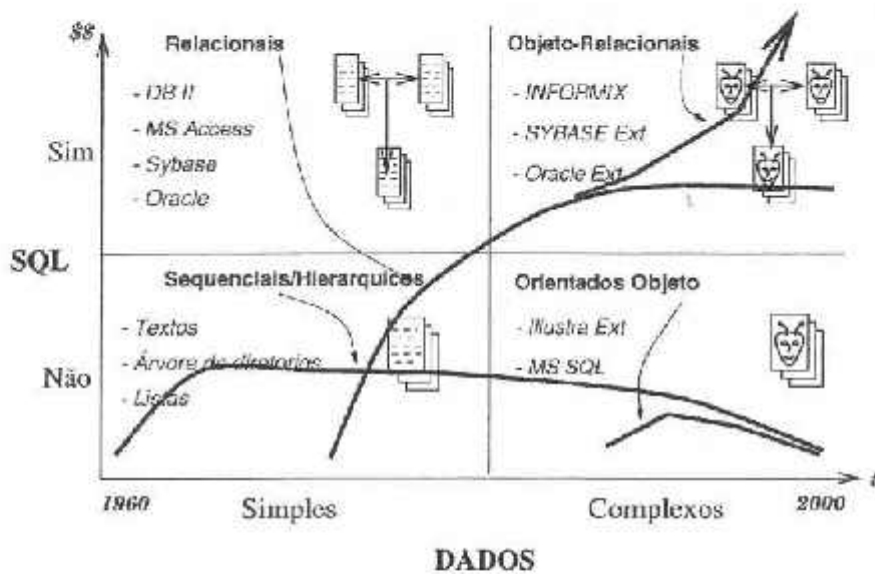


Figura 1.2: Quadro de evolução dos bancos de dados, destacando-se cinco paradigmas.

o de dados, de estrutura, de índices e outros auxiliares. Outros DBMSs, como o MS-SQLserver, Sybase, Informix, geram um arquivo a cada novo banco de dados e lá colocam todas as informações necessárias para a manutenção deste.

DBMS's especialmente concebidos para armazenar grande quantidade de informação, tais como o Oracle, Sybase, Informix, etc, oferecem a alternativa de poder reservar unidades de disco inteiras para a interação direta com o DBMS, dispensando o sistema de arquivos do sistema operacional instalado. Dessa forma, o uso do disco é otimizado para aplicações somente de banco de dados.

Portanto, a vantagem dos DBMSs é sobre os sistemas de arquivos no sentido que utilizam as capacidades do computador voltadas exclusivamente para aplicações em bancos de dados de forma otimizada.

1.3 O Conceito de “Objetividade”

1.3.1 Introdução

Tal qual a evolução dos bancos de dados, o conceito de programação também sofreu mudanças radicais, com o estágio mais avançado englobando os conceitos do estágio imediatamente anterior. Nos dias de hoje, o grande avanço em programação consiste na programação orientada objeto. Em último caso, podemos colocar em um compilador “orientado

objeto" uma programação rigorosamente seqüencial, se bem que esse expediente nos custaria algum trabalho (Corel & Horstman, 1997, [22]). Como nos bancos de dados, a programação seqüencial sob o paradigma da programação orientada objeto provoca uma sensação extenuada e de desperdício. Não há como escapar da tentação da programação estruturada e o conforto que ela proporciona. O mesmo pode-se dizer da programação objeto, que viabiliza as pesadas aplicações da computação gráfica, a cuja técnica está intimamente ligada.

Nessa seção daremos uma descrição resumida das diferentes técnicas de programação.

1.3.2 A Programação Seqüencial

A primeira, e mais simples, forma de programar foi a seqüencial. Essa técnica é aquela que decorre naturalmente da programação em *assembler*. Esta é a linguagem que mais se aproxima à linguagem de máquina¹³. Para facilitar a compreensão, vamos evitar falar do *assembler* propriamente dito e recordaremos a programação em uma calculadora HP-69. A linguagem de programação dessa calculadora é parecida com o *assembler* salvo que as funções matemáticas, interpretação de números a ponto flutuante, além da definição de rotinas para calcular as quatro operações, já estão incorporadas à linguagem (o que não é verdade no *assembler*). Para programarmos essa máquina escrevamos em seqüência as instruções para aplicar funções e operadores ao registrador X. (em *assembler* chamamos **acumulador**), com o auxílio de mais três registradores e alguns endereços de memória. A essa seqüência de operações damos o nome de **fluxo**. De acordo com as necessidades do cálculo o fluxo era interrompido por um teste que, sob condições de falso ou verdadeiro, fazia o fluxo "saltar" (*jump*) ou não para um determinado endereço. Havia, ainda, a possibilidade de desviarmos o fluxo para uma outra posição no programa que à frente fazia o programa retornar (*return*) para o passo seguinte (*branch*).

O exemplo mais emblemático da programação seqüencial, em linguagens de alto nível, é um programa típico FORTRAN, apesar dessa linguagem ter sofrido mutações tremendas, adequando-se heroicamente às transformações mais radicais. Podemos antever que num

¹³No curso "Organização de Computadores" do Prof. Cláudio Mammana no meu currículo de bacharelado do Instituto de Física da USP, tive a inesquecível experiência de programar em *Assembler* em um computador Honeywell, ainda sem o auxílio do "integrador" que traduzia os comandos em mnemônico para linguagem de máquina. Nós, os alunos, escrevamos o programa com auxílio dos mnemônicos, que eram nomes dos comandos do processador em língua humana. Em seguida, usávamos uma tabela de correspondência entre os mnemônicos e seus valores em números binários. Finalmente reuníamos (*assemblávamos*) o programa em números binários para seqüências de números octais, para, em seguida, compormos o programa em um pequeno teclado conectado ao computador.

futuro próximo apareça um “FORTRAN - OO”, dada a sua extraordinária versatilidade. No entanto, é ao FORTRAN-II que queremos nos referir. Aquele dotado tão somente de “do’s” e “if’s” aritméticos. Eis um fluxograma típico:

C Definicao de matrizes e vetores

C

```
DIMENSION X(100), Y(100), AM(100,100)
```

C

C Formatacao de entrada e saida

```
999 FORMAT(1X,F10.2,F10.2)
```

C

C Leitura dos dados de entrada

C

```
DO 99 I=1,100,1
```

```
    READ(5,999) X(I),Y(I)
```

```
99  CONTINUE
```

C

C Modulo de calculos

C

```
9   X(K) = 2.0 * Y(K)
```

```
    ...
```

```
    IF(X(K)-A) 9,8,7
```

```
8   X(K) = A
```

```
    GOTO 6
```

```
7   X(K) = Y(K) + A
```

C

C Segue o fluxo

C

```
6   CONTINUE
```

```
    CALL SUBRANCH(X,Y,Z)
```

```
    ...
```

C

C Imprime resultados

```
C
    DO 98 I=1,100,1
        WRITE(6,998) X(I)
98  CONTINUE
C
C Finaliza
C
    END

    SUBROUTINE SUBBRANCH(A,B,C)
    ...
    ...
    RETURN
    END
```

O que se nota nesse tipo de programação é a característica comum em programação de baixo nível¹⁴, tal como o *assembler*: os conceitos de “*jumps*” e “*branches*” que são os que caracterizam os “*if’s*”, “*do’s*” e “*call’s*” nessa linguagem. A programação de subrotinas e funções é feita na medida em que certas tarefas são repetidas, mudando-se apenas o valor de parâmetros. A construção de módulos, portanto, traduzida na elaboração de subrotinas e funções, tem o objetivo apenas de simplificar o trabalho.

Note, na listagem acima, que a programação seqüencial não necessariamente implica na orientação exclusiva em uma só direção. Os “*if’s*” e “*do’s*” aritméticos podem, e em muitos casos, devem, levar o fluxo de instruções para trás, gerando “*loop’s*” com saída após um certo número de execuções. No entanto, não há como fugir da comparação com as linguagens de mais baixo nível.

1.3.3 A Programação Estruturada

A programação para aplicativos de bancos de dados fez aparecer a figura do “registro” (*record*), meta-variáveis que contêm o bloco das informações sobre uma dada entidade. Um registro possui várias variáveis, cada uma de um determinado tipo que equivalem aos

¹⁴Termos como programação em **baixo nível** ou linguagem de **alto nível** não estão ligados a uma escala de valores e sim à proximidade com a linguagem de máquina. Quanto mais **baixo** é o nível, mais próximo somos obrigados a programar em linguagem binária.

atributos de uma tabela. O conceito de registro evoluiu para o de **estrutura** que contém não somente variáveis como também definições de tipos de dados, módulos de programas, blocos de um determinado fluxo. Na programação estruturada, sem prescindir dos fluxos, aparecem as definições das estruturas como blocos separados e independentes entre si (Mammana, 1973, [43]).

A primeira versão de linguagem de programação estruturada concretiza-se no "ALGOL" (Mammana, 1973, [43]), linguagem inspirada nas ciências dos algoritmos, até então uma área exclusiva da matemática. Apesar de apontar para uma nova era na ciência da informática, essa linguagem não se estabeleceu por problemas ligados à mercadologia. Contudo, em meados da década de 60, Kurt Würt, da Universidade de Zurique, divulgou o "Pascal" (Mammana, 1973, [43]), assentado no código do compilador de uma "pseudo máquina"¹⁵. Esse expediente permitiu a aplicação da linguagem em, virtualmente, qualquer processador. A tática de Kurt Würt permitiu que os fabricantes de software pudessem se dedicar exclusivamente à especificidade do processador no lugar de desenvolver um compilador para cada processador. Com isso a linguagem Pascal ganhou aplicação universal e passou a ser a linguagem preferida daqueles encarregados do desenvolvimento de aplicações comerciais (Mammana, 1973, [43]).

Pouco tempo depois do lançamento do Pascal, criou-se o **micro** processador. Um novo tipo de aplicação dos computadores se apresentou: controles de processos, portabilidade e a abertura do mercado dos computador para uso pessoal. As linguagens presentes nesse novo tipo de equipamento eram do tipo interpretada¹⁶ (os *BASICs*). Essa forma de operação tornava ainda mais lenta a execução de códigos no micro computador, que, por sua vez, eram lentos comparados aos processadores tradicionais. Por outro lado o código da pseudo máquina do Pascal era grande e os micro computadores não tinham, nem aproximadamente, condições de abrigar um código desse. No início da década de 80 um grupo na UCSD, Califórnia (Dennis, 1985, [15]), condensou o código da pseudo máquina permitindo a sua aplicação em um micro computador *Apple-IIe*. Isso permitiu finalmente a liberação dos

¹⁵Na época do lançamento do Pascal não existia o *chip* processador. Os processadores eram fabricados em placas e sua arquitetura determinava o conjunto de instruções. O resultado é que computadores de diferentes marcas e modelos possuem conjuntos totalmente diferentes de instruções. Fabricar um compilador era escrever um para cada processador. Cada processador tinha o seu compilador FORTRAN, COBOL, etc. Para que uma nova linguagem tivesse chances de permanecer no mercado existiam duas alternativas: escrever um compilador para cada processador no mercado ou "augerir" um código universal contendo a intersecção das instruções de todos os processadores. Essa última alternativa foi adotada por Würt no que ele denominou de "pseudo máquina", uma máquina universal contendo o maior conjunto de instruções possível para se aplicar a todos os processadores.

¹⁶Os programas não eram compilados e sim, cada instrução do programa era interpretada na seqüência do fluxo.

micro computadores da linguagem interpretada, abrindo campo para o desenvolvimento de aplicações em código compilado. Muito do crescimento do mercado de micro computadores e computadores pessoais se deve a essa aplicação.

Em 1978, por outro lado, Kernighan e Ritchie lançaram o livro "*The C Programming Language*" (Kenighan & Ritchie, 1978, [38]). A linguagem "C" servia de base para a construção do recém lançado sistema "UNIX". Outras linguagens foram lançadas, com menor popularidade, tais como o PL1¹⁷. Na esteira desse desenvolvimento, as tradicionais linguagens FORTRAN e COBOL ganham extensões permitindo, a elas também, o desenvolvimento através de estruturas. O FORTRAN 77 já se apresenta com instruções do tipo **STRUCTURE** e **RECORD**, que são correspondentes a seus homônimos no "C" e Pascal ([57]). A Figura 1.3 traz reproduzida a listagem de um exemplo extraído do Manual do Usuário do Turbo C ([4]).

Um programa em linguagem estruturada obedece, essencialmente, ao roteiro apresentado em "a)" na Figura 1.4. A definição de uma variável de estrutura complexa envolve definições de várias variáveis e tipos, além de funções que eventualmente tratarão das diferentes instâncias dessa variável. Esse processo refere-se a qualquer nível de programação, aplicando-se tanto ao programa "principal" quanto a subrotinas e funções. Uma outra característica é a introdução do conceito de variável "complexa", variável composta por várias dimensões não necessariamente definidas no espaço numérico. Esse tipo de construção é denominada "estrutura", ou variável **estruturada**, cujo exemplo pode ser visto na Figura 1.3. A definição de uma variável desse tipo é esquematicamente mostrada na Figura 1.4 "b)". Note-se que dentro de tal estrutura cabem definições de funções e procedimentos que eventualmente vão ser usados para a transformação e tratamento da variável ela mesma.

Entre as características que se distinguem na programação em linguagem estruturada encontram-se ([14]):

1. Divisão do programa em módulos segundo critérios conceituais e não mais para a simplificação na programação (ver penúltimo parágrafo de 1.3.2);
2. Definição das variáveis se faz segundo estruturas;
3. A passagem de parâmetros para funções deixa de ser por referência e passa a ser por valor;
4. "*Branches*" e "*Jumps*" passam a ser virtualmente proibidos, ou, na melhor das hipóteses, muito mal vistos. Tais instruções quebram a estruturação do programa. A

¹⁷PL1: Linguagem que a IBM apresentou para concorrer com o ALGOL.

```
#include <alloc.h>

typedef struct {
    char name[25];
    char class;
    short subclass;
    float decl, RA, dist;
} star;

main ()
{
    star *mystar;

    mystar = (star *) malloc(sizeof(star));
    strcpy(mystar->name, "Epsilon Eridani");
    mystar->class = 'K';
    mystar->subclass = 2;
    mystar->decl = -3.5167;
    mystar->RA = 9.633;
    mystar->dist = 0.303;

    /* Rest of function main */
}
```

Figura 1.3: Estrutura e ponteiro.

Exemplo em “C” combinando definição de estrutura e uso de estrutura e ponteiro no programa ([4]). A instrução “`star *mystar;`” é uma declaração definindo a variável como um ponteiro. A área da memória para essa variável é reservada na instrução “`mystar = malloc(...`” e as instruções do tipo “`mystar->[variável da estrutura]`” é a forma do “C” atribuir um valor à variável apontada por “`mystar`”.

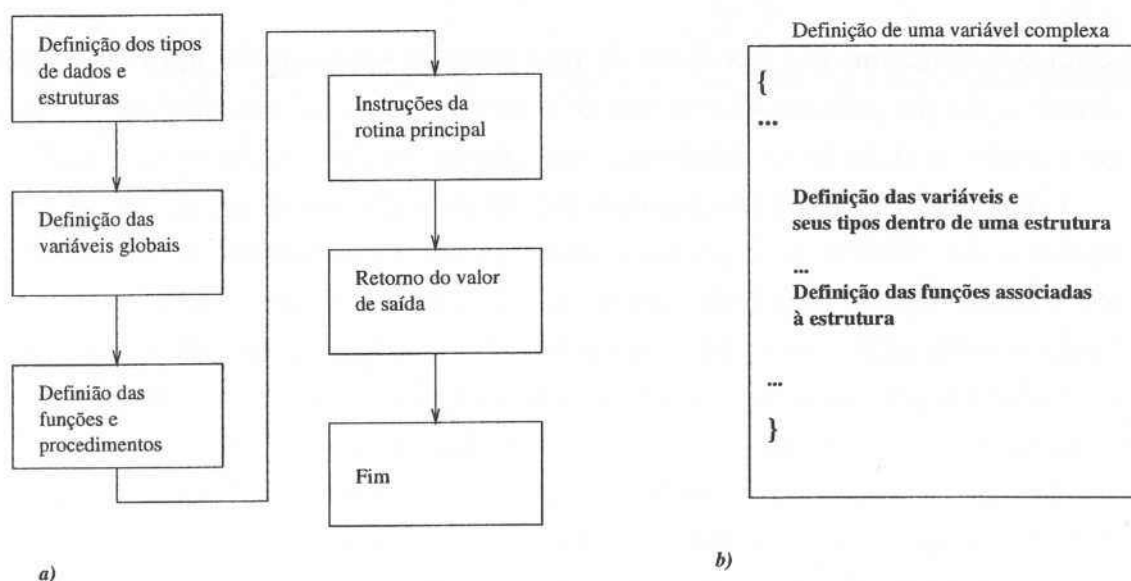


Figura 1.4: Programação estruturada.

a) Fluxograma de um programa em linguagem estruturada; b) definição de uma variável complexa.

conseqüência é que programas em linguagem estruturada contendo *branches* e *jumps* são passíveis de conterem erros difíceis de serem diagnosticados. Ao contrário, a orientação passa a ser a utilização de estruturas definidas por “if, then, else”, “while”, “do, until”, “case of”, etc, para o controle de fluxo;

5. “If’s” e “do’s” aritméticos deixam de existir;

6. Introdução da recursividade interna¹⁸.

¹⁸**Recursividade interna:** Essa é uma poderosa ferramenta da programação estruturada. Consiste em permitir que se “chame” a função dentro dela mesma. Para evitar o *loop* infinito, coloca-se um “escape” para permitir que a recursividade retome o caminho inverso. Como exemplo modificamos a função “poly” no caso do *Numerical Recipes* ([47]), no capítulo que trata do ajuste por mínimos quadrados em funções lineares:

```

FUNCTION POLY(P,X,I)
DIMENSION P(*)
IF (I .EQ. 0) THEN
  P(I) = 1.0
ELSE
  P(I) = POLY(P,X,I-1)
END IF
RETURN P(I)*X
END
  
```

Com o estabelecimento das bases da programação estruturada, aproximou-se definitivamente o ato da programação ao ato de escrever um algoritmo, afastando o programador da posição de engenheiro eletrônico e aproximando-o da posição do matemático ([14]).

Com esse avanço, os compiladores passaram a ser desenvolvidos segundo critérios de otimização. Unidades de pré-compilação tornaram-se capazes de identificar estruturas semelhantes nos códigos fonte, sendo possível, assim, fatores multiplicativos de condensação e aceleração. As estruturas permitiram construir todo código compilado na base de endereçamento indireto. O processo de compilação, portanto, passou a se dividir em duas fases: a geração do código objeto e a “fase de *link*”, que gera o código executável ([14]). Essa divisão, abriu caminho para uma “especialização” entre os fabricantes. De um lado os que desenvolvem o gerador do código objeto, que são os fabricantes dos hoje chamados compiladores: Turbo Pascal, C, Fortran, etc. Do outro lado a fase de *link*, via de regra, ficou a cargo do fabricante do sistema operacional, que é quem tem, teoricamente, o conhecimento da melhor forma de interagir com este ([14]).

A programação estruturada, permitiu, portanto, um avanço substancial na computação, a saber:

1. Aproximou a técnica de programação da teoria dos algoritmos;
2. Permitiu o desenvolvimento de compiladores “otimizantes”;
3. Abriu espaço para a construção de bibliotecas “compiladas”;
4. Abriu espaço para especialização de fabricantes de compiladores;
5. Permitiu uma homogeneização da geração do código executável que passou a ser feita pelo fabricante do sistema operacional.

1.3.4 A Programação Objeto (POO)

Abaixo apresentamos uma definição de **POO** traduzida a partir de Cornell & Horstmann, 1997 ([22]):

Em termos simples, a programação orientada objeto é uma técnica que enfoca a programação nos dados (=objetos) e nas suas interfaces. Para fazer uma analogia com a carpintaria, um carpinteiro “orientado objeto” concerne-se mais

à cadeira que ele está fazendo, e, secundariamente, com as ferramentas usadas para fazê-la. Um carpinteiro “não orientado objeto” pensaria primeiro em suas ferramentas...

Para se ter uma idéia do que isso representa, vejamos um exemplo: quando na programação estruturada pensávamos implantar um certo operador sobre uma variável, escrevíamos uma função ou subrotina que agia sobre essa variável, que era definida, por exemplo, entre os parâmetros dessa função. Exemplo para calcular o quadrado de uma variável “float x” escrevíamos uma função chamada, “squarepow(x)”, e no contexto dessa função multiplicávamos o parâmetro “x” por si mesmo. Na programação orientada objeto, definimos um objeto “X” que contém uma variável do tipo “float”, chamada “x”, e dentro do contexto do objeto “X”, escrevemos o método “squarepow()”, que muda a variável para o seu quadrado. Se, na programação estruturada, calculávamos o quadrado de 2 fazendo: squarepow(2), na programação “OO” fazemos: “2.squarepow()”. Na programação estruturada guardamos o resultado em uma nova variável, por exemplo, chamada “y”. Na programação OO, “instancializamos” o objeto “X” e guardamos a nova instância no objeto “y”. Junto com a POO vêm os conceitos de “construtores”: métodos que geram novas instâncias do objeto; “destruidores”: métodos que ‘apagam’ novas instâncias eventualmente criadas e “mutadores”: métodos que mudam características de instâncias de objetos.

Ao iniciar a década de 80, alguns dos fabricantes de compiladores já percebiam a necessidade de se avançar mais nos conceitos da programação. Essa necessidade é sensível observando-se módulos e exemplos de compiladores como o Turbo Pascal, que apesar de ser, eminentemente, uma linguagem estruturada, já possui elementos tais como “*constructor*” e “*destructor*”¹⁹, que serão chaves na programação orientada ao objeto ([5]). A década de 80 é marcada pela disponibilização no mercado dos primeiros compiladores orientados ao objeto, cujo exemplo mais significativo é o C++. Hoje são produtos bem conhecidos o Super Pascal da Borland, o VisualBasic da Microsoft e o Java da Sun. Em meados dos anos 80, a Apple lançou o McIntosh, com suas janelas e ícones. Essa nova apresentação da interface usuário é possível com os primeiros elementos da orientação ao objeto. É nessa mesma época que surge no MIT o projeto que veio a desembocar no padrão “*windows*”

¹⁹**Constructor**: Módulo declarado no início do código, sem, porém, contar com todas as instruções. Essas instruções serão introduzidas mais tarde ao sabor da aplicação desenvolvida. Mais tarde, na programação JAVA, esse conceito evoluiu para **interface** (referindo-se à declaração inicial) e **implementação** (referindo-se ao preenchimento do código com as instruções). O **Destructor** elimina as definições feitas na implementação, deixando o módulo livre para outras implementações. Esses conceitos não possuem paralelo no FORTRAN.

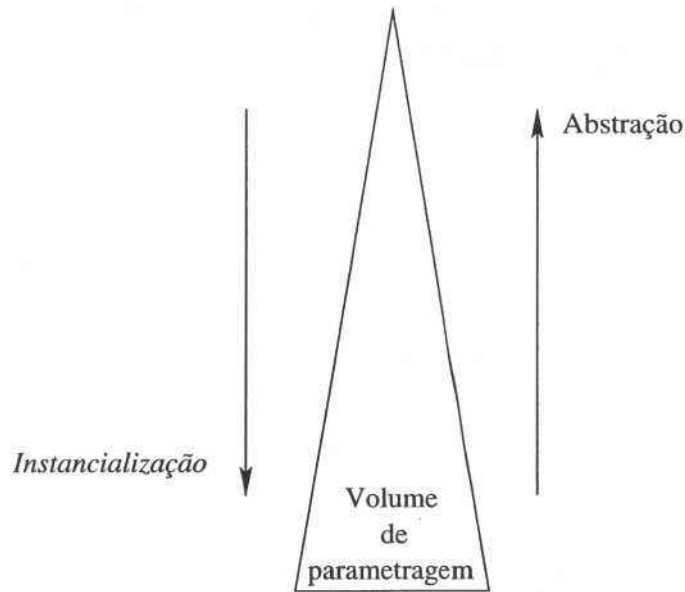


Figura 1.5: Direções possíveis que o analista segue ao desenvolver um programa.

para plataforma UNIX conhecido como X11. Totalmente desenvolvida em C++ essa plataforma fixou a linguagem orientada objeto. Hoje, todas as implementações, sob o UNIX, nessa linguagem, têm por base os conceitos desenvolvidos no projeto “X”²⁰.

Com a orientação ao objeto, a programação compartimentalizou-se de vez em módulos virtualmente independentes. A idéia é a programação de “objetos” que *per se* são independentes (Campione, 1998, [9]). A integração se dá na chamada “instancialização”, cuja característica é partir do abstrato passando mais e mais ao concreto. Há, portanto, uma hierarquia no *modus operandum* da programação. O analista é obrigado a elaborar um modelo do universo a ser desenvolvido, estabelecendo, em seguida, seus elementos mais abstratos, para, na medida em que vai colocando o seu modelo em prática, ir fixando os parâmetros específicos ao modelo. Por outro lado, ele pode partir de objetos concretos e, através da identificação de propriedades comuns, gerar abstrações que incorporem uma parte ou a totalidade desses objetos.

A Figura 1.5 mostra as direções possíveis da programação objeto. Se ele segue a direção que parte da maior abstração para uma maior instancialização, diz-se que ele segue a orientação “*top-down*”. Se, ao contrário, ele seguir o caminho da máxima abstração partindo do máximo concreto, diz-se que ele segue a orientação “*bottom-up*”. Dependendo

²⁰Os sistemas baseados em UNIX da SUN, HP, Digital e IBM, além de sistemas Linux e FreeBSD sustentam seus sistemas gráficos no X (ver documentação, guias e manuais do usuário *on-line* dessas plataformas).

da aplicação, e segundo suas próprias inclinações, o programador segue a orientação dita "top-down" ou "bottom-up". Muitas vezes, o programador é obrigado, a despeito de suas preferências, seguir uma ou outra orientação. Muito similar é a sistematização corrente entre aqueles que discorrem sobre a teoria do conhecimento: existe o raciocínio dedutivo (orientação *top-down*) e o indutivo (orientação *bottom-up*). É enganoso acreditar que existe uma superioridade de uma das duas orientações. Ambas possuem graus peculiares de dificuldade (McNabb, 1997, [44]).

Todos os compiladores orientados ao objeto são providos de um "objeto primordial", aquele a partir do qual todos os objetos derivam (Campione, 1998, [9]). Na verdade, esse "objeto primordial" vem a ser o real **escopo** do compilador, pois, em suma, todos os compiladores devem gerar códigos binários para serem executados pela máquina. Todo objeto, de uma certa maneira, é considerado uma instância de um objeto mais abstrato e este, instância de outro, até se alcançar o objeto primordial que tem relação apenas com o compilador. Ao gerar o código fonte, o programador deve (Cornell & Horstmann, 1997, [22]):

1. Declarar uma nova instância do objeto;
2. Definir as variáveis e tipos de variáveis daquela nova instância;
3. Declarar, se for o caso, quais as variáveis *públicas* e *privadas*²¹ do objeto;
4. Recuperar os parâmetros da nova instância;
5. Implementar os métodos e interfaces pertinentes ao objeto.

Portanto, podemos dizer que implementar um método seria o mesmo que criar uma *function* no "C", ou uma *procedure* no Pascal. Seria, a grosso modo, o mesmo que gerar uma "SUBROUTINE" no FORTRAN.

Os produtos da **POO** estão em virtualmente todas as aplicações modernas, tais como a série MS-Windows e X11. Assim, essa técnica de programação cumpre um de seus principais objetivos: permitir que o usuário execute tarefas complexas sem ter conhecimento das técnicas de programação. Uma operação simples, muito usada em sistemas de

²¹**Variável pública:** equivale à "aproximadamente" instrução COMMON no FORTRAN. Também à variável "global" no C ou Pascal. Na programação objeto o conceito vai um pouco além, pois o acesso a uma variável é sempre possível, dentro de um objeto. A diferença é se ela é disponível (**pública**) em qualquer ambiente ou não (**privada**).

“janelas”, conhecida por “clique e arrastar” é uma operação típica realizada através de técnicas de **POO**. Mesmo as aplicações onde acreditamos “interagir” com a máquina, como programas em FORTRAN, um editor de texto, uma ferramenta de desenho até um leitor de formato especial como o “Adobe Acrobat”, são desenvolvidos em linguagem objeto. Isso porque, com a definitiva compartimentalização da programação, pode-se desenvolver bibliotecas específicas e disponibilizá-las a todos que possam, assim, desenvolver outras bibliotecas que, por sua vez, estarão disponíveis. Essa postura permite gerar a chamada “cultura de empresa” ou de laboratório, com evidente economia e simplificação (Macedo, 1997, [42]). Quando o departamento de desenvolvimento gera o objeto “cilindro”, aos outros departamentos basta instanciarizá-lo para gerar “rolamentos”, “rodas”, “volantes”, etc, sem ser necessário entrar em detalhes da implementação do método que gera o cilindro. Só é preciso, então atentar para a implementação do método que gera o objeto rolamento, o objeto roda, o objeto volante, e todos aqueles que podem ser casos concretos do cilindro. Virtualmente todos os aplicativos são, hoje, desenvolvidos em alguma linguagem orientada ao objeto.

1.3.5 A Extensão da “SQL” à Objetividade

A evolução da **SQL** foi no sentido de transformar o sistema de consultas de forma a que este saiba tratar com **objetos**. Neste particular, entende-se por objeto todo e qualquer conjunto de dados que ganha significação no momento que aplicamos um método a ele de tal maneira que os resultados sejam compreensíveis aos humanos. Um objeto pode ser uma imagem, uma sinfonia, valores de brilhos de estrelas, etc (Cornell & Horstmann, 1997, [22]). Um objeto aqui é entendido como uma evolução do conceito de entidade²².

Acabamos de acompanhar a evolução nos conceitos de programação e suas linguagens, desdobrando, igualmente, na programação orientada ao **objeto**, que nesse contexto é entendido como uma entidade²³ abstrata que se torna concreta na medida em que se detalha a aplicação. A esse processo chamamos instanciarização.

Os diferentes valores que os atributos de uma entidade podem assumir, são chamados “instâncias”.

Como se vê, as tecnologias de programação e banco de dados, seguindo por caminhos diferentes, vão se encontrar nos mesmos conceitos de tal forma a se tornarem indistinguíveis.

Como veremos concretamente em 1.4.2.12, uma consulta em “SQL” pode ser detalhada

²²Ou tabela, no jargão da engenharia.

²³Entidade, aqui, deve ser entendida como uma grandeza ou um conjunto de grandezas matemáticas.

na seguinte gramática (Bowman *et al*, 1998, [6]):

```

verbo (commando)
atributos (passivo do verbo)
verbo (se for o caso) sobre a(s) entidade(s)
condições e junções entre as entidades

```

Essa gramática produz relações que são o resultado das junções entre as instâncias relacionadas. Uma consulta a um banco de dados objeto - relacional pode ser descrita da seguinte forma (Cornell & Horstmann, 1997, [22]):

```

instancialização do objeto
parâmetros da instância
métodos: construtores; mutadores; destrutores
métodos específicos das instâncias

```

Em ambos os casos o resultado final serve tanto como "apoio à decisão"²⁴ quanto como entrada para novas instancializações. O segundo esquema pode ser entendido como uma generalização do primeiro. Também vemos que os comandos, tanto no esquema da "SQL", quanto na linguagem objeto, seguem claramente uma orientação da linguagem natural, isto é, as instruções são passadas como se estivéssemos "interagindo" com a máquina. Geralmente a instrução é dada com um verbo conjugado no imperativo. De maneira geral, esse verbo tem um significado equivalente no "mundo real". Por exemplo, quando passamos a instrução *select* (selecione) em "SQL", estamos passando para a máquina uma instrução que é equivalente a expedirmos uma ordem para que se procure no fichário uma palavra com as características que vamos definir a seguir, nas condições e junções ou nos métodos específicos (Bowman *et al*, 1998, [6]).

Quando definimos um método, estamos construindo um conjunto de instruções que são unificadas em uma dada "ordem". É recomendável, portanto, que esse método seja nomeado de tal forma que faça sentido ao equivalente do objeto no mundo "real", pois essa tem sido a orientação dos analistas que desenvolvem produtos para gerenciamento de banco de dados relacional e objeto - relacional²⁵.

²⁴Sistemas de apoio à decisão são softwares que interagem com os bancos de dados e tratam os dados de forma a apresentar ao usuário informação num formato "compreensível" a ele. Em outras palavras, o que o usuário obtém quando consulta o banco de dados é um relatório na linguagem e apresentação que ele está acostumado. Algumas vezes esse tipo de serviço é chamado "instância de aplicativos". É aqui que se programa a nível de gráficos, diagramas de pizzas, tabelas com termos e números enfatizados, etc ([70]).

²⁵Por exemplo, é melhor chamarmos um objeto que define uma bola como bola e não, digamos,

Item	Matemático	Prático
1	Entidade	Tabela
2	Relação	Tabela
3	Atributo	Coluna
4	Instância	Linha
5	Tupla	Linha na tabela de resultados
6	Domínio	Tipo do dado (<i>data type</i>)
7	Cardinalidade	Número de linhas

Tabela 1.1: Termos matemáticos e usuais.

Correspondência entre os termos matemáticos e os usuais, isto é, usados entre os engenheiros e analistas.

1.4 Fundamentos do Banco de Dados Relacional

1.4.1 A terminologia matemática e seu correspondente nos “DBMS”s

Muitos conceitos possuem diferentes nomes dependendo do contexto. Essa dicotomia pode confundir aqueles que se iniciam na ciência dos bancos de dados. A tabela 1.1 mostra a correspondência entre os termos matemáticos e aqueles usados pelos analistas que desenvolvem os gerenciadores de bancos de dados (*DBMS*), ou o chamado "contexto prático" (Butzen & Forbes, 1997, [8]). A adoção de uma ou de outra terminologia não implica em perda de rigor. Embora com risco de comprometimento da generalidade, opta-se, muitas vezes pelo contexto prático dado o seu caráter intuitivo que torna mais fácil explicar e treinar os usuários. Nessa tese os termos são usados indistintamente, embora priorizando os termos matemáticos no contexto matemático, opta-se pelos termos práticos em casos de exemplos e aplicação.

1.4.2 Definições

Usa-se correntemente, no meio dos profissionais da área, duas expressões cujas diferenças são pouco notadas: o **Banco de Dados** e a **Base de Dados**²⁶. Ambas as expressões são adotadas a partir da terminologia inglesa. Historicamente, o termo **Base de Dados**,

e162004, que teria relação com uma classificação do próprio programador, pois esse nome só faz sentido para o programador, podendo perder sentido mesmo para ele, depois de um certo tempo.

²⁶**Base de dados**: Conjunto de arquivos, organizados e administrados de maneira flexível, tal que os arquivos dessa base de dados podem ser facilmente adaptados a tarefas novas e não previstas inicialmente.

mesmo em inglês, referia-se a qualquer meio físico de armazenamento de dados, enquanto que o termo **Banco de Dados** referia-se ao problema de armazenamento de informação organizada ou coerente (McNabb, 1997, [44]). Com o passar do tempo, o termo **Banco de Dados** caiu em desuso na língua inglesa. Por outro lado o termo **Base de Dados** foi sendo aplicado aos conceitos de “Sistema de Arquivos” (*file system*), árvore de diretórios e arquivos, etc. Nos meios profissionais no Brasil, preferiu-se manter o termo **Banco de Dados**, consagrado como sinônimo da alternativa **Base de Dados** (McNabb, 1997, [44]).

Algumas das definições dos conceitos aqui listados foram sistematizados por Frank, 1988 ([21]) em seu Capítulo 1. Outras foram colhidas de manuais da Informix ([31], [29] e [30]).

O Instituto Nacional Americano de Normas (*ANSI*) estabeleceu três esquemas com que uma base de dados²⁷ pode ser descrita (Frank 1988, [21]):

1. O **Esquema Conceitual** (ou visão conceitual) é a descrição, independente da máquina ou do *software*, da base de dados como um todo. Esse esquema que também é conhecido como a base de dados lógica, é o objeto de interesse dessa tese no que concerne a modelagem de uma base de dados de aplicação na astronomia;
2. O **Esquema Interno** (ou visão interna) é a descrição física da base de dados, como é organizada na máquina, como é feita a organização no disco e seu acesso. Nessa instância são debatidas questões da estrutura de “*chunks*” (geralmente unidade física de disco) e “páginas” (geralmente partições físicas ou virtuais);
3. O **Esquema Externo** (ou visão do usuário) é a orientação da descrição - de maneira a que o usuário compreenda - de como os dados são acessados. Não há obrigatoriedade de se descrever a organização da base de dados. E como existem muitos pontos de vista do que se quer fazer com os dados, existem muitas formas de descrever o esquema externo de uma base de dados.

1.4.2.1 Elementos do Esquema Conceitual

É comum, nos textos, utilizar-se um exemplo ilustrativo no intuito de tornar compreensível os conceitos desenvolvidos no esquema conceitual. Adaptando a teoria à realidade acadêmica, fazemos um paralelo dos conceitos com a estrutura de um catálogo de objetos.

²⁷A despeito de, nessa tese, se dar nítida preferência ao termo **Banco de Dados**, no esmero de se manter fiel ao texto citado, utiliza-se o termo **Base de Dados** que, como foi exposto, deve ser entendido como sinônimo do primeiro.

Matematicamente, uma base de dados é tida como contendo um conjunto de definições descrevendo os conceitos ali tratados. São elas:

Entidade: é um item, uma unidade ou um conceito no mundo “real” cuja informação se quer registrar. Normalmente ela se caracteriza por ter:

- um nome único;
- um campo contendo um identificador único para cada entidade individual. Esse campo geralmente é chamado “**chave primária**” ou “**chave identificadora**”;
- uma descrição dos atributos ou campos do tipo de entidade;
- indicação do número de ocorrências de um tipo de entidade na base de dados. Esse número é chamado “**número cardinal**”;

Atributo: é um aspecto da entidade que se quer registrar;

Instância: uma entidade se apresenta em diferentes possibilidades, cada uma com um conjunto de atributos que a caracteriza. Cada uma dessas possibilidades, é uma “instância” dessa entidade;

Tipo de Entidade: é a classificação de todas as entidades descrevendo o mesmo tipo de item ou conceito do universo sendo modelado. Ela determina que tipo de registro vamos inserir no banco de dados.

Os tipos de entidades que um banco de dados na astronomia pode ter são mostrados na Figura 1.6.

Um atributo de uma entidade pode ser:

Mandatário ou opcional: dependendo se esse atributo deve ter, ou não, pelo menos um valor. Atributos mandatários são marcados com a menção “NN” (*not null*);

Simple ou multivalor: quando um dado atributo pode assumir um ou vários valores para uma entidade. Atributos simples são marcados com a menção “U”;

Agregado ou escalar: um atributo pode ser composto de apenas uma qualidade ou ser o resultado da combinação de vários escalares. Muitas vezes chama-se um atributo agregado de “**campo combinado**” ou “**campo de grupo**”.

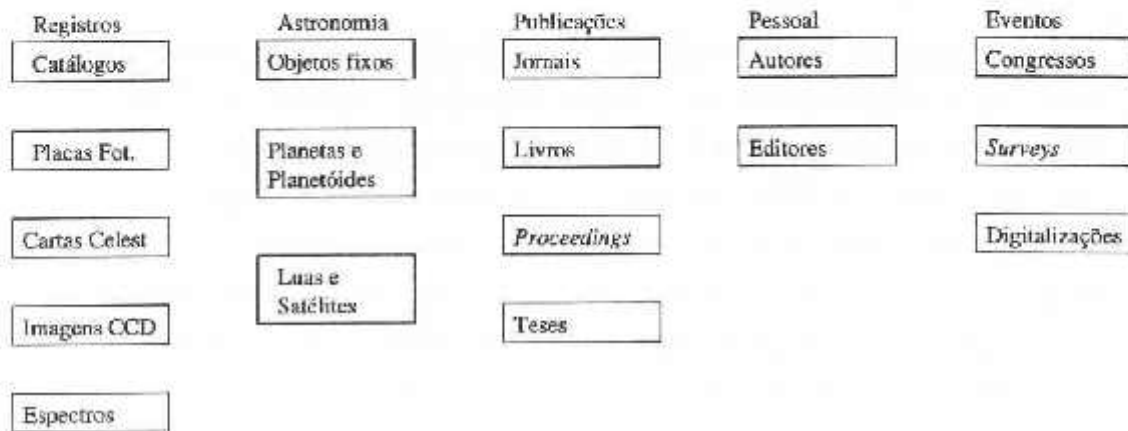


Figura 1.6: Tipos de entidades em um banco de dados na astronomia.

Em um catálogo astronômico (Figura 1.6), *entidades* podem ser **Registros**, **Astronomia** (significando objetos astronômicos), **Publicações**. Cada uma delas é caracterizada por certos *atributos*: Autores: pessoa física; Objetos fixos: descrição; Congressos: datas, locais, decisões. Quanto à *instância*, seriam as listas dessas entidades, isto é, a tipificação concreta das entidades, por exemplo: Autores: Nome, como João, Maria; endereço, *e-mail*, etc.

Na construção do banco de dados, o analista define previamente o tipo das entidades, e se estas manterão relacionamento mandatário ou opcional com outras e qual é a cardinalidade que será mantida. Essa fase da construção da base de dados é conhecida pelo nome de **normalização da base de dados**. O significado desse termo será explicitado adiante em 1.4.2.8.

1.4.2.2 Relacionamento

O relacionamento é a conexão entre duas entidades. Quando duas entidades possuem um ou mais atributos comuns diz-se que estas entidades mantêm um relacionamento. O produto de um relacionamento chama-se **relação**.

O termo **relação** normalmente é associado à “tabela”, mas Codd, 1970, [12] faz questão de distinguir um do outro. Segundo ele, a “tabela” seria uma forma particular de “relação”, sendo a esta última reservado um *status* de entidade matemática, guardando toda a qualidade de abstração. De qualquer forma **relação** pode ser vista como um conjunto de entidades similares.

Uma entidade pode se relacionar uma, duas ou muitas vezes com uma outra entidade, ou mesmo nenhuma vez. Essa multiplicidade de relacionamentos é chamada “**fator de ramificação**” ou “**cardinalidade**”. No que se refere à cardinalidade existem três tipos de

relacionamento: a) o relacionamento 1 para 1, quando a instância de uma entidade possui relacionamento com apenas uma ocorrência em outra entidade; b) o relacionamento 1 para muitos, quando uma instância de uma entidade pode possuir relacionamento com várias outras instâncias de outra entidade, e c) o relacionamento muitos para muitos, quando há a característica deste tipo nos dois sentidos. Esquemáticamente diz-se que para: a) $1 \leftrightarrow 1$ ou $1 : 1$, b) $1 \rightarrow n$ ou $1 : n$ e c) $n \leftrightarrow m$ ou $n : m$. Nos demais relacionamentos, há uma combinação de um ou mais desses esquemas. Do lado “1” do relacionamento, diz-se que essas entidades são “proprietárias”, e do lado “n”, diz-se que são “membros” (Bowman *et al*, 1998, [6]).

Por exemplo, o relacionamento $1 : n$ se dá entre as tabelas “Planetas” e “Luas e satélites”, descrevendo os catálogos em que constam esses objetos. Para cada planeta podem existir vários itens na tabela “Luas e satélites”, pois podem existir diversas luas para um só planeta. Já o relacionamento $1 : 1$ se dá entre as tabelas “Autores” e “Editores”. Um autor pode ser editor e pode possuir apenas uma ocorrência na tabela de editores e vice-versa. Finalmente uma relação $n : m$ pode ser mantida entre as tabelas **Catálogos e Objetos fixos**, pois um catálogo possui descrição para vários objetos e um objeto pode constar de diversos catálogos.

Além do aspecto “cardinal” dos relacionamentos, podemos encontrar os seguintes tipos:

1. O relacionamento seqüencial, quando as entidades se relacionam em ordem segundo um certo critério;
2. O relacionamento de equivalência, quando uma certa entidade pode ser organizada segundo uma propriedade comum;
3. O relacionamento em árvore, é associado diretamente ao processo de procura. A procura se dá numa certa direção, por exemplo, a partir do primeiro item, do último, do item intermediário, etc. Vejamos o exemplo da procura de um arquivo numa dada árvore de diretórios. Na direção de procura a partir do ponto mais alto na hierarquia de diretórios, o fator de ramificação máximo é 1. A entidade final é chamada “raiz”. Quando a procura é na direção oposta, o fator de ramificação máximo é n . As entidades finais da procura são chamadas “folhas”. Esse tipo de relacionamento se aplica especialmente em situações do tipo “árvore genealógica”;
4. O relacionamento em rede, que é um caso em que a ramificação múltipla se dá em todos os sentidos de procura.

<i>Relacionanemto</i>	<i>Diagrama</i>
1 : n	
n : m	
Hierárquico ou Árvore	
Rede	
3ª ordem	
1 : 1	
Seqüencial ou Equivalência	

Tabela 1.2: Tabela de Modelo de Relacionamento entre Entidades

1.4.2.3 Modelagem dos Dados

Dentre as ferramentas existentes para construir o modelo dos dados destacam-se três (Frank, 1988, [21]):

1. Diagrama *E-R*: diagrama “entidade - relacionamento”;
2. Diagrama de Bachman;
3. Modelo Relacional.

Adota-se, nesse trabalho, a ferramenta 1 - o diagrama *E-R* - pois ela se presta bem a todos os tipos de banco de dados, e especialmente ao banco de dados relacional. Esse diagrama é composto de figuras que representam os diversos elementos descritos aqui. Esse tipo de diagrama admite, também, várias representações. Na Tabela 1.2 apresentamos a representação adotada nesse trabalho. A tabela contém um diagrama denominado de 3ª ordem que não será discutido neste trabalho. Este diagrama descreve o que se chama *relacionamento do distribuidor*. Ajusta-se bem a redes e merece um tratamento específico a esse fim. Uma discussão mais detalhada desse tópico poderá ser apresentada quando discutirmos as etapas de evolução do SKICCOSMO (ver em 6.5.4).

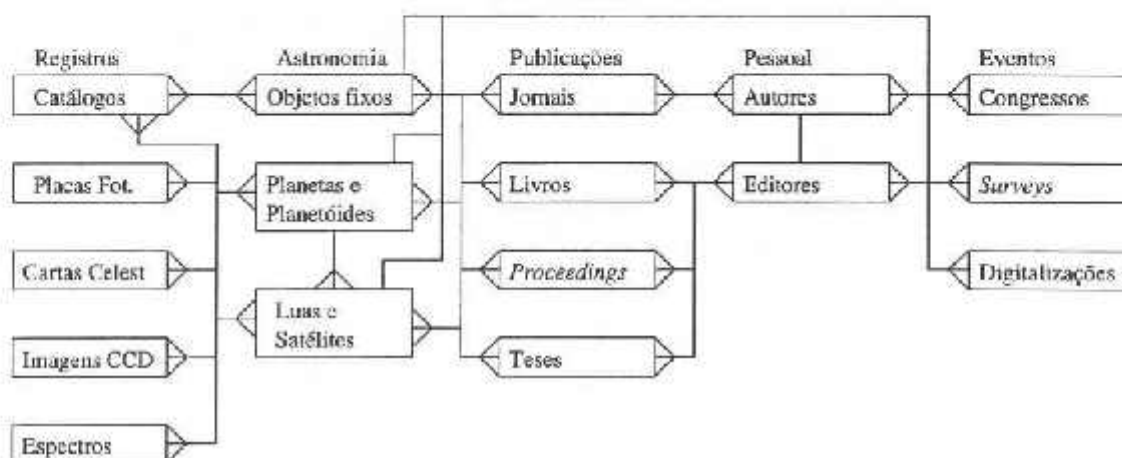


Figura 1.7: Exemplo de modelo de Relacionamento de Entidades (*ER-Model*)

Com respeito ao nosso exemplo, um modelo de nosso banco de dados em astronomia à luz do diagrama *E-R* pode ser visto na Figura 1.7.

1.4.2.4 Detalhamento das Entidades

É a fase em que o analista decompõe uma entidade em seus atributos. Conhecendo o tipo de entidade que está tratando, o analista sabe quais são os atributos relevantes para definir completamente uma entidade, de forma a garantir o que se chama “*integridade de domínio do usuário*”, que discutiremos em 1.4.2.11. A Figura 1.8 mostra uma exemplo dos atributos da entidade “Catálogos” no nosso modelo de banco de dados.

1.4.2.5 Diagrama da Visão do Usuário

Anteriormente foi citado o esquema externo, que é um sinônimo para o diagrama da visão do usuário. Do ponto de vista do usuário, não é necessário saber os detalhes de todas as entidades e seus atributos e formas de relacionamento entre elas. Dependendo do usuário²⁸, apenas parte das informações são disponibilizadas, e, em certos casos, não pode haver permissão para mudar ou inserir dados de algumas ou todas as entidades. Exemplo disso é um operador poder corrigir informações sobre o catálogo FK5, mas é impedido de fazê-lo para outros catálogos. Assim, o administrador do banco de dados cria uma *vista*²⁹, de

²⁸Note que por usuário entende-se, também, o operador que alimenta o banco de dados com informações. Por exemplo, o funcionário de uma revista que atualiza os dados da tabela “Jornais” é entendido como um usuário.

²⁹Espécie de sub-conjunto do banco de dados. Para o usuário, tudo se passa como se o banco de dados fosse constituído apenas pelas tabelas e atributos que ele pode “ver”.

Catálogos
Num Catálogo
Ascensão Reta
Mov.Prop.AR
Declinação
Mov.Prop.Dec
Tipo Espectral
Mag. V
Mag. B
Mag. U
Mag. R
Mag. I

Figura 1.8: Atributos relevantes à entidade **Catálogo**.

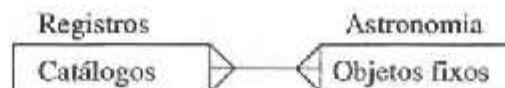


Figura 1.9: Exemplo de “vista” para a astrometria.

Apenas uma parte do banco de dados é exposta ao usuário.

tal forma que o usuário possa “ver” apenas algumas tabelas desse banco de dados e possa, eventualmente, fazer modificações em parte dos atributos das tabelas “visíveis” (Figura 1.9).

1.4.2.6 A Chave Primária e a Chave Estrangeira

Uma tabela ou uma relação deve possuir a chamada “**chave primária**”, que é uma identificação, seja numérica, seja por nome ou sigla, para uma instância de uma entidade. O valor dessa identificação é único em todo o domínio da entidade e, em princípio, identifica completamente a entidade. Uma chave primária pode conter mais de um atributo, contudo essa característica é mais presente nas entidades do tipo “evento”.

Dá-se o nome de “**chave estrangeira**” ao atributo que é derivado de uma chave primária presente em outra relação ou tabela. Geralmente é por meio dela que é feita a “**junção**” entre as duas relações, como veremos mais tarde. É comum encontrar-se, sobretudo nas

entidades do tipo “evento”, chaves estrangeiras fazendo parte da chave primária.

Segundo os preceitos estabelecidos nos trabalhos de E. Codd, as chaves primárias e estrangeiras devem ser usadas para proteger o banco de dados de ações que comprometam a integridade referencial dos dados.

1.4.2.7 Ponto de Partida para a Construção do Modelo

Pode-se partir de duas situações para a construção do modelo dos dados. A primeira e - se quisermos ver a questão do ponto de vista acadêmico - a mais elegante, é quando se elabora a base de dados de forma abstrata, para então construir as diferentes entidades de forma a torná-las mais e mais concretas. Essa é a situação que costuma-se chamar *top-down*, ou a postura da cadeia dedutiva. Concebemos as diferentes entidades em sua forma ideal, pois o banco de dados será munido de dados a partir do nada. Não há informação prévia a ser recuperada, e se há, pode ser transformada e adaptada às condições que o analista concebeu previamente. Se Hiparcos fosse um analista de banco de dados, ao construir o primeiro catálogo da astronomia, estaria construindo esse catálogo na situação *top-down*. Ele usou o conceito de magnitude, para classificar as estrelas pelo brilho, usou as posições das estrelas em coordenadas celestes, etc. Hiparcos, enfim, ao construir seu catálogo astronômico estava criando um banco de dados normalizado.

A segunda situação, conhecida como *bottom-up* ou da cadeia indutiva, é aquela que encontramos com mais frequência nos dias de hoje, pois não há, praticamente, qualquer situação no mundo real que não tenha sido modelada e aplicada a concepções de banco de dados anteriores. Essa é a situação em que o analista se vê diante de informações guardadas em planilhas “Lotus 1-2-3” ou “MS-Excel”, em tabelas “DBASE”, em arquivos “COBOL” e precisa unificar toda esta informação em um banco de dados coerente e normalizado (McNabb, 1997, [44]).

Nas circunstâncias desse trabalho, a situação em que nos encontramos aproxima-se mais da estrutura *bottom-up*, pois esta se caracteriza por instalar dados já existentes em outras bases de dados (“SKICAT”, sob o Sybase) e permitindo a criação de conceitos novos a dados já existentes, como é o caso das imagens IRAS, ROSAT, etc e a transferência para a nova base de dados (“SKICCOSMO” sob o Informix). Por conta disso, nesse trabalho analisamos os dados como são apresentados nas bases de dados já existentes, verificamos se é preciso modificá-los ou adaptá-los aos novos conceitos e aplicamo-los ao novo banco de dados. Parte-se, portanto, da “Visão do Usuário” para a construção das entidades mais abstratas.

É preciso notar que do ponto de vista do "DBMS", ou seja, do gerenciador de banco de dados, tanto o Sybase quanto o Informix se baseiam na tecnologia do banco de dados relacional. O interesse em transferir dados de um para o outro, também se deve ao fato de o Informix conter a tecnologia do "Objeto - Relacional".

1.4.2.8 O Estágio da Normalização

Uma vez construídas as entidades detalhadas em seus atributos e estabelecidas as bases dos relacionamentos entre elas, constitui fase importante na modelagem, a etapa conhecida como "**normalização**". Essa é a fase a qual se supõe que, uma vez completada, evitará redundância de informação e garantirá a chamada "integridade referencial" que veremos em 1.4.2.11.

O processo de normalização representa o trabalho de criar novas entidades auxiliares, remover e/ou transferir atributos de forma a que o banco de dados atenda às "**formas normais**", que são apresentadas a seguir.

1.4.2.9 As Formas Normais

Quando E. Codd, 1970, ([12]) lançou as bases do BDR, e subseqüentemente o aperfeiçoou, ele estabeleceu um conjunto de critérios conhecidos por "Formas Normais". Esses critérios são testados na fase de construção das diferentes "relações" em um banco de dados, fase essa também conhecida como modelagem dos dados. Essa é uma fase crucial na construção de uma base de dados. Rigorosamente, existem sete formas normais (Jennings, 1997, [33]), no entanto é comum se ater às três primeiras, que são:

- **A primeira forma normal** : Todo atributo é composto de um escalar. Não pode conter estrutura ou uma outra relação;
- **A segunda forma normal**: Todo atributo depende da chave primária de forma irredutível;
- **A terceira forma normal**: Estar de acordo com as duas primeiras formas normais e todo atributo que não pertencer à chave primária é independente de qualquer outro atributo na tabela.

Um modelo de banco de dados que obedece a esses três critérios é dito estar **normalizado**. O termo "normalizado" deve ser entendido no sentido matemático. Uma forma intuitiva de

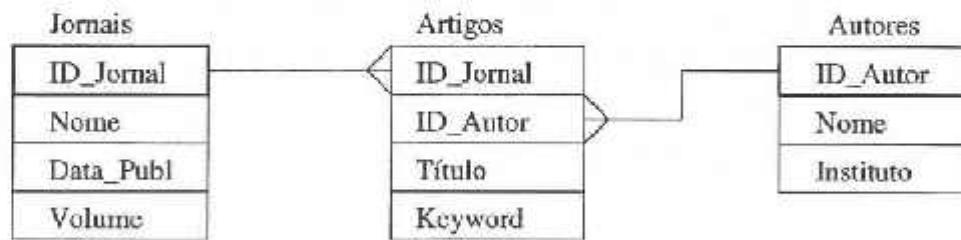


Figura 1.10: Decomposição de entidades.

Exemplo da transformação de um relacionamento $n : m$ para $1 : n$ com a inclusão de uma nova entidade (“Artigos”)

entender o modelo de um banco de dados é fazermos uma associação com um hiper-espaço descrito pelas bases, conforme aprendemos em álgebra linear. Cada relação é entendida como uma base, ou um “eixo”, nesse hiper-espaço. Os pontos de cruzamento dos eixos seriam, portanto, as chaves na relação onde se dá a junção. No jargão dos profissionais em banco de dados, essa modelagem é conhecida pela sugestiva expressão “construir o cubo” (Chapnick, 1997, [10]).

1.4.2.10 As Regras da Normalização

A experiência dos analistas tem levado a algumas regras de normalização de tal forma que as formas normais sejam obedecidas. Tomando-se por base o modelo de astronomia, vamos enumerar as regras, exemplificando-se:

1. Relacionamentos $1 : 1$ representam a mesma entidade. Na Figura 1.7 vê-se esse relacionamento entre “Autores” e “Editores”. Na realidade, esse fato indica que se pode criar uma entidade que abarque as duas;
2. Relacionamentos $n : m$ devem ser evitados. Podemos dizer (sempre com respeito à Figura 1.7) que substituímos o relacionamento entre as entidades “Jornais” e “Autores” por uma nova entidade (chamada, digamos, “Artigos”) que contém as chaves primárias de ambas as entidades originais, mantendo com elas um relacionamento $1 : n$, como na Figura:1.10.
3. Transformar atributos indexados em instâncias. Digamos que a entidade “Catálogos” seja decomposta de atributos tais como mostrado na Figura 1.11 à esquerda. Nessa entidade existem vários atributos representando a mesma grandeza mas que assumem valores diferentes para uma mesma instância. O ideal é, portanto, gerar uma

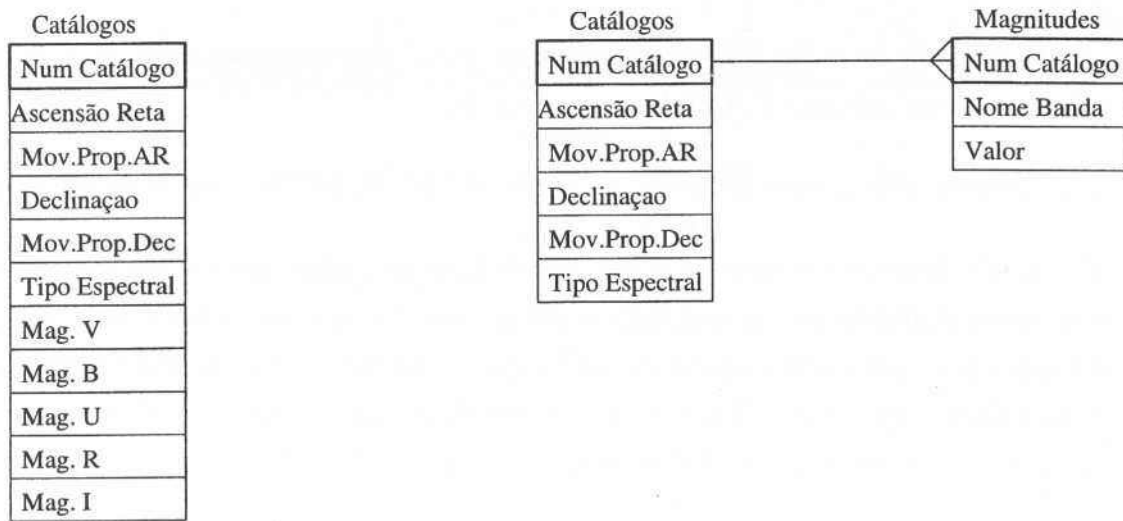


Figura 1.11: Atributos indexados.

Vários atributos indexados (Mag V, Mag B, etc) na tabela “Catálogos” à esquerda, exigem a decomposição dessa entidade em duas, no processo de normalização.

entidade nova (“Magnitudes”) em que os valores desses atributos são transformados em instâncias (Figura 1.11 à direita).

4. Não gerar atributos deduzidos de outros, como raízes de equações, variáveis compostas, nomes compostos de outros atributos ou nomes de outras entidades. É tentador gerar uma tabela ou atributo contendo o valor médio das magnitudes por classes, por exemplo. Contudo, as regras de normalização mandam evitar esse procedimento. Todas as deduções são obtidas no estágio da aplicação e não na armazenagem, dizem os cônegos do Banco de Dados Relacional ([28]).

É na obediência às regras de normalização que se chega a um banco de dados em conformidade com as formas normais. Quando se atinge essa condição diz-se que normalizou-se o modelo de banco de dados e este é dito “normalizado”.

Segundo o que estabelece Codd (1970, [12]), pode-se mostrar que um banco de dados nessas condições é o único capaz de garantir integridade e completeza das informações. No entanto, ao se construir um modelo nessas diretrizes, nada garante essa condição. Isso porque (Butzen & Forbes, 1997, [8]):

1. Um banco de dados nunca é completamente normalizado pois isso poderia significar um custo muito grande, o que, via de regra, é um limitador importante;

2. Os fabricantes de DBMS reconhecem que é praticamente impossível construir um sistema conforme E. Codd vem propondo;
3. Muitas vezes, em nome da viabilidade, é preciso quebrar regras de normalização.

É corrente falar-se nos meios das ciências de base de dados que a normalização é um ideal a se aproximar mas nunca a alcançar. Além disso, nesse meio, a cobrança maior é na fase da aplicação, que é onde o usuário “vê” o que está sendo feito. A “Modelagem dos dados” é uma técnica que vem exigindo maior especialização e experiência. Profissionais com esse perfil são raros no mercado de trabalho (Chapnick, 1997, [10]).

1.4.2.11 Integridade e Segurança

Nas diretrizes estabelecidas por Codd há uma preocupação com as regras de **integridade** da informação e sua **segurança**. Em certos aspectos esses dois conceitos se confundem. Codd enumera 5 tipos de integridade a serem observados (Codd, 1970, [12]):

1. Integridade de domínio (tipo D): garante que nenhum atributo possua valores que estejam fora do domínio definido no momento da criação da tabela;
2. Integridade de coluna (tipo C): garante que todos os valores de uma coluna sejam do mesmo tipo de dado (ex. real, inteiro, caracter, etc);
3. Integridade de entidade (tipo E): garante que as chaves primárias e/ou estrangeiras tenham referência válida;
4. Integridade referencial (tipo R): garante que não exista valor na chave estrangeira sem seu correspondente na chave primária, e
5. Integridade definida pelo usuário (tipo U): garante que as instâncias de uma entidade representam o conjunto completo do universo modelado.

O padrão SQL - ANSI de 1992 estabeleceu a observância dessas regras de integridade e atualmente todos os fabricantes de DBMS incorporaram-nas em seus produtos. Resta a última a ser discutida aqui e diz respeito ao domínio do universo de dados que se distingue um pouco da integridade do tipo “D” que, no interior do banco de dados, é garantida negando inserção de novos valores toda vez que se tenta inserir dados que extrapolam os domínios estabelecidos pelo administrador no momento da criação de uma dada tabela.

A integridade do tipo “D” é aquela que exige que uma dada coluna não tenha valor indeterminado (ou “NULL”) se o administrador que gerou a tabela declarou especificamente que aquela coluna não pode conter valor “NULL”. A **integridade “U”** de domínio na astronomia diz respeito às perguntas: a) os objetos registrados referentes a um certo campo representam a **totalidade** dos objetos naquele campo?; b) Os dados de um objeto representam **toda** a informação desse objeto? À primeira questão vamos nos referir como sendo o domínio numérico e à segunda, como domínio espectral. Sabemos que a resposta às duas questões é **não**, pois é impossível capturar todos os objetos do universo incluídos numa determinada região do céu, pois isso representa o conjunto de todos os objetos integrados naquela linha de visada, como também, pela própria natureza cartesiana do estudo de um objeto, conseguimos obter apenas parte da informação sobre ele.

O objetivo de um banco de dados na astronomia pode ser o de completar um dos dois preceitos da integridade de domínio astronômico: o numérico e o espectral, ou ambos. Em qualquer caso o sucesso de sua construção é limitado pelas técnicas observacionais no que tange a exigüidade de espaço em memória de massa.

Alguns bancos de dados são, por construção, restritos a uma parte da informação sobre os objetos, isto é, pretendem cobrir o domínio numérico. São esses os casos dos bancos de dados obtidos de um programa específico (*survey*). Esse é, originalmente, o caso do SKICAT que se propunha ao armazenamento de dados do POSS-II e imagens CCD em filtros do sistema Thuan-Gunn (Thuan & Gunn, 1976, [60]). Outros bancos de dados não se limitam ao domínio espectral apenas, contudo demandam muito tempo para garantir um maior grau de integridade numérica. Esse é o caso de bancos de dados resultado de compilação e daqueles resultado de campanhas em observatórios.

O SKICCOSMO, a exemplo do SKICAT, propõe-se, antes de tudo, a garantir a integridade do domínio numérico, para, num segundo passo, ampliar seu domínio espectral. São preceitos que serão alcançados apenas com o tempo, na medida em que os recursos materiais vão sendo obtidos.

1.4.2.12 A Linguagem Estruturada de Consulta (SQL)

Uma vez que E. Codd apresentou as bases do modelo de banco de dados relacional, muitos laboratórios, públicos e privados, iniciaram pesquisas no sentido de desenvolver linguagens capazes de lidar com esse modelo. Várias linguagens apareceram. Butzen & Forbes, 1997 ([8]) listam, entre as principais: a “QUEL”³⁰ desenvolvida na Universidade de Berkeley

³⁰QUEL: acrônimo de *Query Language*

pelo grupo de M. Stonebraker que apresentou o sistema “Ingres”, e a “SQL” (*Structure Query Language*), desenvolvida pela IBM que passou a comercializar o “DB2”, gerenciador de banco de dados que dominou o mercado a partir de meados dos anos 70.

Após o sucesso do produto da IBM, muitos sistemas apareceram, entre os proeminentes: “Oracle” e o “Sybase”. O grupo de Berkeley acabou por adotar a SQL e lançou, posteriormente, o “PostgreSQL”. M. Stonebraker ajudou a criar a **Informix**, levando os conceitos mais avançados do PostgreSQL para criar o DBMS “**Illustra**”. Daí, cada sistema seguiu uma vertente e hoje convivem, a Informix como aplicativo comercial e a PostgreSQL distribuída em domínio público. A Informix, particularmente, é estruturada para tratar grande quantidade de informação e garantir segurança, que é a demanda atual no mercado.

As razões principais do sucesso da “SQL” são, nas palavras de Bowman *et al*, 1998, [6], a “solidez matemática e o apelo intuitivo”. De fato, pode-se dizer que a “SQL” seria uma aplicação da ciência da “linguagem natural”. Por um lado, suas instruções não deixam margem de dúvida ou ambiguidade, e é relativamente fácil transformá-las em frases em língua natural, mais especialmente o inglês. Eis um exemplo:

```
select name,age from persons where genre = 'male'
```

Nesse exemplo, a instrução da consulta pede para selecionar o nome e a idade de todos os registros da tabela “persons” que são sexo masculino. Bem entendido, é preciso que a tabela “persons” possua os atributos “name”, “age” e “genre”, definidos em colunas específicas.

Abaixo, um exemplo um pouco mais complexo envolvendo as tabelas “persons” e “office”:

```
select p.name,p.age,o.title from persons p,office o
where p.personID=o.personID and p.genre='male'
```

À relação $p.personID=o.personID$ chamamos **junção interna**. O atributo `personID` está presente em ambas as tabelas, seja em “persons”, seja em “office”. Em princípio esse atributo na tabela “persons” faria parte da chave primária, enquanto na tabela “office”, ele seria uma chave estrangeira. A junção interna permite selecionar para a saída apenas os pares resultantes do produto cartesiano entre as duas tabelas que obedecem àquela relação. No atributo `personID`, diz-se que as tabelas “persons” e “office” mantêm um relacionamento de **um** no lado de “persons” para **muitos** no lado de “office”.

Quando o banco de dados é estruturado completamente com a definição de todas as suas tabelas e estabelecidos todos os seus relacionamentos, uma consulta em “SQL” pode tornar-se longa e extremamente complexa. Apesar disso, é a linguagem que se tornou padrão nos meios de desenvolvimento de bancos de dados relacional por ser, entre todas, a mais compacta e simples. A complexidade, portanto, vem da natureza do problema e não da estrutura da linguagem.

1.4.2.13 Os Índices

Duas são as razões de porque o banco de dados relacional se impôs ao hierárquico em se tratando de aplicação genérica: 1) a criação do conceito de “pilha”³¹ em programação de “kernel”³²; 2) a invenção da árvore binária na geração de índices. A primeira permitiu com que a programação fosse totalmente referenciada a endereçamento por indireção³³, ao contrário do que vinha sendo feito quando os endereçamentos eram absolutos. A segunda livrou os BDR’s da pesquisa seqüencial o que pouco acrescentaria aos primeiros bancos de dados. Com a criação do conceito de pilha, não era mais preciso administrar a ocupação da memória através de seus endereços, bastando apenas saber a posição de uma certa variável na pilha. Do seu endereço ocupa-se o gerenciador da pilha. Com isso houve um avanço no conceito de programação ([14]).

A criação do conceito de pilha, portanto, gerou um grande dinamismo no conceito de programação e tornou anacrônico o banco de dados hierárquico, que baseia-se na geração de endereços absolutos. Finalmente, veio o domínio da técnica da árvore binária cujo desenvolvimento está intimamente ligado ao conceito de pilha (ver Knuth, 1968, [39] §2.3.3, ou Press *et al*, 1988, [47] §8.2). A árvore binária é construída de tal forma que o valor associado a um vértice é sempre superior ou igual a qualquer outro abaixo (ver Figura 1.12). Na árvore binária a procura por um dado reduz-se a um processo recorrente de se responder verdadeiro ou falso.

³¹Pode-se entender a “pilha” intuitivamente pensando em termos da “notação polonesa invertida” adotada nas primeiras calculadoras HP. Os valores a serem operados vão se “empilhando” e sendo recuperados na medida em que se lança as operações. O conceito de “pilha” opõe-se ao conceito de “ponteiro”. O ponteiro contém um endereço de memória. Seu valor varia em função das necessidades do programa. No caso da pilha, o valor do endereço da memória é fixo, variando o conteúdo dessa memória.

³²Entende-se como *kernel* o conjunto de programas que constituem um sistema operacional.

³³**Indireção:** Essa é uma ferramenta que aparece no *assembler* e que permite a construção de vetores e *arrays* nos compiladores das linguagens de alto nível ([43]). Trata-se de interpretar o conteúdo de uma unidade de memória não como valor mas como o endereço da variável. Assim, o programa vai buscar em uma certa unidade de memória o endereço na memória da variável e não o seu valor. Trata-se pois de um **endereçamento indireto** e, em linguagem de alto nível, como o FORTRAN, significa definir uma

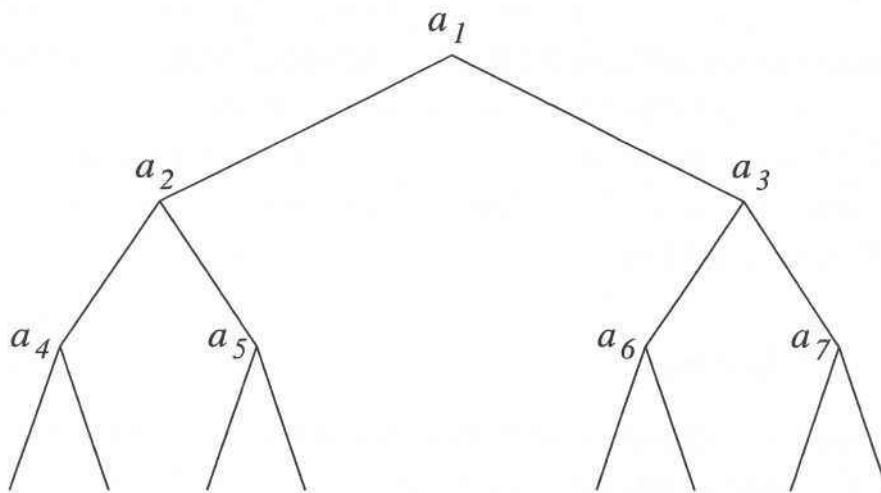


Figura 1.12: Árvore binária.

O valor associado a um vértice é sempre superior ou igual a qualquer outro abaixo.

A construção da árvore binária exige que o DBMS acesse todas as instâncias dos atributos de uma tabela. Essa operação equivale a uma pesquisa seqüencial onde o valor a ser recuperado encontra-se na última linha da tabela. Por isso a construção da árvore binária é onerosa. Existem técnicas para acrescentar “folhas” à árvore de forma a fazê-la atender apenas parcialmente ao preceito da árvore binária (ver legenda da Figura 1.12). Regularmente faz-se uma atualização da estrutura da árvore, onde esta assume sua forma ideal. No Informix, essa tarefa de administrador representa a instrução de consulta: `update statistics` ([31]). Na estrutura apresentada, a árvore binária é um exemplo de aplicação da “árvore genealógica” reversa: se cada vértice representa uma pessoa, a ela está associado um “pai” e uma “mãe”. Sob essas condições, a lógica binária se apresenta como ideal nesses casos. As premissas apresentadas por Codd, porém, exigem uma lógica que passou a ser chamada de “lógica de três camadas”, pois há uma terceira situação: o valor “NULL”. O valor “null” ou nulo, deve ser entendido não como um elemento neutro e sim como uma condição para atender a uma exigência da integridade de domínio. Quando o sistema encontra um valor “null” deve ser entendido que aquela instância do atributo possui um valor desconhecido. A analogia com a “árvore genealógica” é ilustrativa: uma pessoa pode ser declarada sem pai ou sem mãe³⁴. Mesmo que biologicamente qualquer

indexação, por exemplo, em um *array*.

³⁴Soube de um caso recente em que um juiz do estado da Califórnia, EUA, declarou uma criança sem

ser vivo tem um pai e uma mãe (em alguns casos é o mesmo), legalmente aquela situação é possível. Transportando o exemplo para a “árvore binária”, o valor “null” é um valor existente mas desconhecido. A lógica das três camadas, ainda hoje, é objeto de debates teóricos complexos. Codd,1990 ([13]) dedica o Capítulo 8 de seu livro a essa questão.

1.4.3 A Organização de um Banco de Dados orientado para a Astronomia

Sabemos que nem sempre é possível construir um banco de dados rigorosamente de acordo com o estabelecido por Codd (ver 1.4.2.10). Resta saber o que isto representa na construção do SKICCOSMO³⁵. Temos que:

- É necessário definir, sem ambigüidades, uma chave primária;
- Uma entidade é definida por seus atributos, portanto, uma instância é “imutável”;
- Instâncias diferentes de atributos definem diferentes instâncias de um objeto.

Os objetos astronômicos seguem apenas aproximadamente essas diretrizes: um mesmo objeto pode variar seus atributos fotométricos ou espectroscópicos ou pode se mover no céu mudando suas coordenadas ou mesmo essas coordenadas são mutáveis diante dos movimentos da terra. Porém, de 1.4.2.10, sabemos que podemos construir um banco de dados apesar dessas características. Mas a especificidade de nossos dados faz com que:

1. A chave primária seja definida **estatisticamente**. Portanto ela pode ser re-determinada, se novos dados trouxerem mais precisão na sua determinação. Por conseguinte, a **integridade referencial** pode não estar completamente assegurada mas pode ser melhorada com o tempo;
2. Um objeto celeste não necessariamente corresponde a um objeto no banco de dados. Dados que se acreditava pertencerem a um objeto celeste podem ser atribuídos parcial ou integralmente a um outro das imediações (vide 3.5);
3. Um objeto definido no banco de dados pode não existir, sendo apenas um resultado espúrio das ferramentas de identificação. Essas ferramentas utilizam critérios estatísticos, e os resultados guardados devem ser entendidos como tais. Então, um

ambos os pais legais.

³⁵Ver adiante em 1.5 o significado de SKICCOSMO.

objeto armazenado deve possuir parâmetros determinando o nível de confiança de sua classificação.

O resultado de uma consulta a um banco de dados construído nessas condições deve ser entendido dentro das três limitações acima. Consultar o SKICCOSMO tem paralelos com o ato de “observar” o céu real. Processar os resultados dessa consulta equivale à redução dos dados observacionais³⁶.

Na medida em que os métodos de tratamento e análise dos dados avançarem, eles serão aplicados aos dados do SKICCOSMO no sentido de melhorar a confiança das consultas. Da mesma forma, na medida em que as técnicas observacionais vão propiciando significativa melhoria na identificação e classificação dos objetos celestes, o SKICCOSMO propiciará, em consequência, dados mais precisos e confiáveis. A melhoria da tecnologia computacional e de armazenagem de informação fará o SKICCOSMO mais ágil no fornecimento das informações requisitadas.

1.5 O Banco de Dados SKICAT

O SKICAT (*Sky Image Cataloging and Analysis Tool*) é um banco de dados desenvolvido por Weir (1994, [66]). Ele decorre da digitalização das imagens fotográficas do POSS-II (*The Second Palomar Observatory Sky Survey*) pelo STScI (*Space Telescope Science Institute*). Nesse trabalho é introduzida a técnica de classificação baseada no algoritmo *Decision Tree*. Por esse método chega-se a um índice de confiança de 90% na classificação de objetos até a magnitude visual 21^m.

O SKICAT surge a partir da necessidade de armazenar, de forma coerente, os dados advindos da digitalização das placas do POSS-II no STScI. Para tanto ele foi construído com base no *Sybase*, software que está entre os pioneiros dos bancos de dados relacional (ver adiante em 1.2.4) e tem evoluído conjuntamente com seus concorrentes, sendo dos mais populares no mercado. Para armazenar os dados, Weir (1994) escolheu gerar uma tabela para cada imagem tratada, seja ela advinda do DPOSS (*Digitalized Second Palomar Observatory Sky Survey*), seja de imagens CCD. O software que dá suporte à operação com o SKICAT se encarrega de gerar as tabelas e carregá-las na medida em que o usuário vai fornecendo novas observações ou placas digitalizadas. A integração dos dados de diferentes observações de uma mesma região do céu é tratada especialmente pelo processo chamado

³⁶Retomaremos esse assunto em 6.4.2.

de *Matching*³⁷ no contexto do SKICAT. Esse processo é executado somente quando há interesse do usuário em obter dados cruzados de diferentes catálogos. Um catálogo especialmente dedicado a essa operação é, assim, gerado e guardado no banco de dados. Sua disponibilização é possibilitada através de ferramentas especialmente desenvolvidas para esse fim. Uma vez obtidos os dados decorrentes desse procedimento, o catálogo *matched* é, geralmente, apagado do banco de dados.

O acesso ao SKICAT é feito por duas vias (Weir, 1994, [66]):

1. Por aplicativos desenvolvidos em ambientes GUI³⁸ sob o X11³⁹;
2. Por comandos diretamente enviados ao interpretador de SQL (ver adiante em 1.4.2.12).

Ambas as vias têm suas limitações. A primeira via limita a consulta ao ambiente de rede local - pois o X11 GUI é proibitivo para aplicações remotas, uma vez que exige uma troca muito grande de informação no processo cliente - servidor, além de sua liberação representar ameaça à segurança da rede. A segunda via representa a elevação do usuário ao mesmo nível do administrador do sistema, com evidente ameaça à segurança da própria informação contida no banco de dados. Os detalhes da estrutura do SKICAT serão descritos no Capítulo 3.1.

Quando se pensou em disponibilizar os dados do DPOSS para a comunidade astronômica, verificou-se claramente que isso não era possível via SKICAT. Suas características não se enquadram no perfil de um banco de dados de acesso fácil e seguro. Além dos argumentos apresentados acima, a operação com o SKICAT exige conhecimento específico, tanto da estrutura do SKICAT quanto dos processos de consulta. Foi diante desse problema que surgiu a idéia do **SKICCOSMO: *Sky Integrated Catalogs for Cosmology***, cujas características são apresentadas em 3.2.

Apesar da estrutura relacional do Sybase, o SKICAT tem pouco ou quase nada dela. Como conseqüência as instruções em SQL a serem construídas no SKICAT diferem pouco

³⁷Processo em que objetos comuns a duas imagens são identificados. Optamos por não traduzir a palavra *matching*. Sua tradução literal (casamento) tem um significado apenas marginal no conceito aqui exposto. Diz-se, por exemplo, "as cartas do baralho estão casadas", mas quando se fala em "casamento das cartas do baralho" pensa-se em véu e grinalda. Além disso, o termo *matching* é um termo corrente nessa área da astronomia.

³⁸GUI: *Graphic User Interface* é um conceito geral utilizado para se referir a toda interface gráfica que permite o usuário interagir com o computador. Exemplos: MS-Windows, CDE-Solaris, KDE-Linux, SAOimage, Netscape, etc.

³⁹X11: Ambiente gráfico desenvolvido pelo laboratório de informática do MIT, EUA. Tornou-se o padrão GUI de todas as máquinas rodando UNIX.

de instruções das antigas QBE⁴⁰ (ver em 1.2.3) e, por isso mesmo, não se utiliza de propriedades relacionais (Weir, 1994, [66] e Butzen & Forbes, 1997, [8]). Ao construirmos o SKICCOSMO, em bases relacionais, e já concebido para seguir as propriedades objeto-relacionais, foi necessário repensar o papel dos processos de *matching* no interior do SKICAT. Em outras palavras, o usuário deve ter em mãos ferramentas de consulta de forma a obter todas as informações que queira sobre uma dada classe de objetos, sem que, para isso, ele seja obrigado a conhecer todos os detalhes de como os dados desses objetos foram armazenados. O usuário deve poder fazer sua consulta de forma simples e intuitiva e obter como resultado todos os dados pertinentes a sua consulta que estiverem disponíveis no SKICCOSMO.

Para que tudo isso seja possível, devemos abandonar a estrutura com a qual o SKICAT foi concebido. O processo de *matching* dos diferentes campos que são registrados no banco de dados é, forçosamente, parte integrante da construção do banco de dados. Essa é a prerrogativa do banco de dados relacional. O problema é que o que caracteriza o universo de estudo da astronomia é que a identificação de objetos e portanto, o *matching* entre os diferentes conjuntos de medidas de um dado campo é, *per se*, uma área particular de estudos. A consequência é que:

1. A chave estrangeira (ver em 1.4.2.6), que é uma das características de um objeto, não é definitiva nem imutável;
2. A estrutura relacional é uma operação sobre um procedimento estatístico.

Ambas as características estão fora dos preceitos de um banco de dados puramente relacional e são peculiares à astronomia. Um exemplo do primeiro caso é a identificação ótica de quasars. Pode ser que um objeto que se acredita ser a parte ótica de um quasar, efetivamente não esteja relacionado a este, mas sim a um objeto vizinho. Quanto ao segundo caso, trata-se do que se costuma chamar algoritmos de *matching*, que podem ser encontrados relatados na literatura (ver Murtagh, 1992, [45] para uma revisão). Desse assunto, essa tese trata em 3.5.

Uma vez resolvido o problema da construção de um banco de dados relacional com base nos dados do POSS-II deve-se passar pelo estágio de transporte dos dados. A esse respeito deve-se construir uma política de transporte com base nas ferramentas que se

⁴⁰QBE: *Query By Example*. Linguagem imediatamente anterior à SQL (descrita em 1.4.2.12) que serve para se construir consultas em bancos de dados hierárquicos ou pré-relacionais. Contém instruções como "list", "browse", etc. Apesar de sua característica intuitiva não dava conta das possibilidades do relacionamento.

tem a mão. Quando foi assinado o convênio com a Informix para a cessão do DBMS⁴¹ no Observatório Nacional, essa empresa partiu do princípio de que o transporte poderia ser feito simplesmente por um procedimento de “*exportação*” do SKICAT pelo Sybase e “*importação*” para o SKICCOSMO pelo Informix. Para isso, tanto um quanto outro DBMS possuem ferramentas que permitem executar esse procedimento facilmente. No decorrer do trabalho, quando foi-se tomando consciência da estrutura em detalhe do SKICAT e o quanto este se afasta do chamado banco de dados relacional “normalizado” (ver 1.4.2.10), ficou claro que a simples “transposição” dos dados não era a forma mais adequada de transportá-los. Elaborou-se, em consequência, uma estratégia de transferência em que são colocados três servidores em ação: o servidor SKICAT, o servidor SKICCOSMO e um servidor intermediário. Essa estratégia é apresentada em 3.6.

A operação com o SKICCOSMO é apresentada no Capítulo 4. Há uma descrição geral dos comandos, e exemplos são fornecidos no final. Toda a operação se dá, obviamente, através de um *browser WEB*, de maneira que não há restrição para quem possa vir a usar o banco de dados. Uma identificação sumária do usuário é necessária, sobretudo para se ter controle de quanto e como está sendo usado o banco de dados. Uma linguagem de consulta foi elaborada de maneira a permitir que o usuário, com uma leitura introdutória do manual, seja capaz de fazer suas primeiras consultas e testes de consultas. Os exemplos servem para ilustrar o uso dessa interface.

A elaboração da linguagem de consulta trouxe um dilema freqüente para aqueles que desenvolvem aplicativos: de um lado deve-se decidir a “profundidade” com que o usuário pode “alcançar” os dados, questão freqüentemente denominada de “granularidade”, e, de outro, a facilidade de construção da consulta. Em outros termos, podemos colocar que a “facilidade” de uso, seja qual for o sistema, está na razão inversa do que o usuário “pode” fazer. Temos que considerar também que quanto mais um usuário tem acesso “granular” ao sistema, mais ele estará perto de cometer erros, o que traz à tona, também, o problema da segurança dos dados e do sistema. Em resumo, existem dois extremos em que um usuário pode ser enquadrado:

1. O usuário que tem autorização para recuperar toda e qualquer informação nas condições que desejar e no formato que bem entender. Vamos chamar esse usuário de “gerente”;

⁴¹ Acrônimo de *Data Base Manager System* que vem a ser o software que gerencia e administra os dados armazenados (ver em 1.2.1). Existem vários, no mercado, destacando-se o **Oracle**, **Sybase**, **MS-SQL Server** e **Informix**, este último adquirido com o fim de servir de base para o SKICCOSMO.

2. O usuário que não pode recuperar outra coisa, senão dados já devidamente pré estabelecidos, pré formatados e combinados de maneira a que este usuário veja somente a informação estrita e que se adapte aos seus conhecimentos. Vamos chamar esse usuário de "cliente".

Não é difícil estabelecer um paralelo desses usuários com os de uma agência bancária, por exemplo. O gerente de uma agência promove movimentações técnicas transparentes ao cliente, que, em suma, está interessado apenas no extrato do movimento de sua conta. Um cliente, ao cometer um erro, quando operando em um terminal, terá como consequência, no máximo, uma mensagem de erro, e o terminal recusa continuar a operação. Um gerente, por sua vez, ao cometer um erro poderá trazer prejuízos aos clientes ou ao próprio banco.

Da mesma forma, um banco de dados deve estabelecer critérios precisos para determinar em que grau um usuário pode operar: se mais próximo do gerente ou mais próximo do cliente. Como consequência, esse usuário terá de ser treinado com maior ou menor profundidade; as ferramentas de que ele irá dispor terão maior ou menor grau de dificuldade de manipulação. Secundariamente, a linguagem de consulta terá mais ou menos apelo intuitivo.

Da maneira que a linguagem de consulta foi concebida, procurou-se permitir que, de uma forma ou de outra, o usuário tenha acesso a toda informação, ou senão, pelo menos àquela relevante para levar a termo sua pesquisa.

Finalizando esse trabalho, citamos alguns exemplos de pesquisas sendo levadas com o auxílio do SKICCOSMO, particularmente a procura por quasares para $z \simeq 4$ (Capítulo 5). A procura baseia-se no procedimento usado por Kennefick *et al*, 1995 ([37]).

Essa tese está organizada da seguinte forma:

- Capítulo 1: Introdução e apresentação sumária da teoria dos bancos de dados;
- Capítulo 2: Breve descrição dos tipos de bancos de dados na astronomia;
- Capítulo 3: Descrição das estruturas do SKICAT e do SKICCOSMO e suas diferenças. O problema da definição das chaves primárias e estrangeiras são discutidas nesse capítulo, bem como os métodos de *matching*;
- Capítulo 4: Apresentação da operação no SKICCOSMO e sua linguagem de consulta. Alguns exemplos ilustrativos são apresentados no final;
- Capítulo 5: Procura de QSO's em altos *redshifts* a partir do POSS-II;

- Capítulo 6: Discussão das linhas de pesquisa que podem ser levadas a termo com o auxílio do SKICCOSMO.

Capítulo 2

Os Bancos de Dados na Astronomia

2.1 O que é e por que queremos um Banco de Dados?

2.1.1 O que é um banco de dados?

Banco de dados é um conjunto de informações organizadas de forma coerente. Essa coerência deve garantir que a informação seja **íntegra**, seguindo os preceitos de **confiabilidade** e **completeza**. Uma informação é dita confiável se ela realmente diz respeito à entidade ou grandeza que se está estudando, e, ela é completa se a informação da entidade ou grandeza é recuperada com todos os atributos necessários para a sua definição. A matéria que trata da confiabilidade e completeza de um banco de dados é conhecida como **Integridade**.

Codd classificou 5 tipos de integridade (ver 1.4.2.11). Os quatro primeiros, são tratados no interior do DBMS¹. O último tipo, a integridade definida pelo usuário, diz respeito ao universo de que estamos tratando e possui interesse especial na astronomia. Usualmente, nas empresas comerciais, a integridade definida pelo usuário é garantida ao nível do gerenciamento dos dados.

A integridade definida na astronomia tem relação com o universo de objetos astronômicos e com o fato de que os parâmetros que podemos obter são: atributos posicionais, fotométricos e espectroscópicos. Chamaremos a esse universo e seus parâmetros de domínio astronômico, sem confundir com a integridade de domínio de Codd, como descrita em 1.4.2.11.

Como já visto na Seção 1.4.2.11, a integridade de domínio astronômico responde à pergunta: “Os dados que recuperei representam a totalidade da amostra que estou estu-

¹Desde que o banco de dados seja normalizado (ver em 1.4.2.10).

dando?”. Se consultamos uma base de dados como o NED², por exemplo, podemos fazer duas perguntas, no que tange à integridade de domínio numérico: 1) os dados disponíveis no NED englobam todas as galáxias já catalogadas? e 2) As galáxias catalogadas, no NED representam todas as galáxias do universo? À primeira pergunta, pode-se responder que, se o NED não possui informações de todas as galáxias já catalogadas está muito perto de possuí-las. Com respeito à segunda pergunta a resposta é, com certeza, “não”. Além de perguntarmos se todas as galáxias estão catalogadas, podemos, ainda, inquirir, com respeito à integridade espectral: “Os dados catalogados das galáxias contêm tudo o que já foi estudado sobre elas?” e “Dessas galáxias catalogadas já se sabe **tudo** o que se pode saber a respeito delas?”. Bem entendido, essas questões se referem ao que é de interesse astrofísico. Se estamos falando de um cadastro de clientes de um banco comercial, por exemplo, obviamente as informações sobre um cliente não representam tudo sobre a pessoa em questão, mas somente aquilo que tem interesse financeiro. No caso astrofísico, todas as informações materiais sobre uma galáxia têm, *a priori*, interesse de estudo.

2.1.2 Por que queremos um banco de dados?

Dada a crescente qualidade das observações, seria necessária uma quantidade de astrônomos que a comunidade não tem. Muitas vezes as observações são aproveitadas apenas num primeiro momento para eventuais descobertas ou para a determinação de uma grandeza com precisão. Depois disso, os dados são armazenadas em memória de massa aguardando uma análise mais aprofundada, que amiúde, não acontece. Diante dessa situação, os centros operadores de sondas espaciais e grandes telescópios iniciaram, sobretudo no final da década de 80, esforços no sentido de prover acesso a seus dados armazenados de forma a:

1. Disponibilizar os dados obtidos por outros pesquisadores ou de pesquisas (*survey*) a toda a comunidade para permitir análises mais aprofundadas;
2. Oferecer à comunidade informações cruzadas com outras técnicas ou outros observatórios de maneira a enriquecer o banco de dados no que tange a integridade de domínio numérico e espectral.

Na seqüência, veremos a descrição de alguns dos bancos de dados mais proeminentes da comunidade. Entre eles encontram-se tanto os “tradicionais”, tais quanto o SIMBAD e o

²NED: *NASA/IPAC Extragalactic Database*. Banco de dados essencialmente voltado para objetos extragaláticos, goza de grande popularidade entre os astrônomos dessa área.

NED, como alguns serviços que se prestam sobretudo ao cruzamento de dados dos mais diversos bancos de dados à disposição pela internet.

Entre as dificuldades de se montar um banco de dados na astronomia está a heterogeneidade da informação, tanto em parametrização, quanto na forma que os dados são armazenados. O primeiro aspecto, por exemplo, leva a uma grande variação dos valores de um mesmo atributo. É o caso, entre outros, da magnitude de um objeto, pois essa grandeza, estatisticamente, é uma variável observada, portanto sujeita a variações aleatórias e às possíveis variações decorrentes das propriedades intrínsecas do objeto. O segundo aspecto está relacionado com a falta de sistemática normalizada para a armazenagem da informação. Cada centro de pesquisa armazenou suas informações de acordo com suas conveniências, voltado para seus próprios agentes de consulta. Para tentar resolver esse problema, alguns centros estão desenvolvendo agentes baseados na programação orientada objeto (POO) e bancos de dados objeto-relacionais e a partir de suas diretrizes nos propusemos construir o SKICCOSMO de modo a conformá-lo a um projeto global tal qual o "observatório virtual", como veremos no Capítulo 6. Entre esses esforços encontramos o "AMASE" e o "MAST", que são meta gerenciadores de banco de dados, vistos a seguir.

2.2 Os Tipos de Banco de Dados na Astronomia

Em termos gerais encontramos cinco tipos de banco de dados na astronomia:

1. Bancos de dados de objetos catalogados;
2. Bancos de dados de catálogos de observatórios;
3. Bancos de dados de exposições de campos celestes ou espectros (*surveys*);
4. Catálogos de objetos identificados e classificados de um *survey*;
5. "Meta" bancos de dados, integrando dados de vários observatórios, satélites e *surveys*, segundo um critério de associação, seja em função da técnica observacional, seja de catalogação.

O SKICCOSMO, como veremos na Seção 3.2, pode ser classificado como um banco de dados do tipo 3, pois é derivado do POSS-II, que é um *survey*. Ele se reveste de uma especificidade, obtida da própria proposta de sua criação: permitir ao usuário consultas segundo classes ou propriedades de objetos definidas pelo próprio usuário. Enquanto os bancos de dados

encontrados correntemente permitem pesquisas segundo o nome de identificação do objeto, de sua posição aproximada, ou, eventualmente, de magnitude ou tipo espectral dentro de uma certa faixa, o SKICCOSMO pretende permitir relações entre os atributos dos objetos além de propriedades estatísticas de atributos. O SKICCOSMO, de acordo com as propriedades que apresenta, é ideal para o auxílio de descobertas de classes de objetos ou preparação para missões de observação de grandes telescópios. Nesse particular, ele se insere perfeitamente no projeto baseado em Caltech do “Observatório Virtual”, desenvolvido em Caltech, que será descrito no Capítulo 6.

A seguir faremos uma resenha de alguns dos principais bancos de dados “*on line*” disponíveis hoje, na internet. Não pretendemos esgotar o assunto na matéria, senão dar uma idéia de como se apresenta nos dias de hoje esse crescente ramo da astronomia voltado para oferecer ferramentas de pesquisa ao astrônomo.

2.3 Alguns Bancos de Dados “*On line*”

A proliferação de dados disponíveis ao público, via internet, é um indicador do que há muito se observa na astronomia: a “inflação” de dados. Por conta disto a astronomia observacional apresenta-se como sendo um dos campos de investigação mais profícuos da atualidade.

Alguns dos bancos de dados aqui listados são resultado de um enorme trabalho de compilação de informações de catálogos e de referências na literatura. Outros são compostos de ferramentas para procura em diversos catálogos e *archives*³. Outros, ainda, são meramente *archives*. A quase totalidade dos bancos de dados aqui listados é compatível com a tecnologia do banco de dados relacional de forma que nos omitimos em comentar a respeito desse particular na descrição de cada banco de dados, abaixo.

Apesar dos esforços em unificar a informação, ainda não vem sendo adotado um procedimento padrão na construção dos bancos de dados encontrados na internet. Há uma iniciativa recente, elaborada em Caltech que se denomina *Observatório Virtual*, que descrevemos em 6.4.2.

A seguir são listados alguns bancos de dados disponíveis na internet atualmente.

³Sistemas depositários de arquivos a serem baixados via FTP.

2.3.1 SIMBAD

Acronismo: *Set of Identifications, Measurements and Bibliography for Astronomical Data*;

Base: CDS (*Centre de Données Astronomiques*) Strasbourg, FR;

Mantenedor: CNRS (*Centre National de la Recherche Scientifique*), FR;

Iniciadas operações 1981 (em 1983 para galáxias e objetos não estelares);

Origem dos dados: Partiu da unificação dos *Catalogs of Stellar Identifications* e *Bibliography Star Index*, e desde então tem compilado dados de catálogos adicionais e artigos na literatura específica;

Retorna: Tabelas⁴;

Domínio: Objetos galácticos e galáxias; posições, atributos fotométricos e citações na literatura;

URL: <http://simbad.u-strasbg.fr/Simbad>⁵.

Este foi um dos primeiros serviços disponíveis para consulta via acesso remoto. O usuário possui uma conta de acesso e paga por consulta feita. A consulta pode ser feita através de um meio de identificação do objeto astronômico, seja pelo nome, identificação de catálogo, índice espectral, posição ou referência na literatura.

O SIMBAD é um conhecido banco de dados entre os astrônomos, sobretudo aqueles que trabalham com objetos galácticos. Em meados do ano 2000, o SIMBAD conta com cerca de 2.8 milhões de objetos catalogados, 7.6 milhões de identificadores, 115 mil referências bibliográficas e 3.3 milhões de citações de objetos em publicações diversas. O SIMBAD é o pioneiro em oferecer catálogo de citações de objetos. A Figura 2.1 mostra a janela de consulta do SIMBAD.

2.3.2 IUE

Acronismo: *International Ultraviolet Explorer*;

⁴Tabela é o produto de uma consulta. Ela contém colunas de textos descrevendo os atributos dos objetos selecionados.

⁵Alguns endereços URL podem mudar com o tempo.

1. Enter an identifier, coordinates or a reference code:

Examples: sirius, M 31, 12 30 45 +10 20, 1986ABA, 505.33K
How to write an identifier can be found in the [dictionary of nomenclature](#).

a. For identifiers you can choose to query :

b. For coordinate and around objects queries, define a radius :

c. For coordinate queries, define the system : epoch : equinox :

=> You may also create [samples](#) using basic criteria

2. Optional output options :

a. Lists should contain objects.

b. measurements

c. bibliography from to

d. Display coordinates

	1st frame :	2nd frame :	3rd frame :
Coordinate system :	<input type="text" value="FK5"/>	<input type="text" value="FK4"/>	<input type="text" value="Galactic"/>
Epoch :	<input type="text" value="2000.0"/>	<input type="text" value="1950.0"/>	<input type="text" value="2000.0"/>
Equinox :	<input type="text" value="2000.0"/>	<input type="text" value="1950.0"/>	<input type="text" value="2000.0"/>

Figura 2.1: Janela de consulta do SIMBAD.

Base: NSSDC (*National Space Science Data Center*), US;

Mantenedor: NASA,US / ESA, Europa / PPARC (*Particle Physics and Astronomy Research Council*), UK;

Iniciadas operações: 1978 (encerrou coleta de dados em 1996);

Origem dos dados: Dados do satélite IUE;

Retorna: Tabelas, imagens, arquivos FITS de espectros via FTP;

Domínio: UV (1150 - 3200 Å);

URL: <http://archive.stsci.edu/iue/>.

Lançada em 20 de janeiro de 1978, a sonda espacial IUE permaneceu mais de dezoito anos e meio em operação, perfazendo mais de 111 mil observações de 9300 objetos. Disponibiliza cerca de 1.1 Tby de dados integrados no projeto **MAST** (ver abaixo em 2.3.9). Dados do IUE foram distribuídos em forma de *media* LASER para vários observatórios no mundo, inclusive o IAG-USP, que os disponibilizou para a comunidade astronômica brasileira e latino americana através do sistema VAX - VMS. Na Figura 2.2 vemos a janela de consulta do IUE.

2.3.3 STARVIEW

Acronismo: *Space Telescope Archive Catalog Viewer*;

Base: STScI, US / ESO, Europa;

Mantenedor: STScI/NASA, US / ESO, Europa;

Iniciadas operações: 1988;

Origem dos dados: Programas observacionais, acervo público;

Retorna: Visualização, arquivos FITS via FTP, pré-visualização de mapas do céu (*finding charts*) de auxílio à observação no telescópio espacial Hubble;

Domínio: ótico (3400Å- 7500Å), UV, infravermelho, imagens e espectros de alta/baixa resolução;

IUE Archive Search

Object Name		Resolver	
<input type="text" value="I"/>		<input checked="" type="radio"/> SIMBAD <input checked="" type="radio"/> NED <input type="radio"/> Don't resolve	
<input type="button" value="Get coordinates"/>			
RA	Dec	Radius (arcmin)	Equinox
<input type="text"/>	<input type="text"/>	<input type="text" value="10.0"/>	<input type="text" value="J2000"/>
International Ultraviolet Explorer			
Exp Time	Obs Date	Program ID	
<input type="text"/>	<input type="text"/>	<input type="text"/>	
Class		Camera:	<input type="checkbox"/> LWP <input type="checkbox"/> LWR <input type="checkbox"/> SWP
<input type="text" value="01 EARTH"/> <input type="text" value="02 MOON"/> <input type="text" value="03 PLANET"/> <input type="text" value="04 PLANETARY SATELLITE"/> <input type="text" value="05 MINOR PLANET"/>		Dispersion:	<input type="checkbox"/> Low <input type="checkbox"/> High
		Aperture:	<input type="checkbox"/> Large <input type="checkbox"/> Small
		Image ID	<input type="text"/>
<input type="button" value="SEARCH"/>	<input type="button" value="Clear form"/>	<input type="button" value="Reset to defaults"/>	Help...
Output Options			
Output columns	Sort output by:	Maximum number of hits	
<input type="checkbox"/> Mark <input checked="" type="checkbox"/> Image ID <input type="checkbox"/> Camera <input type="checkbox"/> Image <input type="checkbox"/> Object Name <input type="checkbox"/> Disp	1. <input type="text" value="Camera"/> <input type="checkbox"/> reverse 2. <input type="text" value="Image"/> <input type="checkbox"/> reverse 3. <input type="text" value="Disp"/> <input type="checkbox"/> reverse	<input type="text" value="100"/> Output Equinox <input type="text" value="J2000"/> <input type="checkbox"/> Show SQL query	

Figura 2.2: Janela de consulta no IUE.

URL: <http://archive.stsci.edu/starview.html>.

O sistema *Starview* é muito popular, sendo principalmente usado na visualização de campos de auxílio à observação (*finding chart*). Essa visualização é feita seja através de software específico instalado localmente ou via *web browser* na internet.

2.3.4 IPAC/NED

Acronismo: NASA/IPAC *Extragalactic Database*;

Base: IPAC/Caltech, US;

Mantenedor: NASA / Caltech, US;

Iniciadas operações: 1988;

Origem dos dados: Compilação de catálogos e publicações a respeito de objetos extragaláticos;

Retorna: Tabelas, arquivos FITS de imagens e espectros via FTP;

Domínio: Objetos extragaláticos, atributos fotométricos e espectros na literatura e catálogos;


URL: <http://nedwww.ipac.caltech.edu>.






O NED, como é conhecido, é o equivalente ao SIMBAD no que diz respeito à área de astrofísica extra-galáctica. Seu acesso é gratuito. Aqueles que trabalham com esse ramo da astronomia buscam no NED as referências dos objetos extragaláticos, parâmetros que o caracterizam e citações na bibliografia. Em meados do ano 2000 o NED conta com 2.8 milhões de objetos, 3.6 identificações por nome, 1.3 milhões de referências bibliográficas em 44 mil publicações, 3.0 milhões de medidas fotométricas, 144 mil medidas de *redshift* e 123 mil imagens disponíveis em formato FITS.

A consulta é feita através de **filtros**, isto é, escolha por nome, região no céu, *redshift*, ou citação. A recuperação se dá através de *e-mail* enviado ao usuário ou via FTP, após a execução da consulta. A Figura 2.3 mostra a página do NED e seus múltiplos filtros.

NASA/IPAC
EXTRAGALACTIC
DATABASE

▶ [New Features](#)
▶ [Archive](#)
▶ [Connectivity](#) **NEW**
▶ [Frames](#)



 OBJECTS	 DATA	 LITERATURE	 TOOLS	 INFO
By Name	Photometry & SEDs	References	Coordinate & Extinction Calculator NEW	FAQ
Near Name	Catalogs	Author Name	Velocity Calculator	Introduction
Near Position	Positions	Text Search NEW	FTP	Features
IAU Format	Redshifts	LEVELS NEW	Comment	News
By Refcode	Notes	Abstracts		Team
By Parameters	Images	Thesis Abstracts		Batch Jobs
Skyplot				Web Links NEW

Interface last updated: 5 July 2000 + 3.6 million names + 2.8 million objects + 1.3 million references to 44,000 papers + 3.0 million photometric measurements	Database last updated: 31 May 2000 + 144 thousand redshifts + 123 thousand FITS images + 45 thousand notes + 23 thousand abstracts
--	--

If your research benefits from the use of NED, we would appreciate the following acknowledgement in your paper: *This research has made use of the NASA/IPAC Extragalactic Database (NED) which is operated by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.*

Figura 2.3: Página do IPAC/NED.

São mostradas as diferentes formas de interação com o banco de dados.

2.3.5 ADF

Acronismo: *Astrophysics Data Facility*;

Base: GSFC (*Goddard Space Flight Center*), US;

Mantenedor: NASA, US;

Origem dos dados: Permite a consulta integrada das diferentes bases de dados administrados pela NASA;

Retorna: Tabelas, imagens e arquivos FITS via FTP;

Domínio: Altas energias, UV, ótico, infravermelho, submilimétrico, rádio;

URL: <http://adf.gsfc.nasa.gov/adf/adf.html>.

Esse sistema foi desenvolvido pela NASA para permitir o acesso integrado aos dados tornados públicos das suas missões observacionais. Entre os dados disponíveis estão os do ASCA (*Advanced Satellite for Cosmology and Astrophysics*), ADC, COBE, etc. Esse sistema faz parte dos recentes esforços para integrar os dados astronômicos, também conhecidos como **meta** bancos de dados.

As ferramentas disponíveis no ADF são o **AMASE** (*Astrophysics Multi-Spectral Archive Search Engine*), o **IMPRESS** (*Image Perimeters of Sky Surveys*) e **WISARD** (*Web Interface for Searching Archival Research Data*), este último fazendo parte do MAST (ver 2.3.9).

2.3.5.1 AMASE

Este banco de dados tem por base o Goddard Space Flight Center, US, e permite a pesquisa de forma hierárquica de objetos a partir do fornecimento do nome ou busca em uma região no céu, por classe ou atributo. O AMASE foi construído em bases de programação orientada ao objeto (POO) e bases de dados objeto - relacionais, características comuns ao SKICCOSMO. Essa foi a solução encontrada no desenvolvimento do AMASE para lidar com a completa heterogeneidade dos dados, mesmo no seio da administração das bases de dados da NASA. Visando ilustrar a estrutura do AMASE com suas características objeto-relacionais reproduzimos o seu diagrama na figura 2.4, extraída de Cheung *et al*, 1995 ([11]) Recomenda-se comparar a Figura 3.2 que mostra a estrutura do SKICCOSMO no que se refere aos objetos astronômicos com a Figura 2.4. Enquanto a Figura 2.4 serve

para ilustrar os relacionamentos entre objetos, a Figura 3.2 ilustra o SKICCOSMO que é tipicamente relacional. Nosso objetivo é fazer com que a estrutura do SKICCOSMO seja semelhante à do AMASE.

A Figura 2.5 mostra uma das janelas de consulta do AMASE.

2.3.5.2 IMPREeSS

O *Image Perimeter of Sky Surveys* é um sistema que possui uma interface gráfica que auxilia na procura de objetos de interesse do astrônomo. Ele oferece o serviço “*Catseye*”. O sistema é dividido em três janelas. Na primeira escolhe-se os parâmetros astrométricos do objeto: posição, época do equinócio e referências. Na segunda janela escolhe-se as missões espaciais observacionais disponíveis para procura, e a terceira, oferece um *finding chart* permitindo ao usuário verificar a localização do objeto ou objetos procurados. A Figura 2.6 mostra a janela a três *frames* do IMPReSS para consulta gráfica interativa.

2.3.6 Conjunto de ADC

Acronismo: *Astronomical Data Center*;

Base: ADC/NASA, US;

Mantenedor: NASA, US;

Iniciadas operações 1990;

Origem dos dados: Compilação de Catálogos e Artigos;

Retorna: Tabelas, imagens e arquivos FITS via FTP;

Domínio: Objetos astronômicos em geral em várias regiões do espectro ultravioleta, ótico e infravermelho;

URL: <http://adc.gsfc.nasa.gov/>.

Iniciou sua operação distribuindo vários catálogos de interesse astronômico em CD. Hoje, também é possível obter dados via *browser* de alguns dos mais populares catálogos tradicionais tais como o FK5, SAO2000, Henry Draper Catalog, etc. Consultas por publicações também são possíveis.

Pode-se visualizar gráficos de parâmetros escolhidos livremente pelo usuário a partir dos dados oferecidos pelo serviço.

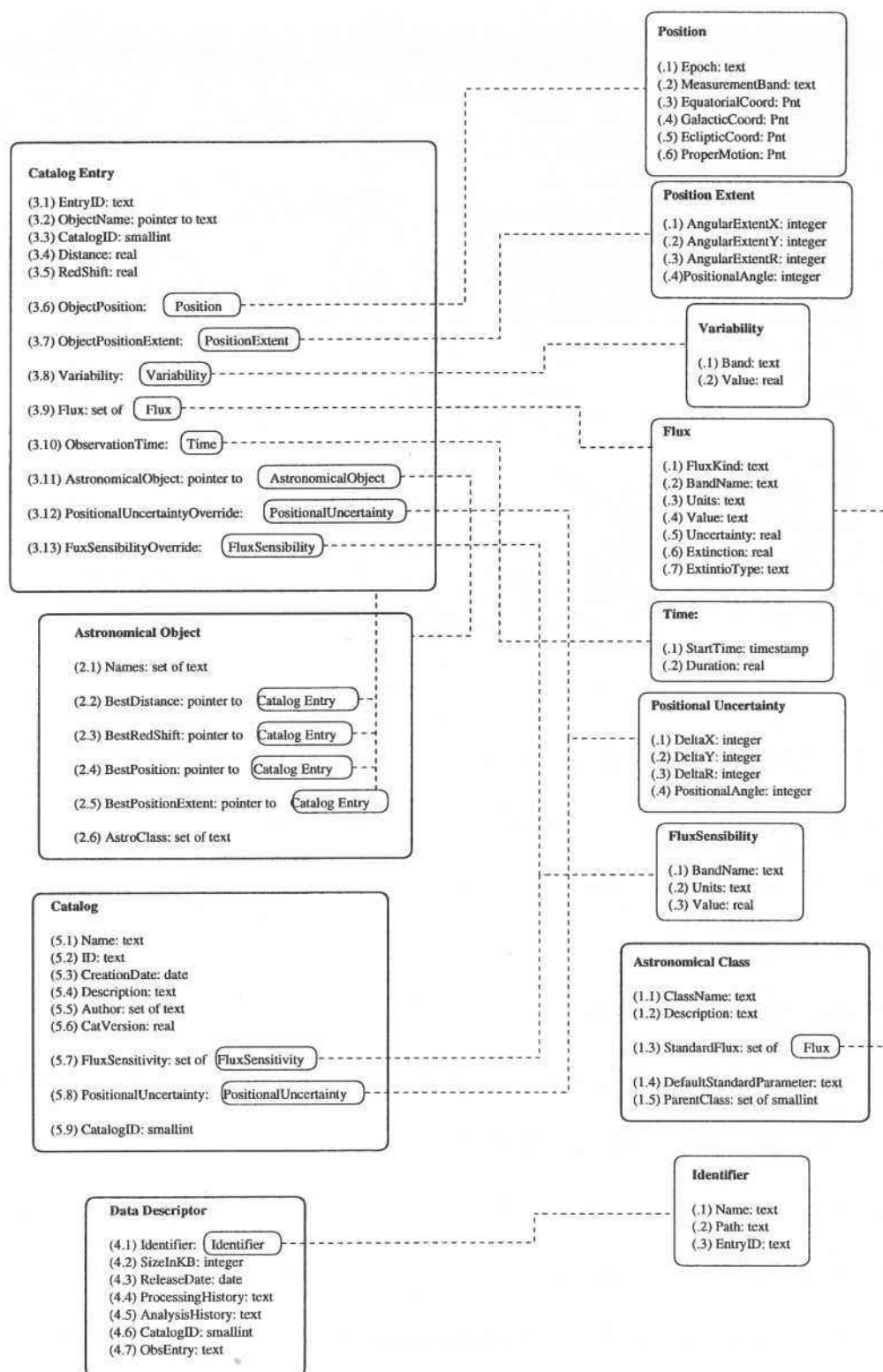


Figura 2.4: Estrutura do AMASE na parte dos objetos astronômicos (Cheung, 1995, [11])

Position Center		Radius (arcsec)	
HH MM SS.SS		180	
DD MM SS.S			

Target Category		Data Attributes	
Class	Type	Spectral Region	Mission
Galactic	Variable Emission	Radio	ALEXIS
HII Region	Extragalactic	IR	ASCA
Compact HII Region	Emission Lines	Optical	CGRO
Galaxy	Absorption Lines	UV	EUVE
Spiral Galaxy	Nuclear Activity	EUV	HST-PC

Figura 2.5: Uma das janelas de consulta do AMASE.

2.3.7 O Serviço HEASARC

Acronismo: *High Energy Astrophysics Science Archive Research Center*;

Base: LHEA (*Laboratory for High Energy Astrophysics*)/NASA) / SAO, US;

Mantenedor: NASA, US;

Iniciadas operações 1990;

Origem dos dados: Conjunto de missões observacionais em várias sondas;

Retorna: Arquivos FITS via FTP;

Domínio: Raios γ , X, extremo UV;

URL: <http://heasarc.gsfc.nasa.gov/>.

Esse serviço oferece dados obtidos em todas as missões da NASA de satélites na faixa ultra violeta até altas energias. Ao todo são cerca de 80 satélites. Os dados dessas missões podem ser correlacionados entre si e com missões da NASA em outras frequências, podendo-se obter informações sobre um objeto nos mais diferentes arquivos de sondas espaciais

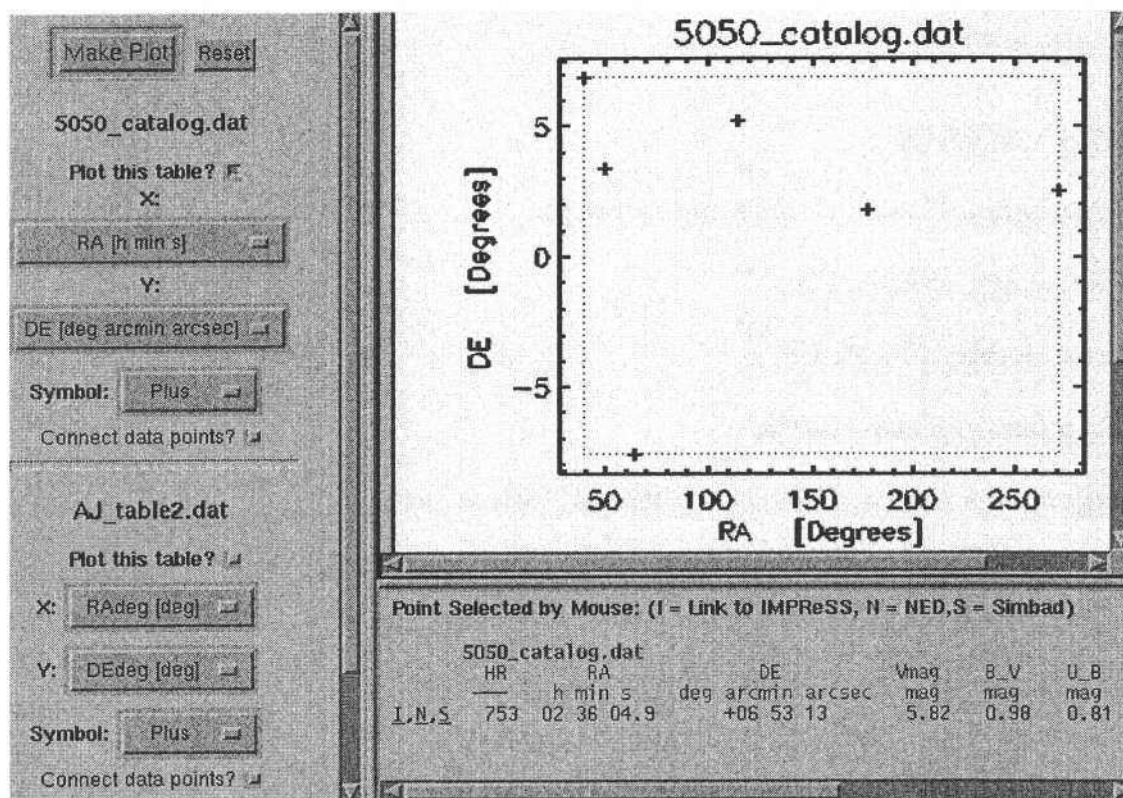


Figura 2.6: Consulta no IMPREeSS.

Escolhendo-se a tabela e as coordenadas a serem colocadas no gráfico (*frame* à esquerda), faz-se o gráfico (*frame* superior à direita). Clicando-se em um ponto nesse gráfico, obtêm-se seus atributos (*frame* abaixo à direita).

ou telescópios nessas faixas de frequência. Ferramentas gráficas permitem visualizar os resultados da pesquisa seja numa página contendo imagem **GIF**, ou *applet* JAVA. São oferecidos pacotes de programas que auxiliam a análise dos dados recuperados nas consultas. O HEASARC é equivalente, no espectro das altas energias, ao IRSA - no espectro infravermelho (ver 2.3.10) - e ao MAST, no espectro ótico / ultravioleta (ver 2.3.9). Todos são **meta** bancos de dados. Na Figura 2.7 vemos uma janela para consulta no HEASARC por palavra chave.

2.3.8 COBE

Acronismo: *Cosmic Background Explorer*;

Base: GSFC/NASA, US;

Mantenedor: NASA, US;

Iniciadas operações 1989;

Origem dos dados: Dados do satélite COBE (*Cosmic Background Explorer*);

Retorna: Arquivos FITS via FTP;

Domínio: Radiação cósmica de fundo ($\sim 3\text{K}$);

URL: <http://space.gsfc.nasa.gov/astro/cobe/>.

O satélite COBE foi desenvolvido pelo *Goddard Space Flight Center* (GSFC/NASA) para medir a radiação infravermelha difusa e a radiação em microondas do universo primordial. Lançado em 18/11/89, levou três instrumentos: *Far Infrared Absolute Spectrophotometer* (FIRAS) para comparar a radiação cósmica de fundo com a radiação de um corpo negro, um *Differential Microwave Radiometer* (DMR) para o mapeamento da radiação cósmica de fundo e o *Diffuse Infrared Background Experiment* (DIRBE) para procurar localizar a radiação cósmica infravermelha de fundo. A precisão da medida da radiação em microondas foi de 0.005%. A anisotropia encontrada na medida foi de uma parte em 100.000. Foi detectada radiação cósmica infravermelha difusa pelo DIRBE nas bandas de 140 e 240 μm . A faixa de procura foi na região de 1.25 μm a 240 μm .

O centro de distribuição dos dados do COBE, localizado no GSFC/NASA oferece programas desenvolvidos em **IDL** (*Interactive Data Language*, [50]) para auxiliar o usuário a manipular os dados obtidos das consultas à base de dados do COBE.

Search for resources by keywords: ([Help](#))

This form allows you to create a custom list of data services to be queried. For example, one can search for data services that provide **infrared** data, for services that provide **images**, or catalogs specializing in **galaxies**. The [search help page](#) has more examples.

Bandpass:	Source type:	Data available:
<input type="checkbox"/> Radio <input type="checkbox"/> Microwave <input type="checkbox"/> Infrared <input type="checkbox"/> Optical <input type="checkbox"/> Ultraviolet <input type="checkbox"/> EUV <input type="checkbox"/> X-ray <input type="checkbox"/> Gamma-ray	<input type="checkbox"/> Survey <input type="checkbox"/> Observations <input type="checkbox"/> Derived	<input type="checkbox"/> Catalog <input type="checkbox"/> Image <input type="checkbox"/> Spectra <input type="checkbox"/> Time-series

Selections will be OR-ed within columns and AND-ed between columns.
 For example: (optical or infrared) and derived and (catalog or image)

Additional keywords: (AND-ed by default)

Example: variab* not (CV or cataclysmic)

NOTE: Do not enter object names in this box; the keywords are searched for in the descriptions of the databases, so search for general terms like classes of objects, not specific targets.

Figura 2.7: A janela de consulta por palavra chave do HEASARC

2.3.9 Conjunto MAST

Acronismo: *Multimission Archive at STScI*;

Base: STScI, US;

Mantenedor: NASA, US;

Origem dos dados: Integração das missões e experimentos da NASA;

Retorna: Arquivos FITS via FTP;

Domínio: Ótico, UV, infravermelho, Radio, X;

URL: <http://archive.stsci.edu/mast.html>.

Serviço do Instituto do Telescópio Espacial, o MAST se propõe a um meta banco de dados e integra todas as missões espaciais de fim astronômico da NASA. Oferece também um serviço de correlação cruzada entre os diferentes catálogos de resultado das missões, inclusive o HIPPARCOS e o Catálogo de Aglomerado de Abell, além de permitir que o usuário compare seu próprio catálogo com o que tem registrado no MAST. Como resultado da pesquisa, o usuário pode fazer um *download* do resultado de suas consultas. A Figura 2.8 mostra a página do MAST que oferece as alternativas para consultas.

2.3.10 IRSA

Acronismo: *Infrared Science Archives*;

Base: IPAC (*Infrared Processing and Analysis Center*)/Caltech, US;

Mantenedor: Caltech, US;

Origem dos dados: Dados de sondas na região do infravermelho e submilimétrico. SIRT (*Space Infrared Telescope Facility*), ISO (*Infrared Satellite Observer*), MSX (*Micro Space Experiment*), IRAS (*Infrared Astronomical Satellite*) e 2MASS (*Two Micron All Sky Survey*);

Retorna: Tabelas, arquivos FITS via FTP;

Domínio: infravermelho próximo e submilimétrico;

MAST
Multimission Archive at STScI

- MAST Introduction
- MAST Data Characteristics
- Cross-Correlation Search
- Quick Search
- Data Analysis Software
- Prepared Datasets
- FAQ
- Related Sites
- Acknowledgments
- Contacts/Help
- Index Search

	Images	Spectra	Other
<u>X-ray</u>	<input type="checkbox"/>	<input type="checkbox"/>	
<u>Extreme UV</u>	<input type="checkbox"/>	<input type="checkbox"/>	
<u>Far UV</u>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<u>Near UV</u>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<u>Optical</u>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<u>Near IR</u>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<u>Radio</u>	<input type="checkbox"/>		

[Help](#)

Select desired data parameters and then click the Submit Form button to find information on available datasets.

[HST | FUSE | IUE | EUVE | Copernicus | ASTRO | ORFEUS | DSS | FIRST | ROSAT | MAST]

Figura 2.8: Página de introdução do MAST e as alternativas de consulta.

Enter Column Constraints below using a SQL *WHERE* clause (ended by an optional *ORDER BY* statement).

Alternatively, you may enter simple range constraints using

[Query by Table \(QBT\)](#)

```

|g| at > +85.0 and k_m
< 11.0 and cc_flg = '000'
  
```

[Column documentation for the 2MASS Second Incremental Release Point Source Catalog \(PSC\)](#)
[Example queries for the 2MASS Second Incremental Release Point Source Catalog \(PSC\)](#)
[Help for SQL WHERE Clause Entry](#)
[String Searches](#)
[List Searches](#)
[Math](#)

Figura 2.9: Consulta no IRSA.

Área de texto para construir a consulta do lado *where* do SQL.

URL: <http://irsa.ipac.caltech.edu>.

Esse sistema, classificado como um **meta** banco de dados, permite a recuperação de dados na faixa do infra vermelho e submilimétrico de missões espaciais, incluindo o IRAS, ISO, etc. A recuperação dos dados é feita através de filtros assim como uma correlação com tabelas fornecidas pelo usuário para cruzar os objetos registrados no sistema a aqueles fornecidos pelo usuário.

Um das possibilidades de consulta no IRSA é construir, numa área de texto, a instrução correspondendo às condições de procura, escrevendo-se a parte que segue o *where* da instrução em SQL. A Figura 2.9 mostra a janela contendo esse tipo de consulta. Essa possibilidade se aproxima da proposta do SKICCSMO (ver em 3.2). A diferença é que o SKICCSMO oferece uma liberdade maior para construir os atributos de saída além de ser possível construir as condições de uma forma um pouco mais amigável. No IRSA, se o usuário quiser poderão ser estabelecidas suas condições através da chamada *Query By Table* (QBT), onde o usuário preenche valores dos limites mínimo e máximo para cada atributo desejado.

O usuário pode escolher o tipo de saída que ele quer para os resultados, se tabela, contagem de linhas ou estatística dos atributos. E pode escolher o catálogo em que ele quer fazer a consulta e o formato da tabela.

2.3.11 DaBIAS/LNA

Acronismo: *Data Bank Information Acquisition System;*

Base: LNA(Laboratório Nacional de Astrofísica)/MST, BR;

Mantenedor: LNA/MST,BR;

Iniciadas operações 1999;

Origem dos dados: Programas de observação no LNA, acervo público dos Telescópios de 1.6m B&C, 0.6 m Zeiss, e 0.6m B&C;

Retorna: Tabelas, arquivos FITS via FTP, *media* magnética via correio;

Domínio: Ótico, infravermelho próximo;

URL: <http://www.lna.br/~databank/databank.html>.

Essa é uma iniciativa de disponibilizar não somente à comunidade brasileira, como também internacional, o acervo de 20 anos de observações nos telescópios instalados no Pico dos Dias em Brasópolis, MG, do LNA, Laboratório Nacional de Astrofísica. Através do sistema DaBias, acessa-se uma tabela contendo as descrições das observações. Na medida que são disponibilizadas, um *link* aponta para a obtenção dos arquivos. O usuário pode, assim, saber que objetos tem à disposição e poderá fazer o *download* do arquivo FITS referente à(s) observação(ões) dos objetos. Na Figura 2.10 podemos ver a janela para construir uma consulta por coordenadas ou por objetos do DaBIAS/LNA.

2.3.12 GSC

Acronismo: *Guide Star Catalog;*

Base: STScI, US;

Mantenedor: STScI, US;

OPD Data Bank

Data bank search form

You may search the data bank for particular objects by name or by coordinates (+ search radius). In the first case, four different alias names for a given object) may be defined.

Search by name:	Search by coordinates:
1. Object <input type="text"/>	<input type="text"/> Right ascension (hh:mm:ss)
2. Object <input type="text"/>	<input type="text"/> Declination ([-]dd:mm:ss)
3. Object <input type="text"/>	<input type="text"/> 2000 Equinox
4. Object <input type="text"/>	<input type="text"/> 15 Search radius (arcmin)(0..60)

You may restrict the search to the data of a particular telescope:

You may restrict the search to a certain range in time:

start of range	<input type="text" value="current year"/>	<input type="text" value="January"/>	<input type="text" value="1"/>
end of range	<input type="text" value="current year"/>	<input type="text" value="December"/>	<input type="text" value="31"/>

Submit request:

Figura 2.10: Janela para consulta por objetos ou coordenadas do DaBIAS/LNA.

Iniciadas operações 1990;

Origem dos dados: *Survey*, placas digitalizadas de câmara Schmidt, do UK Schmidt Telescope e Oschin Schmidt Telescope;

Retorna: Tabelas;

Domínio: Ótico;

URL: <http://www-gsss.stsci.edu/gsc/gsc.html>.

O *Guide Star Catalog* originalmente concebido para servir de calibração para o Telescópio Espacial é uma referência para os mais diversos campos da astronomia, com base terrestre também. Com seus quase 20 milhões de objetos catalogados permite uma pesquisa detalhada e razoavelmente profunda, sem, no entanto chegar aos limites do POSS-II. Tem sido freqüentemente atualizado e melhorado. É disponível também em CD-ROM.

2.4 Características dos Banco de Dados na Astronomia

A propriedade comum dos bancos de dados aqui apresentados, sejam eles classificados como *archives*, ou como **meta**⁶ bancos de dados, é a limitação na consulta. O que se tem, via de regra, é a possibilidade de seleção através de alguma propriedade: nome de objeto, região nas vizinhanças de alguma coordenada, citações na bibliografia, classe de objetos, etc. Não há a possibilidade de escolher objetos que possuam como propriedade, por exemplo, "o quadrado de sua magnitude é a metade de seu raio efetivo". Pode ser que essa propriedade não faça qualquer sentido, atualmente. No entanto não se sabe se ela o fará no futuro. Em outras palavras, não há liberdade do usuário escolher as propriedades de suas consultas. O SKICCOSMO apresenta-se como uma alternativa a mais a ser utilizada pela comunidade. O que o distingue do que foi apresentado até aqui, além de disponibilizar *Web-wide* os dados do POSS-II, é permitir que o astrônomo construa suas consultas estabelecendo relações complexas entre os atributos dos objetos, sem qualquer limitação, salvo as impostas pelo próprio DBMS⁷.

⁶Serviços integrando dados de vários catálogos diferentes, espalhados por diversos centros de distribuição.

⁷Não é possível estabelecer que o brilho superficial de um objeto seja a raiz quadrada de sua classificação. O brilho superficial é armazenado em uma variável do tipo *float*, enquanto que a classificação é do tipo *char*. O DBMS não permite operações aritméticas com variáveis do tipo *char*.

Nome	URL	Domínio	Retorno
SIMBAD	http://simbad.u-strasbg.fr/Simbad	Objetos galáticos extragaláticos posições citações	Tabelas
IUE	http://archive.stsci.edu/iue/	UV (1150Å - 3200Å)	Tabelas Imagens Espectros
STARVIEW	http://archive.stsci.edu/starview.html	Ótico (3400Å - 7500Å) Infravermelho próximo	Imagens Find Charts Espectros
IPAC/NED	http://nedwww.ipac.caltech.edu	Objetos extragaláticos Fotometria Espectros	Tabelas FITS: imagens Espectros
ADF	http://adf.gsfc.nasa.gov/adf/adf.html	Altas energias, UV Ótico, Infravermelho Submilimétrico Rádio	Tabelas FITS: imagens
ADC	http://adc.gsfc.nasa.gov/	Ultravioleta Ótico Infravermelho	Tabelas FITS: imagens Espectros
HEASARC	http://heasarc.gsfc.nasa.gov/	Raios γ Raios X Ultravioleta extremo	FITS: imagens
COBE	http://space.gsfc.nasa.gov/astro/cobe/	Micro ondas (3K) Infravermelho	FITS: imagens Espectros
MAST	http://archive.stsci.edu/mast.html	Ótico Ultravioleta, Raios X Infravermelho, Rádio	FITS: imagens Espectros
IRSA	http://irsa.ipac.caltech.edu	Infravermelho próximo Submilimétrico	Tabelas FITS: imagens Espectros
DaBIAS/LNA	http://www.lno.br/~databank/databank.html	Ótico Infravermelho próximo	FITS: imagens Espectros
GSC	http://www-gsss.stsci.edu/gsc/gsc.html	Ótico	Tabelas

Tabela 2.1: Resumo das características de alguns bancos de dados na internet.

Uma outra situação caracteriza os bancos de dados na astronomia: a dificuldade de homogeneização. A saber: um fóton é irradiado por um objeto emissor no espaço, cruza o meio intergalático (se for o caso), o meio interestelar e o interplanetário para atingir a atmosfera de nosso planeta. Viajará até atingir o espelho ou lente do telescópio, atravessará o dispositivo ótico em utilização e atingirá a cavidade do detector onde vai se juntar aos outros fótons e excitar um contador de acordo com sua eficiência quântica. Todos esses dispositivos de transmissão e reflexão absorvem a luz segundo leis de espalhamento e polarização que modificam significativamente a onda eletromagnética quanto a sua disposição no espaço e distribuição em comprimento de onda. Além disso, essas mudanças têm maior ou menor dependência no tempo. A dependência funcional com o tempo é muitas vezes de difícil modelagem e até impossível de ser especificada em termos analíticos ou extrapoláveis. A medida da energia luminosa de um objeto celeste, portanto, é um "instantâneo", que rigorosamente não possui relação necessária com medidas de outros objetos ou até em relação a ele próprio, em outro momento.

Um equipamento de medida de posição é tão preciso quanto possível, digamos ao limite quântico. Nessas condições, a posição aparente dos astros no céu depende essencialmente da rotação da terra, do estado refringente da atmosfera e, em certos casos, da atividade vulcânica recente.

O maior obstáculo para a construção de um catálogo unificado é a falta de homogeneidade na sua construção. Menos por culpa de seus organizadores do que pela própria natureza do universo de estudo da astronomia. É fato que não há qualquer ponto fixo de referência na astronomia, seja no que diz respeito ao posicionamento quanto ao fluxo radiante nas diversas bandas de comprimento de onda. Essa característica determina que um único objeto celeste possui todos os seus atributos rigorosamente variáveis com o tempo, observatório, tecnologia e equipamento utilizados.

Existe, porém, o preceito da aproximação, que a experiência demonstra funcionar a contento. Essa situação faz da astronomia observacional uma ciência eminentemente empírica e estatística. A construção de um banco de dados voltado para a astronomia será marcada por essa característica, como veremos no próximo Capítulo.

Capítulo 3

Transferindo os dados

Nosso trabalho centra-se na proposta de um novo banco de dados que batizamos SKIC-COSMO. Introduziremos, nesse capítulo, a sua estrutura, não sem antes retomarmos o SKICAT que o antecedeu.

3.1 O SKICAT

Vamos analisar aqui a estrutura das tabelas e aplicativos do banco de dados SKICAT (Fayyad *et al*, 1993, [20]). Apresentamos, nesse capítulo, a descrição de suas diferentes entidades e uma análise crítica das implementações feitas. Mostraremos que o SKICAT não se conforma aos preceitos das formas normais de Codd (ver em 1.4.2.9).

3.1.1 A Estrutura dos Catálogos do SKICAT

3.1.1.1 Os Catálogos registrados e os Catálogos *on-line*

Quando montou a arquitetura do SKICAT, Weir, 1994 ([66]) contava com poucos recursos de disco. A massa de dados a ser armazenada era superior à quantidade de espaço em disco disponível. A solução encontrada foi criar os conceitos de catálogo registrado e catálogo *on-line*. Cada catálogo representa o conjunto da informação obtida de um *frame*, isto é, uma imagem digitalizada de uma placa fotográfica ou uma imagem CCD. Os dados são passados por um processo baseado no FOCAS[62], associado às ferramentas de árvore de decisão (Weir, 1994, [66]) que vai identificar e classificar objetos, sejam estrelas, galáxias e outros (Valdes *et al*, [63]). Se o espaço em disco estiver exíguo e os dados tratados não forem necessários ao trabalho de pesquisa, pode-se fazer um *backup* dos dados e apagá-los

do banco de dados. Restam algumas informações sobre o campo tratado para permitir ao usuário uma decisão futura. Nesse caso diz-se que o catálogo está registrado. Se, pelo contrário, houver espaço suficiente em disco e/ou os dados de determinado catálogo forem essenciais para a análise científica no momento, os dados são mantidos em disco, e o catálogo é dito estar *on-line*.

3.1.1.2 As Tabelas de Características (*Features*)

Entre todas as tabelas do SKICAT, as que fornecem os dados finais para aplicação astronômica são as tabelas de características (*features*). São, no jargão da ciência de banco de dados, aquelas tabelas que possibilitam as aplicações de **apoio à decisão**, pois são elas que possuem os dados finais para a informação científica. Outras tabelas são importantes para uma análise complementar dos dados.

As tabelas de características contêm informações de cada objeto no banco de dados. Essas informações dizem respeito a todas as grandezas obtidas pelo FOCAS, com base nas informações astrométricas e fotométricas advindas da exposição e/ou digitalização no observatório de origem. Como consequência, cada linha na tabela de características é própria a cada objeto, mas não exclusiva, e independente da técnica usada na observação, seja em placas fotográficas, seja em imagens CCD.

Cada objeto nessa tabela possui 60 atributos, ou seja, essas tabelas possuem 60 colunas e tantas linhas quantos forem os objetos identificados e classificados pelo FOCAS. Possuem informações como ascensão reta e declinação, magnitude, medidas de brilho superficial e têm como chaves primárias as posições XC e YC do objeto na placa ou imagem CCD.

3.1.1.3 As Tabelas de Cabeçalho (*Header*)

Cada placa ou imagem CCD introduzida no SKICAT merece uma entrada na Tabela de Cabeçalho (*Header Table*) que contém todas as informações necessárias à obtenção dos dados que estão armazenados nas tabelas de características. Como os atributos dos objetos nessa tabela dependem da técnica observacional, existe uma tabela para placas fotográficas e outra para imagens CCD. Cada objeto nessa tabela possui o nome da tabela de características gerada a partir dela, um identificador (CatID), data da observação, da digitalização se for o caso, da redução e gravação no banco de dados, além de informações obtidas no FOCAS, na digitalização, e no cabeçalho do arquivo FITS original. A tabela de cabeçalho para placas possui 178 colunas e para imagens CCD, 90. Cada uma dessas tabelas possui apenas uma linha.

3.1.1.4 A Tabela das Características *Matched* (*MatchedFeatures Table*)

O SKICAT reserva uma tabela especial para o resultado do procedimento de *matching*, ou seja, a identificação dos objetos equivalentes entre dois campos. Ela possui um conjunto de informações mínimas, podendo ser estendida com outras informações disponíveis nos catálogos dependendo da necessidade do usuário. Como informação mínima destacam-se um identificador de objeto que só aparece nesse estágio do banco de dados e não constitui uma chave primária, ou sequer faz parte dela, ao contrário do que era de se esperar. Outros atributos são repetidos das tabelas originais, criando um grave problema para a manutenção da integridade nesse banco de dados, além de duplicar desnecessariamente o espaço requerido para o armazenamento dos dados.

Todas essas características, em particular, devem ser revistas quando da definição de uma outra estrutura de banco de dados para cosmologia (vide 3.2.1).

3.1.1.5 A Tabela dos Catálogos *Matched* (*MatchedCatalogs*)

A exemplo da tabela de catálogos simples, esta tabela de eventos¹ serve para identificar os procedimentos usados para *match* dois catálogos. Possui apenas duas colunas, uma contendo o CatId e outra, a data do processamento.

3.1.1.6 A Tabela de Contagem de *Matching* (*MatchCount*)

Contém duas colunas, uma com a identificação do objeto, e a outra com o número de vezes que esse objeto aparece nos diferentes catálogos submetidos a esse processo.

3.1.1.7 Tabela de Objetos (*Objects*)

Tabela gerada a partir de uma consulta dos usuários. Normalmente esse é um tipo de tabela temporária usada para certos propósitos e depois é destruída.

3.1.1.8 A Tabela dos Catálogos (*Catalogs*)

Cada catálogo é constituído de uma tabela de cabeçalho e uma tabela de características. Os catálogos inseridos no SKICAT são registrados na Tabela de Catálogos. Essa tabela possui seis colunas especificando o identificador do catálogo (CatId - chave primária), o nome da tabela de características, o tipo (placa ou CCD), a data do primeiro registro e

¹**Entidade do tipo Eventos:** Reportar-se à discussão em 1.4.2.3, particularmente à figura 1.7.

do último registro, além da banda ou filtro da exposição. Nesse particular, é interessante notar que, cada vez que uma mesma placa é reprocessada e re-inscrita no banco de dados, haverá um identificador diferente.

3.1.1.9 A Tabela “*CatVersions*”

Essa tabela contém as versões de cada catálogo registrado no SKICAT. Possui seis colunas. Nessa tabela é possível determinar se um catálogo está “*on-line*”, isto é, está armazenado em disco, ou apenas registrado, mantido em fita magnética. Entre seus atributos estão: *CatId*, *Data*, Número da fita, Nome da fita e uma *flag* para *on-line* ou *off-line*. Suas chaves primárias são: *CatId* e *Data*.

3.1.1.10 A Tabela “*CatTypes*”

Descreve os tipos de catálogos existentes no SKICAT e possui duas colunas e duas linhas. A chave primária é a coluna *Type*. Os tipos podem ser: placa fotográfica (*Plate*) e CCD.

3.1.1.11 A Tabela “*Placas*” (*Plates*) e CCDs

Retém informações sobre os catálogos *on-line*, ou apenas registrados. Cada uma dessas tabelas possui todos os atributos das tabelas de Cabeçalho, seja de imagens digitalizados do POSS-II (*Placas*) seja de imagens CCD. Ela possui os atributos de todas as tabelas de Cabeçalho reunidas. Os atributos dos cabeçalhos das imagens de placas do POSS-II são armazenados na tabela *Placas* enquanto que os atributos dos cabeçalhos das imagens CCD são armazenados na tabela *CCDs*.

Vê-se que o procedimento de reunir os atributos das diferentes placas e imagens CCD em mais de uma tabela produz redundâncias. Além do desperdício de memória, essa situação abre espaço para inconsistências de informação no banco de dados. A mudança inadvertida dos valores de atributos em uma tabela, sem ser seguida da mudança dos mesmos atributos na outra tabela, trará contradição de informações a um usuário que não foi informado dessa mudança.

As tabelas *Placas* e *CCDs* se aproximam mais da solução para um banco de dados normalizado do que as tabelas que fazem redundância. Estas, como já vimos, possuem apenas uma linha, e os atributos que deveriam ser suas chaves primárias, na verdade, compõem o nome dessas tabelas². Já as tabelas *Placas* e *CCDs* compõem o conjunto dos

²A tabela “*f823_Header*” possui em seu próprio nome a referência à banda (f) seguida do número do

atributos de uma classe de objetos, os **cabeçalhos**³.

3.1.1.12 A Tabela “*MatchProc*”

Essa tabela possui sete colunas e apenas uma linha. É usada durante o processo de *matching*. É, portanto, uma tabela temporária, de trabalho.

3.1.1.13 A Tabela “*MatchColumn*”

Possui três colunas. Descreve cada atributo não obrigatório na tabela *matched*.

3.1.1.14 A Operação com o SKICAT

Antes de passarmos a analisar a estrutura do banco de dados SKICAT, observamos que existem várias redundâncias. As tabelas *Plates* e *CCD* acumulam informações com as tabelas *Headers*. Além disso, mesmo que o usuário opte por não introduzir mais colunas na tabela *MatchedFeatures*, esta acumula informações com a tabela *Features*, o que representa não somente a ocupação desnecessária de espaço, mas, principalmente, representa uma ameaça à integridade dos dados no interior do banco de dados.

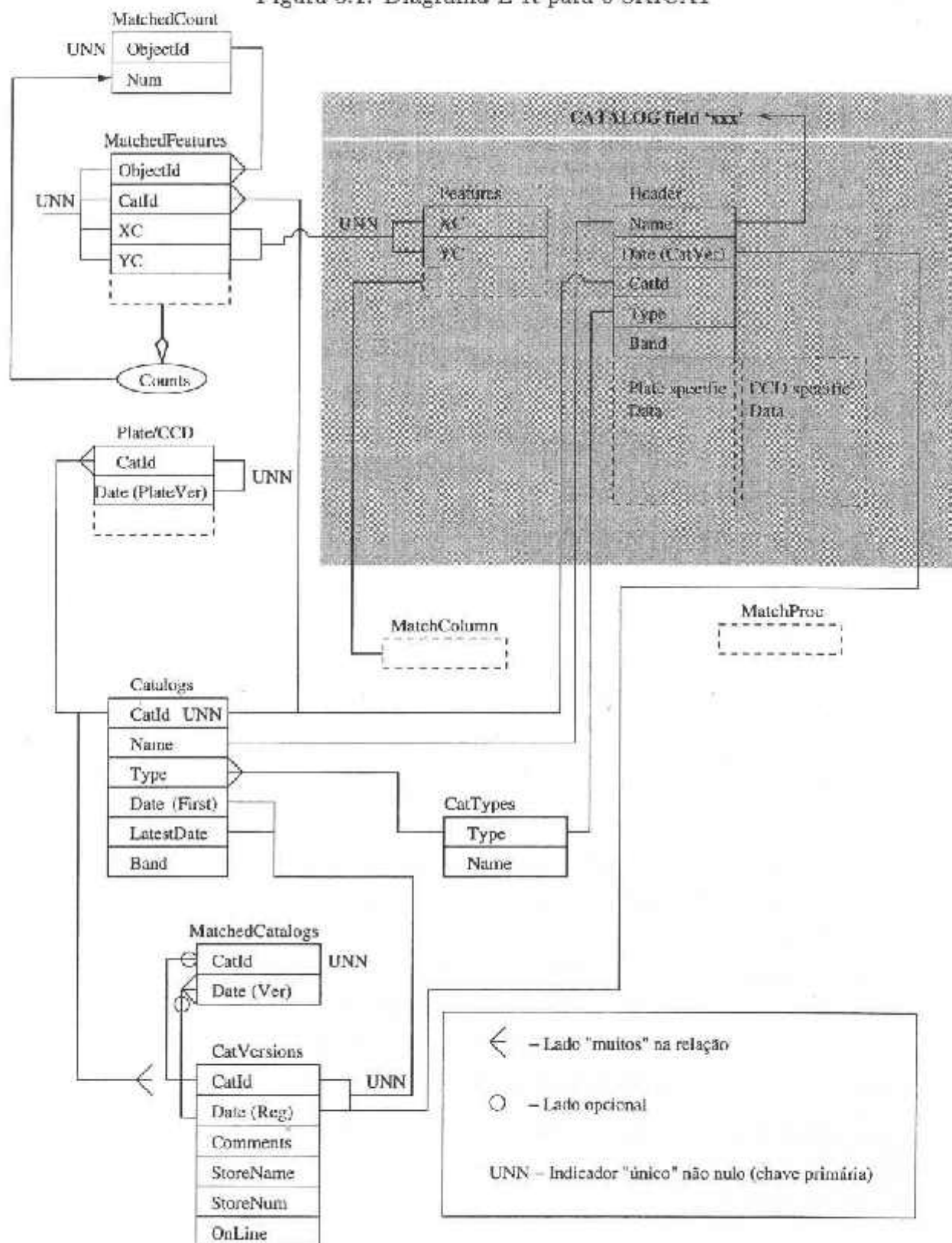
A Figura 3.1 mostra o diagrama E-R (Entidade - Relacionamento)⁴ aproximado para o SKICAT. Esse diagrama é um instrumento interessante que nos permite verificar a arquitetura do banco de dados e a conformidade de sua estrutura aos preceitos de relacionamento estabelecidos por Codd. As colunas que não entram no relacionamento do banco de dados são representadas por caixas de linhas quebradas. Além disso, foi introduzida uma “caixa” em tonalidade cinza indicando o que o autor define por “catálogo”. Esse diagrama não faz parte do padrão do diagrama E-R porque esse conceito não existe na teoria de banco de dados relacional. Uma outra figura introduzida no diagrama, que não consta do padrão, é a seta apontando para o “botão” *Counts*, na saída da tabela *MatchedFeatures*. Essa representação indica que a coluna *Num*, da tabela *MatchedCount*, é um valor derivado de uma operação sobre a tabela *MatchedFeatures*. Em outras palavras, *Num* é o resultado de uma consulta (*query*) e não devia fazer parte das tabelas originais do banco de dados.

campo (823). Rigorosamente, esses atributos deveriam compor a chave primária de uma entidade contendo os mesmos atributos.

³Ver mais a frente em 3.2.

⁴Ver em 1.4.2.3.

Figura 3.1: Diagrama E-R para o SKICAT



O relacionamento entre as tabelas *Catalogs* e *CatVersions* é de “um para muitos”, na chave *CatId*. No entanto, o autor, Weir (1994), construiu um relacionamento de “um para um”, na mesma chave entre as tabelas *CatVersions* e *MatchedCatalogs*, sendo a relação opcional do lado *MatchedCatalogs*. Seria mais consistente fazer a relação *Catalogs - MatchedCatalogs* com o lado “muitos” e opcional com *MatchedCatalogs*. Por outro lado, existem duas colunas na tabela *catalogs* que não são derivadas de consulta, mas são, de certa forma, casos particulares da coluna *Date (Reg)*, da tabela *CatVersions*. Trata-se de mais uma característica não usual em banco de dados relacional por causa da dificuldade da manutenção da integridade referencial dos dados.

A tabela *Header* dentro de *Catalog* não possui chave primária. Quatro chaves estrangeiras são inseridas: *Name* e *CatId*, em relação com a tabela *Catalogs*; *Date(CatVer)*, em relação com a tabela *CatVersion*, e *Type*, em relação com a tabela *CatTypes*. Não há cardinalidade na tabela de *Header* porque ela possui apenas uma linha. Existem muitas tabelas *Header*. Se fôssemos pensar em estruturar o SKICAT segundo as condições de normalização de Codd, não haveria sentido em construir uma tabela, com chaves estrangeiras, possuindo apenas uma linha. Portanto, o que seria o lado “muitos” traduz-se nas diversas tabelas *Header*, bem como *Features*. A tabela *Features* possui uma chave primária composta das colunas *XC* e *YC*. A tabela *MatchedFeatures* mantém relacionamento com esta tabela via *XC* e *YC*. A cardinalidade desse relacionamento é de 1 : 1.

As tabelas *Features* e *MatchedFeatures* mantêm as colunas *XC* e *YC* além de *RA* (ascensão reta) e *DEC* (declinação). No entanto, as duas últimas são obtidas a partir das duas primeiras, o que vai contra a primeira forma normal. Outras violações das formas normais podem ser encontradas: o atributo *Name* de *Catalogs* contém as relações *Features*, e como dito anteriormente, o atributo *Num* da tabela *MatchCount* é resultado de uma consulta em *MatchedFeatures* (segunda forma normal), há redundância de atributos entre *Features* e *MatchedFeatures* e *MatchColumn*, outra redundância se encontra com *Header* e *Plate* ou *CCD*.

Pode-se definir uma tabela primária⁵, do ponto de vista de banco de dados relacional, no SKICAT: trata-se da tabela *Catalogs*. Contudo, essa tabela não é suficiente para obter-se a integridade dos dados. É, então, necessário lançar mão, ou da tabela *Features*, ou de *MatchedFeatures*. A primeira pode ou não estar disponível *on-line* no banco de dados e a segunda depende da ação do usuário em obter os atributos que lhe interessam operando sobre a tabela *Features*. Do ponto de vista do conceito moderno de banco de dados,

⁵Denominamos tabela primária aquela que não possui chave estrangeira.

o SKICAT não possui granularidade imediata pois o usuário não pode obter seus dados, nem construir combinações entre eles, porque depende da intervenção do administrador, e deve aguardar a carga dos dados caso estes não estejam *on-line*, ou, ainda, aguardar que seja procedido o *matching* entre os catálogos que serão usados. Como veremos a seguir (Seção 3.1.2), o SKICAT possui escalabilidade precária⁶ pois, por construção, exige que o usuário execute tarefas de administrador, estando os níveis "gerente" e "cliente" acumulados⁷. Finalmente, é insuficiente em modularidade, pois obstrui a inclusão de informações contendo novos conceitos (paradigmas) no sentido de enriquecer as informações científicas ali armazenadas. Tudo isso decorre da precariedade da normalização do conjunto do banco de dados. Como consequência, o SKICAT é um sistema que proporciona operação lenta e de possibilidades limitadas, além de, como vimos acima, possuir várias falhas no que diz respeito à garantia de integridade. Apesar de pioneiro, perdeu funcionalidade frente aos conceitos mais recentes.

Uma vez analisada a estrutura do banco de dados SKICAT, vamos, a seguir, analisá-lo sob o ponto de vista das aplicações.

3.1.2 Os Aplicativos disponíveis no SKICAT

3.1.2.1 "Stored Procedures"

Valendo-se da facilidade do Sybase, o SKICAT possui três *Stored Procedures* que são programas inseridos no interior do banco de dados com o fim de tornar as operações mais rápidas. São elas: "GETINDEX", "SHOWTABLES" e "WHO". Uma quarta *stored procedure* foi criada para auxiliar a administração dos discos dentro do Sybase e se aplica somente no momento em que o administrador do banco de dados substitui ou introduz um disco novo ao sistema. GETINDEX se destina a mostrar os índices definidos no SKICAT, SHOWTABLES serve para visualizar as tabelas existentes e, finalmente, WHO mostra os usuários que estão utilizando o SKICAT.

3.1.2.2 Utilitários

Eis alguns dos utilitários do SKICAT à disposição do usuário para interagir com o SKICAT:

- **xautoplate**: interface X11 do *script* autoplate. Trata-se de um programa que sintetiza todos os procedimentos de tratamento de imagem, quer seja esta proveniente

⁶Ou seja, não é destinado a servir grande quantidade de usuários.

⁷Os níveis gerente e cliente são apresentados em 1.5.

de uma placa fotográfica digitalizada, ou de um campo CCD. Autoplate utiliza-se de parâmetros previamente introduzidos pelo usuário e executa o FOCAS para a identificação, classificação e calibração dos objetos;

- **xquery**: interface X11 para construção de consultas em tabelas. Através de ferramentas do tipo “botões”, o usuário pode construir uma consulta numa tabela selecionada. É de fácil manejo e a apresentação é simples e clara. Não há facilidade para consultas com junção interna, isto é, aproveitando-se de relacionamentos entre tabelas;
- **xquery_mc**: interface X11 que permite executar consultas em catálogos *matched* (*matched catalogs*). Através dessa ferramenta pode-se fazer consultas no sentido de obter informações de objetos comuns a campos observados em diferentes bandas fotométricas com placas fotográficas ou imagens CCD. Os catálogos devem ter sido previamente *matched* através das ferramentas apropriadas. É, em última instância, a ferramenta que permite obter as informações científicas desejadas;
- **xtable**: interface X11 para o `query_table` - utilitário que permite ao usuário aplicar filtros aos dados, modificar, destruir e criar tabelas dentro do SKICAT. Trata-se de um utilitário que, em princípio, seria direcionado para a administração do SKICAT. No entanto o usuário pode utilizá-lo sem restrição. Essa é uma propriedade do SKICAT: não distingue o usuário do administrador;

Após esta análise das contribuições e problemas com o SKICAT, apresentamos o SKICCOSMO cuja construção só foi possível a partir da referência ao SKICAT.

3.2 Modelo do SKICCOSMO

A construção de um banco de dados para a astronomia, nos moldes do SKICCOSMO, requer a integração dos dados dentro dos modernos conceitos de banco de dados relacional, além de conhecimento da própria astronomia já que esta introduz particularidades e questões singulares para a organização e estruturação dos bancos de dados. Esse trabalho não pode ser feito sem um detalhado conhecimento das diversas entidades e seus atributos do SKICAT, mesmo que este não esteja conforme esses referidos conceitos. Diversas foram as tentativas para fixarmos uma estrutura obedecendo essas prerrogativas. A forma final dessa estrutura pode ser vista na Figura 3.2, cujos detalhes veremos a seguir.

3.2.1 Legado de Catálogos

Para a construção do SKICCOSMO, é preciso se levar em conta a tradição da organização de catálogos na astronomia. Esse conceito é chamado, na terminologia dos profissionais de banco de dados, de *legacy*, ou legado. É uma “herança” da estrutura de armazenamento da informação antes dos modernos conceitos que hoje imperam na área. É a área de trabalho onde não se pode desprezar o que veio na “pré-história” dos bancos de dados relacionais, pois é preciso recuperar os dados contidos nos catálogos originais. Essa é a condição para que se construa o banco de dados na condição *bottom-up* (ver 1.4.2.7).

Essa característica também vale para a astronomia. O que temos, em termos de legado de informação, são os catálogos tradicionais, que ainda hoje constituem uma fonte importante de informação. O SKICCOSMO deverá permitir, portanto, o acesso a dados desses catálogos, integrados ao banco de dados, de maneira a também permitir a recuperação da informação de forma integrada.

3.2.2 Descrição das Diferentes Entidades no Banco de Dados

3.2.2.1 A Tabela *Objects*

Contém os dados universais para cada objeto. Abaixo apresentamos a descrição de cada atributo, suas características no contexto do banco de dados e seu significado astrofísico:

1. ID_Obj (**long**, UNN⁸), número de identificação do objeto. É único na tabela e define univocamente um objeto no banco de dados e por isso será definido como *chave primária*, e, naturalmente, irá compor um dos índices dessa chave. O Informix admite um máximo de 2^{31} para esse tipo de número, ou seja, mantendo-se esse identificador como único, pode-se armazenar até 2.15 giga objetos. Nossa ambição é que esse banco de dados possa armazenar cerca de 10 tera objetos. Para que isso seja, portanto, possível, será necessário adicionar, no seu tempo devido, um novo atributo, por exemplo “ID_Obj(**smallint**)” o que estenderá a capacidade do banco de dados para armazenar até 70 tera objetos;
2. RA (**double**, NN), ascensão reta do objeto para o equador de J2000, em radianos, para permitir que funções trigonométricas sejam aplicadas de forma imediata. Apesar

⁸A sigla “NN” significa “*not null*”, indicando que o atributo nunca deve possuir valor indeterminado, enquanto que “U” significa “*unique*” geralmente indicando ser o atributo parte da chave primária, não podendo possuir valores repetidos na mesma tabela.

da grande cardinalidade, o que significa que seu índice ocupará uma grande área do espaço em disco, não se pode dispensar um índice para esse atributo dada a grande dependência que as consultas terão dele;

3. DEC (**double**, NN), declinação do objeto para o equador de J2000, igualmente em radianos. Mesmo argumento com respeito ao índice;
4. ID_Header (**long**, NN), contém o número de identificação dos catálogos em que o objeto foi extraído. Representa um relacionamento dessa tabela com a tabela **POSSII_Headers**, cujos detalhes veremos a seguir. Esse atributo possui interesse apenas administrativo pois as consultas dos usuários não dependerão dele, como veremos no capítulo 4.

Esses são os atributos que estarão contidos na tabela **Objects**. Outros atributos, tais como magnitude, magnitude isofotal, etc, entrarão nos catálogos especialmente construídos para esse fim. A razão da divisão dos atributos em diferentes tabelas é que os atributos fotométricos são, conceitualmente, diferentes dos astrométricos. Um objeto pode ter várias medidas de fotometria mas é definido por apenas uma posição no céu (mesmo que possa mudar no tempo). As classificações dos objetos são colocadas em uma tabela à parte. Possíveis correções astrométricas poderão ser obtidas no futuro, e constituirão uma nova entidade dentro do banco de dados, que poderá ser acompanhada do aplicativo correspondente que implemente o fornecimento das posições atualizadas.

Garante-se, com essas medidas: 1) maior granularidade na obtenção das diferentes informações científicas dos objetos; 2) maior flexibilidade na atualização e correção das informações dos objetos. Vemos, na Figura 3.2, a representação pictórica da tabela “Objects”.

3.2.2.2 POSS_II: Tabelas de *features*

A construção do SKICAT baseia-se nos “catálogos” da tabela de *features*, que se sustenta na estrutura dos “catálogos” do FOCAS (Fayyad *et al*, 1993, [20] e Valdes *et al*, [63]. Essa estrutura define a tabela *Features*, em cada “catálogo” contendo informações de interesse astrofísico. No entanto, se, como vimos em 3.1, essa estrutura não é conveniente para um banco de dados relacional, é preciso reestruturar o modelo de banco de dados de forma a conformá-lo com as condições do BDR. Decidimos, então, extraí-la de uma tabela tendo por base a tabela *Features* do SKICAT, salvo que esta será estruturada para estar

SKICCOSMO – DPOSS II DIAGRAMA

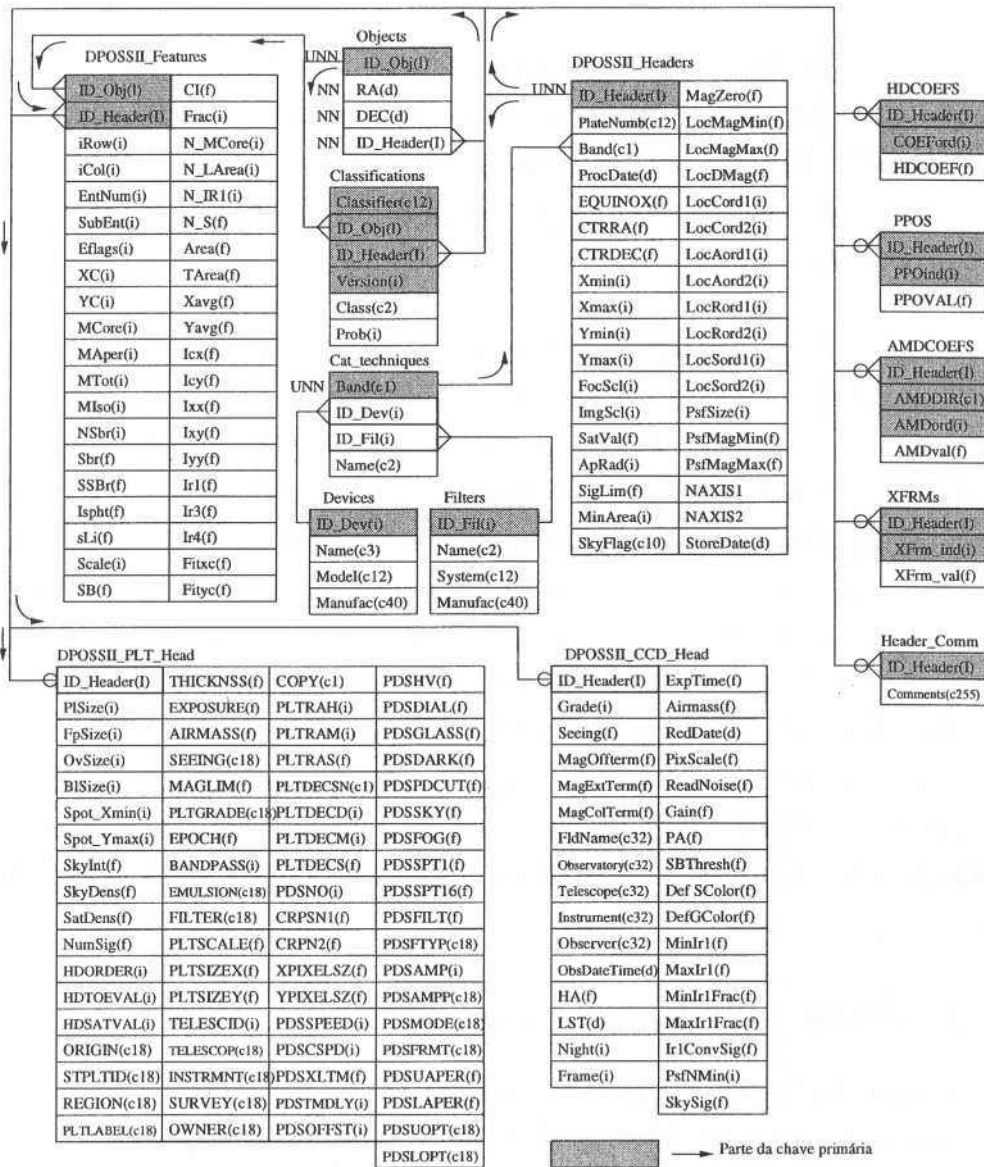


Figura 3.2: Atributos das tabelas do SKICCOSMO.

As setas foram introduzidas para facilitar a visualização do sentido proprietário - cliente (vide 1.4.2.2)

em condições de atender aos preceitos das formas normais de Codd. Para que isso seja possível, é necessário gerar tabelas que sejam composições e desmembramentos de tabelas do SKICAT.

É particularmente importante a noção que uma tabela construída nessas condições ganhará a condição de catálogo (ver 3.2.1), contendo todas as características deste. Cada objeto identificado e classificado em um catálogo SKICAT, ganhará aqui uma linha na tabela dedicada às *Features*. As diferentes instâncias desse objeto nos diferentes campos, tanto em placas fotográficas, quanto em CCD, terão entrada nessa tabela. Um mesmo objeto, medido com o mesmo filtro e mesma técnica, terá mais de uma instância nessa tabela. Por exemplo, quando duas placas do POSS-II se sobreporem no mesmo campo no céu. Será atribuição do usuário escolher aquela que melhor lhe convier ou como ela comporá o resultado final. O nome dessa tabela é **DPOSSII_Features**. A seguir apresentamos uma breve descrição de seus atributos:

1. **ID_Obj(long, NN)**: identificação do objeto na tabela *Objects*. Representa, com **ID_Header**, abaixo descrito, a chave primária dessa tabela;
2. **ID_Header(long, NN)**: identificação do *header*, ou cabeçalho, contendo as informações relativas a técnicas observacionais e do tratamento dos dados relativos ao objeto detectado. É uma chave estrangeira, fazendo junção com a tabela **DPOSSII-Headers**. Detalhes sobre a tabela **Headers** são vistos em 3.2.2.6. **ID_Header** faz, com **ID_Obj**, a chave primária dessa tabela;
3. **iRow(int, NN)**: a fileira do “*footprint*”⁹ (ver Weir, 1994, [66]) é registrada como “*Row*” no SKICAT. Esse atributo, que assume valores que vão de 1 a 13, corresponde ao número da fileira do “*footprint*” na placa. O Informix trata o termo “*row*” como uma palavra reservada. Por isso a mudança de nome;
4. **iCol(int, NN)**: coluna do *footprint*. Como **iRow**, o atributo **iCol** possui valores entre 1 e 13, o par (**iRow**, **iCol**) corresponde ao número do *footprint* no momento do processamento. Em nome da coerência, adotou-se os nomes **iRow** e **iCol** no SKICCOSMO, mesmo que o Informix não determine **Col** como palavra reservada;
5. **EntNum (int, NN)**: número definido pelo FOCAS que identifica o objeto no catálogo FOCAS. Não necessariamente é único em um catálogo;

⁹Ver em 3.6 para mais detalhes a respeito do termo *footprint*.

6. **SubEnt(int, NN)**: definido pelo FOCAS, define junto com **EntNum**, **Row** e **Col** a chave primária da tabela ou **catálogo** do SKICAT. Determina a multiplicidade do objeto. É mantido, além dos outros três atributos, a título de compatibilidade com o uso do SKICAT. Esses atributos poderão ser úteis para resolver possíveis problemas de identificação e classificação de objetos, no futuro;
7. **Eflags(uint)**: atribui valores lógicos aos diferentes objetos detectados, caracterizando, por exemplo, se estes estão saturados, próximos do bordo da imagem, etc;
8. **XC, YC(int)**: posição do objeto em X e Y com respeito ao centro da placa;
9. **MCore(int)**: magnitude do *core* ($\times 1000$);
10. **MAPER(int)**: magnitude de abertura (ver FOCAS, Valdes *et al*, [63]) ($\times 1000$);
11. **MTot(int)**: magnitude total ($\times 1000$);
12. **MIso(int)**: magnitude isofotal ($\times 1000$);
13. **NSbr(int)**: número de pixels da região de fundo de céu correspondente ao objeto;
14. **SBr(float)**: Brilho superficial instrumental do fundo de céu;
15. **SSBr(float)**: Sigma^{10} do brilho superficial do fundo de céu;
16. **Ispht(float)**: brilho isofotal;
17. **sLi(float)**: sigma da luminosidade isofotal em unidades de magnitude;
18. **Scale(float)**: escala da resolução;
19. **Frac(float)**: fração da resolução.
20. **N_MCore**, **MCore** “normalizado” (ver Valdes *et al*, [63]), **N_LArea** log *Area* normalizado, **N_IR1**, **IR1** normalizado e **N_S** brilho superficial normalizado, todos multiplicados por 1000, (**int**);
21. **Area(float)**: área isofotal;
22. **TArea(float)**: área total;

¹⁰Adotamos chamar de “Sigma” o erro padrão das medidas de uma grandeza.

23. **Xavg(float)**: largura média em X;
24. **Yavg(float)**: largura média em Y;
25. **SB(float)**: brilho superficial, obtido pela integração da luminosidade dentro da eclipse ajustada ao último nível isofotal;
26. **CI(float)**: Índice de concentração. Relação do brilho do objeto dentro de uma elipse interna determinada por 0.3 do semi-eixo maior da elipse ajustada ao último nível isofotal;
27. **Icx(float)**: centroide em X ponderado pela intensidade;
28. **Icy(float)**: centroide em Y ponderado pela intensidade;
29. **Ixx(float)**: segundo momento da distribuição em X ponderado pela intensidade;
30. **Ixy(float)**: segundo momento da distribuição em X e Y ponderado pela intensidade;
31. **Iyy(float)**: segundo momento da distribuição em Y ponderado na intensidade;
32. **Ir1(float)**: primeiro momento da distribuição radial ponderado pela intensidade;
33. **Ir3(float)**: terceiro momento da distribuição radial ponderado pela intensidade;
34. **Ir4(float)**: quarto momento da distribuição radial ponderado pela intensidade;
35. **Fitxc e Fityc(float)** Posição do centro do objeto estimado a partir do ajuste da *Point Spread Function* à imagem por máxima verossimilhança;

A Figura 3.2 mostra os atributos dessa tabela, seu relacionamento com a tabela primária e com outras tabelas de controle, como a *Cat_Techniques*, *Devices* e *Filters* (ver a seguir).

3.2.2.3 As tabelas *Devices* e *Filters*

No contexto da definição de uma técnica observacional, a descrição do dispositivo de coleta da luz e o filtro fotométrico são de fundamental importância. Uma descrição sumária desses componentes se faz necessário para que no futuro possamos incorporar novos detalhes a respeito de técnicas de obtenção dos dados, possibilitando ao usuário uma melhor compreensão dos dados recuperados nesse banco de dados. A tabela *Devices* é composta de quatro colunas:

1. **ID_Dev(int,UNN)**: Identificação do dispositivo;
2. **Name(char(3), NN)**: Pode assumir valores tais como CCD ou PLT, a primeira sigla para os dispositivos “CCD” e a segunda, para placas fotográficas. Usado com chave primária;
3. **Model(char(12))**: Modelo do CCD ou da placa;
4. **Manufac(char(40))**: Nome do fabricante.

A tabela *Filters*, por sua vez, também possui quatro colunas:

1. **ID_Fil(int,UNN)**: Identificação do filtro. Adotado como chave primária;
2. **Name(char(12), NN)**: pode assumir valores dos nomes consagrados aos filtros, por exemplo, U, B, V, r, z, etc;
3. **System(char(12))**: assume valores de acordo com o sistema fotométrico a que pertence o filtro, por exemplo, Johnson, Gunn, etc;
4. **Manufac(char(40))**: Nome do fabricante.

3.2.2.4 A tabela *Classifications*

Essa tabela é importante no esquema de definição de atributos de objetos do SKICAT e do SKICCOSMO, por conseqüência. No SKICAT, inicialmente, havia apenas um atributo dando a classificação do objeto que era aquela atribuída pelo FOCAS. Mais tarde, introduziu-se também um outro critério de classificação baseado em árvores de decisão (Weir, 1994, [66]). Antecipando-se a outros eventuais critérios, criamos essa tabela onde as classificações possíveis para um único objeto são virtualmente de número infinito. Dado um campo, um classificador deve identificar e fornecer uma única classificação para o objeto. Os atributos dessa tabela:

1. **Classifier(char(12) NN)**: Nome do classificador, por exemplo, FOCAS, DTREE, etc;
2. **ID_Obj(long NN)**: Junção com a tabela **Objects**;
3. **ID_Header(long NN)**: Junção com a tabela **DPOSSII_Headers**;
4. **Version(int)**: Versão para a classificação;

5. **Class(char(2) NN)**: Classificação do objeto, p.ex., **g**, **s**, etc;
6. **Prob(int NN)**: Probabilidade atribuída pelo classificador. Varia de 0 a 100.

3.2.2.5 A tabela *Cat_techniques*

As tabelas **Filters** e **Devices** mantêm um relacionamento de $n : m$, o que contradiz as regras de normalização descritas em 1.4.2.10. Criamos uma tabela adicional que vai auxiliar o usuário na tarefa de selecionar os diferentes dados à disposição. Esta tabela conterá as informações necessárias para a recuperação dos dados e será a guia para a seleção do tipo de informação que se quer. Deve ser possível determinar, a partir desta tabela, se um dado do catálogo contém informações fotométricas, espectrais, ou campos de imagens, como é o caso do IRAS. Essa tabela, portanto, conterá tanto a chave para a construção dos nomes dos diferentes catálogos, quanto da forma com que se vai recuperar a informação. Essa tabela chama-se **Cat_techniques** e possui os atributos:

1. **Cod_tech(int, UNN)**: identificação da técnica, chave primária;
2. **ID_Dev(int)**: identificação do detetor. Informações do detetor, quer seja placa fotográfica, quer seja CCD, CCD-IR, etc. Mantém relacionamento com a tabela **Detectors**;
3. **ID_Fil(int)**: identificação do filtro fotométrico, se usado. Mantém relacionamento com a tabela **Filters**;
4. **Name(char(2))**: Quando for o caso, como nos nomes das emulsões - que compreendem o conjunto placa fotográfica e filtro - dá-se um nome à técnica de observação, por exemplo, J, F, N, etc.

3.2.2.6 A tabela *DPOSSII_Headers*

Trata-se de uma intersecção entre as informações contidas nos *Headers* dos catálogos *Plate* e *CCD*. Nessa tabela estão contidas algumas informações importantes para o usuário. Por exemplo, qual a precisão das medidas de magnitude. Dada a inhomogeneidade de técnicas observacionais contidas na tabela de *Features*, é importante examinar a origem dos dados na tabela de *Headers*. Através dela e suas tabelas derivadas, recuperamos coordenadas equatoriais de um dado objeto, tais como os obtidos no processamento do campo; mas não as coordenadas contidas na tabela **Objects**, cujos valores foram obtidos por *matching*.

Acrescente-se também, dados dos relatórios da observação, digitação e processamento do campo, ou dos campos relativos ao objeto em questão. Os atributos são:

1. **ID_Header (long, UNN)**: Identificador do campo na tabela. Serve de chave primária a essa tabela;
2. **PlateNumber(char(12), NN)**: Número do campo conforme estabelecido no nome da tabela respectiva no SKICAT. A composição do atributo **Band** (ver abaixo) e o **PlateNumber** dá o nome das tabelas definidas no SKICAT tanto para *Header*, quanto para *Features*. Essa característica do SKICAT é uma das razões que nos levaram a modificar radicalmente a arquitetura do novo banco de dados: não havia qualquer conformidade às formas normais de Codd. Além disso, como consequência, temos, no SKICAT, tabelas de uma só linha. No caso em que o campo a ser carregado é proveniente de uma imagem CCD, o valor de **PlateNumber** será aquele que corresponde ao número da placa que contém esse campo;
3. **Band(char(1))**: Letra descrevendo a banda fotométrica utilizada. Frequentemente é o nome do filtro no sistema utilizado. Os detalhes fotométricos podem ser recuperados da tabela **Cat_techniques** e relacionadas. Representa a chave estrangeira na junção com a tabela **Cat_techniques**;
4. **ProcDate(date)**: Data do processamento;
5. **EQUINOX(float)**: Época do equinócio e do equador das coordenadas equatoriais;
6. **CTRRA e CTRDEC (float)**: Coordenadas equatoriais do centro do campo em graus decimais;
7. **Xmin, Xmax, Ymin, Ymax(int)**: Valores mínimos e máximos de X e Y, definidos durante o pré-processamento do campo, seja da placa fotográfica, seja do CCD, para a execução do *autoplate* ou do *autoccd* (ver Weir, 1994, [66]);
8. **FocSc1(int)**: Fator de escala definido no processamento do FOCAS (Valdes *et al*, [63]);
9. **ImgSc1(int)**: Fator de redução da imagem para o “*snapshot*”. Definido no processo de digitação da placa fotográfica;

10. **SatVal(float)**: Contagem de saturação. Se for para placa fotográfica, o valor da saturação é dado em contagens do microdensitômetro, se for para o CCD, contagem desse próprio dispositivo;
11. **ApRad(int)**: Raio da abertura, definido no FOCAS (Valdes *et al*, [63]);
12. **SigLim(float)**: Limite de detecção (Sigma), variável para o FOCAS (Valdes *et al*, [63]);
13. **MinArea(int)**: Limite inferior da área para o critério de detecção de objetos (FOCAS, Valdes *et al*, [63]);
14. **SkyFlag(char(10))**: Parâmetro do FOCAS determinando o modelo da estimativa do fundo de céu para a medida do céu (FOCAS, Weir, 1994, [66]);
15. **MagZero(float)**: Ponto zero da escala de magnitude;
16. **LocMagMin, LocMagMax, LocDMag (float), LocCord1, LocCord2, LocAord1, LocAord2, LocRord1, LocRord2, LocSord1, LocSord2 (int)**: variáveis definidas pelo processo de classificação dos objetos segundo o método de “árvores de decisão”. Ver Weir, 1994, [66], para detalhes;
17. **PsfSize(int)**: largura, a meia altura, da “*Point Spread Function*” (PSF);
18. **PsfMagMin, PsfMagMax(int)**: Magnitudes mínima e máxima das estrelas escolhidas para construir a “PSF”;
19. **NAXIS1 e NAXIS2 (int)**: Número de pixels nas direções X e Y;
20. **StoreDate(date)**: Data em que os dados foram inseridos no SKICCOSMO

Alguns atributos vindos das tabelas *Header* do SKICAT são “indexados”, ou seja, representam diferentes valores da mesma grandeza. Esses casos são tratados nas regras de normalização (ver em 1.4.2.10), que recomendam a construção de tabelas independentes onde constarão esses atributos. Essas tabelas são mostradas na Figura 3.2 (lado superior direito da figura) e são: **HDCOEFS**, **PPOS**, **AMDCOEFS** e **XFRMs**. As três primeiras derivam do antigo *Header* da placa fotográfica e a última do CCD. Por motivos óbvios essas tabelas são declaradas terem relacionamentos opcionais com a tabela “DPOS-SII_Headers”. Através das instâncias de **HDCOEFS** reconstitue-se a curva de resposta da

placa; de PPOS obtém-se a orientação da placa na observação; de AMDCOEFS obtém-se os coeficientes do polinômio astrométrico. Esse polinômio é ajustado para cada placa a partir das estrelas do GSC (*Guide Star Catalog*, ver Lasker *et al*, 1988, [41]). Pode-se recuperar, portanto os valores das coordenadas equatoriais dos objetos nessa placa utilizando esses coeficientes. Finalmente, com os coeficientes XFRM, obtém-se as coordenadas em α e δ da imagem CCD. Esses coeficientes são dispostos em uma matriz 3×3 . Esses termos, no entanto serão guardados na tabela XFRMs em ordem seqüencial. A recuperação dos termos da matriz pode ser feita através de uma *stored procedure*, ou dentro de uma *view*, em que a transformação se dá segundo:

$$\begin{aligned} i &= k/3 + 1 \\ j &= k\%3 + 1 \end{aligned} \quad k = 0..8$$

onde i e j são os índices da matriz e k o índice guardado na tabela e o símbolo % representa o resto da divisão. Dessa feita, obtém-se k da relação:

$$k = 3(i - 1) + j - 1 \quad i, j = 1..3$$

3.2.2.7 As Tabelas DPOSSII_xxx_Head

Existem duas dessas tabelas: a tabela DPOSSII_PLT_HEAD e DPOSSII_CCD_HEAD. Seguindo a estatística do FOCAS e do SKICAT, é preciso gerar uma tabela contendo informações relativas às observações e tratamento dos dados. Além dessas condições, nessa tabela encontram-se informações a respeito dos classificadores dos objetos e das regras utilizadas para classificar os mesmos. Essas informações foram utilizadas nas diversas fases do tratamento da informação, desde a observação até a classificação, passando pela sua digitalização e redução. A grande maioria desses dados não será de interesse para o astrônomo, salvo se este estiver interessado em uma análise mais profunda, por exemplo, na análise crítica dos procedimentos de redução. Esses dados ficarão mantidos no banco de dados, no interesse da maior granularidade possível para o usuário terminal. Existem ainda, atributos oriundos do SKICAT que não são úteis ao usuário. Alguns desses atributos são nomes de arquivos de controle durante a redução e/ou classificação. Esses atributos serão ignorados na transferência dos dados ao SKICCOSMO.

Uma outra particularidade: haverá uma tabela específica *Headers* para cada técnica observacional utilizada na obtenção das informações de um dado objeto. Imagens obtidas por placas fotográficas ganharão uma tabela *Plate_Headers*, imagens obtidas via CCD

terão uma tabela *CCD_Headers*. A descrição dos atributos dessa tabela na instância do POSS-II é apresentada a seguir:

1. **ID_Header**(long, NN): Identificador do *Header*, junção com a tabela **DPOSSII_Headers**;
2. **PlSize**, **FpSize**, **OvSize**, **B1Size** (int): dimensões da placa, do *footprint*, do *offset* e do bloco da imagem, respectivamente;
3. **Spot_Xmin**, **Spot_Ymax**(int): definição da região dos *spots* sensitométricos;
4. **SkyInt**, **SkyDens**, **SatDens**, **NumSig**(float): intensidade do céu, densidade do fundo de céu, densidade de saturação, limite de detecção na medida dos objetos, respectivamente;
5. **HDORDER**, **HDTOEVAL**, **HDSATVAL**(int): parâmetros da curva característica da placa, os coeficientes estão armazenados em tabela à parte (ver em 3.2.2.6);
6. **STPLTID**, **REGION**, **PLTLABEL**(char(18)): dados do microdensitômetro do STScI, origem, identificação da placa, região digitalizada, *label* da placa;
7. **THCKNSS**, **EXPOSURE**, **AIRMASS** (float), **SEEING** (char(18)), **MAGLIM** (float), **PLTGRADE** (char(18)), **EPOCH** (float), **BANDPASS** (int), **EMULSION**, **FILTER** (char(18)), **PLTSCALE**, **PLTSIZEX**, **PLTSIZEY** (float), **TELESCID** (int), **TELESCOP**, **INSTRMNT**, **SURVEY**, **OWNER** (char(18)), **COPY** (char(1)): informações do observatório, espessura da placa, tempo de exposição, massa de ar, código identificando a medida do *seeing*, limite da magnitude, código identificando a qualidade da placa, época da observação, código da banda fotométrica, tipo de emulsão, filtro, escala nominal da placa, dimensões da placa em X e Y, identificação do telescópio na estrutura do Observatório do Monte Palomar, nome do telescópio, identificação do chassi da placa, nome do levantamento fotográfico, proprietário da placa e código da cópia;
8. **PLTRAH**, **PLTRAM** (int), **PLTRAS** (float), **PLTDECSN** (char(1)), **PLTDECD**, **PLTDECM** (int), **PLTDECS** (float): coordenadas do centro da placa;
9. **PDSNO** (int), **CRPSN1-2** (float), **XPIXELSZ**, **YPIXELSZ** (double), **PDSSPEED**, **PDSCSPD** (int), **PDSPXLTM** (double), **PDSTMDLY**, **PDSOFFST** (int), **PDSHV**, **PDSIDIAL** (double), **PDSGLASS**, **PDSDARK**, **PDSPDCUT**, **PDSSKY**, **PDSFOG**, **PDSSPT1**, **PDSSPT16**, **PDSFILT** (float), **PDSFTYP** (char(18)), **PDSAMP** (int), **PDSAMPP**, **PDSMODE**, **PDSFRMT** (char(18)), **PDSUAPER**,

PDSLAPER (float), PDSUOPT, PDSLOPT (char(18)): dados da digitalização, número PDS, limites da digitalização, tamanho dos pixels em microns, velocidade nominal do PDS, velocidade ajustada do PDS em microsec por pixel, tempo de "delay" do PDS em microsec, tempo de *offset* em microsec, voltagem em kV, leitura da voltagem no monitor, leitura do vidro limpo, corrente de escuro, densidade de corte do PDS, leitura do fundo de céu, leitura da densidade do "fog", leitura no primeiro spot do sensitômetro, identificação da montagem do filtro, tipo de filtro, número do amplificador, amperagem após processamento, modo de digitalização, número de registros em 512 bytes, abertura superior, abertura inferior, ótica superior e ótica inferior;

Para as observações CCD temos a tabela *Headers* definida com os seguintes atributos:

1. ID_Header(long, NN): Identificador do *Header*, junção com a tabela DPOSSII_Headers;
2. Grade(int): código indicando a qualidade da imagem;
3. Seeing(float): *seeing* medido através da FWHM;
4. MagOffTerm, MagExtTerm, MagColTerm(float): para calibração da magnitude instrumental, tendo os seguintes termos: ponto zero, termo da extinção e termo do índice de cor;
5. FldName, Observatory, Telescope, Instrument, Observer (char(32)): nome do campo, observatório, telescópio, instrumento e observador;
6. ObsDateTime(date): data da observação;
7. SBThresh, DefSColor, DefGColor (float): parâmetros para o FOCAS: limite de detecção, termo de cor *default* para estrelas e galáxias, respectivamente;
8. MinIri1, MaxIri1, MinIri1Frac, MaxIri1Frac, Iri1ConvSig (float): parâmetros para o algoritmo de determinação do primeiro momento;
9. PsfNMin(int): Número mínimo de estrelas usadas para a determinação da PSF;
10. SkySig (float): sigma para a detecção do fundo de céu (FOCAS).

Nota-se que essa tabela possui um número menor de atributos que a anterior, relativa a imagens de placa fotográfica. Isso se deve ao fato de que as placas fotográficas passam por um processo adicional, em relação aos *frames* CCD: a digitação.

Os atributos que serão colocados fora do banco de dados *on-line*, sendo armazenados em fitas, para a tabela *Headers CCD* são:

1. **Type(int)**: indica o tipo de imagem (2=CCD). Irrelevante porque constitui um dado redundante;
2. **StoreName (char(32))**, **StoreNum (int)**, **AuxStoreName (char(32))**, **AuxStoreNum (int)**, **ImStoreName (char(32))**, **ImStoreNum (int)**, **ImName (char(32))**: são eliminadas da lista equivalente para placas fotográficas por não serem de interesse ao usuário; **ClassifDef**, **ClassifRules (char(80))**: idem no item 4; **NAXIS1**, **NAXIS2 (int)**: idem no item 5;
3. **AstromRef (char(32))**, **AreaFile (char(13))**: contêm referências a arquivos que não estarão presentes no banco de dados, constituindo um risco de violação de integridade.

3.2.2.8 A Tabela de Comentários DPOSSII

Existe um atributo na tabela de cabeçalho do SKICAT chamado **Comments(char(255))** reservado para comentários do operador, seja do digitalizador “PDS”, seja do administrador do banco de dados. Normalmente esse atributo permanece vazio. No intuito de economizar espaço em disco, e, ao mesmo tempo, ampliar o leque de opções na operação das diversas fases do tratamento dos dados até sua inclusão no banco de dados, foi criada uma tabela à parte, especificamente para guardar os comentários. Dessa forma, se a coluna “Comments” vier em branco, nada se guarda nessa tabela. Se, por acaso, for necessário tecer mais comentários sobre um determinado campo, não haverá limite para armazená-los no banco de dados:

1. **ID_Header(long, NN)**: Identificador do *Header*, junção com a tabela *DPOSSII-Headers*;
2. **Comments(char(255))**: Comentários dos operadores.

3.3 A “Vista” (*view*) *Features*

O Informix, conforme o padrão “ANSI” permite a definição das chamadas “vistas” ou *views*. Na descrição de Bowman *et al* (Bowman *et al*, 1998, [6]), p. 12 uma “*view*” pode ser entendida como uma **tabela virtual**. Outras comparações (Simmons, 1997, [54]) preferem mostrar uma “*view*” como uma “vista” de uma janela, por exemplo de trem, deixando exibir apenas um detalhe da “paisagem”, em nosso caso, um aspecto do banco de dados. A “vista” é um instrumento criado com o principal objetivo de facilitar a construção de consultas.

Foi criado no SKIICCOSMO uma “vista” denominada “Features” que foi construída, conforme a sintaxe recomendada, com a seguinte instrução “SQL”:

```
create view Features
(id_obj, id_header, irow, icol, eflags, xc, yc, mcore, maper, mtot,
miso, sbr, ssbr, ispht, sli, scale, sb, ci, frac, area, xavg, yavg,
icx, icy, ix, ixy, iyy, ir1, ir2, ir3, ir4, pa, asymm, aflag, bflag,
cflag, dflag, eflag, fflag, lflag, pflag, rflag, sflag)
as
select id_obj, id_header, irow, icol, eflags,
xc, yc, mcore/1000., maper/1000., mtot/1000., miso/1000., sbr, ssbr,
ispht, sli, scale/100., sb, ci, frac/100., area, xavg, yavg, icx, icy,
ix, ixy, iyy, ir1, sqrt(ixx+iiy), ir3, ir4, atan(2*ixy/(iiy-ixx)),
sqrt(pow(iiy-ixx,2)+pow(2*ixy,2))/(ixx+iiy),
trunc(mod(eflags,2)/1), trunc(mod(eflags,4)/2),
trunc(mod(eflags,8)/4), trunc(mod(eflags,16)/8),
trunc(mod(eflags,32)/16), trunc(mod(eflags,64)/32),
trunc(mod(eflags,128)/64), trunc(mod(eflags,256)/128),
trunc(mod(eflags,512)/256), trunc(mod(eflags,1024)/512)
from
dpossii_features
```

Entre as vantagens de se utilizar a vista **Features** destacam-se:

- Todos os atributos de magnitude foram divididos por 1000 para se fornecer valores em dimensão esperadas pelo usuário;
- Foram criados dois atributos: “pa” e “asymm”, conforme documentação do FOCAS (Valdes, 1988, [62]);

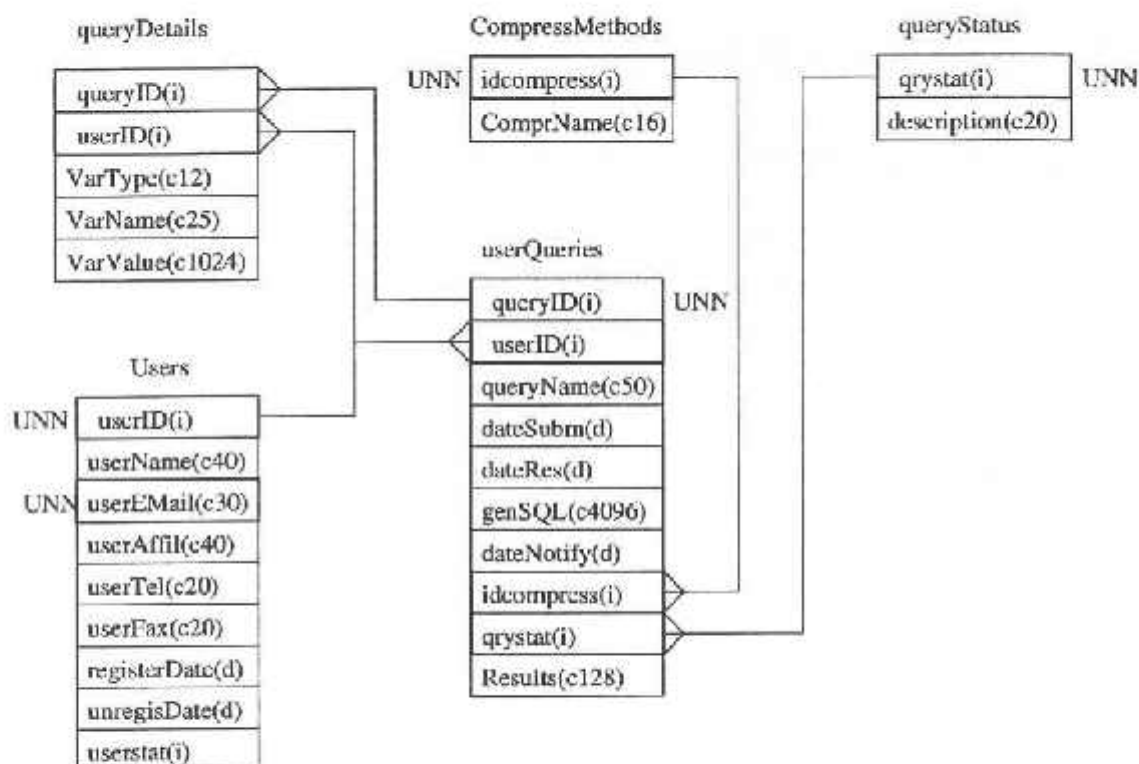


Figura 3.3: Tabelas auxiliares à administração do SKICCOSMO

- Os *flags*, ou indicadores do FOCAS, antes condensados no atributo *eflags* (ver 3.2.2.2) foram recuperados através de operações aritméticas que simulam máscaras lógicas. É o que se pode ver nas expressões envolvendo *trunc* e *mod* em *eflags*.

Uma vez definida a “vista”, as consultas poderão se referir diretamente aos atributos nela definidos, como se estivéssemos recuperando os dados da tabela “real” *DPOSSII_Features*.

3.4 Tabelas de auxílio à administração

Além das tabelas contendo os dados que serão disponibilizados para os usuários, algumas tabelas de auxílio à administração do SKICCOSMO são necessárias. Essas tabelas, cujo modelo de relacionamento de entidades pode ser visto na Figura 3.3, permitem automatizar o processo de coleta de consultas, submissão e redistribuição. A armazenagem dos dados dos usuários e do relatório do que eles fizeram durante a construção da consulta, permite o acompanhamento da operação do SKICCOSMO, bem como a busca de soluções em caso de problemas e eventuais aperfeiçoamentos de sua estrutura.

3.5 A Chave Primária

3.5.1 Como definir a chave primária?

Em um catálogo astronômico, um objeto é definido pela sua posição na esfera celeste e pelo atributo definindo uma classificação. A posição, determinada pelo par de coordenadas celestes, ascensão reta e declinação, não é suficiente para definir um objeto, pois é possível que dois ou mais objetos ocupem a mesma posição no céu. Uma terceira coordenada, a distância do objeto, poderia resolver essa ambigüidade. No entanto, para a maioria dos objetos astronômicos, esse atributo é desconhecido. A atribuição de uma classificação a um objeto ainda apresenta outras dificuldades. Poderíamos classificar um objeto como galáctico ou extra-galáctico. Não obstante esse procedimento não é suficiente, pois não há nada que impeça de haver dois objetos tanto galácticos quanto extra-galácticos ocupando a mesma posição na esfera celeste.

O instrumento de identificação e classificação utilizado na construção desse banco de dados, o FOCAS, não é capaz de eliminar essa indeterminação. Dentro dos limites de precisão astrométrica de nossos dados, o FOCAS, no caso de uma coincidência exata da posição dos objetos, identificará esses objetos como um só objeto, com características fotométricas peculiares, sendo que nossos instrumentos de análise não são capazes de separar os objetos. Um desenvolvimento posterior de técnicas de identificação pode resolver esse problema. Em nosso banco de dados abriremos espaço para esse avanço, definindo uma chave primária única para cada objeto identificado, cujas coordenadas celestes servirão como instrumento de passagem para a definição dessa chave. A chave primária para a determinação de cada objeto será um número seqüencial próprio de nosso catálogo de objetos astronômicos.

3.5.2 A imprecisão astrométrica

Um campo obtido de uma placa fotográfica, ou CCD, define completamente um conjunto de objetos distinguidos pelas suas coordenadas de campo, ou - depois de transformados pelos polinômios astrométricos - pelas coordenadas esféricas. Para integrar esses objetos ao banco de dados, há de se levar em consideração as imprecisões astrométricas que fazem com que um mesmo objeto em um campo tenha coordenadas ligeiramente deslocadas em relação a outro campo (*offset*). Se não levarmos em conta esta correção relativa, incorremos no erro de produzir mais de uma linha, ou mais de uma chave, definindo um mesmo objeto

na tabela **Objects**. É necessário, portanto, aplicarmos um procedimento para *match* dois campos. Precisamos associar cada objeto de um campo a um só objeto de outro campo, ou, em outras palavras, precisamos associar cada objeto aos outros objetos do campo já registrado no SKICCOSMO.

3.5.2.1 Os métodos de *matching* presentes na literatura

P. Stetson (1989, [55]) e E.J. Groth (1986, [23]) desenvolveram, independentemente, métodos semelhantes para o procedimento de *matching*. Groth chama esse método de *Pattern Matching*. A idéia consiste em comparar e ajustar os campos através de formas de triângulos. Três objetos contíguos são escolhidos como referência em um dos campos. Forma-se assim um triângulo, cujos lados terão comprimentos normalizados segundo um critério objetivo. Em seguida, procura-se três objetos no outro campo que satisfaçam à condição de constituírem um triângulo semelhante. Esse método apresenta a limitação da dependência recorrente do número de objetos, o que faz com que o tempo de cálculo seja proporcional à potência de N . Para evitar essa limitação, Stetson baseia seu método em uma hierarquia dependente da magnitude dos objetos.

Murtagh (1992, [45]) faz uma revisão dos principais métodos de *matching* encontrados na literatura e apresenta um método chamado "*Point-Pattern Matching*", baseado na técnica do "*world view*", ponto a ponto, entre dois campos. A técnica consiste no seguinte: escolhe-se um dos pontos no campo A, supostamente contendo n pontos. Em seguida, obtêm-se todos os $n - 1$ vetores representando as distâncias dos outros pontos a esse ponto escolhido, ordenam-se esses vetores em ordem crescente em relação ao ângulo polar e faz-se um mapeamento desses ângulos em intervalos (*bins*) de um grau. Uma vez terminado esse procedimento, procura-se, para todos os pontos no campo B, um mapeamento equivalente. O ponto no campo B que mais se aproximar do mapeamento feito em A será declarado como provável candidato. Esse procedimento é reproduzido para os outros pontos em A. Cada par de pontos *matched* vai acrescentar uma equação que define os termos da translação, rotação e variação aleatória entre os campos A e B. Murtagh estima um tempo de 18s de cálculo em uma Sparc 2 para campos de 2000 objetos no caso de não haver rotação (os campos se diferenciam simplesmente por uma translação). Ele justifica o procedimento argumentando que o algoritmo usado não era dos mais eficientes (dependente de n^2). De qualquer forma, se formos transportar esse método para nosso problema, ou seja, campos de aproximadamente 500000 objetos, esse algoritmo tomaria cerca de 312 horas na CPU de uma Sparc 2.

3.5.2.2 O algoritmo de *matching* desenvolvido no interior do SKICAT

Weir (1994,[66]) descreve um algoritmo que se propõe a associar os objetos de um catálogo aos objetos de outro catálogo da mesma região do céu. Esse algoritmo depende de uma estimativa inicial de quanto ambos os campos estão deslocados.

Para fazer o *matching* faz-se uma correção na posição do objeto candidato e procura-se o objeto mais próximo do catálogo *matched*. Está implícita, aqui, a prévia existência de um catálogo *matched*. Caso não exista catálogo para a respectiva região do céu, lança-se um procedimento para criar um catálogo vazio e que possa ser preenchido. Nesse caso, os objetos do primeiro catálogo são inseridos automaticamente no catálogo, sem *matching* pois não há nada a ser comparado. O catálogo é dividido em segmentos cujas dimensões são pré-estabelecidas pelo usuário e previamente registradas no catálogo "MatchProc". O programa então segue um procedimento iterativo no sentido de minimizar a diferença média entre um catálogo e outro, para cada segmento.

Para proceder à minimização, o programa obedece a uma série de critérios, e uma série de parâmetros são testados para atenderem a condições de qualidade pré estabelecidas. Para cada objeto a distância estimada é definida como:

$$\Delta = \sqrt{\xi^2 + \zeta^2}$$

sendo ξ e ζ as distâncias nos eixos do campo. O desvio padrão será estimado junto com a distância para testar a proximidade da solução e então volta-se ao procedimento de correção da posição a partir do novo valor da distância, até que se chegue a um resultado satisfatório.

A experiência mostra que o procedimento de Weir (1994) é inaceitavelmente longo (72 horas de processamento numa Ultra-Sparc 1). Odewhan, (1999, [46]) fez uma adaptação desse algoritmo em que se obtêm, previamente, as coordenadas do banco de dados, dispensando-se, assim, a interação com o banco de dados durante a execução do programa. Experiências feitas em Caltech mostram que o tempo de processamento então pode ser reduzido para pouco mais de uma hora, na mesma Ultra-Sparc 1. Essa constatação põe em discussão a eficiência da conexão com o "motor" (*engine*) do banco de dados em questão (Sybase) através da Interface de Programação de Aplicação (API). A baixa performance nessa operação pode ter origem no gerenciamento inadequado dos recursos do banco de dados (DBA) como também da sua arquitetura de construção. A leitura do material pertinente (*Sybase - System Administration Guide*, [24]) indica que um estudo mais profundo

da configuração de memória dinâmica (*swap page, cache, etc*) poderia trazer uma maior eficiência nessa interação. Por outro lado, o trabalho de *workbench* de Odewhan é fortemente dependente da memória RAM da máquina em questão. O mesmo pode-se dizer do *workbench* de Murtagh, visto na Seção 3.5.2.1.

3.5.3 Procurando Alternativas

A construção de um modelo de banco de dados relacional para o caso do SKICCOSMO (visto em 3.2) exige que o procedimento de *matching* seja fundamental para a definição da tabela primária de objetos. Para tornar possível o processo de *matching* e gerar o SKICCOSMO é necessário encontrar uma alternativa. Para se ter uma idéia das dificuldades em construirmos um banco de dados baseado em *matchings* de campos de imagens, usando os programas de *matching* do SKICAT, a inscrição das placas, em uma só banda e cobrindo toda a esfera celeste tomaria mais de 14 anos de CPU.

Para utilizarmos um novo banco de dados e orientá-lo para uma base objeto-relacional resolvemos elaborar um processo de *matching* baseado em um conceito diferente, que descrevemos abaixo.

3.5.4 O problema da identificação de um campo para o astrônomo observacional

O problema do relacionamento entre entidades (ou seja, catálogos, ex. FK5) na astronomia passa pela questão da definição de unicidade. À primeira vista, parece que a definição do que é um objeto único não é um problema. No entanto, teremos uma idéia do grau de dificuldade em se determinar a unicidade de um objeto - dado um conjunto de catálogos que supomos representar a mesma região no céu - quando comparamos essa determinação ao trabalho do astrônomo em definir o objeto de interesse no campo de uma "TV-guider" em um telescópio. Geralmente o astrônomo se serve de um mapa do campo (o *finding chart*) e compara o campo com a imagem no monitor para verificar se a "topografia" se "ajusta" uma com a outra. Para ter sucesso nessa ação é preciso que o astrônomo confie que o telescópio esteja apontando para a região correta, isto é, que o campo observado e o de comparação sejam os mesmos, que a escala dos campos sejam equivalentes, e finalmente, que os dois campos sejam obtidos em regiões espectrais próximas, caso contrário, arrisca-se uma comparação de objetos que se comportam distintamente em diferentes comprimentos de onda.

3.5.5 A transposição do problema de identificação para o algoritmo

O algoritmo para o *matching* de catálogos de uma região do céu, que foram gerados a partir de imagens com diferentes filtros, deve seguir a mesma mecânica descrita acima. Parte-se de um dos catálogos que tomamos por referência. Toma-se um objeto nesse catálogo e determina-se a “topografia” de sua vizinhança, ou seja, verifica-se se a disposição dos objetos próximos. Em seguida, toma-se o outro catálogo, fixa-se um objeto que se supõe equivaler ao do catálogo de referência e compara-se a topografia da vizinhança com aquela daquele catálogo. Repete-se o mesmo procedimento para as diferentes possibilidades de translação e rotação até que se chegue ao resultado mais provável. Para que o processo computacional não se torne excessivamente longo parte-se de premissas e informações baseadas em referências aproximadas: mesma região, escalas dos catálogos conhecidas e direções dos eixos conhecidas. Com isso elimina-se as diferenças de escala e rotações. Essas condições são satisfeitas a partir do momento em que é aplicada a determinação da astrometria do campo. O problema, agora, é o ajuste fino das escalas e as translações (ou *offsets*).

3.5.6 O estimador de *Matching*

Um procedimento possível é o ajuste por correlação cruzada. Toma-se dois campos a serem comparados. Por simplicidade, vamos examinar dois campos em apenas uma dimensão. Sejam, pois, dois campos contendo, ambos, “**P**” pixels, cada um. O primeiro campo possui “**N**” objetos e o segundo, “**M**” objetos, sendo “**L**” o número de objetos comuns. Definimos correlação cruzada desses dois campos como:

$$\mathbf{R}(x_i) = \sum_{p=1}^{\mathbf{P}} \phi(x_p) \psi(x_p - x_i) \quad (3.1)$$

sendo ϕ e ψ funções constituídas de *Delta*'s de Kroenecker, que assumem o valor 1 a cada uma dada posição (x_n) onde exista um objeto, sendo nulas quando não for o caso. As coordenadas podem ser em qualquer unidade, contanto que seja coerente. As funções campo podem ser escritas, portanto, como a soma dos *Delta*'s:

$$\phi(x_p) = \sum_{n=1}^{\mathbf{N}} \delta_n(x_p) \quad (3.2)$$

com uma relação similar para ψ . Se os dois campos se ajustam perfeitamente, isto é, se são idênticos, a função $R(0)$ terá como valor numérico o número de objetos do campo. A função de correlação cruzada é um estimador da coincidência das duas funções de distribuição ϕ e ψ como veremos a seguir. Digamos que existam duas distribuições de uma variável que se distingam por um deslocamento Δ : $\phi(x) \cong \psi(x - \Delta)$. O sinal de aproximação se refere ao fato que as duas distribuições não são iguais pois, tomadas de placas de diferentes filtros podem existir objetos em uma que não existem em outra. Podemos estabelecer que a melhor posição relativa entre as duas distribuições, dada por Δ , é aquela que minimiza

$$\chi^2(\Delta) = \sum_{p=1}^P [\phi(x_p) - \psi(x_p - \Delta)]^2 \quad (3.3)$$

onde a soma em "r" se faz no domínio das duas distribuições. Desenvolvendo a expressão (3.3) acima, temos:

$$\chi^2(\Delta) = \sum_{p=1}^P \phi^2(x_p) - 2 \sum_{p=1}^P \phi(x_p) \psi(x_p - \Delta) + \sum_{p=1}^P \psi^2(x_p - \Delta)$$

O primeiro e último termo do lado direito dessa expressão representam, respectivamente, as autocorrelações das funções ϕ e ψ , e não contribuem para o processo de minimização. O termo do meio representa a correlação cruzada definida em (3.1), entre as duas funções, e irá minimizar χ^2 quando atingir seu valor máximo. A função de correlação cruzada, nessas condições, ficará da forma: $R(\Delta) = \sum_{p=1}^N \phi(x_p) \psi(x_p - \Delta)$. Consideremos, então, a transformada de Fourier dessa função de correlação. As funções ϕ e ψ estão convoluídas, de maneira que a transformada de Fourier de R será:

$$H(k) = \Phi \Psi^*$$

onde o símbolo "*" representa o complexo conjugado da função. Considerando que tanto a função ϕ , quanto ψ podem ser consideradas como somas de funções *delta* de Kroenecker, suas transformadas de Fourier podem ser colocadas na forma:

$$\Phi(k) = \frac{1}{P} \sum_{j=1}^P \phi(x_j) = \frac{1}{P} \sum_{j=1}^P \sum_{n=1}^N \delta_n(x_j) \exp(ikx_j) \quad (3.4)$$

com relação similar para Ψ . Sendo " $i = \sqrt{-1}$ ". Pela definição de *delta* de Kroenecker, sabemos que a soma (3.4) acima terá todos os seus componentes nulos, salvo o termo em

que $j = n$. Assim:

$$\Phi(k) = \frac{1}{\mathbf{P}} \sum_{n=1}^N \exp(\imath k x_n)$$

e:

$$\Psi(k) = \frac{1}{\mathbf{P}} \sum_{m=1}^M \exp(\imath k x_m)$$

Portanto, a transformada \mathbf{H} da função de correlação cruzada entre as duas distribuições representando os dois campos pode ser escrita como:

$$\mathbf{H}(k) = \frac{1}{\mathbf{P}^2} \sum_n \sum_m \exp[\imath k (x_n - x_m)]$$

Fazemos, então, a transformação inversa para recuperarmos a função de correlação:

$$\mathbf{R}(x_i) = \mathbf{P} \sum_k \mathbf{H}(k) \exp(-\imath k x_i) = \frac{1}{\mathbf{P}} \sum_{n,m} \sum_k \exp[\imath k (x_n - x_m - x_i)] = \frac{1}{\mathbf{P}} \sum_{n,m} \delta_{n-m}(x_i) \quad (3.5)$$

onde denotamos $\delta_{n-m}(x_i)$ o *delta* de Kroenecker em que o valor unitário se dá quando x_i é igual à diferença $x_n - x_m$. Essa soma é feita para todos os pontos de ambos os campos. No entanto, somente os valores de x_i iguais a $x_n - x_m$ contarão efetivamente para a distribuição. Se os dois campos diferem de fato apenas por um deslocamento, um campo vai definir o outro por uma transformação linear. Se um objeto em um campo for definido pela posição x_n e seu equivalente, no outro campo, pela posição X_n , então obtemos x_n através da transformação:

$$x_n = X_n - \Delta$$

se consideramos que o campo "X" deslocou-se de Δ . Se não há "falhas" em nenhum dos campos, isto é, se não existe objeto sem seu correspondente no outro campo, a distribuição de $\mathbf{P.R}$, em (3.5), terá como valor máximo o número de objetos comuns nos dois campos em $\mathbf{P.R}(x = \Delta) = \mathbf{L}$. Para campos sem "falhas", o valor procurado de Δ pode ser obtido simplesmente por:

$$\Delta = \bar{X} - \bar{x} \quad (3.6)$$

Contudo, na prática, teremos que considerar campos que não possuam todos objetos que se correspondem com os do outro campo dependendo do comportamento espectral do objeto. Por outro lado, a distribuição de \mathbf{R} não pode ser considerada como um resultado de um processo puramente aleatório. Há um "viés" nessa distribuição. A variância do

resultado (3.6) acima é uma composição das variâncias da distribuição dos objetos nos dois campos, o que não pode ser considerado estimador da variância de Δ . Para que tenhamos um estimador sem "viés", temos de considerar a transformação termo a termo:

$$X_i = x_i + \Delta + \epsilon_i \quad (3.7)$$

e determinar a variância σ_ϵ a partir dessa relação. Elevando-se os dois termos ao quadrado e fazendo-se a soma em todos os i 's, obtemos:

$$\sum X_i^2 = \sum x_i^2 + L\Delta^2 + \sum \epsilon_i^2 + 2 \sum X_i \epsilon_i + 2\Delta L\bar{x} + 2\Delta L \sum \epsilon_i$$

Sendo L o número de objetos comuns. O último termo do lado direito nos leva à média de ϵ_i , que, como já comentamos, é nula. O mesmo acontece para o quarto termo da expressão que representa a correlação entre X e ϵ . Se substituirmos cada soma quadrática pelo termo da variância (não normalizada) respectiva, obteremos:

$$\sigma_X^2 + L\bar{X}^2 = \sigma_x^2 + L\bar{x}^2 + L\Delta^2 + \sigma_\epsilon^2 + 2L\bar{x}\Delta$$

Substituindo Δ pela expressão (3.6) podemos demonstrar que:

$$\sigma_\epsilon^2 = \sigma_X^2 - \sigma_x^2 \quad (3.8)$$

3.5.7 Termos para uma relação biunívoca

Da expressão (3.7) concluímos que para cada objeto de um campo deve existir seu equivalente em outro. Não existe identificação *a priori* dos objetos, pois todo esse procedimento visa a constituição da identificação. Digamos que o primeiro campo " ϕ " seja o campo de referência, devidamente identificado e catalogado, e que queiramos atribuir aos objetos do campo " ψ " os índices do campo " ϕ ". Os objetos do campo " ϕ ", desta feita, já foram devidamente identificados e "marcados" com uma chave primária. O campo " ψ " deverá ser identificado e por isso ele poderá ter mais ou menos objetos que " ϕ ". Se tiver menos, os objetos sem par de " ϕ " deverão ser descartados do cálculo da média. Se, por outro lado, tiver mais, os objetos sem par deverão ganhar seus índices, e entrarão no catálogo com seus atributos. Contudo, como no primeiro caso, não entrarão no cálculo da média. Resta, então, o problema de identificar os objetos de " ψ " com respeito a " ϕ ". Para tal, usaremos

a propriedade:

$$(x_n - x_m) - (X_n - X_m) = \epsilon_i$$

e vamos assumir que $c_i \leq \min(x_n - x_m)$ ¹¹. Tomaremos todas as diferenças nos dois campos e ordenaremos em ordem crescente. Se as variações aleatórias não forem suficientemente grandes para inverter posições dos objetos, cada diferença em um campo vai equivaler à diferença dos mesmos objetos no outro campo. Portanto ambas as diferenças terão valores bastante próximos entre si, a menos que uma contenha posições de objetos ausentes ou em excesso. Um valor aceitável, para estimarmos o resíduo da diferença acima, é a variância de ϵ : σ_ϵ .

Se ambos os campos têm o mesmo número de objetos, e estes mantêm relação unívoca entre si, a variância da estatística que estabelecemos acima é:

$$\sigma_\epsilon = \sqrt{\sigma_\phi^2 - \sigma_\psi^2}$$

Para isso, admitimos que a variável aleatória ϵ_i , além de ter média nula, não se correlaciona com as posições dos objetos em ambos os campos. No caso dos números de objetos não serem idênticos, o valor de σ_ϵ será diferente. No entanto, para efeito de teste, podemos adotar esse valor obtido nas condições práticas como uma boa referência para se testar se as diferenças entre $(x_n - x_m)$ e $(X_n - X_m)$ indicam, ou não, o mesmo par de objetos nos dois campos em comparação.

3.5.8 A estimativa de distância e sua variância

O problema do *matching* entre dois campos está relacionado à determinação da distância entre os centros dos dois campos (assunto discutido na seção 3.5.6 sobre o estimador de distância entre dois campos), e à determinação tanto dos objetos comuns quanto dos objetos únicos a cada campo. Segundo o que foi discutido até agora nesta seção, podemos fixar como primeira aproximação para o estimador de distância e sua variância, respectivamente, a diferença entre a posição média dos centros dos campos e a diferença quadrática entre as variâncias dos centros dos campos.

¹¹Examinando-se as propriedades do POSS-II, não há situação que contradiga essa condição. Note-se que não são tratados os campos inseridos no disco galáctico. O tratamento desses casos ainda está para ser desenvolvido.

3.5.9 A identificação dos objetos

Estabelecidos esses valores, saberemos identificar se objetos em dois campos representam o mesmo objeto, testando se a diferença de posição está contida dentro de uma região definida por $[\bar{\Delta} - \sigma_c, \bar{\Delta} + \sigma_c]$, sendo $\bar{\Delta}$ e σ_c a distância e a variância estimadas. Se existe mais de um objeto, no campo em teste, que obedece às mesmas condições, escolhe-se o que possui o menor resíduo de distância.

3.5.10 Ambigüidade na identificação

Aparece aqui a seguinte limitação imposta: a variância deve ser muito menor que a menor distância entre dois objetos distintos de um campo, $\sigma_c \ll \min(|x_n - x_m|)$. Imaginemos que existam dois objetos tão próximos um do outro, que sua distância seja menor que a variância estimada. Suponhamos que um desses objetos tenha propriedades fotométricas tais que ele não seja detectado pelo programa de classificação no outro campo. Pelo procedimento descrito acima, os dois objetos identificados num campo estarão associados a apenas um objeto no outro campo, visto que os dois objetos no primeiro campo obedecem à condição vista em 3.5.9. Se associarmos apenas aquele cuja distância seja a menor, poderemos fazer uma identificação falsa, sem solução para retirarmos essa ambigüidade.

3.5.11 Identificação ponto a ponto

Tomemos dois campos onde pretende-se que existam objetos comuns, devidamente ordenados em ordem crescente de posição. Digamos que $F_1[N]$ ¹² é o vetor que representa os objetos do primeiro campo e $F_2[M]$ é o vetor que representa os objetos do segundo. Vamos definir dois ponteiros¹³ que apontam para os objetos dos campos: j_1 para o primeiro campo e j_2 para o segundo. O valor inicial de j_1 e j_2 é zero. Fixamos j_1 e verificamos se a distância $F_2[j_2] - F_1[j_1]$ obedece à condição estabelecida na Seção 3.5.9 para diferentes valores de j_2 até um limite baseado em um critério qualquer¹⁴. Caso não se encontre um objeto $F_2[j_2]$ obedecendo à condição, declara-se que $F_1[j_1]$ é um objeto único, isto é, sem um par no campo 2. O objeto $F_1[j_1]$ recebe um número de identificação único e é registrado no banco de dados. Em seguida j_1 passa a apontar para o próximo objeto. Tal processo é repetido até que encontre-se um par de objetos, $F_1[j_1]$ e $F_2[j_2]$ que obedeça à condição estipulada.

¹²O termo entre colchetes representa a dimensão do vetor.

¹³Por ponteiro entendemos valores inteiros que os índices dos vetores F_1 e F_2 podem assumir.

¹⁴Por exemplo, o número total de objetos no campo 2.

Nesse caso $F_1[j_1]$ recebe a mesma identificação de $F_2[j_2]$ e é registrado no banco de dados. Para cada par $F_1[j_1]$, $F_2[j_2]$ que obedece à condição em 3.5.9 obtemos a distância Δ_i que irá compor o valor médio e sua variância para proceder-se a uma nova procura, se for o caso, ou estabelecer os valores finais. O χ^2 das diferenças entre os pontos *matched*, descontadas do *offset* obtido, servirá de parâmetro para o cálculo da probabilidade em que os dois campos são semelhantes, através de (Press *et al.*, 1988, [47]):

$$Q(\chi^2|\nu) = Q\left(\frac{\nu}{2}, \frac{\chi^2}{2}\right)$$

com

$$Q(a, x) = \frac{1}{\Gamma(a)} \int_x^{\infty} e^{-t} t^{a-1} dt$$

a Função Gama Incompleta. O parâmetro ν vem a ser o grau de liberdade da estatística, que, em nosso caso, é o número de objetos *matched*.

3.5.12 O histograma das diferenças ponto a ponto

Vimos, pela expressão (3.5), que apenas as diferenças entre as posições dos objetos em comum de cada campo contarão para a correlação cruzada entre esses dois campos. Na prática, devido a variações aleatórias locais, teremos uma distribuição em que a distância entre os dois campos corresponderá à diferença mais provável nesta distribuição. Para determinarmos, portanto, a distância e, conseqüentemente, os pares nos campos, fazemos o histograma de todas as diferenças entre os campos. Esse histograma terá por valor mínimo e máximo as diferenças entre os extremos dos dois campos, e o passo (*bin*) será o menor valor entre a variância estimada (3.8) e a menor diferença em um campo (discussão em 3.5.10).

Sabemos que o valor correto do *offset* está dentro de um intervalo determinado pela largura do "*bin*". Para determiná-lo com maior precisão usamos um procedimento estatístico semelhante ao descrito em 3.5.2.2 e cujos detalhes descrevemos em 3.5.15.1.

O valor inicial do *offset* a ser aplicado à posição de cada objeto no campo tratado será obtido do histograma e corresponderá ao valor mais provável. Teoricamente, no caso da distribuição de objetos no campo ser homogênea, todos os pontos do histograma terão valor igual à unidade¹⁵. Em princípio, a contagem da diferença mais provável será igual à quantidade dos objetos *matched*. Essa propriedade pode ser utilizada como critério de

¹⁵ Admitimos, aqui, que o *bin* seja a distância média entre os objetos.

significância do método. É o que veremos a seguir.

3.5.12.1 Critério de significância de *matching*

É necessário determinarmos quando será possível estabelecer se dois campos podem ser considerados equivalentes. Esse critério deve indicar quando dois campos não se correspondem, no caso do usuário inadvertido tentar combinar dois campos de regiões diferentes. Esse mesmo critério deve indicar se um dos campos, ou mesmo ambos, estão em condições de serem comparados, ou se, por defeito na placa fotográfica ou no CCD, por condições atmosféricas ruins ou qualquer outro motivo, o procedimento de *matching* não pode ser efetuado, mesmo que os campos correspondam à mesma região no céu.

Por construção, o histograma das diferenças possui, em média, uma contagem por intervalo. No intervalo correspondente ao deslocamento entre os dois campos, a contagem chegará ao número de objetos comuns entre os dois campos. Pode haver, no entanto, *matchings* espúrios, isto é, objetos cujas posições fornecem diferenças iguais, ou aproximadas, o suficiente para que essas diferenças estejam no mesmo intervalo no histograma. Vejamos, em termos de probabilidades, a que valor essas contagens espúrias podem chegar.

Seja M o número de objetos distribuídos esparsamente em um campo. Façamos uma projeção das posições desses objetos em um eixo em uma dada direção. Seja N o número de pixels em que esse eixo se divide. A densidade linear de objetos nesse eixo será, então, M/N . Se considerarmos que a distribuição de objetos no céu é homogênea, essa será a probabilidade de encontrarmos um objeto em um dado pixel nesse eixo. Essa, também, é a probabilidade de um objeto coincidir com outro, quando o campo exibe um deslocamento arbitrário na mesma direção do eixo considerado.

Consideremos que as posições dos objetos estão sujeitas a distorções de caráter aleatório. Portanto essa probabilidade deve ser acrescentada de uma distribuição normal, onde σ deve ser menor do que o valor da menor distância entre dois objetos de um mesmo campo, para que a identificação não seja afetada por ambiguidades:

$$N_{\text{espurio}} = Pdx = \frac{M}{N} \frac{1}{\sqrt{2\pi}} \int_{-2\sigma}^{2\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right) dx = 0.95\sigma \frac{M}{N}$$

Os limites de integração são escolhidos de forma a cobrir 95% da faixa de distorção na placa/CCD. O inverso da densidade linear de objetos é a distância média entre os pontos em unidades de pixel. Se considerarmos que σ não pode ser maior que $\min(r_{ij})$ - a menor distância entre dois objetos em um campo -, a substituição de σ por $\min(r_{ij})$ vai estabelecer

o limite superior para a estimativa da probabilidade de um *matching* espúrio. Esse limite superior vai aumentar se substituirmos $\min(r_{ij})$ pela distância média. Assim:

$$N_{\text{espúrio}} = Pdx < 0.95 \sim 1$$

Em outras palavras esse é o valor que vai se somar à contagem no histograma das diferenças devido a *matchings* espúrios. Em suma, à contagem unitária no histograma das diferenças para dois objetos não correspondentes, pode-se somar não mais de uma contagem devido a *matching* espúrio, sob as condições que estamos considerando (ver 3.5.12.4).

A outra consideração que devemos fazer para definir a significância do *matching* é conjecturarmos sobre a fração de objetos que possuam par no campo ou catálogo de referência. Se considerarmos as condições ideais, em que tomamos dois campos cobrindo exatamente a mesma região do céu, discrepantes apenas por uma translação, podemos fazer uma previsão conservadora de 90% de *matching*, isto é, 90% dos objetos possuem, efetivamente, um par. Nessas condições, tomando um campo típico tratado pelo SKICAT, estimamos a significância do procedimento de *matching* em:

$$\frac{N_{\text{objetos matched}} - N_{\text{espúrio}}}{N_{\text{objetos}}} = \frac{90 - 1}{100} = 89\%. \quad (3.9)$$

A experiência mostra que os campos a serem comparados podem ter sido gerados em condições muito diferentes. Pode-se definir uma faixa no céu bem maior no catálogo de referência e, ao mesmo tempo, fazer comparações entre campos CCD e catálogos baseados em placas fotográficas. Campos observados com técnicas diferentes podem possuir diferentes comportamentos de distorção, saturação e sensibilidade mínima (*threshold*). Contudo, se forem mantidas condições aceitáveis de observação e qualidade do detector e/ou emulsão, a efetiva combinação de 50% dos pares de objetos parece ser um critério razoável para considerarmos que os dois campos em questão se correspondem satisfatoriamente. Como critério, portanto, estabelecemos que, no histograma das diferenças, a relação entre o maior pico e o número de objetos do campo a ser comparado deverá representar a significância do *matching*, e esse valor deverá ultrapassar 50% para que os dois campos sejam considerados *matched*.

3.5.12.2 Falha no processo de *Matching*

Em casos onde os dois campos representam duas regiões diferentes do céu, e foram submetidos por um processo de *matching* por inadvertência do operador, espera-se que o algoritmo reconheça esse fato. O programa reconhecerá então os dois campos representando populações de objetos distintos e tomará as providências pertinentes.

Pode acontecer que dois campos representando a mesma região tenham objetos comuns de forma esparsa e eventual, ou então que os campos tenham escalas de placa substancialmente diferentes. Esses são casos em que o algoritmo não funciona de maneira eficiente. Tratamos esses casos de forma não automatizada, mantendo o registro.

Esses casos representam as condições em que o algoritmo apresenta limitações. No primeiro, a solução é óbvia, bastando proceder de forma a tratar um dos campos como “novo”. Os seguintes não contemplam os campos do DPOSS-II.

3.5.12.3 *Matching* em campos densamente populad

Em campos como os do DPOSS-II, onde encontramos cerca de 6 objetos/*minarco*², o histograma das diferenças é contaminado por um efeito adicional. Além do pico representando o *matching*, há uma distribuição que decorre da obtenção dos módulos das distâncias entre os objetos. Chamaremos essa última distribuição de função de janela (*windowing function*) da correlação de campos homoganeamente preenchidos. A Figura 3.4 mostra um exemplo dessa distribuição.

Nessa figura vemos que não basta os dois campos *match* para que um histograma de diferenças bem definido apareça, com um pico correspondendo ao *offset*. De acordo com o intervalo de histograma (*binning*) escolhido, as contagens poderão assumir valores várias vezes superiores à contagem do *matching* esperada. Essa propriedade de campos densos representa um risco de “alarme falso” no algoritmo relativo a esse procedimento. Esse pode ser um problema crítico, se o algoritmo adotado em nosso método partir de valores de *binning* grandes para serem recursivamente diminuídos até se chegar a um valor aceitável, por exemplo quando a contagem do pico for inferior ao mínimo entre os números de objetos dos campos em questão (ver 3.5.15.1). É preciso, portanto, estudar com cuidado esse problema e tratá-lo convenientemente.

Podemos pensar que o histograma das diferenças é equivalente à contagem de objetos dentro de uma faixa circular em torno de um ponto (ver Figura 3.5). Essa faixa, cuja largura corresponde ao tamanho do *bin*, pode tanto varrer o campo de forma cheia, como pode ser seccionada pelas bordas do campo. Para facilitar a análise, vamos considerar que

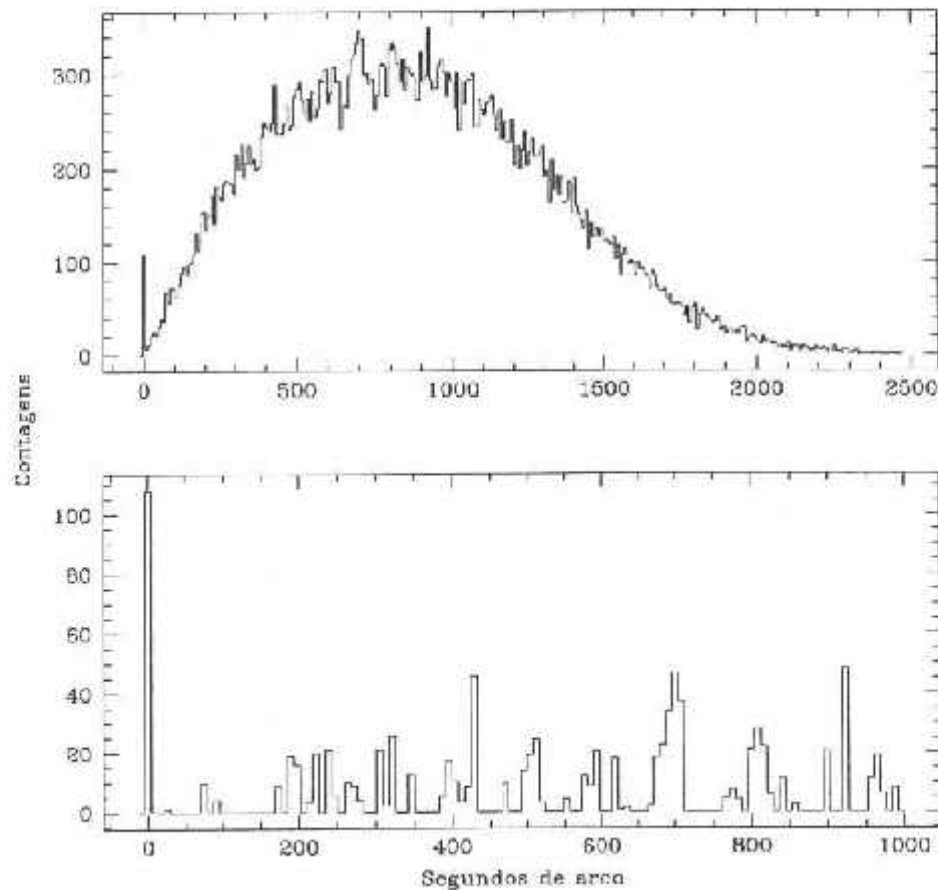


Figura 3.4: Histogramas das diferenças para os campos J823 e F823 no *footprint* 1,1.

O painel superior mostra o histograma das diferenças e o painel inferior mostra o mesmo histograma corrigido por um polinômio de grau 3. O *offset* entre os dois campos é da ordem de décimos de segundo tanto para α quanto para δ . O intervalo do histograma é de $24''.77$ (*binning*). A contagem para o *bin* 0 corresponde ao *matching* entre os dois campos. As outras contagens representam os *matchings* para a função janela da correlação de uma distribuição aproximadamente homogênea.

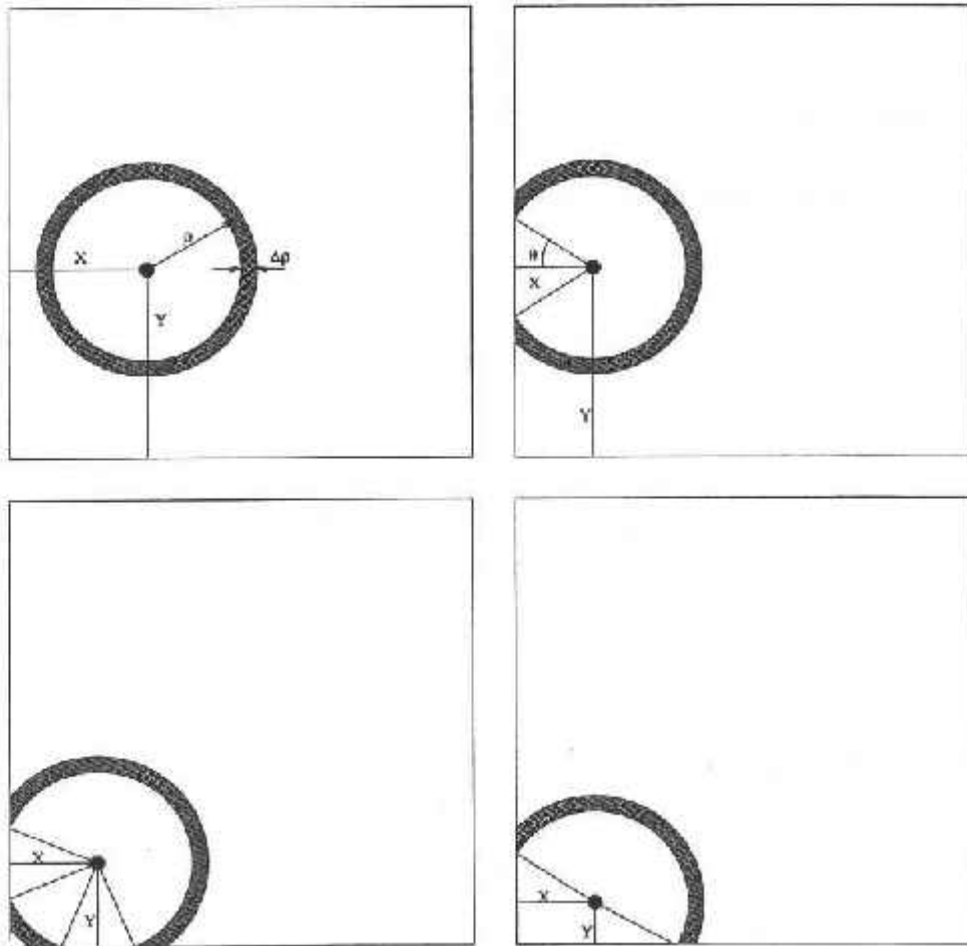
Figura 3.5: Faixas de *matching*.

Figura mostra as diferentes alternativas em que uma faixa circular pode varrer um campo quadrado.

os campos, tanto de referência quanto o de prova sejam quadrados de lado unitário. Vamos estudar apenas os casos em que o raio do círculo seja inferior à metade da dimensão do campo. Temos, portanto, quatro casos, sendo os dois últimos casos particulares de uma mesma situação. Esses casos estão representados na Figura 3.5 e vamos estudá-los a seguir.

Primeiro caso

O primeiro caso (superior esquerdo) tem por domínio:

$$\begin{aligned} 0 &\leq \rho \leq 1/2 \\ \rho &\leq X \leq 1 - \rho \\ \rho &\leq Y \leq 1 - \rho \end{aligned}$$

Sendo a população do campo de referência com N_1 objetos e o campo de prova com N_2 objetos, obtemos diretamente o resultado da função de distribuição de *matchings* para esse caso:

$$F_1(\rho) = 2\pi N_1 N_2 \Delta \rho (1 - 2\rho)^2 \rho$$

Segundo caso

O segundo caso (superior direito) tem por domínio:

$$\begin{aligned} 0 &\leq \rho \leq 1/2 \\ 0 &\leq X \leq \rho \\ \rho &\leq Y \leq 1 - \rho \end{aligned}$$

E a integração pode ser feita segundo:

$$\begin{aligned} F_2(\rho) &= 4 \times 2N_1 N_2 \Delta \rho \left[\int_0^\rho dX \int_\rho^{1-\rho} dY \left(\pi - \arccos\left(\frac{X}{\rho}\right) \right) \right] \rho \\ &= 8N_1 N_2 \Delta \rho (\pi - 1)(2\rho - 1)\rho^2 \end{aligned}$$

A função considera as quatro possibilidades dessa geometria.

Terceiro caso

Observa-se que este é um caso em que a condição anterior se aplica duplamente. O domínio é:

$$\begin{aligned} 0 &\leq \rho \leq 1 \\ \frac{\sqrt{2}}{2}\rho &\leq X \leq \rho \\ \frac{\sqrt{2}}{2}\rho &\leq Y \leq \rho \end{aligned}$$

Com a integração sendo feita:

$$\begin{aligned} F_3(\rho) &= 4 \times 2N_1N_2\Delta\rho \left[\int_{\frac{\sqrt{2}}{2}\rho}^{\rho} dX \int_{\frac{\sqrt{2}}{2}\rho}^{\rho} dY \left(\pi - \arccos\left(\frac{X}{\rho}\right) - \arccos\left(\frac{Y}{\rho}\right) \right) \right] \rho \\ &= 2N_1N_2\Delta\rho \left[5\pi + 4 - \sqrt{2}(3\pi + 4) \right] \rho^3 \end{aligned}$$

Quarto caso

Finalmente, o quarto caso refere-se a um caso particular do terceiro, sem a "ponta" da circunferência no canto inferior. O domínio se restringe a:

$$\begin{aligned} 0 &\leq \rho \leq 2 - \sqrt{2} \\ 0 &\leq X \leq \frac{\sqrt{2}}{2}\rho \\ 0 &\leq Y \leq \frac{\sqrt{2}}{2}\rho \end{aligned}$$

E a integração resulta em:

$$\begin{aligned} F_4(\rho) &= 4N_1N_2\Delta\rho \left[\int_0^{\frac{\sqrt{2}}{2}\rho} dX \int_0^{\frac{\sqrt{2}}{2}\rho} dY \left(\pi - \arccos\left(\frac{X}{\rho}\right) + \arccos\left(\frac{Y}{\rho}\right) \right) \right] \\ &= 2\pi N_1N_2\Delta\rho\rho^3 \end{aligned}$$

Resultado

A função de distribuição final é a soma das quatro funções: $F_1 + F_2 + F_3 + F_4$ o que nos dá um polinômio de 3^o grau:

$$2N_1N_2\Delta\rho \left[\left(2\pi + 12 - \sqrt{2}(3\pi + 4) \right) \rho^3 - 4\rho^2 + \pi\rho \right] \quad (3.10)$$

Sabemos, portanto, que a distribuição no histograma das diferenças é um polinômio de 3º grau, pelo menos até a metade do campo que está sendo estudado. Os coeficientes variam de campo para campo porque a distribuição de pontos não é exatamente homogênea, como foi suposto no desenvolvimento do cálculo acima. Para a correção desse efeito, portanto, os dados obtidos devem ser considerados para o histograma com um polinômio de 3º grau. Como podemos notar, a função descrita na expressão (3.10) é linearmente dependente do tamanho do *bin*, logo seria suficiente ajustarmos o polinômio apenas uma vez. As variações na forma da função, a partir de então, seriam desprezíveis.

3.5.12.4 Limites do método

A identificação de um campo estelar é possível se considerarmos que a distribuição de objetos no campo é única e específica do campo sendo considerado. Se existe semelhança muito grande entre dois campos pode-se identificar erroneamente um campo. É comum os não astrônomos confundirem a chamada "falsa cruz" (a vela da constelação Caravela) com o Cruzeiro do Sul. Nesses casos costumamos chamar a atenção do observador para a estrela "intrometida" que diferencia o verdadeiro Cruzeiro do Sul de sua "imitação". Aos olhos do observador não treinado, há uma uniformidade entre as distribuições dos dois campos.

Os métodos de *matching*, inclusive o do histograma das diferenças, aqui apresentado, são diferentes maneiras de comparar formas dos campos, isto é, de como os objetos estão distribuídos no campo. Como tal, dependem da **não uniformidade** da distribuição dos objetos no campo. O próprio método do histograma das diferenças falha completamente ao se deparar com um campo uniformemente distribuído. Por exemplo, a menor e a maior diferença seriam iguais, e a determinação dos intervalos de acumulação (*binning*) se tornaria singular. A Figura 3.6 nos mostra um caso de distribuição de pontos perfeitamente regular (no caso, um quadrilátero regular). Para fazer o *matching* de dois campos assim construídos, os métodos de Groth (1986, [23]), Murtagh (1992, [45]) e Stetson (1989, [55]) não conseguem separar os pares de objetos (Figura 3.6). O método do histograma das diferenças tampouco é capaz de identificar os pares. Nessa figura, vemos a projeção dos objetos em um eixo com uma pequena inclinação. Mesmo assim, apenas será acessível a identificação de famílias de pares possíveis.

Para que haja viabilidade do método, é preciso que

1. o campo seja não uniforme;
2. a menor e a maior diferença de posições sejam significativamente diferentes.

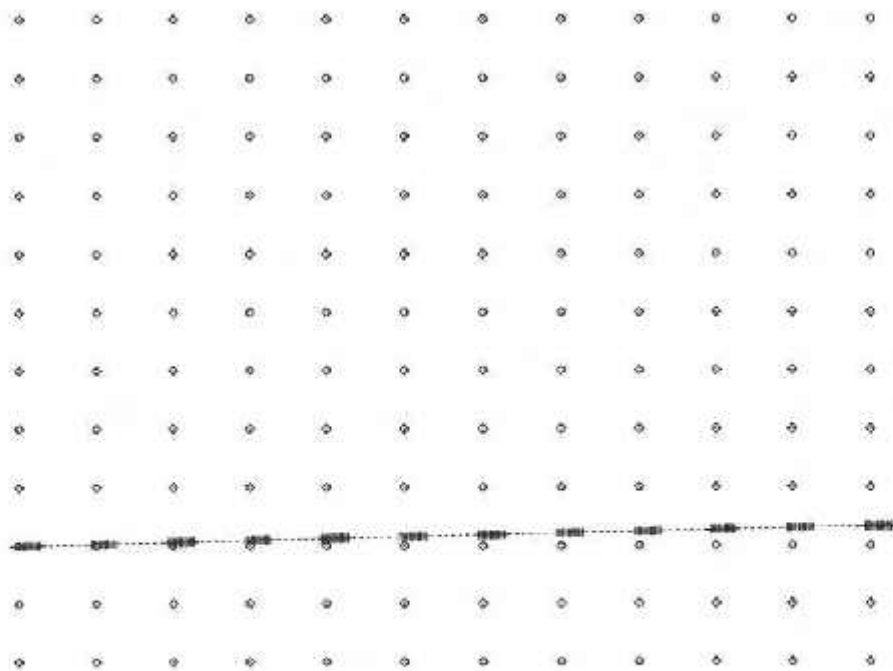


Figura 3.6: Distribuição uniforme.

Exemplo de uma distribuição regular de pontos (no caso, nos vértices de um quadrilátero) projetada em um eixo com pequena inclinação. Se não tivermos singularidade na distribuição de intervalos (*binning*), na melhor das hipóteses teremos uma família de *matchings* possíveis.

Em um campo com distribuição uniforme de objetos, os pontos se distribuem segundo os vértices de um poliedro regular. Nesse caso, como vemos na Figura 3.6, não será possível identificar uma estrutura única, característica em que todos os métodos se baseiam. Para que tenhamos um indicador da confiabilidade de um *matching*, portanto, é preciso que, de alguma forma, possamos estimar o “grau” da não uniformidade na distribuição de objetos em um campo.

Uma forma de verificarmos se existe uniformidade no campo é observarmos o histograma das diferenças de um campo regular com ele mesmo, o que, na prática, corresponde à função de autocorrelação dessa distribuição. A Figura 3.7 mostra esse histograma. Se não existisse qualquer uniformidade, o histograma teria um “pico” no ponto zero, representando a soma de todas as diferenças das posições dos objetos com eles mesmos, enquanto que para outros pontos do gráfico, a contagem não passaria da unidade. Na Figura 3.7 vemos que existe uma família de soluções em que o campo considerado *match* com ele mesmo. Nesse caso, o algoritmo de *matching* apresenta problemas.

A análise do histograma das diferenças vista na Seção 3.5.12.1 mostra também que, a razão entre os pontos máximos do histograma, além de servir para indicar a concordância da região do céu entre os dois campos, também serve para indicar a factibilidade do procedimento de *matching* dos campos, obedecendo às duas condições de viabilidade listadas acima.

Do ponto de vista estatístico usamos o teste de Kolmogorov-Smirnov (K-S) (Press *et al*, 1988, [47]) para verificar se duas amostras são oriundas de uma mesma população. Obtemos dois parâmetros: D , cuja métrica determina a distância entre os conjuntos determinados pelas amostras, e Q nos dá o nível de significância. O parâmetro D é obtido como o valor máximo da distância entre as distribuições acumuladas, enquanto que Q é função de D e do número de pontos das amostras. O nosso propósito é determinar se a distribuição de pontos projetados em um eixo segue ou não uma lei uniforme. Não podemos comparar essa distribuição com funções de origem aleatória, tais como a lei de Poisson ou de Gauss porque não há razão para considerarmos que os objetos em um campo estelar seguem algum processo estocástico. Portanto, a alternativa é escolher como hipótese nula (H_0) a de que os pontos seguem uma lei de distribuição uniforme. Baixos valores de D e Q perto da unidade implica não podermos rejeitar H_0 , e portanto, não podemos admitir que a distribuição é uniforme. Se aceitarmos que a distribuição é uniforme (hipótese nula H_0), a probabilidade de que se trata de um “alarme falso” é de $100Q\%$. Quanto maior for o valor desse parâmetro, maior será a chance de admitirmos erroneamente que a distribuição

é *uniforme*. Obviamente, na medida em que Q diminui, aumentam as chances de não incorreremos nesse erro. Se, por outro lado, obtivermos altos valores de D e baixos de Q , teremos grandes chances de rejeitarmos H_0 sem o risco de errarmos em nossa decisão. Essa última condição é a ideal para que os algoritmos de *matching* sejam eficazes e sem ambiguidades. A Tabela 3.1 mostra alguns resultados de testes feitos em diferentes tipos de distribuição. Em todos os casos, salvo o da distribuição uniforme, o algoritmo de *matching* obteve sucesso. É possível adotar, portanto, as condições $D > 0.15$ e $Q < 0.9$ para rejeitarmos H_0 . Tendo em vista que alguns catálogos são construídos segundo critério de distribuição espacial e que o método histograma das diferenças falha apenas quando a distribuição estudada é **efetivamente uniforme**, como veremos a seguir, os valores de D e Q podem ser adotados numa eventual pré-condição antes de aplicarmos o método.

Há, no entanto, distribuições uniformes que, não obstante a estatística K-S indicar baixo D e alto Q , são sensíveis a diferentes *offsets*, garantindo o sucesso de *matching*. É o caso em que a distribuição espacial de pontos se dispõe segundo diferentes poliedros, ou, em último caso, em poliedros de tamanho variável. Este último é o caso da distribuição de estrelas do catálogo FK5 no campo de coordenadas $-30^\circ < \delta < 30^\circ$ e $3^h < \alpha < 6^h$ (Figura 3.8). Os valores da estatística K-S para esse campo são: $D = 0.11$ e $Q = 0.999$ para declinação e $D = 0.15$ e $Q = 0.94$ para ascensão reta. Apesar desses valores sugerirem que devemos rejeitar os procedimentos clássicos de *matching*, o histograma das diferenças consegue distinguir bem as estruturas estabelecendo um valor único e inequívoco para o eventual *offset*, respeitados, é claro, os limites impostos para a aplicabilidade desse método. A Figura 3.8 mostra muito bem como, no caso do campo descrito acima do catálogo FK5, o histograma das diferenças pode determinar sem ambigüidades o ponto de *matching*.

3.5.13 Aplicações em casos reais

3.5.13.1 Catálogos FK5 e BSC5

Testamos o método do histograma das diferenças para fazer o *matching* entre os catálogos FK5 e BSC5. Esse *matching* pode ser testado, particularmente para esses dois catálogos, porque o BSC5 faz referência ao FK5 nas estrelas comuns.

Usamos o FK5 como catálogo de referência e tomaremos os objetos de BSC5 como os candidatos a serem identificados e catalogados. Dividimos os catálogos em oito regiões de forma a cobrir superfícies no céu com iguais ângulos sólidos. As regiões foram definidas se-

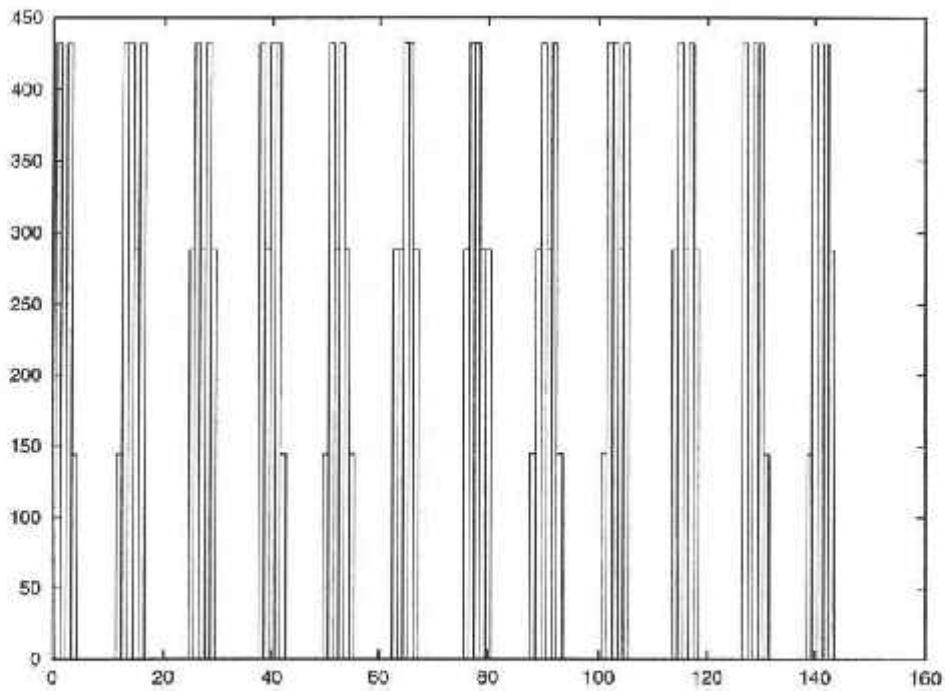


Figura 3.7: Histograma das diferenças do campo uniforme.

Gráfico do histograma das diferenças do campo de pontos distribuídos segundo os vértices de um quadrilátero regular em um eixo inclinado (ver Figura 3.6). Esse histograma representa a função de autocorrelação dessa distribuição. O valor máximo da frequência não representa, nesse caso, o número de pontos no campo, senão o número de vezes em que a mesma estrutura se repete, no campo.

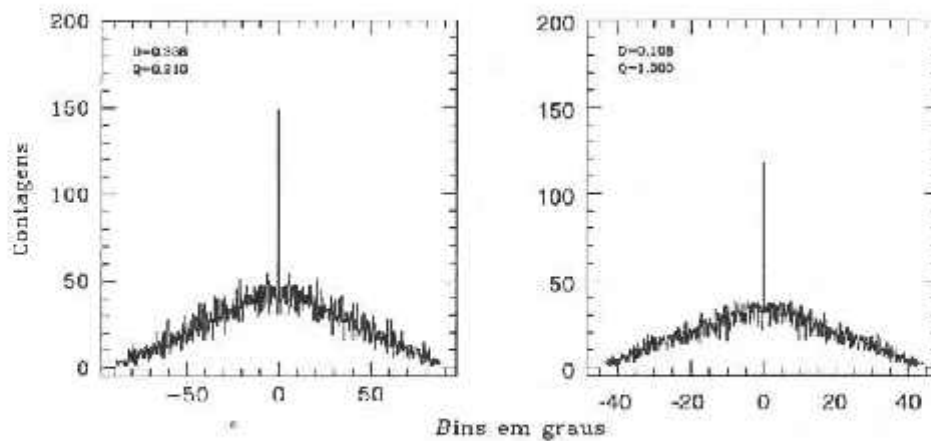


Figura 3.8: Histogramas de distribuição e das diferenças.

Histograma das diferenças, respectivamente para os campos $30^\circ < \delta < 90^\circ$ e $12^h < \alpha < 18^h$ (à esquerda), $-30^\circ < \delta < 30^\circ$ e $3^h < \alpha < 6^h$ (à direita) do catálogo FK5. Os valores da estatística de Kolmogorov-Smirnov estão no canto superior esquerdo dos painéis superiores. Essa estatística, cuja hipótese nula (H_0) adotada é a coincidência com distribuição uniforme, é sensível à repetição de estruturas no campo. Vê-se do valor $Q = 1.00$, no painel à direita, que a distribuição equatorial de estrelas no FK5 é bastante uniforme, sugerindo um critério na escolha de estrelas fundamentais. O histograma das diferenças, por outro lado, mostra-se insensível a essa uniformidade. Para o campo polar (painel à esquerda), a relação entre o pico e o segundo maior ponto é 2.7, enquanto que para o campo equatorial, essa relação é 3.0.

Distribuição	D	Q	H_0
Uniforme (visto acima)	0.069	1.00	Mantido
Série aleatória (média)	0.159	0.870	Rejeitado
Exponencial (média)	0.650	0.000147	Rejeitado
Gaussiana (média)	0.377	0.0856	Rejeitado
FK5 (média em δ)	0.188	0.700	Rejeitado
BSC5 (média em δ)	0.258	0.490	Rejeitado
NGC2000 (média em δ)	0.362	0.245	Rejeitado
Aglomerados de Abell (média em δ)	0.302	0.301	Rejeitado

Tabela 3.1: Valores da estatística Kolmogorov-Smirnov para alguns campos característicos.

Note-se que o catálogo FK5 apresenta, em média, uma distribuição razoavelmente uniforme de estrelas fundamentais. Como critério, adotamos $D > 0.15$ e $Q < 0.9$ como condição para rejeitarmos a hipótese nula.

gundo a Tabela 3.2. Tomando-se o FK5 como referência, a divisão assim definida seleciona cerca de 100 objetos por região.

Valendo-se dos critérios citados em 3.5.12.1 o método foi capaz de realizar o *matching* com 100% de sucesso. Como ilustração, reproduzimos abaixo a listagem do relatório do programa na região entre ($6^h \dots 9^h, -30^\circ \dots 30^\circ$) e colocamos na Figura 3.9 os histogramas das diferenças em α e δ .

```
87 rows selected from FK5
739 rows selected from BSC5
Offset: RA= -0.007 DEC= 0.013
Offset variance: RA= 0.062 DEC= 0.307
86 matched objects
1 single objects in FK5
653 single objects in BSC5
0 mismatched objects
0 lost objects
```

3.5.14 Tempo de Cálculo

Foram feitos testes com o programa, na versão para integrar o processo de carga no SKIC-COSMO. Esses testes indicam que o tempo de cálculo independente do caminho toma-

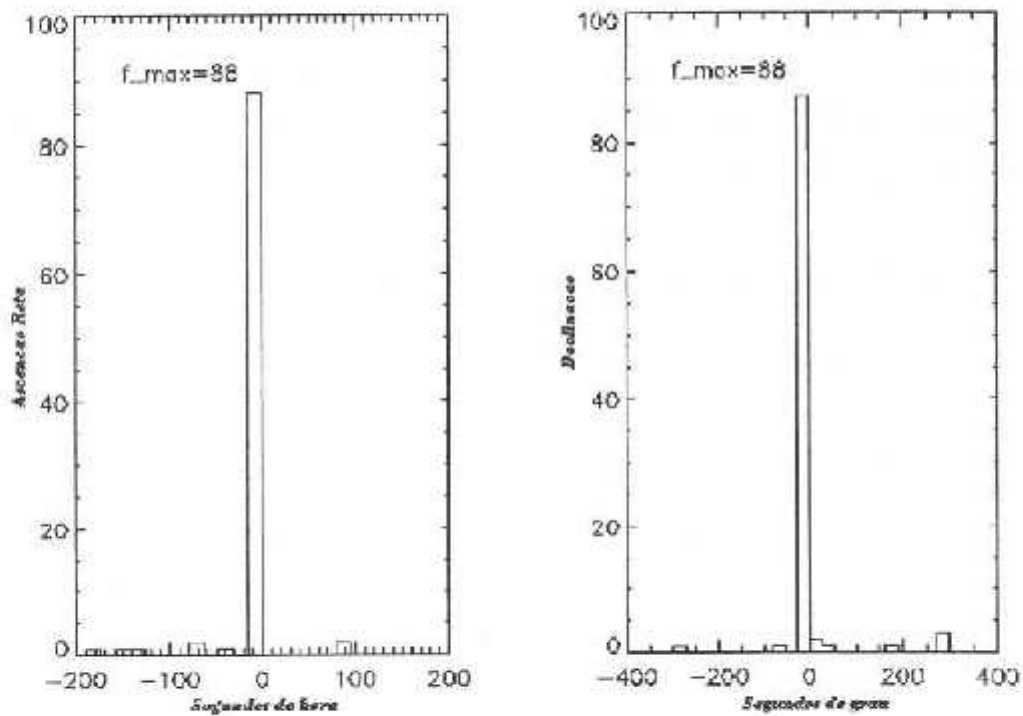


Figura 3.9: Histograma das diferenças do *matching* entre o BSC5 e FK5 para a região ($6^h \dots 9^h, -30^\circ \dots 30^\circ$).

A frequência máxima para ambas direções foi 88. Tendo em vista que foram *matched* 86 objetos, haveria dois falsos *matches* que o algoritmo se encarregou de descartar graças aos critérios de significância.

δ entre	$\Delta\alpha$
-90,-30	4^h
-30,30	2^h
30,90	4^h

Tabela 3.2: Regiões no céu cobrindo números aproximadamente iguais de objetos.

do, tem dependência linear com o número de objetos nos campos sendo processados: $T \propto N_1 N_2$. Apesar da dependência do método com o quadrado do número de objetos, o procedimento mostrou-se muito mais rápido do que aqueles apresentados aqui. O processo de *matching* para um campo de placa fotográfica do POSS-II executado em um Pentium 120MHz é realizado em 45min. Em contraposição ao mesmo processo executado pelo SKICAT, que é de 4320min.

3.5.15 O Processo de *matching* em duas dimensões

O procedimento de *matching* adotado no SKICAT por Weir (1994) leva em conta as distâncias efetivas, isto é, considerando o espaço de duas dimensões do campo, ao contrário de nosso método que considera, inicialmente, o módulo das distâncias. Pode-se dizer que tal procedimento é “seguro”, pois o módulo da distância é insensível às eventuais rotações entre um campo e outro. No entanto, uma questão surge: qual é a magnitude de uma rotação entre uma pose e outra? Em uma placa fotográfica, que acreditamos ser a situação mais delicada, supondo uma granulação de $20\mu m$ e uma extensão de $20cm$, é necessário fazer-se uma rotação de $20''$ para deslocarmos um objeto de um grão de emulsão a outro. Esse valor é ainda maior, se considerarmos o *seeing*.

Consideramos improvável que um campo esteja rotacionado em relação aos outros campos de referência. Um indicativo dessa improbabilidade é a completa desconsideração desse efeito nas análises de *offset* feitas no trabalho de Weir (1994, [66]). Como consequência, consideramos que a rotação é um processo desprezível entre os campos. Se não existe rotação, não há possibilidade de duas coordenadas inverterem posições (*swap*) numa seqüência, seja ela crescente, seja decrescente, quando se passa de um campo a outro. Dessa forma, está garantido que a seqüência de objetos não é modificada em qualquer das duas coordenadas celestes, a menos de deslocamentos aleatórios, que admitimos serem menores que a menor diferença de coordenadas.

3.5.15.1 Algoritmo

As distâncias são obtidas com o módulo do raio vetor entre o objeto a ser estudado e o objeto de referência. Dessa forma, todos os procedimentos são construídos para o estudo da distribuição unidimensional.

O algoritmo usado para o método do histograma das diferenças baseia-se, essencialmente, em dois procedimentos recursivos. O primeiro determina o *bin* onde se encontra o

offset, e o segundo procedimento decide os prováveis pares e estima, de forma robusta, o *offset* exato.

Para levar a cabo o primeiro procedimento, é preciso adotar alguns valores iniciais, tais como o *offset* e o tamanho do *bin*. Para o *offset* adotamos o valor inicial zero. Por experiência, de forma geral, essa primeira estimativa não compromete a convergência do método. Para o tamanho do *bin*, estabelecemos, primeiramente, a expectativa de um *offset* não superior à metade do tamanho do campo. Em geral, um *footprint* de um campo do DPOSS-II possui dimensões de cerca de 30'. Portanto, pelo algoritmo adotado, o *offset* não pode ultrapassar a casa dos 15'. Isso não foi observado e, se for o caso, indicaria que outros condicionantes também estariam comprometidos, sobretudo os coeficientes do polinômio astrométrico, o que quer dizer que o método do histograma das diferenças não se aplicaria. Uma vez estabelecido o limite de *offset*, como sendo a metade de um campo, adota-se o tamanho do *bin* como o de $\Delta D/100$, onde ΔD corresponde ao valor da metade do campo. Tipicamente, o valor inicial do *bin* é, portanto, de 18". Em seguida é necessário estimar o polinômio de 3^o grau correspondente ao da distribuição apresentada na Figura 3.4 (ver 3.5.12.3). Para obter-se o *offset*, uma vez ajustado o polinômio, aplica-se uma correção ao histograma baseada nesse polinômio.

Uma vez obtido o *offset* e o intervalo em que esse pode variar, inicia-se um outro processo recursivo no qual, a partir desses dois parâmetros, escolhe-se os pares prováveis nos dois campos. Se houver ambigüidade, ou seja, mais de um candidato a *matching* para um dado objeto, escolhe-se aquele que mantenha a menor distância com esse objeto. Definidos os pares, obtém-se o *offset* médio que é a média das distâncias entre os pares escolhidos. Com o novo *offset* reinicia-se o processo de escolha de pares, agora definindo as distâncias aceitáveis dentro de um critério de robustez, por exemplo, 2σ . Aqui, são descartados os "falsos" pares, que por ventura foram selecionados com os parâmetros iniciais. Esse processo converge quando o valor do *offset* varia menos do que uma certa tolerância, que adotamos como sendo 10^{-6} . Ao final do procedimento, portanto, temos o *offset* para o campo escolhido e os pares do *matching*, isto é, os objetos que foram identificados como pertencentes também ao campo de referência.

Abaixo listamos o algoritmo que governa o programa de *matching*. Esse programa, na prática, chama-se `match_cat` e foi desenvolvido em linguagem "C".

Program "match_cat"

begin

```

 $\Delta D \leftarrow SizeField/2$ 
 $BinSize \leftarrow \Delta D/100$ 
procedure find_rough_offset
  begin

    Build histogram
    if (first iteration) then Fit  $Ax^3 + Bx^2 + Cx$  to histogram
    Correct Histogram for fitted polynom
    Find histogram peak and deduce offset
    Figure out pairs in both field that fit offset number of bins
    Repeat until peak histogram  $< \min(N_{objs})$ 
  end

procedure find_exact_offset
  begin

    Average offset from guessed pairs
    Find  $\sigma_{offset}$ 
    Redefine pairs
    Repeat until  $\Delta offset/offset < 10^{-6}$ 
  end

procedure sel_by_ra_dec
  begin
    Average offset for  $\alpha$  and  $\delta$ 
    Tag pairs according to criteria  $\Delta\alpha - \alpha_{offset} \leq \sigma_\alpha$  and  $\Delta\delta - \delta_{offset} \leq \sigma_\delta$ 
    Repeat until  $\Delta\alpha_{offset}/\alpha_{offset} < 10^{-6}$  and  $\Delta\delta_{offset}/\delta_{offset} < 10^{-6}$ 
  end

end Program

```

Podemos ver que existem 3 *loops* no programa. O objetivo é refinar cada vez mais o *matching*, através de três diferentes critérios: determinação pelo histograma das diferenças (*loop* 1), refinamento pelo valor absoluto do *offset* (*loop* 2) e, finalmente, a definição da direção do *offset* ao mesmo tempo em que refina-se mais ainda o *offset*.

3.5.15.2 Critério para o intervalo de confiança

No algoritmo mostrado em 3.5.15.1 nos *loops* 2 e 3, é preciso definir o intervalo pelo qual admite-se que o par é um candidato a objeto *matched* ou não. Para isso, define-se um fator multiplicativo que, em última análise vem a ser $k.\sigma$, determinando o critério de robustez da estatística. Alguns valores para esse fator foram usados e a experiência mostrou que o valor de k ideal é da ordem de 2, estabelecendo, assim, um critério de que 95% dos pontos são considerados dentro do intervalo de confiança.

3.5.16 Comparação entre Métodos

Procedemos testes entre o sucesso no *matching* pelo histograma das diferenças e o método do SKICAT (Weir, 1994, [66]). A população *matched* para um e para o outro método, em vários experimentos, diferem de uma média de menos de 1%. A população dos objetos que são discordantes nos dois métodos correspondem àquela descrita em 3.5.9, isto é, objetos muito próximos em um campo a serem *matched* com um único em outro campo. Há uma ambigüidade nesses casos, pois a distância entre os objetos é da ordem da variância da estimativa do *offset*.

3.6 A Estratégia de Transferência

Para o procedimento da transferência dos dados armazenados no SKICAT para o SKIC-COSMO foi necessário contar com as seguintes ferramentas:

1. *Driver* para leitura dos dados no SKICAT;
2. *Driver* para leitura dos dados no SKICCOSMO;
3. Ferramenta para carga de dados no SKICCOSMO.

As duas primeiras ferramentas são necessárias porque não se trata simplesmente de retirar os dados do SKICAT e colocá-los no SKICCOSMO, visto que há uma mudança de paradigma entre um banco de dados e outro (ver em 3.1.1 e 3.2). Isso significa dizer que os dados devem ser coletados do SKICAT, comparados com o que já foi carregado no SKICCOSMO, reorganizados para integrarem outras tabelas e, somente então, serem definitivamente carregados no SKICCOSMO. Em consequência, deve ser desenvolvida uma interface conectando os dois paradigmas.

Para a construção dessa interface foi escolhida a linguagem *Perl* por diversas razões. As principais são, além da facilidade de manipulação de textos, matrizes e "hashes", a possibilidade de instalação de *Drivers* para o Sybase e, simultaneamente, para o Informix, sem qualquer ônus, pois esses módulos são de domínio público. Para a instalação correta dos *drivers* são necessárias duas bibliotecas de software: o *Open Client*, da Sybase, do lado do servidor do SKICAT, e o *ESQL*, da Informix, do lado do servidor do SKICCOSMO. Essas bibliotecas permitem fazer a ligação entre as nossas aplicações e os respectivos motores dos bancos de dados.

Para as transferências dos dados, portanto, armou-se uma triangulação de servidores em rede. O módulo Perl, chamado **loadfields.pl** é executado na servidora do Sybase. Esse programa executa as seguintes tarefas:

1. Recolhe os dados do campo guardado no SKICAT. Para isso o programa seleciona os objetos na seguinte hierarquia: a) objetos "regulares", isto é, marcados como identificados e não saturados; b) resto dos objetos. Os objetos regulares recolhidos são colocados na seguinte ordem: 1) estrelas; 2) galáxias;
2. Determina os limites em α e δ do campo para definir a área em que será processado o *matching* com os objetos já registrados no SKICCOSMO. Em seguida essa área é ampliada de um valor, em torno de $1'$ em todas as direções;
3. Executa, remotamente, um programa alojado no servidor ESQL chamado **data-probj**. Trata-se de um outro programa em Perl que lança uma consulta ao SKICCOSMO recuperando todos os objetos regulares já registrados nesse banco de dados, correspondendo à área descrita no passo 2;
4. Faz o *matching* entre os dois campos. O *offset* é determinado pela comparação dos objetos exclusivamente estelares nos dois campos. Em seguida, verifica-se o *matching* dos objetos resultantes. O critério para esse *matching* é exclusivamente de posição de forma que é possível um objeto já registrado como "galáxia" no SKICCOSMO, seja *matched* com um objeto estelar no novo campo, e vice-versa;
5. Armazena todos os objetos do campo em arquivos a serem carregados no SKICCOSMO. Aplica-se a todos o *offset* determinado no passo 4;
6. Executa, remotamente, o programa **loadfiles**, um *script shell* instalado no servidor do Informix, que carrega todos os dados dos arquivos no SKICCOSMO.

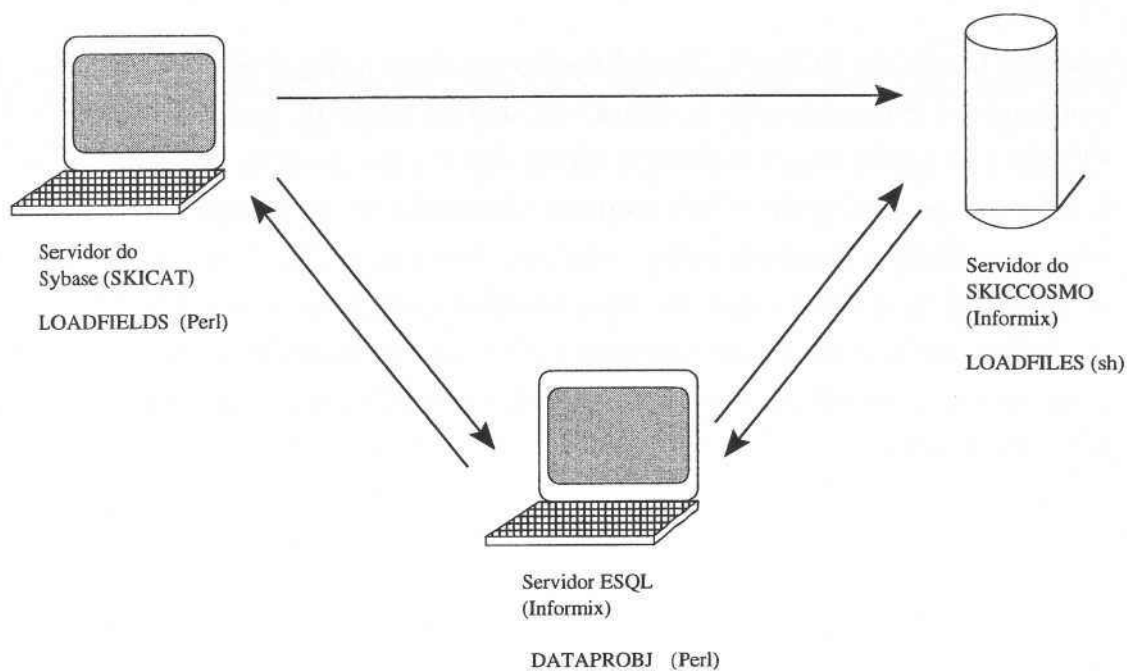


Figura 3.10: Rede em três pontos para a carga do SKICCOSMO

A Figura 3.10 ilustra o esquema de carga do SKICCOSMO. Os módulos executados em cada servidor e sua respectiva linguagem estão em maiúsculo abaixo da descrição de cada servidor.

Quando os dados referem-se a uma placa do POSS-II o programa **loadfields** divide a imagem em uma matriz 13×13 obedecendo à divisão dos *footprints* feita por ocasião do processamento da imagem digitalizada da placa pelo SKICAT. Os dados de um *footprint* são recuperados do SKICAT e reorganizados para adequá-los ao SKICCOSMO. De posse dos dados dos objetos de um dado *footprint*, o programa **loadfields** faz uma consulta ao SKICCOSMO procurando por objetos já registrados na mesma região do céu definida pelo *footprint*. Ao fazer essa consulta o programa acrescenta um minuto de arco aos extremos dessa região. Assim garante-se que não se perca *matchings* dos objetos nos extremos do *footprint*. Em seguida, faz-se o *matching* do campo proveniente do SKICAT com o proveniente do SKICCOSMO, para em seguida registrar os novos dados no SKICCOSMO. O mesmo procedimento é repetido até que todos os *footprints* tenham sido carregados no SKICCOSMO. No caso de uma imagem CCD, a imagem é submetida por inteira a esse processo, sem divisões.

A carga de cada placa digitalizada do POSS-II dura um tempo entre 6 horas para o campo da primeira banda, cerca de 11 horas para a segunda banda e 18 horas para a

terceira banda do POSS-II. Essa diferença se deve, principalmente, ao fato que quando um campo é carregado pela primeira vez, não há perda de tempo em consultas ao SKIC-COSMO procurando por objetos já carregados e o conseqüente processo de *matching* com o campo a ser carregado. Cada conjunto de placas de um campo carregado dá partida a um procedimento chamado *update statistics* (ver em 1.4.2.13) no Informix, com o intuito de atualizar os índices e acelerar sobremaneira as operações com o banco de dados. Esse procedimento leva um tempo que varia com o volume de dados armazenados. Atualmente, com cerca de 16×10^6 objetos armazenados, o procedimento de atualização dos índices leva cerca de 4 horas.

Capítulo 4

Operando com o SKICCOSMO

4.1 O Ambiente de Consulta

4.1.1 Introdução

Um banco de dados relacional é econômico na armazenagem de dados mas por outro lado a consulta pode ser muito complexa. Em outras palavras, se por um lado um banco de dados relacional é o ideal tanto em termos de economia de memória e tempo, quanto de fácil expansão, por outro lado, a construção da consulta não é evidente e em muitos casos - dependendo do número de tabelas envolvidas - tão extensa que compromete o ganho em eficiência por conta de, por exemplo, erros na programação da consulta.

A consulta no SKICCOSMO serve-se freqüentemente da chamada “auto-junção” que é a referência auto recorrente de uma mesma tabela, a saber, “DPOSSII_FEATURES”, pois, com freqüência necessitamos de comparações de diferentes atributos de um mesmo objeto referenciado nessa tabela.

No intuito de facilitar a operação de consulta, foi criado um “tradutor”, um programa com a função de gerar uma instrução em “SQL” a partir de instruções simples. É o que chamamos de “interface” usuário. O objetivo dessa “interface” é evitar que o astrônomo se veja obrigado a construir consultas em “SQL” arriscando gerar uma violação na integridade do resultado da consulta, isto é, produzindo resultados sem qualquer confiança quanto ao seu significado.

Esse “tradutor” é capaz de interpretar um conjunto de instruções sob convenções que se preferiu chamar de “SkiccQL”, ou “Linguagem de Consulta ao SKICCOSMO”. Sob essa linguagem, pretende-se disponibilizar um conjunto de instruções intuitivas ao astrônomo

tendo, este, que se submeter a algumas regras que serão descritas no decorrer desse capítulo. O SkiccQL foi construído para permitir que:

1. O astrônomo se comunique com o SKICCOSMO munido de algumas ferramentas cujas regras são facilmente compreendidas pois são parecidas com a linguagem comum de programação, envolvendo conceitos associados à astronomia;
2. Evite-se programar diretamente em "SQL" o que exigiria conhecimentos da teoria relacional. Essa condição traria dois inconvenientes na consulta ao SKICCOSMO: a) exigindo conhecimentos adicionais para fazer uma consulta, deixa-se o banco de dados em desvantagem frente a outros bancos de dados oferecidos na INTERNET, cujas interfaces usuário são muito mais amigáveis; b) abre condição para se cometer erros que comprometem a operação do SKICCOSMO. Uma consulta mal feita pode onerar sobremaneira o motor do banco de dados, além do resultado não ter garantia de ter o significado científico esperado.

Não se tem a pretensão de gerar um "compilador" FORTRAN para interpretar uma consulta de um astrônomo pois isso demandaria meses de trabalho de diversos programadores. Além disso, o FORTRAN é inadequado para esse propósito. Optou-se por gerar comandos simples e algumas extensões para facilitar a consulta.

Por outro lado, garante-se que os dados que retornam do banco de dados possuem as propriedades requeridas, que sejam confiáveis e íntegros, isto é, o usuário saberá que os dados recuperados representam a integridade dos dados com as propriedades definidas dentro do domínio esperado.

4.1.2 Estratégia Operacional

O endereço *WEB* do SKICCOSMO é <http://skiccosmo.on.br> pois está hospedado no domínio do Observatório Nacional. Esse endereço, que chamaremos de *portal* para usar a notação atual, pode ser qualquer máquina, desde que tenha um "alias" associado ao endereço acima, no servidor de nomes de domínio. O servidor desse portal redireciona, imediatamente, a transferência de dados para um outro servidor aqui denominado servidor *CGI*¹, pois o programa de controle, a interface entre o usuário e o banco de dados, está

¹CGI: *Common Gateway Interface*. Há duas formas de um servidor WEB enviar arquivos para o cliente, chamado de *browser*. Se o endereço solicitado for do tipo *Document*, o servidor envia o arquivo descrito no endereço. Caso esse endereço for do tipo *CGI*, o servidor executa o programa declarado no endereço. Esse programa é suposto se encarregar do envio do "documento" no formato esperado de acordo

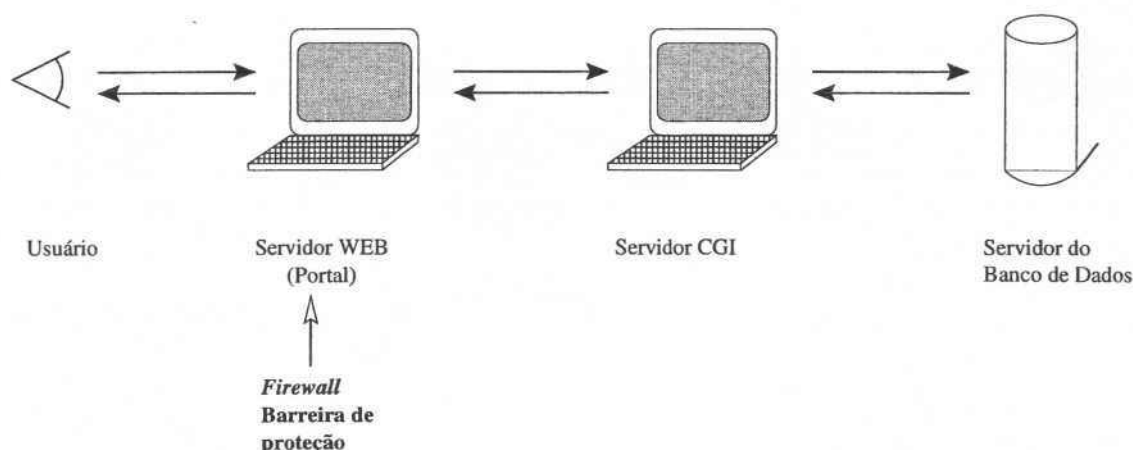


Figura 4.1: Esquema da distribuição em três camadas

ali instalado. É indiferente esses servidores estarem instalados em máquinas diferentes ou na mesma máquina. Por outro lado, todos os contatos com o banco de dados serão feitos através dessa interface. É recomendado pelos profissionais da área que os diferentes servidores estejam, efetivamente, instalados em três máquinas diferentes. Esse cenário é conhecido por *distribuição em três camadas (three layers)*. Esquemáticamente, essa distribuição pode ser vista na Figura 4.1.

Por razões materiais e operacionais, a máquina servidora *WEB* e a do Banco de Dados são as mesmas. Dentro da distribuição do Observatório Nacional, a operação do SKICCOSMO é resumida ao esquema da Figura 4.2. Podemos deduzir da figura que a conexão do ON com a Rede Rio 2 é um limitador significativo. É nosso projeto permitir que as servidoras *WEB* e *CGI* se comuniquem diretamente a 100 Mbps ao roteador e que este se conecte via cabo óptico ao *backbone* da Rede Rio 2.

4.1.3 A Página Web

Após definir o “URL” do SKICCOSMO, o usuário terá em seu *browser* a Figura 4.3. Juntamente com essa página, um outro “quadro” (*frame*), de dimensões de cerca de 1/4 do quadro principal aparecerá. Ele é chamado de *Working Page* e será útil para várias funções do SKICCOSMO. O usuário deverá preencher o campo do formulário com seu endereço eletrônico. Esse endereço eletrônico será usado para todas as comunicações do “gerente” do SKICCOSMO com o usuário. Caso esse endereço não seja encontrado, o “gerente” marcará com a aplicação. Essa medida dá a possibilidade de geração de páginas dinâmicas, permitindo o usuário interagir com serviços diversos, bancos de dados, etc.

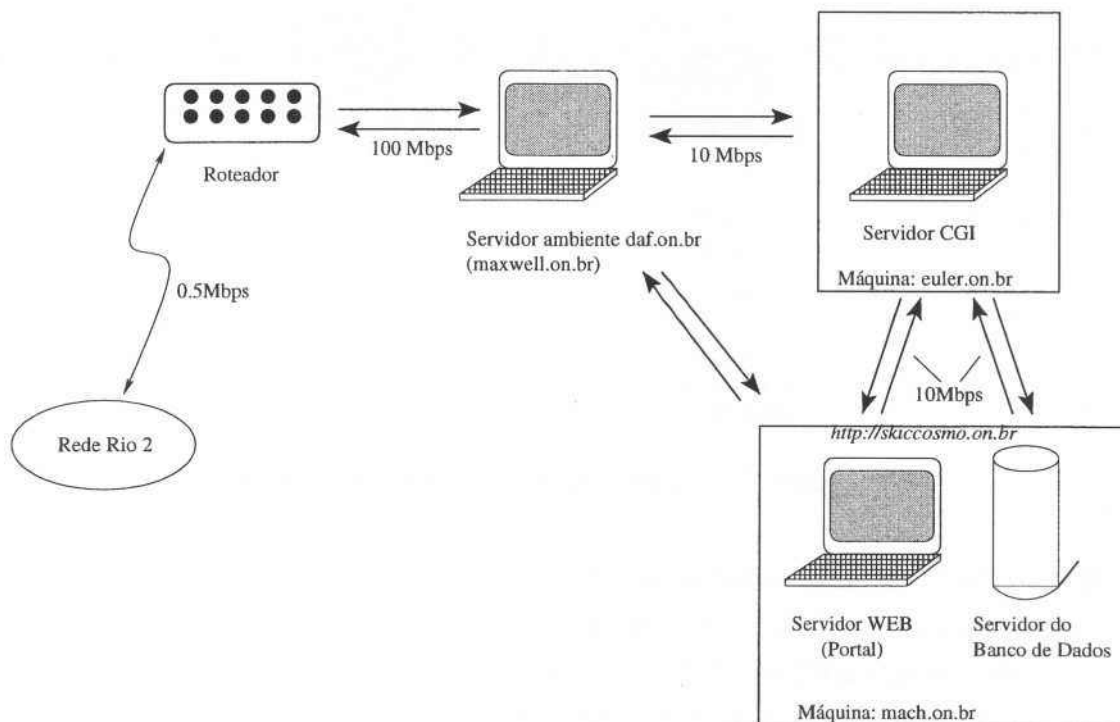


Figura 4.2: Esquema do SKICCOSMO no Observatório Nacional

esse usuário com um indicador “inativo”. É recomendável, também, manter a chamada janela *Working Page* onde algumas informações úteis aparecerão.

Na parte superior da página, abaixo do logotipo do SKICCOSMO, pode se ver dois *botões*, chamados botões de navegação, que permitem visualizar descrições sumárias do projeto SKICCOSMO e algumas indicações de como se operar com o banco de dados.

Após entrar com o endereço eletrônico, o usuário será convidado a preencher um formulário tendo em vista seu cadastramento no SKICCOSMO. Informações como nome completo, instituto a que está associado, número de telefone e fax são requeridas para tornar mais fácil a comunicação entre o pessoal responsável pela manutenção do SKICCOSMO e o usuário, caso seja necessário. Uma vez preenchido e submetido o formulário, o usuário atinge a página conforme mostrado na Figura 4.4. Abaixo do campo contendo o endereço eletrônico e do indicador da contagem de objetos registrados, encontram-se 5 *botões* onde o usuário poderá visualizar - na *Janela de Trabalho (Working Page)* - a listagem de: 1 - Dispositivos de detetores (*Devices*); 2 - Filtros fotométricos (*Filters*); 3 - Bandas, formadas pela combinação do dispositivo e filtro (*Bands*); 4 - Classificadores utilizados no momento do processamento das imagens pelo **Focas** (*Classifiers*); e, 5 - A listagem de todas as imagens e suas descrições conforme seus cabeçalhos em *Headers*.

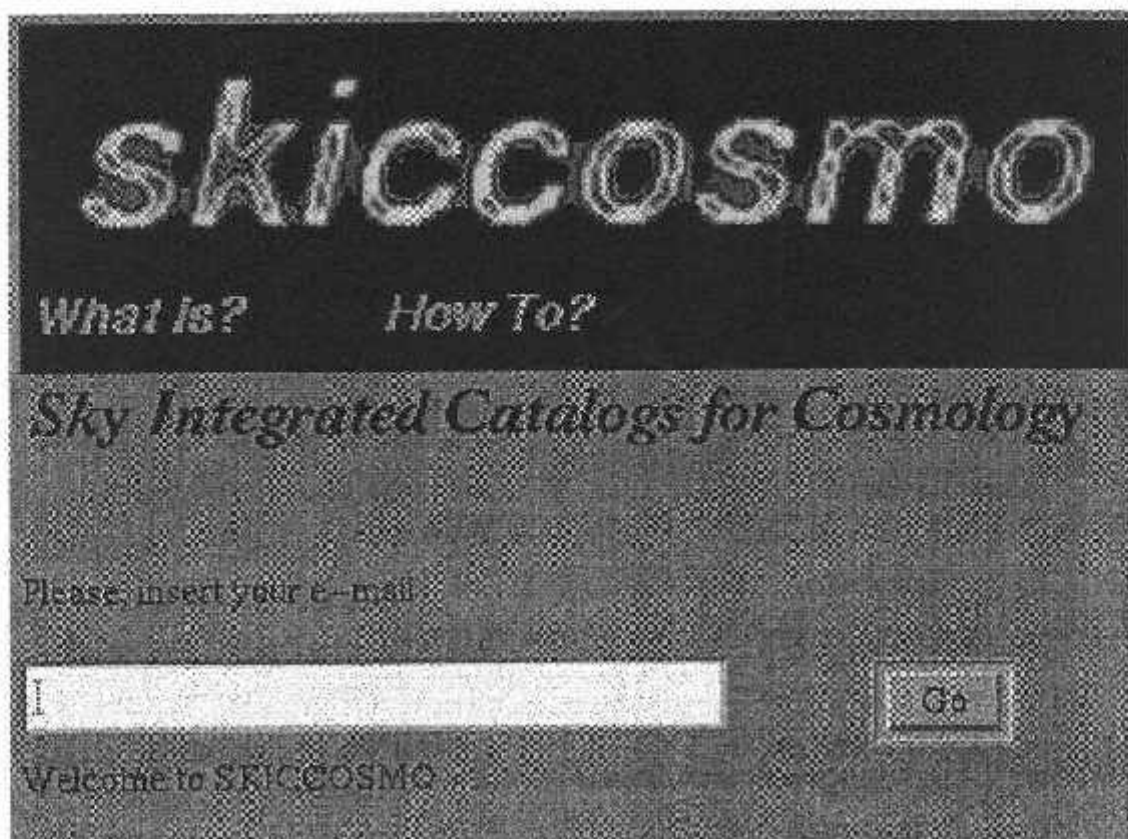


Figura 4.3: Apresentação da página *Web* do SKICCOSMO

Sky Integrated Catalogs for Cosmology

Please, insert your e-mail:

Welcome to SKICCOSMO

Number of registered objects in database: 16320063

List images according to

Device type	Filter type	Band
<input type="text" value="CCD"/> <input type="button" value="Go"/>	<input type="text" value="Hla-F"/> <input type="button" value="Go"/>	<input type="text" value="F"/> <input type="button" value="Go"/>

Figura 4.4: Página após o e-mail ser reconhecido pelo SKICCOSMO

Abaixo, pode-se listar, ainda na *Janela de Trabalho*, as imagens processadas e registradas no SKICCOSMO conforme: o tipo de dispositivo (no caso PLT ou CCD), o tipo de filtro e a banda utilizada na observação.

Abaixo do logotipo do SKICCOSMO, agora, pode-se ver dois *botões* de navegação adicionais. Um que leva à página de Construção de Consultas e outro que permite acompanhar o *status* das consultas submetidas.

4.1.4 A Página de Construção de Consultas

A Página de Construção de Consultas é dividida em três partes, além dos botões de navegação: a primeira parte se refere a informação de atributos e funções disponíveis no SKICCOSMO; a segunda permite recuperar consultas já registradas no Banco de Consultas, e finalmente, a janela de consulta propriamente dita.

A parte informativa contém o campo do endereço eletrônico do usuário e três botões que se destinam a auxiliar o usuário com informações que facilitem a construção de sua consulta. O primeiro lista, na Janela de Trabalho (*Working Page*), os atributos relevantes do cabeçalho das imagens registradas no SKICCOSMO. O segundo lista os atributos da vista *Features* (ver 3.3) com o que o usuário se serve para definir as colunas ou atributos que ele deseja obter. Finalmente, o terceiro botão faz mostrar as funções que o usuário tem à mão para aplicar nos atributos no momento de gerar sua consulta. Essa parte também expõe os botões de informação já comentados em 4.1.3.

A segunda parte da página de consulta permite recuperar consultas realizadas anteriormente pelo próprio usuário, como também recuperar exemplos que são disponibilizados para permitir uma melhor compreensão de como é construída uma consulta. Basta escolher uma consulta pelo nome no menu “*rolante*”². A consulta é recarregada na área de construção de consultas que é constituída da terceira parte da página. Os aspectos da primeira e da segunda parte da página de construção de consulta podem ser vistos na Figura 4.5.

Finalmente, vem a terceira parte que é da construção da consulta propriamente dita, mostrada na Figura 4.6.

O campo *Query Name* deve ser preenchido pois será através do nome que o resultado da consulta deverá ser recuperado, bem como a própria consulta poderá ser recuperada mais tarde através de seu nome. Não podem existir duas consultas com o mesmo nome.

Quando uma consulta é submetida com sucesso, o usuário recebe um aviso na Janela de Trabalho (*Working Page*). Então ele receberá, por *e-mail*, um aviso que o resultado da

²Todas as operações são efetivadas quando a chave **Go** é acionada.

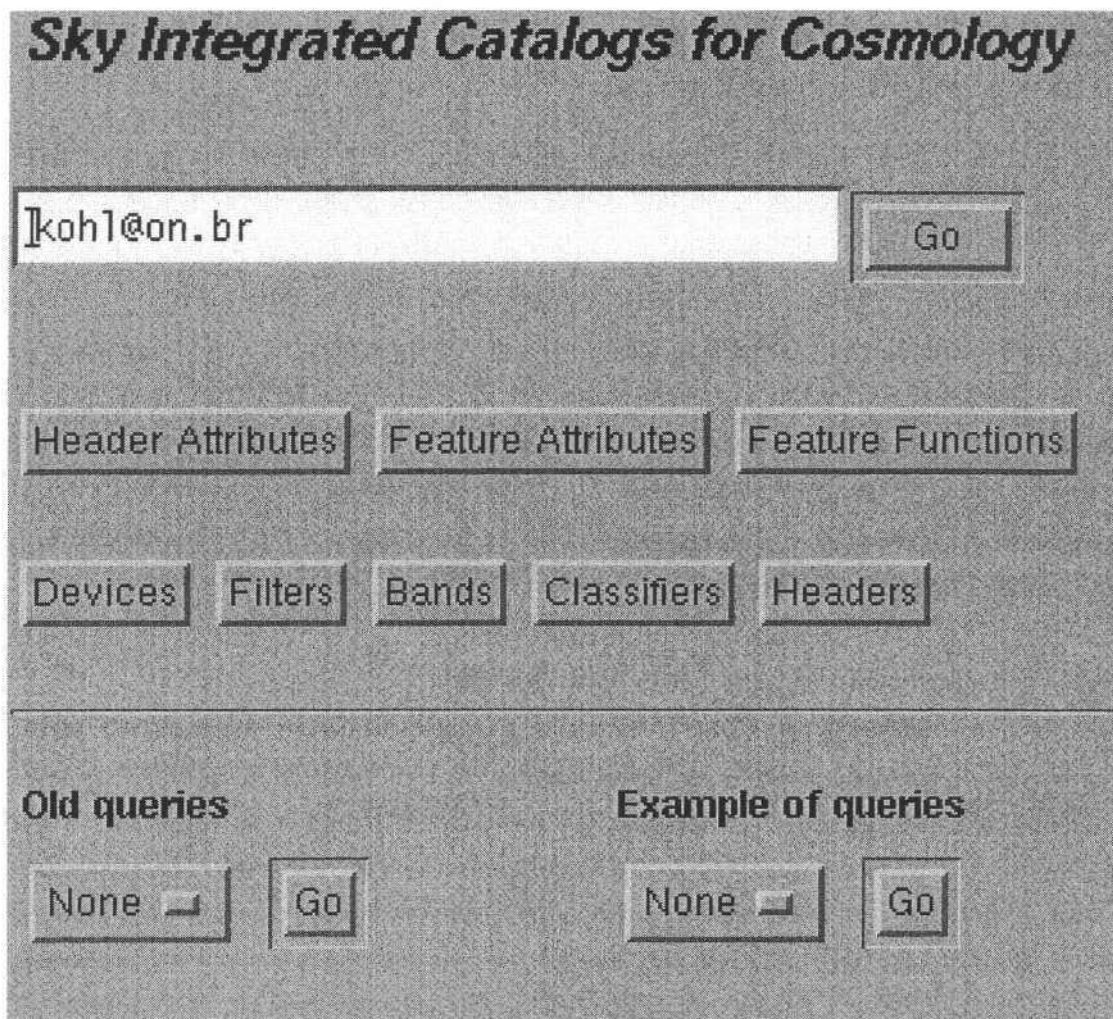


Figura 4.5: Visão da 1a. e 2a. parte da página de construção de consultas

Query Name:		Compression Method:
		Tar Compress
Magnitude Range	Flags	Classification
Min <input style="width: 50px;" type="text" value="15.0"/>	<input type="checkbox"/> AF9 Special attention <input type="checkbox"/> BF9 Edge of field <input type="checkbox"/> CF9 Problem in classifier <input type="checkbox"/> DF9 Below sky <input type="checkbox"/> EF9 Successfully evaluated	<input type="checkbox"/> FF9 Forced classification <input type="checkbox"/> LF9 Exceed limit of Focas <input type="checkbox"/> PF9 Saturated pixel in object <input type="checkbox"/> RF9 Reference point <input type="checkbox"/> SF9 Not split at any level
Max <input style="width: 50px;" type="text" value="19.0"/>		Classifier <input style="width: 100px;" type="text" value="FOCAS"/> Class Type <input style="width: 100px;" type="text" value="s + g"/>
Attribute Definitions		Conditions
<input style="width: 80px;" type="button" value="Submit"/>		<input style="width: 80px;" type="button" value="Reset"/>

Figura 4.6: Visão da 3ª parte da página de construção de consultas

consulta estará disponível e poderá ser recuperado num endereço específico.

A consulta é, dessa forma, interpretada e todos os seus parâmetros são armazenados nas tabelas *userQueries* e *queryDetails*. A consulta em linguagem **SkiccQL** (ver abaixo em 4.2) é traduzida para uma instrução SQL que também é armazenada no atributo *genSQL* da tabela *userQueries*, visto em 3.4.

A instrução **SQL** é recolhida do banco de dados por um utilitário chamado *officeboy*, que periodicamente consulta a tabela *userQueries* a procura de consultas marcadas como *Submitted* no atributo *qrystat*. Quando *officeboy* encontra uma consulta nessas condições, ele a marca com o indicador *Processing* e submete a consulta ao *motor* do banco de dados. Quando a consulta é finalizada, o *officeboy* recolhe o resultado e o guarda num arquivo chamado **Results.dat**. Um relatório contendo os detalhes da consulta é guardado num arquivo chamado **Report.log**. Esses arquivos são colocados num diretório que é comprimido usando um método de compressão escolhido pelo usuário no momento da construção da consulta. Então o *officeboy* grava o resultado numa área acessível pelo servidor **http**, marca a consulta como *Available* e envia um *e-mail* ao usuário, avisando que sua consulta está disponível. A consulta fica disponível ao usuário por um período determinado³. Um usuário tem acesso somente aos arquivos gerados em suas consultas.

Paralelamente ao *officeboy*, um utilitário chamado *collector*, procura por consultas com idade superior ao período estipulado. As consultas nessas condições são marcadas com o indicador *Deleted* e o arquivo comprimido é removido do diretório. Ao mesmo tempo, o *collector* apaga todos os arquivos da área pública com idade superior a 10 dias. Essas medidas são necessárias devido à exigüidade de espaço em disco do servidor *WEB*. A atividade em torno do SKICCOSMO vai ditar uma eventual mudança nesses prazos.

Quando o usuário recebe o *e-mail* do *administrador* do SKICCOSMO, ele poderá consultar a página *QueryStatus* acessível pelos botões de navegação no topo da página do SKICCOSMO. Nessa página, as consultas disponíveis estarão com seus nomes marcados com *links* para os arquivos comprimidos que assim poderão ser transferidos para o computador do usuário. Além delas, todas as outras consultas, inclusive as apagadas, estarão listadas em uma tabela, contendo o *status* da consulta e a data de sua submissão ao SKICCOSMO.

Voltemos, então, à parte da página dedicada à construção da consulta. O usuário deve, portanto, preencher o campo dedicado ao nome da consulta. Ao lado desse campo existe um menu para escolher o tipo de compressão que será usado pelo *officeboy* para produzir

³Atualmente esse período é de uma semana.

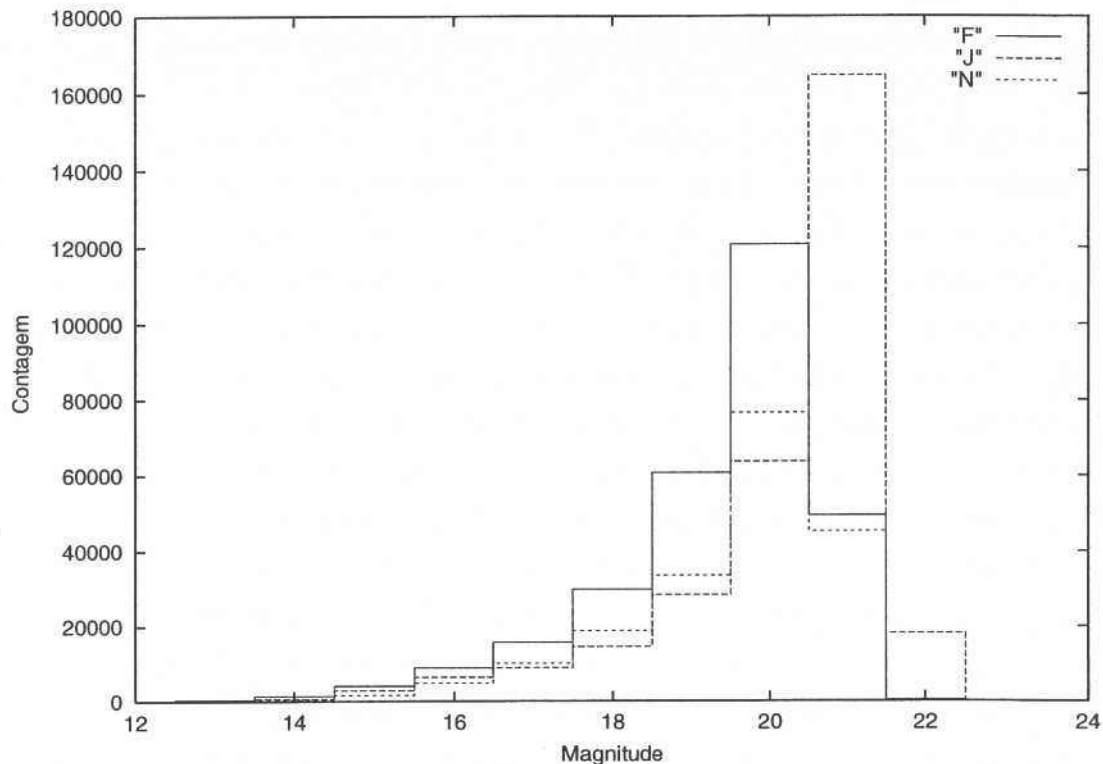


Figura 4.7: Histogramas da magnitude total nas bandas F, N e J de um campo do POSS-II.

o resultado final em um arquivo.

Na Figura 4.6, abaixo do campo do nome da consulta, vê-se três sessões para definição de valores de entrada. O banco de dados precisa ser protegido de eventuais consultas que, por inadvertência do usuário, impõem ao *motor* do banco de dados a execução de tarefas extremamente pesadas, sem que exista uma motivação para isso, ou seja, que o próprio usuário não tenha idéia do que pretende obter. Entre as medidas de proteção encontram-se valores que certos atributos da tabela de *Features* devem estar restritos. São eles: magnitude total, *Flags*, e classificação dos objetos segundo um classificador pré-definido.

A sessão da magnitude estabelece, automaticamente, os limites mínimos e máximos, respectivamente, 15.0 e 19.0. Esses valores podem ser mudados tanto nos campos dedicados a essa sessão como através de definição explícita no campo específico para as definições das condições da consulta, como será visto mais adiante. Os limites de 15.0 e 19.0 servem para, diante de uma consulta inadvertidamente mal feita, fazer com que o “motor” do banco de dados não seja, desnecessariamente, sobrecarregado. Pode-se ver na Figura 4.7 que essa faixa de magnitude não contém um número significativamente alto de objetos.

Na sessão dos indicadores estabelecidos pelo FOCAS, apresenta-se em forma de "chaves", uma para cada indicador. Em princípio, essas chaves, apesar de estarem disponíveis para serem "ligadas" ou "desligadas", não devem ser modificadas. As duas únicas chaves "ligadas" são a **EFlg** e **SFlg** que indicam, respectivamente, *Successfully evaluated* e *No split at any level*. São esses os indicadores que estão ativos para os objetos perfeitamente identificados e avaliados. Objetos com outras características, embora teoricamente de menor interesse, permanecem disponíveis em caso de futuras reclassificações e identificações. Importante notar que "desligar" um indicador é diferente de forçar um dado indicador ao valor "falso". Desligar uma chave de indicador significa apenas que a consulta não fará qualquer teste com o indicador. Se o usuário desejar testar um dado indicador com o valor falso é preciso fazê-lo no campo de construção das condições, apresentado mais adiante.

Finalmente, tem-se a sessão dedicada à classificação. O usuário pode escolher que o classificador, seja o **FOCAS** ou o **DTREE**⁴. As classificações podem ser escolhidas entre **s** (objeto puntiforme), **g** (galáxia) ou **s+g** (estrelas mais galáxias).

Todas as escolhas apresentadas até aqui, são escolhas que o tradutor **SkiccQL**⁵ (ver mais adiante em 4.2) fará no caso em que nada foi especificado nas condições da consulta com respeito a esses itens. Toda condição contraditória com essas escolhas terá precedência, de forma que o usuário é livre para redefinir as condições sobre os atributos de acordo com sua conveniência, mesmo que não faça qualquer modificação nas sessões descritas acima.

De fato, a definição completa da consulta poderá ser feita exclusivamente através das janelas para construção das consultas, como veremos a seguir.

4.1.5 As Janelas para Construção de Consultas

A toda consulta o SKICCOMOS retornará uma tabela. Essa tabela é um subconjunto dos dados armazenados no SKICCOSMO. Esse subconjunto é composto de atributos ou colunas armazenadas cujas instâncias estão sujeitas às condições impostas pela consulta. Portanto uma consulta ao SKICCOSMO é composta de duas partes, a definição das colunas da tabela e a definição das condições dos dados.

Existem duas janelas específicas para a construção da Consulta ao SKICCOSMO: a janela de definição dos atributos ou colunas de retorno e a janela onde se pode construir condições, conforme pode ser visto na parte inferior da Figura 4.6. Essas janelas são do tipo "texto", portanto edita-se à moda de edição de texto. O usuário compõe, com a ajuda

⁴O algoritmo *Neural Network* está em desenvolvimento (Odehian, 1999, [46]).

⁵Fala-se *Skí-quel*, coerentemente com a pronúncia *Seqüel* adotada para **SQL**.

do teclado, os atributos e as condições de sua consulta.

Com o objetivo de facilitar a tarefa do usuário na construção de uma consulta, foi criado um "tradutor", ou um "pré-compilador" chamado **SkiccQL**, significando linguagem de consulta do **SKICCOSMO** (*Skiccosmo Query Language*). Inversamente ao **SQL**, cuja aplicação exige pleno controle, sob pena de comprometer seriamente o processo da consulta e seus resultados (ver prefácio de E. Epstein a Bowman *et al.*, 1998, [6]). Na **SQL** é preciso atentar para as estritas regras de junções internas e auto - junções, além do uso adequado de operadores como o "or" lógico, etc. Um pequeno erro pode levar o *motor* a travar. Com o **SkiccQL** essa possibilidade deixa de existir, pois esse pré compilador se encarrega de gerar as instruções necessárias em **SQL** de maneira que a consulta, em sua forma final, é eficaz e produz os resultados desejados.

Essa condição é possível porque algumas decisões são tomadas automaticamente na geração da instrução **SQL**. Além dos valores "assumidos por falta" (*default*) das sessões expostas anteriormente, o **SkiccQL** força a geração de junções internas, todas as vezes que o usuário quiser informações a respeito de parâmetros dos objetos em mais de uma banda. Em outras palavras, se o usuário pede, em uma mesma consulta, informações sobre objetos na banda **F** e **J**, o **SkiccQL** gerará uma consulta em **SQL** que retornará somente objetos concomitantemente presentes nos campos de **F** e **J**. O mesmo será feito se forem escolhidas mais bandas. Essa imposição é necessária pois, do contrário, a consulta faria retornar o chamado *Produto Cartesiano* que é a primeira instância de dados gerados na consulta, sem significado na astronomia pois é o conjunto de todas as combinações possíveis dos atributos requisitados. Além disso tal procedimento travará o *motor* do banco de dados.

Para a descrição completa da **SkiccQL**, foi dedicada a Seção abaixo.

4.2 A Linguagem de Consulta

4.2.1 Convenção

Algumas convenções nas explicações que se seguem são necessárias em nome da clareza dos conceitos apresentados.

- **Letra maiúscula (Caixa Alta):** nomes de tabelas, banco de dados, DBMS e compiladores;
- **Letra de máquina (Courier):** comandos, nomes de atributos usados em programação;

- **Itálico:** palavras estrangeiras consagradas no jargão profissional.

4.2.2 Conceitos

A recuperação dos dados através da Linguagem de Consulta Astronômica se dá, essencialmente, na recuperação dos atributos da tabela DPOSSII_FEATURES, transformada para a vista FEATURES, acrescentados dos elementos da tabela CLASSIFICATIONS (ver Figura 3.2). Assim, toda operação envolvendo essa ferramenta levará, obrigatoriamente, à utilização dos dados dos objetos catalogados, sendo que cada atributo está forçosamente ligado a uma banda, quer ela envolva uma placa fotográfica ou uma imagem CCD. Todos os atributos da tabela são disponíveis, inclusive as chaves primárias “id_obj” e “id_header”.

Para efeito de programação, define-se uma “banda” como sendo o conjunto de uma ou mais letras seguido imediatamente por um ponto (“.”). Dessa forma, a banda “F” é determinada na SkiccQL como sendo “F.” Um atributo da vista “FEATURES” é completamente definido quando precedido da definição de uma banda. Por exemplo, a magnitude total de um objeto na banda “J” é definido por “J.mtot”⁶.

Toda e qualquer consulta deve possuir imposições para a classificação dos objetos. O SKICCOSMO, a exemplo do SKICAT, tem armazenado todo o conjunto de objetos classificados durante a execução do FOCAS. Assim, parte dos objetos é classificada como ruído, ou objeto múltiplo ou indefinido. Esses objetos estão disponíveis para permitir que o usuário possa encontrar meios de classificar diferentemente alguns deles, se for o caso. Toda possibilidade deve ser acompanhada, evidentemente, de um estudo preliminar cuidadoso, sob pena de fazer retornar da consulta uma miríade virtualmente infinita de objetos sem classificação óbvia. Portanto, para efeito de consulta “trivial”, o tradutor tomará a iniciativa de impor uma classificação por “*default*” para cada banda escolhida e que não tenha classificação definida pelo usuário. Atualmente, a classificação por “*default*” é de objetos classificados como “s” (estrela, objeto pontual) ou como “g” (galáxia).

Cada consulta em SkiccQL será transformada em uma instrução em “SQL” para que seja possível enviá-la ao SKICCOSMO. Essa instrução consta do arquivo “log” que acompanha os dados de retorno da consulta.

Uma vez definidas as bandas, o banco de dados retornará um resultado que será o conjunto de objetos presentes simultaneamente nas imagens das bandas escolhidas. Por

⁶Aqui vale um comentário: enquanto a definição de um atributo pode ser feita, indiferentemente, com letras maiúsculas ou minúsculas, a definição da banda não pode ser feita da mesma maneira. Assim, “F.mtot” e “F.MTOT” quer dizer a mesma coisa, enquanto “f.mtot” e “F.mtot” referem-se a grandezas distintas, pois a banda “f” (se vier a existir) distingue-se de “F”.

exemplo, quando o usuário escolhe atributos na banda "F." retornarão apenas os objetos dessa banda. Quando o usuário acrescenta atributos da banda "J.", apenas retornarão objetos nas bandas "F" e "J". Em outras palavras, o resultado da consulta será o conjunto dos objetos que possuem valores para a banda "F" e que possuem também valores para a banda "J". Essa operação representa uma intersecção entre os conjuntos dos objetos da banda "F" e os da banda "J". Uma consulta nunca retorna uma operação de união entre conjuntos. Essa medida possui uma explicação simples: a disponibilização da operação de união também permitiria ao usuário fazer a operação cujo resultado é conhecido como o "produto cartesiano", que além de poder provocar a super ocupação da memória tampão do banco de dados (*overhead*), não tem utilidade para o astrônomo, pois carece de sentido físico. É, assim, preferível que o usuário se disponha a fazer várias consultas independentes com mudança de alguns parâmetros, se for o caso, a colocar em risco a operação do banco de dados.

Uma consulta nunca é enviada diretamente ao banco de dados. Ela é colocada em uma "pilha" aguardando para ser enviada por um sistema de controle independente. A consulta, portanto, é colocada em um *batch*, cujo resultado será armazenado em um servidor de "ftp" ou equivalente, disponível para ser recuperado oportunamente. São duas as razões desse procedimento: 1) A consulta pode fazer retornar um número apreciável de objetos e o usuário não veria grande vantagem em contemplá-los simplesmente em uma tabela "html"; 2) uma consulta pode ser assaz longa e o retorno direto ao *browser* implicaria em medidas auxiliares de proteção da consulta em caso de corte na conexão, corte esse, muitas vezes feito pelo próprio usuário, impaciente com a demora na resposta. Uma consulta é, dessa feita, "depositada" num monitor de consultas, processada oportunamente, e os resultados são armazenados em diretório apropriado aguardando para ser enviado ao interessado. O usuário é advertido com um *e-mail* assim que a consulta é completada e os dados disponibilizados.

4.2.3 A Definição dos Atributos

O preenchimento da área destinada à definição dos atributos se dá através de linhas como se fosse um programa em uma linguagem como PASCAL ou C. Cada instrução é separada em ";" . A separação em linhas não é considerada. Ela se faz apenas por uma questão de legibilidade. Cada instrução é definida por duas partes separadas por um sinal "=". O lado esquerdo da "igualdade" se refere ao nome que a coluna terá no relatório e o lado direito conterá a definição da expressão contendo os atributos da vista FEATURES (ver

3.3). Assim a definição de uma coluna de saída no relatório se dá na instrução:

```
[nome da coluna] = {expressão contendo atributos da vista features};
```

Podem haver tantas instruções quantas se queira, dentro dos limites impostos pelo *browser* para a janela do tipo “área de texto”.

A simples listagem de ascensão reta e declinação em radianos seria⁷:

```
Right Ascension = RA;
Declination = DEC;
```

O interpretador do SKICCOSMO irá traduzir essa instrução para uma instrução em “SQL” e o resultado será:

```
select RA as Right_Ascension,DEC as Declination from objects o
```

Abaixo vemos um trecho truncado do resultado dessa consulta, pois do contrário não haveria espaço nessa tese para mostrar o arquivo integral⁸:

```
right_ascension      declination
0.06449427978353    -0.0432600184014
0.06447800121136    -0.0461029636710
0.06446336214775    -0.0420751780324
0.06445959627152    -0.0416251246141
0.06446116503985    -0.0472251739812
0.06443939369794    -0.0432708749440
0.06440990335005    -0.0416698407533
0.06440591276961    -0.0412867699976
0.06441797189592    -0.0467324975020
0.06441157614811    -0.0448863443000
0.06440130633316    -0.0452616003424
```

⁷Rigorosamente, a consulta no exemplo não é permitida. Todo objeto está associado a pelo menos uma banda, de maneira que uma consulta deve defini-la de alguma forma. Essa consulta é mostrada apenas para fins de ilustração.

⁸Rigorosamente, essa consulta não é possível no SKICCOSMO pois outras condições aos objetos deverão ser, obrigatoriamente, introduzidas (ver adiante em 4.2.4.2). As consultas nessa seção foram produzidas para fins de ilustração. Não foram elaboradas pelo SKICCOSMO. Elas foram submetidas com o auxílio do utilitário INFORMIX *dbaccess*. Por isso, os resultados são apresentados diferentemente daqueles obtidos com a ajuda do SKICCOSMO (comparar com os resultados da seção 4.3).

```
0.06440202205770 -0.0460544733332
0.06435995659605 -0.0407330239056
```

Podemos notar que os nomes das colunas perderam espaços em troca de “_” e as letras tornaram-se minúsculas. O Informix impõe que espaços entre palavras não sejam aceitos. Ele se encarrega de substituí-los por “_”, além de todos os nomes serem automaticamente transformados para letras minúsculas.

Uma consulta mais complexa é:

```
Right Ascension = RA;
Declination = DEC;
F Total Mag = F.mtot;
F Aper Mag = F.maper;
```

Nesse caso pede-se além das coordenadas do objeto, alguns parâmetros fotométricos. O “F” e “ponto” no lado direito das duas últimas expressões indica a banda relativa à magnitude total e à magnitude de abertura, indicadas nas expressões. Como resultado da compilação o “tradutor” produz a instrução em “SQL”:

```
select RA as Right_Ascension,DEC as Declination,
F.mtot as F_Total_Mag,F.maper as F_Aper_Mag
from objects o, Features F,dpossii_headers FD
where F.id_obj = o.id_obj
and F.id_header = FD.id_header
and FD.band = 'F'
```

E como resultado dessa consulta, temos:

right_ascension	declination	f_total_mag	f_aper_mag
0.06449427978353	-0.0432600184014	21.2830000000000	21.3790000000000
0.06447800121136	-0.0461029636710	20.5320000000000	20.4420000000000
0.06446336214775	-0.0420751780324	19.4700000000000	19.5180000000000
0.06445959627152	-0.0416251246141	17.0010000000000	17.0450000000000
0.06446116503985	-0.0472251739812	18.3160000000000	18.3290000000000
0.06443939369794	-0.0432708749440	20.0740000000000	20.0670000000000
0.06440990335005	-0.0416698407533	21.4230000000000	22.1460000000000

```

0.06440591276961 -0.0412867699976 21.4190000000000 21.3850000000000
0.06441797189592 -0.0467324975020 21.1010000000000 21.8870000000000
0.06441157614811 -0.0448863443000 16.4060000000000 16.4520000000000
0.06440130633316 -0.0452616003424 19.9400000000000 19.6610000000000
0.06440202205770 -0.0460544733332 20.3540000000000 20.3950000000000
0.06435995659605 -0.0407330239056 21.9870000000000 20.7500000000000

```

Note-se que é indiferente registrar letras maiúsculas ou minúsculas com respeito aos nomes dos atributos das tabelas do SKICCOSMO, como já foi dito na nota de pé da página 148, já não se pode dizer o mesmo com respeito aos nomes da banda fotométrica. Em outras palavras, a banda "F" nada tem a ver com a banda "f", se esta, por acaso existir. A definição da coluna "f.mtot" produzirá um resultado vazio, porque não existe banda "f" registrada, até o momento, no banco de dados.

Importante: A definição de uma banda fotométrica se faz através da composição do nome "astronômico" da mesma mais o ".", no final. Em outras palavras, a variável "F.mtot" significa o atributo "mtot" da banda "F". Se quisermos nos referir à banda "F" sem atributos, para o interpretador, devemos escrever "F.". A não colocação do ponto ao final da definição da banda fará o interpretador produzir uma mensagem de erro.

Na consulta em "SQL", vê-se que o compilador tratou de fazer as devidas junções internas, típicas do banco de dados relacional, mas que exigem do usuário conhecimento e experiência nesse tipo de programação. O objetivo é livrar o usuário astrônomo da necessidade de se especializar nessa linguagem e lhe evitar transtornos, para que possa, dessa forma, dedicar-se à questão científica.

Uma vez que as providências para promover as junções internas em uma consulta tenham sido tomadas, o astrônomo pode querer obter os objetos que possuem valores para os três campos do POSS-II. Assim ele faria:

```

F Total Mag = F.mtot;
J Total Mag = J.mtot;
N Total Mag = N.mtot;

```

o que produziria a consulta "SQL":

```

select F.mtot as F_Total_Mag,J.mtot as J_Total_Mag,
N.mtot as N_Total_Mag

```

```

from Features F,dpossii_headers FD,objects o,
Features J,dpossii_headers JD,Features N,
dpossii_headers ND
where F.id_obj = o.id_obj and F.id_header = FD.id_header
and J.id_obj = o.id_obj and J.id_header = JD.id_header
and N.id_obj = o.id_obj and N.id_header = ND.id_header
and FD.band = 'F' and JD.band = 'J' and ND.band = 'N'

```

que daria o resultado:

f_total_mag	j_total_mag	n_total_mag
19.94000000000000	20.16500000000000	20.51500000000000
19.50300000000000	20.24500000000000	19.33100000000000
17.94100000000000	18.04200000000000	18.76900000000000
19.96600000000000	20.44200000000000	20.92300000000000
19.02200000000000	20.43800000000000	18.80100000000000
18.45200000000000	20.09400000000000	18.39200000000000
18.73200000000000	20.10300000000000	19.01500000000000
19.64300000000000	20.86900000000000	20.53900000000000
19.50600000000000	19.83900000000000	21.12600000000000
18.71800000000000	19.21800000000000	19.95900000000000
18.56800000000000	19.75400000000000	19.12000000000000
17.36600000000000	17.73100000000000	18.30600000000000

4.2.3.1 Indexação

Atributos em uma consulta podem ser indexados. A forma mais simples de indexação é a definição do número do campo que se quer usar para recuperar dados fotométricos de objetos. Ex:

```
F Total Mag = F.mtot[826];
```

Essa linha define que os objetos a serem recuperados devem ser da banda 'F' e pertencentes ao campo "826" do POSS-II, isto é, serem objetos da placa "f826" em termos de nomenclatura do POSS-II.

Quando se quer objetos advindos de campos determinados, deve-se colocar uma lista de campos dentro dos colchetes. Ex:

```
F Total Mag = F.mtot[826,853];
```

Nesse exemplo, foram escolhidos os objetos advindos das placas "f826" e "f853". O resultado da consulta conterá o conjunto dos dois campos.

4.2.3.2 Campos com múltipla indexação

O resultado da consulta:

```
F Total Mag = F.mtot[826][827];
```

força que os dados sejam procurados entre os objetos advindos do campo 826, que também sejam pertencentes ao campo 827. A múltipla indexação é permitida, no entanto deve ser usada com atenção, sob o risco de obter-se um conjunto vazio como resultado.

Atenção: Instruções com indexação simples e lista de múltiplos campos promove uma operação equivalente à união entre conjuntos; ao passo que a múltipla indexação promove uma operação equivalente a uma intersecção entre conjuntos.

4.2.3.3 Indexação Indireta

Pode-se promover uma indexação indireta, fazendo o índice referir a uma outra banda que esteja definida na mesma consulta, por exemplo:

```
J Total Mag = J.mtot[F.];
```

```
N Total Mag = N.mtot[F.];
```

```
F Total Mag = F.mtot[826];
```

Nesse caso se está dizendo para o compilador impôs que os atributos `platenumb` das bandas "J" e "N" sejam iguais ao mesmo atributo da banda "F". Como resultado a consulta em "SQL" ficaria:

```
select J.mtot as J_Total_Mag, N.mtot as N_Total_Mag,
       F.mtot as F_Total_Mag
from objects o, Features J, dpossii_headers JH, classifications JC,
       Features N, dpossii_headers NH, classifications NC,
       Features F, dpossii_headers FH, classifications FC
where o.id_obj = J.id_obj and J.id_header = JH.id_header and JH.band = 'J'
and o.id_obj = N.id_obj and N.id_header = NH.id_header and NH.band = 'N'
```



```

and o.id_obj = F.id_obj and F.id_header = FH.id_header and FH.band = 'F'
and ( JH.platenumb = FH.platenumb)
and ( NH.platenumb = FH.platenumb)
and ( FH.platenumb = 826)

```

Pode-se, igualmente, omitir o índice direto, no caso, o número da placa para a banda "F" (826). Assim, poderíamos impor simplesmente que o número da placa para as bandas "J" e "N" fossem os mesmos da banda "F".

Importante: Não são permitidas indexações "aninhadas", isto é, indexações de indexações. Ex: J.Mtot [F. [833]] produz uma instrução SQL errada.

4.2.3.4 Indexação na Classificação do Objeto

O atributo "class" admite um indexador definindo o classificador. Valores válidos para o classificador são FOCAS e DTREE. Outros classificadores poderão estar disponíveis no futuro (ex., o *NeuroNet*). Adota-se, na falta de um índice nesse objeto, o valor padrão FOCAS. Por outro lado, esse atributo não admite indexador para o número do campo. Visto que "class" é um atributo da tabela CLASSIFICATIONS, não cabe indexá-la no número do campo, que pertence à tabela DPOSSII_FEATURES (ver Figura 3.2). A definição do classificador e classificação pela indexação sobrepõem-se aos valores escolhidos na sessão de valores *default*.

4.2.3.5 Indexações esparsas

Se atributos de uma mesma banda são indexados independentemente, a consulta levará ao mesmo resultado de múltiplas indexações. Por exemplo, a consulta:

```

F Mag Tot = F.mtot[833];
F Mag Aper = F.maper[834];

```

é o mesmo que:

```

F Mag Tot = F.mtot[833][834];
F Mag Aper = F.maper;

```

Importante: Tanto nas indexações múltiplas quanto nas esparsas, deve-se tomar o cuidado de verificar se nos campos constando da consulta há, efetivamente, intersecção. Do contrário, obter-se-á um conjunto nulo, como resultado. Por exemplo, a consulta

Função	Expressão	Transformação
Flux	$\exp(-0.9210340372 * M)$	magnitude em fluxo
Mag	$-2.5 * \log(F)$	fluxo em magnitude
Deg	$57.295779513 * R$	radianos em graus
Rad	$0.01745329252 * D$	graus em radianos
Hour	$3.8197186342 * R$	radianos em hora
RadH	$0.2617993878 * H$	hora em radianos

Tabela 4.1: Funções do tipo “macro”.

```
F Class type = F.class[FOCAS][DTREE];
```

vai produzir um resultado nulo porque se requer que os objetos sejam classificados ao mesmo tempo pelo FOCAS e pelo método DTREE. Muito embora a maioria dos objetos possua classificação nesses dois métodos, não existe um “subconjunto” da classificação DTREE no conjunto da classificação FOCAS. A consulta define o classificador e não os objetos classificados. Para se retornar os objetos classificados tanto pelo FOCAS quanto pelo DTREE, faz-se:

```
F Class type = F.class[FOCAS,DTREE];
```

4.2.3.6 Funções do Compilador

Além das funções implícitas do DBMS da Informix, algumas funções específicas de interesse astronômico foram introduzidas com o fim de facilitar a consulta por parte do usuário. O “texto” da consulta é modificado no sentido de produzir o resultado esperado quando é produzida a consulta em linguagem “SQL”. A tabela 4.1 fornece a listagem dessas funções. Do ponto de vista da programação, essas funções representam “macros”, pois as operações:

1. São composição de funções primárias, essas fazendo parte do conjunto de funções do sistema;
2. São aplicadas por substituição no momento da pré-compilação.

Uma consulta contendo uma instrução do tipo:

```
F Flux = Flux(J.mtot);
```

será transformada em uma coluna da forma:

```
exp(-0.9210340372*(J.mtot)) as F_Flux
```

Função	Parâmetros	Operação
<i>Funções algébricas</i>		
ABS	(número)	Valor absoluto
MOD	(dividendo,divisor)	Resto da divisão
POW	(base,expoente)	Potenciação
ROOT	(radicando>0,índice)	Raiz "N" de radicando
ROUND	(número,fator)	Arredonda a um fator
SQRT	(radicando > 0)	Raiz quadrada
TRUNC	(número,fator)	Trunca a um fator
<i>Funções logarítmicas</i>		
EXP	(float)	Exponencial
LOGN	(float>0)	Logarítmo natural
LOG10	(float>0)	Logarítmo base 10
<i>Funções Trigonométricas</i>		
COS	(radiano)	Cosseno
SIN	(radiano)	Seno
TAN	(radiano)	Tangente
ASIN	(float, float <=1)	Arco seno
ACOS	(float, float <=1)	Arco cosseno
ATAN	(float)	Arco tangente
ATAN2	(y,x)	Arco tangente no quadrante

Tabela 4.2: Funções agregadas do Informix

4.2.3.7 Funções Agregadas

O Informix oferece algumas funções agregadas que podem ser usadas sem restrições no SkiccQL. O "tradutor" repassa essas funções ao DBMS via SQL sem qualquer crítica. Erros que porventura sejam cometidos nessa instância serão tratados no nível do banco de dados e retornarão via "relatório" posterior. O SkiccQL não faz diagnóstico desse tipo de erro. A tabela 4.2 fornece uma relação das funções agregadas do Informix.

Algumas funções do Informix não foram listadas, particularmente aquelas de aplicação estatística (Average, Count, etc), porque não foi implementada qualquer regra de agrupamento. Essas regras poderiam se associar a um procedimento de indexação. No entanto, uma discussão futura mais aprofundada a respeito se faz necessária. Por outro lado, na medida em que o usuário receba uma tabela de dados, é possibilitado a ele, localmente, proceder a estatísticas de sua escolha. Para aplicar funções estatísticas no SKICCOSMO o usuário:

1. Abre mão da tabela com os dados para receber apenas o resultado das funções;

2. Tem de aprender a lidar com a programação dos macros do tradutor.

Visto que é possível proceder ao tratamento estatístico de conveniência do usuário e manter, ao mesmo tempo, os dados que recuperou do SKICCOSMO, considera-se que não há perda de operacionalidade em não se disponibilizar, pelo menos de imediato, as funções estatísticas do Informix.

4.2.3.8 Nomes de Variáveis

Os nomes adotados à esquerda da expressão na janela de atributos podem ser usados como variáveis a serem utilizadas em outras expressões, desde que obedecidas algumas condições. Entre as condições para utilizar as variáveis estão:

1. Os nomes de variáveis não podem possuir “espaço” no meio. Do contrário, esses espaços devem ser substituídos pelo sinal “_”, como será feito, efetivamente, quando da listagem dos dados. Por exemplo, se, em dada instrução, foi definido o atributo “Right Ascension = Hour(ra);”, em outra expressão em que se queira usar essa etiqueta como variável, devemos fazer: “Hour Angle = Right_Ascension - 2.1554”. O “tradutor” irá substituir os espaços em todas as etiquetas para, em seguida substituir variáveis por seus equivalentes. Se não se substituir o “espaço” por “_”, o tradutor não reconhecerá a variável;
2. Deve-se obedecer estritamente às maiúsculas e minúsculas. Para o tradutor, “Right Ascension” é diferente de “right ascension”. Para o Informix é indiferente letras maiúsculas e minúsculas, mas para o SkiccQL essa diferença existe.

4.2.3.9 Atributos Expostos e Escondidos

Atributos expostos são os ordinários, já descritos acima. Atributos escondidos são aqueles que não serão listados na tabela resultante. Definir atributos escondidos serve para forçar o retorno de objetos presentes nas bandas determinadas pela consulta, aquelas constando nos atributos expostos e escondidos.

No caso em que queiramos objetos pertencentes a duas bandas, para obter atributos em apenas uma delas, poderemos “esconder” a outra banda colocando o nome da variável “hidden”. Por exemplo, se quisermos os valores de “mtot” de objetos campo “f862” que também estão presentes nos campos “j862” e “n862”, fazemos:

```
F Total Mag = F.mtot[862];  
hidden = J.mtot[F.];  
hidden = N.mtot[F.];
```

O “tradutor” vai construir as junções e condições como se as três bandas tivessem sido escolhidas, salvo que não colocará os atributos como colunas na saída dos resultados.

Importante: a inclusão de atributos escondidos em uma banda já definida em outro atributo “exposto”, não tem qualquer função prática na consulta. O banco de dados vai receber uma instrução SQL contendo condições redundantes, aumentando o tempo de consulta.

4.2.4 As Condições

Uma vez definidos os atributos que retornarão do SKICCOSMO, passa-se à janela de definição das condições. Nela serão definidos os domínios e as equações de vínculo que vão determinar a especificidade da consulta. Um astrônomo, quando se dispõe a fazer uma consulta a um banco de dados, está interessado em uma certa classe de objetos obedecendo a algumas condições. Essas condições são estabelecidas nessa janela.

Diferentemente da determinação dos atributos, as condições são incluídas em uma só instrução. Novamente, a mudança de linha não é considerada. Cada condição é separada da outra por um operador lógico. Esses operadores são dois:

1. O operador “&” representa o “and”;
2. O operador “|” representa o “or”.

Importante: evite o uso do operador “or”. Para utilizá-lo, siga a regra descrita mais abaixo em 4.2.4.1, sob pena de produzir um “*overhead*” na memória tampão do banco de dados.

Operadores como “>”, “<” e “=” são permitidos. Eis, por exemplo, como se define uma região do céu em termos de coordenadas equatoriais:

```
Hour(ra) > 1.5 & Hour(ra) < 2.0
```

Se, por outro lado, já tínhamos definido a variável “Right Ascension = Hour(ra);” na definição das colunas, nas condições, podemos colocar:

```
Right_Ascension > 1.5 & Right_Ascension < 2.0
```

Qualquer expressão matemática é permitida como condição, desde que: 1) o produto da expressão seja do tipo booleano; 2) os atributos e constantes sejam de tipos comparáveis (ex., inteiros com inteiros e/ou números de ponto flutuante, etc).

4.2.4.1 Operador *or*

Consultas em “SQL” com o operador “or” possuem duas características:

1. Promovem a **união** dos conjuntos obtidos pelo resto da consulta;
2. A operação tem precedência sobre todas as outras. Assim, a instrução das condições de uma consulta é dividida em duas partes: a que vem antes do “or” e a que vem depois.

A forma de evitar um sério problema decorrente do uso do operador “or” é inserí-lo em uma expressão com parênteses (“()”) pois assim a operação será forçada a ser avaliada separadamente.

As consultas em geral dispensam o uso desse operador, visto que o astrônomo geralmente deseja informações sobre um universo restrito de dados. Se ocorrer o uso do “or”, a regra geral é a seguinte: procure inserí-lo numa expressão dentro de parênteses. Se o usuário não consegue encontrar, na expressão, os locais onde abrir e fechar os parênteses, então isso é um sinal inequívoco de que sua consulta **não** precisa do operador “or”.

Exemplo da utilização do operador “or” é a condição inserida por “*default*” para a classificação dos objetos (ver 4.2.4.2)

4.2.4.2 Condições Assumidas por Falta (*default*)

As condições assumidas em 4.1.4 onde são impostos valores a alguns atributos são chamadas de condições por falta e são transformadas em instruções do tipo:

```
& 'banda'.classifier = 'FOCAS'
& ( 'banda'.class = 's' or 'banda'.class = 'g' ) ...
```

Note os parênteses delimitando a expressão contendo o "or". Aqui "banda" é um dos diferentes valores das bandas escolhidas.

A definição dos valores por *default*, como já foi visto, pode ser modificada na manipulação dos parâmetros das sessões acima das janelas de construção da consulta ou nas instruções na janela de condições.

4.2.4.3 Inclusão e Exclusão

Dois operadores adicionais podem ser utilizados entre as condições. São eles:

include['banda'] |- Promove a inclusão de uma certa banda, impondo que os dados a serem retornados possuam valores nesta banda;

exclude['banda'] |- Impõe a exclusão da banda. Os objetos a serem retornados **não** podem pertencer aos dados possuindo valores nesta banda.

O comando de inclusão possui um correspondente aproximado na construção ordinária de atributos. É o recurso do atributo escondido. Por exemplo, a consulta:

Attribs:

```
F Total Mag = F.mtot[862];
hidden = J.mtot[F.];
```

é aproximadamente equivalente a:

Attribs:

```
F Total Mag = F.mtot[862];
```

Conds:

```
include[J.]
```

salvo que no primeiro caso serão impostas condições por "*default*" (ver 4.2.4.2), enquanto no segundo caso isso não será feito.

A exclusão não possui qualquer comando parecido ou próximo. É usada sobretudo por aqueles que querem recuperar objetos que existem em outras bandas mas que não podem pertencer ao campo da banda especificada no comando "exclude". Esse comando quando é combinado a outros permite promover uma operação de diferença entre o conjunto dos objetos presentes em uma banda ou combinação de duas bandas e o conjunto dos objetos presentes na banda definida no comando "exclude".

Importante: o comando “include” ou “exclude” de uma banda já escolhida nos atributos provoca uma mensagem de erro.

4.2.5 Erros e Limitações

Erros cometidos ao se construir uma consulta em SkiccQL poderão, ou não, ser diagnosticados pelo “tradutor”. Se forem detectados pelo “tradutor”, o usuário receberá uma nova página contendo uma tentativa de diagnóstico e terá a oportunidade de retornar ao formulário e corrigir o erro. Se não for diagnosticado, provavelmente será produzida uma instrução em “SQL” contendo erros que serão diagnosticados pelo DBMS. Nesse caso, o usuário tomará conhecimento dos erros no relatório após a consulta ser remetida ao DBMS.

Algumas construções da sintaxe do tradutor são proibidas:

- Não é permitido, por exemplo, adotar nomes de variáveis que coincidam com nomes de atributos do SKICCOSMO. Não é permitido fazer: “RA = ra;”, ou equivalente;
- Não é permitido incluir condições na janela de atributos e vice-versa;
- Não é permitido separar condições com “;”.

É fortemente recomendado que se verifique a região do céu e a faixa de magnitude da classe de objetos que se quer pesquisar. Ela deve constar no domínio que o SKICCOSMO cobre no momento da consulta. O SKICCOSMO é carregado permanentemente com mais objetos e campos de placas POSS-II e imagens CCD. No entanto, é possível que a região de interesse não faça parte do domínio coberto pelo SKICCOSMO. Verifica-se a região do céu contida no SKICCOSMO e a faixa de magnitude nos botões *Headers* na primeira página *Web* do SKICCOSMO.

A Tabela 4.3 mostra um resumo das regras do “SkiccQL”.

4.3 Exemplos

Alguns exemplos disponíveis na página de construção da consulta são expostos a seguir. A simplicidade dos exemplos tem o fim de permitir que as diferentes formas de construção de uma consulta sejam mostradas com clareza para o usuário. Fazendo uso combinado das formas de construção de uma consulta em SkiccQL e das funções - sejam do SkiccQL e as agregadas do SQL Informix -, o usuário já pode construir consultas mais complexas.

Tipo de Atributo	Tabela de Origem	Atributos Disponíveis	Regra de Construção	Indexação	Origem do Índice
Coordenadas Equatoriais	Objects	Ra, Dec			
Bandas	Cat_Techniques	Letra minúscula maiúscula	Terminado em ponto		
Atributos Fotométricos	DPOSSII_Features	Todos da tabela	Precedido por definição de banda	Platenumb / Banda	DPOSSII_Features
Classificação	Classifications	Class, Prob	Precedido por definição de banda	Classifier	Classifications
Construção da Consulta, Atributos					
Nome_da_Coluna = Banda.atributo[indice] [indice];					
Índices são opcionais					
Nome_da_Coluna pode constar ao lado direito da igualdade (devidamente definido)					
Construção da Consulta, Condições					
Primeira_condição & Segunda_condição & Terceira_condição ... ou "incluê/exclui (Banda.)"					
Condição é do tipo: Variável = <, >, <> Constante ou variável					
Variável pode ser: Banda.atributo ou Nome_de_Coluna					
Constante é um número ou letra, dependendo do tipo de atributo					
& = and					
= or					
Nota: evitar usar o operador (or)					

Tabela 4.3: Resumo do SkicQL.

Query Name:		Compression Method:	
Total Mag F J N near		Tar Compress	
Magnitude Range		Flags	
Min	15.0	<input type="checkbox"/> A2Flg Special attention	<input type="checkbox"/> A3Flg Perced classification
		<input type="checkbox"/> A32Flg Edge of field	<input type="checkbox"/> A4Flg Exceed limit of Focas
		<input type="checkbox"/> A33Flg Problem in classifier	<input type="checkbox"/> A42Flg Saturated pixel in object
Max	19.0	<input type="checkbox"/> A34Flg Below sky	<input type="checkbox"/> A5Flg Reference point
		<input type="checkbox"/> F1Flg Successfully estimated	<input type="checkbox"/> F2Flg Not split at any level
		Classification	
		Classifier	FOCAS
		Class Type	s + g
Attribute Definitions		Conditions	
F Total Mag = F.mtot[823]; J Total Mag = J.mtot[F.]; N Total Mag = N.mtot[F.];		F.icol = 7 & F.irow = 7 & J.icol = 7 & J.irow = 7 & N.icol = 7 & N.irow = 7	
<input type="button" value="Submit"/>		<input type="button" value="Reset"/>	

Figura 4.8: Aspecto da consulta *Total Mag F J N near equinox*

A Figura 4.8 mostra o aspecto da página de construção da consulta “carregada” com a consulta “*Total Mag F J N near equinox*” que será descrita a seguir.

4.3.1 Exemplo n. 1: *Total Mag F J N near equinox*

Abaixo, apresentamos a listagem do arquivo “Report.log” no diretório recuperado do “adms-kicc” através da página *QueryStatus*. Algumas linhas foram reformatadas para permitir sua legibilidade na listagem.

```

UserName: kohl@on.br
Conds: F.icol = 7 &F.irow = 7 &J.icol = 7 &J.irow = 7 &N.icol = 7
&N.irow = 7
QueryName: Total Mag F J N near equinox - 1
Classifiers: FOCAS
  
```

```

Attribs: F Total Mag = F.mtot[823];J Total Mag = J.mtot[F.];N Total
Mag = N.mtot[F.];
MinMag: 15.0
CompressMeth: Tar Compress
MaxMag: 19.0
userid: 2
SFlg:
EFlg:
Classifications: sg
SQL: select distinct F.mtot as F_Total_Mag, J.mtot as J_Total_Mag,
N.mtot as N_Total_Mag from objects o, features J, dpossii_headers JH,
classifications JC, features F, dpossii_headers FH, classifications
FC, features N, dpossii_headers NH, classifications NC where o.id_obj
= J.id_obj and J.id_header = JH.id_header and o.id_obj = F.id_obj and
F.id_header = FH.id_header and o.id_obj = N.id_obj and N.id_header =
NH.id_header and ( JH.band = 'J' and JH.platenumb = FH.platenumb) and
( o.id_obj = JC.id_obj and J.id_header = JC.id_header and
JC.classifier = 'FOCAS') and ( FH.band = 'F' and FH.platenumb = 823)
and ( o.id_obj = FC.id_obj and F.id_header = FC.id_header and
FC.classifier = 'FOCAS') and ( NH.band = 'N' and NH.platenumb =
FH.platenumb) and ( o.id_obj = NC.id_obj and N.id_header =
NC.id_header and NC.classifier = 'FOCAS') and F.icol = 7 and F.irow =
7 and J.icol = 7 and J.irow = 7 and N.icol = 7 and N.irow = 7 and
(JC.class = 's' or JC.class = 'g') and J.mtot >= 15.0 and J.mtot <=
19.0 and J.EFlg = 1 and J.SFlg = 1 and (FC.class = 's' or FC.class =
'g') and F.mtot >= 15.0 and F.mtot <= 19.0 and F.EFlg = 1 and F.SFlg =
1 and (NC.class = 's' or NC.class = 'g') and N.mtot >= 15.0 and N.mtot
<= 19.0 and N.EFlg = 1 and N.SFlg = 1
Start query: Sat Aug 19 09:25:04 2000
76 objects retrieved.
End query: Sat Aug 19 09:26:35 2000

```

Esse exemplo demonstra como definir atributos que se deseja obter: a magnitude total nas três bandas. A banda **F** está indexada na placa número 823 enquanto as outras bandas estão indiretamente indexadas à banda **F**, o que significa que somente os objetos do campo

823 serão recuperados.

Na janela de condições foram impostos vínculos para os atributos *iraw* e *icol*, atributos esses que representam os elementos da matrix 13×13 na qual o campo da placa é dividido. Em outras palavras, nas condições houve uma restrição para se varrer apenas 1/169 do campo 823.

No arquivo *Report.log*, visto acima, podemos ver qual foi a instrução SQL gerada, o número de objetos recuperados (76), a hora de início e fim da execução, donde podemos deduzir o tempo de processamento da consulta (1m 31s).

Abaixo podemos ver a parte inicial do arquivo *Results.dat*, obtido do diretório de resultados.

```
f_total_mag j_total_mag n_total_mag
DECIMAL(16) DECIMAL(16) DECIMAL(16)
15.9590 17.0690 16.9890
16.0480 16.9640 16.9860
16.2810 17.1960 17.2750
16.3190 17.0790 17.3840
16.4980 16.9410 17.2850
16.5510 16.9240 17.7360
16.5560 17.3350 17.4020
16.70 17.5240 17.6710
16.71 17.22 17.5880
16.7970 17.6110 17.6760
16.80 17.8690 16.8210
16.8280 17.3860 17.6920
16.8430 17.3180 17.61
16.8670 17.0970 17.9290
```

As colunas nessa tabela são separadas pelo caracter **TAB**. Note-se os nomes das colunas segundo a definição dos atributos, salvo que todas as letras são forçadas à caixa baixa, conforme já descrito. Na segunda linha vem listado o tipo numérico da coluna, com o número de dígitos obtido na consulta. Como os dígitos significativos dos atributos retornados são apenas os 8 primeiros (a magnitude total é deduzida a partir de um número inteiro), o resto das casas decimais são omitidas pelo *driver* de Entrada / Saída do *officeboy*. A palavra chave **DECIMAL**, portanto, deve ser substituída por **float** (simples precisão), com o número entre parênteses ignorado.

4.3.2 Exemplo n. 2: *Finding chart for Field F 836*

Abaixo, vemos a listagem do arquivo *Report.log*:

```

UserName: kohl@on.br
Conds: F.irow = 5 &F.icol = 3
QueryName: Finding chart for Field F
Classifiers: FOCAS
Attribs: Right Ascension = Hour(ra);Declination = Deg(dec);hidden =
F.mtot[836];
MinMag: 12.0
CompressMeth: Tar Gzip
MaxMag: 21.0
userid: 2
SFlg:
EFlg:
Classifications: sg
SQL: select 3.8197186342*(o.ra) as Right_Ascension,
57.295779513*(o.dec) as Declination from objects o, features F,
dpossii_headers FH, classifications FC where o.id_obj = F.id_obj and
F.id_header = FH.id_header and FH.band = 'F' and ( FH.platenumb = 836)
and ( o.id_obj = FC.id_obj and F.id_header = FC.id_header and
FC.classifier = 'FOCAS') and F.irow = 5 and F.icol = 3 and (FC.class =
's' or FC.class = 'g') and F.mtot >= 12.0 and F.mtot <= 21.0 and
F.EFlg = 1 and F.SFlg = 1
Start query: Fri May 26 18:25:02 2000
5714 objects retrieved.
End query: Fri May 26 18:27:07 2000

```

Esses dados poderiam servir para um *Finding Chart*, se adicionássemos a classificação dos objetos, - caso sejam ou não galáxias - além de suas propriedades fotométricas. O exemplo serve para demonstrar a utilização de uma variável “escondida” (ver na linha *Attribs*, o atributo definido como *hidden*). Nesse caso não se quis que a variável “F.mtot” fosse listada no resultado, forçando a consulta a procurar os objetos na banda “F”, visto não ter sido definida outra banda. Se não há bandas definidas, nem entre os atributos, nem entre as condições, o banco de dados faz retornar um erro.

Novamente, a banda “F” é indexada no campo 836 do POSS-II na parte contida em “irow = 5” e “icol = 3”. Ainda assim, escolhendo esse pequeno campo, foram selecionados 5714 objetos, porque ampliou-se demasiadamente o intervalo de magnitude (12.0 e 21.0).

4.3.3 Exemplo n. 3: *Color - Color Diagram for stars*

Abaixo, vemos a listagem do arquivo *Report.log*:

```

UserName: kohl@on.br
Conds: F.irow = 6 &F.icol = 6
QueryName: Color - Color Diagram for stars
Classifiers: FOCAS
Attribs: F Color Index = F.mtot[827]-N.mtot[F.];J Color Index =
J.mtot[F.]-N.mtot;
MinMag: 15.0
CompressMeth: Zip File
MaxMag: 19.0
userid: 2
SFlg:
EFlg:
Classifications: s
SQL: select F.mtot-N.mtot as F_Color_Index, J.mtot-N.mtot as
J_Color_Index from objects o, features J, dpossii_headers JH,
classifications JC, features N, dpossii_headers NH, classifications
NC, features F, dpossii_headers FH, classifications FC where o.id_obj
= J.id_obj and J.id_header = JH.id_header and JH.band = 'J' and
o.id_obj = N.id_obj and N.id_header = NH.id_header and NH.band = 'N'
and o.id_obj = F.id_obj and F.id_header = FH.id_header and FH.band =
'F' and ( JH.platenumb = FH.platenumb) and ( o.id_obj = JC.id_obj and
J.id_header = JC.id_header and JC.classifier = 'FOCAS') and (
NH.platenumb = FH.platenumb) and ( o.id_obj = NC.id_obj and
N.id_header = NC.id_header and NC.classifier = 'FOCAS') and (
FH.platenumb = 827) and ( o.id_obj = FC.id_obj and F.id_header =
FC.id_header and FC.classifier = 'FOCAS') and F.irow = 6 and F.icol =
6 and JC.class = 's' and J.mtot >= 15.0 and J.mtot <= 19.0 and J.EFlg

```

```
= 1 and J.SFlg = 1 and NC.class = 's' and N.mtot >= 15.0 and N.mtot <=
19.0 and N.EFlg = 1 and N.SFlg = 1 and FC.class = 's' and F.mtot >=
15.0 and F.mtot <= 19.0 and F.EFlg = 1 and F.SFlg = 1
Start query: Mon May 29 16:00:01 2000
42 objects retrieved.
End query: Mon May 29 16:03:53 2000
```

Esse exemplo serve para mostrar uma construção simples de expressões nos atributos. Vê-se as diferenças aritméticas entre atributos. Essencialmente qualquer operação aritmética é permitida desde que resulte em valores finitos. Mais uma vez, há indexação direta e indireta nos atributos fotométricos. Escolheu-se impor que os objetos estejam perto do equador impondo "F.irow=6". A consulta é muito mais rápida do que restringir valores para a ascensão reta. O valor "F.icol=6" foi imposto apenas para ilustração. Sua função é para restringir a declinação.

4.3.4 Exemplo n. 4: *Choose galaxies in J and N but not in F*

Abaixo, vemos a listagem do arquivo *Report.log*:

```
UserName: kohl@on.br
Conds: exclude(F.) &J.irow = 3 &J.icol = 5
QueryName: Choose galaxies in J and N but not in F
Classifiers: FOCAS
Attribs: Right Ascension = Hour(ra);Declination = Deg(dec);J Aperture
mag = J.maper[830];N Aperture mag = N.maper[J.];
MinMag: 19.0
CompressMeth: Tar Gzip
MaxMag: 21.0
userid: 2
SFlg:
EFlg:
Classifications: g
SQL: select distinct 3.8197186342*(o.ra) as Right_Ascension,
57.295779513*(o.dec) as Declination, J.maper as J_Aperture_mag, N.maper
as N_Aperture_mag from objects o, features J, dpossii_headers JH,
classifications JC, features F, dpossii_headers FH, classifications FC,
```

```

features N, dpossii_headers NH, classifications NC where o.id_obj =
J.id_obj and J.id_header = JH.id_header and o.id_obj = F.id_obj and
F.id_header = FH.id_header and o.id_obj = N.id_obj and N.id_header =
NH.id_header and ( JH.band = 'J' and JH.platenumb = 830) and ( o.id_obj
= JC.id_obj and J.id_header = JC.id_header and JC.classifier = 'FOCAS')
and ( F.id_header not in (select id_header from dpossii_headers where
band = 'F')) and ( o.id_obj = FC.id_obj and F.id_header = FC.id_header
and FC.classifier = 'FOCAS') and ( NH.band = 'N' and NH.platenumb =
JH.platenumb) and ( o.id_obj = NC.id_obj and N.id_header = NC.id_header
and NC.classifier = 'FOCAS') and J.irow = 3 and J.icol = 5 and JC.class =
'g' and J.mtot >= 19.0 and J.mtot <= 21.0 and J.EFlg = 1 and J.SFlg =
1 and FC.class = 'g' and F.mtot >= 19.0 and F.mtot <= 21.0 and F.EFlg =
1 and F.SFlg = 1 and NC.class = 'g' and N.mtot >= 19.0 and N.mtot <=
21.0 and N.EFlg = 1 and N.SFlg = 1
Start query: Mon Aug 7 19:50:04 2000
12 objects retrieved.
End query: Mon Aug 7 19:51:47 2000

```

Essa consulta serve para mostrar a ferramenta “exclude” na janela das condições. Aqui escolheu-se as galáxias que estão presentes (dentro dos limites impostos) nas bandas “N” e “J” e que não estão presentes na banda “F”. Como era de se esperar, o número de casos é pequeno, mesmo para a faixa de magnitude considerada (entre 19.0 e 21.0).

4.3.5 Exemplo n. 5: *Finding Chart for stars where DEX(F-N)>10*

Abaixo, vemos a listagem do arquivo *Report.log*:

```

UserName: kohl@on.br
Conds: Flux(F_total_Mag-N_total_Mag) > 10.0 &F.irow = 5 &F.icol = 5
QueryName: Finding Chart for stars where DEX(F-N)>10
Classifiers: FOCAS
Attribs: Right Ascension = Deg(ra);Declination = Deg(dec);F total Mag
= F.mtot[827];N total Mag = N.mtot[827];
MinMag: 15.0
CompressMeth: Tar Compress
MaxMag: 19.0

```



```

userid: 2
SFlg:
EFlg:
Classifications: ns
SQL: select 57.295779513*(o.ra) as Right_Ascension,
57.295779513*(o.dec) as Declination, F.mtot as F_total_Mag, N.mtot as
N_total_Mag from objects o, features F, dpossii_headers FH,
classifications FC, features N, dpossii_headers NH, classifications NC
where o.id_obj = F.id_obj and F.id_header = FH.id_header and FH.band =
'F' and o.id_obj = N.id_obj and N.id_header = NH.id_header and NH.band =
'N' and ( FH.platenumb = 827) and ( o.id_obj = FC.id_obj and
F.id_header = FC.id_header and FC.classifier = 'FOCAS') and (
NH.platenumb = 827) and ( o.id_obj = NC.id_obj and N.id_header =
NC.id_header and NC.classifier = 'FOCAS') and
exp(-0.9210340372*(F.mtot-N.mtot)) > 10.0 and F.irow = 5 and F.icol =
5 and FC.class != 's' and F.mtot >= 15.0 and F.mtot <= 19.0 and F.EFlg
= 1 and F.SFlg = 1 and NC.class != 's' and N.mtot >= 15.0 and N.mtot
<= 19.0 and N.EFlg = 1 and N.SFlg = 1
Start query: Mon May 29 17:30:02 2000
4 objects retrieved.
End query: Mon May 29 17:32:59 2000

```

Essa consulta serve para mostrar uma expressão matemática na janela das condições. Podemos perceber da instrução SQL gerada que a “função” Flux é substituída para as funções e expressões aritméticas às quais ela corresponde: $\exp(-0.9210340372 * (F.mtot - N.mtot))$.

4.3.6 Exemplo n. 6: *Stars & galaxies within a small area (N-band)*

Nesse exemplo usamos a variável “Declination”, definida na área de atributos (Declination = Deg(dec)) e a usamos para impor, na área de condições, que seu valor em minutos (Declination*60) esteja dentro de uma região entre -1 e 1. Deseja-se, igualmente a recuperação da união dos objetos pertencentes aos campos 823 e 824. A classificação impõe que retornem apenas os objetos classificados como estrela ou galáxia pelo FOCAS. Esse é um exemplo de indexação múltipla nos campos 823 e 824.

Abaixo, vemos a listagem do arquivo *Report.log*:

```

UserName: kohl@on.br
Conds: Declination*60 <= 1 &Declination*60 >= -1
QueryName: Stars and Galaxies within a small area (N-band)
Classifiers: FOCAS
Attribs: Right Ascension = Hour(ra);Declination = Deg(dec);N Total Mag
= N.mtot[823,824];
MinMag: 15.0
CompressMeth: Tar Compress
MaxMag: 19.0
userid: 2
SFlg:
EFlg:
Classifications: sg
SQL: select 3.8197186342*(o.ra) as Right_Ascension,
57.295779513*(o.dec) as Declination, N.mtot as N_Total_Mag from
objects o, features N, dpossii_headers NH, classifications NC where
o.id_obj = N.id_obj and N.id_header = NH.id_header and NH.band = 'N'
and ( NH.platenumb = 823 or NH.platenumb = 824) and ( o.id_obj =
NC.id_obj and N.id_header = NC.id_header and NC.classifier = 'FOCAS')
and 57.295779513*(o.dec)*60 <= 1 and 57.295779513*(o.dec)*60 >= -1 and
(NC.class = 's' or NC.class = 'g') and N.mtot >= 15.0 and N.mtot <=
19.0 and N.EFlg = 1 and N.SFlg = 1
Start query: Mon May 29 17:45:02 2000
1200 objects retrieved.
End query: Mon May 29 18:37:59 2000

```

4.3.7 Exemplo n. 7: *Round Galaxies from F 861*

Queremos recuperar galáxias cuja característica é sua baixa excentricidade na banda “F.” mas somente aquelas que também estejam presentes na banda “J.” e pertençam ao campo 861. Abaixo, vemos a listagem do arquivo *Report.log*:

```

UserName: kohl@on.br
Conds: include(J.) &ABS(F.asymm) <= 0.3 &F.icol > 2 & F.icol < 11

```

```

&F.irow > 2 & F.irow < 11
QueryName: Round Galaxies from F 861
Classifiers: FOCAS
Attribs: Right Ascension = Hour(ra);Declination = Deg(Dec);F Total Mag =
F.mtot;F Core Mag = F.mcore[861];F Postition Angle = Deg(F.pa);F Asymmetry
= F.asymm;
MinMag: 17.0
CompressMeth: Tar Gzip
MaxMag: 20.0
userid: 2
SFlg:
EFlg:
Classifications: g
SQL: select distinct 3.8197186342*(o.ra) as Right_Ascension,
57.295779513*(o.Dec) as Declination, F.mtot as F_Total_Mag, F.mcore
as F_Core_Mag, 57.295779513*(F.pa) as F_Postition_Angle, F.asymm
as F_Asymmetry from objects o, features J, dpossii_headers JH,
classifications JC, features F, dpossii_headers FH, classifications FC
where o.id_obj = J.id_obj and J.id_header = JH.id_header and o.id_obj =
F.id_obj and F.id_header = FH.id_header and ( J.id_header in (select
id_header from dpossii_headers where band = 'J')) and ( o.id_obj =
JC.id_obj and J.id_header = JC.id_header and JC.classifier = 'FOCAS') and
( o.id_obj = FC.id_obj and F.id_header = FC.id_header and FC.classifier =
'FOCAS') and ( FH.band = 'F' and FH.platenumb = 861) and ABS(F.asymm)
<= 0.3 and F.icol > 2 and F.icol < 11 and F.irow > 2 and F.irow < 11
and JC.class = 'g' and J.mtot >= 17.0 and J.mtot <= 20.0 and J.EFlg =
1 and J.SFlg = 1 and FC.class = 'g' and F.mtot >= 17.0 and F.mtot <=
20.0 and F.EFlg = 1 and F.SFlg = 1
Start query: Mon Aug 7 20:00:04 2000
2878 objects retrieved.
End query: Mon Aug 7 20:12:19 2000

```

Esse é um exemplo da ferramenta “include”, que é o inverso de “exclude”. No entanto, enquanto não há outra forma de fazer uma consulta que corresponda àquela construída com “exclude”, a consulta construída com “include” é equivalente àquela construída com

a palavra chave “hidden” entre os atributos. As instruções SQL geradas serão ligeiramente diferentes. A consulta com “hidden” gera inclusão de auto-junções, enquanto a construída com “include” gera uma consulta dotada de “sub-consulta”, isto é, uma consulta gerando um conjunto de atributos que serão usados na consulta mais geral. Não obstante, o resultado das duas consultas será o mesmo, a menos dos valores adotados por *default* (ver 4.2.4.3).

4.3.8 Exemplo n. 8: *QSO at high redshifts*

A procura por QSOs em altos *redshift* se utiliza extensivamente da seguinte consulta no SKICCOSMO (incluindo os termos da calibração fotométrica) :

```

UserName: kohl@on.br
Conds: F.mtot<= 19.5
QueryName: QSO at high redshifts
Classifiers: FOCAS
Attribs: GR Color = 1.062*J.mcore[826]-1.023*F.mcore[J.]-1.055;
RI Color = 1.023*F.mcore-1.016*N.mcore[J.]+1.577;
MinMag: 14.0
CompressMeth: Tar Gzip
MaxMag: 23.0
userid: 2
SFlg:
EFlg:
Classifications: s
SQL: select distinct 1.062*J.mcore-1.023*F.mcore-1.055 as GR_Color,
1.023*F.mcore-1.016*N.mcore+1.577 as RI_Color from objects o, features
J, dpossii_headers JH, classifications JC, features N, dpossii_headers
NH, classifications NC, features F, dpossii_headers FH,
classifications FC where o.id_obj = J.id_obj and J.id_header =
JH.id_header and o.id_obj = N.id_obj and N.id_header = NH.id_header
and o.id_obj = F.id_obj and F.id_header = FH.id_header and ( o.id_obj
= JC.id_obj and J.id_header = JC.id_header and JC.classifier =
'FOCAS') and ( JH.band = 'J' and JH.platenumb = 826) and ( o.id_obj =
FC.id_obj and F.id_header = FC.id_header and FC.classifier = 'FOCAS')
```

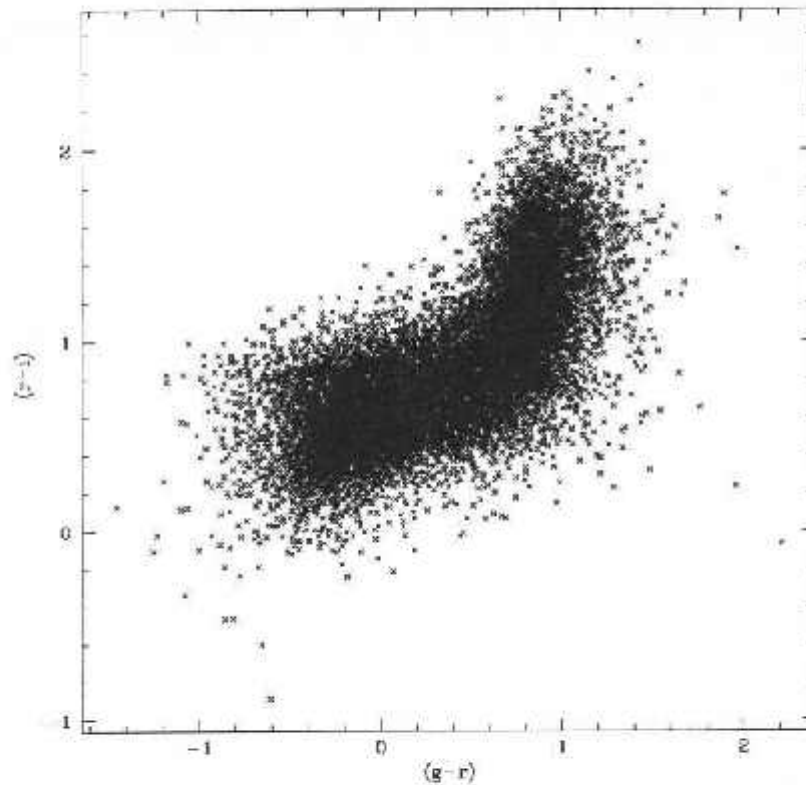


Figura 4.9: Diagrama Cor - Cor para o campo 826 do POSS-II.

```
and ( FH.band = 'F' and FH.platenumb = JH.platenumb) and ( o.id_obj =
NC.id_obj and N.id_header = NC.id_header and NC.classifier = 'FOCAS')
and ( NH.band = 'N' and NH.platenumb = JH.platenumb) and F.mtot<= 19.5
and JC.class = 's' and J.mtot >= 14.0 and J.mtot <= 23.0 and J.EFlg =
1 and J.SFlg = 1 and NC.class = 's' and N.mtot >= 14.0 and N.mtot <=
23.0 and N.EFlg = 1 and N.SFlg = 1 and FC.class = 's' and F.EFlg = 1
and F.SFlg = 1
```

Start query: Mon Aug 21 17:45:06 2000

14877 objects retrieved.

End query: Mon Aug 21 19:18:44 2000

O diagrama cor-cor derivado dessa consulta pode ser visto na Figura 4.9.

Capítulo 5

Utilizando o DPOSS na Procura de QSOs a $z > 4$

5.1 Introdução

Neste capítulo utilizamos as ferramentas desenvolvidas durante o projeto desta tese, na procura de QSOs em redshifts $4 < z < 4.8$. Os dados utilizados neste estudo são os catálogos digitalizados do POSS-II, os quais são descritos em detalhe em Reid *et al.*, 1991 ([48]). Apresentamos aqui resultados preliminares deste levantamento onde 24 novos QSOs foram encontrados através de uma metodologia onde as cores (g-r) e (r-i) são usadas. Considerando o número total de campos disponíveis no POSS-II (894), é fundamental que o processo de *matching* das três imagens em JFN para um dado campo possua uma alta eficiência, garantindo que o levantamento espectroscópico dos candidatos a QSOs selecionados a partir do POSS-II possa ser realizado em curto prazo de tempo.

QSOs, considerados como uma classe distinta de objetos astronômicos, formam provavelmente uma das ferramentas cosmológicas mais poderosas no estudo dos processos de formação de galáxias e no estabelecimento da estrutura em grande escala no Universo. Desde os estágios iniciais do estudo desta classe de objetos percebeu-se que estes não estavam distribuídos uniformemente no espaço. Na verdade, o número de QSOs aumenta com o *redshift* (Schmidt 1968, [52]). Até o momento a forma e a evolução da função de luminosidade dos QSOs é bem conhecida somente até limites de redshift da ordem de 2.2 (Boyle *et al.*, 1988 [7]; Hewett *et al.*, 1993 [27]). Nestes trabalhos é mostrado que o número de QSOs aumenta continuamente até $z \sim 2.2$. É razoável se esperar dentro dos cenários mais aceitos para formação de estruturas, que o número destes sistemas deve começar a

decrecer a partir de um dado redshift, uma vez que estes objetos precisam de uma certa escala de tempo para formar e dar início ao processo de ignição (Haehnelt & Rees, 1993 [25]). O ponto de declínio poderia indicar a época de formação de galáxias e estabelecer um importante vínculo para os modelos de formação de estruturas no Universo, além de obviamente fornecer parâmetros essenciais no estudo da natureza intrínseca dos QSOs.

Dada a importância, então, de se definir a distribuição destes objetos no domínio de redshift $2.2 < z < 5.0$, vários grupos iniciaram projetos de procura de QSOs utilizando diferentes métodos. Schneider *et al.*, 1994 ([53]) completaram um levantamento de 62 graus quadrados procurando por QSOs com $z > 2.7$, e estimaram a função de luminosidade no domínio $2.7 < z < 4.7$ a partir dos 90 QSOs identificados através da emissão em Ly- α (Schmidt *et al.*, 1995 [51]). Warren *et al.*, 1994 ([65]) estimaram a função de luminosidade de QSOs entre $2.2 < z < 4.5$ a partir de um levantamento de placas fotográficas semelhantes ao usado aqui nesta tese. Sua área de cobertura era de 47 graus quadrados. Ambos os levantamentos foram bem sucedidos em encontrar QSOs em altos redshifts. No entanto, esses trabalhos se mostraram pouco eficazes em detectar QSOs brilhantes, dada a pequena cobertura angular utilizada nos dois trabalhos. Irwin *et al.* (1991) conduziram um levantamento de ~ 2500 graus quadrados a partir do catálogo do APM e descobriram 27 QSOs com $z > 4$. Os grupos de Schmidt *et al.* (1995, [51]) e Warren *et al.* (1994, [65]) encontram evidências de que a densidade espacial de QSOs decresce para $z > 3$, enquanto Irwin *et al.* (1991) não encontraram evolução na função de luminosidade de QSOs entre $z = 2$ e $z = 4$. Tal discrepância pode ser devida a efeitos de seleção em alguma dessas amostras ou métodos de procura destes QSOs.

A fim de estabelecer a densidade espacial de QSOs brilhantes em altos domínios de redshift ($z > 4$) e ampliar a lista de QSOs que permitam estudar os sistemas de absorção, Kennefick *et al.* (1995, [36]) iniciaram um levantamento de QSOs brilhantes com redshift entre $4.0 < z < 4.8$ utilizando os catálogos gerados a partir do POSS-II (Djorgovski *et al.* 1992, [17]; Lasker *et al.* 1992 [40]; Reid & Djorgovski 1993, [49]). As placas do POSS-II foram tomadas em JFN e calibradas em gri do sistema de Gunn-Thuan, fazendo deste material uma importante fonte de procura de QSOs no intervalo de redshift de $4.0 < z < 4.8$. A técnica de usar as cores disponíveis se baseia no fato de que neste domínio de redshift a linha de emissão de Ly- α se desloca para a região onde se define a banda fotométrica r do sistema de Gunn-Thuan. Além disso, o contínuo apresenta uma "quebra" significativa na região ao azul de Ly- α onde se encontra a "floresta" de Lyman, resultando num valor alto para a cor (g-r). Esta característica fotométrica faz com que os

QSOs se deslocem no diagrama cor-cor ($g-r$) versus ($r-i$) se separando das estrelas (Figura 5.1). Kenefick *et al.* (1995, [36]), usando esta metodologia, examinaram 27 campos do POSS-II, representando uma área efetiva de 681 graus quadrados, e descobriram 10 QSOs com $z > 4$. A partir deste resultado um esforço considerável foi feito no sentido de tornar o processo o mais eficiente possível. Tanto no Observatório Nacional como em Caltech, foram desenvolvidos sistemas de otimização de criação de catálogos e *matching* dos mesmos, procedimentos estes essenciais para o sucesso deste nosso projeto. O apêndice A descreve o procedimento de preparação de processamento da imagem digitalizada de um campo do POSS-II. Com este procedimento foi possível automatizar parte fundamental da geração de catálogos, além de garantir a consistência dos parâmetros medidos para cada objeto detectado.

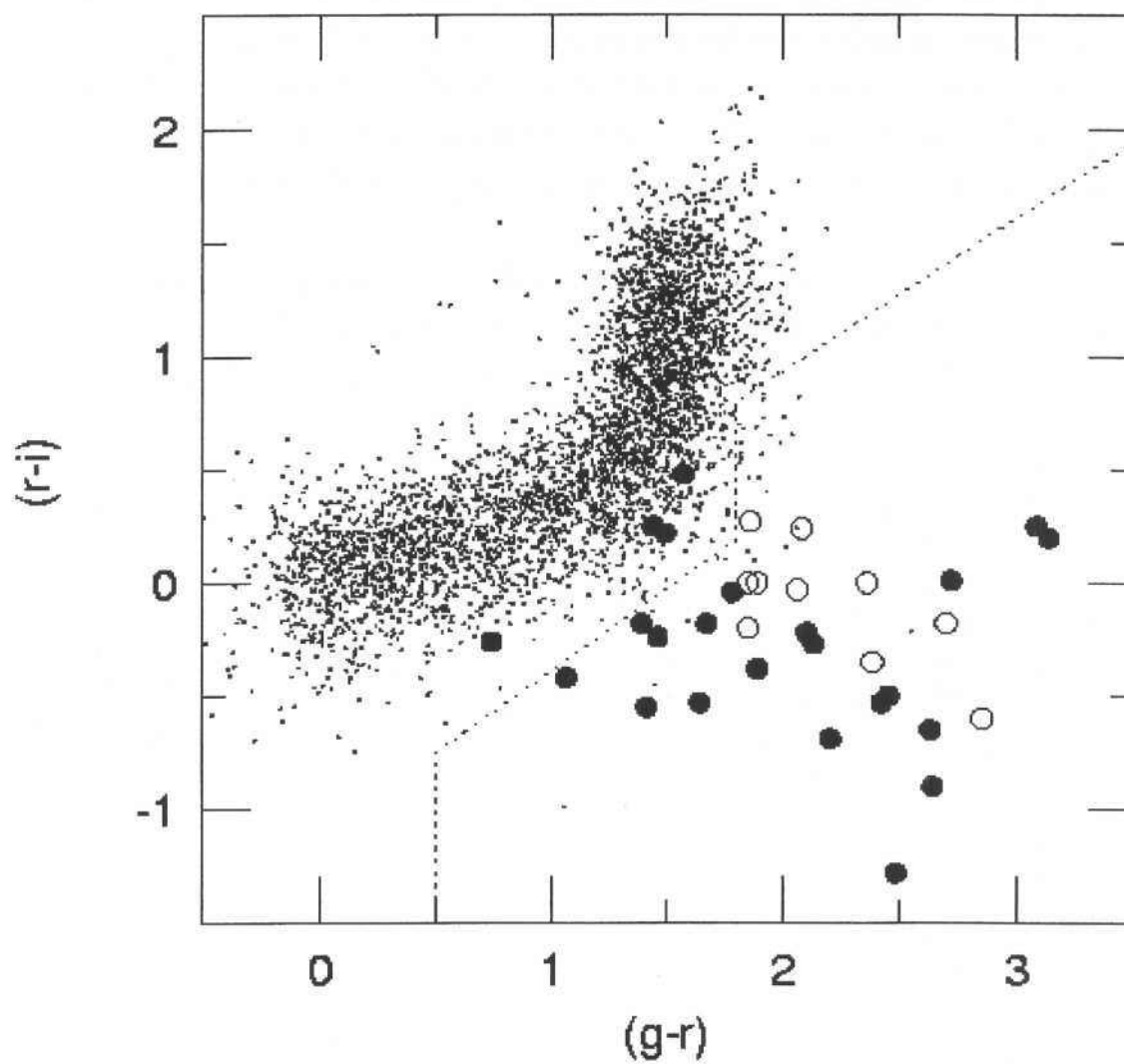
Neste capítulo, dando continuidade ao projeto iniciado por Kenefick *et al.* (1995, [36]), mostraremos resultados que são de natureza preliminar uma vez que o levantamento dos QSOs brilhantes em altos redshifts ainda está sendo realizado (Djorgovski *et al.* 2000 [18]).

5.2 O Levantamento Fotométrico

As placas fotográficas utilizadas neste trabalho foram obtidas como parte do segundo levantamento fotográfico de Palomar (POSS-II) no telescópio Schmidt Oschin de 1m. Este levantamento cobre todo o hemisfério norte em três bandas fotométricas distintas, JFN (Reid *et al.*, 1991 [48]). Maiores detalhes sobre as características deste levantamento podem ser encontradas em Weir, 1995 ([67]).

Nesta tese focalizamos nossa atenção em candidatos a QSOs que foram selecionados a partir de 20 campos diferentes (Vide Tabela 5.1). Todos estes campos estão em latitude galática acima de 30 graus, a fim de minimizar o efeito de contaminação na lista de candidatos por estrelas "vermelhas". A área total coberta por estes campos é da ordem de 700 graus quadrados.

Uma vez que os campos são digitalizados, estas imagens são processadas utilizando um pacote desenvolvido como parte da tese de doutoramento de Weir (1995). Optimizamos, aqui, alguns procedimentos dentro deste pacote a fim de agilizar o processo de procura de QSOs em altos *redshifts* (Vide Apêndice A). O processo de detecção gera catálogos completos até limites de magnitude de ~ 20.5 em g , ~ 21.5 em r , e ~ 20.0 em i . O item classificação é de fundamental importância para o projeto de procura de QSOs em altos

Figura 5.1: Diagrama $(g-r)$ versus $(r-i)$

QSO	z	Campo	Época (UT)	g	r	i
PSS0003-2730	4.26	409	08Oct96	21.10	19.21	19.59
PSS0030+1702	4.305	608	18Sep95	21.30	19.63	19.81*
PSS0034+1639	4.38	609	18Sep95	20.70	19.13	18.65+
PSS0050+2338	4.28	474	02Sep97	21.84	19.21	19.86
PSS0052+2405	4.28	474	02Sep97	20.41	18.21	18.90
PSS0106+2601	4.32	475	09Oct96	20.95	19.46	19.24
PSS0117+1552	4.24	611	18Sep95	20.47	18.34	18.61
PSS0132+1341	4.16	611	20Sep95	21.55	19.13	19.66
PSS0134+3307	4.52	413	11Nov96	22.35	19.22	19.02*
PSS0137+2837	4.30	413	11Nov96	22.17	19.45	19.44*
PSS0152+0735	4.067	684	20Sep95	21.94	19.46	20.74*
PSS0244-0108	4.010	831	20Sep95	20.15	18.69	18.93
PSS0747+4434	4.42	257	06Feb97	21.72	19.08	19.98
PSS1159+1337	4.08	643	19Apr96	18.97	17.58	17.76+
PSS1253-0228	4.0	861	07May97	19.80	19.06	19.32
PSS1456+2007	4.25	581	19Apr96	21.96	19.51	20.01+
PSS1618+4125	4.21	330	19Apr96	21.15	20.09	20.51*
PSS1633+1411	4.35	657	23Sep97	21.08	18.98	19.20
PSS1646+5514	4.05	180	20May96	19.18	17.74	17.49*
PSS1721+3256	4.031	392	16Aug96	20.96	19.18	19.22*
PSS2122-0014	4.18	887	12Nov96	22.75	19.67	19.42*
GB1508+5714	4	177	19Apr96	21.68	20.04	20.57*
BRI0241-0146	4.042	831	—	19.92	18.51	19.06—
BRI0952-0115	4.43	853	—	22.04	18.70	—*

Tabela 5.1: Campos estudados para os candidatos a QSOs.

* Calibração feita usando o campo adjacente.

+ Calibração precária

redshifts. Uma eficiente separação estrela-galáxia minimiza a contaminação por galáxias, de forma que um número menor de candidatos a QSOs deverão ser observados. Neste sentido, dois algoritmos especializados foram aplicados aos nossos dados. Um código de "árvore de decisão" foi desenvolvido por Weir (1995, [68]) especificamente para este trabalho, e mais recentemente utilizamos o código de redes neurais desenvolvido por Odewhan (1999, [46]). Usamos como classificação final para um objeto a classificação média entre as duas classificações descritas anteriormente¹, considerando os três catálogos disponíveis, **JFN**. Utilizamos uma média ponderada pelo *seeing* de cada placa. Vários testes foram feitos com dados CCD tomados no telescópio de 5m de Palomar para verificar a confiabilidade do classificador e atualmente, com a base de dados de treinamento obtida, alcançamos uma completude de 90% com 10% de contaminação em magnitude $r \sim 20.0$ e $g \sim 20.5$. Dada a natureza dos objetos que estamos procurando, QSOs que possuem altos valores de $(g-r)$, baseamos nossa seleção de estrelas no catálogo na banda **F**, e adotamos um limite conservador de 19.6 como limite deste levantamento de QSOs brilhantes em altos redshifts.

Para a calibração dos campos do POSS-II foram obtidos dados CCD de aglomerados de galáxias com o telescópio de 1.5m de Palomar. Estes dados foram tomados nos filtros **gri** do sistema de Gunn-Thuan (Thuan & Gunn 1976, [60]). Atualmente contamos com uma base de dados de cerca de 700 aglomerados de Abell observados nestas três bandas fotométricas.

Uma vez que os catálogos finais, calibrados, são construídos inicia-se o processo de *matching*. Este processo, como discutido em 3.5.2.1, é feito simultaneamente ao processo de armazenamento dos dados na estrutura do banco de dados que foi projetado nesta tese. Para que possamos usar esta amostra na estimativa da densidade espacial de QSOs em altos redshifts, é fundamental que a seleção dos candidatos a QSOs seja feita de maneira homogênea para todos os campos do POSS-II. Inicialmente selecionamos como candidatos a QSO os objetos classificados como estrelas na placa **F** e que tenha uma magnitude de $16.5 < r < 19.6$. A partir desta seleção, temos quatro casos a considerar: um deles, onde o objeto foi detectado nas três bandas; e três casos onde o objeto não foi detectado em uma ou duas das placas **J** e **F**. Na Figura 5.1 mostramos um diagrama cor-cor típico de um campo do POSS-II utilizado neste trabalho. Os círculos fechados representam todos os candidatos a QSO que foram confirmados através de observação espectroscópica (Vide também Tabela 5.1). Nesta figura também apresentamos os contornos de definição do que consideramos ser um objeto afastado o suficiente do locus estelar para ser considerado um

¹Ver em 3.2.2.4.

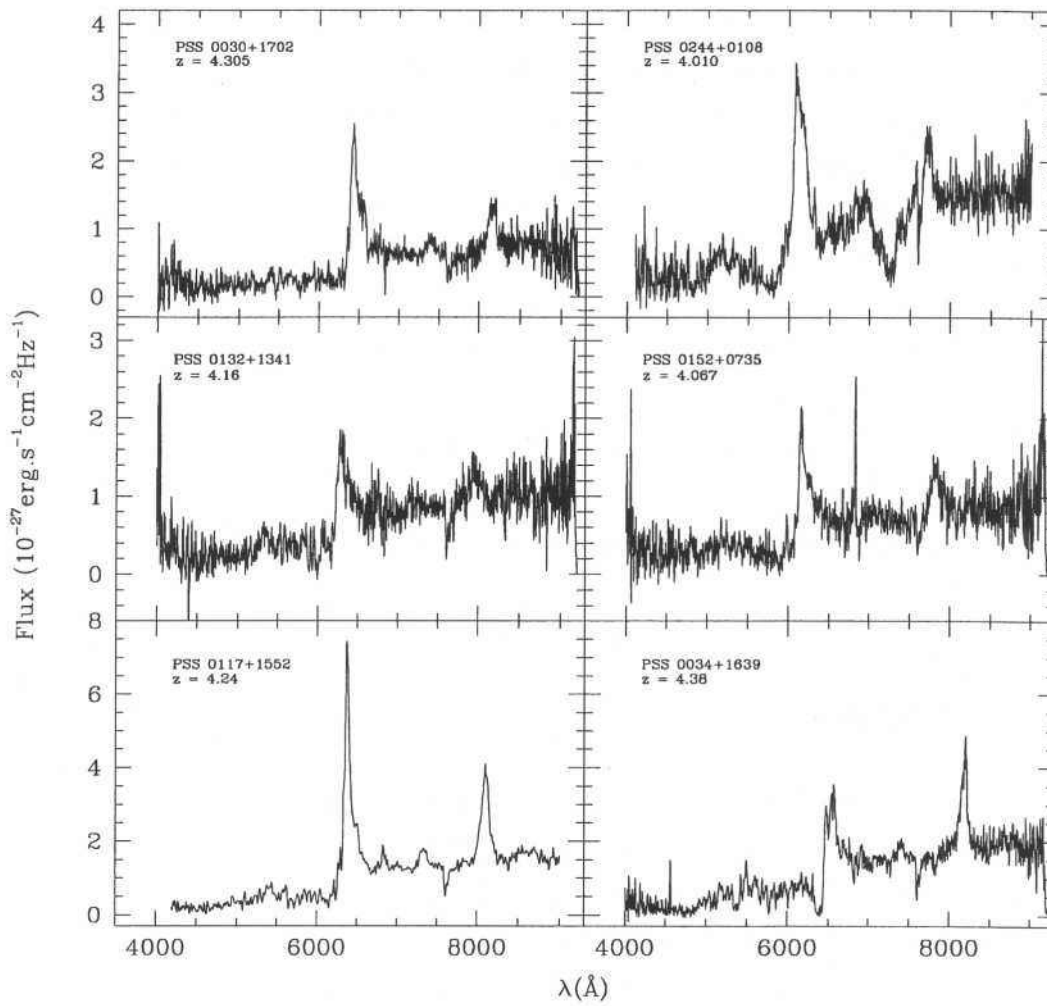
candidato a QSO. Estes limites de cores são usados posteriormente no cálculo da densidade espacial de QSOs que em última instância será comparado com os modelos de formação de estruturas.

5.3 O Levantamento Espectroscópico

A partir da construção da lista de candidatos a QSOs em altos redshifts, iniciamos o levantamento espectroscópico que irá confirmar ou não a natureza do objeto em questão. As observações foram realizadas no telescópio de 5m de Palomar com o espectrógrafo DBSP (*Double Spectrograph*). O tempo de exposição necessário para se determinar se o objeto era ou não um QSO variou entre 300 e 600 segundos, dependendo da magnitude do objeto. Este espectrógrafo possui dois canais independentes. Um no azul, definido no intervalo entre 3350Å e 5100Å, com uma dispersão de $\sim 2\text{Å}/\text{pixel}$. No vermelho temos uma cobertura entre 5000Å e 7500Å, com uma dispersão de $\sim 3\text{Å}/\text{pixel}$. Esta definição instrumental foi feita para podermos observar a linha de Ly- α no intervalo de redshift entre 4.0 e 5.0. A calibração de comprimento de onda foi feita através das lâmpadas de He-Ne. O ajuste polinomial apresentou *rms* típico de 0.3Å. Na Figura 5.2 mostramos os espectros de alguns candidatos que foram confirmados como sendo QSOs em redshifts entre 4.0 e 5.0.

5.4 Discussão

Neste capítulo apresentamos resultados preliminares deste projeto que visa a busca de QSOs localizados em altos desvios para o vermelho ($4 < z < 4.5$). Este trabalho foi viável pelo desenvolvimento do procedimento otimizado de *matching* e do novo conceito de armazenamento dos dados do POSS-II. Além disso, a aplicação de novas ferramentas de inteligência artificial nos permitiu minimizar a contaminação do catálogo de estrelas por galáxias, fazendo com que o levantamento espectroscópico fosse realizado num curto prazo de tempo. Atualmente, este projeto já conseguiu catalogar cerca de 100 novos QSOs no intervalo $4.0 < z < 4.5$. O objetivo primeiro deste projeto, como apresentado em Kenefick et al. (1995, [36]), é estimar a densidade espacial de QSOs. Apresentamos, aqui, dados para 21 novos QSOs em altos z , e mais 3 QSOs redescobertos por este levantamento. Esta amostra é um fator dois maior do que a apresentada em Kenefick (1996, [35]). O desvio para o vermelho mediano da amostra é 4.25, compatível com Kenefick (1996,

Figura 5.2: Espectros de candidatos a QSO para $4.0 < z < 5.0$

[35]). Utilizando o método $1/V_a$ (Avni & Bahcall 1980, [3]), onde V_a é o volume comóvel dentro do qual o QSO poderia ter sido detectado por este levantamento, estimamos que a densidade espacial de QSOs ($4.5 \times 10^{-9} \text{ Mpc}^{-3} \text{ mag}^{-1}$) decresce aproximadamente de um fator sete comparado com a densidade espacial destes objetos em desvios para o vermelho igual a dois. Além disso, corroborando o resultado já encontrado por Kennefick (1996, [35]), não encontramos qualquer evidência de que a função de luminosidade para QSOs menos brilhantes ($M_B > -27$) evolua diferentemente dos QSOs brilhantes. Desta forma, não é necessário invocar uma dependência com a luminosidade na evolução dos QSOs no intervalo $2 < z < 4$. Os resultados finais incluindo todos os QSOs até então descobertos está sendo submetido para publicação em Djorgovski et al. (2000, [18]).

Capítulo 6

Conclusões e Perspectivas

Nessa tese apresentamos o banco de dados SKICCOSMO, voltado para a cosmologia observacional, mas que também pode ser usado em outros campos da astronomia. O SKICCOSMO é constituído a partir do SKICAT (Weir, 1994, [66]) e possui a propriedade de ser integralmente relacional, o que lhe garante integridade e otimização, tanto da ocupação de memória, quanto do tempo de consulta. Os dados podem ser obtidos através de consultas elaboradas em uma página “*Web*” com a ajuda do **SkiccQL**, uma interface que permite o usuário compor, com algumas instruções simples, consultas complexas do ponto de vista do banco de dados relacional.

Os dados disponíveis no SKICCOSMO são constituídos - a exemplo do SKICAT - pelo DPOSS-II (Reid, 1991, [48]). Contudo, sua concepção permite a inclusão de outros catálogos e bancos de dados da astronomia, assim como possui flexibilidade para ser integrado a projetos globais, como veremos a seguir em 6.4.2.

O POSS-II viabilizou pesquisas avançadas em diversos campos (ver 6.4.2), particularmente, foi o ponto de partida para a descoberta de quasares em *redshifts* superiores a 4 (Kennefick, 1995, [36]). Os espectros de quasares nessas condições apresentam a absorção da “*Lyman- α forest*” até $\sim 6000\text{\AA}$. Por sua vez, a linha de emissão Lyman- α desses quasares estão deslocados para maiores comprimentos de onda em função de sua distância. Em consequência o índice de cor (**g-r**) é muito maior do que o dos objetos comuns. Essa propriedade permite que se examine o diagrama (**g-r**) \times (**r-i**) e se identifique os objetos com (**g-r**) discrepantes. Tais objetos são candidatos a QSO nas condições descritas.

6.1 Bancos de Dados

A década de 90 assistiu ao nascimento de uma área que só pode ter sentido considerada a comunidade astronômica global: os bancos de dados. Desde a primeira edição das Conferências “*Astronomical Data Analysis Software and Systems*” o ADASS-I (Worrall *et al.*, 1992, [72]) percebe-se a grande incidência de bancos de dados surgindo em todo o mundo. Desde então tem-se assistido a uma grande profusão de bancos de dados servindo-se uma enorme quantidade de dados de todos os tipos, em praticamente todas as faixas de frequência e técnicas observacionais.

Diante desse quadro, observa-se o aparecimento de esforços no sentido de integrar os bancos de dados pois torna-se cada vez mais difícil manejar tantos serviços de informação, cada um com seu próprio conjunto de ferramentas e formas de consulta. Quando os responsáveis pelo desenvolvimento do banco de dados decidem por qual caminho trilhar na sua construção, muitas vezes não têm muita escolha pois as próprias técnicas observacionais ou o conjunto delas determina como construir a base de dados. Portanto, a integração deve ser feita pelos centros de desenvolvimento dos **meta** banco de dados. Essa é a alternativa para os **padrões de normalização**, que não se tornou viável.

Essa é a tarefa que se tem prestado os sistemas **AMASE**, **IRSA**, **HEASARC** vistos no Capítulo 2. A isso se propõe o ambicioso projeto do Observatório Virtual, visto em 6.4.2.

6.2 BDR, POO e BDOR

Essas são as iniciais dos tópicos mais atuais da ciência das bases de dados. O Banco de Dados Relacional (BDR) é um padrão atual e povoa praticamente todos os serviços que se prestam ao auxílio ao usuário para obter informações. Tratando-se de uma técnica baseada na teoria matemática das relações, parece ser ela o ponto de partida para tudo o que se fizer a partir de agora.

Por outro lado, a Programação Orientada ao Objeto (POO) apareceu como epílogo de uma grande evolução no conceito de programação. Da programação seqüencial, em que os programas eram construídos segundo algoritmos lineares, passou-se à programação estruturada para a qual o **PASCAL** teve grande influência pois trazia em si a evolução nos conceitos de algoritmos. Seguiu-se, finalmente, a programação orientada ao objeto que o **C++** inaugurou. Esta sistematiza, de vez, os conceitos de estruturas que passam a fazer parte do que podemos chamar **super estruturas**, contendo conjuntos de estruturas,

funções, interfaces, etc. Conceitos anteriormente identificados como **variáveis**, **funções** e **estruturas**, são estendidos para **objetos**, **métodos** e **encapsulamentos**. Através de encapsulamentos recorrentes viabiliza-se a possibilidade de construção de riquíssimas bibliotecas de métodos, permitindo assim a alto grau de simplificação e portabilidade na programação. Sem essa técnica de programação, talvez as ferramentas atuais baseadas na interação com o vídeo, tal como a série *Windows* e *X11* não teriam metade das facilidades que possuem hoje.

Por último, a união entre essas duas vertentes da ciência da informática - os bancos de dados relacionais e a programação orientada ao objeto - não parece artificial. Afinal, entre o conceituário de ambas vemos termos coincidentes, tais como **instância**, enquanto que outros podem ser unificados como entidades e encapsulamentos. De um outro ponto de vista, pode-se pensar que tabela é um caso particular de objeto, cabendo aí uma extensão para objetos mais complexos. A essa tarefa os engenheiros dos grandes fabricantes de DBMS e compiladores se dedicaram no início da década de 90 e os primeiros resultados começaram a despontar em meados daquela década. Entre os produtos que apareceram está a ferramenta de construção dos *datablades* da Informix, ex-Illustra.

Escolhemos construir o SKICCOSMO com base no DBMS da Informix Co. por conta da possibilidade de integrá-lo à conceituação de BDOR, que julgamos ser a tendência a ser seguida em futuro próximo.

6.3 Comparação e *matching* de Campos

O problema de *matching* entre poses supostamente do mesmo campo é, em termos práticos, o mesmo do reconhecimento de formas, isto é, comparar uma imagem a outra, uma seqüência sonora ou um padrão na estrutura de objetos complexos. Os algoritmos de Murtagh, 1992 ([45]), Stetson, 1989 ([55]) e Groth, 1986 ([23]) são poderosos e poderiam, com algumas adaptações, ser aplicados em qualquer problema de reconhecimento de forma. Aplicam-se, no entanto, somente a casos de transformações lineares. O método do histograma das diferenças aqui apresentado em 3.5 é mais limitado ainda, sendo aplicável somente para casos de deslocamentos em α e δ . Esse é o caso da imensa maioria dos campos a serem comparados para se proceder à construção da tabela primária **Objects**. Casos excepcionais tiveram de ser tratados diferentemente, mas, via de regra, esse método bastou para o trabalho de *matching* entre campos, o que reduziu significativamente o tempo de transferência dos dados do antigo SKICAT para o SKICCOSMO.

6.3.1 Campos Populosos e Densamente Povoados

Uma dificuldade se interpôs no processo de *matching* entre campos semelhantes: a densidade de objetos no campo. A presença de 6 objetos a cada minarco² faz aparecer, no histograma das diferenças, um *contínuo* do tipo visto na Figura 3.4 que oblitera o pico representando o *offset*. A solução para esse problema é a aplicação de um polinômio do 3^o grau a esse contínuo modelado até pelo menos a diferença de cerca da metade do campo. Essa solução mostrou-se satisfatória para todos os casos que aplicamos, com exceção de algumas casos que apresentaram, também, problemas de escala e erros de operação no momento da digitalização.

6.4 A Disponibilização dos Dados

Um programa CGI em Perl foi construído para dispor os dados do SKICCOSMO. Esse programa recebe as informações via submissão típica do CGI, isto é, através da resultante de um formulário HTML. Essa submissão, no entanto, não está restrita ao formulário disponível no SKICCOSMO dedicado à construção de uma nova consulta como visto na Figura 4.6. Qualquer centro pode construir seu próprio formulário para ser submetido ao SKICCOSMO, desde que sejam preenchidas certas condições, tais como a de repassar as variáveis com seus nomes corretos e com valores compatíveis. Essa característica permite a integração do SKICCOSMO aos *meta* bancos de dados. A conformidade do SKICCOSMO com o padrão do banco de dados relacional (BDR) permite essa integração.

6.4.1 A Recuperação dos Dados

Vimos que, para auxiliar o astrônomo na recuperação dos dados do SKICCOSMO existe um interpretador. Este, num padrão que se aproxima daquilo com que o usuário está habituado a lidar, transforma a consulta feita em uma instrução SQL, que é a linguagem nativa do BDR. Com isso evita-se obrigar o astrônomo de aprender o SQL, cujo manuseio exige algum treinamento. No entanto, essa facilidade limita o poder de ação na consulta já que a simplificação sempre aporta a limitação. Dessa feita elaborou-se o SkiccQL, nome que demos a essa linguagem simples, permitindo que se chegue ao objetivo em mais de uma consulta.

O astrônomo que faz uma consulta a um banco de dados do tipo do SKICCOSMO, muitas vezes, espera um retorno de um grande número de linhas na sua tabela. É con-

traproducente fazer retornar a tabela em forma de HTML, como resposta à submissão da consulta, ainda mais que a consulta pode tomar horas, ou mesmo dias. O melhor a fazer é registrar o pedido da consulta para execução à parte da conexão da *WEB*. Isso é feito através de um processo independente do serviço de página da *WEB*. Um programa residente no servidor ESQL do Informix chamado *officeboy* se encarrega de entregar a consulta registrada ao *motor* do banco de dados e depois da consulta ter sido realizada, avisar o usuário através de uma mensagem eletrônica e colocar o resultado em uma área acessível a esse usuário.

Esse procedimento não deve mudar diante da extensão do SKICCOSMO à sua constituição objeto - relacional, quando forem introduzidas imagens ao banco de dados para serem recuperadas segundo critérios igualmente astronômicos. As propriedades fotométricas podem ser recuperadas através de identificação e classificação executadas no momento da consulta, usando métodos e parâmetros definidos pelo próprio usuário. Do mesmo modo, a imagem propriamente dita pode ser recuperada para análise segundo os critérios do usuário.

Esse cenário se coaduna perfeitamente ao Observatório Virtual, proposto por Caltech, que é descrito a seguir.

6.4.2 O Observatório Virtual

Diante do quadro dos dados atuais advindo tanto de *surveys* quanto de observações, ou tanto de sondas e telescópios espaciais quanto de programas em terra, e olhando para os projetos ora sendo levados adiante e para propostas futuras, é fácil verificar que um problema se apresenta: o da manipulação e tratamento de massa de dados em quantidade nunca antes obtida. Vários são os centros de distribuição de dados. No Capítulo 2 alguns desses centros, inclusive meta banco de dados, foram listados. O problema se coloca, no entanto, na questão da quantidade de informação. A previsão para o futuro próximo é que a quantidade de dados, contada tanto em número de objetos como em bytes, será tão grande que não será possível ao astrônomo manter-se atualizado, mesmo nos assuntos de sua especialidade.

Com base nessas premissas está sendo proposto pelos Drs. S. G. Djorgovski e A. Szalay (2000, [16]) em cooperação com vários institutos americanos liderados por Caltech a criação do Observatório Virtual Nacional (*OVN*) e cuja argumentação vem exposta a seguir.

Por que Observatório Virtual?

O atual *modus operandi* observacional consiste no tratamento de pequenas amostras em faixas estreitas de comprimento de onda. No entanto, a acumulação de informação é crescente a uma progressão geométrica, e em futuro breve, se nada for feito, a comunidade astronômica não será capaz de lidar com toda essa informação, provavelmente caindo em projetos redundantes e conseqüente desperdício de recursos. Esse é o momento, portanto, de planejar o que e como fazer para integrar e tratar coerentemente os dados, tanto do que já foi acumulado (legado), quanto os dados que estão sendo levantados agora e serão no futuro.

Uma vez em operação, o OVN poderá suscitar no astrônomo as seguintes questões: “Será necessário observar um objeto, ou uma classe de objetos, nas condições desejadas, ou essas observações já existem”? Ou então: “Com os dados que possuo de uma classe de objetos eu posso comparar com outros dados em outras condições desses mesmos objetos e chegar a conclusões ainda mais profundas e completas?”. Num certo sentido, o Observatório Virtual poderá economizar enormemente o trabalho de pesquisa, dispensando a necessidade de observar certos objetos. A informação será confiável e instantânea se comparada com o tempo e recursos que são dispensados na preparação e deslocamento aos observatórios.

O que se pode fazer com as “observações” no Observatório Virtual?

Alguns resultados recentes de *surveys* e recuperação de dados proporcionaram descobertas que podem dar uma idéia do que se poderá obter na operação com o OVN:

- Descoberta de anãs marrons e função de massa desses tipos de objetos com o 2MASS;
- Descoberta de anãs *c/* metano com o SDSS;
- Quasares em altos *redshifts* usando o DPOSS, FIRST e SDSS;
- Estudos de variáveis e binárias usando o MACHO e OGLE.

É preciso notar que essas descobertas são de caráter não apenas primário (em que os experimentos foram concebidos para isso), como é o caso das anãs marrons e outras estrelas abaixo da seqüência principal, mas também são de caráter secundário, como é o caso das estrelas variáveis e binárias e por cruzamento de dados como são os casos dos quasares a alto *redshift* e as descobertas de raios X “mole” em binárias (ROSAT e MACHO).

Do que decorre o Observatório Virtual?

A quantidade de dados tem crescido ao longo dos anos de forma geométrica. Espera-se que esse crescimento se prolongue ainda nas próximas décadas. *Surveys* como o Sloan e 2MASS, DPOSS produzirão vários terabytes de dados. Alguns *surveys* que estão por vir produzirão dados da ordem de gigabytes por noite ou mais.

Avanços tecnológicos nas técnicas de armazenagem de massa de dados, capacidade de memória e cálculo, velocidade de execução e a nova rede internet, permitindo transferências à taxa de 100 MByte/sec apontam para o surgimento de um novo paradigma na estocagem e tratamento de dados na astronomia.

Será o Observatório Virtual viável?

Os autores, S. G. Djorgovski e A. Szalay argumentam que, dado o fato da comunidade astronômica ser relativamente pequena e já ter demonstrado ter facilidade em aceitar padrões (vide o formato FITS e os procedimentos de redução em pacotes como o IRAF e MIDAS) ela não apresentará resistência em empenhar-se para gerar uma grande rede de agentes de consultas coerentes e interligados entre si.

O OVN vai exigir que os centros de distribuição de dados se conformem a um padrão de comunicação. Ele exigirá que se integre os dados da forma mais eficiente possível.

O OVN se compara ao projeto do Genoma Humano e ao banco de dados nas ciências da terra (Djorgovski & Szalay, 2000, [16]).

Como se poderá operar com o Observatório Virtual

A geração de consultas é uma das principais preocupações dos autores do projeto do OVN. É esperado um amplo e variado público interessado no que o OVN terá a oferecer: tanto pesquisadores de alto nível como serviços educacionais e de divulgação. Para isso é preciso estabelecer um plano de desenvolvimento de ferramentas de consulta e arquitetura computacional. Entre elas, são citadas:

- módulos inteligentes de geração de estrutura de dados e antecipação e ajuste a consultas subseqüentes;
- implantação efetiva de base de dados a vários terabytes de maneira a disponibilizar intantaneamente os dados;

- desenvolvimento de *motores* de banco de dados para consultas complexas e sofisticadas;
- possibilidade de criação de novas linguagens para a integração das consultas de forma eficiente;
- desenvolvimento de “caixas de ferramentas” estatísticas para proporcionar consultas dedutivas eficazes;
- desenvolvimento de “caixas de ferramentas” de visualização para facilitar a interação do usuário com o sistema.

Pesquisas que poderão ser executadas no Observatório Virtual

Algumas pesquisas serão viáveis em nível de excelência no OVN:

- Preparação de observações em telescópios de grande porte;
- Estudos de população galáctica e extragaláctica;
- Descoberta e estudos de objetos estranhos ou raros ou mesmo únicos.

Sabemos todos que, uma vez diante de um novo paradigma, os horizontes são ampliados mas não é possível antecipar o que virá da imaginação. Estudos e descobertas nunca cogitadas poderão surgir.

6.5 O que há a fazer no SKICCOSMO

Não há pacote na informática que esteja definitivamente pronto. Muito menos esse apresentado nesta tese, principalmente tendo em vista o baixo número de *homens-hora* dedicado ao seu desenvolvimento. Mesmo que se consiga produzir um sistema “imune” a erros, há de haver aperfeiçoamentos e modernizações constantes para acompanhar o desenvolvimento da tecnologia.

Afora, portanto, a necessária manutenção do sistema que ora apresentamos, alguns projetos de aperfeiçoamento se impõem. São esses que apresentamos a seguir.

6.5.1 O Problema astrométrico no SKICCOSMO

O processo de *matching* no SKICCOSMO é descrito em 3.6 quando se apresenta a estratégia de transferências dos dados do SKICAT para o SKICCOSMO. A cada *footprint* que se registra, antes fazendo-o passar por um processo de *matching* com os dados já registrados, obtém-se um deslocamento em coordenadas equatoriais dos dados iniciais relativo ao conjunto dos dados já registrados. Examinando o conjunto dos deslocamentos na matriz de *footprints* inteira (Weir, 1994, [66]) pode-se deduzir que um polinômio astrométrico de 2° grau talvez não seja suficiente para minimizar as distorções produzidas no campo da imagem. Como consequência, os deslocamentos provocados por diferentes distorções em campos contíguos que se sobrepõem (*overlapping*) podem levar a diferenças que chegam a 2 segundos de arco. Se quisermos, por exemplo, preparar programas de observação em grandes telescópios, tais como o Gemini, diferenças dessa magnitude podem induzir a erros incorrigíveis. O devido tratamento astrométrico deve, portanto, exigir uma atenção especial no SKICCOSMO.

6.5.2 Novas Ferramentas de Consulta

Com o final do desenvolvimento do interpretador SkiccQL, testes iniciais mostraram que essa linguagem simples tem espaço para se desenvolver e oferecer mais possibilidades. Algumas possibilidades como a indexação múltipla (ver em 4.2.3.2), embora incluídas no projeto de desenvolvimento, mostraram que é necessário avançar mais na estrutura do programa para se tornarem possíveis.

Entre as facilidades que serão introduzidas estão as chamadas **funções estatísticas agregadas**. São funções que permitem fazer estatísticas de valores no banco de dados. Rigorosamente não existe impedimento para essas funções serem chamadas em uma consulta ao SKICCOSMO. Algumas delas são: *avg()*, *var()*, *count()*, *sum()*, etc, significando a média, variância, contagem de linhas e soma dos valores entre outras. Introduziremos também possibilidades de diretrizes de agrupamento, ou, matematicamente falando, a parametrização dessas funções. A primeira idéia é aproveitar o conceito de indexação já explorada em 4.2.3.1 para a seleção da banda dos atributos fotométricos. Estamos desenvolvendo uma *agulhagem* no fluxograma para desviar a indexação para a função que se impõe.

Outras facilidades devem ser oferecidas como **macros** para a listagem das coordenadas equatoriais, muito embora atualmente seja possível programar o SKICCOSMO para tal tarefa.

6.5.3 Ferramentas Gráficas

A consolidação da linguagem de consulta SkiccQL possibilitará uma interface gráfica mais poderosa para permitir ao usuário uma operação com o SKICCOSMO mais intuitiva. Operações tais como as que o HEASARC oferece, de se poder escolher através de *cliques* em imagens do campo etc (ver em 2.3.7), serão possíveis. A própria visualização do campo, via exposição da própria imagem, como da carta celeste reconstruída a partir dos parâmetros obtidos das tabelas do SKICCOSMO dos objetos identificados, será possível. Para tal, a introdução das imagens no banco de dados será necessária, o que depende de recursos para a aquisição de dispositivos de memória de massa. A evolução tecnológica tem permitido o acesso de discos rígidos, seja por meio magnético quanto ótico, à possibilidade de armazenagem cada vez maior de informação. Ao referir-se à capacidade de discos rígidos, já é comum se ouvir *Terabytes* e o termo *Petabytes* já não parece estranho. Diante de tal quadro, visionar todo o POSS-II em disco não é absurdo, sendo mesmo passível de consideração em planejamentos futuros. Essa é uma das propostas do OVN, visto em 6.4.2, para o que o SKICCOSMO se prepara para se adequar.

6.5.4 Integração de Catálogos do Usuário com o SKICCOSMO

Alguns serviços de **meta** banco de dados, tais como o MAST (ver em 2.3.9) e o IRSA (ver em 2.3.10) permitem que o usuário faça um *upload* de catálogos próprios para que seja feito um cruzamento desse catálogo com os catálogos registrados nesses centros. A proposta original do SKICCOSMO vai além disso. A idéia original do SKICCOSMO era de que bancos de dados fossem integrados para serem disponibilizados ao público.

Com o advento dos **meta** bancos de dados e da proposta do observatório virtual a orientação para integração de banco de dados mudou. A idéia dominante é que *agentes* promovam buscas em endereços pré-determinados, dados necessários para fazer os cruzamentos colocando em bom termo a consulta requisitada pelo usuário.

Esses *agentes* podem evoluir ainda mais. Para que seja possível explicar a maneira pela qual é necessário mostrar um resumo de como se processa uma consulta no interior do *motor* do banco de dados (Butzen & Forbes, 1997, [8]):

- Os dados são distribuídos em blocos chamados de **páginas**;
- As páginas são recuperadas do disco e guardadas em um *buffer* (memória tampão) na memória RAM do computador. A recuperação das páginas se faz segundo a

determinação dos índices, sejam eles internos ou definidos pelo usuário (no caso o administrador do banco de dados). Se não existirem índices ou se a consulta for segundo uma pesquisa *fluida* (usualmente chamada de "wild card") as páginas são carregadas na seqüência das tabelas (geralmente seguindo a ordem de registro dos dados);

- Uma vez estando a página na memória tampão, procede-se a geração dos produtos cartesianos entre as tabelas envolvidas na consulta;
- Os produtos cartesianos vão sendo descartados em função das determinações das junções internas (ver 1.4.2.12).

Essencialmente, essa é a seqüência de eventos que ocorre quando uma consulta é efetuada. O que se acrescentam são procedimentos de verificação de segurança e integridades. Variações na política dessa verificação e a forma de paginação dependem de cada fabricante.

Os *agentes* dos atuais **meta** banco de dados permitem acesso estático aos dados porque não vão além de oferecer a listagem dos bancos de dados que contêm o(s) objeto(s) procurado(s) e suas identificações e possibilidades de recuperação dos dados do catálogo escolhido. Assim, após a escolha dos dados pelo usuário, uma consulta é enviada ao centro distribuidor pertinente dos dados. O processo descrito acima é efetuado, portanto, no interior do *motor* do banco de dados que mantém as referidas informações.

Não é absurdo pensar, tendo em vista a melhoria na velocidade de transmissão na internet que se faz em progressão geométrica, que os *agentes* se encarreguem de fazer as paginações e junções do que seria um **banco de dados único e universal na astronomia**. As tarefas típicas dos *motores* dos bancos de dados deixariam de ser feitas localmente para serem executadas **globalmente** em tarefa distribuída ou concentrada, em função das particularidades da consulta e dos servidores em questão.

Um *agente* localizado na Califórnia poderia solicitar *páginas* advindas de um banco de dados na Austrália e fazer a junção interna com dados do SKICCOSMO no Rio de Janeiro, com a saída do resultado sendo enviada para um usuário na França. Esse esquema é compatível com o modelo de diagrama de **Rede**, descrito em 1.4.2.3. Essa seria uma das possibilidades do esquema descrito nesta tese.

Com essa possibilidade as consultas construídas com bases em critérios complexos e entrelaçados como é possível no SKICCOSMO será uma realidade a nível mundial.

De um certo ponto de vista, as condições impostas nas consultas seriam os *dispositivos observacionais*, espécies de telescópios virtuais no Observatório Virtual.

6.5.5 Integração do SKICCOSMO ao Observatório Virtual

Vimos em 6.4.2 que o Observatório Virtual (*National Virtual Observatory, OVN*) é uma proposta de integração coerente e inteligente de toda a massa de informações obtidas de *surveys* e observações na astronomia. O SKICCOSMO é a proposta de disponibilização de consultas do DPOSS à comunidade de forma integrada e coerente. O SKICCOSMO em si é um banco de dados lidando com uma enorme quantidade de informação. Suas grandes vantagens são: a integração dos dados, determinando previamente o cruzamento dos dados dos objetos, desembocando na identificação de objetos comuns a vários campos (*matching*), e a adequação aos preceitos da programação objeto, conformando-se à tecnologia dos bancos de dados objeto-relacionais.

Pelas razões expostas acima, o SKICCOSMO se integra perfeitamente, tanto em proposição, quanto em operação, ao Observatório Virtual. O SKICCOSMO apresenta-se como o único banco de dados atual que oferece a possibilidade do usuário gerar consultas complexas conforme as condições que ele escolhe. É esse aspecto que o SKICCOSMO pode trazer sua cooperação transferindo essa técnica ao Observatório Virtual. O "SkiccQL" (ver em 4.2) pode ser o embrião de uma linguagem avançada de modo a permitir ao usuário a criação de consultas na complexidade que se queira para recuperar, tanto atributos fotométricos, espectroscópicos ou astrométricos, quanto imagens na banda que se queira, ou até estudos de comportamento estatístico de classes de objetos astronômicos.

Referências Bibliográficas

- [1] Miguel A. Albrecht and Daniel Egret, editors. *Database & On-Line Data in Astronomy*. Kluwer Academic Publ, 1991.
- [2] Lowell J. Arthur. *UNIX Shell Programming*. John Wiley & Sons, Inc., 2nd edition, 1990.
- [3] Y. Avni and J. N. Bahcall. On the simultaneous analysis of several complete samples - The V/Vmax and Vc/Va variables, with applications to quasars. *The Astrophysical Journal*, 235:694-716, February 1980.
- [4] Borland Co. *Turbo C User's Guide*, 1988.
- [5] Borland Co. *Turbo Pascal User's Guide*, 1988.
- [6] Judith S. Bowman, Sandra L. Emerson, and Marcy Darnovsky. *The Practical SQL Handbook*. Addison-Wesley, third edition, 1998.
- [7] B. J. Boyle, T. Shanks, and B. A. Peterson. The evolution of optically selected QSOs. II. *Monthly Notices of the Royal Astronomical Society*, 235:935-948, December 1988.
- [8] Fred Butzen and Dorothy Forbes. *The Linux Database*. MIS:Press, 1997.
- [9] Mary Campione and Kathy Walrath. *The Java Tutorial*. Addison - Wesley Publishing Co., second edition, 1998.
- [10] Philip Chapnick. Análise das tendências da tecnologia de banco de dados. Palestra no DB-forum, São Paulo, 1997.
- [11] C. Cheung, N. Roussopoulos, G. Reichert, Leisawitz D., Kelley S., Silberberg D., and J. Wang. AMASE: An Object-Oriented Metadatabase Catalog for Accessing Multi-Mission Astrophysics Data. SISIC - Science Information Systems Interoperability Conference, 1995.

- [12] E.F. Codd. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, 6:377–387, 1970.
- [13] E.F. Codd. *The Relational Model for Database Management*. Addison-Wesley, second edition, 1990.
- [14] Service Informatique de l'Observatoire de Meudon. Le Langage C. Notes du Cours, 1987.
- [15] Terry L. Dennis. *Apple Pascal: A Problem Solving Approach*. The West Group, 1985.
- [16] George Djorgovski and Alex Szalay. A National Virtual Observatory for Data Exploration and Discovery. White Paper, 2000.
- [17] S. Djorgovski, B. M. Lasker, W. N. Weir, M. Postman, I. N. Reid, and V. G. Laidler. The Palomar Observatory-ST ScI Digital Sky Survey. I. Program Definition and Status. In *American Astronomical Society Meeting*, volume 180, pages 1307+, May 1992.
- [18] S. G. et al Djorgovski. Submetido. *The Astronomical Journal*, 2000.
- [19] D. Egret, M. Wenger, and P. Dubois. *The SIMBAD astronomical database*, pages 79–88. Kluwer Academic Publishers, 1991.
- [20] U.M. Fayyad, J.D. Roden, J.L. Loch, M.P. Burl, S. Burleigh, N. Weir, and S. Djorgovski. *SKICAT DATABASE REFERENCE*. JPL, NASA, 1993.
- [21] Lars Frank. *Database: Theory und Practise*. Addison-Wesley, 1988.
- [22] Cornell Gary and Cay S. Horstmann. *Core Java*. SunSoft Press, 2nd edition, 1997.
- [23] E.J. Groth. A Pattern-Matching Algorithm for Two-Dimensional Coordinate Lists. *AJ*, 95:1244–1248, 1986.
- [24] Server Publications Group. System administration guide. Technical report, Sybase Inc., 1994.
- [25] M. G. Haehnelt and M. J. Rees. The formation of nuclei in newly formed galaxies and the evolution of the quasar population. *Monthly Notices of the Royal Astronomical Society*, 263:168–178, July 1993.

- [26] G. Helou, B.F. Madore, M. Schmitz, M.D. Bicay, X. Wu, and J. Bennett. *The NASA/IPAC Extragalactic Database*, pages 89–106. Kluwer Academic Publishers, 1991.
- [27] P. C. Hewett, C. B. Foltz, and F. H. Chaffee. The evolution of bright, optically selected QSOs. *Astrophysical Letters*, 406:L43–L46, April 1993.
- [28] Informix. Curso Informix de SQL. Notas de Curso.
- [29] INFORMIX Press. *Getting Started with Informix Database Server*, 1996.
- [30] INFORMIX Press. *Informix Guide to SQL*, 1996.
- [31] INFORMIX Press. *OnLine Dynamic Server, Administrator's Guide, V. 7.2*, 1996.
- [32] James Jeans. *The Growth of Physical Science*. Cambridge University Press, 1948.
- [33] Roger Jennings. *Usando Microsoft Access 97*. Editora Campus, 1997.
- [34] Edward Jones. *dBaseIII Plus - User's Guide*. McGraw-Hill International Book Co., 1987.
- [35] J. D. Kennefick. *The Luminosity Function of Quasars at Redshifts Greater than Four*. PhD thesis, CALIFORNIA INSTITUTE OF TECHNOLOGY., January 1996.
- [36] J. D. Kennefick, R. R. de Carvalho, S. G. Djorgovski, M. M. Wilber, E. S. Dickson, N. Weir, U. Fayyad, and J. Roden. The Discovery of Five Quasars at $z > 4$ Using the Second Palomar Sky Survey. *The Astronomical Journal*, 110:78+, July 1995.
- [37] J. D. Kennefick, S. G. Djorgovski, and R. R. de Carvalho. The Luminosity Function of $z > 4$ Quasars from the Second Palomar Sky Survey. *The Astronomical Journal*, 110:2553+, December 1995.
- [38] Brian W. Kernighan and Dennis M. Ritchie. *The C Programming Language*. Prentice-Hall Software Series, 1978.
- [39] Donald E. Knuth. *The Art of Computer Programming - Fundamental Algorithms*, volume 1. Addison-Wesley, 1968.
- [40] B. M. Lasker, S. Djorgovski, M. Postman, V. G. Laidler, W. N. Weir, I. N. Reid, and C. Sturch. The Palomar Observatory - ST ScI Digital Sky Survey. II. The Scanning Process. In *American Astronomical Society Meeting*, volume 180, pages 0907+, May 1992.

- [41] Barry M. Lasker, Conrad R. Sturch, Carlos Lopez, Anthony D. Mallamas, Steven F. McLaughlin, Jane L. Russell, Wieslaw Z. Wisniewski, Bruce A. Gillespie, Helmut Jenkner, Elizabeth D. Siciliano, Deborah Kenny, John H. Baumert, Alan M. Goldberg, Gregory W. Henry, Edward Kemper, and Michael J Siegel. The guide star photometric catalog. *ApJS*, 68:1–90, 1988.
- [42] Ricardo D. Macedo. Adaptando SGBDS às Necessidades de Globalização das Empresas. Palestra no DB-forum, São Paulo, 1997.
- [43] Claudio Mammana. Organização dos Computadores. Notas de Curso, 1973.
- [44] Dustin McNabb. Object Orient Database Technologies and the Internet. Palestra no DB-forum, São Paulo, 1997.
- [45] F. Murtagh. A New Approach to Point-Pattern Matching. *Publications of the Astronomical Society of the Pacific*, 104:301–307, 1992.
- [46] S. Odewhan. Matching Algorithm in SKICAT. Private Communication, 1999.
- [47] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, 1988.
- [48] I. N. Reid, C. Brewer, R. J. Brucato, W. R. McKinley, A. Maury, D. Mendenhall, J. R. Mould, J. Mueller, G. Neugebauer, J. Phinney, W. L. W. Sargent, J. Schombert, and R. Thicksten. The second Palomar Sky Survey. *Publications of the Astronomical Society of the Pacific*, 103:661–674, July 1991.
- [49] I. N. Reid and S. G. Djorgovski. . In B.T. Soifer, editor, *Sky Surveys: Protostars to Protogalaxies*, volume 43, page 125. ASPCS, 1993.
- [50] Research Systems, Inc. *IDL User's Guide*, 1995.
- [51] M. Schmidt, D. P. Schneider, and J. E. Gunn. Spectroscopic ccd surveys for quasars at large redshift.iv.evolution of the luminosity function from quasars detected by their lyman-alpha emission. *The Astronomical Journal*, 110:68 , July 1995.
- [52] Maarten Schmidt. Space Distribution and Luminosity Functions of Quasi-Stellar Radio Sources. *The Astrophysical Journal*, 151:393–409, February 1968.

- [53] D. P. Schneider, M. Schmidt, and J. E. Gunn. Spectroscopic CCD surveys for quasars at large redshift. 3: The Palomar Transit GRISM Survey catalog. *The Astronomical Journal*, 107:1245–1269, April 1994.
- [54] Scott Simmons. Accessing Multimedia Data Base. Palestra no DB-forum, São Paulo, 1997.
- [55] P. Stetson. The Techniques of Least Squares and Stellar Photometry with CCDs. In B. Barbuy, E. Janot-Pacheco, A.M. Magalhães, and S.M. Viegas, editors, *V Advanced School of Astrophysics*, pages 1–151. Instituto Astronômico e Geofísico - USP, 1989.
- [56] Madnick Stuart and John J. Donovan. *Operating Systems*. McGraw-Hill International Book Co., 1981.
- [57] SUN Microsystems. *Solaris Answerbook, Fortran 77*, 1998.
- [58] SUN Microsystems, Ca. *Answerbook, The Solaris Administrator's Guide*.
- [59] The PostgreSQL Development Team. *PostgreSQL User's Guide*. Lockhart, Thomas, 1998.
- [60] T. X. Thuan and J. E. Gunn. A new four-color intermediate-band photometric system. *Publications of the Astronomical Society of the Pacific*, 88:543–547, August 1976.
- [61] UNESCO. *ISIS User's Guide*.
- [62] F. Valdes. *Focas User's Manual*. Kitt Peak National Observatory, Comp. Supp. Off., Tucson, Az, 1982.
- [63] F. Valdes, L.E. Campusano, J.D. Velásquez, and P.B. Stetson. Focas automatic catalog matching algorithms. NOAO Publications.
- [64] M.E. Van Steenberg and J.L. Green. *The NSSDC Services*, pages 151–160. Kluwer Academic Publishers, 1991.
- [65] S. J. Warren, P. C. Hewett, and P. S. Osmer. A wide-field multicolor survey for high-redshift quasars, Z greater than or equal to 2.2. 3: The luminosity function. *The Astrophysical Journal*, 421:412–433, February 1994.
- [66] N. Weir. *Automated Analysis of the Digitized Second Palomar Sky Survey: System Design, Implementation, and Initial Results*. PhD thesis, Caltech, 1994.

- [67] N. Weir, U. M. Fayyad, and S. Djorgovski. Automated Star/Galaxy Classification for Digitized POSS-II. *The Astronomical Journal*, 109:2401+, June 1995.
- [68] N. Weir, U. M. Fayyad, S. G. Djorgovski, and J. Roden. The SKICAT System for Processing and Analyzing Digital Imaging Sky Surveys. *Publications of the Astronomical Society of the Pacific*, 107:1243+, December 1995.
- [69] J.R. Weiss and J.C. Good. *The NASA Astrophysics Data System*, pages 139–150. Kluwer Academic Publishers, 1991.
- [70] James C. Wetherbe. *Systems Analysis for Computer-Based Information Systems*. West Publishing Co., 1976.
- [71] William P.D. Wightman. *The Growth of Scientific Ideas*. Oliver and Boyd, 1950.
- [72] D.M. Worrall, C. Biemesderfer, and J. Barnes, editors. *Astronomical Data Analysis Software and Systems*, 1992.

Apêndice A

PRESKI

Assistente para Carga de Dados no SKICAT

Quando ingressei nos trabalhos para a instalação e manutenção do SKICAT tomei conhecimento dos procedimentos de registro das imagens do DPOSS. O procedimento se resume em:

1. Criar diretórios de trabalho;
2. Baixar um conjunto de 23×23 arquivos da imagem original;
3. Visualizar uma imagem "comprimida" (*snapshot*) da original para inspecionar sua qualidade e verificar se é necessário promover o *byte-swap* das imagens;
4. Editar os arquivos de cabeçalho IRAF e completá-lo com os valores dos coeficientes do polinômio astrométrico e da transformação pixel-densidade. Esses valores estão em um arquivo texto à parte que geralmente acompanha o diretório contendo a imagem;
5. Chamar um programa que ajusta a sensitometria com a ajuda de uma interface gráfica em que o usuário marca as regiões dos *spots* e ajuda o algoritmo a ajustar a melhor curva de sensibilidade interativamente. Tudo isso o usuário faz com o auxílio do cursor;
6. Interagir com o programa de ajuste do céu na imagem marcando com o cursor os pontos onde se presume que exista apenas sinal do céu;
7. Executar o programa de identificação e classificação fornecendo a ele parâmetros que dependem do que foi feito até o momento.

Esses procedimentos são executados através de operações em diferentes ambientes: IRAF, interfaces gráficas como o PGPLOT e SAOimage. Ficou claro que tal tarefa exigia um razoável conhecimento dos propósitos do SKICAT além de exigir grande concentração. Dependendo da fase, um erro pode levar à necessidade de se iniciar novamente todo o procedimento. Apesar de todo o cuidado que os operadores dispensavam, não se conseguia evitar erros de manipulação em qualquer dos passos do procedimento. Além disso, era difícil treinar estagiários, tanto pelo tempo necessário para que estes entendessem o processo como pelo desinteresse que tarefa tão enfadonha suscita.

A primeira tarefa a que nos propusemos foi a de elaborar um assistente para se executar o procedimento descrito acima, no sentido de: a) diminuir a margem de erro do operador; b) tornar o trabalho mais confortável. Foi criado, desta feita, um *shell-script* que gera os arquivos e diretórios necessários, recupera os valores dos cabeçalhos, cria as imagens, gera e executa as tarefas do IRAF, dispara o servidor PGPLOT, roda os programas de calibração, executa o SAOimage, e roda os programas do SKICAT para identificar, classificar e registrar os objetos no banco de dados. Tudo isso é feito de forma automática e transparente para o usuário que é solicitado intervir apenas para tomar as decisões que dependem da decisão humana, como verificar a qualidade da imagem, marcar os *spots* de sensitometria na placa, escolher os pontos para o ajuste da curva de sensibilidade.

O **preski** foi desenvolvido no Observatório Nacional e, uma vez testado e pronto para operar, foi instalado nos computadores de Caltech para pleno funcionamento. Esse foi um módulo que se adaptou imediatamente ao ambiente de trabalho do SKICAT. Com ele, foi possível transferir as tarefas de carga dos dados no SKICAT para pessoal não especializado, por exemplo, estagiários. Graças a isso, o processo de tratamento das placas está praticamente terminado.

Abaixo vê-se a listagem do programa:

```
#!/bin/sh
#
wait_for_infile() {
FROW=$1
FCOL=$2
blkfile='printf "%s_%d_%.2d.blk" ${PLATE} ${FROW} ${FCOL}'
echo Waiting for downloading image blocks...
while [ ! -f ${BLOCKS}/${blkfile} ]
do
```

```
sleep 5
done
}

usage() {
echo "Usage: preski {-tv} \"[plate name]\""
exit 1
}

clear

# Here we are deciding if tape is tar-type or vms-type
TP_TYPE="v" # Default is vms-type
while getopts tv OPTN
do
case $OPTN in
\?) usage;;
t | v) TP_TYPE=$OPTN;;
esac
done
shift `expr $OPTIND - 1`

#
# Define variables useful to the program
#
if [ $# -ne 1 ]; then
usage;
fi

PLATE=$1
PREFIX=`echo $PLATE | cut -c1`
AUX=$HOME/AUX/$PREFIX/$PLATE
HOSTNAME=`hostname`
SKICAT=/usr/local/skicat
```

```

SKIBIN=$SKICAT/bin
AUXPROC=$HOME/aux_proc
PREVEXTRACT=$AUXPROC/prev_extract
DEV=/dev/rmt/0lbn
#
# Working dir depends on host.
#
case $HOST in
"minerva")
WORKDIR=/scr/skicat2
;;
"kalypso")
WORKDIR=/scr2/skicat2
;;
"hecate")
WORKDIR=/scr2/skicat2
;;
"phaeton")
WORKDIR=/scr4/skicat2
;;
*)
echo Can run it only on kalypso,minerva,hecate, or phaeton
exit 1
;;
esac

ST=$WORKDIR/$PLATE/st
LISTS=$WORKDIR/$PLATE/lists
BLOCKS=$WORKDIR/$PLATE/blocks
FOOTPRINTS=$WORKDIR/$PLATE/footprints
SPOTS=$WORKDIR/$PLATE/footprints/spots
CHISQ=$WORKDIR/$PLATE/chisq
#
# Go to working directory and create needed directories

```

```
#
cd $WORKDIR

# Warn user in case he has already processed the same plate...
printf "Testing for existence of \"\$PLATE\" directory..."
if [ -d $PLATE ]; then
    echo Yes.
else
    echo No. It will be created.
fi

if [ ! -d $PLATE ]; then mkdir $PLATE; fi
if [ ! -d $CHISQ ]; then mkdir $CHISQ; fi
if [ ! -d $ST ]; then mkdir $ST; fi
if [ ! -d $LISTS ]; then mkdir $LISTS; fi
if [ ! -d $BLOCKS ]; then mkdir $BLOCKS; fi
if [ ! -d $FOOTPRINTS ]; then mkdir $FOOTPRINTS; fi
if [ ! -d $SPOTS ]; then mkdir $SPOTS; fi

# Retrieve image blocks
cd $BLOCKS
if [ ! -f ${PLATE}_23_23.blk ]; then
    echo "Insert tape of \"\$PLATE\" saveset in \"\$HOST\" tape device"
    printf "Hit [ENTER] when ready... "; read dummy
    cd $BLOCKS; mt -f /dev/rmt/0 rewind
    cd $BLOCKS; $SKIBIN/restoreblocks -$TP_TYPE -f $DEV $PLATE 2>&1 >
    restoreblocks.log &
fi

#
# Erase everything in ST dir, if there is something there...
#
cd $ST
```

```
rm -f *
cp $AUX/* $ST
gunzip *.gz 2> /dev/null

#
# Define snap and spot images, but you must be sure that the
# correspondent blocks are copied onto disk already.
#
SNAPH='ls *.hhh | grep snp | cut -d. -f1'
SPOTH='ls *.hhh | grep spot | cut -d. -f1'
#
# Test for SaoImage existing processes. If they exist, must kill them
# before continuing. Or use the only existing process.
#
/bin/ps -u $USER | grep saimage | awk '{print $1}' > /tmp/saoid
nsao='wc -l /tmp/saoid | awk '{print $1}'
if test $nsao -gt 0; then
  resp=
  until [ "$resp" ]
  do
    echo There are SaoImage processes running in this session.
    printf "Kill them? "; read resp
  done
  if [ $resp = "y" ]; then
    for p in `cat /tmp/saoid`
    do
      echo $p | kill -9 $p
    done
  else
    echo Please finish your other tasks before running PRESKI.
    exit 1
  fi
fi
```



```

#
# Must run "xhost" and "setenv DISPLAY" if it is the case.
# Otherwise send a warn message and run out.
#
(saoimage 2>&1 > /dev/null &
if [ $? -ne 0 ]; then
    echo No DISPLAY environment defined, or no Xhost authorized client
    exit 1
fi )

#
# Swap images. You can "re-swap" if it is the case inside Iraf
# if you need it.
#
snapswap $SNAPH
rm -f $HOME/sw.cl
echo "cd $ST" > $HOME/sw.cl
echo "set stdimage=imt2048" >> $HOME/sw.cl
echo "display $SNAPH 1" >> $HOME/sw.cl
echo "logout" >> $HOME/sw.cl
rm -f run_sw
echo "$HOME/.init_skicat < $HOME/sw.cl" > run_sw
echo "exit" >> run_sw
chmod +x run_sw
/usr/local/bin/xterm -sb -fn 7x14 -ls -iconic -e run_sw &
if [ $? -ne 0 ]; then
    echo No DISPLAY environment defined, or no Xhost authorized client
    exit 1
fi
# run_sw
sswap="y"
while [ $sswap = "y" ]
do
    printf "Snapswap $SNAPH again? (n): "; read sswap

```

```

if [ ! "${sswap}" ]; then sswap="n"; fi
if [ $sswap = "y" ]; then
  snapswap $SNAPH
  /usr/local/bin/xterm -sb -fn 7x14 -ls -e run_sw &
  if [ $? -ne 0 ]; then
    echo No DISPLAY environment defined, or no Xhost authorized client
    exit 1
  fi
else
  echo No Snapswap $SNAPH again.
fi
done

GSHFILE='ls -l *.gsh | line'
# Save good gshfile
mv $GSHFILE SAVEGSH

wait_for_infile 4 23

if [ $SPOTH ]; then
  snapswap $SPOTH
  rm -f $HOME/ss.cl
  echo "cd $ST" > $HOME/ss.cl
  echo "set stdimage=imt2048" >> $HOME/ss.cl
  echo "display $SPOTH 1" >> $HOME/ss.cl
  echo "logout" >> $HOME/ss.cl
  rm -f run_ss
  echo "$HOME/.init_skicat < $HOME/ss.cl" > run_ss
  echo "exit" >> run_ss
  chmod +x run_ss
  /usr/local/bin/xterm -sb -fn 7x14 -ls -iconic -e run_ss &
  if [ $? -ne 0 ]; then
    echo No DISPLAY environment defined, or no Xhost authorized client
    exit 1
  fi

```

```

fi
#run_ss
  sswap="y"
  while [ $sswap = "y" ]
  do
    printf "Snapswap $SPOTH again? (n): "; read sswap
    if [ ! "${sswap}" ]; then sswap="n"; fi
    if [ $sswap = "y" ]; then
      snapswap $SPOTH
      /usr/local/bin/xterm -sb -fn 7x14 -ls -iconic -e run_ss &
      if [ $? -ne 0 ]; then
        echo No DISPLAY environment defined, or no Xhost authorized client
        exit 1
      fi
    else
      echo No Snapswap $SPOTH again.
    fi
  done
#
# There must exist only one "gsh" file
#
for f in `ls -1 *.gsh 2> /dev/null`
do
  mv $f ${f}.2
done
fi
ncoefs=`grep AMDX SAVEGSH | wc -l`
if [ $ncoefs -eq 0 ]; then
echo "Astrometric Coefficients are not present in the file $GSHFILE "
mv SAVEGSH $GSHFILE
exit 1
fi
if [ $ncoefs -eq 40 ]; then
# LSTV is the last line with AMDx zero

```

```

LSTV='cut -d/ -f1 SAVEGSH | awk -F= '$1 ~ /^AMDx1 /{if($2 != 0) print NR-1}'
# Retrieve gshfile trimmed of zero values
sed /PLTVEC1/,${LSTV}d SAVEGSH > $GSHFILE
#
# If there are only 20 coeffs, then its assumed that there is no NULL coeff.
#
elif [ $ncoeffs -eq 20 ]; then
mv SAVEGSH $GSHFILE
fi
if [ $SPOTH ]; then
#
# Create a "spot2.cl" script to be read by iraf
#
CONST='grep -i setup_true_sky *.tp_nml | awk -F= '$1 !~ /spot/{print $2}'
$SKIBIN/mkspot2cl $ST $SPOTH $CONST > spot2.cl
fi
#
# Do the same for iraf_skicat
#
/bin/ps -u $USER | grep "cl\.e" | awk '{print $1}' > /tmp/irafid
niraf='wc -l /tmp/irafid | awk '{print $1}'
if test $niraf -gt 0; then
  resp=
  until [ "$resp" ]
  do
    echo There is at least a "cl" process running.
    printf "Kill it? "; read resp
  done
  if [ $resp = "y" ]; then
    for p in `cat /tmp/irafid`
    do
      echo $p | kill -9 $p
    done
  else

```

```

    echo Please finish your other tasks before running preski.
    exit 1
fi
fi
rm -f $HOME/temp.cl
echo "cd $ST" > $HOME/temp.cl
if [ $SPOTH ]; then
    rm -f spot2.dens
    echo "set SPOT=$SPOTH" >> $HOME/temp.cl
    echo "cl < spot2.cl" >> $HOME/temp.cl
fi
#
# Here it should be interesting to introduce something that control
# the user run snapswap as many times as he wants.
#
echo "set stdimage=imt2048" >> $HOME/temp.cl
echo "imexam $SNAPH" >> $HOME/temp.cl
echo "cd $SPOTS" >> $HOME/temp.cl
echo "cl < spots.cl" >> $HOME/temp.cl

echo "logout" >> $HOME/temp.cl

rm -f run_iraf
echo "$HOME/.init_skicat<$HOME/temp.cl">>run_iraf
if [ $SPOTH ]; then
    echo "cp spot2.dens ../footprints/spots">>run_iraf
    echo "cd ../footprints/spots">>run_iraf
    echo "rm -f temp.dens">>run_iraf
    echo "sed s/INDEF/999/g spots.dens > temp.dens">>run_iraf
    echo "rm -f Constants.dummy" >> run_iraf
    echo "paste spot2.dens temp.dens | cat -n | \
        awk -f $HOME/bin/preski.awk > Constants.dummy" >> run_iraf
    echo "cat Constants.dummy" >> run_iraf
    echo "p=0" >> run_iraf

```

```

echo "c=0" >> run_iraf
echo "rm -f run.bc" >> run_iraf
echo "echo s=0.0 > run.bc" >> run_iraf
echo "while [ \"\$p\" ]" >> run_iraf
echo "do" >> run_iraf
echo "printf \"Pointer: \";read p">>run_iraf
echo "if [ \"\$p\" ]; then" >> run_iraf
echo "c=\`expr \$c + 1\`" >> run_iraf
echo "if [ \$c -eq 1 ]; then p1=\$p; fi" >> run_iraf
echo "q=\`expr \$p - 1\`" >> run_iraf
echo "sed 1,\${q}d Constants.dummy | line | awk -f \$HOME/bin/tobc.awk >> \
run.bc" >> run_iraf
echo "fi" >> run_iraf
echo "done" >> run_iraf
echo "if [ ! \$c -eq 0 ]; then" >> run_iraf
echo "echo c=\${c}.0 >> run.bc" >> run_iraf
echo "echo \"s/c\" >> run.bc" >> run_iraf
echo "v=\`(cat run.bc | bc -l )\`" >> run_iraf
echo "fi" >> run_iraf
echo "echo \$v | \$HOME/bin/match_cc">>run_iraf
echo "rm -f dummy.dens">>run_iraf
echo "head -\${p1} spot22.dens > dummy.dens">>run_iraf
echo "p1=\`expr 16 - \$p1\`">>run_iraf
echo "tail -\${p1} spots.dens >> dummy.dens">>run_iraf
echo "cp dummy.dens spots.dens">>run_iraf
fi
echo "$HOME/bin/kill.sao">>run_iraf
echo "sleep 1">>run_iraf
chmod +x run_iraf

#
# Idem ...
#
/usr/local/bin/xterm -sb -fn 7x14 -ls -e run_iraf &

```

```

if [ $? -ne 0 ]; then
  echo No DISPLAY environment defined, or no Xhost authorized client
  exit 1
fi

echo If there is a secondary spots image, mark those spot areas, then
printf "Press [ENTER] to start initplate after the snapshot is displayed";\
read dummy
cd ..
initplate ${PLATE}.pars
rm -f $SPOTS/*
spotplate ${PLATE}.pars

rm -f run_xauto
echo "#!/bin/csh" >> run_xauto
echo "source $SKICAT/skicat.init" >> run_xauto
echo "cd $WORKDIR/$PLATE" >> run_xauto
echo "xautoplate" >> run_xauto
echo "exit 0" >> run_xauto
chmod +x run_xauto

wait_for_infile 23 23

/usr/local/bin/xterm -sb -fn 7x14 -ls -n $HOST -e run_xauto &
if [ $? -ne 0 ]; then
  echo No DISPLAY environment defined, or no Xhost authorized client
  exit 1
fi

cd $WORKDIR/$PLATE
touch ${PLATE}.log
tail -1 ${PLATE}.log | cut -d: -f1 | grep Finished 2>&1 >/dev/null
FINI=$?
while [ $FINI -ne 0 ]

```

```
do
  sleep 600
tail -1 ${PLATE}.log | cut -d: -f1 | grep Finished 2>&1 >/dev/null
FINI=$?
done

mkdir catalog
cd catalog

# now do the batch procedure
rm -f batch_${PLATE}.log
date>batch_${PLATE}.log
$SKIBIN/radec_plate $PLATE 0 2>&1 >>batch_${PLATE}.log

date>>batch_${PLATE}.log
$SKIBIN/classify_plate $PLATE 0 2>&1 >>batch_${PLATE}.log
date>>batch_${PLATE}.log
$SKIBIN/register_catalog -catalog $PLATE 2>&1 >>batch_${PLATE}.log
date>>batch_${PLATE}.log
$SKIBIN/save_catalog -catalog $PLATE -dir . 2>&1 >>batch_${PLATE}.log
date>>batch_${PLATE}.log

rm -r $AUX

echo "Insert a tape in the tapedrive and save the data."
echo "Also ftp the catalog files to"
echo "kalypto /scr3/skicat2/catalogs/"

echo "                THE END                "
```

```
# Notify local users that processing is done
echo Processing of $PLATE completed on $HOST > notify1
date >> notify1
mail rrg < notify1
```



```
rm notify1
```

```
exit 0
```


Apêndice B

CAROL (*Carry On-Line*)

Utilitário de Transformação de Formato VMS para TAR

Um dos problemas do processo de transferência de dados no SKICAT era o assincronismo de algumas ferramentas disponíveis. A geração das imagens em disco era feita por um VAX da Digital porque os dados vêm do STScI em fitas exabyte no format *backup* do VAX - VMS. Os dados eram então transferidos para o disco do VAX, para em seguida serem transferidos para as estações SUN - SPARC para o tratamento com o *preski*. Esse processo era oneroso e lento.

Para resolver esse problema, criamos o utilitário **CAROL**, cujo objetivo é dispensar a necessidade do VAX. A função do **CAROL** é ler as fitas exabyte em formato *VMS-backup* e gerar outras fitas - sejam exabyte ou não - em formato *tar*. Uma vez que a fita é gerada, ela passa a aguardar a recuperação dos dados para tratamento. A transferência dos dados para o disco passa a ser uma simples tarefa de "untargar" uma fita. Dessa forma, a necessidade do VAX é eliminada, tornando a recuperação dos dados digitalizados no STScI uma tarefa simples e imediata.

CAROL é um *script shell* (*/bin/sh*), lê os dados da fita na unidade cujo nome é fornecido no momento da execução. Ele lê não somente a imagem binária, mas os cabeçalhos, e arquivos auxiliares. Em seguida ele cria os diretórios e rearranja os arquivos de maneira a que estes fiquem dispostos no padrão do *preski* (essa tarefa era feita manualmente) e, igualmente, transfere tudo para uma unidade que se declara na execução.

A tarefa do **CAROL** é essencialmente de transferência e, portanto, de *swap*. Exige grande quantidade de área de trabalho em disco. No entanto, o nome dessa área pode ser previamente definido. Por ser praticamente automatizado, o **CAROL** foi instalado em

uma SPARC II especialmente dedicada a esse fim, em Caltech. O que era uma onerosa tarefa de leitura de fita em um VAX, passou a ser quase despercebida em uma SPARC II.

Juntos, o PRESKI e o CAROL trouxeram um grande salto de qualidade no projeto do POSS-II e permitiram que o Observatório Nacional pudesse participar do projeto do POSS-II sem qualquer restrição.

```
#!/bin/sh
# Usage: carol [WorkDir]

# To install this script in your own site you should set properly
# the variables "VMSBAKUP", "EXAHOST" and "AUXDIR"
# You should have the program "imgsplit" in your
# executable path.

MINSPPDIR=1200000
MINSPTOT=`expr $MINSPPDIR \* 2`

VMSBAKUP=/usr/local/bin/vmsbackup

case $HOST in
    "aedon")
        AUXDIR=/scr2/rrg/aux
        THISDIR=/scr2/rrg/convert
        WDIR=/scr2/rrg/convert
        ;;
    "elektra")
        AUXDIR=/scr/skicat2/aux
        THISDIR=/scr/skicat2/convert
        WDIR=/scr/skicat2/convert
        ;;
    "poseidon")
        AUXDIR=/scr2/skicat2/aux
        THISDIR=/scr2/skicat2/convert
        WDIR=/scr2/skicat2/convert
        ;;

```

```
"fritz")
AUXDIR=/scr/skicat2/aux
THISDIR=/scr/skicat2/convert
WDIR=/scr/skicat2/convert
;;
"kalypso")
AUXDIR=/scr3/skicat2/aux
THISDIR=/scr2/skicat2/convert
WDIR=/scr2/skicat2/convert
;;
"hecate")
THISDIR=/scr2/skicat2/convert
AUXDIR=/scr2/skicat2/aux
WDIR=/scr2/skicat2/convert
;;
"ganesha")
THISDIR=/scr2/skicat2/convert
AUXDIR=/scr2/skicat2/aux
WDIR=/scr2/skicat2/convert
;;
"aceses")
THISDIR=/scr2/skicat2/convert
AUXDIR=/scr2/skicat2/aux
WDIR=/scr2/skicat2/convert
;;
*)

echo You may convert tapes only on kalypso,hecate,fritz,shango, \
  or aceses

exit 1
;;
esac
```

```
# The directory this script is installed
# It contains some useful files like default.hhh
CAROLHOMEDIR=/usr/local/skicat/conv_tape/csh
DEFAULT_HHH=${CAROLHOMEDIR}/default.hhh
```

```
VIMXDIM=23040
VIMYDIM=23040
BLKXDIM=1024
BLKYDIM=1024
PXSIZE=2
SPTXSIZE=2304
EXAHOST=$HOST
```

```
LS=/usr/bin/ls
RM=/usr/bin/rm
WC=/usr/bin/wc
CP=/usr/bin/cp
MV=/usr/bin/mv
```

```
availsp () {
av='df -b $1 | sed 1d'
FSYS='echo $av | awk '{print $1}''
AVSP='echo $av | awk '{print $2}''
}
```

```
bkvms () {
DEV=/dev/rmt/0n
cd $THISDIR; $VMSBAKUP -xv -s $1 -f $DEV
}
```

```
bcktape () {
mt -f $DEV offl
```

```

}

availsp $THISDIR
fsthdir=$FSYS
avthdir=$AVSP

totalsp=$avthdir

fswdir=$FSYS
avwdir=$AVSP
if [ $fsthdir != $fswdir ]; then
totalsp='expr $totalsp + $AVSP'
fi

printf "Insert tape in exabyte device of \"%s\" and press [Return]" $EXAHOST
read dummy
bkvms 1
bkvms 2
bcktape
eval $RM -f copyright.txt
eval $RM -f *shadin*
for f in *spot*.vim
do
    suffix='echo $f | cut -d. -f2'
done
for f in *.vim
do
    old_prefix='echo $f | cut -c1-5'
    prefix='echo $old_prefix | cut -c2-5 | tr pi fn'
    echo $prefix
done
# Brings a good header file to assure that snp_file be a real STSci file
SNPFILE='$LS ${old_prefix}*_snp.hhh'
$RM -f $SNPFILE

```

```

eval "awk '{if ((\ $1 == \"NAXIS1\") || (\ $1 == \"NAXIS2\"))\
printf(\"%-8s=%21s %s\n\",\$1,700,\$4); else print \$0}' \$DEFAULT_HHH\"
> \$SNPFILE
SPOTVIM='eval \$LS \${old_prefix}*_spot*.vim'; ISFILE=?
if [ \${ISFILE} -eq 0 ]; then
    RTSPOT='echo \$SPOTVIM | cut -d. -f1'
    SFSPOT='echo \$SPOTVIM | cut -d. -f2'
# Here it is assumed that spot.vim file is actually a spot.hhd file
# I tested in one of these files and it works if we bring a good
# header file
    mv \${SPOTVIM} \${RTSPOT}.hhd
fi

if [ -f \$RTSPOT.hhd ]; then
    SPTSIZE='eval \$WC -c \${RTSPOT}.hhd | awk '{print \$1}''
    SPTYSIZE='expr \$SPTSIZE \/ \$SPTXSIZE \/ \$PXSIZE'
    echo Creating IRAF file from \$RTSPOT
    eval "awk '{if (\ $1 == \"NAXIS2\")\
printf(\"%-8s=%21s %s\n\",\$1,\$SPTYSIZE,\$4); else print \$0}'\
\$DEFAULT_HHH" > \${RTSPOT}.hhh
else
    echo The file \$RTSPOT does not exist. No SPOTS IRAF file generated
fi
VIM='eval \$LS *.vim'
if [ -f \$VIM ]; then
    echo Splitting \$VIM in blocks
    imgsplitt -X \$VIMXDIM -Y \$VIMYDIM -x \$BLKXDIM -y \$BLKYDIM -b \$PXSIZE\
-p \$prefix -s ".blk" -o \$WDIR \$VIM

else
    echo The file \$VIM does not exist. No block files generated
fi
FILTER='echo \$prefix | cut -c1'
# Put good sequence in a dummy file for tar

```



```
echo Creating sequence for tar backup
cd $WDIR
eval $RM -f tarlist.txt
xcont=0
while [ $xcont -lt 23 ]
do
    xcont=`expr $xcont + 1`
    ycont=0
    while [ $ycont -lt 23 ]
    do
        ycont=`expr $ycont + 1`
        printf "%s_%d_%2.2d.blk\n" $prefix $xcont $ycont >> tarlist.txt
    done
done
done
printf "Insert blank tape for tar in exabyte device of \"%s\" and press\  
[Return]" $EXAHOST

read dummy
tar cvf $DEV `cat tarlist.txt`
echo Offloading tape...
bcktape
echo Deleting unnecessary files
eval $RM -f *.blk
cd $THISDIR
eval $RM -f *.blk
eval $RM -f *.vim
mkdir $AUXDIR/$FILTER/$prefix
mv * $AUXDIR/$FILTER/$prefix

cd $AUXDIR/$FILTER/
echo Executing: rcp -r $prefix kalypso:/scr3/skicat2/aux/$FILTER/
rcp -r $prefix kalypso:/scr3/skicat2/aux/$FILTER/
rm -r $prefix
echo Done: $prefix
```

228

APÊNDICE B. CAROL (CARRY ON-LINE)

exit 0