

EXTENSÃO DO SGBD POSTGRESQL PARA SUPORTAR RECUPERAÇÃO DE IMAGENS POR CONTEÚDO

Por

Ernani Viriato de Melo

DISSERTAÇÃO APRESENTADA À
UNIVERSIDADE FEDERAL DE UBERLÂNDIA, MINAS GERAIS,
COMO PARTE DOS REQUISITOS EXIGIDOS
PARA OBTENÇÃO DO TÍTULO DE MESTRE
EM CIÊNCIA DA COMPUTAÇÃO

JULHO DE 2006

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Computação a aceitação da dissertação intitulada “**Extensão do SGBD PostgreSQL para Suportar Recuperação de Imagens por Conteúdo**” por **Ernani Viriato de Melo** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação**.

Uberlândia, 27 de Julho de 2006

Orientadora:

Prof^a. Dr^a. Denise Guliato

Universidade Federal de Uberlândia UFU/MG

Banca Examinadora:

Prof. Dr. Mauro Biajiz

Universidade Federal de São Carlos UFSCAR/SP

Prof. Dr. Ilmério Reis da Silva

Universidade Federal de Uberlândia UFU/MG

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Data: Julho, 2006

Autor: **Ernani Viriato de Melo**
Título: **Extensão do SGBD PostgreSQL para Suportar Recuperação
de Imagens por Conteúdo**
Faculdade: **Faculdade de Computação**
Grau: **Mestrado**

Fica garantido à Universidade Federal de Uberlândia o direito de circulação e impressão de cópias deste documento para propósitos exclusivamente acadêmicos, desde que o autor seja devidamente informado.

Autor

Dedicatória

Aos meus pais, Aires e Leni

Agradecimentos

Agradeço...

A Deus, por minha vida.

Aos meus pais, Aires e Leni, pela dedicação, apoio e carinho em todos os momentos que precisei.

Às minhas irmãs, Alexandar e Adriana, pelos conselhos, incentivos e carinho durante toda minha trajetória profissional.

À Marilene, por ter me ajudado tanto nos momentos finais deste projeto. Você me deu muita tranquilidade, alegria, amor e paz na minha vida.

Aos meus chefes Eduardo e Urbano pela compreensão, paciência, tolerância e principalmente pela amizade.

Aos meus amigos da ABCZ que me ajudaram muito nos momentos em que tive que deixar a empresa para dedicar-me ao mestrado.

Aos meus amigos do LCC que, diretamente ou indiretamente, contribuíram para a realização deste trabalho.

Aos meus grandes amigos, Ricardo e Robson, que caminharam comigo no desenvolvimento dos projetos relacionados ao AMDI.

Principalmente à professora Denise Guliato, pela amizade, paciência, apoio e orientação em todos os momentos da realização deste trabalho.

*"A maior recompensa do nosso trabalho não é o que nos pagam por ele, mas aquilo
em que ele nos transforma."*

(John Ruskin)

Resumo

Este projeto descreve uma extensão para o SGBDR PostgreSQL para suportar recuperação de imagens por conteúdo. Esta extensão, denominada PostgreSQL-IE, é independente de aplicação e possui as vantagens de ser portátil e de ter o código aberto. A linguagem SQL estendida (SQL-IE), disponibilizada pelo PostgreSQL-IE, é composta por um conjunto de funções que inclui comandos para criar novas funções de extração do conteúdo das imagens, novos vetores de características com uma combinação de extratores de características previamente definidos, e novos métodos de acesso para dados multimídia. SQL-IE também inclui recursos para definir consultas combinando dados convencionais e dados visuais. O PostgreSQL-IE disponibiliza um novo tipo de dados para imagens que permite associar várias imagens em um único atributo. Este recurso permite a combinação de características visuais de diferentes imagens em um mesmo vetor de características. Para ilustrar as facilidades e os recursos da extensão do SGBDR proposta, este trabalho descreve o SISPRIM, um sistema de pesquisa que permite a recuperação de mamografias associadas ao histórico clínico e estilo de vida da paciente.

Palavras chave: extratores de características, recuperação de imagens por conteúdo, mamografias digitais e extensão para SGBDR.

Abstract

This work presents an extension to RDBMS PostgreSQL for supporting content-based image retrieval. This extension, called PostgreSQL - IE, is independent of application, and takes advantage to be open source and portable. SQL - IE command language extension is composed by a set of functions that includes commands to create new feature extractor procedures, new feature vectors as a combination of previously defined feature extractors, and new access methods for multimedia data. SQL-IE also includes resources for defining queries by combining conventional and visual data. PostgreSQL - IE makes a new image data type available that permits associating various images to one unique attribute. This resource makes the combining of visual features from different images in the same feature vector possible. To illustrate the facilities and power of the proposed RDBMS extension, this work presents SISPRIM, a research system that supports mammographic images retrieval associated to the clinical history and data related to the style of the patient.

Keywords: feature extractors, content-based image retrieval, digital mammograms and extension to RDBMS.

Sumário

1	Introdução	1
1.1	Considerações Iniciais	1
1.2	Histórico	2
1.3	Objetivos	3
1.4	Organização do Trabalho	4
2	Extensões de Sistemas de Gerenciamento de Banco de Dados Relacionais para Suportar Recuperação de Imagens por Conteúdo	5
2.1	Introdução	5
2.2	Padrão SQL/MM Part 5: Still Image	6
2.3	DB2 Image Extender	8
2.4	Informix Excalibur Image DataBlade Module	13
2.5	Informix Image Foundation DataBlade Module	16
2.6	Oracle interMedia	19
2.7	Considerações Finais do Capítulo	22
3	O PostgreSQL Image Extender (POSTGRESQL-IE)	26
3.1	Introdução	26
3.2	A Estrutura do Sistema de Catálogo Estendido	27
3.3	Tipo de Dados para Imagens	28
3.4	A Linguagem Estendida (SQL-IE)	30
3.4.1	Criação de extratores de características e definição de vetores de características	30

3.4.2	Criação de métodos de acesso	35
3.4.3	Associação entre um vetor de características e um atributo <i>PGImage</i>	37
3.4.4	Recuperação de imagens por conteúdo	40
3.4.5	Inserção, atualização e manipulação de imagens	45
3.4.6	Funções diversas	50
3.5	Métodos de acesso no PostgreSQL-IE	53
3.6	Considerações Finais do Capítulo	54
4	SISPRIM - Sistema de Pesquisa com Suporte para Recuperação de Imagens Mamográficas Baseada em Conteúdo	56
4.1	Introdução	56
4.2	Visão Geral do SISPRIM	57
4.3	Base de Dados Mamográfica	58
4.4	Interface de Configuração do SISPRIM	59
4.5	Interface de Consulta do SISPRIM	62
4.6	Considerações Finais do Capítulo	63
5	Conclusão	66

Lista de Figuras

3.1	Principais tabelas do catálogo estendido.	28
3.2	Inserção de uma tupla com atributo do tipo imagem. (a) tabela <i>paciente</i> ; (b) a tabela <i>pge_regimage</i> com as informações das imagens armazenadas no atributo <i>mama</i> ; (c) tabela interna que relaciona todos os vetores de características associados ao atributo <i>mama</i> da tabela <i>paciente</i>	47
4.1	Esquema geral do fluxo de informações no AMDI. Fonte: Guliato et al. [1].	58
4.2	SISPRIM: O sistema de pesquisa do AMDI. Fonte: Melo et al. [2].	58
4.3	Interface para a administração do SISPRIM: configurando extratores de características.	60
4.4	Interface para a administração do SISPRIM: definindo vetores de carac- terísticas.	61
4.5	Interface para a administração do SISPRIM: definindo consultas.	62
4.6	Interface de pesquisa do SISPRIM para realizar a consulta <i>Query1</i>	64
4.7	Resultado da consulta <i>Query1</i>	65
5.1	Modelagem da base de dados mamográfica do AMDI (Parte 1).	73
5.2	Modelagem da base de dados mamográfica do AMDI (Parte 2).	74
5.3	Modelagem da base de dados mamográfica do AMDI (Parte 3).	75
5.4	Modelagem da base de dados mamográfica do AMDI (Parte 4).	76
5.5	Alguns comandos SQL-IE para construção da base de dados mamográfica do AMDI.	77

Lista de Tabelas

2.1	Comparação de tipos de dados e características entre as extensões de SGB-DRs.	23
2.2	Comparação de operações de manipulação de imagens entre as extensões de SGBDRs.	24

Lista de Abreviaturas

AMDI	Atlas Indexado de Mamografias Digitais
BLOB	<i>Binary large object</i>
CBIR	<i>Content-Based Image Retrieval</i>
CBIRS	<i>Content-Based Image Retrieval System</i>
DBA	<i>Database Administrator</i>
INPI	Instituto Nacional da Propriedade Industrial
ISO	<i>International Organization for Standardization</i>
SGBD	Sistema Gerenciador de Banco de Dados
SGBDR	Sistema Gerenciador de Banco de Dados Relacional
SQL	<i>Structured Query Language</i>
SQL-IE	<i>Structured Query Language - Image Extender</i>
TOAST	<i>The Oversized-Attribute Storage Technique</i>
UDF	<i>User-Defined Function</i>
UDT	<i>User-Defined Type</i>

Capítulo 1

Introdução

1.1 Considerações Iniciais

Em algumas aplicações, como sistemas multimídia, museus e galerias de arte, gerenciamento de projetos de arquitetura e engenharia, dados criminais, informações hospitalares, e dados geográficos, as imagens armazenadas podem conter informações preciosas. A necessidade de extrair estas informações e os avanços na área de processamento digital de imagens têm incentivado o surgimento de técnicas para o armazenamento e a recuperação de imagens.

A recuperação de imagem baseada em conteúdo (CBIR - *Content-Based Image Retrieval*) vem se destacando em trabalhos científicos, onde características como forma, textura e cor são utilizadas para caracterizar as imagens, para posterior recuperação destas imagens de acordo com critérios de similaridade entre estas características.

A CBIR é indicada para tratar problemas do tipo: dada uma base de dados mamográfica, recuperar as 10 imagens mais similares, a uma imagem dada como referência, para auxiliar no processo de diagnóstico de câncer de mama; ou um catalogador de um museu que procura por objetos com formas e texturas particulares semelhantes a uma determinada imagem.

Os dados de um sistema (hospitalar, multimídia, etc), normalmente estão armazenados em base de dados relacionais. Os SGBDRs (sistemas gerenciadores de bases de dados relacionais) foram projetados para armazenar e gerenciar grande quantidade de informações

em atributos e tabelas (relações). Estes atributos armazenam tipos de dados textuais e numéricos. As imagens são armazenadas em um bloco de memória não interpretável, ou seja, BLOB (*Binary Large Objects*).

As consultas aos registros em SGBDRs convencionais são realizadas usando SQL (*Structured Query Language*) que realiza as comparações entre os atributos textuais e/ou numéricos das relações, não disponibilizando comparações entre imagens, ou seja, consultas em dados do tipo BLOB. Uma solução proposta por Leung [3] e Bordogna [4] é a utilização de atributos textuais e numéricos, que contêm descrições adicionais das imagens, para realizar consultas. Nesta solução, chamada recuperação de imagem baseada em texto, diferentes usuários podem ter diferentes interpretações para uma mesma imagem.

Para que os SGBDRs considerem comparações de atributos que contêm imagens, ou seja, permitirem recuperação de imagens baseada em conteúdo e não em texto, existe a necessidade de estender suas funcionalidades e o gerenciamento dos dados. Com isso, os SGBDRs são capazes de armazenar imagens em um novo tipo de dado e realizar consultas por similaridade.

1.2 Histórico

O termo CBIR surgiu em 1992, com Kato [5], que descreve experiências em recuperação automática de imagens pelo uso de características de cor e textura. Desde então, o termo foi usado para descrever o processo de recuperação de imagens desejadas de um banco de imagens, usando a similaridade entre suas características visuais [6], [7], [8]. Entretanto, a maioria dos trabalhos que desenvolvem CBIR descrevem técnicas específicas para determinadas aplicações [9], [10], [11], [12].

Um sistema de recuperação de imagens por conteúdo (CBIRS - *Content-Based Image Retrieval System*) disponibiliza técnicas, ferramentas e algoritmos que são utilizados em áreas como estatística, reconhecimento de padrões, processamento de sinais, e computação visual. Existem vários CBIRS descritos na literatura, tais como: o QBIC da IBM [13], onde as imagens são importadas para dentro do sistema e, as características baseadas em cor, textura e forma são obtidas. As consultas são realizadas levando em consideração

as similaridades entre estas características; o VisualSEEk [14] é um sistema híbrido que integra recuperação de imagem baseada em conteúdos visuais e em propriedades espaciais (como tamanho, localização e relações com outras regiões); e o Blobworld [15] é uma ferramenta baseada em segmentação de regiões, onde propriedades destas regiões, como cor, textura e forma, são utilizadas nas consultas.

Para suportar sistemas CBIR independente da aplicação, algumas extensões de SGBDRs foram desenvolvidas e estão disponíveis no mercado, como o DB2 Image Extender [16], o Informix com o Excalibur Image DataBlade [17] e o Oracle com o Oracle interMedia [18]. Estes sistemas representam a imagem como um tipo de dados especial e incluem estratégias para a realização de consultas combinando dados convencionais com dados visuais.

No entanto, os sistemas CBIR descritos na literatura e as extensões de SGBDRs são limitados quanto ao número de descritores de características e quanto às consultas por similaridade das imagens.

1.3 Objetivos

O objetivo deste trabalho é desenvolver uma extensão para o SGBD PostgreSQL, denominada PostgreSQL-IE, que seja capaz de suportar o desenvolvimento de sistemas de recuperação de imagens por conteúdo independente de aplicação, com baixo custo financeiro, código aberto e que supere algumas das limitações impostas pelas extensões dos SGBDRs atualmente comercializadas.

O PostgreSQL-IE disponibiliza a linguagem de manipulação e definição de dados, denominada SQL-IE, que estende as funcionalidades do SQL padrão e permite a criação de novos extratores de características e novos métodos de acesso.

O PostgreSQL-IE disponibiliza um novo tipo de dado para representar imagens. Este tipo de dado permite que várias imagens sejam armazenadas em uma mesma coluna da tabela, o que aumenta a flexibilidade da consulta por similaridade ao mesmo tempo que permite reduzir o número de tabelas do modelo da aplicação.

Para exemplificar a utilização dos recursos propostos pelo PostgreSQL-IE, foi desen-

volvido um sistema de recuperação de imagens por conteúdo aplicado a imagens mamográficas.

1.4 Organização do Trabalho

Este trabalho está organizado da seguinte forma:

- No capítulo 2 são mostradas as extensões dos SGBDRs comerciais existentes para manipular imagens e o padrão da ISO, SQL/MM StillImage Part5: Image, criado para padronizar a sintaxe e funcionalidade das extensões dos SGBDRs. As extensões aqui apresentadas são: o DB2 Image Extender, o Excalibur Image DataBlade, o Image Foundation DataBlade, e o Oracle interMedia.
- No capítulo 3 é descrita a extensão PostgreSQL-IE proposta neste trabalho.
- No capítulo 4 é apresentado o SISPRIM - Sistema de Pesquisa com Suporte para Recuperação de Imagens Mamográficas Baseada em Conteúdo, que utiliza a extensão PostgreSQL-IE proposta.
- No capítulo 5 são apresentadas as conclusões do trabalho.

Capítulo 2

Extensões de Sistemas de Gerenciamento de Banco de Dados Relacionais para Suportar Recuperação de Imagens por Conteúdo

2.1 Introdução

As extensões de SGBD foram desenvolvidas com o objetivo de facilitar o desenvolvimento de sistemas CBIR.

Este capítulo apresenta quatro produtos diferentes, que estão disponíveis em três SGBDs: o DB2 Image Extender [16], que faz parte do DB2 Audio, Image e Video Extenders e é oferecido pela IBM como uma extensão do DB2 Universal Database; o Excalibur Image DataBlade [17] e o Image Foundation DataBlade [19], que são oferecidos pela Informix da IBM; e o Oracle interMedia [18] que suporta o gerenciamento e consultas de imagens em banco de dados Oracle.

A abordagem principal das extensões é a introdução de um novo tipo de dados de imagem que encapsula o conteúdo da imagem, seu formato, sua altura e sua largura em pixels, e outros atributos que são julgados importantes. Este novo tipo de dados é disponível para colunas de uma tabela da mesma forma que os tipos tradicionais como

integer e *varchar*. Normalmente, o acesso direto aos atributos da imagem, considerando sua representação interna não é possível, nem desejável. O acesso é realizado usando funções, como *height(imagem)*, que retorna a altura da imagem.

Algumas funções de manipulação, como escala e rotação, são utilizadas em várias aplicações e disponíveis nos quatro produtos. Porém existem algumas funções que são particulares de cada produto, como a *bghcolor()* do Image Foundation DataBlade que altera a cor de fundo da imagem.

Os nomes das características que expressam o conteúdo das imagens diferem entre os produtos. Entretanto, a funcionalidade é semelhante. Em 2001, a ISO (*International Organization for Standardization*) criou um padrão SQL, o SQL/MM Part 5: Still Image [20], que define tipos de dados, funções de manipulação e características de imagens visando a unificação da linguagem SQL.

2.2 Padrão SQL/MM Part 5: Still Image

As extensões dos SGBDRs apresentam linguagens diferentes, fazendo com que a portabilidade das aplicações entre sistemas de banco de dados seja uma tarefa difícil.

Para superar esta divergência, a ISO definiu uma linguagem padronizada para o gerenciamento de imagens em banco de dados relacionais. O primeiro resultado foi o padrão SQL/MM Part 5: Still Image, cuja meta é superar todas as linguagens e funcionalidades diferentes e simplificar a portabilidade.

O padrão SQL/MM define um tipo de dados *SI_StillImage* para encapsular imagens. Alguns atributos que constituem o *SI_StillImage* estão descritos a seguir:

- *SI_content*: representa o conteúdo binário da imagem;
- *SI_contentLength*: representa o tamanho da imagem;
- *SI_format*: indica o formato da imagem;
- *SI_width*: representa a largura da imagem;
- *SI_height*: representa a altura da imagem.

Para acessar as informações das imagens, existem os métodos que constituem o tipo de dados *SI_StillImage*. Alguns estão descritos a seguir:

- *SI_setContent*: ajusta os valores dos atributos que representam as características das imagens;
- *SI_changeFormat*: altera o formato da imagem;
- *SI_Scale* e *SI_Resize*: redimensiona a imagem alterando a sua largura e altura;
- *SI_Rotate*: rotaciona a imagem de acordo com o ângulo dado;
- *SI_Thumbnail*: exibe a imagem com o seu tamanho reduzido.

O padrão SQL/MM também define uma linguagem de consulta de imagens baseada em conteúdo usando características da imagem, definidas como tipo de dados. São quatro tipos de dados representando as características básicas e uma composta (*SI_FeatureList*), que é uma lista de uma ou mais características básicas associadas a um determinado peso. As características básicas são:

- *Average color (SI_AverageColor)*: cálculo da média de cores dos pixels de uma imagem. *Average color* é útil para recuperar imagens que possuam uma cor predominante (se uma imagem tem uma cor predominante, o *Average color* será similar à cor predominante);
- *Color histogram (SI_ColorHistogram)*: descreve a frequência da ocorrência das cores de uma imagem;
- *Positional color (SI_PositionalColor)*: cálculo da média de cor dos pixels em uma determinada área da imagem. Por exemplo, somente no canto superior esquerdo possui grande quantidade de pixels amarelos, pois existe um sol na imagem. *Positional color* é útil para representar imagens que têm cor predominante em determinadas regiões conhecidas;
- *Texture (SI_Texture)*: cálculo de granularidade, contraste e direcionamento da imagem. A Granularidade indica o tamanho dos itens repetidos em uma imagem, por

exemplo, pedregulhos. O Contraste identifica as variações de brilho e a direcionalidade indica se uma direção predomina em uma imagem. *Texture* é usada para procurar imagens que têm um padrão particular.

Os valores das características descritas anteriormente, são usadas para medir a similaridade entre duas imagens definidas como *SI_StillImage*. Por exemplo, sejam as imagens *img1* e *img2*, se *FT2* é a característica de textura da imagem *img2*, então a medida de similaridade entre as imagens *img1* e *img2* é obtida pelo método *SI_Score* de *FT2* usando a *img1* como parâmetro de entrada (*FT2.SI_Score(img1)*).

O padrão SQL/MM define uma extensão para o catálogo do sistema, que é um conjunto de tabelas relacionais contendo informações sobre os formatos de imagens e características suportadas pelo banco de dados.

Apesar dos esforços de padronização, as novas versões dos SGBDRs comerciais não aderiram ao padrão SQL/MM Part 5: Still Image. Somente o Oracle, a partir de 2005, disponibiliza, além dos recursos existentes nas versões anteriores, o suporte para o padrão SQL/MM Part 5: Still Image.

2.3 DB2 Image Extender

O DB2 Image Extender adere ao padrão SQL/MM em termos de funcionalidade das características visuais e manipulação de imagens, mas não em termos da linguagem. Além das funcionalidades propostas pelo padrão, o DB2 Image Extender oferece outras operações de manipulação e armazenamento de imagens usando o sistema de arquivo do servidor e não o banco de dados. As características visuais para consultas baseadas em conteúdo são as mesmas do padrão SQL/MM, mas não são tratadas como tipos de dados separados.

O tipo de dados usado para encapsular imagens e seus atributos é o *DB2Image*. Este tipo de dado é um UDT (*user-defined type*) não estruturado, onde internamente, é armazenado um identificador de objeto, chamado *handle*, que é relacionado com os dados da imagem e seus atributos que ficam armazenados em tabelas administrativas (tabelas do catálogo que são criadas com a utilização do DB2 Image Extender). O conteúdo

da imagem pode ser armazenado em um arquivo no servidor ou em um BLOB de uma tabela administrativa. As principais informações da imagem, como altura, largura, formato e número de cores, são armazenadas em tabelas únicas do módulo Image Extender, e outras informações, como última atualização da imagem, são armazenadas em tabelas compartilhadas com os módulos Video Extender e Audio extender.

Ao criar um banco de dados, o usuário-administrador deve habilitar o uso do DB2 Image Extender com o comando SQL: *ENABLE DATABASE FOR DB2IMAGE*. Internamente, o DB2 cria a UDT *DB2Image* para objetos de imagens, cria tabelas administrativas e cria UDFs (*user-defined functions*), como por exemplo:

- *size()*: função que retorna o tamanho em bytes da imagem;
- *replace()*: atualiza o conteúdo e comentário de usuários da imagem;
- *format()*: retorna o formato da imagem (por exemplo, GIF);
- *DB2Image()*: armazena o conteúdo da imagem;
- *Filename()*: atualiza o nome do arquivo que contém a imagem;
- *NumColor()*: retorna o número de cores de uso da imagem.

O DBA também deve habilitar o uso para as tabelas e colunas que vão trabalhar com imagens. Ao habilitar a tabela com o comando *ENABLE TABLE* o DB2 cria tabelas administrativas que conterão informações dos atributos dos objetos imagem, e ao habilitar a coluna, com o comando *ENABLE COLUMN*, cria-se *triggers* que vão atualizar várias tabelas em resposta a comandos *INSERT*, *UPDATE* e *DELETE*. O exemplo a seguir mostra a criação de uma tabela e a configuração para o uso do DB2 Image Extender:

```
CREATE TABLE paciente (  
    id            INTEGER NOT NULL,  
    registro      VARCHAR(50) NOT NULL,  
    data_nasc     DATE,  
    mama         DB2IMAGE);  
  
ENABLE TABLE paciente FOR DB2IMAGE;  
  
ENABLE COLUMN paciente mama FOR DB2IMAGE;
```

Uma imagem pode ser inserida em uma coluna do tipo *DB2Image* apenas usando uma UDF, chamada *DB2Image()*. Em resposta a esta função o Image Extender lê informações no cabeçalho da imagem referentes a altura, largura e número de cores, cria um *handle* único e armazena informações em tabelas administrativas. A atualização de imagens e de informações de controle interno é feita usando as UDFs *Replace()* e *Content()*, utilizada também na recuperação de imagens. O exemplo a seguir mostra a inserção de um registro na tabela *paciente*:

```
INSERT INTO paciente (id,registro,data_nasc,mama)
VALUES (1, '5042002-7', '22/03/1958',
DB2Image(
    CURRENT SERVER,          /*nome do servidor de BD*/
    'C:/Imagens/mamografia.bmp', /*arquivo da imagem*/
    'BMP',                   /*formato da imagem*/
    /*'ASIS' : Permite o Image Extender encontrar o formato*/
    MMDB_STORAGE_TYPE_INTERNAL, /*armazenar no BD*/
    /*MMDB_STORAGE_TYPE_EXTERNAL : armazenar no arquivo*/
    'Mamografia da Paciente'    /*comentário*/
    ) )
```

O DB2 Image Extender oferece várias funções para alterar o conteúdo de uma imagem. As operações não são desenvolvidas separadamente, e sim atribuídas na forma de parâmetros quando a imagem estiver sendo inserida, atualizada ou recuperada. Os parâmetros são usados para indicar a altura e largura da imagem, o tipo de compressão, o grau de rotação limitado a 90°, 180° e 270° graus, redução do espaço de cor, e outras manipulações. O exemplo a seguir mostra a inserção de uma imagem, convertendo o formato BMP para GIF, alterando a largura para 110 pixels e altura para 150.

```
INSERT INTO paciente (id,registro,data_nasc,mama)
VALUES (1, '5042002-7', '22/03/1958',
DB2Image(
    CURRENT SERVER,          /*nome do servidor de BD*/
    'C:/Imagens/mamografia.bmp', /*imagem origem*/
```

```

'BMP',                                /*formato origem*/
'GIF',                                /*formato destino*/
'-x 110 -y 150',                      /* opções de conversão */
'C:/Imagens/mamografia.gif',          /*imagem destino*/
'Mamografia da Paciente'              /*comentário*/
) )

```

Quando uma imagem é armazenada em um formato particular, é possível também armazenar um *thumbnail*, uma versão minimizada da imagem. O formato e o tamanho do *thumbnail* são controláveis, caso o tamanho não seja especificado e o formato da imagem seja reconhecido pelo Image Extender, um *thumbnail* é criado no formato GIF de tamanho 112 x 84 pixels. Os *thumbnails* são armazenados em tabelas administrativas, com isso, a recuperação de imagens minimizadas em uma consulta tem um maior desempenho.

Além das operações de conversão de imagens, o Image Extender possibilita a recuperação de imagens através do catálogo QBIC (*Query By Image Content*), que é um conjunto de arquivos de índices que contém as características visuais das imagens. Por razões de desempenho e por ter sido desenvolvido após a primeira versão do Image Extender, os índices são armazenados no sistema de arquivos do servidor.

Quando o catálogo QBIC é criado, devem ser identificadas as características que serão necessárias para analisar, armazenar e posteriormente consultar, lembrando que as características podem ser adicionadas ou removidas do catálogo QBIC após sua criação. As características visuais são as mesmas descritas no padrão SQL/MM, porém não são tratadas como tipos de dados.

Com o comando de criação de catálogo (*CREATE QBIC CATALOG*) o Image Extender armazena no catálogo do sistema a imagem após a mesma ser armazenada na tabela, e confere periodicamente as imagens que estão esperando para serem catalogadas (características processadas e armazenadas). O intervalo de conferência é medido em segundos pela variável de ambiente *DB2CATALOGDELAY*, onde o valor *default* é 60 segundos. O comando *CREATE QBIC CATALOG* tem um parâmetro opcional "*on*" que indica a atualização automática do catálogo QBIC, ou seja, sem a necessidade de esperar a checagem periódica do Image Extender.

Para dar manutenção às características de um catálogo QBIC, é preciso seguir os seguintes passos: abrir o catálogo com o comando *OPEN QBIC CATALOG*, com isso, o catálogo fica fechado para outras sessões; adicionar ou remover características com os respectivos comandos *ADD QBIC FEATURE* e *REMOVE QBIC FEATURE*; e por fim fechar o catálogo com o comando *CLOSE QBIC FEATURE*. Um exemplo é dado a seguir:

```
CREATE QBIC CATALOG paciente mama on ;
OPEN QBIC CATALOG paciente mama;
ADD QBIC FEATURE QbDrawFeatureClass; /* Positional Color */
REMOVE QBIC FEATURE QbDrawFeatureClass;
CLOSE QBIC CATALOG;
```

Nas consultas por conteúdo, deve-se identificar a imagem de referência, as características, os pesos das características e o conjunto de imagens catalogadas que serão comparadas, através de uma *query string* ou uma *query named*. A *query string* é composta por uma string de parâmetros, e a *query named* é o nome de um objeto *query*. A comparação entre as imagens é feita pelo cálculo de *score* definido por uma das quatro UDF descritas a seguir:

- *QbScoreFromStr*: recebe o *score* de uma única imagem catalogada (usando uma *query string*);
- *QbScoreTBFromStr*: recebe os *scores* de múltiplas imagens catalogadas em uma coluna de uma tabela (usando uma *query string*);
- *QbScoreFromName*: recebe o *score* de uma única imagem catalogada (usando uma *query named*);
- *QbScoreTBFromName*: recebe os *scores* de múltiplas imagens catalogadas em uma coluna de uma tabela (usando uma *query named*).

No exemplo a seguir é feita uma consulta que retorna os campos *id* e *registro*, e o *score* da comparação das imagens catalogadas da coluna *mama* com a imagem de referência "c:/pesq.bmp" usando característica de textura. A consulta é ordenada pelas 10 imagens da coluna *mama* mais similares:

```

SELECT id, registro,
      (QbScoreFromStr(
        mama,
        'QbTextureFeatureClass file=<server,"c:/pesq.bmp">'
      ) AS ScoreMama
FROM paciente
WHERE CAST(mama AS varchar(250)) IN
      (SELECT CAST(image_id AS varchar(25))
      FROM TABLE (
        QbScoreTBFromStr(
          'QbTextureFeatureClass file=<server,"c:/pesq.bmp">',
          /* query string */
          paciente, /* tabela de consulta */
          mama,     /* campo da tabela de consulta */
          10 )      /* Nro de imagens mais similares */
        ) AS Tabela1 )
ORDER BY ScoreMama;

```

A sintaxe das UDFs de consultas por conteúdo permitem o uso de uma das 4 características básicas (*QbColorFeatureClass*: Média de cor, *QbColorHistogramFeatureClass*: Histograma de cor, *QbDrawFeatureClass*: Cor local, e *QbTextureFeatureClass*: Textura) ou a combinação com os pesos dela. O valor do *score* é um valor maior que 0, onde 0 significa total semelhança.

2.4 Informix Excalibur Image DataBlade Module

O Excalibur Image DataBlade Module tem como principal objetivo a consulta baseada em conteúdo. Embora o módulo tenha algumas operações de manipulação de imagens, as operações são limitadas a conversões de formatos, funções de escala e codificação de cores. O Excalibur aborda as funcionalidades do padrão SQL/MM. Porém, o produto também não segue exatamente a linguagem padronizada.

Na primeira versão do Excalibur Image DataBlade, os vetores de características eram

indexados pelo método de acesso baseado em redes neurais, chamado *frnet* [21]. A partir da versão 1.2 nenhum mecanismo de indexação de imagem foi desenvolvido. Isto implica que os valores de *score* devem ser obtidos percorrendo todos os registros da tabela, o que reduz o desempenho do sistema.

As imagens e seus atributos são representados no banco de dados usando o tipo estruturado *IfdImageDesc*. Este tipo de dados armazena o formato da imagem, tipo, altura, largura e o identificador da localização da imagem. As imagens podem ser armazenadas em um objeto grande (BLOB) ou no sistema de arquivo do servidor. Todas as características extraídas da imagem são armazenadas em um vetor de característica utilizando o tipo de dados de tamanho variável, chamado *IfdFeatVect*. Na primeira versão do Excalibur Image DataBlade, cada característica era armazenada em um único vetor e indexada pelo método de acesso *frnet*, criado anteriormente pelo comando SQL *CREATE INDEX*. O exemplo a seguir mostra a criação de uma tabela e as colunas que armazenaram a imagem e suas características.

```
CREATE TABLE paciente (
    id            INTEGER NOT NULL,
    registro      VARCHAR(50) NOT NULL,
    data_nasc     DATE,
    mama          IfdImageDesc,    /*Armazenar imagens*/
    mamaVetorCar  IfdFeatVect      /*Armazenar vetores de características*/ )
```

O atributo *IfdFeatVect* é atualizado pela função *GetFeatureVector()*. Esta função extrai 6 características visuais da imagem (*Color Content*, *Shape Content*, *Texture Content*, *Brightness Structure*, *Color Structure* e *Aspect Ratio*), e as armazenam em um vetor de características que está presente no próprio atributo imagem. O exemplo a seguir ilustra a inserção de registros na tabela *paciente*.

```
INSERT INTO paciente (id,registro,data_nasc,mama)
VALUES (1, '5042002-7', '22/03/1958',
        IfdImageDescFromFile('C:/Imagens/mamografia.bmp') );
```

```
UPDATE paciente SET mamaVetorCar = GetFeatureVector(mama)
WHERE id = 1;
```

O Excalibur Image DataBlade disponibiliza funções separadas para acessar os atributos e manipular as imagens. A função *CorvertImgFormat()* pega uma imagem e uma indicação de formato como parâmetros de entrada, e retorna a imagem convertida. A função *ConvertImgType()* é usada para modificar a condificação de cor. Dependendo do formato da imagem, o esquema de codificação não pode ser aplicado. As funções *ImgHeight()* e *ImgWidth()* retornam a altura e largura da imagem, respectivamente, o seu tamanho pode ser alterado informando um fator de escala usando a função *ScaleImgBy()* ou informando o tamanho desejado usando a função *ScaleImgTo()*.

A partir da versão 1.2 do Excalibur Image DataBlade, foram disponibilizadas seis funções para extração de características combinadas em um único vetor de características. São elas:

- *Color Content*: equivalente à característica *Color histogram* do padrão SQL/MM;
- *Shape Content*: descreve a medida da orientação relativa, curvatura, e contraste das linhas em uma imagem. A cor e posição não importam;
- *Texture Content*: descreve a medida, em uma escala menor, do fluxo e granularidade de uma imagem. A cor, posição e orientação destas características não importam;
- *Brightness Structure*: descreve o brilho de cada ponto da imagem;
- *Color Structure*: similar à característica *Positional Color* do padrão SQL/MM;
- *Aspect Ratio*: descreve a relação da largura à altura na imagem.

Note que nenhuma característica similar à *Average Color* (definida pelo padrão SQL/MM e disponível no DB2 Image Extender) é disponível no Excalibur Image DataBlade. A razão disto é que esta característica em particular é de pouca utilidade. Ao calcular a *Average Color* de uma imagem no espaço de cor RGB, a cor resultante poderia ser muito diferente da percepção do usuário. Por exemplo, neste espaço de cor, uma

imagem que é a metade amarela e a metade azul resulta computacionalmente em uma *Average Color* perto de cinza.

A função *Resembles()* é responsável pela comparação de dois vetores de características. Ela permite a utilização de pesos atribuídos a cada característica, podendo variar de 0 a 100, no cálculo de similaridade. No exemplo a seguir é feita uma consulta na tabela *paciente*, onde deseja-se os 10 primeiros registros em ordem de similaridade entre a imagem "c:/pesq.bmp" e as mamografias da coluna *mama*, ponderando as características utilizadas no cálculo de similaridade.

```
SELECT FIRST 10 id, registro, rank
FROM paciente
WHERE Resembles( mamaVetorCar, /* coluna a ser comparada */
    GetFeatureVector(IfdImgDescFromFile('C:/pesq.bmp')),
    /* imagem de referencia */
    0.0, /* % mínima de similaridade */
    10, /* peso da característica Color Content */
    30, /* peso da característica Shape Content */
    40, /* peso da característica Texture Content */
    10, /* peso da característica Brightness Structure */
    5, /* peso da característica Color Structure */
    5, /* peso da característica Aspect Ratio */
    Rank #REAL)
ORDER BY rank;
```

A função *Resembles()* só pode ser executada na cláusula *WHERE* do comando *SELECT*. Ela retorna um valor booleano que indica se dois vetores (*IfdFeatVect*) são similares de acordo com um grau de similaridade especificado pelo usuário. A similaridade, medida pelo *score*, varia entre 0 (0% de similaridade) e 1 (100% de similaridade).

2.5 Informix Image Foundation DataBlade Module

O módulo Informix Image Foundation DataBlade não disponibiliza nenhuma funcionalidade para recuperação de imagens por conteúdo. A especialidade deste produto são

funcionalidades de manipulação e transformação de imagens, como rotacionar imagens e converter tipos de espaço de cor.

O armazenamento de imagens pode ser feito tanto no banco de dados como em sistemas de arquivos externo. As informações sobre as imagens, como formato e espaço de cor, são disponíveis no tipo de dados *IfxImage*.

Assim como o Excalibur Image DataBlade, o Informix Image Foundation DataBlade é um módulo do SGBD Informix Dynamic Server (IDS), mas ambos os produtos não são integrados e nem possuem uma interface de compatibilidade. Um exemplo de incompatibilidade são os próprios tipos de dados de imagens. No Excalibur Image DataBlade, as imagens e seus atributos são armazenados no tipo de dados *IfdImageDesc*, já no Informix Image Foundation DataBlade são disponíveis no tipo de dados *IfxImage*.

O tipo de dados *IfxImage* encapsula os atributos da imagem *pixeltype* (tipo do pixel da imagem), *colorspace* (espaço de cor da imagem, por exemplo: RGB e LAB), *compression* (tipo de compressão), *width* (largura) e *height* (altura).

Existem três funções para inserir ou atualizar imagens ou seus atributos. A função *IfxImage()* recebe uma imagem como parâmetro de entrada e retorna uma linha do tipo *IfxImage* com seus atributos. A função *NewIfxImage()* recebe uma imagem e algumas propriedades para a imagem de destino como parâmetros de entrada e retorna uma linha do tipo *IfxImage* com seus atributos. A função *NewImage()* recebe uma imagem e alguns parâmetros para a imagem de destino do tipo BLOB, que é armazenada no sistema de arquivos ou no banco de dados.

As manipulações de uma imagem são disponibilizadas em uma única função, chamada CVT. As operações são disponíveis através de parâmetros. Alguns exemplos de operações são dados a seguir:

- *BGCOLOR*: configura a cor de fundo da imagem;
- *ROT*: rotaciona a imagem ao redor do canto superior-esquerdo da imagem;
- *HEI*: especifica a altura em pixels da imagem;
- *WID*: especifica a largura em pixels da imagem;

- *ROI*: especifica a região de interesse;
- *ORG*: translada a imagem para uma nova posição;
- *CNT*: especifica um ajuste de contraste;
- *CUT*: corta a imagem dado um retângulo.

O exemplo a seguir ilustra como criar tabelas de imagens e como inserir um registro usando a função *NewImage()*.

```
/* Criação da tabela */
CREATE TABLE paciente (
    id            INTEGER NOT NULL,
    registro      VARCHAR(50) NOT NULL,
    data_nasc     DATE,
    mama          Blob,
    mamaAtributos IfxImage );

/* Inserir uma imagem mamografia.bmp alterando o formato
   para jpeg e a largura para 300 pixels. */
INSERT INTO paciente (id, registro, data_nasc, mama )
VALUES (1, '5042002-7', '22/03/1958',
        NewImage(CVT('mamografia.bmp', 'wid=300'),
                 'blob:///?FMT=image/jpeg')::blob );

/* Inserir os atributos da imagem */
UPDATE paciente SET mamaAtributos = IfxImage('mamografia.bmp')
WHERE id = 1;
```

O uso de uma única função para várias operações permite um melhor desempenho, pois as operações são executadas de uma só vez, não havendo necessidade de converter e mover resultados de um lado para outro entre o servidor de banco de dados e o software de manipulação de imagens.

2.6 Oracle interMedia

O Oracle interMedia permite armazenar, gerenciar, e recuperar imagens, áudio, vídeo, ou outro dado heterogêneo de mídia em uma interface integrada. O Oracle interMedia estende a confiabilidade, funcionalidade e gerenciamento do banco de dados Oracle para aplicações multimídia.

O armazenamento de dados do tipo imagem pode ser realizado no banco de dados na forma de BLOBs ou em um servidor de dados. Os atributos e o conteúdo das imagens podem ser armazenados em tipos de dados definidos pelo padrão (SQL/MM) ou em dois tipos de dados *ORDImage* e *ORDImageSignature* respectivamente.

O Oracle interMedia suporta a primeira edição da ISO, o SQL/MM Part5: StillImage, contendo todos os tipos especificados pelo padrão. Além disso, o Oracle interMedia disponibiliza mais recursos que o padrão com a utilização dos tipos *ORDImage* e *ORDImageSignature*. Os recursos são: cálculo de similaridade baseado em forma; indexação do tipo de dados *ORDImageSignature*; o método *SI_Score* do tipo *ORDImageSignature* oferece uma melhor performance; armazenamento de imagens fora da base de dados; e API java-client para os tipos *ORDImage* e *ORDImageSignature*.

O tipo de dados *ORDImage* é usado para armazenar, gerenciar e manipular imagens. Este tipo encapsula informações como altura, largura, tamanho, formato de arquivo e formato de compressão.

O tipo de dados *ORDImage* possui o método construtor *init()* e outros métodos para acessar os atributos, como *getWidth()*, *getCompressionFormat()* e *getDicomMetadata()*. O Oracle interMedia disponibiliza características e manipulação de imagens no formato DICOM (*Digital Imaging and Communications in Medicine*).

O exemplo a seguir mostra a criação de uma tabela e as colunas que armazenaram a imagem e suas características.

```
CREATE TABLE paciente (  
    id                INTEGER NOT NULL,  
    registro          VARCHAR(50) NOT NULL,  
    data_nasc         DATE,  
    mama              ORDSYS.ORDIMAGE,
```

```

mama_thumbnail  ORDSYS.ORDIMAGE,
mamaVetorCar    ORDSYS.ORDIMAGESIGNATURE )

```

A alteração de algum atributo da imagem é feito pelo método *process()*. Este método permite alterar o formato de compressão, qualidade de compressão, codificação de cor e redução de cor, ajuste de contraste, conversão de formatos, tamanho da imagem e rotação da imagem. Um exemplo de como alterar o formato do arquivo e criar um *thumbnail* é dado a seguir em PL/SQL.

```

DECLARE
    obj    ORDSYS.ORDImage;
    obj_t  ORDSYS.ORDImage;
BEGIN
    SELECT mama INTO obj FROM paciente
    WHERE id = 35 FOR UPDATE;

    obj_t := obj;
    obj.process('fileFormat=GIFF');
    obj_t.process('maxScale=32 32');
    UPDATE paciente SET mama = obj, mama_thumbnail = obj_t
    WHERE product_id = 35;
END;

```

O tipo de dados *ORDImageSignature* armazena o vetor de características das imagens. Este tipo possui um método construtor *init()*, e é atualizado pelo método *generateSignature()*. Este tipo de dados pode ser indexado, o que torna as consultas baseadas em conteúdo mais rápidas. O exemplo a seguir ilustra a inserção de registros na tabela *paciente* e a criação de um índice na coluna *mamaVetorCar*.

```

INSERT INTO paciente (id,registro,data_nasc,mama,mama_thumbnail,
                      mamaVetorCar)
VALUES (1, '5042002-7', '22/03/1958',
        ORDSYS.ORDImage.init('FILE', 'FILE_DIR', 'mamografia.bmp'),
        ORDSYS.ORDImage.init(),

```

```
ORDSYS.ORDImageSignature.init() );
```

```
UPDATE paciente SET mamaVetorCar = mamaVetorCar.generateSignature(mama)
WHERE id = 1;
```

```
CREATE INDEX ix1 ON paciente(mamaVetorCar) INDEXTYPE IS
ORDSYS.ORDIMAGEINDEX;
```

O vetor de características é extraído utilizando uma técnica de segmentação de imagem em regiões baseada em cor. Cada região tem associada informação de cor, textura e forma. As características visuais da imagem disponíveis no Oracle interMedia são:

- *Color*: equivalente à característica *Color histogram* do padrão SQL/MM;
- *Shape*: descreve as formas das regiões obtidas pela técnica de segmentação. Uma forma é caracterizada por uma região de cor uniforme;
- *Texture*: descreve os níveis de textura da imagem, como granularidade ou suavidade;
- *Location*: representa as posições de acordo com forma, cor e textura. Por exemplo, a cor azul poderia ser localizada no topo da imagem e uma certa textura no fundo do canto esquerdo da imagem.

Para calcular a similaridade entre dois vetores de características, existe o método *isSimilar()* e o operador *IMGSimilar()*. Ambos recebem como parâmetros dois vetores de características a serem comparados, as características com pesos que serão utilizados no cálculo de similaridade e um valor tolerável de *score*. O retorno é 1 para verdadeiro e 0 para falso. Para retornar o *score* da similaridade (varia de 0 a 100, onde quanto menor o *score* mais similar são as imagens) na consulta é necessário o uso do operador *IMGScore()* que recupera o *score* calculado pelo operador *IMGSimilar()*. Cada *IMGScore* e *IMGSimilar()* compartilham um mesmo número de referência. No exemplo a seguir, dado em PL/SQL, é feita uma consulta na tabela *paciente*, onde desejam-se os registros que possuem o vetor de característica (coluna *mamaVetorCar*) com similaridade de até 30 *scores*, considerando cor, de um vetor de características *query_vet1*, e com similaridade

de até 20 *scores*, considerando 70% forma e 30% textura, de um vetor de características *query_vet2*.

```
DECLARE
    query_vet1    ORDSYS.ORDImageSignature;
    query_vet2    ORDSYS.ORDImageSignature;
BEGIN
    ...
    SELECT id, registro, ORDSYS.IMGScore(1) ScoreCor,
           ORDSYS.IMGScore(2) ScoreTextura,
    FROM paciente
    WHERE ORDSYS.IMGSimilar(mamaVetorCar, /* coluna a ser comparada */
                           query_vet1,    /* Vetor de referência */
                           'color="1.0"', /* características a serem
                                           consideradas na consulta */
                           30,             /* score máximo considerado */
                           1              /* número de referência para o
                                           operador IMGScore() */
                           ) = 1
    AND ORDSYS.IMGSimilar(mamaVetorCar,query_vet2,'shape="0.7"
                           texture="0.3"',20,2) = 1;
    ...
END
```

2.7 Considerações Finais do Capítulo

As extensões de SGBDRs discutidas neste capítulo possuem muitos recursos para trabalhar com imagens. Entretanto, os mesmos são muito distintos em termos de linguagem e funcionalidades. Cada produto usa um tipo de dados diferente para representar imagens em banco de dados, e um conjunto diferente de funções e métodos para administrá-las e manipulá-las. A Tabela 3.1 resume as características de cada extensão apresentada, e a Tabela 3.2 resume algumas operações de manipulação de imagens.

Extensão SGBDR	Tipo Dado (Imagem)	Tipo Dado (Vetor Car)	Forma	Textura	Cor Global	Cor Local	Média de Cor
SQL/MM	SI_Stillimage	SI_AverageColor SI_ColorHistogram SI_PositionalColor SI_Texture SI_FeatureList	☒	✓	✓	✓	✓
DB2	DB2Image	DB2Image	☒	✓	✓	✓	✓
Informix Excalibur	IfdImageDesc	IfdFeatVect	✓	✓	✓	✓	☒
Informix I. F. M.	IfxImage (atributos)	☒	☒	☒	☒	☒	☒
Oracle	ORDImage	ORDImage Signature	✓	✓	✓	✓	☒

✓: Apresenta, ☒ : Não apresenta.

Tabela 2.1: Comparação de tipos de dados e características entre as extensões de SGBDRs.

Os SGBDs DB2, Informix, e as versões anteriores a do Oracle 10 respeitam o padrão SQL da ISO para construção de tabelas, tipos de dados, operadores, entre outros, mas não aderem ao padrão da ISO SQL/MM Part 5: Still Image. O Oracle, a partir da versão 10, possui suporte ao padrão SQL/MM Part 5: Still Image, além de suas funcionalidades já desenvolvidas nas versões anteriores.

O padrão SQL/MM define um tipo de dados *StillImage* com métodos para armazenar e manipular imagens. O DB2 Image Extender define o tipo de dados *DB2Image* para armazenar imagens e encapsular suas propriedades. As operações para alterar as propriedades das imagens são feitas apenas com a função *DB2Image()*, disponibilizando assim, várias operações na imagem em uma única requisição ao servidor de banco de dados. O Excalibur Datablade armazena imagens no tipo de dados *IfdImageDesc* e cada propriedade da imagem pode ser alterada usando uma função específica, por exemplo, a *ScaleImgBy()*. O Informix Image Foundation disponibiliza o *IfxImage* para armazenar

Extensão SGBDR	Escala	Rotação	Redução do Espaço de Cor	Conversão de Formato	Cortar Imagem	Cor de Fundo
SQL/MM	✓	☒	☒	✓	☒	☒
DB2	✓	✓	✓	✓	☒	☒
Informix Excalibur	✓	☒	✓	✓	☒	☒
Informix I. F. M.	✓	✓	✓	✓	✓	✓
Oracle	✓	☒	✓	✓	✓	☒

✓: Apresenta, ☒ : Não apresenta.

Tabela 2.2: Comparação de operações de manipulação de imagens entre as extensões de SGBDRs.

as propriedades das imagens que são alteradas com a função *NewIfxImage()*. Os dois principais diferenciais do Informix Image Foundation é a alta capacidade em alterar as propriedades das imagens com vários operadores, e não suportar recuperação de imagens por conteúdo. O Oracle interMedia permite seguir o padrão SQL/MM ou usar o tipo de dados *ORDImage* e o método *process()* para realizar operações de manipulação.

Para consultar as propriedades das imagens, o padrão SQL/MM e o Oracle interMedia define métodos referentes aos tipos de dados que armazenam imagens. O DB2 Image Extender e o Excalibur Datablade disponibilizam funções para cada propriedade. Somente o Informix Image Foundation oferece as propriedades das imagens apenas com uma função, chamada *GetImageProperties()*.

As três extensões que suportam recuperação de imagens por conteúdo armazenam o vetor de características de forma diferente. O DB2 Image Extender armazena os vetores de características (somente as características especificadas pelo administrador do banco de dados) em um sistema de arquivos, chamado catálogo QBIC. O Excalibur Datablade armazena em um tipo de dados *IfdFeatVect* e atualiza todas as características usando a função *GetFeatureVector()*. O Oracle interMedia armazena em um tipo de dados

ORDImageSignature e atualiza todas as características usando o método *generateSignature()*.

O Excalibur DataBlade e Oracle interMedia vão além das características padronizadas e implementam a característica baseada em forma. As consultas baseadas em conteúdo são funcionalmente semelhantes, entretanto a sintaxe e a interpretação de pesos e score diferem nos três produtos.

As extensões propostas pela IBM e Oracle permitem alterar o conteúdo das imagens, solucionando problemas que antes eram tratados apenas em nível de aplicação. A análise do conteúdo das imagens sendo feita no próprio SGBDs torna possível uma integração entre imagens e dados convencionais muito maior que tratada ao nível de aplicação. Entretanto, estas extensões comerciais são limitadas quanto ao número de descritores de características que podem ser utilizados nas consultas e a falta de flexibilidade para combiná-los. Outra limitação é o fato do vetor de características ser composto pelo conteúdo de uma única imagem.

Para superar algumas das limitações das extensões de SGBDs apresentadas, foi desenvolvido o PostgreSQL-IE cujos recursos serão detalhados no próximo capítulo.

Capítulo 3

O PostgreSQL Image Extender (POSTGRESQL-IE)

3.1 Introdução

O PostgreSQL é um Sistema de Gerenciamento de Banco de Dados Objeto-Relacional que foi desenvolvido a partir de um projeto chamado Postgres Versão 4.2, iniciado por volta de 1986, no Departamento de Ciência da Computação da Universidade da Califórnia (Berkeley) [22], [23]. Atualmente, o PostgreSQL é gratuito, possui o código aberto e tem um grande potencial em estender suas funcionalidades, por exemplo, adicionar novos tipos de dados, funções, operações, e métodos de indexação.

A necessidade de se ter um SGBDR gratuito que suporte recuperação de imagens por conteúdo e a extensibilidade do PostgreSQL contribuíram para o desenvolvimento do PostgreSQL Image Extender (PostgreSQL-IE) [1] em versões para os sistemas operacionais Windows e Linux.

O PostgreSQL-IE é uma extensão do PostgreSQL, desenvolvida em linguagem C e PLPGSQL [24], que permite a extração automática de características da imagem, e possui recursos para criar novos extratores de características, novos vetores de características, novos métodos de acesso e novas estratégias para a elaboração de consultas que permitem combinar dados visuais com dados convencionais, usando a extensão dos comandos SQL (*Structured Query Language*), denominada SQL-IE.

Como será mostrado nas seções seguintes, o PostgreSQL-IE não apresenta uma estrutura compatível com os padrões da ISO SQL/MM Part 5: Still Image. Esta decisão foi tomada em função das restrições impostas ao tipo de dados *SI_StillImage* e à necessidade de incorporar mais flexibilidade às linguagens de definição e manipulação de dados.

3.2 A Estrutura do Sistema de Catálogo Estendido

O PostgreSQL armazena os metadados, tais como informações sobre tabelas, colunas, *triggers*, índices, restrições de integridade, restrições de segurança e assim por diante, no catálogo do sistema. O catálogo é composto por tabelas comuns, que são mantidas por comandos SQL. O catálogo do sistema possui tabelas globais, ou seja, aquelas que pertencem a todos os bancos de dados, e também tabelas locais que são próprias do banco de dados da aplicação. Na criação de um banco de dados, as tabelas locais do catálogo são geradas pela duplicação de um banco de dados modelo, normalmente o *template1*.

Como o PostgreSQL-IE estende as funcionalidades do PostgreSQL, as informações relacionadas às imagens devem ser armazenadas em novas tabelas do catálogo gerando um novo sistema de catálogo, que está pronto para manipulação de imagens e é obtido usando o banco de dados modelo do PostgreSQL-IE, chamado *extended_template1*.

O sistema de catálogo estendido é composto pelas tabelas do catálogo clonadas do *template1* e 6 novas tabelas, descritas na Figura 3.1. Além do sistema de catálogo estendido, o banco de dados modelo *extended_template1* possui algumas funções de manipulação de imagens que utilizam bibliotecas compartilhadas do PostgreSQL e bibliotecas próprias do PostgreSQL-IE, onde estão desenvolvidas algumas funções de controle interno, extratores de características e métodos de acesso. Estas funções são responsáveis pela atualização das tabelas descritas na Figura 3.1.

Para criar um banco de dados de uma nova aplicação usando o PostgreSQL-IE, basta executar o comando SQL *create database nome_novo_banco with template = extended_template1*. Note que ao executar este comando, o banco de dados *extended_template1* é clonado e pode ser visualizado no esquema *public* do novo banco de dados sendo criado.

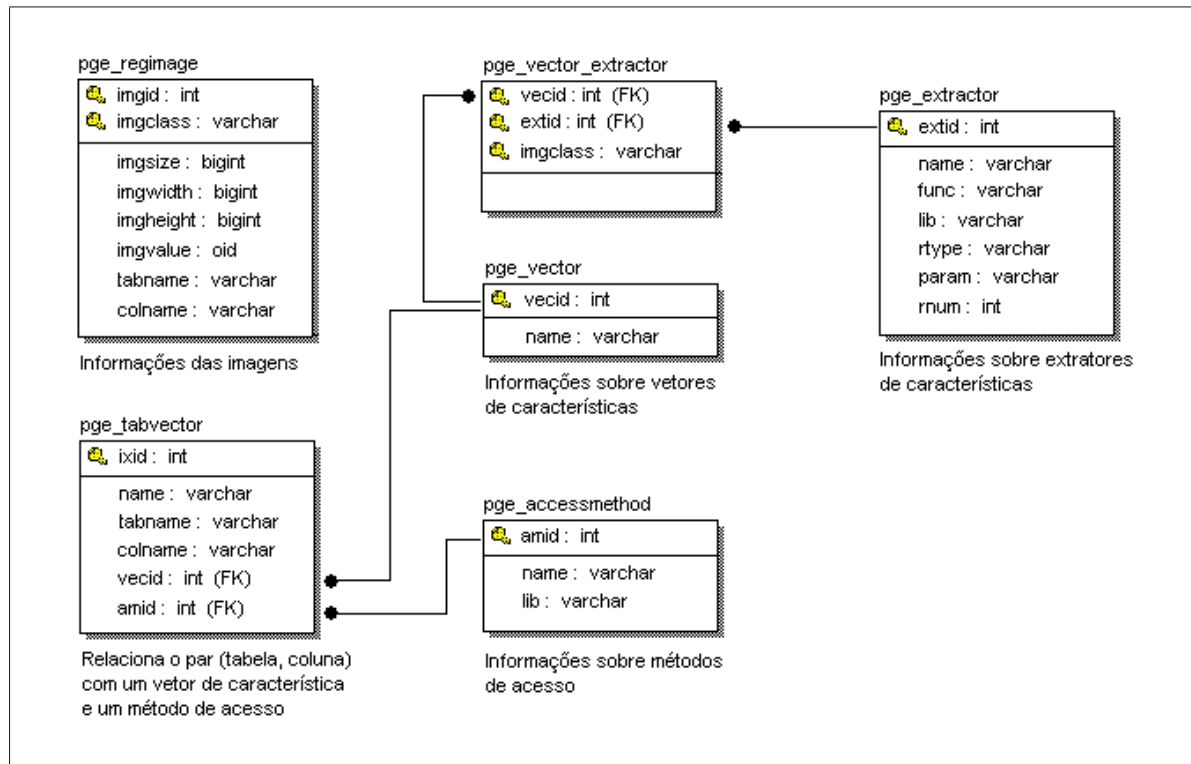


Figura 3.1: Principais tabelas do catálogo estendido.

3.3 Tipo de Dados para Imagens

Em um banco de dados relacional as imagens são armazenadas em tipos de dados BLOBs (Binary Large Objects) permitindo a recuperação das mesmas apenas através de chaves (texto ou números). Para superar esta limitação, o PostgreSQL-IE suporta um novo tipo de dados de imagens chamado *PGImage*, que é um tipo de dados composto pelo campo *imgid* do tipo integer, que identifica unicamente uma tupla, e pelo campo *imgclass* do tipo varchar, que define as classes das imagens de uma tupla. O *PGImage* é um UDT, ou seja, criado pela extensão utilizando o comando *CREATE TYPE* do PostgreSQL.

Quando uma tabela previamente criada processa os atributos do tipo *PGImage*, um valor *imgid* é associado a cada um deles. O *imgid* é um número único em todo o banco de dados que identifica uma imagem em duas tabelas. A primeira é a tabela do catálogo estendido *pge_regimage*, onde são armazenadas informações de altura, largura, tipo da imagem, o identificador de objeto grande (do inglês *large-object*) e a classe da imagem. A segunda tabela é criada dinamicamente na configuração do atributo e é usada para

armazenar características visuais da imagem, como será descrito na seção 3.4.3. A tabela do catálogo *pge_regimage* armazena o identificador de *large-object* para identificar o conteúdo binário da imagem armazenado em uma tabela TOAST (*The Oversized-Attribute Storage Technique*) do sistema. Um *large-object* é um método de armazenamento (não um tipo de dados) que habilita o PostgreSQL-IE a armazenar dados binários grandes separadamente da tabela *pge_regimage*.

Um atributo do tipo *PGImage* pode suportar uma ou mais imagens desde que elas sejam de classes diferentes. A atribuição de uma classe é necessária para que seja possível distinguir várias imagens com o mesmo formato, armazenadas em um atributo *PGImage*. Sendo assim, um vetor de características construído para um atributo do tipo *PGImage*, pode combinar descritores de características de diferentes classes de imagens armazenadas em um mesmo atributo. As classes da imagem são definidas pelo usuário durante a criação da tabela, usando uma cláusula chamada CHECK.

Como exemplo, considere o comando *CREATE TABLE*, apresentado a seguir, que cria uma tabela *paciente* com quatro atributos: identificador da paciente (*id*), registro (*registro*), data de aniversário (*data_nasc*), e imagens mamográficas (*mama*) associadas. Note que o atributo *mama* representa duas imagens com diferentes classes, uma relacionada à região de interesse da mamografia original (*mamografia*) e outra relacionada ao contorno da lesão desenhado pelo radiologista na mamografia original (*contorno*). A definição do tipo *PGImage* permite extrair características distintas das duas imagens associadas ao atributo *mama* (textura da classe *mamografia* e características de forma da classe *contorno*) e armazená-las em um único vetor de características que será utilizado para realizar consultas, facilitando o processo de modelagem de uma base de dados e incorporando flexibilidade às consultas por similaridade.

```
CREATE TABLE paciente(  
    id            INTEGER NOT NULL,  
    registro      VARCHAR(50) NOT NULL,  
    data_nasc     DATE,  
    mama         PGIMAGE,  
    CHECK ( (mama).classes in ('mamografia','contorno') )
```

```

/* O atributo mama pode armazenar imagens da classe
   'mamografia' e/ou da classe 'contorno' */
)

```

3.4 A Linguagem Estendida (SQL-IE)

O PostgreSQL-IE disponibiliza um pacote de funções, chamado SQL-IE, que incorpora ao padrão SQL novas funções para manipular e atualizar atributos de imagens, e definir consultas complexas. Além disso, apresenta facilidades para criar novos extratores de características, novos métodos de acesso, e para definir novos vetores de características. O pacote de funções foi desenvolvido usando o PLPGSQL [24] e linguagem de programação C. As próximas seções apresentam as funções disponíveis no SQL-IE.

3.4.1 Criação de extratores de características e definição de vetores de características

O PostgreSQL-IE permite incluir novas funções de extração de características em suas bibliotecas. Estas funções devem ser desenvolvidas em linguagem C e disponibilizadas em forma de bibliotecas: *dll* (*Dynamically Linked Library*) no Windows e *so* (*Shared Object*) no Linux. O PostgreSQL-IE usa este recurso para tornar novos extratores de características disponíveis para qualquer aplicação CBIR.

Cada função de extração de característica possui uma ou mais imagens de entrada e retorna um ou mais valores. O usuário-administrador da aplicação pode criar extratores de características em seu banco de dados informando qual função e biblioteca vai utilizar, desde que as funções de extração de características tenham os parâmetros de entrada e saída exigidos pelo PostgreSQL-IE.

Criação de extratores de características

Os comandos SQL-IE para manipular e criar extratores de características estão descritos a seguir.

1. Create_Extractor

Esta função cria um novo extrator de características. Ao criá-lo, o PostgreSQL-IE insere informações na tabela *pge_extractor* do catálogo estendido e cria funções de controle interno que se comunicam com as funções das bibliotecas de extratores. Note que nenhum processamento nas imagens é realizado.

Sintaxe

Create_Extractor (<NomeExtractor>, < Função>, <TipoRetorno>, <QtdeRetorno>, <Biblioteca> [, <ParEntrada>])

<i>NomeExtractor</i>	Nome do extrator de característica.
<i>Função</i>	Função que extrai a característica.
<i>TipoRetorno</i>	O tipo de dado que o extrator retorna.
<i>QtdeRetorno</i>	Quantidade de informações retornadas. <i>Default</i> = 1
<i>Biblioteca</i>	Biblioteca onde a função de extração está implementada.
<i>ParEntrada</i>	Lista de parâmetros de entrada requeridos pela função de extração. (<i>Opcional</i>)

Exemplos

```
/* Criar um extrator chamado 'area' que se comunica com a função  
   getArea da biblioteca ExtShape e retorna um valor do tipo float */
```

```
SELECT Create_Extractor('area','getArea','float','ExtShape');
```

```
/* Criar um extrator chamado 'histograma' que comunica com a  
   função getHistogram da biblioteca ExtColor e retorna 256  
   valores do tipo integer */
```

```
SELECT Create_Extractor('histograma', 'getHistogram', 'integer',  
                        256, 'ExtColor');
```

2. Delete_Extractor

Esta função exclui um extrator de características, desde que este extrator não esteja sendo utilizado por algum vetor de características. Ao remover o extrator, algumas funções de controle interno, anteriormente criadas pelo *Create_Extractor*, e algumas informações da tabela *pge_extractor* do catálogo estendido são excluídas.

Sintaxe

Delete_Extractor (<NomeExtractor>)

NomeExtractor Nome do extrator de característica no banco de dados.

Exemplo

```
/* Excluir o extrator de características 'area' */
```

```
SELECT Delete_Extractor('area')
```

Definição de vetores de características

Para a realização de consultas por similaridade, os extratores de características devem ser combinados em vetores de características.

Os comandos SQL-IE para definir vetores de características estão descritos a seguir.

1. Define_Feature_Vector

Esta função define um vetor de características combinando extratores previamente criados. A cada extrator é associada uma classe da imagem, indicando de quais das imagens armazenadas no atributo *PGImage* serão extraídas as características a serem utilizadas nas consultas.

Sintaxe

Define_Feature_Vector (<NomeVetorCaracteristicas>, <ListaExtratores>)

<i>NomeVetorCaracteristicas</i>	Nome do vetor de características sendo definido no banco de dados.
<i>ListaExtratores</i>	Lista de extratores, que compõem a definição do vetor de características, com as classes de imagens que serão utilizadas pelos extratores.

Onde: $\langle ListaExtratores \rangle :: NomeExtrator: ClasseImagem /$
 $NomeExtrator: ClasseImagem; \langle ListaExtratores \rangle$

Exemplos

Para os exemplos a seguir, considere que os extratores de características: área, perímetro, compacidade e Fourier foram previamente criados, e as classes: mamografia e contorno configuradas no atributo *mama* da tabela *paciente*, na seção 3.3.

```
/* Definir que o vetor 'Vetor_Textura' armazenará característica
   retiradas do extrator 'contraste' que utiliza a classe mamografia */
```

```
SELECT Define_Feature_Vector('Vetor_Textura','contraste:mamografia');
```

```
/* Definir que o vetor 'Vetor_Forma' armazenará característica
   retiradas dos extratores 'area', 'perimetro', 'compacidade' e
   'fourier', que utiliza a classe contorno */
```

```
SELECT Define_Feature_Vector('Vetor_Forma','area:contorno;
   perimetro:contorno;compacidade:contorno;fourier:contorno');
```

```
/* Definir que o vetor 'Vetor_Misto' armazenará características
   de textura e forma */
```

```
SELECT Define_Feature_Vector('Vetor_Misto', 'contraste:mamografia;
   area:contorno; perimetro:contorno');
```

2. Delete_Feature_Vector

Esta função exclui a definição de um vetor de características previamente definido. A exclusão resulta na atualização das tabelas do catálogo *pge_vector* e *pge_vector_extractor*.

Sintaxe

Delete_Feature_Vector (<NomeVetorCaracterísticas>)

NomeVetorCaracterísticas Nome do vetor de características a ser excluído do banco de dados.

Exemplo

```
/* Excluir a definição do vetor 'Vetor_Textura' */
```

```
SELECT Delete_Feature_Vector('Vetor_Textura')
```

3. Show_Extractors

Esta função exibe todos os extratores de características definidos no banco de dados. Também é possível exibir somente os extratores de determinado vetor de características, passando seu nome como parâmetro.

Sintaxe

Show_Extractors ([<NomeVetorCaracterísticas>])

NomeVetorCaracterísticas Nome do vetor de características, o qual se deseja listar os seus extratores.

Exemplos

```
/* Exibir todos os extratores de características do banco de dados */
```

```
SELECT Show_Extractors() AS TodosExtratores;
```

```

/* Exibir todos os extratores de características do vetor
   'Vetor_Forma' */

SELECT Show_Extractors('Vetor_Forma') AS ExtratoresVetorForma;

```

3.4.2 Criação de métodos de acesso

O PostgreSQL disponibiliza vários tipos de índice: *B-tree* [25], *R-tree* [26], *Hash* [27] e *Gist* [28]. Os índices *B-tree* suportam operadores de igualdade e de intervalo, como: $<$; $<=$; $=$; $>=$; $>$. Os índices *R-tree* são adequados para consultas a dados espaciais (*point*, *box*, *lseg*, *line*, *path*, *polygon* e *circle*) e suportam operadores geométricos, como: $\&\&$ (Se sobrepõem?); $\ll$ (Está à esquerda?); $\gg$ (Está à direita?); $\@$ (Está contido ou sobre?); e outros. Os índices *hash* suportam apenas comparações de igualdade simples. O *Gist* (*Generalized Search Trees*) é constituído de um índice (árvore balanceada) que fornece as funcionalidades tanto de uma *B-tree* quanto de uma *R-tree* e suas variantes.

Os índices disponíveis no PostgreSQL podem ser usados para tipos de dados e funções definidos pelo usuário, bastando para isso registrar no catálogo do sistema as informações de operações necessárias para cada índice. Entretanto, os índices *B-tree*, *R-tree*, *Hash* e *Gist* não são apropriados para indexar dados multimídia, uma vez que estes são representados por vetores multidimensionais e a similaridade entre diferentes objetos é obtida pela avaliação de função de distância entre os vetores multidimensionais [29], [30].

O PostgreSQL-IE ainda não disponibiliza um método de acesso multidimensional. Quando um operador de similaridade é usado em determinado atributo *PGImage*, todas as tuplas daquele atributo são lidas e realizadas o cálculo de similaridade. Esta ação é referenciada neste trabalho como *Standard Access Method* (SAM).

Entretando, o PostgreSQL-IE incorpora a flexibilidade de criar novos métodos de indexação para dados multimídia. Estes métodos de indexação devem ser desenvolvidos em linguagem C, estar disponíveis no formato de biblioteca (*dll* ou *so*) e possuir os seguintes operadores: *Create_Index_Structure()*, *Insert_Element_Index()*, *Delete_Element_Index()*,

Update_Element_Index(), *Drop_Index_Structure()*, *KNN()*, e *RANGE()*.

Os comandos SQL-IE para manipular novos métodos de acesso para indexar dados multimídia no PostgreSQL-IE estão descritos a seguir.

1. **Create_AccessMethod**

Esta função cria um método de acesso para indexar vetores de características que representam as imagens. Ao criá-lo, o PostgreSQL-IE insere informações na tabela do catálogo *pge_accessmethod* e cria funções de controle interno que se comunicam com as funções da biblioteca do novo método.

Sintaxe

Create_AccessMethod (<NomeMAcesso>, <Biblioteca>)

NomeMAcesso Nome do novo método de acesso no banco de dados.

Biblioteca Nome da biblioteca que contém o método de acesso.

Exemplo

```
/* Criar o método de acesso 'SlimTree' que está disponível na
   biblioteca 'ExtSlimTree' */

SELECT Create_AccessMethod('SlimTree','ExtSlimTree')
```

2. **Delete_AccessMethod**

Esta função exclui um método de acesso previamente criado, desde que nenhum campo de uma tabela esteja utilizando-o. Ao remover o método de acesso, algumas informações da tabela do catálogo *pge_accessmethod* e funções de controle interno, anteriormente criadas, são excluídas.

Sintaxe

Delete_AccessMethod(<NomeMAcesso>)

NomeMAcesso Nome do método de acesso a ser excluído.

Exemplo

```
/* Excluir o método de acesso 'SlimTree' */
```

```
SELECT Delete_AcessMethod('SlimTree')
```

3.4.3 Associação entre um vetor de características e um atributo

PGImage

Até agora foram definidos extratores de características, vetores de características, e métodos de indexação. Para realizar uma consulta baseada em conteúdo, um atributo do tipo *PGImage* deve ser associado a um ou mais vetores de características previamente definidos, e opcionalmente a um índice.

O atributo do tipo *PGImage* armazena as características visuais, relacionadas aos vetores de características associados ao próprio atributo, em uma tabela de dados visuais do catálogo criada dinamicamente, utilizando a função *Set_Feature_Vector* do SQL-IE. O número de colunas desta tabela varia de acordo com o número de extratores de características que compõe o vetor de características.

A criação de um índice de uma coluna *PGImage* é realizada pela função *Set_Feature_Vector* que associa um método de acesso e um vetor de características a um atributo *PGImage*. Note que não é permitida a criação do índice usando o comando *CREATE INDEX*.

Os comandos SQL-IE responsáveis pela configuração dos atributos *PGImage* serão descritos a seguir:

1. *Set_Feature_Vector*

Esta função associa um atributo *PGImage* de uma determinada tabela a um vetor de características previamente definido e cria uma estrutura de índice se um método de acesso estiver definido. Uma tabela interna associada ao par (nome da tabela,

nome do atributo) é criada ou alterada (novas colunas são adicionadas, se a tabela já havia sido criada) no sistema de catálogo estendido, como ilustrado na Figura 3.2(c). Cada coluna desta tabela corresponde a um extrator de características do vetor de características sendo configurado. Note que mais de um vetor de características pode ser associado a um mesmo atributo *PGImage* de uma tabela. Para evitar alocação de espaço de armazenamento e tempo de processamento extras, o PostgreSQL-IE reusa as colunas já criadas na respectiva tabela interna para definir novos vetores de características. A tabela *pge_tabvector* no sistema de catálogo estendido relaciona colunas das tabelas internas a seus respectivos vetores de características. Quando um vetor de características é configurado para uma tabela que já está populada, todos os extratores de características definidos pelo vetor de características sendo definido são executados para todas as tuplas da tabela.

Sintaxe

Set_Feature_Vector (<NomeIndice>, <NomeTabela>, <NomeColuna>, <NomeVetorCaracterísticas>[, <MetodoAcesso>])

<i>NomeIndice</i>	Nome do índice sendo criado.
<i>NomeTabela</i>	Nome da relação onde o atributo está sendo modelado.
<i>NomeColuna</i>	Nome do Atributo do tipo <i>PGImage</i> a ser indexado.
<i>NomeVetorCaracterísticas</i>	Nome do vetor de características definido no banco de dados.
<i>MetodoAcesso</i>	Método de acesso a ser utilizado pelo atributo. Caso não seja definido, a consulta usará o SAM. (<i>Opcional</i>)

Exemplos

```
/* Considerando os vetores de características definidos na seção 3.4.1:
   Vetor_Textura, Vetor_Forma e Vetor_Misto */
```

```
(a) SELECT Set_Feature_Vector('Ix_paciente_mama1', 'paciente',
                               'mama','Vetor_Textura');

(b) SELECT Set_Feature_Vector('Ix_paciente_mama2', 'paciente',
                               'mama','Vetor_Forma', 'slimtree');

(c) SELECT Set_Feature_Vector('Ix_paciente_mama3', 'paciente',
                               'mama','Vetor_Misto');
```

Neste exemplo, ao executar o comando (a), o PostgreSQL-IE cria uma tabela de dados visuais do catálogo, chamada *pge_paciente_mama*, com uma coluna que armazenará os valores de contraste. Note que o *Ix_paciente_mama1* não utilizará nenhum método de indexação, ou seja, qualquer consulta no atributo *mama* que utilize o *Ix_paciente_mama1* será realizada uma leitura de todos os registros da *pge_paciente_mama*. Executando o comando (b) após o comando (a), algumas colunas serão adicionadas à *pge_paciente_mama* para armazenar os valores de área, perímetro, compacidade e descritores de Fourier. Note que o índice sendo criado utilizará o método de acesso SlimTree [31]. Por último, executando o comando (c) nenhuma coluna será criada na *pge_paciente_mama*, pois todos os extratores do *Vetor_Misto* já estão definidos na *pge_paciente_mama*, e o *Ix_paciente_mama3* não utilizará nenhum método de indexação. Todos os comandos (a), (b) e (c) atualizam a tabela do catálogo *pge_tabvector* e cria funções de controle interno.

2. UnSet_Feature_Vector

Esta função exclui um índice previamente criado para uma determinada coluna *PGImage*. Ao executá-lo, a tabela administrativa que armazena os vetores de características da coluna será excluída, além de funções e *triggers* de controle interno.

Sintaxe

UnSet_Feature_Vector (<NomeIndice>)

NomeIndice

Nome do índice a ser excluído no banco de dados.

Exemplo

```
/* Excluir o índice 'Ix_paciente_mama1' */  
  
SELECT UnSet_Feature_Vector('Ix_paciente_mama1')
```

3.4.4 Recuperação de imagens por conteúdo

Atualmente o PostgreSQL-IE suporta dois operadores de similaridade: consulta aos K vizinhos mais próximos (KNN - *K nearest neighbors*) e consulta por raio de abrangência (RANGE). Os dois operadores devem estar sempre em subconsultas, onde a seleção é o próprio operador KNN ou RANGE, e a projeção é o conjunto de imagens mais similares à imagem de referência especificada pelo operador.

Os comandos SQL-IE responsáveis pelas operações de similaridade em atributos *PG-Image* serão descritos a seguir:

1. KNN

O operador KNN faz parte de uma subconsulta que retorna as k imagens de um atributo mais similares à imagem de referência.

Sintaxe

KNN ($\langle NomeScore \rangle$, $\langle K \rangle$, $\langle NomeIndice \rangle$, $\langle ImagensReferência \rangle$)

<i>NomeScore</i>	O parâmetro nome do <i>score</i> faz a integração com a função <i>Score_IE</i> que exibe o <i>score</i> entre a imagem de referência e as imagens sendo comparadas, permitindo usar mais de um operador de similaridade com a mesma consulta e relacionar os <i>scores</i> resultantes de cada operador adequadamente. Note que quanto menor o <i>score</i> maior a similaridade.
<i>K</i>	Indica o número máximo de imagens mais similares a serem recuperadas.
<i>NomeIndice</i>	O nome do índice, definido a partir da função <i>Set_Feature_Vector</i> (veja seção 3.4.3), identifica o vetor de característica a ser utilizado na consulta.
<i>ImagensReferência</i>	Atributo do tipo <i>PGImage</i> ou lista de imagens necessárias para a construção do vetor de característica da imagem de referência.

Onde: $\langle ImagensReferência \rangle :: Atributo_PGImage \mid \langle ListaImagens \rangle$
 $\langle ListaImagens \rangle :: ClasseImagem, CaminhoImagem \mid$
 $ClasseImagem, CaminhoImagem; \langle ListaImagens \rangle$

Exemplos

```
/* Exemplo 1:
Retornar id, registro e o score de similaridade da mama da
tabela paciente, onde a coluna mama deve ser uma das 10 mais
similares a imagem 'contorno.bmp'. Note que o nome do score deve
ser o mesmo utilizado pela função Score_IE e que as
características utilizadas na comparação são apenas aquelas
definidas pelo índice 'Ix_paciente_mama2' */
```

```
SELECT id, registro, Score_IE('sc_1',mama) AS score
FROM paciente
```

```

WHERE (mama).content in
      (SELECT * FROM KNN('sc_1', 10, 'Ix_paciente_mama2',
                        'contorno,C:/imagens/contorno.bmp') );
ORDER BY score;

```

/* Exemplo 2:

Este exemplo é o mesmo do anterior com as seguintes diferenças: as características estão ponderadas, ou seja, descritores de Fourier tem um maior peso na consulta em relação à compacidade que tem um maior peso em relação a perímetro, assim por diante; no exemplo anterior, o índice 'Ix_paciente_mama2' utiliza o método de acesso slimtree, já neste isso não é possível por causa da ponderação. Com isso, o operador KNN realizará um scan em todos os vetores de características do atributo mama */

```

SELECT id, registro, Score_IE('sc_2',mama) AS score
FROM paciente
WHERE (mama).content in
      (SELECT * FROM KNN('sc_2', 10,
                        'Ix_paciente_mama2:area=0.1;perimetro=0.2;
                        compacidade=0.3;fourier=0.4',
                        'contorno,C:/imagens/contorno.bmp') );

```

/* Exemplo 3:

Retornar id, registro e o score de similaridade da mama da tabela paciente, onde a coluna mama deve ser uma das 10 mais similares a coluna mama da paciente com id = 1 utilizando as características definidas no índice 'Ix_paciente_mama1' */

```

SELECT p1.id, p1.registro, Score_IE('sc_3',p1.mama) AS score
FROM paciente p1, paciente p2
WHERE p2.id = 1

```

```

AND p1.id <> 1
AND (p1.mama).content in
      (SELECT * FROM KNN('sc_3',10,'Ix_paciente_mama3',p2.mama) )
ORDER BY score

```

Todos os 3 exemplos retornam os registros em ordem de *score*, ou seja, de similaridade em relação a coluna *mama* da tabela *paciente*.

2. Raio de Abrangência (RANGE)

O operador RANGE faz parte de uma subconsulta que retorna as imagens de um atributo mais similares à imagem de referência dado um raio de abrangência *r*.

Sintaxe

RANGE (<NomeScore>, <r>, <NomeIndice>, <ImagensReferência>)

<i>NomeScore</i>	Nome do <i>score</i> para fazer a integração com a função <i>Score_IE</i> .
<i>r</i>	O raio <i>r</i> de abrangência.
<i>NomeIndice</i>	Nome do índice.
<i>ImagensReferência</i>	Atributo do tipo <i>PGImage</i> ou lista de imagens necessárias para a construção do vetor de característica da imagem de referência.

Onde: <ImagensReferência> :: Atributo_PGImage / <ListaImagens>

<ListaImagens> :: ClasseImagem, CaminhoImagem /

ClasseImagem, CaminhoImagem; <ListaImagens>

Exemplos

/* Exemplo 1:

```

Retornar id, registro e o score de similaridade da mama da
tabela paciente, onde a distância global entre a mama e a imagem
'contorno.bmp' deve ser menor ou igual a 0.9 */

```

```

SELECT id, registro, Score_IE('sc_1',mama) AS score
FROM paciente
WHERE (mama).content in
      (SELECT * FROM RANGE('sc_1', 0.9, 'Ix_paciente_mama2',
                           'contorno,C:/imagens/contorno.bmp') );
ORDER BY score;

```

/* Exemplo 2:

O mesmo exemplo só que agora com ponderação */

```

SELECT id, registro, Score_IE('sc_2',mama) AS score
FROM paciente
WHERE (mama).content in
      (SELECT * FROM RANGE('sc_2', 0.9,
                           'Ix_paciente_mama2:area=0.1;perimetro=0.2;
                           compacidade=0.3;fourier=0.4',
                           'contorno,C:/imagens/contorno.bmp') );

```

/* Exemplo 3:

Retornar id, registro e o score de similaridade da mama da
tabela paciente, onde a distância global entre mama da paciente
com id = 1 e a mama das pacientes restantes deve ser menor ou
igual a 1.2*/

```

SELECT p1.id, p1.registro, Score_IE('sc_3',p1.mama) AS score
FROM paciente p1, paciente p2
WHERE p2.id = 1
AND p1.id <> 1
AND (p1.mama).content in
      (SELECT * FROM RANGE('sc_3',1.1,'Ix_paciente_mama3',p2.mama) )
ORDER BY score;

```

3.4.5 Inserção, atualização e manipulação de imagens

O PostgreSQL-IE oferece várias funções para acessar e alterar o conteúdo das imagens. Para inserir uma imagem no banco de dados é necessário utilizar a função *Insert_Image* do SQL-IE no comando *INSERT*, e para atualizar o conteúdo de um atributo *PGImage* são utilizadas as funções *Update_Image_Attr* e *Replace_Image_Attr* do SQL-IE no comando *UPDATE*. Estas funções importam os arquivos das imagens e armazenam no PostgreSQL em forma de *large-objects*, além de atualizar a tabela *pge_regimage* e a tabela interna associada ao atributo sendo atualizado no catálogo estendido.

Ao excluir uma tupla que contenha atributos *PGImage*, usando o comando *DELETE*, não é necessário utilizar nenhuma função do PostgreSQL-IE, pois a remoção dos vetores de características da tupla é feita automaticamente por *triggers* que foram criados anteriormente por comandos de configuração do PostgreSQL-IE.

Os comandos SQL-IE responsáveis pela manipulação dos atributos *PGImage* serão descritos a seguir:

1. *Insert_Image*

Esta função é usada como parte do comando convencional *INSERT* do SQL. A função *Insert_Image* permite armazenar uma ou mais imagens no banco de dados, e dispara a execução de todos os extratores de características de todos os vetores de características associados ao atributo *PGImage* sendo inserido. Informações como largura e altura também são armazenadas.

Sintaxe

Insert_Image (*<NomeTabela>*, *<NomeColuna>*, *<ListaImagens>*)

<i>NomeTabela</i>	Nome da relação onde a tupla está sendo inserida.
<i>NomeColuna</i>	Nome do Atributo do tipo <i>PGImage</i> que está armazenando as imagens.
<i>ListaImagens</i>	Lista de imagens. Para cada imagem a ser inserida, deve-se informar a classe e o caminho do arquivo separados por ",". Caso exista mais de uma imagem, basta separá-las com ";".

Onde: $\langle ListaImagens \rangle :: ClasseImagem, CaminhoImagem /$
 $ClasseImagem, CaminhoImagem; \langle ListaImagens \rangle$

Exemplo

```
/* Inserir na tabela paciente a tupla com os valores 51 para a
   coluna id, '5042002-7' para a coluna registro, '22/03/1958' para a
   coluna data_nasc, a imagem mamografia.bmp para a classe mamografia
   da coluna mama e a imagem contorno.bmp para a classe contorno da
   coluna mama. */
```

```
INSERT INTO paciente (id,registro,data_nasc,mama)
VALUES ( 51, '5042002-7', '22/03/1958',
        Insert_Image('paciente', 'mama',
                     'mamografia,C:/Imagens/mamografia.bmp ;
                     contorno,C:/Imagens/contorno.bmp' )
```

A Figura 3.2 ilustra a inserção de uma tupla na tabela *paciente* e a atualização das tabelas *pge_regimage* e *pge_paciente_mama* que relacionam os vetores características associados à tabela *paciente*. Note que uma nova tupla é inserida na tabela *pge_paciente_mama* contendo todos os extratores de características relacionados com a nova tupla da tabela *paciente*.

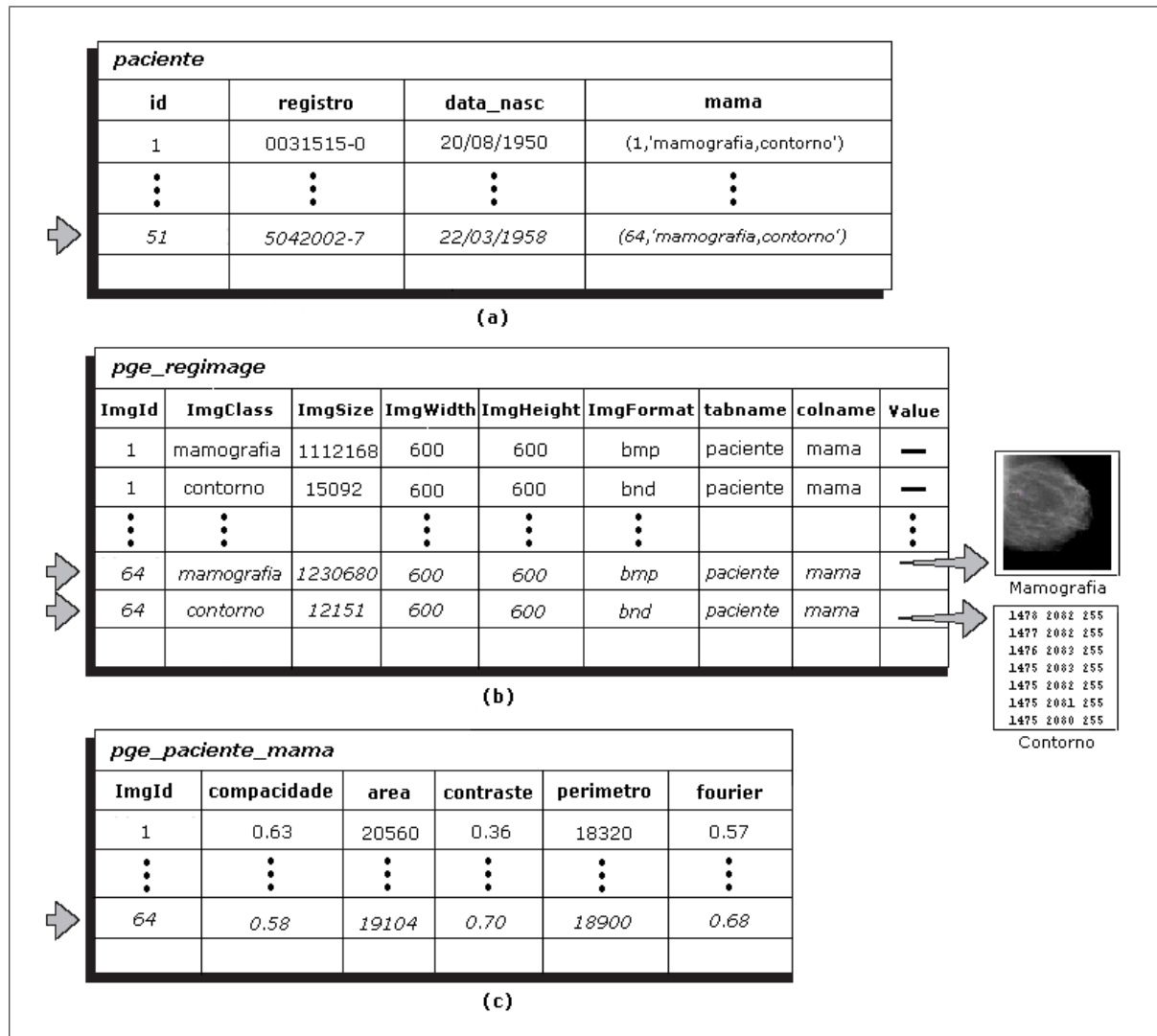


Figura 3.2: Inserção de uma tupla com atributo do tipo imagem. (a) tabela *paciente*; (b) a tabela *pge_regimage* com as informações das imagens armazenadas no atributo *mama*; (c) tabela interna que relaciona todos os vetores de características associados ao atributo *mama* da tabela *paciente*.

2. Update_Image_Attr

Ao atualizar uma tupla com o comando *UPDATE*, as colunas do tipo *PGImage* atualizam as informações somente com as funções *Update_Image_Attr* e *Replace_Image_Attr*. A função *Update_Image_Attr* atualiza as imagens cujas as classes estão sendo especificadas na função, calcula as características das novas imagens e atualiza os vetores de características na tabela de dados visuais do catálogo estendido.

Sintaxe

`Update_Image_Attr ( <NomeTabela>, <NomeColuna>, <Coluna>, <ListaImagens> )`

<i>NomeTabela</i>	Nome da relação onde as tuplas estão sendo atualizadas.
<i>NomeColuna</i>	Nome do Atributo do tipo <i>PGImage</i> que está atualizando as imagens.
<i>Coluna</i>	Atributo do tipo <i>PGImage</i> que está atualizando as imagens.
<i>ListaImagens</i>	Lista de imagens. Para cada imagem a ser inserida, deve-se informar a classe e o caminho do arquivo separados por " ". Caso exista mais de uma imagem, basta separá-las com "; "

Onde: `<ListaImagens> :: ClasseImagem, CaminhoImagem /
ClasseImagem, CaminhoImagem; <ListaImagens>`

Exemplo

```
/* Atualizar a coluna data_nasc com '22/03/1958' e a classe  
mamografia da coluna mama com a imagem 'mamografia.bmp'  
da tabela paciente, onde a coluna id seja igual a 1. */
```

```
UPDATE paciente  
SET data_nasc = '22/03/1950',  
    mama = Update_Image_Attr('paciente', 'mama', mama ,  
                             'mamografia,C:/Imagens/mamografia.bmp')  
WHERE id = 1
```

3. Replace_Image_Attr

A diferença entre esta função e a *Update_Image_Attr* é que para as imagens, cujas as classes não estão especificadas na função, é atribuído um valor nulo.

Sintaxe

`Replace_Image_Attr ( <NomeTabela>, <NomeColuna>, <Coluna>, <ListaImagens> )`

<i>NomeTabela</i>	Nome da relação onde as tuplas estão sendo atualizadas.
<i>NomeColuna</i>	Nome do Atributo do tipo <i>PGImage</i> que está atualizando as imagens.
<i>Coluna</i>	Atributo do tipo <i>PGImage</i> que está atualizando as imagens.
<i>ListaImagens</i>	Lista de imagens. Para cada imagem a ser inserida, deve-se informar a classe e o caminho do arquivo separados por ", ". Caso exista mais de uma imagem, basta separá-las com "; ".

Onde: `<ListaImagens> :: ClasseImagem, CaminhoImagem /
ClasseImagem, CaminhoImagem; <ListaImagens>`

Exemplo

```
/* Atualizar a coluna data_nasc com '22/03/1958', a classe  
mamografia da coluna mama com a imagem 'mamografia.bmp'  
e remover as outras classes da coluna mama da tabela  
paciente, onde a coluna id seja igual a 1. */
```

```
UPDATE paciente  
SET data_nasc = '22/03/1950',  
    mama = Replace_Image_Attr('paciente', 'mama', mama ,  
                             'contorno,C:/Imagens/contorno.bmp')  
WHERE id = 1
```

Note neste exemplo, a imagem associada à classe mamografia, caso exista, será excluída da base de dados, permanecendo apenas a imagem associada à classe contorno.

3.4.6 Funções diversas

Algumas funções, como a *Height_IE*, *Width_IE* e a *Value_Extractor* fornecem características da imagem e podem ser utilizadas tanto na projeção, quanto na seleção de um comando *SELECT* do SQL. Outras funções, como a *Show_Image*, podem ser executadas apenas na projeção.

1. Height_IE

As imagens com formatos conhecidos pelo PostgreSQL-IE, como *bmp*, *jpg* e *bnd*, possuem algumas informações adicionais, sendo a altura uma delas. Para obter a altura de uma imagem, basta utilizar a função *Height_IE* tanto na projeção quanto na seleção de um comando *SELECT*.

Sintaxe

Height_IE ([*<ClasseImagem>*,] *<Imagem>*)

ClasseImagem Classe da imagem do atributo PGImagem. Caso contenha somente uma classe definida para o atributo, não é necessário informar. (*Opcional*)

Coluna Atributo *PGImage* o qual vai extrair a altura da imagem.

Exemplos

```
/* Retornar o id, nome e altura das mamografias da coluna  
mama de todas as tuplas da tabela paciente. */
```

```
SELECT id, nome, height_IE('mamografia',paciente.mama)  
FROM paciente;
```

```
/* Retornar o id e registro das tuplas da tabela paciente  
que possuem a altura da mamografia da coluna mama maiores  
que 100. */
```

```
SELECT id, registro
```

```
FROM paciente
WHERE heigth_IE('contorno',paciente.mama) > 100;
```

2. Width_IE

Assim como a altura, a largura é outra informação adicional das imagens. Para obter a largura de uma imagem, basta utilizar a função *Width_IE* tanto na projeção quanto na seleção de um comando *SELECT*.

Sintaxe

Width_IE ([*<ClasseImagem>*,] *<Imagem>*)

ClasseImagem Classe da imagem do atributo PGImagem. Caso contenha somente uma classe definida para o atributo, não é necessário informar. (*Opcional*)

Coluna Atributo *PGImage* o qual vai extrair a largura da imagem.

Exemplos

```
/* Retornar o id, registro e largura das mamografias da coluna
   mama de todas as tuplas da tabela paciente. */
```

```
SELECT id, registro, width_IE('mamografia',paciente.mama)
FROM paciente;
```

```
/* Retornar o id e registro das tuplas da tabela paciente
   que possuem a largura da mamografia da coluna mama maiores
   que 100. */
```

```
SELECT id, registro
FROM paciente
WHERE width_IE('contorno',paciente.mama) > 100;
```

3. Value_Extractor

O conteúdo dos vetores de características podem ser consultados usando a função *Value_Extractor* tanto na projeção quanto na seleção de um comando *SELECT*. Com esta função, também é possível obter o valor de uma característica de imagens que não estão armazenadas no banco de dados e sim em arquivos.

Sintaxe

Value_Extractor (*<NomeExtrator>*, *<ImagensReferência>*)

NomeExtrator Nome do extrator de característica.

ImagensReferência Atributo *PGImage* ou lista de imagens necessárias para a execução do extrator.

Onde: *<ImagensReferência>* :: *ClasseImagem*, *Atributo_PGImage* / *<ListaImagens>*

<ListaImagens> :: *ClasseImagem*, *CaminhoImagem* /
ClasseImagem, *CaminhoImagem*; *<ListaImagens>*

Exemplos

```
/* Retornar a área da imagem 'imagem.bmp'. */
```

```
SELECT Value_Extractor('area', 'contorno,C:/Imagens/imagem.bmp')
```

```
/* Retornar o id, registro e área das tuplas que tiverem a  
  área da mama maior que 100 da tabela paciente. */
```

```
SELECT id, registro, Value_Extractor('area', paciente.mama)  
FROM paciente  
WHERE Value_Extractor('area','contorno', paciente.mama) > 100;
```

4. Show_Image

Esta função é necessária para obter a imagem a partir de uma tupla no banco de

dados. A função *Show_Image* extrai os *large-objects* do banco de dados relacionados às tuplas da consulta e os armazenam em um arquivo na máquina servidora.

Sintaxe

Show_Image (*<Imagem>*, *<ArquivoDestino>*)

Imagem Atributo *PGImage*.

ArquivoDestino Classe da imagem e o caminho do arquivo onde a imagem vai ser armazenada.

Exemplo

```
/* Retornar a mamografia da coluna mama para o arquivo de destino  
  'destino.bmp' da tabela paciente com o id = 1. */
```

```
SELECT Show_Image(mama,'mamografia,C:/Imagens/destino.bmp')  
FROM paciente  
WHERE id = 1
```

3.5 Métodos de acesso no PostgreSQL-IE

Apesar do PostgreSQL-IE permitir a inclusão de novos métodos de acesso multidimensionais, atualmente o PostgreSQL-IE disponibiliza apenas o SAM (*Standard Access Method*). No SAM, quando um operador de similaridade é usado em determinado atributo *PGImage*, todas as tuplas daquele atributo são lidas e realizadas o cálculo de similaridade.

Embora não haja otimização na consulta, o SAM permite a recuperação de imagens usando um vetor de características com uma combinação de vários extratores de características associados a diferentes pesos.

No cálculo de similaridade, cada característica (dimensão) é considerada separadamente. O vetor da imagem de referência é dado como objeto *x*, tal que $x = (x_1, x_2, \dots, x_n)$ e o vetor da imagem de consulta como objeto *y*, tal que $y = (y_1, y_2, \dots, y_n)$, onde *n* é o número de características dos vetores. Para cada característica *i* é identificado o valor

mais baixo b_i e o valor mais alto a_i da característica i de todos os objetos de pesquisa. A função de distância D_i entre os objetos x e y é dada pela Equação 3.1:

$$D_i(x_i, y_i) = \frac{|x_i - y_i|}{a_i - b_i} \quad (3.1)$$

O valor da distância D_i está no intervalo $[0,1]$ e representa o grau de dissimilaridade entre as características i das imagens, ou seja, quanto menor for o valor de D_i mais similares serão as imagens considerando a característica i . Com isso, todas as características (dimensões) ficam normalizadas. Os operadores de similaridade, vistos na seção 3.4.4, permitem atribuir pesos às características, com isso o cálculo da distância global entre os vetores de características é definido pela Equação 3.2:

$$D(x, y) = \sum_{i=1}^n W_i D_i \quad (3.2)$$

O SAM visa normalizar cada característica da imagem, para que a distância entre uma característica de duas imagens fique no intervalo $[0,1]$. Caso uma consulta qualquer dar mais importância a uma determinada característica, basta atribuir um peso a mesma. Quanto maior o peso, mais a característica será importante no cálculo de similaridade entre duas imagens. A avaliação do SAM está além do escopo deste trabalho. Maiores detalhes sobre as métricas utilizadas no cálculo de similaridade podem ser encontradas em Guang-Ho Cha [32].

3.6 Considerações Finais do Capítulo

Este capítulo descreveu a extensão PostgreSQL-IE para suportar recuperação de imagens por conteúdo. A extensão proposta possui o código aberto, é portátil, e é disponível em versões para os sistemas operacionais Windows e Linux. O PostgreSQL-IE suporta um novo tipo de dados para representar imagens que permite modelar um esquema relacional armazenando várias imagens de diferentes classes em um mesmo atributo. Esta abordagem torna possível a combinação de extratores de características de diferentes imagens em um mesmo vetor de características.

O PostgreSQL-IE é apropriado para suportar qualquer tipo de aplicação, pois os descritores de características e métodos de acesso são criados e configurados pelo próprio administrador da aplicação, o que não acontece com as extensões de SGBDRs comerciais disponíveis no mercado, nem com os padrões estabelecidos pela ISO SQL/MM Part 5: Still Image.

A linguagem estendida (SQL-IE) é de fácil utilização, flexível e possui todos os recursos necessários para a aplicação CBIR. Qualquer software, seja com interface *web* ou *desktop*, pode acessar um banco de dados com imagens e utilizar recursos do PostgreSQL-IE sem a necessidade de instalar softwares ou pacotes adicionais.

Um exemplo de aplicação que está utilizando o PostgreSQL-IE é o sistema SISPRIM (Sistema de Pesquisa para Recuperação de Imagens Mamográficas) [2]. Este sistema faz parte do projeto AMDI - Atlas Indexado de Mamografias Digitais [1], [33], [34], e será descrito no próximo capítulo.

Capítulo 4

SISPRIM - Sistema de Pesquisa com Suporte para Recuperação de Imagens Mamográficas Baseada em Conteúdo

4.1 Introdução

Câncer de mama é o segundo tipo de câncer mais freqüente no mundo e o primeiro entre as mulheres (cerca de um milhão de casos estimados para 2006). No Brasil o número estimado de casos em 2006 é de 48.930 com um risco estimado de 52 casos a cada 100 mil mulheres [35]. Apesar de ser considerado um câncer de bom prognóstico se diagnosticado e tratado precocemente, as taxas de mortalidade por câncer de mama continuam elevadas no Brasil, provavelmente devido ao diagnóstico tardio [35].

A mamografia é um exame de raio-x das mamas que utiliza baixa dose de radiação e que tem sido largamente utilizada para analisar e diagnosticar tumores de mama. Casos ambíguos que apresentem características suspeitas são avaliados usando procedimentos adicionais como vistas complementares, ultra-sonografia, ressonância magnética ou medicina nuclear. A biopsia é indicada quando os métodos de diagnóstico por imagem indicam uma suspeita de malignidade, sendo necessária uma confirmação.

Embora a mamografia seja atualmente o melhor método para a detecção de câncer de mama, entre 10% e 30% das mulheres com presença de câncer de mama têm diagnóstico

falso negativo, ou seja, o radiologista interpretou o exame erroneamente como normal. Em aproximadamente 67% destes falsos negativos o radiologista falhou na detecção do câncer que era evidente em uma análise retrospectiva [36]. Esses dados têm justificado o desenvolvimento de métodos de processamento digital de imagens para a detecção e análise de características obtidas a partir das mamografias, de tal maneira a auxiliar na redução de erros de diagnóstico, na redução de procedimentos auxiliares, na redução da mortalidade bem como dos custos com a saúde. Estes dados têm justificado também o desenvolvimento de sistemas computacionais que auxiliem o radiologista em atividades de ensino-aprendizagem para leitura e interpretação de mamogramas, como também em atividades de pesquisa e na rotina de uma clínica de radiologia.

Para endereçar este tipo de aplicação está sendo desenvolvido o AMDI - Atlas Indexado de Mamografias Digitais [1], [33], [34]. O AMDI é um sistema que integra módulos para permitir a adição de novos casos em uma base de dados mamográfica, e auxiliar atividades de pesquisa e educação em câncer de mama através de uma interface flexível, fácil e disponível pela *web*. A Figura 4.1 mostra a representação esquemática da arquitetura do AMDI.

Este capítulo apresenta o SISPRIM (Sistema de Pesquisa com suporte para Recuperação de Imagens Mamográficas baseada em conteúdo) a ser integrado no AMDI. O SISPRIM foi implementado utilizando o módulo PostgreSQL Image Extender (PostgreSQL-IE) apresentado no capítulo anterior.

4.2 Visão Geral do SISPRIM

O SISPRIM, ilustrado na Figura 4.2, é um software livre e disponível pela *web*, que permite elaborar consultas à base de dados mamográfica utilizando filtros convencionais, como histórico clínico e estilo de vida da paciente, e operações de similaridade entre imagens mamográficas. Os descritores de características atualmente disponíveis no SISPRIM são: *area*, *perimeter*, *compactness*, *fourier descriptors*, *fraccional concavity* e *spiculation index* [37], [38], [39], [40], [41]. A avaliação destes extratores está além do escopo deste trabalho.

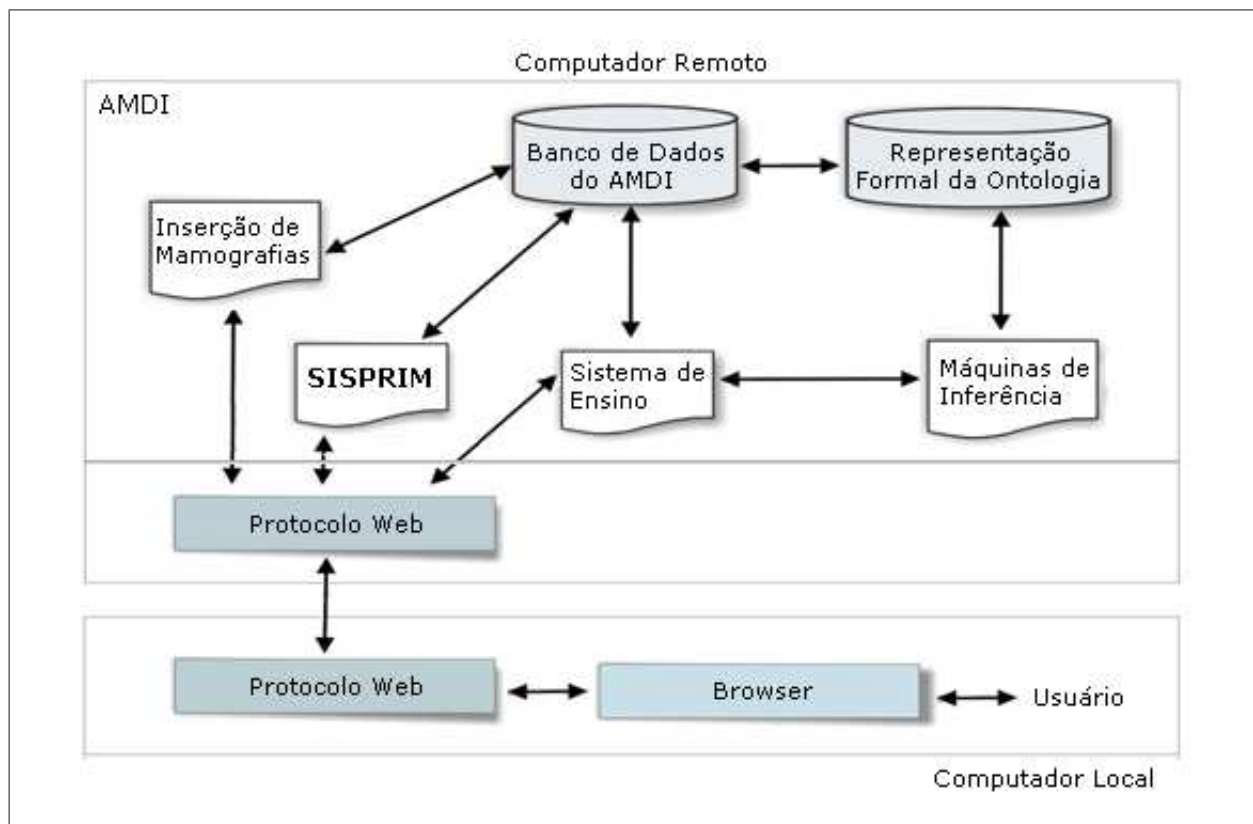


Figura 4.1: Esquema geral do fluxo de informações no AMDI. Fonte: Guliato et al. [1].

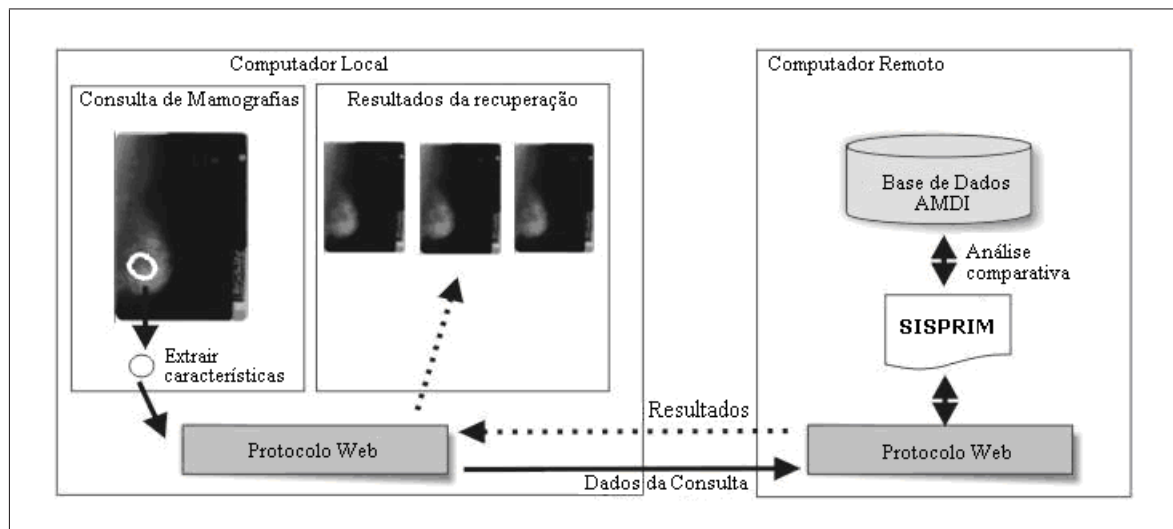


Figura 4.2: SISPRIM: O sistema de pesquisa do AMDI. Fonte: Melo et al. [2].

4.3 Base de Dados Mamográfica

A base de dados mamográficas do AMDI foi modelada para que cada caso contivesse informações sobre o histórico clínico da paciente, seu estilo de vida, as duas vistas padrões

de cada mama [crânio caudal (CC) e médio-lateral oblíqua (MLO)], os achados radiológicos incluindo distorção arquitetural, densidade assimétrica, presença de nódulo palpável, presença de grupamentos de calcificações e neste caso, localização do agrupamento e características das calcificações de acordo com sua morfologia e distribuição. A base de dados mantém ainda informações sobre a presença de lesões, e suas características de acordo com a margem, forma, densidade e textura. As características dos achados radiológicos presentes na mamografia, e o seu respectivo diagnóstico são descritos de acordo com a classificação BI-RADS [42]. A base de dados mamográfica do AMDI permite que novos exames de uma mesma paciente sejam armazenados ao longo do tempo, além de exames complementares. O contorno da mama, o contorno do músculo peitoral (apenas para a vista MLO), o contorno da lesão e os limites do agrupamento de calcificações, desenhados por radiologistas experientes, usando uma interface gráfica adequada, também são armazenados na base de dados mamográficos. O Anexo A apresenta a base de dados mamográfica modelada utilizando o PostgreSQL-IE.

Note que a tabela *Imagem*, veja Figura 5.2 - Anexo A, possui um atributo *IMG_Imagem* do tipo *PGImage*. Este atributo está modelado para conter até 9 imagens diferentes, conforme ilustrado na Figura 5.5 - Anexo A. A classe *mammography* está associada à imagem original, *contour_breast* está associada à imagem que contém apenas o contorno da mama, *contour_pectoral_muscle* está associada à imagem que contém a linha que separa o músculo peitoral da mama, *contour_lesion1*, *contour_lesion2*, *contour_lesion3* são três classes de imagens que contém o contorno de uma lesão, se presente na imagem original, cada uma desenhada por um radiologista de forma independente, e *calcification1*, *calcification2*, *calcification3* são três classes de imagens que contém os limites de agrupamentos de calcificações, se presentes na imagem original, cada agrupamento detectado por um radiologista.

4.4 Interface de Configuração do SISPRIM

O acesso e consultas à base de dados mamográfica do AMDI usando comandos SQL-IE diretamente pode ser uma tarefa difícil para profissionais não treinados na área de

banco de dados. Visando disponibilizar um sistema fácil de ser configurado e utilizado, o SISPRIM foi projetado de tal modo a conter duas interfaces: a de configuração para o usuário-administrador, e a de consulta para o usuário-pesquisador.

Para o usuário-administrador é exibida uma interface que permite a criação e/ou remoção de descritores de características, de vetores de características e de métodos de indexação no banco de dados do AMDI.

Considere o esquema relacional do AMDI apresentado no Anexo A. O campo *IMG_Imagem* da tabela *Imagem* armazena várias imagens, entre elas a imagem cuja classe é *contour_lesion1*. Para exemplificar a interface de configuração do SISPRIM, serão consideradas a criação de um novo extrator de características, a definição de um novo vetor de características e a associação do atributo *IMG_Imagem* a um vetor de características.

De acordo com a Figura 4.3, o usuário-administrador já incluiu no sistema os extratores de características: *area*, *compactness*, *fraccional concavity*, *fourier descriptors* e *perimeter* e está em processo de criação do extrator de características *spiculation index*.

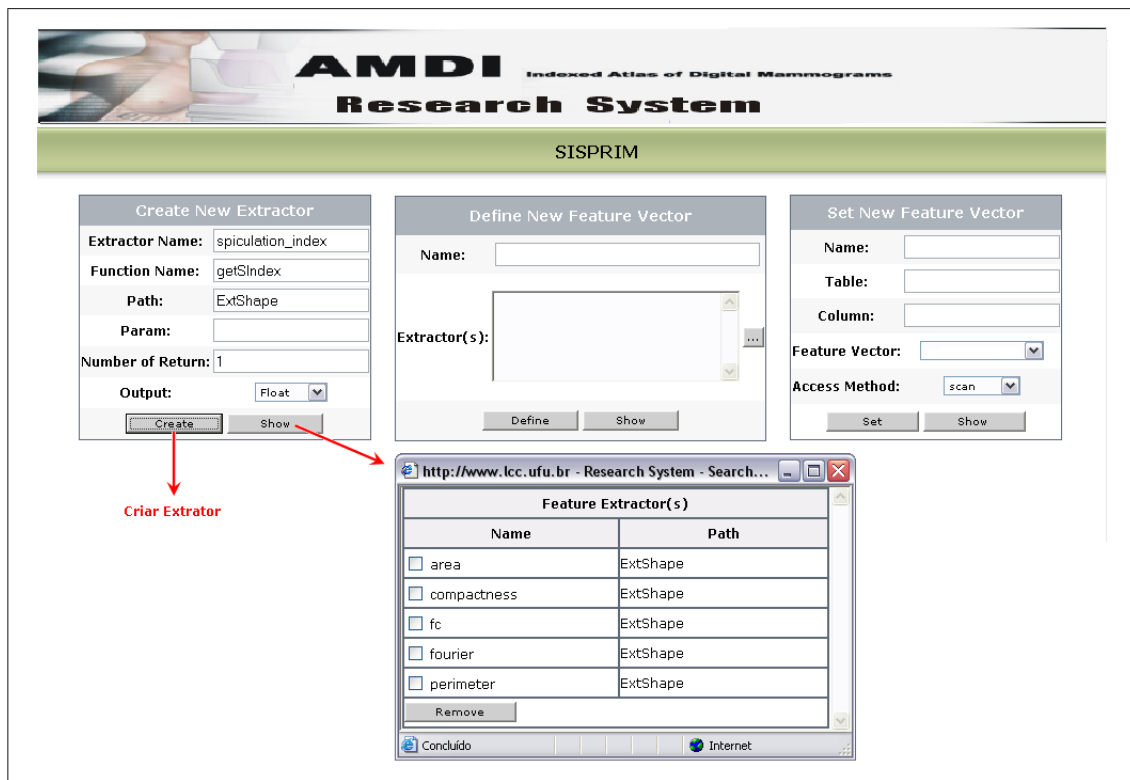


Figura 4.3: Interface para a administração do SISPRIM: configurando extratores de características.

Uma vez criados todos os extratores de características, o usuário-administrador deve definir os vetores de características. A Figura 4.4 ilustra o processo de definição do vetor de características *Vector1* combinando as características: *compactness*, *fourier descriptors* e *spiculation index*.

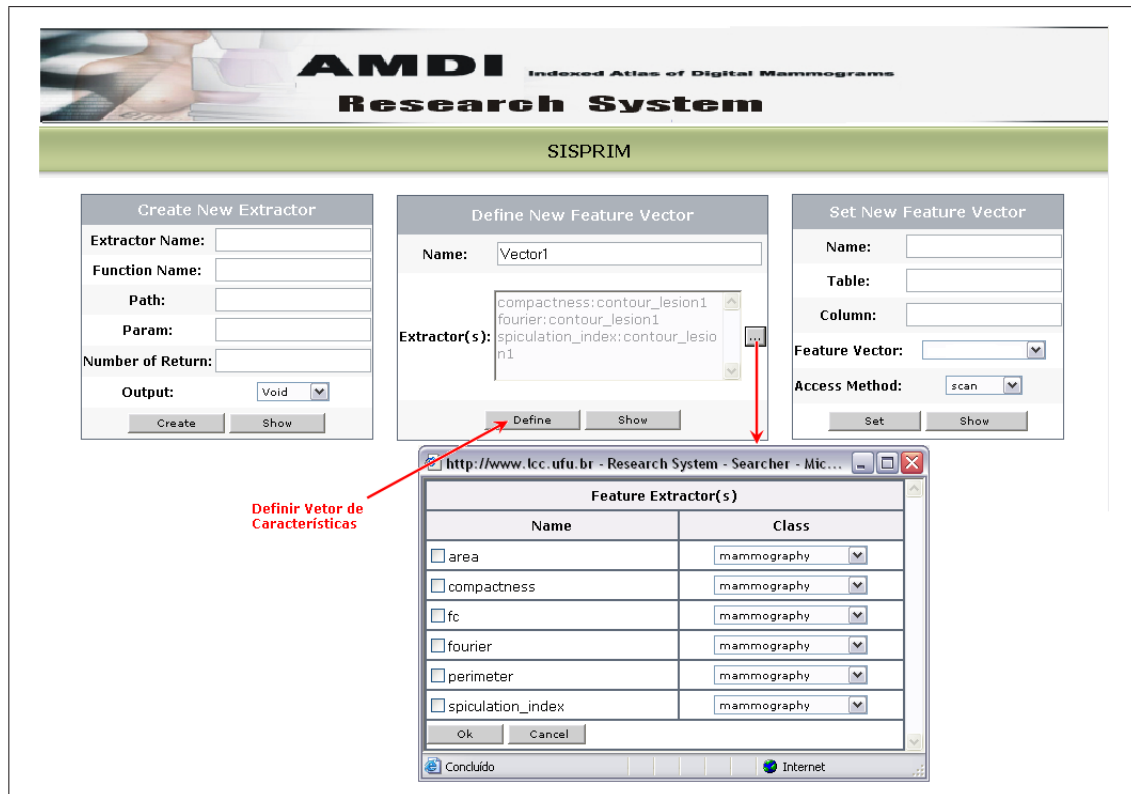


Figura 4.4: Interface para a administração do SISPRIM: definindo vetores de características.

Uma vez definidos os vetores de características, o usuário-administrador deve associar a cada um destes vetores uma chave de pesquisa do tipo *PGImage* e, opcionalmente, um método de acesso. Caso nenhum método de acesso seja informado, o SISPRIM considera que não há índice associado ao vetor de características e executa a consulta percorrendo todas as tuplas da tabela (ou das tabelas) envolvida(s). Como exemplo foi criada a consulta *Query1* associando o vetor de característica *Vector1* à chave de pesquisa *IMG_Imagem* da tabela *Imagem*, veja a Figura 4.5. Neste momento o PostgreSQL-IE calcula as instâncias dos vetores de características para cada tupla da tabela *Imagem* e armazena estas instâncias no catálogo estendido do PostgreSQL-IE.

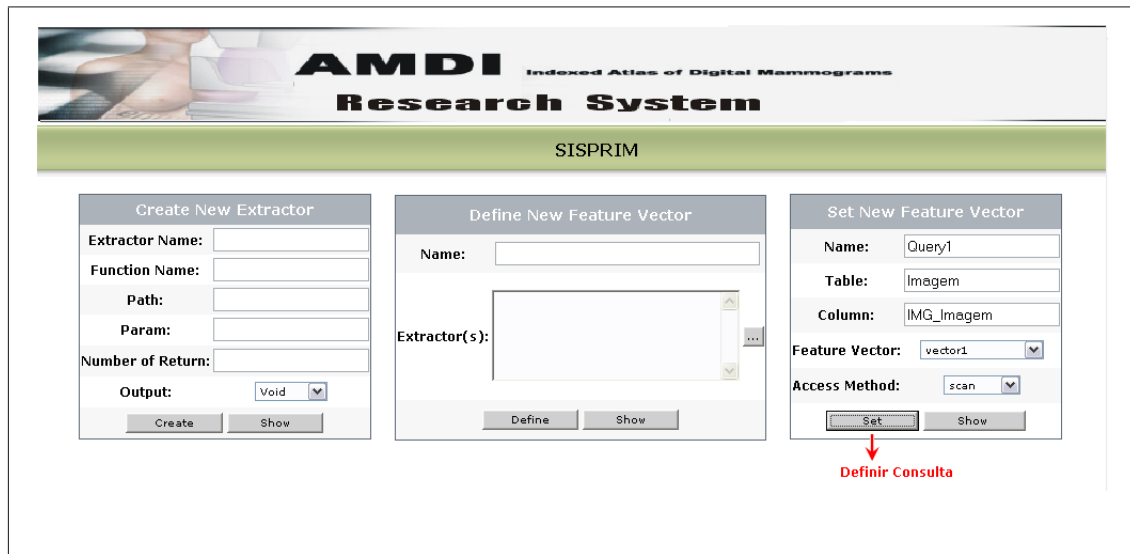


Figura 4.5: Interface para a administração do SISPRIM: definindo consultas.

4.5 Interface de Consulta do SISPRIM

Uma vez configurado o sistema de pesquisa, o usuário-pesquisador poderá elaborar suas pesquisas, correlacionando dados clínicos da paciente com achados radiológicos e imagens mamográficas. Um exemplo é ilustrado na Figura 4.6. Considere a seguinte consulta:

"Retorne as 2 imagens mais similares à imagem do contorno de uma lesão de mama, dada como imagem de entrada, de acordo com Query1 e cuja idade da paciente esteja no intervalo entre 30 e 50 anos. Exiba no relatório final da consulta o registro da paciente, idade da paciente, idade da primeira gravidez, o score e o thumbnail da lesão."

Como à consulta *Query1* está associado o vetor de características *Vector1* com 3 descritores de características, estes descritores são mostrados na interface do SISPRIM, podendo ser associados a pesos que variam no intervalo real $[0,1]$, caso o usuário-pesquisador julgue necessário. Para a realização desta consulta, o vetor de características *Vector1* é calculado para a imagem de referência e comparado com todas as instâncias do vetor característica *Vector1* disponíveis no catálogo da extensão do PostgreSQL-IE. Note que o comando SQL-IE executado é apresentado na base da interface. As duas imagens mais similares à imagem de referência e cujas pacientes possuírem entre 30 e 50 anos retornam como resposta à consulta, veja Figura 4.7. A resposta à consulta é mostrada na ordem crescente do score, isto é, as lesões armazenadas na base de dados são exibidas na ordem

crescente de similaridade em relação à lesão dada como entrada.

A flexibilidade para realizar a consulta, combinando dados textuais com dados visuais, definindo as informações que deverão constar no relatório de resposta, e ponderando os descritores de características da imagem de referência, possibilita aos especialistas recuperarem informações facilmente, e aos pesquisadores, um alto poder de análise da base de dados mamográficos.

4.6 Considerações Finais do Capítulo

O AMDI é um sistema público, acessado via Web, que integra ferramentas que auxiliam na construção e edição de uma base de dados de mamografias digitais e nas atividades de pesquisa e de ensino na área da radiologia de mama. O SISPRIM é um módulo integrado ao AMDI que possibilita correlacionar o histórico clínico da paciente e seu estilo de vida com características extraídas automaticamente das imagens mamográficas usando uma interface gráfica amigável. Atualmente o SISPRIM inclui seis descritores de características e dois operadores de similaridade, mas novos extratores podem ser facilmente incluídos no sistema pelo usuário-administrador. Esta flexibilidade é devido ao fato do SISPRIM ter sido desenvolvido usando o PostgreSQL-IE.

O PostgreSQL-IE permite que as aplicações, seja *web* ou *desktop*, possam disponibilizar CBIR sem a necessidade de implementações adicionais, bastando apenas utilizar comandos SQL-IE na conexão com o banco de dados. Além disso, novas estratégias de consultas por similaridade de imagens podem ser facilmente adicionadas ao banco de dados.

Text-Based Retrieval

Field: Age

Value:

initial age

end age

Insert Field

Text-Based Query

Field	Value
☐ Age	40 - 60
Remove	

Return Fields

Field: ---

Insert Field

All

Fields

☐ Age
☐ First Pregnancy
☐ Id-Patient
Remove

Features

Feature Index: query1

Feature Vector: vector1

Table: imagem

Column: img_imagem

Access Method: scan

Features Extractor(s)

Weight	Name	Path
1.0	compactness	ExtShape
1.0	fourier	ExtShape
1.0	spiculation_index	ExtShape

Content-Based Image Retrieval

Reference:

☒ Id-Case: 10

☐ File:

Upload

Similarity:

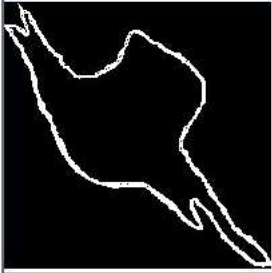
☒ KNN: 2 K

☐ RANGE: R

Image(s)

View

Class: contour_lesion1



SQL - IE Command

```

SELECT Paciente.pac_registro, Paciente.pac_primeiragravidez,
EXTRACT(YEAR FROM age(current_date,Paciente.pac_datanasc)) AS Idade,
Imagem.IMG_Imagem, Score_IE('sc_1',Imagem.IMG_Imagem) AS score
FROM Paciente, Caso, Imagem
WHERE Paciente.pac_registro = Caso.pac_registro
AND Caso.cas_codigo = Imagem.cas_codigo
AND (Imagem.IMG_Imagem).imgid in
(SELECT * FROM KNN ('sc_1', 2, 'Query1',
'contour_lesion1,imagens/contour.bnd' ) )
AND (EXTRACT(YEAR FROM age(current_date,Paciente.pac_datanasc))>=30
AND
EXTRACT(YEAR FROM age(current_date,Paciente.pac_datanasc))<=50)
ORDER BY score

```

Create

Execute

Figura 4.6: Interface de pesquisa do SISPRIM para realizar a consulta *Query1*.

64

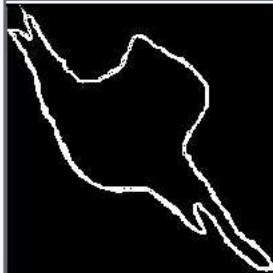
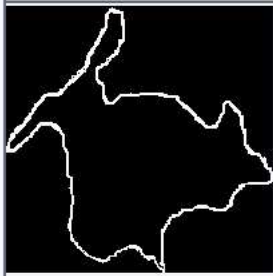
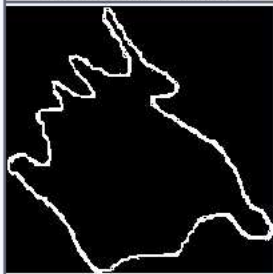
Result	
(Reference)	
Data	Image(s)
Id-Patient: ---	View Class: contour_lesion1 
Age: ---	
First Pregnancy: ---	
Score: ---	
(Register 1)	
Data	Image(s)
Id-Patient: 25	View Class: contour_lesion1 
Age: 55	
First Pregnancy: 44	
Score: 0.3050	
(Register 2)	
Data	Image(s)
Id-Patient: 38	View Class: contour_lesion1 
Age: 55	
First Pregnancy: 44	
Score: 0.3350	

Figura 4.7: Resultado da consulta *Query1*.

Capítulo 5

Conclusão

Recuperação de imagens por conteúdo (CBIR) é uma área que vem crescendo com impacto importante na área de computação. Inicialmente os sistemas envolvendo recuperação de imagens por conteúdo eram desenvolvidos para uma aplicação específica e suportados por Sistemas de Gerenciamento de Banco de Dados Relacionais (SGBDR) convencionais. Nestes sistemas as imagens eram armazenadas em diretórios separados da base de dados, os descritores e vetores de características, e métodos de indexação podiam ser vistos como funções da aplicação. Exemplos de sistemas CBIR dependentes de aplicação são VisualSEEk, PhotoBook e MARS.

Para endereçar a necessidade crescente por plataformas que suportassem o desenvolvimento de sistemas CBIR independente de aplicação, alguns SGBDR comerciais como DB2, Informix e Oracle foram estendidos, permitindo o armazenamento de imagens na base de dados, como um tipo especial de dados, e disponibilizando consultas por similaridade limitado a um conjunto pré-definido de extratores de características. No entanto, estas extensões são de alto custo financeiro, possuem arquitetura fechada e não permitem a inclusão de novos descritores de características, ou de novos métodos de acesso.

Este trabalho apresentou o PostgreSQL - IE, uma extensão do SGBDR PostgreSQL que tem como principais características ser portátil, independente de aplicação, código aberto e de livre acesso. Além destas características, o PostgreSQL - IE disponibiliza a linguagem de consulta e manipulação de dados SQL - IE, que define novas funções e um novo tipo de dados para tratar imagens denominado *PGImage*. Um atributo declarado

como sendo do tipo *PGImage* pode conter uma ou mais imagens de diferentes classes, o que torna possível combinar em um mesmo vetor, características de diferentes imagens armazenadas em um mesmo atributo, aumentando assim, a flexibilidade das consultas por similaridade. Um atributo do tipo *PGImage* também facilita a etapa de modelagem de uma base de dados, uma vez que reduz a necessidade de tabelas de relacionamento.

A linguagem SQL - IE é composta de um conjunto de funções que incluem comandos para criar novos extratores de características, novos métodos de acesso, e para definir novos vetores de características como uma combinação dos extratores previamente criados. SQL - IE também inclui recursos que permitem definir consultas combinando dados convencionais com dados visuais.

Para ilustrar a potencialidade da extensão proposta, foi desenvolvido o SISPRIM - Sistema de Pesquisa para Recuperação de Imagens Mamográficas. A base de dados mamográfica foi armazenada usando o POSTGRESQL - IE. Para o desenvolvimento do SISPRIM foram criados 6 extratores de características: *area*, *compactness*, *fourier descriptors*, *fractional concavity* e *spiculation index*, e 2 operadores de similaridade: aos K vizinhos mais próximos e por intervalo de abrangência. Como ainda não há um método de acesso implementado de acordo com o padrão exigido pela extensão, estes operadores são executados percorrendo todas as tuplas das tabelas envolvidas na consulta. Para facilitar tanto a etapa de configuração como a de consulta, o SIPRIM disponibiliza interfaces gráficas que tornam estas tarefas mais fáceis para o usuário.

Como principais contribuições deste trabalho, destacam-se: a extensão do catálogo do sistema do POSTGRESQL, e o conjunto de funções, que combinados, compõem o POSTGRESQL - IE, um SGBDR que suporta o desenvolvimento de sistemas independente de aplicação para recuperação de imagens por conteúdo à baixo custo financeiro. A extensão proposta está em processo de registro junto ao INPI (Instituto Nacional da Propriedade Industrial).

Para aumentar a potencialidade do POSTGRESQL - IE novos extratores, novos métodos de acesso devem ser desenvolvidos e incorporados ao SGBDR.

Referências Bibliográficas

- [1] D. Guliato, R. S. Boaventura, E. V. Melo, and R. M. Rangayyan. AMDI - indexed atlas of digital mammograms that integrates case studies, e-learning, and research systems via the web. In *J.S. Suri and R.M. Rangayyan (Ed.). Recent Advances in Breast Imaging, Mammography, and Computer-aided Diagnosis of Breast Cancer*. SPIE, Bellingham, WA, USA, pages 529-555, 2006.
- [2] E. V. Melo, D. Guliato, R. M. Rangayyan, and R. C. Soares. SISPRIM - sistema de pesquisa com suporte para recuperação de imagens mamográficas baseada em conteúdo. *WIM 2006 - VI Workshop de Informática Médica*, 2006.
- [3] C. H. C. Leung. Architecture on an image database system. *Information Services & Use*, 10:391–397, 1990.
- [4] G. Bordogna. Pictorial indexing for an integrated pictorial and textual information retrieval environment. *Information Services & Use*, 16:165–173, 1990.
- [5] T. Kato. Database architecture for content-based image retrieval, image storage and retrieval systems. In *Proc SPIE 1662*, pages 112–123, 1992.
- [6] H. Alto, R. M. Rangayyan, and J. E. L. Desautels. Content-based retrieval and analysis of mammographic masses. *Journal of Electronic Imaging*, 14(2):1–17, 2005.
- [7] M. A. Nascimento, V. Sridhar, and X. Li. Effective and efficient region-based image retrieval. *Journal of Visual Languages and Computing*, 14:151–179, 2003.

- [8] A. Traina, C. Traina Jr, B. F. Chino, and P. M. A. Marques. Efficient content-based image retrieval through metric histograms. *World Wide Web Journal*, 6(2):151–185, 2003.
- [9] Y. Xiao, Y. Wang, and D.J. Felleman. A spatially organized representation of colour in macaque cortical area v2. *Nature*, 421:535–539, 2003.
- [10] J. Felipe, A. Traina, and C. Traina Jr. Retrieval by content of medical images using texture for tissue identification. *16th IEEE Symposium on Computer-Based Medical Systems (CBMS03)*, pages 175–180, 2003.
- [11] T.M. Lehmann B.B. Wein and H. Greenspan. Integration of content-based image retrieval to picture archiving and communication systems. In *Medical Informatics Europe Conference*, 2003.
- [12] S. Antani, Xu L.R. Long, and G.R. Thoma. Partial shape matching for cbir of spine xray images. *SPIE Electronic Imaging, Storage and Retrieval Methods and Applications for Multimedia*, 5307:1–8, 2004.
- [13] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. The QBIC project: Querying image by content using color, texture, and shape. In *Proc. SPIE Storage and Retrieval for Image and Video Database*, pages 173–187, 1993.
- [14] J. R. Smith and S. F. Chang. Tools and techniques for color image retrieval. *Symposium on Electronic Imaging: Science and Technology - Storage & Retrieval for Image and Video Databases IV*, 2670:426–437, February 1996.
- [15] Chad Carson, Megan Thomas, Serge Belongie, Joseph M. Hellerstein, and Jitendra Malik. Blobworld: A system for region-based image indexing and retrieval. In *Third International Conference on Visual Information Systems*. Springer, 1999.
- [16] IBM Corporation. Db2 universal database image, audio, and video extenders administration and programming. 2000.

- [17] Informix Corporation. Excalibur image datablade module, user's guide. 2000.
- [18] Oracle Corporation. Oracle8i intermedia audio, image, and video - user's guide and reference. 2005.
- [19] Informix Corporation. Image foundation datablade module, user's guide. 2000.
- [20] ISO. ISO/IEC IS 13249-5:2001 SQL/MM, information technology database languages SQL multimedia and application packages part 5: Still image. 2001.
- [21] Informix Corporation. Excalibur image datablade module, user's guide. 1998.
- [22] M. Stonebraker, L. A. Rowe, and M. Hirohama. The implementation of postgres. *Knowledge and Data Engineering*, 2(1):125–142, 1990.
- [23] The POSTGRES Group. The POSTGRES reference manual. *Computer Science Division, University of California, Berkeley*, January 1993.
- [24] The POSTGRES Group. Postgresql 8.0.0 documentation. 2005.
- [25] Douglas Comer. The ubiquitous b-tree. *Computing Surveys*, 11(2):121–137, 1979.
- [26] Antonio Guttman. R-tree: A dynamic index structure for spatial searching. In *International Conference on Management of Data*. ACM-SIGMOD, pages 47-57, 1984.
- [27] W. Litwin. Linear hashing: A new tool for file and table addressing. In *6th Conference on Very Large Databases*, pages 212–223, 1980.
- [28] Joseph M. Hellerstein, Jeffrey F. Naughton, and Avi Pfeffer. Generalized search trees for database systems. In Umeshwar Dayal, Peter M. D. Gray, and Shojiro Nishio, editors, *Proc. 21st Int. Conf. Very Large Data Bases, (VLDB)*. Morgan Kaufmann, pages 562-573, 1995.
- [29] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of 23rd International Conference on Very Large Data Bases*, pages 426–435, 1997.

- [30] C. E. Martins. Slim⁺-tree: Métodos de acesso métrico baseado em medidas de dispersão. Dissertação de mestrado, Universidade Federal de Uberlândia, Faculdade de Computação, 2005.
- [31] Caetano Traina Jr., Agma Traina, Bernhard Seeger, and Christos Faloutsos. Slim-Trees: High performance metric trees minimizing overlap between nodes. *Lecture Notes in Computer Science*, 1777:51–65, 2000.
- [32] Guang-Ho Cha. Bitmap indexing method for complex similarity queries with relevance feedback. In *Proceedings of the 1st ACM international workshop on Multimedia databases*, pages 55–62. ACM Press New York, NY, USA, 2003.
- [33] D. Guliato, M. Caetano, F. R. Janones, S. C. Lima, V. Deus, R. M. Rangayyan, R. S. Boaventura, P. M. A. Marquez, and J. A. H. Rodrigues. AMDI - um atlas indexado para mamografias digitais via web. *III Congresso Latinoamericano de Engenharia Biomédica*, pages 529–532, 2004.
- [34] D. Guliato, R. S. Boaventura, E. V. Melo, V. Deus, F. R. Janones, and R. M. Rangayyan. AMDI: An atlas to integrate case studies, e-learning, and research systems via the web. In *Proc. IASTED International Conference on Telehealth*, pages 69–74, 2005.
- [35] INCA Instituto Nacional do Câncer. Incidência de câncer no brasil. 2006. <http://www.inca.gov.br>. Visitado em 05/06/2006, 2006.
- [36] M. Giger. Computer aided diagnosis of breast lesions in medical images. *Computing in Medicine*, pages 39–45, September/October 2000.
- [37] R. M. Rangayyan, N. R. Mudigonda, and J. E. L. Desautels. Boundary modeling and shape analysis methods for classification of mammographic masses. *Medical and Biological Engineering and Computing*, 38:487–495, 2000.
- [38] B. S. Sahiner, H. P. Chan, N. Petrick, M. A. Helvie, and L. M. Hadjiiski. Improvement of mammographic mass characterization using speculation measures and morphological features. *Medical Physics*, 28(7):1455–1465, 1997.

- [39] D. Guliato, R. M. Rangayyan, J. D. Carvalho, and S. A. Santiago. Spiculation-preserving polygonal modeling of contours of breast tumors. *IEEE International Conference of the Engineering in Medicine and Biology Society (EMBC)*, 2006.
- [40] Z. Huo, M. L. Giger, C. J. Vyborny, D. E. Wolverton, and C. E. Metz. Computerized classification of benign and malignant masses on digitized mammograms: A study of robustness. *Academic Radiology*, 7(12):1077–1084, 2000.
- [41] Z. Huo, M. L. Giger, and C. J. Vyborny. Computerized analysis of multiple-mammographic views: Potential usefulness of special view mammograms in computer-aided diagnosis. *IEEE Transactions on Medical Imaging*, 20(12):1285–1292, 2001.
- [42] American College of Radiology. Breast imaging - reporting and data system birads, 4th edition. 2004.

APÊNDICE A

A modelagem do banco de dados mamográficos do AMDI é ilustrada nas Figuras 5.1, 5.2, 5.3 e 5.4. Alguns comandos SQL-IE para a construção do banco de dados são ilustrados na Figura 5.5.

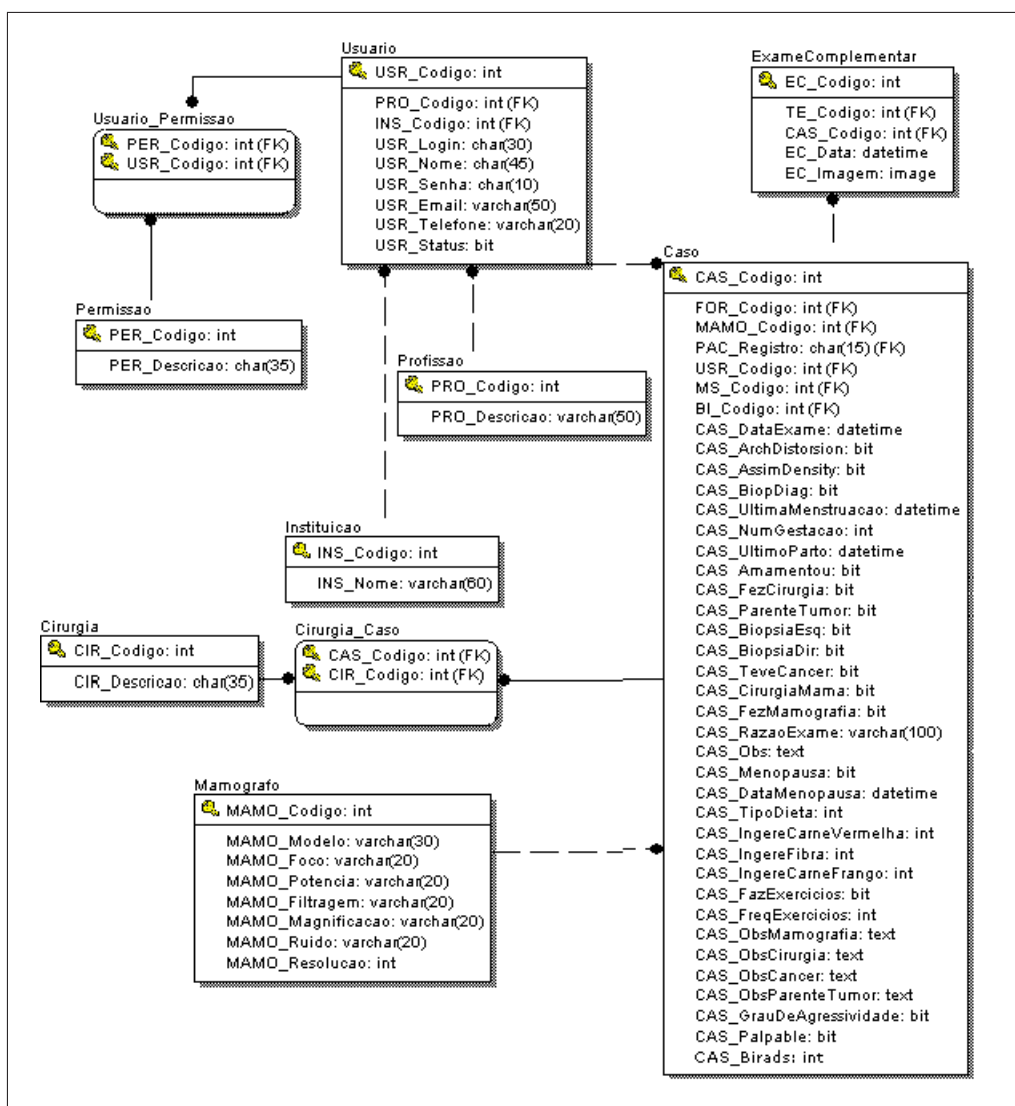


Figura 5.1: Modelagem da base de dados mamográfica do AMDI (Parte 1).

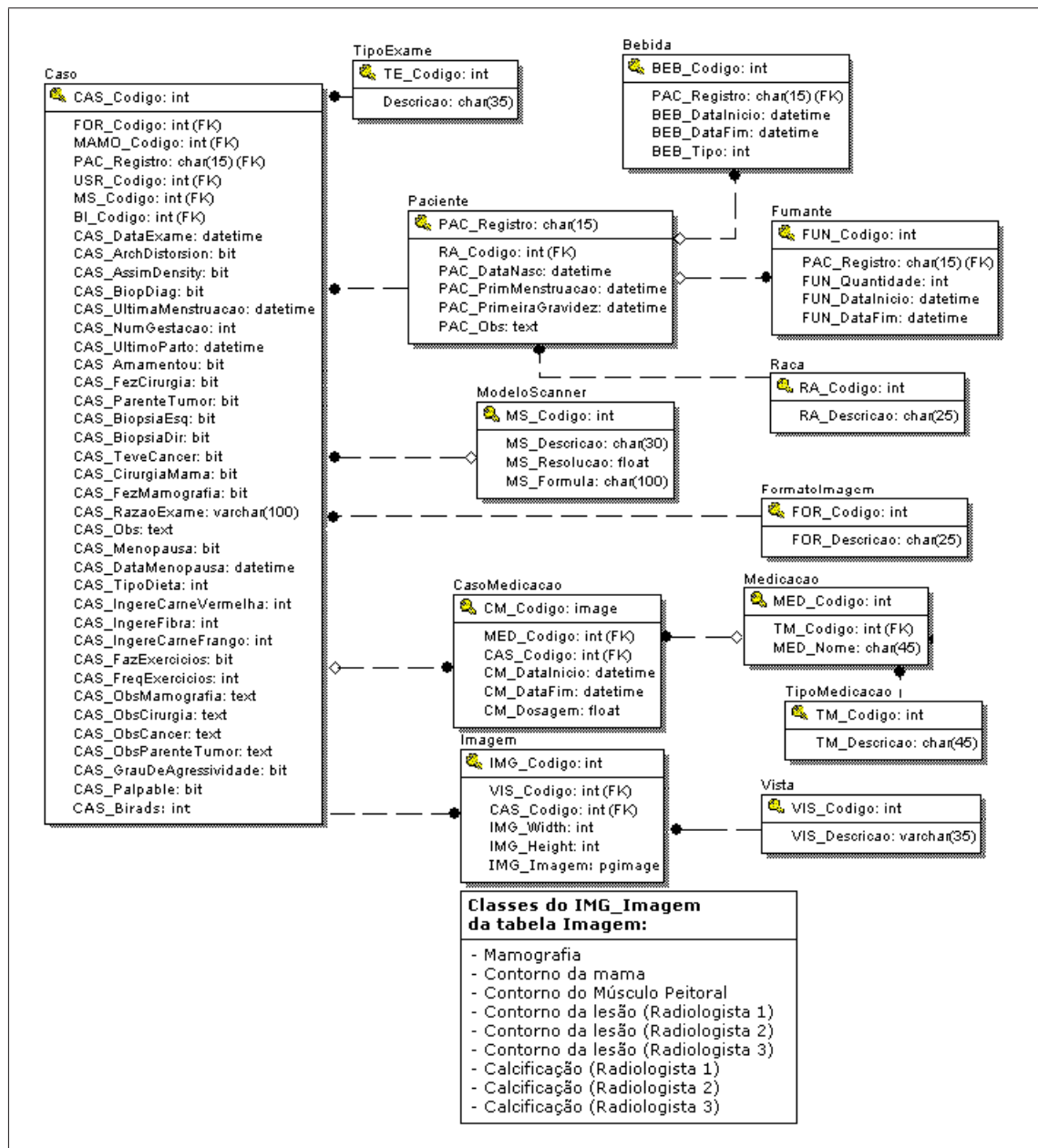


Figura 5.2: Modelagem da base de dados mamográfica do AMDI (Parte 2).

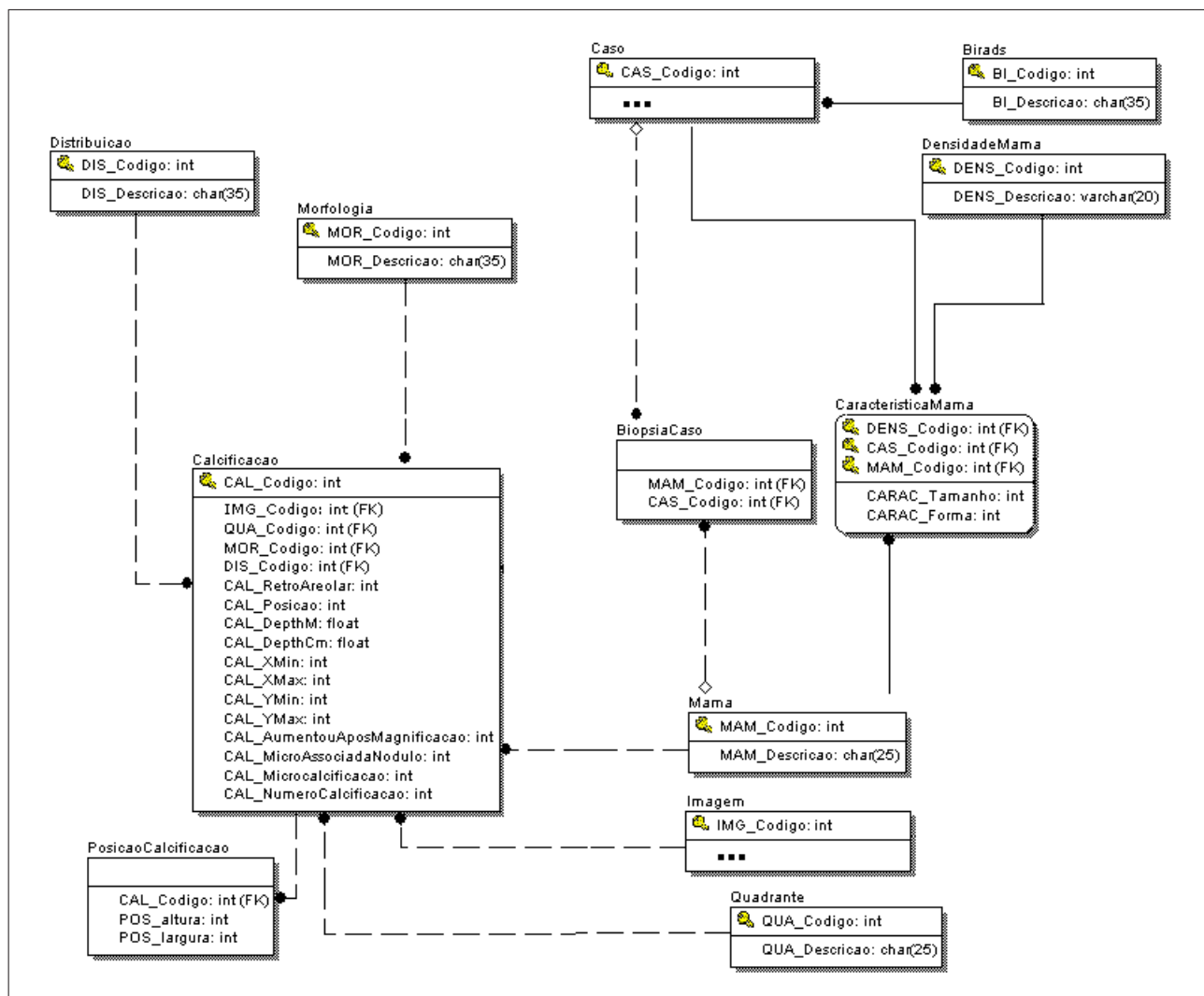


Figura 5.3: Modelagem da base de dados mamográfica do AMDI (Parte 3).

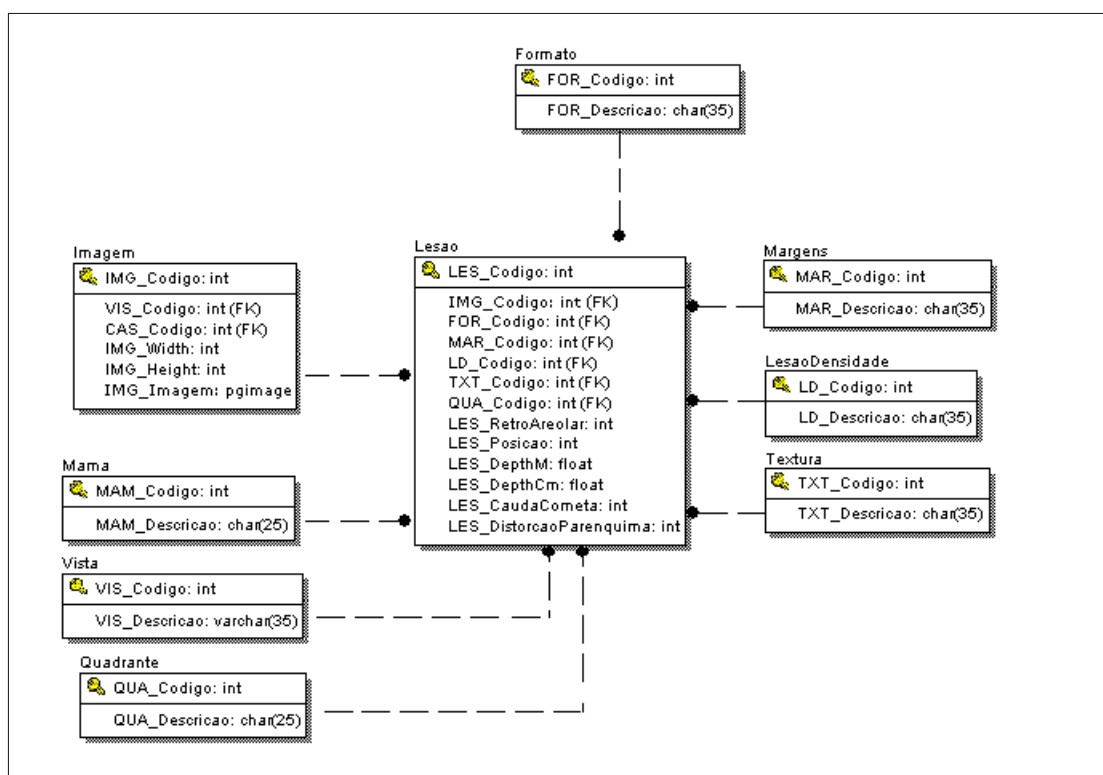


Figura 5.4: Modelagem da base de dados mamográfica do AMDI (Parte 4).

```

CREATE DATABASE AMDI WITH TEMPLATE = extended_template1;

CREATE TABLE imagem (
    img_codigo integer NOT NULL,
    img_imagem pgimage,
    cas_codigo integer NOT NULL,
    vis_codigo integer NOT NULL,
    img_width integer,
    img_height integer,
    img_imagem pgimage,
    CHECK ( (img_imagem).classes in ('mammography','contour_breast',
        'contour_pectoral_muscle','contour_lesion1',
        'contour_lesion2','contour_lesion3','calcification1',
        'calcification2','calcification3') )
);

SELECT Create_Extractor('area','getArea','float','ExtShape');
SELECT Create_Extractor('perimeter','getPerimeter','float','ExtShape');
SELECT Create_Extractor('compactness','getCompactness','float','ExtShape');
SELECT Create_Extractor('fourier','getFourierDescriptor','float','ExtShape');
SELECT Create_Extractor('fc','getFractionalConcavity','float','ExtShape');
SELECT Create_Extractor('spiculation_index','getSIndex','float','ExtShape');

SELECT Define_Feature_Vector('Shape-Vector1','area:contour_lesion1;
    perimeter:contour_lesion1;compactness:contour_lesion1;
    fourier:contour_lesion1;fc:contour_lesion1;
    spiculation_index:lesion_contour1');
SELECT Define_Feature_Vector('SI-Vector','spiculation_index:contour_lesion1;
    spiculation_index:contour_lesion2;spiculation_index:contour_lesion3');
SELECT Define_Feature_Vector('CC-F-SI-Vector','compactness:contour_lesion1;
    compactness:contour_lesion2;compactness:contour_lesion3;
    fourier:contour_lesion1;fourier:contour_lesion2;
    fourier:contour_lesion3;spiculation_index:contour_lesion1;
    spiculation_index:contour_lesion2;spiculation_index:contour_lesion3');

SELECT Set_Feature_Vector('Query1','imagem','img_imagem','Shape-Vector1');
SELECT Set_Feature_Vector('QuerySI','imagem','img_imagem','SI-Vector');
SELECT Set_Feature_Vector('QueryCCFSI','imagem','img_imagem','CC-F-SI-Vector');

```

Figura 5.5: Alguns comandos SQL-IE para construção da base de dados mamográfica do AMDI.