UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Estendendo *Quadtrees* para Suporte ao Armazenamento e Recuperação de Dados Espaço-Temporais

Por

JOICE SELEME MOTA

Dissertação submetida à avaliação,
como requisito parcial para a obtenção do grau de Mestre
em Ciência da Computação

Prof. Dr. Cirano Iochpe
Orientador

Porto Alegre, dezembro de 2000.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Mota, Joice Seleme

Estendendo *Quadtrees* para suporte ao armazenamento e recuperação de dados espaço-temporais / por Joice Seleme Mota. – Porto Alegre: PPGC da UFRGS, 2000.

79p.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2000. Orientador: Iochpe, Cirano.

1. Sistemas de Informação Geográfica 2. *Quadtrees* 3. Dados Espaço-Temporais .I. Iochpe, Cirano. II. Título.

Sumário

Lista de Abreviaturas

ADT	Tipo Abstrato de Dados
CAD	Computer Aided Design
DETMA	Dados Espaço-Temporais em Formato Matricial
DML	Linguagem de Manipulação de Dados
MA	Módulo de Armazenamento
MGD	Módulo de Gerência de Dados
O2	Object 2
OQL	Object Query Language
SGBD	Sistema Gerenciador de Banco de Dados
SGBDOO	Sistema Gerenciador de Banco de Dados Orientado a Objetos
SIG	Sistema de Informação Geográfica
SQL	Structured Query Language
TEMPEST	Temporal Geographic Information System
VIS	Sistemas de Informações Visuais

Lista de Figuras

Lista de Tabelas

Resumo

Nos Sistemas de Informação Geográfica (SIG), os aspectos temporais são importantes, principalmente, para representar o histórico de dados georreferenciados. Vários modelos conceituais de dados para SIG propõem classes e operações que permitem representar os aspectos espaciais e temporais das aplicações. Porém, ao nível do modelo interno dos sistemas atuais, as estruturas de dados armazenam e manipulam somente os aspectos espaciais dos dados geográficos, não contemplando os aspectos espaço-temporais propostos nos modelos conceituais. O objetivo desse trabalho é estender estruturas de dados do tipo *quadtree* para suporte ao armazenamento e à recuperação de dados espaço-temporais.

Palavras-chave: Sistemas de Informação Geográfica, *Quadtrees*, Dados Espaço-Temporais

TITLE: "EXTEND QUADTREES FOR SUPPORTS THE STORAGE AND RETRIEVAL OF SPATIO-TEMPORAL DATA".

Abstract

Support of temporal aspects is an important issue for geographic applications that need both to maintain and query historical series of georeferenced spatial data. Although various spatio-temporal data models have been proposed in the literature lately, there exists no commercial product yet that supports temporal aspects of spatial data. One of the reasons for that is probably the lack of efficient spatio-temporal data structures at the physical level of existing spatial database management systems. This work proposes an extension of a *quadtree* data structure that supports the storage and retrieval of spatio-temporal data in raster format.

Keywords: Geographic Information Systems, Quadtrees, Spacio-Temporal Dates

1 Introdução

Uma das áreas que mais tem se desenvolvido, nos últimos anos, é a de aplicações geográficas, as quais armazenam e processam dados georreferenciados. Dados georreferenciados são dados referenciados em relação à superfície terrestre [LIS97]. As aplicações geográficas são implementadas com base em Sistemas de Informação Geográfica (SIGs).

Os SIGs são sistemas automatizados, usados para armazenar, analisar, manipular e visualizar dados geográficos, ou seja, dados que representam fenômenos geográficos, cuja localização em relação à superfície terrestre é uma característica inerente à informação e indispensável para analisá-la [FAR98]. SIGs representam, em computador, fenômenos geográficos da natureza e aqueles criados pelo homem (fenômenos antrópicos). Cada tipo específico de fenômeno geográfico pode ser representado de diversas formas. A maioria dos modelos conceituais os representam como entidades ou objetos ([LIS00], [FAR98], [PEU98], [BOT95], [CAR93], [OLI97]). Um modelo conceitual permite representar como os objetos geográficos são e como se comportam perante a realidade que está sendo modelada, independentemente do sistema de gerência do banco de dados que irá armazená-los.

Os objetos geográficos apresentam características espaciais (ex: geometria, localização), descritivas (ex: atributos escalares) e temporais. Estas últimas estão relacionadas, usualmente, ao histórico de estados assumidos pelos objetos ao longo do tempo. Alguns dos modelos conceituais propostos para representar objetos geográficos já tratam da temporalidade dos dados, ou seja, permitem que se modelem aspectos temporais dos objetos. A face da terra se altera, dia a dia, pela intervenção do homem ou de fenômenos da natureza, o que aumenta a complexidade dos processos de armazenamento e manipulação de dados geográficos. Tal complexidade deve ser tratada pelo sistema de gerência do banco de dados geográficos que armazena os objetos geográficos para posterior manipulação pelos demais módulos do SIG.

A maioria dos softwares de bancos de dados atuais armazena somente as características espaciais e descritivas dos objetos geográficos. Mais recentemente, novos estudos estão apresentando propostas para que se possa tratar, no banco de dados, os aspectos temporais de objetos geográficos ([BOT95], [FAR98]).

Como já indicado, acima, o tempo é importante, em um SIG, pela necessidade que este tipo de sistema possui de tratar o histórico das informações por ele armazenadas e manipuladas. Por exemplo, em um estudo sobre a vegetação presente em uma área, pode ser importante saber se as matas e florestas vem sendo preservadas ou desmatadas ao longo de um período específico de tempo. Para isso seria necessário ter armazenado a seqüência de estados pelos quais passou a vegetação da região ao longo do tempo. Cada um destes estados deveria, então, ser associado, no banco de dados, ao período de tempo em que foi válido.

O fato de já existirem modelos conceituais que suportam a modelagem de aspectos temporais dos objetos geográficos não soluciona, de todo, o problema. Persiste,

ainda, a necessidade de que tais aspectos possam ser tratados no nível de armazenamento e manipulação dos objetos geográficos, ou seja, no modelo físico do banco de dados geográficos e no nível de sua linguagem de manipulação de dados (DML).

Com base na modelagem conceitual, é desenvolvido o modelo lógico que expressa os dados na visão da interface de definição do software de SIG. Neste momento, deve-se considerar se o modelo lógico implementado pelo SIG suporta os aspectos temporais, o que facilitaria o mapeamento do modelo conceitual. Caso o modelo lógico não suporte a temporalidade dos dados geográficos, o tratamento dos aspectos temporais deve ficar sob a responsabilidade dos programas de aplicação.

Quando se trata do armazenamento dos dados, existem ainda as estruturas de dados oferecidas pelo modelo físico do banco de dados implementado. No caso de dados geográficos no formato *raster*, por exemplo, existem várias estruturas de armazenamento possíveis. Dentre as mais utilizadas, em SIGs, estão as *quadtrees* ([SAM89a], [SAM89b]), pela possibilidade de compactação no processo de armazenamento e manipulação de dados *raster* (ex: imagens de satélite).

O termo *quadtree* é utilizado para descrever uma família de estruturas de dados em forma de árvores quaternárias com propriedades comuns baseadas na decomposição recursiva de formas geométricas, as quais são representadas, matricialmente, a partir de seus conteúdos ([SAM89a], [SAM89b]).

As estruturas *quadtrees* decompõem dados espaciais representados por matrizes de pontos ou de células. Atualmente, as *quadtrees* armazenam, apenas, os aspectos espaciais de dados geográficos, não sendo capazes de armazenar suas características temporais.

Quando se deseja armazenar, utilizando *quadtrees*, os aspectos temporais de dados geográficos como, por exemplo, o histórico de imagens de uma determinada região, a imagem completa deve ser novamente armazenada a cada atualização.Além disso, a aplicação deve controlar a ordem relativa entre as versões da imagem, já que o sistema de banco de dados ignora qualquer possível relacionamento entre duas ou mais *quadtrees*.

O principal problema desta estratégia de projeto verifica-se na utilização do espaço de armazenamento do SIG. Ainda que espaço de memória esteja cada vez mais acessível, o espaço em disco ainda é um problema para bancos de dados geográficos e, até mesmo, a transferência de dados entre diferentes computadores pode ter seu tempo de resposta afetado. Além disso, o volume de dados armazenados por aplicações de SIGs é, usualmente, maior que em outras aplicações, mesmo sem o armazenamento das características temporais.

Em uma seqüência de cinco imagens de uma mesma área, por exemplo, mesmo que as diferenças entre elas sejam mínimas, como numa região onde cresceu vegetação, há a necessidade de se armazenar novamente toda a imagem, desperdiçando-se, assim,

espaço de memória e tempo de processamento na execução de operações de acesso aos dados.

Para que os SIGs, no futuro, suportem modelos de dados espaço-temporais é necessário que, entre outras coisas, seus modelos internos de dados contemplem estruturas de dados que permitam o armazenamento e a consulta a dados espaço-temporais. Atualmente, porém, não se conhece qualquer SIG comercial que implemente alguma estrutura de dados baseada em *quadtrees* com essas características.

O principal objetivo desse trabalho foi estender estruturas de dados espaciais do tipo *quadtrees* para armazenar e recuperar dados espaço-temporais em formato matricial (*raster*).

Para se chegar ao objetivo descrito acima, foram estabelecidas três metas principais.

A primeira foi o estudo de modelos conceituais para, se possível, identificar um conjunto genérico de operações espaço-temporais sobre dados *raster.* A partir daí, foi selecionado um subconjunto de operações básicas a serem adaptadas para *quadtrees.*

Foram estudados alguns modelos conceituais de dados temporais específicos para SIG ou não [MOT99], sendo que alguns deles já apresentam conceitos de modelagem de dados espaço-temporais. Cinco modelos foram identificados ([EDE96], [GOR93], [FAR98], [BOT95], [PEU98]) com relação a suas operações espaço-temporais comuns.

Na escolha do subconjunto de operações espaço-temporais para extensão de estruturas *quadtrees*, foram observados alguns critérios de seleção tais como: operações bem definidas; facilidade de implementação; disponibilidade de algoritmos adaptáveis na interface das *quadtrees;* e forma de utilização das operações.

A segunda meta foi a investigação de classes de estruturas *quadtrees* existentes, para escolha da mais adequada para a extensão. Na escolha da classe de *quadtree* também foram observados alguns critérios de seleção, tais como: estruturas já implementadas em sistemas comerciais; adequação para extensão, facilidades de implementação e compactação.

A terceira meta foi o desenvolvimento de uma nova proposta para quadtrees espaço-temporais, proposta esta baseada no subconjunto de operações básicas escolhido e na classe de *quadtrees* a ser estendida.

O restante da dissertação está estruturado como segue. O segundo capítulo apresenta o estudo sobre os modelos conceituais, tendo como foco a comparação das suas operações, o conjunto genérico das mesmas, e o subconjunto escolhido para a extensão. No terceiro capítulo é apresentado o estudo das estruturas espaciais *quadtrees*. O quarto capítulo apresenta a proposta de extensão da classe de *quadtrees* escolhida e as alterações nos algoritmos que implementam suas operações básicas. O quinto capítulo apresenta as considerações finais sobre o trabalho desenvolvido. Em anexo, encontra-se

a descrição de um protótipo que foi implementado para avaliar o funcionamento dos módulos de armazenamento e consulta da nova classe de *quadtrees* espaço-temporais proposta.

2 Estudo de Modelos Conceituais

No início da pesquisa, estudou-se três modelos temporais para a compreensão de como a realidade espaço-temporal pode ser modelada. O objetivo deste estudo era identificar os tipos de dados e as operações que deveriam ser suportadas pelo modelo físico de um SIG temporal. Na ocasião foram estudados três modelos temporais ([EDE94], [GOR93], [FAR98]), sendo que um único deles [FAR98] específico para SIG. Procurou-se identificar um conjunto de operações relevantes e comuns aos modelos, para que a estrutura a ser desenvolvida tivesse condições de suportar tais operações. Essa idéia partiu do pressuposto de que a estrutura de armazenamento deveria oferecer, aos SIGs, tais operações em sua interface.

Para aprimorar os conhecimentos sobre operações espaço-temporais, foram estudados mais dois modelos específicos para SIGs, o modelo TEMPEST [PEU98] e o modelo de BOTELHO [BOT95], totalizando cinco modelos estudados.

Através desses estudos, identificou-se um conjunto genérico de operações, ou seja, um conjunto de operações comuns aos modelos.

O modelo TF-ORM ([EDE94], [EDE96]), não apresenta operações espaço-temporais por não se tratar de um modelo específico para SIG. Porém, seu estudo foi importante para a compreensão dos aspectos temporais incorporados em todos os outros modelos estudados. O modelo TIGUKAT [GOR93] apresenta, algumas operações temporais para SIGs, no qual foi concentrado o estudo. Os outros modelos ([FAR98], [BOT95], [PEU98]) são específicos para SIGs.

Todos eles apresentam operações descritivas, espaciais, temporais e espaço-temporais. Esse capítulo apresenta brevemente os modelos estudados, principalmente as operações espaço-temporais (quando existentes).

Ao final do capítulo, é apresentado o conjunto de operações espaço-temporais comuns aos modelos estudados.

2.1 O Modelo TF-ORM

O TF-ORM *(Temporal Functionality in Objects with Roles Model)* ([EDE94], [EDE96]) é um modelo de dados orientado a objetos. Utiliza o conceito de papéis *(roles)* para representar, separadamente, os diferentes comportamentos dos objetos de uma classe.

O conceito de papéis, associado à orientação a objetos foi introduzido através do modelo ORM *(Object with Roles Model)* [PER90]. A partir do modelo ORM foi feita uma primeira extensão, o modelo F-ORM (*Functionality in Objects with Roles Model)*[DEA91], no qual se procurou representar as funcionalidades de sistemas de informação de escritórios. Novas extensões foram realizadas para permitir a representação de aspectos temporais dando origem ao modelo TF-ORM [EDE96].

Nos modelos orientados a objetos, um objeto é criado como uma instância de uma classe, com um identificador próprio que o distingue dos demais objetos da classe. Uma vez criado, o objeto pertence a esta classe durante todo o seu tempo de vida. Devido a esse fator, um objeto não pode ser instância de mais de uma classe ou subclasse e nem migrar de uma classe para outra.

Essas limitações do modelo orientado a objetos podem ser resolvidas, utilizando-se o conceito de papéis. No conceito de papéis, um objeto continua a ser uma instância de uma única classe, mas poderá assumir ao longo de sua existência um ou mais papéis, isolada ou simultaneamente, em diferentes tempos. Deste modo é possível representar a evolução do objeto no tempo [EDE94].

Através desse conceito, uma classe pode apresentar diversos papéis, os quais podem ser instanciados dinamicamente, ou seja, em tempo de execução. Um objeto continua sendo instância de uma só classe, mas pode desempenhar diferentes papéis durante a sua existência[EDE96].

O Modelo TF-ORM é bitemporal, ou seja, o tempo de transação e o tempo de validade são utilizados como rótulos associados às propriedades que descrevem as instâncias e às propriedades que variam com o tempo. O tempo de transação é implicitamente definido pelo Sistema Gerenciador de Banco de Dados (SGBD). O tempo de validade deve ser fornecido pelo usuário, representando o instante do início da validade no mundo real, de características de objetos armazenados no banco de dados.

Os aspectos temporais do modelo TF-ORM se caracterizam como segue:

- Ordem no tempo: Linear (linearmente ordenado);
- Variação Temporal: Discreta;
- Elemento temporal primitivo: Ponto no tempo;
- Instante no tempo: instante atual *(NOW)*;
- Granularidade temporal: minuto (definido como *chronon*);
- Intervalo Temporal: representado através de um par de pontos no tempo, os quais definem seus limites;
- Marcas no tempo: evento, intervalos, tempo de duração;

As propriedades dos objetos no modelo TF-ORM, são classificadas como estáticas ou dinâmicas [EDE96]. As propriedades estáticas não apresentam rótulos temporais. Uma propriedade dinâmica consiste de um conjunto de triplas (tipo de dado, tempo de transação, tempo de validade).

2.1.1 Operações Temporais

O modelo TF-ORM apresenta um conjunto de mensagens pré-definidas, que são utilizadas para a manipulação dos objetos e papéis. Essa manipulação constitui-se na criação de instâncias de objetos e papéis, término da existência de classes, suspensão e retomada de atividades de instâncias.

Uma classe possui um nome (C_n) e um conjunto de papéis (R_n). Cada papel é constituído de um nome e um conjunto de propriedades (Pi), estados (S_i), regras (Ru_i) e mensagens (M_i).

- Operação Criar Classes
  Dados de Entrada: classe = (C_n, R_0, R_1, R_2,.....R $_n$);
- Operação Criar Papéis
  Dados de Entrada: Ri = < Rn_i, P_i, S_i, M_i, Ru_i>,
- Operação Criar Instâncias
  Dados de Entrada: P_i (Estática) <tipo>
  P_i (Dinâmica) <tipo, transação, validade>

As operações de consulta são baseadas na linguagem SQL *(Structured Query Language)* [EDE94].

A forma geral de um comando de recuperação de dados SQL é a seguinte:

SELECT < cláusula de especificação>
FROM <cláusula de identificação>
WHERE <cláusula de busca>

A cláusula SELECT define a estrutura do resultado a ser obtido pela consulta. A cláusula FROM identifica os objetos sobre os quais será efetuada a consulta. A cláusula WHERE apresenta a condição que deve ser satisfeita pelos objetos buscados.

Para condições que avaliam predicados temporais o modelo TF-ORM utiliza a cláusula WHEN no lugar da cláusula WHERE. Os operadores lógicos temporais que podem ser utilizados com a cláusula WHEN são apresentados na tabela 2.1.

Tabela 2.1 - Operadores Lógicos Temporais

Operador	Semântica
sometime past A	A valeu em algum momento do passado
immediately past A	A valeu no momento imediatamente anterior
always past A	A valeu em todos os momentos do passado
sometime future A	A será válido em algum momento do futuro
immediately future A	A será válido no momento imediatamente seguinte
always future A	A será válido em todos os momentos do futuro
A *since* B	A valeu em todos os momentos desde que B valeu
A *until* B	A valeu em todos os momentos até que B valha
A *before* B	A valeu em algum momento anterior ao em que B valeu
A *after* B	A vale em algum momento depois que B valeu

As saídas para operações de consulta podem ser de: dados, temporal e mixta. Em uma consulta de saídas de dados, as informações apresentadas correspondem somente a dados. Uma consulta de saída temporal apresenta as informações temporais associadas aos dados. Em uma consulta de saída mixta, os valores de dados e valores temporais são apresentados simultaneamente.

2.2 Extensão temporal para o modelo TIGUKAT

O modelo de objetos TIGUKAT é orientado a objetos. Em [GOR93] é proposta uma extensão temporal para esse modelo. O modelo TIGUKAT é definido com uma semântica uniforme para objetos. O modelo é comportamental no sentido de que todos os acessos e manipulações dos objetos são restritos para a aplicação de procedimentos (funções) sobre objetos, e a uniformidade do modelo se deve ao fato de que todo objeto faz parte de uma classe que pertence a um tipo primitivo já estabelecido pelo modelo.

O objeto é o conceito fundamental do modelo TIGUKAT. Todo componente de informação, incluindo a sua semântica, é representado como um objeto. Desta maneira, até o nível mais básico, todos os elementos expressivos no modelo incorporam a semântica de objetos. As propriedades do objeto, os relacionamentos entre objetos e as operações sobre objetos são todos modelados de uma maneira uniforme através de funções aplicadas a objetos.

A base do modelo é construída com um número de objetos primitivos que incluem [GOR93]:

- tipos de dados: real, inteiro, string, caracter (T_atomic);
- tipos para definição e estruturação de objetos com informações em comum, incluindo as operações que podem ser efetuadas sobre eles (T_type);
- propriedades e métodos que compõem as características e o comportamento de objetos (T_behavior);

- funções para especificação da implementação dos métodos de vários tipos (T_function);
- classes para classificação automática de tipos objetos relacionados e coleção para suportar grupos heterogêneos de objetos definidos pelo usuário (Tcollection);

Em relação à extensão temporal, [GOR93] apresenta as seguintes características:

- Ordem no tempo: linear e ramificada;
- Variação temporal: contínua, discreta ou densa;
- Dimensão temporal: tempo de validade;
- Instante no tempo: intervalo temporal;
- Granularidade: definida pelo usuário;
- Marcas no tempo: instante, intervalos e duração;

2.2.1 Operações Temporais

Na inserção, toda a classe de objetos deve pertencer a um tipo primitivo, e cada objeto pertencente a uma classe pode ser agrupado com objetos de classes diferentes.

Os prefixos utilizados nas classes e objetos determinam qual é o seu tipo primitivo. Tipos de objetos iniciam com o prefixo T_, classe de objetos por C_, as propriedades estáticas por B_ e as propriedades temporais por BT_.

A criação de uma nova classe é definida pelo nome da classe e suas propriedades.

- Operação Criar Classes
  Dados de Entrada: C_Nome : B_propriedade: <tipo>
  BT_propriedade: <tipo>

O modelo TIGUKAT apresenta, em sua extensão, um conjunto de operações temporais para os tipos de objetos pré-definidos.

Algumas operações podem ser utilizadas em SIGs, tais como união, interseção, diferença e sobreposição (*overlap)* e se encontram na classe temporal T_interval apresentadas na tabela 2.2.

Tabela 2.2 - Operações temporais do modelo TIGUKAT.

Nome	Descrição	Entradas	Saídas
B_Overlap	Verifica se dois objetos se sobrepõem em algum período de tempo	objetos, intervalo temporal	Valor booleano
B_Union	Retorna o período quando dois objetos possuem o mesmo tempo de validade	objetos, intervalo temporal	Intervalo Temporal
B_Intersection	Retorna o período no tempo quando dois objetos possuem as mesmas propriedades	objetos	Intervalo Temporal
B_difference	Retorna a posição no tempo quando em dois objetos de entrada as propriedades do primeiro não pertencem ao segundo	objetos	Intervalo Temporal

A linguagem de recuperação do modelo TIGUKAT possui um conjunto de predicados e operações baseadas na álgebra relacional e sua estrutura baseia-se no SQL. Dessa forma é chamada de TQL (*TIGUKAT Query Language*) e sua consulta básica possui a seguinte sintaxe:

SELECT<lista de objetos>
INTO <nome da coleção>
FROM <lista de classes e propriedades>
WHERE <condição>

A cláusula SELECT retorna a lista de objetos ou o objeto que se deseja consultar. INTO determina em qual coleção encontram-se os objetos desejados (se houver). A cláusula FROM referencia as classes e as propriedades onde devem ser buscadas as informações. Na cláusula WHERE se estabelece a condição desejada para a consulta

As saídas das consultas no modelo TIGUKAT podem ser de dados, temporais ou mixtas.

2.3 Modelo *BOTELHO*

Com o objetivo de facilitar a criação e o desenvolvimento de SIG temporais, Botelho descreve um modelo geográfico para representar as informações espaço-temporais desses sistemas. As principais características do modelo são descritas abaixo [BOT95]:

- É baseado nos conceitos de orientação a objetos utilizados no desenvolvimento do banco de dados O2 [O2t93], e pode ser integrado em um Sistema Gerenciador de Banco de Dados Orientado a Objetos (SGBDOO), tornando-o capacitado a prover as classes básicas necessárias para a construção de um SIG temporal.

- Utiliza uma estrutura de abstração multi-nível como em [CAM94], provendo o isolamento entre as classes geográficas da aplicação e o nível físico dos dados.

- A estrutura de dados proposta para armazenar os dados do modelo geográfico é baseada na estratégia de [BRA94], onde há agregação de atributos temporais (*timestamps*) ao esquema das classes cuja variação no tempo deve ser armazenada. Esta estratégia facilita a conversão de bases de dados atemporais em temporais.

- A estrutura de armazenamento provê métodos espaço-temporais necessários para a efetivação de uma consulta espaço-temporal.

- A taxonomia de consultas espaço-temporais apresentada está fundamentada no conceito de dominância dimensional de Langran [LAN95], e divide uma consulta em componentes básicos: seleção, projeção e agrupamento.

O modelo proposto é semelhante ao modelo de [CAM94], porém estendido para incorporar a dimensão temporal.

Para o modelo Botelho [BOT95], um objeto é uma instância de classes, caracterizado pela sua estrutura, comportamento e estado, ou seja, seus atributos, métodos e valores a ele atribuídos. Como qualquer modelo orientado a objetos, implementa os conceitos de herança e composição para possibilitar a reutilização de objetos previamente definidos.

O modelo de Botelho segue a filosofia de abstração de dados multi-nível adotada por Camara [CAM94] que divide o processo de modelagem dos dados do SIG em quatro níveis distintos dispostos em uma hierarquia:

- O nível do mundo real representa as entidades reais a serem modeladas como, por exemplo, plantações, bacias hidrográficas, rios ou redes de serviços.

- O nível conceitual introduz ferramentas para formalizar a modelagem das entidades geográficas definidas no nível anterior. Neste nível definem-se quais características das entidades do mundo real deverão ser representadas no banco de dados.

- O nível de representação espaço-temporal trata dos detalhes pertinentes às possíveis representações geométricas e topológicas das propriedades espaciais das entidades geográficas. Esse nível cuida de questões como escala, precisão, projeção cartográfica e outras.

- O nível de implementação (físico) cobre as estruturas de dados e operações usadas nas representações, incluindo as estruturas de índice para agilizar o acesso a dados, como as *Rtrees* e *Quadtrees*.

No nível conceitual, o modelo de Botelho limita-se ao tratamento de objetos geográficos representados por geo-objetos, que são instâncias de classes de uma hierarquia cuja raiz é a classe GeoObjeto. Geo-objetos tem atributos não-espaciais e um atributo espacial (denominado Localização).

2.3.1 Operações Espaço-Temporais

Para a temporalização das operações, o modelo propõe a alteração do esquema de uma classe ou de parte dela para que suas instâncias se tornem capazes de armazenar a evolução temporal das entidades que elas modelam.

A temporalização se resume em estender uma classe com os atributos necessários para armazenar todos os valores válidos para o objeto em algum período de tempo. Cada atributo do objeto tem associado um atributo temporal para armazenar pares de valor e tempo de validade, ou seja, uma lista de intervalos de tempo, cada um associado ao respectivo valor válido no período.

Botelho define um conjunto de consultas geográficas espaço-temporais básicas, as quais podem utilizar combinações para formar as consultas mais complexas existentes na maioria dos sistemas geográficos. Uma consulta é dividida em três componentes: predicado, domínio e resultado.

- Predicado é a parte interrogativa da consulta. A função do predicado é restringir o domínio da consulta e selecionar o resultado.

- Domínio da consulta é o conjunto de dados sobre o qual será aplicado o predicado para se extrair o resultado. O domínio da consulta pode ser um conjunto de objetos geográficos ou um esquema de uma classe geográfica, podendo envolver o estado, restrições ou comportamento dessas informações.

- Resultado é a resposta dada conforme a aplicação de um predicado sobre um domínio. Uma consulta pode resultar em valores de atributos convencionais, valores de atributos espaciais, valores de atributos temporais ou em um conjunto de objetos geográficos (temporais ou não).

Existe uma série de operações ([DAV93],[CIF95]) aplicáveis aos modelos geográficos. Botelho estende a semântica dessas operações para incorporar a dimensão temporal.

O modelo de Botelho trata apenas do tempo de validade de consultas temporais, e cria uma classificação de consultas espaço-temporais. As consultas espaço-temporais buscam relações espaciais entre objetos ao longo do tempo.

As operações espaço-temporais citadas em [BOT95] são apresentadas na tabela 2.3.

Tabela 2.3 - Operações espaço-temporais do modelo Botelho

Nome	Descrição	Entradas	Saídas
Interseção	Identifica os objetos geométricos que intersectaram outro objeto em algum instante de tempo.	objetos, intervalo temporal	Valor Booleano
Inclusão	Determina se um objeto geométrico esteve contido em outro em um período de tempo	objetos, intervalo temporal	Valor booleano
Continência	Determina se um objeto geométrico conteve outro em um período de tempo	objetos, intervalo temporal	Valor booleano
Proximidade	Determina quais objetos geométricos estiveram próximos de outro em determinado período de tempo	objetos, intervalo temporal	Valores numéricos
Adjacência	Determina vizinhança entre dois objetos geométricos	objetos	Valor Booleano
União	Reúne vários objetos geométricos em um único objeto	objetos	Objeto
Sobreposição	Sobrepõe vários objetos geométricos formando um único objeto	objetos	Objeto
Área	Calcula a área de um objeto em um determinado período	objeto, intervalo temporal	Valor numérico
Perímetro	Calcula o perímetro de um objeto em um determinado período	objeto, intervalo temporal	Valor Numérico
Comprimento	Calcula o comprimento de um objeto linear em um determinado período;	objeto, intervalo temporal	Valor Numérico

2.4 Modelo Espaço-Temporal de FARIA

A proposta de um modelo espaço-temporal feita por Faria [FAR98], visa contemplar os aspectos temporais de interesse no contexto de SIGs, e foi desenvolvida a partir do modelo de Botelho descrito acima. Já a parte que envolve as operações e consultas permitidas pelo modelo, ou seja, a linguagem de recuperação de informações foi baseada no modelo TOODM [OLI93].

Segundo [FAR98], as consultas em SIGs podem ser divididas em: temáticas; espaciais; temporais; e espaço-temporais.

As consultas temáticas referem-se ao conteúdo das propriedades (atributos) não espaciais. As consultas espaciais podem ser de localização, métricas, de orientação, ou topológicas. As consultas temporais consideram relações temporais entre ocorrência de

fenômenos. As consultas epaço-temporais combinam características de consultas espaciais e temporais.

2.4.1 Operações Espaço-Temporais

O modelo de Faria trata apenas das variações temporais de dados espaciais. Um objeto com propriedades espaço-temporais é inserido como uma especialização da classe SpatioTempObject, que trata dos atributos temporalizados.

A temporalização consiste em anexar à classe os atributos necessários para armazenar toda a história do objeto e seus componentes.

As consultas espaço-temporais combinam métodos temporais e funções espaciais. Nesse tipo de consulta é adicionada a dimensão temporal aos operadores espaciais. A dimensão temporal no caso do modelo é apenas o tempo de validade da informação. As operações espaço-temporais podem ser localizacional-temporal, direcional-temporal, métrico_temporal, e topológico-temporal.

As operações localizacional-temporal retornam o componente espacial do objeto referenciado, ou seja, as informações associadas a sua representação espacial georreferenciada válidas no período de tempo estabelecido.

As operações direcionais-temporais verificam se existe um determinado relacionamento de orientação entre dois objetos em um instante de tempo determinado. A tabela 2.4 apresenta os operadores direcionais-temporais do modelo de Faria.

Tabela 2.4 - Operações Direcionais-Temporais do modelo Faria

Nome	**Descrição**	**Entrada**	**Saída**
Above	Verifica se um objeto está abaixo de outro num instante de tempo	objeto1, objeto2, intervalo temporal	Valor booleano
Below	Verifica se um objeto está acima de outro num instante de tempo	objeto1, objeto2, intervalo temporal	Valor booleano
North	Verifica se um objeto está ao norte de outro num instante de tempo	objeto1, objeto2, intervalo temporal	Valor booleano
South	Verifica se um objeto está ao sul de outro num instante de tempo	objeto1, objeto2, intervalo temporal	Valor booleano
East	Verifica se um objeto está ao leste de outro num instante de tempo	objeto1, objeto2, intervalo temporal	Valor booleano
West	Verifica se um objeto está a oeste de outro num instante de tempo	objeto1, objeto2, intervalo temporal	Valor booleano
Left	Verifica se um objeto está a esquerda de outro num instante de tempo	objeto1, objeto2, intervalo temporal	Valor booleano
Right	Verifica se um objeto está a direita de outro num instante de tempo	objeto1, objeto2, intervalo temporal	Valor booleano
Same_position	Verifica se um objeto faz vizinhança em qualquer posição a outro num instante de tempo	objeto1, objeto2, intervalo temporal	Valor booleano
Northeast	Verifica se um objeto está a nordeste de outro num instante de tempo	objeto1, objeto2, intervalo temporal	Valor booleano

As operações métricos-temporais calculam o valor escalar para um ou mais objetos em um determinado período de tempo. A tabela 2.5 apresenta os operadores métricos-temporais do modelo de Faria.

Tabela 2.5 - Operações métricas-temporais do modelo Faria

Nome	Descrição	Entradas	Saídas
Área	Calcula a área ocupada pelo objeto geométrico poligonal em um período de tempo	objeto, intervalo temporal	Valor Numérico
Lenght	Retorna o comprimento do objeto geométrico linear em um período de tempo	objeto, intervalo temporal	Valor Numérico
Perimeter	Retorna o perímetro do objeto geométrico poligonal em um período de tempo	objeto, Intervalo temporal	Valor Numérico
Distance	Calcula a distância entre dois objetos geométricos em um período de tempo	Objeto1, objeto2, Intervalo temporal	Valor Numérico

As operações topológicos-temporais retornam um valor lógico verdadeiro se há um determinado relacionamento dois objetos geométricos num determinado período de tempo. A tabela 2.6 apresenta os operadores topológicos-temporais.

Tabela 2.6 - Operações topológicas-temporais do modelo Faria

Nome	Descrição	Entradas	Saídas
Disjoint	Verifica se existem objetos geométricos que não se tocam em determinada área em um instante de tempo	objeto1, objeto2, intervalo temporal	Valor booleano
Touch	Verifica se há fronteira entre dois objetos geométricos em um período de tempo	objeto1, objeto2, intervalo temporal	Valor booleano
Overlap	Verifica se dois objetos geométricos se sobrepõem em um determinado instante	objeto1, objeto2, intervalo temporal	Valor booleano
Inside	Verifica se um objeto está no interior de outro objeto em um instante de tempo	objeto1, objeto2, intervalo temporal	Valor booleano
Cross	Verifica se há um cruzamento entre dois objetos geométricos em determinado instante	objeto1, objeto2, intervalo temporal	Valor booleano

A linguagem de recuperação utiliza a estrutura em OQL (Object Query Language), que é uma linguagem baseada em SQL voltada para a orientação a objetos. Na cláusula WHERE são colocados os métodos temporais.

2.5 Modelo TEMPEST – (*Temporal Geographic Information System*)

Na busca de um modelo que pudesse ser genérico, ou seja, independente das aplicações para Sistemas de Informação Geográfica (SIG), e conhecendo a necessidade de uma ferramenta para auxiliar na modelagem e manipulação de dados geográficos começam-se as pesquisas do Projeto *TEMPEST* [PEU98], um projeto de pesquisa em SIG preocupado em apresentar uma solução para SIG com operações e modelos básicos e comuns para todos os tipos de usuários de SIG.

O *TEMPEST* é um *framework* [PEU84] derivado de um conjunto de modelo de dados e operações independentes dos aplicativos de SIG que proporciona a base para uma moderna interface de SIG.

O principal objetivo do *TEMPEST* é fornecer um modelo independente de aplicativos, onde qualquer domínio de desenvolvimento de SIG possa ser modelado e suas informações manipuladas, por isso não oferece um modelo específico, porém apresenta operações básicas e uma linguagem de manipulação [QIA98A, QIA98b] que considera adequada para resolver a complexidade das consultas de SIG.

Para que se pudesse chegar a esse conjunto universal de operações básicas, a linguagem de manipulação, e ao protótipo levou em consideração alguns aspectos que serão descritos a seguir.

O modelo *TEMPEST* prevê a multi-representação de uma informação geográfica em banco de dados, levando em consideração os atributos dos objetos, de localização e aspectos temporais. Essa representação, "objeto-localização-tempo" é chamada de modelo *TRIAD*.

Para uma realidade genérica de SIG, o *TEMPEST* concentra-se no modelo operacional, ou seja, no modelo de dados e informação e suas operações para modelar esta realidade. As operações podem ser aplicadas sobre a base de dados, ou sobre imagens gráficas. Essa distinção não é bem clara para o usuário, porém a lógica de processamento para as operações é a mesma.

Tanto para a visualização de tarefas quanto para consultas em banco de dados é apresentado um conjunto de operações que podem ser algoritmizadas.

Para que se defina um conjunto de operações para serem utilizadas em consultas de bancos de dados e visualizações e composições de imagens deve existir um modelo de informações genérico e flexível, para que possa se adaptar a qualquer domínio de aplicação de SIG.

2.5.1 Operações Espaço-Temporais

As operações espaciais são estendidas para espaço-temporais através do modelo TRIAD, ou seja, cada objeto tem associado o seu tempo de validade e as operações espaço-temporais podem ser para identificar, por exemplo, em que momento na história uma determinada rua passou a cruzar todo o bairro.

No modelo TEMPEST percebe-se que para a realização de operações espaço-temporais, existe um módulo que carrega os objetos envolvidos na consulta na memória para depois realizar as operações que são gerenciadas pelo SIG. Isso significa que as operações espaço-temporais são realizadas na memória lógica, após a recuperação dos objetos envolvidos, ou seja, essas operações são realizadas e gerenciadas pelo SIG e não pela estrutura que armazena os objetos.

As operações espaço-temporais propostas por Peuquet [PEU98] são baseadas em conjuntos. Essa escolha partiu do princípio de que conjuntos são bem definidos por possuírem conceitos e construções matemáticas. Usando conjuntos pode-se representar restrições impostas por modelos relacionais sobre Sistemas Gerenciadores de Banco de Dados ou SIGs. Ao mesmo tempo conjuntos permitem resultados de visualizações geográficas através de suas operações.

As operações de conjuntos envolvem todos os elementos participantes do conjunto. As operações de União, Diferença e Interseção, são aplicadas entre dois conjuntos proporcionando um novo resultado e são apresentadas na tabela 2.7.

Tabela 2.7 - Operações espaço-temporais do modelo *TEMPEST*

Nome	Descrição	Entrada	Saída
União	Une dois objetos geométricos em sua representação em um instante de tempo	Objeto1, objeto2	objeto
Diferença	Distingue um objeto geométrico que entre dois objetos, pertence ao primeiro e não ao segundo	Objeto1, Objeto2	Objeto
Interseção	Apresenta um objeto geográfico que pertença a dois objetos ao mesmo tempo em um determinado período	Objeto1, Objeto2	
Select	Seleciona objetos que satisfaçam uma condição pré-estabelecida	Objetos	Objetos
Associate	Recupera pares de objetos geométricos que pertençam ao mesmo conjunto ou conjuntos diferentes	Objetos	Objetos
Signify	Determina como objetos geométricos podem ser exibidos, reportados ou visualizados;	Objetos	Objetos
Measure/Topology	Determinam o relacionamento espaço-temporal entre os elementos de um conjunto (Vizinhança, distância, direção)	Objeto1, Objeto2	Valor booleano, Valor Numérico
Transformation	Troca a configuração de um objeto geométrico em um determinado período;	Objeto	Objeto
Compose	Combina vários objetos geométricos em um único objeto num período de tempo;	Objeto1, Objeto2, $Objeto_n$	Objeto

2.6 Conjunto de Operações Genéricas

Percebeu-se uma peculiaridade entre os modelos que oferecem operações espaço-temporais. Na verdade são colocadas as características temporais nas operações espaciais, tornando assim possível a realização dessas operações de acordo com o tempo que foram válidas, ou de acordo com a marca de tempo associada a elas.

A Tabela 2.8 apresenta as operações e os modelos estudados.

Tabela 2.8 - Modelos X Operações.

	TF-ORM	TIGUKAT	BOTELHO	FARIA	TEMPEST
Inclusão	■	■	■	■	■
Atualização	■	■	■	■	■
Consultas	■	■	■	■	■
Exclusão	■	■	■	■	■
Sobreposição		■	■	■	
União		■	■		■
Interseção		■	■		■
Diferença		■			■
Inclusão			■	■	
Continência			■		
Proximidade			■		
Adjacência			■	■	■
Área			■	■	
Perímetro			■	■	
Comprimento			■	■	
Above				■	
Below				■	
North				■	
South				■	
East				■	
West				■	
Left				■	
Right				■	
Northeast				■	
Distância				■	■
Disjoint				■	
Cross				■	
Select					■
Associate					■
Signify					■
Transformation					■
Compose					■

Todos os modelos apresentam as operações básicas de inserção, atualização, consulta e exclusão, seja dos seus objetos ou geo-objetos.

Botelho [BOT95] apresenta as operações de interseção, inclusão, continência, proximidade e adjacência como topológicas. Cita ainda operações como área e distância como métricas e apresenta duas operações de conjuntos a união e a sobreposição.

Faria [FAR98] apresenta como topológicas as operações *disjoint, touch, cross, inside* e *overlap*. Como métricas apresenta as operações de área, comprimento, perímetro e distância.

Já o modelo TEMPEST [PEU98] apresenta como de conjuntos as operações de interseção, união, diferença, seleção e cita como métricas/topológicas as operações de vizinhança, conectividade, distância/direção.

Apesar das diferenças no tratamento das operações foram observadas operações comuns entre os modelos tais como: interseção, vizinhança, proximidade, distância e sobreposição.

O modelo TF-ORM não apresenta operações espaço-temporais. Uma comparação entre as operações temporais dos modelos TF-ORM, TIGUKAT e FARIA, bem como as operações temporais comuns entre eles é encontrada em [MOT99].

3 Estruturas de Armazenamento

Diversos Tipo Abstratos de Dados (ADTs) são propostos para a manipulação de dados espaciais em formato matricial, não só em SIGs, mas em outras áreas como, por exemplo, computação gráfica, CAD (*Computer-Aided-Design*), cartografia digital. A maior preocupação é o grande espaço de memória que os dados espaciais matriciais ocupam em seu armazenamento.

Os ADTs são compostos pelas estruturas de dados e um conjunto de operações oferecidas em sua interface.

Existem vários ADTs espaciais ([ORE82], [SAM89a], [SAM90]), porém a maioria delas trata apenas da representação espacial, não associando temporalidade a sua representação.

Para buscar um conjunto de ADTs a ser estudado, levou-se em consideração alguns critérios tais como:

- *utilização*: se as estruturas são bastante utilizadas para o armazenamento de dados matriciais especialmente em SIG;
- *implementação*: facilidade de implementação e algoritmos disponíveis para extensão;
- *estudos anteriores*: propostas de extensão para suporte a temporalidade que utilizam determinada estrutura espacial;

Dentre as estruturas utilizadas por SIG para o tratamento de dados matriciais e, de acordo com os critérios de seleção estabelecidos, optou-se por realizar o estudo sobre o tipo abstrato de dados composto pelas estruturas de dados espaciais do tipo *quadtrees* ([SAM89a], [SAM89b]).

3.1 ADTs *Quadtrees*

Atualmente, muitas das estruturas de dados, utilizadas para representar dados espaciais em formato matricial (*raster*), são hierárquicas, baseadas no princípio de decomposição recursiva. Dentre as estruturas utilizadas [SAM90], pode-se citar as quaternárias (q*uadtrees* de regiões e *quadtrees* pontuais).

Essas estruturas pertencem, cada qual, a uma classe de estruturas que apresentam propriedades comuns e são diferenciadas pelas seguintes características:

- o tipo de dado que representam;
- o princípio utilizado no processo de decomposição;
- a resolução (variável ou não) que suportam.

Os tipos de dados a serem representados são pontos, áreas e curvas, entre outros. A decomposição é realizada com divisões quaternárias, ou ainda gerenciada pela inclusão de dados.

A resolução (o número de vezes que o processo de decomposição é aplicado) pode ser fixada no início do processo ou ser gerenciada através das propriedades dos dados de entrada [SAM90].

A classificação das estruturas *quadtrees* é apresentada na Figura 3.1.

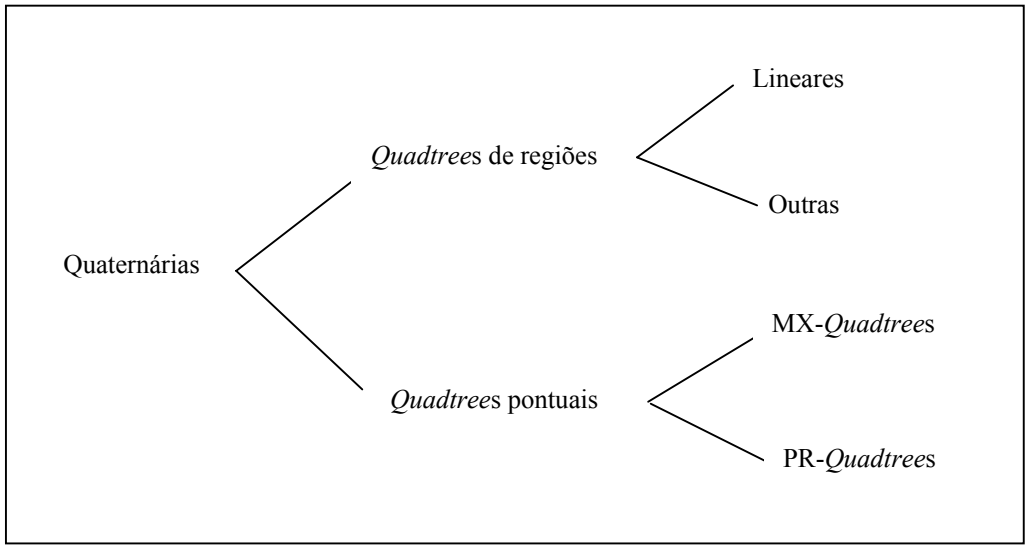


Figura 3.1- Classificação das estruturas *quadtrees*

As estruturas de regiões servem para representar dados do tipo região, enquanto que as pontuais representam o tipo de dado ponto.

Cada classe de estruturas *quadtrees* possui um conjunto de operações para a sua manipulação, tais como: inserção, deleção e busca de dados espaciais.

As operações são implementadas através de algoritmos, porém cada algoritmo varia de acordo com a estrutura que implementa e qual o tipo de dado espacial que manipula. Por exemplo, o algoritmo de inserção de dados do tipo ponto para uma *quadtree* região é diferente do algoritmo de inserção de dados do tipo retângulo para uma *quadtree* região.

Existem consultas típicas a serem aplicadas aos dados e estruturas de representação espacial como, por exemplo, distância entre pontos, vizinhança entre regiões.

Geralmente as pesquisas são feitas recuperando-se as coordenadas dos pontos ou regiões desejadas, mas as árvores também podem armazenar valores a serem recuperados.

Existem algoritmos próprios para determinar adjacência e vizinhança entre regiões, distância entre pontos, união e interseção entre DEMs [COR90].

As operações de inclusão utilizam métodos de ordenação no seu armazenamento. Os métodos de ordenação são algoritmos de caminhamento na matriz que contém dos dados espaciais para gerar a *quadtree.*

O propósito é a otimização do processo de mapeamento no armazenamento de dados espaciais ([SAM84], [SAM89a]).

Goodchild e Grandfield [GOO83] discute uma série de métodos de ordenação apresentadas na Figura 3.2.

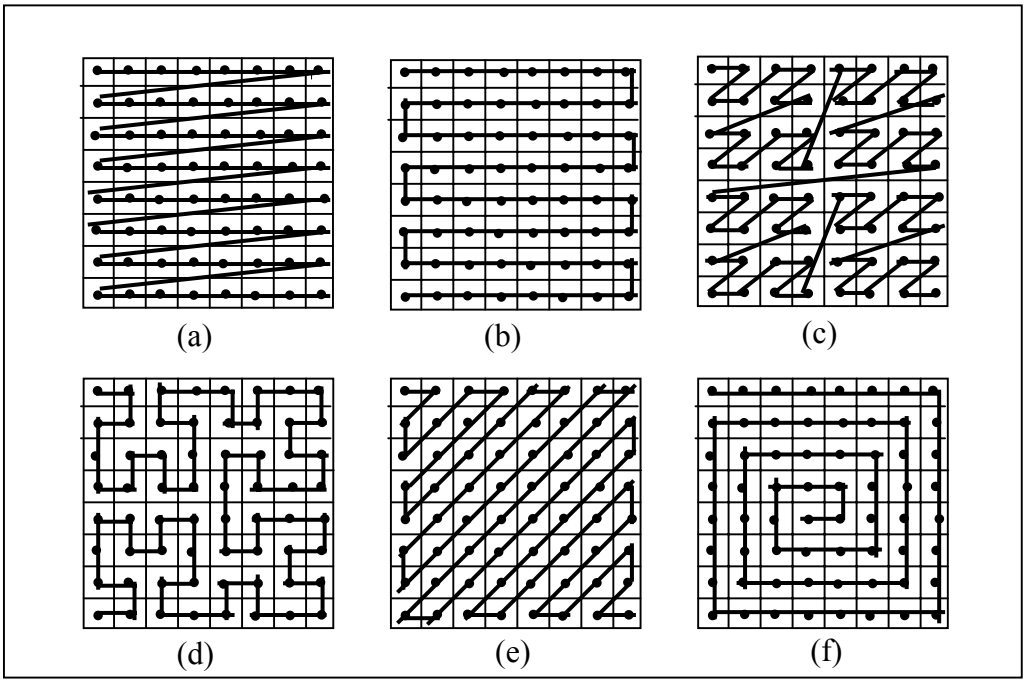


Figura 3.2 - Métodos de Ordenação

Cada método possui características distintas. O método *raster-scan* (3.2a) e *row-primer* (3.2b) são respectivamente similares à ordem de Morton [MOR66] (3.2c), e ao método de Peano-Hilbert [SAM89a] (3.2d).

Para os métodos de Morton e Peano-Hilbert é necessário conhecer o número máximo de valores das coordenadas dos dados espaciais sendo armazenados. A ordem de Morton é simétrica, diferentemente de Peano-Hilbert. Algumas vantagens da ordem de Morton [SAM89a] são que a posição de cada elemento pode ser determinada na intercalação das coordenadas x e y dos elementos, e que o processo para a recuperação das posições é fácil de ser especificado.

As outras ordens [SAM89a], Cantor-diagonal (3.2e) e ordem espiral (3.2f), são similares a *row-prime* e são de utilização interessante quando há dois pontos de origem na representação dos dados espaciais.

A ordem de Morton é a mais utilizada devido a simplicidade da conversão do processo de mapeamento de dados bi-dimensionais para estruturas de armazenamento unidimensionais.

3.2 Classes de *Quadtrees*

3.2.1 *Quadtree*s de Pontos

Nas *quadtree*s pontuais, para cada ponto de entrada, a área é subdividida em quatro quadrantes. A Figura 3.3 apresenta um exemplo de *quadtree* pontual. Quando se insere o primeiro ponto nas coordenadas específicas, subdivide-se a região de estudo em quatro quadrantes. Os outros pontos serão inseridos de acordo com os quadrantes já estabelecidos. Além disso, cada novo ponto inserido subdivide, em quatro novos quadrantes, o quadrante cuja área contém suas coordenadas.

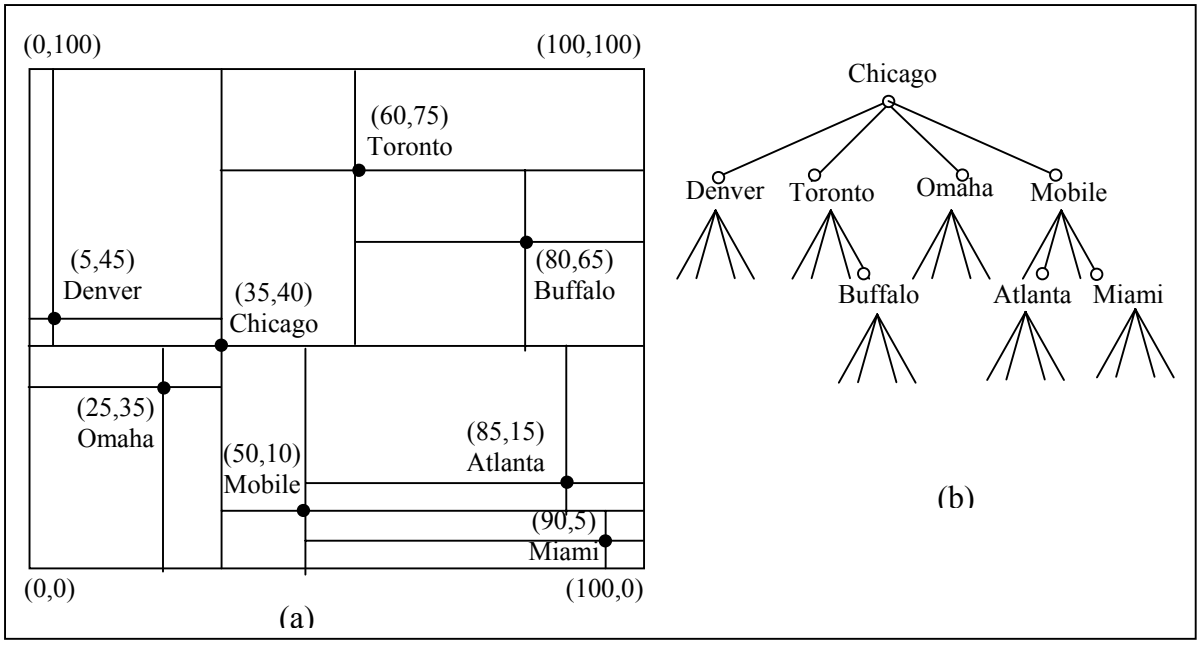


Figura 3.3 - *Quadtree* Pontual e sua representação.

A Figura 3.3 (a) apresenta um conjunto de dados espaciais do tipo ponto, com suas coordenadas específicas em uma região de coordenadas conhecidas. Essas coordenadas são especificadas para a geração da *quadtree*. A figura 3.3 (b) apresenta a árvore quaternária construída a partir de uma ordenação do conjunto de pontos de entrada.

A cada ponto inserido, subdivide-se um novo quadrante. O primeiro ponto inserido na árvore da figura 3.3 (b) é o nó raiz da árvore. Pelo método de ordenação de Morton, os quatro nós filhos de Chicago são Denver, Toronto, Omaha e Móbile.

Cada nova subdivisão compõe o próximo nível da árvore e seus respectivos nós filhos seguindo mesmo método de ordenação.

3.2.1.1 *Quadtree*s MX

Na *quadtree* MX os nodos "pretos", correspondem à presença do ponto na posição da matriz, enquanto que os nodos brancos correspondem a ausência do ponto na posição da matriz.

A Figura 3.4 ilustra uma *quadtree* MX para uma matriz de dimensões 2^3 x 2^3.

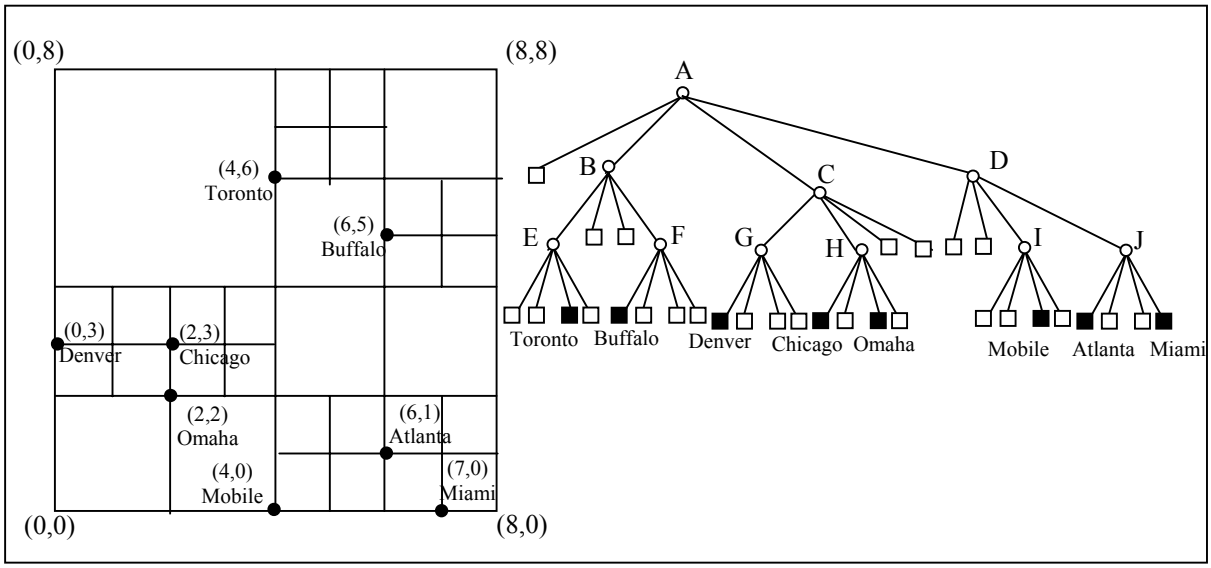


Figura 3.4 - Exemplo de *Quadtree* MX

A *quadtree* MX é adequada para representar um conjunto de pontos com domínio discreto e finito. Procurou-se adequar a representação para aproximá-la mais da representação de *quadtrees* de região.

Na representação em forma de árvore, a disposição dos pontos também obedece a ordenação de Morton [MOR66]. A área é dividida em quadrantes e os pontos incluídos pertencem a esses quadrantes que são utilizados para o desenvolvimento da árvore. Por exemplo, na primeira subdivisão percebe-se que o primeiro quadrante não contém pontos, sendo representado por um nodo "branco" na árvore. O segundo, terceiro e o quarto quadrantes são subdivididos por apresentarem novos pontos, dando, assim, mais níveis à árvore.

3.2.1.2 *Quadtree*s PR.

Nas *quadtree*s PR os conjuntos de dados do tipo ponto não necessitam ser discretos em seus quadrantes, ou seja, os quadrantes são divididos de forma regular, independente da localização dos dados tipo pontos. Os quadrantes podem conter dados do tipo ponto não exatamente nas coordenadas da subdivisão do quadrante.

Nas *quadtree*s PR os nós folhas ou estão vazios ou contém dados do tipo ponto. No exemplo da Figura 3.5 é apresentada a *quadtree* PR correspondente à *quadtree*

pontual da Figura 3.3 Uma próxima subdivisão em quadrantes só ocorre se o quadrante pai contém mais de um ponto.

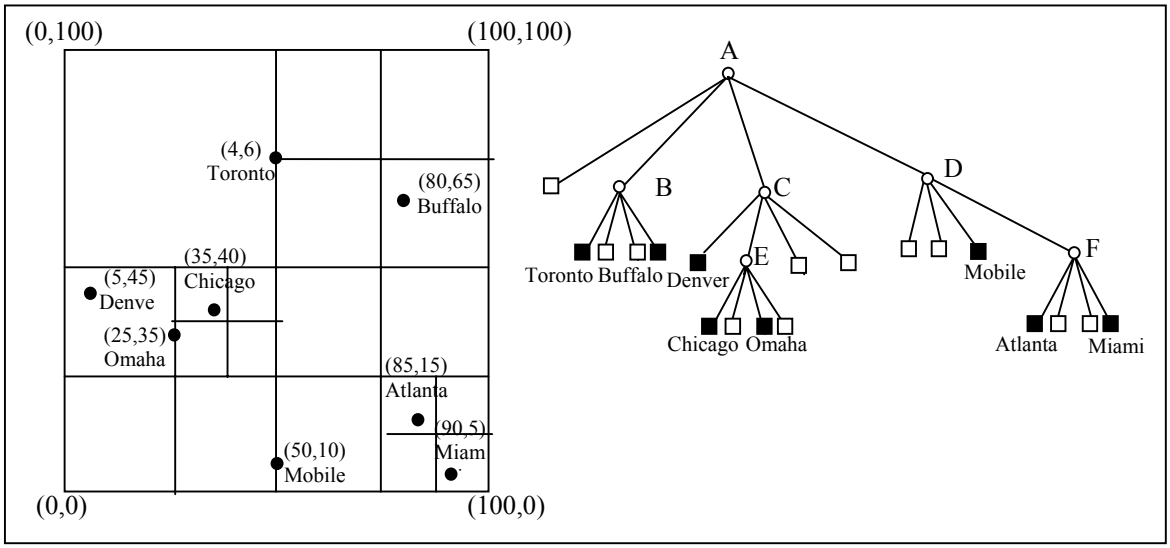


Figura 3.5 - Exemplo de *Quadtree* PR

Na representação do exemplo acima, também predomina a ordenação de Morton [MOR66] onde, para o nó raiz, o primeiro quadrante está vazio, o segundo contém dois pontos (Toronto e Buffalo), o terceiro contém três pontos (Denver, Chicago e Omaha) e o quarto contém três pontos (Mobile, Atlanta e Miami). A subdivisão é feita de acordo com a inserção dos pontos em seus respectivos quadrantes.

Percebe-se que a representação pontual não é a mais adequada quando se deseja trabalhar com dados espaciais matriciais, pois são próprias para trabalharem com um conjunto de pontos dispersos que não representam, em sua formação, toda a superfície de uma imagem.

3.2.2 *Quadtrees* de Região

As *quadtree*s de região [SAM90] são baseadas em subdivisões sucessivas de dados matriciais que podem representar imagens ou modelos numéricos quaisquer. Na representação da imagem, uma matriz pode conter 0s (fora de região de interesse) ou 1s (espaço ocupado por alguma região de interesse). A área que contém a imagem será subdividida até que cada subdivisão contenha, em sua totalidade, um conjunto de 1s ou 0s.

A *quadtree* de região possui resolução variável, uma vez que suas divisões podem possuir tamanhos diferentes.

A Figura 3.6 apresenta uma *quadtree* de região para uma matriz 2^3 x 2^3. Os espaços que contém 1s correspondem a elementos da imagem dentro da região de interesse. O processo de subdivisão produz uma árvore de grau 4 (onde a raiz e cada nó não folha possui quatro nós filhos).

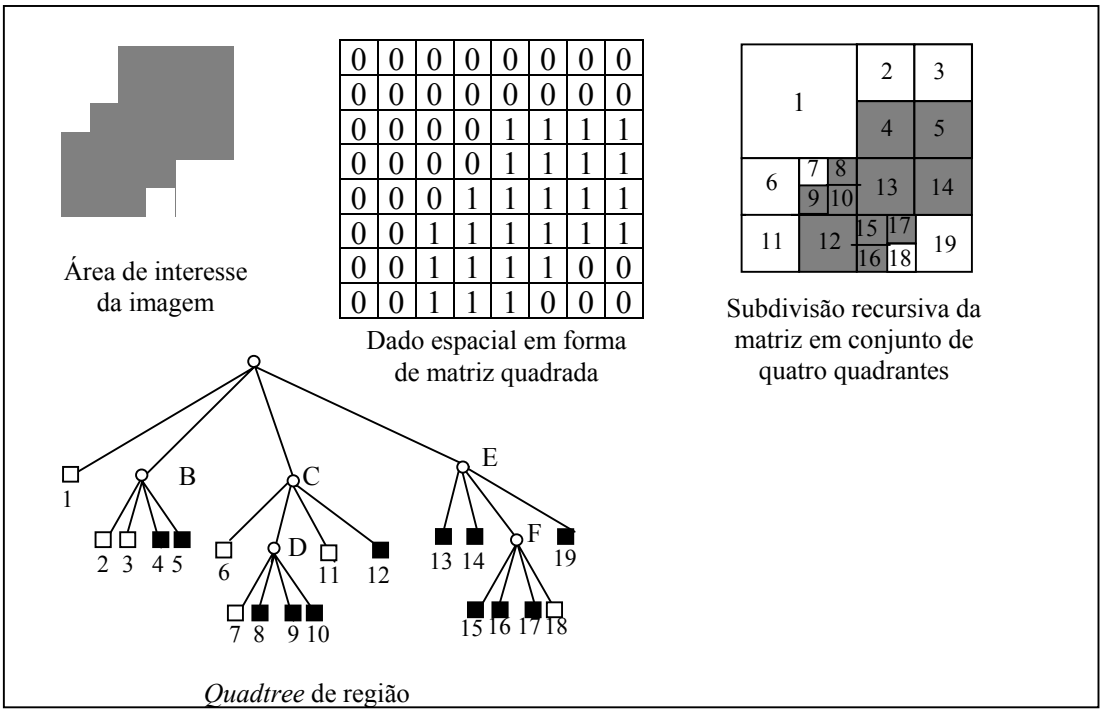


Figura 3.6 - *Quadtree* Região – Ordenação de Morton

Na representação em árvore, o nó raiz corresponde à matriz original. Cada nó filho corresponde a um quadrante da subdivisão que foi realizada na região do sucessor daquele nó. Os nós folhas representam os blocos preenchidos somente com 1s ou somente com 0s, o que significa que não é necessário mais subdivisões.

Cada nova subdivisão de um quadrante indica um nível a mais na árvore. Os nós folhas são marcados com números, enquanto os nós não folhas são marcados com letras.

São utilizados dois métodos de ordenação na construção de uma *quadtree* de região[SAM90]. O primeiro método é o de Morton [MOR66] quando o conteúdo útil da matriz não é extenso. O segundo método é aplicado quando as imagens são muito grandes. Nesse caso utiliza-se o método *raster-scan* como ilustra a figura 3.7.

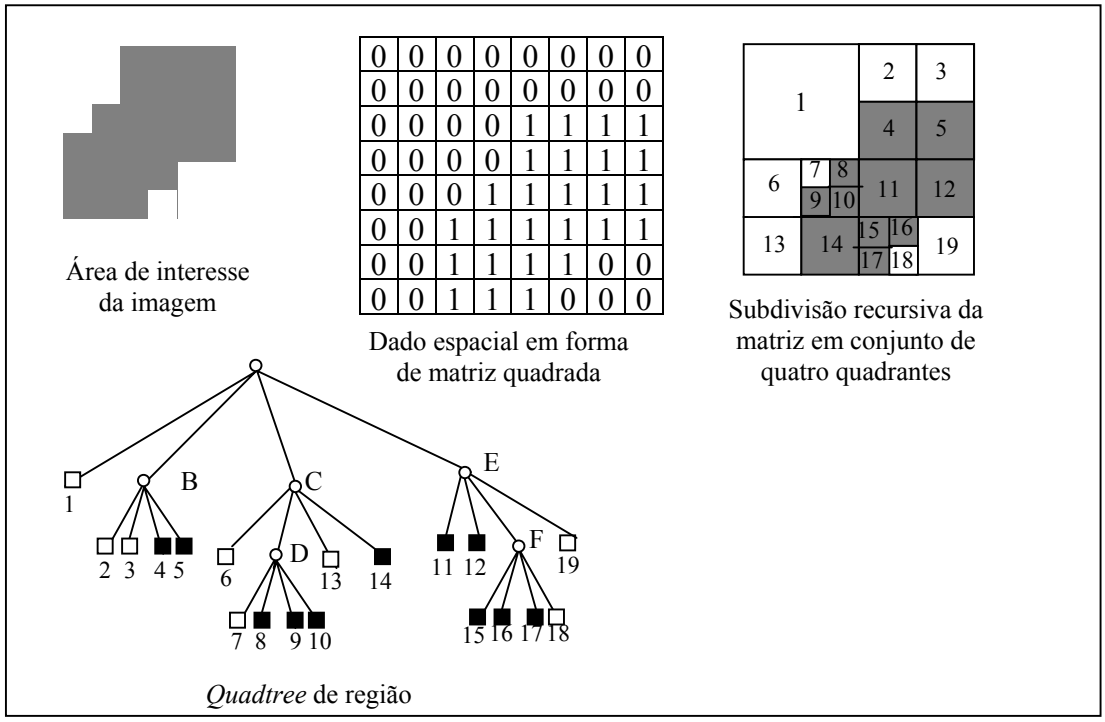


Figura 3.7 - *Quadtrees* região – Ordem *Raster Scan*

3.2.2.1 *Quadtrees* Lineares

Para o uso eficiente do espaço de armazenamento, ([GAR82a], [GAR82b]) propôs um método de representar *quadtrees* de região em uma estrutura linear, mantendo nela um código para cada nó ocupado. O código consiste de dígitos na base quatro que expressam o caminho da raiz até um nó localizado em uma subdivisão qualquer da matriz de origem. Essa estrutura apresenta as seguintes características [COR90]:

- somente os nós pretos são armazenados;
- a codificação inerente a cada nó incorpora propriedades de adjacência nas quatro direções e;
- o código atribuído a cada um dos nós especifica, implicitamente, o caminhamento desde a raiz até ele;

O processo para codificação dos pixels pretos de uma região contida em uma matriz quadrada de dimensão 2^n x 2^n é apresentada em [GAR82a] seguindo as seguintes convenções:

- Cada um dos quadrantes das sucessivas subdivisões é codificado com um dígito que o distingue dos demais, seguindo a ordem de Morton:

NW (noroeste)→ 0; NE (nordeste)→ 1; SW(sudoeste)→ 2;

SE (sudeste)→ 3;

Desta maneira, para cada um dos pixels pretos é atribuído um código quaternário ponderado, ou seja, um número composto dos dígitos 0, 1,2 e 3, na base 4, onde cada dígito sucessivo representa a subdivisão do quadrante do qual ele se originou como é mostrado na Figura 3.8.

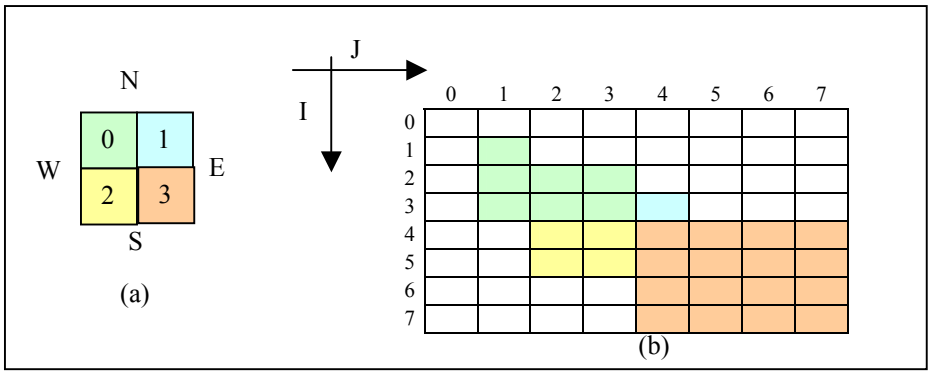


Figura 3.8 - Representação "*quadtree* linear" de uma região
(a) – ordem para identificação dos quadrantes
(b) – códigos quaternários, para cada pixel, com base na divisão recursiva da matriz em quadrantes.

O par de inteiros (I, J), com I, J = 0,1,....., $2^n - 1$, identifica a posição do pixel na matriz 2^n x 2^n. O processo de codificação consiste em mapear a região atribuindo, a cada par de coordenadas (I, J), um número inteiro K, na Base 4, que expresse os quadrantes sucessivos aos quais o pixel pertence. Por exemplo, se n = 3 e (I, J) = (6,5), o código será 321. Este código indica que o pixel (6,5) pertence ao quadrante SE da primeira subdivisão, ao quadrante SW da segunda divisão e ao quadrante NE da última subdivisão como mostra a Figura 3.8 acima.

Para se determinar o código K, torna-se necessário escrever o par (I, J) segundo as expressões abaixo, denotando uma representação binária:

$I = C_{n-1} * 2^{n-1} + C_{n-2} * 2^{n-2} + ...+ C_0$

$J = D_{n-1} * 2^{n-1} + D_{n-2} * 2^{n-2} + ...+ D_0$

onde:

n – indica a resolução, ou seja, o último grau da decomposição recursiva;

C_i e $D_i \subset <0,1>$ - estes valores indicam o quadrante ao qual pertence o par (I, J), para cada nível da subdivisão. A identificação é feita a partir da mais alta potência na representação binária (2^{n-1}), tendo-se por base as seguintes condições:

se $C_i = 0$ e $D_i = 0$, então $K_i = 0$;

$C_i = 0$ e $D_i = 1$, então $K_i = 1$;

$C_i = 1$ e $D_i = 0$, então $K_i = 2$;

$C_i = 1$ e $D_i = 1$, então $K_i = 3$;

Aplicando-se as fórmulas ao exemplo numérico mencionado, temos:

$I = 1*2^2 + 1*2^1 + 0*2^0 = 6$

$J = 1*2^2 + 0*2^1 + 1*2^0 = 5$

$K = (3 \quad 2 \quad 1)_4$

Ao término do processo inicial de atribuição dos códigos quaternários, tem-se uma lista desordenada, correspondente à codificação dos pixels "pretos". No passo seguinte, são efetuadas a ordenação e o posterior armazenamento em um vetor.

Um processo de compactação pode ser aplicado sempre que ocorrerem seqüências de pixels da mesma cor. Seus códigos são substituídos por um código constituído de n-1 dígitos quaternários, seguido por algum tipo de letra ou marca. No exemplo a seguir, denota-se por X:

310, 311, 312, 313.....31X

30X, 31X, 32X, 33X.....3XX

Para que se tenha como resultado da compactação um vetor ainda ordenado é necessário que a marca de compactação tenha valor de ordenação maior que 3 na seqüência de caracteres usada pelo sistema de computação. Este vetor ordenado constitui a *quadtree* linear.

Aplicando-se o processo de codificação na região apresentada na Figura 3.7, tem-se como resultado a seqüência:

< 003 021 023 03X 122 21X 3XX>

As principais vantagens da *quadtree* linear, em relação às demais *quadtrees* são:

- os requisitos de espaço de armazenamento e tempo de execução nos processos de recuperação dependem unicamente da quantidade de nós pretos;
- a eliminação da necessidade de armazenamento de ponteiros, de uso comum na implementação das estruturas em árvore dos demais tipos de *quadtree*;
- a facilidade de implementação da operação de compactação;

Por ser uma estrutura que não utiliza ponteiros, sua principal desvantagem é não permitir o acesso randômico aos dados espaciais matriciais, sendo que o processo de recuperação de parte de um dado espacial matricial pode se tornar muito lento por ser realizado seqüencialmente.

3.3 Uma proposta de *quadtrees* para dados espaço-temporais

Teodoros Tzouramanis [TZO98], utiliza *quadtrees* lineares para propor um método de acesso espaço-temporal a imagens *raster.*

Sua estrutura armazena imagens consecutivas levando em consideração o tempo de transação das imagens, quando do seu armazenamento.

A proposta é de que quando uma imagem sofra alguma alteração, possa-se armazenar apenas a diferença ocorrida e o tempo de transação desse armazenamento. Nesse caso utilizando essa estrutura, ter-se-ia o histórico das imagens, o momento em que essas alterações ocorreram no banco de dados, e uma considerável economia de espaço de armazenamento, uma vez que apenas as diferenças estariam sendo armazenadas a cada alteração.

A *quadtree* linear utilizada, chamada de FD *quadtree* linear [SAM90], é a variação que trata a estrutura *quadtree* como uma coleção de nós "folhas" ou terminais. Cada nó é representado por um par de números (altura e código de localização). O "código de localização" é representado através de um número fixo de bits.

No método utilizado por Tzouramanis [TZO98], imagens raster são convertidas para representações FD *quadtree* linear, e tem como resultado a lista com os nós que devem ser armazenados. Para cada nova sub-árvore é gerada uma nova lista.

A lista de códigos geradas na FD *quadtree* linear é armazenada em uma *B+Tree*. Essa nova estrutura, chamada de Sobreposição de *Quadtrees* Lineares [TZO98] é formada por sobreposições consecutivas de *B+Trees*.

A atualização de uma imagem é realizada em dois estágios. Primeiramente, é criada a lista com os códigos de localização da nova versão da imagem sobre a lista de códigos da última versão da imagem, que residem em uma tabela binária. O segundo estágio é a construção da nova instância da árvore.

Na realidade, a novidade da estrutura não está no nível da sobreposição de *quadtrees* lineares, mas na forma com que se trabalha com a sobreposição das árvores *B+Trees* pela extensão dos ponteiros com capacidade para trabalhar com o tempo de transação em modificações de imagens *raster*.

Essa estrutura apresenta algumas limitações, pois trabalha apenas com *pixels* pretos (representando imagem) e brancos (ausência de imagem) e com o tempo de transação. Em SIGs o tempo de validade da imagem é mais importante, pois se deve representar o mais próximo possível a realidade, e existe um intervalo de tempo entre a leitura das imagens e a sua inserção no banco de dados.

Outro ponto a ser considerado é o fato de que as estruturas de *quadtrees* lineares são utilizadas para a economia de memória, inclusive pelo fato de não utilizarem ponteiros. A não utilização de ponteiros também facilita a implementação,

pois operações baseadas em conjuntos, tais como união e intersecção são facilmente implementadas nesse tipo de estrutura.

Apesar de utilizar *quadtrees* lineares logicamente, a proposta de Teodoros [TZO98], retoma a complexidade da utilização de ponteiros, quando mapeia o seu sistema de *quadtrees* lineares para *B+Trees*, o que permite um questionamento em relação ao fato da não utilização da estrutura *quadtrees* com ponteiros, uma vez que o mapeamento seria direto.

A principal vantagem da estrutura proposta por Tzouramanis é a possibilidade de acesso randômico as quadtrees, permitindo a recuperação de partes das imagens *raster*

As desvantagens dessa estrutura são a complexidade de implementação dos algoritmos e a impossibilidade de compactação no armazenamento das imagens *raster*

4 Proposta de um ADT para Dados Espaço-Temporais de Formato Matricial

O ADT aqui apresentado tem por objetivo armazenar e manipular Dados Espaço-Temporais de Formato Matricial (DETMAs).

Através de operações básicas, armazenam-se os DETMAs originais e todas as suas atualizações futuras como uma seqüência temporal, onde cada novo DETMA contém somente as diferenças em relação aos anteriores.

Para a definição do ADT foi necessário desenvolver uma estrutura de dados e um conjunto de operações.

Dentre os ADTs do tipo *quadtree* apresentados no Capítulo 3 foi selecionada uma classe para o desenvolvimento da extensão. A seleção da classe foi realizada de acordo com critérios de correção e eficiência. O critério de correção refere-se ao fato da estrutura poder armazenar, manter e disponibilizar os dados corretamente. O critério de eficiência refere-se à economia de espaço de memória para a operação de armazenamento e ao tempo de resposta das operações de consulta.

As *quadtrees* lineares são vantajosas para a representação e manipulação de DETMAs devido à facilidade de implementação, clareza conceitual, possibilidade de compactação e ao não armazenamento das células vazias que compõem os DETMAS.

Na proposta em [TZO98], essas vantagens não são aproveitadas quando é realizado o mapeamento para B+Trees, devido a reutilização de ponteiros na estrutura.

O que se propõem é a extensão da estrutura de *quadtree*s lineares bem como de suas operações básicas para que se cumpram os seguintes objetivos:

- possibilidade de armazenar somente as diferenças entre estados de um dado capturado durante um período de tempo;
- associar rótulos temporais na própria *quadtree.*

O conjunto de operações a ser implementado partiu dos estudos realizados no Capítulo dois, onde se identificou operações espaço-temporais comuns a vários modelos conceituais de dados.

Percebeu-se que as operações básicas de manipulação de objetos tais como, inclusão, atualização, consulta e exclusão são comuns a todos os modelos e bastam para a manipulação de DETMAs, já que outras operações são implementadas pelos SIGs diretamente na memória principal do sistema.

A escolha das operações básicas se baseou nos seguintes fatores. A *quadtree* linear é utilizada para recuperar toda a imagem, uma vez que o acesso randômico a

partes específicas não é trivial. Os SIGs, quando realizam operações entre DETMAs, primeiro as recuperam, por inteiro, na memória principal para, depois, realizarem as operações espaço-temporais solicitadas. As operações básicas de manipulação dos modelos estudados atendem às necessidades dos SIGs, permitindo a recuperação, armazenamento e a atualização de DETMAs associados a seus respectivos tempos de validade.

O novo ADT proposto, é composto por uma lista de estruturas *quadtrees* lineares ordenadas por tempo de validade, e um conjunto de operações básicas para sua manipulação.

Como em [TZO98], a proposta aqui apresentada também armazena apenas as diferenças dos DETMAs que vão sendo inseridos nos banco de dados ao longo do tempo. Porém, a nova estrutura trabalha com tempo de validade e implementa as *quadtrees* lineares através de listas ordenadas, além de permitir DETMAs coloridos. Além disso, listas contíguas podem ser integradas, permitindo a compactação de DETMAs, o que aumenta a eficiência no que diz respeito à economia de memória.

Pretende-se, então, que esse ADT possa ser utilizado através de módulos que gerenciam a representação espaço-temporal de objetos geográficos em produtos de SIGs.

Propõe-se que cada DETMA seja mantido, ao longo do tempo, em um sistema de *quadtrees* lineares temporais. A primeira lista representa o estado inicial do dado inserido no banco de dados. A cada nova atualização do dado, uma próxima *quadtree* linear é criada, a qual armazena somente as diferenças do novo estado do dado em relação a seu estado anterior.

A cada instante, o estado mais atual do dado é representado pela seqüência temporal das listas do sistema. Se uma imagem for alterada, por exemplo, duas vezes após sua inserção no sistema, sua versão mais atual será representada pela *quadtree* linear que armazena sua versão original, mais a *quadtree* que armazena as diferenças entre sua versão original e sua primeira atualização, mais uma terceira *quadtree* linear que representa as diferenças da última atualização em relação a seqüência de listas mais antigas. A Figura 4.1 ilustra este exemplo.

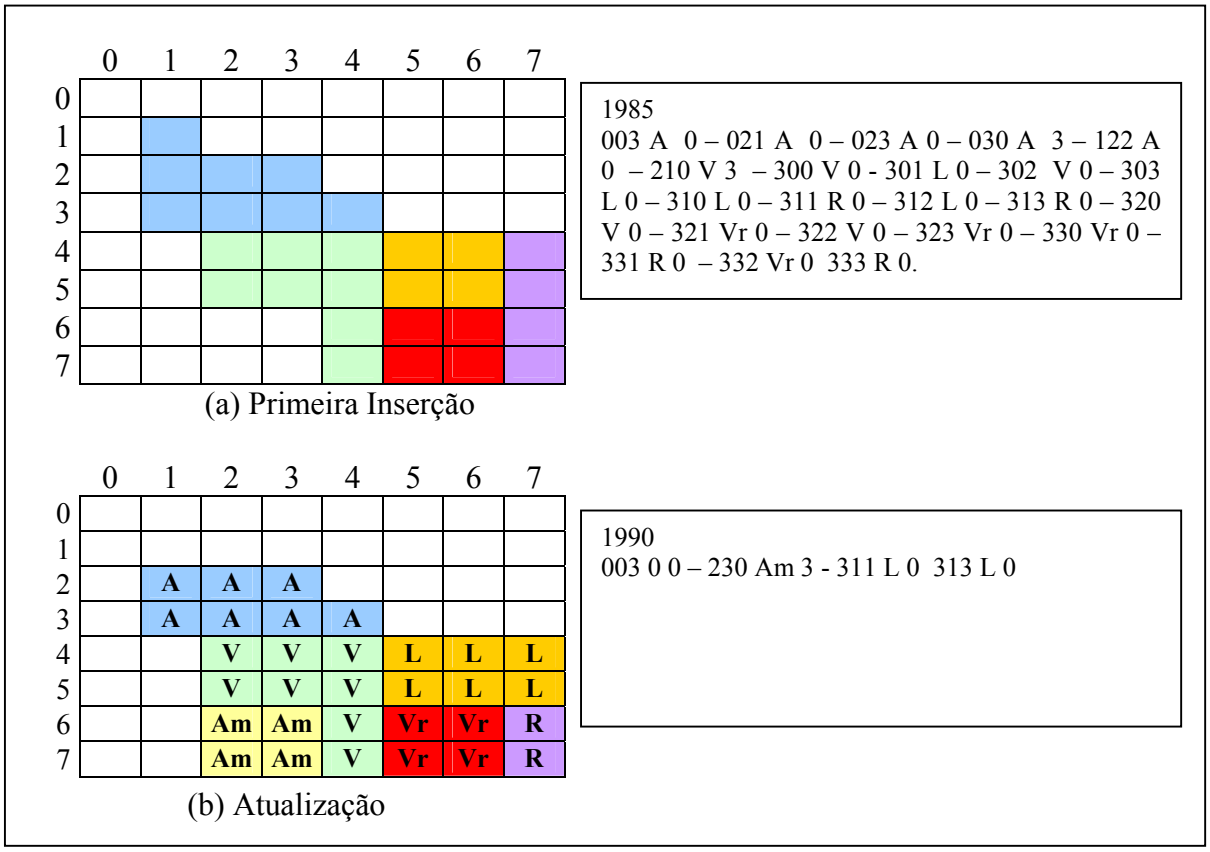


Figura 4.1 - Exemplo de Sistema de *Quadtrees* Lineares Temporais

4.1 Arquitetura do Sistema

O Sistema de *Quadtrees* Lineares Temporais é um tipo abstrato de dado com um conjunto de operações para SIG. A Figura 4.2 indica sua localização na arquitetura de software de um ambiente SIG.

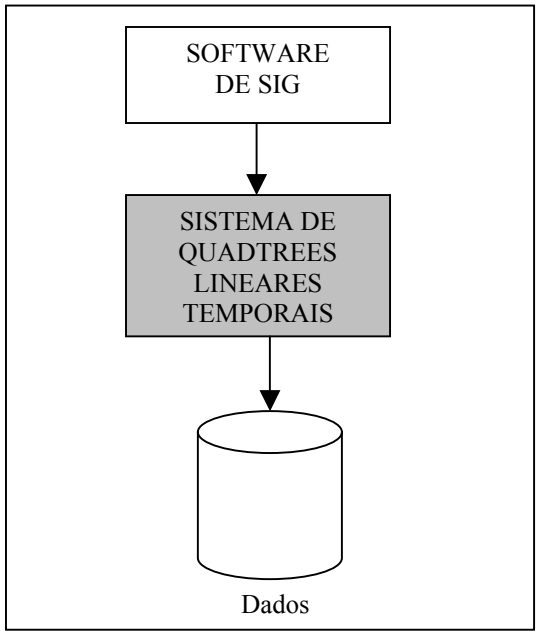


Figura 4.2 - Nível do Sistema de *Quadtrees* Lineares Temporais

O Sistema de *Quadtrees* Lineares Temporal para DETMAs apresenta um Módulo de Armazenamento (MA) que trata da forma como as *quadtrees* lineares são armazenadas pelo sistema, e um Módulo de Gerência dos Dados (MGD), o qual implementa as operações a serem solicitadas pelo SIG conforme apresentado na Figura 4.3.

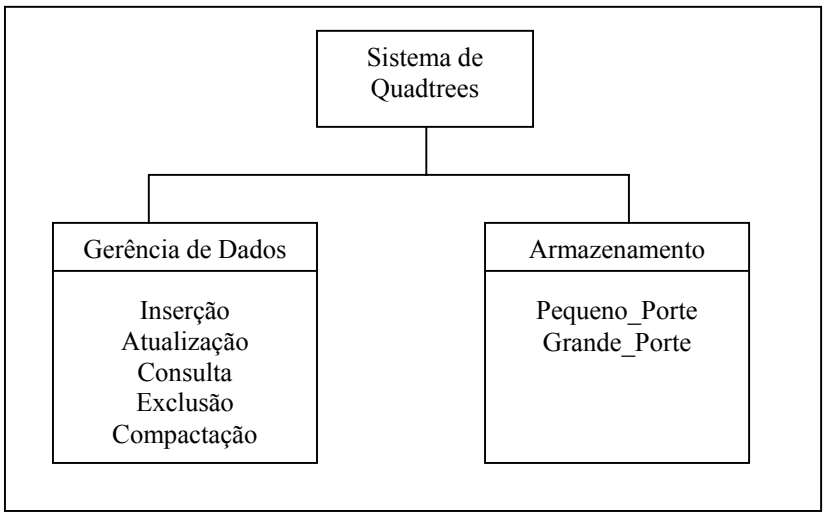


Figura 4.3 - Módulos do Sistema de *Quadtrees* Lineares Temporais

O armazenamento pode ocorrer de duas formas:

- um arquivo para armazenar a lista que representa a versão original do dado e as listas que representam o histórico de seus demais estados, ou seja, as listas com as diferenças,
- um arquivo para a lista que armazena os códigos referentes ao estado inicial, e um arquivo para cada atualização.

O Módulo de Gerência de Dados implementa as operações básicas de inserção, atualização, consulta e exclusão, bem como a operação de compactação do Sistema de *Quadtrees* Lineares Temporais. Essas operações são implementadas por seqüências de funções como apresenta a Figura 4.4.

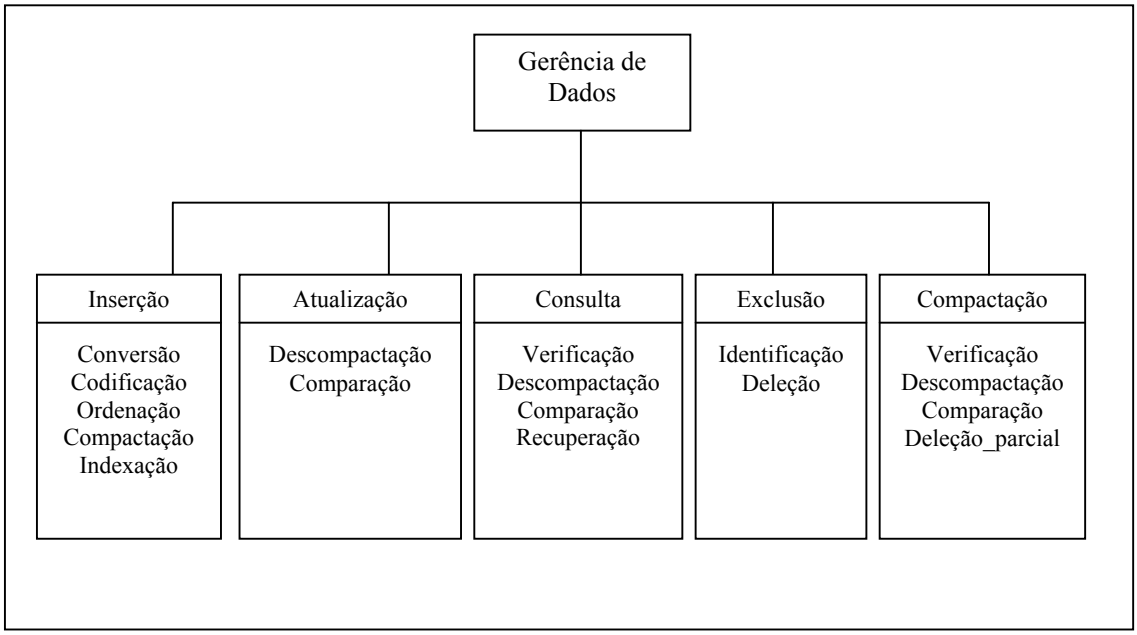


Figura 4.4 - Módulo de Gerência de Dados

4.2 Operações

A tabela 4.1 apresenta a descrição das operações do Sistema de *Quadtrees* Lineares Temporais, bem como as entradas e saídas das mesmas.

Tabela 4.1 - Operações do Sistema de *Quadtrees* Lineares Temporais

Operação	Descrição	Entradas	Saídas
Inserção	Possibilita a inserção do primeiro estado de um DETMA	Primeiro estado de DETMA: matriz 2^n X 2^n com tempo e validade (TV) {(coordenadas, cor)}	Lista Linear com TV {(código, cor))}
Atualização	Possibilita a atualização de um DETMA através da inserção de um novo estado	Novo estado de DETMA: matriz 2^n X 2^n com tempo e validade (TV) {(coordenadas, cor)}	Lista Linear com TV {(código, cor))}
Consulta	Possibilita a recuperação de um estado de um DETMA completo	Intervalo de Tempo	Vários, um ou nenhum estado completo de DETMA
Exclusão	Possibilita a exclusão de um DETMA	Identificador	
Compactação	Possibilita a integração de estados de um DETMA	Intervalo de Tempo	Novo Sistema de Listas Lineares Temporais

4.2.1 Operação de Inserção

As funções utilizadas na operação de inserção devem primeiramente gerar, a partir dos endereços da matriz que representa o estado inicial do dado, a *quadtree* linear. Uma *quadtree* linear é composta por códigos de localização, onde cada código corresponde ao endereço na matriz de uma determinada região. O sistema de *quadtrees* lineares deve armazenar em cada lista, o código de localização e a cor da região. Deve haver um índice, indicando o tempo de validade do estado representado na lista.

O tempo de validade pode possuir representação pontual, ou seja, cada data é representada como um ponto no tempo.

Parte-se do princípio que o dado de entrada está em uma matriz com seus respectivos endereços. Esses endereços são os da memória onde a imagem foi carregada e não as coordenadas em relação à superfície terrestre. Cada região, além do endereço da célula em relação às coordenadas da matriz, traz também as cores já codificadas.

As coordenadas reais dos vértices na matriz quadrada são mantidas pelo SIG em um índice que associa regiões geográficas a instâncias do Sistema de *Quadtrees* Lineares Temporais .

Os passos para o processo de inserção são os seguintes:

- receber a matriz contendo as coordenadas e cores;
- definir o número de cores para o trabalho;
- converter a matriz, a partir dos seus endereços, em uma *quadtree* linear representada por uma lista, contendo pares de códigos de localização e cor da região;
- ordenar a lista por código de localização;
- compactar a lista por cor;
- acrescentar data de validade à lista (pontual);

O Diagrama de Seqüências da Figura 4.5 ilustra a seqüência de passos da operação de inserção.

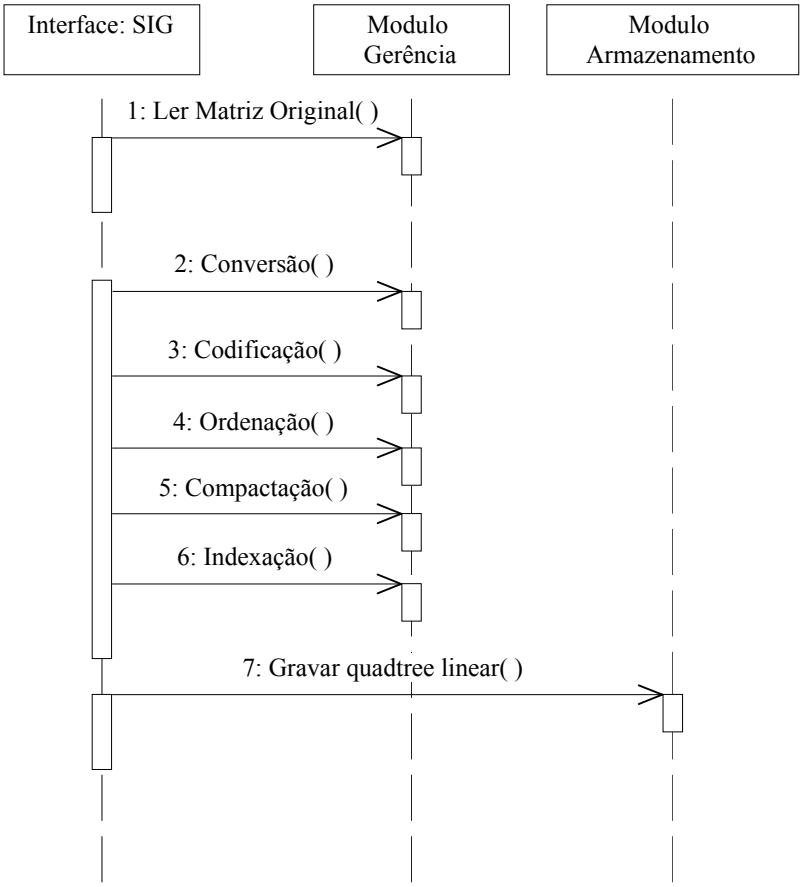


Figura 4.5 - Diagrama de Seqüências de Inserção

A operação de inserção cria um sistema com uma primeira *quadtree* linear para um novo DETMA. Os algoritmos foram desenvolvidos em português estruturado.

Para a primeira inserção são necessários cinco procedimentos:

a) Conversão: converte as coordenadas da matriz para números binários;

b) Codificação: gera os códigos de localização das listas lineares;

c) Ordenação: ordena a lista gerada;

d) Compactação: compacta a lista por cor;

e) Cria índice: acrescenta a data de validade da imagem a lista;

a) PROCEDIMENTO CONVERSÃO (N,GRAU, I, J)
{ Dados N (tamanho matriz), grau, I, J (coordenadas)}
Valor N = tamanho matriz inteiros;
A, grau, I, J: inteiros;
C, D: vetores de inteiros com tamanho grau (nível)
Início
A:= grau - 1;
enquanto I <> 0 faça

```
C[A] := I mod 2
I:= I div 2
A:= A-1;
fimenquanto
enquanto J <> 0 faça
C[A] := J mod 2
J:= J div 2
A:= A-1;
Fimenquanto;
```

Esse algoritmo transforma os endereços da matriz que representa o dado em números binários para posterior codificação da *quadtree* linear.

Cada coordenada é representada por um número binário que será passado como parâmetro para o procedimento que gera os códigos de localização. Por exemplo, uma imagem representando estado da vegetação de uma área chega em forma de matriz. As coordenadas dessa matriz que representa a imagem são transformadas para números binários e passados para codificação junto com a cor da vegetação situada em cada coordenada.

b) PROCEDIMENTO CODIFICA (n, c, d, k, cor)

```
Valor N = registros de inteiros
cor: parâmetro recebido da matriz original;
referências C, D, K = vetores de inteiros
I: inteiro;
{dados C, D, e o nível encontrar código quaternário 'K'};
Início
 Para I:= N-1 até 0 passo N-1 faça
        se C[I] = 0 e D[I] = 0 então K[I].codigo := 0;
        se C[I] = 0 e D[I] = 1 então K[I].codigo := 1;
        se C[I] = 1 e D[I] = 0 então K[I].codigo := 2;
        se C[I] = 1 e D[I] = 1 então K[I].codigo := 3;
        K[I].cor := cor;
Fim.
```

O procedimento de codificação, apresentado acima, gera o código de localização a partir da comparação dos endereços da matriz em binário.

Esse procedimento gera a *quadtree* linear com os códigos de localização e as respectivas cores. Após esse processo deve ser aplicado um processo de ordenação sobre a *quadtree* linear gerada.

Existem vários algoritmos de ordenação de listas como, por exemplo, linear, linear otimizado, bolha, *merge* e *heap* [LOP99]. Apesar de não possuir o melhor desempenho, foi utilizado o algoritmo Bolha por ser o que possui maior facilidade de implementação. Nada impede, contudo que este algoritmo de ordenação seja substituído por outro em uma implementação específica do Sistema de *Quadtrees* Lineares Temporais.

c) PROCEDIMENTO ORDENAÇÃO (Bolha)

```
        {Dados: lista a ser ordenada, N: tamanho da lista}
        i, j, auxcod, auxcor :  inteiro;
        Inicio
        J:= N;
        Enquanto j > 1 faça
          Para I:= 1 até j-1 faça
                Se lista[I].codigo > lista[I+1].codigo então
                  auxcod:= lista[I].codigo;
                  auxcor:= lista[I].cor;
                   lista[I].codigo := lista[I+1].codigo;
                   lista[I].cor := lista[I+1].cor;
                   lista[I+1].codigo := auxcod;
                   lista[I+!].cor := auxcor;
                FimSe
        FimPara
        j:=j-1;
FimEnquanto.
```

Após a ordenação da lista pelos códigos de localização é realizada a compactação da lista por cor de região. A compactação proporciona economia de memória, pois, se em uma subdivisão quaternária regiões contíguas apresentarem uma mesma cor, tais regiões podem ser representadas por apenas um código. Samet [SAM89a] propõem um algoritmo para compactação de *quadtrees* lineares apresentado a seguir.

d) PROCEDIMENTO COMPACTAR LISTA

```
        {Dados: lista a ser compactada}
        {Saída: lista compactada}

        I, J, N, proximo: inteiro;
        ok:boolean;
        Inicio
        n:=1;
        i:=1;
        ok:= falso;
        repita
           proximo:=i;
           enquanto não ok faça
                se ((q[i].codigo)+1 = q[i+1].codigo) .e. ((q[i].cor) = (q[i+1]).cor)  então
                    j:=j+1;
                senão
                    ok:= .verdadeiro.;
                i:=i+1;
           fimenquanto
           c[n].codigo:= q[proximo].codigo;
           c[n].cor:= q[proximo].cor;
           c[n].com:= j;
           n:=n+1;
```

```
    j:=0;
    ok:= false;
até i > 10;
```

O algoritmo de compactação compara as posições contíguas da lista, verificando se estas se encontram em seqüência e possuem a mesma cor. A lista é percorrida até que seja encontrado um código fora da seqüência ou com cor diferente. Na nova lista compactada gerada são armazenados o código, a cor, e o número que indica quantos códigos subseqüentes estão compactados.

Após a geração da *quadtree* linear ordenada e compactada cria-se um índice de acordo com a data de validade da imagem representada. A data de validade que aponta para lista indica o momento inicial que a mesma passou a valer para o Sistema de *Quadtrees* Lineares Temporais. A lista é válida desde a sua data de validade inicial até a data de validade inicial da próxima lista.

```
PROCEDIMENTO CRIAR INDICE (LISTA: lista, data_validade: data)
{Dado de entrada lista ordenada e compactada e data de validade}
Lista: lista;
Indice: ponteiro;
Inicio
   Indice.dado := data_validade;
   Indice.endereço := lista;
Fim;
```

Através desses algoritmos pode-se executar todos os passos necessários para a operação de inclusão do sistema de *quadtrees* lineares.

4.2.2 Operação de Atualização

Após a inserção de um primeiro estado do dado matricial no sistema de *quadtrees*, é permitida a atualização do dado através da inserção de novos estados. O novo estado deve ser armazenado com a sua respectiva data de início de validade. O Sistema de *Quadtrees* Lineares Temporais armazena, em uma nova *quadtree* linear, somente as diferenças entre o novo estado e os estados anteriores já armazenados.

Após o processo de conversão para *quadtree* linear, é feita a comparação da lista atual (a nova atualização) com todas as listas anteriores onde:

- se a região já existe nas versões anteriores, e possuir a mesma cor, não se armazena novamente;
- se a região já existe, porém mudou de cor, acrescenta-se na nova lista o mesmo código com a nova cor;
- se a região não existe nas listas anteriores, acrescenta-se a região na nova lista;

- se a região existe nas listas anteriores e deixou de existir, acrescenta-se o código da região na nova lista, com um registro de cor correspondente a deletado, por exemplo, -0;
- armazena-se a nova lista, contendo somente as diferenças do novo estado com a respectiva data de início de validade (pontual);

Se a atualização possuir data de validade inferior ao estado inicial do Sistema de *Quadtrees* Lineares Temporais, o processo de comparação é o mesmo, porém a lista de atualização que está entrando, deve ser considerar a lista com o estado inicial. Para prever essa situação um algoritmo verifica a validade da lista de atualização. Se for menor que o estado inicial, acrescenta a lista completa no sistema e chama o algoritmo de atualização passando como parâmetro as listas superiores uma de cada vez.

O Diagrama de Seqüências da operação de Atualização é apresentado na Figura 4.6.

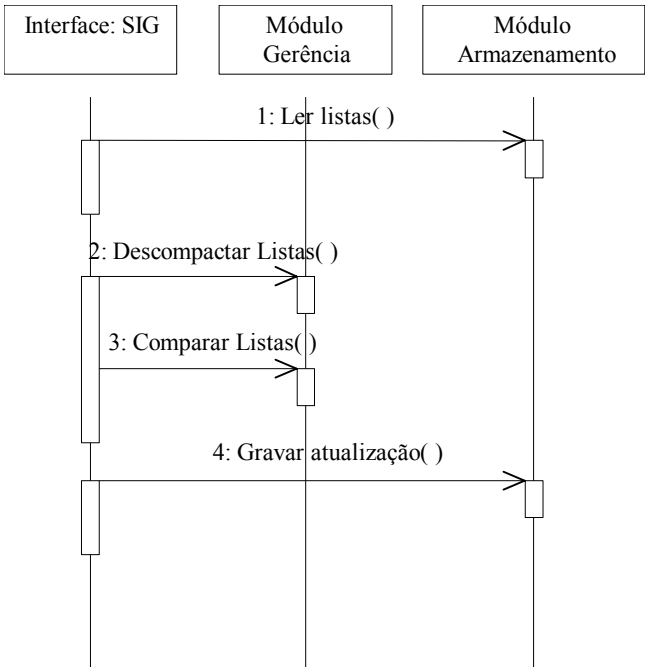


Figura 4.6 - Diagrama de Seqüências de Atualização

O algoritmo de atualização apresentado, a seguir, parte da entrada de duas listas a serem comparadas para gerar a diferença entre elas. Esse algoritmo compara as listas partindo da mais atual para a mais antiga. As listas são comparadas duas a duas.

Para identificar regiões deletadas, deve-se sempre comparar as listas já armazenadas com a lista inicial de entrada da atualização para que, ao final das comparações, seja gerada a nova diferença com todas as deleções contempladas.

```
        PROCEDIMENTO COMPARAR LISTAS (Atual, Anterior: lista; N, M: inteiro);
        {Dados listas Atual, Anterior, N (tamanho da atual), M (tamanho anterior}).
        Variáveis:
        Achou: lógico;
        Diferença: lista;
        A, I, J: inteiro;
        Início
        Achou:= false;
        A:=1;
        Para J:= 1 até N faça
        Início
           Para I:= 1 até M faça
           Início
           se Atual[J].codigo = Anterior[I].codigo então
              se (Atual[J].cor = Anterior[I].cor) então
                    achou:= true;
                    I:= M;
              senão
                    diferença[A].codigo := Atual[J].codigo;
                    diferença[A].cor := Atual[J].cor;
                    achou:= true;
                   I:=M;
                   A:= A +1;
              Fimse;
           Fimse;
           Se (I=M) e não(achou) então
             Diferença[A] := Atual[J];
             A:= A+1;
           Fimse;
          FimPara;
         achou:= false;
        FimPara;

{o bloco abaixo verifica a deleção sendo que as diferenças devem ser sempre verificadas com a lista original da atualização}
achou:= false
Para I:= 1 até M faça
  Para J:= 1 até N faça
  Inicio
      Se Anterior[I].codigo = Atual[J].codigo então
            J:= n;
      Senão
            Se Anterior[I].codigo < Atual[J].codigo então
              Diferença[A].codigo:= Anterior[I].codigo;
              Diferença[A].cor := -0; {representação de deleção}
      FimSe;
FimPara;
```

O algoritmo funciona da seguinte maneira:

- compara a primeira posição da lista que chegou para atualização com todas as posições da lista armazenada que apresenta maior tempo de validade ainda menor que o da lista de entrada;
- se encontrar código e cor iguais, passa para a próxima posição da lista que chegou para atualização, pois não há necessidade de armazenar essa região;
- se encontrar o código igual, porém a cor é diferente, armazena o código e a nova cor na lista que conterá a diferença e passa para a próxima posição;
- se percorrer toda a lista anterior e não encontrar o código de localização, acrescenta o seu código e cor na lista, pois trata-se de nova região;
- até esse momento a lista mantém-se ordenada;
- quando há a verificação de códigos deletados, deve-se verificar se os códigos da lista anterior existem na lista original de atualização;
- se não existirem, devem ser acrescentados na lista de diferenças, sendo gravada com o código para cor referente ao de deleção, ex: (-0);
- esse código é colocado ao fim da lista, tirando a ordenação da mesma, então é necessário aplicar o processo de ordenação para que a lista de diferença mantenha-se ordenada;

Esse processo de comparação deve se repetir até que todas as listas, com tempo de validade mais antigo que o da nova lista, tenham sido percorridos pelo algoritmo acima.

Foi desenvolvido um procedimento para verificar se a lista que está entrando realmente possui a data de validade mais atual, para que através dessa verificação passe como parâmetro as listas corretas.

```
PROCEDIMENTO VERIFICAR (Data_Validade, Sistema de Listas)

Anterior, Atual, Auxiliar: listas;
Inicio
   se data_validade (ATUAL) < data_validade (Anterior) então
        Auxiliar:= Anterior;
        Anterior:= Atual;
        Atual:= Anterior;
        Comparar_listas(atual, anterior);
Senão
     Comparar_listas (atual, anterior);
Fim.
```

O procedimento acima compara a lista de entrada com a de estado final do sistema de listas. Se a data for inferior, troca as listas e passa como parâmetro para o procedimento de comparação. Esse processo é realizado em cascata até a lista chegar no seu estado de acordo com a data de validade.

4.2.3 Operação de Consulta

O processo de consulta utiliza o tempo de validade como critério de seleção. O Sistema de *Quadtrees* Lineares Temporais devolve sempre a matriz completa, contendo o dado no estado válido para o tempo de validade indicado na consulta.

Supondo que se tivesse o seguinte sistema de *quadtrees* lineares:
1980 - 1.azul 2.verde 6.vermelho 10.roxo 12.preto
1985 - 2.marinho 3.cinza
1990 - 3.-0 4.abacate 9.preto 12.lilás
1995 - 3.-0 10.-0 15.amarelo

Se a consulta fosse: 'Recuperar o estado válido em 1990', o resultado seria o seguinte:
1.azul 2.marinho 3.-0 4.abacate 6.vermelho 9.preto 10.roxo 12.lilas

Se a consulta fosse: 'Recuperar o estado válido de 1985 a 1990', o resultado seria o seguinte:
1.azul 2.marinho 3.-0 4.abacate 6.vermelho 9.preto 10.roxo 12.lilas

Figura 4.7 - Condição de seleção baseada em tempo pontual e intervalo de tempo.

No exemplo da Figura 4.7, percebe-se que o resultado se repete, pelo fato de que o processo de consulta sempre deve recuperar o estado representado pelas listas das diferenças a partir da data solicitada, em cascata mantendo sempre os dados das listas mais atuais.

Os passos do algoritmo para processamento de consultas são apresentados abaixo:

- verificar no índice de datas, se a data existe:
- se existe lista com respectiva data, recuperar o estado associado;
- senão, recuperar estado a partir da data mais próxima inferior a data especificada;
- se a data for inferior ao estado inicial, notificar de estado não existente;

Para o processo de recuperação de estados, seguem-se os seguintes passos:

- partir da lista mais atual do conjunto de listas selecionado;
- recuperar os códigos das listas mais atuais acrescentando-os na lista resultado da consulta;
- buscar os códigos que ainda não aparecem no resultado da consulta, em cascata,a partir da segunda lista mais atual até a lista com menor tempo de validade;

A figura 4.8 mostra o diagrama de seqüências da operação de consulta.

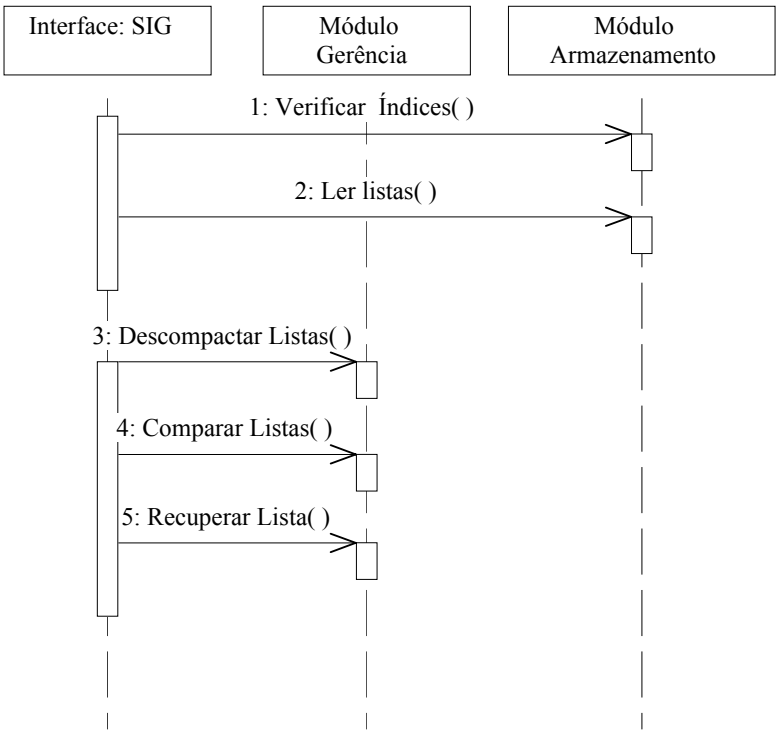


Figura 4.8 - Diagrama de Seqüências para Consultas

O algoritmo para consulta suporta consultas a dados matriciais por data de validade. Partindo da lista que é apontada pela data de validade, deve-se buscar os códigos de todas as regiões da matriz, mantendo sempre os códigos das listas mais atuais até chegar à lista de tempo menor de validade.

Primeiro, o algoritmo cria uma lista de recuperação que recebe toda a lista mais atual. Em seguida as listas anteriores são comparadas, uma a uma, com a lista de recuperação, na qual são acrescentadas somente as regiões que ainda não foram referenciadas.

PROCEDIMENTO CONSULTAR
{Dados de entrada lista mais atual e anterior de acordo com a data de validade estipulada para consulta}
Variaveis
achou: logica;
I, J, A, M, N: inteiro;
Anterior, Atual, Consulta: listas;
Inicio

```
        achou:= false;
        Para J:= 1 até N faça
         Inicio
           Consulta[J].codigo := Atual[J].codigo
           Consulta[J].cor := Atual[J].cor
         FimPara
        A:= N+1;
        Para I:= 1 até M faça
           Para J:= 1 até N faça
           Início
              Se Anterior[I].codigo = Atual[J].codigo então
                  achou:= true;
                   J:= N;
              senão
                 se Anterior[I].codigo < Atual[J].codigo então
                 Consulta[A].codigo := Anterior[I].codigo;
                 Consulta[A].cor :=  Anterior[I].cor;
                 A:= A+1;
                 achou:= true;
                 J:=N;
              Fimse
       FimPara
       Se não (achou) então
           Consulta[A].codigo := Anterior[I].codigo;
            Consulta[A].cor :=  Anterior[I].cor;
       FimSe
      achou:= false;
FimPara;
Procedimento Ordenar (Consulta: listas);
Fim;
```

Na primeira rotina, a lista mais atual é copiada para a lista que deverá recuperar o dado completo, uma vez que a consulta parte da lista com data maior e vai até a lista que representa o estado mais antigo do dado armazenado no Sistema de Q*uadtrees* Lineares Temporais.

A segunda rotina compara a lista anterior com a atual, caso um código de localização exista nas duas listas, permanece a versão do código da lista mais atual, sendo acrescentados os códigos da lista anterior que não são encontrados na lista de recuperação.

Após a comparação entre duas listas é chamado o processo de ordenação uma vez que a lista com o resultado da consulta não está ordenada.

Esse processo deve seguir, em cascata, até se chegar à lista original para a total recuperação da imagem sendo que, na próxima execução do algoritmo, as duas entradas serão sempre a lista de recuperação até o momento, e a próxima lista de data anterior do Sistema de *Quadtrees* Lineares Temporais.

O procedimento verificar_datas verifica se a data de consulta solicitada é inferior a lista que possui o estado inicial.

```
PROCEDIMENTO VERIFICAR_DATAS (Data_Consulta)
Início
se data_consulta < data_inicial então
   mensagem (‘Estado menor que o estado inicial do Sistema ’);
Fim.
```

4.2.4 Operação de Exclusão

Para o processo de exclusão, cujo diagrama de seqüências é apresentado na Figura 4.9, deve-se excluir todo o Sistema de *Quadtrees* Lineares Temporais.

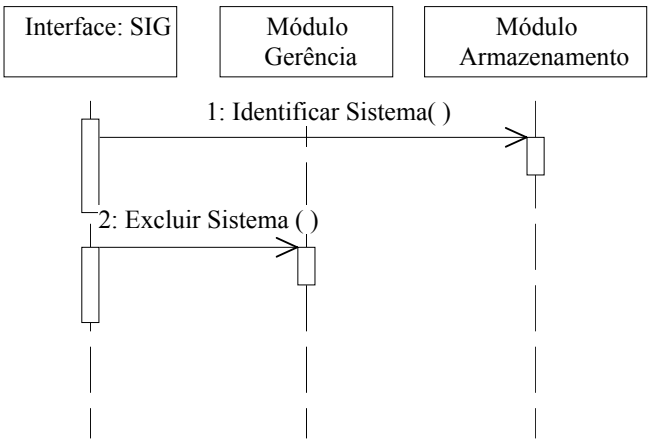


Figura 4.9 - Diagrama de Seqüência de Exclusão

O algoritmo de exclusão exclui todo o sistema de *quadtree* quando se deseja deletar o DETMA e, conseqüentemente, seu histórico.

Para que isso seja possível, existe um ponteiro para todo o índice de listas, ou seja, um ponteiro para todos os ponteiros das listas. Na chamada do procedimento para excluir todo o sistema, esse índice é que deve ser passado como parâmetro.

```
PROCEDIMENTO EXCLUIR SISTEMA_QUADTREES (identificador)
    {Entra-se com o identificador do Sistema  a ser excluído}
    Indice_Geral: ponteiro;
    Inicio
    Se indice_geral.dado = indice_geral então
    Sistema_quadtrees.delete;
    Fim.
```

4.2.5 Operação de Compactação

No processo de compactação de listas, transforma-se uma seqüência temporal de *quadtrees* lineares, pertencentes a um Sistema de *Quadtrees* Lineares Temporais, em uma única *quadtree* linear, cujo tempo de validade é igual ao da *quadtree* de maior tempo de validade presente na seqüência. Esta operação visa reduzir espaço, em disco, ocupado por um Sistema de *Quadtrees* Lineares Temporais. A Figura 4.10 ilustra o processo. Parte-se da lista com a maior data de validade até a lista com a menor data, comparando os códigos das listas e armazenando, na nova lista, sempre o código mais atual.

Supondo que se tivesse o seguinte sistema de *quadtrees* lineares:
1980 - 1.azul 2.verde 6.vermelho 10.roxo 12.preto
1985 - 2.marinho 3.cinza
1990 - 3.-0 4.abacate 9.preto 12.lilás
1995 - 3.-0 10.-0 12.amarelo

Se a operação de compactação fosse: 'Compactar as listas de 1995 a 1990', o resultado seria o
seguinte:

1995 - 3.-0 4.abacate 9.preto 10.-0 12.amarelo

O sistema ficaria:
1980 - 1.azul 2.verde 6.vermelho 10.roxo 12.preto
1985 - 2.marinho 3.cinza
1995 - 3.-0 4.abacate 9.preto 10.-0 12.lilás 15.amarelo

Figura 4.10 - Exemplo de compactação de listas

O Digrama de Seqüências das operações para compactação é ilustrado pela Figura 4.11.

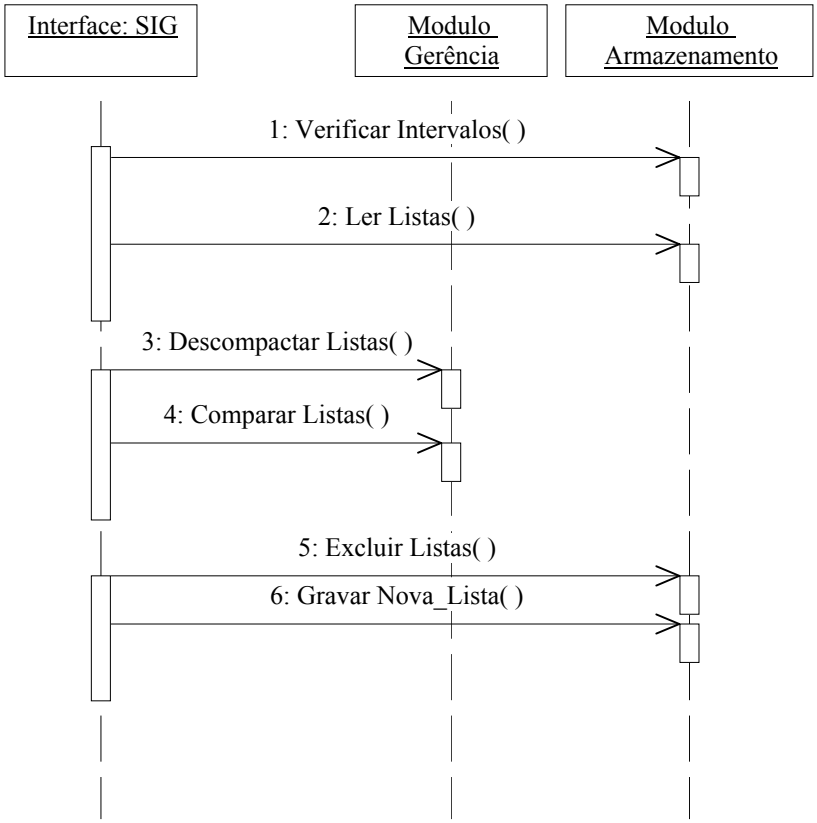


Figura 4.11 - Diagrama de Seqüências de Compactação

O algoritmo para compactação de listas é o mesmo algoritmo utilizado para a consulta com exceção de que o processo de recuperação em cascata não segue até a lista original, porém até a menor data estipulada para compactação. Após o processo de compactação de listas e ordenação da lista gerada, deve-se chamar um procedimento que exclua as listas de entrada e o procedimento para criar o novo índice contendo a maior data do intervalo.

a) PROCEDIMENTO COMPACTAR LISTAS

{O procedimento de compactação, na realidade, chama o procedimento de consulta, ordenação, criação de índices e exclusão de listas}

Para a compactação de listas, primeiramente, define-se o intervalo das listas a serem compactadas. Informa-se esse intervalo e o procedimento de consulta é chamado. O procedimento de consulta gera a lista com as regiões do intervalo desejado. Após, aplica-se o algoritmo de ordenação, cria-se o novo índice e deleta-se as listas que foram compactadas.

b) PROCEDIMENTO EXCLUIR LISTAS (Data_Validade)

{Entra-se com as datas de validade que apontam para as listas a serem excluídas}

```
Indice: ponteiro;
Inicio
Se indice.dado = data_validade então
Lista.delete;
Fim.
```

Esse algoritmo de exclusão de listas só pode ser chamado pelo algoritmo de compactação, pois é só dessa maneira que se permite deletar listas sem deletar todo o Sistema de *Quadtrees* Lineares Temporais .

5 Conclusões

Os modelos conceituais de SIGs representam os fenômenos geográficos como entidades ou objetos. Os objetos geográficos apresentam características descritivas, espaciais e temporais.

Alguns modelos conceituais tratam das características temporais dos objetos geográficos, porém, a temporalidade não é tratada no nível de armazenamento e manipulação dos objetos geográficos.

Para o armazenamento e manipulação de dados geográficos no formato matricial, as estruturas mais utilizadas são as *quadtrees*.

É possível o armazenamento de várias versões de um dado espacial no formato matricial pelas *quadtrees*, porém a estrutura não consegue identificar as versões como sendo do mesmo dado.

Além disso, em uma nova versão, todo o dados espacial no formato matricial deve ser armazenado novamente, mesmo que a diferença entre as versões seja mínima.

O objetivo desse trabalho foi o de estender estruturas *quadtrees* para suportar o armazenamento e a recuperação de dados espaço-temporais em formato matricial. Para se atingir o objetivo foram traçadas três metas a serem alcançadas: o estudo de modelos de dados conceituais, o estudo dos diversos tipos de *quadtree* e de suas operações e a investigação sobre a possibilidade de extensão de algum tipo de *quadtree*.

O Sistema de *Quadtrees* Lineares Temporais aqui desenvolvido possui um Modulo de Armazenamento (MA), e um Módulo de Gerenciamento de Dados (MGD). No MGD encontram-se as operações utilizadas pelo Sistema para armazenar e manipular Dados Espaço-Temporais no formato Matricial (DETMAs).

A operação de inserção possibilita a inclusão do primeiro estado de um DETMA. A operação de atualização insere somente as diferenças de um novo estado de um DETMA. A operação de consulta possibilita a recuperação de um DETMA completo. A operação de exclusão deleta um DETMA completo e a operação de compactação possibilita a integração de estados de um DETMA.

A possibilidade de compactação do Sistema de *Quadtrees* Lineares Temporais se faz útil ao economizar espaço de memória e tempo de processamento.

O Sistema de *Quadtrees* Lineares Temporais aqui desenvolvido procurou solucionar algumas limitações da estrutura na qual se baseou [TZO98], propondo trabalhar com imagens coloridas, tempo de validade e listas ordenadas, preservando a estrutura de *quadtrees* lineares. Apesar de ponteiros serem mais velozes nas consultas, ocupam mais espaço de memória e tornam os algoritmos de implementação mais complexos. Devido a esses fatores procurou-se minimizar a utilização de ponteiros.

O protótipo foi desenvolvido para comprovar o critério de correção, ou seja, testar se os algoritmos realmente funcionam. Apesar dos critérios de eficiência não terem sido testados, acredita-se que o tempo de resposta pode ser melhorado pelo simples fato de armazenar apenas as diferenças dos DETMAS nas atualizações.

Existem algumas limitações na estrutura aqui apresentada, ela não representa o tempo de transação e executa as operações de atualização e consulta na memória principal. Isso significa que o Sistema continua atuando como os SIGs, recupera as listas na memória, para depois executar as operações. Além disso, o Sistema de *Quadtrees* Lineares Temporais não permite o acesso randômico para recuperação de parte das listas, permitindo apenas a sua recuperação completa.

Em relação à exclusão, quando se deseja excluir um Sistema de *Quadtrees* Lineares Temporais, todas as listas pertencentes a esse sistema devem ser excluídas. Deve-se observar esse fator no momento de implementação, uma vez que existe um algoritmo que exclui somente as listas que passaram pelo processo de compactação. Esse algoritmo só pode ser acessado pelo processo de compactação, pois qualquer outro acesso pode ocasionar um erro de consistência (falta de lista) no sistema de *quadtrees* lineares.

Esse trabalho se propunha apenas a estender *quadtrees* para que pudessem armazenar e manipular dados espaço-temporais, provando sua funcionalidade. Não foi realizado nenhum teste de verificação e otimização de consultas. Mesmo assim, o problema de economia de memória e tempo de processamento de consultas foi minimizado, pois o espaço ocupado pela diferença da imagem em uma atualização é consideravelmente menor em relação a imagem completa e esse fato, por si só economiza espaço de memória.

Propõem-se para trabalhos futuros:

- O desenvolvimento completo do protótipo com todas as operações aqui desenvolvidas (inclusão, consulta, exclusão, atualização e compactação);
- Realização de testes de verificação para estimativas de tempo de processamento em atualizações e consultas;
- Estudo de outras formas de implementação para otimizar tempo de acesso e uso de memória.

Anexo 1 Desenvolvimento de um Protótipo para o Sistema de *Quadtrees* Lineares Temporais

O protótipo foi desenvolvido na linguagem Pascal 7.0, para demonstrar o critério de correção dos algoritmos propostos.

O protótipo possui módulos que implementam as rotinas para as operações de inclusão, atualização e consulta como apresenta a Figura A.1.

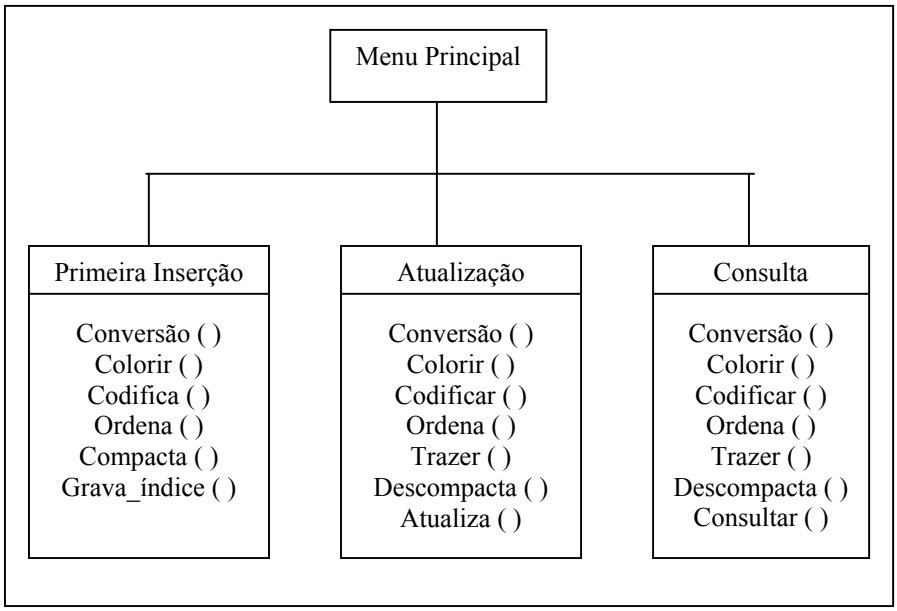


Figura A.1 - Módulos do Protótipo

Partiu-se do princípio que o arquivo de entrada é um DETMA com data de validade e a tupla {(coordenadas, cor)}.

Inicialmente é realizada a leitura desse arquivo e a sua conversão para uma *quadtree* linear para posterior armazenamento em um arquivo.

As rotinas que compõem o módulo da inserção do estado inicial do DETMA são:

- *Procedure Conversão:* lê o arquivo de entrada e converte para números binários;
- *Procedure Colorir:* lê as cores das células do arquivo de entrada;
- *Procedure Codifica:* gera a lista linear, com o código de localização e cor;
- *Procedure Ordena*: ordena a lista linear gerada;
- *Procedure Compacta:* compacta a lista já ordenada;

- *Procedure grava_índice:* grava a data inicial de validade da lista linear e armazena em arquivo;

Na operação de atualização o arquivo de entrada passa pelo mesmo processo que a inclusão até o processo de ordenação. Após, é recuperada a lista com data inferior a lista de entrada, feita a comparação entre as listas e gerada a lista com a diferença. Esse processo é executado em cascata até se chegar a lista com estado inicial para posterior armazenamento. A estratégia utilizada no protótipo foi de armazenar cada atualização em um arquivo. Não foi prevista na implementação do protótipo a situação de uma atualização possuir data de validade inferior ao estado final do Sistema de *Quadtrees* Lineares Temporais.

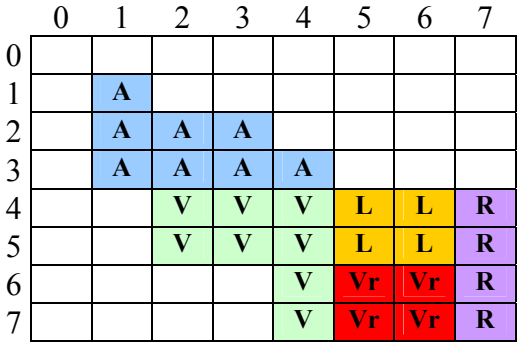
As rotinas para o módulo de atualização são:

- *Procedure Conversão:* lê o arquivo de entrada e converte para números binários;
- *Procedure Colorir:* lê as cores das células do arquivo;
- *Procedure Codifica:* gera a lista linear, com o código de localização e cor;
- *Procedure Ordena:* ordena a lista linear gerada;
- *Procedure Trazer*: carrega o arquivo com data de validade inferior a ser comparado com a lista de entrada;
- *Procedure Descompacta:* descompacta o arquivo para comparação com a lista de entrada;
- *Procedure Atualiza:* compara as listas, gerando uma nova lista com a diferença (esse processo é realizado em cascata até se chegar a lista com estado inicial);
- *Procedure Ordena:* ordena a lista com a diferença;
- *Procedure Compacta*: compacta a lista já ordenada;
- *Procedure grava_índice:* grava a data de validade da lista e armazena em arquivo;

A operação de consulta é realizada de acordo com a data solicitada pelo usuário. Não foi previsto no desenvolvimento do protótipo a situação de solicitação de uma consulta com data de validade inferior ao estado inicial do Sistema de *Quadtrees* Lineares Temporais. As rotinas para o módulo de consulta são;

- *Procedure Trazer*: recupera o arquivo de acordo com a data solicitada e o arquivo com data anterior a essa;
- *Procedure Descompacta*: descompacta os arquivos para recuperação;
- *Procedure Consulta*: gera a lista a ser recuperada;
- *Procedure Ordena*: ordena a lista gerada;

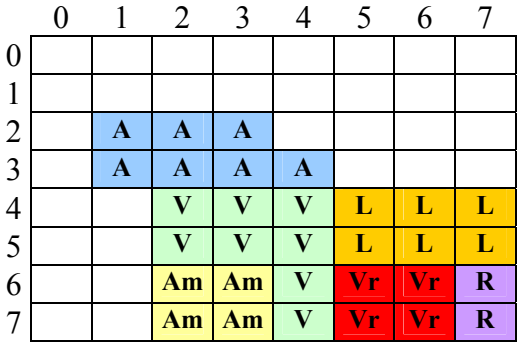
Para a verificação do funcionamento do protótipo foram realizadas as operações para um conjunto de imagens representadas por um DETMA. A Figura A.2 apresenta as matrizes, bem como a lista linear correspondente gerada pelo protótipo.



	0	1	2	3	4	5	6	7
0								
1		A						
2		A	A	A				
3		A	A	A	A			
4			V	V	V	L	L	R
5			V	V	V	L	L	R
6					V	Vr	Vr	R
7					V	Vr	Vr	R

1985
003 A 0 – 021 A 0 – 023 A 0 – 030 A 3 – 122 A 0 – 210 V 3 – 300 V 0 - 301 L 0 – 302 V 0 – 303 L 0 – 310 L 0 – 311 R 0 – 312 L 0 – 313 R 0 – 320 V 0 – 321 Vr 0 – 322 V 0 – 323 Vr 0 – 330 Vr 0 – 331 R 0 – 332 Vr 0 333 R 0.

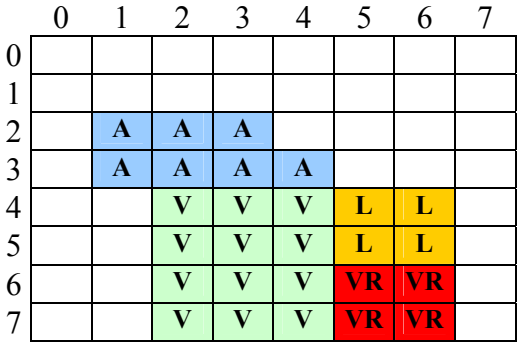
(a) Primeira Inserção



	0	1	2	3	4	5	6	7
0								
1								
2		A	A	A				
3		A	A	A	A			
4			V	V	V	L	L	L
5			V	V	V	L	L	L
6			Am	Am	V	Vr	Vr	R
7			Am	Am	V	Vr	Vr	R

1990
003 0 0 – 230 Am 3 - 311 L 0 313 L 0

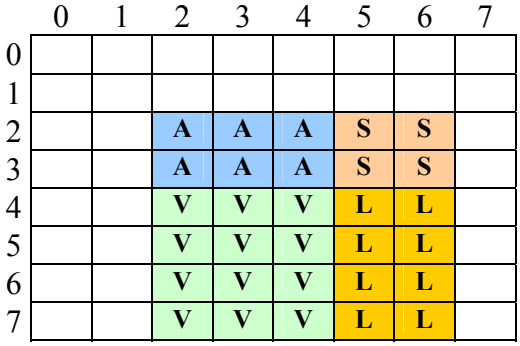
(b) 1ª Atualização



	0	1	2	3	4	5	6	7
0								
1								
2		A	A	A				
3		A	A	A	A			
4			V	V	V	L	L	
5			V	V	V	L	L	
6			V	V	V	VR	VR	
7			V	V	V	VR	VR	

1995
230 V 3 311 0 – 313 0 0 – 331 0 0 – 333 0 0.

(b) 2ª Atualização



	0	1	2	3	4	5	6	7
0								
1								
2			A	A	A	S	S	
3			A	A	A	S	S	
4			V	V	V	L	L	
5			V	V	V	L	L	
6			V	V	V	L	L	
7			V	V	V	L	L	

2000
021 0 0 – 023 0 0 – 120 A 0 – 121 S 0 – 123 S 0 130 S 0 – 132 S 0 – 321 L 0 – 323 L 0 – 330 L 0 – 332 L 0

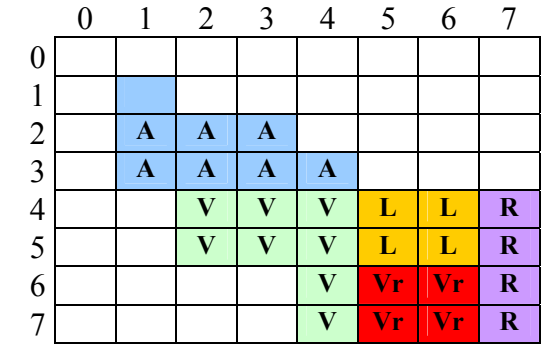
(b) 3ª Atualização

Figura A.2 - Exemplo de imagens representadas por matrizes

Na Figura A.2 acima as listas lineares são representadas por seu tempo de início de validade. Os três campos correspondem a:

- código de localização: código quaternário que corresponde as coordenadas da matriz;
- cor: cor da célula na matriz;
- marca de compactação: indica se existe compactação de células, quando for 0 não há compactação, quando for maior que 0, significa que existem mais células seqüenciais da mesma cor. Por exemplo, 230 V 4, indica na verdade 230, 231, 232, 233 na cor verde.

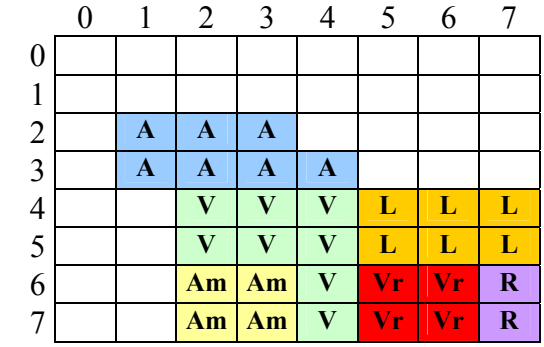
As operações de consulta realizadas no protótipo, dependendo do tempo de validade são apresentadas na figura A.3.



	0	1	2	3	4	5	6	7
0								
1								
2		A	A	A				
3		A	A	A	A			
4			V	V	V	L	L	R
5			V	V	V	L	L	R
6					V	Vr	Vr	R
7					V	Vr	Vr	R

003 A 0 – 021 A 0 – 023 A 0 – 030 A 3 – 122 A 0 – 210 V 3 – 300 V 0 - 301 L 0 – 302 V 0 – 303 L 0 – 310 L 0 – 311 R 0 – 312 L 0 – 313 R 0 – 320 V 0 – 321 Vr 0 – 322 V 0 – 323 Vr 0 – 330 Vr 0 – 331 R 0 – 332 Vr 0 333 R 0.

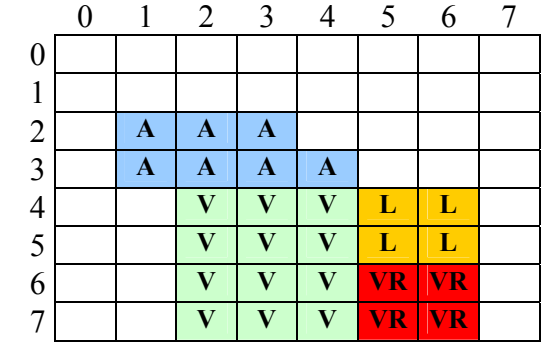
Recuperação da Imagem válida em 1987



	0	1	2	3	4	5	6	7
0								
1								
2		A	A	A				
3		A	A	A	A			
4			V	V	V	L	L	L
5			V	V	V	L	L	L
6			Am	Am	V	Vr	Vr	R
7			Am	Am	V	Vr	Vr	R

003 0 0 – 021 A 0 – 023 A 0 – 030 A 3 – 122 A 0 – 210 V 3 – 230 Am 4 – 300 V 0 - 301 L 0 – 302 V 0 – 303 L 0 – 310 L 0 – 311 L 0 – 312 L 0 – 313 L 0 0 – 320 V 0 – 321 Vr 0 – 322 V 0 – 323 Vr 0 – 330 Vr 0 – 331 R 0 – 332 Vr 0 333 R 0.

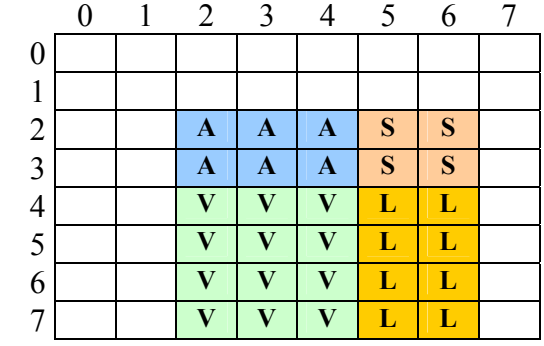
Recuperação da imagem válida entre 1991 e 1984



	0	1	2	3	4	5	6	7
0								
1								
2		A	A	A				
3		A	A	A	A			
4			V	V	V	L	L	
5			V	V	V	L	L	
6			V	V	V	VR	VR	
7			V	V	V	VR	VR	

003 0 0 – 021 A 0 – 023 A 0 – 030 A 3 – 122 A 0 – 210 V 3 – 230 V 4 – 300 V 0 - 301 L 0 – 302 V 0 – 303 L 0 – 310 L 0 – 311 0 – 312 L 0 – 313 0 0 – 320 V 0 – 321 Vr 0 – 322 V 0 – 323 Vr 0 – 330 Vr 0 – 331 0 0 – 332 Vr 0 – 333 0 0.

Recuperação da imagem válida em 1998



	0	1	2	3	4	5	6	7
0								
1								
2			A	A	A	S	S	
3			A	A	A	S	S	
4			V	V	V	L	L	
5			V	V	V	L	L	
6			V	V	V	L	L	
7			V	V	V	L	L	

003 0 0 – 021 0 0 – 023 0 0 – 030 A 3 – 120 A 0 – 121 S 0 – 122 A 0 – 123 S 0 130 S 0 – 132 S 0 – 230 V 4 – 300 V 0 - 301 L 0 – 302 V 0 – 303 L 0 – 310 L 0 – 311 0 – 312 L 0 – 313 0 0 – 320 V 0 – 321 L 0 – 322 V 0 – 323 L 0 – 330 L 0 – 331 0 0 332 L 0 – 333 0 0.

Recuperação da imagem válida em 2000

Figura A.3 - Exemplo de operação de consulta

ANEXO 2 Codificação das Rotinas

```
procedure conversao;
{Lê o arquivo de entrada e converte as coordenadas para numeros binarios}
var
g,i,j,n,a,cont1,cont2,s,tam:integer;
c,d: array[0..2] of integer;
x:real;
begin
cor1:=1;
c[0]:=0;
c[1]:=0;
c[2]:=0;
d[0]:=0;
d[1]:=0;
d[2]:=0;
assign(arqmat,'imagem.dat');
reset(arqmat);
read(arqmat,arqm);
x:= (sqrt(filesize(arqmat)-1)-1);
n:=trunc(x);
colorir;
g:=3;
for I:= 0 to n do
 begin
  d[0]:=0;
  d[1]:=0;
  d[2]:=0;
for J:= 0 to n do
   begin
   cont1:=i;
   a:=g-1;
   repeat
   c[a]:= cont1 mod 2;
   cont1:=cont1 div 2;
   a:=a-1
   until cont1 = 0;
    cont2:=j;
      a:=g-1;
    repeat
     d[a]:= cont2 mod 2;
     cont2:=cont2 div 2;
     a:=a-1;
   until cont2 = 0;
   codifica(c,d);
  end;
 end;
ordena;
if opcao = '1' then
```

```
begin
compacta;
criar_abrir;
grava_indice;
end
else
if opcao = '2' then
begin
trazer;
ordena;
compacta;
criar_abrir;
grava_indice;
end;

procedure colorir;
{Lê as cores dos pixels do arquivo original de entrada para um vetor}
var
i:integer;
begin
seek(arqmat,1);
for i:= 1 to 64 do
begin
read(arqmat,arqm);
cores[i]:= arqm.cor;
seek(arqmat, filepos(arqmat));
end;
end;

procedure codifica(c1,d1:array of integer);
{gera o codigo de localização e atribui a cor ao arquivo linear}
var
i,n,v,code: integer;
k:array [0..2] of char;
begin
n:=2;
k[0]:='0';
k[1]:='0';
k[2]:='0';
for i:= n downto 0 do
begin
if (c1[i] = 0) and (d1[i] = 0) then k[i]:='0';
if (c1[i] = 0) and (d1[i] = 1) then k[i]:='1';
if (c1[i] = 1) and (d1[i] = 0) then k[i]:='2';
if (c1[i] = 1) and (d1[i] = 1) then k[i]:='3';
end;
val((k[0]+k[1]+k[2]),v,code);
clrscr;
gotoxy(20,12);
if cores[cor1] <> 0 then
```

```
begin
   original[x].codigo:=v;
   original[x].cor:= cores[cor1];
   x:=x+1;
   cont:=cont+1;
end;
cor1:=cor1+1;
end;

procedure ordena;
var
i,j,auxcodigo:integer;
auxcor:byte;
begin
j:=cont;
while j > 1 do
begin
 for i:=1 to j-1 do
 begin
 if (original[i].codigo > original[i+1].codigo) then
    begin
      auxcodigo:= original[i].codigo;
      auxcor:=original[i].cor;
      original[i].codigo:= original[i+1].codigo;
      original[i].cor:=original[i+1].cor;
      original[i+1].codigo:= auxcodigo;
      original[i+1].cor:= auxcor;
    end;
 end;
j:=j-1;
end;
end;

procedure compacta;
var
n,i,j,proximo:integer;
ok:boolean;
begin
i:=1;
j:=0;
n:=1;
ct:=0;
ok:=false;
repeat
proximo:=i;
while not ok do
Begin
if ((original[i].codigo)+1 = original[i+1].codigo) and ((original[i].cor) =
(original[i+1]).cor) then
 begin
```

```
   j:=j+1;
 end
 else
  ok:= true;
i:=i+1;
 end;
 atual[n].codigo:= original[proximo].codigo;
 atual[n].cor:= original[proximo].cor;
 atual[n].numero:= j;
 ct:= ct+1;
 n:=n+1;
 j:=0;
 ok:= false;
 until i > cont;
end;

procedure grava_indice;
{Procedimento para gravar a data de validade e o arquivo compactado}
var  pes:arquivo;
     indi, i1:integer;
     data_val:word;
  begin
  writeln('Digite a data de validade da imagem');
  readln(data_val);
  i1:=1;
  while  i1 <= ct do
  begin
  arq.cods[i1].codigo:= atual[i1].codigo;
  arq.cods[i1].cor:= atual[i1].cor;
  arq.cods[i1].numero:= atual[i1].numero;
  i1:=i1+1;
  end;
  arq.data:= data_val;
  seek(arqlista, filesize(arqlista)-1);
  write(arqlista,arq);   {grava o arquivo compactado}
 end;

procedure atualiza;
var
achou: boolean;
pos,i,j,n,m: integer;
begin
clrscr;
achou:=false;
pos:=1;
n:=cont;
m:=28;
for j:=1 to n do
begin
  for i:= 1 to m do
```

```
  begin
    if original[j].codigo = anterior[i].codigo then
      if original[j].cor = anterior[i].cor then
        begin
        achou:=true;
        i:=m;
        end
      else
        if original[j].cor <> 0 then
        begin
         diferenca[pos].codigo:= original[j].codigo;
         diferenca[pos].cor:= original[j].cor;
         achou:=true;
         i:=m;
         pos:=pos+1;
        end;
    if (i=m) and not(achou) then
      begin
       diferenca[pos].codigo:= original[j].codigo;
       diferenca[pos].cor:= original[j].cor;
       pos:=pos+1;
      end;
  end;
   achou:=false;
end;
for i:= 1 to m do
begin
  for j:= 1 to n do
  begin
    if anterior[i].codigo = original[j].codigo then
    begin
     achou:=true;
     j:=n;
    end;
  end;
 if not(achou) then
  begin
    diferenca[pos].codigo:= anterior[i].codigo;
    diferenca[pos].cor:= 0;
    pos:=pos+1;
  end;
end;
for i:= 1 to pos-1 do
begin
original[i].codigo:= diferenca[i].codigo;
original[i].cor:=diferenca[i].cor;
end;
cont:=pos-1;
end;
```

```
procedure descompacta;
var
i,qtd:integer;
begin
a:=1;
for i:= 1 to 28 do
begin
if compara[i].numero = 0 then
begin
anterior[a].codigo:= compara[i].codigo;
anterior[a].cor:= compara[i].cor;
a:=a+1;
end
else
begin
qtd:= compara[i].numero;
anterior[a].codigo:= compara[i].codigo;
anterior[a].cor:= compara[i].cor;
a:=a+1;
repeat
anterior[a].codigo:= ((anterior[a-1].codigo))+1;
anterior[a].cor:= compara[i].cor;
a:=a+1;
qtd:=qtd-1;
until qtd=0;
end;
end;
end;

procedure trazer;
var
teste,i1,n,m:integer;
begin
criar_abrir;
while not eof(arqlista) do
begin
read(arqlista, arq);
writeln(arq.data);
for i1 := 1 to 28 do
begin
compara[i1].codigo:= (arq.cods[i1].codigo);
compara[i1].cor:= (arq.cods[i1].cor);
compara[i1].numero:= (arq.cods[i1].numero);
end;
descompacta;
atualiza;
end;
end;
```

```
procedure consultas;
var
achou:boolean;
i,j,a,m,n:integer;

begin
clrscr;
achou:=false;
for j:= 1 to n do
begin
consulta[j].codigo:= atual[j].codigo;
consulta[j].cor:= atual[j].cor;
end;
a:=n+1;
for i:= 1 to m do
begin
 for j:= 1 to n do
 begin
 if anterior[i].codigo = atual[j].codigo then
 begin
 achou:=true;
 j:=n;
 end
 else
  if anterior[i].codigo < atual[j].codigo then
  begin
    consulta[a].codigo:= anterior[i].codigo;
    consulta[a].cor:= anterior[i].cor;
    a:= a+1;
    achou:=true;
    j:=n;
  end;
end;
 if not (achou) then
  begin
  consulta[a].codigo:= anterior[i].codigo;
  consulta[a].cor:= anterior[i].cor;
  a:= a+1;
  end;
  achou:=false;
end;
writeln('O resultado da consulta e:');
for i:= 1 to 20 do
begin
writeln(consulta[i].codigo);
writeln(consulta[i].cor);
readln;
end;
```

Bibliografia

[BOT95] BOTELHO, M.A. **Incorporação de facilidades espaço-temporais em bancos de dados orientados a objetos.** São Paulo: IMECC-UNICAMP, 1998.

[BRA94] BRAYNER, A. **Implementação de um sistema temporal em um banco de dados orientado a objetos.** Campinas: UNICAMP, 1994. Master´s Thesis.

[CAM94] CAMARA, G.; CASANOVA, M.; MEDEIROS, C.B.; SOUZA, M..; FREITAS, U. A model to cultivate objects and manipulate fields. In: ACM Workshop on Advances in GIS, 2., 1994. **Proceedings…** ACM Press, New York, 1994. p. 20-28.

[CAR93] CARON, C.; BÉDARD, Y. Extending the individual formalism for a more complete modeling of urban spatially referenced data. **Computers, Environment and Urban Systems,** [S.1.], v.17, p.337-346,1993.

[CIF95] CIFERRI, R. R. **Um benchmark voltado para análise de desempenho de um SIG.** Campinas: UNICAMP, 1995. Master´s Thesis.

[COR90] CORDEIRO, M. J. **Estruturas de dados aplicadas a cartografia automatizada.** Rio de Janeiro: Instituto Militar de Engenharia, 1990. Tese de Doutorado.

[DAV93] DAVID, B; VOISARD, B. **A unified approach to geographic data modeling.** Ludowig-Maximilians – Universitat Munchen, Institute Fur Informatik, September, 1993. Technical Report.

[DEA91] DeANTONELLIS, V.; PERNICI, B. **ITHACA Object Oriented Methodology Manual – Application Development Manual**. Milano: Politecnico di Milano, 1991. ITHACA ESPRIT Project.

[EDE94] EDELWEISS, N.; PALAZZO, J. **Modelagem de aspectos temporais.** Recife: UFPE-DI, 1994.

[EDE96] EDELWEISS, N. **TF-ORM:** O modelo e uma especificação. Porto Alegre: CPGCC da UFRGS, 1996. (RP - 258)

[FAR98] FARIA, G. **Um banco de dados espaço-temporal para desenvolvimento de aplicações em sistemas de informação geográfica**. Campinas: Instituto de Computação da UNICAMP, 1996.

[GAR82a] GARGANTINI, I. An effective way to represent *quadtrees*. **Communications of the ACM**, New York, v. 25, p. 905-910, Dec. 1982

[GAR82b] GARGANTINI, I. Detection of connectivity for regions represented by linear *quadtrees*. **Comput Mathe Appl**, New York, v. 8, p. 319-327, 1982.

[GOO83] GOODCHILD, M.F.; GRANDFIELD, A. W. Optimizing raster storage: an examination of four alternatives, **Proceedings of Auto-Carto** 6, vol. 1, Ottawa, October 1983, 400-407.

[GOR93] GORALWALLA, I. A. **Temporal Extension to a Uniform Behavioral Object Model.** Canada: University of Alberta, 1993.

[LAN95] LANGRAN, G. **Time in Geographical information systems.** Bristol, Pensilvania: Taylor Francis, 1995.

[LIS97] LISBOA, J. **Modelos Conceituais de Dados para Sistemas de Informações Geográficas**. Porto Alegre: CPGCC da UFRGS, 1997. 119p. (EQ-12).

[LIS00] LISBOA, J. **Projeto conceitual de banco de dados geográficos através da reutilização de esquemas, utilizando padrões de análise e um *framework* conceitual.** Porto Alegre: PPGC da UFRGS, 2000. Tese de doutorado.

[LOP99] LOPES, A.V. **Estrutura de Dados para Construção de Software.** Canoas: Ed. ULBRA, 1999.

[MOR66] MORTON, G.M. **A computer oriented geodetic data base and a new technique in file sequencing**, Ottawa, Canada: IBM, 1966.

[MOT99] MOTA, J.S. **Modelos de Dados Temporais:** um estudo comparativo. Porto Alegre: PPGC da UFRGS, 1999. (TI-904).

[OLI93] OLIVEIRA, L. C. **Incorporação da dimensão temporal em bancos de dados orientados a objetos.** Campinas: Universidade Estadual de Campinas, 1993. Master's thesis.

[OLI97] OLIVEIRA, J.; PIRES, F.; MEDEIROS, C.B. Na Environment for integrated modelling and analysis of geographic information. **GeoInformatica**, Boston, n.1, p.29-58, 1997.

[ORE82] ORENSTEIN, J. A. Multidimensional tries used for associative searching, **Information Processing Letters**, New York, v. 14, n. 4, p. 150-157, June, 1982.

[O2t93] O2TECHNOLOGY. **The O2 user manual, version 4.3.** Versailles, France, 1993. Technical Report.

[PER90] PERNICI, B. et al. Object with roles. **SIGOIS Bulletin**, v.11, n.2-3, p.205-15, 1990. Trabalho apresentado na: ACM/IEEE CONFERENCE ON OFFICE INFORMATION SYSTEMS, 1990, Cambridge, MA.

[PEU84] PEUQUET, D. A conceptual framework and comparison of spatial data models. **Cartographica**, Zurich, v. 21, p. 66 – 113, 1984.

[PEU98] PEUQUET, D. et al. **Delineating operations for visualization and analysis of space-time data in GIS**. Pennsylvania: Department of Geography. The Pennsylvania State University, 1998.

[QIA98a] QIAN, L.; PEUQUET, D. **Desing of a visual query language for GIS.** Pennsylvania: Department of Geography. The Pennsylvania State University, 1998.

[QIA98b] QIAN, L.; PEUQUET, D. **A systematic strategy for high performance GIS.** Pennsylvania: Department of Geography. The Pennsylvania State University, 1998.

[SAM84] SAMET, H. The *quadtree* and related hierarchial data structures. **Computing Surveys**, New York, v. 16., n. 2, p. 187-222, June 1984.

[SAM89a] SAMET, H. **The desing and analysis of spatial data structures.** Boston: Addison Wesley, 1989.

[SAM89b] SAMET, H. **Applications of Spatial Data Structures.** Boston: Addison Wesley, 1990.

[SAM90] SAMET, H. **Applications of spatial data structures:** computer graphics, image processing and GIS. Addison Wesley, 1990.

[TZO98] TZOURAMANIS, T. Overlapping linear *quadtrees*: a spatio-temporal access method. In: ACM-GIS, 1998. **Proceedings…,** Bethesda, MD, November 1998, p. 1-7.