AVALIAÇÃO DE JUNÇÕES EM BANCOS DE DADOS ESPACIAIS

Geraldo Zimbrão da Silva

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Jano Moreira de Souza, Ph.D.

Profa. Claudia Bauzer Medeiros, Ph.D.

Prof. Rubens Nascimento Melo, Ds.C.

Profa. Maria Luiza Machado Campos, Ph.D.

Prof. Marta Lima Queirós Mattoso, D.Sc.

Prof. Claudio Esperança, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

JUNHO DE 1999

SILVA, GERALDO ZIMBRÃO

Avaliação de Junções em Bancos de Dados Espaciais [Rio de Janeiro] 1999.

VII, 137 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 1999)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Bancos de Dados Espaciais

I. COPPE/UFRJ II. Título (série)

AGRADECIMENTOS

Agradeço ao meu Orientador, Prof. Jano, pelo apoio em todos os momentos deste longo e difícil trabalho, pela sua compreensão e por ter acreditado em meu potencial.

Agradeço à minha esposa, Renata, por estar sempre ao meu lado, e à minha filha, Helena, por compreender as tantas vezes que tive de me ausentar.

Agradeço aos meus familiares, em especial a minha mãe Ivanilde, pelo apoio nas horas difíceis, pelo tempo de lazer perdido e tantos outros contratempos.

Agradeço a todos os meus amigos, especialmente ao Lúcio e ao Marcos, por todo o apoio que sempre me deram.

Agradeço a todos os colegas do programa, entre professores, alunos e funcionários, e que muito contribuíram para que essa Tese fosse realizada.

Finalmente, agradeço ao Senhor por me permitir realizar este trabalho.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

AVALIAÇÃO DE JUNÇÕES EM BANCOS DE DADOS ESPACIAIS

Geraldo Zimbrão da Silva

Junho/1999

Orientador: Jano Moreira de Souza

Programa: Engenharia de Sistemas e Computação

Este trabalho trata de alguns aspectos práticos de implementação de operações de Bancos de Dados Espaciais. Tais aspectos dizem respeito a conceitos teóricos que já são bem aceitos e definidos. Em particular, estaremos tratando da realização eficiente de junções em Bancos de Dados Espaciais. Para tanto, propomos e implementamos novos índices espaciais e novas aproximações para filtragem que permitem a realização mais eficiente de tais junções, considerando o aproveitamento racional do espaço em disco e melhorando o tempo de execução de operações e de resposta às consultas. Os resultados experimentais observados foram melhores quando comparados às técnicas atualmente empregadas.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

PERFORMING JOINS IN SPATIAL DATABASES

Geraldo Zimbrão da Silva

June/1999

Advisor: Jano Moreira de Souza

Department: Systems and Computing Engineering

This work presents some practical implementation issues of Spatial Databases. Such issues are concerned with theoretical concepts that now are well accepted and defined. Mainly, we will be investigating the efficient execution of joins in Spatial Databases. So, we have proposed and implemented new spatial indexes and new filter approximations that allow a more efficient accomplishment of such joins, considering the rational use of the disk space while decreasing the time spent performing queries and operations. The observed experimental results were superior when compared to current techniques.

Índice

1. Introdução

Nos últimos anos a área de bancos de dados espaciais tem adquirido uma importância cada vez maior tanto para a indústria quanto para o setor de pesquisa. Em decorrência disso, essa área de bancos de dados vem experimentando um rápido desenvolvimento. Novos algoritmos e estruturas de dados vêm sendo desenvolvidos e aperfeiçoados para satisfazerem às exigências peculiares aos bancos de dados espaciais, entre as quais podemos citar o grande volume de dados e a natureza espacial das operações sobre os mesmos.

Existe uma grande quantidade de sistemas que representam dados com atributos espaciais. No entanto, apenas duas categorias de aplicações respondem pela quase totalidade de tais sistemas. Tais categorias são CAD/CAM e SIG. Os sistemas de CAD/CAM (*Computer Aided Design / Computer Aided Manufacturing* - Projeto Assistido por Computador / Fabricação Assistida por Computador) representam informações espaciais, em geral tridimensionais, de peças e outros componentes, com nível de detalhes suficientes para a sua fabricação. Muitas vezes tais sistemas lidam com milhões de elementos, caso por exemplo dos projetos de circuitos integrados. Já os SIGs (Sistemas de Informações Geográficas) tratam, em sua maioria, de informações espaciais bidimensionais, porém georeferenciadas, ou seja, representadas em termos de suas coordenadas geográficas. Também nesse caso há sistemas que lidam com milhões de elementos.

Enquanto que os atuais sistemas de CAD/CAM em geral foram construídos a partir do zero ou quase isso após algumas tentativas frustradas de se utilizar um SGBD convencional, os SIGs atuais são, em sua maioria, construídos sobre bancos de dados espaciais ou mesmo relacionais, sendo essa categoria de aplicação a grande responsável pelo rápido desenvolvimento da área de bancos de dados espaciais. A demanda constante por sistemas que suportem cada vez mais informações tem levado ao aprimoramento dos algoritmos já existentes e implementados para cada tipo de problema.

No entanto, os bancos de dados espaciais, por serem em geral construídos utilizando-se alguma forma os bancos de dados relacionais, carecem ainda de modelos e algoritmos especialmente desenvolvidos para as suas necessidades específicas. Não existe ainda um modelo a ser seguido para a construção de um banco de dados espaciais.

Como exemplo, podemos citar os atuais esforços que ainda são feitos para criação de um padrão SQL para consultas espaciais (SQL3, 1995; SAIF, 1994; KAUSHIK e RUNDENSTEINER, 98).

Embora um grande número de questões nessa área ainda esteja em aberto, já há consenso sobre a necessidade de se satisfazer a alguns requisitos identificados: por exemplo, um banco de dados espaciais deve oferecer as mesmas funcionalidades para os dados espaciais que um banco de dados relacional oferece aos dados convencionais. Em vista disso, um banco de dados espaciais deve fornecer suporte para responder consultas *ad-hoc* sobre os atributos espaciais armazenados. Por analogia com os bancos de dados relacionais, tais consultas devem ser decompostas em consultas menores, mais simples, que possam ser implementadas através de um conjunto pequeno de operadores espaciais, análogos ou complementares aos operadores definidos pela álgebra relacional.

É correto afirmar que a teoria que oferecerá um embasamento mais sólido permitindo o desenvolvimento e disseminação do uso de Bancos de Dados Espaciais ainda se encontra em sua infância, o que inclusive justifica o montante de pesquisa atualmente realizada nessa área. Por outro lado, alguns tópicos práticos já podem ser investigados esperando-se obter resultados confiáveis e aplicáveis, sobretudo por dois motivos:

1) muitas aplicações e protótipos já foram ou estão sendo desenvolvidos, apresentando um conjunto de requisitos que terão de ser contemplados de alguma forma;
2) parte dos conceitos teóricos desenvolvidos já podem ser considerados como bem aceitos e cristalizados, tanto pelos pesquisadores como pelos usuários.

A presente Tese trata, portanto, de alguns aspectos práticos de implementação de operações de Bancos de Dados Espaciais. Tais aspectos dizem respeito a conceitos teóricos que já são bem aceitos e definidos. Em particular, estaremos tratando da realização eficiente de junções em Bancos de Dados Espaciais. Para tanto, iremos propor novos índices espaciais e outras técnicas que permitirão a realização mais eficiente de tais junções, considerando o aproveitamento racional do espaço em disco e melhorando o tempo de execução de operações e de resposta às consultas.

1.1 Definindo o Problema

Chamamos de Sistemas de Informações Geográficas, ou simplesmente SIGs, aos sistemas que permitem o registro de informações geográficas. SIGs caracterizam-se principalmente por utilizarem informações georeferenciadas, que são informações que descrevem objetos e fenômenos do mundo real associados a uma localização na superfície terrestre e válidas por um determinado período de tempo. Assim como para o desenvolvimento de sistemas de informação convencionais são utilizados os chamados Sistemas Gerenciadores de Bancos de Dados, ou simplesmente Bancos de Dados, para o desenvolvimento de SIGs são utilizados Sistemas Gerenciadores de Bancos de Dados Espaciais, ou simplesmente Bancos de Dados Espaciais.

O dado mais importante armazenado em um SIG é o objeto espacial georeferenciado. Nesse trabalho, objetos espaciais são caracterizados por possuírem pelo menos um atributo que descreve a sua extensão espacial através de uma ou mais linhas poligonais. É importante observar que a única restrição feita quanto à forma de um objeto espacial é que ele seja fechado, podendo no entanto possuir buracos (p.e. lagos) ou ser desconexo (p.e. um arquipélago). Em outras palavras, estaremos interessados principalmente em objetos que possuem área ou superfície.

1.1.1 Tipos de Consultas Espaciais

Para os bancos de dados de uma forma geral podemos distinguir dois tipos de consultas. O primeiro tipo é chamado de consulta de única passada, e requer a leitura de cada objeto apenas uma vez, e por isso o seu tempo de execução é no máximo linearmente proporcional ao número total de objetos - n - armazenados no repositório de dados em questão. Na maior parte dos casos esse tempo é apenas uma fração de n, sendo que em outras é até mesmo proporcional ao logaritmo de n. Consultas envolvendo intervalos de valores para determinados atributos são exemplos desse tipo de consulta em bancos de dados relacionais. De forma análoga, consultas envolvendo "janelas" ou "recortes" de uma região, ou ainda as consultas ditas pontuais são exemplos de consultas de passada simples em bancos de dados espaciais. Várias abordagens para o processamento eficiente de tais consultas já foram discutidas na literatura, sendo que SAMET (1990) apresenta uma seleção abrangente das mesmas para bancos de dados espaciais.

O segundo tipo de consulta compreende as consultas de múltiplas passadas, onde os objetos têm de ser lidos diversas vezes e, consequentemente, o tempo de

execução em geral não é linear, mas superlinear quando tomado em proporção ao número de objetos contidos na relação. Como exemplo de consulta de passadas múltiplas podemos citar a operação de junção em um banco de dados relacional. Nos Sistemas de Informações Geográficas atuais, duas das operações mais frequentes são as operações de sobreposição e interseção de mapas, genericamente chamadas de junções espaciais, e que são exemplos de consultas de múltiplas passadas em bancos de dados espaciais.

1.1.2 Usuários das Consultas

Para melhor compreender a natureza das consultas que iremos estudar, vamos considerar o perfil simplificado de alguns usuários típicos de Sistemas de Informações Geográficas: o usuário comum, o especialista e os tomadores de decisões.

Um usuário comum de um sistema de informações geográficas em geral tem interesse em saber a distribuição de determinada(s) propriedade(s) em relação a uma região em uma dada época. Em outras palavras, o que tal usuário deseja são mapas convencionais armazenados eletronicamente. Além disso, tal usuário também terá um certo interesse em informações sobre a evolução de uma ou outra entidade geográfica.

Um especialista estará frequentemente interessado na obtenção de novas informações a partir das já existentes, em geral cruzando informações distintas sobre uma mesma região, possivelmente ao longo do tempo. Além disso, um outro trabalho frequentemente desenvolvido por especialistas é a elaboração de hipóteses e conjecturas e a sua verificação contra os dados reais. Assim, pesquisadores usualmente estão atrás de fatos na base que comprovem ou refutem alguma teoria. No entanto, embora para tais usuários seja relativamente fácil saber o quê se procura, em geral não se tem a menor idéia de onde o encontrar, muitas vezes sendo necessário procurar na base inteira.

Usuários cujo conhecimento está concentrado em outras áreas, como tomadores de decisão, gostam de ver a evolução de um ou mais atributos em uma região ao longo do tempo. Além disso, tais usuários frequentemente estarão interessados na obtenção de resultados que sintetizem as informações contidas em tal base, que serão apresentados sob a forma de mapas, gráficos, tabelas ou ainda subconjuntos da base original. Resultados sintéticos envolvem o acesso a informações oriundas de várias regiões ou de várias épocas, ou ainda ambos.

Uma consulta é, então, uma operação de busca de determinada informação na base, possivelmente envolvendo o acesso a um grande volume de dados, não todos necessariamente relevantes. Chamaremos de consultas espaciais todas as consultas que envolvam o acesso a dados espaciais de uma ou mais regiões. A rigor, não trataremos de dados não espaciais em tais consultas, sendo a sua presença considerada de importância secundária para a presente tese - afinal, o volume de dados espaciais em geral é muito maior do que os dados não espaciais, e consultas em bases de dados não espaciais têm sido o tema de vários outros trabalhos. Por outro lado, somente consideraremos a finalidade da consulta na medida em que essa informação influencia o padrão de acesso. Em outras palavras, dividiremos as consultas em classes segundo os seus padrões de acesso, e não segundo a formulação original das mesmas.

1.2 Uso de BDEs para a Construção de SIGs

Nessa seção, mostraremos alguns argumentos para justificar o uso de banco de dados espaciais para o desenvolvimento de sistemas de informações geográficas, e a necessidade de técnicas eficientes para realização de consultas (especialmente as junções) nas bases dessas aplicações.

A tecnologia de banco de dados surgiu para preencher uma lacuna existente no tratamento dado pelos sistemas operacionais aos arquivos. Inicialmente, a idéia era centralizar as informações disponíveis em uma organização, disciplinando o acesso, evitando redundância e padronizando a organização dos dados. A funcionalidade oferecida não era muito grande, mas já era um passo além do sistema de arquivos. Foi a época dos chamados arquivos indexados — ISAM e VSAM (DATE, 1991).

Posteriormente, novas funcionalidades foram sendo incorporadas: era o início dos Sistemas Gerenciadores de Banco de Dados propriamente ditos. Sugiram então o sistema hierárquico, o sistema de rede e finalmente o sistema relacional. Novamente segundo DATE (1991), os dois primeiros sistemas não foram desenvolvidos a partir de um modelo de dados abstrato pré-definido. Foram simplesmente definidos por um processo de abstração realizado sobre implementações já existentes. O modelo relacional teria sido o primeiro exemplo de um modelo de dados definido *antes* da implementação propriamente dita.

Por sua vez, os sistemas de informações geográficas trilharam um caminho semelhante ao de outros tipos de sistemas de informações. Inicialmente, não havia

nenhum banco de dados por trás das aplicações. Toda a funcionalidade desejada teria de ser implementada a partir de no máximo algumas bibliotecas, e isso para cada sistema novo a ser construído. Além disso, o compartilhamento de informações entre aplicações diferentes era uma tarefa extremamente trabalhosa. Por analogia com os outros tipos de sistemas de informações, em pouco tempo percebeu-se que havia algumas vantagens em utilizar um banco de dados (possivelmente relacional) para o armazenamento das informações textuais associadas aos mapas. Logo em seguida, surgiram os banco de dados especializados em armazenar as informações espaciais propriamente ditas, e que dispunham de funcionalidades para realizar a integração entre os dois tipos de dados.

Dessa forma, os dados geográficos, assim como quaisquer outros, são melhores organizados e mais seguros em termos de armazenamento se estiverem estruturados em um banco de dados, tirando proveito de todas as vantagens daí decorrentes: eliminação da redundância, segurança, compartilhamento, padronização, recuperação depois de falhas etc.

Já existe uma razoável quantidade de Bancos de Dados Espaciais disponíveis comercialmente (ESRI, 1991, SMALWORLD, 1995) e que podem ser utilizados para o desenvolvimento de SIGs. No entanto, nenhum desses sistemas incorpora técnicas sofisticadas para a realização eficiente de junções espaciais, o que as torna operações extremamente custosas.

Um exemplo simples de consulta que envolve a realização de várias junções é a que nos é apresentada em VEENHOF et al. (1995): "recupere todas as áreas rurais abaixo do nível do mar que possuam tipo de solo igual a areia e distantes três milhas de lagos poluídos". Usualmente, as informações representadas em um SIG são agrupadas em camadas temáticas, que são conjuntos de informações sobre determinadas variáveis: altitude, uso da terra etc. Assim, com os planos temáticos uso da terra, solo, poluição e elevação, várias junções espaciais terão de ser realizadas para responder a tal consulta.

Atualmente já existem bases de dados geográficas realmente grandes, e a tendência é aumentar ainda mais o volume de informação na medida em que a tecnologia evolui. Novas fontes disponibilizadoras de informações geográficas (satélites, bibliotecas digitais, empresas de serviços públicos) estão aparecendo a cada dia. Como a avaliação de junções espaciais é uma consulta de passadas múltiplas, o tempo necessário para a sua execução, que nas bases atuais já é considerável quando

comparado às outras operações, tenderá a crescer a uma taxa maior do que o tamanho das bases.

Consequentemente, podemos afirmar que, mais ainda que nos bancos de dados relacionais, nos bancos de dados espaciais a operação de junção é de grande importância. Portanto, estabelecer algoritmos eficientes para a realização de junções espaciais é também fundamental para a avaliação efetiva de consultas espaciais, de forma a possibilitar o surgimento de aplicações que hoje são tecnologicamente inviáveis ou mesmo economicamente contraproducentes.

1.3 Aplicações que Usam Junções Espaciais

Em geral, todas as aplicações envolvendo SIGs possuem algum tipo de demanda para a realização de junções espaciais. Estaremos citando aqui, portanto, apenas aquelas áreas onde a realização de junções espaciais desempenha um papel central na aplicação em si. Grande parte das aplicações de SIG hoje em dia estão localizadas em agências ou órgãos governamentais. Uma razão para esse fato é que a informação geográfica se constitui em um fator estratégico, muitas vezes confidencial. Do restante das aplicações, boa parte pode ser encontrada em empresas que prestam serviços ao público em geral envolvendo algum tipo de rede, como as companhias de energia, telefones, transportes etc.

1.3.1 Sistemas Cadastrais

Um sistema cadastral mantém informações relacionadas a propriedades. Um exemplo bem típico são os cadastros que o governo mantém sobre as propriedades urbanas, e que são utilizados para o cálculo do IPTU (Imposto Predial e Territorial Urbano), ou ainda o cadastros sobre as propriedades rurais, utilizado para calcular o ITR (Imposto Territorial Rural) e para identificar propriedades que possam ser utilizadas para fins de reforma agrária. Embora tais cadastros atualmente não sejam feitos utilizando-se técnicas que os caracterizem como SIGs, esse certamente seria o próximo passo. Por outro lado, as junções, especialmente aquelas decorrentes da sobreposição de mapas, têm um papel de destaque em tais aplicações.

1.3.2 Gerência de Redes de Infra-estruturas Públicas

A criação e manutenção de redes para a oferta de serviços públicos, tanto por parte do governo como de companhias que explorem esses serviços, é uma tarefa com forte demanda por junções em SIGs. De fato, saber a localização exata de cada

interseção de uma rede de serviços com os objetos geográficos ou outras redes de serviços pode permitir uma operação e manutenção mais eficiente.

1.3.3 Gerência de Recursos Naturais

É do interesse de um país que agências governamentais tenham informações sobre os recursos naturais dentro de suas fronteiras, estejam eles sendo explorados comercialmente ou não. Em tais sistemas, junções espaciais são importantes sobretudo para a obtenção de variáveis econômicas potenciais, estimativas das reservas de recursos, identificação de áreas propícias à determinada atividade ou exploração econômica e também para a obtenção de estatísticas diversas.

1.3.4 Mapas para Navegação Eletrônica

Já está em curso nos EUA uma iniciativa para preparar o país para a implantação - em um futuro não muito distante - de um sistema que permitirá, entre outros serviços, uma gerência mais inteligente das rodovias. Esse sistema, conhecido como IVHS (*Intelligent Vehicle-Highway Systems*), oferecerá serviços que vão desde o controle de tráfego até a operação automática de veículos, passando por serviços do tipo rastreamento de veículos e alertas para evitar colisões (GORDON et al., 1994). O IVHS necessitará de grande quantidade de informações georeferenciadas em tempo real, o que incluirá a capacidade de realizar eficientemente junções espaciais *on the fly*.

1.3.5 Mineração de Dados Espaciais

Mineração de dados espaciais significa extrair informação e conhecimento espacial de alto nível de grandes bases de dados espaciais, objetivando a identificação de padrões e relações de causa e efeito, que permitam entre outras coisas a previsão da evolução de determinados cenários. No entanto, com o enorme crescimento do volume de informações espaciais, a habilidade humana foi sobrepujada em sua capacidade de interpretação. Surge então a necessidade de ferramentas e técnicas para análise de bancos de dados inteligentes e automatizados. A Prospecção de Conhecimento em Base de Dados (*Knowledge Discovery in Databases*) é o campo emergente na solução deste tipo de problema. Claro está que para o uso de tais ferramentas é necessária a realização de diversos tipos de consultas sobre os dados, inclusive as junções espaciais.

1.4 Contribuições desta Tese

Esse trabalho caracteriza as junções espaciais, mostrando os seus aspectos mais relevantes do ponto de vista computacional, bem como exemplos de situações práticas onde tais junções surgem.

Além disso, mostramos o estado da arte na realização de junções espaciais, o que por si só constitui-se em um valioso guia para a implantação de módulos processadores de tais junções. Para tanto, uma revisão da literatura correlata cuidadosa e abrangente é apresentada, com ênfase nos melhores resultados já obtidos.

Finalmente, projetamos, implementamos e avaliamos métodos e técnicas novas, especialmente desenvolvidas para a realização de junções espaciais, e que se mostraram muito eficientes quando comparadas às alternativas existentes. Analisar o motivo do sucesso de tais abordagens, indicando também os caminhos mais promissores para futuros melhoramentos nas estruturas de dados e algoritmos propostos.

1.5 Estrutura da Tese

Essa tese está estruturada da seguinte forma: o presente capítulo é a introdução, que também mostra a motivação para o trabalho a ser apresentado. No capítulo dois é feita uma revisão das áreas correlatas, mostrando uma revisão dos conceitos mais importantes relacionados com a área de bancos de dados espaciais que sejam úteis para a compreensão do restante da tese. No capítulo três é feita uma caracterização das consultas significativas, descrevendo com detalhes a operação de junção espacial e métodos eficientes para implementá-la. No capítulo quatro são apresentados os melhores índices para a realização das junções espaciais bem como as melhores aproximações e os melhoramentos propostos. Finalmente, no capítulo cinco temos as conclusões dessa tese.

2. Bancos de Dados Espaciais

Nesse capítulo iremos mostrar uma revisão dos conceitos mais importantes relacionados com a área de bancos de dados espaciais que sejam úteis para a compreensão do restante da tese. Se o leitor já estiver familiarizado com Sistemas de Informações Geográficas e bancos de dados espaciais, poderá, sem perda de contexto, seguir direto para o capítulo seguinte, que trata das junções espaciais.

Assim, esse capítulo terá a seguinte estrutura: na seção 1 teremos uma visão geral das áreas afins, e na seção 2, apresentaremos os principais conceitos relativos a Bancos de Dados Espaciais Orientado a Objetos.

2.1 Visão Geral das Áreas Afins

Vamos apresentar nessa seção uma breve introdução dos principais conceitos relativos a Sistemas de Informações Geográficas, bancos de dados espaciais e mineração de dados. O objetivo com isso é permitir ao leitor pouco familiarizado com tais sistemas uma visão geral dos mesmos, sem o quê o restante da presente tese se tornaria pouco compreensível, e uma padronização de termos e conceitos.

2.1.1 Sistemas de Informações Geográficas

Aprofundando a definição apresentada no capítulo anterior, iremos agora considerar alguns aspectos relevantes em relação aos Sistemas de Informações Geográficas.

2.1.1.1 Definição

Existem diversas definições para SIG, cada uma delas baseada no tipo do usuário ou domínio da aplicação. Segundo BAUZER e PIRES e BOTELHO (1994), a definição mais genérica seria um sistema digital de informação cujos registros são de alguma forma geograficamente referenciados (georeferenciados). Para uma definição mais precisa, seria necessário enfatizar as funcionalidades ou as aplicações suportadas. Por exemplo, um SIG é um sistema computadorizado que permite coletar dados georeferenciados das mais diversas fontes, processar, armazenar, recuperar e analisar os mesmos com o objetivo de gerar informações a partir dos dados existentes e apresentar resultados em um formato possível de ser compreendido pelo usuário.

2.1.1.2 Dados Armazenados em um SIG

Os dados armazenados em um SIG descrevem algo acerca do mundo real que é modelado durante uma determinada época, representando uma abstração da realidade. Esses dados são provenientes do estudo de fenômenos geográficos, relacionamentos temporais e a descrição de características que causam impacto no ambiente (através da observação direta ou indireta do mundo real). Pode-se classificar os dados de um SIG em três grupos: espaciais, pictóricos e não espaciais. Dados espaciais são dados que descrevem a geometria, a localização ou topologia de fenômenos geográficos. Dados pictóricos são imagens armazenadas, como fotos de satélite. Dados não espaciais são dados sobre o meio, atividades humanas e infra-estruturas usadas para conduzi-las.

2.1.1.3 Tipos de SIGs

Os SIGs podem ser classificados de acordo com a forma em que os dados foram modelados, o que em última análise depende da visão de mundo que o usuário possui: a visão baseada em regiões e a visão baseada em entidades.

A classe de SIGs baseada em regiões tende a ver o mundo como uma superfície contínua, sobre a qual estão definidas distribuições, também contínuas, representando os diversos atributos em estudo. Cada uma dessas distribuições é também chamada de camada temática. Uma camada temática pode ser formalizada como sendo uma função matemática $f(x,y)$ em D, que associa a uma determinada região do espaço um determinado valor (ou conjunto de valores) do domínio do atributo em questão. As entidades são criadas ao longo do processo de modelagem como forma de dividir a superfície em áreas (por exemplo, tipo de vegetação). Nesse modelo, entidades não existem por si só - a ênfase está no conteúdo de determinada área, e não na definição clara de sua identidade e fronteira. Com isso, torna-se difícil que duas entidades ocupem o mesmo espaço na mesma camada temática.

SIGs baseados em regiões em geral usam estruturas de dados em formato de células de um mosaico, como por exemplo reticulados regulares. Embora a maior parte dos sistemas utilizem células quadradas e do mesmo tamanho (chamadas de *pixels*), nada impede o uso de células irregulares (por exemplo, diagramas de Voroni). Um dos formatos mais difundidos é o formato *raster*, que nada mais é do que um reticulado regular de células retangulares ordenadas por linha - o que torna desnecessário o armazenamento das coordenadas de cada célula. Cada célula de uma estrutura *raster*

possui um e apenas um valor em uma camada temática - daí a sua imediata associação com funções.

Já a classe de SIGs baseada em entidades trata o espaço de informação como povoado de entidades identificáveis discretas (ou objetos), cada um com uma georeferência própria. Tais entidades existem por si, independente de camadas temáticas, e em geral são perfeitamente identificáveis com entidades do mundo real. Nesse modelo, é permitido que várias entidades se sobreponham, já que a ênfase está na descrição das mesmas.

SIGs baseados em entidades em geral usam estruturas de dados que comportam a representação de pontos, linhas e polígonos sob a forma de listas de coordenadas. As fronteiras das regiões são armazenadas sob a forma de linhas poligonais precisas. O conceito de entidade é o mesmo existente em BDs relacionais, de forma que uma entidade geográfica pode possuir uma série de atributos. Em contraste com o formato *raster*, esse formato de dados é chamado de formato vetorial. Melhorando a representação vetorial, é possível armazenar os dados de forma a conter informações topológicas explicitamente. Para tanto, os elementos básicos passam a ser: arco (série de pontos que começam e terminam em um nó), nó (início ou fim de um arco, ou o ponto de interseção de dois ou mais arcos) e polígono (uma cadeia fechada de arcos que representam seus limites).

2.1.1.4 Comparação Entre Representações Vetoriais e Raster

As estruturas vetoriais possibilitam uma boa representação dos fenômenos geográficos, possuem uma estrutura de dados compacta, permitem a descrição explícita da topologia, fornecem uma boa precisão gráfica. Por outro lado, a aquisição dos dados requer muito tempo, é trabalhosa e depende da qualidade do operador. Além de dificultar a simulação de resultados, é uma tecnologia bem mais cara em relação à representação em formato *raster*.

As estruturas *raster* apresentam a vantagem de serem mais simples, facilitando a aquisição dos dados, que dependerão apenas do equipamento e não mais do operador. Facilitam a análise e a simulação, a tecnologia é das mais baratas e vem se desenvolvendo muito rapidamente. Por outro lado, têm como desvantagens o grande volume de dados gráficos, problemas em relação aos tamanhos das células e à escala, e a baixa acurácia.

2.1.1.5 Interação entre a Cartografia e SIG

A Cartografia é o conjunto de estudos e operações científicas, artísticas e técnicas que intervêem a partir dos resultados de observações diretas ou da exploração de uma documentação existente, tendo em vista a elaboração e a preparação de plantas, mapas e outras formas de expressão, bem como a sua utilização. A Cartografia tem como objeto de estudo a distribuição espacial, a combinação e a interação dos fenômenos da natureza e da sociedade, bem como suas alterações através dos tempos, por um método de informação a base de um sistema especial de sinais, que constituem os símbolos cartográficos.

Mapas, plantas e cartas são formas visuais que permitem ao usuário ser criativo e pensar sobre os dados espaciais, servindo como meio gráfico que armazena dados e veicula informações. Os dados espaciais e seus atributos são armazenados na Base de Dados do SIG como uma representação cartográfica digital, no qual as coordenadas dos objetos ou fenômenos representados correspondem à localização absoluta deles com referência a um sistema de coordenadas geográficas do mundo real.

Além disso, é com o uso da Cartografia aplicada aos dados que serão resolvidos problemas do tipo: padronização de representações dos objetos do mundo real, uniformização de escalas, sistemas de projeções, georeferenciamento, efeitos da curvatura da Terra etc.

Portanto, os SIGs permitem a organização e a análise da informação geográfica levando a soluções mais rápidas e precisas para problemas relacionados ao comportamento espacial dos dados contidos no sistema, independentemente do modo como eles foram introduzidos no sistema ou de como eles serão apresentados.

2.1.2 Bancos de Dados Espaciais

Em BAUZER e PIRES e BOTELHO (1994) podemos encontrar uma definição precisa do que deve ser um banco de dados para GIS, e que pode ser também utilizada no contexto de Bancos de Dados Espaciais. Além disso, naquele trabalho também é apresentado uma lista de requisitos que os Bancos de Dados Espaciais devem atender para suportarem a construção de Sistemas de Informações Geográficas. Aqui abordaremos os BDEs sem nos restringir ao campo de aplicação de SIGs.

2.1.2.1 Definição

Em vários campos do conhecimento há uma demanda para a administração de dados com características pouco convencionais, como os dados geométricos, geográficos ou espaciais. Como exemplo, podemos citar modelos tridimensionais do cérebro humano, mapas da superfície terrestre, desenhos esquemáticos de cadeias de átomos em moléculas grandes, como DNA, proteínas ou mesmo polímeros, ou ainda projetos de circuitos integrados do tipo VLSI. Desde o advento da tecnologia de Banco de Dados têm-se tentado lidar com tais dados em sistemas gerenciadores de bancos de dados, embora o sucesso até aqui obtido tenha sido relativo. No entanto, conseguiu-se obter nessas áreas algumas das principais vantagens obtidas com o uso de um banco de dados em outras áreas: controle de acesso, segurança e integridade dos dados e recuperação contra falhas.

Uma característica que distingue tais sistemas dos sistemas de CAD (*Computer Aided Design*) é que o enfoque principal em tais sistemas é lidar com grandes coleções de polígonos relativamente simples, ao passo que sistemas de CAD em geral lidam com pequenos grupos de objeto muito complexos, tão complexos que em geral são hierarquicamente organizados para poderem ser manipulados.

Vários termos foram utilizados para designar sistemas de banco de dados que oferecem suporte a tal tipo de dados, tais como banco de dados pictoriais, de imagens, geográficos, geométricos, ou ainda espaciais. Segundo GÜTING (1994), até certo ponto o termo banco de dados espaciais se tornou popular devido à série de conferências SSD (*Symposium on Large Spatial Databases*), que vêm acontecendo bianualmente desde 1989, e está associado a uma visão de banco de dados contendo conjuntos de objetos no espaço, ao invés de imagens ou figuras de um determinado espaço.

Os requisitos para lidar com objetos no espaço que possuem identidade, extensão, localização e relacionamentos bem definidos são bastante diferentes dos requisitos para se lidar com imagens do tipo raster. Foi sugerido então que, para distinguir claramente as duas classes de sistemas, devemos chamá-los banco de dados espaciais e banco de dados de imagem, respectivamente (GÜNTHER e BUCHMANN, 1990). Neste trabalho, estaremos interessados apenas em banco de dados espaciais.

Assim, definiremos um banco de dados espacial como sendo um sistema de banco de dados que oferece suporte a tipos de dados espaciais tanto em seu modelo de

dados com em sua linguagem de consulta, implementando tais tipos e oferecendo pelo menos índices espaciais e algoritmos eficiente para junções espaciais.

2.1.2.2 Dados Armazenados em um BD Espacial

A principal área de aplicação que direciona os esforços da pesquisa em banco de dados espaciais é a área de SIG. Portanto, iremos considerar aqui a necessidade de alguns tipos de dados que, embora sejam utilizados em SIGs, certamente são úteis em outras áreas.

Dessa forma, os tipos de dados armazenados em um SIG servirão de requisito básica para os tipos de dados a serem armazenados em um banco de dados espaciais. Como estes tipos de dados estão relacionados com a modelagem do mundo real, que como vimos depende do enfoque dado à visão de mundo do usuário, temos que os tipos de dados necessários a um banco de dados espaciais estão intimamente relacionados a essa visão. Assim, bancos de dados espaciais devem suportar os seguintes tipos de dados: espaciais (espaço e os objetos nele contidos), pictoriais (imagens) não espaciais.

2.1.2.3 Abordagem Dual

Para uma melhor compreensão dos problemas com os quais nos depararemos mais adiante, vamos ter nessa seção uma visão geral de como é o modelo atualmente dominante nos banco de dados espaciais disponíveis comercialmente e quais as suas principais deficiências. Além disso, analisaremos como tais deficiências causam impacto na incorporação dos novos índices e técnicas para a realização de junções, justificando a elaboração de um novo modelo de implementação que satisfaça os principais requisitos para os bancos de dados espaciais, inclusive com a mudança de paradigma envolvida: de um modelo relacional para um modelo orientado a objetos.

Praticamente todos os sistemas comercialmente disponíveis utilizam algum banco de dados relacional, que pode ser proprietário ou não, para lidar com os atributos alfanuméricos. Já os atributos espaciais são usualmente armazenados em um formato proprietário, de modo que é necessário manter ligações específicas entre os dados espaciais e os dados não espaciais. Essa abordagem porém, conhecida como dual, possui um claro problema de desempenho: realizar operações de união em tabelas relacionais nunca se igualou ao desempenho apresentado tanto nos antigos modelos de banco de dados em rede e hierárquicos quanto nos recentes modelos orientado a objetos. A Figura 2-1 ilustra a separação entre dados espaciais e não espaciais na abordagem dual.

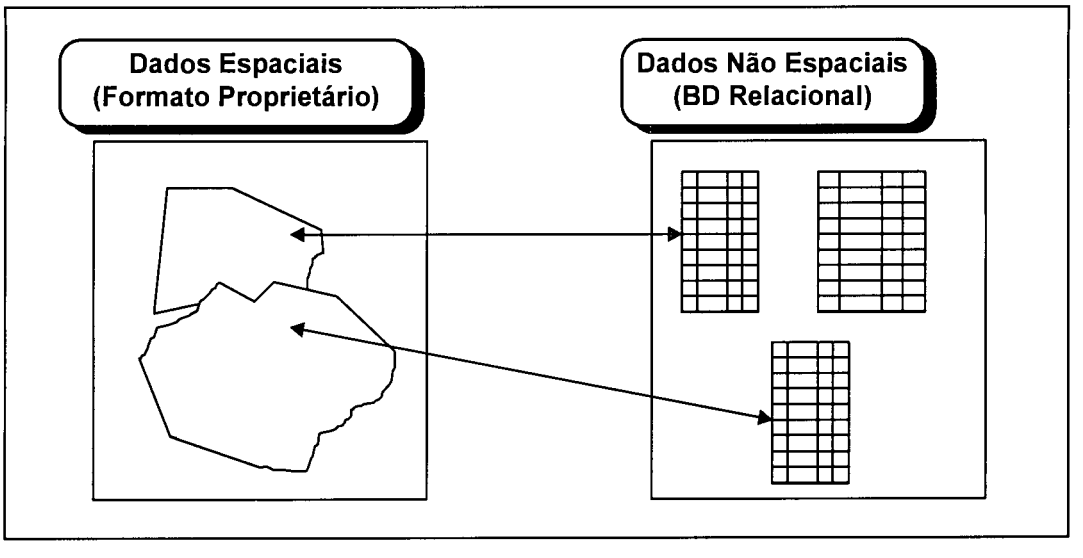


Figura 2-1 Separação de Dados na Abordagem Dual

Com a tecnologia de banco de dados orientado a objetos, tanto os atributos espaciais quanto os atributos não espaciais e seus índices podem ser armazenados no SGBD. No entanto, a maior oportunidade oferecida é a chance de se reprojetar toda a funcionalidade hoje presente em bancos de dados espaciais. Além disso, com o uso da orientação a objetos é possível se conseguir um tratamento uniforme e integrado para os dados espaciais nos dois formatos dominantes atualmente: raster e vetorial. Na seção 3.1 examinaremos com mais detalhes a abordagem orientada a objetos.

Cabe ainda ressaltar que os bancos de dados espaciais que utilizam a abordagem dual enfrentam problemas de desempenho com as bases atuais. É quase consenso na literatura (LANGRAN, 1992) que os SIGs no futuro deverão incorporar a dimensão temporal. A extensão de tais bancos para comportarem e tratarem a dimensão temporal das informações implicará em um agravamento desses problemas de desempenho, principalmente devido ao fato de que o volume de informações tende a crescer consideravelmente. Além disso, também não existem implementações de extensões satisfatórias para bancos de dados relacionais tratarem a dimensão temporal, o que implicaria em um esforço adicional para poder gravar as informações textuais temporalizadas. Portanto, o desenvolvimento dos bancos de dados espaço-temporais deverá romper com a atual abordagem dual, adotando um novo modelo de implementação. Ao que tudo indica, tal modelo será baseado na abordagem orientada a objetos.

Por outro lado, independentemente da abordagem utilizada para a implementação dos bancos de dados espaço-temporais orientados a objetos, será

necessário uma adequação das técnicas disponíveis atualmente para a realização das consultas em tais bases. De fato, as técnicas utilizadas para a realização de consultas em bancos de dados orientadas a objetos diferem bastante das técnicas utilizadas em bancos de dados relacionais. Por sua vez, tais técnicas são completamente diferentes das utilizadas para a realização de consultas espaciais, e diferem bastante das técnicas utilizadas em bancos de dados temporais. Além disso, devemos levar em conta que, de uma forma geral, o volume de dados de sistemas de informações geográficas e temporais é muito maior e sempre crescente. Por estes motivos, podemos afirmar que haverá a necessidade do desenvolvimento de novas técnicas (ou a adaptação/fusão das já existentes) para a realização de consultas em bancos de dados espaço-temporais.

Portanto, os trabalhos atualmente desenvolvidos para uma melhor realização de junções espaciais devem estar em conformidade com os possíveis cenários futuros expostos nos dois parágrafos anteriores: a mudança de paradigma relacional para orientado a objetos e a incorporação da dimensão temporal ao modelo de dados. Nesse sentido, devem apresentar o máximo de independência em relação à forma de armazenamento dos dados, principalmente aqueles espaciais; independência em relação aos índices espaciais e/ou temporais; e apresentar um bom desempenho para massas de dados muito grandes quando comparadas aos *buffers* disponíveis.

2.1.3 Mineração de dados

Novamente aprofundando a definição apresentada no capítulo anterior, iremos agora considerar alguns aspectos relevantes de mineração de dados e sua aplicação em Sistemas de Informações Geográficas.

2.1.3.1 Definição

Nas últimas décadas, o enorme crescimento das bases de dados comerciais, governamentais e científicas ultrapassou em muito a capacidade humana de interpretá-las. A partir deste fato, surge a necessidade de ferramentas e técnicas automatizadas para análise inteligente de grandes bases de dados. Como consequência temos o surgimento da área de mineração de dados (*data mining*).

O termo mineração de dados descreve o conceito de prospecção de conhecimento em banco de dados, do Inglês *knowledge discovery in databases - KDD* (SHAPIRO e FRAWLEY, 1991). O termo prospecção de conhecimento em bancos de dados foi formalizado em 1989 em referência ao conceito genérico de buscar

conhecimento de alto nível sobre os dados. O termo mineração de dados é, então, a aplicação em alto nível das técnicas e ferramentas usadas para apresentar e analisar dados para tomadores de decisões. Este termo, mineração de dados, tem sido usado pela comunidade de estatísticos, analistas de dados e MIS (Management Information Systems - Sistemas de Informações para Gerência), enquanto que o termo prospecção de conhecimento em banco de dados foi usado principalmente por pesquisadores da área de inteligência artificial e de aprendizagem de máquina.

2.1.3.2 A necessidade da mineração de dados

Algoritmos complexos usados em mineração de dados existem já há duas décadas. O governo dos EUA usou software de mineração de dados baseados em redes neurais, lógica nebulosa e reconhecimento de padrões para identificar fraudes em recolhimento de impostos e escuta de comunicações estrangeiras (DEPOMPE, 1996). Estas ferramentas, até agora, foram de domínio exclusivo das grandes corporações ou das agências governamentais, devido principalmente ao alto custo envolvido.

Os avanços em aquisição de dados científicos (por exemplo, sensoreamento remoto e satélites), o processamento de código de barras e as operações governamentais aumentaram muito o volume de dados. Somando-se isso às novas tecnologias de armazenamento de dados e o uso extenso de sistemas gerenciadores de banco de dados, a magnitude do volume de dados evolui dramaticamente. O projeto da criação do banco de dados do código genético humano (genoma) e as pesquisas de astronomia estão produzindo terabytes de dados. Sensoreamento remoto, imagens geradas por satélite e outras ferramentas são capazes de produzir 50 gigabytes de dados por hora (FAYAD et al., 1996).

Vários fatores combinados trouxeram a mineração de dados para a vanguarda de tomada de decisões em nível empresarial. Dentre eles, podemos citar:

- o valor intrínseco de grandes bases de dados
- o surgimento do conceito de data warehouse
- a redução dramática na relação custo/benefício de sistemas para armazenamento e processamento de dados.
- a intensa competição em esferas comerciais já saturadas
- a habilidade para manufatura, comercialização e propaganda direcionados a segmentos de mercado pequenos ou até mesmo indivíduos, e

- o mercado de produtos para mineração de dados está em franca expansão.

A tecnologia de mineração de dados é caracterizada por computações intensivas em grandes volumes de dados. Um grande poder de processamento é crítico, e paralelismo é a chave para capacitar mineração de dados em larga escala.

As companhias perceberam que o mais valioso recurso delas é a informação que elas possuem sobre os clientes, em particular sobre os seus padrões de compra. Aumentar a competitividade depende da qualidade da tomada de decisões e do aperfeiçoamento desse processo, utilizando-se das decisões antigas. Um melhor conhecimento sobre os clientes e sobre os mercados permitirá às empresas melhorar o direcionamento dos seus produtos e serviços. Por exemplo, varejistas podem direcionar para determinados clientes promoções de vendas específicas; companhias de telefones podem prever padrões de demanda, podem perfilar e podem segmentar grupos de cliente, melhorando a análise de rentabilidade; e, instituições financeiras podem obter informações para a segmentarização de produtos financeiros (DEPOMPE, 1996).

2.1.3.3 Mineração de Dados x Consultas Tradicionais

Consultas de banco de dados tradicionais contrastam com mineração de dados já que estas são simbolizados por perguntas simples como "qual o valor das vendas de suco laranja em janeiro de 1995 para a cidade X"?. A análise multidimensional, freqüentemente chamado de processamento analítico on-line (on-line analytical processing - OLAP), permite aos usuários realizar consultas muito mais complexas, tais como a comparação entre o volume real e o planejado de vendas em uma região durante determinado período. Novamente, a ênfase em ambos os casos, é que os resultados derivados são valores extraídos da agregação de dados existentes. Por outro lado, a mineração de dados através do uso de algoritmos específicos ou máquinas de busca tenta extrair padrões e tendências nos dados e deduzir regras a partir destes padrões. Com estas regras, o usuário pode então apoiar, revisar e examinar decisões em outros negócios relacionados (EDELSTEIN, 1996).

No contexto de SIGs, a mineração de dados assume importância estratégica para o desenvolvimento de uma região ou país. É possível, por exemplo, através de cuidadosa análise dos tipos de solo, vegetação e outras variáveis geográficas determinar as chances de ocorrência de determinados minerais no subsolo, ou qual cultura é a mais apropriada para determinada região. Esse tipo de inferência demanda uma grande

quantidade de informações cruzadas, ou seja, uma grande quantidade de junções entre vários tipos de conjuntos de dados espaciais diferentes.

2.2 Bancos de Dados Espaciais Orientados a Objetos

O conceito de programação orientado a objetos já está bem consolidado no mundo da computação, o que pode ser observado inclusive pela grande difusão do uso de linguagens orientadas a objetos, como o C++ (STROUSTRUP, 1990). Sistemas Gerenciadores de Banco de Dados Orientado a Objetos, embora também já tenham se tornado uma realidade (BANCILHON, 1992), ainda sofrem a falta de um modelo único comum apesar dos esforços de padronização (CATTELL, 1994). Porém na comunidade de usuários de SIGs tem havido uma aceitação surpreendentemente lenta dos conceitos de orientação para objetos. Esse fato ocorre porque até o momento nenhum dos produtos comercialmente disponíveis atingiu o nível de funcionalidade oferecido pelos sistemas tradicionais. Contudo, essa situação finalmente está mudando. Dessa forma, tentaremos manter nessa tese, nos algoritmos e melhoramentos propostos, sempre que possível, a independência do modelo de banco de dados: relacional ou orientado a objetos.

2.2.1 Arquiteturas Integradas

Para superar os problemas de desempenho decorrentes da arquitetura dual dos SIGs tradicionais começaram a ser pesquisados e desenvolvidos sistemas com uma arquitetura integrada. A abordagem utilizada foi a de desenvolver SIGs a partir de SGBDs Estendidos, que possuem suporte aos tipos de dados mais complexos presentes em tais aplicações. Dois importantes exemplos dessa abordagem são os SIG desenvolvidos sobre o Postgres (STONEBRAKER e KEMNITZ, 1991) e o Starburst (HAAS e CODY, 1991). Em particular, Starburst é a implementação da Arquitetura de Extensão da Gerência de Dados proposta em LINDSAY et al. (1987).

Tais exemplos servem para mostrar que, com tipos de dados e funções apropriadas definidos pelos usuário, um SIG bem flexível pode ser construído, capaz de lidar com tipos de dados bastante variados de uma forma uniforme. Já que tipo de dados abstratos, definidos pelo usuário, é um dos pilares da orientação a objetos, nada mais natural do que considerar que tais resultados também podem ser obtidos com Sistemas Gerenciadores de Banco de Dados Orientados a Objetos.

2.2.2 Smallworld

O sistema *Smallworld* (SMALWORLD, 1995), pode ser considerado o primeiro banco de dados espaciais orientado a objetos disponível comercialmente. Ainda que o armazenamento final dos dados seja feito em um banco de dados relacional, existe uma camada que abstrai e implementa todos os conceitos relativos a orientação a objetos. Além disso, a linguagem utilizada para a programação no sistema é uma variação de Smalltalk, sabidamente uma das linguagens orientadas a objetos mais puras.

Os relatos de casos de sistemas desenvolvidos utilizando o sistema *SmallWorld* são bastante animadores, servindo como exemplo encorajador para a utilização da tecnologia de orientação a objetos em sistemas de informações geográficas.

2.2.3 Tópicos Relevantes

O conceito de Banco de Dados Espaciais Orientado a Objetos ainda não está bem definido. Existem vários aspectos ou componentes necessários para podermos designar algum sistema como um arquétipo para SIG, cada um dos quais podendo ser implementado de várias formas diferentes. Porém, embora muitos sistemas comercialmente disponíveis usem o termo "orientado a objetos", poucos realmente o são, no significado atribuído ao termo no contexto de linguagens de programação. Dessa forma, é fundamental definir o que entendemos por um SIG orientado a objetos para evitar confusões devido ao mal uso desses termos.

2.2.3.1 Classificação dos SIGs

As classificações mais comuns para SIG versam basicamente sobre o tipo de dados armazenados: raster ou vetorial. No entanto, veremos aqui alguns aspectos mais abrangentes do que simplesmente a forma dos dados armazenados - veremos principalmente o modelo da realidade adotado pelo SIG.

Os SIGs, no que diz respeito ao modelo da realidade adotado, podem ser baseados na localização ou baseados em entidades.

I. Modelos Baseados na Localização

Esses modelos tratam a informação geográfica como coleções de distribuições espaciais, onde cada distribuição pode ser formalizada como uma função matemática que leva um conjunto (*grid*) de pontos geográficos no domínio de

um atributo. Padrões de altitude, temperatura, uso do solo e outros são enquadrados nessa categoria.

Mais especificamente, modelos baseados na localização assumem que o mundo é contínuo, formado por uma infinita quantidade de regiões. A posição de cada região é determinada por um sistema de coordenadas, sendo suas condições descritas para o sistema através de um determinado número de variáveis. Cada uma dessas variáveis pode ser encarada como uma camada que captura os valores dessa variável através da superfície da Terra, ou seja, funções que mapeam as coordenadas nos valores da variável.

É importante observar que, sob essa classificação, uma camada pode ser armazenada tanto no formato raster como vetorial. No formato raster, cada coordenada corresponde a uma célula que armazenará o valor da função naquele ponto. No formato vetorial, as linhas e polígonos desenhados representam mudanças de valor da função (curvas de nível, p.e.).

II. Modelos Baseados em Entidades

A idéia básica por trás desse modelo é a de que os humanos vêem o mundo como um espaço vazio povoado por vários tipos de entidades. As entidades não são construções artificiais, mas sim existem no mundo real e são parte fundamental de nosso modelo da realidade. Cada entidade é descrita em termos de seus atributos, e cada região pode conter nenhum, um ou vários objetos. Os modelos baseados em entidades podem ser subdivididos em centrados na geometria ou centrados nos objetos.

Nos modelos centrados na geometria, a classificação primária das entidades é feita de acordo com a sua forma geométrica primitiva: pontos, linhas, polígonos. Cada um desses tipos geométricos é então subdividido em classes que representam entidades do mundo real. Por exemplo, uma linha pode representar uma estrada, um rio ou um duto de gás, e ter atributos associados apropriados a cada caso.

Em contraste, nos modelos centrados em objetos a classificação primária das entidades é baseada no mundo real, de forma que uma entidade pode ser uma estrada ou uma escola. Cada entidade possui vários atributos, que podem ser geométricos ou não. Dessa forma, uma entidade pode ter várias representações geométricas, uma para cada escala: uma cidade pode ser um ponto ou uma região, por exemplo. Ainda,

esse modelo permite a representação tanto no formato raster quanto vetorial para a mesma entidade.

2.2.3.2 SIGs Realmente Orientados a Objetos

Podemos então agora estabelecer o que é um SIG orientado a objetos: é um SIG que adota o modelo Baseado em Entidades e seja Centrado em Objetos. Não importa muito, porém, se esse SIG lida com dados no formato raster ou vetorial: na verdade, seria desejável que lidasse com os dois formatos. Fundamental portanto para caracterizar um Sistema de Informação Geográfica como orientado a objetos é o seu modelo da realidade. Detalhes internos de implementação escapam dessa classificação, embora o uso de um banco de dados orientado a objetos com extensões espaciais para a implementação de um SIG orientado a objetos nos pareça a escolha óbvia, principalmente devido a homogeneidade e uniformidade dos conceitos envolvidos.

Conforme mencionado anteriormente, o primeiro banco de dados espacial orientado a objetos, voltado para o desenvolvimento de sistemas de informação geográfica, disponível comercialmente, o Smallworld, é implementado sobre um banco de dados relacionais. Mas apesar disso, esse aspecto se torna transparente para o desenvolvedor da aplicação, pois ele não necessita lidar diretamente com as tabelas - ao contrário, as tabelas são geradas automaticamente a partir das classes implementadas pelo usuário.

2.2.3.3 Vantagens

Em KIM (1995), temos uma ampla lista de vantagens dos banco de dados orientados a objetos sobre os banco de dados relacionais. Além dessas vantagens, um banco de dados espacial orientado a objetos terá algumas vantagens específicas sobre os congêneres relacionais. As vantagens que aqui são mencionadas foram relatadas em sistemas implementados com o auxílio do Smallworld e de um protótipo chamado OO-DNC (ARCTUR et al., 1995). É importante ressaltar também que o padrão SAIF (SAIF, 1994) para intercâmbio de dados foi totalmente desenvolvido utilizando-se tecnologia de orientação a objetos, e é composto de uma biblioteca de classes que deve servir de base para a construção de outros sistemas.

Diminuição da Impedância: a construção de sistemas em geral, e SIGs em particular, envolve a passagem por várias etapas que vão da análise à manutenção do sistema. Cada uma dessas etapas possui atividades distintas e formas de representação

também distintas. Abstrações são feitas em cada uma delas. No entanto, ao se cruzar as fronteiras entre uma etapa e outra, há o que se chama de descasamento ou impedância: conceitos e abstrações utilizados para representar o conhecimento sobre o sistema em uma etapa têm de ser modificados e adaptados à representação em outra etapa. Tais mudanças de representação introduzem muitas das vezes erros ou dificuldades que resultam no aumento de complexidade (e dos custos) do desenvolvimento dos sistemas. Hoje em dia é fato consumado que as técnicas de orientação para objetos diminuem a impedância por utilizarem uma notação e abstração uniforme ao longo dessas várias etapas de desenvolvimento. Logo, a modelagem de um fenômeno que é intrinsecamente orientado a objetos como sistemas de informação geográfica (especialmente quando se utiliza o modelo baseado em entidades) será realizada com menor impedância se forem empregados os conceitos de orientação a objetos ao longo de todo o desenvolvimento do sistema (WORBOYS, 1994).

Velocidade: devido ao fato de que os bancos de dados orientados a objetos não apresentam a arquitetura dual, que separa os atributos gráficos e espaciais dos atributos alfanuméricos, pode-se obter um ganho considerável na velocidade de execução. Além disso, com os recursos disponíveis em tais bancos de dados para o agrupamento (*clustering*) de estruturas de dados, que oferecem um grande controle por parte do usuário, muitas otimizações podem ser realizadas. Especificamente, em ARCTUR et al. (1995) temos relatos de operações realizadas de seis a quinze vezes mais rápidas.

Além das vantagens citadas no parágrafo anterior, temos também as seguintes:

1) uma grande estruturação da informação, devida à utilização direta dos conceitos de classe e herança;
2) encapsulamento de dados e a sobrecarga de operações facilita a manipulação dos dados e torna a representação física e lógica dos dados mais independente uma da outra;
3) decomposição de objetos em partes, muito usual em SIGs, é um dos conceitos que mais são beneficiados em técnicas de orientação a objetos, tanto do ponto de vista semântico quanto do ponto de vista da eficiência;
4) escalas e representações diversas para o mesmo objeto de uma maneira transparente para o usuário.

É importante ressaltar também que o padrão SAIF (SAIF, 1994) para intercâmbio de dados foi totalmente desenvolvido utilizando-se tecnologia de orientação

a objetos, e é composto de uma biblioteca de classes reutilizável que deve servir de base para a construção de outros sistemas.

2.3 Sumário

Este capítulo apresentou os conceitos necessários entendimento do restante desta Tese: Sistemas de Informações Geográficas e Bancos de Dados Espaciais e os tipos de dados armazeandos em tais bancos.

3. Consultas Espaciais e Junções

Este capítulo descreve com detalhes o problema que será tratado no capítulo seguinte desta tese e métodos eficientes para resolvê-lo: a junção espacial. Para tanto, abordaremos tópicos como índices espaciais, também chamados de métodos de acesso espaciais, e técnicas de junção espacial. Iremos também justificar a utilidade desse tipo de consulta, propor alguns exemplos, verificar o seu grau de dificuldade e mostrar algumas alternativas de implementação.

Não adotaremos nenhum modelo de linguagem de consulta espacial pois os padrões ainda não estão consolidados, sendo esse ainda um campo onde muito trabalho está sendo realizado. Além disso, presume-se que o modelo atual de realização de consultas utilizado em bancos de dados relacionais continuará valendo, modelo esse que prevê a decomposição de uma consulta em operações mais simples, envolvendo apenas um predicado, por meio de alguma álgebra ou cálculo.

Assim, este capítulo tem a seguinte estrutura: na seção 1 teremos uma breve introdução a determinadas características relevantes para o problema da tecnologia atual dos computadores. Na seção 2, apresentaremos as consultas espaciais consideradas mais importantes: restrição e junção. Na seção 3 iremos caracterizar as principais dificuldades e desafios encontrados na realização de tais consultas. Na seção 4 iremos estudar a realização das restrições. Na seção 5 veremos com mais detalhes a realização de junções, inclusive com a exposição e análise dos algoritmos existentes tanto para dados não espaciais quanto para dados espaciais. Finalmente, na seção 6 iremos apresentar o modelo corrente mais aceito para o processamento de junções espaciais bem como uma visão geral do estado da arte, com a citação dos trabalhos mais importantes.

3.1 Índices e Blocos

Iremos nesta seção explicar em detalhes algumas premissas importantes para o projeto de estruturas de dados e algoritmos. Em particular, iremos ver a necessidade de se utilizar blocos para organizar a informação nos índices. Além disso, para a melhor compreensão dos algoritmos que iremos propor, apresentaremos algumas propriedades e características das tecnologias atuais que são utilizadas para a computação sequencial[1].

[1] Nesta tese, não consideraremos computadores que operem em paralelo ou com processamento distribuído.

Discos magnéticos são o meio de armazenamento por excelência dos bancos de dados. Uma característica importante dos discos é sua divisão em setores agrupados em trilhas e a existência de uma (ou mais) cabeças de leitura/gravação da informação apoiadas sobre um mecanismo chamado de *braço*. O disco permanece girando sobre o braço, que é móvel, e pode ser posicionado sobre cada uma das trilhas em que o disco está dividido. Esse posicionamento é a operação que consome mais tempo quando se faz um acesso a alguma informação contida no disco. As outras operações envolvidas são chamadas de latência (espera pelo setor correto passar sobre a cabeça do braço) e a leitura/escrita propriamente dita. Essa é uma tecnologia em constante evolução, de forma que os discos atuais são mais rápidos do que memórias principais de algumas décadas atrás.

Em geral, gasta-se mais tempo para posicionar o braço de um disco na posição correta de leitura (posicionamento + latência) do que para se ler um bloco de 1 Kb, e o tempo gasto na leitura de um bloco de 1 K será virtualmente o mesmo para se ler um bloco de 2 K. De fato, com tais parâmetros torna-se vantajosa a leitura de blocos grandes do disco — o limite será o tamanho de bloco que irá caber em uma mesma trilha de modo que não seja necessário outro posicionamento. Este, contudo, é um valor típico da construção da unidade de disco, e que depende de outros parâmetros tais como: quantas cabeças possui a unidade, quantas superfícies de disco, a densidade da superfície magnética e assim por diante. No entanto, apenas para ilustração, supondo-se que a soma dos tempos de posicionamento e latência seja de 3 ms, se forem lidos 32 Kb a 80 Mb/s iremos acrescentar algo em torno de 0,4 ms — aproximadamente 12% do tempo total gasto no acesso ao disco.

Por outro lado, uma característica comum dos discos é que quanto menor o deslocamento do braço entre a origem e o destino, menor será o tempo gasto no posicionamento, embora essa relação não seja linear devido em parte à inércia e em parte a limitações tecnológicas. De qualquer maneira, é sempre mais rápido ler blocos no disco que residam em trilhas próximas do que ler blocos distantes várias trilhas um do outro. É importante notar que o conceito de proximidade não é relativo aos blocos mas sim às trilhas: na mesma trilha existem setores diametralmente opostos, no entanto o acesso a um deles a partir do outro é usualmente mais rápido do que mudar de trilha. O nome dado a esse padrão de acesso, feito em trilhas próximas, é chamado de sequencial — pelo fato de que, em geral, dados gravados sequencialmente estarão em

trilhas sequenciais. Na verdade, essa proximidade irá depender do sistema operacional e de sua política de gravação em disco. No entanto, justamente para aproveitar essa característica dos discos, virtualmente todos os sistemas operacionais procuram armazenar sequencialmente no disco os dados que forem enviados sequencialmente.

Já a memória principal, por não possuir partes mecânicas, funciona à velocidade máxima o tempo inteiro, o que significa dizer que a localização física perde a importância no tempo gasto para recuperar a informação. De fato, esse tempo é constante e muito pequeno quando comparado ao acesso ao disco. Dessa forma, o que irá determinar o tempo de acesso é a quantidade de informação a ser manipulada pela CPU.

É precisamente essa característica intrínseca dos discos, o acesso vantajoso de blocos relativamente grandes de informação e que residam próximos (em trilhas próximas), que vem norteando o desenvolvimento dos algoritmos e estruturas de dados que fazem uso de memória secundária. Assim, enquanto que em geral os algoritmos feitos para memória principal trabalham com unidades pequenas de informação, usualmente um objeto de cada vez, os algoritmos que trabalham com disco tendem a lidar com grandes blocos contendo vários objetos para tirar melhor proveito das características de acesso dos discos. Além disso, alguns algoritmos e estruturas foram projetados com uma certa preocupação em permitir o acesso sequencial à informação sempre que isso for útil, pois como vimos o acesso sequencial é quase sempre mais rápido do que o acesso não sequencial.

Em muitos casos, como veremos no restante desta Tese, o uso de índices acelera o processamento de consultas. No caso específico de consultas espaciais, uma grande gama de estruturas de indexação foi desenvolvida a partir das chamadas R-Trees.

3.1.1 R-Trees

As R-Trees e suas variantes são estruturas que dividem o espaço em retângulos, correspondendo a blocos no disco, os quais podem possuir sobreposição. Uma exceção é a R^+-Tree, a qual iremos considerar mais adiante. Essa sobreposição poderá ocorrer em qualquer nível da árvore, desde os nós mais internos até as folhas.

A estrutura interna de uma R-Tree é bastante simples. Existem dois tipos de nós: internos e folhas. Cada nó irá corresponder a um bloco no disco. Cada nó interno possui entradas para outros nós, que como em toda estrutura em árvore são chamados de

filhos. A cada nó é associado um retângulo, que é o menor retângulo que englobe os retângulos de todos os seus filhos. Cada nó folha contém uma certa quantidade de MBRs do conjunto de dados, sendo que também possuem um retângulo associado cobrindo todos os seus MBRs. A Figura 3-1 ilustra uma R-Tree.

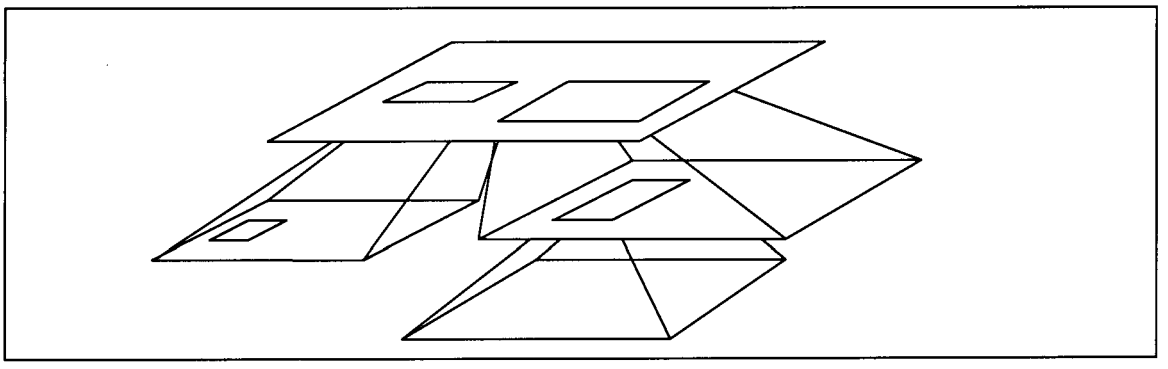


Figura 3-1 - Uma R-Tree

Seja então M o número máximo de entradas em um nó e $m \leq M / 2$ um parâmetro identificando o número mínimo de entradas em um nó, seis propriedades são propostas como descrito a seguir:

R1. Cada nó folha contém entre m e M registros de índice a menos que seja um nó raiz;

R2. Para cada registro de índice < *MBR*, *identificador de tupla* > em um nó folha, *MBR* é o menor retângulo que espacialmente contém os objetos de dados *n*-dimensionais representados pela tupla indicada;

R3. Cada nó interno contém entre m e M filhos ao menos que seja um nó raiz;

R4. Para cada entrada < *MBR*, *ponteiro para nó filho* > em um nó interno, *MBR* é o menor retângulo que espacialmente contém todos os retângulos do nó filho;

R5. O nó raiz tem pelo menos dois filhos a menos que seja uma folha;

R6. Todas as folhas aparecem no mesmo nível da árvore.

Uma árvore contendo N registros de índice com essas características acima descritas possui altura de no máximo $\lfloor \log_m N \rfloor - 1$, o que mantém uma boa eficiência assintótica de tempo e utilização de espaço. Detalhes da ocupação da estrutura que garantem a sua eficiência e os algoritmos associados podem ser encontrados em GUTTMAN (1984) e BECKMANN et al. (1990).

3.2 Consultas Espaciais

Nesta seção iremos apresentar de forma genérica os diversos tipos de consultas espaciais, ressaltando quais resultam em junções espaciais. O nosso modelo de consulta básica envolve a seleção de objetos pertencentes a uma ou mais coleções, possivelmente à coleção de todos os objetos da base, com restrições especificadas sobre os atributos espaciais — é o que chamaremos de predicado de seleção espacial. Ou seja, estará em questão apenas a operação de seleção de objetos espaciais baseada tão somente nas suas propriedades geográficas e geométricas. Não estudaremos as consultas sobre os atributos não espaciais por considerarmos que essa é uma área de pesquisa já bastante explorada e independente de bancos de dados espaciais.

3.2.1 Metodologia de Processamento para Consultas Relacionais

Para mostrar o nosso modelo de processamento de consultas espaciais, iremos antes fazer algumas considerações sobre como são implementadas, de forma eficiente, as consultas em bancos de dados relacionais. Uma abordagem simplificada, segundo DATE (1991), seria a seguinte:

1) Converter a consulta em uma representação interna: as consultas realizadas nos bancos de dados relacionais são em geral especificadas em uma linguagem de consulta estruturada, chamada simplesmente de SQL. A SQL é uma linguagem declarativa, que apenas descreve as características do conjunto resposta, não se preocupando com os passos necessários para obtê-lo. A consulta especificada em SQL é então traduzida para uma representação interna, em geral uma *árvore de sintaxe abstrata* ou *árvore de consulta*, que contudo terá um equivalente em álgebra e cálculo relacional.
2) Conversão à forma padrão: algumas manipulações algébricas são então realizadas para tornar a execução da consulta mais eficiente, utilizando-se para isso certas regras de transformação bem definidas. Um exemplo típico é a transformação de "(A JOIN B) WHERE *restrição-sobre*-A AND *restrição-sobre*-B" para a expressão equivalente porém quase sempre mais eficiente "(A WHERE *restrição-sobre*-A) JOIN (B WHERE *restrição-sobre*-B)".
3) Escolha dos procedimentos de nível inferior: nesse estágio, considerações como a existência de índices, o agrupamento físico dos registros, a

distribuição dos valores, etc, são levados em conta para a escolha das operações de mais baixo nível (junção, restrição etc) que serão utilizadas para a implementação da consulta.

4) Geração e escolha do melhor dos planos de execução da consulta: o estágio final do processo de otimização envolve a construção de um conjunto de planos de consulta candidatos, seguido da escolha do melhor. Cada plano é construído pela combinação de um conjunto de procedimentos de implementação candidatos. Certamente haverá diversos planos, que deverão ser avaliados segundo alguma função de custo, e a tarefa de escolher o melhor pode ser mais custosa do que a própria consulta. Este é um tema fortemente pesquisado atualmente, no entanto, algumas estratégias já existem e são amplamente utilizadas. Assim, após a escolha do plano que se supõe melhor, a consulta é enfim realizada.

3.2.2 Metodologia de Processamento para Consultas Espaciais

Devemos lembrar que o sucesso dos bancos de dados relacionais pode ser em parte creditado ao forte suporte teórico fornecido pelo par álgebra e cálculo relacional, que formam a essência do modelo relacional. Além disso, pelo que temos observado dos esforços atuais, podemos realizar duas suposições sobre o padrão de linguagem de consulta espacial a ser estabelecido e adotado (SQL3, 1995, SAIF, 1994):

1) Será teoricamente sustentado por algum par de álgebra e cálculo que desempenhará papel equivalente ao desempenhado pelo par álgebra e cálculo relacional nos atuais bancos de dados relacionais.
2) Será parecido o suficiente com o atual SQL para permitir que uma estratégia de decomposição de consultas complexas em operações primitivas e a sua otimização através de manipulações algébricas sobre as mesmas, semelhante à que é usada nos bancos relacionais, possa ser adotada.

Portanto, supomos que iremos realizar consultas já simplificadas e tidas como provavelmente as mais eficientes. Não estaremos pois interessados em métodos de decomposição de consultas nem de otimização e de geração de planos de consulta — iremos assumir que tal trabalho já foi feito em uma etapa anterior. Estaremos então lidando com os procedimentos primitivos de nível inferior que servirão para implementar os diversos tipos de operações.

Além disso, o conjunto resultado de uma operação não deverá ser necessariamente materializado, ou seja, muitas vezes bastará que os objetos sejam identificados, especialmente quando esse conjunto vier a ser utilizado como entrada de uma outra operação. De fato, dificilmente será atraente criar uma tabela para conter um conjunto resposta devido ao fato de que os dados espaciais ocupam muito espaço.

Finalmente, as suposições que estamos assumindo estão coerentes com o que é proposto nos recentes trabalhos de PAPADIAS et al. (1999a, 1999b) e MAMOULIS e PAPADIAS (1999). De fato, estes trabalhos, através do uso de funções de custo (HUANG et al., 1997a) para as junções espaciais, apresentam modelos que permitem realizar a otimização das consultas de forma análoga ao modelo relacional: geração de vários planos alternativos de consulta e tentativa de seleção do melhor deles.

3.2.3 Operações Primitivas de Um BD Espacial

Por analogia com a álgebra relacional podemos citar cinco operações primitivas (DATE, 1991): restrição, projeção, produto, união e diferença. A junção, embora possa ser expressa como o resultado de uma restrição aplicada a um produto, é útil o bastante para ser suportada diretamente, principalmente por razões de eficiência. Na mesma situação estão as operações de divisão e interseção. No entanto, para os bancos de dados espaciais, de especial interesse será a implementação das operações de restrição e de junção, isto porque irão diferir fundamentalmente das implementações relacionais. As outras operações são facilmente implementadas tomando-se como base a operação equivalente relacional, como podemos ver na Tabela 3-1:

Tabela 3-1 - Operações a Serem Implementadas por um BD espacial

Operação	Descrição	Implementação
Restrição	Extrai elementos de um conjunto que satisfaçam a uma condição.	Específica, dada a natureza espacial da condição de restrição.
Junção	Conjunto formado pelos pares do produto cartesiano dos elementos de dois conjuntos que satisfaçam a uma condição de restrição (os conjuntos podem ser idênticos).	Específica, dada a natureza espacial da condição de restrição.
Projeção	Extrai atributos específicos dos elementos de um conjunto.	Semelhante à implementação relacional.
Produto	Produto cartesiano dos elementos de dois conjuntos (os conjuntos podem ser idênticos).	Semelhante à implementação relacional.
União	Dados dois conjuntos, é o conjunto formado pelos elementos que pertencem a pelo menos um dos dois conjuntos dados.	Semelhante à implementação relacional.
Diferença	Dados dois conjuntos, é o conjunto formado pelos elementos que pertencem ao primeiro conjunto e que não pertencem ao segundo.	Semelhante à implementação relacional.
Interseção	Dados dois conjuntos, é o conjunto formado pelos elementos que pertencem aos dois conjuntos.	Semelhante à implementação relacional.

É importante observar que somente as operações de restrição e de junção lidam com condições ou expressões que envolvem os atributos espaciais. As demais operações são definidas sobre conjuntos, sem considerar valores específicos de nenhum atributo mas sim o elemento como um todo. Disso resulta a total compatibilidade de tais operações primitivas tanto para bancos de dados relacionais quanto para bancos de dados espaciais. A interseção citada na Tabela 3-1, por exemplo, não é a interseção atributos espaciais, tais como polígonos, mas sim a interseção de conjuntos — ou seja, avalia-se a existência de um mesmo elemento em conjuntos diferentes.

Portanto, iremos nas subseções seguintes apresentar formalmente a definição das operações primitivas de restrição e de junção para bancos de dados espaciais. Além disso, definiremos também o que é um índice sobre um conjunto. Consideraremos os elementos de um conjunto como n-uplas, ou tuplas, formadas por atributos (a_1, a_2, ... a_n) $\in$ $\boldsymbol{C}$, onde $\boldsymbol{C}$ é o produto cartesiano $A_1 \times A_2 \times \ldots \times A_n$, tais que $a_i \in A_i \; \forall \; i$, $1 \leq i \leq n$, onde cada A_i representa o domínio do i-ésimo atributo a_i da tupla, A_i não necessariamente distinto de A_j. Note que essa definição de conjunto de tuplas é

compatível tanto com o modelo de banco de dados relacional como com o modelo de banco de dados orientado a objetos. De fato, a única ressalva que temos a fazer é que A_i pode ser um conjunto de natureza espacial, como por exemplo um conjunto de polígonos.

3.2.3.1 Operação Primitiva de Restrição

Iremos agora definir formalmente a operação primitiva de restrição como uma função que recebe como parâmetros um conjunto de tuplas e um predicado de restrição, e retorna um subconjunto de tuplas do mesmo tipo da entrada que satisfazem ao predicado dado. Para tanto, iremos antes definir o que é um predicado de restrição. Além disso, chamaremos de *bool* ao conjunto *booleano*, e utilizaremos a sua definição usual.

Definição 3.1 - predicado de restrição : Seja t uma tupla $(a_1, a_2, \ldots a_n) \in \boldsymbol{C} = A_1 \times A_2 \times \ldots \times A_n$, tal que $a_i \in A_i \; \forall \; i$, $1 \leq i \leq n$. Chamamos de *predicado de restrição sobre* $\boldsymbol{C}$, ou simplesmente *predicado de restrição*, a qualquer função do tipo p: $\boldsymbol{C} \rightarrow bool$, ou seja, uma função que dada uma tupla $t \in \boldsymbol{C}$, associam à mesma um valor *verdadeiro* ou *falso*.

Dado um predicado de restrição, chamaremos aos atributos envolvidos no mesmo de *atributos de restrição*. Quando o contexto assim o permitir, escreveremos apenas predicado. Se pelo menos um dos atributos de restrição de um predicado for de natureza espacial teremos um *predicado de restrição espacial*. Neste trabalho iremos considerar apenas os predicados de restrição envolvendo um único atributo, que será espacial.

Definição 3.2 - operador de restrição: Seja $\boldsymbol{C} = A_1 \times A_2 \times \ldots \times A_n$ o conjunto das tuplas $t = (a_1, a_2, \ldots a_n)$ tais que $a_i \in A_i \; \forall \; i$, $1 \leq i \leq n$, e seja $\boldsymbol{P}$ o conjunto de predicados de restrição sobre $\boldsymbol{C}$. Chamamos de operador de restrição a qualquer função *Res*: $\boldsymbol{C} \times \boldsymbol{P} \rightarrow \boldsymbol{C}$, definida por: *Res*: $(\boldsymbol{A} , p) \rightarrow \boldsymbol{B}$, onde $p \in \boldsymbol{P}$ e $\boldsymbol{B} \subseteq \boldsymbol{A}$ são conjuntos de tuplas t, com $t \in \boldsymbol{C}$, tais que $\forall$ tupla $t \in \boldsymbol{B}, p(t) =$ *verdadeiro* e $\forall$ tupla $t \in \boldsymbol{A} - \boldsymbol{B}, p(t) =$ *falso*.

Chamaremos de operação de restrição espacial à aplicação de um operador de restrição cujo predicado seja um predicado de restrição espacial. Além disso, chamaremos de operação de restrição sobre um atributo x à operação de restrição cujo predicado de restrição envolva apenas o valor do atributo x, onde x é um atributo espacial ou não.

3.2.3.2 Operação de Junção

Conforme já mencionamos anteriormente, a operação de junção não é uma operação primitiva. No entanto, tendo em vista a realização eficiente da operação de junção, é vantajoso ter uma implementação específica para a mesma. Por esse motivo, iremos agora definir formalmente a operação de junção como uma função que recebe como parâmetros dois conjunto de tuplas e um predicado de junção, e retorna o conjunto de tuplas pertencentes ao produto cartesiano dos conjuntos de entrada que satisfaçam ao predicado dado. Para tanto, iremos antes definir o que é um predicado de junção.

Definição 3.3 - predicado de junção: Sejam t e v tuplas tais que $t = (a_1, a_2, \ldots a_n) \in \boldsymbol{C} = A_1 \times A_2 \times \ldots \times A_n$, onde $a_i \in A_i \; \forall \; i, \; 1 \leq i \leq n$, e $v = (b_1, b_2, \ldots b_m) \in \boldsymbol{D} = B_1 \times B_2 \times \ldots \times B_m$, onde $b_j \in B_j \; \forall \; j, \; 1 \leq j \leq m$. Chamamos de *predicado de junção sobre* $\boldsymbol{C} \times \boldsymbol{D}$ ou simplesmente *predicado de junção*, a qualquer função p: $\boldsymbol{C} \times \boldsymbol{D} \rightarrow$ *bool*, ou seja, funções que dada uma tupla (t, v), $t \in \boldsymbol{C}$ e $v \in \boldsymbol{D}$, associam à mesma um valor *verdadeiro* ou *falso*.

De forma análoga ao predicado de restrição, dado um predicado de junção, chamaremos aos atributos envolvidos no mesmo de *atributos de junção*. Quando o contexto assim o permitir, escreveremos apenas predicado. Se pelo menos um dos atributos de junção de um predicado for de natureza espacial teremos um *predicado de junção espacial*. Nesse trabalho iremos considerar apenas os predicados de junção envolvendo atributos espaciais.

Definição 3.4 - operador de junção: Sejam $\boldsymbol{C} = A_1 \times A_2 \times \ldots \times A_n$ o conjunto das tuplas $t = (a_1, a_2, \ldots a_n)$ tais que $a_i \in A_i \; \forall \; i, \; 1 \leq i \leq n$, $\boldsymbol{D} = B_1 \times B_2 \times \ldots \times B_m$ o conjunto das tuplas $v = (b_1, b_2, \ldots b_m)$ tais que $b_j \in B_j \; \forall \; j, \; 1 \leq j \leq m$, e $\boldsymbol{P}$ o conjunto de predicados de junção sobre $\boldsymbol{C} \times \boldsymbol{D}$. Chamamos de operador de junção a qualquer função *Jun*: $\boldsymbol{C} \times \boldsymbol{D} \times \boldsymbol{P} \rightarrow \boldsymbol{C} \times \boldsymbol{D}$ definida por *Jun*: ($\boldsymbol{A}$, $\boldsymbol{B}$, p) $\rightarrow \boldsymbol{E}$, onde $p \in \boldsymbol{P}$, $\boldsymbol{A}$ é um conjunto de tuplas t, com $t \in \boldsymbol{C}$, $\boldsymbol{B}$ é um conjunto de tuplas v, com $v \in \boldsymbol{D}$, e $\boldsymbol{E}$ é um conjunto de tuplas $u \in \boldsymbol{A} \times \boldsymbol{B}$, tais que $\forall$ tupla $u \in \boldsymbol{E}$, $p(u) =$ *verdadeiro* e $\forall$ tupla $u \in \boldsymbol{A} \times \boldsymbol{B} - \boldsymbol{E}$, $p(u) =$ *falso*.

Chamaremos de operação de junção espacial à aplicação de um operador de junção cujo predicado seja um predicado de junção espacial. Além disso, chamaremos de operação de junção sobre os atributos x e y à operação de junção cujo predicado de

junção envolva apenas os valores dos atributos x e y, onde ambos são atributos espaciais ou ambos não o são, e x pertence à tupla t e y pertence à tupla v.

Conceitualmente, para realizar a operação de junção devemos verificar para quais tuplas $p(u)$ é *verdadeiro, onde* u = $(t, v) \in \boldsymbol{A} \times \boldsymbol{B}$. Ou seja, devemos calcular o produto cartesiano $\boldsymbol{A} \times \boldsymbol{B}$ e aplicar p a cada um de seus elementos, colocando no conjunto resposta todas as tuplas u para as quais $p(u)$ for *verdadeiro*. Conforme já mencionado, se a operação de junção for descrita nestes termos é fácil observar que a mesma pode ser definida como uma restrição sobre o conjunto formado pelo produto cartesiano dos conjuntos de entrada $\boldsymbol{A} \times \boldsymbol{B}$, onde o predicado de junção se tornaria então um predicado de restrição.

Por outro lado, é possível que o predicado de junção p seja tal que se possa determinar quais os elementos $u = (v, t)$, $v \in \boldsymbol{A}$ e $t \in \boldsymbol{B}$, para os quais $p(u)$ é verdadeiro, sem que seja necessário verificar todos os pares do produto cartesiano $\boldsymbol{A} \times \boldsymbol{B}$. Possivelmente esta forma de se realizar a operação de junção será mais eficiente do que a mencionada no parágrafo anterior, especialmente quando o conjunto resposta possuir um número de elementos bem menor do que o produto $\boldsymbol{A} \times \boldsymbol{B}$. De fato, essa é a justificativa para definirmos e implementarmos a operação de junção independentemente das operações de restrição e de produto.

3.2.3.3 Índices

Iremos definir um índice sobre um conjunto de tuplas como sendo uma projeção sobre um ou mais atributos, mantida em uma estrutura de armazenamento especial, e que contém também informações sobre a localização da tupla que permitam sua recuperação. Chamaremos de *atributos indexados* aos atributos utilizados na projeção. Um índice deverá ter informação que permita recuperar uma determinada tupla pelo valor do atributo indexado: fornecendo-se um conjunto de valores de busca para os atributos indexados, um conjunto de tuplas deve ser retornado. Para cada tupla pertencente ao conjunto retornado, o valor de seu atributo indexado deve estar contido no conjunto de valores de busca. A princípio, o conjunto retornado pode ser vazio, unitário ou ainda conter vários elementos.

De uma forma geral, um índice permite o acesso e recuperação de uma tupla específica pelo valor do atributo em tempo sublinear, ou seja, menor do que $O(n)$, onde

n é o número de elementos no conjunto. Além disso, o índice pode possuir alguma forma de ordenação envolvendo o(s) atributo(s) indexado(s).

Para uso em bancos de dados, os tipos mais comuns de índices são as árvores B e as tabelas de *hash*. As árvores B são ordenadas e possuem tempo de acesso médio da ordem de $O(\ log\ n\ )$, enquanto que as tabelas de *hash* não são ordenadas e possuem tempo de acesso médio da ordem de $O(\ 1\ )$, ou seja, constante.

3.3 Dificuldades na Realização de Consultas Espaciais

Nesta seção iremos apresentar algumas das dificuldades inerentes à realização de consultas espaciais. As mais importantes são: a complexidade dos atributos espaciais, a complexidade das operações sobre atributos espaciais, o grande volume de dados, a dificuldade na implementação de índices espaciais eficientes e a ausência de funções de ordenação total para atributos espaciais que preservem a proximidade.

3.3.1 Atributos Espaciais

Consideraremos como atributos espaciais aos atributos utilizados para representar dados espaciais do tipo ponto, linha e região. Em particular, nesta Tese estaremos interessados em regiões representadas por polígonos. A própria natureza dos atributos espaciais certamente representa uma dificuldade para a realização de consultas. No caso específico dos polígonos, usualmente teremos listas de vértices que não têm um tamanho conhecido *a priori*. Na prática, poderíamos até determinar um número máximo para o número de vértices de cada polígono, porém esse número poderia ser muito grande, bem como a razão entre o polígono com o maior e o menor número de vértices. Qualquer que seja a representação utilizada, a quantidade de informação será variável, o que inevitavelmente levará também a um tamanho variável para tais atributos.

É certo que podemos ter atributos não espaciais com tamanho variável, tais como atributos de texto longos, porém tais atributos são a exceção e não a regra. Além disso, não há muitas possibilidades de predicados envolvendo tais tipos de atributos, o que restringe o uso de consultas *ad hoc* sobre os mesmos, tornando assim a sua implementação mais direcionada para a aplicação.

3.3.2 Operações sobre Atributos Espaciais

Uma outra dificuldade que surge em qualquer tipo de consulta espacial está relacionada com a natureza complexa dos atributos espaciais. Assim, em uma consulta sobre um atributo não espacial estaremos realizando operações relativamente simples

tais como: igualdade, pertinência em faixa de valores, operações aritméticas, ou ainda operações sobre datas, cadeias de caracteres e outras. No entanto, os operadores espaciais, definidos sobre uma representação espacial de algum objeto, que no nosso caso específico são polígonos com muitas arestas, são intrinsecamente mais complexos do que os operadores relacionais.

Apenas como exemplo, o conhecido algoritmo de *plane-sweep* (PREPARATA e SHAMOS 1985, BRINKHOFF et al., 1994), que serve para determinarmos se existe interseção entre dois polígonos, possui complexidade $O(n \log n)$, onde n é a soma do número de vértices dos polígonos dados. Vários operadores espaciais muito úteis, que aparecem com frequência em aplicações de SIGs, são implementados através da interseção de polígonos. Além disso, existem outras operações que podem ser ainda mais complexas.

Enquanto que nas consultas sobre atributos não espaciais o tempo gasto para calcular o predicado de uma operação é mínimo quando comparado ao tempo gasto para recuperar a(s) tupla(s) envolvidas do disco, nas consultas com predicados espaciais isso pode não ser verdadeiro. Dessa forma, em muitos algoritmos, é necessário levar em conta não somente as operações de E/S, mas também o tempo de CPU gasto no processamento de tais operadores espaciais. Em BRINKHOFF (1994) temos exemplos de algoritmos onde o peso do cálculo de interseções de polígono no tempo total de uma consulta chega a ser maior do que o peso do tempo gasto nas operações de E/S.

3.3.3 Volume de Dados

Uma base espacial com o mesmo número de objetos que uma base convencional tende a ocupar muito mais espaço em disco. Isso ocorre porque, como vimos, os atributos espaciais são intrinsecamente maiores do que os atributos mais comuns encontrados nos bancos de dados convencionais. Por outro lado, ao se recuperar um objeto espacial no disco possivelmente várias operações de leitura terão de ser realizadas — o que certamente se constituirá em uma operação bem mais custosa do que buscar uma simples tupla em um banco de dados relacional, por exemplo.

3.3.4 Índices Espaciais

Encontraremos dificuldades também na implementação dos índices: para os atributos não espaciais devemos ter índices não espaciais, como as B-Trees (BAYER e MCCREIGHT, 1972, COMER, 1979); para os atributos espaciais devemos ter índices

espaciais, como as R-Trees (GUTTMAN, 1984). Conforme apontamos anteriormente, os objetos espaciais possuem tamanho variável, possivelmente muito grande. Dessa forma, a implementação eficiente dos índices espaciais é em geral feita utilizando-se aproximações, tais como os retângulos mínimos envolventes (MBR, do inglês *minimum bounding rectangle*) e ponteiros em disco para os atributos espaciais propriamente ditos. Com essa indireção diminui-se muito a eficiência do índice espacial, pois em geral a informação presente no mesmo não é suficiente para a avaliação completa de um predicado espacial. Em outras palavras, um índice espacial dificilmente será uma projeção do atributo indexado.

3.3.5 Função de Ordenação Total

Finalmente, os dados espaciais não possuem uma função de ordenação total que preserve a proximidade espacial. Embora existam técnicas baseadas em curvas de preenchimento do espaço (SAMET, 1990, GAEDE, 1995) que permitam a definição de funções de ordenação total sobre os conjuntos de dados espaciais, tais funções não preservam uniformemente a proximidade espacial, de modo que durante a ordenação dois objetos adjacentes no espaço podem ser mapeados bem distantes no disco.

3.4 Realização de Restrições

Nesta seção iremos mostrar como pode ser implementada a operação primitiva de restrição espacial. Para tanto, mostraremos como pode ser implementada a operação de restrição sobre atributos não espaciais nos banco de dados relacionais, e identificaremos o que puder ser implementado de forma análoga para os atributos espaciais.

3.4.1 Restrições sobre Atributos não Espaciais

A realização de uma restrição envolve a avaliação de um predicado para um determinado conjunto de tuplas. Dessa forma, a entrada do algoritmo para a realização de uma operação de restrição é um conjunto de tuplas e o predicado de restrição. Além disso, esse conjunto tanto pode ter existência permanente quanto temporária, como por exemplo o resultado de outra consulta.

O predicado de restrição para atributos não espaciais pode ser arbitrariamente complexo: basta lembrar que qualquer função p que receba uma tupla e retorne um valor booleano preenche os requisitos de nossa definição. No entanto, a complexidade do predicado poderá ser limitada pela linguagem de consulta. SQL, por exemplo, embora

permita as operações aritméticas usuais, não provê muitas funções matemáticas. Funções para manipulação de cadeias de caracteres também não são de forma alguma abundantes. Por outro lado, quanto mais simples o predicado de uma consulta, mais chances esta possuirá de tirar proveito do otimizador de consultas.

Vamos nos deter com mais detalhes no caso em que o predicado p envolve apenas um atributo, já que iremos considerar também apenas um atributo no caso espacial. Quando o predicado envolve apenas um atributo, digamos a_i, podemos ter duas situações: o atributo está ou não indexado.

Se o atributo está indexado, o predicado p pode ser calculado diretamente sobre o índice e apenas as tuplas relevantes serão incluídas na resposta, não havendo pois a necessidade de se percorrer toda a coleção de dados. Se for possível inferir que o conjunto resposta possuirá cardinalidade muito menor do que a coleção completa, o otimizador pode tirar proveito deste fato e percorrer o índice. A vantagem consistirá no volume de dados: certamente o índice será bem menor do que a coleção completa. Por outro lado, se o predicado p for tal que possamos fazer uso da estrutura do índice para avaliá-lo, nem mesmo o índice precisará ser completamente varrido. Como exemplo temos o caso em que o predicado especifica um conjunto finito de valores para a_i, ou ainda o caso em que o índice está ordenado e o predicado especifica uma faixa de valores para a_i.

Se o atributo não está indexado, o predicado p deverá ser calculado percorrendo-se todo o conjunto de tuplas. Uma situação especial que ocorre em alguns casos, especialmente em bases de dados consolidados ou com poucas alterações, é quando a própria relação está ordenada pelo atributo a_i — nesse caso, a avaliação do predicado poderá ser realizada da mesma forma que quando existe um índice ordenado.

3.4.2 Restrições sobre Atributos Espaciais

A realização de uma restrição espacial envolve a avaliação de um predicado espacial para um determinado conjunto de tuplas. De forma idêntica ao caso dos atributos não espaciais, podemos considerar como entrada para os nossos algoritmos de restrição um conjunto de tuplas e o predicado de restrição.

Embora o conjunto de tuplas possa ser especificado da maneira mais diversa possível, como por exemplo, uma latitude, uma região arbitrária, uma altitude ou uma propriedade de um atributo não espacial qualquer, iremos considerar que tal conjunto já

está predeterminado. Afinal, se a especificação do conjunto em si é realmente complexa, então poderá ser feita através de uma consulta, o que dará margem a uma decomposição em operações primitivas e possivelmente otimizações. No entanto, assumimos que essa é uma tarefa a ser realizada em uma etapa anterior. Portanto, o conjunto de tuplas será conhecido e determinado *a priori*.

Existem vários tipos de predicados de restrição espacial — na verdade, qualquer função p que receba uma tupla que contenha um ou mais atributos espaciais e retorne um valor booleano pode ser utilizada. No entanto, estaremos considerando predicados que possam ser calculados utilizando-se apenas um atributo dessa tupla, necessariamente espacial. Como exemplos de predicados que aparecem com frequência em sistemas de informações geográficas temos: interseção com determinada região, linha ou ponto; distância e posição relativa à alguma região, linha ou ponto; área ou perímetro entre determinados valores. Novamente, os tipos possíveis de predicados irão estar diretamente relacionados ao poder de expressão da linguagem de consulta. De fato, o ponto mais importante quanto à complexidade do predicado é a limitação do ser humano para expressá-lo, e também a sua correspondência com algum fenômeno do mundo real que se deseje analisar. Por outro lado, algumas funções tais como área e perímetro são informações tão utilizadas que em geral são calculadas e armazenadas com um atributo comum, não espacial, e atualizadas sempre que o respectivo atributo espacial também o for.

Vários dos predicados possíveis para uma restrição sobre atributos espaciais poderão ser avaliados definindo-se uma região e descobrindo-se se determinado objeto possui interseção com a mesma. Certamente esse é o caso da interseção propriamente dita, mas também dos predicados envolvendo distância e posição relativa. Assim, para recuperarmos todos os objetos distantes n metros ou menos do ponto x podemos simplesmente procurar os objetos que possuam interseção com o círculo de centro x e raio n. É importante observar que um caso específico desse tipo de consulta é quando a região demarcada se degenera em apenas um ponto ou linha. Por outro lado, regiões complexas poderão ser descritas por um polígono ou uma expressão matemática. Nesse caso, convém observar que definir a região desejada pode ser mais difícil até do que realizar a consulta propriamente dita. No entanto, esse é um problema diretamente relacionado com a expressividade da linguagem de consulta, de forma que não o abordaremos aqui. Além disso, a escolha entre avaliar o predicado como ele se apresenta

(p.e. calculando a distância de cada objeto ao ponto x) ou definindo-se uma região, deverá ser feita na etapa anterior à da realização da operação primitiva de restrição.

Da mesma forma que na avaliação da restrição sobre atributos não espaciais, para a avaliação da restrição espacial teremos dois casos: quando não existe ou quando existe um índice espacial sobre o atributo a_i envolvido no predicado.

Quando o atributo da restrição não estiver indexado, a coleção inteira de objetos deverá ser percorrida. Em geral, tal coleção não estará ordenada, mas se a mesma possuir alguma ordenação, dificilmente o será por algum atributo espacial — e mesmo que assim seja, a sua utilidade será muito limitada, pois já vimos que não existem funções de ordenação total que preservem a proximidade espacial.

Quando existe um índice sobre o atributo da restrição, o mesmo será utilizado para avaliar o predicado p. Porém, devemos lembrar que em geral um índice espacial possui apenas informação sobre a posição do objeto em relação ao espaço, não sendo uma projeção do atributo a_i propriamente dita. Ou seja, em geral o índice não conterá o valor do atributo espacial. Para muitos predicados de restrição espacial, a informação sobre o atributo a_i contida no índice pode ser suficiente para a avaliação do mesmo — especialmente predicados que envolvam apenas a interação com uma região predefinida. A própria estruturação espacial presente no índice pode ser utilizada para a determinação do valor do predicado, permitindo que o mesmo seja calculado sem que o índice inteiro tenha de ser percorrido. Finalmente, para muitos predicados a informação contida no índice não será suficiente para a sua avaliação completa: o resultado poderá ser *verdadeiro*, *falso* ou *inconclusivo*, o que significa que o próprio atributo espacial terá de ser investigado para que o valor do predicado seja calculado. No entanto, como apenas os casos *inconclusivos* forçarão a recuperação do objeto, essa abordagem ainda é melhor do que percorrer toda a coleção de objetos.

Em vários artigos, onde os mais citados são GUTTING (1994) e BRINKHOFF et al. (1993a, 1994), é proposta a utilização de índices espaciais para a realização eficiente da operação de restrição espacial, em particular as R*-Trees. Em SAMET (1990) temos vários outras estruturas espaciais para a implementação de índices que permitam a realização eficiente de restrições espaciais.

3.5 *Realização de Junções*

Nesta seção iremos mostrar como pode ser implementada a operação de junção espacial. Para tanto, mostraremos como pode ser implementada a operação de junção sobre atributos não espaciais. No entanto, ao repetirmos a abordagem anterior, ou seja, ao tentarmos implementar a junção espacial por analogia com a junção de atributos não espaciais, encontraremos uma série de dificuldades. Por ser esse o tema da tese, iremos apresentar também algumas considerações sobre a complexidade de cada abordagem.

3.5.1 A Operação de Junção

A operação de junção é uma das operações mais poderosas dos bancos de dados relacionais (DATE, 1991) e de extrema importância para os atributos espaciais (GÜNTHER e BUCHMANN, 1990). Duas das operações mais frequentes nos Sistemas de Informações Geográficas atualmente são as operações de sobreposição e interseção de mapas (Figura 3-2). Estas operações resultam de junções espaciais, podendo mesmo serem consideradas casos especiais das mesmas. Um exemplo de consulta simples que se enquadra em tal categoria é a que nos é apresentada em VEENHOF et al. (1995): "recupere todas as áreas rurais abaixo do nível do mar que possuam tipo de solo igual a areia e distantes três milhas de lagos poluídos". Com os planos temáticos "uso da terra", "solo", "poluição" e "elevação", várias junções espaciais terão de ser realizadas para responder a tal consulta.

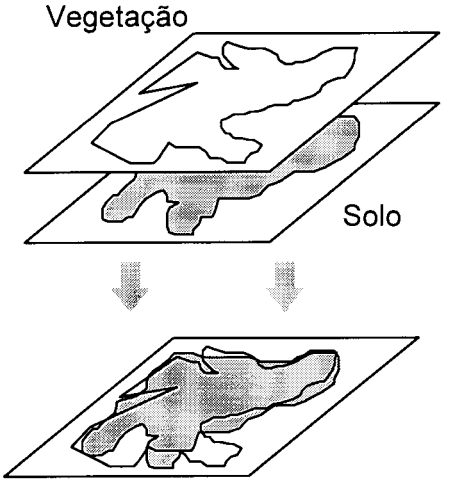


Figura 3-2 - junção espacial para a sobreposição de mapas.

Entendemos a junção espacial como um subconjunto do produto cartesiano de dois conjuntos, A e B, não necessariamente distintos, contendo respectivamente m e n elementos. Esse subconjunto é formado pelos pares de elementos de $\boldsymbol{A}$ e $\boldsymbol{B}$ para os quais um determinado predicado p é avaliado como *verdadeiro*. De especial interesse para

aplicações práticas é a sobreposição de objetos espaciais, de forma que é na interseção de polígonos que concentraremos a nossa atenção. No entanto, como é apontado em BRINKHOFF (1994), os resultados obtidos para a interseção em particular podem ser facilmente aplicados em vários outros tipos de operadores.

Há uma peculiaridade nos predicados de junção envolvendo atributos espaciais. Enquanto que para os atributos não espaciais o predicado de junção mais comum é a igualdade, para os atributos espaciais isso não é verdade. Predicados que envolvam atributos espaciais usualmente são de interseção, ou podem ser calculados utilizando-se a interseção de polígonos. Conforme veremos adiante, essa característica irá dificultar sobremaneira, quando não inviabilizar, a adaptação dos algoritmos já estabelecidos para a realização de junções em bancos de dados relacionais. Em particular, quanto mais eficiente o método relacional, mais complicada é a adaptação para o seu uso em consultas espaciais.

3.5.2 Junções sobre Atributos não Espaciais

Existem três abordagens básicas para a realização de junções em bancos de dados relacionais, a saber: *loops* aninhados, *sort-merge join* e *hash-join*. Nesta subseção iremos mostrar cada uma delas, bem como um estudo sobre a complexidade de cada uma dessas abordagens.

É importante observar que as junções mais comuns realizadas em bancos de dados relacionais possuem um predicado simples: como vimos, em geral procura-se tuplas que possuam valores idênticos para um determinado atributo (*equi-join*). Assim, os algoritmos aqui apresentados foram em sua maioria projetados para a eficiência máxima nesse tipo de junção. No entanto, é possível aplicá-los a vários outros predicados com pouca ou nenhuma adaptação.

De qualquer forma, mesmo se o predicado de junção for tal que não seja possível aplicar os algoritmos mais sofisticados, a simples existência dos índices pode ser aproveitada. Como vimos, um índice é uma projeção de um (ou mais) atributo(s), e o seu uso pode melhorar significativamente o desempenho de uma junção pelo simples fato de que o índice certamente ocupará menos espaço do que a tabela completa. Essa propriedade pode inclusive justificar a realização de uma projeção dos atributos da junção antes da realização da junção em si — mas essa é uma tarefa a ser realizada em uma etapa anterior à da avaliação de operações primitivas de junção.

3.5.2.1 Abordagem de Loops Aninhados

A abordagem dos *loops* aninhados consiste em avaliar o predicado de junção para o todos os pares do produto cartesiano $A \times B$, ou seja, para cada elemento do conjunto A percorrer todos os elementos do conjunto B e avaliar o predicado para cada um dos pares de elementos. Exceto em situações extremas, esta abordagem tem desempenho extremamente fraco para ser considerada seriamente como uma opção para a implementação. Um exemplo desta possibilidade ocorre quando um dos conjuntos possui cardinalidade muito maior do que o outro e não há nenhum tipo de índice para os atributos envolvidos na junção.

Note no entanto que esta abordagem irá funcionar para qualquer tipo de predicado, pois parte da definição da operação de junção como sendo uma restrição sobre um produto.

3.5.2.2 Abordagem de Junção por Sort-Merge

O algoritmo de junção por *sort-merge* consiste em estabelecer uma ordenação para os conjuntos sobre os respectivos atributos de junção e utilizar essa ordenação para restringir a necessidade de calcular o predicado de junção.

Exemplos de predicados de junção que possuem total compatibilidade com a ordenação convencional, e portanto podem ser avaliado desta forma, são os predicados envolvendo os operadores: $=$, $<$, $\leq$, $\geq$, $>$. Para ilustrar o método, vejamos o que ocorre quando temos um predicado da forma $t.a_i = u.a_j$, onde t e u são tuplas pertencentes respectivamente aos conjuntos de entrada **A** e **B**, e a_i e a_j são os respectivos atributos. Suponha que já exista uma ordenação para as tuplas (p.e. existem índices sobre a_i e a_j) para cada um dos atributos envolvidos, em ordem crescente de valor. Suponha também, sem perda de generalidade, apenas para simplificar o exemplo, que os valores assumidos por a_i e a_j são únicos (sem repetição), ou seja, cada valor de a_i está asssociado a apenas uma tupla de A (idem para a_j). Iremos então tomar o primeiro elemento $t.a_i$ de A, segundo a ordenação, e compará-lo ao primeiro elemento $u.a_j$ de B, também segundo a ordenação. Se forem iguais, o par (t, u) faz parte do conjunto resposta. Caso $t.a_i < u.a_j$, então o par (t, u) não faz parte do conjunto resposta, e além disso, podemos ter a certeza de que t não fará parte de nenhum outro par, pois todas as tuplas de B seguintes a u terão o atributo $u.a_j$ maior do que o valor atual, ou seja, maior do que $t.a_i$. Assim, avançamos para o próximo elemento de A e repetimos a comparação. Procederemos de

forma análoga para o caso $t.a_i > u.a_j$. Desta forma, mesmo sem avaliar o predicado, poderemos concluir que ele é falso para vários pares de tuplas — de fato, não chegaremos nem mesmo a materializar a maioria dos pares. É importante observar que tanto podemos utilizar índices ordenados sobre os atributos de junção como podemos ter a própria relação em si ordenada.

Veremos agora com mais detalhes critérios para determinar a compatibilidade entre o predicado de junção e a ordenação das tabelas. Definimos o predicado como uma função p: (u, t) $\rightarrow$ *bool*, onde u e t são tuplas dos conjuntos de entrada $\boldsymbol{A}$ e $\boldsymbol{B}$, respectivamente. Note que, sem perda de generalidade, podemos considerar que apenas um atributo de cada tupla será utilizado no predicado de junção: se houverem dois ou mais atributos de uma mesma tupla envolvidos no predicado, podemos considerá-los como um atributo que é uma tupla. Por outro lado, como já afirmamos anteriormente, nos predicados de junção espacial estaremos considerando apenas um atributo de cada tupla. Novamente, sem perda de generalidade, podemos assumir que a_i e a_j possuem o mesmo domínio $\boldsymbol{D}$: se assim não for, então $a_i \in \boldsymbol{D_1}$ e $a_j \in \boldsymbol{D_2}$, e definimos $\boldsymbol{D} = \boldsymbol{D_1} \cup \boldsymbol{D_2}$. Assim, o predicado de junção pode ser visto como uma função p: $\boldsymbol{D} \times \boldsymbol{D} \rightarrow$ *bool*, e podemos definir a seguinte relação binária **P** em $\boldsymbol{D}$: a **P** b, ou seja, (a, b) $\in$ **P**, se e somente se p(a, b) = *verdadeiro*. O predicado passa então a ser definido por uma relação binária **P** em $\boldsymbol{D}$.

Uma ordenação sobre um atributo, no sentido usual da palavra, pressupõe a existência de uma relação de ordem total[2] **R** para o domínio $\boldsymbol{D}$ desse atributo. O que garante o funcionamento do algoritmo é a propriedade transitiva da relação de ordem total: ao percorrermos a tabela na ordem especificada para o atributo a_i, assim que $t.a_i$ assume o valor x, para todas as tuplas seguintes os valores $t.a_i = y$ serão tais que valerá x **R** y. Por outro lado, para todas as tuplas anteriores $t.a_i = w$ valerá w **R** x. Se o valor do predicado p(x, v) puder ser utilizado para: afirmar que p(x, w) é falso ou descobrir o valor de p(y, v) para todo y tal que x **R** y; então o algoritmo irá funcionar, pois não será necessário percorrer tais valores. De forma análoga podemos verificar que o atributo $u.a_j$ também deve estar ordenado pela mesma relação de ordem total, de forma que possamos usar o mesmo artifício para descobrir o valor do predicado para os pares correspondentes em u. A Figura 3-3 fornece uma visualização dessa condição. As linhas

contínuas e grossas representam os pares que faltam ser investigados no momento em que os índices estão fornecendo os elementos (x, v); as linhas contínuas e um pouco mais finas representam os pares que ou já foram investigados em um momento anterior ou que o predicado é tal que podemos afirmar que tais pares não pertencem à relação definida pelo predicado.

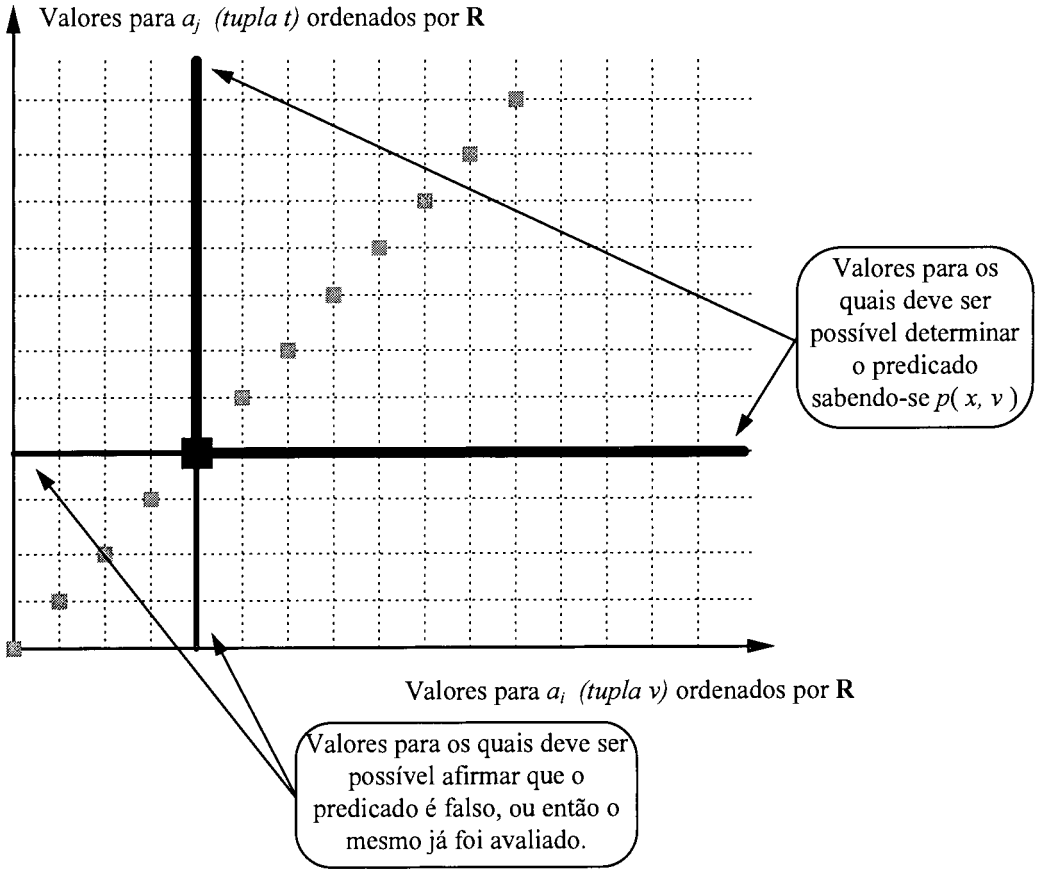


Figura 3-3 - Representação gráfica da junção por *sort-merge join*

Posto dessa forma, o algoritmo *sort-merge join* somente poderia ser utilizado para predicados de igualdade ou então predicados definidos por uma relação idêntica à utilizada para a criação dos índices. No entanto, algumas manipulações podem ser feitas para aumentar a aplicabilidade do método. A primeira é que o predicado $t.a_i = u.a_j$ pode ser substituído por $f_1(t.a_i) = f_2(u.a_j)$ desde que o índice seja construído sobre $f_1(t.a_i)$ e $f_2(u.a_j)$, ou seja possível obtê-lo a partir do índice existente. Um exemplo é um índice construído utilizando-se a relação $\leq$ usual, e f_1 e f_2 crescentes: se $y < w$ e $f_1(x) \leq f_2(y)$ então $f_1(x) < f_2(w)$, pois $f_2(y) < f_2(w)$. Podemos proceder de forma análoga pois f_1 é crescente também, logo o método pode ser utilizado para deduzir o valor do predicado

[2] Uma relação binária **R** em conjunto ***D*** é de ordem total se é uma relação reflexiva, transitiva e antissimétrica onde $\forall\ a, b \in \boldsymbol{D}$, ou a **R** b ou b **R** a.

utilizando os índices já existentes. Por outro lado, mesmo que os índices para f_1 e f_2 não possam ser deduzidos a partir dos índices já existentes, pode ser vantajoso construí-los para realizar a junção. Nesse caso, devemos encarar $f_1(t.a_i)$ e $f_2(u.a_j)$ como se fossem atributos das respectivas tuplas, porém calculados ao invés de armazenados.

Por sua vez, uma relação **R** de ordem total no conjunto ***D*** pode ser usada para definir uma outra relação **R'** em ***D***, que chamaremos de inversa de **R**, da seguinte forma: $\forall\ a, b \in \boldsymbol{D}$, se $(a, b) \in \mathbf{R}$ então $(b, a) \in \mathbf{R'}$, ou seja, se $a\ \mathbf{R}\ b$ então $b\ \mathbf{R'}\ a$. É fácil verificar que **R'** assim definido é uma relação de ordem total em ***D*** também. Mais do que isso uma tabela ordenada por **R'** conterá os elementos da mesma tabela ordenada por **R** na ordem exatamente inversa. O que significa que, se tivermos duas tabelas ordenadas por **R** e por **R'**, respectivamente, podemos considerar que as duas tabelas estão ordenadas pela mesma relação se percorrermos uma delas na ordem inversa. Mais do que isso, poderemos usar uma ordenação por **R** para calcular um predicado compatível com **R'**.

Em bancos de dados relacionais, esse algoritmo é em geral implementado utilizando índices do tipo Árvore-B. Os atributos são ordenados na árvore, e as duas árvores índices são percorridas simultaneamente, sendo a coordenação feita com base na ordem do valor das entradas na árvore. A eficiência desse algoritmo é satisfatória para a maior parte das aplicações, especialmente se os índices são mantidos armazenados na base de dados de forma que não seja necessário criá-los a cada consulta. Além disso, os pares do conjunto resposta da junção serão produzidos por esse algoritmo já ordenados, o que pode facilitar junções ou restrições encadeadas.

Em alguns casos, especialmente bases de dados que sofrem poucas alterações, os próprios conjuntos de dados são mantidos ordenados. A desvantagem dessa estratégia é que somente é possível escolher um atributo (ou composição de atributos) para realizar a ordenação.

3.5.2.3 Abordagem Hash-Join

As técnicas de realização de junções utilizando o paradigma *hash-join* na área relacional são bem estudadas e possuem excelente desempenho (DEWITT et al., 1984), particularmente para relações que são grandes quando comparadas ao tamanho disponível para *buffers*. No entanto, a aplicabilidade desse algoritmo é bastante restrita, como veremos a seguir.

O algoritmo de *hash-join* pode ser dividido em duas fases distintas: a fase de partição e a fase de junção. Na fase de partição, os conjuntos internos e externos são particionados e seus elementos são colocados nos respectivos *buckets*, utilizando-se para isso funções de partição (funções de *hash*). Na fase de junção, os *buckets* correspondentes aos conjuntos internos e externos são examinados, aplicando-se o predicado a cada um dos elementos e gerando-se assim o conjunto resposta.

Uma partição de um conjunto é uma família de subconjuntos, chamados de blocos da partição, tais que: nenhum bloco é vazio, a união dos blocos forma o conjunto original, os blocos que formam a partição são disjuntos (possuem interseção vazia). A primeira restrição é de ordem apenas teórica. Na prática, significa que devemos fazer um bom uso do espaço reservado para o algoritmo de *hash*, não permitindo que muitos *buckets* fiquem vazios.

Uma função de partição será uma função *f* tal que, quando aplicada a cada um dos elementos de um conjunto, o associe a um e apenas um dos blocos da partição. Note que, para o algoritmo de *hash* funcionar, é necessário que os elementos dos conjuntos que possam satisfazer ao predicado proposto sejam mapeados para o mesmo *bucket* para que possam ser comparados na fase de junção (Figura 3-4). Em outras palavras, se *p*(*x*, *y*) = *verdadeiro*, então *x* e *y* estarão no mesmo *bucket*. O contrário não é necessário: se *x* e *y* estão no mesmo *bucket*, então *p*(*x*, *y*) pode ser *verdadeiro* ou *falso*. Porém é desejável que ocorra o mínimo possível de casos onde *x* e *y* estão no mesmo *bucket* e o predicado é *falso* para que o algoritmo seja eficiente.

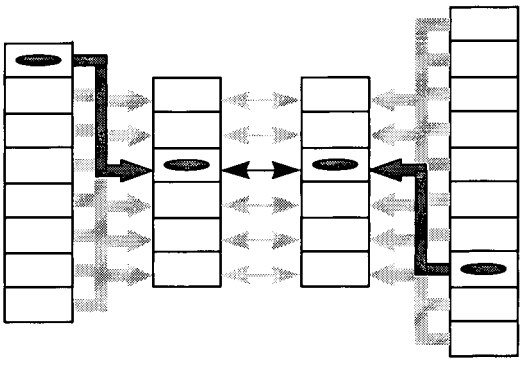


Figura 3-4- Elementos a serem comparados devem ser mapeados para o mesmo *bucket*

Uma forma simples de garantir que os elementos para os quais o predicado é *verdadeiro* serão mapeados para o mesmo *bucket* é utilizar uma relação de equivalência[3] para particionar os conjuntos. Na verdade, esta é uma condição necessária e suficiente

para a aplicabilidade do algoritmo de *hash*, conforme ficará claro mais adiante. Primeiro, vamos mostrar que essa é uma condição suficiente.

Claro está que a relação de equivalência deverá ser escolhida levando-se em conta o predicado da junção. Assim, para junções envolvendo o operador de igualdade aplicado a números inteiros, podemos utilizar a relação de equivalência *módulo n*, onde *n* é o número de *buckets* da tabela de *hash*. A escolha da relação de equivalência, na prática, é enormemente simplificada pelo fato de que junções, de uma forma geral, são realizadas com predicados de igualdade, que já é uma relação de equivalência.

Por outro lado, quando o predicado não for de igualdade, podemos encontrar uma relação de equivalência, possivelmente em uma tabela de relações pré-definida, tal que se x e y são tais que o predicado $p(x, y)$ é *verdadeiro*, então x e y pertencerão à mesma classe de equivalência[4], ou seja, $x \approx y$. Assim, se $x \approx y$ temos uma forma simples de obter $f(x) = f(y)$: basta tomarmos sempre o mesmo elemento representativo de cada classe e aplicar nesse elemento a função de *hash*. Como exemplo, considere o operador de igualdade aplicado a cadeias de caracteres, porém sem levar em consideração a presença de acentos e nem maiúsculas e minúsculas. A primeira observação a ser feita é que esse operador também define uma relação de equivalência. Logo, basta aplicarmos a função de *hash* sempre ao mesmo representante de cada classe de equivalência — por exemplo, utilizarmos sempre a cadeia de caracteres formada por caracteres maiúsculos e sem acento. Dessa forma, *Estêvão* ≈ *Estevao*, e aplicaremos a função de *hash* a *ESTEVAO* nos dois casos.

Caso o predicado não seja uma relação de equivalência, então devemos definir uma de tal forma que o predicado esteja contido na mesma. É um resultado conhecido que o fecho transitivo-reflexivo de uma relação **R**, denotado por **R***, é a menor relação simultaneamente transitiva e reflexiva que contém **R**. Logo, uma relação de equivalência que contenha **R** também conterá **R*** — o que nos indica um bom caminho para se encontrar tal relação de equivalência. Considerando o exemplo do parágrafo anterior, se definirmos um novo predicado que considere "á" equivalente a "a" , mas não a "ã", ou seja, ou possui o acento correto ou não possui acento nenhum, poderíamos

[3] Uma relação de equivalência é uma relação simultaneamente reflexiva, transitiva e simétrica.

[4] Utilizamos a notação $x \approx y$ para indicar que x e y são da mesma classe de equivalência, e $[x]$ para denotar essa classe.

tomar como relação de equivalência a relação definida pelo primeiro predicado pois ela contém a relação do segundo predicado.

Se não encontramos tal relação, ou se essa relação tiver poucas classes de equivalência, não é recomendável o uso do algoritmo de *hash* para realizar a junção. Por exemplo, no caso do operador > (*maior*) aplicado a algum atributo pertencente ao conjunto dos números inteiros **I**, teremos que *maior** não define uma relação de equivalência: *maior** é a relação *maior-ou-igual*, que não é simétrica. De fato, a única relação de equivalência que contém a relação *maior* é a representada pelo próprio produto cartesiano $\mathbf{I} \times \mathbf{I}$, que define uma única classe de equivalência — ele mesmo. Logo, o algoritmo de *hash* não é uma boa escolha para realizar junções cujo predicado seja *maior*.

Para mostrar que esta é uma condição necessária, basta lembrar que toda partição de um conjunto define uma relação de equivalência. Logo, a função de *hash* será uma relação de equivalência. Como a função de *hash* precisa mapear para o mesmo *bucket* todos os elementos que possam atender ao predicado da junção, a relação de equivalência definida pela função de *hash* deverá conter a relação definida pelo predicado.

Finalmente, vale mencionar alguns predicados para os quais aparentemente não é possível definir uma relação de equivalência mas é possível utilizar o algoritmo de *hash*. Como exemplo, temos o predicado $A.idade = B.idade + 1$, que associa tuplas tais que a idade do elemento pertencente a A seja um ano maior do que B. Sabe-se que o fecho reflexivo-transitivo dessa relação, $y=x+1$, é a relação *maior-ou-igual*, a qual já consideramos anteriormente. No entanto, podemos considerar esse predicado como o predicado de igualdade aplicado ao atributo *sucessorIdade*, definido pela expressão "*idade* + 1" de B, ou seja, um atributo que por acaso é calculado — e nesse caso, teremos uma junção com o predicado de igualdade, que como já vimos, é uma relação de equivalência.

De fato, podemos generalizar essa condição da mesma forma que o fizemos para o algoritmo *sort-merge join*: basta que o predicado $f_1(\ t.a_i\ )\ \mathbf{R}\ f_2(\ u.a_j\ )$ seja uma relação de equivalência (ou esteja contida em uma que não seja o próprio produto cartesiano) para podermos então aplicar o algoritmo de *hash* sobre os valores de $f_1(\ t.a_i\ )$ e $f_2(\ u.a_j\ )$, e não diretamente sobre os atributos $t.a_i$ e $u.a_j$.

3.5.3 Complexidade das Junções

Nesta subseção faremos algumas considerações sobre a complexidade dos algoritmos de junção citados anteriormente, restrita porém aos atributos não espaciais. A discussão sobre a complexidade dos algoritmos para junções espaciais será realizada no capítulo seguinte.

3.5.3.1 Complexidade do Algoritmo de Junção por Loops Aninhados

Iremos supor que os conjuntos de entrada possuem cardinalidade m e n. Então, utilizando a técnica dos *loops* aninhados, devemos realizar $n \times m$ comparações, o que é ineficiente. A complexidade pode ser estimada como segue: vamos supor que n é a cardinalidade do maior dos conjuntos, então $m = n - b$, para algum $b > 0$. Como são necessárias $m \times n$ comparações, teremos:

$(n - b) \times n \Rightarrow n^2 - b \times n \Rightarrow O(n^2)$.

A complexidade desse algoritmo é, pois, $O(n^2)$.

3.5.3.2 Complexidade do Algoritmo de junção por Sort-Merge

O algoritmo de junção utilizando a abordagem *sort-merge* se divide em duas fases distintas: ordenar os conjuntos e percorrê-los de forma sincronizada.

A primeira fase implica na ordenação de dois conjuntos com n e $n + b$ elementos. Como os melhores algoritmos de ordenação possuem complexidade $n \log n$, teremos: $(n + b) \log (n + b) + n \log n \Rightarrow O(n \log n)$.

Para a segunda fase, no melhor caso os elementos máximos e mínimos dos índices permitem inferir que o conjunto resposta da junção será vazio, e nesse caso, a complexidade será O(1). No pior caso, iremos percorrer, intercaladamente, todos os elementos de $\boldsymbol{A}$ e de $\boldsymbol{B}$. Logo, a complexidade será: $2n + b \Rightarrow O(n)$.

Portanto, a complexidade total desse algoritmo será dominada pela primeira etapa: $O(n \log n)$.

3.5.3.3 Complexidade do Algoritmo de junção por Hash

O ótimo desempenho atingido pelo *hash* nas abordagens relacionais pode ser explicado como se segue. Se escolhermos a função de *hash* de modo a particionarmos os conjuntos de entrada em p partições (*buckets*), sabendo-se que teremos de realizar comparações apenas entre as partições correspondentes a um e a outro conjunto, o número de comparações pode ser estimado como (supondo-se que as partições de cada conjunto contenham o número médio de elementos n/p e m/p, respectivamente):

$$p \cdot \frac{n}{p} \cdot \frac{(n+b)}{p} \Rightarrow \frac{n^2 + b \cdot n}{p}$$

pois teremos que investigar p partições, com n/b elementos de $\boldsymbol{A}$ para serem comparados com $(n + b)/b$ elementos de $\boldsymbol{B}$ cada uma. Agora, se escolhermos p como sendo proporcional a n, encontraremos complexidade $O(n)$:

$$p = n/c,$$

$$\frac{n}{c} \cdot \frac{n}{n/c} \cdot \frac{n+b}{n/c} \Rightarrow \frac{n}{c} \cdot c \cdot \frac{c \cdot (n+b)}{n} \Rightarrow c \cdot (n+b)$$

O que corresponde a uma complexidade $O(n)$. Na prática, a boa escolha da função de *hash* é o que irá garantir que as partições tenham um número de elementos próximo da média. Se isso não ocorrer, o desempenho será um pouco pior. Além disso, a constante c deve ser próxima de 1 para garantir um número de partições adequado. Já existem, no entanto, algoritmos de *hash* com número de partições variável, o que tem garantido uma boa distribuição dos elementos, como por exemplo o *hash* linear SAMET (1990).

Dessa forma, na primeira fase, de partição, teremos de aplicar a função de *hash* a cada um dos elementos dos dois conjuntos A e B. Logo, a complexidade será $O(n)$. Na segunda fase, como vimos, a complexidade também será de $O(n)$, resultando em uma complexidade total de $O(n)$. De fato, na prática as duas fases são realizadas de forma intercalada: primeiro, um dos conjuntos é inserido em uma tabela de *hash* — o menor deles pode levar a menos operações em disco. Depois, cada elemento do segundo conjunto é apenas comparado, e não inserido, nessa mesma tabela, resultando nos pares do conjunto resposta. Dessa forma, embora o algoritmo continue tendo complexidade $O(n)$, as constantes envolvidas serão ligeiramente menores.

3.5.3.4 Considerações a Respeito da Complexidade dos Algoritmos

É importante observar que a complexidade dos algoritmos serve apenas para mostrar a escalabilidade do algoritmo. No caso de determinados predicados sobre determinados conjuntos de dados, as constantes envolvidas poderão ser grandes o suficiente para não serem ignoradas, levando a melhores resultados o algoritmo que aparentemente é o mais complexo, ou seja, o algoritmo de *loops* aninhados. Tais casos extremos podem ocorrer quando o conjunto $\boldsymbol{A}$ é muito maior (ou menor) do que o conjunto $\boldsymbol{B}$, ou ainda quando o conjunto resposta é grande quando comparado a $\boldsymbol{A} \times \boldsymbol{B}$ (p.e. o predicado $t.a_i \neq u.a_j$). São os chamados casos de baixa seletividade.

Se os conjuntos de dados A e B possuírem índices mantidos pelo banco de dados sobre os atributos utilizados no predicado de junção (ou se as tabelas estiverem ordenadas pelo atributo da junção), é possível realizar a junção através do algoritmo *sort-merge* direto a partir da segunda fase, ou seja, percorrer de forma sincronizada os elementos dos dois conjuntos. Como vimos, esta parte do algoritmo possui complexidade $O(n)$, o que a torna de fato uma alternativa atraente ao algoritmo de *hash-join*. Por outro lado, a preexistência das estruturas de *hash* sobre os atributos envolvido no predicado da consulta exerce apenas uma influência constante no desempenho total do algoritmo de *hash-join*, pois a complexidade de ambas as fase é $O(n)$.

3.5.4 Junções sobre Atributos Espaciais

Nesta subseção iremos esboçar a versão espacial das três abordagens mais comuns para a implementação de junções: *loops* aninhados, *sort-merge join* e *hash-join*. Em particular, conforme já estabelecemos anteriormente, iremos mostrar os principais problemas e as possíveis soluções relacionadas apenas ao predicado de interseção de polígonos.

Os últimos congressos e publicações relacionadas com as áreas de bancos de dados e de sistemas de informações geográficas têm uma grande quantidade de artigos na área de processamento de junções espaciais quando o predicado é de interseção de polígonos. Sem pretender enumerar todos os trabalhos já publicados, iremos comentar os principais métodos propostos, dispensando mais atenção àqueles que tenham se destacado tanto por eficiência quanto por impacto nos trabalhos posteriores.

3.5.4.1 Abordagem de Loops Aninhados

Uma primeira abordagem para a realização de junções espaciais consiste em utilizar o algoritmo de *loops* aninhados, proveniente das junções em bancos de dados relacionais. Tal algoritmo consiste em confrontar cada elemento do conjunto $\boldsymbol{A}$ com cada um dos elementos do conjunto $\boldsymbol{B}$, verificando se o predicado da junção espacial é satisfeito. Este algoritmo simples serve para ilustrar as operações mais caras na realização de junções espaciais: a transferência de objetos grandes do disco para a memória e a avaliação do predicado de junção (p.e. o teste de interseção de polígonos).

De fato, essa é a única abordagem que pode ser imediatamente transportada do domínio dos atributos não espaciais para o domínio dos atributos espaciais, e a única

também que não requer nenhuma adaptação para os diferentes predicados possíveis. Infelizmente, essa costuma ser a abordagem com pior desempenho.

3.5.4.2 Abordagem Junção por Sort-Merge

Vimos que a compatibilidade entre a ordenação do índice e o predicado de junção é fundamental para o funcionamento do algoritmo usado para realizar junções não espaciais pela abordagem *sort-merge*. Portanto, se quisermos aplicar nas junções espaciais o mesmo algoritmo, devemos garantir tal compatibilidade.

Por outro lado, as suposições feitas para garantir tal compatibilidade terão de ser completamente revistas no domínio dos atributos espaciais, a começar pela relação de ordem total utilizada para a construção do índice. Como vimos, não existe uma relação de ordem total que preserve a proximidade espacial, e o predicado de interseção está fortemente relacionado com a proximidade.

Portanto, embora possamos construir um índice espacial baseado em uma relação **R** de ordem total, o mesmo terá utilidade limitada para a realização de uma junção espacial cujo predicado seja o de interseção. O predicado de interseção de polígonos define uma relação que é simétrica e reflexiva, mas não é transitiva — ou seja, é um predicado totalmente inadequado para se tirar proveito de qualquer ordenação mantida pelo índice. De fato, tendo calculado $p(x, y)$ e chegando à conclusão que o polígono x possui interseção com y, saber que y **R** w não nos permite inferir nada a respeito de $p(x, w)$.

Por outro lado, se utilizarmos um método de acesso espacial que nos permita estabelecer relações de proximidade sem utilizar uma relação de ordem total para construirmos um índice, poderemos utilizar esse índice para a avaliação de predicados que estejam relacionados com a proximidade espacial (p.e. interseção e distância). Porém, ao relaxarmos a suposição de que o índice mantém uma relação de ordem total sobre o atributo da junção teremos de adaptar o algoritmo para lidar com a nova estrutura do índice, o que exigirá um algoritmo diferente para cada tipo de índice.

Um bom exemplo de método de acesso espacial que mantém informações sobre a proximidade de objetos espaciais são as R-Trees (GUTTMAN, 1984). Em SAMET (1990) temos vários outros exemplos, como o *grid file* e o *excell*. Em BRINKHOFF (1993a) temos a descrição completa de um algoritmo para a realização de junções, utilizando o operador de interseção, percorrendo duas R*-Trees. A justificativa

para o uso de R*-Trees é feita com base em seu desempenho superior quando comparada às outras estruturas. Basicamente, cada entrada na raiz de uma árvore é comparada com as entradas da raiz da outra árvore. Quando o algoritmo encontra uma interseção entre duas entradas, os nós de cada uma das R-Trees correspondentes a essas entradas devem ser investigados. Assim, todas as sub-árvores de cada um desses nós serão percorridas também, recursivamente até as folhas. Iremos detalhar esse e outros algoritmos mais adiante na seção 3.6.

A primeira deficiência deste algoritmo surge quando comparamos as propriedades de cada índice utilizado: enquanto no domínio dos atributos não espaciais os nós de uma B-Tree garantidamente não possuem interseção, propriedade garantida pelo uso de uma relação de ordem total, o mesmo não ocorre na R-Tree. Ao contrário, praticamente em todas as R-Trees haverá interseção entre os nós, o que certamente implicará em visitar alguns nós mais de uma vez, e por conseguinte, toda a sub-árvore desse nó também será visitada mais de uma vez.

O problema da superposição dos nós afeta grande parte das estruturas espaciais. Ele ocorre principalmente devido ao fato de que os dados em si podem possuir áreas de sobreposição — mesmo que seja apenas na sua fronteira. Por outro lado, se exigirmos que todos os objetos que possuem sobreposição estejam situados no mesmo nó, poderemos chegar à situação limite de ter apenas um nó e, neste caso, o índice seria inútil.

A alternativa utilizada em alguns índices para dados espaciais é a decomposição de alguns objetos, em particular, aqueles que cruzam a fronteira definida pelo nó. A desvantagem dessa abordagem é a duplicação da referência para o objeto espacial, o que resultará em ter acesso ao mesmo mais de uma vez. Além disso, ocorrerá um problema novo nessa abordagem: já que um mesmo objeto aparecerá mais de uma vez no índice, nada impede que se sobreponha em mais de um lugar a um outro objeto que também apareça mais de uma vez, levando o par a aparecer mais de uma vez no conjunto resposta. É necessário pois, nesta abordagem, eliminar os pares de polígonos duplicados, o que é uma tarefa trivial porém custosa: ou se utiliza uma tabela de *hash* ou se ordena o conjunto resposta, sendo que provavelmente estaremos lidando com um conjunto com elementos demais para caber na memória.

Finalmente, é importante lembrar que, como já vimos para o caso das restrições espaciais, em geral o objeto espacial em si não é armazenado no índice, mas apenas

informações sobre a sua posição e extensão. Nem sempre será possível avaliar o predicado de junção baseado somente em tais informações, o que fatalmente nos obrigará a buscar os objetos para avaliar o predicado — ou seja, o conjunto resposta da junção não poderá ser calculado apenas percorrendo-se os índices.

3.5.4.3 Abordagem Hash-Join

Algoritmos de *hash-join* são os que requerem mais modificações para serem utilizados em junções espaciais. Por esse motivo, explicaremos brevemente alguns conceitos e problemas já levantados por outros autores ao se tentar aplicar tal abordagem em junções espaciais. Em particular, no trabalho de MING e RAVISHANKAR (1996), podemos ver uma caracterização das escolhas a serem feitas para a aplicação de técnicas envolvendo *hash-join* às junções espaciais. Faremos aqui um resumo rápido de tais idéias.

Conforme apontado em BRINKHOFF (1994) e MING e RAVISHANKAR (1996), ao se utilizar *hash-join* para realizar junções espaciais as dificuldades surgem principalmente por dois motivos. Primeiro, devido à natureza do predicado da junção espacial: não se busca polígonos exatamente iguais, mas que possuam alguma interseção. Essa diferença é fundamental para a compreensão dos problemas relativos à junção espacial. No caso relacional, como o predicado usual é de igualdade ou uma relação de equivalência, sempre que x e y obedecerem ao predicado teremos como garantir que $f(x)$ seja igual a $f(y)$, fazendo com que as respectivas tuplas sejam mapeadas para o mesmo *bucket*. No entanto, no caso da junção espacial, nem sempre isso ocorre. Sejam p_1 e p_2 dois polígonos que se interceptam, porém tais que $p_1 \neq p_2$ (certamente, esse é o caso padrão): nem sempre será possível garantir que $f(p_1) = f(p_2)$, por melhor que seja a nossa escolha para f, já que $p_1 \neq p_2$. Isso ocorre por que um predicado envolvendo interseção de polígonos não define uma relação de equivalência, e o fecho reflexivo-transitivo da operação de interseção é formado pelas partes conexas do conjunto de polígonos — possivelmente o conjunto inteiro. Em segundo lugar, conforme já vimos, devido ao fato de os dados espaciais carecerem de uma função de ordenação total que preserve a proximidade espacial.

Já que vimos que não é possível definir uma relação de equivalência para o predicado de interseção, vamos alterar o algoritmo de *hash-join*. Os algoritmos utilizados em bancos de dados relacionais têm concordado com duas restrições, cada

uma delas aplicável a uma das fases já citadas: partição e junção. A primeira restrição é a atribuição simples, aplicável a fase de partição, significando que cada elemento da entrada é colocado em apenas um *bucket*. A segunda é o casamento simples, aplicável a fase de junção, significando que cada *bucket* do conjunto interno será comparado com apenas um *bucket* do conjunto externo. Assim, para se adaptar algoritmos de *hash-join* para a realização de junções espaciais, devido ao problema do predicado não ser uma relação de equivalência, é necessário que pelo menos uma das duas restrições acima seja relaxada. As possíveis escolhas são (Figura 3-5):

- Atribuições múltiplas: nesse caso, um elemento da entrada pode ser mapeado para mais de um *bucket*.
- Casamentos múltiplos: nesse caso, cada *bucket* do conjunto interno pode necessitar ser comparado com mais de um *bucket* do conjunto externo.

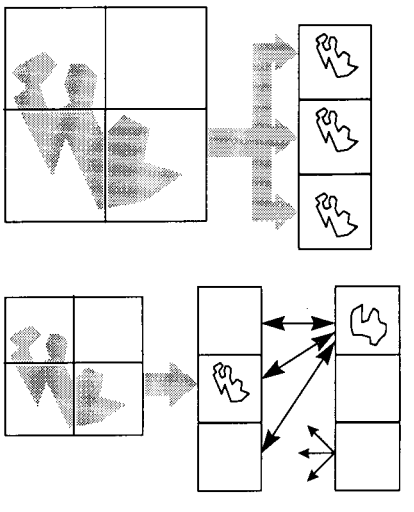


Figura 3-5 - Exemplo de atribuição múltipla e de casamento múltiplo.

Se relaxarmos a primeira restrição e permitirmos atribuições múltiplas, teremos como vantagem a simplicidade do algoritmo. A única diferença conceitual em relação à abordagem relacional é que a função de *hash* leva um objeto em um conjunto de *buckets* ao invés de apenas um *bucket*. A desvantagem nesse caso será um aumento no tamanho do conjunto de dados, resultando em uma maior quantidade de operações de E/S, além do trabalho extra para encontrar e eliminar do conjunto resposta os pares de objetos duplicados. Exemplos dessa abordagem são ZIMBRÃO e SOUZA (1997) e PATEL e DEWITT (1996).

Por outro lado, se relaxarmos a segunda restrição, sendo necessária a realização de casamentos múltiplos, não haverá um aumento no tamanho do conjunto de dados,

mas poderá haver a necessidade de se ler um *bucket* mais de uma vez, o que também poderá aumentar a quantidade de operações de E/S. Além disso, identificar os pares de *buckets* que necessitam ser comparados não é uma tarefa trivial. Um exemplo dessa abordagem pode ser encontrado em MING e RAVISHANKAR (1996). É importante observar que o relaxamento dessa restrição nos leva a um algoritmo semelhante ao utilizado nas junções dos nós em R-Trees, em que cada nó deve ser comparado com todos os outros nós com os quais ele possui interseção. De fato, o trabalho citado utiliza uma estrutura em árvore para particionar os dados.

3.5.5 Complexidade das Junções Espaciais

Nesta subseção iremos investigar um pouco mais a complexidade obtida em cada abordagem, sendo que por razões óbvias essa complexidade será bem mais dependente da estrutura de dados utilizada e de características dos conjuntos de dados envolvidos do que nos casos não espaciais.

3.5.5.1 Complexidade do Algoritmo de Junção por Loops Aninhados

A complexidade da realização de junções espaciais através da abordagem dos *loops aninhados* é a única que pode ser feita independentemente de algum conhecimento prévio sobre os conjuntos de dados. Como esse algoritmo é basicamente igual ao utilizado para os atributos não espaciais, a sua complexidade será a mesma: $O(n^2)$, onde n é o número de elementos do maior dos conjuntos de entrada. No entanto, como o cálculo do predicado espacial pode ser uma operação bastante custosa, haverá uma constante envolvida que não deve ser desprezada.

3.5.5.2 Complexidade do Algoritmo de Junção por Sort-Merge

A complexidade do algoritmo de junção espacial através da abordagem *sort-merge* está diretamente relacionada ao método de acesso espacial utilizado. Iremos aqui discutir aspectos genéricos e o caso particular da R-Tree.

Para qualquer método de acesso espacial utilizado, a complexidade será no máximo $O(n^2)$, onde n é o número de elementos do maior dos conjuntos de entrada, pois o pior caso que pode acontecer é a informação de proximidade espacial não servir para evitar o cálculo do predicado e recairmos no algoritmo de *loops* aninhados. Por outro lado a complexidade estará diretamente relacionada aos conjuntos de dados e ao tipo de predicado: no caso da interseção de polígonos, que não é transitiva, podemos ter $\boldsymbol{A}$ com n elementos, $\boldsymbol{B}$ com m elementos e o produto $\boldsymbol{A} \times \boldsymbol{B}$ com $n \times m$ pares diferentes

de polígonos, todos se sobrepondo. Certamente, a complexidade para calcular esse conjunto será $O(\ n^2\ )$, onde n é o número de elementos do maior dos conjuntos de entrada.

Assim, iremos realizar algumas suposições sobre os conjuntos de entrada. Em FALOUTSOS e KAMEL (1994) e GAEDE e FALOUTSOS (1996) há análises mais detalhadas que permitem uma estimativa mais acurada da complexidade das buscas em uma R-Tree, e que podem ser utilizadas para estimar a complexidade das junções espaciais. Um resultado porém fica evidente em FALOUTSOS e KAMEL (1994) e GAEDE e FALOUTSOS (1996) e que nos será de certa forma útil: a complexidade das buscas e das junções em uma R-Tree irá depender fortemente de determinados parâmetros dos conjuntos de dados. Contudo, para os nossos propósitos, a análise mais simples que iremos fazer é suficiente, pois também chegaremos a evidenciar tal dependência.

Em primeiro lugar, iremos estimar que os dados espaciais não estão uniformemente distribuídos, mas que seja possível construir para cada conjunto de entrada uma R*-Tree onde os nós possuem uma taxa de sobreposição pequena o bastante para ser desprezada. Além disso, iremos supor que a sobreposição média de cada nó da R*-Tree do conjunto $\boldsymbol{A}$ com cada nó da R*-Tree do conjunto $\boldsymbol{B}$ é r. Seja pois d o número mínimo de elementos em um nó, e D o número máximo de elementos em um nó. Para simplificar, suporemos que tanto as raízes como cada um dos nós das duas árvores possuem d elementos e que as alturas das árvores serão iguais, representadas por h, onde $log_D\ m \leq h \leq log_d\ m + 1$, e m o número de elementos do maior dos conjuntos de entrada.

Estabelecidos esses parâmetros, a complexidade de uma junção espacial, em termos de visitas a nós, o que de certa forma reflete o número máximo de acessos a disco, pode ser assim estimada: as raízes das duas árvores terão d elementos cada uma, onde apenas $(r \times d) \times (r \times d)$ pares de retângulos se sobrepõem. Isso significa que teremos de realizar r^2d^2 junções espaciais nas sub-árvores correspondentes, cada uma delas de altura $h - 1$. No nível seguinte, para cada uma das junções espaciais das sub-árvores das raízes, como supomos os mesmos parâmetros médios, deveremos ter novamente o mesmo número de junções espaciais para cada sub-árvore de altura $h - 2$, ou seja, r^2d^2, e assim sucessivamente. Portanto, teremos como total de junções, até o nível das folhas, $r^2d^2 \times r^2d^2 \times \ldots \times r^2d^2$, onde os fatores desse produto irão aparecer $h -$

1 vezes, e teremos então $r^{2(h-1)}d^{2(h-1)}$ junções. Como $h \leq log_d\ m + 1$ e usando a propriedade $r^{log_d m} = m^{log_d r}$, temos $r^{2(h-1)}d^{2(h-1)} \leq r^{2log_d m}d^{2log_d m} = r^{2log_d m}d^{\,log_d m^2} = r^{2log_d m}m^2 = m^{2log_d r}m^2 = m^{2(\,1\,+\,log_d r)}$. Se r for menor do que $1/d$, pelo nosso modelo não teremos nenhuma sobreposição em nenhum nível, logo podemos assumir que $1/d \leq r \leq 1$, e logo teremos $-1 \leq log_d r \leq 0$ qualquer que seja d, ou seja, $m^0 \leq m^{2(\,1\,+\,log_d r)} \leq m^2$, e teremos que a complexidade da junção estará entre $O(\ 1\ )$ e $O(\ n^2\ )$. Esse resultado era de se esperar, pois se tivermos $r < 1/d$, significa que os dois conjuntos não possuem nenhuma interseção, fato detectável ao nível da raiz das árvores independentemente dos tamanhos dos conjuntos de entrada. Por outro lado, se tivermos $r = 1$, significa que cada MBR do conjunto $\boldsymbol{A}$ possui interseção com todos MBRs do conjunto $\boldsymbol{B}$, e consequentemente o índice não irá nos ajudar a eliminar nenhum par de polígonos.

Para estimarmos o número de comparações necessárias, podemos considerar que cada junção de um nó com outro, para a identificação dos MBRs que se sobrepõem, será realizada por um algoritmo mais ingênuo (força bruta p.e.), e que irá requerer d^2 comparações de MBRs. Assim, teremos para cada subjunção d^2 comparações, o que nos leva ao valor de $d^2m^{2(\,1\,+\,log_d r)}$ comparações ao todo. No entanto, como d é um parâmetro de construção da R-Tree independente de m, a complexidade também será $O(\ m^{2(\,1\,+\,log_d r)}\ )$.

Como exemplo, tomemos $d = 50$ e $r = 0{,}15$ (em cada nó, 15% dos MBRs possuem interseção com os MBRs do nó correspondente na outra árvore). A complexidade da junção espacial estimada por nosso modelo, tanto em acessos a disco quanto em número de comparações será $O(\ m^{1{,}03}\ )$ — apenas um pouco maior do que a complexidade máxima do algoritmo de junção não espacial por *sort-merge*.

Devemos ressaltar contudo que fizemos a estimativa da complexidade apenas da fase de junção, ou seja, não incluímos nessa análise o custo de construção da R-Tree. Além disso, as suposições acima são por demais uniformes para serem verdadeiras para algum conjunto de dados real, o que significa que na prática a complexidade real será um pouco maior do que a estimada — no entanto, certamente entre $O(\ 1\ )$ e $O(\ n^2\ )$.

Finalmente, vale lembrar que enquanto o algoritmo de junção não espacial por *sort-merge* possui, na fase de junção, complexidade máxima $O(\ n\ )$, o algoritmo correspondente em sua versão espacial terá complexidade máxima de $O(\ n^2\ )$. Isso

evidencia a dificuldade existente na adaptação para as junções espaciais dos algoritmos existentes para junções não espaciais.

3.5.5.3 Complexidade do Algoritmo de Junção por Hash

A complexidade do algoritmo de junção por *hash-join* estará diretamente relacionada com qual das duas restrições será relaxada: atribuições simples ou casamentos simples. Contudo, como já mencionamos anteriormente, a abordagem utilizando casamentos múltiplos se aproxima bastante da abordagem utilizando *sort-merge*, inclusive utilizando-se de uma estrutura em árvore, de modo que é fácil de verificar que as respectivas complexidades não serão muito diferentes.

Iremos pois estimar a complexidade para a abordagem *hash-join* no caso das atribuições múltiplas. Suporemos que os conjuntos de dados não são uniformes mas que produzam p *buckets* com d elementos em média e $d + k$ elementos no máximo. Seja m o número de elementos do maior dos conjuntos de entrada, e a a medida do aumento do volume de dados devido à atribuição múltipla. É importante observar que o número máximo de elementos em um *bucket* está relacionado não só à média de elementos d, mas também ao número máximo de polígonos que possuem sobreposição na mesma área. Esse fator é representado pelo parâmetro k.

Na fase de partição, cada elemento de cada conjunto terá de ser atribuído a um ou mais *bucket*, de forma que teremos o equivalente a inserir am elementos para cada um dos conjunto de entrada, ou seja, teremos complexidade O(am) para criar as tabelas de *hash*. Para realizarmos as comparações cada *bucket* deve ser lido uma única vez, de forma que realizaremos p acessos a disco, levando a uma complexidade de $O(p)$.

Na fase de junção, podemos assumir que não teremos de realizar mais do que $p \times (d + k)^2$ comparações. Realizando as mesmas transformações algébricas realizadas em 3.5.3.3, d pode ser estimado como am/p e p pode ser escolhido em função de m, de forma que $p = m/c$. Podemos estimar o número máximo necessário de comparações em:

$$\frac{m}{c}\cdot\left(\frac{am}{m/c}+k\right)^2 \Rightarrow \frac{m}{c}\cdot(ac+k)^2 \Rightarrow \frac{m}{c}\cdot\left(a^2c^2+2ack+k^2\right) \Rightarrow m\left(a^2c+2ak+k^2/c\right)$$

Precisamos então analisar os termos entre parênteses para verificar as suas correlações com m. De fato, c não depende de m, pois é um parâmetro do algoritmo relacionando o número de partições com o número de elementos. O parâmetro k, que é o número máximo de polígonos que possuem interseção na mesma área, não depende do

número de partições, de forma que ao aumentarmos p, não necessariamente estaremos reduzindo o número máximo de elementos em um *bucket*. Por outro lado, k é um fator característico de cada conjunto de dados, não dependendo diretamente de m. Finalmente, o aumento dos dados a é relacionado à área de cada *bucket* da partição que por sua vez é relacionada ao número de partições p na mesma área. Para simplificar o modelo, podemos assumir uma correspondência linear entre a e p (na realidade, o limite de a/p quando $p \to \infty$ é uma c^{te} — ver a subseção 4.2.1, item 1), de forma que podemos escrever $a = pe \Rightarrow a = me/c$. Assim, o número máximo estimado de comparações ficará em:

$$m\left(a^2c + 2ak + k^2/c\right) \Rightarrow m\left(\left(\frac{me}{c}\right)^2 c + \frac{2me}{c}k + k^2/c\right) \Rightarrow m\left(\frac{m^2e^2}{c} + \frac{2me}{c}k + k^2/c\right)$$

$$\Rightarrow \left(m^3e^2 + 2m^2ek + mk^2\right)/c$$

o que nos levará a $O(\ m^3\ )$ comparações — um resultado surpreendente se levarmos em consideração que estimamos a complexidade máxima em $O(\ m^2\ )$ comparações para a abordagem de *loops aninhados*. Essa degradação de performance contudo pode ser evidenciada da seguinte forma: suponha dois conjuntos de dados com m polígonos cada um, tais que todos eles sejam iguais e estejam sobrepostos. Se tivermos apenas uma partição, todos os polígonos serão mapeados para a mesma, e teremos de realizar m^2 comparações. Por outro lado, se dividirmos essa partição em m partes, e cada polígono sendo mapeado para todas as m partições, cada uma delas continuará tendo m elementos, o que nos levará a realizar m^3 comparações. Certamente esse é um caso extremo, no qual a utilidade da abordagem *hash-join* será duvidosa. Na prática, contudo, podemos verificar que o parâmetro e é quase sempre pequeno o suficiente para diminuir bastante o número total de comparações para conjuntos de dados reais.

Um outro fator que torna a abordagem *hash-join* competitiva, como podemos ver em PATEL e DEWITT (1996), é o fato de que o acesso a disco é uma operação em torno de 10^6 vezes mais lenta do que comparações de números inteiros (MBRs) em memória, o que indica que o acesso a disco domina o tempo de execução do algoritmo para valores razoáveis de m. Esse algoritmo, como vimos, possui complexidade $O(\ m\ )$ para acessos a disco, menor portanto do que a abordagem de junção por *sort-merge*.

Finalmente, vale lembrar que o aumento do volume de dados provocado pela inserção de um polígono (ou MBR) em mais de um *bucket* e a consequente duplicação de pares de polígonos no conjunto resposta poderá ser resolvida por algoritmos que possuam complexidade da ordem de $O(m)$ — uma tabela de *hash*, por exemplo, onde seriam inseridos os identificadores dos pares de polígonos já colocados no conjunto resposta. Essa abordagem apresenta como inconveniente o fato de que se o conjunto resposta for muito grande, a tabela de *hash* terá de ser feita em disco. Contudo, como veremos no Capítulo 4, há formas mais eficientes de se detectar e eliminar os pares de polígonos repetidos em memória.

3.5.5.4 Considerações a Respeito da Complexidade dos Algoritmos

Novamente cabe observar que a complexidade dos algoritmos serve apenas para mostrar se ele é escalável. O parâmetro r utilizado na estimativa do número necessário de comparações para o algoritmo de junção por *sort-merge* somente pode ser obtido após a execução da junção, e vale apenas para as R-Trees construídas sobre dois conjuntos de dados específicos. Para outro par de conjuntos, teremos outro valor para r. Por outro lado, as constantes envolvidas na estimativa tanto de comparações quanto de acessos a disco para a abordagem de *hash-join* podem ser de tal forma influentes que para valores práticos de m o número estimado de comparações seja aceitável.

Embora nossas considerações tenham sido apenas sobre complexidade, elas foram suficientes para guiar o desenvolvimento de alternativas para o processamento de junções espaciais. Em termos de escalabilidade, por exemplo, algoritmos que utilizam a abordagem *hash-join* tendem a realizar menos acessos a disco do que algoritmos que usam a abordagem de *sort-merge*. Por outro lado, estes tendem a consumir menos tempo de CPU em comparações do que aqueles.

Finalmente, existem estudos mais aprofundados que vão além da estimativa de complexidade e fazem estimativas sobre o custo total de uma consulta, atualmente com taxas de acertos em torno de 10%, o que é bastante razoável para uso em otimizadores. Os primeiros trabalhos sobre estes custos, tais como HUANG et al. (1993), GÜNTHER (1993) e PAGEL et al. (1993), necessitavam de informações sobre as R-Trees envolvidas na junção tais como número e ocupação média dos nós, ao passo que FALOUTSOS e KAMEL (1994), HUANG et al. (1997b), GÜNTHER et al. (1998) e THEODORIDIS et al. (1998b) realizam estimativas baseadas apenas em informações

sobre os conjuntos de dados. Em THEODORIDIS et al. (1998a) podemos encontrar um apanhado geral dos trabalhos relativos às estimativas de custo das restrições e junções espaciais de uma forma geral.

3.6 Uma Metodologia para o Processamento para Junções Espaciais

Nesta seção iremos apresentar o processador de junções espaciais que servirá de arquétipo (*framework*) para as novas abordagens que iremos propor. Para isso, justificaremos a sua construção modularizada investigando a maneira corrente de se realizar junções espaciais através de MBRs.

3.6.1 O Processador de Consultas Espaciais em Múltiplos Passos

Em um trabalho freqüentemente citado, BRINKHOF (1994) apresenta o processador de consultas espaciais em múltiplos passos (Multi-step Spatial Query Processor - MSQP), projetado para a realização de junções espaciais. Em tal processador, uma junção espacial é dividida em três passos, descritos mais adiante. O processador de junções espaciais definido por BRINKHOF (1994) permite uma modularização sem precedentes na realização de junções espaciais. Cada passo é implementado por um módulo que pode ser substituído através de outro módulo equivalente. De fato, nesse trabalho procuramos ajustar os algoritmos propostos a tal modularização, de forma a poder substituir com vantagens os módulos descritos no MSQP. Alguns dos outros trabalhos que citaremos podem também ser adaptados para se conformarem com o MSQP, implementando alguns de seus passos.

3.6.1.1 Passo 1 do MSQP: Seleção de Candidatos

Primeiro, ao invés de uma representação exata dos polígonos, os seus MBRs são utilizados para computar uma junção espacial aproximada. Este passo retorna um assim chamado conjunto de candidatos pois contém todos os elementos do conjunto resposta e mais alguns elementos do produto cartesiano que não se interceptam, porém cujos respectivos MBRs se interceptam.

3.6.1.2 Passo 2 do MSQP: Aplicação do Filtro

No segundo passo, aproximações dos polígonos mais acuradas que MBRs são utilizadas para refinar a resposta, eliminando alguns dos falsos candidatos. As aproximações são representações mais simples do objeto em questão. Além disso,

respostas positivas também podem ser identificadas através de tais aproximações sem ter acesso à representação geométrica exata dos objetos espaciais.

3.6.1.3 Passo 3 do MSQP: Comparação Geométrica Exata

Finalmente, nesse passo todos os pares candidatos remanescentes são examinados. Este passo requer acesso à representação geométrica exata dos objetos espaciais, e freqüentemente é o passo mais demorado: exige tempo de CPU para computar o teste de interseção exato, e tempo de I/O para ler os objetos espaciais do disco. No entanto, o tempo gasto nesse passo pode ser reduzido através do uso de aproximações melhores no passo anterior.

3.6.2 MBRs e Junções Espaciais

Iremos nesta subseção justificar a realização da junção espacial inicialmente sobre os MBRs dos polígonos. Essa é inclusive uma característica comum a todos os métodos citados no capítulo anterior. De fato, o passo 1 do MSQP prevê a realização de uma junção espacial sobre os MBRs dos polígonos dos dois conjuntos de entrada. Utiliza-se os MBRs dos polígonos, e não os polígonos propriamente ditos, por três motivos principais.

O primeiro motivo é que, como vimos, a representação usual dos polígonos como lista de vértices pode ocupar uma grande quantidade de espaço de armazenamento, e além disso essa quantidade irá variar muito de um polígono para o outro. Se formos armazenar o próprio polígono na estrutura utilizada como índice iremos dificultar sobremaneira a sua construção eficiente, pois o tamanho dos blocos deverá ser muito grande — resultando em tempos de acesso e transferência para a memória principal muito superiores do que se utilizarmos blocos menores. Além disso, o uso de blocos grandes poderá resultar em muito espaço desperdiçado. Uma maneira de se evitar esse desperdício pode ser a utilização de blocos de tamanho variável, mas essa é uma alternativa que traz mais dificuldades ainda na implementação dos algoritmos.

O segundo motivo é que a maior parte dos índices eficientes para áreas utiliza estruturas com certa regularidade para realizar a decomposição do espaço (SAMET, 1990), sendo o retângulo a mais comum. De fato, o nome R-Tree vem de *Rectangle-Tree*. Poucas estruturas podem lidar com polígonos diretamente, como é o caso da Cell-Tree (GÜNTHER, 1989) — mas mesmo essa estrutura lida apenas com polígonos convexos.

Finalmente, o terceiro e último motivo importante é o fato de que os MBRs preservam a informação de proximidade espacial, propriedade útil na avaliação de muitos predicados espaciais. Particularmente no caso do operador de interseção, que é o foco principal desse trabalho, como o MBR é uma aproximação que contém todo o polígono, podemos afirmar que sempre que dois polígonos possuírem interseção então seus MBRs também se interceptarão. Esse fato garante que o conjunto resposta gerado por uma junção espacial realizada sobre os MBRs dos polígonos irá ser na verdade um super-conjunto do conjunto resposta da junção sobre os polígonos propriamente ditos. Por esse motivo, chamamos o conjunto resposta da junção de MBRs dos polígonos de conjunto de pares de candidatos: para obter o conjunto resposta da junção espacial, basta investigar os elementos do conjunto de pares de candidatos.

Assim, invariavelmente todos os algoritmos de junção espacial que têm sido propostos nos últimos anos realizam uma pré-junção sobre os MBRs dos polígonos, e somente então passam a investigar quais os pares desse conjunto resposta que farão parte do resultado da junção original. Muitos desses trabalhos param nesse passo, não chegando a realizar os passos 2 e 3 do MSQP: realizam apenas a junção de MBRs. Por outro lado, alguns métodos não implementam o passo 2, passando o conjunto produzido pela junção de MBRs diretamente para o passo 3.

Cabe aqui observar que qualquer que seja o método para realizar a junção dos MBRs o seu resultado será necessariamente o mesmo conjunto de pares de MBRs que se interceptam. Com isso, pode-se propor e avaliar métodos diferentes para a implementação dos passos 1 e 2 do MSQP de forma independente: basta que os requisitos de espaço para o armazenamento das aproximações utilizadas no passo 2 sejam observados. É por esse motivo que muitos trabalhos realizam apenas a junção de MBRs: melhorias realizadas nesse passo irão quase que imediatamente resultar em melhorias no tempo total gasto na realização da junção, pois não irão afetar o desempenho dos passos seguintes. De fato, as únicas variações possíveis no conjunto resposta da junção de MBRs são a ordem de geração dos pares, e se relaxarmos um pouco a definição de conjunto, a ocorrência de pares repetidos.

Até onde sabemos, nenhum dos algoritmos propostos para implementar o passo 2 é sensível à ordem em que os pares são gerados, pois as aproximações usualmente são armazenadas no índice. Já o passo 3 é mais sensível a essa ordem, pois como os polígonos em geral não serão armazenados no índice, para investigar uma possível

interseção que não foi decidida nos passos anteriores será necessário realizar um acesso a disco para recuperar cada polígono. Essa é uma operação que pode ser custosa, pois envolve um acesso a disco em uma posição aleatória. Para minimizar o tempo gasto para recuperar o polígono do disco, devemos utilizar estruturas de *cache*, ou seja, manter um subconjunto dos polígonos na memória principal evitando assim que um polígono tenha de ser lido diversas vezes. Claro está que a ordem em que os pares forem apresentados ao passo 3 irá afetar o desempenho do *cache*. Afinal, se um polígono aparece em vários pares, será maior a chance dele estar no *cache* se todos pares em que o mesmo aparece forem passados para o passo 3 em tempos próximos.

No entanto, os índices utilizados para a realização da junção dos MBRs preservam alguma informação sobre a proximidade espacial útil para a avaliação do predicado de interseção. Assim, não importando qual seja o método empregado para implementar o passo 1, há uma boa chance de que os pares de candidatos sejam gerados mantendo-se uma certa proximidade dos polígonos. Desta forma, diferenças na ordem de geração dos pares somente terão impacto significativo se a proximidade dos pares que contêm um mesmo polígono não for minimamente preservada. Para diminuir tal impacto, em ABEL et al. (1996) é proposto um algoritmo que estabelece uma nova ordenação na entrada do passo 3, de forma a aproveitar melhor o cache. Na verdade, o algoritmo estudado naquele trabalho prevê que essa ordenação será aplicada na saída do passo 1, pois o modelo de processamento utilizado por eles não prevê a realização do passo 2; contudo essa é apenas uma questão de nomenclatura.

Alguns métodos de acesso espacial possuem mais de uma entrada para o mesmo objeto. Uma junção de MBRs que use algum destes métodos de acesso possivelmente irá produzir um conjunto de pares de candidatos com entradas duplicadas, ou seja, haverá pares de MBRs que serão listados duas vezes. A existência de pares duplicados no conjunto de candidatos é desastrosa para o bom desempenho dos passos 2 e 3, pois a avaliação do predicado de junção usualmente é uma operação custosa, mesmo quando é realizada sobre aproximações. Por este motivo, ainda que a resposta da junção espacial tolere a existência de pares replicados (se o objetivo da junção for a exibição de um mapa, p.e.), pode ser vantajoso eliminá-los no passo 1 para se evitar a execução repetida dos algoritmos mais lentos dos passos 2 e principalmente do passo 3. Desta forma, o quanto antes os pares replicados forem eliminados, menor

será o tempo total da junção. É por este motivo que muitos autores consideram que a eliminação dos pares duplicados é parte intrínseca da junção dos MBRs.

Finalmente, cabe aqui dizer que os passos do MSQP são em geral realizados de forma intercalada, de forma que podemos pensá-los como processos independentes sendo realizados de maneira concorrente, onde a saída do passo 1 está conectada à entrada do passo 2 e a saída do passo 2 está conectada à entrada do passo 3. Em um sistema de computação sequencial, contudo, o que realmente ocorre é uma alternância entre a execução dos módulos correspondentes a cada passo. A vantagem dessa abordagem é que torna desnecessário materializar um conjunto possivelmente grande de pares de MBRs candidatos. Por outro lado, a desvantagem surge quando o método de realização da junção de MBRs cria duplicatas, pois uma alternativa simples para eliminá-las é materializar a coleção sob a forma de um índice. Ilustrando essa alternativa, em MING e RAVISHANKAR (1996) a coleção de pares de MBRs candidatos é materializada e ordenada para a eliminação de duplicatas. A Figura 3-6 ilustra esse processo.

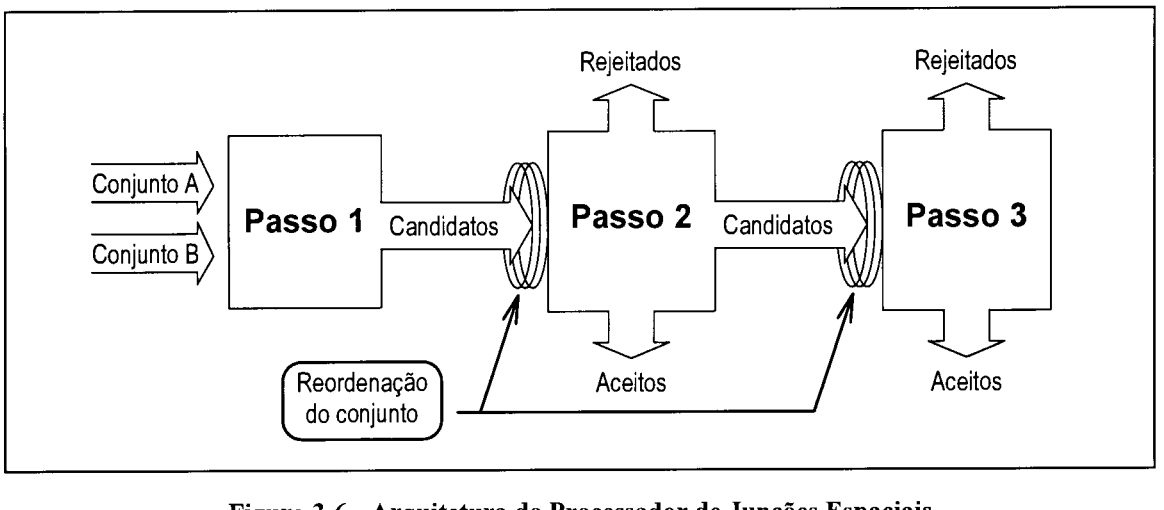
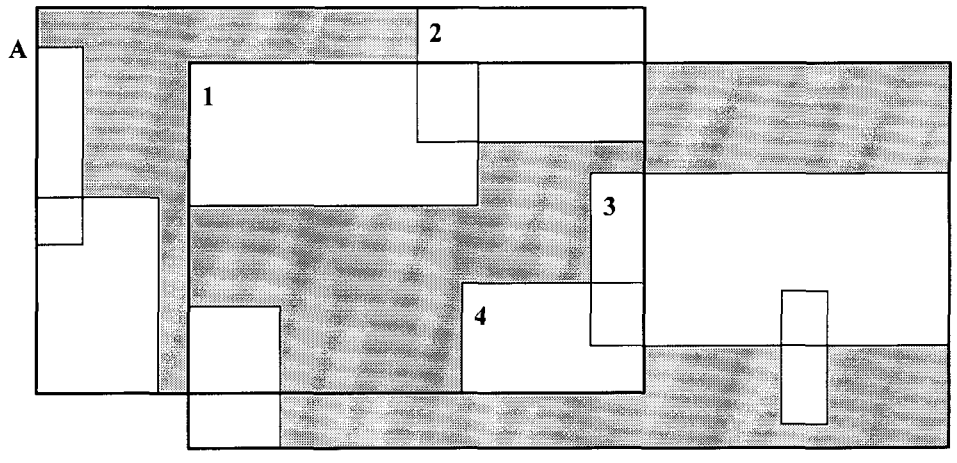


Figura 3-6 - Arquitetura do Processador de Junções Espaciais

É importante observar que há casos de algoritmos em que as ordenações entre os passos podem ser totais ou parciais, ou seja, sobre toda a coleção ou sobre apenas um segmento ou subconjunto dela. A vantagem de se utilizar apenas um segmento da mesma é que podemos escolher o subconjunto de modo que caiba na memória, evitando assim materializar a coleção inteira. De fato, mesmo que a ordenação seja utilizada para remover pares duplicados, é possível segmentá-la de tal forma que apenas parte da coleção seja necessária a cada instante — como veremos mais adiante, na seção 4.1.2.

3.6.3 A Realização da Junção Espacial sobre os MBRs através de R-Trees

Sendo a R-Tree uma estrutura recursiva, o algoritmo que a utiliza para realizar a junção espacial também é recursivo. Assim, dadas duas árvores representadas pelos retângulos associados aos respectivos nós de mais alto nível (o nó raiz), iremos investigar os filhos de cada um desses nós, aos pares como em um produto cartesiano. Realizaremos junções nas sub-árvores representadas pelos pares de nós filhos cujos retângulos possuam interseção. Em cada junção de sub-árvore iremos repetir o processo recursivamente até chegarmos às folhas, quando então iremos obter os pares de MBRs que formam o conjunto resposta da junção de MBRs. A Figura 3-7 ilustra esse processo.



A junção dos blocos **A** e **B** será efetivada mediante a realização de duas junções nas subárvores representadas pelos blocos **1** e **2** e pelos blocos **3** e **4**. Os demais blocos não serão avaliados pois os respectivos retângulos não se interceptam.

Figura 3-7 - Junção através de uma R-Tree

Uma das vantagens deste algoritmo é que cada par de MBRs somente será listado uma vez no conjunto de candidatos, pois cada MBR só pode ser inserido em um nó. A grande desvantagem do algoritmo é resultante da sobreposição dos retângulos correspondentes aos nós da R-Tree. Para investigarmos os MBRs que estão em determinada área onde haja sobreposição de dois ou mais nós teremos de percorrer mais de uma sub-árvore. Quando formos realizar uma junção, o problema irá se agravar, e de forma quadrática: se m retângulos de uma árvore possuem interseção com um ponto i e se n retângulos da outra árvore também possuem interseção com esse mesmo ponto i, então precisaremos de realizar $m \times n$ junções nas sub-árvores.

Ler um nó do disco mais de uma vez é definitivamente prejudicial para o desempenho do algoritmo. No entanto, esse é um problema que não poderá ser resolvido

apenas com o uso de *cache* ou mesmo de uma ordenação mais eficiente, pois se ocorrer sobreposição em dois nós próximos da raiz possivelmente toda a sub-árvore terá de ser lida, o que pode representar uma quantidade muito grande de nós para serem mantidos no *cache*. Assim, garantir que cada nó de uma R-Tree será lido apenas uma vez é uma tarefa bastante difícil. Podemos verificar isso em HUANG e JING (1997), onde é apresentado um novo algoritmo para a realização da junção espacial com R-Trees na tentativa de minimizar os acessos a disco. Isso é feito alterando-se a ordem em que as junções nas sub-árvores são realizadas. É interessante observar que esse problema está diretamente relacionado ao da ordenação (ou falta de) dos pares de MBRs candidatos.

Apesar da R^+-Tree ser uma estrutura sem sobreposição, isso é conseguido através da duplicação das entradas para os objetos espaciais, ao nível das folhas. Daí resulta uma árvore com mais objetos do que uma R-Tree comum, o que pode levar a uma altura também maior (e consequentemente, mais acessos a disco). Além disso, teremos de lidar com o problema da remoção de pares de elementos duplicados do conjunto de MBRs candidatos. Essa abordagem irá ter portanto mais semelhança com as abordagens de *hash*, sem no entanto ter as vantagens das mesmas, ou seja, o acesso ao disco com complexidade $O(n)$, a construção do índice também com complexidade $O(n)$ e a implementação mais simples.

Finalmente, cabe dizer que não há nenhuma diferença fundamental no caso de não termos dois conjuntos distintos de entrada, ou seja, a junção a ser realizada irá avaliar o predicado utilizando pares de elementos do mesmo conjunto. A grande diferença que irá ocorrer é que será necessário realizar testes para se evitar que um par no conjunto resposta da junção de MBRs seja formado por MBRs do mesmo polígono, ou seja, determinarmos que um polígono intercepta ele mesmo. Como vantagem gratuita, o uso do *cache* tenderá a ser mais eficiente, já que teremos apenas uma estrutura a percorrer compartilhando o mesmo espaço reservado para o *cache*.

3.6.4 Uso de Aproximações

Uma aproximação de um polígono (ou de qualquer outro objeto espacial) é uma representação simplificada do mesmo, usualmente ocupando menos espaço de armazenamento, e que possui determinadas propriedades características de cada tipo de aproximação. Essas propriedades podem ser úteis para a avaliação de um determinado

predicado espacial sem que haja a necessidade de se ter acesso à representação do polígono.

Um método de acesso espacial organiza os objetos em uma estrutura espacial de acordo com chaves geométricas (BRINKHOFF et al., 1993b), que nada mais são do que aproximações. Devido à sua simplicidade, o MBR é a aproximação mais utilizada pelos métodos de acesso espacial. Usando MBRs, a complexidade de um objeto espacial é reduzida para quatro parâmetros onde as mais importantes características do objeto (posição e extensão) são mantidas. No entanto, objetos cartográficos do mundo real são muito mal aproximados através de MBRs. Iremos portanto investigar o uso de outras aproximações para os objetos espaciais.

Podemos dividir as técnicas de aproximações em três classes (BRINKHOFF et al., 1993b): conservadoras, progressivas e generalizadoras. Uma aproximação é dita conservadora se e somente se o contorno do objeto original está inteiramente contido na mesma. Analogamente, uma aproximação é dita progressiva se todos os pontos pertencentes à aproximação estão contidos no objeto. Uma aproximação generalizadora tenta simplificar o contorno do objeto, por exemplo reduzindo o número de vértices. Em geral, não existe nenhuma relação topológica entre a aproximação generalizadora e o objeto original, ou seja nem o objeto está inteiramente contido na aproximação nem a aproximação está inteiramente contida no objeto. Por esse motivo, aproximações generalizadoras em geral não podem ser utilizadas para ajudar na avaliação de predicados como a interseção de polígonos.

Exemplos de aproximações conservadoras são os menores retângulos envolventes (**MBR**, de *minimum bounding rectangle*). Um outro exemplo de aproximação conservadora são os polígonos convexos de *n*-vértices (***n*C**, de *n corners*), que é o menor (em área) polígono com *n* vértices que contém o objeto sendo aproximado. Uma variante da aproximação *n*C é o envelope convexo, que é menor polígono convexo que contém o objeto (**CH**, de *convex hull*). Exemplos de aproximações progressivas são os maiores retângulos contidos no polígono (**ER**, de *enclosed rectangle*) e as maiores segmentos de linha ortogonais contidos no polígono (**EL**, de *enclosed line*). Outros exemplos e propriedades de aproximações podem ser encontrados em BRINKHOFF et al. (1993b). A Figura 3-8 ilustra algumas aproximações para um polígono.

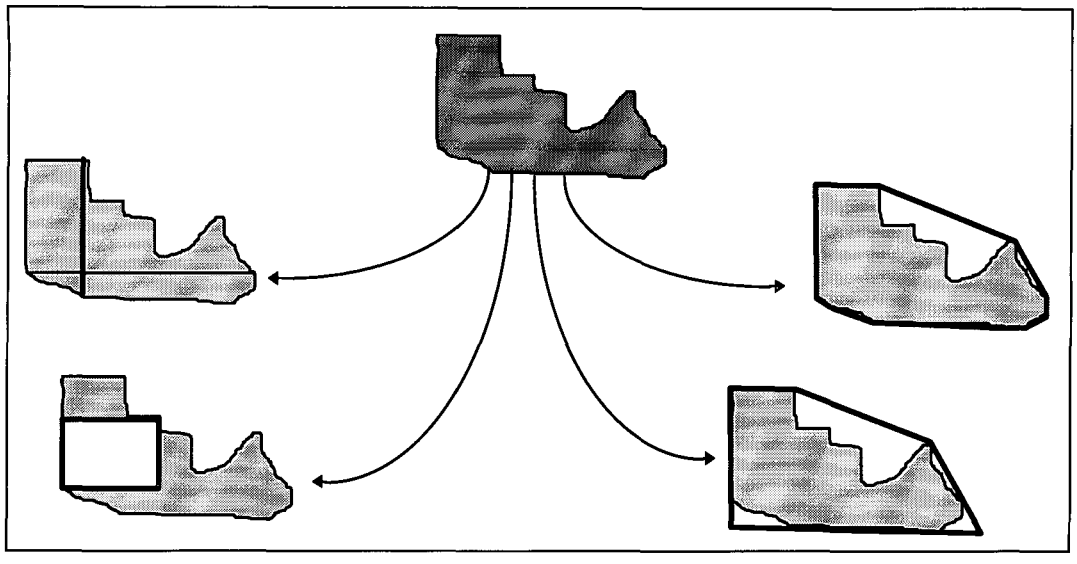


Figura 3-8 Exemplos de aproximações. No sentido antihorário temos EL, ER, 5C e CH

Em se tratando de predicados de interseção de polígonos, como regra geral, temos que aproximações conservadoras podem auxiliar principalmente na identificação de casos onde não há interseção: se não houver interseção entre os MBRs de dois polígonos, não é possível que haja interseção entre eles. Já as aproximações progressivas podem auxiliar principalmente na identificação de casos onde há interseção: se os **ERs** de dois polígonos possuem interseção, certamente os polígonos se interceptam. Por outro lado, as aproximações generalizadoras são de pouca utilidade nesse tipo de predicado. Por esse motivo, em geral, para se construir um filtro geométrico para auxiliar na avaliação de junções espaciais com o predicado de interseção, duas aproximações devem ser utilizadas: uma conservadora e outra progressiva. Na Figura 3-9 vemos exemplos do uso de aproximações para classificar pares de candidatos: A) rejeitar; B) testar; C) Aceitar. No capítulo quatro, apresentaremos uma nova aproximação que substitui as aproximações conservadora e progressiva com vantagens para a avaliação de tais predicados.

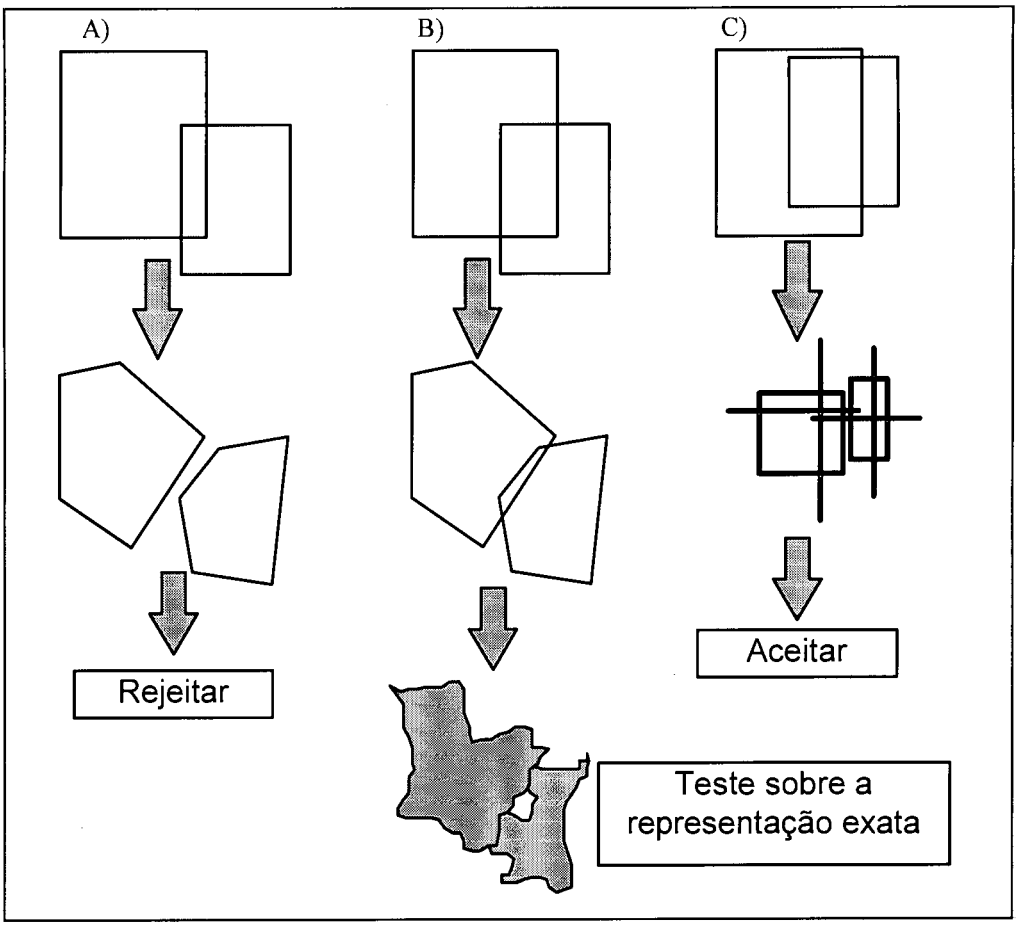


Figura 3-9 - Uso das aproximações para classificar pares de candidatos: 5C (A e B) e EL-ER (C)

3.6.5 Resultados Relevantes do MSQP

Dois grandes resultados vieram de BRINKHOFF et al. (1994): primeiro, a conclusão que o teste de interseção exata é o passo mais demorado no processo de junções espaciais; e segundo, a estrutura modular proposta para o processador. Em outras palavras, seu trabalho nos mostra direções de pesquisa para melhorar o processo de junções espaciais: a junção de MBRs usando R*-Trees ou outra estrutura de índice, e o passo de filtro. Melhorias feitas no terceiro passo tendem a ser de pequeno impacto no tempo total gasto no processo de junções espaciais, já que seus efeitos podem ser cancelados por melhorias feitas nos passos iniciais.

3.6.6 Estado da Arte

Na Figura 3-10 resumimos outros trabalhos relacionados que descrevem algoritmos que podem ser usados para substituir quaisquer dos passos do MSQP. Comentamos alguns destes trabalhos brevemente e indicamos qual passo pode substituir cada um deles. Esta não é uma lista ampla de artigos recentemente publicados

relacionada a esta área, mas inclui os resultados mais importantes alcançados, até onde sabemos.

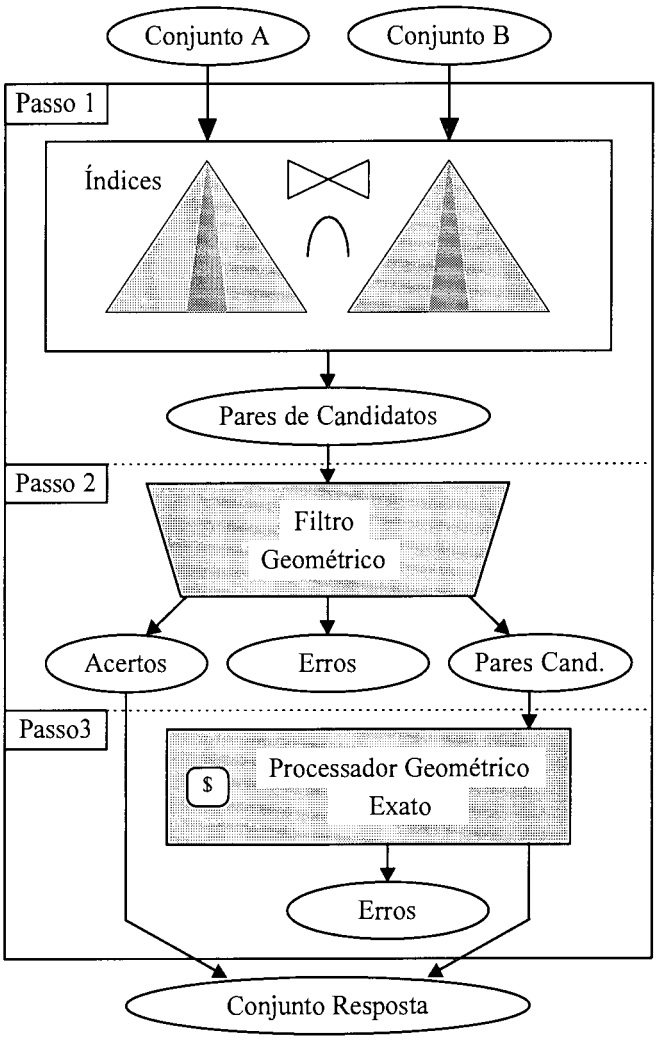


Figura 3-10 Estado de arte em Processamento de Junções Espaciais

Passo 1: neste passo, BRINKHOFF et al. (1993a) e BRINKHOFF et al. (1994) usam R*-Tree como índice para executar uma junção espacial. PATEL e DEWITT (1996), e MING e RAVISHANKAR (1995) e MING e RAVISHANKAR (1996) apresentam abordagens baseadas em *hash*, enquanto BRINKHOFF et al. (1996) apresentam junções espaciais baseadas em algoritmos paralelos, usando R*-Trees. BERCHTOLD et al. (1996) sugere outra árvore, a X-Tree que também pode ser usada para implementar este passo. HUANG e JING (1997) propõem um algoritmo novo para a travessia da R*-Tree chamado de BFT. KOUDAS e SEVCIK (1997) utilizam um algoritmo baseado na separação por tamanho dos MBRs em uma árvore especial chamada de árvore de filtro. Em ARGE et al. (1998) é proposto um algoritmo baseado no conhecido algoritmo de *plane-sweep* para realizar a junção quando nenhum índice

existe sobre os conjuntos. Finalmente, MAMOULIS e PAPADIAS (1999) propõem um novo algoritmo para ser utilizado quando apenas um dos conjuntos possui índice, situação típica que ocorre quando uma consulta está encadeada com outra.

Passo 2: neste passo, BRINKHOFF et al. (1994) usam aproximações de polígonos de cinco-vértices, e BRINKHOFF et al. (1993b) discutem outras aproximações que podem ser usadas para filtrar o conjunto de candidato. VEENHOF et al. (1995) usam aproximações que são construídas girando duas linhas paralelas ao redor do objeto. ESPERANÇA e SAMET (1997) usam polígonos ortogonais para filtrar algumas consultas espaciais. Finalmente, em BRINKHOFF et al. (1995) é apresentado um estudo sobre a complexidade de polígonos, o que está indiretamente relacionado a qualidade das aproximações.

Passo 3: neste passo, BRINKHOFF et al. (1994) usam o algoritmo de *plane-sweep* e conjuntos de trapezóides para computar o teste de interseção exato, e HUANG et al. (1997a) usam um algoritmo chamado Descoberta Simbólica de Interseções para reduzir o tempo gasto em um algoritmo baseado no de *plane-sweep*.

De forma coerente com a constatação de BRINKHOFF et al. (1994) de que as melhorias nas primeiras fases do algoritmo tendem a anular as melhoria feitas nas fases seguintes, podemos notar que a maior parte dos esforços têm se concentrado no passo 1, seguido de longe pelo passo 2, enquanto que o passo 3 tem recebido pouca atenção. Nesta Tese, não lidaremos com o passo 3 por se tratar de um problema mais relacionado à área de Computação Gráfica do que à área de Banco de Dados.

3.7 Sumário

Neste capítulo fizemos uma revisão detalhada do problemas de processamento e otimização de consultas, comparando o tratamento dado a casos não espaciais e espaciais. Em especial, o capítulo analisou a operação de junção espacial.

Apresentamos uma abordagem para o processamento de junções espaciais que será utilizada no restante desta Tese. Esta abordagem é baseada em 3 passos: junção de MBRs, filtragem por aproximações e comparação geométrica exata. Ressaltamos que as únicas variações possíveis no conjunto resposta da junção de MBRs, se relaxarmos um pouco a definição de conjunto, são a ordem de geração dos pares e a ocorrência de pares repetidos. Portanto, no capítulo seguinte da tese iremos propor novos algoritmos para uma melhor implementação dos passos 1 e 2 do MSQP.

4. Melhorias no Processamento de Junções Espaciais

Este capítulo apresenta as principais contribuições dessa tese. Os algoritmos e resultados aqui apresentados deram origem a dois artigos submetidos, aprovados e apresentados em conferências de Banco de Dados: ZIMBRÃO (1997) e ZIMBRÃO (1998).

Tais algoritmos estão em conformidade com a abordagem para o processamento de junções espaciais apresentada no capítulo anterior, chamada de MSQP. Esta abordagem é baseada em 3 passos: junção de MBRs, filtragem por aproximações e comparação geométrica exata.

No primeiro artigo propomos e avaliamos um novo método, baseado na abordagem *hash-join*, para a realização da junção espacial sobre os MBRs dos polígonos, o que corresponde ao passo 1 do MSQP. Nós o chamamos de 2DHM, um acrônimo para Matriz de Hash Bidimensional (*Two Dimensions Hash Matrix*). Os resultados práticos obtidos foram bons: o número de comparações de MBRs foi proporcional ao tamanho do conjunto resposta, o espaço utilizado foi próximo do obtido por outros índices e o tempo total para a execução das consultas foi menor, o que torna a nossa proposta uma alternativa viável para a avaliação de junções espaciais.

No segundo artigo nós propomos e avaliamos uma nova aproximação geométrica para polígonos chamada de 4CRS, um acrônimo para assinatura raster de quatro cores (*Four Color Raster Signature*), que substitui com vantagens as atualmente utilizadas no passo 2 do MSQP. Os resultados práticos foram bons: a nova aproximação não acarretou em nenhum incremento de tempo nem de espaço, e possui taxa de identificação de candidatos bem maior do que as concorrentes, resultando em uma diminuição significativa do tempo total gasto em uma junção espacial.

Assim, este capítulo está dividido em duas seções, delas correspondendo ao tema principal de cada um dos algoritmos gerados. Além disto, as seções contêm alguns resultados ou melhoramentos menores que não apareceram nos artigos, principalmente por restrições de espaço peculiares a trabalhos apresentados em conferências. Na seção 4.1 apresentaremos a matriz de *hash* bidimensional e na seção 4.2 a assinatura raster de quatro cores.

4.1 Matriz de Hash Bidimensional

Nesta seção apresentaremos a abordagem 2DHM, os problemas resultantes da mesma, as soluções propostas para tais problemas, e justificaremos algumas decisões de implementação, tais como a não utilização das técnicas de *hash* já desenvolvidas para os dados pontuais no espaço. Por fim, apresentaremos também os resultados práticos obtidos.

4.1.1 Conceitos Iniciais

Como toda abordagem baseada em *hash-join*, a nossa se divide em duas fases: partição e junção. As dificuldades inerentes às abordagens *hash-join*, como replicação dos MBRs e dispersão dos dados espacialmente próximos por muitos blocos no disco, serão atacadas por meio de uma distribuição hierárquica de blocos de *overflow*, agrupamentos de blocos e blocos de tamanhos diferentes porém predeterminados. Embora a nossa proposta possua um desempenho muito bom nas duas fases, é na fase de partição que o seu desempenho é melhor, sendo portanto sua aplicação uma ótima alternativa quando nenhum dos conjuntos de entrada possui índices.

Como vimos anteriormente, uma das possíveis alternativas quando se aplica uma abordagem *hash-join* é a inserção de um MBR de um polígono em mais de um bloco de disco, ou seja, o relaxamento da condição de "atribuição única". A nossa abordagem relaxa exatamente essa condição, e prevê que um polígono poderá ser inserido em vários blocos no disco.

O número médio de *buckets* em que cada polígono é inserido configurará o **inchaço** dos dados (*data inflating*). Esse número representa a taxa de replicação dos dados. Claro está que esse inchaço deve ser minimizado, pois ele significa redundância de informação e aumento do espaço utilizado em disco, o que poderá resultar em um acréscimo do número de operações de entrada e saída.

A idéia principal é dividir o espaço de busca em células de tamanho fixo, formando uma grade regular, ou simplesmente *grid*. Por convenção, cada célula será unicamente identificada pelas coordenadas de seu canto inferior esquerdo, sendo essas coordenadas a chave da célula. Será aplicada a função de *hash* na chave de cada célula que um determinado polígono interceptar, de forma a produzir um conjunto de *buckets* onde tal polígono deve constar. A Figura 4-1 ilustra esse processo.

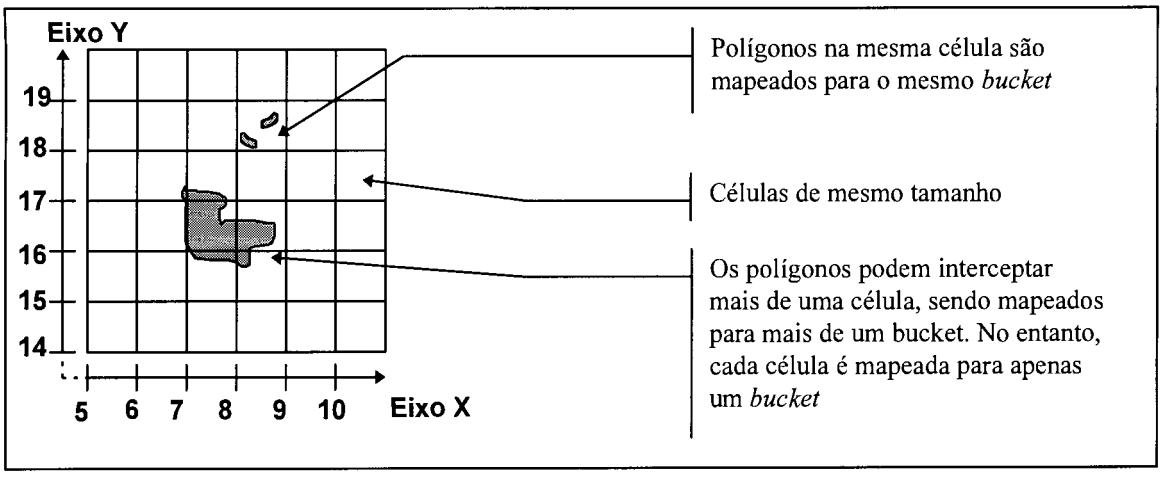


Figura 4-1 - Mapeamento de um polígono

4.1.1.1 Bit-interleaving

A abordagem tradicional para o hash espacial quando os dados são pontos no espaço é realizar uma combinação intercalando cada um dos bits das coordenadas de um ponto (*z-ordering*, *bit-interleaving*), ou aplicar uma função de ordenação, como curvas de Hilbert, gerando uma chave escalar SAMET (1990). Após a criação da chave escalar para cada ponto, aplica-se à mesma a função de *hash*. Espera-se assim obter uma distribuição homogênea, com poucas colisões, embora haja uma perda da proximidade espacial. Essa abordagem, contudo, não pode ser utilizada diretamente em polígonos, pois embora pontos e polígonos possuam natureza espacial, o predicado mais utilizado para pontos é o de igualdade, sendo portanto muito semelhante ao predicado utilizado para dados escalares na abordagem relacional. Já na junção de polígonos o predicado mais comum é a interseção, que, como vimos, possui muitas particularidades.

4.1.1.2 Matriz de Hash *Bidimensional*

Para abordar tais particularidades propomos uma forma alternativa de realizar o *hash* espacial. Nossa proposta consiste em não reduzir a informação bidimensional. Preservando a dimensão original podemos preservar também informação sobre a proximidade espacial. Dessa forma, ao invés de uma tabela de *hash*, mapearemos as chaves para uma matriz de *hash*. Por exemplo, tomando a célula de coordenadas x e y, teremos: $(x, y) \rightarrow (h_1(x), h_2(y))$, onde h_1 e h_2 são funções simples de *hash*, usualmente envolvendo o resto da divisão por um número primo. Note que utilizando duas funções diferentes a matriz de *hash* não precisa ser quadrada, podendo obedecer a algum critério de proporcionalidade em relação aos conjuntos de entrada. A Figura 4-2 ilustra o mapeamento de diversas células para a mesma posição na matriz de *hash*.

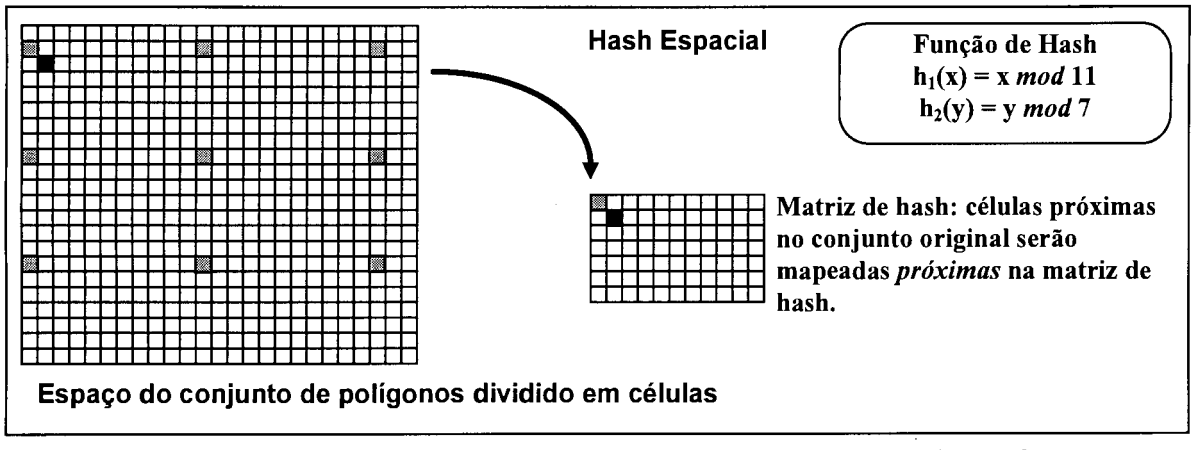


Figura 4-2: Cada posição da matriz de *hash* será um *bucket* contendo várias células.

Certamente, a matriz resultante será armazenada no disco de forma sequencial segundo algum critério simples, como por exemplo uma linha após a outra. Portanto, quando afirmamos que polígonos próximos serão mapeados para células próximas, na verdade queremos dizer que tais células são próximas na matriz de *hash*, e não no disco. Em outras palavras, o que é preservado é a *informação* da proximidade espacial, e não a proximidade de agrupamento em si. Como veremos adiante, esta informação é de grande utilidade para a eliminação de pares duplicados.

4.1.1.3 Conjuntos de Dados para Testes

Utilizamos dados de municipalidades (ou equivalente) de alguns países europeus, condados de alguns estados americanos, províncias da América do Sul e a malha municipal brasileira (IBGE, 1996). Além disso, alguns conjuntos de dados foram criados artificialmente conforme proposto em BRINKHOFF et al. (1994), ou seja, através de um deslocamento aleatório nas coordenadas x e y, e possivelmente uma rotação também aleatória. Tipicamente, a média de pontos por polígono de cada conjunto de dados varia de 22 a 96, conforme a Tabela 4-1.

Tabela 4-1- número de polígonos e vértices em cada conjunto de dados.

Conjunto	# Polígonos	# médio de vértices
Sul dos EUA	496	22
Europa-B	249	96
América do Sul	228	54
Brasil	5081	80
Brasil-A	15647	80
Brasil-B	23007	82

4.1.2 Tamanho das Células

Ao utilizar tal abordagem diretamente verificamos que o inchaço nos dados foi muito grande, o que também é apontado em MING (1996) e PATEL (1996), tornando o

algoritmo muito ineficiente tanto do ponto de vista do uso de espaço em disco (e consequentemente, em operações de entrada e saída) quanto do número de comparações. O problema pode ser caracterizado da seguinte forma (GAEDE, 1995): se utilizarmos células pequenas quando comparadas ao tamanho médio dos polígonos, cada célula no espaço de busca será interceptada por poucos polígonos, embora cada polígono intercepte várias células inchando os dados. Por outro lado, se utilizarmos células grandes, teremos um inchaço menor nos dados, mas em compensação cada célula conterá mais polígonos. A Figura 4-3 ilustra essa situação para o conjunto de dados Brasil, que contém 5081 polígonos representando municípios.

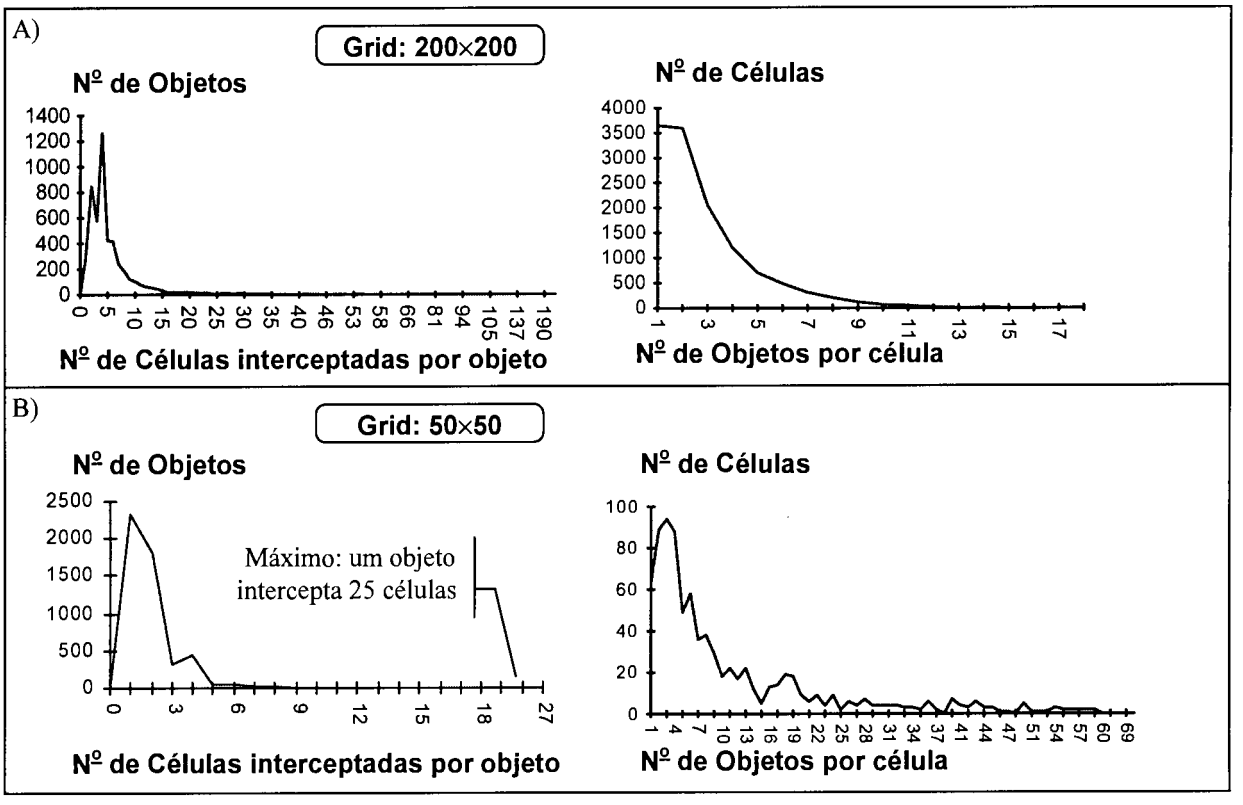


Figura 4-3 - Distribuição das células e objetos para o conjunto de dados Brasil.

Um meio termo deve ser encontrado. Para tanto, em geral é necessário ter um grande conhecimento prévio sobre os dados. Conforme podemos ver na Figura 4-3-A, ao utilizarmos células pequenas temos um grande número de células (mais de 3500) interceptando apenas um polígono, porém teremos polígonos interceptando mais de 190 células. Já na Figura 4-3-B, ao utilizarmos células maiores, teremos poucas células interceptando apenas um polígono (em torno de 60), porém apenas um objeto intercepta mais de 23 células.

Em outras palavras, se tivermos muitas células, várias delas serão mapeadas para o mesmo *bucket*, aumentando consideravelmente a ocupação dos mesmos. Além

disso, como cada polígono interceptará muitas células, o resultado é um aumento enorme no número de comparações redundantes, ou seja, o mesmo par de MBRs será investigado mais de uma vez. Por outro lado, se utilizarmos células grandes, teremos um grande número de polígonos interceptando a mesma célula, e como consequência, será preciso um grande número de comparações dentro da célula para descobrirmos os MBRs que se interceptam. É necessário portanto uma escolha criteriosa para obtermos um tamanho de célula eficiente.

A Tabela 4-2 ilustra esse fato para estes conjuntos de dados, supondo uma junção do conjunto com ele mesmo. Vale observar que esse é um dos testes de pior caso, pois se houver uma área com grande concentração de polígonos, ela o será nos dois conjuntos. Como estimativa otimista, assumiremos uma distribuição uniforme dos polígonos pelas células. Embora pela Figura 4-3 possamos constatar que isso não corresponde à realidade, esta estimativa representa o menor número possível de comparações para determinado *grid*, de forma que teremos um parâmetro para avaliar o quão boa é uma determinada abordagem. Se α é o número de células ocupadas, η_i a ocupação da célula *i* e μ é a ocupação média das células (contando apenas as células ocupadas), o número estimado de comparações é $\alpha.\mu.(\mu-1)/2$, correspondendo às linhas "# comp. estimada", e o número real de comparações é $\Sigma(\eta_i-1)^2/2$, correspondendo às linhas "# comp. real". Convém lembrar que não é necessário comparar o polígono a si próprio.

Tabela 4-2- Número de comparações em função do tamanho das células do *grid*.

Tamanho do *Grid*		Sul dos EUA	Europa-B	América do Sul	Brasil	Brasil-B
	Ocupação média	1,47...	3,60...	1,38...	2,78...	4,01
200	# comp. Estimada	4.247	56.785	2.355	30.817	194.155
x	# comp. real	6.928	108.202	4.021	55.501	365.451
200	**Inchaço dos MBRs**	**36,19×**	**175,36×**	**54,68×**	**6,82×**	**5,60×**
	Inchaço dos dados	12,73×	59,12×	18,89×	2,94×	2,53×
	Ocupação média	2,00...	4,00...	1,72...	5,16...	6,85...
100	# comp. estimada	3.129	18.780	1.460	34.679	190.382
x	# comp. real	4.440	36.610	2.411	71.588	375.074
100	**Inchaço dos MBRs**	**12,62×**	**50,25×**	**17.70×**	**3,28×**	**2,83×**
	Inchaço dos dados	4,87×	17,42×	6,57×	1,76×	1,61×
	Ocupação média	3,18...	4,88...	2,51...	11,77...	14,83...
50	# comp. estimada	2.881	8.055	1.181	54.253	286.246
x	# comp. real	3.594	16.656	1.968	130.534	606.312
50	**Inchaço dos MBRs**	**5,31×**	**16.65×**	**6,85×**	**1,98×**	**1,80×**
	Inchaço dos dados	2,44×	6.22×	2,95×	1,33×	1,27×
	Ocupação média	6,07...	6,85...	4,36...	30,51...	39,43...
25	# comp. estimada	3.509	4.710	1.260	108.525	606.079
x	# comp. real	4.347	10.391	2.163	293.322	1.406.822
25	**Inchaço dos MBRs**	**2,79×**	**6,47×**	**3,29×**	**1,45×**	**1.37×**
	Inchaço dos dados	1,60×	2,82×	1,76×	1,15×	1,12×
	Ocupação média	14,00...	11,60...	8,28...	88,23...	123,25...
13	# comp. estimada	5.733	4.183	1.599	269.362	1.657.404
x	# comp. real	7.147	9.473	2.828	747.141	4.069.951
13	**Inchaço dos MBRs**	**1,78×**	**3,17×**	**1,93×**	**1,22×**	**1,18×**
	Inchaço dos dados	1,26×	1,72×	1,31×	1,07×	1,06×
MBRs que se interceptam		1.475	3.201	722	18.935	130.140

Como vimos na subseção 3.4.5, item 3, a eficiência do algoritmo de *hash-join* depende da relação entre o número de células de uma partição e da cardinalidade dos conjuntos participantes da junção. Portanto, o tamanho das células deverá depender da área total ocupada pelos conjuntos de dados, ou melhor, da área onde os conjuntos possuem interseção, e da cardinalidade dos conjuntos. Retomando a última equação do item 3 da subseção 3.5.5, temos como estimativa para o número máximo de comparações:

$$\left(m^3e^2 + 2m^2ek + mk^2\right)/c\,.$$

Esse resultado nos leva a concluir que quanto menos partições, ou seja, c maior, menor será o número máximo de comparações necessárias, o que poderia resultar em um menor número médio de comparações. No entanto, observando a Tabela 4-2 podemos observar que isso não corresponde à realidade — há espaço para uma escolha de *grid* que resulte em um número menor de comparações com mais células: tanto a

coluna **Sul dos EUA** quanto a coluna **América do Sul** possuem o menor número de comparações com um *grid* de 50 por 50 células. De fato, podemos observar pela Figura 4-4 que o gráfico *tamanho da célula* × *número de comparações* é uma curva com cavidade para cima.

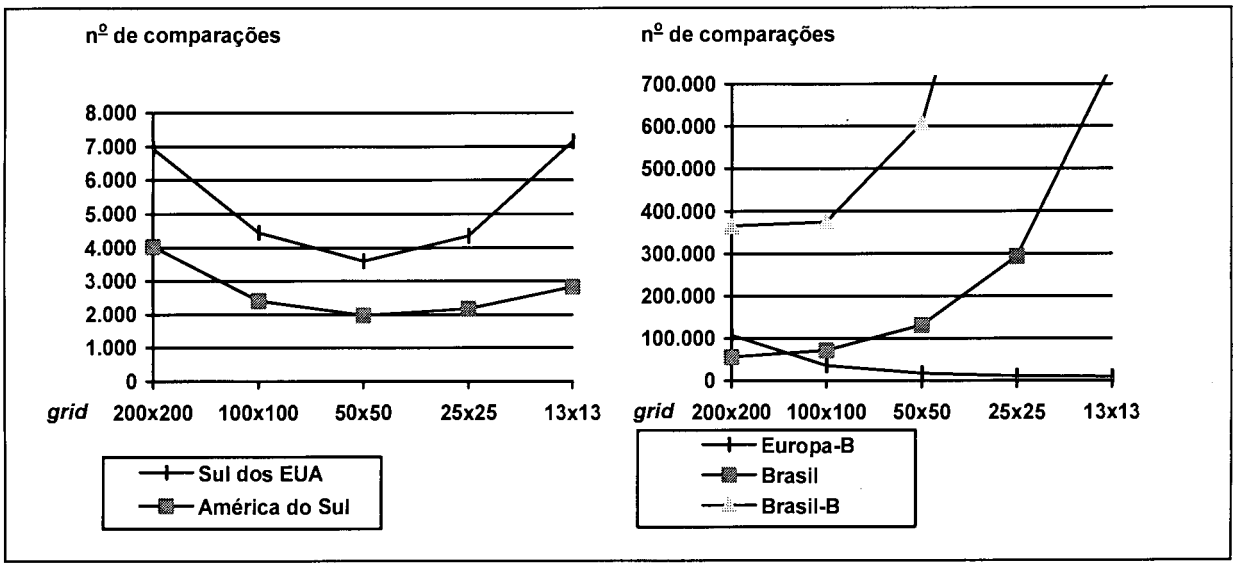


Figura 4-4 - Gráfico do *tamanho da célula* × *número de comparações*

Certamente, haverá então um *grid* ótimo a ser escolhido. O aparente erro foi introduzido ao assumirmos que a, o fator de incremento ou *inchaço* nos dados, é linearmente dependente de p, quando na verdade a real relação será mostrada no item seguinte.

4.1.2.1 Estimativas para Calcular o Inchaço dos dados

Usando uma abordagem probabilística, o inchaço dos dados causado por um *grid* regular pode ser estimado para conjuntos de dados que não tenham nenhuma anomalia grave, ou melhor, cujas anomalias sejam pequenas em relação ao restante do conjunto — por anomalia entende-se uma grande concentração de polígonos se interceptando uns aos outros. De fato, veremos aqui uma abordagem que se mostrou bem ajustada para todos os conjuntos de dados que nós utilizamos como teste, com erro máximo de ±3,5%. A Figura 4-5 ilustra algumas das nossas premissas.

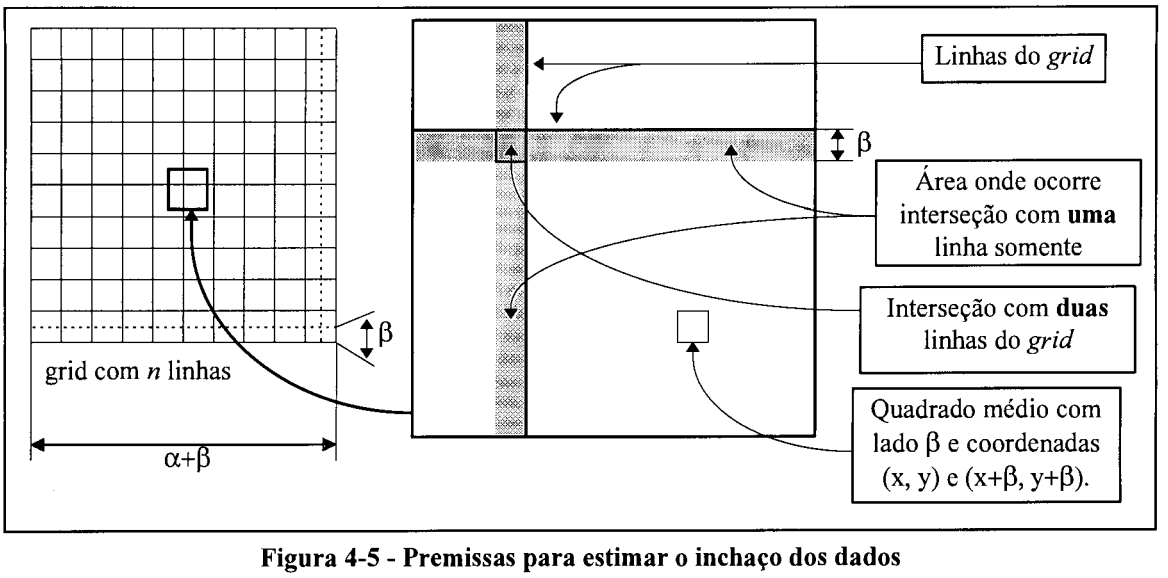


Figura 4-5 - Premissas para estimar o inchaço dos dados

Como estamos lidando com estimativas e probabilidades, iremos simplificar bastante o modelo. O conjunto de dados será formado por quadrados com lado médio β, ao invés de retângulos. O espaço a ser dividido será um quadrado de lado α+β, e iremos dividi-lo em $(n+1)^2$ células, de forma a termos um *grid* com n linhas horizontais e n verticais. Além disso, assumiremos que o lado de cada célula do *grid*, dado por (α+β)/(n+1), é maior do que β. Nos interessa saber a probabilidade de uma linha do *grid* cruzar um retângulo, dividindo-o em dois e provocando a inserção do mesmo em mais de uma célula. Mais ainda, queremos calcular o número esperado de quadrados que serão divididos – essa é a medida do inchaço dos dados.

Ao traçarmos o *grid*, dado um retângulo fixo r, seja e_1 o evento de uma linha vertical de coordenada x igual a v cruzar esse retângulo. Considerando-se o retângulo definido pelas coordenadas (x, y) e (x+β, y+β), de acordo com a nossa modelagem o evento e_1 é equivalente ao evento do ponto (x, y) estar contido na faixa vertical definida pelo intervalo fechado $[v-\beta, v]$ no eixo x (faixa em cinza na Figura 4-5). A probabilidade desse evento ocorrer é $P(e_1) = \beta/\alpha$, como pode ser facilmente deduzido da Figura 4-5, pois β é o comprimento da faixa onde ocorre interseção com a linha escolhida e α é o intervalo horizontal disponível para que o retângulo esteja contido no espaço estabelecido. Chamaremos β/α de γ. De forma análoga, sendo e_2 o evento do retângulo ser interceptado por uma linha horizontal, temos $P(e_2) = \gamma$.

Por serem eventos independentes, a probabilidade de um retângulo ser cortado por duas linhas, necessariamente uma horizontal e uma vertical, é $P(e_1 \cap e_2) = \gamma^2$. Ou alternativamente, a probabilidade do quadrado hachurado da Figura 4-5 conter o ponto

(x, y) é dada por β^2/α^2, ou seja, γ^2. Também é fácil observar que se tivermos n linhas horizontais e verticais, as probabilidades serão: $P(e_1) = P(e_2) = n\gamma$ e $P(e_1 \cap e_2) = n^2\gamma^2$. Finalmente, a probabilidade de um retângulo interceptar uma e apenas uma linha será $P(e_1 \cup e_2) = P(e_1) + P(e_2) - 2 \times P(e_1 \cap e_2) = 2n\gamma - 2n^2\gamma^2$.

Podemos agora calcular o inchaço dos dados da seguinte forma: cada retângulo que não interceptar nenhuma linha é contado apenas uma vez; cada retângulo que interceptar apenas uma linha deve ser contado duas vezes e cada retângulo que interceptar duas linhas deve ser contado quatro vezes. A Tabela 4-3 ilustra esses cálculos.

Tabela 4-3 - Número esperado de vezes que um retângulo será dividido (inchaço dos dados)

Número de Linhas interceptadas	Probabilidade de ocorrência	Número de partes em que o retângulo será dividido	Expectativa de partes em que cada retângulo será dividido (inchaço dos dados)
0	$1 - 2n\gamma + n^2\gamma^2$	1	$1 - 2n\gamma + n^2\gamma^2$
1	$2n\gamma - 2n^2\gamma^2$	2	$4n\gamma - 4n^2\gamma^2$
2	$n^2\gamma^2$	4	$4n^2\gamma^2$
Total			$1 + 2n\gamma + n^2\gamma^2 = (1+n\gamma)^2$

Portanto, podemos estimar o inchaço dos dados para cada partição sabendo-se o lado médio de cada quadrado e a extensão total do espaço ocupado pelo conjunto de dados. Note que $\gamma^2 = \beta^2/\alpha^2$ é a área média sobre a área total, de forma que essa estimativa pode ser aplicada a conjuntos de retângulos também. A Tabela 4-4 reflete o cálculo desta estimativa aplicado aos conjuntos de dados da Tabela 4-1.

Tabela 4-4 - Estimativa do Inchaço dos Dados

Conjunto de Dados	Número de células $(n+1)^2$	Inchaço Real	Inchaço Estimado $(1+n\gamma)^2$	Erro (%)
Sul dos EUA 496 polígonos γ=0,0259	40000	36,19	37,89	4,7%
	10000	12,62	12,71	0,7%
	2500	5,31	5,15	3,0%
	625	2,79	2,63	5,7%
	169	1,78	1,72	3,4%
			Erro Médio	**3,5%**
Europa-B 249 polígonos γ=0,0621	40000	175,36	178,5	1,8%
	10000	50,25	51,11	1,7%
	2500	16,65	16,35	1,8%
	625	6,47	6,2	4,2%
	169	3,17	3,05	3,8%
			Erro Médio	**2,7%**
América do Sul 228 polígonos γ=0,0326	40000	54,68	55,98	2,4%
	10000	17,7	17,85	0,8%
	2500	6,85	6,74	1,6%
	625	3,29	3,17	3,6%
	169	1,93	1,93	0,0%
			Erro Médio	**1,7%**
Brasil 5081 polígonos γ=0,008177	40000	6,82	6,9	1,2%
	10000	3,28	3,27	0,3%
	2500	1,98	1,96	1,0%
	625	1,45	1,43	1,4%
	169	1,22	1,21	0,8%
			Erro Médio	**0,9%**
Brasil-B 23007 polígonos γ=0,006872	40000	5,6	5,61	0,2%
	10000	2,83	2,82	0,4%
	2500	1,8	1,79	0,6%
	625	1,37	1,36	0,7%
	169	1,18	1,17	0,8%
			Erro Médio	**0,5%**

Finalmente, cabe dizer que embora seja trabalhoso obter um valor preciso de γ, pois seria necessário percorrer todos os MBRs do conjunto de dados e calcular a área média, é possível obter por amostragem uma estimativa razoável para γ com um número reduzido de MBRs escolhidos ao acaso. Essa é uma alternativa fácil de ser implementada e que permitirá uma estimativa razoável do inchaço dos dados.

4.1.2.2 Tamanho de Célula Variável

Destes resultados segue que devemos escolher o tamanho da lateral da célula de forma que contenha poucos objetos sem contudo inflar muito os dados. Ora, então o tamanho ideal de célula dependerá de dois parâmetros: o tamanho dos polígonos e sua distribuição pelo espaço. Como em dados reais ambos os parâmetros variam bastante dentro do mesmo conjunto de dados, o tamanho das células deveria ser variável. É

importante notar porém que se houver uma grande concentração de polígonos na mesma área, devemos esperar uma grande quantidade de MBRs se interceptando, o que inevitavelmente resultará em um grande número de testes. Por isso, escolheremos o tamanho da célula dando maior importância ao tamanho dos polígonos do que à sua distribuição espacial.

Por outro lado, se o tamanho das células for variável, ele será definido por uma função $f(x,y)$ onde x e y são as coordenadas base de cada célula. O problema com esta abordagem é que a função f deveria ser específica para cada conjunto de dados. No entanto, se assim procedermos não poderemos realizar a comparação entre os conjuntos interno e externo da junção de forma simples, pois não existirão partições correspondentes. Na realidade, estaremos relaxando também a restrição de "casamentos simples". Portanto, f deve ser escolhida levando-se em conta cada par de conjuntos interno e externo, o que aumenta consideravelmente a sua complexidade - note que, neste caso, ela não pode ser precalculada e armazenada. Em MING (1996), abordagens utilizando partições com células de tamanho variável são propostas. A função f, que determina a partição, é definida através de R-Trees (GUTTMAN, 1984, BECKMANN et al., 1990). Já em PATEL (1996), tamanhos de células fixos são utilizados – bastante semelhante ao particionamento aqui proposto.

Nossa proposta é utilizar uma função de partição que dependa exclusivamente do tamanho médio dos polígonos de determinado conjunto, sendo auto-adaptável para regiões em que o tamanho seja muito distante da média ou haja grande concentração de polígonos. Para isso, ao invés de utilizarmos um tamanho de célula variável, utilizaremos três tamanhos de célula fixos, predeterminados para o conjunto de dados a partir do tamanho médio de seus polígonos. Em ordem crescente de tamanho, classificaremos as células em tipo 0, tipo 1 e tipo 2 (Figura 4-6). Além disso, os tamanhos das células devem ser múltiplos uns dos outros, e cada célula deve ter lado 2^n para algum $n > 0$, e coordenadas $(x.2^n, y.2^n)$. Assim, sendo os tamanhos das células tipo 0, tipo 1 e tipo 2 respectivamente t_0, t_1 e t_2, teremos $t_2 = 2^w.t_1$ e $t_1 = 2^v.t_0$, conforme a figura seguinte.

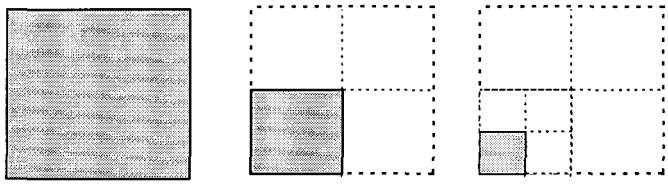


Figura 4-6 - Tipos de células

Apenas as células tipo 2 serão mapeadas para a matriz de *hash* baseadas na sua própria coordenada base. As células tipo 1 e tipo 0 utilizarão a mesma coordenada base da célula tipo 2 que as contém, e consequentemente serão mapeadas para a mesma partição e para o mesmo *bucket*. Dessa forma, um polígono que intercepte uma célula tipo 2, algumas células tipo 1 e várias células tipo 0 não será replicado no *bucket* correspondente: haverá apenas uma entrada referente ao mesmo. Com isso é possível utilizar uma grade de células pequenas (grade fina) sem causar uma explosão de dados desnecessários, replicados inutilmente.

Em um *bucket*, se houver mais de uma célula tipo 2, elas estarão ordenadas segundo a coordenada base e organizadas hierarquicamente segundo o tipo, conforme a Figura 4-7. Dessa forma, durante a comparação entre *buckets* correspondentes, apenas os MBRs contidos nas células correspondentes serão testadas.

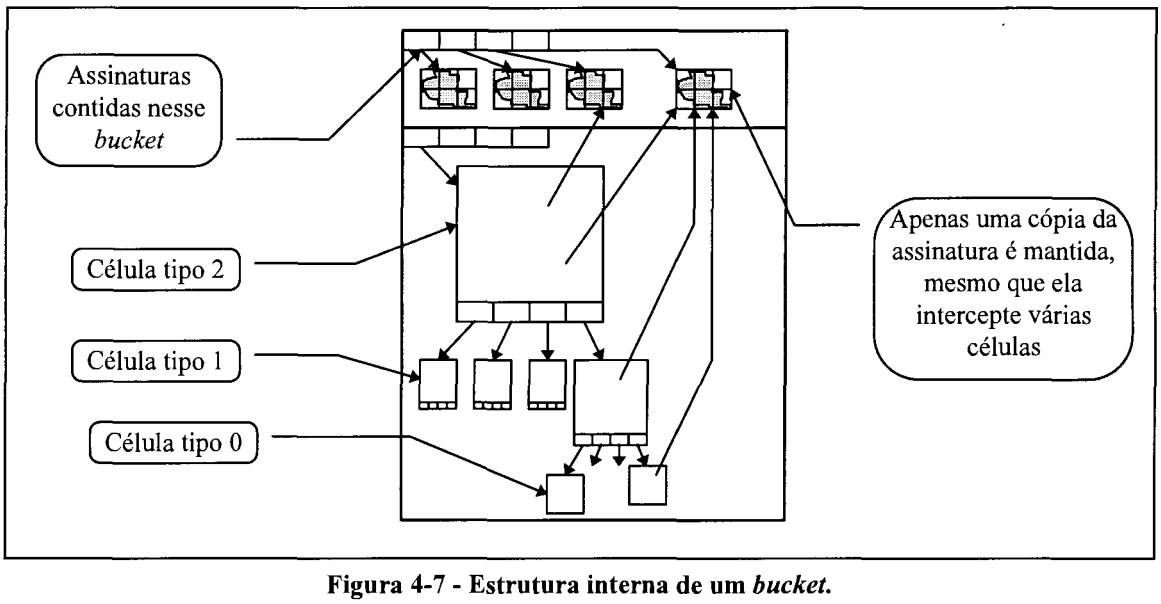


Figura 4-7 - Estrutura interna de um *bucket*.

O algoritmo escolherá a opção de comparação entre células baseado na distribuição de polígonos pelas células tipo 0, 1 e 2, preferindo sempre o caso ótimo. A Tabela 4-5 ilustra essas possibilidades, relativas à Figura 4-8. Convém notar que se uma célula contém 4 polígonos, seriam necessárias 4×3 comparações para se descobrir o

número de MBRs que se interceptam, já que se trata da junção de um conjunto com ele mesmo.

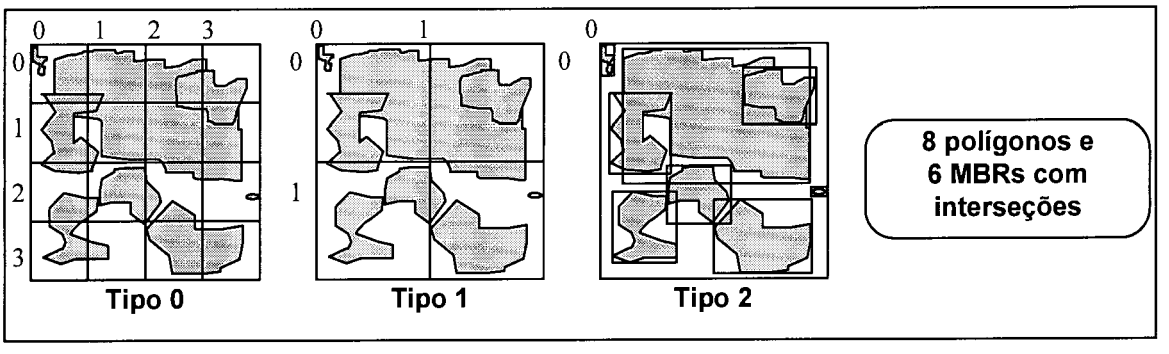


Figura 4-8 - Ocupação das células segundo o tipo.

Tabela 4-5 - Possibilidades de comparação para descobrir o no de MBRs que se interceptam.

Tipo 0	Célula (x,y)	0,0	1,0	0,1	1,1	2,0	3,0	2,1	3,1	0,2	1,2	0,3	1,3	2,2	3,2	2,3	3,3	Total
	#Polígonos	1	1	2	2	1	1	3	2	2	2	3	3	2	2	2	2	
	#Comparações	0	0	1	1	0	0	3	1	1	1	3	3	1	1	1	1	!Erro de sintaxe ,)
	Total Parcial	2				4				8				4				!Erro de sintaxe ,)
Tipo 1	Célula (x,y)	0,0				1,0				0,1				1,1				
	#Polígonos	3				4				3				2				
	#Comparações	3				6				3				1				!Erro de sintaxe ,)
Tipo 2	Célula (x,y)	0,0																
	#Polígonos	8																
	#Comparações	28																28
Misto	Célula (x,y)	0,0				0,1				1,0				1,1				
	Total Ótimo	2 (Tipo 0)				4 (Tipo 0)				3 (Tipo 1)				1 (Tipo 1)				10

É possível realizar comparações mistas ao nível de células tipo 0, 1 ou 2. Sabendo-se quantos polígonos há em cada tipo de célula teremos a opção de escolher a combinação que resulte no menor número possível de comparações. Assim, ao invés de realizarmos 3 comparações para a célula tipo 1 (0,0), realizamos apenas duas comparações nas células tipo 0 correspondentes: (0,0), (1,0), (0,1) e (1,1). Com isso, ao invés das 28 comparações necessárias na célula tipo 2 realizaremos apenas 10. Há casos em que o contrário ocorrerá, ou seja, será mais vantajoso realizar a comparação utilizando a célula tipo 2 - isto dependerá justamente do tamanho dos polígonos e de sua distribuição. Em PATEL (1996), algoritmos de *plane-sweep* são utilizados no que seria o equivalente à nossa célula tipo 2. Como tal algoritmo depende do número de

polígonos, nada impede que seja utilizado após a definição da estratégia de comparações a ser realizada, resultando em um menor número de comparações de MBRs do que o obtido naquele trabalho. Contudo, no momento o algoritmo não está sendo utilizado em nossa abordagem.

4.1.3 Eliminação de Duplicatas

Se a área de interseção de dois MBRs interceptar mais de uma célula, isso significa que o par de MBRs será avaliado e incluído na resposta várias vezes. Esse fato não é desejável, pois acarretaria em trabalho redundante nos passos seguintes do processador de consultas, além de aumentar sem necessidade o tamanho do conjunto resposta, podendo inclusive invalidar as consultas que pedem a cardinalidade do conjunto resposta.

Em GAEDE (1995) vemos uma solução simples para esse problema: antes de incluir um par de MBRs no conjunto resposta, uma consulta é feita a uma tabela de *hash* que conterá todos os pares já enumerados. Claramente, essa abordagem introduz um novo complicador: manter uma tabela de *hash*, com o tamanho proporcional ao conjunto resposta (inicialmente, esse tamanho é desconhecido). Além disso, certamente essa tabela auxiliar não residirá na memória principal, o que resultará em um grande aumento do número de acessos a disco.

Já em PATEL (1996), tal trabalho é postergado para os passos seguintes do processador de junções espaciais - às custas, porém, de uma ordenação de todo o conjunto de saída. Cabem aqui as mesmas considerações sobre memória e disco já feitas no parágrafo anterior, com o agravante de que o melhor algoritmo de ordenação possui complexidade $O(n.\log n)$.

A nossa abordagem, contudo, facilita que se encontre e elimine da resposta os pares de MBRs duplicados: como o espaço de busca está dividido segundo um *grid* uniforme, apenas as células vizinhas àquela que está sendo processada poderão conter pares de MBRs duplicados. Aproveitaremos aqui a proximidade espacial mantida pela matriz de *hash*. Manteremos várias tabelas de *hash*, uma para cada *bucket*, contendo somente os pares de MBRs que não estão inteiramente contidos na célula (e não o conjunto resposta inteiro), e iremos eliminando aquelas tabelas cujas células vizinhas já foram todas processadas. Essa alternativa foi testada com sucesso, e em nossos

conjuntos de dados as tabelas de *hash* geradas foram pequenas o suficiente para permanecerem na memória, mesmo no caso do maior conjunto de dados.

Finalmente, se a junção estiver sendo realizada sem a utilização de aproximações, ou seja, apenas sobre os MBRs, pode-se eliminar os pares replicados através de uma estratégia bem simples: processar cada par de MBRs apenas na célula que contém o canto inferior direito (ou qualquer outro previamente escolhido) da interseção dos MBRs. Assim, cada par de MBRs que se interceptar somente será processado em uma única célula, como podemos ver na Figura 4-9. Convém observar que essa mesma estratégia também pode ser utilizada se as aproximações forem replicadas inteiramente em cada entrada repetida de um polígono. No entanto, se apenas parte da aproximação estiver representada em cada entrada não será possível deixar o processamento do par de MBRs para a célula que contém o canto inferior direito pois esta célula não irá conter o restante da aproximação e portanto não poderá detectar se houver interseção em outra célula.

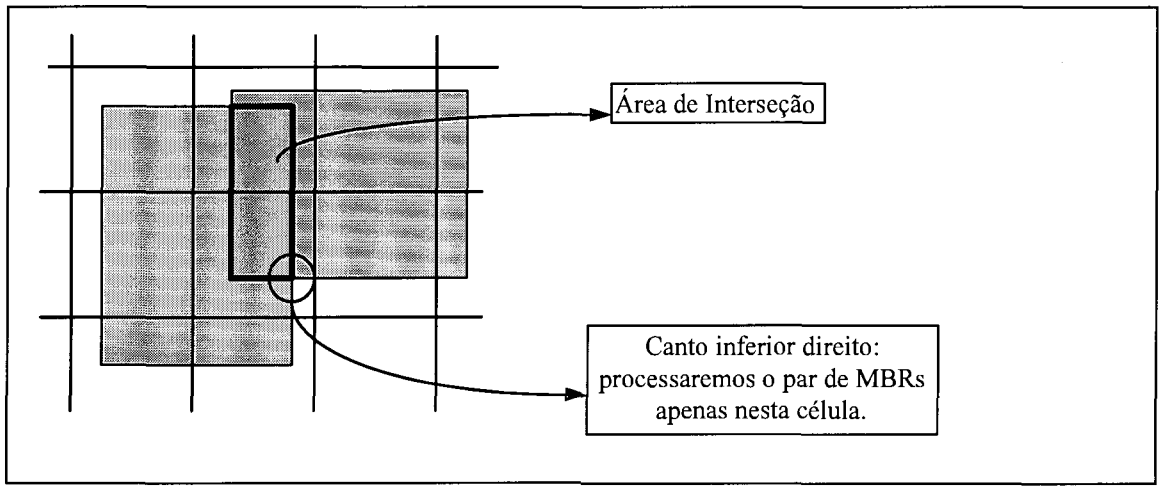


Figura 4-9 - Estratégia para eliminar pares de MBRs duplicados

4.1.4 Tratamento do *Overflow*

Embora o tamanho dos *buckets* possa ser dimensionado de forma a não haver *overflow* em um *bucket*, em geral essa abordagem não é satisfatória por desperdiçar muito espaço, aumentando sobremaneira o acesso ao disco. Portanto, certamente haverá *overflows* em alguns *buckets*. Nossa abordagem trata o *overflow* de uma forma hierárquica, semelhante a uma árvore: cada grupo predeterminado de *buckets* terá um *bucket* extra para *overflow*. Por sua vez, cada grupo de *buckets* de *overflow* terá também um *bucket* extra para *overflow*. A vantagem de tal abordagem é que, na fase de junção,

como os *buckets* são percorridos em uma ordem predeterminada (e que deve ser a mesma utilizada para a criação dos *buckets* de *overflow*), o *bucket* de *overflow* do grupo deve ser mantido na memória. Dessa forma, é possível evitar acessos extras ao disco quando formos processar os *buckets* restantes do grupo. Novamente, podemos fazer uso da proximidade espacial mantida pela matriz de *hash* para determinar a ordem de leitura dos *buckets*.

Em nossos testes, utilizando-se um *bucket* de *overflow* para cada grupo de 16 *buckets*, obtivemos uma hierarquia de apenas dois níveis (os outros níveis não chegaram a ser utilizados). É importante observar que o *bucket* de *overflow* apenas agrupa as células que não couberam no *bucket* original em um bloco para a transferência do disco para a memória. As células em si continuam separadas na estrutura interna do *bucket*, não sendo pois necessário comparar cada um das células do *bucket* de *overflow*, mas apenas as células que originalmente vieram do *bucket* sendo processado.

4.1.5 Testes Realizados

Nossos dados de testes serão sempre dois conjuntos de polígonos, contendo cada um de 200 até 23.000 polígonos. Além disso, conjuntos derivados dos conjuntos apresentados na Tabela 4-1 foram gerados conforme já mencionado. Assim, trabalharemos com dois tipos básicos de junções: o conjunto contra ele mesmo e o conjunto contra ele mesmo porém deslocado aleatoriamente de (x,y). Dos conjuntos de dados da Tabela 4-1, apenas Brasil-B possui polígonos sobrepostos. No entanto, todos possuem MBRs que se interceptam. Realizamos uma série de testes para obter três tipos de resultados: o número de comparações, que afeta o desempenho em termos de tempo de CPU, o número de acessos a disco que corresponde a operações de E/S e o tempo total para realizar uma consulta, que é a soma dos outros dois.

A Tabela 4-6 ilustra os resultados obtidos para o conjunto de dados "Brasil", que possui 5.081 polígonos e 18.935 pares de MBRs com interseção. Já a Tabela 4-7 ilustra os resultados obtidos para o conjunto de dados "Brasil-B", que possui 23.007 polígonos e 130.140 pares de MBRs com interseção. Repare que a coluna **Tipo 2**, número de testes utilizando apenas células do tipo 2, reflete o desempenho do algoritmo apresentado em PATEL (1996). Finalmente, vale dizer que o número de comparações obtidos por nossa abordagem, que é representado na coluna ***Otimizado***, se manteve razoavelmente estável para uma faixa grande de valores para o *grid*, o que é muito

vantajoso por não dispormos de uma metodologia mais precisa sobre como escolher o tamanho dos *grids*. De fato, embora possamos estimar o inchaço dos dados e por conseguinte o número máximo estimado de comparações, esse é um valor que reflete apenas a junção do conjunto com ele mesmo. Não temos como estimar precisamente o número máximo de comparações para conjuntos distintos sem levar em conta a distribuição espacial dos objetos, o que não foi feito em nossos modelos. Um trabalho que aponta nessa direção, aparentemente com bons resultados para as R*-Trees, é GAEDE (1996). A coluna ***Razão sobre a Resposta*** é o número de testes sobre o tamanho do conjunto resposta. Note que quanto menor o valor nesta coluna melhor, pois menos comparações desnecessárias são realizadas.

Tabela 4-6 - Conjunto Brasil: no de comparações de MBRs e a razão sobre o tamanho do conjunto resposta (quanto menor, melhor) para vários grids.

Grid	Número de Comparações			Razão sobre a resposta	Escolha das Células		
	Estimado	Tipo 2	Otimizado		Tipo 2	Tipo 1	Tipo 0
200x200	30.817	55.501	48.750	2,57x	51.60%	29.51%	18.88%
100x100	34.679	71.588	48.257	2,55x	49.11%	31.75%	19.14%
50x50	54.253	130.534	51.204	2,70x	38.48%	29.60%	31.91%
25x25	108.525	293.322	70.731	3,74x	25.74%	25.08%	49.17%

Tabela 4-7 - Conjunto Brasil-B, idem ao anterior.

Grid	Número de Comparações			Razão sobre a resposta	Escolha das Células		
	Estimado	Tipo 2	Otimizado		Tipo 2	Tipo 1	Tipo 0
300x300	194.155	365.451	346.113	2,66x	49.85%	27.75%	22.40%
150x150	190.382	375.074	312.133	2,40x	37.67%	36.45%	25.88%
75x75	286.246	606.312	323.367	2,70x	21.14%	33.98%	44.88%
38x38	606.079	1.406.822	378.066	2,91x	18.63%	15.48%	65.89%

A Tabela 4-8 a seguir mostra os custos das operações de entrada e saída para a junção de alguns conjuntos. É importante observar que, no conjunto Europa-B, gerado com distribuição espacial aleatória, não existe nenhum tipo de ordenação. Ou seja, dados sequenciais no disco estão completamente dispersos no espaço. Nos outros conjuntos, existe um agrupamento simples, que pode ser por região (América do Sul) como por latitude (Brasil). De uma forma geral, os dados reais possuem uma certa localidade. Note que o objetivo é minimizar o número de vezes em que uma página é lida – o ideal é que cada página seja lida no máximo uma vez.

Tabela 4-8 - Custos de entrada e saída em blocos de 0.5 Kbyte (Obs: a última coluna mostra blocos de 1Kbyte).

Conjunto de dados	América do Sul	Europa-B	Brasil	Brasil	Brasil
Grid	25x25	25x25	200x200	200x200	100x100
tamanho do buffer (em relação ao tamanho dos dados)	9.30%	6,81%	8.22%	2.05%	1.98%
# acessos por página	1,66	4,62	1,04	1,41	2,19
# acessos a disco - 0.5kb	287	1.086	13.022	17.628	7.073 (1kb)
% acertos do buffer	61,73%	32,55%	62,41%	49.11%	57.53%

Para a avaliação de desempenho, nós usamos uma máquina Sun Sparc Ultra 1 de 64Mb, com um disco local e isolada do resto da rede. A Figura 4-10 mostra o tempo gasto na construção do índice (A) e o custo em termos de espaço (B), que foi calculado dividindo-se o tamanho total gasto na estrutura de dados pelo tamanho total das aproximações do conjunto de dados, ou seja, as chaves. Note que o *overhead* provocado pela R*-Tree é aproximadamente constante e provém do fato de que a utilização média dos nós fica abaixo da sua capacidade máxima. O conjunto **Brasil'** é um conjunto artificialmente gerado a partir do conjunto **Brasil** conforme mencionado anteriormente.

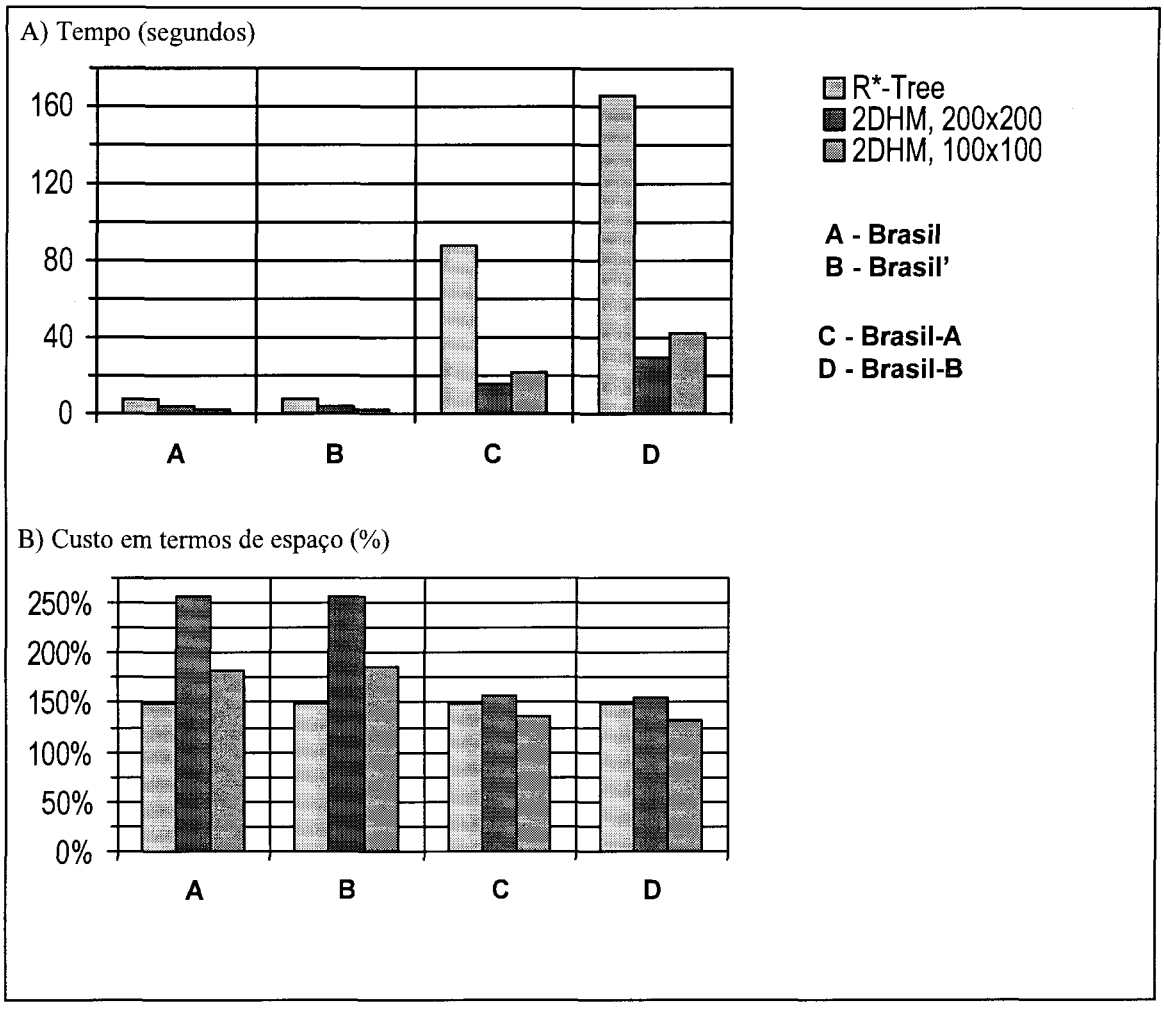


Figura 4-10 A e B - Tempo gasto na construção dos índices para alguns conjuntos e custo em termos de espaço.

A seguir, mostramos os resultados encontrados quando realizamos apenas a junção de MBRs entre os conjuntos A e B, denotada por A/B, e C e D, denotada por C/D. Para mostrar a estabilidade do método, realizamos os testes com buffers de tamanhos bem diferentes : 37, 67 e 197 páginas de *buffer*. Por outro lado, o desemprenho da junção utilizando a R-Tree é melhorado quando utilizamos mais e mais buffers - embora de forma não linear. A Figura 4-11-A mostra o número de comparações, e a Figura 4-11-B mostra o número de operações de E/S.

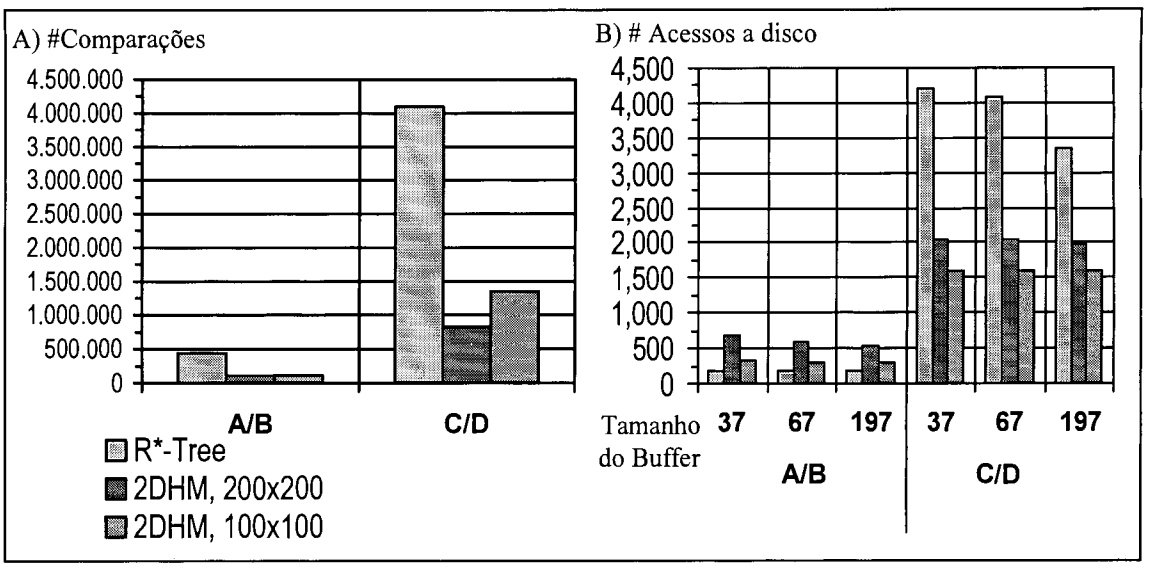


Figura 4-11 A e B - Número de comparações e acessos a disco.

Finalmente, as Figura 4-12 A e B mostram o tempo total de execução medido para as junções de MBRs de polígonos. Adotamos a mesma metodologia usada em BRINKHOFF et al. (1993a) e BRINKHOFF et al. (1994) para medir os tempos E/S, e que consiste em basicamente medir o número de acessos a disco feitos pelo programa e não pelo sistema operacional, subtrair do tempo total de execução o tempo gasto pelo s. o. em acessos a disco e somar o tempo médio esperado para o número de acessos medidos pelo programa. Além disso, todas as técnicas de otimização de travessias de R-Trees propostas em BRINKHOFF et al. (1993a) foram implementadas.

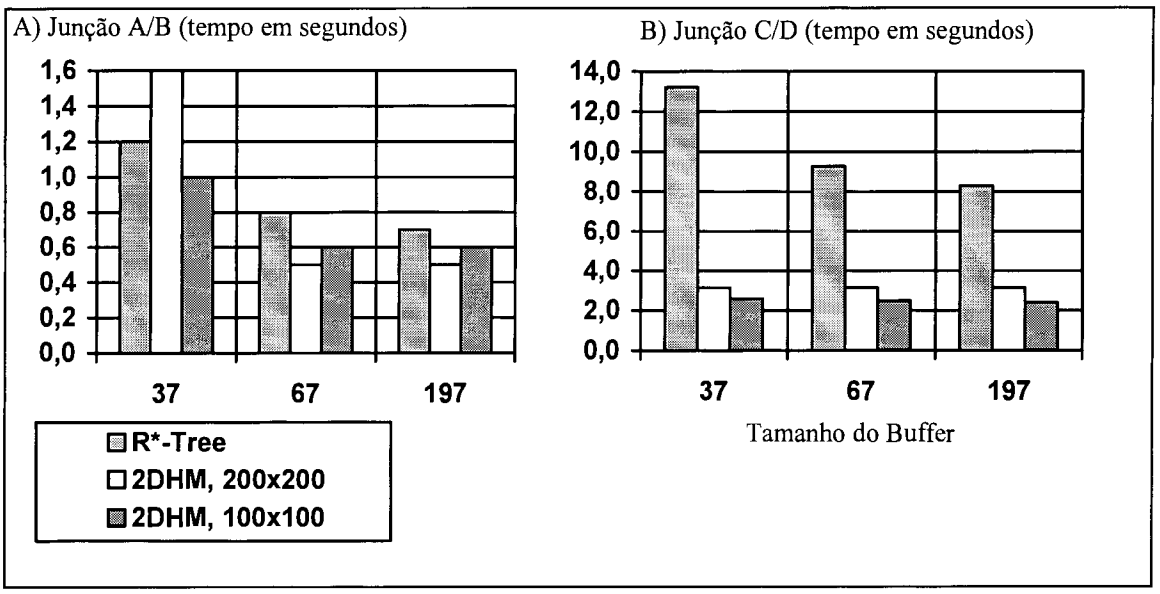


Figura 4-12 A e B - Avaliação de Desempenho das junções A/B (A) e C/D (B)

4.1.6 Comentários

Podemos analisar o número de acessos a disco em dois momentos distintos: durante a fase de partição e durante a fase de junção. Durante a fase de partição, o

número de acessos a disco é muito influenciado pela ordem em que os dados dos conjuntos estão dispostos. Se esses dados estão distribuídos aleatoriamente pelo espaço de busca, teremos em consequência um grande número de acessos a disco, pouco adiantando o tamanho do *buffer* utilizado. Porém, conforme apontado em BRINKHOFF et al. (1993a), outras abordagens também sofrerão com tal situação. Já na fase de junção, como os *buckets* são percorridos em uma ordem predeterminada, cada *bucket* é lido apenas uma vez - inclusive os *buckets* de *overflow*, conforme visto anteriormente. Cabe observar também que, na prática, em geral teremos várias junções encadeadas, de forma que a primeira a ser realizada terá como consequência uma ordenação nos dados, se porventura os mesmos estivessem distribuídos aleatoriamente. A partir daí, os conjuntos que utilizarem tanto o conjunto resposta da primeira junção como qualquer um dos conjuntos envolvidos poderão tirar proveito dessa ordenação.

Embora o acesso ao disco seja a operação que consome mais tempo nesta fase do processamento de junções espaciais, o tempo utilizado para a comparação e identificação de pares de MBRs que se interceptam não é desprezível. No entanto, poucos artigos sobre o assunto mencionam o número de comparações necessárias entre MBRs. Uma exceção é o trabalho de BRINKHOFF et al. (1993a), que mostra diversos resultados nesse sentido, utilizando R*-trees. Apenas a título de ilustração, o melhor desempenho dos algoritmos apresentados lá necessitou realizar um número de comparações de MBRs que ficou em torno de dez vezes o tamanho do conjunto resposta, havendo casos em que esse fator chegou a 25. Nossa abordagem conseguiu taxas que ficaram em torno de duas a três vezes o tamanho do conjunto resposta.

O resultado da fase de partição, que é a fase que realiza mais acessos a disco de forma não sequencial, pode ser armazenado e mantido como um índice para determinados conjuntos de dados (PATEL, 1996). Conforme observamos, o aumento da massa de dados é menor do que o inchaço dos MBRs, resultando em uma estrutura de dados viável de ser armazenada na base de dados. Assim, tendo índices armazenados na forma aqui descrita para os conjuntos de dados, o processamento de junções espaciais poderá ser realizado diretamente a partir da fase de junção - que faz acessos tipicamente sequenciais.

Finalmente, uma observação importante: se combinarmos a abordagem 2DHM com a abordagem 4CRS, que apresentamos na seção 4.2 a seguir, o inchaço nos dados não provocará um aumento correspondente no espaço utilizado, pois a abordagem 4CRS

permite que apenas parte da chave seja armazenada em uma página. Podemos dividir uma chave 4CRS em duas partes: a parte que contém as informações genéricas e a parte que contém a aproximação em si. Será necessário replicar apenas a parte genérica da chave, sendo a parte que contém a aproximação dividida entre as várias células. Além disso, a compactação será pouco afetada por poder ser realizada de forma independente nas partes da aproximação, conforme pode ser observado em ZIMBRÃO (1997), ZIMBRÃO (1998) e na seção 4.2. Tipicamente, uma aproximação 4CRS de 750 pontos ocupa 40 bytes. Incluindo o MBR e Identificador do polígono, teremos uma chave de 60 bytes. Portanto, apenas um terço da chave será duplicada para cada polígono que interceptar mais de uma célula, o que significa dizer que um inchaço de 45% na realidade terá um efeito multiplicador de apenas 15% sobre os dados.

4.2 Assinatura Raster de Quatro Cores

Nesta seção apresentaremos a aproximação 4CRS e algoritmos para utilizá-la para detectar interseções de polígonos. Além disso, iremos mostrar como computar a aproximação para um polígono qualquer e como compactar a aproximação de forma a economizar espaço em disco. Finalmente, iremos avaliar os ganhos de desempenho obtidos com o uso da mesma.

4.2.1 Conceitos Iniciais

Como vimos, uma das conclusões obtidas através do MSQP é que o processamento de junções espaciais pode ser realizado de forma mais eficiente se utilizarmos filtros, que permitam uma avaliação dos pares de candidatos, classificando-os sem a necessidade de recuperá-los do disco. Os filtros são implementados através do uso de aproximações, em geral uma aproximação conservadora e outra progressiva. Através do filtro, é possível classificar um par de candidatos (MBRs que se interceptam) em grupos: aceitos, rejeitados e inconclusivos. Quando a consulta deve retornar apenas o conjunto de identificadores de polígonos, o teste de interseção exata somente se fará necessário para os pares inconclusivos. É o caso por exemplo de consultas com várias junções espaciais aninhadas. Por outro lado, mesmo que seja necessário retornar a interseção dos polígonos explicitamente calculada, o cálculo somente precisará ser realizado em pares de polígonos que realmente se interceptam. É o caso por exemplo da sobreposição de duas camadas temáticas para a produção de uma terceira.

4.2.2 A Aproximação Raster

Apresentaremos agora um novo tipo de aproximação: a aproximação raster. Esta aproximação é generalizadora, porém combinando em uma única aproximação propriedades das aproximações conservadoras e progressivas. Desta forma, ela pode ser utilizada para a construção de filtros geométricos para identificar a interseção de polígonos.

A aproximação raster de um polígono é um pequeno mapa de *bits* do polígono e que utiliza algumas poucas cores. Esta aproximação pode ser vista como uma assinatura de *bits* do polígono. Como qualquer outro tipo de assinatura, pode ser computada uma vez, armazenada e utilizada posteriormente. Como são relativamente pequenas, as assinaturas dos polígonos podem ser armazenadas nos índices espaciais, o que habilita a sua utilização na avaliação de consultas espaciais.

4.2.2.1 Assinatura Raster de Polígonos

Dois dos resultados obtidos por BRINKHOFF et al. (1994) são particularmente importantes para o nosso trabalho. O primeiro resultado é que o passo de comparação geométrica exata, por exigir a busca de muitos objetos no disco, é o que mais consome tempo. Dessa forma, esforços devem ser dispendidos em aperfeiçoar os filtros para que um número menor de polígonos tenha de ser trazido para a memória para comparação. Segundo, cerca de dois terços dos candidatos obtidos no passo um são de pares de polígonos que realmente se interceptam. Embora esta proporção possa variar um pouco de conjunto para conjunto, essa é uma proporção relacionada ao uso de MBRs para a realização da junção sobre as estruturas de dados espaciais. Desta forma, é interessante aperfeiçoar os filtros para uma maior detecção de interseções logo no segundo passo do processador de consultas espaciais.

Assim, propomos uma aproximação para substituir, no esquema do processador proposto, a aproximação **5C** (conservadora) e **EL&EC** (progressiva) , ambas utilizadas em BRINKHOFF et al. (1994): a aproximação raster de quatro cores (**4CRS**, de *4 colors raster signature*). Essa abordagem consiste em manter para cada polígono uma aproximação raster contendo $m \times n$ células, cada uma das quais com dois bits de informação indicando uma das seguintes possibilidades (Tabela 4-9):

Tabela 4-9 - Tipos de células em uma assinatura

Valor dos *bits*	Tipo da Célula	Descrição
00	Vazia	A célula não intercepta o polígono
01	*Pouco*	A célula contém uma interseção de 50% ou menos com o polígono
10	*Muito*	A célula contém uma interseção de mais de 50% com o polígono
11	*Cheia*	A célula está totalmente ocupada pelo polígono

Como exemplo, temos o polígono da Figura 4-13.

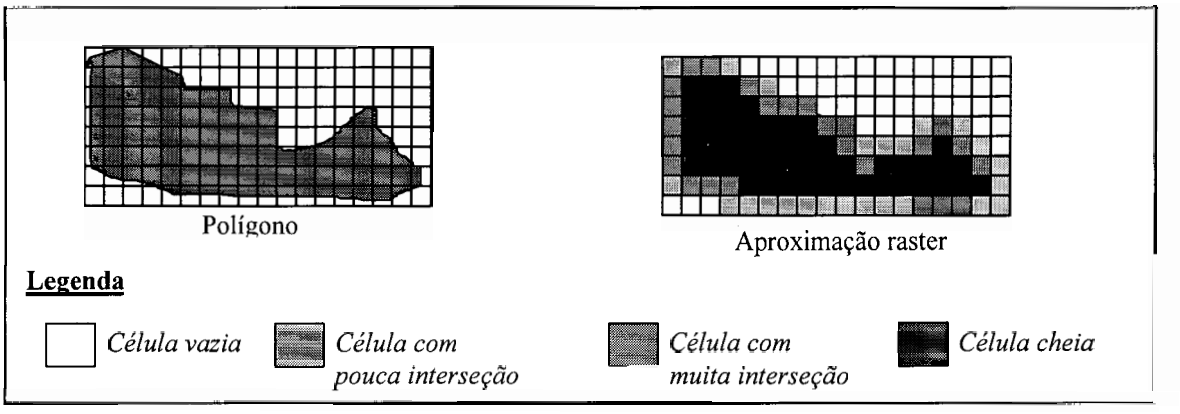


Figura 4-13 - Células em uma aproximação 4CRS

Para realizar a classificação de um par de polígonos candidatos devemos sobrepor as suas aproximações raster na área onde seus MBRs se interceptaram. Se for necessário, veremos adiante como realizar qualquer mudança de escala. Analisando cada par de células sobrepostas, teremos as possibilidades ilustradas na Tabela 4-10.

Tabela 4-10 -Resultado das comparações de células: apenas três casos são inconclusivos (candidato).

Tipo de célula × Interseção	Célula Vazia	Célula com Pouca Interseção	Célula com Muita Interseção	Célula Cheia
Célula Vazia	Rejeitado	Rejeitado	Rejeitado	Rejeitado
Célula com Pouca Interseção	Rejeitado	***Candidato***	***Candidato***	*Aceito*
Célula com Muita Interseção	Rejeitado	***Candidato***	*Aceito*	*Aceito*
Célula Cheia	Rejeitado	*Aceito*	*Aceito*	*Aceito*

Se duas células sobrepostas possuem mais de 50% de área do polígono cada uma é óbvio que os polígonos se interceptam. É importante observar que basta que um par de células sobrepostas seja aceito para que o par de polígonos em questão também o seja, como pode ser observado na Figura 4-14.

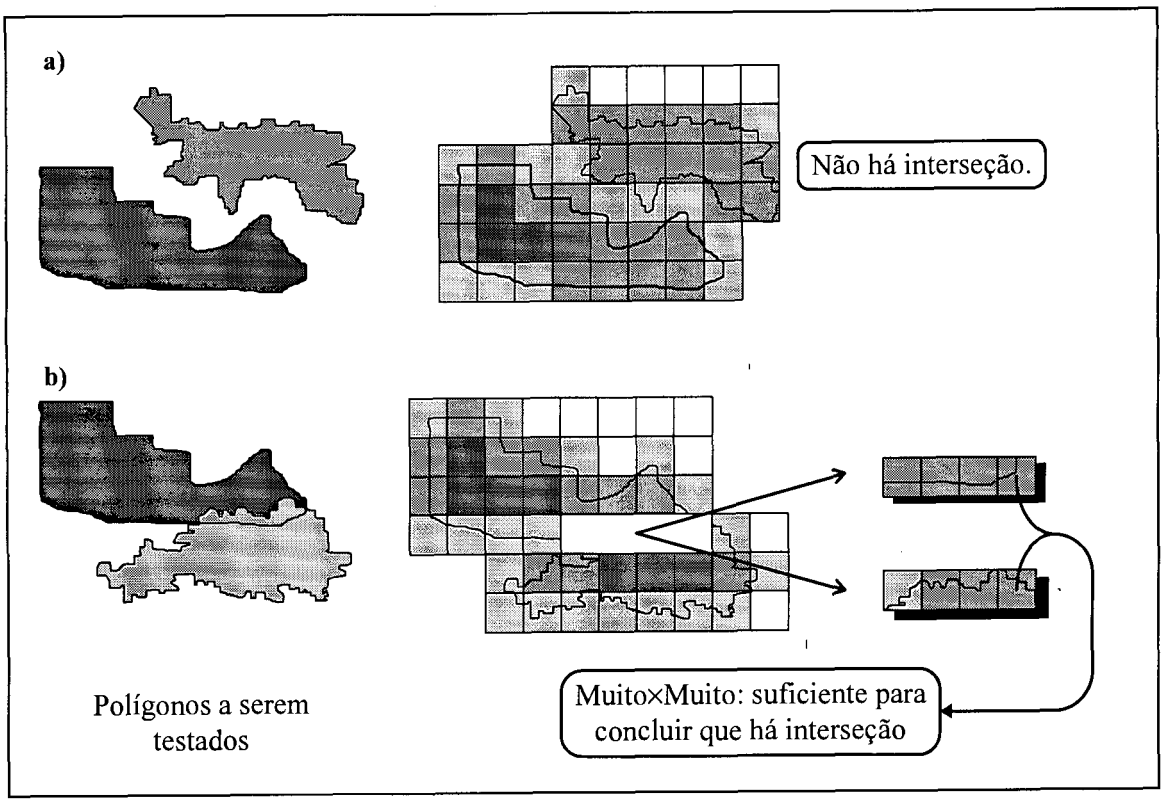


Figura 4-14 - (a) polígonos sem interseção, (b) polígonos com interseção

4.2.2.2 Mudança de Escala

Para podermos comparar duas aproximações de polígonos distintos devemos antes assegurar que as suas células possuem o mesmo tamanho e que as células que se interceptam possuem as mesmas coordenadas. Para lidar com as mudanças de escala mais facilmente, estabelecemos que cada célula tem lado com comprimento igual a uma potência de dois (2^n), e o início de cada célula é múltiplo da mesma potência de dois ($2^n a$). Assim, garantimos que se duas células de mesmo tamanho se interceptam, então elas se sobrepõem perfeitamente (Figura 4-15).

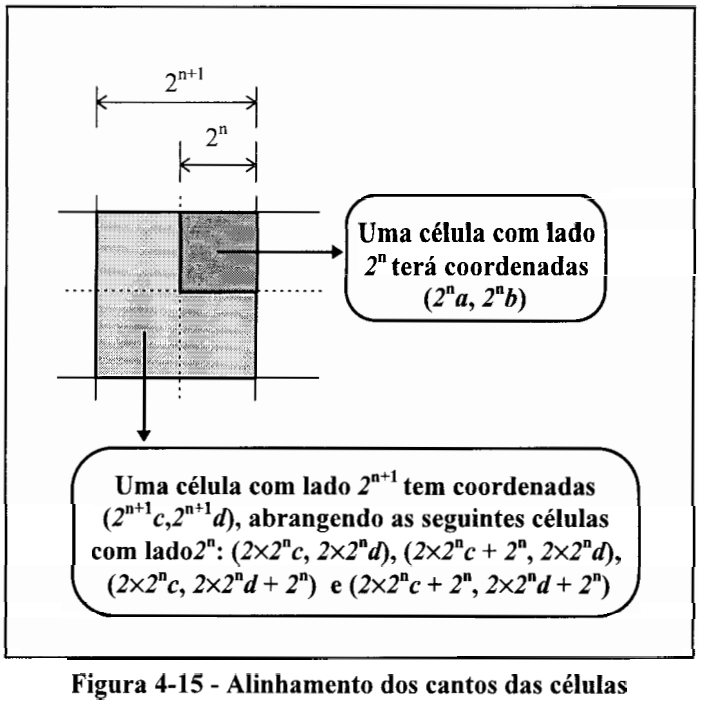


Figura 4-15 - Alinhamento dos cantos das células

Note que não é possível subdividir uma célula, ou seja, diminuir o seu lado, em uma alteração de escala, visto que nos casos de células do tipo *Pouco* ou *Muito* estaríamos pressupondo levianamente que a área do polígono está uniformemente distribuída pela célula. Desta forma, sempre que se fizer necessária uma mudança de escala, esta será feita agrupando-se 2^m células, respeitando-se o fato de que o valor das coordenadas do início de cada célula será múltiplo de seu lado. A Figura 4-16 ilustra este processo.

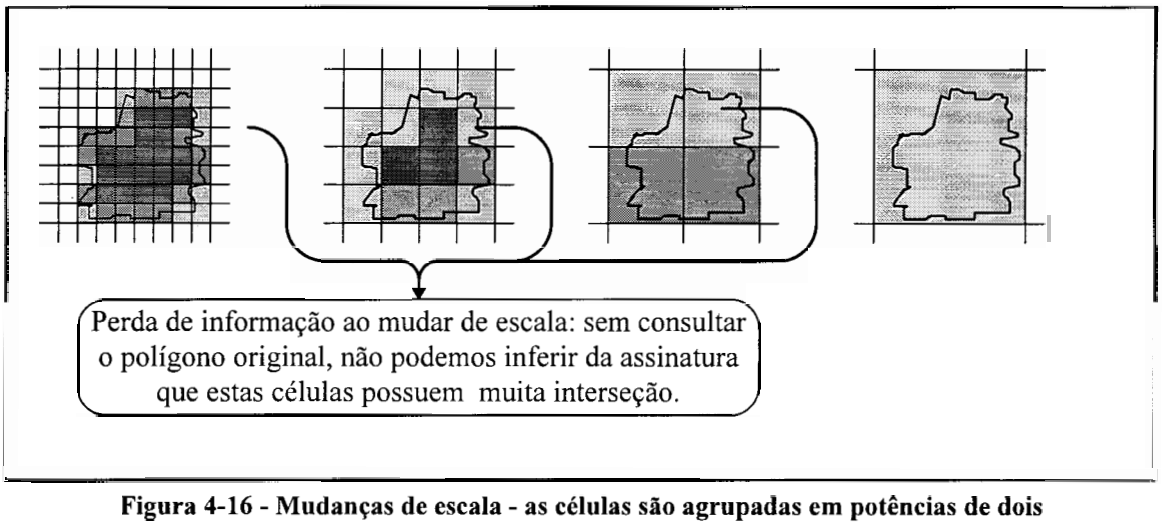


Figura 4-16 - Mudanças de escala - as células são agrupadas em potências de dois

Para realizar a mudança de escala devemos utilizar uma abordagem pessimista. Note que é importante realizar a conversão diretamente da assinatura original para minimizar a perda de informação. Segue o algoritmo para a mudança de escala (entrada: um grupo $m \times m$ de células; saída: o valor da célula que contém este grupo):

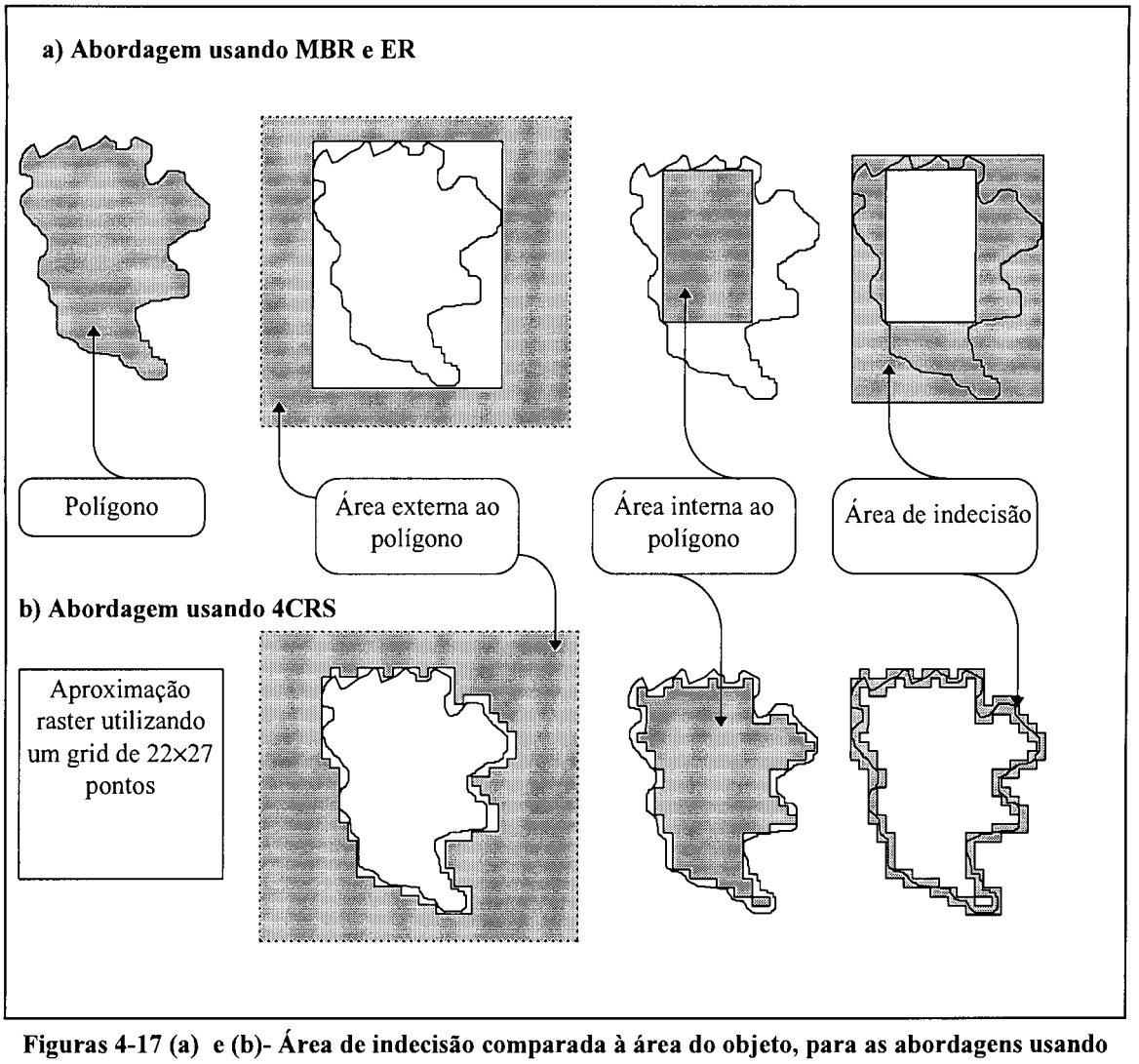
1) *se todas as células forem Vazias, o resultado é célula Vazia;*

2) *se todas forem Cheias, o resultado é célula Cheia;*

3) *caso contrário, devemos realizar um somatório onde cada célula Vazia ou Pouca contará como valendo 0, Muita contará como valendo 0.5 e Cheia contará como valendo 1; se a média for menor do que 0.5 o resultado é célula com Pouca interseção, se não o resultado é célula Muita interseção.*

4.2.2.3 Área de Indecisão

Como vimos, a aproximação raster combina em uma única aproximação as informações características da aproximações conservadoras e progressiva. Quando utilizamos uma aproximação conservadora, somente podemos ter certeza da interseção, pois a área da aproximação está totalmente contida no polígono. Por outro lado, a aproximação progressiva contém todo o polígono e mais alguma área que não pertence ao mesmo (o MBR é uma aproximação progressiva). Obviamente, a aproximação conservadora está totalmente contida na aproximação progressiva. No entanto, esta abordagem dual nada nos informa a respeito da área contida na aproximação progressiva e que não está contida na aproximação conservadora. Chamaremos esta área de área de indecisão, pois se as aproximações de dois polígonos possuírem interseção apenas nesta área nada poderemos afirmar, postergando a decisão para o passo de comparação geométrica exata (3úmero passo do MSQP). É importante observar que nossa definição de área de indecisão está diretamente relacionada com o conceito de *Qualidade da Aproximação* definido em BRINKHOFF et al. (1993b). As Figuras 4-17 (a) e (b) mostram a área de indecisão de um polígono segundo a abordagem MBR/ER e 4CRS.



Figuras 4-17 (a) e (b)- Área de indecisão comparada à área do objeto, para as abordagens usando aproximações MBR/ER (a) e 4CRS (b)

Já a aproximação raster, além de combinar as duas aproximações em apenas uma, mantém informação sobre a quantidade de área do polígono contida na área de indecisão. Assim, conseguimos diminuir o grau de indecisão da aproximação, conforme podemos observar na Tabela 4-11 e na Figura 4-18. O grau de indecisão é menor não só por obtermos uma área de indecisão menor como também por conseguirmos em alguns casos decidir mesmo na área de indecisão (comparação de duas células do tipo *Muito*).

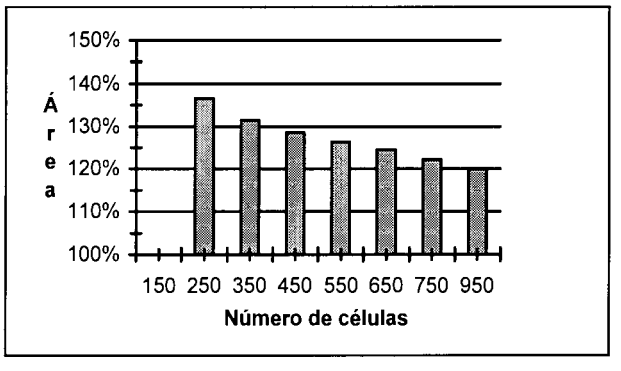


Figura 4-18 - Média da área da aproximação sobre a área do polígono (%) - média dos vários conjuntos de dados.

Tabela 4-11 - Comparação das diferentes aproximações - média dos vários conjuntos de dados

	Área da Aproximação / Área do Polígono
4CRS (750 células)	**122%**
Convex Hull	125%
5-Corner	133%
MBR	193%

Conforme pode ser observado, a Figura 4-18 apresenta uma distribuição aparentemente assintótica. Assim, aumentar o número de células em uma aproximação não resultará em uma melhoria correspondente na qualidade da aproximação. Aqui, uma escolha entre qualidade de aproximação e tamanho de chave deve ser feito. Para manter o tamanho da chave pequeno, decidimos usar aproximações com no máximo 750 células. Como veremos no item 4 da subseção 4.2.2, este limite na resolução resulta em um tamanho da aproximação 4CRS (em bytes) comparável ao tamanho das outras abordagens.

Além disso, analisando nossos conjuntos de dados, oriundos do mundo real, observamos que em uma resolução de 750 pontos as células apresentaram a seguinte distribuição aproximada: para cada célula do tipo *muito*, há uma célula do tipo *pouco*, quatro células *vazias* e quatro células *cheias*. Embora esta distribuição seja animadora, veremos adiante que ela é dependente da escala: como a comparação de polígonos cujas células possuem tamanhos diferentesexige uma mudança de escala, esta distribuição somente valerá para uma das aproximações envolvidas na comparação.

Ainda que haja uma diferença muito grande de tamanho entre dois polígonos sendo comparados, na pior das hipóteses um dos polígonos será reduzido a uma célula

com pouca área de interseção. Ainda assim, a aproximação do outro polígono não será alterada, o que significa que ele conterá cerca de 80% de células cheias ou vazias (no caso apresentado de 750 pontos), o que permitirá ainda um bom resultado na realização da comparação.

4.2.2.4 Número de Células

Experimentalmente, conforme a Figura 4-18, verificamos que a qualidade do filtro melhora à medida que aumentamos o número de células na aproximação. Intuitivamente, podemos explicar este fato da seguinte forma: quanto maior o número células, mais próxima a aproximação estará do polígono original, até que no limite estaremos comparando os próprios polígonos e não mais aproximações, o que nos daria uma taxa de acerto de 100%. Porém, este é um resultado de interesse puramente teórico. Na prática, uma aproximação com 10,000 células seria maior que o próprio polígono, o que a tornaria inútil.

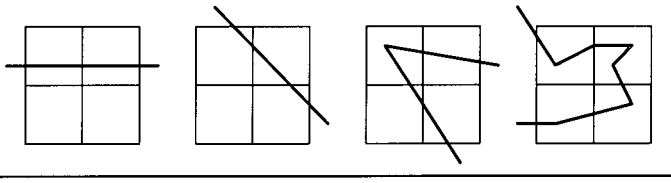
Anteriormente, dissemos que a distribuição *área* × *número* de pontos era aparentemente assintótica. Não daremos aqui uma prova desta afirmação, apenas traçaremos um roteiro para tal prova da seguinte forma, deixando a demonstração a cargo do leitor:

1) Ao dividirmos por dois o tamanho das células, teremos multiplicado por quatro o número de células *cheias* e também o número de células *vazias*, mais algumas células (*cheias* ou *vazias*) obtidas no caso seguinte;
2) Ao dividirmos uma célula com *muita* ou com *pouca* interseção com o polígono, teremos a seguinte situação: no máximo, r células terão divisão homogênea, ou seja, darão origem a quatro células com *muita* ou com *pouca* interseção com o polígono, onde r é o número de pontos do polígono, pois para isso acontecer é necessário que a célula contenha pelo menos um vértice do polígono. As demais células, que não contêm vértices do polígono, conforme podemos observar na Figura 4-19, darão origem a pelo menos uma célula *cheia* ou uma célula *vazia*, e no máximo a três células com *pouca* ou *muita* interseção.
3) Sejam P, M, V e C são respectivamente células com **p**ouca e **m**uita interseção, **v**azia e **c**heia, e *resto* é o número de células *cheias* ou *vazias*

originárias de uma célula com *muita* ou com *pouca* interseção com o polígono. A cada aumento *d* na escala, teremos como estimativas:

(a) número máximo de células com *pouca* ou *muita* interseção com o polígono: $3^d(P + M) + 4r$ - *resto*:

(b) número mínimo de células *cheias* ou vazias: $4^d(V + C)$ + *resto*.



Diversas formas de uma célula ser subdividida em quatro. Note que apenas quando existem vértices dentro da célula é possível a mesma dar origem a quatro células que interceptam arestas. De outra forma, no máximo três células irão possuir interseção com alguma aresta.

Figura 4-19 - Diferentes formas de uma célula ser dividida

À medida em que *d* cresce, os termos exponenciais dominam as funções, o que significa que a razão (a)/(b) tende a diminuir, chegando a zero quando *d* tende a infinito. Ora, (a) é justamente o número de células de indecisão: células com pouca ou muita interseção. Se a aproximação raster for constituída somente de células cheias ou vazias, a taxa de acerto será de 100%. Porém, podemos observar também que a melhoria da qualidade do filtro não é linear, refletindo a função que domina a razão (a)/(b), que é $(3/4)^n$.

Portanto, podemos desenvolver critérios para a escolha do limite *p* de células para as aproximações raster. Como vimos, esse limite influirá no tamanho médio de cada chave, e portanto uma relação de custo benefício está envolvida. Esta escolha estará relacionada com o número médio de pontos por polígono, bem como com a distância mínima entre os pontos de cada polígono. No entanto, em geral a distância mínima entre dois pontos está diretamente relacionada com a precisão, e portanto, com o número médio de pontos por polígono. A exceção fica para os polígonos que possuem grandes arestas, em geral definidas artificialmente - por exemplo o caso de vários estados e condados norte-americanos, bem como grandes construções, cidades e propriedades de uma forma geral.

Em BRINKHOFF et al. (1995) é apresentado um estudo sobre a complexidade de polígonos e em GAEDE (1996) é apresentado um estudo sobre a dimensão de fractal de polígonos que pode ser utilizada como base para estimativas do número de pontos

necessários para se obter uma boa aproximação raster. Embora os resultados preliminares sejam animadores, ainda são necessários testes mais detalhados para a obtenção de resultados conclusivos.

Como afirmamos antes, em nossa pesquisa utilizamos 750 células para manter o tamanho da aproximação competitivo. Claro que qualquer outro número perto de 750 poderia ser usado com resultados semelhantes para este conjunto de dados.

4.2.2.5 Computando a Aproximação 4CRS

Para computar a aproximação 4CRS implementamos um algoritmo simples que foi desenvolvido usando rotinas que já estavam disponíveis. Não nos preocupamos com a otimização do mesmo já que ele será usado apenas uma vez para computar as aproximações, que por sua vez serão armazenadas e usadas várias vezes depois. Os testes mostraram que o tempo gasto no cálculo das aproximações 4CRS, embora um pouco maior, é comparável ao tempo gasto no cálculo das aproximações 5-C e ER&EL.

O MBR do polígono é usado para calcular o MBR-2^n, ou seja, um MBR cujos vértices são da forma ($2^n x_0$, $2^n y_0$) e ($2^n x_k$, $2^n y_k$), onde x_0, x_k, y_0, y_k e n são inteiros. Além disso, n é escolhido de tal um modo que $(x_k - x_0)(y_k - y_0) < N+1$, onde N é o número máximo de células da aproximação (por exemplo 750). Algoritmo usado (força bruta): inicialmente o maior n é escolhido de forma que 2^n seja menor que o lado do MBR original. Então, em repetições sucessivas, n é diminuído de 1 enquanto as condições anteriores valerem.

Os pontos x_0, x_1,... x_k definem um conjunto de linhas paralelas ao eixo vertical que chamaremos de X_0, X_1,... X_k. De um modo semelhante, definimos as linhas horizontais Y_0, Y_1,... Y_k. Para cada par de linhas verticais (X_i, X_{i+1}) computamos sua interseção com o polígono original. Para tanto, estamos usando uma versão simplificada do algoritmo de Sutherland-Hodgman (ROGER, 1986). Nós chamamos cada polígono resultante de P_i. Para cada polígono P_i, computamos a sua interseção com os pares de linhas horizontais (Y_j, Y_{j+1}). O resultado para cada par de linhas é a interseção do polígono original com a célula (i, j). Finalmente, para cada célula, computamos a área dessa interseção e a classificamos: *Cheio*, *Muito*, *Pouco* ou *Vazio*. As células *Cheio* e *Vazio* podem ser classificadas por inspeção, sem a necessidade de se computar sua área. A única otimização realizada é armazenar os pontos de interseção da linha X_{i+1} que

foram computadas para o par de linhas (X_i, X_{i+1}) e as usar no cálculo do par (X_{i+1}, X_{i+2}). O mesmo é feito para as linhas horizontais.

4.2.2.6 Compactação

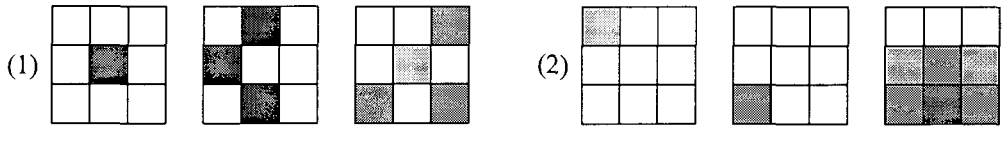
A observação do fato de que a maior parte das células é vazia ou cheia (80%, no caso de 750 pontos) nos levou a estudar a viabilidade da aplicação de algoritmos de compactação (sem perda) da aproximação raster. De fato, utilizando-se padrões de 3×3 células, foi possível obter índices de compactação muito bons, resultando na aproximação compactada em apenas 40% do tamanho original. Utilizando-se padrões de 2×2 células, a taxa de compressão não foi tão boa. Por outro lado, utilizando-se padrões de 4×4 células obtivemos uma taxa de compressão um pouco melhor (em torno de 35%), mas às custas da velocidade de descompactação e exigências maiores em temos de espaço para as estruturas intermediárias. Assim, optamos por usar os padrões de 3×3 células.

A compactação da aproximação raster permite uma densidade maior de informação, possibilitando o uso de uma maior quantidade de células, o que como vimos significa uma melhor capacidade de decisão e filtragem no passo 2. No entanto, esta abordagem torna evidente um problema que até então não havíamos tratado: o fato de que as aproximações terão tamanhos variáveis. Cabe lembrar no entanto que esse problema de forma alguma surgiu com a compactação: quando afirmamos que utilizaremos p células para uma aproximação, na verdade estamos anunciando o limite superior de células para tal aproximação. Certamente haverá aproximações $m \times n$ tais que o número de células seja diferente (e portanto menor) do que p. Além disso, como estabelecemos que as células têm tamanho igual a potências de dois, para aumentarmos o número de células em uma aproximação qualquer devemos dividir o tamanho de cada célula pela metade, o que resultaria em geral em multiplicar o número total de células por quatro. Como somente podemos afirmar que $2m \times 2n$ é menor do que p se $m \times n$ é menor do que $p/4$, temos que se $m \times n$ é maior do $p/4$ não é possível aumentar o número de células na aproximação. Esse resultado significa que teremos uma diferença de até quatro vezes entre a maior e a menor aproximação. Assim, mesmo se não utilizássemos a compactação, teríamos de lidar com aproximações de tamanhos diferentes.

4.2.2.6.1 O Algoritmo de Compactação

O algoritmo de compactação foi desenvolvido através de testes estatísticos sobre os dados, que demonstraram claramente a ocorrência predominante de células cheias e vazias sobre as células de pouca e muita interseção. Além disso, a simplicidade foi um dos requisitos mais importantes para determinar o algoritmo de compactação, já que a descompactação de forma alguma deveria penalizar a execução da consulta como um todo.

Tendo isso em vista, as células de uma assinatura foram separadas em grupos de 3×3 células, e a cada grupo foi atribuído um código fixo. Como temos 4^9 possibilidades diferentes de células, gastaremos 18 bits para codificar cada uma. No entanto, alguns tipos de células são altamente improváveis de ocorrer (Figura 4-20), enquanto outras possuem uma frequência bastante alta. Apenas para ilustrar esse fato, algumas estatísticas foram realizadas sobre os diversos conjuntos de dados com assinaturas de 750 pontos. Os resultados obtidos mostraram que o grupo cheio (grupo em que todas as células estão cheias) juntamente com o grupo vazio (grupo em que todas as células estão vazias) representam aproximadamente 40% de todos os grupos de células.

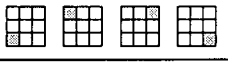


(1) Grupos de ocorrência improvável, e (2) grupos de ocorrência bastante comum em qualquer conjunto de dados. Na prática, dificilmente observaremos a ocorrência de uma célula cheia ao lado de uma célula vazia ou "tabuleiros de xadrez", pois os contornos dos objetos espaciais tendem a ser mais complexos e irregulares, dificilmente coincidindo os seus vértices com as coordenadas das células.

Figura 4-20 - Exemplos de padrões 3×3 de células

O nosso algoritmo de compactação é uma adaptação do algoritmo de Huffman, no qual o conjunto de padrões foi dividido em quatro classes. Basicamente, o algoritmo de compactação atribui códigos com poucos bits aos padrões que ocorrem com maior frequência, e códigos com muitos bits aos padrões que têm poucas chances de ocorrerem. A Tabela 4-12 mostra os códigos gerados para alguns padrões. É importante notar que o prefixo do código para cada classe deve ser único.

Tabela 4-12 - Exemplos de padrões, os códigos e frequência deles (Brasil, 550 células). O prefixo 11 é usado para codificar outros padrões que usem mais *bits*.

Padrão	Número de ocorrências (%)	Código binário
	22%	00
	18%	01
	14%	10000, 10001 10010, 10011
	12%	10100, 10101 10110, 10111

A principal diferença entre eles está na construção e no armazenamento da árvore de códigos: a árvore é construída uma vez, baseada na análise de um conjunto grande de dados, e armazenada como um parâmetro do algoritmo, em separado dos dados compactados. Dessa forma, a compactação, embora não tão eficiente, pode ser realizada de forma mais rápida e sem a necessidade do armazenamento da árvore junto aos dados. Além disso, o descompactador pode ser manualmente otimizado para uma determinada árvore, determinada antes da compilação do descompactador. Em nossos testes, as árvores de compactação alcançaram um tamanho em torno de 15 Kbytes e taxas de compactação, como vimos, em torno de 40% do tamanho original. O tamanho total das chaves com a assinatura compactada é menor que o tamanho das chaves em BRINKHOFF et al. (1994). Os resultados obtidos indicam uma quase independência entre o conjunto de dados e a árvore gerada. Uma maior dependência foi no entanto notada entre a nossa medida de qualidade da aproximação e a taxa de compactação: quanto melhor a qualidade da aproximação, maior a taxa de compressão.

A Tabela 4-13 mostra o tamanho médio das assinaturas e a sua compactação. Além disso, para fins de comparação, mostramos também os valores para a combinação de aproximações 5C e ER-EL. A compactação usada para tal fim foi a abordagem *base + deslocamento*, onde a *base* é um par de coordenadas contendo a parte mais significativa dos bytes e o *deslocamento* são coordenadas com a parte menos significativa a serem somadas à *base*. Finalmente, cabe dizer que dependendo da precisão das coordenadas e do tamanho do objeto, nem sempre será possível compactar as aproximações 5C e ER-EL. Note que a melhor compactação possível segundo esse esquema é de 44 bytes, maior do que a média de 4CRS com 750 células (62 e 80 correspondem à outra compactação possível e nenhuma compactação, respectivamente).

Tabela 4-13 - Tamanho médio em bytes das assinaturas compactadas conjunto (Brasil)

	4CRS (250 células)	4CRS (350 células)	4CRS (550 células)	4CRS (750 células)	5C & ER-EL
Tamanho original	35.8	50.2	77.0	104.5	80
Tamanho compactado	19.7	25.44	33.8	42.5	44/62/80

4.2.2.6.2 Lidando com Aproximações de Tamanhos Diferentes

Conforme explicado anteriormente, as aproximações farão parte das chaves dos polígonos, o que significa que serão armazenadas no índice espacial, por exemplo uma R*-Tree. Conforme BRINKHOFF et al. (1994), tal abordagem é razoável pois calcular uma aproximação, tanto a 4CRS quanto 5C e ER-EL, é uma operação relativamente demorada e que envolve percorrer em geral uma grande base de dados contendo os polígonos. Além disso, podemos esperar utilizar estas aproximações várias vezes depois.

Para podermos lidar com o tamanho variável das aproximações sem desperdiçar muito espaço (o que aumentaria o número de acessos ao disco) devemos agrupar um número razoável de aproximações em um *bucket*. Este número deve ser escolhido levando-se em conta o tamanho médio das chaves, mantendo-se em mente que pode haver casos de *overflow*. Experimentalmente, para cada limite p de pontos podemos determinar o tamanho médio da chave compactada. Adicionando uma margem de segurança podemos reduzir o número de *overflows* para limites que não comprometam o desempenho geral. A Figura 4-21 mostra a distribuição para o tamanho da aproximação 4CRS quando o limite p é fixado em 750 células.

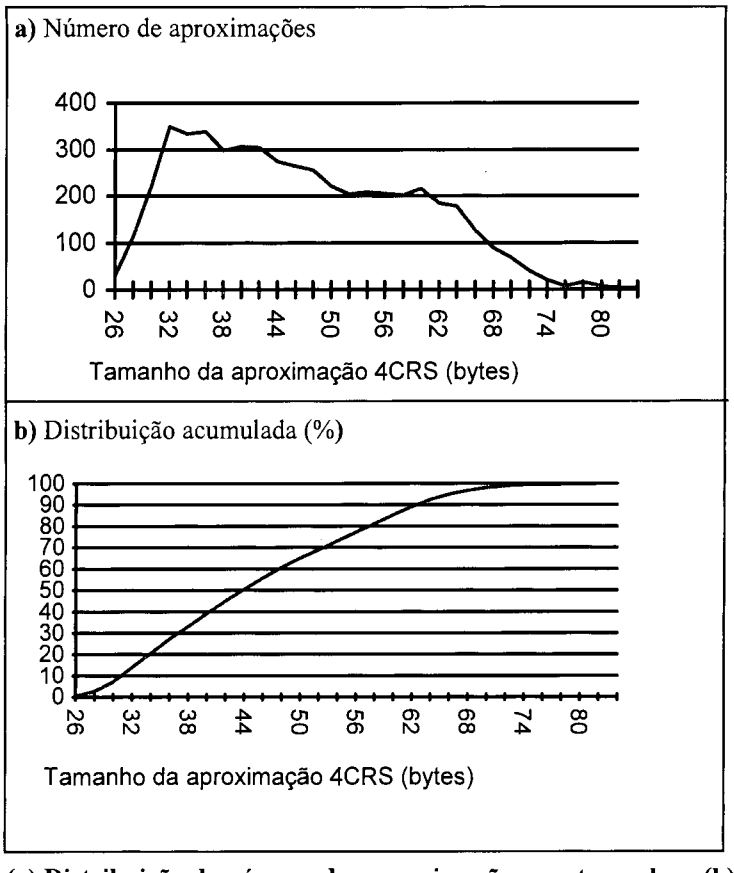


Figura 4-21 - (a) Distribuição do número das aproximações por tamanho e (b) distribuição acumulada.

Portanto, devemos adaptar a estrutura de dados espaciais utilizada como índice no passo 1 do processador para lidar com *buckets* e *overflows* para podermos resolver o problema das chaves de tamanho variável. É importante observar que em geral tais estruturas já trabalham com *buckets* (SAMET, 1990) para obter um acesso eficiente ao disco, o que significa que a adaptação de tais estruturas não é muito trabalhosa. Além disso, os *buckets* normalmente usados na família de estruturas de dados das R-Trees têm taxa de utilização de espaço entre 50% e 100%. Isto significa que a adaptação a ser feita em tais estruturas não é muito difícil como pudemos verificar adaptando a R*-Tree.

4.2.3 Resultados Experimentais

Conduzimos uma série de experimentos para confirmar a superioridade da aproximação 4CRS sobre as outras aproximações propostas. Para tanto, implementamos o MSQP completo, utilizando para o passo 1 a R*-Tree e para o passo 3 o algoritmo de *plane-sweep*. O passo 2 foi implementado com a abordagem 4CRS e com a abordagem utilizada em BRINKHOFF et al. (1994), que é a 5C & ER-EL, e que vinha a ser a melhor abordagem até o presente momento.

4.2.3.1 Dados de Teste

Nossa massa de teste é composta de vários conjuntos de polígonos que contêm até quarenta e cinco mil polígonos. Usamos dados de várias origens como algumas municipalidades em países europeus, condados americanos e a malha municipal brasileira (IBGE, 1996). O número médio de pontos por polígono em cada conjunto de dados variou de 22 a 96 (Tabela 4-14). Como sugerido em BRINKHOFF et al. (1994) nós geramos um outro conjunto de dados (chamado de *conjunto'*) deslocando os polígonos originais aleatoriamente em suas coordenadas x e y, e executando a junção espacial usando *conjunto* × *conjunto'*. A única exceção a esta regra é a junção *Brasil-A* × *Brasil-B*. Estes conjuntos de dados foram gerados como segue: o conjunto de dados *Brasil* foi expandido aleatoriamente, deslocado, girado e replicado 9 vezes (*Brasil-A*) e 4 vezes (*Brasil-B*).

Tabela 4-14 - Número de pontos e vértices em cada conjunto de dados

Conjunto de dados	Número de Polígonos	Média de vértices
América do Sul	228	54
Europa	249	96
SUL dos EUA	496	22
Brasil	5081	80
Brasil-A	45.729	80
Brasil-B	20.324	80

4.2.3.2 Comparação das Junções Espaciais

Apresentaremos agora a média dos resultados obtidos em nossos testes, de forma a permitir a comparação com as outras aproximações. Para computar os valores médios, descartamos os resultados do conjunto "Sul dos EUA" devido a sua extrema uniformidade — muitos dos condados são quadriláteros. Realizamos a comparação da abordagem 4CRS com duas outras aproximações: 5C-ER (56 bytes fixos) e 5C & ER-EL (80 bytes fixos). Apenas para relembrar, o tamanho médio da aproximação 4CRS foi de 42,5 bytes. Além disso, excluímos também os resultados dos conjuntos de dados "Brasil-A" e "Brasil-B" pois os mesmos são derivados do conjunto "Brasil".

O conjunto de entrada para esse passo é o conjunto de pares de candidatos, resultado da junção de MBRs do passo 1, implementado através de uma R*-Tree. É importante notar que não importa qual seja o método utilizado para computar o passo 1, sempre será obtido o mesmo conjunto de candidatos. Isto ocorre porque o resultado do

passo 1 é uma junção de MBRs. Apenas a ordem em que os candidatos aparecem é que pode ser diferente.

Tabela 4-15 - Características dos conjuntos de dados

	Brasil	Sul dos EUA	Europa	América do Sul
% de pares de MBRs candidatos que possuem interseção	65.49%	82.21%	68.63%	67.86%

A Tabela 4-15 descreve os conjuntos de dados em termos da relação entre polígonos que se interceptam e pares de MBRs candidatos. Nas Tabela 4-16 e Tabela 4-17, as três primeiras colunas mostram a classificação dos filtros para cada par de candidatos. A coluna **Aceitos** mostra a porcentagem de pares candidatos que foram identificados como pares de polígonos que se interceptam. São interseções verdadeiras de polígonos que foram identificadas sem que fosse necessário ter acesso à sua representação exata. A coluna **Rejeitados** mostra a porcentagem de pares de candidatos que foram identificados como polígonos que não possuem interseção. De novo, não foi necessário ter acesso à representação exata dos polígonos para decidir. A coluna **Candidatos** mostra a porcentagem dos pares de candidatos que o filtro não consegui identificar, e que servirão de entrada para o passo 3. A coluna **Rejeitados Identificados** mostra a porcentagem de pares de polígonos sem interseção que foram identificados como tal pelo filtro, e a coluna **Aceitos Identificados** mostra porcentagem dos pares de polígonos com interseção que foram identificados pelo filtro.

Tabela 4-16 - Desempenho dos filtros (a) 4CRS-350 células, (b) 4CRS-750 células e (c) 5-C/ER&EL (os melhores resultados estão em negrito)

	Aceitos	Rejeitados	Candidatos (menor é melhor)	Rejeitados Identificados	Aceitos Identificados
Brasil					
(a)	57.23%	23.23%	19.55%	67.29%	87.39%
(b)	**59.96%**	**26.30%**	**13.74%**	**76.20%**	**91.56%**
(c)	37.15%	25.75%	37.10%	73.69%	57.10%
Sul dos EUA					
(a)	78.78%	12.17%	9.04%	68.41%	95.84%
(b)	**79.37%**	13.74%	**6.89%**	**77.01%**	**96.60%**
(c)	62.51%	**16.26%**	21.24%	80.22%	77.92%
Europa					
(a)	59.73%	20.49%	19.77%	65.33%	87.04%
(b)	**62.28%**	**23.48%**	**14.24%**	**74.85%**	**90.75%**
(c)	41.39%	22.48%	36.13%	70.36%	60.82%
América do Sul					
(a)	60.45%	21.22%	18.33%	66.02%	89.08%
(b)	**62.48%**	**24.18%**	**13.34%**	**74.88%**	**92.28%**
(c)	41.50%	24.49%	34.01%	72.58%	62.63%

Tabela 4-17 - Comparação dos métodos (taxa média, sem o conjunto "Sul dos EUA")

	Aceitos	Rejeitados	Candidatos (menor é melhor)	Rejeitados Identificados	Aceitos Identificados
4CRS (350 células)	59.1%	21.6%	19.2%	66.2%	87.8%
4CRS (750 células)	**61.5%**	**24.6%**	**13.7%**	**75.2%**	**91.53%**
5-C & ER (nossos resultados)	23.5%	23.6%	52.8%	71.4%	35.2%
5-C & ER-EL (nossos resultados)	40.0%	23.6%	36.4%	71.4%	60.2%
5-C & ER (Europa A) BRINKHOFF et al., 1994	23%	23%	54%	66%	30%
5-C & ER-EL (Europa A) BRINKHOFF et al. 1993b e 1994	40.5%	20.8%	40.84%	66.3%	59.2%

É importante notar que a coluna **Candidatos** nas Tabela 4-16 e Tabela 4-17 é o fator determinante do problema. Esses números mostram o tamanho do conjunto de entrada para o passo 3, que como vimos domina o tempo total da execução da consulta. Desta forma, taxas menores significam que poucos testes de interseção exata de polígonos deverão ser realizados. Como pode ser observado, os resultados médios obtidos com a abordagem 4CRS (13,7% dos MBRs que se interceptam) mostram uma taxa reduzida em mais de 60% quando comparada com os melhores resultados já obtidos (5-C & ER-EL, 36.4% dos MBRs que se interceptam) anteriormente ao uso de nosso filtro. Conforme mencionamos anteriormente, a combinação 5-C & ER-EL ocupa mais espaço em bytes, mesmo quando compactada, do que a abordagem 4CRS. Por isso,

podemos esperar custos de entrada e saída menores para a nossa abordagem, já que teremos de manipular menos bytes.

BRINKHOFF et al. (1994) omitiram o tempo gasto no algoritmo que testa a interseção usando a aproximação 5C & ER-EL por que este tempo não afeta significativamente o desempenho global da junção espacial. No entanto, decidimos mostrar este tempo na Tabela 4-18, por se tratar da única parte do algoritmo que poderia afetá-lo, o que como pode ser observado não ocorre. A Tabela 4-18 mostra que o tempo gasto nos dois algoritmos é aproximadamente o mesmo e que não há impacto desta parte do algoritmo no desempenho global da junção espacial.

Tabela 4-18 - Média do tempo gasto no algoritmo de comparação de aproximações — conjunto de dados Brasil, 5081 polígonos e 33.188 pares de MBRs candidatos.

	Tempo (segundos)
5C & ER-EL	2.34
4CRS	2.48

A descompactação da aproximação 4CRS e a comparação entre as assinaturas gastou aproximadamente o mesmo tempo que a comparação pela abordagem usando as aproximações 5C & ER-EL. Este resultado reflete o fato que a abordagem com a aproximação 4CRS utiliza apenas as rápidas operações com inteiros: deslocamento de *bits* e operações como AND, OR e XOR. Em contraste, a abordagem com as aproximações 5C & ER-EL usa algumas operações em ponto flutuante, como multiplicação e divisão, algumas com precisão dupla — tais operações sempre são um pouco mais lentas do que as operações com inteiros.

Os tempos foram medidos da seguinte forma: primeiro medimos o tempo de executar apenas o passo 1 e depois o subtraímos do tempo gasto para executar o passo 1 e 2. O processo foi repetido dez vezes para cada conjunto de dados e foi tomado o tempo médio. Tivemos de adotar essa abordagem por que o processador de junções espaciais em passos múltiplos não é sequencial: a execução dos passos ocorre de maneira intercalada. A máquina de teste foi uma SUN Sparc20 com 64 Mb de memória principal. Embora este teste tenha sido realizado apenas com o conjunto de dados "Brasil", o resultado mostra que o impacto no tempo total de execução da consulta é bem pequeno — menos de 0,8%, e inclui o tempo gasto na descompactação das assinaturas. Além disso, este tempo poderia ser melhorado com o uso de um cache para guardar as últimas assinaturas descompactadas e evitar o trabalho repetido. No entanto,

como isso significaria uma melhoria apenas marginal no tempo total de execução da consulta, optamos por não implementar tal melhoramento.

De forma a não favorecer a nossa abordagem, no passo 3 apenas o tempo gasto com o acesso ao disco foi medido e incluído nos resultados finais. A principal razão de termos assim procedido foi por termos utilizado o algoritmo padrão de *plane-sweep* para implementar o passo 3, e que BRINKHOFF et al. (1994) mostrou ser muito ineficaz. De fato, ele pode ser substituído com vantagens pela R*-tree de trapézios com bons ganhos de desempenho em termos de CPU às custas de um pouco mais de tempo para acesso a disco. Ou ainda pela abordagem SID (detecção simbólica de interseções) apresentada em HUANG (1997a). Assim, computando apenas o tempo gasto com o acesso a disco, garantimos que qualquer melhoria realizada nesse passo não afetará para menos a vantagem da nossa abordagem sobre a concorrente — afinal, a nossa abordagem realiza menos testes de interseção, influindo decisivamente no tamanho do conjunto de entrada para esse passo e consequentemente no tempo gasto para executá-lo. Qualquer que seja o algoritmo utilizado para implementar o passo 3, certamente ele dependerá do tamanho do conjunto de entrada — quanto menor, menos tempo será necessário. Conforme nós mencionamos anteriormente, nossa abordagem chegou a reduzir em 60% o tamanho desse conjunto quando comparado com a melhor das outras abordagens.

Finalmente, implementamos uma suíte completa para a realização de experimentos com junções espaciais em C++, na mesma plataforma citada anteriormente. Utilizamos os compiladores GNU para melhorar a portabilidade: foi necessário apenas uma recompilação para executar testes em um PowerPC IBM com sistema operacional Linux. A suíte de testes inclui uma implementação do MSQP com todos os melhoramentos propostos em BRINKHOFF et al. (1993a) e BRINKHOFF et al. (1994), mais rotinas de importação de dados, criação das assinaturas e medição do tempo de CPU e acessos a disco, utilizando a mesma metodologia usada em BRINKHOFF et al. (1994) para medir esses tempos.

Implementamos o passo 1 através de uma R*-Tree e também através da abordagem 2DHM, cujos resultados estão na subseção 4.1.5. Aqui optamos por mostrar apenas os resultados obtidos com a R*-Tree de forma a podermos isolar os ganhos obtidos apenas com o uso da abordagem 4CRS. Na Tabela 4-19 mostramos o tamanho dos conjuntos de saída dos passos 1 e 3, que não sofrem alteração.

Tabela 4-19 - Descrição das junções

	Pares de MBRs (saída do passo 1)	Polígonos com Interseção (saída da junção)
Brasil × Brasil'	31.998	20.885
Brasil-A × Brasil-B	108.153	59.887

A Tabela 4-20 mostra os nossos resultados. Ambas as junções foram realizadas utilizando-se um *buffer* LRU de 23 páginas de 4Kb, o que representa cerca de 2% do tamanho do conjunto de dados "Brasil".

Tabela 4-20 - Desempenho global da junção - 4CRS (750 células) e 5C & ER-EL

	Número de testes de interseção exata a realizar (saída do passo 2)		Número de acessos a disco (todos os passos)		Tempo total de execução da consulta (em segundos)	
	5-C ER&EL	4CRS	5-C ER&EL	4CRS	5-C ER&EL	4CRS
Brasil × Brasil'	11.872	4.696	11.451	6.069	130.8	70,8
Brasil-A × Brasil-B	41.962	19.984	44.192	26.610	529.8	332,0

A coluna "Número de testes de interseção exata a realizar" mostra o tamanho do conjunto de saída do passo 2 e que serve de entrada para o passo 3, o passo que consome mais tempo de CPU e de acesso a disco segundo BRINKHOFF et al. (1994). Estes são os pares de candidatos que não puderam ser identificados pelos filtros geométricos, e que nas Tabelas Tabela 4-16Tabela 4-17 apareceram como porcentagem na coluna "Candidatos". A coluna "Número de acessos a disco" representa a soma de todos os acessos a disco. Cabe destacar que o tamanho das R*-Tree da junção de MBRs e o tamanho do conjunto de dados de entrada do passo 3 têm grande influência nesse número. Inclusive, o tamanho do conjunto de entrada do passo 3 pode ter uma influência muito grande no número total de acessos a disco se os polígonos possuírem muitos pontos — afinal, cada MBR no conjunto de entrada do passo 3 representa um polígono a mais a ser recuperado do disco. Cabe dizer ainda que as aproximações utilizadas no passo 2 são recuperadas pelo passo 1, pois são armazenadas junto com os MBRs no índice espacial.

O tamanho dos municípios brasileiros varia muito: há municípios na região Amazônica, como Altamira, que são maiores do que países como Bélgica e Holanda, enquanto outros, como Búzios, possuem poucos km^2. Além disso, na segunda junção, os conjuntos de dados "Brasil-A" e "Brasil-B" possuem um valor diferente para a área

média dos polígonos, o que levou a muitas mudanças de escala na aplicação da abordagem 4CRS. Essas mudanças de escala tiveram pouco impacto no nosso algoritmo: os ganhos de desempenho foram 45% e 37%, respectivamente, quando comparados com a melhor abordagem conhecida até então.

4.2.4 Variações da Abordagem 4CRS

Uma série de variações pode ser proposta para melhorar o desempenho da aproximação 4CRS. Em particular, proporemos aqui apenas o uso de um número maior de cores e de células de formatos diferentes.

4.2.4.1 Utilizando mais Cores (8CRS)

É possível utilizar mais cores na aproximação raster, refletindo melhor o grau de interseção entre as células e o polígono. Assim utilizando oito cores (3 bits), poderíamos ter a abordagem 8CRS, segundo a Tabela 4-21.

Tabela 4-21 - Tipos de células da aproximação 8CRS

Tipo de Célula	Interseção
Cheia	100%
Muito 3	entre 83,33% e 100%
Muito 2	entre 66,66% e 83,33%
Muito 1	entre 50% e 66,66%
Pouco 3	entre 33,33% e 50%
Pouco 2	entre 16,66% e 33,33%
Pouco 1	entre 0% e 16,66%
Vazio	0%

As possíveis interseções seriam então dadas pela Tabela 4-22.

Tabela 4-22 - Comparações usando a abordagem 8CRS

Tipo de Célula	Vazia	Pouco 1	Pouco 2	Pouco 3	Muito 1	Muito 2	Muito 3	Cheia
Vazia	Rejeitada	Rejeitada	Rejeitada	Rejeitada	Rejeitada	Rejeitada	Rejeitada	Rejeitada
Pouco 1	Rejeitada	*Candidato*	*Candidato*	*Candidato*	*Candidato*	*Candidato*	*Candidato*	Aceita
Pouco2	Rejeitada	*Candidato*	*Candidato*	*Candidato*	*Candidato*	*Candidato*	Aceita	Aceita
Pouco 3	Rejeitada	*Candidato*	*Candidato*	*Candidato*	*Candidato*	Aceita	Aceita	Aceita
Muito 1	Rejeitada	*Candidato*	*Candidato*	*Candidato*	Aceita	Aceita	Aceita	Aceita
Muito 2	Rejeitada	*Candidato*	*Candidato*	Aceita	Aceita	Aceita	Aceita	Aceita
Muito 3	Rejeitada	*Candidato*	Aceita	Aceita	Aceita	Aceita	Aceita	Aceita
Cheia	Rejeitada	Aceita	Aceita	Aceita	Aceita	Aceita	Aceita	Aceita

É importante observar que, comparando com o esquema adotado para a 4CRS, as novas cores foram utilizadas para redividir as células que antes pertenciam apenas aos tipos Muito e Pouco. As quantidades de células do tipo Cheia ou Vazia serão as mesmas tanto na aproximação 4CRS quanto na 8CRS. Assim, devemos comparar apenas as mudanças realizadas na área de indecisão. Na abordagem 4CRS teremos 25% dos casos de comparação entre as células de indecisão (Muito e Pouco) levando à aceitação, enquanto que na abordagem 8CRS essa taxa será de 41,66%.

No entanto, podemos observar que na área de indecisão, tanto na abordagem 4CRS quanto na 8CRS somente podemos decidir por aceitar ou tornar candidato, nunca rejeitar. Dessa forma, utilizando oito cores, podemos esperar apenas uma melhora na taxa de identificação de candidatos, e não na de rejeição. Como essa já é a nossa melhor taxa, por volta de 91,5% em média, as melhorias obtidas utilizando-se oito cores não foram muito significativas, de forma que resolvemos omitir esses resultados.

4.2.4.2 Utilizando Células Especiais

Uma outra abordagem para melhorar a qualidade das aproximações é definir algumas células com formato especial, de forma a manter alguma informação sobre a distribuição da área do polígono em determinada célula. A Tabela 4-23 ilustra alguns exemplos. Foram omitidas as células Muito, Pouco, Vazia e Cheia por serem idênticas às apresentadas nas abordagens anteriores.

Tabela 4-23 - Comparações usando a abordagem 8CRS

Célula	Descrição
	Célula com interseção apenas a Nordeste
	Célula com interseção apenas a Sudeste
	Célula com interseção apenas a Noroeste
	Célula com interseção apenas a Sudoeste

Célula	Descrição
	Célula com interseção predominante a ?
	Célula com interseção predominante a Sudeste
	Célula com interseção predominante a Noroeste
	Célula com interseção predominante a Sudoeste

Embora em tais abordagens seja possível decidir na chamada área de indecisão, os resultados assim obtidos não foram muito melhores que a abordagem anterior. Isso ocorre porque foram necessários vários bits para termos um número razoável de formatos diferentes de células (p.e, 4 bits por célula para podermos ter 16 tipos de células), o que aumentou muito o espaço necessário para armazenar a assinatura sobrecarregando a estrutura da R*-Tree.

4.2.4.3 Utilizando mais Pontos

A utilização de uma grande quantidade de pontos se mostrou como a alternativa mais vantajosa, capaz de proporcionar melhores resultados. Para tanto, é necessário desenvolver ou aperfeiçoar mecanismos de compactação mais eficientes, que permitam a utilização de um grande número de pontos sem ter o ônus de chaves muito grandes. Conforme podemos observar na Tabela 4-24, embora haja um aumento do tamanho da chave compactada a medida em que o número de pontos aumenta, esse incremento não é linear. Isso ocorre por que há um aumento muito grande do número de células cheias e vazias, diminuindo consideravelmente a área de indecisão da assinatura. Dessa forma, a entropia da assinatura também diminui, resultando em uma taxa de compactação mais alta. Assim, outros algoritmos para a compactação da aproximação 4CRS devem ser investigados, tais como lzw e gif.

Tabela 4-24 - Taxa de compactação para cada tamanho de assinatura

	4CRS (250)	4CRS (350)	4CRS (550)	4CRS (750)
Tamanho original	35,8	50,2	77,0	104,5
Tamanho compactado	19,7	25,4	33,8	42,5
Taxa de compactação	45%	50%	57%	60%

4.3 Sumário

Este capítulo apresentou as principais contribuições dessa tese: a Matriz de Hash Bidimensional (2DHM) e a Assinatura Raster de Quatro Cores (4CRS). Tais algoritmos estão em conformidade com a abordagem para o processamento de junções espaciais apresentada no capítulo anterior, chamada de MSQP, o que permite que possam ser combinados com outras abordagens. A Matriz de Hash Bidimensional implementa o passo 1 do MSQP e a Assinatura Raster de Quatro Cores o passo 2.

5. Conclusões

Neste capítulo apresentamos as conclusões obtidas com os trabalhos realizados. Esta Tese tratou do desempenho da realização de junções espaciais em sistemas de computação sequenciais, enfatizando os fatores que limitam a escalabilidade de tais junções e propondo novos algoritmos para superar tais limites. Embora os algoritmos tenham sido desenvolvidos tendo por base o modelo relacional de decomposição de consultas, o trabalho apresentado é relevante para o desempenho sistemas que suportem qualquer modelo de dados, tais como os bancos de dados espaciais, espaço-temporais e espaciais orientados a objetos. A operação de junção de polígonos, em especial com o uso do operador de interseção, é representativa de toda uma família de operadores que devem ser suportados por qualquer sistema, independentemente do seu modelo de dados. Além disso, os principais algoritmos e melhoramentos projetados foram implementados e testados, sendo também apresentados em fóruns de pesquisa reconhecidos pela comunidade acadêmica, tendo obtido boa aceitação, e já receberam citações em outros trabalhos.

A seção 5.1 apresenta as contribuições originais desta Tese. A Seção 5.2 indica as direções para a continuidade da pesquisa aqui realizada.

5.1 Contribuições dessa Tese

Estudamos o "estado da arte" em métodos para se realizar uma junção espacial, principalmente em seus aspectos teóricos e de projeto. Assim, propusemos e testamos alternativas que resultaram em melhorias significativas no desempenho de junções espaciais. As contribuições originais da Tese podem ser resumidas da seguinte forma:

5.1.1 Matriz de Hash Bidimensional

Apresentamos uma forma de se realizar junções espaciais segundo o paradigma *hash-join*, a 2DHM, e que está em conformidade com os modelos correntes de processamento de junções espaciais. Ao contrário de quase todas as outras abordagens, a abordagem 2DHM conseguiu manter a restrição de ter partições que não se sobrepõem, às custas de muito pouco inchaço nos dados. Além disso, conseguimos resolver satisfatoriamente o problema de se eliminar os pares de MBRs duplicados. A nossa abordagem se mostrou superior tanto no tempo gasto para a construção do índice quanto no tempo gasto para a realização da junção em si. Como foi visto, em nossos testes a

abordagem 2DHM superou em muito o desempenho da abordagem usando a R*-Tree (seis vezes mais rápido no melhor caso). Finalmente, foi proposta e avaliada uma fórmula para se estimar com precisão aceitável o inchaço nos dados provocados pelo uso de um *grid* regular em uma estrutura de dados do tipo *hash*.

5.1.2 Aproximação Raster de Quatro Cores

Apresentamos uma nova aproximação, chamada de 4CRS, para a implementação de filtros no processamento de junções espaciais, em particular mas não somente, cujo predicado seja o de interseção. Os resultados foram muito bons quando comparados com as técnicas existentes. Mesmo nos casos onde o ganho foi menor, nós obtivemos uma redução de 50% no número de testes de comparação exata de polígonos. Essa é a parte da junção espacial mais custosa pois envolve a transferência de grandes objetos do disco para a memória, e a tendência é que tais objetos se tornem cada vez maiores já que os SIGs estão se tornando cada vez mais precisos. Finalmente, conseguimos reduzir o tempo total de execução das consultas de teste em 37% e 45%.

5.2 Direções de Pesquisa para Trabalhos Futuros

A continuidade deste trabalho está garantida pela sua grande abrangência. De fato, a aplicação das técnicas aqui apresentadas foi explorada apenas no contexto das junções espaciais com o predicado de interseção. Apontaremos aqui as sequências naturais desse trabalho, com sugestões para a aplicação dos nossos algoritmos em outras áreas.

5.2.1 Paralelização dos Algoritmos

Dada a crescente disponibilidade de dados espaciais, paralelismo é uma boa alternativa para se lidar com junções espaciais. É importante notar que a abordagem 2DHM possui características que facilitam a paralelização do algoritmo. Um trabalho nesse sentido está ainda em sua fase inicial, de forma que não dispomos ainda de resultados práticos. Já a abordagem 4CRS não facilita mas também não interfere no uso de paralelismo.

5.2.2 Outras Consultas

Tanto a abordagem 4CRS quanto a 2DHM podem ser utilizadas para a realização de outros tipos de consultas espaciais, tais como a consulta com seleção de área (*query window*) e a junção de distância (*distance-join*).

5.2.3 Junções Espaço-Temporais

A abordagem 4CRS pode ser utilizada em um análogo temporal do MSQP para a realização eficiente de junções espaço-temporais. Um trabalho nesse sentido já está em andamento, com alguns resultados bem promissores (ZIMBRÃO et al., 1999).

6. Bibliografia

ABEL, D. J., et al. "Resequencing and Clustering to Improve the Performance of Spatial Joins", Technical Report ISS-37, CSIRO, Canberra, Australia, Dec 1996.

ARCTUR, D., ANWAR, J. A. and CHAKRAVARTHY, S. 1995. "Comparison and benchmarks for import of VPF geographic data from object-oriented aand relational database files". In: *Fourth International Symposium on Large Spatial Databases, SSD'95*, Portland, ME, USA, Jun 1995.

ARGE, L., et al. 1998. "Scalable Sweeping Based Spatial Join". In: *24th International Conference on Very Large Databases*, New York City, NY, USA, August, 1998.

BANCILHON, F., 1992. "The O2 Object-Oriented Database System". *Proceedings of the ACM SIGMOD Intl. Conf. on Management of Data,* San Diego, USA, Jun 1992.

BATTY, P. M. 1990 "Exploiting Relational Database Technology in a GIS", In: *Mapping Awareness Magazine*, July/August 1990.

BAUZER, C. e BOTELHO, M. 1996 "Tratamento do Tempo em SIG", In: *Proceedings of the GIS Brazil`96*, Curitiba, Brasil, Maio 1996.

BAUZER, C., PIRES, F. 1994. "Databases for GIS", *SIGMOD Record*, Vol 23, No 1, pp. 107-115, 1994.

BAYER, R., C. MCCREIGHT, 1972, "Organization and maintenance of large ordered indexes". In: *Acta Informatica*, Vol. 1, No. 3, pp. 173-189, 1972.

BECKMANN, N., KRIEGEL, H. P., SCHNEIDER, R. et al. 1990. "The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles". In: *Proceedings of the ACM SIGMOD Intl. Conf on Management of Data*, Atlantic City, NJ, 1990.

BELLER, A. 1991 "Spatial/Temporal Events in a GIS". In: *Proceedings of GIS/LIS 91*, pp. 666-775. Bethesda, Maryland, Feb 1991.

BERCHTOLD, S., KEIM, D. A., KRIEGEL, H. P. 1996. "The X-Tree: An Index Structure for High-Dimensional Data", In: *Proceedings of 22th International Conference on Very Large Data Bases*, Bombay, India 1996.

BRINKHOFF, T. KRIEGEL, H. P, SCHNEIDER, R. 1993b. "Comparison of Approximations of Complex Objects Used for Approximation-based Query Processing in Saptial Databse Systems". In: *Proceedings of 9th International Conference on Data Engineering*, Vienna, Austria, Mar 1993.

BRINKHOFF, T. KRIEGEL, H. P, SCHNEIDER, R. et al. 1994. "Multi-step Processing of Spatial Joins". In: *Proceedings of the 1994 ACM-SIGMOD Intl. Conference*, Minneapolis, USA, May 1994.

BRINKHOFF, T. KRIEGEL, H. P, SCHNEIDER, R. et al. 1995. "Measuring the Complexity of Polygonal Objects". In: *Proceedings of ACM International Workshop on Advances in Geographic Informatin Systems*, Baltimore, MD, USA, Dec 1995.

BRINKHOFF, T. KRIEGEL, H. P, SEEGER, B. 1993a. "Efficient processing of Spatial Joins Using R-Trees". In: *Proceedings of the 1993 ACM-SIGMOD Intl. Conference*, Washington, DC, USA, May 1993.

BRINKHOFF, T. KRIEGEL, H. P, SEEGER, B. 1996. "Parallel Processing of Spatial Joins Using R-trees", In: *Proceedings of 12th International Conference on Data Engineering*, New Orleans, LA, USA 1996.

CAMBRAY, B. 1993. "Three-Dimensional (3D) Modelling in a Geographical Database". In: *Proceedings of the Auto-Carto 11th International Conference on Computer Assisted Cartography*, pp. 338-347, Minneapolis, USA, Nov 1993.

CANDY, J. T. 1995. "Development of a Prototype Temporal Geographic Information System". Tese de M.Sc., Department of Geography - Simon Fraser University. USA, 1995.

CATTELL, R. G. G. 1994. "ODMG-93: A Standard for Object-Oriented DBMSs". In: *Proceedings of the ACM SIGMOD Intl. Conf. on Management of Data,* Minneapolis, USA, May 1994.

COMER, D. 1979. "The ubiquitous B-Tree". In: *ACM Computing Surveys,* Vol 11, No. 2, pp. 121-137, Jun 1979.

DATE, C.J. 1991. "Introdução a Sistemas de Banco de Dados". 4a Edição, Rio de Janeiro, Editora Campus, 1991.

DEPOMPE, B. 1996. *There's gold in databases*, 1st Edition, CMP Publications, NYC, Jan 1996.

DEWITT, D. et al. 1984. "Implementation Techhniques for main memory database systems". In: *SIGMOD'84, Proceedings of Annual Meeting,* Boston, Massachusetts, June 1984.

EDELSTEIN, H. 1996. *Technology how to: Mining data warehouses*, 1st Edition, CMP Publications, NYC, Jan 1996.

ESPERANÇA, C., SAMET, H. 1997. "Orthogonal Polygons as Bounding Structures in Filter-Refine Query Processing Strategies". In: *Proceedings of the 5th International Symposium on Spatial Databases*, Berlin, Germany, Jun 1997.

ESRI, ARC /Info User's Guide 1991. "ARC command references". *Environmental Systems Research Institute*, Redlands, 1991.

FALOUTSOS, C., KAMEL, I. 1994. "Beyond Uniformity and Independence: Analysis of R-trees Using the Concept of Fractal Dimension", In: *Proceedings of the 13th ACM Symposium on Principles of Database Systems (PODS)*, Minneapolis, Minnesota, USA, May 1994.

FAYAD, U. M. PIATETSKY-SHAPIRO, G., SMYTH, P. 1996. "From data mining to knowledge discovery", In: *Advances in Knowledge Discovery and Data Mining*, AAAI Press/MIT Press, CA, 1996.

GAEDE, V. 1995. "Optimal Redundancy in Spatial Database Systems". In: *Fourth International Symposium on Large Spatial Databases, SSD'95*, Portland, ME, USA, Jun 1995.

GAEDE, V., FALOUTSOS, C. 1996. "Analysis of n-dimensional Quadtrees Using the Hausdorff Fractal Dimension". In: *Proceedings of the 22nd VLDB Conference Mumbai* (Bombay), India, Aug. 1996.

GORDON, S. R. et al. 1994. *Final Report on Status of Spatial/Map Databases.* Technical Report of Oak Ridge National Laboratory, June 1994.

GÜNTHER, O. 1989. "The cell tree: An object oriented index structure for geometric databases". In: *Proc. 5th IEEE Int. Conf. on Data Engineering*, Los Angeles, California, USA, Feb 1989.

GÜNTHER, O. 1993. "Efficient Computation of Spatial Joins". In: *Proceedings of 9th International Conference on Data Engineering*, Vienna, Austria, Mar. 1993.

GÜNTHER, O. et al. 1998. "Benchmarking Spatial Joins A la Carte", in: *Proc. 10th Int. Conf. on Scientific and Statistical Databases*, IEEE Publications, New York, Aug. 1998.

GÜNTHER, O., BUCHMANN, A. 1990. "Research Issues in Spatial Databases". In: *ACM SIGMOD Record*, Vol. 19, pp. 61-68, 1990.

GÜTING, R. H. 1994. "An Introdution to Spatial Database Systems", *Invited Contribution to a Special Issue on Spatial Database Systems of the VLDB Journal*, Vol. 3, No. 4, October 1994.

GUTTMAN, A. 1984. "R-Trees: A Dynamic Index Structure for Spatial Searching". In: *Proceedings of the ACM SIGMOD Intl. Conf on Management of Data*, Boston, MA, USA, May 1984.

HAAS, L. M., CODY, W. F. 1991. "Exploiting extensible DBMS in Integrated GIS", In: Lecture Notes in Computer Science, Springer-Verlag, No. 525, pp. 423-450, 1991.

HUANG, Y. W., JING, N. 1997.: "Spatial Joins Using R-Trees: Breadth-First Traversal with Global Optimizations". In: *Proceedings of the 23rd VLDB Conference,* Athens, Greece, Aug. 1997.

HUANG, Y. W., JONES, M. C., RUNDENSTEINER, E. A. 1997a.: "Improving Spatial Intersect Joins Using Symbolic Intersect Detection". In: *Proceedings of the 5th International Symposium on Advances in Spatial Databases, SSD'97*, Berlin, Germany, July 1997.

HUANG, Y.W., JING N., RUNDENSTEINER, E. A. 1997b. "A Cost Model for Estimating the Performance of Spatial Joins Using R-trees", In: *9th International Conference on Scientific and Statistical Database Management (SSDBM'97)*, Olympia, Washington, USA, Jul 1997.

IBGE, 1996. *Malha Municipal Digital do Brasil - 1994*, Fundação Instituto Brasileiro de Geografia e Estatística, Rio de Janeiro, 1996.

KAMEL, I., FALOUTSOS, C. 1993. "On Packing R-trees", In: *Proceedings of the 2nd International Conference on Information and Knowledge Management (CIKM)*, Washington, DC, USA, Nov 1993.

KAUSHIK, S., RUNDENSTEINER, E. A. 1998. "SVIQUEL: A Spatial Visual Query and Exploration Language". In: *9th International Conference on Database and Expert Systems Applications - DEXA,* pp 290-299, Vienna, Austria, Aug 1998.

KOUDAS, N., SEVCIK, K. 1997. "Size Separation Spatial Join". In: *Proceedings of ACM SIGMOD International Conference on Management of Data*, Tucson, Arizona, USA. May 1997.

LANGRAN, G. 1992. *Time in Geographical Information Systems.* 1st Edition, Taylor & Francis, London, 1992.

LAURINI, R. 1993. "Sharing Geographic Information in Distributed Databases", In: *Proceedings of the 16thUrban Data Management Symposium*, pp 26-41, Vienna, Austria, Sep 1993.

LINDSAY, B., MCPHARSON, J., PIRAHESH, H. 1987. "A data management extension arqchitecture", in: *Proceedings of the ACM-SIGMOD International Conference on Management of Data,* . 220-226, S. Francisco, USA, May 1987.

MAMOULIS, N., PAPADIAS, D. 1999. "Integration of Spatial Join Algorithms for Joining Multiple Inputs". In: *Proceedings of the ACM Conference on the Management of Data (SIGMOD)*, Philadephia, PA, USA, May 1999.

MCKEOWN, D.M. JR. 1984. "Digital cartography and photo interpretation from a data base viewpoint". In: *New Applications of Data Bases*, 1st Edition, G. Gardarin and E. Gelenbe Academic Press, pp 19-42, London, 1984.

MING, L. L., RAVISHANKAR, C. V. "Spatial Hash-Joins". In: *Proceedings of the 1996 ACM-SIGMOD Conference*, Montreal, Canada, Jun 1996.

MING, L. L., RAVISHANKAR, C. V. 1995. "Generating Seeded Trees From Data Sets". In: *Proceedings of 4th International Symposium on Large Spatial Databses*, Portland, ME, USA, August, 1995.

NSDI, THE WHITE HOUSE 1994. "Coordinating Geographic Data Acquisition and Access: The National Spatial Data Infrastructure (NSDI)", *Executive Order 12906*, April 1994.

PAGEL, B. U. et al. 1993. "Towards an Analysis of Range Query Performance", In: *Proceedings of the 12th ACM Symposium on Principles of Database Systems (PODS),* Washington, DC, USA, May 1993.

PAPADIAS D., MAMOULIS, N., THEODORIDIS, Y. 1999a. "Constraint-based Processing of Multi-way Spatial Joins". Technical Report Series HKUST-CS99-02 - The Hong Kong University of Science & Technology, Department of Computer Science, January 1999.

PAPADIAS D., MAMOULIS, N., THEODORIDIS, Y. 1999b. "Processing and Optimization of Multi-way Spatial Joins Using R-trees", In: *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (PODS), Philadephia, PA, USA, May 1999.

PATEL, J. M., DEWITT, D. 1996. "Partition Based Spatial-Merge Join". In: *Proceedings of the 1996 ACM-SIGMOD Conference*, Montreal, Canada, June 1996.

PREPARATA, F. P., SHAMOS, M. L. 1985. *Computational Geometry*, 1st Edition, Springer Verlag, Germany, 1985.

ROGER, D. F. 1986. *Procedural Elements For Computer Graphic.* 1st Edition, McGraw-Hill Book Company, NYC, 1986.

SAIF, MELP 1994. *Spatial Archive and Interchange Format (SAIF): Formal Definition Release 3.1*, Reference Series Volume 1 - Surveys and Resource Mapping Branch - Ministry of Environment, Lands and Parks (MELP) - Province of British Columbia - Canada, April 1994.

SAMET, H. 1990. *The Design and Analysis of Spatial Data Structure*, 1st Edition, Addison-Wesley Publishhing Company, NYC, 1990.

SHAPIRO, G. P., FRAWLEY W. J. (Eds.). *Knowledge Discovery in Databases.* AAAI/MIT Press, 1991.

SMALWORLD 1995, *Technical Papers* - disponíveis ao público no endereço Internet: "http://www.w3net.com/sworld/tech".

SQL3, 1995. *X3H2-95-084/DBL:YOW-004, (ISO-ANSI Working Draft) Database Language SQL (SQL3),* Jim Melton Editor, March 1995.

STONEBRAKER, M., KEMNITZ, G.: 1991b. "The Postgres Next Generation Database Management System". In: *Communications of ACM*, Vol. 34 No.10, 1991.

THEODORIDIS, Y., STEFANAKIS, E., SELLIS, T., 1998a. "Cost Models for Join Queries in Spatial Databases". In: *Proceedings of the 14th IEEE Conference on Data Engineering, ICDE'98,* Orlando - FL, USA, February 1998.

THEODORIDIS, Y., STEFANAKIS, E., SELLIS, T., 1998b. "Efficient Cost Models for Spatial Queries Using R-trees", In: *IEEE Transactions on Knowledge and Data Engineering*, 1998.

VEENHOF, H. M., et al. "Optimisation of Spatial Joins Using Filters". In: *Advances in Databases, 13th British National Conference on Databases,* BNCOD 13, Manchester, United Kingdom, July 1995.

WORBOYS, M.F. "Object Oriented Approaches to Geo-referenced Information", In: *International Journal of Geographical Information Systems* Vol 8, pp. 385-399, 1994.

ZIMBRÃO, G., SOUZA, J. M. 1997a. "Using Raster Approximations For Processing of Spatial Joins". COPPE/UFRJ - Relatório Técnico ES-442/97, 1997.

ZIMBRÃO, G., SOUZA, J. M. 1997b. "Realização Eficiente de Junções Espaciais Utilizando Hash-Join". In: *Anais do XII Simpósio Brasileiro de Banco de Dados* - Fortaleza, CE, Brasil, Outubro, 1997.

ZIMBRÃO, G., SOUZA, J. M. 1998. "A Raster Approximation For Processing of Spatial Joins". In: *Proceedings of VLDB'98 - 24th International Conference on Very Large Databases*, New York City, NY, USA, Aug 1998.

ZIMBRÃO, G., SOUZA, J. M. 1996: "Futuras Direções Em SIGs: O Que Os Usuários Devem Esperar Dos Novos Sistemas". In: *Anais do I SEGEO* - RJ, outubro de 1996.

ZIMBRÃO, G., ALMEIDA, V. T., SOUZA, J. M., 1999 "The Temporal R-Tree". Technical Report ES492/99, COPPE/Federal University of Rio de Janeiro, Brazil, March 1999.