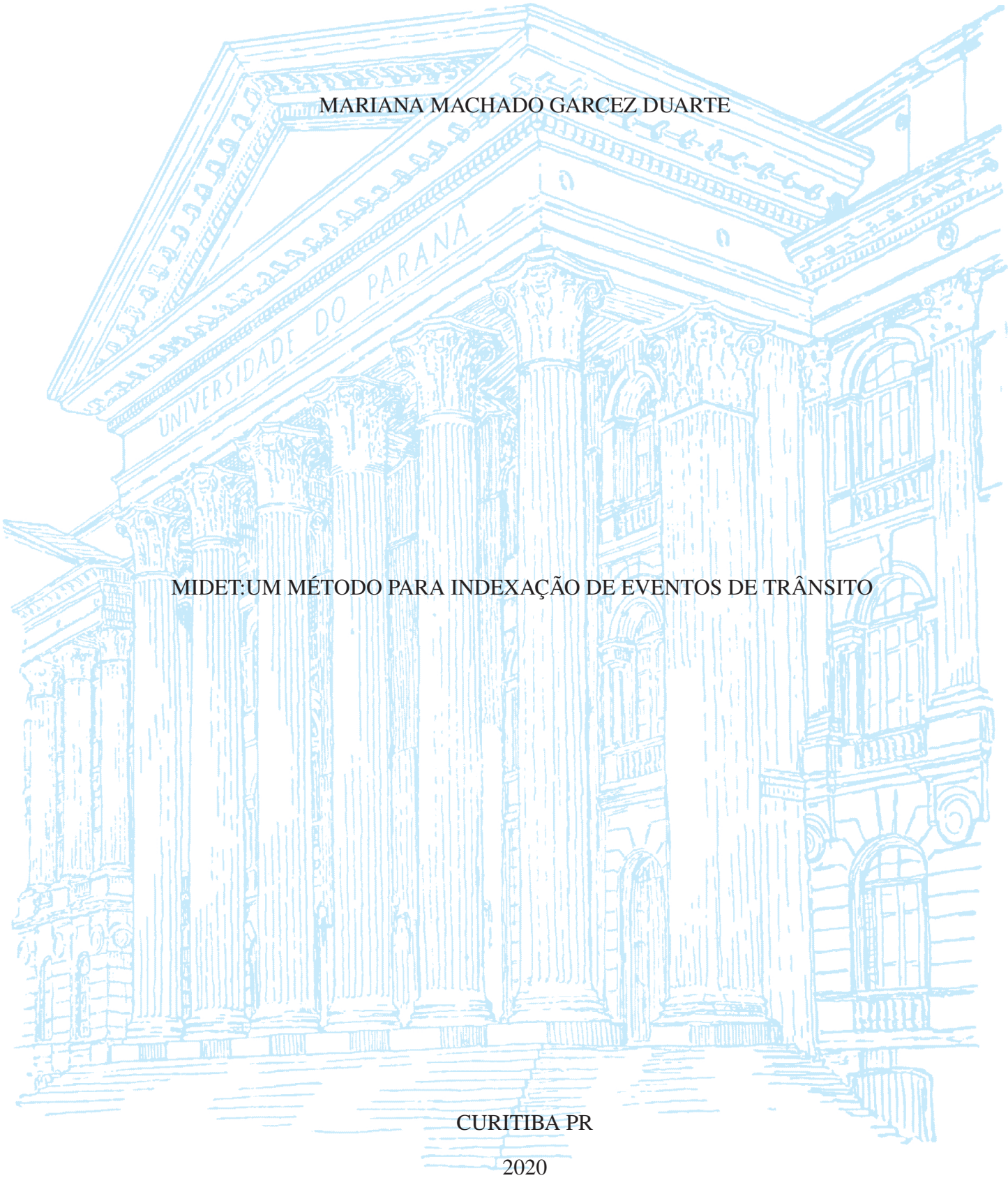
UNIVERSIDADE FEDERAL DO PARANÁ

MARIANA MACHADO GARCEZ DUARTE



MIDET:UM MÉTODO PARA INDEXAÇÃO DE EVENTOS DE TRÂNSITO

CURITIBA PR

2020

MARIANA MACHADO GARCEZ DUARTE

MIDET:UM MÉTODO PARA INDEXAÇÃO DE EVENTOS DE TRÂNSITO

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Profa Dra Carmem Satie Hara.

Coorientador: Profa Dra Rebeca Schroeder Freitas.

CURITIBA PR

2020

CATALOGAÇÃO NA FONTE – SIBI/UFPR

D812m Duarte, Mariana Machado Garcez

MIDET: um método para indexação de eventos de trânsito [recurso eletrônico]/ Mariana Machado Garcez Duarte, 2020.

Dissertação (Mestrado) - Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná. Área de concentração: Ciência da Computação.
Orientador: Profa Dra Carmem Satie Hara.
Coorientador: Profa Dra Rebeca Schroeder Freitas.

1. Planejamento urbano. 2. Espaço. 3. Trânsito. I. Hara, Carmem Satie. II. Freitas, Rebeca Schroeder. III. Título.

CDD 910.1711

Bibliotecária: Vilma Machado CRB9/1563

MINISTÉRIO DA EDUCAÇÃO
SETOR DE CIENCIAS EXATAS
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO INFORMÁTICA - 40001016034P5

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da dissertação de Mestrado de **MARIANA MACHADO GARCEZ DUARTE** intitulada: **MIDET: Um Método para Indexação de Eventos de Trânsito**, sob orientação da Profa. Dra. CARMEM SATIE HARA, que após terem inquirido a aluna e realizada a avaliação do trabalho, são de parecer pela sua Aprovação no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 06 de Abril de 2020.

Carmem Satie Hara

CARMEM SATIE HARA
Presidente da Banca Examinadora (UNIVERSIDADE FEDERAL DO PARANÁ)

Cristina Ciferri

CRISTINA DUTRA DE AGUIAR CIFERRI
Avaliador Externo (INSTITUTO DE CIêNCIAS MATEMáTICAS E DE COMPUTAçãO ICMC - USP)

REBECA SCHROEDER FREITAS
Coorientador - Avaliador Externo (UNIVERSIDADE DO ESTADO DE SANTA CATARINA)

WAGNER MACHADO NUNAN ZOLA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

MINISTÉRIO DA EDUCAÇÃO - UNIVERSIDADE FEDERAL DO PARANÁ - Setor de Ciências Exatas - Coordenação de Pós-Graduação em Informática

RESUMO

No ambiente urbano, os dados coletados a partir de eventos de trânsito podem se tornar elementos de estudo para fomentar o planejamento de cidades e metrópoles. Apesar de muitos dados já terem sido coletados, o desafio é transformar este conjunto de dados espaço-temporais em conhecimento de mobilidade para auxiliar nesse planejamento. Eventos de trânsito, como engarrafamentos e alertas, são continuamente produzidos através de aplicativos como o Waze. Em razão da velocidade com que estes dados são reportados, aplicativos em geral armazenam eventos como registros individuais. Embora este modelo de armazenamento possa garantir um baixo custo de inserção em uma base de dados, ele produz baixo desempenho em consultas espaço-temporais. Para tratar este problema, esta dissertação propõe o particionamento da área geográfica de interesse em forma matricial, criando uma tesselação composta por Células Geográficas (CGs) justapostas. Eventos ocorridos em uma mesma CG são então armazenados de forma agrupada, para otimizar a sua recuperação. Esta estratégia foi utilizada para desenvolver um ***M**étodo para **I**ndexação de **E**ventos de **T**rânsito (MIDET)*. O MIDET adota um armazenamento misto, com arquivos compostos por blocos de eventos ocorridos em um mesmo CG, e uma base relacional para a indexação dos dados e a utilização de bitmaps. A motivação do MIDET é a indexação de eventos de trânsito históricos, ocorridos em uma determinada região geográfica, relativos a um período de tempo, com o objetivo de obter um bom desempenho para *consultas espaço-temporais* sobre tais registros. Como eventos de trânsito são reportados em grandes volumes de dados, um objetivo adicional é que o método tenha um baixo custo de *inserção*. A criação do modelo de armazenamento proposto consiste em 2 fases: (1) criação da tesselação e (2) geração de conjunto de eventos e índices bitmap na base de dados. Com isso, o método visa diminuir o tempo de busca e tornar o processamento de consultas mais eficiente. O método possui aplicação em diversos contextos. O estudo de caso adotado neste trabalho foi a base do aplicativo Waze[1] referente a eventos da cidade de Joinville-SC. Consultas espaço-temporais executadas com o MIDET apresentaram desempenho superior em comparação com o armazenamento na base de dados relacional clusterizada por CGs. Os testes realizados demostraram que o armazenamento misto e a utilização de bitmaps propostos pelo MIDET são vantajosos.

Palavras-chave: Índices Bitmap, Dados Espaço-temporais, Células Geográficas, Aplicativo Waze, Consultas analíticas, Eventos de Trânsito, Planejamento Urbano, Tesselação

[1]https://github.com/joinvalle/Joinville-Smart-Mobility

ABSTRACT

In the urban environment, data collected from traffic events can become elements of study to help plan cities and metropolises. Although much data has already been stored, the challenge is transforming this set of spatio-temporal data into mobility knowledge to assist in this planning. Traffic events, such as traffic jams and alerts, are continuously produced using apps like Waze. Due to the speed with which this data is reported, applications generally store events as individual records. Even though this storage model can guarantee a low insertion cost in a database, it produces low performance in spatial-temporal queries. To address this problem, this dissertation proposes a geographic partitioning of the area of interest in a matrix form, creating a tessellation composed of juxtaposed Geographic Cells (CGs). Events that occurred in the same CG are then stored in a grouped manner to optimize their recovery. This strategy constitutes the basis of a ***M**ethod for **I**ndexation of **T**raffic **E**vents (MIDET)*. MIDET adopts a mixed storage, with files composed of groups of events that occurred in the same CG and a relational database for indexing data based on bitmaps. MIDET's motivation is indexing historical traffic events, which occurred in a given geographic region, related to a period, with the goal of obtaining good performance for *spatial-temporal queries*. As traffic events are reported on large volumes of data, an additional objective is that the method should have a low cost of *insertion*. The method has application in several contexts. The adopted study case uses real data obtained from the application Waze [1], containing events from Joinville city. Spatial-temporal queries performed with MIDET shows superior performance compared to storage in the relational database clustered by CGs. The experiments demonstrates that the mixed storage and use of a bitmap-based indexing proposed by MIDET are advantageous compared to traditional spatial tree-based indexing adopted by relational databases.

Keywords: Bitmap index, Spatial-Temporal Data, Geographic Cells, Waze App, Analytical Query, Traffic Events, Urban Planning, Tessellation

[1]https://github.com/joinvalle/Joinville-Smart-Mobility

Lista de Figuras

Lista de Tabelas

LISTA DE ACRÔNIMOS

DINF	Departamento de Informática
PPGINF	Programa de Pós-Graduação em Informática
UFPR	Universidade Federal do Paraná
DW	Data Warehouse
GDW	Data Warehouse Geográfico
DWE	Data Warehouse Espacial
MBR	Retângulo Envolvente Mínimo
SGBD	Sistema Gerenciador de Banco de Dados
MIDET	Método de Indexação de Eventos de Trânsito
CGs	Células Geográficas

Sumário

1 INTRODUÇÃO

No ambiente urbano, os dados coletados a partir de eventos de trânsito podem ser elementos de estudo para fomentar o planejamento de cidades e metrópoles. Apesar de muitos dados já terem sido coletados, o desafio é transformar este conjunto de dados espaço-temporais em conhecimento de mobilidade para auxiliar nesse planejamento. Eventos de trânsito, como engarrafamentos, alertas e irregularidades são continuamente produzidos através de aplicativos como o Waze, tornando o conjunto de dados cada vez maior. Em razão da velocidade com que estes dados são reportados, estes aplicativos em geral armazenam eventos como registros individuais, cada um deles com as suas próprias informações espaço-temporais. Embora este modelo de armazenamento possa garantir um baixo custo de inserção em uma base de dados, ele possui um baixo desempenho em consultas espaço-temporais, pois elas precisam varrer todos os registros da base. Para evitar esta busca exaustiva, estruturas de índice podem ser utilizadas.

As principais estruturas adotadas são as Árvores R (PostgreSQL (2020), SQLite (2018)) e Árvores KD (Oracle (2019), ExtremeDB (2018)). Shahnawaz e Kannapiran (2019) destacam os principais problemas na indexação e gerenciamento de dados espaço-temporais utilizando estas árvores. Estas estruturas possuem um alto custo de manutenção do índice e um processamento de consultas espaço-temporais lento em alguns casos. Além disso, dependem de algoritmos de agrupamento de dados para um melhor desempenho em consultas. Por exemplo, Vu e Eldawy (2018) apresentam duas limitações de uma árvore R: (1) a árvore pode produzir nós folha com uma grade variedade de tamanho, resultando em uma carga de distribuição do índice desbalanceada e (2) dados podem pertencer a mais de um nó folha, criando sobreposição de dados. Já a estrutura de uma Árvore KD é sensível à ordem pela qual os pontos são inseridos [Gan et al. (2007)], o que pode resultar em uma árvore não otimizada para o conjunto de dados. Além disso, os pontos podem estar distribuídos de uma maneira que pode gerar desbalanceamento.

Esta dissertação propõe o particionamento da área geográfica de interesse, criando uma tesselação composta por Células Geográficas (CGs) justapostas, sobre uma área de interesse. As CGs justapostas eliminam a possibilidade de dados pertencerem a mais de uma CG, uma vez que não se intersectam. A área de interesse é inserida como parâmetro, o que evita que parte da área não seja representada. Eventos ocorridos em uma mesma CG são então armazenados de forma agrupada, para otimizar a sua recuperação. Esta estratégia foi utilizada para desenvolver um ***M**étodo para **I**ndexação de **E**ventos de **T**rânsito (MIDET)*. O MIDET adota um armazenamento misto, com arquivos compostos por blocos de eventos ocorridos em uma mesma CG e em um mesmo período, e uma base relacional para a indexação dos dados e a utilização de bitmaps. O agrupamento de eventos em blocos visa reduzir o custo de manutenção da indexação, em comparação com registros armazenados individualmente. Além disso, os eventos são distribuídos por toda a grade e os blocos possuem tamanho máximo fixo. Dessa forma, eles não geram conjuntos com uma grande variedade de tamanho. O uso de bitmaps visa diminuir o espaço de busca, filtrando apenas agrupamentos que tenham possibilidade de atender à consulta. O MIDET particiona fisicamente os registros em subconjuntos pela sua proximidade espaço-temporal, o que evita a indexação de cada elemento, diminuindo o tempo de criação e o tamanho da estrutura de indexação. Além disso, ele reduz o espaço de busca em consultas espaço-temporais que envolvem as operações de seleção, junção, projeção e agregação.

1.1 MOTIVAÇÃO

A motivação do MIDET é a indexação de eventos históricos de trânsito, ocorridos em uma determinada região geográfica, relativos a um período de tempo. Assim, o método tem como objetivo obter um bom desempenho para *consultas espaço-temporais* sobre tais registros. Como eventos de trânsito são reportados em grandes volumes de dados, um objetivo adicional é que o método tenha um baixo custo de *inserção*. Para atender estes dois objetivos, o MIDET definiu a seguinte hipótese.

Hipótese: É possível otimizar consultas de eventos de trânsito, agrupando os registros por proximidade espaço-temporal, baseado em uma tesselação sobre a área geográfica de interesse.

A estratégia de agrupamento é similar à representação em forma matricial (tesselação) utilizada para dados em sistemas de informação geográfica, na qual a área de interesse é caracterizada por uma matriz de células de tamanhos regulares, cada uma associada a um conjunto de valores que representam as características geográficas da região [Silva (2002)]. Para o MIDET, eventos ocorridos dentro dos limites de cada célula são armazenados em conjunto. Além disso, para tratar a questão temporal, eventos que ocorreram dentro da célula são ordenados cronologicamente e particionados por intervalos de tempo pré-determinados pelo usuário.

Para otimizar as consultas espaço-temporais, são utilizados índices bitmap. A criação do modelo de armazenamento proposto consiste em 2 fases: (1) criação da tesselação e (2) geração de conjunto de eventos e índices bitmap na base de dados. Com isso, o método visa diminuir o tempo de busca e tornar o processamento de consultas mais eficiente. O método possui aplicação em diversos contextos. O estudo de caso adotado neste trabalho foi a base do aplicativo Waze[2] referente a eventos da cidade de Joinville-SC.

1.2 CONTRIBUIÇÕES

As contribuições deste trabalho tem aplicação na área de planejamento urbano, principalmente na indexação espaço-temporal de eventos de trânsito. As principais contribuições são:

- Uma abordagem de agrupamento de registros pela sua proximidade espaço-temporal e pelo tipo de evento (engarrafamento, alerta e irregularidade).
- Um método para indexação de eventos de trânsito baseado no agrupamento de registros.
- Um modelo de armazenamento misto com arquivos compostos por agrupamento de registros ocorridos em uma mesma CG e em um mesmo período, bem como uma base relacional para a indexação dos dados baseada em bitmaps, que descarta a necessidade da avaliação registro a registro no processamento de consultas espaço-temporais.
- Estudos experimentais que validam o MIDET através de comparações com formas de armazenamento relacionadas utilizando consultas espaço-temporais.

1.3 ORGANIZAÇÃO DO DOCUMENTO

O restante da dissertação está dividido da seguinte forma: o Capítulo 2 apresenta fundamentos teóricos de indexação de árvore e bitmaps, pois são as estruturas, em geral, utilizadas para a indexação em trabalhos relacionados. O Capítulo 3 apresenta e compara

[2]https://github.com/joinvalle/Joinville-Smart-Mobility

abordagens relacionadas ao tema desta dissertação. O Capítulo 4 apresenta o MIDET, bem como suas estruturas de dados, armazenamento, detalhes de seu funcionamento e um exemplo de sua utilização. O Capítulo 5 descreve a implementação do MIDET e experimentos realizados. O Capítulo 6 conclui esta dissertação.

2 INDEXAÇÃO DE DADOS

Neste Capítulo são apresentados alguns conceitos importantes para a dissertação. Na Seção 2.1 são apresentadas estruturas de indexação que foram consideradas pertinentes para este trabalho pois, em geral, são as utilizadas para a indexação de dados espaço-temporais em sistemas gerenciadores de banco de dados e em trabalhos relacionados.

2.1 INDEXAÇÃO

Nesta Seção são apresentadas as estruturas de indexação consideradas como árvores B+, R e KD. Na Seção 2.1.1 diferentes estruturas de árvore são abordadas. Na Seção 2.1.2 índices baseados em bitmap são apresentados.

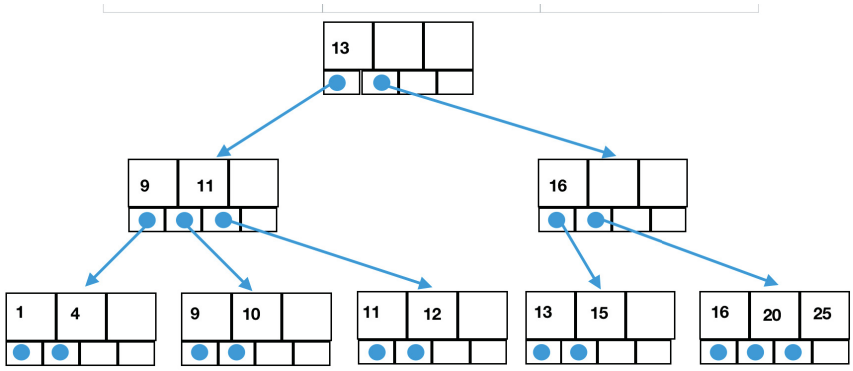
2.1.1 Árvores

Árvores B + são semelhantes a árvores binárias, porém podem possuir mais de 2 nós filhos por nó pai. Esta estrutura utiliza índices chave-valor. Os conteúdos de nós intermediários são apenas ponteiros para os próximos níveis e os nós folha armazenam os dados ordenados. A árvore B+ requer que todos os nós folha tenham a mesma distância da raiz, como ilustrado na Figura 2.1a. No exemplo, a busca dos 11 valores dos nós folha, envolverá o carregamento de três nós do disco (o bloco da raiz, um bloco de segundo nível e a folha).

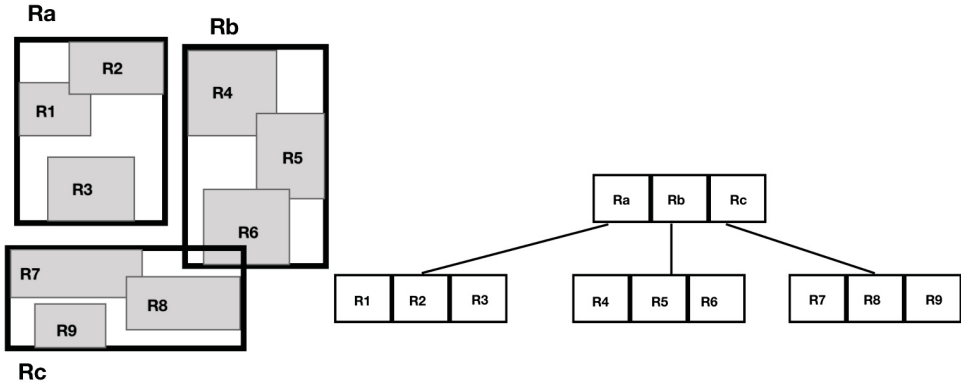
Árvores R são estruturas de dados baseadas nas árvores B+ [Manolopoulos et al. (2003)]. São utilizadas para a representação de um conjunto de objetos bidimensionais geométricos, utilizando Retângulo Delimitador Mínimo (MBR). Cada nó na estrutura corresponde aos MBRs que limitam seus filhos. Os nós folha possuem ponteiros para objetos de dados. Assim como a árvore B+, a árvore R é uma estrutura balanceada. Cada nó folha contém pares *(R,O)*, onde *R* corresponde ao MBR e *O* corresponde ao objeto espacial. Todos os outros nós possuem pares do tipo *(R,P)*, onde *R* é o MBR que contém os MBRs de seus filhos re *P* é o ponteiro para o nó filho. A Figura 2.1b ilustra o espaço geográfico à esquerda sendo representado pela árvore R à direita. Os retângulos de dados *R1-R9* são armazenados em nós folha e *Ra*,*Rb* e *Rc* são os MBRs de regiões que contém os objetos.

Segundo Papadias et al. (2001), há duas vantagens da utilização da árvore R: (1) a árvore modela a região em múltiplos MBRs e (2) permite buscar por MBRs similares. Segundo Manolopoulos et al. (2006), árvores R possuem 2 principais desvantagens: (1) a execução de uma busca de localização de um ponto pode levar a investigação de diversos caminhos da raíz à folha, o que pode resultar em deterioramento da performance, especialmente quando a sobreposição de MBRs é significante e (2) o grau de sobreposição pode aumentar significativamente mesmo com poucos retângulos, deteriorando a performance durante uma busca de intervalo, devido ao espaço vazio dentro deles. Já, Vu e Eldawy (2018) apresenta mais uma limitação a árvore pode produzir nós folha com uma grade variedade de tamanho, resultando em uma carga de distribuição do índice desbalanceada.

Árvores KD, ou árvores *k* dimensionais, são árvores para particionamento binário de espaço. Estas árvores organizam em suas estruturas um número *n* de pontos no plano com *k* dimensões. Estas estruturas são principalmente usadas para buscas de vizinhos mais próximos e para buscas por intervalos. A Figura 2.2 apresenta um exemplo de árvore KD com duas dimensões, com a região à esquerda representada na árvore à direita. Cada nó interno divide



(a) Árvore B+



(b) Árvore R

Figura 2.1: Árvores B+ e R

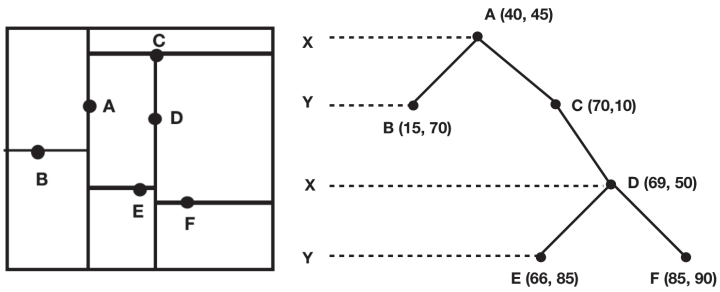


Figura 2.2: Árvore KD

o espaço de busca, assim cada divisão é representada por uma linha vertical para nós com discriminadores em x e horizontal para nós com discriminadores em y. O nó raiz divide o espaço em duas partes e cada filho seu subdivide o espaço em partes menores. Árvores KD são otimizadas para realizar buscas em paralelo e pesquisas de vizinhos mais próximos. Streisky e Borges (2019) apresentam 2 vantagens: (1) possibilita a busca e inserção de dados eficiente e (2) facilidade de armazenamento da estrutura. Segundo Gan et al. (2007), as árvores KD apresentam três desvantagens: (1) a estrutura da árvore é sensível à ordem pela qual os pontos são inseridos, (2) os pontos podem estar distribuídos em qualquer parte da árvore, o que pode gerar desbalanceamento e (3) operações de remoção de dados existentes são complexas e podem causar a reorganização da sub-árvore abaixo do dado removido.

Estruturas baseadas em árvore B+ oferecem uma vantagem em relação a outras estruturas clássicas utilizadas na literatura como hash [Kleppmann (2017)]. As árvores e suas variações dividem o banco de dados em blocos ou páginas fixas. Assim a leitura e a escrita é feita em

Atrbuto
A,B
C
C
B
A

(a) Tabela

A	B	C
1	1	0
0	0	1
0	0	1
0	1	0
1	0	0

(b) Índice bitmap simples

Figura 2.3: Índice bitmap simples

uma página por vez [Kleppmann (2017)]. Este esquema é apropriado ao projeto do hardware subjacente, pois os discos também são organizados em blocos de tamanho fixo.

2.1.2 Bitmaps

Segundo Wu e Buchmann (2008), a criação e manutenção de índices de bitmap simples geralmente apresentam custos menores do que a utilização de uma árvore, além de reduzir o espaço de pesquisa antes mesmo de acessar os dados.

Árvores B+ e suas variantes possuem vantagem em relação a bitmaps quando o conteúdo é constantemente atualizado. Porém, para banco de dados históricos, bitmaps possuem vantagem pois não recebem atualizações frequentemente. Estes índices podem reduzir o espaço de busca antes mesmo de acessar os dados e também melhoram a velocidade do processamento de consultas [Keawpibal et al. (2019)].

A ideia por trás de indexação com bitmap simples é utilizar um vetor de bits (0 ou 1) para representar se um atributo na tupla é igual a um valor específico ou não. A posição do bit no vetor denota a posição de uma tupla na tabela T. A Figura 2.3 ilustra um exemplo de bitmap simples. A tabela T tem o domínio *A, B, C* (Figura 2.3a). É criado um vetor de bits para representar o elemento da tabela no vetor correspondente. É utilizado o bit 1 na posição equivalente à tabela. Na tabela (Figura 2.3a) T[0]= A,B e T[4]=A, e no vetor *A*(Figura 2.3b) A[0]=1 e A[4]=1, e no vetor *B* B[0]=1.

2.1.2.1 Consultas em Bitmap

As consultas com índices bitmaps são processadas através de operações *Bitwise*. O tempo de processamento das consultas depende da quantidade de bitmaps acessados e das operações realizadas sobre eles. As operações no bitmap podem ser processadas mais eficientemente do que outros índices por serem *Bitwise*. As operações principais *Bitwise* são negação (Not), conjunção (And), disjunção (Or) e disjunção binária exclusiva (Xor). Para esta dissertação, as operações de conjunção e disjunção foram utilizadas. A Figura 2.4 ilustra a operação de conjunção. É retornado o bit 1 sempre que ambos os operandos sejam '1'. Já a Figura 2.5 ilustra a operação de disjunção. É retornado o bit 1 sempre que algum dos operandos seja '1' ou ambos.

Assim, nesta dissertação são considerados bitmaps por apresentarem vantagens em processamento nas consultas, conforme apresentado.

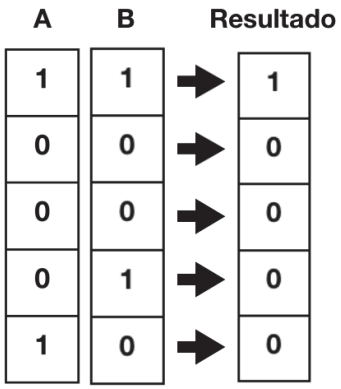


Figura 2.4: Exemplo Conjunção de bitmap

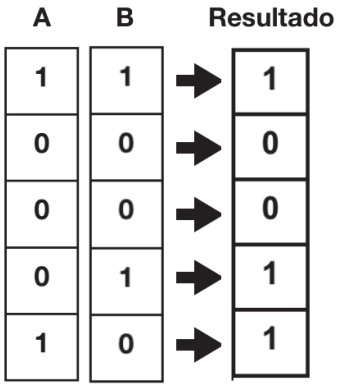


Figura 2.5: Exemplo Disjunção de bitmap

2.2 CONSIDERAÇÕES

Neste Capítulo, foram apresentados conceitos importantes para a dissertação sobre estruturas de indexação, em geral, utilizadas para a indexação de dados espaço-temporais em sistemas gerenciadores de banco de dados. Estas estruturas estão presentes em trabalhos relacionados que serão apresentados a seguir.

3 TRABALHOS RELACIONADOS

Neste Capítulo são apresentados trabalhos relacionados, que estão divididos em duas categorias. Os focados em utilização de indexação por árvore (Seção 3.1) e os baseados em bitmaps (Seção 3.2).

3.1 INDEXAÇÃO

Nesta Seção são apresentados trabalhos relacionados que utilizam indexação com árvores. Nas Seções 3.1.1 a 3.1.4 os modelos com árvore são detalhados. Na Seção 3.1.5 é apresentada a tabela comparativa e uma descrição sucinta do modelo proposto.

3.1.1 Árvore STIG

O modelo de indexação baseado em GPUs, chamado de árvore STIG [Doraiswamy et al. (2016)], corresponde à uma árvore KD na qual as folhas são compostas por partições, que são divisões de uma unidade de disco rígido e que podem ser processadas em paralelo. A árvore STIG possui `k` dimensões, sendo `k=2*s+m` sendo `s` os atributos espaciais e `m` os atributos temporais. A Figura 3.1 ilustra a estrutura do índice STIG. Os nós internos da árvore armazenam a mediana dos pontos de coordenada por ele divididos e, também, ponteiros para seus filhos da esquerda e direita. Os filhos com valores menores que a sua mediana ficam para a esquerda e os maiores para a direita. O nó folha aponta para o bloco folha que armazena dados correspondentes a uma coleção de registros. Esses dados consistem em valores dos atributos nos quais o índice foi criado junto com um ponteiro para o local do registro real. Cada bloco folha satisfaz todas as restrições especificadas pelo caminho da raiz à folha correspondente. O tamanho do bloco folha é um parâmetro especificado ao criar o índice. Um nó folha armazena um ponteiro para um bloco folha e uma caixa k-dimensional que limita todos os registros nesse bloco. Uma vantagem da utilização dos blocos é a diminuição da altura da árvore por um fator igual ao tamanho dos blocos folha.

O objetivo é acelerar o processamento de consultas OLAP em dados históricos e permitir o processamento tanto de consultas sobre dados na memória principal quanto de dados no disco utilizando GPU. Os blocos folha facilitam esta estratégia ao possibilitar a busca em paralelo pelo GPU. O foco do trabalho é a representação de informações geográficas para análises estatísticas. Com o uso da árvore STIG, o número de testes para definir se o objeto faz parte da célula geográfica diminui, segundo os autores, pois é feita a filtragem simultânea de múltiplas dimensões diminuindo o espaço de busca. Já o aspecto temporal é apenas um atributo. Este modelo não realiza atualização no índice.

3.1.2 Índice TQ

O modelo de indexação criado e implementado por Imawan et al. (2015) tem a proposta de prever congestionamentos de fluxo de veículos utilizando o índice TQ, que leva em consideração as características espaço-temporais de dados de trânsito ainda não tratados *(raw data)*. O modelo é otimizado para pesquisas analíticas com linha de tempo. Além disto, é implementada uma plataforma para a visualização dos dados. Para encontrar padrões do tráfego diário e estruturas da rede rodoviária, o índice TQ mantém dois componentes principais: um índice de localização e um índice de tempo. A Figura 3.2 ilustra a estrutura do índice TQ. O índice de localização

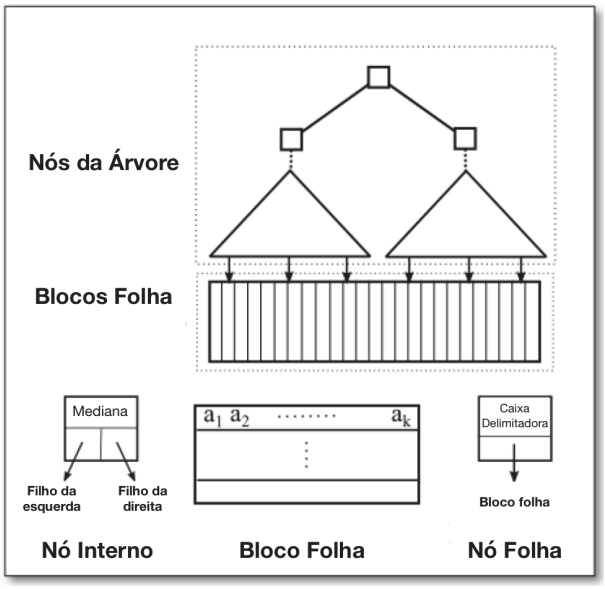


Figura 3.1: Estrutura do índice STIG, visto em Doraiswamy et al. (2016)

representa a rede de ruas e conecta a visão da linha do tempo dos engarrafamentos das ruas com os segmentos das ruas. Este índice é baseado em hash e tem como chave o identificador do segmento da rua e a tripla (identificador da próximo segmento de rua, identificador do segmento de rua anterior e um conjunto de ponteiros para linhas do índice temporal). Este índice está exemplificado na parte superior central da Figura 3.2. O índice de localização acelera as buscas com linha de tempo pois os usuários devem especificar a conexão de ruas relevantes para a busca. O índice temporal mantém uma visão de uma linha do tempo de eventos de congestionamento e a sua estrutura está exemplificada na parte inferior central da Figura 3.2. Os dados ainda não processados são extraídos dos segmentos de rua e indexados por tempo e localização. Com a junção da localização e o tempo, é possível criar a visão da linha do tempo, onde os eventos de congestionamento (começo e fim) podem ser identificados no segmento de rua em qual ocorreram e seu horário, como mostrado à direita na Figura 3.2.

Além da definição do índice é demonstrado como criá-lo eficientemente [Imawan et al. (2015)]. Existem dois algoritmos definidos para esta criação. O primeiro consiste em detectar eventos de congestionamento de trânsito, inserir em um índice TQ e atualizar as dependências dos eventos. Como primeiro passo, é verificado se existe congestionamento em cada trecho de rua, observando o valor da velocidade em um arquivo de log de trânsito. Um trecho de rua que contém a velocidade abaixo do limiar indica o status de congestionamento e a busca é focada na mudança desse status do trânsito, ou seja, o começo ou o término deste.

Na segunda fase é inserido um conjunto de eventos de congestionamento em um índice TQ. O índice de localização armazena a informação da rede de estradas e o índice de tempo mantém a visão da linha do tempo dos eventos. São utilizados valores do índice hash do identificador do segmento da rua. Se nenhum valor for recuperado, isso significa que esse segmento de rua não armazena uma visão da linha do tempo. Assim, é criado um novo nó hash contendo a linha do tempo. Por fim, é calculada a dependência de cada congestionamento para manter as informações sobre os status de congestionamento atualizadas. Se um segmento da rua mudar para o status de congestionado, então a dependência é atualizada ao verificar o status dos segmentos de rua relacionados.

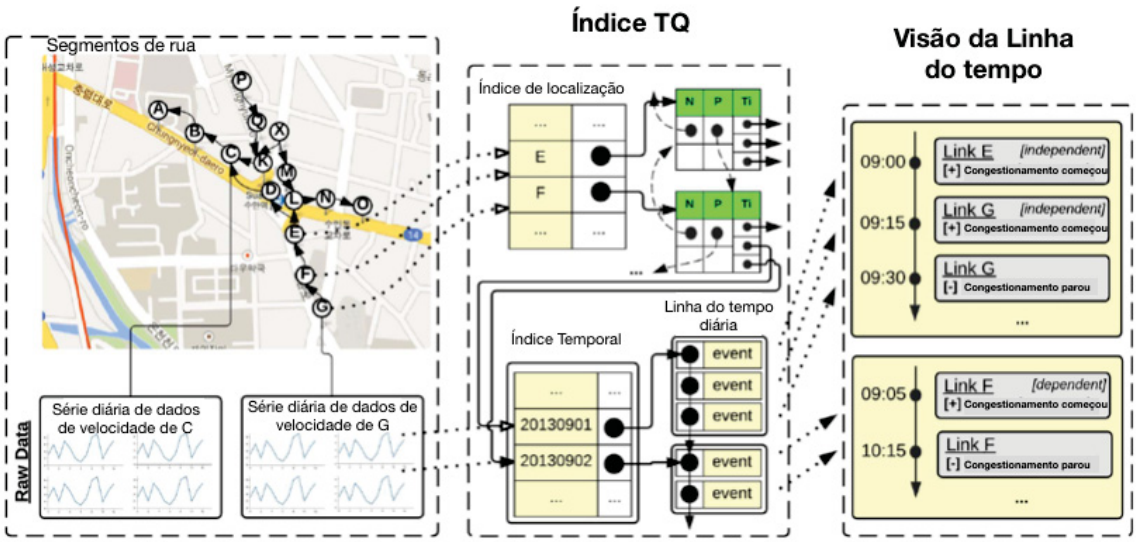


Figura 3.2: Estrutura do índice TQ, visto em Imawan et al. (2015)

3.1.3 *Smart City*

O modelo de armazenamento proposto por Krylovskiy et al. (2015) e implementado para um caso real em Brundu et al. (2016), tem como intuito aplicar uma arquitetura de microsserviço para projetar uma plataforma de larga escala para cidades inteligentes (Smart City) e para a internet das coisas (IoT). Como o MIDET, o trabalho utiliza dados reais espaço-temporais. Segundo os autores, os experimentos sugerem benefícios provenientes da adoção de microsserviços em comparação com a arquitetura orientada a serviços mais genérica. Também são destacados alguns dos desafios que esta arquitetura introduz. Por exemplo, implementá-la com sucesso na prática pode ser um desafio, pois a decomposição de sistemas distribuídos em componentes independentes na forma de microsserviços, traz uma sobrecarga operacional. No momento da escrita do artigo, os autores consideravam consistência eventual e outros desafios devido aos *trade-off* da decomposição do serviço para se obter benefícios. Como trabalhos futuros os autores mencionam o interesse em realizar um estudo mais completo para avaliação da plataforma à medida que mais plataformas de serviços e aplicações forem desenvolvidas.

A Figura 3.3 ilustra a arquitetura do sistema. No nível conceitual, são incluídas tecnologias de sensores heterogêneas e modelos de informação dos distritos, integrados na plataforma de serviço usada pela aplicações da Smart City. A plataforma de serviços engloba uma junção de serviços de IoT e dados de tecnologias de sensores heterogêneos. Também um número de serviços de plataforma para Serviços de Smart City. Uma variedade de aplicações podem utilizar a plataforma como aplicativos da Web, de desktop e móveis tanto para a administração municipal quanto para os cidadãos. Além do que, a plataforma pode ser usada por sistemas de terceiros e aplicativos, incluindo os clientes da Web semântica, para acessar dados de sistemas de sensores integrados e modelos de informação. O *middleware* é um elemento crucial desta plataforma de IoT pois ele integra dispositivos heterogêneos na plataforma. As tarefas do *middleware* incluem fornecer abstrações de modelagem de dispositivos de IoT e tecnologias de sensores, habilitar a pesquisa e a descoberta desses dispositivos e seus recursos por aplicativos e serviços e fornecer protocolos unificados para acesso a dados históricos de sensores e quase em tempo real.

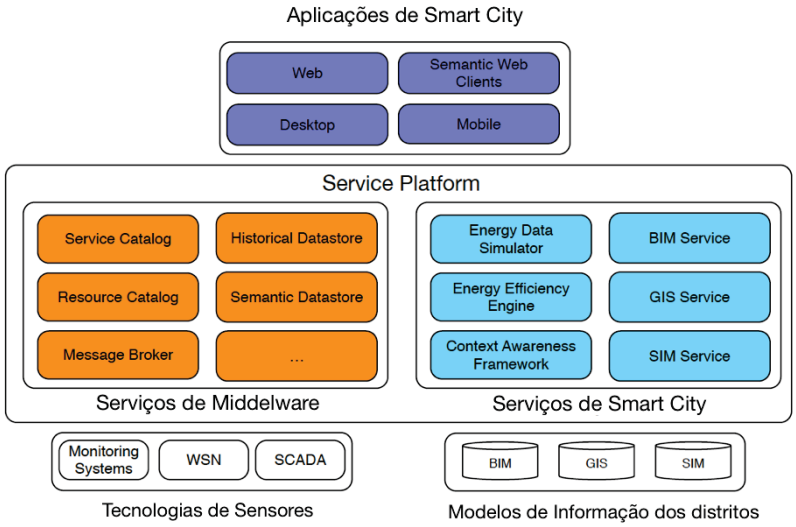


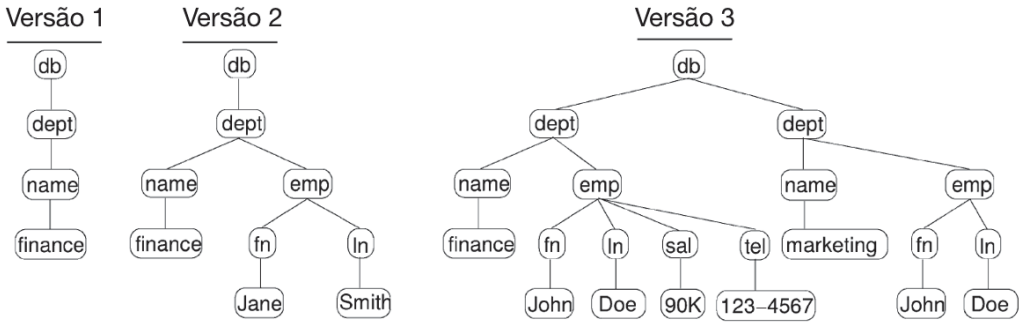
Figura 3.3: Arquitetura da plataforma Smart City, visto em Krylovskiy et al. (2015)

3.1.4 Árvore Buneman

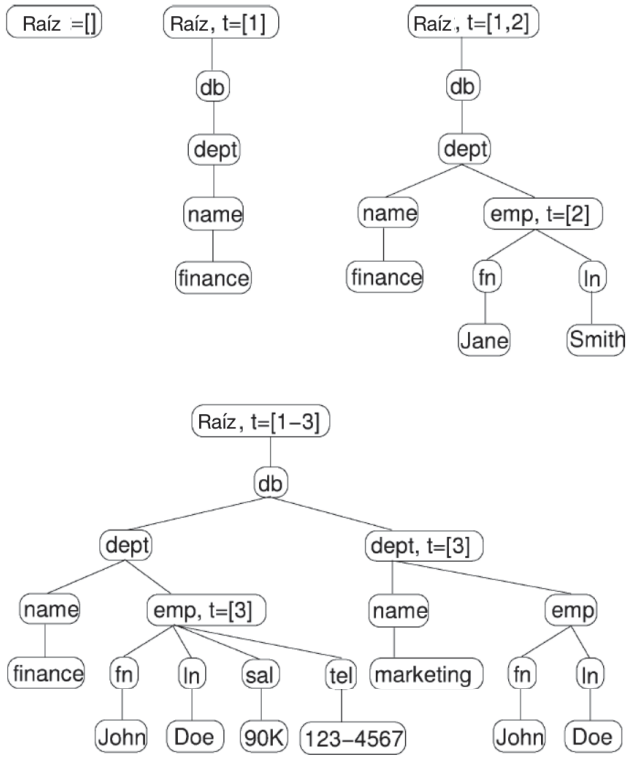
O modelo de armazenamento de Buneman et al. (2004) para XML, chamado de Árvore Buneman, é proposto para realizar o versionamento da base de dados, trabalhando com dados focados em objeto e com o aspecto temporal. Esta estrutura é otimizada para buscas históricas focadas em objeto. A técnica de arquivamento é eficiente em seu uso do espaço e preserva a continuidade dos elementos através de versões do banco de dados. Um conjunto de experimentos também demonstra que o arquivo não gera sobrecarga de espaço significativo quando comparado com outras abordagens. Outra propriedade da abordagem, segundo os autores, é que é usado o formato XML para representar dados hierárquicos e o arquivo resultante também está em XML. Consequentemente, ferramentas para XML podem ser aplicadas diretamente no arquivo. Os autores aplicam um compactador XML, e os experimentos mostram que o volume da árvore Bunemam compactada é menor que o arquivo compactado utilizando outras estratégias. Também é mostrado como é possível estender a ferramenta para um arquivador de memória externa para prover maior escalabilidade. São descritas ainda estruturas de índice que podem melhorar a eficiência de algumas consultas temporais. Esta árvore foi implementada também por Besenbacher et al. (2004), porém pouco conteúdo se tem sobre esta implementação.

Na Figura 3.4a é ilustrada uma sequência de versões de um banco de dados de uma empresa contendo informações sobre funcionários em cada departamento. Todo departamento pode ser identificado por seu nome de departamento, sendo o nome a chave para os departamentos. Dentro de cada departamento, cada funcionário pode ser identificado exclusivamente pelo seu nome e sobrenome. É possível notar que é permitido que dois funcionários tenham o mesmo nome e sobrenome se pertencerem a departamentos distintos (por exemplo, John Doe da versão 3). Cada empregado também tem no máximo um valor salarial e um ou mais números de telefone. As versões mudam quando há inserção ou remoção de elementos na árvore.

A Figura 3.4b ilustra a anotação temporal da árvore Buneman. A raiz começa nula e com o tempo nulo. Ao inserir a versão 1 da Figura 3.4a a árvore aponta para o primeira subárvore e anota o tempo da inserção. Após, com a segunda versão, a árvore tem o período atual, ou seja, do t=1 a t=2. A nova subárvore inserida no t=2 tem a anotação na subárvore o que indica que essa subárvore só é válida a partir de t=2. Ao inserir a versão 3 a raiz vale do t=1 a t=3, e os nós que não contém anotação em algum nó pai valem para todo o período da raiz.



(a) Sequência de versões



(b) Árvore com anotação de versões

Figura 3.4: Sequência de versões e árvore com anotação de versões Buneman et al. (2004)

Uma questão trazida por Buneman et al. (2004) é como a técnica apresentada pode ser estendida para arquivar dados sob circunstâncias em que a estrutura-chave muda, pois os esquemas tendem a mudar ao longo do tempo.

3.1.5 Análise Comparativa

Os critérios de avaliação definidos para efeito de comparação entre os trabalhos de indexação consistem em: suporte para dados temporais e geográficos, e se o modelo suporta dados relacionais. Por fim, se o modelo foi criado com o propósito de otimizar consultas com dados de mobilidade. A Tabela 3.1 apresenta a comparação entre os trabalhos relacionados de indexação por árvore.

Foram analisados 4 modelos de indexação sendo que 3 destes Imawan et al. (2015) (Índice TQ), Krylovskiy et al. (2015) (Smart City) e Doraiswamy et al. (2016) (Árvore STIG) são

Estrutura	Temporal	Geográfico	Dados relacionais	Dados de mobilidade
Árvore STIG	X	X	X	X
Índice TQ	X	X		X
Smart City	X	X		X
Árvore Buneman	X			
MIDET	X	X	X	X

Tabela 3.1: Comparação de trabalhos relacionados com indexação

espaço-temporais. O trabalho de Buneman et al. (2004) (Árvore Buneman) é somente temporal. Apenas a Árvore STIG (Doraiswamy et al. (2016)) lida com dados relacionais. Os trabalhos Imawan et al. (2015) (Índice TQ) e Doraiswamy et al. (2016) (Árvore STIG) tem como foco a indexação de dados de mobilidade, e o trabalho de Krylovskiy et al. (2015) (Smart City) visa a integração de dados de uma cidade, como por exemplo, dados de trânsito integrados a ocorrências policiais e em hospitais.

O MIDET tem como foco a indexação de eventos de trânsito, portanto, são espaço-temporais como os trabalhos de Imawan et al. (2015) (Índice TQ), Krylovskiy et al. (2015) (Smart City) e Doraiswamy et al. (2016) (Árvore STIG). O trabalho de Krylovskiy et al. (2015) (Smart City) assim como o MIDET utiliza dados reais. O trabalho de Doraiswamy et al. (2016) (Árvore STIG) orientou a criação de uma grade geográfica (tesselação), bem como a criação de blocos de registros. O trabalho de Buneman et al. (2004) (Árvore Buneman) inspirou a criação de bitmaps por tipo de evento e para cada período por ser armazenado em versões. As consultas realizadas pelo MIDET são consultas espaço-temporais que envolvem as operações de seleção, junção, projeção e agregação.

Algumas desvantagens nos trabalhos relacionados conduziram à concepção do MIDET. Como por exemplo, o trabalho de Imawan et al. (2015) necessita que sejam especificadas todas as conexões de rua para realizar uma busca. O trabalho de Doraiswamy et al. (2016) (árvore STIG) é o mais semelhante ao MIDET. Porém, a árvore STIG é baseada em uma árvore KD. Esta árvore apresenta desvantagens, como (1) a estrutura da árvore é sensível à ordem pela qual os pontos são inseridos e (2) os pontos estão distribuídos por qualquer parte da árvore, o que pode gerar desbalanceamento [Gan et al. (2007)]. Além disso, a árvore STIG para ser usada em consultas que utilizam diferentes tipos de evento. Porém cada uma das árvores teria divisões espaço-temporais diferentes, tornando impossível combinar a árvore como indexação. Já os trabalhos de Krylovskiy et al. (2015) e Buneman et al. (2004) possuem intuitos diferentes do MIDET. O trabalho de Krylovskiy et al. (2015) integra com diferentes modelos de dados mas não é focado à indexação dos dados relacionais. Já o trabalho de Buneman et al. (2004) tem como objetivo o versionamento de um banco de dados.

Além disso, diferente dos trabalhos apresentados acima, bitmaps também serão utilizados visando diminuir o espaço de busca em consultas espaço-temporais. A seguir, os trabalhos de bitmap serão apresentados.

3.2 BITMAPS

A seguir são apresentados trabalhos de indexação com bitmap. A Seção 3.2.6 apresenta a tabela comparativa entre os trabalhos de bitmap e o MIDET.

3.2.1 Bitmap duplo

O índice de bitmap duplo usa menos espaço do que os índices de bitmap previamente existentes, mantendo as mesmas melhorias na velocidade de processamento de consultas [Wattanakitrungroj e Vanichayobon (2006)]. Este índice representa cada valor de atributo usando apenas dois vetores de bitmap, com cada vetor representando valores de atributos. A Figura 3.6 ilustra este índice de bitmap. Como o domínio da tabela da Figura 3.5 é 15, são necessários 6 vetores de bitmap vBit D (dual) `D0,D1,D2,D3,D4,D5` para a representação dos valores. A equação é usada para a codificação:

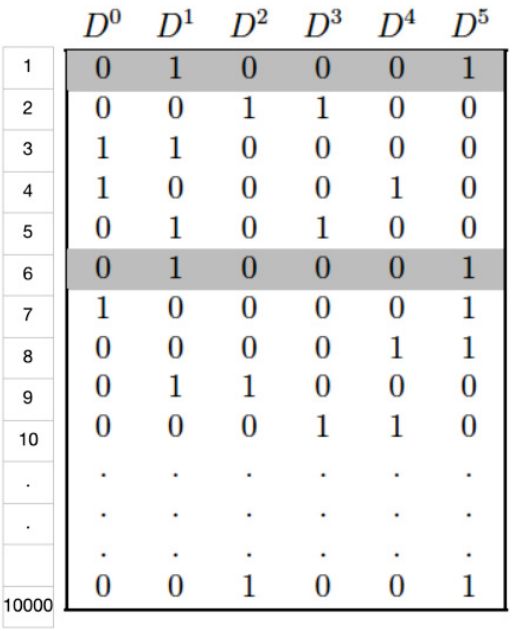
	A
1	3
2	9
3	14
4	8
5	10
6	3
7	4
8	0
9	12
10	5
.	.
.	.
.	.
10,000	2

Figura 3.5: Tabela A

$$D^j = \begin{cases} 1 & \text{se j=r e j=s} \\ 0 & \text{senão} \end{cases}$$

Hic=$n(n-1)/2$, r=$\sqrt{2(Hic - v) + 0.25} + 0.5$, $s = r - 1 - [(v - ((n - r)(n - r - 1)/2))modr]$

Sendo `v` o valor a ser representado, `j` a coluna e `n` o número total de vetores bitmap usados. Para representar o valor `3` em 1 os bits de Vbit[D1] e Vbit[D5] recebem 1 e os outros recebem 0. Apesar de outros modelos de bitmap resolverem o problema da esparsidade encontrado no bitmap simples, como o bitmap codificado que produz um número de vetores menor quando comparado ao bitmap simples, o tempo de processamento da busca é maior que de um bitmap duplo [Keawpibal et al. (2019)]. O bitmap duplo proposto por Wattanakitrungroj e Vanichayobon (2006) demonstrou melhorar o desempenho de consulta para uma consulta de igualdade com um *trade-off* entre espaço necessário e tempo consumido. Porém, consultas de intervalo ainda não eram otimizadas. Em Keawpibal et al. (2019) o algoritmo Dual-simRQ é proposto para melhorar o processamento dessas consultas. O dual-simRQ elimina a necessidade de varredura de vetores de bitmap e execução de operações booleanas desnecessárias. Como trabalhos futuros em Keawpibal et al. (2019) é proposto investigar um novo algoritmo de codificação para índice bitmap para otimizar o desempenho de todos os tipos de consultas seletivas em termos de compensação de espaço e tempo.



	D^0	D^1	D^2	D^3	D^4	D^5
1	0	1	0	0	0	1
2	0	0	1	1	0	0
3	1	1	0	0	0	0
4	1	0	0	0	1	0
5	0	1	0	1	0	0
6	0	1	0	0	0	1
7	1	0	0	0	0	1
8	0	0	0	0	1	1
9	0	1	1	0	0	0
10	0	0	0	1	1	0
.	.	.	.	.	.	.
.	.	.	.	.	.	.
	.	.	.	.	.	.
10000	0	0	1	0	0	1

Figura 3.6: Índice bitmap duplo, visto em Keawpibal et al. (2016)

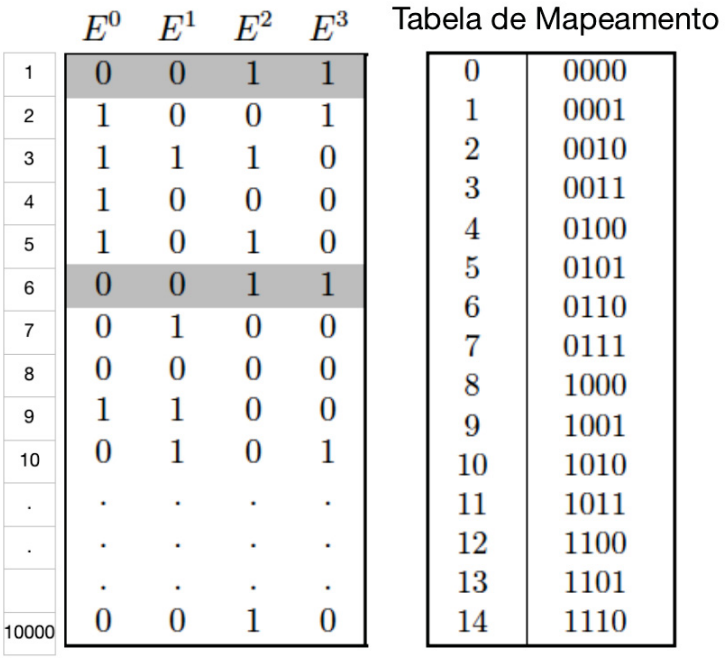
3.2.2 *Encoded Bitmap Indexing*

Com o aumento da cardinalidade de atributos a serem representados por um bitmap, a utilização do espaço degrada. Além disso, para buscas com intervalos grandes, o número de vetores de bit a serem processados aumenta. Neste caso, índices com bitmap simples podem ter um desempenho mais baixo que árvores B [Wu e Buchmann (2008)].

Com o intuito de reduzir a cardinalidade de vetores de bitmap simples, o bitmap codificado é proposto em Wu e Buchmann (2008). O bitmap codificado tem como princípio a codificação do domínio do atributo. Na Figura 3.7 este índice é ilustrado, assim como a tabela de mapeamento. São utilizados 4 vetores de bit vBit E (encoded) para representar o domínio A da Figura 3.5, onde a correspondência é representada na tabela de mapeamento. Para representar o atributo *3* na posição 1, é adicionado o bit 1 nos vetores de bitmap , Vbit[E2] e Vbit[E3], o código correspondente, no caso, *0011*. É possível observar que para representar o atributo *3* na posição 6 é usada a mesma codificação. Em Wu e Buchmann (2008) é proposto este método com o objetivo de reduzir o espaço necessário para um bitmap simples e o ganho de desempenho da busca. Porém, o custo de achar uma função de codificação interativa bem definida, que lide com o aumento do domínio não é trabalhado. Entretanto, Wu e Buchmann (2008) afirma que com definições customizadas das codificações, os índices de bitmap codificados são adequados e capazes de indexar dados OLAP. Algumas das aplicações para este tipo de bitmap são a codificação de hierarquia das dimensões, codificação com a preservação de ordem para atributos numéricos e índices de bitmap codificados baseados em intervalos. Wu e Buchmann (2008) conclui que, o bitmap codificado herda as facilidades e custo baixo do bitmap original, e ao mesmo tempo resolve a esparsidade.

3.2.3 *Enhanced Encoded Bitmap*

O trabalho de Keawpibal et al. (2012) apresentou o algoritmo E-EBI para otimizar buscas de igualdade em bitmap codificado. Segundo os autores, o algoritmo torna este modelo de



	E^0	E^1	E^2	E^3
1	0	0	1	1
2	1	0	0	1
3	1	1	1	0
4	1	0	0	0
5	1	0	1	0
6	0	0	1	1
7	0	1	0	0
8	0	0	0	0
9	1	1	0	0
10	0	1	0	1
.	.	.	.	.
.	.	.	.	.
	.	.	.	.
10000	0	0	1	0

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110

Figura 3.7: Índice bitmap codificado Keawpibal et al. (2016)

bitmap mais otimizado em termos de *trade-off* espaço tempo. Ele usa o menor número de vetores bitmap e menor tempo de consulta usando operações booleanas de baixo custo para responder a consulta de igualdade. O estudo comparativo mostra que o método proposto é melhor do ponto de vista do *trade-off* espaço-temporal que outras técnicas de indexação de bitmap existentes para uma consulta de igualdade. Como trabalhos futuros, uma representação de codificação bem definida para cada valor distinto de um atributo indexado é sugerida para otimizar outros tipos de desempenho de consultas. Assim, os autores planejam aplicar técnicas de mineração de dados. Essas técnicas, segundo os autores, podem ser usadas para agrupar atributos de valores que são frequentemente consultados conjuntamente. Sendo possível então usar esses agrupamentos para criar um esquema de codificação otimizado para reduzir o número de vetores de bitmap.

3.2.4 *Roaring Bitmap*

O trabalho de Lemire et al. (2018) apresenta um bitmap comprimido, o que diminui o espaço de representação. A maioria das alternativas ao Roaring faz parte de um conjunto de bitmaps compactados que são bitmaps codificados [Lemire et al. (2018)].

O *roaring bitmaps* divide todos os números inteiros em sub-conjuntos de 2^{16} números inteiros (`65535`). Esses sub-conjuntos são chamados de *containers*. O Roaring possui compressão híbrida pois armazena os dados de seus *containers* em 3 formatos diferentes, com base em heurísticas. Estes 3 formatos são compostos por *Array Container*, *Run Container* e *Bitmap Container*.

O *Array Container* armazena os dados como uma matriz de números inteiros, sem bitmaps. No entanto, ele não armazena prefixos, economizando espaço [Lemire et al. (2018)]. As pesquisas são feitas através da busca binária. O *Array Container* cresce dinamicamente com base nas heurísticas. Por exemplo, se o número de elementos no *Array Container* exceder `4096`, ele será convertido em *Bitmap Container*.

O *Run Container* é composto de uma matriz compactada, com pares de números inteiros de `16 bits`. O primeiro valor de cada par representa um valor inicial, enquanto o segundo valor é o comprimento de uma execução. Por exemplo, os valores `11, 12, 13, 14, 15` são armazenados como, o par `11, 4`, em que `4` significa que além do número `11`, existem 4 valores contíguos a seguir.

O *Bitmap Container* é armazenado como `1024 * palavras de 64 bits`, sem compactação no bitmap. Seu tamanho é fixo na alocação e não cresce dinamicamente. O *Bitmap Container* usa instruções de 64 bits no processador, obtendo um processamento eficiente [Lemire et al. (2018)].

A implementação foi realizada nas linguagens Go, C, Java e Swift [3].

3.2.5 Índice I-DWE, Índice SB e HSB

O trabalho de Siqueira et al. (2008) propõem uma estrutura de indexação, chamada I-DWE, para Data Warehouse espacial (DWE), que trata hierarquias predefinidas. É uma adaptação do índice bitmap. É proposto que o sbitvector (*Spatial Bitvector*) tenha três campos: um valor de chave da dimensão geográfica primitiva, um ponteiro para um vetor de bits e um retângulo envolvente mínimo (MBR). O trabalho posterior de Siqueira et al. (2009), propõem um índice eficiente para DW espacial que suporta hierarquias de atributos espaciais pré definidas e lida com a multidimensionalidade, chamado índice SB. O índice permite consultas multidimensionais com predicado espacial. Um índice SB representado na Figura 3.9 é uma matriz do tipo sbitvector, cuja i-ésima entrada aponta para o i-ésimo vetor de bits do índice Bitmap de junção estrela criado sobre a chave primária da tabela de dimensão espacial. Além disso, o índice SB é mantido persistentemente no disco juntamente com o índice Bitmap da junção estrela. Uma hierarquia espacial pré-definida no DWE determina a existência de um índice SB pelo nível da granularidade da hierarquia espacial. Para ilustrar a estrutura de dados do índice SB é utilizado o conjunto de dados mostrados na Figura 3.8. Cidade é uma tabela de dimensões espaciais contendo atributo espacial city_geo e a chave primária city_pk. A tabela de dimensão Fornecedor é associada a tabela Cidade e a tabela de fato pedidos. Por exemplo, city_pk = 1 é associado a s_suppkey = 3 e lo_suppkey = 3. O índice SB [0] mantém o valor da chave `1` e o MBR do objeto espacial identificado por city_pk = 1, como mostrado na Figura 3.9. Além disso, B é o índice Bitmap da junção estrela definido sobre o atributo city_pk da tabela de dimensões Cidade para indicar o conjunto de linhas na tabela de fatos a ser unida a um determinado valor de city_pk. O vetor de bit apontado por B[0] especifica as tuplas na tabela de fato onde city_pk = 1. Para acessar o vetor de bits relacionado ao SB-index[0] é utilizado o índice 0 para ler B, ou seja, B [0].

Em Siqueira et al. (2012) os autores propõem mais um índice, chamado índice HSB, onde se possa realizar a escolha entre estes dois índices. O HSB é um índice espacial baseado em árvore com cada nó folha apontando para um vetor de bits de um índice Bitmap de junção estrela.

A Figura 3.10 mostra o índice HSB para o conjunto de dados mostrado na Figura 3.8. Os objetos espaciais da tabela de dimensões City (3.10a) são representados por seus MBRs na Figura 3.10b, que compõem a estrutura de dados do índice espacial e executam algoritmo de clusterização como na Figura3.10c. O índice HSB também contém um ponteiro para um vetor de bits em cada entrada de nó folha na Figura 3.10d. Portanto, um objeto espacial é identificado por um valor de chave, representado por um MBR e também está associado a um conjunto de tuplas através de um vetor de bits. Por exemplo, o polígono identificado pelo valor da chave `1` na Figura 3.10a é aproximado de `R1` na Figura 3.10b e agrupado pela região `M1` no índice espacial na Figura 3.10c. No índice HSB as regiões espaciais resultantes ao englobar MBRs, ou seja, as

[3] https://github.com/RoaringBitmap

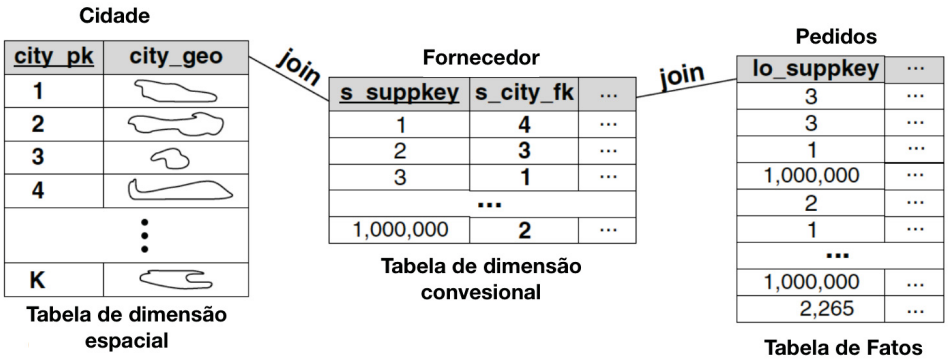


Figura 3.8: A tabela de dimensões espaciais Cidade, a tabela de dimensões Fornecedor e a tabela de fatos de pedidos, visto em Siqueira et al. (2012)

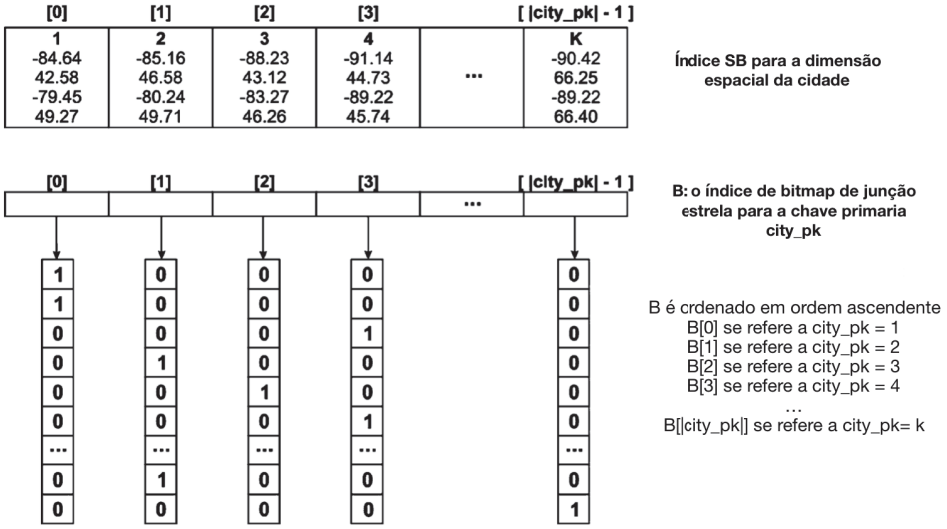


Figura 3.9: Índice SB, visto em Siqueira et al. (2012)

regiões M1 e M2 são as primeiras do índice espacial. Cada região generalizada aponta para os objetos espaciais que pertencem a ela. Os objetos espaciais pertencem a MBR e são associados a um vetor de bit. Como o objeto espacial `1` que possui o MBR `R1` e é associado a um vetor de bit 3.10d, cujos primeiro e segundo bits com valor 1 indicam que a primeira e a segunda tupla da tabela de fatos fazem referência ao polígono com valor de chave `1`.

Consultas de baixa seletividade beneficiam-se com o índice HSB. Já o índice SB oferece melhor desempenho para consultas seletivas e de igualdade. As vantagens introduzidas pelo índice SB e pelo índice HSB foram analisadas através de testes de desempenho destinados a investigar a sua eficiência, características e aplicabilidade. Os resultados, segundo os autores, mostraram que ambos os índices propostos foram muito eficientes em comparação com os recursos típicos do SGBDs para processar operações espaciais. Em comparações entre o índice SB, o índice HSB, o cálculo de junção estrela e o uso de visões materializadas demonstrou que o ganho de desempenho dos índices variou de 95% a 99% sobre o cálculo de união em estrela, e de pelo menos 68% para visões materializadas. Estes resultados foram obtidos para consultas com diferentes complexidades, ou seja, os autores investigaram o processamento com uma única janela de consulta espacial, o processamento com duas janelas de consultas espaciais e o processamento de operações ininterruptas espaciais.

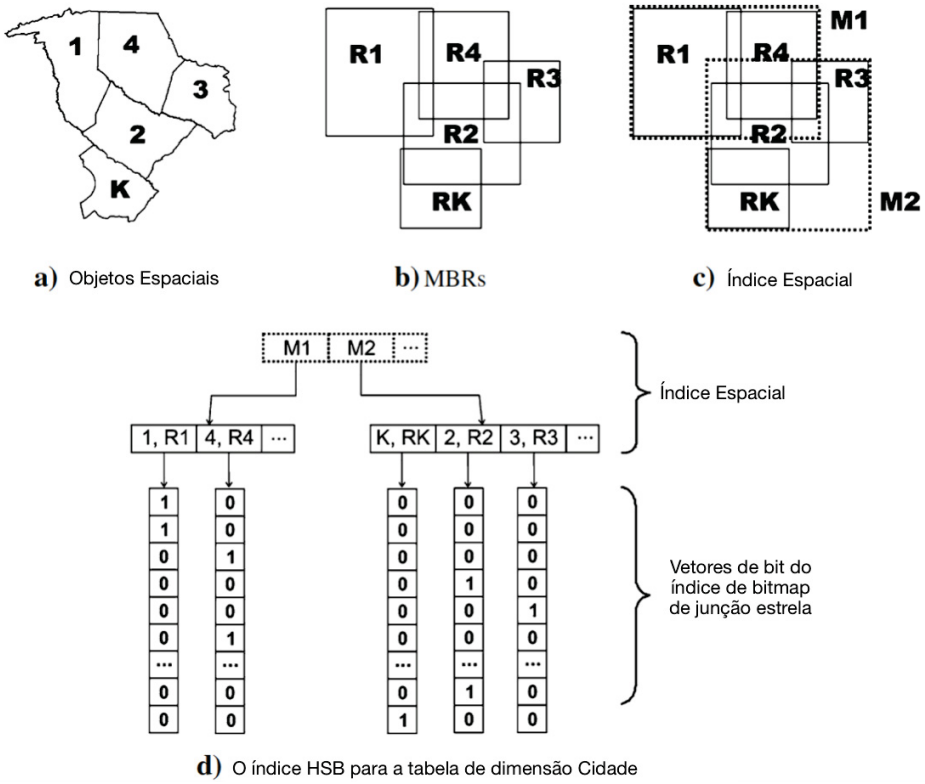


Figura 3.10: Índice HSB, visto em Siqueira et al. (2012)

3.2.6 Análise Comparativa

Os trabalhos foram classificados em suporte para dados temporais e/ou geográfico, se o foco da utilização do índice é para redução de espaço de armazenamento e/ou de processamento de consulta. Também, se a implementação foi focada em consultas de igualdade, pois é um dos objetivo do MIDET tratar consultas que envolvem as operações de seleção, junção, projeção e agregação. A Tabela 3.2 apresenta trabalhos relacionados com os critérios mencionados.

Foram comparados 6 trabalhos de bitmap sendo que os 3 primeiros (Encoded Bitmap Indexing, Enhanced Encoded Bitmap e Roaring Bitmap) são focados apenas em redução do custo de espaço e de consultas de igualdade. Estes trabalhos apresentam algoritmos para redução do espaço da representação dos dados. Já os trabalhos (I-DWE, Índice SB e Índice HSB) visam a representação geográfica de objetos espaciais e não a redução do espaço de armazenamento. Somente os trabalhos de (Enhanced Encoded Bitmap) e de (Índice HSB) visam a otimização de consultas de igualdade.

O MIDET visa permitir a utilização de modelagem de ponto e linhas como representação de espaço, e intervalos de tempo como modelagem temporal, como o trabalho de Savary et al. (2003), porém acrescentando dados temporais pontuais. Como o trabalho de Papadias et al. (2001) a utilização de mais de uma estrutura de indexação e sua combinação será considerada. Neste trabalho, o intuito é o processamento eficiente de consultas em eventos de trânsito promovendo a indexação espaço-temporal. O MIDET utiliza o Roring Bitmap de Lemire et al. (2018) para a implementação de bitmaps devido a sua capacidade de compressão e desempenho em consultas.

O trabalho de Siqueira et al. (2008) (I-DWE) e Siqueira et al. (2012) (SB e HSB) é implementado para consultas analíticas espaciais em *Data Warehouse*. Portanto, possui um intuito diferente do MIDET. Já os trabalhos de bitmap Keawpibal et al. (2019),Wu e Buchmann (2008), Keawpibal et al. (2012) e Lemire et al. (2018) (Bitmap duplo, Encoded Bitmap Indexing,

Estrutura	Temporal	Geográfico	Redução custo de espaço	Consultas de igualdade
Bitmap duplo			X	
Encoded Bitmap Indexing			X	
Enhanced Encoded Bitmap				X
Roaring Bitmap			X	
I-DWE		X		
Índice SB		X		
Índice HSB		X		X
MIDET	X	X	X	X

Tabela 3.2: Comparação de trabalhos relacionados de Bitmap

Enhanced Encoded Bitmap e Roaring bitmap) poderiam ser utilizados pelo MIDET para a implementação do bitmnap. Wang et al. (2017) afirmam que o Roaring possui um desempenho de criação, compressão e descompressão superior a outro bitmaps semelhantes. Devido a essa constatação, o Roaring foi escolhido. Como o MIDET indexa apenas dados históricos, o uso de bitmaps é justificado com a a utilização de operações *bitwise* e também pela não atualização dos dados.

3.3 CONSIDERAÇÕES

Neste Capítulo foram apresentados os trabalhos relacionados de indexação e bitmaps. Procedeu-se uma análise das características de cada trabalho, suas contribuições e diferenças para o MIDET.

O MIDET utiliza conceitos de trabalhos relacionados para criar um novo método de indexação de dados espaço-temporais. O objetivo do MIDET é possuir um bom desempenho para consultas espaço-temporais que envolvam as operação de seleção, junção, projeção e agregação. O trabalho de Doraiswamy et al. (2016) (Árvore STIG) orientou a criação de uma grade geográfica (tesselação), bem como a criação de blocos de registros. O trabalho de Buneman et al. (2004) (Árvore Buneman) inspirou a criação de bitmaps por tipo de evento e para cada período por ser armazenado em versões. Porém, diferente dos trabalhos de indexação, bitmaps também serão utilizados visando diminuir o espaço de busca em consultas espaço-temporais. Ao analisar os diferentes modelos de bitmap, foi escolhido o Lemire et al. (2018) (Roaring). A escolha foi baseada em um estudo realizado por Wang et al. (2017), que afirmam que o Roaring possui um desempenho de criação, compressão e descompressão superior às alternativas relacionadas.

A seguir, detalhes do modelo e do funcionamento do MIDET serão apresentados.

4 MIDET

Eventos de trânsito, como engarrafamentos e alertas, são continuamente produzidos através de aplicativos como o Waze. Em razão da velocidade com que estes dados são reportados, tais aplicativos armazenam eventos como registros individuais. Embora este modelo de armazenamento possa garantir um baixo custo de inserção em uma base de dados, ele produz baixo desempenho em consultas que requerem agregar os registros por seus atributos espaço-temporais. Para tratar deste problema, esta dissertação propõe um método para a indexação de dados de eventos de trânsito, chamado de MIDET (*Método para Indexação De Eventos de Trânsito*). O capítulo está estruturado da seguinte forma: na Seção 4.1 são apresentadas as estruturas de dados e de armazenamento, bem como detalhes do seu funcionamento; em seguida, a Seção 4.2 finaliza o capítulo com a apresentação de um exemplo de utilização do MIDET.

4.1 UM MÉTODO PARA INDEXAÇÃO PARA EVENTOS DE TRÂNSITO

A motivação do MIDET é a indexação de eventos de trânsito históricos, ocorridos em uma determinada região geográfica, relativos a um período de tempo. Assim, o método tem como objetivo obter um bom desempenho para consultas que necessitam fazer agrupamentos espaço-temporais sobre tais registros. Como eventos de trânsito são reportados em grandes volumes de dados, um objetivo adicional é que o método tenha um baixo custo de inserção. Para atender estes dois objetivos, o MIDET adota a estratégia de manter os eventos armazenados como registros individuais, mas agrupando-os pela sua proximidade espaço-temporal.

A estratégia de agrupamento é similar à representação em forma matricial (tesselação) utilizada para dados em sistemas de informação geográfica, na qual a área de interesse é caracterizada por uma matriz de células de tamanhos regulares, cada uma associada a um conjunto de valores que representam as características geográficas da região [Silva (2002)]. Para o MIDET, eventos ocorridos dentro dos limites de cada célula são armazenados em conjunto. Além disso, para tratar a questão temporal, eventos que ocorreram dentro da célula são ordenados cronologicamente e particionados por intervalos de tempo pré-determinados pelo administrador. A definição desses parâmetros pode impactar no desempenho do modelo, portanto testes para determinar os melhores tamanhos devem ser realizados.

O método toma como base as diferentes estruturas de indexação e de índices bitmaps apresentadas no Capítulo 2 e os trabalhos relacionados no Capítulo 3. O diferencial desta dissertação é a combinação da utilização de tesselação com a indexação temporal e a indexação de bitmaps, para cada intervalo de tempo e tipo de evento. Nesta dissertação são considerados os eventos de engarrafamento, alerta e irregularidade.

4.1.1 O Modelo MIDET

Os eventos de trânsito são registros de ocorrências no trânsito e podem ser de diversos *tipos*, como engarrafamento, alerta e irregularidade. Cada registro possui um *timestamp* (momento de ocorrência do evento) e uma dimensão geográfica, que pode ser um ponto ou um conjunto de pontos, definidos pela sua latitude e longitude. Além dos atributos espaço-temporais, eventos podem conter outros atributos, tais como a gravidade ou o nível de alerta.

O tamanho vertical de cada célula é definido por *PassoLat* e horizontal por *PassoLong* que correspondem *(latID-LatSE)/L* e *(longID-longSE)/C*, respectivamente. Para definir as

coordenadas de limite de cada CG, o MIDET utiliza as linhas e colunas da correspondência matricial, multiplicado pelo seu *PassoLat* e *PassoLong* para a coordenada do ponto superior esquerdo, e adicionando mais um passo para coordenada do ponto inferior direito. Assim, a CG_0 tem o canto superior esquerdo na coordenada (*latSE, longSE*) e o canto inferior direito na coordenada (*latSE + PassoLat, longSE + PassoLong)* e a CG_{11} é delimitada pelos pontos *(LatSE + 2 * PassoLat, LongSE + 3 * PassoLong)* e *(LatSE + 3 * PassoLat, LongSE + 4 * PassoLong))*.

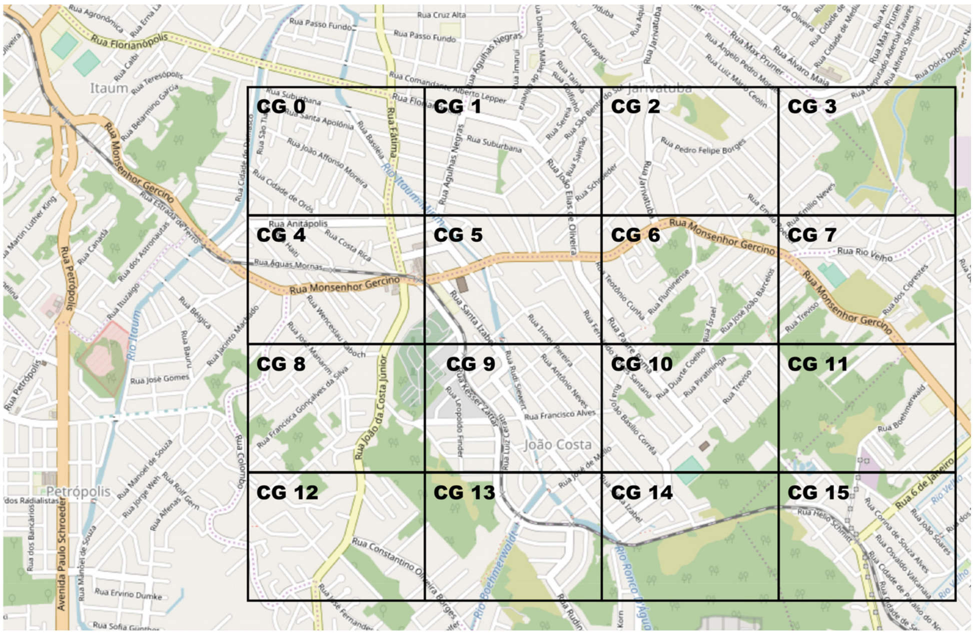


Figura 4.1: Grade com CGs

Para identificar em qual CG as coordenadas *(lat, long)* de cada ponto de um evento se encontra é utilizada a seguinte fórmula:

$$linha = \left\lfloor \frac{lat - latSE}{passoLat} \right\rfloor \qquad coluna = \left\lfloor \frac{long - longSE}{passoLong} \right\rfloor$$

Para registros de eventos de trânsito que possuam a dimensão geográfica informada por um conjunto de pontos (engarrafamento e irregularidade) o MIDET analisa se dois pontos sequenciais do registro, p_n e p_{n+1}, encontram-se em CGs distintas. Caso isto ocorra há a necessidade da divisão desse conjunto de pontos. São criados dois registros, r_1 e r_2, onde r_1 contém o ponto inicial até p_n, além de um ponto adicional referente ao ponto de interseção do limite da CG e do segmento de reta formado por p_n e p_{n+1}. Este mesmo ponto é inserido em r_2, além dos pontos a partir de p_{n+1}. Este mesmo processo se repete até que o conjunto seja inteiramente analisado.

4.1.2 Armazenamento

O MIDET utiliza um modelo de armazenamento misto, com arquivos binários e uma base de dados relacional. Os registros de eventos são armazenados em arquivos e a base de dados mantém informações para agilizar o acesso aos arquivos.

Organização dos arquivos.
Um conjunto de registros de eventos compõe blocos. *Blocos* são unidades de transferência entre o disco e a memória e contem registros de um mesmo tipo de evento e que ocorreram em uma

mesma CG. Quando a quantidade de registros em uma CG ultrapassa o tamanho máximo de bloco pré-definido, um novo bloco é criado. Caso não ultrapasse o tamanho máximo, este espaço livre não é preenchido. Uma sequência de blocos do mesmo tipo de evento que ocorreram dentro de um intervalo de tempo compõe um arquivo. O *intervalo de tempo* é definido pelo adiministrador em número de dias e deve ser o mesmo para todos os tipos de eventos. Em outras palavras, um *arquivo* está associado a um *intervalo de tempo*, refere-se a um tipo de evento e é composto por blocos, que por sua vez são compostos por registros de eventos que ocorreram dentro de uma mesma CG. A Figura 4.2 apresenta um exemplo desta organização.

Para a otimização da busca de registros, é criado um *header* para cada bloco durante a sua gravação. A Figura 4.3 retrata a organização de registros em *blocos* e a estrutura do *header*. O *header* é composto por: ID do bloco, ID da CG referenciada, quantidade de registros no bloco e o endereço de cada registro dentro do bloco. O tamanho do *header* é definido pela quantidade de *slots*, ou seja, pela quantidade máxima de registros que o bloco pode conter. No exemplo da Figura 4.3, o primeiro bloco tem id `0`, o id da CG referenciada `1`, a quantidade de registros no bloco `2` , o endereço de cada registro `20` e `93` e o fim do bloco no endereço `168`.

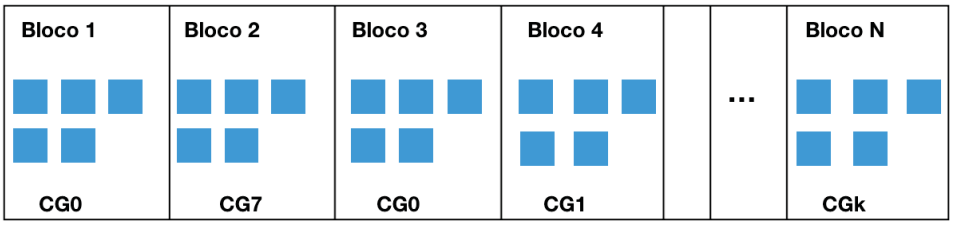


Figura 4.2: Organização do arquivo em blocos

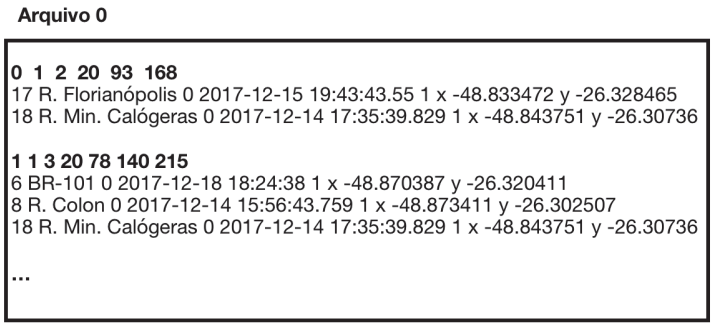


Figura 4.3: Bloco e estrutura do header

Como os registros de eventos contém campos de tamanho variável , são utilizados campos adicionais. Por exemplo, o tamanho do nome da rua pode evitar a pesquisa de cada caractere para se obter o nome, e a quantidade de pontos da geometria do registro facilita seu processamento. Por exemplo, na Figura 4.3, o primeiro registro do bloco *0* (*R. Florianópolis*) tem tamanho de nome *17* e a quantidade de pontos (x e y) é *1*.

Para otimizar as consultas, cada arquivo é associado a um índice bitmap (*vBit*), indexado pelas CGs, de forma que *vBit[n]=1* se o arquivo contém registros de eventos que ocorreram na CG_n e *vBit[n]=0*, caso contrário. A Figura 4.4 apresenta 3 índices bitmap, cada um associado a um arquivo com tipos de eventos distintos. Para o bitmap de engarrafamento, o índice mostra

que o arquivo associado contém registros que ocorreram nas CGs 0, 3 e 5, mas nenhum registro nas CGs 1, 2, 4, 6 e 13.

Organização da base relacional.
Os índices bitmaps associados aos arquivos são armazenados em uma base de dados relacional em uma tabela chamada de Bitmap. Cada linha da Tabela Bitmap corresponde a um intervalo de tempo, com a conversão de anos e meses em dias. Os atributos da tabela são: o início e o fim do intervalo em dias, os índices bitmap para cada tipo de evento de trânsito e o contador do arquivo. Por exemplo, de acordo com a linha da tabela ilustrada na Figura 4.5, os registros de eventos para o período entre `Período Dia` e `Período Dia Fim` estão armazenados nos arquivos `Alerta0`, `Engarrafamento0` e `Irregularidade1`. Ou seja, é o primeiro arquivo criado para alerta, o primeiro arquivo para engarrafamento e o segundo arquivo para irregularidades. A tabela também contém os bitmaps ilustrados na Figura 4.4.

Para cada tipo de evento de trânsito é criada uma Tabela de vetor de bloco. Esta tabela associa cada CG aos blocos que compõem os arquivos, como ilustrado na Figura 4.6. A tabela possui 3 informações: o contador do arquivo, a identificação da CG e o endereço do bloco neste arquivo.

Também é criada uma Tabela CG com L * C (linhas * colunas) tuplas, que correspondem a matriz de CGs. Ela composta pelo identificador da CG e um retângulo definido por suas 4 coordenadas limites, com o ponto superior esquerdo, ponto superior direto, ponto inferior esquerdo e ponto inferior direito.

	CG 0	CG 1	CG 2	CG 3	CG 4	CG 5	CG 6	…	CG13	…
Engarrafamento	1	0	0	1	0	1	0		0	
Alerta	1	1	1	0	0	0	0		1	
Irregularidade	0	1	0	0	0	1	0		1	

Figura 4.4: Índices bitmap do domínio de CGs, onde se tem engarrafamento, alerta e irregularidade

Detalhes da criação, bem como da utilização do índice bitmap e das demais estruturas que compõem o MIDET são apresentados na próxima seção.

4.1.3 Criação do Modelo

Na Figura 4.7, o processo de criação do método de armazenamento e indexação MIDET é mostrado. Como entrada para o processo são dados: o banco de dados com eventos de trânsito, a área de interesse com as coordenadas superiores da esquerda e inferiores da direita, o tamanho máximo dos blocos em bytes, a quantidade de *slots* no *header* dos blocos para cada tipo de evento, quantidade de linhas e colunas da matriz de tesselação, data inicial e o intervalo de tempo em dias.

O método consiste em duas fases de criação: *criação da matriz* sobre a área geográfica de interesse e *criação dos arquivos* de registros de eventos e da *indexação* com a Tabela de bitmap e as Tabelas de vetor de bloco para cada tipo de evento (Figura 4.7). Após essas duas fases, as consultas são realizadas utilizando a Tabela Bitmap e as Tabelas de vetor de bloco, sobre as informações armazenadas nos arquivos.

Na primeira fase, é feita a divisão geográfica utilizando o número de linhas (L) e colunas (C) e a área de interesse pré-definidos, em uma matriz sobre a área de interesse dividindo-a

Tabela Bitmap

Periodo Dia	Periodo Dia Fim	Bitmap Alerta	Bitmap Engarrafamento	Bitmap Irregularidade	Arquivo Alerta	Arquivo Engarrafamento	Arquivo Irregularidade
736532	736632	{1,1,1,0,0,0,0,..., 1...}	{1,0,0,1,0,1,0,..., 0 ...}	{0,1,0,0,0,1,0,..., 1, ...}	0	0	1
...	...	...	...	...	...	...	...

Arquivo	CG_id	Endereço Bloco
0	0	End Bloco 1
0	0	End Bloco 3
0	1	End Bloco 4
0	7	End Bloco 2
...	...	..
0	K	End Bloco N

Tabela Vetor de Bloco de Alerta

Tabela CG

CG_id	Geom
0	POLYGON((-49.3 -26.5,-49.2765 -26.5,-49.2765 -26.475,-49.3 -26.475,-49.3 -26.5),(-49.3 -26.5,-49.2765 -26.5,-49.2765 -26.475,-49.3 -26.475,-49.3 -26.5))
1	POLYGON((-49.3 -26.475,-49.2765 -26.475,-49.2765 -26.45,-49.3 -26.45,-49.3 -26.475), (-49.3 -26.475,-49.2765 -26.475,-49.2765 -26.45,-49.3 -26.45,-49.3 -26.475))
2	POLYGON((-49.3 -26.45,-49.2765 -26.45,-49.2765 -26.425,-49.3 -26.425,-49.3 -26.45), (-49.3 -26.45,-49.2765 -26.45,-49.2765 -26.425,-49.3 -26.425,-49.3 -26.45))
3	POLYGON((-49.3 -26.425,-49.2765 -26.425,-49.2765 -26.4,-49.3 -26.4,-49.3 -26.425), (-49.3 -26.425,-49.2765 -26.425,-49.2765 -26.4,-49.3 -26.4,-49.3 -26.425))
...	..
K	POLYGON((-48.8535 -26.025,-48.83 -26.025,-48.83 -26,-48.8535 -26,-48.8535 -26.025), (-48.8535 -26.025,-48.83 -26.025,-48.83 -26,-48.8535 -26,-48.8535 -26.025))

Figura 4.5: Tabela de Bitmap e Tabela de Vetor de Bloco

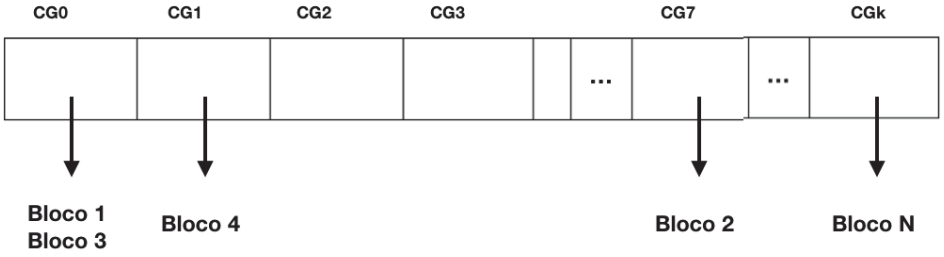
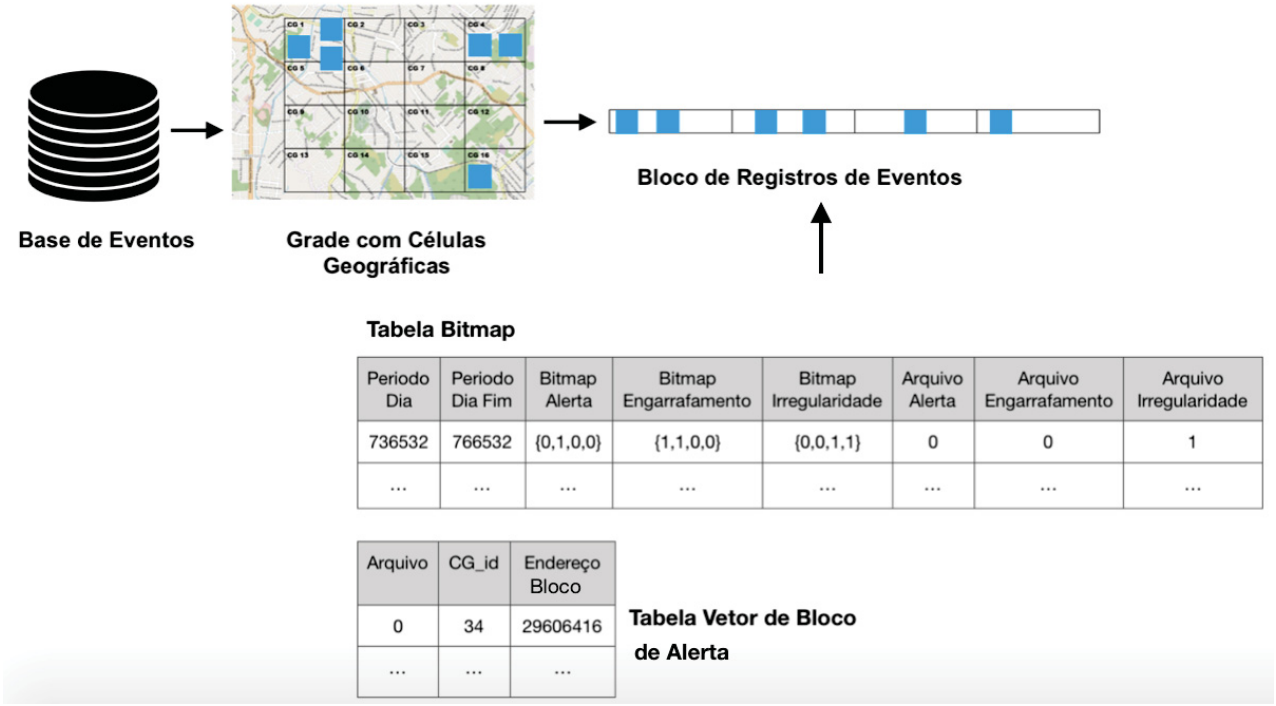


Figura 4.6: Vetor de CGs e blocos

em células geográficas (CGs) (Figura 4.1). Assim, são obtidos os *PassoLat* e *PassoLong* que servirão de base para o cálculo dos limites das CGs a partir das coordenadas do canto superior esquerdo e do canto inferior direito da área de interesse. O resultado desta divisão, para cada CG, é gravado na Tabela CG na base relacional, com a identificação da CG e as coordenadas limites.

Na segunda fase é feita a leitura da base de origem, com os registros em ordem cronológica. O MIDET efetua a distribuição de registros de eventos de trânsito separando os registros por CGs. As coordenadas de cada ponto, são utilizadas para determinar em qual(is) CG(s) o evento ocorreu. Esse processo é executado para cada intervalo de tempo definido e para cada tipo de evento (engarrafamento, alerta e irregularidade).

Primeiramente, é calculado o período utilizando a data inicial e o tamanho do intervalo de tempo. Caso o registro tenha um *timestamp* maior que este intervalo, é encerrado o período em processamento, os arquivos são fechados e é iniciado um novo período com novos arquivos e uma nova data de início. Após esta verificação, é calculado em qual CG as coordenadas de cada ponto desse registro se encontram. Caso um ponto mude de CG, é aplicado o processo de inserção de pontos intermediários (Seção 4.1.1) e um novo registro é criado. Em seguida, é verificado se a soma do tamanho do registro com o tamanho dos registros já inseridos (no bloco)



Periodo Dia	Periodo Dia Fim	Bitmap Alerta	Bitmap Engarrafamento	Bitmap Irregularidade	Arquivo Alerta	Arquivo Engarrafamento	Arquivo Irregularidade
736532	766532	{0,1,0,0}	{1,1,0,0}	{0,0,1,1}	0	0	1
...	...	...	...	...	...	...	...

Arquivo	CG_id	Endereço Bloco
0	34	29606416
...	...	...

Figura 4.7: Método proposto

é superior ao tamanho máximo pré-definido do bloco. Caso seja superior, o bloco é armazenado no arquivo, juntamente com os seus registros e um bloco vazio é criado para aquele registro. Caso seja inferior, o registro é adicionado ao bloco. Ao final da leitura, todos os registros são armazenados em blocos no arquivo.

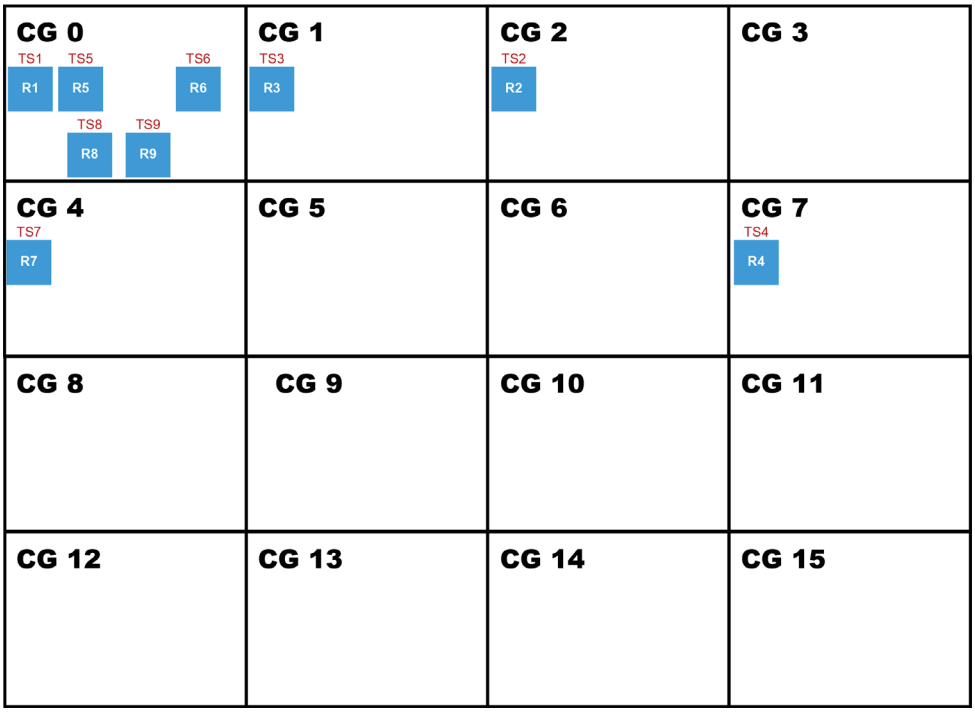


Figura 4.8: Inserção de registros em suas CGS

Nesta segunda fase são criadas, também, a Tabela Bitmap e as Tabelas de Vetor de Bloco (4.5). Primeiramente é criado o índice bitmap. Ao ler um registro, é verificado a qual

CG ele pertence e o *tipo* de registro. Assim é colocado o bit *1*(verdadeiro) no vetor de bit. Por exemplo, considere um registro r que pertence à *CG0* e é do tipo *engarrafamento*, o índice bitmap de *Engarrafamento* na Figura 4.4 tem o *vBit[0]* como verdadeiro. O vetor de bloco é criado no início do período e, a medida em que os blocos são *armazenados* os endereços destes blocos são inseridos na lista de bloco na CG correspondente. Por exemplo, na Figura 4.6 o *bloco 1* e o *bloco 3* são blocos referentes à *CG0*.

A criação dos índices bitmap e dos vetores de bloco é realizada para cada período e cada tipo de evento. No final de um período, os índices bitmap são armazenados na Tabela Bitmap no Banco de Dados, juntamente com o contador do arquivo de cada evento correspondente ao período. Cada tupla da Tabela Bitmap corresponde a um período. Existe uma Tabela de Vetor de Bloco para cada tipo de evento de trânsito. Os vetores de bloco criados a cada período são armazenados na tabela correspondente ao final do período. O contador do arquivo indica qual é o arquivo correspondente para aquele bloco. Por exemplo, a Figura 4.6 ilustra um vetor de bloco para o *período 0* que ao ser inserido na Tabela de Vetor de Bloco tem o *Arquivo 0* como contador de arquivo.

Detalhes do processamento são apresentados a seguir.

4.1.4 Processamento de Consultas

Esta Seção apresenta a definição das consultas que são processadas pelo MIDET e o processamento de consultas sobre o ambiente criado pelo MIDET.

As consultas utilizada pelo MIDET envolvem as operações de seleção, junção, projeção e agregação.

A Figura 4.9 apresenta os passos do processamento de um consulta no MIDET. Ele envolve as Tabela de Bitmap e as Tabelas de Vetor de Bloco para cada tipo de evento, para obter o número sequencial de arquivo, a CG e o endereço em bytes dos blocos de interesse. Por exemplo, uma consulta que envolva engarrafamento vai obter informação da Tabela de Bitmap e a Tabela de Vetor de Bloco de engarrafamento. Após, é feita a leitura do bloco, utilizando o *header* para obter o endereço de cada registro dentro do arquivo correspondente. Na Figura 4.9, o arquivo 0 ilustra a estrutura do arquivo para o período 0.

A consulta executada pelo MIDET, segue os seguintes passos:

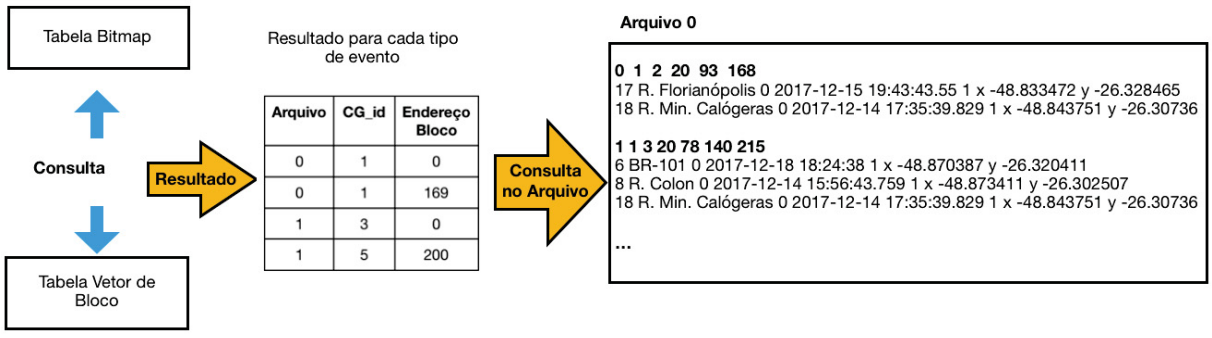


Figura 4.9: Esquema de uma consulta no MIDET

I) Identificação de quais são os eventos envolvidos nessa consulta. Por exemplo, para a consulta “Quantos engarrafamentos e alertas ocorreram na rua X, de 2010 a 2015? “ apenas seriam considerados os eventos de engarrafamento e alerta.

II) Identificação do conjunto de períodos que deverão ser analisados. Por exemplo, na mesma consulta usada no item *I*, e considerando que o parâmetro para intervalo de tempo foi definido como 365 dias, apenas seriam considerados 6 períodos anuais.

III) Para cada período a ser analisado os passos são:

- Conjunção entre os bitmaps de CGs dos tipos identificados no passo *I*, objetivando encontrar as CGs que são comuns a todos estes eventos. Por exemplo, considerando a mesma consulta do item *I*, seriam identificadas as CGs cujos vBit[n] estivessem como *verdadeiro* tanto no bitmap de engarrafamento quanto de alerta.

- Para cada evento envolvido efetua a seleção dos arquivos e blocos a serem analisados, através da pesquisa de cada CG comum na relacionando a Tabela Bitmap com as Tabelas de Vetor e seleciona os registros a serem processados através da busca direta no arquivo pelo endereço do *header* de bloco. Por exemplo, se a CG identificada na conjunção realizada no passo fosse a *CG 1* e considerando o resultado da consulta Figura 4.9 a pesquisa seria efeituada nos blocos com endereço *0* e *169*.

- Pesquisa diretamente em cada um dos registros, utilizando os seus campos de informação. Por exemplo, considerando o *bloco 0*, a pesquisa seria efetuada apenas nos registros deste bloco.

IV) Ao processar todos os eventos de todos os períodos envolvidos, o resultado da consulta estará completo

A seguir, um exemplo de criação e utilização do MIDET será demonstrado.

4.2 EXEMPLO PRÁTICO

Nesta Seção é apresentado um exemplo de criação e utilização do MIDET utilizando dados extraídos do aplicativo Waze [4]. Todos os passos da fase de criação das estruturas e dos índices são exemplificados. Posteriormente, o processo para realização de uma consulta é ilustrado.

A Tabela apresentada na Figura 4.10 foi extraída da base Waze. Ela possui dados sobre engarrafamentos (*Jam*) e alertas (*Alert*). O campo *line* indica o conjunto de pontos do engarrafamento e o ponto do alerta.

Jam

ID	End_node	Road_type	Street	Street_FK	City	Country	Delay	Length	Level	Datafile_id	Line
1	R. Dr. João Colin	2	R. Itaiópolis	1	Joinville	BR	70	549	2	171540	"[{"x": -48.843926, "y": -26.287405}, {"x": -48.844654, "y": -26.287378}, {"x": -48.845189, "y": -26.287358},
2	R. Valença	2	R. Florianópolis	2	Joinville	BR	69	512	2	171540	"[{"x": -48.836247, "y": -26.325694}, {"x": -48.835558, "y": -26.32665}, {"x": -48.835106, "y": -26.32725}, {"
3	R. Da. Francisca	1	V. Exclusiva para ônibus	3	Joinville	BR	81	685	2	171540	"[{"x": -48.841604, "y": -26.301808}, {"x": -48.841867, "y": -26.301496}, {"x": -48.842022, "y": -26.301255},
4	R. Tuiuti	1	R. Rouxinol	4	Joinville	BR	78	427	2	171541	"[{"x": -48.816698, "y": -26.256093}, {"x": -48.81986, "y": -26.25868}]"
5	R. Gen. Rondon	2	R. Florianópolis	2	Joinville	BR	125	435	3	171541	"[{"x": -48.83981, "y": -26.320857}, {"x": -48.839656, "y": -26.321065}, {"x": -48.839319, "y": -26.321519}, {
6	R. João da Costa Jr	2	R. Mons. Gercino	5	Joinville	BR	191	358	4	171541	"[{"x": -48.837846, "y": -26.326728}, {"x": -48.838013, "y": -26.326406}, {"x": -48.838245, "y": -26.326094},
7	R. Itaiópolis	2	Av. José Vieira	6	Joinville	BR	132	193	4	171541	"[{"x": -48.844021, "y": -26.284109}, {"x": -48.843527, "y": -26.284729}, {"x": -48.843451, "y": -26.284941},
8	R. Areia Branca	2	R. do Porto	7	Joinville	BR	94	206	4	171542	"[{"x": -48.807743, "y": -26.259652}, {"x": -48.807533, "y": -26.259898}, {"x": -48.807183, "y": -26.260255},
9	R. Valença	2	R. Florianópolis	2	Joinville	BR	69	512	2	171542	"[{"x": -48.836247, "y": -26.325694}, {"x": -48.835558, "y": -26.32665}, {"x": -48.835106, "y": -26.32725}, {"

Alert

ID	Road_type	Street	Street_FK	City	Country	Realiability	Type	SubType	Datafile_id	Line
1	2	R. Florianópolis	2	Joinville	BR	5	WEATHERHAZARD	HAZARD_ON_ROAD_POT_HOLE	162113	{"x": -48.836772,"y": -26.324968}
2	2	R. Fátima	8	Joinville	BR	5	WEATHERHAZARD	HAZARD_ON_ROAD_POT_HOLE	162117	"{"x": -48.817513, "y": -26.329484}"
3	2	R. Fátima	8	Joinville	BR	5	WEATHERHAZARD	HAZARD_ON_ROAD_POT_HOLE	162118	"{"x": -48.817513, "y": -26.329484}"
4	2	R. Visconde de Taunay	9	Joinville	BR	5	JAM	JAM_HEAVY_TRAFFIC	162121	"{"x": -48.85453, "y": -26.308517}"
5	1	R. Maj. Vieira	10	Joinville	BR	5	WEATHERHAZARD	HAZARD_ON_ROAD_POT_HOLE	162123	"{"x": -48.845459, "y": -26.333522}"
6	2	R. Ottokar Doerffel	11	Joinville	BR	5	ACCIDENT	ACCIDENT_MINOR	162121	"{"x": -48.861588, "y": -26.314766}"
7	2	R. Ottokar Doerffel	11	Joinville	BR	5	ACCIDENT	ACCIDENT_MINOR	162122	"{"x": -48.861588, "y": -26.314766}"
8	2	Av. Rolf Wiest	12	Joinville	BR	5	JAM	JAM_HEAVY_TRAFFIC	162128	"{"x": -48.851824, "y": -26.257107}"
9	2	R. Da. Francisca	13	Joinville	BR	5	JAM	JAM_STAND_STILL_TRAFFIC	162121	"{"x": -48.889209, "y": -26.241511}"

Figura 4.10: Tabela Exemplo

É feita a divisão geográfica em CGs (tesselação) (Figura 4.11). As CGs são armazenadas na Tabela de CGs na base relacional.

A Figura 4.12 ilustra a disposição de engarrafamentos da Tabela da Figura 4.10 respeitando o esquema de ruas apresentado na Figura 4.11. A Figura 4.13 ilustra a distribuição dos alertas. É possível observar que os tipos engarrafamento e alerta possuem tratamento

[4] https://github.com/joinvalle/Joinville-Smart-Mobility

geográfico diferente. O tipo engarrafamento é um trecho de rua e é representado por uma sequência de pontos formando trechos lineares ou não. Já o tipo alerta é caracterizado por apenas um ponto.

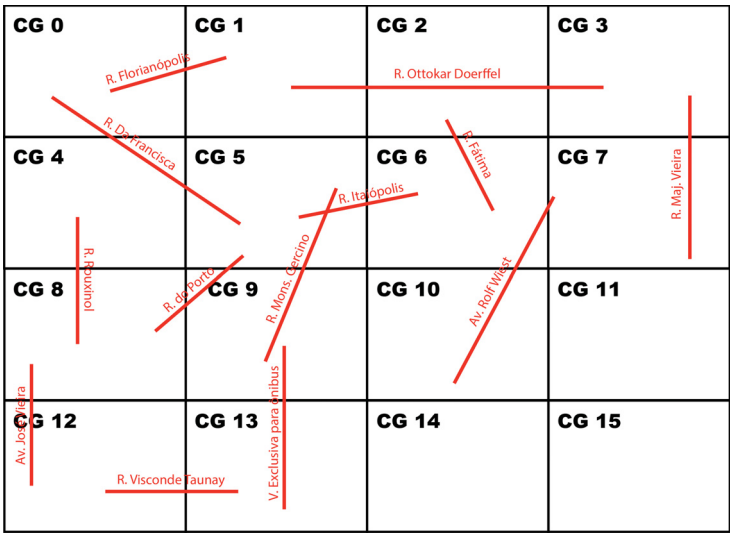


Figura 4.11: Grade de CGs com disposição de ruas

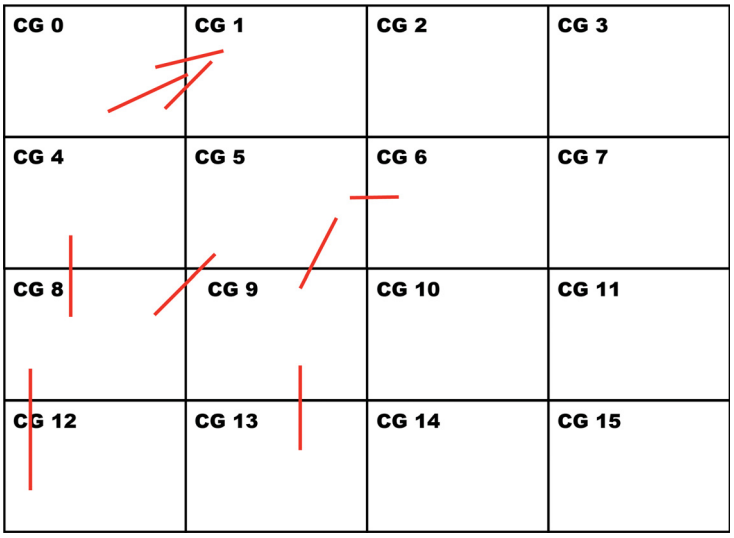


Figura 4.12: Distribuição de Engarrafamento em CGs

A Figura 4.14 ilustra o começo da distribuição dos registros da Tabela de engarrafamento da Figura 4.10. O registro `J1` equivale ao primeiro registro da Tabela engarrafamentos (*Jam*). Como o engarrafamento `Id 1` na rua Itaiópolis pertence a mais de uma CG, o registro é inserido em ambas as CGs. Porém, este registro não é copiado completamente, cada ponto pertencente a ele é separado por sua geografia e inserido na CG à qual este ponto pertence, como visto na Seção 4.1. Adicionalmente, é inserido no índice bitmap o *vBit[n]* como verdadeiro. No exemplo, o engarrafamento `Id 1` teria seus pontos divididos entre as CGs C5 e C6 e o índice bitamp de engarrafamento vBit[5]=1 e vBit[6]=1.

A Figura 4.15 mostra o processo de distribuição de registros já mais avançado. O tamanho máximo do bloco foi pré-definido como 126 Bytes, que equivale a 3 slots.

É possível observar na Figura 4.16 que os registros que estavam na `CG5` na Figura 4.15 foram armazenados no `Bloco 1`, com a CG correspondente. Posteriormente, os registros nas

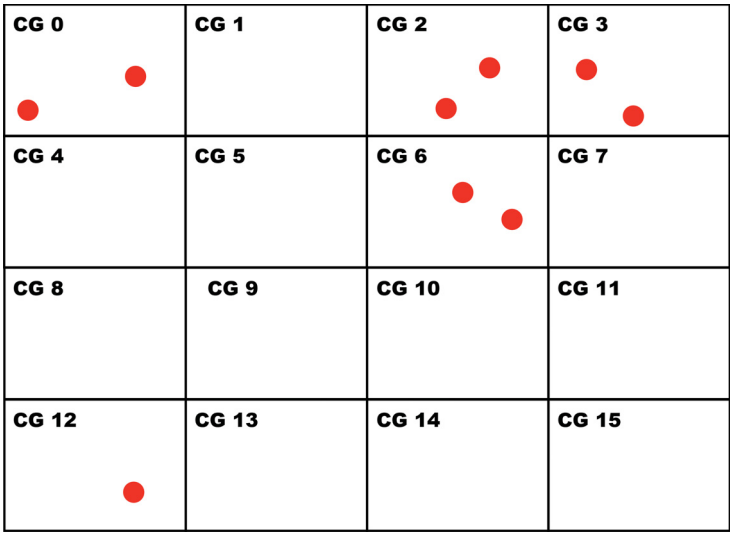


Figura 4.13: Distribuição de Alerta em CGs

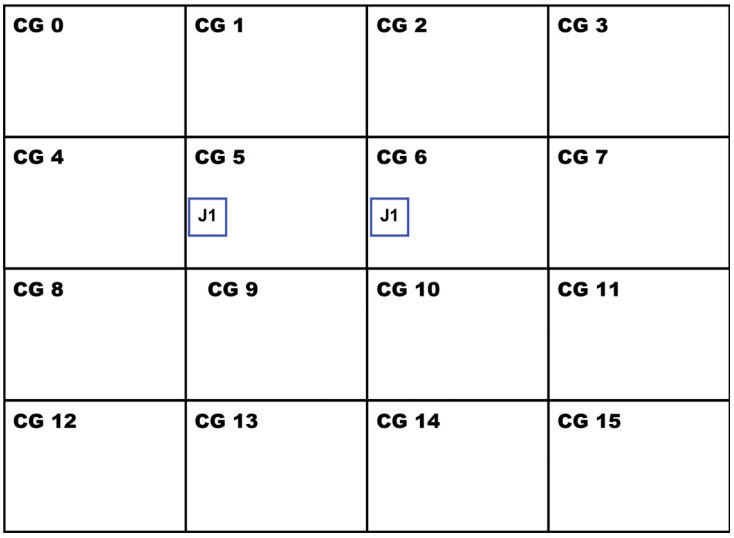


Figura 4.14: Começo da distribuição de registros da Tabela de engarrafamento em CGs

`CG8` e `CG9` também serão armazenados em blocos do arquivo. A medida que as CGs possuam o tamanho do bloco pré-definido, esses registros serão igualmente armazenados no arquivo em novos blocos e seus endereços no vetor de blocos correspondente.

Ao final da leitura, todos os registros são armazenados em blocos no arquivo. A Figura 4.17 representa o arquivo de registros de engarrafamento resultante. É possível observar que o arquivo de engarrafamento possui blocos com tamanho menor que o tamanho máximo pré-definido, pois não é preenchido o espaço livre. Os vetores de bloco de alerta e engarrafamento são *armazenados* nas tabelas de Vetor de bloco correspondentes e os índices bitmap são *armazenados* na Tabela Bitmap.

A Figura 4.18 ilustra o arquivo de alertas da Tabela 4.10 que passou pelo mesmo processo que a Tabela de engarrafamento.

Após este processo, são criados os índices bitmaps e as Tabelas Bitmap e Vetores de Bloco, como descrito na Seção 4.1. Ao fim desta criação, é possível realizar consultas.

Como exemplo prático, o processo para realizar a consulta "Algum ponto na rua Florianópolis tem engarrafamento e alerta no tempo 736537?"será demonstrado. Para responder a esta consulta é necessário utilizar as seguintes Tabelas: engarrafamento e alerta. Para realizar a

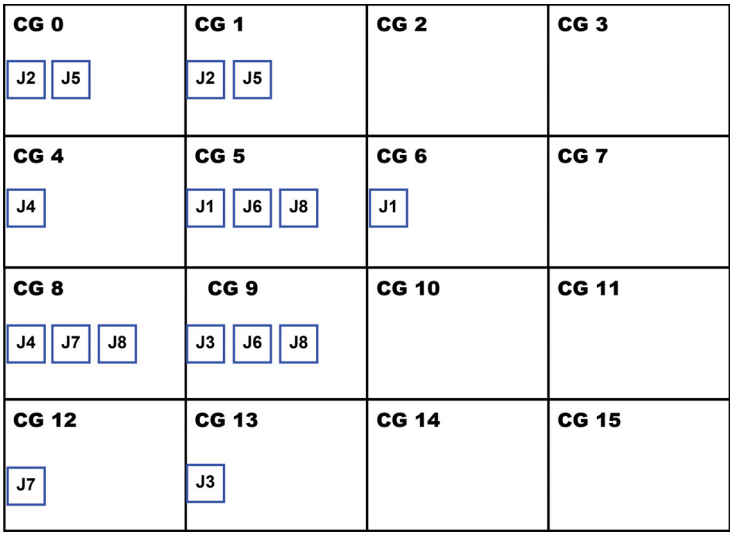


Figura 4.15: Processo de distribuição de registros da Tabela de engarrafamento em CGs

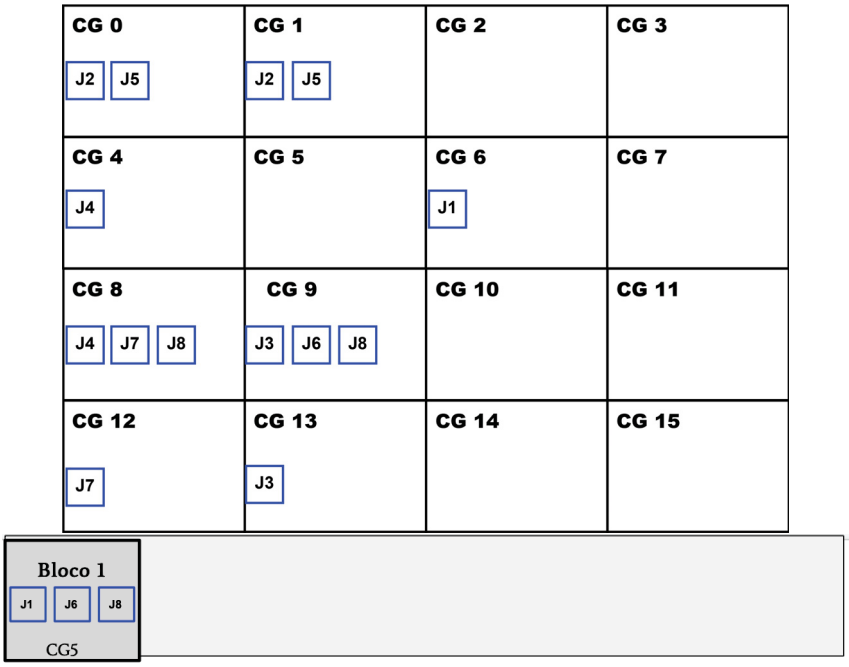


Figura 4.16: Distribuição de registros da Tabela de engarrafamento em CG e inserção no arquivo

Figura 4.17: Arquivo Engarrafamento

busca, é feita a seleção do `tempo 736537` na Tabela bitmap. A Figura 4.19 mostra o índice bitmap sobre o domínio de CGs onde se tem engarrafamento e alerta.

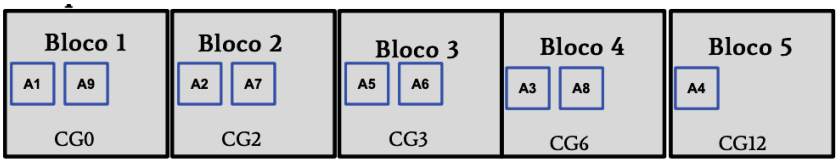


Figura 4.18: Arquivo Alerta

A Figura 4.19 mostra a conjunção do índice de bitmap de engarrafamento e o índice de bitmap de alerta realizada para determinar quais CGs são de interesse para a consulta. Os retângulos em vermelho destacam as CGs que possuem todas as características. Portanto as `CGs` *0, 6 e 12* são de interesse para a consulta.

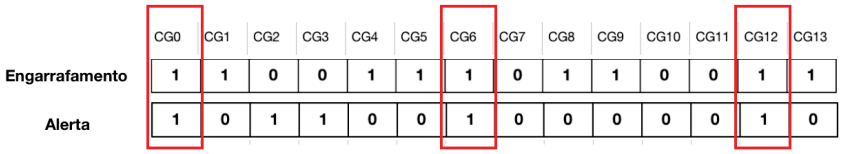


Figura 4.19: Conjunção de bitmaps

Com este resultado, a última etapa consiste em selecionar os blocos de cada arquivo que se relacionam com as `CGs` *0, 6 e 12*. Para isto, utiliza-se a Tabela de vetor de CG de cada tipo de evento de trânsito: alerta e engarrafamento. A Figura 4.20 mostra o vetor de blocos de engarrafamento por CG. Como as `CGs` *0, 6 e 12* são o alvo, a seleção é feita na lista de blocos delas, na imagem em vermelho. Neste caso somente será necessário consultar os registros nos blocos `4, 7 e 8` do arquivo de engarrafamento. Do mesmo modo, esta seleção é feita no vetor de blocos de alerta por CG, na Figura 4.21, onde só será necessário consultar os blocos `1, 4 e 5` do arquivo de alerta.

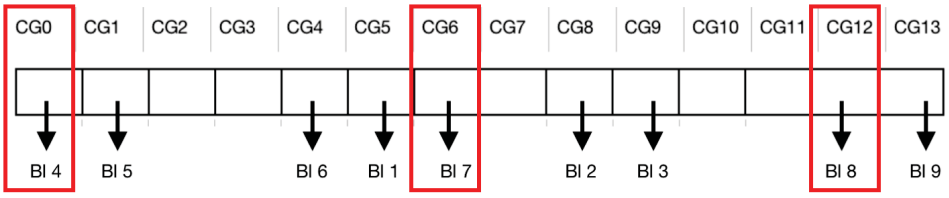


Figura 4.20: Vetor de CG e blocos de engarrafamento

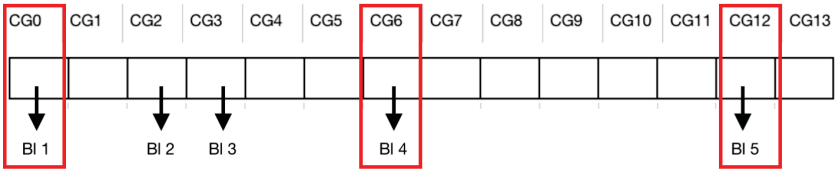


Figura 4.21: Vetor de CG e blocos de alerta

Para responder à consulta, os registros dos blocos `4, 7 e 8` de engarrafamento e os blocos `1, 4 e 5` de alerta serão combinados. Primeiramente será verificado para todos os registros do bloco de alerta se o nome da rua corresponde à Florianópolis, caso corresponda, este será inserido em uma lista de possíveis candidatos. Ao final da leitura do bloco de alerta, todos os registros da lista de candidatos resultante serão combinados com cada um dos registros do

bloco de engarrafamento, com o nome e a posição na rua comparados. A criação da lista de candidatos otimiza a consulta ao diminuir a quantidade de registros de alerta a serem combinados com registros de engarrafamento.

4.3 CONSIDERAÇÕES

Neste Capítulo foi apresentado o MIDET (*Método para Indexação De Eventos de Trânsito*). A motivação, modelo, estruturas de armazenamento e o funcionamento do MIDET foram apresentados na Seção 4.1. A Seção 4.2 apresenta um exemplo de criação e utilização do MIDET com dados extraídos do aplicativo *Waze*.

As seguir, o Capítulo 5 descreve a implementação do MIDET e experimentos realizados.

5 IMPLEMENTAÇÃO

Este capítulo descreve a implementação do MIDET e os experimentos realizados. A Seção 5.1 descreve a configuração do ambiente de implementação e bibliotecas utilizadas. A Seção 5.2 descreve a abordagem de armazenamento utilizada para comparação do desempenho. A Seção 5.3 descreve os experimentos realizados e o ambiente no qual foram executados, bem como cada consulta realizada. A Seção 5.4 conclui o capítulo.

5.1 AMBIENTE DE IMPLEMENTAÇÃO

A implementação foi realizada em um computador pessoal, executando o sistema operacional Mac OS X 10.15.2, com 1.1 Ghz Dual-Core Intel Core m3, com 8GB de memória RAM. A linguagem utilizada para a implementação foi em linguagem C, com o gcc 11.0.0. O código pode ser encontrado em gitlab [5].

5.1.1 Bibliotecas e Ferramentas

Para o armazenamento dos dados foi utilizada base de dados Postgres e arquivos binários. Para a criação, armazenamento e processamento de bitmaps foi utilizado o roaring bitmap (Lemire et al. (2018)). A escolha foi baseada no estudo realizado por Wang et al. (2017), que afirmam que o roaring bitmap é a alternativa superior aos outros bitmaps comprimidos e recomendam utilizá-lo sempre que possível. Para isto, foi utilizado o CRoaring [6] que é a implementação do roaring para a linguagem C e a extensão pg_roaringbitamp [7] para o Postgres.

Para a conexão com a linguagem C e o Postgres foi utilizada a biblioteca libpq[8].

5.1.2 Base de Dados

A base de origem *Waze* está armazenada em Postgres e foi obtida a partir do projeto *Smart Mobility* organizado pela prefeitura da cidade de Joinville-SC. A concessão dos dados do referido projeto foi fornecida à Universidade do Estado de Santa Catarina, parceira deste trabalho. A Base de origem *Waze* possui 13 Gigabytes (GB), contendo dados no período de setembro 2017 a setembro 2018. A partir desta base foi criada a base chamada de *base Waze*, apenas com atributos envolvidos nas consultas dos experimentos. Foi também criada uma árvore R sobre o atributo *Geometry*.

Foram criadas as bases (*B20*), (*B30*), (*B40*), (*B50*) e (*B100*) que correspondem a 20%, 30%, 40%, 50% e 100% da base Waze. A Tabela 5.1 apresenta a quantidade de registros correspondente ao percentual da base Waze, para cada *tipo* de evento. Para as bases de teste, foi escolhida como a área preferencial a região *central* da cidade de Joinville, por conter uma maior concentração de eventos de trânsito. As bases foram criadas em forma crescente, ou seja, a base *B 30* contém todos os registros contidos na *B20*, acrescidos dos próximos 10% de eventos. A base *B100* contém a totalidade de registros originários da base Waze.

[5] https://gitlab.com/mmgd/midm
[6] https://github.com/RoaringBitmap/CRoaring
[7] https://github.com/ChenHuajun/pg_roaringbitmap
[8] https://www.postgresql.org/docs/9.5/libpq.html

Base	Porcentagem	Num Regs Alerta	Num Regs Engarrafamento	Num Regs Irregularidade
B20	20 %	1024311	587816	22063
B30	30 %	1536466	887724	33095
B40	40 %	2048622	1183632	44126
B50	50 %	2560777	1479540	55158
B100	100 %	5121554	2959080	110316

Tabela 5.1: Quantidade de registros para cada tipo de evento

5.2 ARMAZENAMENTO DE REGISTROS

Os registros de eventos de trânsito são armazenados em arquivos compostos por *blocos*, que por sua vez correspondem a um conjunto de registros (Seção 4.1.3). Para determinar a eficiência do modelo de armazenamento do MIDET, que faz o armazenamento dos registros em arquivos, uma alternativa de armazenamento foi utilizada. A alternativa armazena os registros em tabelas em uma base de dados, como apresentado na próxima Seção.

5.2.1 Armazenamento Relacional

Para avaliar a eficiência do modelo de armazenamento do MIDET, que faz o armazenamento dos registros em arquivos, uma alternativa de armazenamento foi utilizada. A alternativa armazena os registros em tabelas em uma base de dados Postgres versão 11.5. A base é chamada de *Base Waze-CG*. A Figura 5.1 ilustra a organização dos registros neste modelo. É criada uma tabela para *cada tipo de evento*, contendo os atributos *nome*, *pub_utc_date*, *idCG* e *geometry*, que correspondem ao nome da rua, a data de publicação, a identificação da CG que contém a localização do evento e a geometria. Observe que, ao contrário do arquivo do MIDET, os registros de eventos encontram-se em um armazenamento relacional e são agrupados pelo atributo (IdCG). Foi também criada uma árvore R sobre o atributo *Geometry*. Para o atributo temporal, os registros são inseridos em ordem cronológica crescente.

Nome	Pub_utc_date	IdCG	Geometry
R. Florianópolis	2017-12-15 19:43:43.55	1	(x -48.833472, y -26.328465)
R. Min. Calógeras	2017-12-14 17:35:39.829	1	(x -48.843751, y -26.30736)
BR-101	2017-12-18 18:24:38	1	(x -48.870387, y -26.320411)
R. Colon	2017-12-14 15:56:43.759	1	(x -48.873411, y -26.302507)
R. Min. Calógeras	2017-12-14 17:35:39.829	1	(x -48.843751, y -26.30736)
...	...	...	...
...	...	...	...

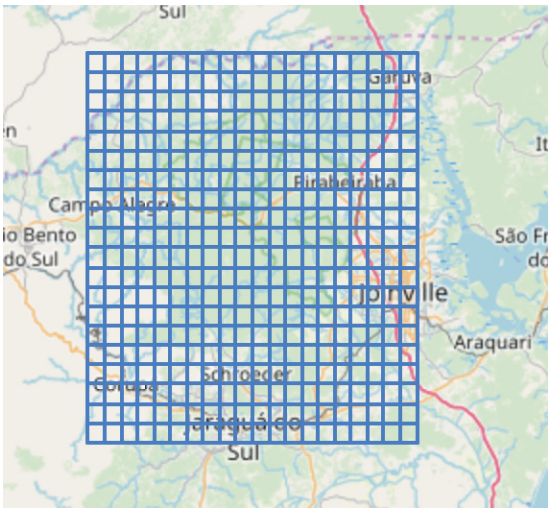
Figura 5.1: Armazenamento Relacional

Comparado com a base *Waze*, a Base *Waze-CG*, além de considerar apenas os atributos de interesse, agrupa os registros pela sua proximidade espacial, ao utilizar a coluna *IdCG* para definir um índice clusterizado dos dados espaciais.

A consulta executada na Base *Waze-CG* segue os seguintes passos:

Parâmetros	
Linha	20
Coluna	20
Intervalo de Tempo	365 dias
LatSE	-48,83
LongSE	-26,5
LatID	-49,3
LongID	-26
Tamanho Bloco	8KB
Slots Header	180

(a) Parâmetros



(b) Área de Interesse dividida em CGs

Figura 5.2: Parâmetros de Entrada e Área de Interesse

I) Identificação de quais são os eventos envolvidos na consulta. Por exemplo, para a consulta "Quantos engarrafamentos e alertas ocorreram na rua X, de 2017 a 2018?" apenas são considerados os eventos de engarrafamento e de alerta.

II) Identificação do período da consulta, para filtragem de registros. Por exemplo na consulta, o período é de 2017 a 2018.

III) Resultado da consulta com a junção dos registros entre os tipos de evento que ocorreram na mesma CG. Por exemplo na consulta, os eventos de engarrafamento e alerta serão combinados caso tenham ocorrido na mesma CG.

5.3 EXPERIMENTOS

Esta Seção apresenta os experimentos, o ambiente no qual eles foram executados, a fase de criação do MIDET, duas consultas e seus resultados. Para a realização dos experimentos foram desenvolvidas duas consultas espaço-temporais.

5.3.1 Ambiente de Experimentos

As consultas SQL foram realizadas no Postgresql versão 11.5 e sua extensão Postgis. Para a conexão com a linguagem C e o Postgres foi utilizada a biblioteca libpq.

Os parâmetros foram definidos como relacionados na Figura 5.2a. O número de linhas e colunas foi definido como L=20 e C=20, formando uma matriz de CGs com 400 células. O intervalo de tempo foi pré-definido com 365 dias. A área de entrada definida para o experimentos foi de (-48.83, -26,5) a (-49.3, -26) que correspondem a matriz (latSE, LongSE) e (latID, LongID). A área corresponde a Figura 5.2b. O tamanho máximo de um bloco foi definido como 8 *KiloBytes*(KB), por ser o mesmo tamanho da paginação utilizada pelo Postgres. O tamanho do *header* foi definido para conter 180 slots, ou seja, cada bloco contém no máximo 180 registros.

5.3.2 Geração da Base

Esta Seção apresenta a comparação entre a Base *Waze-CG* e o MIDET com relação aos tempos de criação e ao tamanho da base.

Base	MIDET	Base Waze	Base Waze-CG
B20	39s	1m 33s 990ms	8m 31s 591ms
B30	55s	2m 39s 70ms	12m 46s 598ms
B40	1m 12s	3m 3s	18m 24s 201ms
B50	1m 39s	3m 36s	23m 30s 195ms
B100	3m 5s	6m 44s	43m 24s 496ms

Tabela 5.2: Tempo de criação do MIDET e da base Waze-CG

5.3.3 Comparação do tempo de criação dos modelos de armazenamento

Esta Seção apresenta a comparação do tempo de criação entre os modelos de armazenamento. A Tabela 5.2 e o gráfico (Figura 5.3) apresentam os tempos de criação dos registros no MIDET, na Base Waze e a Base Waze-CG.

É possível observar que o MIDET, em comparação com a base *Waze*, apresenta uma diminuição de tempo. Este tempo consiste no tempo total de execução do MIDET, com as duas fases de criação: *criação da matriz* de CGs, criação dos arquivos de registros de evento e da indexação com a tabela bitmap e as tabelas de vetor de bloco. O tempo da base *Waze* consiste na inserção de registros no banco Postgres dos três tipos de evento.

Em comparação com a base *Waze-CG*, o tempo de inserção do MIDET apresenta-se menor (Tabela 5.2). O tempo da base *Waze-CG* é constituído pela primeira fase de criação do MIDET (criação da matriz de CGs), pela leitura da base *Waze* no programa em linguagem C, pela obtenção dass CGs e distribuição dos pontos em registros, caso necessário, armazenamento dos registros no banco Postgres e a clusterização do índice sobre as CGs. Um dos objetivos do MIDET é que o método tenha um baixo custo de inserção. Este objetivo é atingido com a criação de *blocos* no arquivo pois a medida que estes blocos são preenchidos e armazenados, os endereços são inseridos no vetor de bloco, sem necessidade de clusterização. Ao contrário da base Waze-CG, que necessita ser clusterizada e isto ocorre somente depois de serem gravados todos os eventos, o que implica em acréscimo do tempo de processamento.

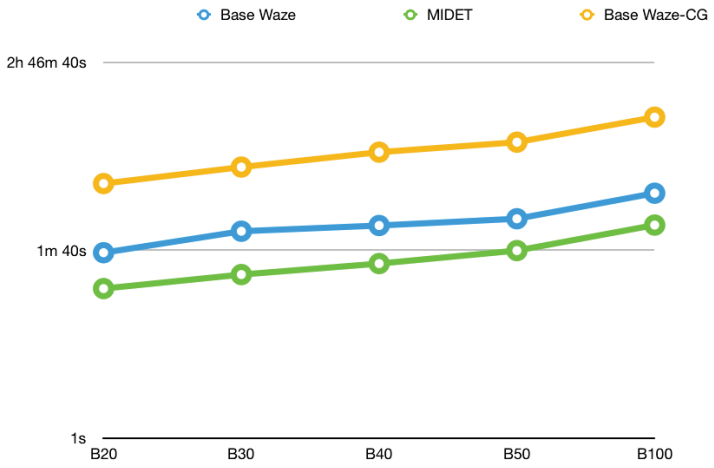


Figura 5.3: Gráfico dos tempos de criação

Base	Tam bitmap (KB)	Tam vetor bloco Alerta (KB)	Tam vetor bloco Engarrafamento (KB)	Tam vetor bloco Irregularidade (KB)
B20	16	352	480	56
B30	16	544	688	64
B40	16	704	928	72
B50	16	872	1152	88
B100	16	3432	4632	304

Tabela 5.3: Tamanho da base com Tabela de Bitmap e Tabelas de Vetores de Bloco

5.3.4 Tamanho da Base Waze-CG e dos elementos do MIDET

Esta Seção apresenta o tamanho dos elementos do MIDET comparados com a Base *Waze-CG*.

A Tabela 5.3 representa o tamanho das tabelas de bitmap e de vetores de bloco para cada tipo de evento no MIDET, armazenadas no Postgres. A tabela bitmap se mostra constante pois a quantidade de dados representados não aumentou o suficiente para precisar de mais páginas contendo dados relacionais (*relpages*) que fornecem uma área de espaço em disco contínuo para as tabelas com um melhor desempenho, mantendo a flexibilidade na alocação e na colocação de dados no disco [PostgreSQL (2020)]. Já as tabelas de vetor de bloco, para todos os tipos de evento cresce em proporção os número de blocos gerados.

As Tabelas 5.4, 5.5 e 5.6 apresentam, para cada tipo de evento e porcentagem da Base Waze, os tamanhos dos elementos do MIDET. A segunda coluna apresenta os tamanhos de cada header. Os testes foram gerados para as bases B20, B30, B40, B50 e B100. A terceira coluna (*Num blocos*) apresenta a quantidade de blocos gerados pelo MIDET. A quarta coluna (*Tam total headers*) corresponde ao tamanho do *header* para todos os blocos, ou seja, o número de blocos (*Num blocos*) multiplicado pelo tamanho do *header* (*Tam header*). A coluna (*MIDET Postgres*) apresenta o tamanho da tabela de bitmap e da tabela do vetor de bloco do tipo de evento. Por exemplo, o tamanho do MIDET Postgres na tabela de alerta é igual ao tamanho da tabela bitmap e do vetor de bloco de alerta. A coluna *Tam MIDET Total* corresponde ao tamanho total do arquivo do MIDET, juntamente com o tamanho do MIDET Postgres. Por exemplo, o tamanho total do MIDET na tabela de alerta é igual ao tamanho do arquivo de alerta, somado ao tamanho do MIDET postgres de alerta. A coluna *Tam Waze-CG* é o tamanho da tabela da Base *Waze-CG*. A coluna *% MIDET/ Waze-CG* indica a percentual do espaço ocupado pelo MIDET total, em relação a Base *Waze-CG*, considerando somente os espaços ocupados pelos registros.

O tamanho do header foi padronizado para permitir uma melhor análise da quantidade de blocos gerados e do tamanho do arquivo gerado pelo MIDET. Como por exemplo, nas Tabelas 5.4, 5.5 e 5.6 há um crescimento na quantidade de blocos correspondente ao aumento na quantidade de registros.

Para a análise do espaço utilizado pelos registros propriamente ditos, é utilizada a coluna *Tam MIDET só regs* que foi obtida retirando-se do tamanho total do arquivo (*Tam MIDET total*) o tamanho dos *headers* (*Tam Total headers*). Como resultado da comparação, é possível concluir que, para o tipo *alerta* o arquivo ocupa menor espaço que a Base *Waze-CG* e para os demais tipos, houve acréscimo. Isso se justifica em razão da informação da geometria de alerta ser constituída por somente um ponto e os outros tipos por um conjunto de pontos, o que pode resultar na divisão de pontos em múltiplos registros.

ALERTA	Tam header (Bytes)	Num blocos	Tam Total headers (MB)	MIDET Posgres (MB)	Tam MIDET Total (MB)	Tam Waze-CG (MB)	% MIDET/ Waze-CG
B20	470	7503	3,53	0,35	60,07	77	78 %
B30	470	11905	5,60	0,54	95,46	123	78 %
B40	470	15543	7,31	0,70	124,72	160	78 %
B50	470	19427	9,13	0,87	155,99	200	78 %
B100	470	38771	18,22	3,43	313,15	399	78 %

Tabela 5.4: Tamanho dos elementos do MIDET para o tipo alerta e a Base Waze-CG

ENGARR	Tam header (Bytes)	Num blocos	Tam Total headers (MB)	MIDET Posgres (MB)	Tam MIDET Total (MB)	Tam Waze-CG (MB)	% MIDET/ Waze-CG
B20	470	10473	4,92	0,48	77,30	68	114 %
B30	470	15228	7,16	0,69	112,80	100	113 %
B40	470	20866	9,81	0,93	154,64	138	112 %
B50	470	26067	12,25	1,15	193,17	172	112 %
B100	470	52524	24,68	4,63	391,55	345	113 %

Tabela 5.5: Tamanho dos elementos do MIDET para o tipo engarrafamento e a Base Waze-CG

IRREG	Tam header (Bytes)	Num blocos	Tam Total headers (MB)	MIDET Posgres (MB)	Tam MIDET Total (MB)	Tam Waze-CG (MB)	% MIDET/ Waze-CG
B20	470	569	0,27	0,06	4,07	3,27	124 %
B30	470	762	0,36	0,06	5,48	4,34	126 %
B40	470	1092	0,51	0,07	7,79	6,09	128 %
B50	470	1320	0,62	0,09	9,40	7,42	127 %
B100	470	2623	1,23	0,30	18,92	15,00	126 %

Tabela 5.6: Tamanho dos elementos do MIDET para o tipo irregularidades e a Base Waze-CG

5.3.5 Processamento de Consultas

Esta Seção apresenta as consultas realizadas. Com o objetivo de comparação do desempenho de consultas espaço-temporais, foram realizadas estas consulta em *3* plataformas diferentes: na base *Waze*, no MIDET e na Base *Waze-CG*.

Consulta 1

A primeira consulta foi definida para analisar o impacto da utilização de bitmaps com o MIDET para consultas com junção de tipos diferentes de evento de trânsito e a utilização do índice Cg_id na base Waze-CG. "Quais ruas tiveram eventos de engarrafamento e alerta, que ocorreram exatamente em um mesmo ponto, de uma rua, nos primeiros 7 dias de 2018?"

Consulta na base *Waze*.

A consulta SQL no banco de dados na base *Waze* é apresentada a seguir.

```
SELECT i.street
FROM waze.jams i,waze.alerts j
WHERE i.pub_utc_date > '2017-12-31 23:59:59'
        and j.pub_utc_date > '2017-12-31 23:59:59'
        and i.pub_utc_date < '2018-01-08 00:00:00'
        and j.pub_utc_date < '2018-01-08 00:00:00'
        and i.street=j.street
        and st_Intersects(i.geom,j.geom);
```

Consulta no MIDET.

A partir do esquema apresentado na Seção 4.1.4, foram identificados os eventos de engarrafamento e alerta, que aconteceram na primeira semana de 2018. Foi realizada a consulta na Tabela Bitmap com a utilização de SQL e libpq, realizando a conjunção entre os bitmps de cada tipo de evento, encontrando as CGs que são comuns aos dois evento. Em seguida, é feita a consulta nas Tabelas de Vetor de Bloco de cada tipo de evento envolvido, obtendo os blocos e endereços de cada CG comum que devem ser pesquisados nos arquivos. Para realizar a intersecção dos tipos de evento (alerta e engarrafamento) é criado um vetor de registros. É iniciada a leitura dos aquivos de alertas correspondentes, utilizando fseek e fread. Para cada CG, é criado um vetor com os registros de alerta da CG. Este vetor é comparado com os registros de engarrafamento da mesma CG, que são lidos diretamente dos arquivos de engarrafamento. O vetor evita a releitura dos registros de alerta da mesma CG, pois é reutilizado para comparar com registros de engarrafamento. Se o nome das ruas for igual, é feita a verificação do *timestamp*, e somente assim, é feita a leitura do restante dos elementos daquele registro de engarrafamento (data de publicação, quantidade de pontos e geometria) e posteriormente, é verificada a existência da intersecção entre os pontos.

Consulta na Base Waze-CG.

Para realizar a consulta na Base Waze-CG foi utilizado o postgis e sql com a função *ST_Intersects* e o sql no postgres com os filtros de nome de registro e filtros temporais. Foi também utilizado o índice *clusterizado* sobre a coluna de CGs. A consulta utilizada diretamente no banco de dados é apresentada a seguir.

```
SELECT i.nome
FROM registrosaler i,registrosjam j
where i.id_cg=j.id_cg
        and i.pub_utc_date > '2017-12-31 23:59:59'
        and j.pub_utc_date > '2017-12-31 23:59:59'
        and i.pub_utc_date < '2018-01-08 00:00:00'
        and j.pub_utc_date < '2018-01-08 00:00:00'
```

Base	Base Waze	MIDET	Base Waze-CG	Resultado
B20	41s 995ms	37s	33s 47ms	789
B30	1min 32s	1min 23s	1min 2s	15844
B40	2min 25s	1min 30s	1min 40s	34075
B50	3min 7s	2min 4s	2min 30s	573543
B100	5min 12s	2min 35s	3min 56s	1012052

Tabela 5.7: Resultados Consulta 1

```
and i.nome=j.nome
and st_Intersects(i.geom,j.geom);
```

Resultado da Consulta 1

A Tabela 5.7 e a Figura 5.4 apresentam os resultados da consulta 1 a partir das bases B20-B50 e B100. A coluna *Base Waze* mostra o tempo de processamento da consulta na base *Waze*. A coluna *MIDET* representa o tempo de consulta nos registros armazenados pelo MIDET, incluindo o tempo de busca na *Tabela bitmap e Tabelas de vetor*. Já a coluna *Base Waze-CG* demonstra o tempo da consulta na Tabela *Waze-CG*. A coluna *Resultado* mostra a quantidade de registros resultantes da consulta em cada Base de B20-B50 e B100.

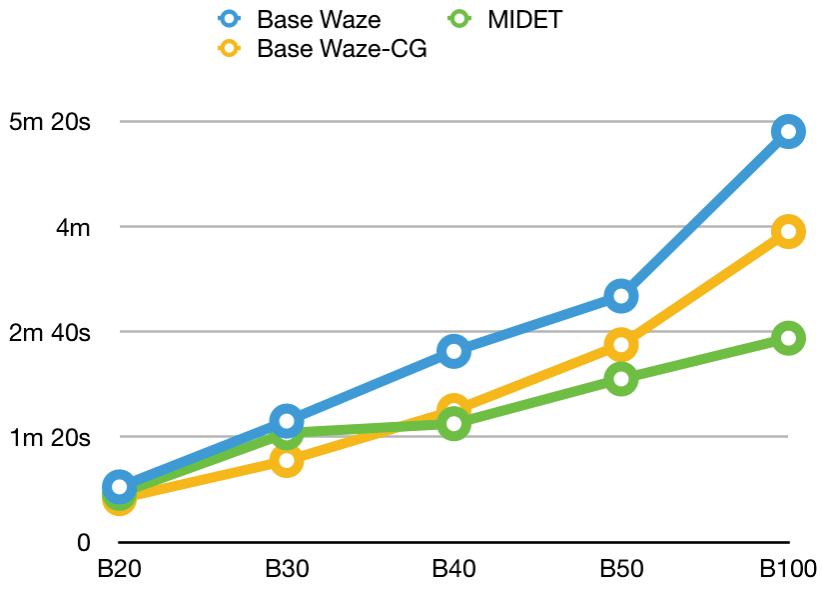


Figura 5.4: Gráfico Consulta 1

O processamento do MIDET é composto pelo tempo de busca pela biblioteca libpq no Postgres na tabela bitmap e nas tabelas de vetor de bloco de alerta e vetor de bloco de engarrafamento e, o tempo de consulta nos registros armazenados pelo MIDET, utilizando fseek e fread. O tempo de busca no Postgres equivale a apenas, em média *96,6 ms*.

É possível observar que ambos os modelos de armazenamento (MIDET e base *Waze-CG*) apresentam vantagem no tempo de processamento da consulta com junção com relação a base *Waze*. Para a base *Waze-CG* a vantagem é dada por realizar a indexação geográfica (coluna CG) com um índice clusterizado. O que permite a filtragem de registros, pois os dois tipos devem estar em uma mesma CG e somente asssim, é feita a junção. Já a vantagem do MIDET com relação a base *Waze* é dada pela utilização de CGs e bitmaps relacionados a eles. Com o resultado da

conjunção dos bitmaps é possível diminuir o espaço de busca, uma vez que os registros só são acessados se pertencem às CGs identificados na conjunção do bitmap.

A base Waze-CG tem o processamento inicialmente mais eficiente que o MIDET por utilizar SQL para o processamento, assim, fazendo uso do otimizador SQL. Porém, os resultados dos testes demostram que à medida que a quantidade de dados aumenta (a partir da B40 na Figura 5.4), a vantagem da utilização do MIDET cresce, em razão do agrupamento em subconjuntos pela proximidade espaço-temporal (*blocos*) que permite remover do espaço de busca o subconjunto que não pertence à(s) CG(s) identificado(s).

Consulta 2.

A segunda consulta foi definida para identificar o impacto, em uma consulta espacial, da utilização de blocos definidos pelas CGs pelo MIDET e a utilização da tabela de CG e o índice id_CG pela base Waze-CG. A consulta 2 é uma consulta espaço-temporal "Qual é a quantidade de registros de engarrafamentos no mês de outubro de 2017 em uma área de interesse informada?"O tamanho da área de interesse é de 25 km2, para a consulta é usada a região ilustrada na Figura 5.5 na cidade de Joinville.

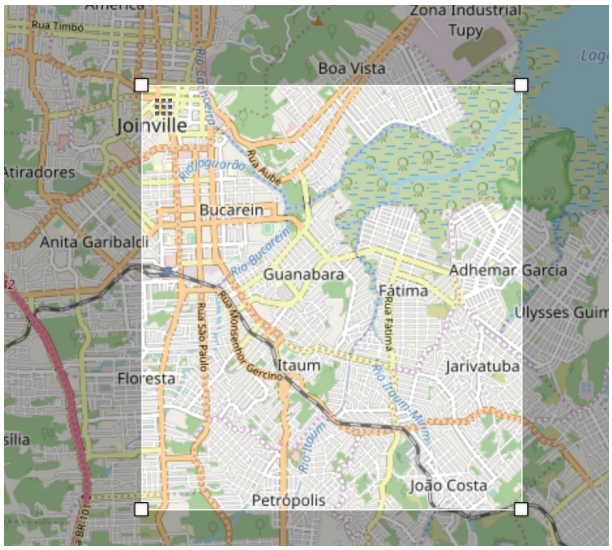


Figura 5.5: Área de Interesse

Consulta na base *Waze*.

A consulta SQL diretamente no banco de dados na base *Waze* é apresentada a seguir.

```
SELECT COUNT(street)
FROM waze.jams
WHERE pub_utc_date > '2017-09-30 23:59:59'
        and pub_utc_date < '2017-11-01 00:00:00'
        and st_Intersects(geom,
        'POLYGON((-48.85 -26.35,-48.80 -26.35,-48.80 -26.3,-48.85 -26.3,-48.85 -26.35),
        (-48.85 -26.35,-48.80 -26.35,-48.80 -26.3,-48.85 -26.3,-48.85 -26.35)');
```

Consulta no MIDET.

A partir do esquema apresentado na Seção 4.1.4, foi identificado o evento do tipo engarrafamento para o mês de outubro de 2017. Foi realizada a consulta SQL pela biblioteca libpq na Tabela Bitmap e nas Tabelas de Vetor de Bloco, obtendo os blocos e endereços que devem ser pesquisados para os eventos de engarrafamento, selecionando somente as CGs que pertencem à área de interesse a partir da Tabela CG. É feita a leitura dos arquivos correspondentes (utilizando fseek e

Base	Base Waze	MIDET	Base Waze-CG	Resultado
B20	3s 29ms	2s 86ms	1s 192ms	516
B30	5s 53ms	2s 935ms	2s 10ms	755
B40	7s 521ms	3s 116ms	3s 814ms	966
B50	9s 106ms	4s 68ms	4s 904ms	1009
B100	14s 980ms	6s 875ms	8s 200ms	2737

Tabela 5.8: Resultados Consulta 2

fread) e para cada registro, é utilizada uma função implementada em linguagem C que define se este pertence à área de interesse e é verificado se o seu *timestamp* encontra-se no intervalo dado. Caso o registro esteja dentro da área e do intervalo, é adicionado no contador da consulta.

Consulta na base *Waze-CG*.

Para realizar a consulta na base *Waze-CG* foram utilizados e os passos descritos na Seção 5.2.1, com a função *ST_Intersects* e o sql no postgres com os filtros temporais. Foi também utilizado o índice *clusterizado* sobre a coluna de CGs. A consulta na base *Waze-CG* é apresentada a seguir.

```
SELECT COUNT(nome)
FROM registrosjam
WHERE id_cg IN (SELECT i.cg_id
                    FROM cg_id i
                    WHERE ST_Intersects(geom,
                    'POLYGON((-48.85 -26.35,-48.80 -26.35,-48.80 -26.3,-48.85 -26.3,-48.85 -26.35),
                    (-48.85 -26.35,-48.80 -26.35,-48.80 -26.3,-48.85 -26.3,-48.85 -26.35))'))
and pub_utc_date > '2017-09-30 23:59:59'
and pub_utc_date < '2017-11-01 00:00:00'
and ST_Intersects(geom,
'POLYGON((-48.85 -26.35,-48.80 -26.35,-48.80 -26.3,-48.85 -26.3,-48.85 -26.35),
(-48.85 -26.35,-48.80 -26.35,-48.80 -26.3,-48.85 -26.3,-48.85 -26.35))');
```

Resultado Consulta 2

A Tabela 5.8 e a Figura 5.6 apresentam os resultados da consulta 2 a partir dos resultados nas bases B20-B50 e B100. A coluna *Base Waze* mostra o tempo de processamento da consulta na base *Waze*. A coluna *MIDET* representa o tempo de consulta nos registros armazenados pelo MIDET, incluindo o tempo de busca na *Tabela bitmap e Tabelas de vetor*. Já, a coluna *Base Waze-CG* demonstra o tempo da consulta na Tabela *Waze-CG*. A coluna *Resultado* mostra a quantidade de registros resultantes da consulta em cada Base de B20-B50 e B100

O processamento do MIDET é composto pelo tempo de busca pela biblioteca libpq no Postgres na tabela bitmap e na tabela de vetor de bloco de engarrafamento e, o tempo de consulta nos registros armazenados pelo MIDET, utilizando fseek e fread. O tempo de busca no Postgres equivale a apenas, em média *96,4 ms*.

É possível observar que ambos os modelos de armazenamento (MIDET e base *Waze-CG*) apresentam vantagem no tempo de processamento da consulta espacial com relação a base *Waze*. Para a base *Waze-CG* a vantagem é dada por realizar a indexação geográfica (coluna CG) com um índice clusterizado. A sub-consulta identifica as CGs que intersectam a área de interesse, pesquisando na Tabela CG que possui apenas L * C elementos. Reduzindo o espaço de busca pois permite filtrar a Tabela de registros de engarrafamento selecionando apenas os registros dessas CGs. Neste caso, a Tabela *CG* possui um tamanho de 400 linhas (128 KB) e a sub-consulta retorna apenas 3 CGs que intersectam a área de interesse. Assim, esta filtra os registros de

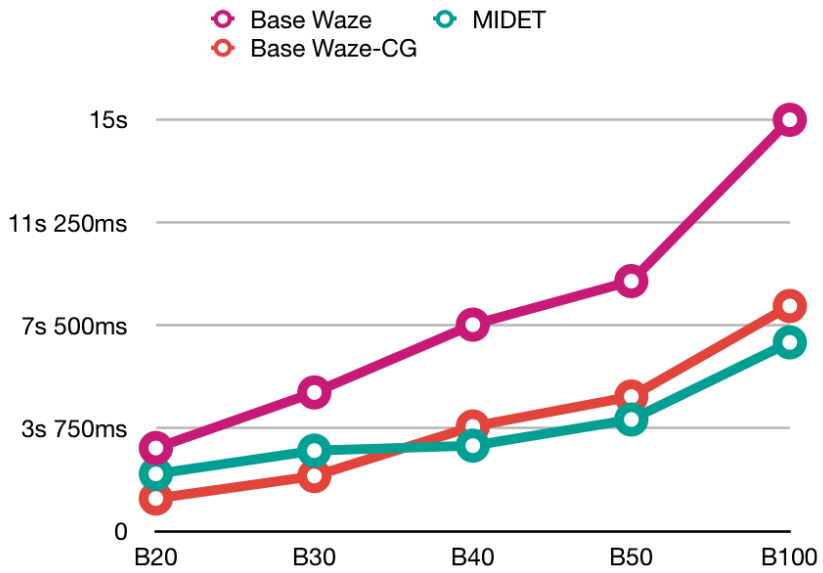


Figura 5.6: Gráfico Consulta 2

engarrafamento e diminui a quantidade de intersecções a serem avaliadas. Já, a vantagem do MIDET com relação a base *Waze* é dada pela utilização de CGs e bloco, pois os registros são filtrados pelas CG.

A base Waze-CG tem o processamento inicialmente mais eficiente que o MIDET por utilizar SQL para o processamento, assim, fazendo uso do otimizador SQL. Porém, os resultados dos testes demostram que à medida que a quantidade de dados aumenta (a partir da B40 na Figura 5.4), a vantagem da utilização do MIDET cresce, em razão do agrupamento em subconjuntos pela proximidade espaço-temporal (*blocos*) que diminui o espaço de busca, pois faz a pesquisa bloco por bloco e não, registro a registro.

5.4 CONSIDERAÇÕES

Neste Capítulo 5 foi descrita a implementação do MIDET e os experimentos realizados. Foram descritos o ambiente de implementação e as bibliotecas utilizadas. Foi apresentada a abordagem de armazenamento utilizada para a comparação do desempenho com o MIDET. Os experimentos com duas consultas foram analisados.

Os testes demonstraram que o uso de CGs, em ambos os modelos de armazenamento (MIDET e base *Waze-CG*) é vantajoso, pois diminui o tempo de processamento de consultas com relação à base *Waze*.

Para as base B20 e B30, a base Waze-CG apresenta um processamento de consulta mais rápido que o MIDET. Porém, para todos os tamanhos o tempo de construção da base Waze-CG é significativo e em média 15 vezes maior que o tempo de construção do MIDET. Para testes com bases maiores, o MIDET apresenta vantagem, também, no tempo de processamento de consultas por utilizar bitmaps e o agrupamento de registros pela sua proximidade espaço-temporal.

O Capítulo 6 conclui a dissertação.

6 CONCLUSÃO

Este Capítulo apresenta as considerações finais desta dissertação e trabalhos futuros. A Seção 6.1 apresenta as considerações e a Seção 6.2 apresenta Trabalhos Futuros.

6.1 CONSIDERAÇÕES

Esta dissertação utilizou como base a hipótese de *ser possível otimizar consultas de eventos de trânsito, agrupando os registros por proximidade espaço-temporal, baseado em uma tesselação sobre a área geográfica de interesse*.

Foram testadas duas formas de armazenamento de dados baseado em tesselação, na qual se definem células geográficas (CGs): (1) em uma base relacional incluindo a CG da localidade como atributo adicional e geração de um índice clusterizado sobre a CG (Base Waze-CG) e (2) com a criação de um ***M**étodo para **I**ndexação de **E**ventos de **T**rânsito (MIDET)*. O MIDET adota um armazenamento misto, com arquivos compostos por blocos de eventos ocorridos em uma mesma CG e um mesmo período, e uma base relacional para a indexação dos dados e a utilização de bitmaps.

Os testes foram realizados sobre 3 modelos: o MIDET, a Base Waze-CG e a base original orientada a eventos (Base Waze). Foram criadas as bases (*B20*), (*B30*), (*B40*), (*B50*) e (*B100*) que correspondem a 20%, 30%, 40%, 50% e 100% da base Waze. Os testes efetuados mostram vantagem com relação à Base Waze, quanto ao tempo de processamento das consultas espaço-temporais para ambos os modelos de armazenamento (MIDET e Waze-CG). A vantagem para a base Waze-CG justifica-se pela indexação geográfica sobre a CG com um índice clusterizado. Já, a vantagem do MIDET com relação à base Waze se justifica em razão da indexação geográfica (CGs) e também pelo agrupamento dos registros em blocos. O processamento de consultas espaço-temporais utilizando SQL, nas bases Waze-CG B20 e B30, é mais eficiente que o do MIDET em razão do uso do otimizador SQL. Porém, para testes com as bases maiores, o MIDET apresenta vantagem no tempo de processamento de consulta. Essa vantagem é dada em razão de utilizar bitmaps e blocos, diminuindo o espaço de busca.

As duas consultas espaço-temporais executadas com o MIDET apresentaram um desempenho superior com relação ao tempo de processamento, destas mesmas consultas na Base Waze. Portanto, os testes realizados demostraram ser vantajosa a utilização do agrupamento de eventos pela sua proximidade espaço-temporal e a utilização de bitmaps.

A implementação do MIDET permitiu que fosse definida uma abordagem de agrupamento de registros pela sua proximidade espaço-temporal e pelo tipo de evento (engarrafamento, alerta e irregularidade). Por meio de um método para indexação de eventos de trânsito baseado no agrupamento de registros. Composto por um modelo de armazenamento misto com arquivos e uma base relacional. Os arquivos são compostos por agrupamentos de registros ocorridos em uma mesma CG e em um mesmo período. A base relacional é utilizada para a indexação dos dados e uso de bitmaps. O que descarta a necessidade de avaliar registro a registro durante consultas espaço-temporais. Os estudos experimentais validaram o MIDET através da comparação da performance em consultas espaço-temporais de formas de armazenamento relacionadas.

Cabe destacar que a forma como o índice foi modelado pode comprometer a sua dinamicidade geográfica pois esta deve ser definida a priori. O que necessita que o usuário defina um tamanho adequado, com a quantidade de linhas e colunas. O MIDET também desconsidera a densidade de entradas de acordo com a localização geográfica e as variações de densidade

nas regiões em diferentes instantes de tempo. Para encontrar o tamanho ótimo para a grade geográfica de um dado conjunto de dados, ou seja, o melhor tamanho, é necessária a realização de uma análise experimental. Além disso, uma possível solução para o problema de densidade dos dados seria a utilização de divisões dentro de CGs, dividindo as que possuírem alta densidade. Caso fosse implementada a sub-divisão, seria necessário manter a informação de quais CGs foram divididas. Além disso, seria preciso modificar a criação dos bitmaps, podendo utilizar duas hierarquias de bitmap, um bitmap para as CGs completas e um bitmap para cada CG que tenha sido sub-dividida. Caso uma CG tenha sido dividida em um dado período de tempo, seria necessário que o bitmap criado para ela fosse criado para todos os períodos existentes para realizar a conjunção em consultas que utilizam diversos períodos. Porém, essa abordagem aumentaria o tamanho da representação do MIDM. Na modelagem atual, que possui as CGs de tamanho regular, a grade não é efetivamente criada, apenas calculada a partir do tamanho definido. Assim, é possível saber se um evento pertence a uma CG somente pelo cálculo.

6.2 TRABALHOS FUTUROS

Alguns trabalhos futuros para o MIDET são definidos a seguir:

- Criar bitmaps para outros elementos, como por exemplo ruas onde tiveram engarrafamento. O que pode reduzir o tempo de uma busca, por exemplo : "A Rua Florianópolis teve engarrafamento na primeira semana de 2018?". Para responder à consulta seria necessário apenas consultar o bitmap deste período e desta rua.

- Utilizar entrada de dados em fluxo. O MIDET foi modelado levando em consideração a possibilidade de estendê-lo para entrada de fluxo. A medida que seria feita a inserção, novos períodos seriam criados, juntamente com novos arquivos e vetores de bloco.

- Paralelizar o processamento da leitura dos registros nos arquivos. O processamento de consultas no MIDET foi implementado de uma maneira que fosse possível paralelizá-lo. Seria possível realizar a leitura de cada bloco no arquivo paralelamente.

- Implementar novas consultas para realização de testes. Duas consultas espaço-temporais foram testadas nessa dissertação, a primeira com o intuito de testar o impacto da utilização de bitmaps para realizar junções e a segunda para determinar o impacto da utilização de blocos. Ao implementar novas consultas seria possível verificar o impacto da estrutura em diversas operações de busca.

- Inserir o período dos registros na Tabela Vetor de Bloco. Como na dissertação todos os registros de eventos pertencem a apenas um período, o impacto da utilização de diversos períodos não foi testado.

- Realizar testes com diversos períodos de tempo, modificando as áreas de interesse, com diferentes quantidades de linhas e colunas e diferentes tamanhos do bloco. Nesta dissertação, a área de interesse, o tamanho de linhas, colunas e do bloco foram fixados em apenas um valor. Testes adicionais variando estes parâmetros permitiriam definir a relação entre cada um e sugerir ao usuário tamanhos que possam melhorar a performance.

- Realizar a otimização do armazenamento da estrutura do arquivo. Por exemplo, movendo os campos fixos para o início do bloco e os campos variáveis para o final.

Referências

Besenbacher, S., Mailund, T., Westh-Nielsen, L. e Pedersen, C. N. S. (2004). RBT — a tool for building refined Buneman trees . *Bioinformatics*, 21(8 2005):1711–1712.

Brundu, F. G., Patti, E., Osello, A., Giudice, M. D., Rapetti, N., Krylovskiy, A., Jahn, M., Verda, V., Guelpa, E., Rietto, L. e Acquaviva, A. (2016). IoT Software Infrastructure for Energy Management and Simulation in Smart Cities. Em *IEEE Transactions on Industrial Informatics*, página 99.

Buneman, P., Khanna, S., Tajima, K. e Wang-Chiew (2004). Archiving Scientific Data. *ACM Transactions on Database Systems (TODS)*, 29(1):2–42.

Doraiswamy, H., Vo, H. T., Silva, C. T. e Freire, J. (2016). A GPU-Based Index to Support Interactive Spatio-Temporal Queries over Historical Data. Em *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, Helsinki, Finland.

ExtremeDB (2018). *ExtremeDB Documentation*. McObject, 8.0 edition.

Gan, G., Ma, C. e Wu, J. (2007). *Data Clustering: Theory, Algorithms, and Applications*. SIAM.

Imawan, A., Putri, F. e Kwon, J. (2015). TiQ: A Timeline query processing system over Road Traffic Data. Em *2015 IEEE International Conference on Smart City*, Chengdu, China.

Keawpibal, A., Wattanakitrungroj, N. e Vanichayobon, S. (2012). Enhanced Encoded Bitmap Index for Equality Query. Em *2012 8th International Conference on Computing Technology and Information Management (NCM and ICNIT)*, Seoul, South Korea.

Keawpibal, N., Preechaveerakul, L. e Vanichayobon, S. (2016). HyBiX: A novel encoding bitmap index for space-an time-eficient query processing. Songkhla, Thailand.

Keawpibal, N., Preechaveerakul, L. e Vanichayobon, S. (2019). Optimizing Range Query Processing for Dual Bitmap Index. *Walailak Journal of Science and Technology*, 16(2):133–142.

Kleppmann, M. (2017). *Designing Data-Intensive Applications*. O'Reilly Media, Inc.

Krylovskiy, A., Jahn, M. e Patti, E. (2015). Designing a Smart City Internet of Things Platform with Microservice Architecture. Em *The 3rd International Conference on Future Internet of Things and Cloud (FiCloud)*, páginas 25–30, Rome, Italy.

Lemire, D., Ssi-Yan-Kai, G. e Kaser, O. (2018). Consistently faster and smaller compressed bitmaps with roaring. *Software: Practice and Experience*, 46:1547–1569.

Manolopoulos, Y., Nanopoulos, A., Papadopoulos, A. N. e Aristotle (2003). R-trees have grown everywhere.

Manolopoulos, Y., Nanopoulos, A., Papadopoulos, A. N. e Theodoridis, Y. (2006). *R-Trees: Theory and Applications*. Springer.

Oracle (2019). *Data Cartridge Developer's Guide*. Oracle, 19c edition.

Papadias, D., Tao, Y., Kalnis, P. e Zhang, J. (2001). Indexing Spatio-Temporal Data Warehouses. Hong Kong.

PostgreSQL (2020). *PostgreSQL 12.2 Documentation*. The PostgreSQL Global Development Group, 12 edition.

Savary, L., Wan, T. e Zeitouni, K. (2003). Spatio-temporal Data Warehouse Design for Human Activity Pattern Analysis. Em *PRISM Laboratory*, Versailles University.

Shahnawaz e Kannapiran, T. (2019). Indexing Issues in Spatial Big Data Management. Em *International Conference on Advances in Engineering Science Management Technology 2019*, Uttaranchal University, Dehradun, Uttarakhand, India.

Silva, R. (2002). Bancos de dados geográficos: Uma análise das arquiteturas dual (spring) e integrada (oracle spatial). Dissertação de Mestrado, Escola Politécnica da Universidade de São Paulo, São Paulo, SP.

Siqueira, T. L. L., Ciferri, R. R. e Times, V. C. (2008). I-DWE: Uma Estrutura de Indexação para Data Warehouse Espacial. Em *VII Workshop de Teses e Dissertações em Bancos de Dados*, Campinas.

Siqueira, T. L. L., Ciferri, R. R., Times, V. C. e de Aguiar Ciferri, C. D. (2009). A Spatial Bitmap-based Index for Geographical Data Warehouses. Em *Proceedings of the 2009 ACM symposium on Applied Computing*, Honolulu, Hawaii, USA.

Siqueira, T. L. L., de Aguiar Ciferri, C. D., Times, V. C. e Ciferri, R. R. (2012). The SB-index and the HSB-index: efficient indices for spatial data warehouses. *Geoinformatica*, 16(1):165–205.

SQLite (2018). *SQLite Documentation*. SQLite, 2.1.0 edition.

Streisky, M. e Borges, A. P. (2019). Aplicação de Algoritmo Para Recuperação de Casos No Problema de Condução de Trens de Carga. Monografia (Bacharel em Ciência da Computação), UTFPR (Universidade Tecnológica Federal do Paraná), Ponta Grossa, Brazil.

Vu, T. e Eldawy, A. (2018). R-Grove: Growing a Family of R-trees in the Big-Data Forest. Em *SIGSPATIAL*, Seattle, WA, USA.

Wang, J., Lin, Y. C. e S., S. (2017). An experimental study of bitmap compression vs. inverted list compression. Em *Proceedings of the 2017 ACM SIGMOD International Conference on Management of Data*, New York, NY.

Wattanakitrungroj, N. e Vanichayobon, S. (2006). Dual Bitmap Index: Space-Time Efficient Bitmap Index for Equality and Membership Queries. Em *2006 International Symposium on Communications and Information Technologies*, Bangkok, Thailand.

Wu, M.-C. e Buchmann, A. (2008). Encoded Bitmap Indexing for Data Warehouses. Em *Proceedings 14th International Conference on Data Engineering*, Orlando, FL, USA.