Geotecnologias na Agregação de Valor de Negócio dos Correios - Aplicação de Melhores Práticas em Grandes Volumes de Dados Para Estimativa de Melhor Ponto de Interesse

Por

Emilson Ribeiro Neto

Orientador:

Prof. Dr. Henrique LLacer Roig

Monografia apresentada como requisito parcial para a obtenção do título de Especialista em Geoprocessamento Ambiental pelo Programa de Pós-Graduação em GeoCiência da Universidade de Brasília.

UnB - Brasília - DF

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

RR484g

Ribeiro Neto, Emilson
Geotecnologias na Agregação de Valor de Negó cio dos Correios - Aplicação de Melhores Práticas em Grandes Volumes de Dados Para Estimativa de Melhor Ponto de Interesse / Emilson Ribeiro Neto; orientador Henrique LLacer Roig. -- Brasília, 2018.
72 p.

Monografia (Especialização - Geoprocessamento Ambiental) - Universidade de Brasília, 2018.

1. Banco de dados espacial. 2. Melhores práticas banco de dados espacial de grandes volumes. 3. Consultas performáticas em banco de dados espaciais. 4. Clusters espaciais. 5. PostgreSQL e PostGIS. I. Roig, Henrique LLacer , orient. II. Título.

EMILSON RIBEIRO NETO

GEOTECNOLOGIAS NA AGREGAÇÃO DE VALOR DE NEGÓCIO DOS CORREIOS - APLICAÇÃO DE MELHORES PRÁTICAS EM GRANDES VOLUMES DE DADOS PARA ESTIMATIVA DE MELHOR PONTO DE INTERESSE

Monografia apresentada como requisito parcial para a obtenção do título de Especialista em Geo-Processamento Ambiental pelo Programa de Pós-Graduação em GeoCiência.

Aprovado em 09 de Março de 2018

BANCA EXAMINADORA

Professor Dr. Henrique LLacer Roig - Presidente da Banca
Universidade de Brasília

Professor Dr. Gervásio Barbosa Soares Neto - Membro Convidado
Instituto Federal de Brasília

Professor Dr. Elder Yokoyama - Membro Efetivo
Universidade de Brasília

Dedico este trabalho à minha amada esposa Sâmia Machado Ribeiro e filhos Alice e Benajamin Ribeiro Ponce de Leon.

\/

Agradecimentos

Dedico e agradeço este trabalho primeiramente aquele que em todas as coisas tem me sustentado, dado conhecimento, sabedoria, aberto oportunidades e cuidado de mim e minha família, comprovando a cada dia seu imensurável amor, me sustentando em todas as necessidades e me feito encontrar no caminho pessoas maravilhosas. Ao Deus onisciente, onipresente e onipotente; ao único que é digno de receber a honra e a glória, a força e o poder. Ao Rei Eterno imortal, invisível mas real que é digno de todo louvor: Iavé; à Yeshua (Jesus) seu filho, o Cristo, e ao Espírito Santo que transformou minha vida e tem feito de mim um trabalho de aperfeiçoamento me tornando um ser humano melhor a cada dia por sua misericórdia.

Ao gerente Clediomar do Carmo Santos que, na época em que atuava em sua gerência, apoiou, batalhou e fez o possível para a realização deste curso; a minha atual gerente Thais Ribeiro dos Santos Pessoa que tem crido e apoiado meu trabalho nas atividades de Geo dos Correios; em especial ao colegas de trabalho Eduardo Henrique Soares Altgott que é um verdadeiro mostro (no bom sentido) nas questões técnicas de banco de dados e transferido seu precioso conhecimento com paciência e muito apoio; ao ex-chefe de departamento do DETEC Paulo Afonso Andrade de Santana que batalhou em conjunto com Clediomar intercedendo à Vice-Presidência corporativa para realizar este curso no interesse da ECT; a atual chefe de departamento Fabiana de Assunção Cruvinel Nascimento por acreditar e incentivar meu trabalho nos desafios de Geo.

Agradeco em especial ao meu orientador Prof. Dr. Henrique LLacer Roig que tem com muito empenho apoiado este trabalho e ao Prof. Dr. Paulo Roberto Menezes (Paulo Menezes) pela intensa dedicação, explêndida didática, notório saber, pela amizade que cultivamos, pelos almoços e cafezinhos que tomamos juntos.

Aos professores que tanto se empenharam nesta jornada Drª Tati Almeida, Drª Rejane Cicerelli, Drº Edilson de Souza Bias, Drº Edson Eyji Sano, Me. Alexandre Moreno Richwin Ferreira e Drª Maristela Holanda por todo empenho em nossa formação.

Agradeço a minha esposa Sâmia que tem me apoiado com sua imensa compreensão, paciência, amor e incentivo nos momentos de ausência em que estudava; e aos meus filhos Alice e Benjamin que têm me proporcionado imenso prazer e alegria no dia a dia, combustível perfeito para motivar meus trabalhos e minha vida.

Resumo

Este trabalho apresenta um caso de uso de aplicação de técnicas de alta performance em banco de dados espaciais de grande volume para extração de pontos de interesse aplicado na Empresa Brasileira de Correios e Telégrafos - ECT.

São abordadas técnicas de alta performance em consultas espaciais envolvendo operadores espaciais e estruturais em banco de dados, análise comparativa entre indexadores espaciais, aplicação de consultas espaciais utilizando índices com *"bounding boxes"* (caixas delimitadoras) como exemplo de melhores práticas, análise das funções de clusterização do PosGis e aplicação de consultas performáticas, também conhecidas como *"tuning"*, aplicadas com os conceitos de *"clusters"* espaciais para obtenção de pontos de interesses em bases de dados de enorme volume de dados além de funções geoestatísticas do Postgis.

O método utilizado envolveu escolha da origem de dados, aplicação de filtro de dados em banco de dados, aplicação de *"tuning"*, aplicação de filtros espaciais, aplicação de funções espaciais e validação dos resultados obtidos.

Os resultados obtidos atingiram 100% de todos os requisitos funcionais com tempo de execução dentro dos limites aceitáveis pelo negócio sem esgotamento dos recursos computacionais.

PALAVRAS-CHAVE: alta performance em banco de dados espaciais, tuning em banco espacial, PostGis, PostgreSQL, cluster espacial, operadores espaciais performáticos, índices espaciais, bounding boxes espaciais, filtros espaciais.

Abstract

This paper presents a case of use of high performance techniques in a bigest spatial databases to points of interest-POI extraction performed in the Brazilian Postal Office and Telegraph Company (ECT).

High-performance techniques in spatial queries involving spatial and structural operators in databases, comparative analysis between spatial indexes, application of spatial queries using indexes with bounding boxes examples like best practices, analysis of the PosGis clustering functions, application of performative queries, know as cluster, applied with clusters to obtain points of interest in databases with geostatistics functions in huge data volume are submited in this work.

The used method consist in the data source choose, database data filtering , database tuning, spatial filters applied, spatial functions applied and results validation.

All results approach reached 100% of all requirements with execution in a acceptable time by the business without exhaustion the computational resources.

KEYWORDS: high performance in spatial database, spacial database tuning techniques , PostGIS, PostgreSQL, spacial cluster, performatic spacial operators, spatial indexes, spatial bounding boxes, spatial filters.

Sumário

Lista de Figuras

Lista de Abreveaturas e Siglas

SGBD - *Data Base Management System* (Sistema Gerenciador de Banco de Dados)

SQL - *Structured Query Language* (Linguagem de Consulta Estrutura)

ECT - Empresa Brasileira de Correios e Telégrafos

SIG - Sistema de Informação Geográfica

1 Introdução

O uso de funções espaciais em bancos de dados de grande volume sem a aplicação de técnicas especiais para consultas em grandes volumes de dados tornam inviável, em termos computacionais ou comerciais, a obtenção de informações e agregação de valor ao negócio em tempo hábil e/ou com os recursos computacionais existentes. Isso ocorre seja por esgotamento de recursos computacionais, que são finitos (estouro de memória no banco de dados, elevado tempo de processo, esgotamento de área de armazenamento de dados, superutilização de processadores ou perda de resposta por sobrecarga de I/O), ou pelo tempo de resposta necessário ao atendimento corporativo podendo trazer danos ao negócio. Para extração de informações a partir de imensos volumes é necessário o uso de técnicas de otimização, conhecidas como *"tuning"*, que consistem no desenvolvimento de *scripts* em linguagem SQL - *Structured Query Language* (Linguagem de Consulta Estrutura) e ajustes nas estruturas de banco de dados a partir da ciência da carga computacional das funções espaciais e não espaciais utilizadas, torna-se um conhecimento indispensável para obtenção de resultados e agregar valor ao negócio na utilização de geoprocessamento.

1.1 Sobre este trabalho

Este trabalho abordará técnicas de otimização em consultas em banco de dados que viabilizem o processamento a partir de grande volume de dados, funções de clusterização em dados espaciais e o desenvolvimento de códigos em linguagem SQL para se obter pontos geocodificados de pontos de interesse (POI) a partir da coleta de pontos geocodificados no processo de entrega de objetos postais para cada endereço em território brasileiro.

A definição de grande volume de dados, neste trabalho, significa, aproximadamente 3TB de volume de dados geoespaciais, mais de 3 bilhões de registros de pontos espaciais geocodificados. Neste cenário o atingimento dos objetivos se dá pelo desenvolvimento de códigos para processamento com suporte de perspectiva de processamento de 52 bilhões de registros em menos de um ano;

Para se atingir o objeto foram aplicadas técnicas de *"tuning"*, que são um conjunto de consultas em SQL performáticas com indexadores espaciais associadas a mudanças das estruturas do modelo do banco de dados visando performance, uso de técnicas de clusterização espacial aplicada em geoprocessamento, que consiste no agrupamento dados que possuem dada correlação espacial para tomada de decisão, a partir de um algoritmo escolhido de acordo com estudo de caso e parâmetros definidos, com o objetivo a ser alcançar a seleção dos *"clusters"* com maior significação e, ao final, aplicação de funções de geoestatísticas para extração das informações de interesse do trabalho.

2 Do caso de uso dos Correios - ECT

O Correios é a única empresa brasileira presente em todo território nacional com capilaridade em todos os municípios brasileiros. Ela oferece, além de serviços postais, vários outros tipos de produtos com fins sociais tais como serviços bancários em localidades que não possuem bancos, sendo muitos dos serviços de cunho social (pagamento de programas sociais do Governo Federal como bolsa família, aposentadorias, seguro DPVAT e outros).

O Correios possui hoje 1.403 unidades em todo o Brasil sendo que destas 103 são agências fraqueadas e 1.300 são desde agências próprias, comunitárias, centros de distribuição, subsidiárias, permissionárias etc. A grande parte das suas unidades encontram-se geocodificadas dentro de um total aproximado de 38.400 pontos de interesse utilizados pela instituição.

A empresa possui como demanda institucional e estratégica diversos projetos que dependem de soluções de sistemas de informações geográficas - SIGs. Dentro dessas demandas o Correios possui quase 60 mil celulares com tecnologia de geolocalização (GPS) gerando dados espaciais. Em 2017 foram distribuídos 7 mil aparelhos, como piloto do projeto, enviando informações geográficas relativas ao ponto de localização do carteiro para controle de percorrida e baixa de objetos postais entregue. Em fevereiro de 2018 foram distribuídos 44.424 aparelhos e até marco 44.514 entrarão em operação. A partir das informações espaciais geradas pelos aparelhos os sistemas internos geram estatísticas e controle das percorridas, efetividade e eficiência das atividades de cada carteiro, assim como o controle pelo código de cada objetos postais (Sedex, PAC, carta registrada, malotes etc). Também há sistema de controle de frota viária que consome e produz informações espaciais com quase totalidade dos veículos gerenciando e controlando de roteirização e sensores diversos (abertura de porta, desvio de rotas etc).

Hoje o maior volume de dados espaciais é produzido pelo envio do ponto geolocalizado pelos celulares dos carteiros em uma periodicidade de cinco segundo ou 10 passos caminhados. São registrados os dados de percurso dos carteiros, do ponto de entrega no momento da baixa do objeto postal *in-loco* e em março será efetivada para todos os celulares a coleta das assinaturas dos recebedores para fins de auditoria e disponibilização aos clientes contendo a geolocalização. Em poucos dias, estima-se que até março de 2018, estaremos próximos de obter um quantitativo diário de 262,2 milhões de novos registros diários com pontos espaciais coletados pelos 44,5 mil aparelhos celulares em território brasileiro. Tal quantidade de registros significa uma estimativa anual de quase 63 bilhões de registros, somente da percorrida das atividades dos carteiros e 14TB de volume de dados. Há uma estratégia para diminuir a quantidade de registros transformando os milhares de pontos das percorridas em uma única multilinha de percorrida e expurgo dos pontos processados mas que não faz parte do escopo deste trabalho.

2.1 Das necessidades institucionais e problemas de geotecnologia

A ECT utiliza processos automatizados de roteirização de transporte e entrega com uso dos pontos geocodificados existentes, controle dos objetos postais e dos índices operacionais por geocontroles. O escopo deste trabalho visou oferecer uma solução para obtenção do melhor ponto de entrega geolocalizado, POI, dos endereços brasileiros a partir dos dados espaciais de entrega de objetos postais cruzados CEP, número do endereço e geolocalização a partir de um imenso volume de dados. Isso foi possível exclusivamente utilizando um conjunto de técnicas e conhecimentos descritos neste trabalho tais como indexação espacial, *"clusters"* espaciais e *"tuning"* no banco de dados geoespacial. Os mesmos resultados, em tese, podem se obtidos utilizando funções simples como de distância e centroide, porém, na prática, esgotam-se os quase sem-

pre recursos computacionais de memória e processamento do Sistema Gerenciador de Banco de Dados (*Data Base Management System* - SGBD) inviabilizando a obtenção dos resultados esperados.

A obtenção dos pontos geocodificados de cada endereço, os POIs, são insumos para roteirização de entrega/transporte, composição de uma base de dados qualitativa para efetivação dos projetos como Geomarketing e outros serviços definidos nos horizontes estratégicos da empresa.

2.2 Das tecnologias utilizadas na ECT

Existindo diversas tecnologias que envolvem geotecnologias, em termos de ferramentas SIG, a instituição utiliza as tecnologias ArcGis Server 10.3, Quantum GIS (Qgis), ferramenta institucional Geo-Editor de Mapas, SGBD principal PostgresSQL com extensão PostGis, SGBD Oracle SDO que armazena dados geográficos e diversos outros sistemas internos desenvolvidos pela própria corporação que trabalham com informações espaciais.

2.3 Do escopo do trabalho do caso de uso

O escopo deste trabalho consiste no desenvolvimento de rotina otimizada para grandes volumes de dados utilizando linguagem PgSQL, aplicando técnicas de *tuning*, para obter o melhor ponto de entrega dos endereços geocodificado a partir da base de dados corporativa dos correios. A base de dados utilizada será a do projeto Geo que possui todos os pontos de coordenadas geocodificados de baixas de objetos e de percorridas dos carteiros. A rotina deve atender regras definidas pelas áreas funcionais (questões de distância da linha geoespacial do CEP do objeto, significação dos dados e serem utilizados e outros parâmetros). Todos os dados obtidos são provenientes apenas do

SGBD tecnologia PosgreSQL com extensão PostGis.

3 Banco de dados e SGBD espacial

O conceito de banco de dados consiste na operação de uma coleção de dados armazenados, relacional, usados por um dado sistema [10]. O conceito de banco de dados não está intrinsecamente ligado a tecnologia da informação, porém esta área, aproveitando-se do conceito, desenvolveu sistemas gerenciadores de banco de dados, os SGBDs, que oferecem funcionalidades de fornecimento facilitado aos dados (armazenamento e recuperação) transparente em relação a questões de hardware do computador, com parâmetros de controles incluindo acesso. Os SGBDs também garantem integridade, confidencialidade, disponibilidade e relacionamentos; facilitados pelo uso de uma linguagem intuitiva e fácil assimilação de consultas estruturadas (Structured Query Language - SQL) que são utilizadas pela maioria das tecnologias (com exceção dos SGBDs não relacionados).

Os SGBDs foram inicialmente desenvolvidos para trabalhar com vários tipos de dados e arquivos por volta de 1960 e posteriormente, por volta de 1970 [**?**], foram aperfeiçoados à trabalhar com relações topológicas por meio de relacionamentos espaciais, sistemas de referências espaciais, dados de geometria e atributos geoespaciais , nascendo assim os SGBDs espaciais.

Entre os tipos de tecnologias de SGBDs o núcleo de funcionamento baseiam-se em SGBDs relacionais (SGBD-R) e os mais jovens não relacionais (NoSQLs) ou orientados a objetos (SGBD-OO) [14]. Cada um com especificidades e essências diferentes. Enquanto o SGBD-R trabalha com armazenamento de tabelas e atributos relacionados, o SGBD-OO (orientado a objetos) trabalha com armazenamento de objetos relacionados [10, 14] e o SGBD NoSQL com dados tipo chave-valor orientado a consultas/documentos e grafos [13]. Cada tecnologia de ser adotada de acordo com a arquitetura do sistema e peculiaridade de cada projeto.

Em questão de tecnologias SGBDs-R espaciais existem as soluções em softwares livres, tendo como o mais maduro no momento o PostgresSQL com a extensão espacial PostGis, e as soluções proprietárias, como o SGBD da gigante de tecnologia Oracle com o Spatial and Graph Databse; os SGBDs-OO (orientados a objetos) com a solução proprietária da ESRI Arc Geodatabase e em modalidade freewere oferecidos pela SIG freewere SPRING e, por final, os SGBDs NoSQL com as tecnologias MongoDB (Orientados a Documentos), que é oferecido pelas gigantes Amazon, Google Cloud e Microsoft Azure; a solução da Amazon DynamoDB (chave-valor), o Apache CouchDB (Orientados a Documentos) que pode ser usado localmente; e os que trabalham como grafos (Graph Databases) OrientDB, Neo4J, OrientDB [14] ou CosmoDB disponível também na nuvem da Microsof Azure[2].

O SGBD utilizado neste caso trabalho foi o SGBD-R de tecnologia em software livre PostgresSQL com extensão PostGis com desenvolvimento de códigos utilizando linguagem PgSQL e funções geoespaciais do PostGis.

3.1 Indexação espacial e operadores lógicos espaciais

Os SGBDs relacionais usufruem de uma poderosa linguagem declarativa, a linguagem SQL, para realizar pesquisas, inserções e modificações de dados com uso de operadores lógicos que satisfaçam às necessidades dos utilizadores de SGBDs. São extensivamente utilizados os operadores PgSQL:

Operador/Elemento	Descrição
.	Separador de nome de tabela / coluna
::	Tópico do estilo PostgreSQL
[ ]	Seleção de elemento de matriz
-	Operador menos unitário
^	Operador de exponenciação
* / %	Operadores de multiplicação, divisão, módulo
+ -	Operadores de adição e subtração
IS	Operador condicionado para VERDADEIRO, FALSO, DESCONHECIDO e nulo
ISNULL	Teste nulo
NOTNULL	Teste para não-nulo
IN	Operador com conjunto de membros
BETWEEN	Operador de contenção de alcance (entre valores de atributos)
OVERLAPS	Sobreposição de intervalo de tempo
LIKE ILIKE SIMILAR	Operador para correspondência de padrões
< >	Operador menor que, maior que
=	Operador de igualdade ou atribuição
NOT	Operador de negação lógica
AND	Operador de conjunção lógica
OR	Operador de disjunção lógica

Tabela 1: Tabela de Operadores SQL
Fonte:[5]

Através do uso dos operadores da tabela 2 é possível estruturar toda e qualquer consulta ou manipulação dos dados. A grande problemática é que, em termos performáticos, uso dos operadores convencionais em grandes volumes de dados, sem uso e aplicação de técnicas adequadas, podem inviabilizar a extração das informações de interesse por sobrecarga computacional (esgotamento dos recursos físicos do servidor que abriga o SGBD) impactando em questões de tempo de execução para obtenção de informações para o negócio.

3.2 Índices geoespacias para processamento de grandes volumes de dados

As informações contidas em um banco de dados são visíveis e acessíveis aos usuários por meio de tabelas. Um índice, ou também conhecido como o termo inglês *"index"*, é uma estrutura com associações à uma tabela com apontamentos para os dados armazenados. Um dado indexado possui um ponteiro que atinge diretamente o dado acelerando a busca e recuperação dos dados. Uma pesquisa feito ao dado indexado é ocorre sobre o índice, que possui um apontador direto para o dado. Quando se realiza uma pesquisa em um item não indexado é realizado um "*full table scan*", que é uma leitura bloco a bloco, e/ou sequencial, por toda a tabela selecionada checando item a item todos os critérios dos filtro de dados estabelecidos nos critérios de seleção. A leitura de sequencial bloco a bloco requer um custo computacional elevado em relação a se ter um ponteiro (um índice). A título de exemplo, um "full table scan" em uma tabela extensa pode demorar horas enquanto uma pesquisa em dados que possuam índices podem levar segundos ou milissegundos [11, 9].

3.2.1 Da estrutura e disposição das tabelas no banco de dados

Segundo os manuais do PostgresSQL [1], todas as tabelas e índices são armazenadas em uma matriz de páginas de tamanho fixo. Em uma tabela, um item é uma linha; em um índice, um item é uma entrada do índice. Item se refere a um valor de dado individual armazenado na página. Em uma tabela todas as páginas são logicamente equivalentes, portanto um determinado item (linha) pode ser armazenado em qualquer página. Nos índices, a primeira página geralmente é reservada para uma metapágina contendo informações de controle, e podem existir tipos diferentes de página dentro do índice, dependendo do método de acesso do índice [12].

3.2.2 Dos tipos de índex

Existem vários tipos de índices e cada qual é implementado por diferentes algorítimos. Isso significa que há resultados de performance diferentes de acordo com o tipo de operador, dado e algorítimo utilizado. Isso quer dizer que um índice que possui excelentes resultados para se trabalhar com intersecção entre campos espaciais pode ter um desempenho menor se utilizado em comparações utilizando operadores convencionais como $=$, $>$ ou $<$ [1]. O PostgresSQL trabalhar com os tipos de índices B-tree, Hash, GiST, SP-GiST and GIN:

	Índices	Descrição
Não-espacial	Hash	Adequado caso se planeja utilizar os tipos de operadores abaixo sempre que uma coluna indexada estiver envolvida em uma comparação usando: < <= = >= > Between equivalente a in IS NULL ou IS NOT NULL podem ser usados
	B-tree	
Espacial	GiST	O GiST não é apenas um singelo tipo de índex mas uma infraestrutura com diferentes tipos de estratégias de índices internos que podem ser aplicados. Trabalha com vários tipos de dados geométricos bidimensionais e suporta os operadores: && que se sobrepõem &< sobrepõe a direita &> sobrepõe a esquerda << a direita de >> a esquerda de <<| está estritamente abaixo de |>> está estritamente acima de &<| está sobreposto ou está abaixo de |&> está sobreposto ou está acima de @ se contém @> se contém a esquerda <@ se contém a esquerda ~= igual a
	SP-GiST	Como o GiST possui uma infraestrutura com diferentes tipos buscas além de uma ampla gama de diferentes estruturas de dados baseadas em disco não balanceadas (quadrieres, k-d trees e árvores de sufixos (tries)) e suporta os operadores: << a direita de >> a esquerda de ~= igual a <@ se contém a esquerda <^ abaixo de >^ acima de
	GIN	Índices invertidos que manipulam valores com mais de uma chave (*arrays*)

Tabela 2: Algoritmos de indexação e operadores espaciais
Fonte: [1]

Embora o B-tree nativamente não foi projetado para trabalhar com dados e operadores espaciais, no PostgresSQL ele foi adaptado e existe a possibilidade de trabalhar também com operadores espaciais e utilizá-lo como algoritmo indexador. Comparando-o com o GiST, que indexa o bounding box (fronteira do objeto espacial), o B-Tree indexa a geometria espacial inteira podendo ocasionar falhas de espaço de armazenamento uma vez que o tamanho do índice pode ser maior do que o próprio índice pode lidar [3].

A criação do índice no PostgreSQL possui a seguinte sintaxe:

```
CREATE INDEX [nome_do_indice] ON [nome_da_tabela] USING [tipo_de_index]
(nome_da_coluna_interesse)
Exmplo:
CREATE INDEX ie1_ponto_de_interesse
ON geo.ponto_de_interesse USING btree
(numero_endereco);
```

3.3 Do conceito e uso de *"Bounding Boxes"* com operadores espaciais

É importante saber que operadores dos índices nativamente espacias, como o GiST, trabalham com o conceito *"bounding boxes"* (que significa caixa delimitadora em português) e não com a geometria do objeto específico propriamente. Isso ocorre para otimizar o processo da operação já que, dependendo do tipo de geometria, o custo computacional para algumas operações seria demasiadamente elevado.

O conceito de *"bounding boxes"* consiste em aproximar as formas dos objetos da geometria a objetos simples tais como retângulos ou caixas. Neste caso, o uso de operadores espaciais, tais como &&, não garantes que as geometrias em si se intersectam, mas garante a intersecção entre os *"bounding boxes"* [3]. Isso quer dizer que, no caso

prático, duas geometria esféricas pode o operar && obter e intersecção dos *"bound boxes"* pelas arestas e o dado ser validado como intersectado, porém as esferas estas não se intersectam de fato. Um exemplo típico de *"bounding box"* é do modelo 3D da figura 1 que o *"bounding box"* do objeto é a caixa, não o contorno do modelo. Neste ultimo exemplo um toque entre as caixas e não no objeto pode ser interpretado como um toque no objeto.

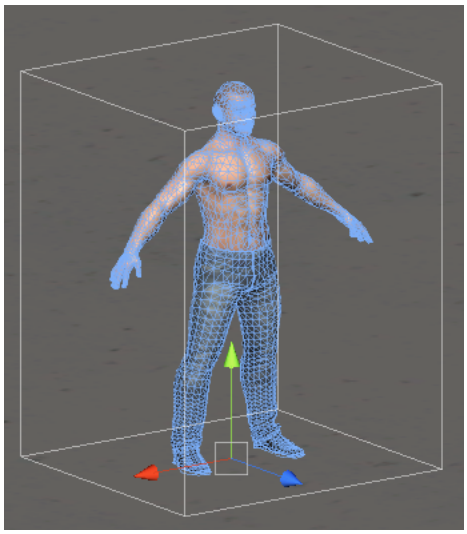

Figura 1: Exemplo de "bouding box" em modelo 3D
Fonte: http://www.xboxblast.com.br/2013/03/gamedev-9-trabalhando-com-sistemas-de.html
Acessado em 29/01/2018

Para melhor compreensão de que dependendo das condições dos objetos com o uso de *"bounding box"* resultados não esperados pode ser obtidos (ratificando que a intersecção entre *"bounding boxes"* não significa a intersecção entre objetos), na 2 visualizamos que, na imagem a esquerda, apenas a linha vermelha cruza todos os objetos (polígono e as linhas verde e azul), na imagem central formação dos bounding boxes, e na imagem da direita o que seria a intersecção entre os dois ou mais *"boundig boxes"* dos objetos porém em termos reais apenas a linha vermelha intersecta o polígono e as linhas verde e azul.

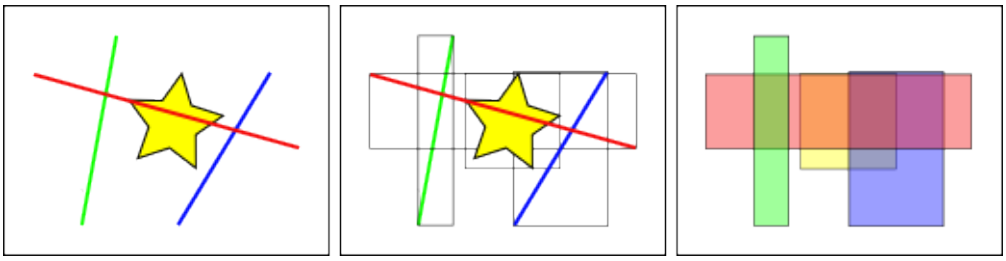

Figura 2: Complicações de índices utilizando bounding boxes
Fonte: [3]

4 Funções de agrupamento espacial (clusterização espacial)

Clusterização é a transliteração do termo em língua inglesa da palavra *"cluster",* que significa grupo ou algo próximo do que seria agrupamento daquilo que é distinto mas possui verosimilhanças entre si. Em termos de computação, *"clusters*", ou métodos de clusterização (agrupamento), também são utilizados para prover arquiteturas computacionais como alta disponibilidade de ambientes, alto desempenho, balanceamento de carga. Dependendo do sentido temos de compreender como a aplicação do termo está relacionado a agrupamento de objetos, dados ou informações por verosimilhança, ou até mesmo algum critério de interesse específico baseado em algoritmos hierárquicos ou particionados [8]. Aplicado em Geoprocessamento, *"clusters"* podem ser utilizados para agrupar dados que apresentam naturezas verosimilhantes separando-os de dados que não são de interesse para uma dada análise estatística e/ou espacial.

É importante ressaltar que existem vários modelos matemáticos/algoritmos aplicáveis à clusterização que não estão disponíveis como funções na extensão PostGis, mas podem ser codificados por meio de códigos em SQL. Um exemplo típico é o algoritmo de *Mahalanobis* que funciona com correlações de fatores de distância, que apresentam grande eficiência dependendo do tipo de objetivo de análise a ser desenvolvida, e não está disponível na extensão PostGis.

O processo de escolha da função de clusterização depende diretamente do tipo de resultado que se espera e das variáveis existentes. Como por exemplo, o uso de clusterização que utiliza o métodos K-means particiona os pontos geocodificados em grupos rigidamente definidos utilizando a média do grupo mais próximo. Tal rigidez, dependendo do tipo de aplicação, torna o K-means inaplicável para este caso de uso criando a necessidade do uso de outro tipo de método. Sendo assim veremos as características

principais das funções de clusterização para uso em tomadas de decisões, com exemplos a partir de uma nuvem de pontos obtidos do banco de dados de um número de endereços de entregas de objetos postais de um CEP aleatório para se gerar e estudar os *"clusters"* com os dados de mais significativo e utilizá-los, após aplicação de funções estatísticas, para obter a geocodificação do endereço mais próximo possível da geocodificação real.

O uso destas técnicas foram utilizadas pois os pontos obtidos a partir da baixa de entrega de objetos registrados pelo Carteiro para um mesmo endereço possuíam muita heterogeneidade (para um mesmo endereço diversos pontos em locais muito distantes e diferentes) ocasionados por fatores de imprecisão do GPS ou até mesmo baixa em locais distintos do endereço de entrega.

Para este trabalho foi utilizado como critério de seleção para clusterização filtragem dos pontos que estavam a uma distância de até 30m da linha do CEP (tabela cep_base).

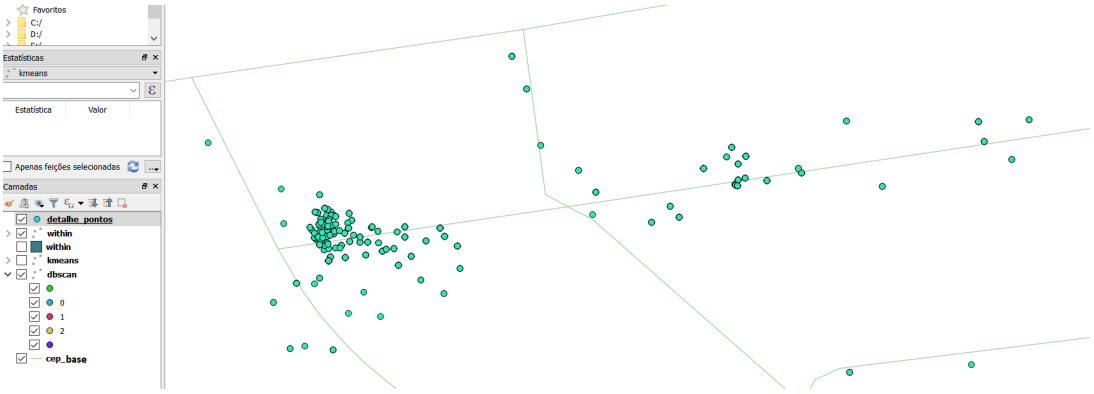

Figura 3: Nuvem de pontos de endereço aleatório

4.1 Função ST_ClusterKMeans

O métodos K-means particiona os dados em grupos definidos e utiliza a média do grupo mais próximo [4]. Uma das variáveis principais é o processo rígido de definição do número de grupos. Tal rigidez torna o uso do método inaplicável quando se trabalha

com dados heterogêneos pois quando não se tem dados suficientes para formação dos grupos. Caso o requisito de quantitativo mínimo de pontos não seja atingido a função não permite o processamento e paralisa todo o processo. Como exemplo caso a função seja definida com formação de um quantitativo mínimo de 10 *"clusters"* mas existindo um quantitativo de dados menor que o número mínimo de *"clusters"*, exemplo 9 dados, a função não pode ser processada.

Para análises de clusterização de grande volume de dados com bases significativamente heterogênicas (em relação a quantidade de pontos), a rigidez da quantidade mínima de grupos definido pelo método inviabiliza o seu uso.

Em termos de acurácia de agrupamento dos pontos em grupos, atendidos os requisitos de quantitativo mínimo de pontos para agrupamento, apresentou grande acurácia no agrupamento de pontos por proximidade em relação a todas as demais funções existentes no PostGis.

```
–Script exemplo uso "cluster" st_clusterkmeans
–Observações:
–Utilização de index especais
–restrição de uso apenas dos pontos com distância menor ou igual a
30 metros da linha do CEP
–Número de clusters: 10
select
st_clusterkmeans(geom, 10) over () as cluster,
geom
into kmeans
from
(SELECT b.geom, (EXISTS ( SELECT g.cep
FROM conexao_geometria f
JOIN cep_base g ON f.registro_nu = g.registro_nu
WHERE f.numero_celula =
(( SELECT e.numero_celula
FROM linha e
WHERE
e.celula_geom &&
st_transform(st_expand(st_transform(b.geom, 3857), 30), 4326)
AND st_distanceSphere(b.geom,e.celula_geom) <= 30
ORDER BY (st_distanceSphere(b.geom,3857, e.celula_geom) ) LIMIT 1))
AND g.cep = b.numero_cep)) AS cep_correto,
b.numero_cep, b.numero_endereco
FROM detalhe_ponto b
WHERE b.status = 'E'
AND b.numero_cep = '80035260'
AND b.numero_endereco = '33') resultado
WHERE cep_correto = 't' ;
```

Resultado da aplicação do *"cluster"* da função ST_ClusterKMens com representação gráfica dos agrupamentos (*"clusters"*) resultantes segmentado por cores:

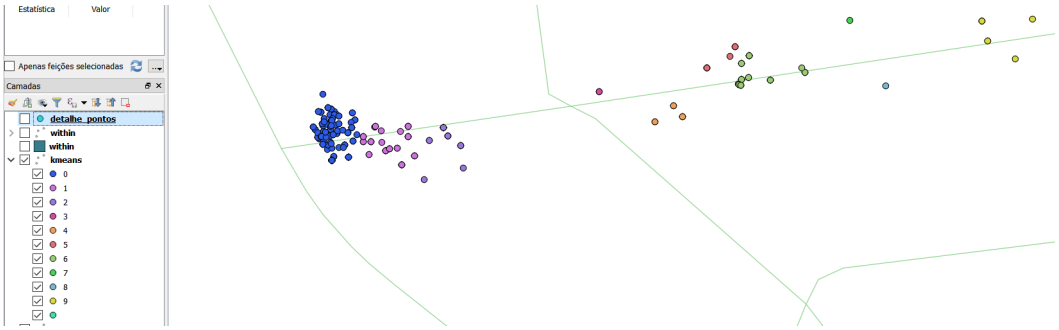

Figura 4: Resultado ST_ClusterKmeans

4.2 Função ST_ClusterWithin

A função ST_ClusterWithin tem como característica gerar um resultado com *"array"* de coleção de geometria a partir de uma distância cartesiana definida. Internamente seleciona os *"clusters"* por uma particular verossimilhança de distância ou aglomerados [4]. A documentação oficial da função é pobre para descrever o algoritmo deste mecanismo interno e para maior entendimento é necessário estudar o código da função.

Resultando um excelente resultado de clusterização é permitido a utilizações de funções espaciais como centroide (ST_Centroid), obtenção de círculo geométrico dos *"clusters"* para representação de esferas de calor (ST_MinimumBoundingCircle), estatísticas diretamente no *"array"* gerado e algumas outras funções interessantes, porém apresenta a problemática de que a coleção de geometria resultante não pode ser trabalhada diretamente em alguns softwares de SIG como por exemplo o QGis, tornando necessário utilizar funções de extração de coleções (ST_CollectionExtract) previamente caso se deseja plotar seus resultados graficamente nestes sistemas.

```
–Script exemplo uso "cluster" ST_ClusterWithin
–Observações:
–Utilização de index especais
–restrição de uso apenas dos pontos com distância menor ou igual a 30 metros da linha do CEP
–Distância do "clusters" 10m
SELECT
row_number() over () AS id,
ST_NumGeometries(gc) qtpontos,
ST_CollectionExtract(gc,1) AS geom_collection,
into Within
from
(select unnest(ST_ClusterWithin(st_transform(geom,3857), 10)) gc
from
(SELECT b.geom, (EXISTS ( SELECT g.cep
FROM conexao_geometria f JOIN cep_base g ON f.registro_nu = g.registro_nu
WHERE f.numero_celula = (( SELECT e.numero_celula FROM linha e
WHERE e.celula_geom && st_transform(st_expand(st_transform(b.geom, 3857), 30), 4326)
AND st_distance(st_transform(b.geom, 3857), st_transform(e.celula_geom, 3857)) <= 30
ORDER BY (st_distance(st_transform(b.geom,3857), st_transform(e.celula_geom, 3857))) LIMIT
1))
AND g.cep = b.numero_cep)) AS cep_correto, b.numero_cep, b.numero_endereco
FROM detalhe_ponto b
WHERE b.status = 'E'
AND b.numero_cep = '80035260'
AND b.numero_endereco = '33') resultado
WHERE cep_correto = 't') as resultado2
```

Resultado da aplicação do *"cluster"* da função ST_ClusterWithin com representação gráfica dos agrupamentos (*"clusters"*) resultantes segmentado por cores:

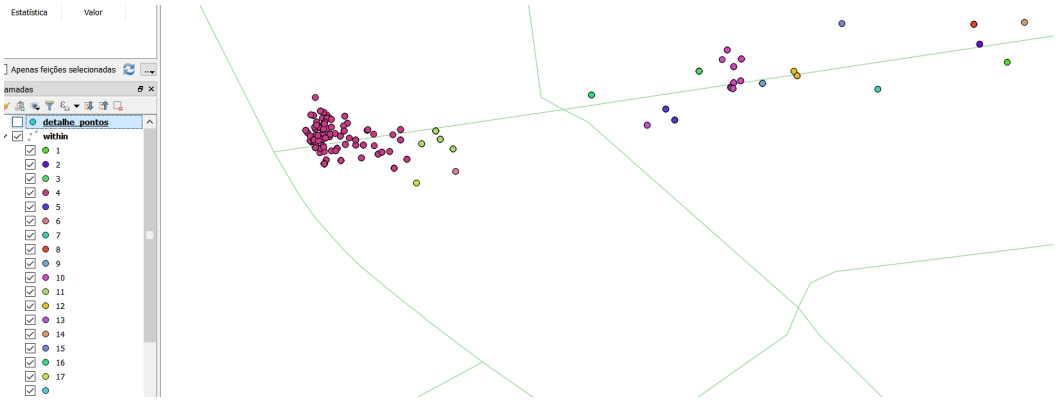

Figura 5: Resultado ST_ClusterWithin

4.3 Função ST_ClusterDBSCAN

A função ST_ClusterDBSCAN classifica os dados, dentro das geometrias submetidas à função, associando a um quantitativo de e vinculação de *"clusters"* dinamicamente, retornando o seu ID atribuído a partir de uma relação de definição de distância cartesiana e um quantitativo mínimo de pontos que satisfaça uma dada condição da função. É muito útil para extensas massas de dados quando se necessita que, obrigatoriamente, um *"cluster"* tenha como parâmetro de formação apenas o quantitativo mínimo de amostras.

Para que um dado geométrico faça parte do *"cluster"* da função ele tem de atender as seguintes condições: 1- Ser um núcleo da geometria que esteja dentro da distância cartesiana mínima definida com o quantitativo mínimo de pontos definido; ou 2- Ser a borda da geometria que esteja dentro da distância de um núcleo de geometria [4].

```
–Script exemplo uso "cluster" ST_ClusterDBSCAN
–Observações:
–Utilização de index especais
–restrição de uso apenas dos pontos com distância menor ou igual a 30 metros da linha do CEP
–Número mínimos de pontos por "cluster" 10
–Distância do "cluster" 10m
select
row_number() OVER (PARTITION BY "cluster" ORDER BY "cluster" DESC),
cluster,
geom
into dbscan
from (select ST_ClusterDBSCAN(st_transform(geom,3857), eps:=10, minpoints:= 10) over () as
cluster, geom
from
(SELECT b.geom, (EXISTS ( SELECT g.cep
FROM conexao_geometria f JOIN cep_base g ON f.registro_nu = g.registro_nu
WHERE f.numero_celula = (( SELECT e.numero_celula FROM linha e
WHERE e.celula_geom && st_transform(st_expand(st_transform(b.geom, 3857), 30), 4326)
AND st_distance(st_transform(b.geom, 3857), st_transform(e.celula_geom, 3857)) <= 30
ORDER BY (st_distance(st_transform(b.geom,3857), st_transform(e.celula_geom, 3857))) LIMIT
1))
AND g.cep = b.numero_cep)) AS cep_correto, b.numero_cep, b.numero_endereco
FROM detalhe_ponto b
WHERE b.status = 'E'
AND b.numero_cep = '80035260'
AND b.numero_endereco = '33') resultado
WHERE cep_correto = 't') as resultado3
order by cluster,row_number
```

Resultado da aplicação do *"cluster"* da função ST_ClusterDBSCAN com representação gráfica dos agrupamentos (*"clusters"*) resultantes segmentado por cores:

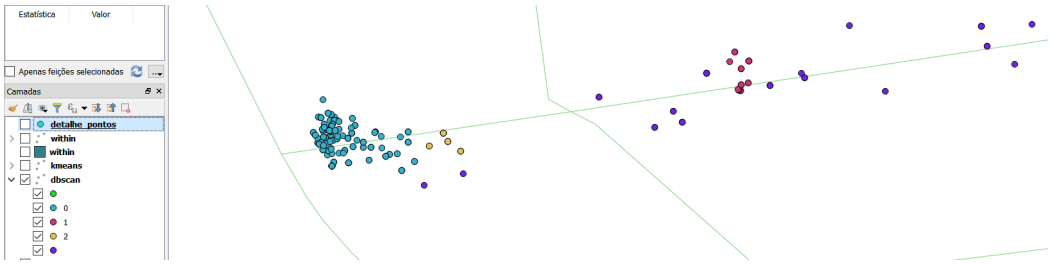

Figura 6: Resultado ST_ClusterDBSCAN

4.4 Função ST_ClusterIntersecting

A função ST_ClusterIntersecting retorna um *"array"* com coleção de geometrias de todos os componentes que estão conectados dentro de um conjunto de geometrias [4]. Assim como a função ST_ClusterWithin, é necessário, para visualização em softwares de edição de SIG (QGis por exemplo), utilizar função de extração de coleção (ST_CollectionExtract). Neste processo esta função apresntou um problema, em todos os casos testados, perda o sistema de referência das informações sendo necessário reatribuir manualmente no sistema SIG para que as informações pudessem ser plotadas corretamente. Uma outra alternativa é a utilização da função ST_Transform para transformar os dados ao sistema de referência correto no campo geom.

```
–Script exemplo uso "cluster" ST_ClusterIntersecting
–Observações:
–Utilização de index especais
–restrição de uso apenas dos pontos com distância menor ou igual a 30 metros da linha do CEP
–Número mínimos de pontos por "cluster" 10
–Distância do "cluster" 10m
SELECT
row_number() over () AS id,
ST_NumGeometries(gc) qtpontos,
ST_CollectionExtract(gc,1) as geom
into clusterintersecting
from (select unnest(ST_ClusterIntersecting(st_transform(geom,3857))) gc from
from
(SELECT b.geom, (EXISTS ( SELECT g.cep
FROM conexao_geometria f JOIN cep_base g ON f.registro_nu = g.registro_nu
WHERE f.numero_celula = (( SELECT e.numero_celula FROM linha e
WHERE e.celula_geom && st_transform(st_expand(st_transform(b.geom, 3857), 30), 4326)
AND st_distance(st_transform(b.geom, 3857), st_transform(e.celula_geom, 3857)) <= 30
ORDER BY (st_distance(st_transform(b.geom,3857), st_transform(e.celula_geom, 3857))) LIMIT
1))
AND g.cep = b.numero_cep)) AS cep_correto, b.numero_cep, b.numero_endereco
FROM detalhe_ponto b
WHERE b.status = 'E'
AND b.numero_cep = '80035260'
AND b.numero_endereco = '33') resultado
WHERE cep_correto = 't') as resultado4
```

Resultado da aplicação do *"cluster"* da função ST_ClusterIntersecting com representação gráfica dos agrupamentos (*"clusters"*) resultantes segmentado por cores:

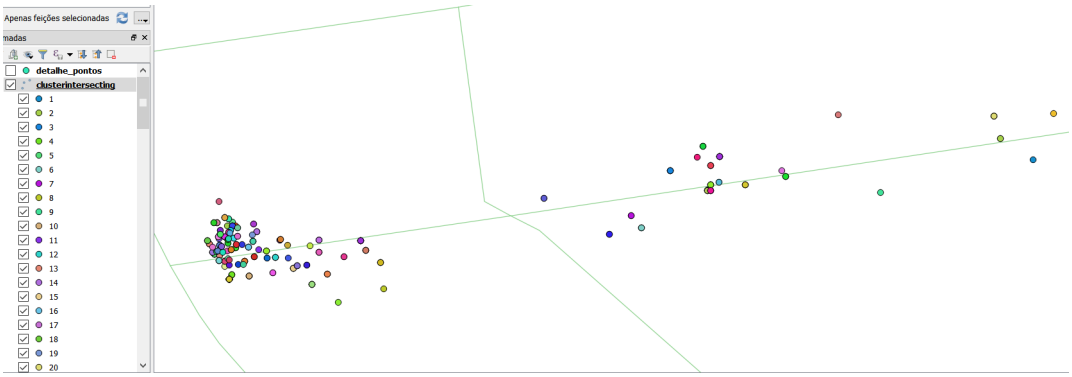

Figura 7: Resultado ST_ClusterIntersecting

4.5 Comparação e melhor escolha de funções para o estudo da caso

A função de clusterizaçao ST_ClusterKMeans apresentou excelentes resultados finais porém sua limitação quando a quantitativos de dados heterogêneos, que não pode ser definido dinamicamente, resulta em restrição de processamento. Tanto ST_ClusterDBSCAN quanto ST_ClusterWinthin são compatíveis com dados heterogêneos porém o ST_ClusterWinthin apresentou resultados mais acurados quanto a seleção dos objetos. O ST_ClusterDBSCAN possui a vantagem de criar dinamicamente o quantitativo de *"cluster"* e efetivação da atribuição dos dados baseado a partir de um quantitativo mínimo de dados (pontos) que tenham dada semelhança além da distância, porém seu resultado é apresentado como uma coleção de *"array"* que não pode ser interpretado pelos sistemas SIGs necessitando do uso de funções de extração aumentando a carga computacional. A função ST_ClusterWinthin, a escolhida para aplicação no caso de uso deste trabalho, é bastante útil quando o fator distância for a questão principal do problema além da semelhança entre os pontos. A função ST_ClusterIntersecting gera *"cluster"* resultantes a partir de geometrias que se tocam porém é inaplicável para o

estudo de caso deste trabalho.

5 Caso de uso dos Correios - Obtenção de POIs a partir de grandes volumes de dados

Uma das principais demandas estratégicas do Correios é a otimização da roteirização e desenvolvimento do projeto de GeoMarketing. Ambas demandas dependem da existência de uma base de endereços geocodificados com abrangência de todo território nacional. Foi definida a estratégia de geocodificar todos os endereços a partir da criação de uma base própria a processada pelos dados espaciais captados em sua extensa capilaridade presente em todos os municípios brasileiros. Foi levado em conta a plena capacidade de produção de dados suficiente para atender o objetivo proposto. Foi estabelecido como fonte para processamento das informações a base de dados dos pontos geolocalizados de entrega de objetos postais para obter os POIs a partir da geolocalização dos pontos de entrega de objetos postais e associando-os ao endereço postal .

É importante registrar que, embora exista um padrão de numeração dos endereços dos logradouros, há demasiadas exceções em território nacional, em especial às localidades que cresceram sem planejamento urbano ou localidades históricas que não possuíam qualquer padronização. Caso se adotasse a norma para geocodificar os endereços, haveria a necessidade de informação manual dos sentido de onde começa e onde termina cada rua além do conhecimento da regras dos lados de arruamento, o que seria impossível ser feito manualmente, com pouco eficiência e baixa acurácia. Não há base de dados com registros consolidados no território nacional para se estabelecer algoritmos que obtenham a geocodificação dos endereços com base nas regras. Embora sistemas companhias como o Google e ESRI tenham a geocodificação de muitos endereços utilizando um algoritmo baseado no padrão de arruamento brasileiro, os dados acabam "errando" demasiadamente em razão das exceções em locais históricos ou padrões de

arruamentos diferentes. A exceção é via de regra no Brasil em termos de definição de arruamentos.

No momento os carteiros já operam mais de 7 mil celulares ativos, de um total de 44.426 aparelhos entregues, com tecnologia de geolocalização com perspectiva de operar 44.514 mil aparelhos em operação até março de 2018. Isso significa um salto para quase pouco mais de 61 bilhões de registros anuais de contendo geoespaciais.

Os aparelhos possuem um sistema próprio da ECT (SRO Móvel) que captura dados espaciais no momento da entrega de objeto postais (incluindo situações de ausência, recusa do destinatário ou outros) além de capturar dados para controle da percorrida do carteiro por meio do envio da geolocalização a cada cinco segundo ou dez passos. Tais informações visam obter e mensurar os trabalhos a partir dados estatísticos além de controlar e comprovar que o carteiro esteve no local de entrega indicado para melhoria da qualidade dos serviços em casos de contestação pelos clientes. A próxima versão do aplicativo fará também a captura da assinatura do recebedor do objeto geolocalizada para disponibilidade em processos online.

Os resultados do trabalho aplicados nos processo de roteirização, a geocodificação objetiva maximizar a eficiência das rotas, diminuição do tempo de entrega, melhoria dos serviços e economia de gastos. O projeto de GeoMarketing será uma nova modalidade de serviço oferecido pelo Correios para a carteira de clientes incluindo serviços que variam de mala postal com *analisitcs* por perfil de negócio a venda de dados de altimetria por uma malha de pontos das cidades brasileiras.

O grande desafio deste trabalho não é apenas aplicar funções espaciais conhecidas do PostGis. É como obter informações espaciais a partir de um gigantesco volume de dados. Para tando foi necessário abstrair o processo de codificação considerando a aplicação uma série de técnicas de desenvolvimento voltado para processamento de grande volume de dados espaciais, com cenário de recursos computacionais finitos, com

obtenção de informações em prazo de negócio aceitável. As funções espaciais conhecidas sem a aplicação de técnicas, dependendo do tipo de função e operadores lógicos, geram gargalos que tornam o banco indisponível e inoperável pelo esgotamento de recursos gerando prejuízos ao negócio. Um código funcional não projetado para operar performaticamente em grandes volumes de banco de dados pode paralisar o SGBD e os negócios da empresa que utilizam informações espaciais para operação, tomada de decisão e prestação de serviços.

Para atender as metas estratégicas e demandas tecnológicas de geotecnologias contidas no Plano Estratégico Institucional e no Plano Diretor de Tecnologia da Informação houve a iniciativa de buscar capacitação em geotecnologias na Universidade de Brasília assim como fomento ao estudo dos manuais das geotecnologias.

5.1 Da problemática inicial

Impossibilitados de utilizar algoritmos de arruamento para obter, por aproximação, a geocodificação dos endereços no Brasil, por falta de padronização na maioria dos municípios, decidiu-se utilizar os pontos de entrega de cada objeto postal como fonte de informações para processar e geocodificar os pontos de intersse POIs, que são os endereços nacionais geocodificados.

Analisando previamente os dados em um projeto piloto diversos problemas de inconsistência ou qualidade dos dados foram identificados. Tais inconsistências tiveram de ser tratadas adotando técnicas distintas. Entre os problemas de inconsistência encontrados registramos:

- Pontos com significativa imprecisão do GPS;
- Baixa dos objetos ao longo do trajeto da percorrida e não no ponto de entrega;
- Baixa dos objetos em locais distintos da proximidade do endereço de entrega, como nos centros de distribuição;

- Falta de informe de parada das atividades de trabalho com envio automático de coordenadas descartáveis em períodos noturos e/ou finais de semana/feriados.
- Linhas espaciais de arruamento sem vinculação com a base de CEP;
- Vinculação dos sistemas de referência de tabelas (GEOMETRY_COLUMNS) diferentes dos indicados nos campos internos da própria tabela;
- Campos de número do endereço com caracteres não numéricos (espaços, traços, nomes etc);
- Esgotamento de recursos ou tempo de resposta extenso (dias) com uso de consultas espaciais em grande volume de dados sem aplicação de técnicas de alta performance (*"tuning"*).

Após identificação dos problemas tornou-se evidente a necessidade do uso de um método de processamento otimizado com fase de tratamento e pré-processamento prévio dos dados, eliminação dos dados insignificantes, aplicação de técnicas de performáticas *("tuning")* para diminuição do tempo de resposta, aplicação de funções espaciais com inteligência suficiente para retorno das informações de interesse associadas com geoestatística.

5.2 Do método utilizado

1. Escolher origem de fonte de dados e tecnologias:
   (a) Identificar bases de dados;
   (b) Identificar tecnologias das fontes de dados;
   (c) Pesquisar documentação técnica da tecnologia PostgrSQL/PostGis
2. Aplicar filtros de dados:

(a) Utilizar apenas pontos de baixa de objetos;

(b) Utilizar apenas registros que o CEP possua vinculação com linha espacial;

(c) Utilizar apenas registros que a agência do carteiro tenha o ponto geolocalizado;

(d) Tratar as dados de número de endereço extraindo exclusivamente números;

3. Aplicar tuning (refinamento):

   (a) Indexar tabelas e campos espaciais;

   (b) Utilizar operadores e consultas performáticas em SQL espacial;

   (c) Particionar tabelas;

4. Aplicar filtros espaciais:

   (a) Eliminar pontos próximos de um raio delimitado da agência

   (b) Eliminar pontos que ultrapassem um raio delimitado da linha indexada do CEP

   (c) Validação do sistema de referência do ponto

5. Aplicar funções espaciais:

   (a) Clusterizar os pontos;

   (b) Escolher o *"cluster"* mais significativo;

   (c) Extrair POI a partir do *"cluster"* escolhido;

6. Validar resultados:

   (a) Validar informações geradas

5.3 Origem da fonte de dados e tecnologias adotadas

O Correios desenvolveu ao longo dos últimos três anos tecnologias que consomem e produzem dados georeferenciados. O sistema que produze o maior volume de dados espaciais é Sistema de Rastreamento de Objeto, que registra o ponto dos objetos entregues e a percorrida do carteiro a cada cinco segundos gravando na base do sistema ECTGeo. Os dados produzidos são principalmente consumidos pelo Sistema de Gerenciamento de Desempenho Operacional, que obtém os dados estatísticos e gerenciais das percorridas dos carteiros para cada unidade operacional para melhoria dos seus processos de trabalho e qualidade para os clientes; o sistema de roteirização e controle de frota de veículos e sensores remotos que consomem informações de endereços e unidades geocodificadas; e o subsistema Geo Editor de Mapas que faz a vinculação de linhas espaciais com CEPs da base de negócio. Existem outras sistemas que produzem e consumem minoritariamente dados espaciais que controlam serviços de clientes do Governo. O coração das informações espaciais é o sistema ECTGeo.

O Correios adotou a tecnologia de SGBD PostgresSQL com a extensão PostGis para o projeto ECTGeo. Este banco de dados possui hoje aproximadamente 3 bilhões de registros espaciais incluindo diversos tipos de geometrias (pontos, linhas e polígonos), com perspectiva de crescimento para 64 bilhões ainda este ano. A ECT também possui a tecnologia SIG da ESRI Arcgis Desktop e portal utilizando a extensão *Network* para roteirização, porém sua base de dados de pontos geocodificados de endereços demonstrou-se insatisfatória para a instituição em termos de acurácia da geocodificação dos endereços com erro de pontos de até 800km e teve de ser descartada.

Em atendimento as normas internas de segurança da informação todos os nomes de tabelas, colunas, campos e siglas de sistemas serão alterados neste trabalho em atendimento as diretrizes de sigilo institucional das informações.

No SGBD PostgresSQL o banco de dados de estudo possui extensão PostGis com

nome geo e esquema público com as seguintes tabelas:

- cep_base: Contém o código de todos os CEPs de arruamentos existentes em território nacional contendo endereço e UF;
- linha: Contém os vetores de linhas de arruamentos geocodificadas;
- conexao_geometria: Contém a vinculação das linhas de arruamento geocodificadas com a tabela cep_base;
- geo_agencia_correios: Contém a geocodificação de cada unidade institucional;
- ponto: Contém as informações do registro de trabalho do dia de cada carteiro, a unidade operacional, a matrícula do funcionário, o distrito postal, data e hora de início e fim das atividades do turno de trabalho.
- detalhe_ponto: Contém a nuvem de pontos geocodificados coletada pelos aparelhos celulares para cada registro de "ponto" do dia, o código dos objetos postais e o status de baixa (entrega, recusa, ausência etc).

5.4 Aplicação de filtro de dados

Após definição com os gestores funcionais das áreas de negócio foi acordado que para atendimento da demanda os filtros de dados deveriam ser aplicados a fim de:

(a) Utilizar apenas pontos de baixa de objetos: tabela "detalhe_ponto" processar apenas marcadores do tipo "E" na coluna "status";

```
select * from detalhe_ponto dp where dp.status = 'E'
```

(b) Utilizar apenas registros que o CEP possuam vinculação com linha espacial: a tabela "conexao_geometria", que faz a ligação entre CEP e linha espacial, deve

possuir registro da tabela que contém os registros CEPs, campo “registro_nu” da tabela “cep_base”, com as linhas espaciais, campo “numero_celula” da tabela “linha”;

```
select * from conexao_geometria cg
where exists (select 1 from linha ln
where cg.numero_celula = ln.numero_celula)
and exists (select 1 from cep_base cb
where cb.registro_nu = cg.registro)
```

(c) Utilizar apenas registros que a agência do carteiro tenha o ponto geolocalizado: a tabela ponto possui o número da agência que distribuiu a encomenda e deve existir o registro correspondente da tabela da dados geocodificados das agências ;

```
select * from ponto p
where exists (select 1 from geo_agencia_correios a
where p.agencia = a.agencia)
```

(d) Tratar as dados de número de endereço extraindo apenas números: o tipo do campo é de caracteres e no momento do cadastro o atendente cadastra manualmente o número do endereço podendo conter caracteres (como é o caso de Sem número (S/N) ou “Casa A”), e deve ser tratado para trabalhar apenas com valores numéricos tratando os casos em que existam outros caracteres associados ao número de endereço;

```
select
trim(dp.numero_cep),
replace(trim(dp.numero_endereco),'.','')::int, dp.geom
from
detalhe_ponto dp
where
dp.status = 'E'
and textregexeq(trim(dp.numero_endereco),'^[[:digit:]]+(\.[[:digit:]]+)?$')=true
–última expressão checa se a variavel do texto pode ser convertido para numérico
```

5.5 Aplicação de *tuning* (refinamento)

"Tuning", em termos de computação, pode ser descrito como otimização com finalidade de melhoria na performance, algo que é necessário com a gravação de milhares de registros espaciais que incrementam impactos negativos de performance conforme se aumenta o número de registros. São exemplos de aplicação de *"tuning"* a criação de índices para acelerar consultas, utilização de operadores mais performáticos e até mesmo particionamento de tabelas para melhoria de performance na recuperação e gravação de dados pela execução de ações definidas a partir de um planejamento prévio [6, 12].

5.5.1 Indexação de tabelas e campos espaciais

Tipos diferentes de consultas para extração do mesmo dado ou informação podem ter performance diferentes de acordo com a técnica utilizada. Indexação de dados espaciais tratam as geometrias a partir da transformação do dado *"bounding boxes"*, que é mais leve computacionalmente, podendo ser aplicado vários operadores espaciais os índices e não sobre o dado bruto em si. Trabalhando com índices se obtém performance muito maior do que sobre dados não indexados. Consultas a dados não indexados ou uso de funções espaciais que não trabalham com índices forçam leituras completas de tabelas (*full table scan*) que têm custo computacional muito mais elevado. Para fins de exemplo uma consulta que dura 1500ms em uma tabela não indexada pode levar 200ms quando há índices espaciais [7].

Há vários algoritmos diferentes de indexadores e que possuem performances e consumo de área de armazenamento diferentes. Quando se trabalha com grandes volumes de dados cria-se a necessidade de se avaliar performance, uso de área de armazenamento, finalidade e funcionalidades.

Dois algoritmos de indexação foram testados e comparada a performance em uma

base de dados de desenvolvimento a partir de geometrias contidas em três tabelas executando o *script* de produção final do caso de uso deste trabalho em ambiente de testes.

As tabelas utilizadas em ambiente de produção e testes para análise e comparação de indexadores na data de 08/02/2018 as 11:05:

Tabela	Num Registros	Descrição Tabelas	Espaço de Tabela
detalhe_ponto	3.100.894.656 (ambiente de produção) 78.834.276 (teste)	19 colunas de dados de chaves de registros, geometria de pontos, status do tipo entrega, CEP, número do logradouro, código do objeto, elevação, horário, outros dados	341,40GB (produção) 11GB (teste)
ponto	2.029.752 (ambiente de produção) 1.894.219 (teste)	8 colunas de dados de chaves de registros, código da agência, distrito postal, registro funcional do carteiro, confirmação de transação, outros dados	211MB (produção) 200MB (teste)
linha	2.232.473	3 colunas de dados de chaves de registros, geometria de linha, UF	439MB
cep_base	1.038.023	3 colunas de dados de chaves de registros, CEP, UF	76MB
conexao_geometria	1.351.725	3 colunas de dados de chaves de registros, chave de registro da cep_base, chave de registro da linha	87MB
geo_agencia	4946	9 colunas de dados de chaves de registros, geometria de ponto, limites inferiores, limites superiores, endereço, nome da unidade, UF	56KB

Tabela 3: Descrição de tabelas no banco de dados

Em novembro de 2017 houve um esgotamento de recursos de armazenamento e a tabela detalhe_pontos, que estava com 18.2 bilhões de registros com dados espaciais e aproximadamente 2.01TB de área de armazenamento e teve de ser sanitarizada. Neste processo os dados de pontos de percorrida que não eram de baixa de entregas de objetos postais foram descartados para fins de processamento deste trabalho.

5.5.2 Comparação de performance e uso de recurso de indexadores B-Tree e GiST

Fora realizados testes de performance (análise do tempo médio de processamento para cada algoritmo de índice) e verificação de uso de recursos de armazenamento de dados para fins comparativos entre os indexadores GiST e B-Tree.

Embora o B-Tree não tenha sido desenvolvido nativamente para trabalhar com dados espaciais, sua utilização é mais performática do que o GiST mas com uma grande limitação quanto ao tamanho do índice (8191 bytes). Isso significa que o algoritmo B-Tree não tem a capacidade de trabalhar com geometrias de significativo tamanho espacial como polígonos com muitas arestas ou até mesmo linhas com muitos vetores, restringindo o trabalha apenas com dados que gerem *"bouding boxes"* com área de armazenamento de índices pequenos como pontos, poucos segmentos de linhas ou polígonos simples. O algoritmo GiST trabalhou bem com geometrias de tamanhos distintos sem nenhum tipo de problema.

Resultados de desempenho de criação de índices Em ambiente de testes o B-Tree utilizado como indexador para pontos geolocalizados (pequenos índices de *"bounding boxes"* que não estouram o limite do indexador) apresentou desempenho de criação dos índice para a tabela detalhe_ponto, com 11GB de dados e quase 79 milhões de registros espaciais (todos executados em ambiente de teste), média de 08 minutos e 17 segundos. O GiST tempo médio de 21 minutos 16 segundos. A média foi obtida após 25 repetições de criação de índices. O B-Tree apresentou resultado final de aproximadamente apenas 2/5 avos de tempo de execução do índice do GiST no ambiente de testes contendo apenas geometria de pontos.

Resultados de utilização de volume de armazenamento O B-Tree utilizado como indexador para pontos geolocalizados da tabela detalhe_ponto apresentou volume de armazenamento de 3739MB enquanto o GiST 4329MB. O B-Tree apresentou volume de área de armazenamento 14% menor que o GiST neste estudo de caso.

Resultados de desempenho de tempo de execução de *script* Em ambiente de testes o *script* para atendimento da demanda institucional de criação de pontos de interesse (endereços geolocalizados a partir dos pontos de entrega de objetos registrados) nas tabelas de banco de dados informadas na 3 apresentou os seguintes tempos de processamento para cada tipo de índice resultando em 692.667 registros com 15 repetições:

- B-Tree: média de 26 minutos e 56 segundos;
- GiST: média de 27 minutos e 11 segundos;
- Processamento sem uso de indexadores: média de 32 minutos e 27 segundos;

O B-Tree apresentou uma média de 1% de maior eficiência em relação ao GiST no processamento do *script* objeto deste trabalho no ambiente de teste. Já uma comparação de performance dos resultados com e sem o uso de índices, o resultado final sem o uso de índice comparado com o resultado obtido no B-Tree foi de 120% o tempo total de processamento do B-Tree .

Conclusão de comparação de resultados entre indexadores A definição do algoritmo de indexação espacial é primordial quando se trabalha com dados espaciais, pois o seu incremento de performance, em relação ao uso de dados não

indexados, é substancial e fator decisivo para se obter respostas com maior agilidade. Em termos de performance não há significativas diferenças entre utilizar os algoritmos B-Tree ou GiST no processamento de dados, porém em termos de uso área de armazenamento o B-Tree é uma ótima escolha quando se trabalha com dados espaciais pequenos pois o algoritmo utiliza menor volume de armazenamento. Em contrapartida seu uso é inviável caso se tenha muita heterogeneidade de tamanhos de objetos espaciais, como é o caso de polígonos ou linhas com muitas arestas/segmentos de vetores ou tamanho espacial. Cabe ao analista de banco de dados geoespacial definir um equilíbrio entre as vantagens e desvantagens de cada algoritmo que devem observar os tipos de dados e disponibilidade de armazenamento para escolha do indexador. Neste estudo também foi identificado que no uso do GiST, para algumas geometrias (polígonos com muitas arestas ou linhas com muitos segmentos), houve um crescimento do tamanho dos índices ao ponto de que o próprio índice possuía um volume de armazenamento mais elevado que o próprio dado da base (em um caso especial o volume de dados do índice GiST consumia 30GB e própria tabela de dados possuía aproximadamente um valor menor de 20GB).

5.5.3 Utilizando operadores e consultas performáticas em SQL espacial

Consultas indexadas É importante ter ciência que nem todas as funções e operadores do PostGis usam nativamente os índices. O que deve ser levado em consideração em primeiro lugar é que consultas espaciais devem ser precedidas de consultas indexadas antes da elaboração de qualquer utilização de operador e as funções espaciais devem ser escolhidas com preferências aquelas que fazem consultas a índices. Segundo [7], as boas práticas de consulta em bancos de dados espaciais implementam uma forma

"de duas fases" de processamento espacial:

1- A primeira fase é a pesquisa de indexada dos *"bounding boxes"*, que é executada em toda a tabela por meio dos operadores espaciais 2 tais como "&&" que significa sobreposição de *"bonding boxes"* análogo ao operador lógico matemático "=" (igual) [7];

2- A segunda fase é o teste de processamento espacial preciso, que é executado apenas nos dados retornados pelo subconjunto pela primeira fase, que seria equivalente ao uso de funções espaciais propriamente dita, como é o caso das funções ST_Distance, ST_Intersects(), ST_DWithin() etc.

```
SELECT * FROM linha ln, pontos p
WHERE ln.geom && st_expand(p.geom,20) –1ª fase
and ST_Intersects(ln.geom,p.geom) –2ª fase
–&& retorna todos os bounding boxes que se sobrepões (linha e pontos)
–expandindo pela função ST_expand os bounding boxes dos pontos para 20 metros.
```

Neste caso de uso todas as consultas espaciais foram otimizadas pelo uso de operadores em dados indexados para retornar apenas os pontos próximos a linha do CEP por sobreposição dos *"bounding boxes"* dos pontos com as linhas (expandindo o *"bounding box"* dos pontos em 20 metros) e que fossem excluídos os pontos que estavam a 20 metros de distância das agências (também expandindo os *"bounding boxes"* e eliminando tais recorrências). Também foi submetida a função a segunda faze do processamento espacial preciso utilização funções espaciais que filtravam os dados entre as distâncias do ponto a linha. Testes realizados utilizando apenas a primeira fase indexada retornou pontos muito além da distância de 20 metros definida pois o cruzamento dos *"bounding boxes"* não possui refinamento suficiente para garantir o fator distância estabelecido pois o uso de operadores espaciais sobre os *"bounding boxes"*, tais como &&, garantem os resultados apenas entre os *"bounding boxes"* mas não entre as geometrias em si, como é o caso visto no conceito de *"bounding boxes"*. Temos de nos lembrar do exemplo de

duas geometria esféricas que podem apresentar intersecção entre seus *"bounding boxes"* pelas aresta porém os objetos não se intersectam de fato. Este é o caso da figura 1em que pode haver intersecção do *"bounding box"* do boneco com o de outro objeto porém o boneco em si pode não ter sofrido intersecção.

Há a opção de se obter o mesmo resultado utilizando a técnica de *buffer*, que consiste em criar uma área espacial virtual a partir seja da linha ou do ponto, porém o custo computacional é superior se comparado a solução de expansão dos *"bounding boxes"* e submissão à consulta indexada que tem um custo computacional e tempo muito menor aplicados no resultado deste uso deste trabalho.

A expansão dos *"bounding boxes"* dos pontos com aplicação nos operadores espaciais, na primeira fase de pesquisa, é boa prática pois apresenta melhor performance se comparado ao uso puro de funções espaciais de cálculo de distância, apesentando economia aferida de 9/10 de tempo, aproximadamente, em relação a consulta sem a aplicação das técnicas de expansão e intersecção de *"bounding boxes"* em ambiente de testes neste caso de uso.

A segunda fase, fase de teste processamento espacial preciso, é quem tratá a acurácia do filtro espacial porem é importante que se leve em consideração o custo computacional de cada função de acordo com a estratégia de performance adotada. Funções que trazem o mesmo resultado no processamento espacial preciso, como a ST_Distance, ST_Intersects() ou ST_DWithin, vão apresentar tempos de processamento e performances diferentes.

```
SELECT * from linha ln
where
ln.geom &&
st_expand(st_transform(select dp.geom from detalhe_ponto dp
where dp.registro = 15421254, 3857), 20)
```

Atualização de índices espaciais e estatísticas das tabelas Em termos de funcionamento de inclusão, exclusão, alteração de registros e índices, em um cenário de exclusão de registros de uma dada tabela no SGBD, os dados não são apagados fisicamente mas sim marcados como inúteis por uma questão de performance (exclusão física dos dados é muito mais demorada) e o índice no instante permanece inalterado. Caso se verifique o tamanho do volume de armazenamento da tabela antes e depois de excluir uma grande quantidade de registros percebe-se a mesma área de uso de volume de dados e índices se mantêm. Quando ocorre inserções ou atualizações pode-se perceber o incremento de volume de dados utilizados, porém os índices permanecem os mesmos. Isto ocorre porque o banco não atualiza instantaneamente os índices e nem os registros excluídos não são fisicamente eliminados do banco no instalante.

Muitas alterações sem atualização dos índices e da composição das tabelas degradam a performance quando há busca de um dado indexado que não existe. Cada SGBD tem uma estratégia para tratar as questões de atualizações físicas das tabelas e seus índices.

O PostgresSQL possui nativamente funcionalidade que realiza atualização dos índices periodicamente para inclusões, exclusões e modificações. Há funcionalidade *"VACUUM ANALYZE"* que verifica dados excluídos, fazendo a exclusão física dos dados, seguido de atualização dos índices das tabelas tantos para os dados incrementados, excluídos e atualizados quando executado pelo administrados de banco de dados. Há também o processo automático que ocorre periodicamente ou por configuração de algum parâmetro, como por contador de registros por exemplo, que se chama *"AUTOVACCUM"*.

Quando se trabalha com bases de registros e operações muito dinâmicas (modificações constantes de registros no banco para inclusão, exclusão ou modificação), tem de ser adotar estratégias para que em um dado processo, na ocorrência de muitas atualizações, executar manualmente o *"VACUUM ANALYZE"*. Este efetiva a atualização dos índices e estatísticas das tabelas para não ocorrer o fato de um registro importante

que acabou de ser inserido não entrar no processamento de uma dada consulta ou execução de um script/função/procedimento (quando não se pode esperar a atualização automática do *"AUTOVACUUM"*). Esta última funcionalidade tem a capacidade de corrigir toda a tabela excluindo linhas em branco e outros tipos de "lixos" de dados que degradam a performance.

Operadores performáticos e *subquerys* O operador *"IN"* é utilizado para filtrar valores a partir de um conjunto de possibilidades definidas. Em termos práticos, ao utilizá-los em uma tabela como filtro de um campo do tipo inteiro, por exemplo, que se queira obter todos os registros que possuam um dado valor, toda a tabela será lida e todas as incidências daquele valor serão retornadas. Isso quer dizer toda a tabela será lida e o campo checado para cada registro (*"full table scan"*). Se em uma consulta o objetivo é trazer todos os registros que possuam um dado valor específico, o uso do operador *"IN"* é indispensável, porém nos casos em que se saiba que os valores não se repetem ou que se tenha a necessidade apenas de checar a existência do valor de interesse, este operador não é o mais adequado em termos de performance. Isso por que tendo em sua massa de dados valores que não se repetem ou que é necessário apenas a checagem da existência de registro a partir de um dado valor específico o uso do *"IN"* faz com que o banco cheque os dados em toda tabela mesmo que o dado já tenha sido encontrado alguma vez para procurar uma nova incidência daquele valor. Isso resulta em um maior esforço no banco de dados e maior custo computacional. Este custo certamente será sentido em termos de tempo de resposta. Se há um filtro de incidência entre muitos valores, por certo esse acréscimo de tempo de resposta entrará em uma progressão aritmética comprometendo a performance.

A ideia principal do uso do operado *"EXISTS"* consiste em checar booleanamente se um dado existe ou não a partir de um filtro estabelecido. Havendo incidência da

condição do filtro o valor verdadeiro é retornado e ali termina o processamento daquele filtro. A performance depende muito da cardinalidade, tipo de dado e tamanho das tabelas. A aplicação deste recurso é utilizado substituindo o operador "IN" por *"EXISTS"* seguido pela utilização de *"subquerys"* na consulta. Em termos práticos se substitui um *"full table scan"* por uma checagem do dado e encerramento do processamento caso o dado seja encontrado.

Neste projeto o uso do operador *"EXISTS"* foi utilizado como condição de checagem prévia de uma *subquery* que verifica, num primeiro momento, se o CEP possui vinculação com a linha geométrica, e num segundo momento pela consulta indexada de sobreposição da linha do CEP com o *"bounding box"* do ponto.

```
SELECT
unnest(ST_ClusterWithin(st_transform(b.geom,3857), 20)) gc,
cep,
numero_endereco
FROM (SELECT cep,
replace(trim(c.numero_endereco),'.','')::int as numero_endereco,
c.geom
from
detalhe_ponto c, ponto per, geo_agencia ga
where
c.numero_celula=per.numero_celula
and cod_agencia = lpad( numero_agencia, 12, ' ')
and textregexeq(trim(c.numero_endereco),'^[[:digit:]]+(\.[[:digit:]]+)?$')=true
and c.status = 'E'
and cep != lpad( '', 8, ' ')
and exists( select 1 from cep_base cb
where cb.cep=c.numero_cep)
and not geom && st_transform( st_expand( st_transform( ga.geom_ag, 3857), 20), 4326)
group by cep, replace(trim(c.numero_endereco),'.','')::int, geom) b
WHERE exists( select 1 from
(select cel.celula_geom from linha_geometria f
inner JOIN cep_base g ON f.registro_nu = g.registro_nu
inner JOIN linha cel ON f.numero_celula = cel.numero_celula
where g.cep = b.numero_cep) w
where
w.celula_geom && st_transform(st_expand(st_transform(b.geom, 3857), 20), 4326)
and ST_DistanceSphere(w.celula_geom,b.geom) <=20 limit 1)
AND g.numero_cep= b.cep) f
```

5.5.4 Particionamento de tabelas

O particionamento consiste em fragmentar uma tabela de grande tamanho em pedaços fisicamente menores com fim de se obter melhor performance de forma transparente logicamente pelas aplicações. Isso quer dizer que as aplicações não vão trabalhar com segmentos de tabelas, mas enxergar apenas uma única tabela lógica. O banco que é responsável por gerenciar e criar os segmentos com base em uma estratégia definida. O particionamento substitui as principais colunas de índices reduzindo o seu tamanho tornando mais provável que as partes altamente utilizadas dos índices se encaixem na memória. Além disso, as consultas aos dados fisicamente ocorrem de forma paralela, em algumas tecnologias, permitindo aumento significativo da performance [1].

O PostgresSQL possui duas formas de realizar particionamento: por faixa, quando realizada por coluna de chaves ou conjunto de colunas, ou por lista, que consiste informando quais valores de chave deve pertencer a qual partição.

A estratégia adota foi a de particionamento da tabela detalhe_ponto por faixa escolhendo o critério da data para se criar partições e manter estes dados agrupados. Nesta estratégia é criada uma tabela por cada mês e ano corrente, pois não seria possível usar particionamento em lista e remeter para a aplicação a inserção dos dados escolhendo a tabela física que o dado seria gravado em razão da dinâmica de funcionamento da aplicação.

Aplicando as técnicas de particionamento houve uma significativa redução nos tempos de consulta em alguns momentos em torno de 80%.

5.6 Aplicação de filtros espaciais

Os filtro espaciais foram escritos para eliminar pontos além de um raio delimitado da geolocalização da agência, pontos de entrega que além de um raio da linha indexada

do CEP e aqueles pontos em que, embora estivesse no sistema de referência WGS84 esferoide 4326, foram obtidos a partir de outro sistema de referência ou com aberrações das coordenadas de eixos de latitude e longitude irreais.

A área funcional definiu que os pontos elegíveis para entrar no processamento deveriam estar no limite máximo de 20 metros de distância da linha com CEP visando também tratar os casos dos pontos de entrega fora dos limites do CEP. A distância de 20 metros foi considerada considerando os casos que pontos de entrega não são necessariamente sobre a linha do CEP já que na dinâmica de entrega esta pode ocorrer dentro do terreno ou até mesmo em vielas com divisões dentro de um mesmo terreno havendo a necessidade de uma margem do endereço de entrega geocodificado com a linha do CEP.

Também foi definido que as linhas espaciais elegíveis para o processamento são as com vinculação com CEP por meio da tabela conexao_cep. O filtro espacial deve utilizar dados indexados com intersecção de *"bounding boxes"* na primeira fase e refinamento pelo filtro da distância dos dados já previamente filtrados na segunda fase. A estratégia se demonstrou muito eficiente pois elimina qualquer ponto que esteja fora dos parâmetros estabelecidos com perfeita acurácia.

```
exists( select 1 from
(select cel.celula_geom from linha_geometria f
inner JOIN cep_base g ON f.registro_nu = g.registro_nu
inner JOIN linha cel ON f.numero_celula = cel.numero_celula
where g.cep = b.numero_cep) w
where
w.celula_geom && st_transform(st_expand(st_transform(b.geom, 3857), 20), 4326)
and ST_DistanceSphere(w.celula_geom,b.geom) <=20 limit 1)
```

A estratégia para eliminar os pontos que estivem em um raio próximo as agências/-centros de distribuição objetivava eliminar todos os pontos que teriam sido dado baixa dentro das unidades nos casos que o CEP do endereço coincidisse com o da unidade,

porém, para isso, deveria se ter a geocodificação de todas as unidades com precisão. Como ainda há unidades em processo de geocodificação, todos os endereços com esta condição e com agência sem codificação estariam descartados. Como há muitas unidades não geocodificadas ou em processo de validação em interiores com agências únicas, regiões inteiras estariam fora do cálculo de processamento.Ainda na fase de desenvolvimento a regra foi suspensa e o processamento ocorreu para todos os pontos assumindo os casos em que entregas no mesmo CEP que as unidades de distribuição poderiam ter geocodificação de endereços coincidentes com as agências.

```
SELECT
cep, replace(trim(c.numero_endereco),'.','')::int as numero_endereco, c.geom
from
detalhe_ponto c, ponto per, geo_agencia ga
where
not geom && st_transform( st_expand( st_transform( ga.geom_ag, 3857), 20), 4326)
limit 1
```

Na fase de testes do *script* de processamento do trabalho, se deparou com o erro *"couldn't project point (30220 4052 0): latitude or longitude exceeded limits".* Foi identificado que mesmo o sistema apenas reconheço o DATUM WGS84 esférico, em um universo de bilhões de registros, 31 apresentaram eixos de latitude maior que 180 ou menor que -180, e longitude maior que 90 ou menor que -90 e foram descartados como aberrações. Não encontramos o motivo da falha na geração do ponto geocodificado, mas um filtro de contorno teve de ser adiciono para evitar dados errados checando a integridade do ponto que não ferisse os limites de longitude e latitude do DATUM. Acredita-se que um aplicativo corrompido possa ter gerado tais aberrações.

```
SELECT * from
from
detalhe_ponto c
where
ST_X(c.geom) between -180 and 180 and ST_Y(c.geom) between -90 and 90
```

5.7 Aplicação de funções espaciais

As funções espaciais objetivam obter informações a partir dos dados filtrados espacialmente armazenados no banco de dados. Tendo como estratégia principal extrair obter a geocodificação dos endereços a partir da geolocalização de entrega dos objetos para cada endereço referenciado pelo CEP e número de destino.

Com a incidência, e muitos casos, da existência de uma nuvem de pontos ao longo da linha do CEP, foi adotada a estratégia de *clusterizar* a nuvem de pontos e escolher o *"cluster"* com maior incidência de pontos e considerá-lo o mais significativo para posteriormente utilizar funções estatísticas. A aplicação de técnicas de *"clustering"* e escolha do resultado mais significativo foi adotada em razão de identificarmos, na faze piloto, que muitos objetos foram baixados no percurso do carteiro, sob alegação de necessidade de "agilidade" da entrega, portanto fora da proximidade da unidade de entrega. Tal ação para se ganhar "agilidade" produziu em diversos casos uma nuvem de pontos com diversas marcações fora do ponto de entrega impossibilitando a aplicação de uma função de obtenção do ponto mediano diretamente sobre a nuvem de pontos num processo mais simplificado.

Aplicada a função de clusterização do PostGis ST_ClusterWithin com escolha do *"clustes"* mais significativo, em seguida os dados são processados para extrair a informação do POI utilizando funções estatísticas estatísticos (ponto mediano).

Os testes iniciais foram realizado obtendo resultado a partir do cálculo simplificado

de obtenção do centroide a partir da coleção do *"array"* mais significativo, porém identificou-se pouca acurácia das informações geradas. Em novos testes alcançamos melhor acurácia dos pontos utilizando funções espaciais que levam em consideração questões estatísticas de mediana dos pontos.

É importante registrar que antes de se obter a informação do ponto de interesse a função de *"clusters"* escolhida retorna coleção de pontos, *"arrays"*, necessitando assim de um tratamento prévio para extração dos dados para posterior processamento estatístico.

5.7.1 Aplicação de clusterização

No caso específico deste trabalho o uso de *"clusters"*, pela aplicação de funções que identificam dados a partir de uma dada semelhança ou parâmetro, foi fundamental para agrupar pontos espaciais dentro de uma região de interesse e, a partir das suas características, obter a geocodificação do que seria o ponto de interesse real aproximado de endereços para geocodifica-los a partir de um nuvem de pontos. A aplicação desta técnica, compreendendo fatores de imprecisões de GPS ou captura de pontos em locais diferentes dos quais seria o ponto de interesse esperado, viabilizou o descarte de pontos geolocalizados que estavam fora dos locais esperados. Para aplicar os critérios de descarte supracitados, tendo disponíveis diversas funções de *"cluster"* do PostGis, seus algoritmos precisaram ser estudados e a partir das suas características o mais adequado foi escolhido.

Todos os pontos geocodificados obtidos utilizam sistema de referência esférico WGS84 SRID 4326, porém algumas funções espaciais funcionam apenas com *DATUM* projetado. Para utilização das funções de interesse os pontos geocodificados são transformados previamente utilizando a função ST_transform para o *DATUM* WGS84 projetado 3857 e, em seguida, armazenados em WGS84 esférico novamente após obtenção dos re-

sultados. Acredita-se que o histórico da escolha do WGS84 esférico se deu porque seria praticamente impossível configurar cada aparelho para a zona específica da localidade caso se adotasse um sistema *UTM*. Também não se tem o histórico do projeto o porquê da não adoção do sistema WGS84 projetado seis anos atrás.

O algoritmo de clusterização escolhido levou em conta que há elevada heterogeneidade de pontos por CEP e números de endereço de entrega. Isso quer dizer que há casos em que para um dado número de endereço de entrega e CEP há apenas um único registro e há casos em que existem centenas de registros. Outro fator a ser considerado é o tamanho em metros de cada *"cluster"* por que uma rua muito extensa pode ter vários *"clusters"* e o mais significativo, ou seja, com maior quantidade de pontos, deve ser escolhido.

Considerando os aspectos supracitados, o algoritmo de *"cluster"* não poderia posuir regras rígidas quanto ao quantitativo mínimo de *"clusters"* gerados, como acontece na função ST_ClusterKMeans, que embora se tenha uma excelente aproximação, nos casos que existam poucos registros espaciais, menores que a quantidade de *"clusters"*, não havia processamento dos dados e foi descartado. O ST_ClusterIntersecting não atendia o objetivo pois os *"cluster"* eram criados a partir de objetos espaciais que necessariamente se intersectam podendo não atender os objetivos finais e foi descartado. A função ST_ClusterDBSCAN trabalhava com fator distância do tamanho dos *"clusters"* como quantitativo mínimo que poderia ser ajustado para um mínimo de um único ponto mas a função ST_ClusterWithin foi escolhida porque trabalhava com o fator distância juntamente com índices se adequando melhor para extração das informações de interesse e apresentou em testes um desempenho um pouco melhor que as funções viáveis neste caso de uso. Isso não significa que a função ST_ClusterDBSCAN para outros tipos de dados não possa ser mais adequada.

A função ST_ClusterWithin foi aplicada a partir dos filtros de dados e espaciais

previamente já comentados utilizando a função *"unnest"* que converte os dados trazidos em um *"array"* de linhas :

```
(SELECT unnest(ST_ClusterWithin(st_transform(b.geom,3857), 20)) gc,
numero_cep,
numero_endereco
FROM( select numero_cep,
replace(trim(c.numero_endereco),'.','')::int as numero_endereco,
c.geom
from detalhe_percorrida c
where textregexeq(trim(c.numero_endereco),'^[[:digit:]]+(\.[[:digit:]]+)?$')=true
and c.status = 'E'
and c.numero_cep != lpad( '', 8, ' ')
and ST_X(geom) between -180 and 180
and ST_Y(geom) between -90 and 90
group by c.numero_cep,
replace(trim(c.numero_endereco),'.','')::int, geom) b
WHERE exists( select 1 from
(select cel.celula_geom from linha_geometria f
inner JOIN cep_base g ON f.registro_nu = g.registro_nu
inner JOIN linha cel ON f.numero_celula = cel.numero_celula
where g.cep = b.numero_cep) w
w.celula_geom && st_transform(st_expand(st_transform(b.geom, 3857), 20), 4326)
and ST_DistanceSphere(w.celula_geom,b.geom) <=20 limit 1)
group by numero_cep,numero_endereco) f
```

5.7.2 Obtenção de informação a partir do cluster

O uso de *"subquerys"* é interessante pois torna possível realizar tratamentos obtendo informações de campos das tabelas de interesse com funções de agregações de informações simultaneamente. Utilizando funções de agregação não se pode retornar dados

dos campos agrupados com algumas repetições de interesse. Neste caso foi utilizada a estratégia de usar *"subquerys"* com a função Row_number() associada com *"Partition"* para enumerar os *"clusters"* a partir da quantidade de geometrias e filtrá-las em seguida trazendo apenas os *"clusters"* mais significativos.

A função para se obter o ponto mediano a partir de um conjunto de pontos, a função ST_GeometricMedian, não possui a capacidade de trabalhar com coleções diretamente, sendo necessária a sua extração prévia para processamento consecutivo. O uso de *"subquerys"* é recomendado para este tipo de desenvolvimento.

```
(SELECT row_number() over () AS id,
numero_cep as cep,
numero_endereco as numeroendereco,
ST_NumGeometries(gc) qtpontos,
st_transform(ST_GeometricMedian(ST_CollectionExtract(gc,1)) ,4326) AS mediana,
row_number() OVER (PARTITION BY numero_cep, numero_endereco
ORDER BY ST_NumGeometries(gc) DESC)
FROM
(SELECT unnest(ST_ClusterWithin(st_transform(b.geom,3857), 20)) gc,
numero_cep,
numero_endereco
...) f
```

5.7.3 Extração do Ponto de Interesse

Utilizando também *"subquerys"* foi possível filtrar, dentro do conjunto de todos os *"clusters"* processados com agregação de dados, o dado mais significante a parti da ordenação dos *"clusters"* gerados com maior quantidade de pontos, retornando assim as informações de interesse deste trabalho. É uma grande vantagem que se necessita utilizar os resultados das funções de agregação (*"group by"*) mas não se pode usá-las

pois estas suprimem os dados dos campos pela própria função.

```
select
cep as edr_co_cep,
numeroendereco as edr_nu_endereco,
mediana as edr_cg,
qtpontos as edr_qt_pontos
from ( SELECT row_number() over () AS id,
...) a
where row_number = 1
ORDER BY EDR_QT_PONTOS desc
```

O *script* completo encontra-se disponível no apêndice 1.

5.8 Validação dos resultados

A área funcional foi a responsável pela validação dos resultados a partir da análise dos resultados como critérios:

- Avaliação dos resultados gerados nas cidades de Formosa-GO, São Paulo-SP Centro (com maior densidade) e Regiões periféricas (menor densidade) da Grande São Paulo;
- Avaliação de todos os bairros pelas áreas funcionais locais com apoio dos carteiros que realizam as rotas do distrito da localidade;
- Análise de quantitativo mínimo de 50 amostras de pontos por bairro em ruas de CEPs aleatórias (inexistindo o quantitativo indicado o mínimo existente será utilizado);
- Validação espacial do ponto que esteja até 20 metros da linha do CEP;
- Validação dos dados sem repetição de pontos com o mesmo número para o mesmo CEP;

5.9 Conclusão dos resultados

Após avaliação dos resultados gerados em ambiente de testes, o trabalho geocodificou 1.905.641 endereços e, posteriormente, em produção, geocodificou um total 2.469.349 endereços em 2 horas e 03 minutos de processamento apenas. Foi constatado que todos os pontos gerados atendiam todos os critérios estabelecidos sem encontrar nenhuma discrepância, inconsistências, repetição de número no mesmo CEP; apresentaram os *"clusters"* os mais significativos e inexistência de situações de pontos a mais de 20m da linha do CEP. Os resultados finais foram homologados pela área funcional e o objetivo de automatizar a geolocalização dos endereços como POI foi atendido analisando os resultados na cidade de Formosa-GO 8, São Paulo Capital 9 e periferia.

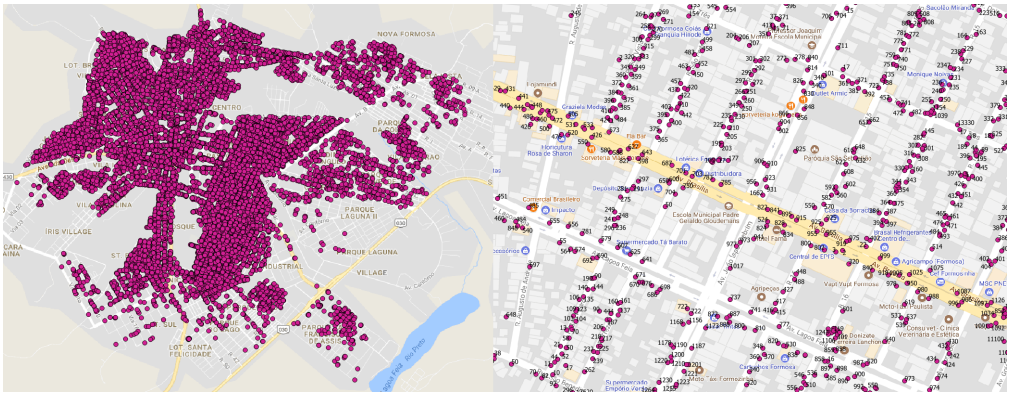

Figura 8: Resultado geocodificação em Formosa-GO

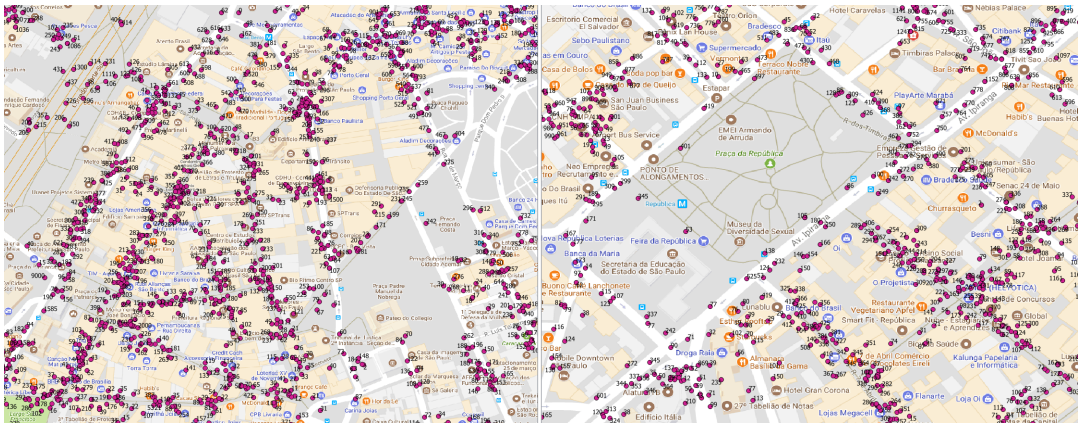

Figura 9: Resultado geocodificação em São Paulo-SP

O armazenamento das informações geradas foi efetivado em tabela para consulta qualitativa objetivando utilização pelo sistema de roteirização, que trabalha com pontos geocodificados, para ganho de maior eficiência nas rotas, economia de combustível, tempo, maior agilidade na entrega e eficiência do processo finalístico da empresa. A tabela com as informações também trouxe viabilidade para iniciar o projeto de Geo-Marketing dos Correios.

A partir da homologação uma rotina será criada com uma pequena modificação no código para que os endereços com CEP e número criados a partir de *"clusters"* com 50 pontos não entrarão em processamentos no futuro para diminuir eventuais cargas computacionais.

6 Conclusão

Este trabalho objetivou a obtenção de geocodificação de pontos de interesse, conhecido como POI, que seriam os pontos de endereços geocodificados a partir da base de dados de entrega de objetos postais registrados pelos carteiros no momento da entrega.

Vimos que para grandes volumes de dados é impossível a obtenção de resultados em tempo razoável ou sem esgotar os recursos computacionais caso não se utilize técnicas de otimização em banco de dados, também conhecido como *"tuning"*.

Este trabalho comparou os tipos de indexadores, como trouxe a luz comparativos entre funções espaciais de clusterização como forma de se obter a região com maior significação para processamento de cálculo estatístico de geolocalização mediada, a fim de obter o ponto de interesse de geocodificação de endereços com maior acurácia a partir de filtro de dados utilizando linguagem PgSQL com extensão PostGis, com uso de diversos filtros (dados e espaciais) associados a técnicas para agregação de dados de interesse.

Os resultados finais dos pontos de interesse com endereços geocodificados foram homologados pelas áreas funcionais com apoio dos gestores e carteiros que conhecem as regiões para validação das informações obtidas. A aceitação do resultado foi de 100%.

O processamento das informações a partir de um gigantesco volume de dados foi obtido em menos de duas horas satisfazendo as necessidades da empresa no tempo de levantamento de dados espaciais.

Os impactos deste trabalho resultarão em uma melhor performance do sistema de roteirização, em razão da sua dificuldade em geocodificar endereços e trabalho com o repasse de pontos geocodificados, diminuição do tempo de entrega, diminuição de esforço, economia de combustível e aumento da capacidade de entrega de objetos por percorrida do profissional. O projeto GeoMarketing poderá ser iniciado a partir da

existência da base de dados de endereços geolocalizados com associações qualitativas comerciais para potencializar e criar novas oportunidades de negócio do Correios.

7 Referências

Referências

[1] *Disposição das páginas de banco de dados PostgreSQL.* The PostgreSQL Global Development Group. Disponível em: <http://pgdocptbr.sourceforge.net/pg80/storage-page-layout.html>. Acessado em: 02 fev. 2018.

[2] Documentação do banco de dados do azure cosmos. Disponível em: <https://docs.microsoft.com/pt-br/azure/cosmos-db>. Acesso em: 02 fev. 2018.

[3] *Introdution to PosGIS - Spatial Indexing. Disponível em: <http://revenant.ca/www/postgis/workshop/indexing.html>. Acesso em: 02 fev. 2018.*

[4] *Spatial Relationships and Measurements - ST_ClusterDBSCAN. Disponível em: <postgis.net/-docs/ST_ClusterDBSCAN.html>. Acesso em: 02 fev. 2018.*

[5] Sql lexical structure (syntax). The PostgreSQL Global Development Group, Disponível em: <https://www.postgresql.org/docs/9.0/static/sql-syntax-lexical.html>. Acesso em: 02 fev. 2018.

[6] Marquez A. *Postgis Essentials.* PACKT PUB, 2015.

[7] Neufeld K.; Alam A. *Introdution to Postgis.* Refractions Research, 2010.

[8] Pinheiro L. C. Método de representção espacial de clustering. Master's thesis, UFPR, 2006.

[9] Korry D.; Susan D. *PostgreSQL: The comprehensive guide to building, programming, and administering PostgreSQL databases.* Sams Publishing, second edition, 2005.

[10] Dante C. J. *An Introduction to Database Systems.* Pearson/Addison Wesley, 2004.

[11] Stones R.; Matthew N. *Beginning Databases with PostgreSQL: From Novice to Professional.* SPRINGER A PR TRADE, 2007.

[12] Obe R. O.; Hsu L. S. *PostGIS in Action.* MANNING, 2015.

[13] F. Sadalage, P. J.; Fowler. *NoSQL Distilled.* Addison Wesley, 2012.

[14] Queiroz G. R.; Monteiro M. V. Bancos de dados geográficos e sistemas nosql: Onde estamos e para onde vamos. *Revista Brasileira de Cartografia*, 2013.

A Apêndice - Script SQL de obtenção de melhor POI

```
Script 1: 1-Calcula_coordenada.sql
Linguagem e programa compilador: PgSQL - Postgres/PostGis
#*********************************************************************
#       INICIO DO SCRIPT  1
#*********************************************************************
select
cep as ed_cep,
numeroendereco as ed_numero_endereco,
mediana as ed_geom,
qtpontos as ed_qt_ponto
    from (SELECT row_number() over () AS id,
        numero_cep as cep,
        numero_endereco as numeroendereco,
        ST_NumGeometries(gc) qtpontos,
        st_transform(ST_GeometricMedian(ST_CollectionExtract(gc,1)) ,4326) AS mediana,
        row_number() OVER (PARTITION BY numero_cep, numero_endereco
        ORDER BY ST_NumGeometries(gc) DESC)
        FROM

(SELECT unnest(ST_ClusterWithin(st_transform(b.geom,3857), 20)) gc,
numero_cep,
numero_endereco
FROM(select numero_cep,
 replace(trim(c.numero_endereco),'.','')::int as numero_endereco,
 c.geom
 from detalhe_percorrida c
 where textregexeq(trim(c.numero_endereco),
       '^[[:digit:]]+(\.[[:digit:]]+)?$')=true
 and c.status = 'E'
```

```
  and c.numero_cep != lpad( '', 8, ' ')
  and ST_X(geom) between -180 and 180
  and ST_Y(geom) between  -90 and 90
  group by c.numero_cep, replace(trim(c.numero_endereco),'.','')::int, geom) b
WHERE exists( select 1 from
       (select cel.celula_geom from  linha_geometria f
       inner JOIN cep_base g ON f.registro_nu = g.registro_nu
       inner JOIN linha cel ON f.numero_celula = cel.numero_celula
       where g.cep = b.numero_cep) w
       w.celula_geom && st_transform(st_expand(st_transform(b.geom, 3857), 20), 4326)
       and ST_DistanceSphere(w.colecao,b.geom) <=20  limit 1)

     group by numero_cep,numero_endereco) f

) a
where row_number = 1
ORDER BY ED_QT_PONTO desc

##############################################################################
#FIM DO SCRIPT 1
##############################################################################
```