Anais do XXVII Congresso Brasileiro de Cartografia e XXVI Exposicarta
6 a 9 de novembro de 2017, SBC, Rio de Janeiro - RJ, p. 1220-1225

AUTOMAÇÃO DE CADASTRO TÉCNICO DE USO COMPARTILHADO DE POSTES DE DISTRIBUIÇÃO

H. N. Uchoa[1], C. S. Sanz[2], F. S. N. Pinho[2], A. L. F. Sá[2], D. C. L. Mesquista[2]

[1]UTEI, Brasil
[2] CADIC, Brasil

Comissão VI - Sistemas de Informações Geográficas e Infraestrutura de Dados Espaciais

RESUMO

O objetivo do presente trabalho é apresentar um estudo de caso de aplicação de banco de dados espacial em plataforma de código aberto para automação do cadastro técnico do uso compartilhado de postes. A solução projetada foi adotada em alguns estados do Brasil gerenciando informações relacionadas à infraestrutura de transmissão de dados e voz de alguns milhões de postes de distribuição de energia elétrica. Este trabalho irá apresentar conceitos, soluções práticas e tecnologias adotadas no projeto. As principais plataformas tecnológicas utilizadas neste projeto foram o PostgreSQL (PostGIS) e o MapServer, sendo planejada a integração do GeoServer ainda para o ano de 2017. O principal resultado do projeto foi a otimização do processo de campo e de gabinete com grande redução de tempo em comparação às antigas metodologias analógicas ainda adotadas por algumas empresas.

Palavras-chave: SIG, Código Aberto, Software Livre, MapServer, GeoServer, Banco de Dados Espacial, Telecomunicações.

ABSTRACT

The purpose of this paper is to explain a case study of spatial database application in open source platform for automation of the technical register of the shared use of poles. The projected solution was adopted in some states of Brazil managing information related to the data and voice transmission infrastructure of some millions of electricity distribution poles. This work will present concepts, practical solutions and technologies adopted in the project. The main technological platforms used in this project were PostgreSQL (PostGIS) and MapServer, and the integration of GeoServer is planned for the year 2017. The main result of the project was the optimization of the process of survey and office processing with great reduction of time in comparison to the old analog methodologies still adopted by some Brazilian companies.

Keywords: GIS, Open Source, Free Software, MapServer, GeoServer, Sspatial Database, Telecommunications.

1- APRESENTAÇÃO

Atualmente uma significativa parcela de internet que é distribuída para os brasileiros é feita através de cabos aéreos que são lançados através dos postes de distribuição de energia. As empresas de telecomunicações que desejam utilizar esta infraestrutura das concessionárias do setor elétrico precisam pagar um valor mensal pelo uso de cada poste, sendo este processo regulamentado pela Resolução Conjunta nº 4, de 16 de dezembro de 2014 (Aneel e Anatel).

As concessionárias precisam gerenciar o uso dos postes pelas empresas de telecomunicações e contratam periodicamente um censo de uso compartilhado para identificar quais empresas estão utilizando os postes, retirando os cabos “piratas” (sem contrato) e cobrando os valores corretos das empresas que possuem contrato. O processo utilizado neste mapeamento de cabos aéreas evoluiu passando por algumas fases:

1. Processo de levantamento em campo por meio analógico (registro em formulários em papel), com consolidação de informações finais em planilhas eletrônicas.

2. Processo de levantamento em campo com dispositivo digital (PDA, smartphones, etc), com consolidação de informações finais em planilhas eletrônicas e/ou banco de dados alfanumérico (não espacial).
3. Processo orientado a banco de dados espacial/geográfico, com o levantamento de campo sendo feito com um aplicativo que organiza as informações num banco de dados local que posteriormente alimenta um banco de dados espacial centralizado.

Atualmente ainda existem empresas que trabalham com as metodologias 1 e 2. O foco deste trabalho é apresentar a metodologia 3 com ênfase ao poder de análise proveniente do uso do banco de dados espacial.

2- ARQUITETURA

A evolução do modelo analógico para o modelo digital exige uma mudança de paradigma, sendo maior esta mudança quando passamos a trabalhar com um banco de dados espacial/geográfico.

A primeira evolução conceitual importante do projeto foi pensar no processo de censo de forma contínua, evoluindo o conceito apresentado pelo próprio contratante do serviço (empresas concessionárias de distribuição de energia). Neste sentido, o novo modelo conceitual foi implementado no banco de dados espacial tendo o poste como uma entidade cujo ciclo de vida passaria a ter rastreabilidade espaço temporal.

Este novo conceito exigiu um desenho específico da arquitetura de software, pois os dados tratados nos trabalhos de campos pela contratada precisavam estar vinculados com os dados da contratante. Desta forma, todos os processos e sistemas foram adequados para garantir a vinculação dos postes georreferenciados, ou seja, a informação interna da concessionária deveria estar compatível com o levantamento de campo. A nova solução foi batizada de GeoPoste (http://geoposte.com.br/) e rapidamente foi percebida pelos contratantes como uma significativa inovação em relação ao processo antigo.

Para suportar todas as exigências técnicas, a estrutura inicial contou com 2 servidores, separando o servidor de aplicativos do servidor de banco de dados (SGBD espacial).

Fig. 1 - Organização inicial de servidores e sistemas

O processo de automação foi estruturado utilizando-se principalmente programação em shell script (Linux), possibilitando o tratamento de dados de diversas fontes sem a interferência humana e com alta performance, pois shell script roda diretamente no nível do Sistema Operacional. Através do shell script é possível chamar nativamente inúmeras bibliotecas, incluindo as que tratam dados geoespaciais, possibilitando automação de complexos processos.

O shell script também foi a base para a construção da ferramenta ETL (*Extract, Transform, Load* - Extração, Transformação e Carregamento) para solução GeoPoste. Este componente foi denominado GeoETL e representa um conjunto de scripts que rodam de duas formas:

- Autônoma: são executados em determinados horários sem intervenção humana.
- Assistida: são executados através de uma ação humana.

A figura a seguir apresenta alguns componentes da solução, destacando o papel do GeoETL.

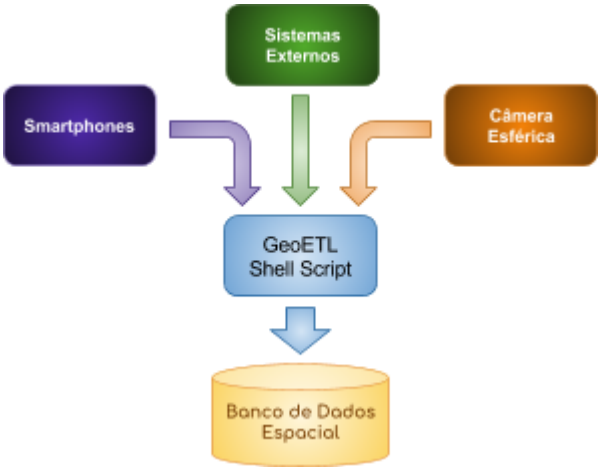

Fig. 2 - Funcionamento do GeoETL

3- AUTOMAÇÃO: ALGORITMOS TOPOLÓGICOS

O verdadeiro potencial do SGBD Geoespacial ainda é pouco explorado no Brasil. Independente da marca do SGBD, em geral os usuários utilizam o banco de dados geoespacial (BDGeo) basicamente como um repositório de dados e fazem poucas análises topológicas.

No presente projeto, inúmeros processos foram automatizados com o uso das funcionalidades do BDGeo e um dos principais algoritmos tratou de uma análise para inferência de empresas em postes. O desafio que foi solucionado com o uso de análises geográficas e topológicas era o seguinte: apresentados 3 (três) postes numa sequência onde uma empresa aparece no primeiro e no terceiro poste, então essa empresa provavelmente está também no segundo poste. A malha de pontos não fornecia informação sobre o trajeto dos cabos aéreos, impossibilitando que o problema fosse tratado como uma análise linear de uma sequência de postes. A solução final foi implementada através da linguagem procedural nativa do PostgreSQL (PL/pgSQL) combinando uma modelagem matemática com as funções do PostGIS.

O algoritmo (modelo matemático) será detalhado a seguir:

3.1- DADOS DE ENTRADA

I. Menor ângulo para alinhamento: Valor em graus que indica o menor ângulo a ser aceito para considerar que os 3 postes analisados estão alinhados. Na figura a seguir, este valor está indicado pela letra **A**.

II. Distância entre postes: Valor em metros que indica qual a distância máxima entre postes a ser aceita para considerar que os mesmos sejam próximos e possam fazer parte de uma mesma sequência. Na figura a seguir, este valor está indicado por **d1** e **d2**.

III. Ângulo para teste do paralelismo: Valor em graus que indica qual o valor máximo do ângulo entre os postes e os arruamentos mais próximos. Na figura a seguir, este valor está indicado pela letra **B**.

IV. Distância para análise do paralelismo: Valor em metros que representa a distância do poste para o eixo dos logradouros (rua, avenida, etc) mais próximos para análise do paralelismo. Na figura a seguir, este valor está indicado por **d3**.

V. Dias passados da última edição do quadrante: Este valor serve para garantir que o censo do quadrante já tenha sido concluído há alguns dias para evitar um processamento de um quadrante ainda em fase de execução do censo. Se o valor for 5, por exemplo, somente será processado o quadrante no qual a última edição do censo tenha sido feita há mais de 5 dias. Este parâmetro é importante somente para a execução em lote (grande quantidade de quadrantes de uma só vez).

Fig. 3 - Representação gráfica (no mapa) dos elementos utilizados no algoritmo.

3.2- MODELO MATEMÁTICO

I. A partir da distância máxima entre os postes o algoritmo localiza os postes do entorno que atendem ao critério de distância máxima (figura a seguir).

II. Depois de localizar os postes próximos, o algoritmo faz uma combinação de 2 a 2 para analisar a condição de alinhamento entre o poste em análise e os outros 2 localizados nas proximidades do primeiro. Na figura anterior, é indicado por **A**.

III. Se a condição for correta, o algoritmo analisa as empresas presentes nos 3 postes.

IV. Se houver empresas faltando no poste em análise, o algoritmo analisa a condição do paralelismo (ângulo **B** da figura anterior) e da distância para rua (**d3**).

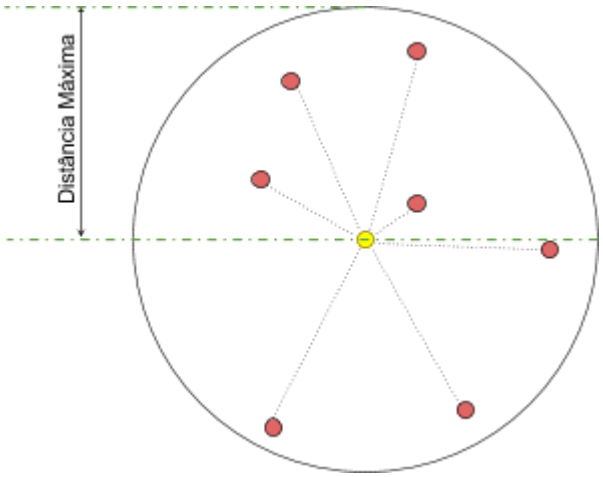

Fig. 4 - Identificação dos postes que se encontram a uma determinada distância.

3.3- CÓDIGO DA SOLUÇÃO ADOTADA

```
CREATE OR REPLACE FUNCTION cadic001_identifica_empresas_inferencia (angulo_menor DOUBLE PRECISION, dias INTEGER, distancia_postes DOUBLE PRECISION, id_quadrante_selecionado INTEGER, angulo_paralelismo DOUBLE PRECISION, distancia_paralelismo DOUBLE PRECISION) RETURNS VARCHAR AS $$
-- Objetivo: função para identificar as empresas que podem estar sem identificação em determinados postes.
--       O algoritmo se baseia na análise do poste anterior e posterior que precisam formar um ângulo
--       maior que o 'algulo_menor' (graus, recomendado: 165, ou seja, 15 graus de variação)
-- ChangeLog (v1.09): criacao dos recursos para gravacao das analises espaciais na tabela (buffer zone).
-- Execução: SELECT cadic001_identifica_empresas_inferencia (165, 15, 35, $quad, 15, 15);
-- Para forçar que o algoritmo identifique o maior vão, devemos colocar o valor igual a 0:
-- SELECT cadic001_identifica_empresas_inferencia (165, 15, 0, 48441, 15, 15);

DECLARE
        last_id_poste                   INTEGER;
        last_id_cabo                    INTEGER;
        contador                        INTEGER;
        id_municipio_var                INTEGER;
        id_status_var                   INTEGER;
        distancia_vao                   DOUBLE PRECISION;
        distancia_vao_maior             DOUBLE PRECISION := 0;
        dados_postes                    RECORD;
        dados_quadrantes                RECORD;
        dados_postes_atendem            RECORD;
        dados_empresas_inferencia       RECORD;
        dados_paralelismo               RECORD;
        dados_maior_vao                 RECORD;
        conta_paralelismo               INTEGER;
        postes_entre_postes             INTEGER;
BEGIN
  -- percorre os quadrantes para identificar se possuem um censo executado há mais de XX dias ou estão com inferencia_forca = TRUE
  FOR dados_quadrantes IN SELECT id_quadrante, max(p.modified) data_modificacao, DATE_PART('day', now() - max(p.modified)) tempo_dias FROM postes p, quadrantes q WHERE (q.inferencia_forca = TRUE OR q.inferencia_data IS NULL) AND ST_Contains(q.geom, p.geom_atual) GROUP BY id_quadrante HAVING max(p.modified) IS NOT NULL AND DATE_PART('day', now() - max(p.modified)) >  dias AND id_quadrante = id_quadrante_selecionado LOOP
    RAISE NOTICE '[1]: [INF_EMP] Processando quadrante: %',dados_quadrantes.id_quadrante;
    -- Se o valor da distância do vão for igual a zero, então calcula o valor do maior vão do quadrante
    IF distancia_postes = 0 THEN
      -- Este loop identifica o maior vão dentro de um quadrante
      FOR dados_maior_vao IN SELECT id_poste, geom_atual geom FROM postes p WHERE quadrante_id = dados_quadrantes.id_quadrante LOOP
        -- Pega, a partir de um poste, a segunda maior distância de vão.
        SELECT INTO distancia_vao st_distance(ST_Transform(p1.geom_atual, 3857), ST_Transform(p2.geom_atual, 3857)) AS distancia
          FROM postes p1, postes p2
          WHERE p1.quadrante_id = dados_quadrantes.id_quadrante AND p2.quadrante_id = dados_quadrantes.id_quadrante AND p1.id_poste <> p2.id_poste
            AND p1.id_poste = dados_maior_vao.id_poste
          ORDER BY st_distance(p1.geom_atual, p2.geom_atual) LIMIT 1 OFFSET 1;
        IF distancia_vao > distancia_vao_maior THEN
          RAISE NOTICE ' [1.1]: [INF_EMP] Distancia analisada: %',distancia_vao;
          distancia_vao_maior := distancia_vao;
        END IF;
      END LOOP;
      RAISE NOTICE ' [1.2]: [INF_EMP] Distancia vao: %',distancia_vao_maior;
      distancia_postes := distancia_vao_maior;
    END IF;     -- condição da distancia_poste = 0

    FOR dados_postes IN SELECT p.id_poste, p.geom_atual FROM postes p, quadrantes q WHERE q.id_quadrante = dados_quadrantes.id_quadrante AND ST_Contains(q.geom, p.geom_atual) LOOP
      RAISE NOTICE '   [2]: [INF_EMP] Processando poste: %',dados_postes.id_poste;
      -- selecionar os postes que estão a uma distância inferior a distancia_postes
      -- montar as combinações possíveis com os postes mais próximos
      FOR dados_postes_atendem IN
        WITH lados_triangulo AS (
          WITH seleciona_postes AS (
            WITH RECURSIVE t(i) AS (SELECT unnest(array_agg(p2.id_poste)::text[]) FROM postes p1, postes p2
              WHERE p1.id_poste = dados_postes.id_poste AND p2.id_poste <> dados_postes.id_poste
                AND st_distance(ST_TRANSFORM(p1.geom_atual, 3857), ST_TRANSFORM(p2.geom_atual, 3857)) < distancia_postes) ,cte AS (
                  SELECT i AS combo, i, 1 AS ct
                    FROM t
                      UNION ALL
                        SELECT cte.combo || ','|| t.i, t.i, ct + 1
                          FROM cte
                          JOIN t ON t.i > cte.i
                  )
            SELECT dados_postes.id_poste::INTEGER vertice_a, (('{' || combo || '}')::text[])[1]::INTEGER vertice_b, (('{' || combo || '}')::text[])[2]::INTEGER vertice_c FROM cte WHERE array_length(('{' || combo || '}')::text[],1)=2 ORDER BY ct, combo
            )
            SELECT ST_DISTANCE (ST_TRANSFORM(b_geom, 3857), ST_TRANSFORM(c_geom, 3857)) a_dist,
              ST_DISTANCE (ST_TRANSFORM(a_geom, 3857), ST_TRANSFORM(c_geom, 3857)) b_dist,
              ST_DISTANCE (ST_TRANSFORM(a_geom, 3857), ST_TRANSFORM(b_geom, 3857)) c_dist, vertice_a, vertice_b, vertice_c FROM
                (SELECT (SELECT geom_atual FROM postes WHERE id_poste = vertice_a) a_geom,
                  (SELECT geom_atual FROM postes WHERE id_poste = vertice_b) b_geom,
                  (SELECT geom_atual FROM postes WHERE id_poste = vertice_c) c_geom,
                  vertice_a, vertice_b, vertice_c FROM seleciona_postes) AS foo
          )
          SELECT DEGREES(ACOS( (POWER(b_dist, 2) + POWER(c_dist, 2) - POWER(a_dist, 2)) / (2 * b_dist * c_dist) )) angulo, vertice_a, vertice_b, vertice_c, a_dist, b_dist, c_dist FROM lados_triangulo WHERE DEGREES(ACOS( (POWER(b_dist, 2) + POWER(c_dist, 2) - POWER(a_dist, 2)) / (2 * b_dist * c_dist) )) >= angulo_menor LOOP
        -- inicio do LOOP dos postes que atendem aos critérios
        RAISE NOTICE '        [3]: [INF_EMP] Processando conjunto: %, %, % -- ANG: %',dados_postes_atendem.vertice_a, dados_postes_atendem.vertice_b, dados_postes_atendem.vertice_c, dados_postes_atendem.angulo;
        -- LOOP para itentificar as empresas que estão em B e C, mas não estão no A
        FOR dados_empresas_inferencia IN
          WITH seleciona_empresas AS (
            SELECT id_poste, e.descricao, e.id_empresa, '{' || dados_postes_atendem.vertice_a || '}' vertice_a FROM cabos c
              JOIN postes p ON (poste_id = id_poste) AND id_poste IN (dados_postes_atendem.vertice_a, dados_postes_atendem.vertice_b, dados_postes_atendem.vertice_c)
              JOIN empresas e ON (empresa_id = id_empresa) AND id_empresa <> 41
              GROUP BY id_poste, e.descricao, e.id_empresa
```

```
 86              ORDER BY id_poste
 87            )
 88          SELECT id_empresa, array_agg(id_poste)
postes_agrupados
 89            FROM seleciona_empresas
 90            GROUP BY id_empresa, vertice_a
 91            HAVING array_length(array_agg(id_poste),1) = 2 AND
NOT (array_agg(id_poste) &&  vertice_a::INTEGER[]) LOOP
 92          -- Inseri a empresa que foi inferida
 93          RAISE NOTICE '              [4]: [INF_EMP] Empresa
Selecionada: % - % - %',dados_empresas_inferencia.id_empresa,
dados_empresas_inferencia.postes_agrupados[1],
dados_empresas_inferencia.postes_agrupados[2];
 94
 95          -- Analisa se existe algum algum poste entre os 2
postes selecionados.
 96          -- Se existir, tem que descartar o vão pois ele pode
ter pulado um poste.
 97          -- Utilizar um buffer zone com endcap=flat, exemplo:
 98          -- SELECT ST_Buffer(ST_GeomFromText('LINESTRING(50
50,150 150,150 50)'), 10, 'endcap=flat join=round');
 99          SELECT INTO postes_entre_postes count(*) FROM postes
p1, postes p2, postes p3 WHERE p1.id_poste =
dados_postes_atendem.vertice_b AND p2.id_poste =
dados_postes_atendem.vertice_c AND p3.id_poste NOT IN
(dados_postes_atendem.vertice_b, dados_postes_atendem.vertice_c)
AND ST_Contains(ST_Buffer(ST_MakeLine(ST_Transform(p1.geom_atual,
3857), ST_Transform(p2.geom_atual, 3857)), 10, 'endcap=flat
join=round'), ST_Transform(p3.geom_atual, 3857));
100         RAISE NOTICE '              [4.1]: [INF_EMP]
postes_entre_postes: %',postes_entre_postes;
101
102         -- Somente analisa o paralelismo caso
103         IF postes_entre_postes = 0 THEN
104           -- analisa se existe algum paralelismo com as ruas
mais proximas (consultar a tabela ways)
105           -- Inicio do LOOP do paralelismo
106           conta_paralelismo := 0;
107           FOR dados_paralelismo IN
108             WITH ajusta_angulos AS (
109               WITH gera_angulos AS (
110                 WITH selecao_poste_inferidos AS (
111                   SELECT
112                       p.id_poste, ST_TRANSFORM(p.geom_atual,
900913) geom_poste,
113
ST_MakeLine(ST_Transform(p1.geom_atual,900913),
ST_Transform(p2.geom_atual,900913)) geom_line,
114
degrees(ST_Azimuth(ST_Transform(p1.geom_atual,900913),
ST_Transform(p2.geom_atual,900913))) AS deg_ang_poste, p1.id_poste
idp1, p2.id_poste idp2
115                     FROM cabos ci
116                     JOIN postes p ON (p.id_poste =
dados_postes.id_poste)
117                     JOIN postes p1 ON (p1.id_poste =
dados_empresas_inferencia.postes_agrupados[1])
118                     JOIN postes p2 ON (p2.id_poste =
dados_empresas_inferencia.postes_agrupados[2])
119                     LIMIT 1
120                   )
121                   SELECT id_poste, geom_poste, geom_line,
osm_id, deg_ang_poste, way,
122                     st_distance(ST_TRANSFORM(geom_poste,
900913), way) as distancia,
123                     idp1, idp2
124                   FROM selecao_poste_inferidos,
planet_osm_line
125                   WHERE id_poste = dados_postes.id_poste
AND name not in ('footway', 'steps' , 'pedestrian', 'cycleway',
'construction')
126                   ORDER BY ST_TRANSFORM(geom_poste, 900913)
<-> way LIMIT 40
127                 )
128                 SELECT DISTINCT id_poste, osm_id,
deg_ang_poste, idp1, idp2,
129                   CASE WHEN ST_LineLocatePoint(way,
geom_poste) >= 0.5
130                   THEN
degrees(ST_Azimuth(ST_StartPoint(ST_LineSubstring(way,
ST_LineLocatePoint(way, geom_poste)-0.09, ST_LineLocatePoint(way,
geom_poste)-0.01)), ST_EndPoint(ST_LineSubstring(way,
ST_LineLocatePoint(way, geom_poste)-0.09, ST_LineLocatePoint(way,
geom_poste)-0.01))))
131                   ELSE
degrees(ST_Azimuth(ST_StartPoint(ST_LineSubstring(way,
ST_LineLocatePoint(way, geom_poste)+0.01, ST_LineLocatePoint(way,
geom_poste)+0.09)), ST_EndPoint(ST_LineSubstring(way,
ST_LineLocatePoint(way, geom_poste)+0.01, ST_LineLocatePoint(way,
geom_poste)+0.09))))
132                   END deg_ang_rua,
133                   CASE WHEN ST_LineLocatePoint(way,
geom_poste) >= 0.5
134                   THEN ABS(deg_ang_poste -
degrees(ST_Azimuth(ST_StartPoint(ST_LineSubstring(way,
ST_LineLocatePoint(way, geom_poste)-0.09, ST_LineLocatePoint(way,
geom_poste)-0.01)), ST_EndPoint(ST_LineSubstring(way,
ST_LineLocatePoint(way, geom_poste)-0.09, ST_LineLocatePoint(way,
geom_poste)-0.01)))))
135                   ELSE ABS(deg_ang_poste -
degrees(ST_Azimuth(ST_StartPoint(ST_LineSubstring(way,
ST_LineLocatePoint(way, geom_poste)+0.01, ST_LineLocatePoint(way,
geom_poste)+0.09)), ST_EndPoint(ST_LineSubstring(way,
ST_LineLocatePoint(way, geom_poste)+0.01, ST_LineLocatePoint(way,
geom_poste)+0.09)))))
136                   END diferenca,
137                   distancia FROM gera_angulos WHERE distancia
<= distancia_paralelismo
138               )
139               SELECT id_poste, osm_id, deg_ang_poste,
deg_ang_rua, diferenca, idp1, idp2,
140                 CASE WHEN diferenca > 180
141                   THEN 360 - diferenca
142                   ELSE diferenca
143                   END dif_corrigido,
144                   distancia FROM ajusta_angulos LOOP
--LOOP do paralelismo
145
146             RAISE NOTICE '              [5]: [INF_EMP]
osm_id %, deg_ang_poste %, deg_ang_rua %, dif_ang %, idp1 %, idp2
%, distancia %', dados_paralelismo.osm_id,
dados_paralelismo.deg_ang_poste, dados_paralelismo.deg_ang_rua,
dados_paralelismo.dif_corrigido, dados_paralelismo.idp1,
dados_paralelismo.idp2, dados_paralelismo.distancia;
147
148             -- analisa condições do paralelismo:
149             IF (dados_paralelismo.distancia <=
distancia_paralelismo AND dados_paralelismo.dif_corrigido <=
angulo_paralelismo) THEN
150               conta_paralelismo := 1;
151             END IF;
152
153           END LOOP;           -- Final do Loop do paralelismo
154
155           IF (conta_paralelismo > 0) THEN
156             RAISE NOTICE '                [6]: [INF_EMP]
Angulo e distancia atendem criterios. Inseri Empresa';
157               -- verifica se o registro já possui a empresa
e, caso não possua, inseri a nova empresa
158               INSERT INTO cabosinferidos
159                 (empresa_id, poste_id, poste_id_ref1,
poste_id_ref2, created, modified)
160                 SELECT
dados_empresas_inferencia.id_empresa, dados_postes.id_poste,
161
dados_empresas_inferencia.postes_agrupados[1],
dados_empresas_inferencia.postes_agrupados[2],
162                   NOW(), NOW()
163                 WHERE
164                   NOT EXISTS (
165                     SELECT empresa_id, poste_id FROM
cabosinferidos
166                       WHERE empresa_id =
dados_empresas_inferencia.id_empresa
167                       AND poste_id = dados_postes.id_poste
168                   );
169           END IF;
170
171         END IF;       -- final do IF postes_entre_postes = 0
172
173       END LOOP;       -- Final do Loop das empresas inferidas
(dados_empresas_inferencia)
174     END LOOP;         -- Final do Loop dos postes que atendem
aos critérios
175   END LOOP;   -- Final do Loop dos postes
176   -- atualiza parâmetros do quadrante
177   UPDATE quadrantes SET inferencia_executado = TRUE,
inferencia_data = NOW() WHERE id_quadrante =
dados_quadrantes.id_quadrante;
178 END LOOP;     -- Final do Loop dos quadrantes (principal)
179
180 RETURN 'Processamento Concluído';
181 END;
182 $$ LANGUAGE plpgsql;
```

4- CONCLUSÃO

A implementação de modelos matemáticos no nível do banco de dados geoespacial com funções geográficas e topológicas fazendo o uso adequado da indexação espacial (GIST) possibilitou a construção de

uma solução com elevado nível de automação e alta performance.

A próxima fase do projeto irá integrar o GeoServer à solução para agilizar a criação de relatórios que exigem apresentação de mapas, deixando o MapServer ainda como parte da arquitetura.

REFERÊNCIAS BIBLIOGRÁFICAS

Resolução Conjunta nº 4, de 16 de dezembro de 2014 (Aneel e Anatel), link acessado em 25/09/2017: http://www.anatel.gov.br/legislacao/resolucoes/resolucoes-conjuntas/820-resolucaoconjunta-4

Site oficial do PostgreSQL, link acessado em 25/09/2017: https://www.postgresql.org/

Site oficial do PostGIS, link acessado em 25/09/2017: http://postgis.net/