Revista Brasileira de Cartografia (2016), Nº 68/8: 1531-1540
Sociedade Brasileira de Cartografia, Geodésia, Fotogrametria e Sensoriamento Remoto
ISSN: 1808-0936

AGREGAÇÃO HIERÁRQUICA DE GEO-OBJETOS: UMA ABORDAGEM BASEADA EM *SOFTWARE* LIVRE

Hierarchical Aggregation of Geo-objects: A Free Software based approach

Luiz Claudio Oliveira de Andrade, Philipe Borba & Maurício Carvalho Mathias de Paulo

Diretoria de Serviço Geográfico – DSG
2° Centro de Geoinformação – 2° CGEO
EPCT Km 4,5 DF 001 - Setor Habitacional Taquari - Lago Norte - CEP: 71559-901 - Brasília, DF, Brasil
{luizclaudio.andrade, borba.philipe, mauricio.paulo@eb.mil.br

Recebido em 10 de Fevereiro, 2016/ Aceito em 7 de Junho, 2016
Received on February 10, 2016/ Accepted on June 7, 2016

RESUMO

A utilização da Especificação Técnica para Estruturação de Dados Geoespaciais Vetoriais (ET-EDGV) já é uma realidade para várias instituições produtores de dados cartográficos no Brasil. O Decreto 6.666/08 institui a Infraestrutura Nacional de Dados Geoespaciais (INDE) e determina que todos os órgãos e entidades do Poder Executivo Federal devem seguir as normas e padrões homologados pela INDE. Neste sentido, a Diretoria de Serviço Geográfico do Exército Brasileiro (DSG) desenvolveu uma solução baseada em *software* livre para prover, de forma interoperável, a criação, edição e remoção de agregações hierárquicas de geo-objetos. Tais agregações são chamadas de Elementos Complexos. Os Elementos Complexos estendem a noção de feição geográfica, permitindo uma agregação semântica de geo-objetos com diferentes primitivas geométricas. Neste trabalho é mostrada a solução desenvolvida para o *software* livre *QGIS* utilizando o *PostGIS* e o *SpatiaLite* como bancos de dados.

Palavras Chave: EDGV, *QGIS*, Complexos, *PostGIS*, *SpatiaLite*, INDE.

ABSTRACT

The Technical Specification for Vector Geospatial Data Structure (ET-EDGV) usage is already a reality for several cartographic data producing institutions in Brazil. The Decree 6,666/08 stablishes the Geospatial Data Infrastructure (INDE) and determines that all entities and organizations of the Federal Executive Branch of Government must follow the norms and standards approved by INDE. In this sense, the Brazilian Army Geographical Service Bureau (DSG) developed a free software based solution to provide, in an interoperable way, the creation, edition and removal of hierarchical aggregations of geo-objects. Such aggregations are called Complex Elements. Complex Elements extend the notion of geographical feature, allowing semantic aggregation of geo-objects with different geometric primitives. In this paper it's shown a solution developed for the *QGIS* free *software* using *PostGIS* and *SpatiaLite* as databases.

Keywords: EDGV, *QGIS*, Complexes, *PostGIS*, *SpatiaLite*, INDE.

1. INTRODUÇÃO

A Infraestrutura de Dados Espaciais (INDE) foi instituída pelo Decreto 6.666/08. Tal decreto prevê que, no âmbito do Poder Executivo Federal, todos os órgão produtores de dados cartográficos no Brasil devam obedecer aos padrões estabelecidos para a INDE e às normas Cartográficas Nacionais (BRASIL, 2008).

Como um dos resultados dos trabalhos desenvolvidos nesse sentido, foi criada a Especificação Técnica para Estruturação de Dados Geoespaciais Vetoriais, a ET-EDGV, para padronizar estruturas de dados e seu compartilhamento de modo interoperável e para racionalizar recursos entre os produtores de geoinformação. Tal padronização é materializada por meio de um modelo conceitual onde estão documentados os Diagramas de Classe dos dados geoespaciais, referentes ao mapeamento sistemático para escalas de 1:25.000 até 1:250.000, que são agrupados em 13 Categorias de Informação (CONCAR, 2010), a saber:

a. Hidrografia;
b. Relevo;
c. Vegetação;
d. Sistema de Transporte;
e. Energia e Comunicações;
f. Abastecimento de Água e Saneamento Básico;
g. Educação e Cultura;
h. Estrutura Econômica;
i. Localidades;
j. Pontos de Referência;
k. Limites;
l. Administração Pública; e
m. Saúde e Serviço Social.

A versão 2.1.3 da ET-EDGV, última homologada pela CONCAR, modela todas as classes de objetos presentes nas 13 Categorias de Informação presentes nela. As classes geométricas da EDGV podem ter primitivas geométricas do tipo Ponto, Linha e Polígono. Adicionalmente está prevista a primitiva geométrica chamada Complexo (CONCAR, 2010).

A forma de trabalho com as primitivas Ponto, Linha e Polígono já é conhecida pelos produtores de dados cartográficos, porém, a forma de trabalho com a primitiva Complexo, definida pela EDGV, ainda necessita de maior disseminação. Desta forma, este trabalho propõe uma forma interoperável de trabalhar com Elementos Complexos em ambiente livre usando os Bancos de Dados *PostGIS* e *SpatiaLite* apoiados no *software* livre *QGIS*.

Diante do exposto, o objetivo deste artigo é mostrar como a Diretoria de Serviço Geográfico do Exército Brasileiro (DSG) chegou a uma solução para trabalhar com Elementos Complexos de maneira interoperável, com uma interface gráfica para facilitar sua manipulação e totalmente baseada em *software* livre.

Para atingir o objetivo, este artigo é dividido em 8 seções. A primeira seção é uma introdução ao assunto, que faz um resumo do que é abordado no artigo. A segunda seção contém uma descrição detalhada de como são estruturados os Elementos Complexos. A seção 3 contém trabalhos relacionados com o presente tema. A seção 4 explica de modo geral a estruturação da solução proposta para a manipulação de objetos complexos. A seção 5 mostra detalhes da implementação do modelo conceitual da ET-EDGV desenvolvida pela DSG. A seção 6 refere-se aos detalhes de implementação da camada de aplicação desta solução. A seção 7 mostra os resultados obtidos pela DSG após a implementação do que foi proposto no artigo. A seção 8 dedica-se à conclusão.

2. ELEMENTOS COMPLEXOS

De acordo com a EDGV, Elementos Complexos podem ser definidos da seguinte forma (CONCAR, 2010):

"Elemento complexo é aquele cuja geometria poderá ser constituída por mais de uma primitiva geométrica, isto poderá ocorrer em classes de objetos onde:

1. pelo menos uma instância possua mais que uma primitiva geométrica; ou,

2. as instâncias sejam representadas pela agregação de instâncias de classes de objetos com diferentes primitivas geométricas"

Na EDGV, os Elementos Complexos são compostos por instâncias de classes de objetos com diferentes primitivas geométricas. Para exemplificar esse conceito, foi elaborado um diagrama OMT-G (CASANOVA *et al.*, 2005) do Complexo Aeroportuário como se pode ver na Figura 1. Tal complexo é composto por Construções Aeroportuárias (Edif_Constr_Aeroportuaria), por Pista de Pouso ou Heliponto (Pista_Ponto_Pouso), Faixa de Segurança (Faixa_Seguranca).

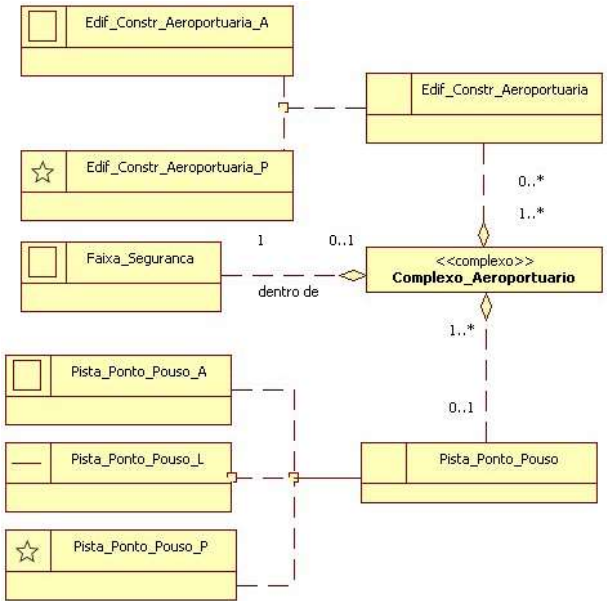

Fig. 1 - Complexo Aeroportuário.

Existe também a possibilidade de se agregar Elementos Complexos em Elementos Complexos conforme o item 1 da definição apresentada acima. Para exemplificar este conceito, na Figura 2 é mostrado o caso do complexo Instituição Pública que agrega elementos complexos do tipo Organização Pública Civil e, da mesma forma, agrega elementos complexos do tipo Organização Pública Militar.

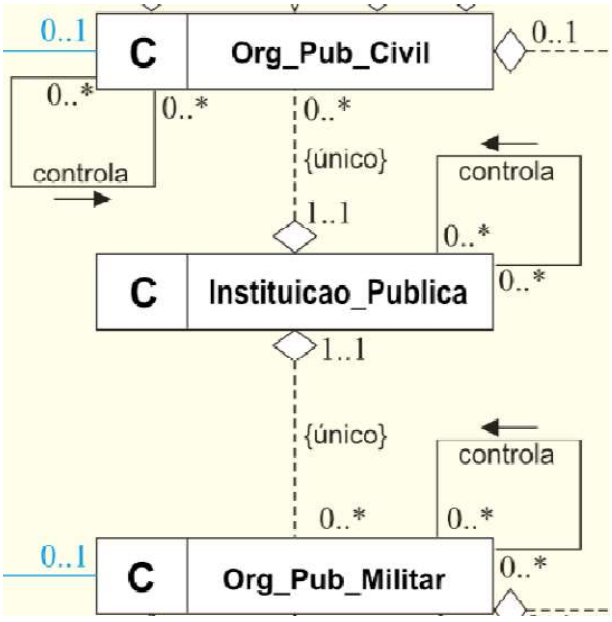

Fig. 2 - Complexo Organização Pública Civil. Fonte: ET-EDGV (CONCAR, 2010).

As classes que compõem o Complexo Aeroportuário podem ser implementadas em qualquer Banco de Dados com extensão espacial, pois são classes simples com primitivas únicas. O problema reside na representação e manipulação de Elementos Complexos, pois eles são compostos por elementos com diversas primitivas. Para contornar tal problema, a DSG desenvolveu funcionalidades para manter tais estruturas, bem como visualizá-las.

3. TRABALHOS RELACIONADOS

A ePING (BRASIL, 2011) é uma arquitetura que define padrões de interoperabilidade do governo eletrônico em nível federal. Ela também determina que sejam usados, preferencialmente, padrões abertos em suas especificações técnicas. Sendo assim, nesse sentido, no que tange a publicação de informações georreferenciadas vetoriais foram adotados como padrão arquivos *GML* versão 2.0 ou superior e arquivos *ShapeFile*.

Quanto ao formato *ShapeFile,* é importante ressaltar que somente feições do tipo ponto, linha ou área são suportados (ESRI, 1998). Sendo assim, a definição de objetos complexos, conforme definidos na ET-EDGV necessita de tabelas auxiliares (arquivos DBF) que sejam controladas no nível da aplicação, dificultando a interoperabilidade e o manuseio pelos usuários.

Quanto ao *GML*, é possível que sejam criados objetos complexos porém, por ser uma linguagem baseada em *XML* (BRAY *et al.*, 1998) necessita de um uso considerável de recursos computacionais para processamento e armazenamento (PIRAS *et al.*, 2004), que são características possivelmente indesejáveis em ambientes de produção de geoinformação, dependendo do volume de informações a produzir.

Desta forma, considerando as limitações do uso dos formatos adotados pela ePING se escolheu usar bancos de dados objeto-relacionais como o *PostgreSQL* e o *SQLite* para a implementação presente neste trabalho. O *PostgreSQL* tem uma arquitetura cliente/servidor que permite que múltiplos usuários manipulem a mesma base simultaneamente, gerenciados por um Sistema Gerenciador de Banco de Dados (SGDB), fato que favorece ambientes de produção com múltiplos usuários.

A escolha de bancos de dados espaciais, gerenciados por SGDB, para implementação da ET-EDGV já foi analisada na literatura (BORBA *et al.*, 2008), (DA PENHA *et al.*, 2012) e (GOUVEIA *et al.*, 2012).

Alguns autores (BORBA *et al.*, 2008) (GOUVEIA *et al.*, 2012) apresentam implementações em bancos de dados objeto-relacionais. Porém, a solução apresentada por (GOUVEIA *et al.*, 2012) implementa de forma simplificada e incompleta o conceito de objetos complexos. A solução de BORBA *et al.* (2008) representou uma referência de que a implementação de objetos complexos em bancos de dados relacionais é viável, corroborando a opção feita neste trabalho.

A ET-EDGV apresenta o modelo conceitual do banco de dados. Os autores Borba *et al.* (2008) e Gouveia *et al.* (2012) apresentaram estratégias de implementação diferentes, gerando bancos de dados distintos à partir do mesmo modelo conceitual. Isso ocorre pois não há um padrão para as implementações nas diversas tecnologias, fazendo com que ambas implementações sejam válidas.

O fato de o usuário de geoinformação manipular diretamente as informações armazenadas nas tabelas dos bancos de dados por meio dos aplicativos de SIG faz com que usuários que não estejam familiarizados com as decisões de implementação de cada desenvolvedor possam sentir dificuldades em compreender o mapeamento entre o modelo conceitual e as tabelas existentes no banco de dados de cada um. Sendo assim, as diferenças entre o modelo conceitual da ET-EDGV e suas correspondentes tabelas no banco de dados representam uma etapa de compreensão adicional ao usuário do produto.

4. SOLUÇÃO PROPOSTA

Tendo em vista a literatura atual e o que se encontra disponível no mercado buscou-se uma solução interoperável com ferramentas para manipulação de objetos complexos por usuários sem familiaridade com bancos de dados relacionais. Tal solução, conforme determinado na ePING (BRASIL, 2011), deve preferencialmente adotar padrões abertos.

Sendo assim, toda a solução foi pautada em software livre e de código aberto e se divide em duas partes principais:

a. implementação do modelo conceitual em bancos de dados objeto-relacionais, em *PostgreSQL* e *SQLite* conforme descrito na seção 5 deste trabalho; e

b. implementação de uma extensão, denominada DSG Tools, para o aplicativo de sistemas de informação geográficas (SIG) *QGIS,* na linguagem de programação *Python,* conforme descrito na seção 6 deste trabalho.

Tal proposta de solução tem por objetivo permitir que usuários do *QGIS* possam criar bancos de dados ET-EDGV, tanto em *PostgreSQL* quanto em *SQLite* e manipular objetos complexos armazenados neles por meio da extensão DSG Tools.

5. IMPLEMENTAÇÃO DO MODELO CONCEITUAL

A implementação do modelo conceitual definido pela ET-EDGV seguiu os seguintes requisitos:

c) O modelo lógico deve poder ser implementado em *PostgreSQL* (POSTGRESQL, 2016), com sua extensão espacial *PostGIS* (REFRACTIONS RESEARCH INC, 2005) e *SQLite* (SQLITE DEVELOPMENT TEAM, 2000) com sua extensão espacial *SpatiaLite* (FURIERI, 2008);

d) O modelo lógico deve ser compatível com os principais plataformas de SIG utilizados na DSG e pelos usuários da geoinformação produzida pela DSG;

e) O modelo físico deve ser o mais próximo possível, em termos tecnológicos, do modelo conceitual da ET-EDGV para facilitar a consulta do significado de classes e atributos, posto que este é o documento que o usuário final terá acesso (não há norma que defina o modelo lógico/físico da ET-EDGV);

f) O modelo poderá servir como um padrão nacional para distribuição de dados espaciais vetoriais em conformidade com a ET-EDGV, implementando todos os conceitos presentes na norma (Complexos, herança, domínios, etc.);

g) O modelo deve permitir a coexistência de diversas versões de ET-EDGV nos mesmos sistemas;

h) As implementações em *PostgreSQL* e *SpatiaLite* devem ser as mais próximas possíveis, ficando as diferenças de implementação restritas a especificidades técnicas.

Apesar da implementação ter seguido os requisitos supracitados, algumas decisões de projeto tiveram que ser tomadas para contornar particularidades do *PostgreSQL*. Como exemplo de especificidades da tecnologia em questão, podem-se citar:

a) Falta de controle ao se utilizar herança múltipla

conforme capítulo 5 da documentação oficial do *PostgreSQL*; (POSTGRESQL, 2016); e

b) Não acionamento de chaves estrangeiras em classes com herança conforme capítulo 5 da documentação oficial do *PostgreSQL* (POSTGRESQL, 2016).

Para contornar tais limitações, foram tomadas as seguintes decisões de projeto:

a) Não foi implementada a herança múltipla. Para casos em que isso ocorria, se usava apenas uma das heranças e as demais foram implementadas como agregações;

b) No caso dos complexos não houve uso de chaves estrangeiras. Esta decisão foi tomada devido ao problema de acionamento de chaves estrangeiras em classes herdadas. Desta forma, todo o controle dos complexos foi implementado na camada de aplicação.

Para construir mecanismos auxiliares para que a camada de aplicação pudesse controlar os elementos complexos, foram executados os passos seguintes, a saber:

a) Todos os relacionamentos de composição, definidos pela ET-EDGV, que definem como os Elementos Complexos podem ser formados foram materializados utilizando-se chaves estrangeiras;

b) A partir de uma consulta SQL se extraiu todas as possíveis agregações materializadas por meio de chaves estrangeiras e, desta consulta, criou-se uma tabela auxiliar chamada *complex_schema;*

c) Após a construção da tabela *complex_schema*, as chaves estrangeiras citadas na letra a) foram eliminadas do banco para evitar as limitações do *PostgreSQL* citadas anteriormente.

A tabela *complex_schema,* possui como colunas: o esquema da classe complexa (coluna *complex_schema*), classe de elemento complexo (coluna *complex*), esquema da classe agregada (*aggregated_schema),* classe agregada (*aggregated_class*) e coluna de ligação da classe agregada (*column_name*).

Cabe ressaltar que, visando um desenvolvimento mais ágil, foi feito o uso do *pgModeler* (ARAÚJO & SILVA, 2012) para a implementação do banco em *PostgreSQL*, e posteriormente foi feita a conversão deste banco para um banco em *SpatiaLite* (FURIERI, 2008) por meio do uso do *FME* (SAFE SOFTWARE, 1993). Desta forma, a EDGV 2.1.3 pode ser plenamente utilizada por meio desses dois SGBD.

6. IMPLEMENTAÇÃO DO CÓDIGO DA CAMADA DE APLICAÇÃO

A implementação do código foi feita após a implementação do modelo conceitual da EDGV. Com o modelo físico do banco pronto, foi necessário definir os casos de uso para a manipulação dos Elementos Complexos. O diagrama de Casos de Uso para manipulação de Elementos Complexos pode ser visto na Figura *3*.

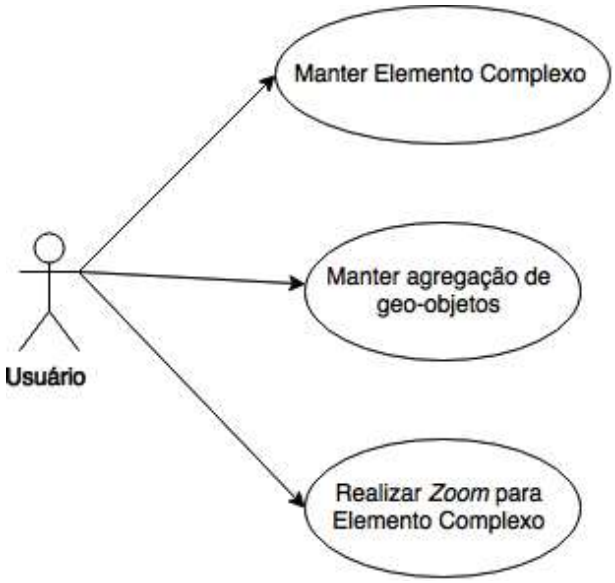

Fig. 3 - Casos de Uso para manipulação de Elementos Complexos.

O caso de uso *Manter Elemento Complexo* refere-se à criação, remoção e atualização de elementos complexos. O Caso de Uso Manter *Agregação de Geo-objetos* refere-se à criação, remoção e atualização de agregações de geo-objetos em Elementos Complexos. Por fim, o Caso de Uso *Realizar Zoom para Elemento Complexo* refere-se à realização de um zoom para a área do retângulo envolvente de todos os geo-objetos que são agregados em um dado Elemento Complexo.

Cabe ressaltar que a agregação de geo-objetos em um Elemento Complexo deve ser genérica permitindo até mesmo a agregação de Elementos Complexos em outros Elementos Complexos. Não existe um limite para a quantidade de agregações que podem existir. A hierarquia das agregações deve ser livre, desde que coerente com o previsto na EDGV.

A implementação do código referente aos casos de uso citados acima foi toda feita em *Python* com o objetivo de criar uma

extensão para o *QGIS*. O código desenvolvido para cumprir o objetivo deste artigo consiste basicamente de 10 classes: *SqlGenerator e suas derivadas, SqlGeneratorFactory, AbstractDb e suas derivadas, DbFactory, ComplexWindow e ManageComplex*. O diagrama UML simplificado dessas classes pode ser vista na Figura 4.

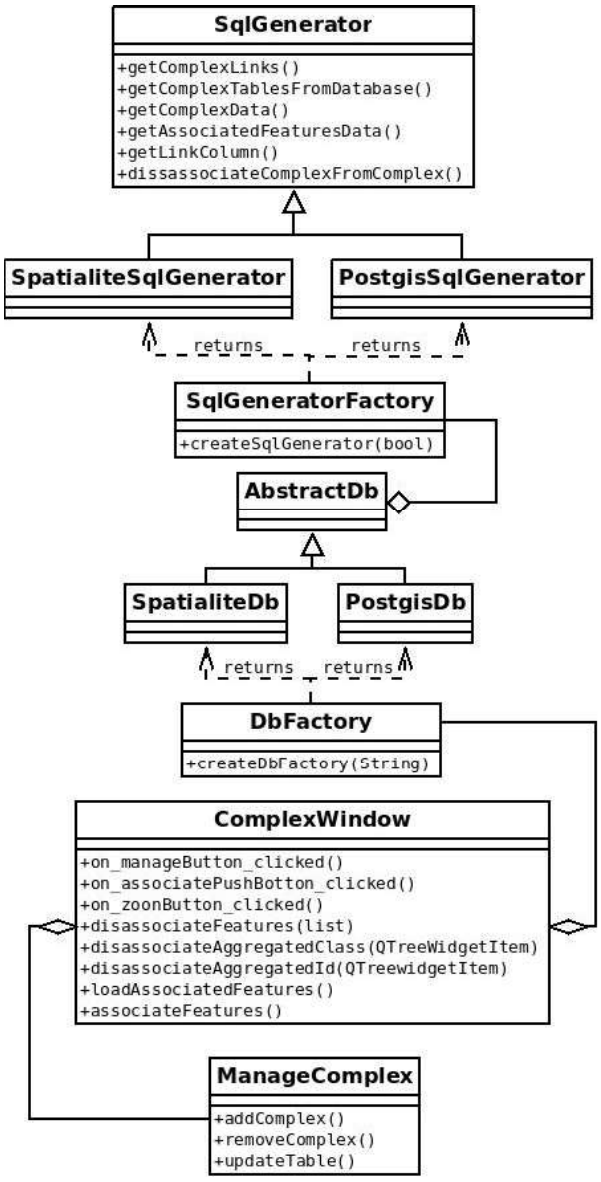

Fig. 4 - Diagrama UML de classes simplificado.

A classe *ManageComplex* é responsável pela criação de instâncias de Elementos Complexos que ainda não possuem classes agregadas, remoção e atualização de Elementos Complexos já existentes. Os métodos utilizados para tal são, respectivamente *addComplex*, *removeComplex*, *updateTable.*

A classe *SqlGenerator* é responsável por prover consultas SQL, as quais são executadas pelos métodos da classe AbstractDb, que é utilizada na manipulação dos Elementos Complexos. A seguir são mostrados detalhes de cada um dos métodos utilizados:

d) getComplexLinks: obtém todas as classes que podem ser agregadas em uma dada classe de Elemento Complexo.

e) getComplexTablesFromDatabase: obtém todas as classes de Elementos Complexos existentes na EDGV.

f) getComplexData: obtém o uuid (chave primária) e o nome do Elemento Complexo.

g) getAssociatedFeaturesData: obtém o id (chave primária) dos geo-objeto agregados a um determinado complexo. Essa obtenção é feita para cada uma das classes agregadas.

h) getLinkColumn: obtém a coluna de ligação entre um dado Elemento Complexo e uma dada classe agregada.

i) disassociateComplexFromComplex: remove de um dado Elemento Complexo um outro Elemento Complexo agregado.

A classe AbstractDb possui duas classes herdadas que especializam comportamentos específicos de banco. Os objetos especializados desta classe são criados por meio da classe DbFactory que implementa o padrão de projeto criacional *Factory* (GAMMA, 1995), sendo necessário que somente seja informado se o banco que está em trabalho é *SQLite* ou *PostgreSQL*. A classe AbstracDb é composta pela classe SqlGeneratorFactory, a qual também implementa o padrão de projeto criacional *Factory*.

Da mesma forma que a classe AbstractDb, a classe SqlGeneratorFactory é especializada em duas classes que implementam consultas de acordo com a tecnologia de banco de dados empregada.

A classe *ComplexWindow* é a responsável por utilizar as classes já citadas para executar todos os casos de uso na manipulação de Elementos Complexos. A seguir são mostrados detalhes dos métodos mais importantes utilizados:

j) on_manageButton_clicked: executa o diálogo ManageComplexDialog já citado anteriormente.

k) on_associatePushButton_clicked: executa o método *associateFeatures.*

l) on_zoomButton_clicked: calcula o retângulo envolvente de todos os objetos que são agregados por um determinado Elemento Complexo e

realiza uma aproximação para enquadramento deste elemento.

m) disassociateFeatures: desassocia dos Elementos Complexos Marcados para remoção todas as classes a eles agregadas.

n) disassociateAggregatedClass: desassocia uma determinada classe agregada de um dado Elemento Complexo.

o) loadAssociatedFeatures: obtém todos os objetos agregados a um Elemento Complexo.

p) associateFeatures: agrega à um determinado Elemento Complexo objetos selecionados no SIG.

7. RESULTADOS OBTIDOS

A implementação do código citado na seção 6 foi feita dentro do contexto de implementação da extensão para o *QGIS* chamado DSG Tools. Tal extensão contempla uma gama de ferramentas voltadas para a produção cartográfica e foi desenvolvido pela

DSG. O DSG Tools fornece, de forma transparente para o usuário, o ferramental para trabalhar com bancos de dados da EDGV 2.1.3 (última versão homologada para o público em geral) tanto em bancos de dados PostGIS quanto em *SpatiaLite*.

Na Figura 5, pode-se ver um projeto do *QGIS* com dados em torno da região do Aeroporto Internacional Juscelino Kubitschek, em Brasília-DF. Todos os dados desse projeto foram desenvolvidos de acordo com a EDGV 2.1.3.

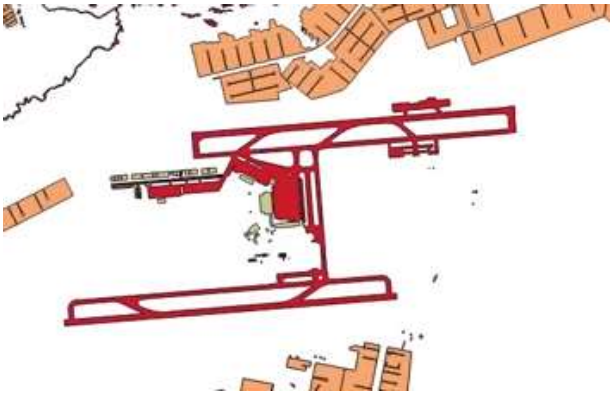

Fig. 5 - Aeroporto Juscelino Kubitschek.

Com o uso desses dados, desejou-se criar um Elemento Complexo do tipo Complexo Aeroportuário. O primeiro passo para tal é criar um Elemento Complexo do tipo Complexo Aeroportuário. Em seguida, deve-se selecionar o banco de dados em trabalho e a classe de Elemento Complexo que se deseja trabalhar, no caso a classe *tra_complexo_aeroportuario*. Isso pode ser visto na Figura 6.

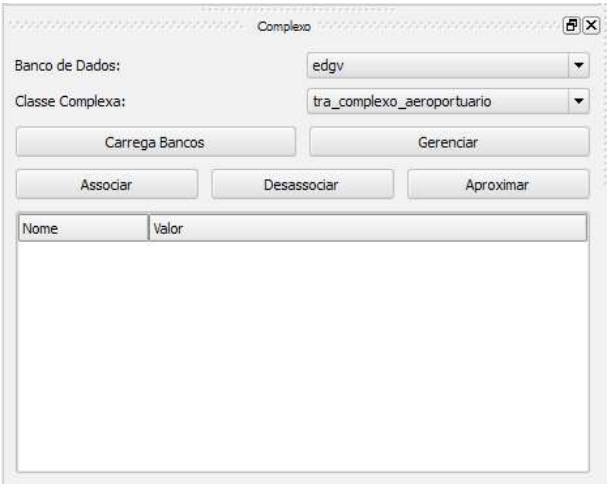

Fig. 6 - Abrindo a janela para Gerenciar Complexos.

Tendo sido feito isso, o DSG Tools irá mostrar o diálogo de gerenciamento de Elementos Complexos. Para o caso específico de complexo aeroportuário, é mostrado ao usuário o diálogo da Figura 7. Após a criação do complexo já é possível ver o item referente a ele criado na árvore de Elementos Complexos.

Fig. 7 - Criando o Complexo Aeroportuário.

De acordo com a ET-EDGV, um Complexo Aeroportuário possui agregações das classes Pista de Pouso e Construção Aeroportuária, como já foi apresentado na Figura 2. Desta forma, todos os geo-objetos referentes às classes citadas, desde que relacionadas a Complexos Aeroportuários, foram selecionados com o uso da extensão *Multiple Layer Selection* (DSG, 2014), também desenvolvido pela DSG para apoiar a produção cartográfica, que permite a seleção de

feições de diversas camadas carregadas e visíveis no *QGIS*. O resultado da seleção supracitada pode ser visualizado na Figura 8.

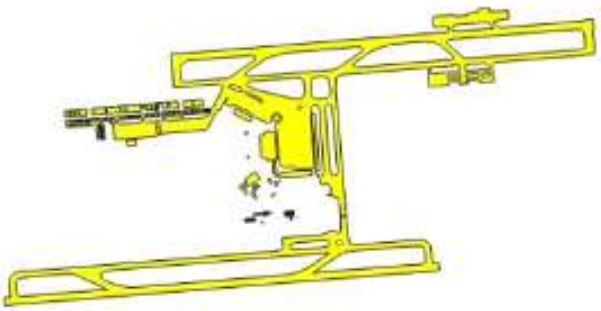

Fig. 8 - Seleção dos Geo-objetos que devem compor o Complexo.

Após a seleção dos geo-objetos que devem ser agregados, deve-se clicar em *Associar* para que a agregação seja de fato concretizada como se pode ver na Figura 9. Na árvore da Figura 9 é possível ver as classes agregadas e os identificadores (chaves primárias) dos objetos agregados de forma hierárquica.

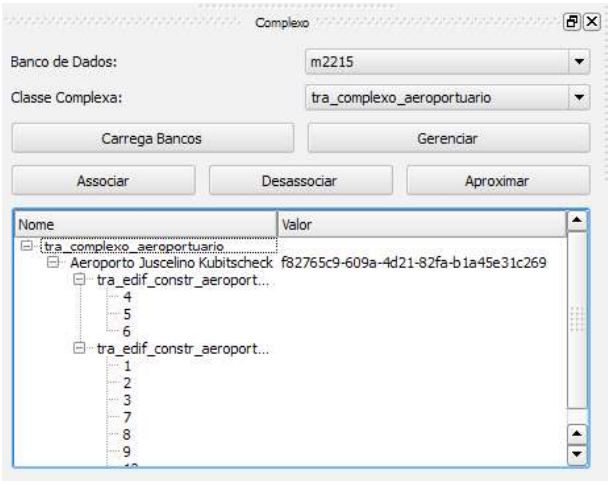

Fig. 9 - Agregação de geo-objetos realizada.

Após a criação do complexo é possível que o mesmo seja inspecionado ao se clicar em *Aproximar* (botão em destaque na Figura 10). Tal ação irá calcular o Mínimo Retângulo Envolvente do Elemento Complexo e fará uma aproximação para envolver o mesmo.

Desta forma, conclui-se a criação de agregações hierárquicas em um Elemento Complexo. Existe ainda, caso haja necessidade, a possibilidade de se atualizar um Elemento Complexo, desassociar geo-objetos agregados e, até mesmo, mudar agregações existentes. A mudança de agregações existentes se faz com a reagregação de um geo-objeto selecionado, ou seja, um geo-objeto já agregado a um Elemento Complexo a outro Elemento Complexo diferente.

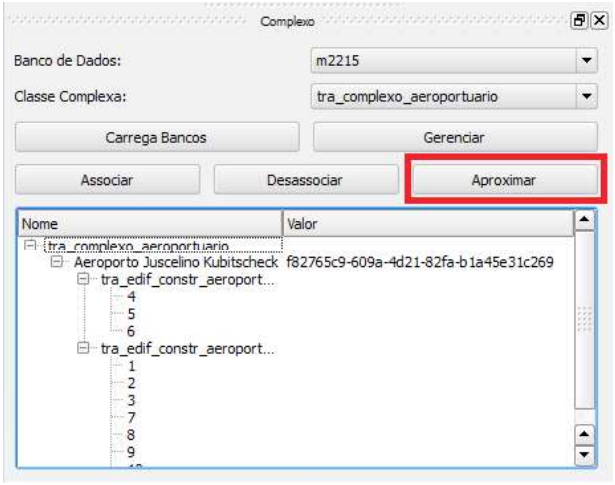

Fig. 10 - Realizando Zoom para o Elemento Complexo.

CONCLUSÃO

Neste trabalho, após análise de soluções presentes na literatura, foi proposta uma forma direta e transparente de se realizar a manipulação de objetos complexos conforme definição na ET-EDGV 2.1.3 com o objetivo de democratizar esta norma em nível nacional.

Os diferenciais deste trabalho foram a implementação de uma ferramenta que permita que não somente usuários com conhecimentos de Tecnologia da Informação (TI), mas também usuários de *software* de SIG em geral possam manipular a ET-EDGV sem necessitar de um conhecimento técnico em TI mais aprofundado e a implementação da ET-EDGV adotando estratégias que geraram um modelo lógico o mais próximo possível do modelo conceitual da ET-EDGV, única documentação oficial que os usuários podem acessar como referência.

A ferramenta desenvolvida neste artigo, juntamente com outras ferramentas voltadas para a produção cartográfica de acordo com a ET-EDGV, estão disponíveis na extensão DSG Tools para o *software QGIS*. O DSG Tools está disponível para *download*, de maneira gratuita (assim como o *QGIS*), desde 25 de março de 2015 e conta com mais de 18.000 de *downloads* (DSG, 2015).

Cabe ainda ressaltar que o que foi apresentado neste artigo foi desenvolvido totalmente em ambiente livre, usando a linguagem de programação Python. Como o código é livre e aberto, existe a oportunidade de melhorias e contribuições por parte da sociedade como um todo, visando sempre ao desenvolvimento da tecnologia nacional e ao desenvolvimento da produção cartográfica nacional.

REFERÊNCIAS BIBLIOGRÁFICAS

ARAÚJO E SILVA, R. **pgModeler**. Brasil, 2012. Disponível em http://www.pgmodeler.com.br. Acesso em 05 fev. 2016.

BRASIL. Decreto nº 6.666, de 27 de novembro de 2008. Institui, no âmbito do Poder Executivo federal, a Infraestrutura Nacional de Dados Espaciais - INDE, e dá outras providências. **Diário Oficial [da] República Federativa do Brasil, Brasília**, DF, 28 nov. 2008. Seção 1, pt. 1, p. 57, 2008.

BRASIL. e-PING: Padrões de interoperabilidade de governo eletrônico - versão 2011. Brasília: **Comitê Executivo de Governo Eletrônico**, 2011.

BORBA, ROGÉRIO L. R. & MOTA, GUILHERME L. A. & NUNES, JORGE L. N. S. Metodologia para Representação da Estrutura de Dados Geoespacial Vetorial da Mapoteca Nacional Digital em Bancos de Dados Geográficos Relacionais. **Anais do X Simpósio Brasileiro de Geoinformática (Geoinfo 2008)**, p 25-30, 2008.

BORGES, K. A. V.; DAVIS JR, C. A.; LAENDER, A. H. F. Modelagem conceitual de dados geográficos. In: CASANOVA, M. A.; CÂMARA, G.; DAVIS JR., C. A.; QUEIROZ, G. R**, Banco de dados geográficos.** Curitiba, Mundogeo, p. 93-146, 2005.

BRAY, T., PAOLI, J., SPERBERG-MCQUEEN, C. M., MALER, E., & YERGEAU, F. Extensible markup language (XML)**. World Wide Web Consortium Recommendation REC-xml-19980210**, v. 16, p. 16, 1998.

CONCAR. Comissão Nacional de Cartografia. **Especificação Técnica Para A Estruturação De Dados Geoespaciais Vetoriais (ET-EDGV)**. Versão 2.1.3, 2010.

DA PENHA, A. L. T.; MORITA, C. Y.; DE CERQUEIRA, R. W. Geração de base cartográfica digital a partir de produtos fotogramétricos para a geração de ortofotocarta, carta topográfica e banco de dados geográficos–o caso do projeto de mapeamento do Estado da Bahia. **Simpósio Brasileiro de Ciências Geodésicas e Tecnologias da Geoinformação**, v. 4, p. 1-9, 2012.

DSG. **Multiple Layer Selection**. Brasília, Brasil, 2014. Disponível em http://plugins.qgis.org/plugins/ MultipleLayerSelection/. Acesso em 05 fev. 2016.

ESRI. **ESRI Shapefile Technical Description, An ESRI White Paper**, 1998.

FURIERI, A. **SpatiaLite**, 2008. Itália. Disponível em http://www.gaia-gis.it/gaia-sins/. Acesso em 05 fev. 2016.

GAMMA, E, VLISSIDES, J., HELM, R. & JOHNSON, R. **Design patterns: Elements of reusable object-oriented software**. Reading: Addison-Wesley, v. 49, n. 120, p. 107-116, 1995.

GOUVEIA, A. L.; NAGATOMI, R. C. M. E.; DA SILVA, R. L. Migração Da Base Cartográfica Do Brasil, Ao Milionésimo, Para Os Padrões Da Inde. **IV Simpósio Brasileiro de Ciências Geodésicas e Tecnologias da Geoinformação,** p 1-6, 2012.

OPEN GEOSPATIAL CONSORTIUM. OpenGIS® Geography Markup Language (GML) Encoding Standard. Ref Number: **OGC 07-036 v3.2.1**, 2007.

PIRAS, A., DEMONTIS, R., DE VITA, E., & SANNA, S. Compact GML: merging mobile computing and mobile cartography. **Proceedings Of The 3rd GML And Geo-Spatial Web Services Conference**, 2004.

POSTGRESQL. **PostgreSQL**. Califórnia, Estados Unidos da América. Disponível em http://www.postgresql.org. Acesso em 05 fev. 2016.

POSTGRESQL. **PostgreSQL 9.3 Documentation**. Disponível em http://www.postgresql.org/docs/9.3/static/ddl-inherit.html. Acesso em 05 fev. 2016.

REFRACTIONS RESEARCH INC. **PostGIS**. Canada, 2005. Disponível em http://postgis.net. Acesso em 05 fev. 2016.

SAFE *SOFTWARE*. **FME**. Vancouver, Canadá, 1993. Disponível em http://www.safe.com. Acesso em 05 fev. 2016.

SQLITE DEVELOPMENT TEAM. **SQLite**, 2000. Disponível em https://www.sqlite.org. Acesso em 05 fev. 2016.