UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

GERAÇÃO DE DADOS ESPACIAIS VAGOS BASEADA EM MODELOS EXATOS

FERNANDO ROBERTO PROENÇA

ORIENTADOR: PROF. DR. RICARDO RODRIGUES CIFERRI

São Carlos - SP
Maio/2013

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

GERAÇÃO DE DADOS ESPACIAIS VAGOS BASEADA EM MODELOS EXATOS

FERNANDO ROBERTO PROENÇA

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Engenharia de Software, Banco de Dados e Interação Humano Computador.

Orientador: Prof. Dr. Ricardo Rodrigues Ciferri.

São Carlos - SP
Maio/2013

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

P964gd

Proença, Fernando Roberto.
Geração de dados espaciais vagos baseada em modelos exatos / Fernando Roberto Proença. -- São Carlos : UFSCar, 2013.
111 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2013.

1. Banco de dados. 2. Sistema de informação. 3. Sistemas de informação geográfica. 4. Java (Linguagem de programação de computador). I. Título.

CDD: 005.74 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Geração de Dados Espaciais Vagos Baseada em Modelos Exatos"

Fernando Roberto Proença

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação

Membros da Banca:

Ricardo Rodrigues Ciferri

Prof. Dr. Ricardo Rodrigues Ciferri
(Orientador - DC/UFSCar)

Prof. Dr. Renato Bueno
(DC/UFSCar)

Prof. Dr. Geovane Cayres Magalhães
(UNICAMP)

São Carlos
Maio/2013

Dedico este trabalho aos meus pais Miguel Lídio Proença e Maria Iolanda de Queiroz Proença, que são meus maiores exemplos de superação e fé. Ao meu irmão Mailson de Queiroz Proença, minha namorada Régila da Silva Fidélis e a todos parentes e amigos que sempre estiveram ao meu lado, acreditando na minha capacidade para a realização deste projeto.

Em memória de meu pai Miguel Lídio Proença.

Fernando Roberto Proença

AGRADECIMENTO

Agradeço primeiramente a Deus, pela capacitação concedida, sem a qual não realizaria este projeto de Mestrado. Agradeço também ao meu orientador, professor doutor Ricardo Rodrigues Ciferri pela ideia de tema, pela parceria, colaboração e orientação concedida durante todo o processo de elaboração deste trabalho. À professora doutora Cristina Dutra de Aguiar Ciferri da USP/ICMC e aos professores do Grupo de Banco de Dados do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos pelos conselhos e comentários valiosos que contribuíram para a realização deste trabalho. Aos amigos do Departamento de Computação da UFSCar que sempre estiveram dispostos a esclarecer minhas dúvidas técnicas para o desenvolvimento do projeto. Agradeço também pelo apoio financeiro recebido pela CAPES e CNPq. Enfim, a todas as pessoas que contribuíram direta e indiretamente para realização deste projeto.

RESUMO

Sistemas de informação geográfica com o auxílio de bancos de dados espaciais armazenam e gerenciam dados espaciais exatos, cujas formas (fronteiras) são bem definidas e que possuem uma localização exata no espaço. Entretanto, vários dados espaciais reais não possuem os seus limites precisamente conhecidos ou possuem uma localização incerta no espaço, os quais são denominados dados espaciais vagos. Os limites de um dado espacial vago podem encolher ou estender, portanto, podem ter uma extensão mínima e máxima. Nuvens de poluição, desmatamentos, focos de incêndios, rota de um avião, habitats de plantas e de animais são exemplos de dados espaciais vagos. Na literatura, atualmente existem modelos de dados espaciais vagos, tais como *Egg-Yolk*, QMM e VASA. No entanto, segundo o nosso conhecimento, estes enfocam apenas no aspecto formal da definição do modelo. Com isso, dados espaciais vagos reais ou sintéticos não estão disponíveis para uso. O principal objetivo deste trabalho de mestrado consiste no desenvolvimento de algoritmos para a geração de dados espaciais vagos sintéticos baseados nos modelos exatos de dados espaciais vagos *Egg-Yolk*, QMM e VASA. Também foi implementada uma ferramenta, chamada *VagueDataGeneration*, para auxiliar no processo de geração desses dados. Nos algoritmos propostos e na ferramenta desenvolvida, o usuário define as propriedades referentes ao tipo de dado de um modelo, tais como tamanho, formato, volume, complexidade, localização e distribuição espacial dos dados espaciais vagos a serem gerados. Por meio do uso dos algoritmos propostos e da ferramenta *VagueDataGeneration*, os pesquisadores podem gerar grandes amostras de dados espaciais vagos, possibilitando novas pesquisas, como exemplo, testar índices para dados espaciais vagos ou testar técnicas de processamento de consultas em *Data Warehouses* que armazenam dados espaciais vagos. A validação da geração de dados espaciais vagos foi efetuada usando um estudo de caso com dados de fenômenos rurais vagos.

Palavras-chave: Dados Espaciais Vagos, Modelos Exatos de Dados Espaciais Vagos, Geração de Dados Espaciais Vagos, Dados Sintéticos, *Egg-Yolk*, QMM, VASA.

ABSTRACT

Geographic information systems with the aid of spatial databases store and manage crisp spatial data (or exact spatial data), whose shapes (boundaries) are well defined and have a precise location in space. However, several spatial data do not have precisely known boundaries or have an uncertain location in space, which are called vague spatial data. The boundaries of a given vague spatial data may shrink or extend, therefore, may have a minimum and maximum extension. Clouds of pollution, deforestation, fire outbreaks, route of an airplane, habitats of plants and animals are examples of vague spatial data. In the literature, there are currently vague spatial data models, such as Egg-Yolk, QMM and VASA. However, according to our knowledge, they focus only on the formal aspect of the model definition. Thus, real or synthetic vague spatial data is not available for use. The main objective of this master thesis is the development of algorithms for the generation of synthetic vague spatial data based on the crisp models of spatial data vague Egg-Yolk, QMM and VASA. It was also implemented a tool, called *VagueDataGeneration*, to assist in the process of generation such data. For both the algorithms and the tool, the user is able to set the properties related to the data type of model, such as size, shape, volume, complexity, location and spatial distribution. By using the proposed algorithms and the *VagueDataGeneration* tool, researchers can generate large samples of vague spatial data, enabling new research, such as testing indexes for vague spatial data or evaluating query processing over data warehouses that store vague spatial data. The validation of the vague spatial data generation was conducted using a case study with data from vague rural phenomena.

Keywords: Vague Spatial Data, Vague Spatial Data Crisp Models, Vague Spatial Objects, Vague Spatial Data Generation, Synthetic Data, Egg-Yolk, QMM, VASA.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BD – Banco de Dados

BDE – Banco de Dados Espacial

DC – Desconectado

DR – Regiões distintas

DW – *Data Warehouse*

DWE – *Data Warehouse* Espacial

DWEV – *Data Warehouse* Espacial Vago

DWG – *Data Warehouse* Geográfico

EC – Externamente Conectado

EQ – Igual

IBGE – Instituto Brasileiro de Geografia e Estatística

IDE – *Integrated Development Environment*

INEP – Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira

MBR – Retângulo Envolvente Mínimo

NTPP – Parte própria não-tangencial

NTPPI – Parte própria não-tangencial inversa

PL/pgSQL – *Procedural Language/PostgreSQL Structured Query Language*

PO – Se sobrepõem parcialmente

PP – Parte própria

PPI – Parte própria inversa

QMM – Qualitativo Mínimo-Máximo

RCC – Cálculo de Conexão de Região

SGBD – Sistema de Gerenciamento de Banco de Dados

SIG – Sistema de Informação Geográfica

TPP – Parte própria tangencial

TPPI – Parte própria tangencial inversa

VASA – Álgebra Espacial Vaga

SUMÁRIO

Capítulo 1

INTRODUÇÃO

*Este capítulo apresenta o **contexto** em que este trabalho está inserido e a **motivação** que deu origem a este projeto de pesquisa em nível de mestrado. Em seguida são discutidos os **objetivos** da pesquisa, finalizando com a descrição da **organização** desta monografia*.

1.1 Contexto do Trabalho

Os bancos de dados espaciais (BDE) são utilizados para armazenar e gerenciar grandes volumes de dados espaciais, tanto dados espaciais vetoriais, formados por geometrias, quanto dados espaciais matriciais, compostos por imagens no formato *raster*. Os dados espaciais vetoriais, foco deste trabalho, são formados por formas geométricas, os quais podem ser de tipos simples (ex. ponto, linha e polígono) ou de tipos complexos (ex. múltiplos pontos, múltiplas linhas e múltiplos polígonos). Geralmente os dados espaciais são armazenados em um BDE considerando as suas coordenadas bem definidas, ou seja, os objetos espaciais possuem uma localização exata no espaço e uma extensão bem definida. Por exemplo, o traçado de uma rodovia localizada em um estado é representado por uma linha cujos pontos que a compõem são todos conhecidos.

No entanto, muitos objetos do mundo real localizados no espaço não possuem uma localização exata ou os seus limites não são precisamente conhecidos. Florestas, nuvens de poluição, habitats de plantas e de animais são alguns exemplos. Tais objetos são conhecidos como **dados espaciais vagos** (os termos objetos e dados serão usados como sinônimos ao longo do texto).

Um objeto espacial vago pode representar a incerteza sobre os caminhos ou a extensão espacial dos fenômenos no espaço. Esses objetos podem encolher ou estender e, portanto, ter uma extensão mínima e máxima. Como exemplo, pode-se citar um lago cujo nível de água depende da quantidade de chuvas. No período da seca, o lago tem um tamanho menor devido ao volume reduzido de água, enquanto que em outras épocas do ano, o lago pode aumentar de tamanho devido ao maior volume de água. Outros exemplos do mundo real nos quais os dados espaciais vagos podem ser aplicados são:

- Recursos naturais (ex. minério de ferro): para algumas áreas, sabe-se certamente da existência de minério de ferro por causa de amostras de solo e poços artesianos. Para outras áreas, não se tem certeza da incidência desse mineral. Esses objetos são exemplos de dados espaciais vagos, e nesse caso, são representados por regiões (polígonos) vagas;
- Áreas e pontos de pragas e parasitas em plantações: para algumas áreas, tem-se certeza que a área está infectada por pragas e parasitas. Já para outras áreas, não se tem certeza se elas estão infectadas ou não. Esses dados espaciais vagos são representados por regiões vagas ou por pontos vagos; e
- Trilhas (caminhos) de animais: Algumas trilhas de animais são conhecidas, já outras trilhas são apenas uma suposição. Nesse exemplo, as trilhas são representadas por linhas vagas.

Existem alguns modelos de dados espaciais vagos, como por exemplo, modelos difusos (ou nebulosos) e exatos. A seguir, esses modelos são detalhados.

- Modelos difusos (*fuzzy*)*:* os modelos difusos (ZADEH, 1965; DUBOIS; PRADE, 1993; DILO, 2006; DILO; BY; STEIN, 2007) permitem uma modelagem mais refinada dos objetos espaciais vagos, considerando diferentes graus de pertinência. Os graus de pertinência estão no intervalo de **0** e **1**, onde considerando um determinado ponto de um objeto espacial vago, o nível **0** significa que esse ponto definitivamente não pertence ao objeto, enquanto o nível **1** significa que esse ponto certamente pertence ao objeto. Esse modelo requer um grande conhecimento sobre os graus de pertinências, uma vez que nem sempre as informações sobre os graus de pertinências estão disponíveis, além

de, computacionalmente, esse modelo ser muito caro, devido suas estruturas de dados e algoritmos;

- Modelos exatos: os modelos exatos (LEHMANN E COHN, 1994; COHN; GOTTS, 1996; BEJAOUI et al., 2009, 2010; PAULY; SCHNEIDER, 2010) estendem os modelos de dados espaciais exatos, ou seja, os tipos de dados, operações e predicados topológicos são baseados em seus correspondentes exatos. A vantagem da abordagem deste modelo é que as definições existentes, técnicas, estruturas de dados e algoritmos não precisam ser reconstruídos, mas apenas modificados, ampliados, ou simplesmente usados. Assim como os modelos grosseiros, as aproximações mínima e máxima de uma região são consideradas. No entanto, alguns modelos exatos consideram a região exata disjunta em relação à região vaga, em contraste com os modelos grosseiros.

Este trabalho de pesquisa aborda apenas os modelos de dados espaciais vagos baseados nos modelos exatos, apesar da existência e do modelo difuso (*fuzzy*).

1.2 Motivação

Atualmente existem na literatura propostas de modelos de dados espaciais vagos, como *Egg-Yolk* (LEHMANN E COHN, 1994), QMM (BEJAOUI et al., 2009) e VASA (PAULY E SCHNEIDER, 2010). Esses modelos definem os tipos de dados, operações, predicados topológicos e relacionamentos existentes entre os dados espaciais vagos. No entanto, segundo o nosso conhecimento, não há amostras de dados espaciais vagos disponíveis para os pesquisadores usarem em suas pesquisas, por exemplo, para testar um índice para dados espaciais vagos ou para testar técnicas de processamento de consultas em *data warehouses* (DW) que armazenam dados espaciais vagos. Ademais, não foi encontrado até o momento nenhuma proposta de implementação para a geração de dados espaciais vagos sintéticos de maneira controlada, ou seja, gerar os dados espaciais vagos controlando suas propriedades e características, de acordo com a necessidade do pesquisador.

Em particular, este trabalho de mestrado visa apoiar diretamente dois outros trabalhos que estão sendo desenvolvidos no Grupo de Banco de Dados da UFSCar e que necessitam de dados espaciais vagos. Ambos os trabalhos são orientados pelo Prof. Ricardo Rodrigues Ciferri. O primeiro trabalho, em nível de mestrado e sendo desenvolvido pelo aluno Anderson Chaves Carniel, visa implementar um tipo abstrato de dados para a VASA usando o sistema gerenciador de banco de dados PostgreSQL/PostGIS. O segundo trabalho, em nível de doutorado e sendo desenvolvido pelo aluno Thiago Luís Lopes Siqueira, possui como alguns de seus objetivos a proposta de esquemas lógicos de *data warehouses* com dados espaciais vagos e a proposta de um índice para *data warehouses* com dados espaciais vagos. Ambos esses trabalhos de mestrado e doutorado usarão os dados espaciais vagos sintéticos gerados nesta pesquisa de mestrado para validar a eficiência de suas propostas. Portanto, é de interesse e necessidade do Grupo de Banco de Dados da UFSCar a proposta de um gerador de dados espaciais vagos para os modelos *Egg-Yolk*, QMM e VASA.

Tais argumentos relatam a necessidade de pesquisas que auxiliem a geração de dados espaciais vagos, baseando-se nos modelos existentes na literatura. No presente trabalho são propostos algoritmos para a geração de conjuntos de dados espaciais vagos bastante volumosos, os quais podem ser armazenados em um banco de dados espacial ou em um *data warehouse* espacial (DWE) para o processamento de consultas sobre esses dados. Também é proposta uma ferramenta para apoiar a geração dos dados espaciais vagos.

1.3 Objetivo

O principal objetivo deste trabalho de pesquisa em nível de mestrado consiste no desenvolvimento de algoritmos para a geração de dados espaciais vagos sintéticos a partir dos modelos de dados espaciais vagos que são baseados nos modelos de dados espaciais exatos. Mais precisamente, neste trabalho foram desenvolvidos algoritmos para a geração de dados espaciais vagos sintéticos a partir dos modelos *Egg-Yolk*, QMM e VASA. Outro objetivo deste trabalho consiste

na construção da ferramenta *VagueDataGeneration*, a qual realiza a geração dos dados espaciais vagos sintéticos a partir dos algoritmos desenvolvidos.

A partir dos objetivos propostos, os algoritmos de geração de dados espaciais vagos sintéticos foram desenvolvidos, considerando as definições e as características de cada tipo de dado de cada modelo considerado. Foram exploradas diversas características dos dados, tais como tamanho, formato e complexidade (ou seja, número de pontos que compõem a geometria). Por fim, foi desenvolvida a ferramenta *VagueDataGeneration*, por meio da qual o usuário define os parâmetros dos algoritmos para a geração dos dados espaciais vagos sintéticos.

O manual de instalação e uso da ferramenta está disponível também no *Apêndice A* desta monografia. A ferramenta *VagueDataGeneration* e o manual de instalação e uso também podem ser obtido através do endereço eletrônico http://gbd.dc.ufscar.br/vaguedatageneration.

1.4 Hipóteses

Para o desenvolvimento deste trabalho de pesquisa foram propostas três **hipóteses**, apresentadas a seguir:

1. É possível gerar dados espaciais vagos sintéticos considerando e controlando na geração dos dados características tais como formato, tamanho, volume, complexidade, localização e distribuição espacial dos dados espaciais vagos;
2. É possível gerar dados espaciais vagos sintéticos a partir das definições dos modelos de dados espaciais vagos *Egg-Yolk*, QMM e VASA;
3. É possível posicionar os dados espaciais vagos sintéticos gerados em uma determinada localização do espaço.

As respectivas respostas e justificativas das hipóteses descritas são apresentadas na seção 5.2.

1.5 Organização da Monografia

Além desta introdução, esta monografia está organizada da seguinte forma:

- O Capítulo 2 - descreve os fundamentos teóricos necessários para a compreensão deste trabalho;
- O Capítulo 3 - sumariza os modelos de dados espaciais vagos *Egg-Yolk*, QMM e VASA;
- O Capítulo 4 - descreve o trabalho desenvolvido. Mais precisamente, esse capítulo apresenta os algoritmos propostos para a geração dos dados espaciais vagos e descreve a ferramenta *VagueDataGeneration*, desenvolvida para auxiliar a geração de dados espaciais vagos;
- O Capítulo 5 - descreve um estudo de caso voltado à validação dos algoritmos propostos; e
- O Capítulo 6 - apresenta as conclusões do trabalho, juntamente com as contribuições e os trabalhos futuros.

Capítulo 2

FUNDAMENTAÇÃO TEÓRICA

*Este capítulo descreve os **principais conceitos** relacionados à proposta da dissertação de mestrado. Primeiramente, são definidos os conceitos relacionados aos **dados espaciais** e seus **tipos de dados espaciais**, essenciais para se entender os modelos dos dados espaciais vagos. Em seguida, são descritos os conceitos referentes ao **cálculo de conexão de região (RCC)**, uma vez que o modelo de dado espacial vago Egg-Yolk baseia-se no RCC, e os conceitos referentes ao modelo de **9-Interseção**, os quais são essenciais para o entendimento do modelo de dado espacial vago QMM (Qualitativo Mínimo-Máximo) e da Álgebra Espacial Vaga (VASA). Por fim, são resumidos alguns **geradores de dados espaciais exatos** existentes na literatura.*

2.1 Estruturação do Capítulo

Este capítulo está estruturado da seguinte forma. Na seção 2.2 são descritos conceitos básicos de dados espaciais vagos. Nas seções 2.3 e 2.4 são detalhados os conceitos de cálculo de conexão de região e do modelo de 9-interseção, respectivamente. Na seção 2.5 são resumidos geradores de dados espaciais exatos. O capítulo é finalizado na seção 2.6 com as considerações finais.

2.2 Dados Espaciais

Os dados espaciais podem ser usados em diversas situações para representar a localização georreferenciada de um objeto na superfície terrestre (DAVIS *et al*., 2005). Por exemplo, um *outdoor* de uma loja pode exibir os locais de todas as lojas como pontos em um mapa; um motorista pode verificar a distância entre dois locais e planejar uma rota; e um gerente de vendas pode definir regiões de vendas, e usá-las para relacionar clientes a representantes de vendas e realizar análises do desempenho das vendas. Esses exemplos representam apenas algumas das possibilidades criadas pela integração de dados espaciais aos sistemas de gerenciamento de banco de dados e às aplicações de *software*.

Os tipos de dados espaciais podem ser representados por meio de diferentes objetos geométricos. Ciferri (2002) e Câmara *et al*. (1996) classificam os tipos de dados espaciais em ponto, linha, polígono no espaço Euclidiano bidimensional.

Um ponto é formado por um par de coordenadas espaciais (x, y), sendo a menor unidade possível para representar um objeto espacial, não possuindo extensão nem dimensão (zero-dimensional). Em geral, pontos são usados para representar localizações discretas, como exemplo, uma parada de ônibus em um mapa de uma cidade.

Uma linha é uma sequência de pontos conectados. Uma linha poligonal é uma linha onde os pontos não estão dispostos de forma retilínea, representando formas unidimensionais. Em ambas, cada par de pontos conectados representa um segmento de linha. Linhas são utilizadas para representar objetos espaciais lineares como rodovias, ferrovias e rios em um mapa de uma região. Uma linha obrigatoriamente deve conter pelo menos dois pontos diferentes conectados entre si.

Por fim, um polígono é uma região formada por duas ou mais sequências de linhas fechadas ou linhas poligonais conectadas, ou seja, com pelo menos três pontos distintos, de tal forma que o último ponto seja igual ao primeiro ponto da linha. Polígonos representam objetos espaciais bidimensionais, como exemplo, a área de um estado em um mapa de um determinado país. Os polígonos podem ser complexos, possuindo buracos ou partes disjuntas chamadas de ilhas.

Os tipos dados espaciais são divididos em simples e complexos, dependendo da complexidade espacial que eles são capazes de modelar. Os tipos de dados

espaciais simples são compostos por apenas um único objeto espacial. Um ponto simples, uma linha contínua, uma região (polígono) simples são exemplos de tipos de dados espaciais simples. Já os tipos de dados espaciais complexos são formados por um objeto espacial com vários componentes em uma única instância. Múltiplos pontos, múltiplas linhas, polígono com buraco, polígono com ilha são exemplos de dados espaciais complexos. A Figura 2.1 ilustra diferentes tipos de dados espaciais.

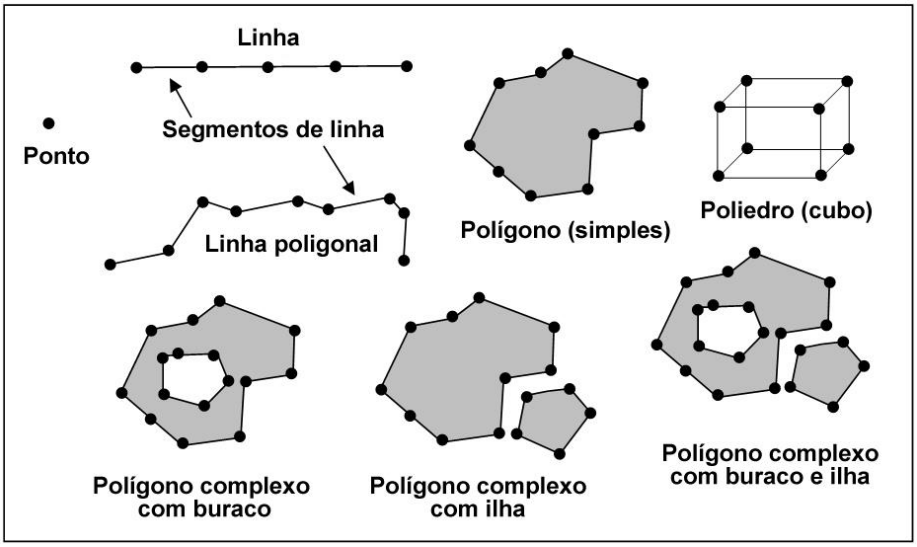

Figura 2.1 Tipos de Dados Espaciais. Reproduzido de (Ciferri, 2002).

2.3 Cálculo de Conexão de Região (RCC)

A teoria do cálculo de conexão de região (RCC) descreve formalmente os relacionamentos entre duas regiões exatas distintas em um determinado espaço. Formalmente, a teoria da RCC é definida por meio do relacionamento *C*(*X*, *Y*), onde "*X*" e "*Y*" representam as regiões distintas e "*C*" representa a relação entre as duas regiões. A relação *C*(*X*, *Y*) é lida como "*X*" se conecta com "*Y*" (se *X* e *Y* forem regiões) (RANDELL; COHN, 1989; RANDELL *et al*., 1992).

Cohn *et al*. (1997) apresentam um conjunto de oito relacionamentos bases entre duas regiões, denominado RCC-8. O RCC-8 relaciona apenas duas regiões distintas no mesmo espaço. Os oito relacionamentos são: DC (Desconectado), EC (Externamente Conectado), PO (se sobrepõem parcialmente), TPP (parte própria tangencial), NTPP (parte própria não-tangencial), EQ (igual), TPPI (parte própria tangencial inversa), NTPPI (parte própria não-tangencial inversa). Esses

relacionamentos são descritos a seguir, e exemplificados na Figura 2.2, considerando duas regiões "*X*" e "*Y*":

- Desconectado (DC): ocorre quando a região "*X*" é disjunta (desconectada) da região "Y". Isto é, as regiões "*X*" e "*Y*" não possuem nenhum ponto em comum;
- Igual (EQ): ocorre quando a região "*X*" é igual à região "*Y*", ou seja, ambas as regiões possuem todos os pontos em comum, tanto no interior quanto nas bordas (ou limites) das regiões;
- Externamente Conectado (EC): ocorre quando a borda da região "*X*" toca a borda da região "*Y*". Isto é, as regiões "*X*" e "*Y*" possuem apenas alguns pontos de suas bordas em comum;
- Se sobrepõem parcialmente (PO): ocorre quando parte da região "*X*" se sobrepõe a parte da região "*Y*", ou seja, parte das regiões "*X*" e "*Y*" possuem pontos em comum tanto em seus interiores quanto em suas bordas;
- Parte própria tangencial (TPP): ocorre quando a região "*X*" está contida (dentro) na região "*Y*" e a borda da região "*X*" toca a borda da região "*Y*" em determinados pontos;
- Parte própria não-tangencial (NTPP): ocorre quando a região "*X*" está contida na região "*Y*" e a borda da região "*X*" não toca a borda da região "*Y*" em nenhum ponto;
- Parte própria tangencial inversa (TPPI): ocorre quando a região "*X*" contém a região "*Y*" (isto é, a região "*Y*" está dentro da região "*X*") e a borda da região "*X*" toca a borda da região "*Y*" em determinados pontos;
- Parte própria não-tangencial inversa (NTPPI): ocorre quando a região "*X*" contém a região "*Y*" e a borda da região "*X*" não toca a borda da região "*Y*" em nenhum ponto.

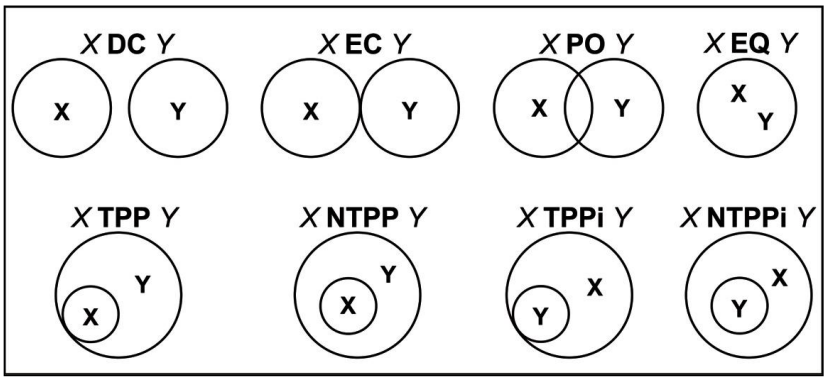

Figura 2.2 RCC-8 Bases da teoria do cálculo de conexão de regiões. Adaptada de (Cohn *et al*., 1997)

A partir dos relacionamentos da RCC-8 foi desenvolvido outro conjunto de cinco novos relacionamentos, denominado RCC-5 (COHN *et al*., 1997). O RCC-5 é formado pelos relacionamentos PO (se sobrepõem parcialmente), PP (parte própria), EQ (igual), PPI (parte própria inversa) e DR (regiões distintas). Esses relacionamentos são derivados de combinações entre alguns relacionamentos do RCC-8. Esses relacionamentos são descritos a seguir, e ilustrados na Figura 2.3:

- Parte própria (PP): obtida por meio dos relacionamentos TPP e NTPP do RCC-8. Esse relacionamento ocorre quando a região "*X*" está contida na região "*Y*", desconsiderando se a borda da região "*X*" toca ou não a borda da região "*Y*" em determinados pontos.
- Parte própria inversa (PPI): obtida por meio dos relacionamentos TPPI e NTPPI do RCC-8. Esse relacionamento ocorre quando a região "*X*" contém a região "*Y*", desconsiderando se a borda da região "*X*" toca ou não a borda da região "*Y*" em determinados pontos.
- Regiões distintas" (DR): obtida por meio dos relacionamentos EC e DC do RCC-8. Esse relacionamento ocorre quando a região "*X*" é disjunta (desconectada) da região "*Y*", desconsiderando se a borda da região "*X*" toca ou não a borda da região "*Y*".
- Se sobrepõem parcialmente (PO): mesmo relacionamento PO do RCC-8;
- Igual (EQ): mesmo relacionamento EQ do RCC-8.

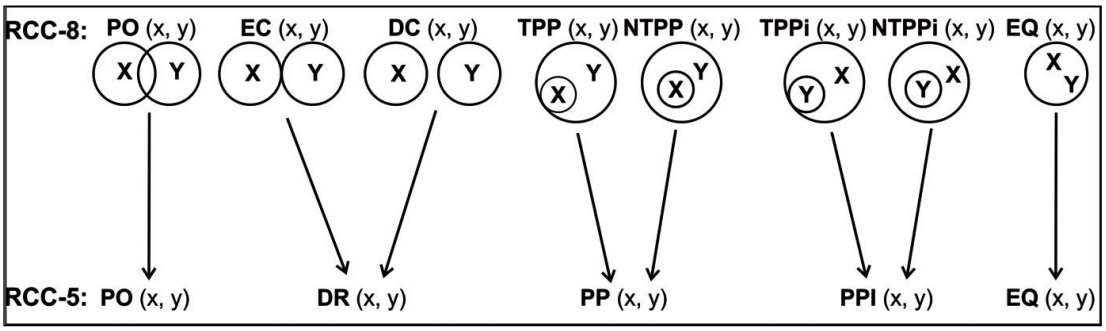

Figura 2.3 Relacionamentos topológicos do RCC-8 e RCC-5 entre duas regiões. Adaptada de (Cohn e Gotts, 1996).

As teorias do RCC-8 e do RCC-5 consideram apenas relacionamentos entre objetos espaciais exatos, ou seja, objetos que possuem seus limites precisamente conhecidos e localização exata no espaço. Entretanto, elas foram descritas neste capítulo porque são usadas como base para a definição de relacionamentos entre duas regiões no modelo de dados espaciais vagos *Egg-Yolk*, foco de interesse desta dissertação.

2.4 Modelo de 9-Interseção

Egenhofer e Herring (1991) definem um conjunto de nove interseções entre dois objetos espaciais, denominado 9-Interseção. O modelo de 9-interseção é um modelo que formaliza as relações topológicas binárias, ou seja, relações entre dois objetos espaciais. Para um relacionamento entre dois objetos espaciais segundo o modelo de 9-interseção são considerados três partes dos objetos espaciais: (*i*) Interior, (*ii*) Exterior e (*iii*) Limite. O **interior** (***º***) representa a parte (região) que realmente pertence ao objeto espacial, enquanto que o **exterior** (‾) consiste na parte que certamente não pertence ao objeto. Por fim, o **limite** (***∂***) (ou fronteira) representa a divisão entre o interior e o exterior do objeto espacial. A fronteira também pertence ao objeto espacial.

Os relacionamentos entre dois objetos espaciais são definidos a partir de comparações de seus interiores, limites e exteriores (EGENHOFER; HERRING, 1991). Considerando dois objetos espaciais "*A*" e "*B*", respectivamente, têm-se as comparações entre: o interior de "*A*" (***Aº***); o limite de "*A*" (***∂A***) e o exterior de "*A*" (***A***‾)

com o interior de "*B*" (***B°***); e o limite de "*B*" (***∂B***) e o exterior de "*B*" (***B***⁻). O relacionamento entre as três partes dos dois objetos espaciais resultam em nove comparações, chamadas de 9-Interseção, as quais são representadas a partir de uma matriz 3 x 3. A matriz a seguir explicita os nove relacionamentos, considerando o relacionamento "*R*" entre duas regiões "*A*" e "*B*", respectivamente:

$$R(A, B) = \begin{pmatrix} A^{o} \cap B^{o} & A^{o} \cap \partial B & A^{o} \cap B^{-} \\ \partial A \cap B^{o} & \partial A \cap \partial B & \partial A \cap B^{-} \\ A^{-} \cap B^{o} & A^{-} \cap \partial B & A^{-} \cap B^{-} \end{pmatrix}$$

Utilizando como base a matriz *3 x 3*, são definidos os tipos de relacionamentos entre dois objetos espaciais definidos a seguir. Considerando dois objetos "*A*" e "*B*", eles podem ter os seguintes relacionamentos: *A* e *B* são **disjuntos**; *A* **toca** *B*; *A* é **igual a** *B*; *A* **contém** *B*; *A* **cobre** *B*; *A* está **contido em** *B*; *A* é **coberto por** *B*; *A* **se sobrepõe** a *B* (com limites disjuntos); *A* **se sobrepõe** a *B* (com limites se intersectando). Esses relacionamentos são ilustrados na Figura 2.4.

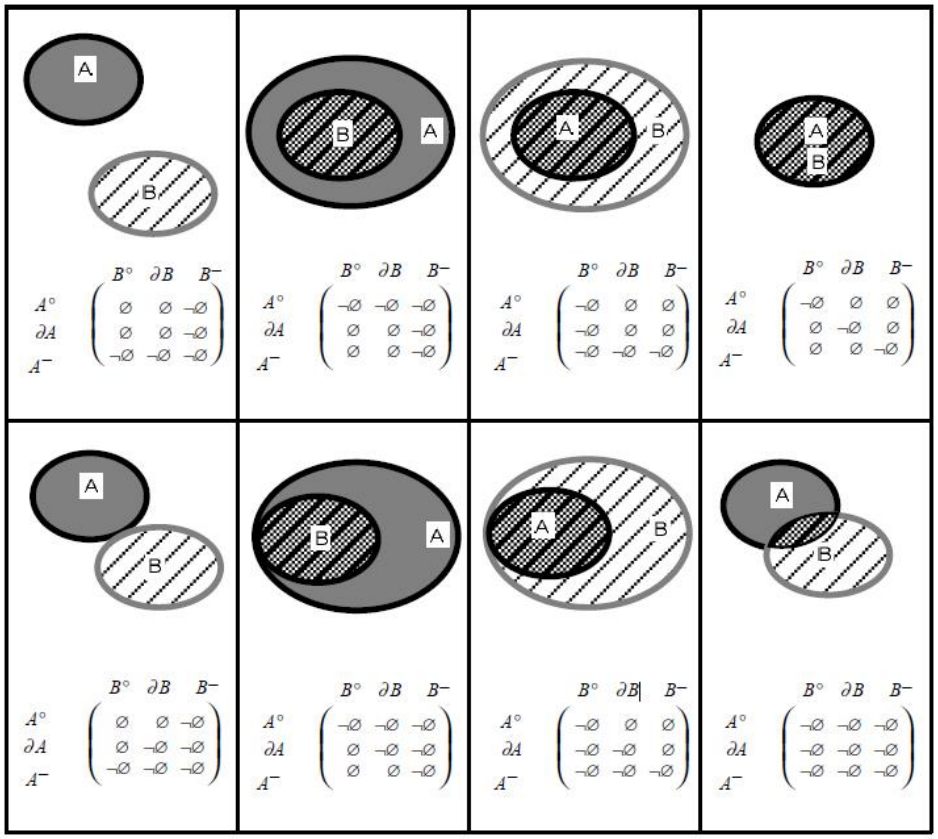

Figura 2.4 Relacionamentos topológicos do modelo de 9-Interseção entre duas regiões. Egenhofer e Herring (1991).

Segundo Egenhofer e Herring (1991), um relacionamento topológico entre dois objetos espaciais pode resultar em dois valores: **Vazio (Ø)** e **Não-vazio(¬Ø)**. Um resultado vazio é caracterizado quando um determinado relacionamento entre dois objetos espaciais não acontece, enquanto que um resultado não-vazio consiste na ocorrência de um determinado relacionamento entre dois objetos.

Os conceitos descritos nessa seção são usados como base para o modelos de dados espaciais vagos QMM e VASA, focos de interesse desta dissertação.

2.5 Geradores de Dados Espaciais Exatos

Diversos geradores de dados espaciais exatos são encontrados na literatura. Nesta seção são descritos os principais trabalhos relacionados a esta pesquisa de mestrado. As seções 2.5.1, 2.5.2, 2.5.3 e 2.5.4 descrevem os geradores *spatial geometry*, *spadawan*, geração de conjuntos de dados espaço-temporal e gerador de polígonos e pontos exatos, respectivamente. Todos esses geradores são geradores de dados espaciais exatos sintéticos.

2.5.1 *Spatial Geometry*

No trabalho de Nascimento *et al.* (2011) foi desenvolvido um gerador de dados espaciais exatos sintéticos denominado *Spatial Geometry* que enfoca a construção de data warehouses espaciais, ou seja, a construção de data warehouses que armazenam dados espaciais exatos. O *Spatial Geometry* gera polígonos simples, mais especificamente, retângulos. Para a geração de um conjunto de retângulos, primeiramente é gerado um retângulo, o qual é dividido em outros retângulos, sendo que esses retângulos podem ter tamanhos diferentes. Essa geração se baseia no modelo de particionamento da *quadtree* (GHAZEL, 2000), no qual quatro novas sub-regiões são decompostas (geradas) recursivamente. A Figura 2.5 representa a geração dos retângulos.

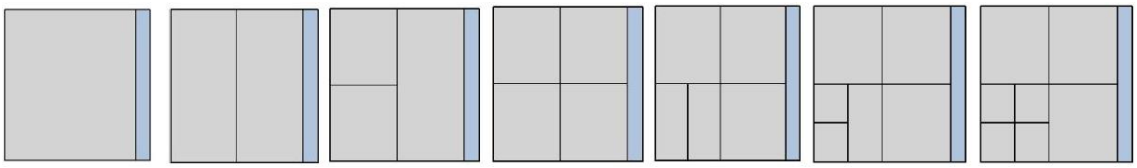

Figura 2.5 Particionamento do Espaço. (adaptado de Nascimento *et al*., 2011).

Os retângulos gerados são distribuídos no espaço inicialmente considerando uma margem de 0,1% em todos os lados do retângulo (Figura 2.6a), e a partir dessa margem são gerados os pontos que compõem o polígono (Figura 2.6b). O valor da margem pode ser configurado e a quantidade total de pontos é dividida pelo número de lados do retângulo inicial. Esses retângulos representam regiões, nações e cidades, sendo que as regiões contêm nações, as quais contêm cidades que contêm ruas e endereços.

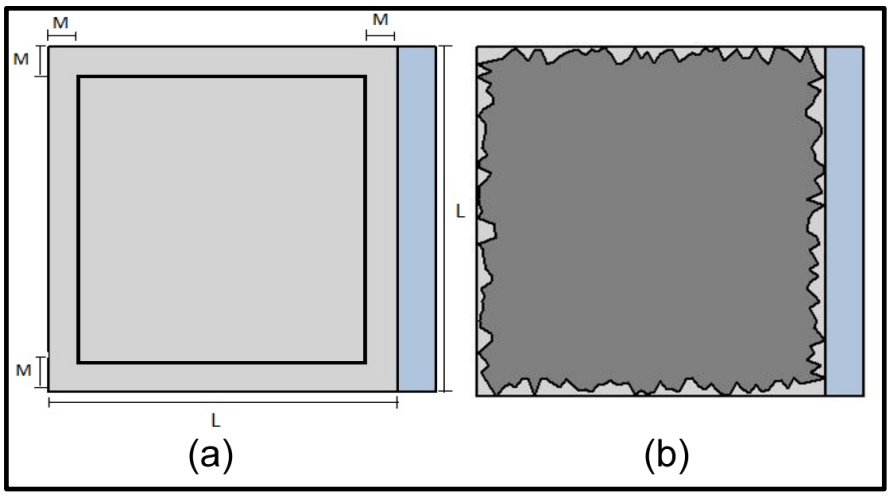

Figura 2.6 (a) Margem do MBR e (b) Geração dos pontos que compõem o polígono. Adaptada de (adaptado de Nascimento *et al*., 2011).

Segundo Nascimento *et al.* (2011), além da geração de polígonos, são geradas linhas exatas simples e pontos exatos simples que representam as ruas e os endereços, respectivamente. As ruas estão contidas nas cidades e os endereços estão contidos nas ruas de forma a representar uma hierarquia de atributos em uma dimensão de um *data warehouse* espacial. As ruas são geradas no centro de cada cidade a partir de um cálculo do tamanho da margem em cada eixo da cidade e são representadas linearmente nos dois eixos *x* e *y*. É possível gerar ***n*** ruas por cidade, conforme ilustrado na Figura 2.7a.

Por fim, são gerados os endereços. Os endereços são distribuídos em um determinado ponto da rua, sendo que nenhum endereço é atribuído aos cruzamentos das ruas, ou seja, o endereço não é um ponto comum entre duas ruas (linhas). A Figura 2.7b mostra o resultado da geração dos endereços.

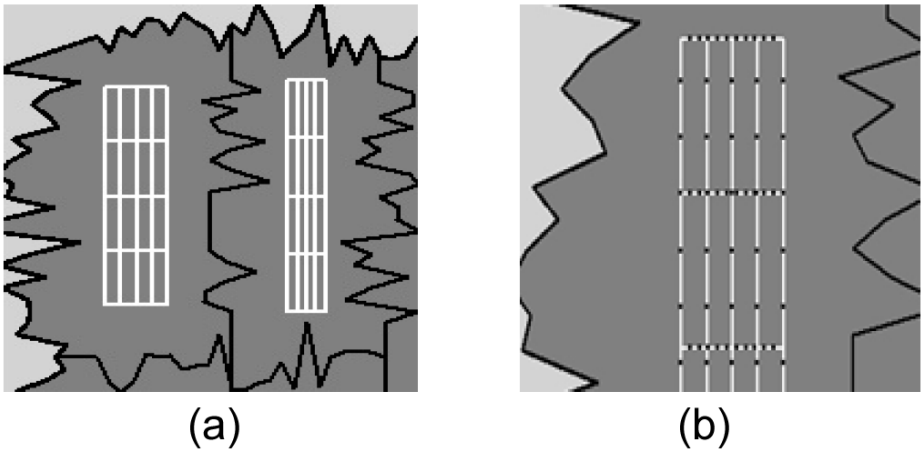

Figura 2.7 (a) Geração das ruas e (b) Geração dos endereços. Adaptada de (Nascimento *et al*., 2011).

Os dados espaciais exatos sintéticos (pontos, linhas e polígonos) gerados pelo gerador *Spatial Geometry* são armazenados em um arquivo texto. Nesse arquivo texto são armazenadas as coordenadas referentes ao dado espacial exato gerado, juntamente com o identificador (*id*) do objeto correspondente.

2.5.2 *Benchmarking spatial data warehouses* (*Spadawan*)

Siqueira *et al.* (2010) desenvolveram um benchmark no qual são gerados e distribuídos aleatoriamente pontos exatos sintéticos dentro de uma determinada região com o intuito de carregar um *data warehouse* geográfico (DWG). Os pontos gerados representam os endereços de clientes e de fornecedores, sendo que esses endereços são únicos e distintos.

Os endereços são gerados dentro de cada cidade e a quantidade de endereços gerados varia de acordo com um fator de escala definido pelo usuário da ferramenta. A Figura 2.8 ilustra a geração e distribuição de pontos que representam fornecedores por cidades.

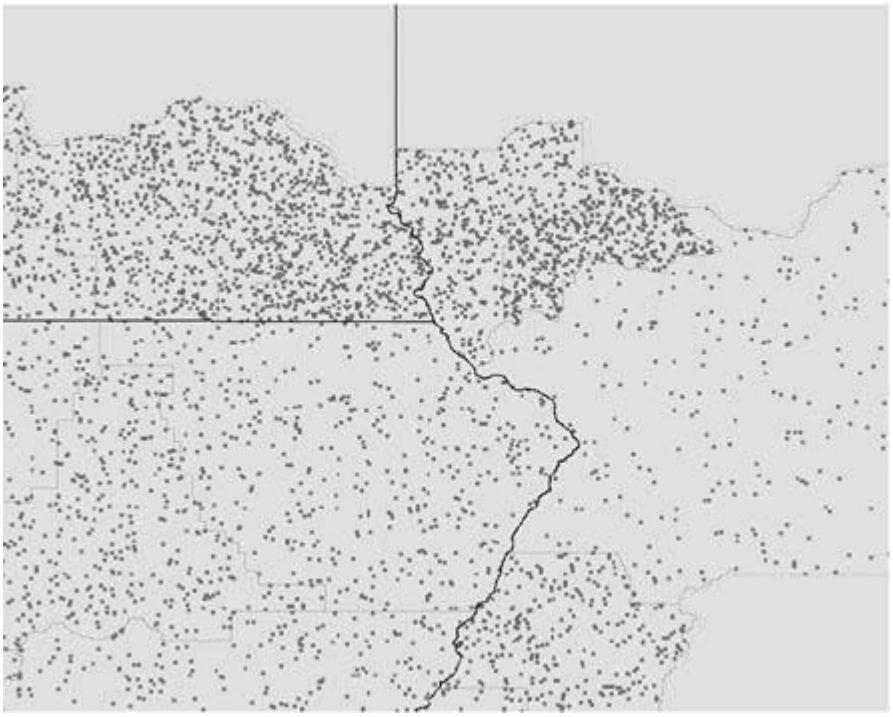

Figura 2.8 Vários endereços de fornecedores por cidade. Siqueira *et al.* (2010).

2.5.3 *Geração de conjuntos de dados espaço-temporal*

Theodoridis *et al.* (1999) abordam em seu trabalho a geração de dados espaciais exatos, mais precisamente pontos exatos sintéticos. Esses pontos são gerados aleatoriamente dentro de uma determinada região, denominada *extent*. Os pontos gerados representam fenômenos do mundo real, tais como plantas, animais e pessoas.

Adicionalmente, Theodoridis *et al.* (1999) consideram que os fenômenos do mundo real se movimentam com o passar do tempo. Por isso, periodicamente os pontos gerados são reposicionados (redistribuídos) no *extent* de acordo com uma função de movimentação. Caso um determinado ponto seja reposicionado fora do *extent*, o mesmo é redistribuído dentro do *extent* novamente a partir de uma função que reposicionamento de pontos. A Figura 2.9 ilustra os conjuntos de dados (pontos) sintéticos gerados em diferentes épocas de tempo.

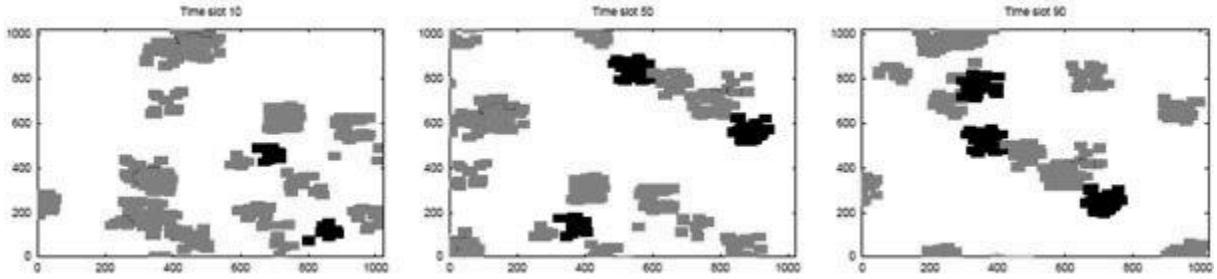

Figura 2.9 Conjuntos de dados (pontos) sintéticos gerados em diferentes épocas de tempo. Adaptado de (Theodoridis *et al.*, 1999).

2.5.4 Gerador de Polígonos Exatos e Pontos Exatos

No trabalho de Ciferri (2002) são gerados dados espaciais exatos sintéticos. Primeiramente é gerado um quadrado e, em seguida, o mesmo é divido em quatro novos quadrados, utilizando como base o modelo de particionamento da *quadtree* (GHAZEL, 2000).

Após o particionamento do quadrado inicial são gerados pontos exatos sintéticos. A geração dos pontos é efetuada em três passos (CIFERRI, 2002): *i*. Define-se a localização na qual os pontos serão gerados; *ii*. Determina-se a organização dos dados, ou seja, é definido se os pontos serão gerados aleatoriamente ou em grupos; *iii*. Por fim, os pontos são gerados dentro da localização definida.

Ademais, é possível gerar regiões exatas simples sintéticas a partir do agrupamento dos pontos localizados no mesmo espaço. Tais regiões podem possuir diversos tamanhos e formas geométricas, dependendo da quantidade de pontos e da distribuição desses pontos no espaço (CIFERRI, 2002).

2.6 Considerações Finais

Neste capítulo foram descritos os principais conceitos de modelos de dados espaciais exatos, os quais servem de base para o entendimento do próximo, capítulo 3, o qual descreve modelos de dados espaciais vagos. Em detalhes, foram detalhados conceitos de dados espaciais, de cálculo de conexão de região e do modelo de 9-interseção.

Ademais, foram descritos diferentes geradores de dados espaciais exatos sintéticos existentes na literatura. Apesar de gerarem diferentes tipos de dados exatos, como pontos exatos, linhas exatas e polígonos exatos, esses geradores não são voltados à geração de dados espaciais vagos. Em especial, não foi encontrado na literatura até o momento nenhum trabalho referente à geração de dados espaciais vagos, bem como geradores de dados espaciais vagos sintéticos, apesar da importância e aplicabilidade desses tipos de dados. A proposta de algoritmos para a geração de dados espaciais vagos, bem como o desenvolvimento de uma ferramenta para auxiliar essa geração são os objetivos deste trabalho de mestrado, e são descritos no capítulo 4.

Capítulo 3

MODELOS DE DADOS ESPACIAIS VAGOS

*Este capítulo descreve conceitos e definições referentes aos modelos de dados espaciais vagos **Egg-Yolk**, **QMM** e **VASA**, considerando que este trabalho de pesquisa baseia-se nestes modelos para a geração de dados. Também é feita uma comparação entre esses modelos.*

3.1 Estruturação do Capítulo

Este capítulo está estruturado da seguinte forma. Na seção 3.2 é descrito o modelo *Egg-Yolk*, enquanto que nas seções 3.3 e 3.4 são resumidos os modelos QMM e VASA, respectivamente. Na seção 3.5 é feita uma comparação entre os modelos descritos. O capítulo é finalizado na seção 3.6, com as considerações finais.

3.2 Modelo *Egg-Yolk*

Lehmann e Cohn (1994) propuseram um modelo de representação de regiões espaciais com fronteiras incertas, ou seja, regiões cujas fronteiras não são precisamente conhecidas (regiões vagas), chamado de modelo *Egg-Yolk*. Esse

modelo foi estendido a partir da teoria do RCC, utilizando duas regiões exatas para representar uma região vaga.

No modelo *Egg-Yolk* uma região vaga é formada por duas sub-regiões exatas concêntricas, sendo que, obrigatoriamente, uma das duas sub-regiões deve estar contida na outra sub-região e suas bordas (limites) devem ser disjuntas, ou seja, seus limites não podem possuir nenhum ponto em comum. Essas duas sub-regiões são denominadas clara e gema.

A primeira sub-região, denominada gema, representa a área mínima da região vaga, ou seja, a área que certamente faz parte dessa região e, todos os pontos localizados dentro dessa região certamente pertencem à região vaga. A segunda sub-região, denominada clara, representa a área máxima da região vaga, que inclui a área sobreposta pela gema. Todos os pontos fora da clara não pertencem à região vaga. A sub-região que pertence à clara e não pertence a gema (isto é parte cinza claro da Figura 3.1) é a parte hipotética da região vaga, ou seja, a área que pode ou não pertencer à região vaga.

A Figura 3.1 ilustra uma região vaga segundo o modelo *Egg-Yolk* com duas regiões exatas, na qual a região cinza escuro (interior) representa a região da gema, a região cinza claro (exterior) representa a região da clara, que não pertence à região da gema.

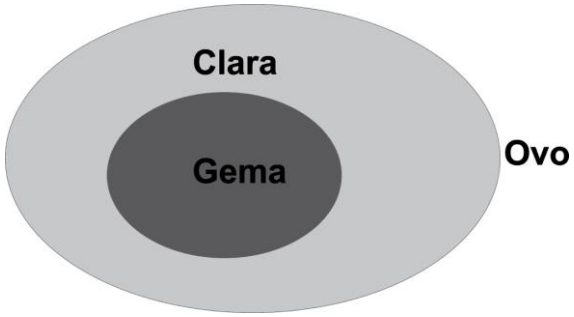

Figura 3.1 Representação do Modelo *Egg-Yolk*. Adaptada de (Cohn *et al*., 1997)

O modelo *Egg-Yolk* possui algumas restrições. A primeira delas refere-se ao fato de que a sub-região gema sempre deve estar contida na sub-região clara. Ademais, uma região vaga obrigatoriamente deve possuir uma sub-região com fronteiras bem definidas (gema), mas pode possuir ou não uma sub-região com fronteiras vagas, ou seja, a clara pode ser uma região vazia (SIQUEIRA *et al.*, 2011). Outra restrição relevante é que as fronteiras (limites) das sub-regiões clara e gema não podem se tocar. Por fim, o modelo *Egg-Yolk* aborda apenas regiões vagas, não

tratando de outros tipos de dados espaciais vagos, como pontos vagos e linhas vagas.

3.3 Modelo Qualitativo Mínimo-Máximo (QMM)

O modelo QMM (BEJAOUI *et al.,* 2009) considera três tipos de objetos espaciais vagos: pontos vagos, linhas vagas e regiões vagas. Cada um desses objetos espaciais vagos é formado por um par de conjuntos denominados **extensão mínima** e **extensão máxima**. A extensão mínima refere-se às partes que certamente pertencem ao objeto espacial e a extensão máxima resulta na união da extensão mínima com as partes hipotéticas, ou seja, as partes nas quais a imprecisão é considerada (isto é, partes que podem ou não pertencer ao objeto espacial). A parte hipotética de um objeto espacial vago é sempre representada por uma região exata, independentemente do tipo de objeto espacial (BEJAOUI *et al.*, 2010). As partes externas à extensão máxima representam os pontos que definitivamente não pertencem ao objeto. A Figura 3.2 exemplifica (a) um ponto vago, (b) uma linha vaga e (c) uma região vaga, considerando suas extensões mínima e máxima.

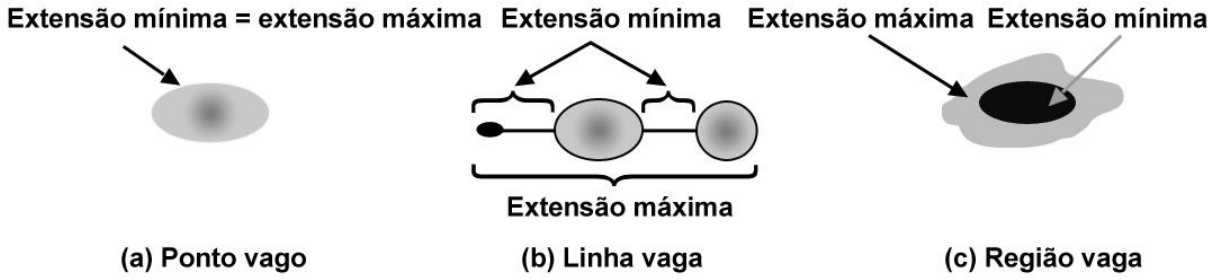

Figura 3.2 Extensões mínima e máxima para (a) um ponto vago, (b) uma linha vaga e (c) uma região vaga. Adaptada de (Bejaoui *et al.*, 2009).

Nas seções 3.3.1, 3.3.2 e 3.3.3 são detalhadas as características de ponto vago, linha vaga e região vaga, respectivamente. Note que o modelo QMM aborda apenas objetos espaciais simples, desconsiderando objetos espaciais complexos, como múltiplos pontos, linhas complexas e polígonos com buraco ou com ilhas.

3.3.1 Ponto Vago

Um ponto vago no modelo QMM ocorre quando há uma falta de conhecimento (incerteza) em relação à sua posição na superfície. No modelo QMM, um ponto vago é representado por uma região exata. Os pontos localizados dentro dessa região exata representam as possíveis posições em que o ponto pode ser encontrado na superfície. Como exemplo do mundo real, pode-se considerar um foco de incêndio, o qual os bombeiros não podem precisamente localizá-lo. No entanto, pode-se localizar uma área em que o foco de incêndio ocorre, a qual é representada por uma região exata. A Figura 3.3 ilustra um ponto exato simples e um ponto vago.

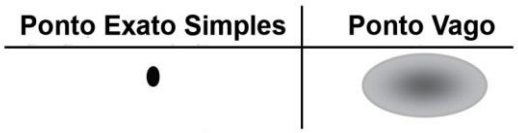

Figura 3.3 Ponto Vago do Modelo QMM. (Bejaoui *et al.*, 2009).

3.3.2 Linha Vaga

Segundo o modelo QMM, uma linha vaga pode conter qualquer um dos invariantes topológicos vagos, ou seja, tanto os limites (extensão) quanto o interior podem ser vagos. Os limites de uma linha correspondem ao ponto inicial e ao ponto final da mesma, e podem ser parcialmente ou totalmente vagos. O interior de uma linha vaga corresponde aos demais pontos que formam a linha, ou seja, todos os pontos menos o ponto inicial e o ponto final. O interior de uma linha vaga pode ser parcialmente ou completamente vago. Uma linha é completamente vaga quando seus limites e o seu interior são vagos. Assim, uma linha completamente vaga corresponde a uma região exata que representa o conjunto de posições que a linha pode preencher. Por outro lado, em uma linha completamente exata, os pontos do interior e os dois pontos limites são bem definidos.

Para definir os níveis de imprecisão de uma linha vaga, Bejaoui *et al.* (2009) usam quatro advérbios: (*i*) fracamente, (*ii*) parcialmente, (*iii*) fortemente e (*iv*) completamente. O termo **fracamente** indica que partes e um dos invariantes topológicos é vago, ou seja, uma parte do interior ou um dos dois limites é vago. O termo **parcialmente** é usado para a imprecisão completa de um dos invariantes

topológicos (limites ou interior) ou caso o interior e limites sejam parcialmente vagos, ao mesmo tempo. Já o termo **fortemente** especifica imprecisão completa para um dos invariantes topológicos e imprecisão parcial para o outro, ou seja, o interior é completamente vago e um dos limites é vago ou os dois limites e parte do interior são vagos. Finalmente, o termo **completamente** indica tanto a precisão exata total quanto a imprecisão total dos componentes da linha vaga, tanto dos limites quanto do interior da linha. Esses advérbios definem os níveis de imprecisão e fornecem uma caracterização qualitativa de uma linha vaga. A partir dessa caracterização qualitativa, são definidos nove tipos de linhas vagas, conforme ilustrado na Figura 3.4.

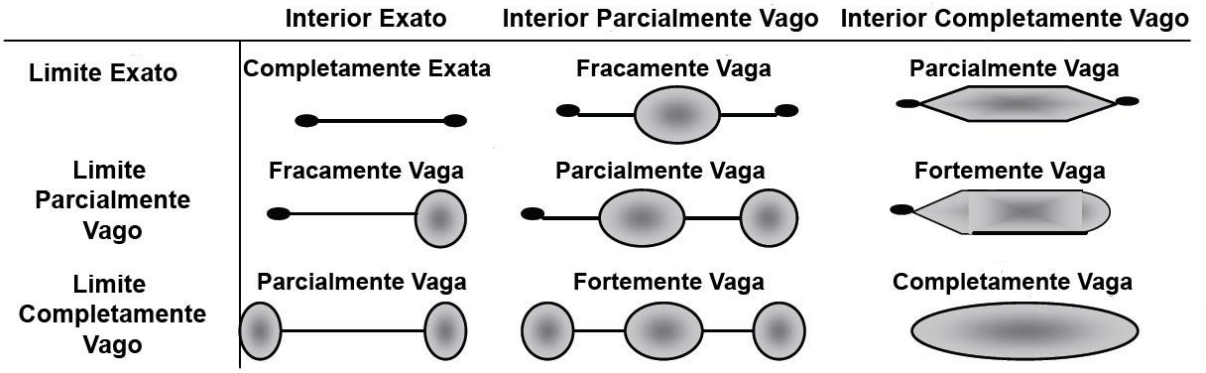

Figura 3.4 Linhas Vagas do Modelo QMM. Adaptada de (Bejaoui *et al*., 2009).

Como exemplo de uma linha vaga no mundo real, mais precisamente uma linha fortemente vaga, pode-se considerar uma linha que representa a trajetória de um avião que desapareceu das telas de um radar alguns minutos após a sua decolagem. A única informação certa é a última posição do avião antes da interrupção da comunicação do avião com o radar. Essa posição (ponto) representa a parte exata da linha e, consequentemente, refere-se à extensão mínima da mesma. A partir desse ponto, a trajetória do avião é desconhecida (vaga), podendo ter qualquer forma e posição dentro de uma área de imprecisão (região exata), representando a parte hipotética da linha. A união da área de imprecisão com a extensão mínima da linha refere-se à extensão máxima da mesma.

O modelo QMM para linhas vagas possui algumas restrições (BEJAOUI *et al.*, 2009): As linhas vagas não podem se cruzar, não podem ser fechadas e não podem ser complexas. Ademais, caso os pontos inicial e final da linha sejam vagos, as duas regiões que os representam não podem se sobrepor.

3.3.3 Região Vaga

Uma região vaga no modelo QMM é formada por duas regiões exatas que definem o limite mínimo e o limite máximo. O limite mínimo refere-se à área que certamente pertence à região vaga (parte exata), enquanto que o limite máximo refere-se à união do limite mínimo com a parte hipotética, ou seja, a área que pode ou não pertencer à região vaga. Uma região vaga surge da dificuldade em distinguir precisamente o interior e o exterior da região por meio de um limite bem definido (BEJAOUI *et al.*, 2009).

Os limites mínimo e máximo são regiões exatas que possuem três invariantes topológicos para cada um dos limites: um interior, um limite e um exterior. O limite mínimo é a região mais interna e a extensão mínima de uma região vaga, enquanto que o limite máximo é a região mais externa, correspondendo à extensão máxima de uma região vaga. Segundo Bejaoui *et al*. (2009), a extensão mínima de uma região vaga obrigatoriamente deve estar contida na sua extensão máxima e os limites da extensão mínima e da extensão máxima podem ser adjacentes (tocar) entre si, ou seja, podem possuir pontos em comum. As regiões vagas segundo o modelo QMM não oferecem suporte para polígonos complexos (polígonos com ilhas, polígonos com buracos e polígonos com ilhas e com buracos).

Bejaoui *et al*. (2010) define três tipos de regiões vagas, de acordo com seus níveis de níveis de imprecisão: (*i*) região exata; (*ii*) região com fronteira parcialmente vaga; (*iii*) região com fronteira completamente vaga.

Uma **região exata** é uma região cujo limite máximo é igual ao limite mínimo, não possuindo nenhum ponto vago.

Uma **região com fronteira parcialmente vaga** ocorre quando o limite mínimo está contido no limite máximo, porém alguns pontos de fronteiras do limite mínimo e do limite máximo são iguais.

Por fim, uma **região com fronteira completamente vaga** ocorre quando o limite mínimo está contido no limite máximo, no entanto, todos os pontos de fronteiras do limite mínimo são diferentes dos pontos de fronteiras do limite máximo, ou seja, o limite mínimo e o limite máximo não possuem nenhum ponto de fronteira em comum.

A Figura 3.5 ilustra um exemplo de cada um desses três tipos de regiões vagas do modelo QMM.

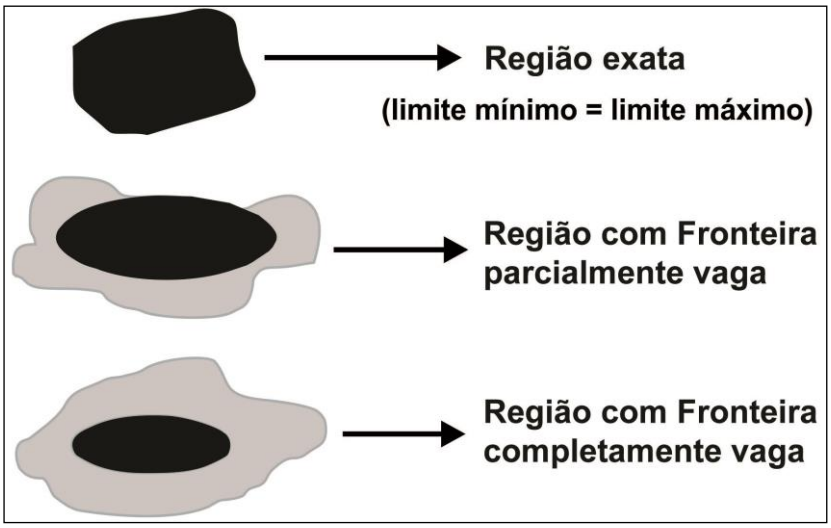

Figura 3.5 Regiões Vagas do Modelo QMM. Adaptada de (Bejaoui *et al*., 2010).

3.4 Álgebra VASA

A Álgebra Espacial Vaga (VASA) proposta por Pauly e Schneider (2010) é um modelo de dados formal que define um conjunto de tipos de dados espaciais vagos, operações espaciais vagas e relacionamentos topológicos vagos. A VASA é baseada nos modelos de dados espaciais exatos para definir os tipos de dados espaciais vagos, aproveitando as definições, técnicas, estruturas de dados e algoritmos existentes. A álgebra VASA reaproveita as definições dos modelos exatos, modificando e ampliando tais definições. Por esse motivo, pode-se executar diretamente operações sobre um conjunto de objetos e predicados espaciais vagos sem a necessidade de se projetar e implementar novos algoritmos.

Um objeto espacial vago pode representar a incerteza sobre caminhos ou a extensão espacial dos fenômenos no espaço (PAULY e SCHNEIDER, 2008), ou seja, os objetos podem encolher ou estender e, portanto, ter uma extensão mínima e máxima. Segundo Pauly e Schneider (2004), um objeto espacial vago é formado por um par de objetos espaciais exatos que são distintos. Este par de objetos pode ser disjunto ou adjacente (ou seja, o par de objetos podem se tocar). O primeiro objeto espacial exato representa a parte do **núcleo**, que corresponde à parte que certamente pertence ao objeto espacial, enquanto que o segundo objeto espacial

exato representa a parte da **conjectura**, que corresponde à parte que pode ou não pertencer ao objeto espacial.

Como exemplo, por meio da VASA, é possível representar um lago cujo nível de água depende da quantidade de chuvas e da evaporação da água do mesmo. A Figura 3.6 ilustra um lago onde a parte cinza escuro representa o **núcleo**, cujo interior definitivamente faz parte do lago (área que possui água o ano todo), ou seja, a parte exata do objeto. A parte cinza claro representa a **conjectura**, ou seja, a área que talvez possa fazer parte do lago (parte hipotética do objeto). A parte cinza claro pode representar tanto a água quanto a praia do lago. Por fim, a parte branca indica a área que definitivamente não faz parte do lago, ou seja, a parte **exterior** do objeto.

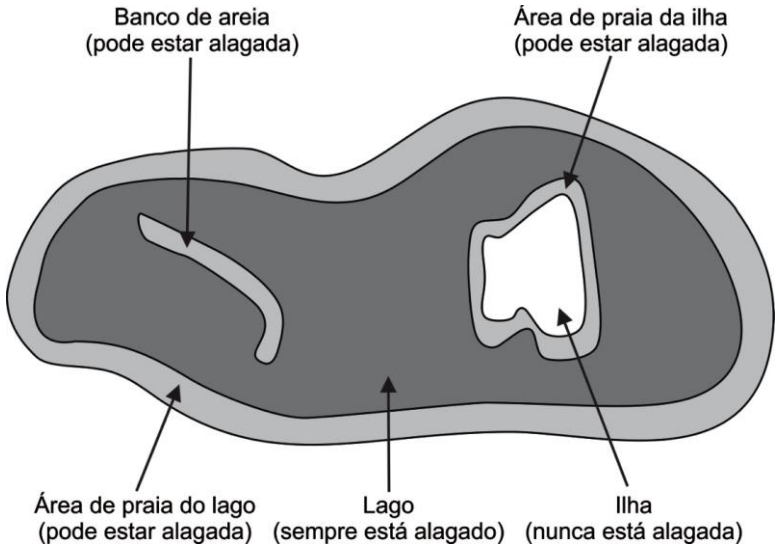

Figura 3.6 Extensão de um lago, dependendo da quantidade de chuva e da evaporação da água. Adaptada de (Pauly e Schneider, 2010)

Pauly e Schneider (2010) definem os tipos de dados espaciais vagos a partir dos modelos de dados espaciais exatos, já bem definidos e consolidados na literatura. Os tipos de dados permitidos pela VASA são genéricos e denominados ponto2D, linha2D e polígono2D e podem se referir a objetos simples ou complexos. Segundo o modelo de dados da VASA, um objeto espacial vago (*point2D, line2D, region2D*) é formado por um par de objetos simples ou complexos exatos, sendo que tais objetos obrigatoriamente devem ser do mesmo tipo de dado espacial exato.

Na álgebra VASA o núcleo e a conjectura devem possuir interiores disjuntos, uma vez que um ponto de um objeto vago não pode pertencer a ambas as partes. Outra característica relevante na VASA é que os limites do núcleo e da conjectura podem ser adjacentes, ou seja, os limites do núcleo e da conjectura podem se

encontrar (tocar), possuindo determinados pontos em comum. Por fim, em casos específicos, tanto o núcleo quanto a conjectura, ou ambos podem ser vazios.

A Figura 3.7 ilustra exemplos de objetos espaciais vagos segundo a VASA. As Figuras 3.7a, 3.7b e 3.7c representam um ponto vago simples, uma linha vaga simples e uma região (polígono) vaga simples, respectivamente, enquanto que as Figuras 3.7d, 3.7e e 3.7f representam um ponto vago complexo, uma linha vaga complexa e uma região vaga complexa, respectivamente. O ponto vago simples Figura 3.7a) é formado um ponto preto, que representa a parte exata (núcleo) e um ponto cinza, que representa parte hipotética (conjectura) do mesmo. A linha vaga simples (Figura 3.7b) é composta por uma linha contínua que pertence ao núcleo, e uma linha tracejada que pertence à conjectura. A região vaga simples (Figura 3.7c) é formada por um polígono preto que representa o núcleo e um polígono cinza que representa a conjectura. O ponto vago complexo (Figura 3.7d) é composto por vários pontos pretos e cinzas, sendo que os pontos pretos pertencem ao núcleo e os pontos cinza pertencem à conjectura. A linha vaga complexa (Figura 3.7e) é formada por várias linhas contínuas e tracejadas, cujos traçados contínuos pertencem ao núcleo, enquanto que os tracejados pertencem à conjectura. Por fim, a região vaga complexa (Figura 3.7f) é composta por diversas regiões pretas e cinzas, onde as regiões pretas pertencem ao núcleo e as regiões cinza pertencem à conjectura.

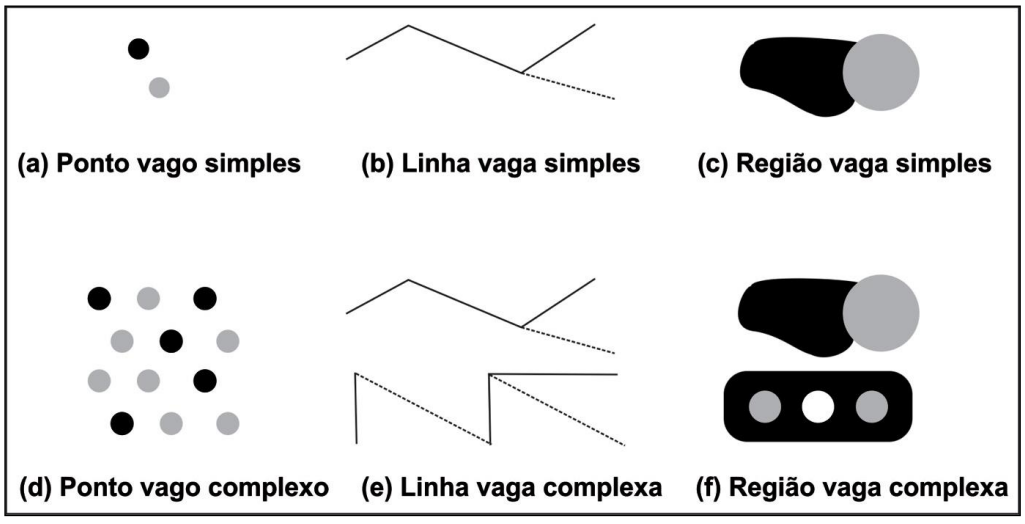

Figura 3.7 Objetos espaciais vagos segundo a VASA.

3.5 Comparação entre os Modelos *Egg-Yolk*, QMM e VASA

O modelo *Egg-Yolk* aborda apenas regiões vagas, não considerando outros tipos de dados espaciais vagos. Uma região vaga é formada por duas regiões exatas denominadas **clara** e **gema**, sendo que a região da gema sempre está contida na região da clara e os limites das duas regiões exatas devem ser disjuntos. Uma região vaga deve obrigatoriamente possuir uma gema, podendo possuir ou não uma clara.

Já o modelo QMM define a representação de objetos espaciais vagos e relacionamentos topológicos entre dois objetos vagos em níveis de imprecisão, considerando três tipos de objetos espaciais vagos: pontos vagos, linhas vagas, e regiões vagas. Cada um desses objetos espaciais é formado por um par de conjuntos denominado **extensão mínima** e **extensão máxima**, sendo que a extensão mínima representa a parte que certamente pertence ao objeto vago e a extensão máxima representa a união da extensão mínima (parte exata) com a parte que pode ou não pertencer ao objeto espacial vago (parte hipotética). A parte hipotética de um objeto espacial vago é sempre representada por uma **região exata**. Em uma região vaga do modelo QMM, a extensão mínima obrigatoriamente está contida na extensão máxima e seus limites podem ser adjacentes. Ademais, o modelo QMM aborda apenas objetos espaciais simples, não enfocando objetos espaciais complexos.

Por fim, a álgebra VASA representa objetos espaciais vagos simples ou complexos. Um objeto espacial vago é formado por um par de objetos espaciais exatos simples ou complexos, sendo que tais objetos devem ser do mesmo tipo de dado. Esse par de objetos é denominado **núcleo** e **conjectura**. O núcleo representa a parte exata do objeto vago enquanto que a conjectura representa a hipotética do objeto espacial vago. Na álgebra VASA o núcleo e a conjectura devem possuir interiores disjuntos e seus limites podem ser adjacentes. Tanto o núcleo quanto a conjectura, ou ambos, podem ser vazios.

Quanto aos relacionamentos topológicos e aos predicados espaciais, o modelo *Egg-Yolk* baseia-se na teoria da RCC, enquanto o modelo QMM e a álgebra VASA baseiam-se na teoria de 9-interseção. A Tabela 3.1 resume os tipos de dados disponíveis para cada um dos modelos.

Tabela 3.1 Tipos de dados disponíveis

	Egg-Yolk	QMM	VASA
Ponto		X	X
Múltiplos Pontos			X
Linha		X	X
Múltiplas Linhas			X
Polígono	X	X	X
Múltiplos Polígonos			X

3.6 Considerações Finais

Neste capítulo foram descritos os modelos de dados espaciais vagos *Egg-Yolk*, QMM e VASA. Também foi realizada uma comparação entre esses modelos. Apesar dos modelos definirem conceitos relacionados aos tipos de dados vagos, bem como as combinações de formas geométricas que definem um determinado tipo de dado vago, definirem operações e relacionamentos para tais modelos, no melhor de nosso conhecimento, não existe na literatura algoritmos para a geração de dados espaciais sintéticos a partir desses modelos de dados espaciais vagos. Esse é o objetivo principal desta dissertação de mestrado, sendo que os algoritmos propostos são detalhados no capítulo 4. No capítulo 4 também é descrita a ferramenta *VagueDataGeneration*, a qual foi desenvolvida para auxiliar a geração dos dados espaciais vagos sintéticos.

Capítulo 4

ALGORITMOS PARA A GERAÇÃO DE DADOS ESPACIAIS VAGOS

*Este capítulo descreve o **desenvolvimento** do trabalho de mestrado. Mais especificamente, este capítulo detalha o **trabalho desenvolvido**, descreve a **metodologia que foi utilizada**, apresenta e detalha os **algoritmos** propostos para a geração de dados espaciais vagos segundo os modelos exatos Egg-Yolk, QMM e VASA e por fim descreve as principais funcionalidades de uma **ferramenta de geração de dados espaciais vagos**, chamada "VagueDataGeneration".*

4.1 Considerações Iniciais

Neste capítulo os resultados do desenvolvimento do trabalho de mestrado são detalhados. A seção 4.2 sumariza os principais objetivos e restrições do trabalho visando destacar o seu escopo, enquanto a seção 4.3 discorre sobre a metodologia utilizada. Nas seções 4.4 e 4.5 são apresentados os algoritmos propostos. Na seção 4.4 são descritos os algoritmos de geração das geometrias (geração de pontos exatos, linhas exatas e polígonos exatos), enquanto na seção 4.5 são descritos os algoritmos de geração dos dados espaciais vagos sintéticos (pontos vagos, linhas vagas e regiões vagas) baseados nos modelos de dados espaciais vagos considerados neste trabalho (modelos *Egg-Yolk*, QMM e VASA). Por fim, a seção 4.6 detalha a ferramenta *VagueDataGeneration*, a qual foi desenvolvida para auxiliar o usuário no processo de geração dos dados espaciais vagos. O capítulo é concluído na seção 4.7 com as considerações finais.

4.2 Detalhamento do Trabalho

O principal objetivo deste trabalho é o desenvolvimento de **algoritmos** para a **geração de dados espaciais vagos** a partir dos modelos de dados espaciais vagos *Egg-Yolk*, QMM e VASA, os quais foram descritos detalhadamente no capítulo 3.

O trabalho desenvolvido dentro deste contexto possui as seguintes características. Primeiro, ele considera apenas dados espaciais vagos no espaço Euclidiano bidimensional. Além disso, no processo de geração de dados espaciais vagos, o trabalho considera as variações das principais propriedades (ou características) dos dados espaciais vagos, a saber: tamanho, formato, volume (ou seja, quantidade de objetos espaciais), complexidade (ou seja, número de pontos que formam os limites dos objetos espaciais vagos, sejam os limites da parte exata ou os limites da parte hipotética), localização e distribuição espacial.

A partir dos algoritmos propostos, é possível gerar dados espaciais sintéticos que representam todos os tipos de dados espaciais vagos presentes nos modelos *Egg-Yolk*, QMM e VASA.

Outro objetivo do trabalho consiste na construção da ferramenta *VagueDataGeneration*, a qual auxilia na geração de dados espaciais vagos de acordo com os algoritmos propostos. Com esta ferramenta, o usuário pode definir qual o modelo de dados espaciais vago e o tipo de dado espacial vago para os quais pretende gerar dados e também pode estabelecer os valores das propriedades dos tipos de dados de cada modelo de dado espacial vago considerado neste projeto.

4.3 Metodologia

A metodologia utilizada para alcançar os objetivos propostos incluiu a constante pesquisa sobre o estado da arte nos temas de geração de dados, modelagem de dados espaciais vagos e *benchmarks* de banco de dados. Foram realizados diversos estudos teóricos sobre esses temas por meio da leitura de livros, artigos, dissertações de mestrado e teses de doutorado.

Os estudos teóricos foram realizados a partir de bibliografias convencionais nas bibliotecas da UFSCar e do ICMC/USP e a partir de bibliografias eletrônicas nos principais portais acadêmicos. Esses estudos teóricos ofereceram a fundamentação para o desenvolvimento das atividades propostas. Ademais, foram realizadas reuniões periódicas com o orientador, além da participação e realização de seminários nas reuniões periódicas do Grupo de Banco de Dados da UFSCar, as quais permitiram a realização de discussões a respeito das soluções propostas para este projeto de mestrado.

Primeiramente, foi realizado o estudo teórico de conceitos relacionados aos tipos de dados espaciais vagos, ao cálculo de conexão de região e ao modelo de 9-interseção. Na sequência, foram estudados os modelos de dados espaciais vagos *Egg-Yolk*, QMM e VASA, os quais são os modelos considerados neste trabalho. Os resultados obtidos nos estudos teóricos realizados foram descritos nos capítulos 2 e 3, respectivamente.

Na sequência, foi feita a proposta dos algoritmos de geração de dados espaciais vagos usando como base os modelos considerados. Para o desenvolvimento desses algoritmos foi utilizado o ambiente computacional Linux, juntamente com o sistema gerenciador de banco de dados (SGBD) *PostgreSQL* com a extensão *PostGIS*. Os algoritmos de geração dos dados espaciais vagos foram implementados na linguagem de programação *PL/pgSQL*, linguagem nativa no SGBD *PostgreSQL*. O SGBD *PostgreSQL* com a extensão *PostGIS* foi o escolhido por se tratar de um banco de dados espacial gratuito e de grande uso. Vale ressaltar que os algoritmos são genéricos e podem ser implementados em qualquer banco de dados espacial.

Posteriormente, foi construída a ferramenta *VagueDataGeneration* para o auxílio à geração dos dados espaciais vagos. A ferramenta foi implementada na linguagem Java utilizando a *IDE Netbeans*. Em especial, a ferramenta executa os algoritmos de geração de dados espaciais vagos sintéticos implementados no SGBD *PostgreSQL*.

Ademais, para validar os algoritmos propostos, foram pesquisados domínios de dados espaciais vagos e definiu-se um estudo de caso para a representação de fenômenos rurais vagos existentes.

4.4 Algoritmos de Geração de Geometrias

Antes do desenvolvimento dos algoritmos de geração de dados espaciais vagos propriamente dito, foi necessário o desenvolvimento de algoritmos para a geração de diversas geometrias exatas. As geometrias exatas geradas são pontos, linhas e polígonos. Especificamente para polígonos, foram gerados quadrados, retângulos, triângulos, círculos e elipses. A geração dessas geometrias foi muito importante para o desenvolvimento do trabalho, pois os dados espaciais vagos segundo os modelos considerados neste projeto são formados por um conjunto de geometrias exatas.

Nas seções 4.4.1, 4.4.2 e 4.4.3 são descritos os algoritmos de geração de pontos exatos, linhas exatas e polígonos exatos, respectivamente. Vale ressaltar novamente que esses algoritmos estão descritos de forma genérica, podendo ser implementados em diferentes linguagens de programação.

4.4.1 Geração de Pontos Exatos

O Algoritmo 4.1 detalha o algoritmo para a geração de pontos exatos, os quais são gerados de forma aleatória dentro de uma determinada região, chamada de “região base”. O algoritmo tem como entrada a região base e retorna o ponto gerado. Suas principais variáveis são:

- **xmin:** armazena o valor mínimo da coordenada *X* do MBR da região base;
- **xmax:** armazena o valor máximo da coordenada *X* do MBR da região base;
- **ymin:** armazena o valor mínimo da coordenada *Y* do MBR região base;
- **ymax:** armazena o valor máximo da coordenada *Y* do MBR da região base;
- **x:** armazena o valor escolhido entre o valor mínimo e máximo da coordenada *X* da região base para a definição do ponto;
- **y:** armazena o valor escolhido entre o valor mínimo e máximo da coordenada *Y* da região base para a definição do ponto;
- **ponto_valido:** recebe verdadeiro se o ponto gerado estiver dentro da região base ou falso caso contrário;
- **ponto_gerado:** recebe o ponto formado pelos valores de x e y.

Algoritmo 4.1 Gera_retorna_ponto

```
gera_retorna_ponto(regiao_base)
Saída: Ponto gerado.
01  xmin ← valor mínimo da coordenada X da regiao_base
02  xmax ← valor máximo da coordenada X da regiao_base
03  ymin ← valor mínimo da coordenada Y da regiao_base
04  ymax ← valor máximo da coordenada Y da regiao_base
05  ponto_valido ← falso
06  enquanto (ponto_valido = falso) faça
07     x ← Random(xmin, xmax)
08     y ← Random(ymin, ymax)
09     ponto_gerado ← ponto(x,y)
10     se (Intersecta(regiao_base, ponto_gerado))
11        então ponto_valido ← verdadeiro
12  fim-enquanto
13  retorna ponto_gerado
```

O Algoritmo 4.1 é detalhado da seguinte forma. Da linha 01 à linha 04 são selecionados os pontos mínimos e máximos das coordenadas *X* e *Y* do MBR da região base. Nas linhas 07 e 08, os valores escolhidos estão entre os valores mínimos e máximos das coordenadas *X* e *Y* da região base e são armazenados nas variáveis relacionadas "*x*" e "*y*". A linha 09 armazena o ponto gerado a partir dos valores atribuídos às variáveis "*x*" e "*y*". Após a geração do ponto, é verificado se ele está dentro da região base (linha 10). Em caso afirmativo, o ponto gerado é aceito (linha 11). Por fim, o ponto gerado é retornado (linha 13).

A Figura 4.1 ilustra um exemplo de geração de pontos em uma região base usando o algoritmo 4.1. A região base do Algoritmo 4.1 e dos demais algoritmos apresentados no capitulo 4 é formada pela união das fronteiras dos municípios de São Carlos, Ibaté e Araraquara.

Figura 4.1 Representação da geração de pontos exatos em uma região base.

4.4.2 Geração de Linhas Exatas

Assim como os pontos, as linhas também são geradas de forma aleatória dentro de uma determinada região, chamada de “região base”. Na geração de uma determinada linha, primeiramente são gerados os pontos inicial e final da linha e, posteriormente são adicionados os demais pontos da mesma. Após a inclusão de todos os pontos que compõem a linha, pode-se rotacionar (girar) a mesma.

O Algoritmo 4.2 ilustra o algoritmo de geração de linhas proposto, sendo que seus parâmetros e variáveis são:

- ✓ **Lista de parâmetros:**
  - **quant_pts_linha:** quantidade de pontos que a linha terá, ou seja, a sua complexidade;
  - **porcentagem:** porcentagem que define o comprimento da linha em relação à média do comprimento do MBR da região base;
  - **giro:** define a rotação da linha gerada (de 0º a 360º);
  - **regiao_base:** região onde será gerada a linha.
- ✓ **Lista de variáveis:**
  - **linha_valida:** recebe verdadeiro se a linha gerada estiver dentro da região base ou falso caso contrário;
  - **xmin, xmax, ymin e ymax:** recebe os valores mínimos e máximos das coordenadas *X* e *Y* do MBR da região base, respectivamente.
  - **x e y:** recebem valores entre o valor mínimo e máximo da coordenada *X* e entre o valor mínimo e máximo da coordenada *Y* do MBR da região base para a definição de um determinado ponto da linha;
  - **deltax e deltay:** recebem o comprimento da coordenada *X* (*xmax - xmin*) e o comprimento da coordenada *Y* (*ymax - ymin*), respectivamente;
  - **delta:** recebe a média do comprimento da coordenada *X* e da coordenada *Y* (*(deltax + deltay) / 2*);
  - **comp_inicial_linha:** recebe o comprimento que a linha inicial terá;
  - **array_pontos[n]:** recebe o conjunto de pontos que compõem a linha gerada;
  - **distancia_entre_pontos:** recebe um valor que representa a distância entre os pontos do interior da linha gerada;

- **linha_gerada:** recebe a linha gerada formada pelo conjunto de pontos armazenados no vetor "*array_pontos[n]*".

Algoritmo 4.2 Gera_retorna_linha

```
gera_retorna_linha(quant_pts_linha, porcentagem, giro, regiao_base)
Saída: Linha gerada.
01  linha_valida ← falso
02  enquanto (linha_valida = falso) faça
03    Defina os pontos mínimos e máximos das coordenadas X e Y a partir
      da Região Base (xmin, xmax, ymin e ymax)
04    deltax ← xmax – xmin
05    deltay ← ymax - ymin
06    delta ← (deltax + deltay) / 2
07    comp_inicial_linha ← (SQRT(porcentagem / 100)) * delta
08    array_pontos ← nulo
09    array_pontos[1] ← gera_retorna_ponto(regiao_base)
10    y ← array_pontos[1].y
11    se (array_pontos[1].x > ((xmin + xmax) / 2)
12      então x ← array_pontos[1].x - comp_inicial_linha
13      senão x ← array_pontos[1].x + comp_inicial_linha
14    fim-se
15    array_pontos[2] ← ponto(x,y)
16    linha_gerada ← constroe_linha(array_pontos)
17    Redefine os pontos mínimos e máximos da coordenada X da Linha
      Gerada (xmin e xmax)
18    ymin ← Y(array_pontos[1]) - (comp_inicial_linha / 10)
19    ymax ← Y(array_pontos[1]) + (comp_inicial_linha / 10)
20    distancia_entre_pontos ← (xmax - xmin) / (quant_pts_linha + 1)
21    x ← xmin
22    i ← 3
23    enquanto (i <= quant_pts_linha) faça
24      x ← x + distancia_entre_pontos
25      Y ← Random(ymin, ymax)
26      linha_gerada ← Adiciona(linha_gerada, ponto(x,y))
27      Incrementa i
28    fim-enquanto
29    Rotaciona (Gira) a linha gerada
      se (Contém(regiao_base, linha_gerada) e
30    Linha_Simples(linha_gerada)) {Função que verifica se a linha não
    cruza}
31      então linha_valida ← verdadeiro
32  fim-enquanto
33  retorna linha_gerada
```

O Algoritmo 4.2 é detalhado da seguinte forma. Nas linhas 01 a 06 são selecionados os pontos mínimos e máximos das coordenadas *X* e *Y* do MBR da região base e são definidos os comprimentos das coordenadas *X* e *Y*. Na linha 07 a variável "*comp_inicial_linha*" recebe o comprimento que a linha inicial (linha base) terá. Isso é feito calculando-se a raiz quadrada da porcentagem passada por parâmetro, multiplicada por delta.

Na linha 09, é gerado o primeiro ponto da linha, ou seja, o ponto inicial, a partir da função *gera_retorna_ponto(regiao_base)* (Algoritmo 4.1). Esse ponto é

adicionado ao vetor "*array_pontos*". Na sequência (linhas 10 a 13), é verificado se a coordenada *X* do primeiro ponto da linha é maior do que média dos valores mínimo e máximo da coordenada *X* da região base. Caso verdadeiro, a variável "*x*" recebe o valor da coordenada *X* do primeiro ponto menos o comprimento inicial da linha (linha 12). Caso falso, a variável "*x*" recebe o valor da coordenada *X* do primeiro ponto mais o comprimento inicial da linha (linha 13). O valor obtido para a variável "x" é então armazenado juntamente com a variável "y" como o segundo ponto da linha, ou seja, o ponto final da linha (linha 15).

Após a definição dos dois primeiros pontos da linha (ponto inicial e ponto final) é construída a linha "base" (linha 16). Essa linha é chamada de linha "base" e o restante dos pontos que a comporão serão adicionados nessa linha gerada. Para tanto, os pontos mínimos e máximos ("*xmin*" e "*xmax*") da coordenada *X* são redefinidos a partir da linha "base" gerada (linha 17). Na linha 18, a variável "*ymin*" recebe o valor da coordenada *Y* do primeiro ponto menos 10% do comprimento inicial da linha, enquanto que na linha 19 a variável "*ymax*" recebe o valor da coordenada *Y* do primeiro ponto mais 10% do comprimento inicial da linha. Este intervalo definido para as variáveis "*ymin*" e "*ymax*" (linhas 17, 18 e 19) consiste no intervalo onde poderá ser atribuído os valores da coordenada y dos demais pontos que farão parte da linha.

Na linha 20, a variável "*distancia_entre_pontos*" recebe o valor da distância entre um ponto i e o próximo ponto (*ponto i + 1*), obtido por meio do comprimento da linha "base" (ou seja, valores mínimo e máximo da coordenada *X* da linha base (*xmax – xmin*)) dividido pela quantidade de pontos da linha mais um (*quant_pts_linha + 1*).

Da linha 21 a 28 os demais pontos são adicionados à linha gerada. A linha 24 adiciona ao valor da coordenada *X* o valor da distância entre os pontos (variável "*distancia_entre_pontos*"), enquanto que na linha 25 é escolhido o valor da coordenada *Y*. Na linha 26 é construído um ponto com as novas coordenadas definidas nas linhas 24 e 25 (*ponto(x,y)*), e esse novo ponto é acrescentado à linha gerada através da função *Adiciona(linha_gerada, ponto(x,y))*.

Após a adição de todos os pontos na linha, a mesma é rotacionada a partir do valor do parâmetro "*giro*", sendo que a linha pode ser rotacionada de 0º a 360º (linha 29). Na linha 30 é verificado se a linha gerada é uma linha simples e se ela está

dentro da região base. Caso a linha gerada satisfaça às duas condições, ela é aceita (linha 31). Por fim, a linha gerada é retornada (linha 33).

A Figura 4.2 ilustra um exemplo de geração de linhas em uma região base usando o Algoritmo 4.2, considerando como parâmetros o tamanho (comprimento) e a quantidade de pontos das linhas.

Figura 4.2 Representação da geração de linhas exatas em uma região base.

4.4.3 Geração de Polígonos Exatos

Assim como os pontos e as linhas, os polígonos também são gerados dentro de uma determinada região de forma aleatória, sendo essa região chamada de "região base". Neste trabalho foram gerados diversos tipos de polígono exatos, a saber: quadrados, retângulos, triângulos, círculos e elipses, os quais são detalhados nas seções 4.4.3.1 a 4.4.3.5, respectivamente.

4.4.3.1 Geração de Quadrados Exatos

O algoritmo proposto para a geração de quadrados exatos gera quadrados em uma determinada região base, onde o primeiro ponto do quadrado é gerado aleatoriamente dentro desta região base. Os demais pontos que compõem o

quadrado são distribuídos uniformemente dentro desta região base. Após a geração do quadrado o mesmo pode ser rotacionado. O Algoritmo 4.3 detalha esse algoritmo, sendo que seus parâmetros e variáveis são descritos como segue.

- ✓ **Lista de Parâmetros:**
  - **porcentagem:** porcentagem que define o tamanho (área) do quadrado em relação ao tamanho (área) da região base;
  - **giro:** define a rotação do quadrado gerado (de 0º a 360º);
  - **regiao_base:** região na qual será gerado o quadrado.
- ✓ **Lista de Variáveis:**
  - **quadrado_valido:** recebe verdadeiro se o quadrado gerado estiver dentro da região base ou falso caso contrário;
  - **xmin, xmax, ymin e ymax:** recebem os valores mínimos e máximos das coordenadas *X* e *Y* do MBR da região base;
  - **x e y:** recebem valores entre o valor mínimo e o valor máximo da coordenada *X* e entre o valor mínimo e o máximo da coordenada *Y* do MBR da região base para a definição de um determinado ponto do quadrado;
  - **deltax e deltay:** *deltax* recebe o comprimento da coordenada *X* (*xmax - xmin*), enquanto que *deltay* recebe o comprimento da coordenada *Y* (*ymax - ymin*);
  - **delta:** recebe a média do comprimento da coordenada *X* e da coordenada *Y* (*(deltax + deltay) / 2*);
  - **valor_lado:** recebe o comprimento que o lado do quadrado (linha) terá;
  - **array_pontos[n]:** recebe o conjunto de pontos que compõem o quadrado gerado;
  - **quadrado_gerado:** recebe o quadrado gerado formado pelo conjunto de pontos armazenados no vetor "*array_pontos[n]*".

Algoritmo 4.3 Gera_retorna_quadrado.

```
gera_retorna_quadrado(porcentagem, giro, regiao_base)
Saída: Quadrado gerado.
01  Defina os pontos mínimos e máximos das coordenadas X e Y a partir
    da Região Base (xmin, xmax, ymin e ymax)
02  deltax ← xmax – xmin
03  deltay ← ymax - ymin
04  delta ← (deltax + deltay) / 2
05  valor_lado ← (SQRT(porcentagem / 100)) * delta
06  quadrado_valido ← falso
```

```
07  enquanto (quadrado_valido = falso) faça
08    array_pontos ← nulo
09    array_pontos[1] ← gera_retorna_ponto(regiao_base)
10    y ← array_pontos[1].y
11    se (array_pontos[1].x > ((xmin + xmax) / 2)
12      então x ← array_pontos[1].x - valor_lado
13      senão x ← array_pontos[1].x + valor_lado
14    fim-se
15    array_pontos[2] ← ponto(x,y)
16    x ← array_pontos[2].x
17    se (array_pontos[2].y > ((ymin + ymax) / 2)
18      então y ← array_pontos[2].y - valor_lado
19      senão y ← array_pontos[2].y + valor_lado
20    fim-se
21    array_pontos[3] ← ponto(x,y)
22    x ← X(array_pontos[1])
23    y ← Y(array_pontos[3])
24    array_pontos[4] ← ponto(x,y)
25    array_pontos[5] ← array_pontos[1]
26    quadrado_gerado ← constroe_poligono(array_pontos)
27    Rotaciona (Gira) o quadrado gerado
28    se (Contém(regiao_base, quadrado_gerado))
29      então quadrado_valido ← verdadeiro
30    fim-se
    fim-enquanto
32  retorna quadrado_gerado
```

O Algoritmo 4.3 é detalhado da seguinte forma. Nas linhas 01 a 06 são selecionados os pontos mínimos e máximos das coordenadas *X* e *Y* do MBR da região base, são definidos os comprimentos das coordenadas *X* e *Y*, e os valores iniciais das variáveis. A linha 09 gera o primeiro ponto do quadrado a partir da função "*gera_retorna_ponto(regiao_base)*" (Algoritmo 4.1). Esse ponto é adicionado ao vetor "*array_pontos*". Na linha 11, é verificado se a coordenada *X* do primeiro ponto é maior do que média dos valores mínimo e máximo da coordenada *X* da região base. Caso verdadeiro, a variável "*x*" recebe o valor da coordenada *X* do primeiro ponto menos o valor do lado do quadrado (linha 12). Caso contrário, a variável "*x*" recebe o valor da coordenada *X* do primeiro ponto mais o valor do lado do quadrado (linha 13).

Na linha 15, o segundo ponto é adicionado ao vetor "*array_pontos*". Na linha 17, é verificado se a coordenada *Y* do segundo ponto do quadrado é maior do que média dos valores mínimo e máximo da coordenada *Y* da região base. Caso verdadeiro, a variável "*y*" recebe o valor da coordenada *Y* do segundo ponto menos o valor do lado do quadrado (linha 18). Caso contrário, a variável "*y*" recebe o valor da coordenada *Y* do segundo ponto mais o valor do lado do quadrado (linha 19).

Na linha 21, o terceiro ponto é adicionado ao vetor "*array_pontos*". Por fim, nas linhas 24 e 25, são adicionados ao vetor "*array_pontos*" o quarto e o quinto pontos, respectivamente. Para a geração de um quadrado é necessário acrescentar um quinto ponto, pois é uma restrição do *PostgreSQL*, onde o primeiro e o último ponto de um polígono devem ser iguais.

Após a definição dos pontos, o quadrado é construído na linha 26, e o mesmo é rotacionado a partir do valor do parâmetro "*giro*", sendo que o quadrado gerado pode ser girado de 0º a 360º (linha 27). A função "*constroe_poligono(array_pontos)*" consiste numa função que gera polígonos a partir de um conjunto de pontos. Na linha 28 é verificado se o quadrado gerado está dentro da região base. Se essa condição for válida, o quadrado é aceito (linha 29). Por fim, o quadrado gerado é retornado (passo 32).

A Figura 4.3 ilustra um exemplo de geração de quadrados em uma região base utilizando o algoritmo 4.3, variando o parâmetro "tamanho" do quadrado.

Figura 4.3 Representação da geração de quadrados exatos em uma região base.

4.4.3.2 Geração de Retângulos Exatos

Os retângulos exatos são gerados a partir do MBR de uma "região base". O algoritmo de geração de retângulos é semelhante ao algoritmo de geração de quadrados. O que difere os dois algoritmos é que na geração de retângulos são definidos duas linhas, correspondentes à coordenada (lado) *X* e *Y* do retângulo. O

Algoritmo 4.4 descreve o algoritmo proposto para a geração de retângulos, o qual usa os seguintes parâmetros e variáveis:

- ✓ **Lista de Parâmetros:**
  - **porcentagem:** define o tamanho (área) do retângulo em relação ao tamanho (área) região base;
  - **giro:** define a rotação do retângulo gerado (de 0º a 360º);
  - **regiao_base:** região onde o retângulo será gerado.
- ✓ **Lista de Variáveis:**
  - **retangulo_valido:** recebe verdadeiro se o retângulo gerado estiver dentro da região base ou falso caso contrário;
  - **xmin, xmax, ymin e ymax:** recebem os valores mínimos e máximos das coordenadas *X* e *Y* do MBR da região base;
  - **x e y:** recebem valores para a definição de um determinado ponto do retângulo;
  - **deltax e deltay:** recebem o comprimento da coordenada *X* (*xmax* - *xmin*) e da coordenada *Y* (*ymax* - *ymin*), respectivamente;
  - **valor_lado_x:** recebe o comprimento do lado do retângulo (linha) da coordenada *X* do retângulo gerado;
  - **valor_lado_y:** recebe o comprimento do lado do retângulo (linha) da coordenada *Y* do retângulo gerado;
  - **array_pontos[n]**: recebe o conjunto de pontos que compõem o retângulo gerado;
  - **retangulo_gerado:** recebe o retângulo gerado formado pelo conjunto de pontos armazenados no vetor "*array_pontos[n]*".

Algoritmo 4.4 Gera_retorna_retangulo

```
gera_retorna_retangulo(porcentagem, giro, regiao_base)
Saída: Retângulo gerado.
01  Defina os pontos mínimos e máximos das coordenadas X e Y a partir
    da Região Base (xmin, xmax, ymin e ymax)
02  deltax ← xmax – xmin
03  deltay ← ymax - ymin
04  valor_lado_x ← (SQRT(porcentagem / 100)) * deltax
05  valor_lado_y ← (SQRT(porcentagem / 100)) * deltay
06  retangulo_valido ← falso
07  enquanto (retangulo_valido = falso) faça
08    array_pontos ← nulo
09    array_pontos[1] ← gera_retorna_ponto(regiao_base)
```

```
10    y ← array_pontos[1].y
11    se (array_pontos[1].x > ((xmin + xmax) / 2)
12      então x ← array_pontos[1].x - valor_lado_x
13      senão x ← array_pontos[1].x + valor_lado_x
14    array_pontos[2] ← ponto(x,y)
15    x ← array_pontos[2].x
16    se (array_pontos[2].y > ((ymin + ymax) / 2)
17      então y ← array_pontos[2].y - valor_lado_y
18      senão y ← array_pontos[2].y + valor_lado_y
19    array_pontos[3] ← ponto(x,y)
20    x ← array_pontos[1].x
21    y ← array_pontos[3].y
22    array_pontos[4] ← ponto(x,y)
23    array_pontos[5] ← array_pontos[1]
24    retangulo_gerado ← constroe_poligono(array_pontos)
25    Rotaciona (Gira) o quadrado gerado
26    se (Contém(regiao_base, retangulo_gerado))
27      então retangulo_valido ← verdadeiro
28  fim-enquanto
29  retorna retangulo_gerado
```

A Figura 4.4 ilustra um exemplo de geração de retângulos em uma região base utilizando o Algoritmo 4.4, variando o parâmetro “tamanho” do retângulo.

Figura 4.4 Representação da geração de retângulos exatos em uma região base.

4.4.3.3 Geração de Triângulos Exatos

Os triângulos exatos são gerados a partir dos quadrados produzidos pelo Algoritmo 4.3. Após a geração do quadrado (Algoritmo 4.3) é gerado um triângulo retângulo, desprezando o quarto ponto do quadrado gerado ou é gerado um

triângulo isósceles, a partir de um novo ponto gerado através da média do coordenada *X* do primeiro ponto e da coordenada *X* do segundo ponto. Após a geração do triângulo o mesmo pode ser rotacionado. O Algoritmo 4.5 detalha o algoritmo para a geração de triângulos, sendo que os parâmetros e variáveis são detalhados como segue.

- ✓ **Lista de Parâmetros:**
  - **porcentagem:** porcentagem que define o tamanho (área) do triângulo em relação ao tamanho (área) região base;
  - **tipo_triangulo:** recebe o tipo de triângulo a ser gerado (triângulo retângulo ou triângulo isósceles);
  - **giro:** define a rotação do triângulo gerado (de 0º a 360º);
  - **regiao_base:** região onde será gerado o quadrado base para a geração do triângulo.
- ✓ **Lista de Variáveis:**
  - **quadrado_base:** recebe o quadrado base necessário para a geração do triângulo;
  - **triangulo_valido:** recebe verdadeiro se o triângulo gerado estiver dentro da região base ou falso caso contrário;
  - **x e y:** recebem valores para a definição de um determinado ponto do triângulo;
  - **array_pts_triangulo[n]:** recebe o conjunto de pontos que formam o triângulo gerado;
  - **triangulo_gerado:** recebe o triângulo gerado formado pelo conjunto de pontos armazenados no vetor "*array_pts_triangulo[n]*".

Algoritmo 4.5 Gera_retorna_triangulo

```
gera_retorna_triangulo(porcentagem, tipo_triangulo, giro, regiao_base)
Saída: Triângulo gerado.
01  triangulo_valido ← falso
02  enquanto (triangulo_valido = falso) faça
03    quadrado_base ← gera_retorna_quadrado(porcentagem, 0,
      regiao_base)
04    array_pts_triangulo ← nulo
05    array_pts_triangulo[1] ← PegaPonto(quadrado_base, 1)
06    array_pts_triangulo[2] ← PegaPonto(quadrado_base, 2)
07    array_pts_triangulo[3] ← PegaPonto(quadrado_base, 3)
08    array_pts_triangulo[4] ← PegaPonto(quadrado_base, 1)
09    se (tipo_triangulo = 'isosceles') então
10      x ← (array_pts_triangulo[4].x + array_pts_triangulo[3].x)/2
```

```
11      y ← (array_pts_triangulo[3].y)
12      array_pts_triangulo[3] ← ponto(x,y)
13    fim-se
14    array_pts_triangulo[4] ← array_pts_triangulo[1]
15    triangulo_gerado ← construoe_poligono(array_pts_triangulo)
16    Rotaciona (Gira) o triângulo gerado
17    se (Contém(regiao_base, triangulo_gerado))
18      então triangulo_valido ← verdadeiro
19    fim-se
20  fim-enquanto
21  retorna triangulo_gerado
```

O Algoritmo 4.5 é detalhado como segue. A linha 03 gera o quadrado “base”, sendo que a partir desse quadrado são definidos os pontos do triângulo a ser gerado. Esses pontos são armazenados no vetor “*array_pts_triangulo*” (da linha 05 à linha 08). Na linha 09 é verificado se o tipo de triângulo a ser gerado é “*isosceles*”. Em caso afirmativo, a variável “*x*” recebe a média da soma da coordenada *X* do terceiro ponto do triângulo com a coordenada *X* do quarto ponto (linha 10), a variável “*y*” recebe a coordenada *Y* do terceiro ponto do triângulo (linha 11), e o terceiro ponto armazenado no vetor “*array_pts_triangulo*” é alterado (passo 12).

Após a definição dos pontos, o triângulo é construído (linha 15) e o mesmo é rotacionado a partir do valor do parâmetro “*giro*”, sendo que o triângulo pode ser girado de 0º a 360º (linha 16). Na linha 17 é verificado se o triângulo gerado está dentro da região base. Em caso afirmativo, ele é aceito (passo 18). Por fim, o triângulo gerado é retornado (passo 21).

A Figura 4.5 ilustra um exemplo de geração de triângulos a partir de quadrados gerados em uma região base utilizando o Algoritmo 4.5, variando o parâmetro “tamanho” do quadrado base e o tipo de triângulo a ser gerado. Os triângulos gerados estão na cor "rosa".

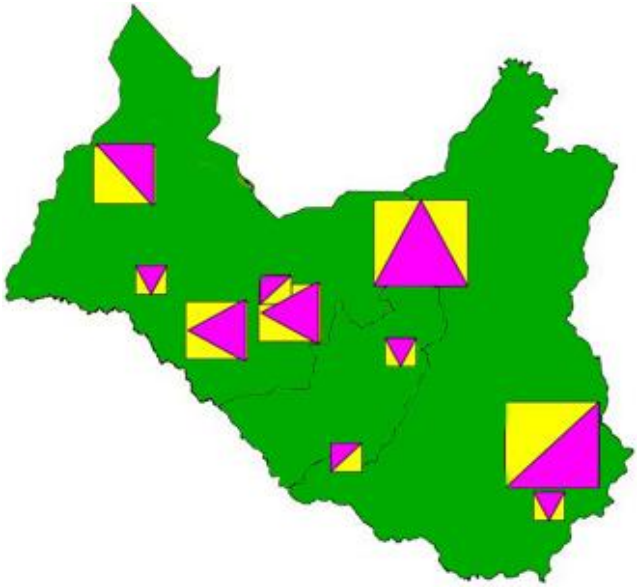

Figura 4.5 Representação da geração de triângulos exatos a partir de quadrados gerados em uma região base.

4.4.3.4 Geração de Círculos Exatos

Um círculo é gerado a partir de um "ponto base" gerado aleatoriamente dentro do MBR de uma "região base". Após o sorteio do "ponto base", um círculo é gerado ao redor desse ponto usando a função de *Buffer*, considerando um valor aproximado para o *buffer*, onde o valor do *buffer* é incrementado ou decrementado de acordo com a porcentagem que esse círculo em relação à região base. O Algoritmo 4.6 detalha o algoritmo para a geração de círculos, sendo seus parâmetros e variáveis descritos a seguir.

- ✓ **Lista de Parâmetros:**
  - **porcentagem:** porcentagem que define o tamanho (área) do círculo em relação ao tamanho (área) região base;
  - **regiao_base:** região onde será gerado o círculo.
- ✓ **Lista de Variáveis:**
  - **xmin, xmax, ymin e ymax:** recebem os valores mínimos e máximos das coordenadas *X* e *Y* do MBR da região base;
  - **deltax e deltay:** recebem o comprimento da coordenada *X* (*xmax - xmin*) e da coordenada *Y* (*ymax - ymin*);
  - **area_regiao_base:** recebe o valor da área da região base;
  - **valor_buffer:** recebe o valor real do *buffer*;

- **circulo_teste:** recebe o círculo gerado para verificar o valor exato do valor do *buffer* para gerar o circulo final com o tamanho exato;
- **area_circulo:** recebe o valor da área do círculo de teste gerado;
- **incremento:** recebe o valor de incremento ou decremento do valor do *buffer*;
- **circulo_valido:** recebe verdadeiro se o círculo gerado estiver dentro da região base ou falso caso contrário;
- **ponto_base:** recebe o ponto base para a geração do círculo, gerado a partir da função "*gera_retorna_ponto(regiao_base)*" (Algoritmo 4.1);
- **circulo_gerado:** recebe o círculo gerado a partir do ponto base e do valor do *buffer* definido.

Algoritmo 4.6 Gera_retorna_circulo

```
gera_retorna_circulo(porcentagem, regiao_base)
Saída: Círculo gerado.
01  Defina os pontos mínimos e máximos das coordenadas X e Y a partir
    da Região Base (xmin, xmax, ymin e ymax)
02  deltax ← xmax – xmin
03  deltay ← ymax - ymin
04  area_regiao_base ← AREA(regiao_base)
05  valor_buffer ← area_regiao_base / 10000
06  circulo_teste ← BUFFER(Ponto(0,0), valor_buffer)
07  area_circulo ← AREA(circulo_teste)
08  se (area_circulo <= ((area_regiao_base * porcentagem)/100)) então
09    incremento ← 0.0001
10    enquanto (area_circulo <= ((area_regiao_base * porcentagem) /
      100))) faça
11      valor_buffer ← valor_buffer + (incremento)
12      circulo_teste ← BUFFER(Ponto(0,0), valor_buffer)
13      area_circulo ← AREA(circulo_gerado)
14    fim-enquanto
15  senão
16    incremento ← -0.0001
17    enquanto (area_circulo > ((area_regiao_base * porcentagem) /
      100))) faça
18      valor_buffer ← valor_buffer + (incremento)
19      circulo_teste ← BUFFER(Ponto(0,0), valor_buffer)
20      area_circulo ← AREA(circulo_gerado)
21    fim-enquanto
22  fim-se
23  circulo_valido ← falso
24  enquanto (circulo_valido = falso) faça
25    ponto_base ← gera_retorna_ponto(regiao_base)
26    circulo_gerado ← BUFFER(ponto_base, valor_buffer)
27    se (Contém(regiao_base, circulo_gerado)) então
28      circulo_valido ← verdadeiro
30  fim-enquanto
31  retorna circulo_gerado
```

O Algoritmo 4.6 trabalha da seguinte forma. Nas linhas 01 a 03 são selecionados os pontos mínimos e máximos das coordenadas *X* e *Y* do MBR da região base e são definidos os comprimentos das coordenadas *X* e *Y*. Na linha 04 a variável "*area_regiao_base*" recebe o valor da área da região base. Na linha 05, a variável "*valor_buffer*" recebe o valor da área da região base dividido por dez mil. Esse valor é o valor inicial, ou seja, o valor aproximado do valor exato do *buffer* (variável "*valor_buffer*") para a geração do círculo.

A partir do valor aproximado, o algoritmo realiza um *loop* (linhas 08 a 14) para definir o valor exato do *buffer* para a geração do círculo. Em detalhes, na linha 06 é gerado um círculo de teste para verificar se o valor do *buffer* está coerente ao tamanho do círculo a ser gerado. Na linha 08, é verificado se a área do círculo gerado para teste é menor ou igual à porcentagem definida no parâmetro "*porcentagem*" da área da região base. Caso seja menor ou igual, a variável referente ao valor do *buffer* (variável "*valor_buffer*") é incrementada (linha 11), é gerado um novo círculo de teste (linha 12) e o valor da área do círculo teste é atualizado (linha 13), até que a área do círculo de teste seja maior que a área da porcentagem da região base (linha 10).

Caso a área do círculo gerado para teste seja maior do que a porcentagem da área da região base, a variável referente ao valor do *buffer* (variável "*valor_buffer*") é decrementada (linha 18), é gerado um novo círculo de teste (linha 19) e o valor da área do círculo teste é atualizado (linha 20), até que a área do círculo de teste seja menor ou igual à área da porcentagem região base (linha 17).

Após a definição do valor exato do *buffer,* um ponto base é gerado dentro da região base (linha 25), sendo que o círculo será gerado a partir desse ponto base e do valor do *buffer* definido anteriormente. Na linha 26 o círculo é gerado e na linha 27 é verificado se esse círculo está dentro da região base. Se essa condição for satisfeita (linha 27), ele é aceito (passo 28). Por fim, o círculo gerado é retornado (linha 31).

A Figura 4.6 ilustra um exemplo de geração de círculos em uma região base utilizando o Algoritmo 4.6, considerando o parâmetro da porcentagem da área do círculo em relação a região base.

Figura 4.6 Representação da geração de círculos exatos em uma região base.

4.4.3.5 Geração de Elipses Exatas

Uma elipse é gerada a partir de um círculo gerado anteriormente, dentro de uma região base. Após a geração do círculo, é gerada a elipse utilizando a função "*Scale*" do *PostgreSQL*. A função "*Scale*" gera uma elipse a partir de um determinado círculo.

O Algoritmo 4.7 detalha o algoritmo para a geração de elipses, sendo seus parâmetros e variáveis descritas a seguir.

- ✓ **Lista de Parâmetros:**
  - **porcentagem:** porcentagem que define o tamanho (área) da elipse em relação ao tamanho (área) região base;
  - **giro:** define a rotação da elipse gerada (de 0º a 360º);
  - **regiao_base:** região onde será gerada a elipse.
- ✓ **Lista de Variáveis:**
  - **xmin, xmax, ymin e ymax:** recebem os valores mínimos e máximos das coordenadas *X* e *Y* do MBR da região base;
  - **deltax e deltay:** recebem o comprimento da coordenada *X* (*xmax - xmin*) e da coordenada *Y* (*ymax - ymin*);
  - **area_regiao_base:** recebe o valor da área da região base;
  - **valor_buffer:** recebe o valor real do *buffer*;

- **elipse_teste:** recebe a elipse que vai ser testada;
- **area_elipse:** recebe o valor da área da elipse de teste gerada;
- **incremento:** recebe o valor de incremento ou decremento do valor do *buffer*;
- **elipse_valida:** recebe verdadeiro se a elipse gerada estiver dentro da região base ou falso caso contrário;
- **ponto_base:** recebe o ponto base para a geração da elipse, o qual é gerado a partir da função "*gera_retorna_ponto(regiao_base)*";
- **elipse_gerada:** recebe a elipse gerada a partir do ponto base e do valor do *buffer* definido.

Algoritmo 4.7 Gera_retorna_elipse

```
gera_retorna_elipse(porcentagem, giro, regiao_base)
Saída: Elipse gerada.
01  Defina os pontos mínimos e máximos das coordenadas X e Y a partir
    da Região Base (xmin, xmax, ymin e ymax)
02  deltax ← xmax – xmin
03  deltay ← ymax - ymin
04  area_regiao_base ← AREA(regiao_base)
05  valor_buffer ← area_regiao_base / 10000
06  elipse_teste ← SCALE(BUFFER(Ponto(0,0), valor_buffer))
07  area_elipse ← AREA(elipse_teste)
08  se (area_elipse <= ((area_regiao_base * porcentagem)/100)) então
09    incremento ← 0.0001
10    enquanto (area_elipse <= ((area_regiao_base * porcentagem) /
      100))) faça
11      valor_buffer ← valor_buffer + (incremento)
12      elipse_teste ← SCALE(BUFFER(Ponto(0,0), valor_buffer))
13      area_elipse ← AREA(elipse_teste)
14    fim-enquanto
15  senão
16    incremento ← -0.0001
17    enquanto (area_elipse > ((area_regiao_base * porcentagem) /
      100))) faça
18      valor_buffer ← valor_buffer + (incremento)
19      elipse_teste ← SCALE(BUFFER(Ponto(0,0), valor_buffer))
20      area_elipse ← AREA(elipse_teste)
21    fim-enquanto
22  fim-se
23  elipse_valida ← falso
24  enquanto (elipse_valida = falso) faça
25    ponto_base ← gera_retorna_ponto(regiao_base)
26    elipse_gerada ← SCALE(BUFFER(ponto_base, valor_buffer))
27    Rotaciona (Gira) a elipse gerada
```

```
28    se (Contém(regiao_base, elipse_gerada))
29      então elipse_valida ← verdadeiro
30    fim-se
31  fim-enquanto
32  retorna elipse_gerada
```

O algoritmo geração de elipses segue a mesma estrutura do algoritmo de geração de círculos, pois uma elipse é gerada a partir de um círculo, acrescentando uma função de geração de elipse previamente definida no SGBD (função "*Scale*" - linhas 6, 12, 19 e 26).

A Figura 4.7 ilustra um exemplo de geração de elipses em uma região base utilizando o Algoritmo 4.7, considerando o parâmetro porcentagem da área da elipse em relação a região base.

Figura 4.7 Representação da geração de elipses exatas em uma região base.

4.5 Algoritmos para a Geração de Dados Espaciais Vagos

Para a geração dos dados espaciais vagos primeiramente foi necessário à geração de diversas formas geométricas (isto é, dados espaciais exatos) conforme descrito na seção 4.4, pois os dados espaciais vagos são gerados a partir de combinações das geometrias exatas descritas na seção 4.4. Como exemplo, para se gerar uma região vaga simples do modelo *Egg-Yolk* é necessário gerar dois polígonos, sendo que o primeiro polígono corresponde à clara e o segundo polígono

corresponde à gema. Outro exemplo, para se gerar uma região vaga simples do modelo QMM também é necessário gerar dois polígonos, sendo que o primeiro polígono corresponde a extensão máxima do objeto espacial vago e o segundo polígono corresponde a extensão mínima do objeto espacial vago. Para o modelo QMM, a extensão máxima e a extensão mínima podem ser de diferentes tipos de dados, por exemplo, a extensão máxima pode ser representada por um círculo, enquanto que a extensão mínima pode ser representada por um quadrado.

Os algoritmos de geração dos dados espaciais vagos propostos neste trabalham visam a obtenção de conjuntos de dados sintéticos. Ademais, eles constroem dados espaciais vagos genéricos, ou seja, um algoritmo proposto que gera um determinado tipo de dado vago a partir de um determinado modelo de dados espaciais vagos pode ser utilizado para a geração e representação de diversos objetos espaciais vagos de diferentes domínios do mundo real.

Cada algoritmo gera apenas um único tipo de dado. Todavia, um mesmo tipo de dado pode ser considerado em um ou mais modelos de dados espaciais vagos. Portanto, um determinado algoritmo pode gerar apenas um tipo de dado vago, porém esse tipo de dado pode ser gerado a partir das definições de diferentes modelos de dados espaciais vagos, alterando apenas características específicas do dado vago gerado.

A seguir são descritos os algoritmos de geração de dados espaciais vagos sintéticos a partir dos modelos de dados espaciais vagos e seus tipos de dados vagos considerados neste projeto. Os algoritmos descritos a seguir abordam apenas um subconjunto dos principais procedimentos para a geração de dados espaciais vagos para os modelos *Egg-Yolk*, QMM e VASA. A quantidade de algoritmos desenvolvidos foi grande e escolheu-se o subconjunto mais representativo para ser apresentado nesta monografia.

Ademais, os algoritmos descritos enfocam especificamente a geração de dados espaciais usando funções do SGBD PostgreSQL/PostGIS. Porém, qualquer SGBD com funções equivalentes às funções do SGBD PostgreSQL/PostGIS também pode ser usado.

Nas seções 4.5.1 a 4.5.5 são descritos os algoritmos de geração de pontos vagos agrupados da VASA, linhas vagas da VASA, linhas vagas do QMM, regiões vagas simples do *Egg-Yolk*, QMM e da VASA e regiões vagas distintas agrupadas da VASA, respectivamente.

4.5.1 Algoritmo de Geração de Pontos Vagos Agrupados (VASA)

O algoritmo de geração de pontos vagos agrupados gera pontos vagos (simples ou complexos) contidos em uma determinada região base. Esses pontos vagos são gerados a partir das definições da VASA (seção 3.4). Para a geração dos pontos vagos agrupados, primeiramente é gerada uma região exata dentro da região base (região mãe) já existente, como por exemplo, um município ou uma área de uma plantação. Esta região gerada é denominada "*regiao_base_elementos*", sendo que esta região possui um tamanho percentual em relação à região base e onde primeiramente são gerados os elementos de núcleo e posteriormente os elementos de conjectura dos pontos vagos.

O Algoritmo 4.8 detalha o algoritmo para a geração de pontos vagos agrupados, sendo seus parâmetros e variáveis descritos a seguir.

- ✓ **Lista de Parâmetros:**
  - **nome_objeto:** nome do objeto que será representado por um ponto vago gerado (ex.: ponto de pragas e/ou parasitas);
  - **quant_pontos_vagos:** quantidade de pontos vagos a serem gerados;
  - **quant_elem_nucleo:** quantidade de elementos (pontos) que o núcleo de cada ponto vago terá;
  - **quant_elem_conjectura:** quantidade de elementos (pontos) que a conjectura de cada ponto vago terá;
  - **porc_area_nucleo:** porcentagem da região onde serão gerados os elementos do núcleo de um ponto vago em relação à região base;
  - **porc_area_conj:** porcentagem da região onde serão gerados os elementos da conjectura de um ponto vago em relação à região base;
  - **tipo_geom_area_nucleo:** tipo da forma geométrica da área base para a geração dos elementos de núcleo do ponto vago (ex.: retângulo, triângulo, círculo);
  - **tipo_geom_area_conj:** tipo da forma geométrica da área base para a geração dos elementos de conjectura do ponto vago;
  - **id_regiao_base:** código da região onde serão geradas as "*regiao_base_elementos*" para a geração dos elementos de núcleo e de conjectura de um ponto vago. Por exemplo, supondo que serão gerados

pontos vagos para representação de pontos de pragas, então antes de gerar os pontos vagos, é gerado uma região exata (região base elementos) que estará contida nessa região base, e os elementos desse ponto vago gerado estarão contidos na região base elementos.

- ✓ **Lista de Variáveis:**
  - **regiao_base_nucleo:** armazena o núcleo da região base (região "mãe"), obtida a partir do *id_regiao_base*;
  - **regiao_base_conj:** armazena a conjectura da região base (região "mãe");
  - **regiao_base_gerada_valida:** verifica se a região base elementos gerada é válida ou não (verdadeiro ou falso);
  - **regiao_nucleo_gerado:** região gerada dentro da "*regiao_base_nucleo*". Consiste na região onde serão gerados os elementos (pontos) do núcleo de um ponto vago;
  - **regiao_conj_gerada:** região gerada dentro da "*regiao_base_conj*". Consiste na região onde serão gerados os elementos (pontos) da conjectura de um ponto vago;
  - **ponto_gerado:** recebe o ponto gerado que, após validado, será armazenado no vetor de pontos do núcleo ou da conjectura;
  - **array_pontos_nucleo:** armazena os elementos (pontos) referentes ao núcleo do ponto vago a ser gerado;
  - **array_pontos_conjectura:** armazena os elementos referentes à conjectura do ponto vago a ser gerado;
  - **nucleo_geom[n]:** representa os elementos de núcleo dos pontos vagos já armazenados em uma estrutura de dados, como exemplo, armazenados em uma tabela, recuperados através de consultas para verificações do algoritmo;
  - **conjectura_geom[n]:** representa os elementos de conjectura dos pontos vagos já armazenados em uma estrutura de dados.

Algoritmo 4.8 Gera_pontos_vagos_agrupados

```
gera_pontos_vagos_agrupados(nome_objeto, quant_pontos,
quant_elem_nucleo, quant_elem_conjectura, porc_area_nucleo,
porc_area_conj, tipo_geom_area_nucleo, tipo_geom_area_conj,
id_regiao_base)
```

Saída: *Confirmação de sucesso ou falha da geração dos pontos vagos agrupados.*

```
01  regiao_base_conj ← conjectura_regiao_base(id_regiao_base)
```

```
02 regiao_base_nucleo ← nucleo_regiao_base(id_regiao_base)
03 regiao_base_gerada_valida ← falso
04 enquanto (regiao_base_gerada_valida = falso) faça
05    regiao_conj_gerada ← gera_poligono(porc_area_conj,
       regiao_base_conj, tipo_geom_area_conj)
06       se(Intersecta(regiao_conj_gerada, regiao_base_nucleo)
07          então regiao_base_gerada_valida ← verdadeiro
08 fim-enquanto
   regiao_nucleo_gerado ← gera_poligono(porc_area_nucleo,
09 Interseção(regiao_base_nucleo, regiao_conj_gerada),
   tipo_geom_area_nucleo)
10 i ← 1
11 enquanto (i <= quant_pontos) faça
12    j ← 1
13    array_pontos_nucleo ← nulo
14    enquanto (j <= quant_elem_nucleo) faça
15       ponto_gerado ← gera_ponto(regiao_nucleo_gerado)
         se (Não Igual(ponto_gerado, array_pontos_nucleo[n]) e
16           Não Igual(ponto_gerado, nucleo_geom[n]) e
             Não Igual(ponto_gerado, conjectura_geom[n]))
17          então
18             array_pontos_nucleo[j] ← ponto_gerado
19             j ← j + 1
20          fim-então
21       fim-se
22    fim-enquanto
23    j ← 1
24    array_pontos_conjectura ← nulo
25    enquanto (j <= quant_elem_conjectura) faça
26       ponto_gerado ← gera_ponto(regiao_conjectura_gerada)
27       se (Não Igual(ponto_gerado, array_pontos_nucleo[n]) e
             Não Igual(ponto_gerado, nucleo_geom[n]))
28          então
29             array_pontos_conjectura[j] ← ponto_gerado
30             j ← j + 1
31          fim-então
32       fim-se
33    fim-enquanto
34    Armazena o ponto vago gerado
35    i ← i + 1
36 fim-enquanto
```

O Algoritmo 4.8 trabalha da seguinte forma. Nas linhas 01 e 02 são executadas consultas que buscam o núcleo e a conjectura da região base, respectivamente, a partir do código de identificação da região base (*id_regiao_base*). Essas consultas retornam as regiões “mães”, ou seja, as regiões onde serão geradas as regiões bases (“*regiao_base_elementos*”) para a geração dos elementos (pontos) do núcleo e da conjectura dos pontos vagos.

Na linha 05 é gerado um polígono de acordo com os valores dos parâmetros “*tipo_geom_area_conj” e* “*porc_area_conj*” e “*regiao_base_conj*”. O polígono gerado é atribuído à variável “*regiao_conj_gerada*”. Após a geração do polígono, é verificado se o mesmo intersecta a região base de núcleo (linha 06). Em caso

afirmativo, o polígono é aceito (linha 07) e os elementos da conjectura do ponto vago a ser gerado estarão contidos nesse polígono. O processo descrito nesse parágrafo é repetido até que seja gerado um polígono que intersecta com a região base de núcleo.

Na linha 09 é gerado um polígono de acordo os valores dos parâmetros "*tipo_geom_area_nucleo*", "*porc_area_nucleo*" e da interseção da região base do núcleo (*regiao_base_nucleo*) com a região de conjectura gerada (*regiao_conj_gerada*). O polígono gerado é armazenado na variável "*regiao_nucleo_gerado*", sendo que os elementos do núcleo de um ponto vago a ser gerado estarão contidos nesse polígono. Os elementos de conjectura poderão ser gerados tanto na região de conjectura gerada e quanto na de núcleo gerada, pois os elementos de conjectura do ponto vago poderão estar contidos também na região de núcleo gerada.

Na linha 15 é gerado um ponto a partir da função "*gera_ponto(regiao_base)*" (Algoritmo 4.1), passando como parâmetro a região base, sendo que o ponto será gerado e estará contido nessa região. O ponto gerado é atribuído à variável "*ponto_gerado*". Após a geração do ponto, é verificado na linha 16 se ele é diferente dos pontos gerados que estão armazenados no vetor (*array_pontos_nucleo[n]*) e se ele é diferente dos pontos já armazenados em alguma estrutura de dados (correspondentes às variáveis "*nucleo_geom[n]*" e "*conjectura_geom[n]*), como por exemplo, uma tabela de um banco de dados. Se essa condição for verdadeira, o ponto gerado é adicionado ao vetor de elementos de núcleo (linha 18) e a quantidade elementos de núcleo é incrementada (linha 19).

Na linha 26 são gerados os pontos que serão armazenados no vetor "*conjectura_geom[n]*". Cada ponto é gerado partir da função "*gera_ponto(regiao_base)*" (Algoritmo 4.1) e é atribuído à variável "*ponto_gerado*". Na linha 27 é verificado se o ponto gerado na linha 26 é diferente dos pontos armazenados no vetor de pontos do núcleo (*array_pontos_nucleo[n]*) e se ele é diferente dos pontos já armazenados armazenados em alguma estrutura de dados (variável "*nucleo_geom[n]*"). Note que na linha 27 são realizados apenas dois testes, enquanto na linha 16 são realizados três testes. Isso ocorre, pois o ponto gerado na linha 26 refere-se a um elemento de conjectura do ponto vago e ele pode ser igual aos pontos armazenados na variável "*conjectura_geom[n]*". Em caso afirmativo, o ponto gerado é adicionado ao vetor de elementos de conjectura (linha 29) e a

quantidade elementos de conjectura é incrementada (linha 30). Note que, o ponto gerado na linha 15 refere aos elementos de núcleo e o ponto gerado na linha 26 refere-se aos elementos de conjectura.

Por fim, na linha 34 o ponto vago agrupado gerado é armazenado em uma tabela de um banco de dados. Um ponto vago agrupado gerado é composto por dois objetos, sendo que um dos objetos contém os elementos (pontos) que formam o núcleo e o outro objeto contém os elementos que formam a conjectura do ponto vago agrupado. Em particular, as linhas 04 a 36 são executadas até que todos os pontos vagos desejados sejam gerados.

A Figura 4.8 ilustra a representação da geração de pontos vagos a partir de regiões geradas em uma região base, usando o Algoritmo 4.8, considerando os parâmetros quantidade de pontos de núcleo (quatro pontos vermelhos) e de conjectura (seis pontos azuis). Note que nessa figura também é ilustrada a região base onde são gerados os pontos vagos.

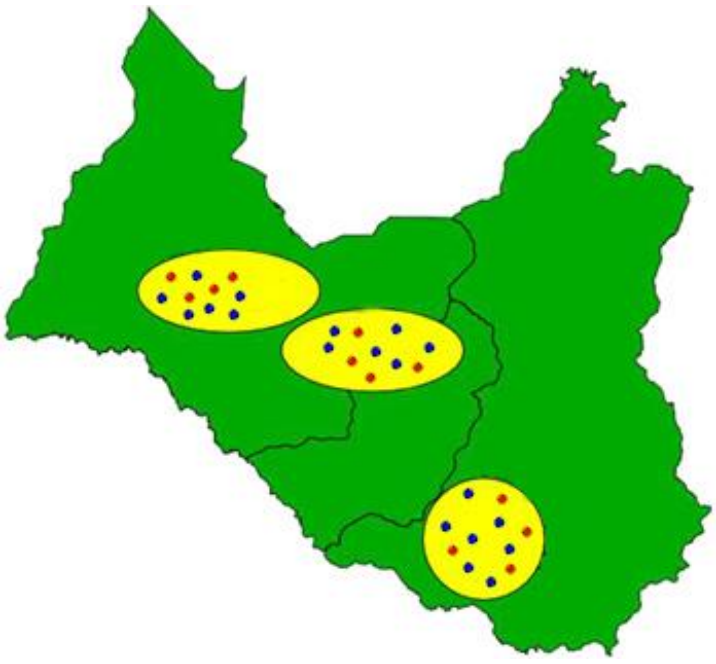

Figura 4.8 Representação da geração de pontos vagos a partir de regiões geradas em uma região base.

O Algoritmo 4.8 gera pontos vagos considerando apenas as características da VASA, pois o modelo *Egg-Yolk* não possui ponto vago e um ponto vago do modelo QMM é representado por um polígono exato. Portanto, para a geração de pontos vagos segundo o modelo QMM, pode-se utilizar os algoritmos de geração de

quadrados, retângulos, triângulos, círculos e elipses, propostos nas seções 4.4.3.1 a 4.4.3.5, respectivamente.

4.5.2 Algoritmo de Geração de Linhas Vagas Agrupadas (VASA)

O algoritmo de geração de linhas vagas agrupadas gera linhas vagas simples ou complexas baseadas nas definições da VASA. Para a geração das linhas vagas agrupadas, primeiramente é gerada uma região exata dentro da região base, como por exemplo, um município ou uma área de uma plantação. Esta região gerada é denominada "*regiao_base_elementos*", possuindo um tamanho em relação a uma determinada porcentagem da região base e onde primeiramente são gerados os elementos de núcleo e posteriormente os elementos de conjectura das linhas vagas.

O Algoritmo 4.9 detalha a geração de linhas vagas agrupadas, sendo seus parâmetros e variáveis descritos a seguir.

- ✓ **Lista de Parâmetros:**
  - **nome_objeto:** nome do objeto que será representado pela linha vaga (ex.: Trilha de Animais);
  - **quant_linhas:** quantidade de linhas vagas a serem geradas;
  - **quant_elem_nucleo:** quantidade de elementos (linhas) que o núcleo de cada linha vaga terá;
  - **quant_elem_conjectura:** quantidade de elementos (linhas) que a conjectura de cada linha vaga terá;
  - **quant_pts_linha_nucleo:** quantidade de pontos que cada elemento (linha) do núcleo terá;
  - **quant_pts_linha_conj:** quantidade de pontos que cada elemento (linha) da conjectura terá;
  - **porc_area_elementos:** porcentagem da região onde serão gerados os elementos de núcleo e conjectura de uma linha vaga em relação à região base;
  - **porc_elem_nucleo:** porcentagem do comprimento de cada elemento do núcleo em relação à região base dos elementos (região onde serão gerados os elementos de núcleo e conjectura de uma determinada região vaga);

- **porc_elem_conj:** porcentagem do comprimento de cada elemento da conjectura em relação à região base dos elementos;
- **giro_nucleo:** rotação de cada elemento do núcleo da linha vaga (de 0º a 360º);
- **giro_conj:** rotação de cada elemento da conjectura da linha vaga (de 0º a 360º);
- **id_regiao_base:** código da região base onde serão geradas as regiões bases elementos (variável *regiao_base_elementos*) para a geração dos elementos de núcleo e de conjectura de uma linha vaga.

✓ **Lista de Variáveis:**

- **regiao_base:** armazena a região base (região "mãe"), obtida a partir do "*id_regiao_base*";
- **regiao_base_elementos:** região gerada dentro da região base. Consiste na região onde serão gerados os elementos de núcleo e de conjectura de uma linha vaga;
- **linha_gerada:** recebe a linha gerada que, após validada, é armazenada no vetor de linhas do núcleo ou da conjectura;
- **array_linhas_nucleo:** armazena os elementos (linhas) referentes aos núcleo da linha vaga a ser gerado;
- **array_linhas_conjectura:** armazena os elementos (linhas) referentes à conjectura da linha vaga a ser gerado;
- **nucleo_geom[n]:** representa os elementos de núcleo das linhas vagas já armazenadas em uma estrutura de dados, como exemplo, armazenados em uma tabela, recuperados através de consultas para verificações do algoritmo;
- **conjectura_geom[n]:** representa os elementos de conjectura das linhas vagas já armazenadas em uma estrutura de dados.

Algoritmo 4.9 Gera_linhas_vagas_vasa

```
gera_linhas_vagas_vasa(nome_objeto, quant_linhas, quant_elem_nucleo,
quant_elem_conjectura, quant_pts_linha_nucleo, quant_pts_linha_conj,
porc_area_elementos, porc_elem_nucleo, porc_elem_conj, giro_nucleo,
giro_conj, id_regiao_base)
```

Saída: *Confirmação de sucesso ou falha da geração das linhas vagas simples ou complexas agrupadas.*

```
01  regiao_base ← regiao_base(id_regiao_base)
02  i ← 1
03  enquanto (i <= quant_linhas) faça
```

```
04      regiao_base_elementos ← gera_poligono(porc_area_elementos,
        regiao_base)
05      j ← 1
06      array_linhas_nucleo ← nulo
07      enquanto (j <= quant_elem_nucleo) faça
08         linha_gerada ← gera_retorna_linha(quant_pts_linha_nucleo,
           porc_elem_nucleo, giro_nucleo, regiao_base_elementos)
           se (Não Igual(linha_gerada, array_linhas_nucleo[n]) e
09              Não Igual(linha_gerada, nucleo_geom[n]) e
                Não Igual(linha_gerada, conjectura_geom[n]))
10             então
11                array_linhas_nucleo[j] ← linha_gerada
12                j ← j + 1
13             fim-então
14         fim-se
15      fim-enquanto
16      j ← 1
17      array_linhas_conjectura ← nulo
18      enquanto (j <= quant_elem_conjectura) faça
19         linha_gerada ← gera_retorna_linha(quant_pts_linha_conj,
           porc_elem_conj, giro_conj, regiao_base_elementos))
           se (Não Igual(linha_gerada, array_linhas_nucleo[n]) e
20              Não Igual(linha_gerada, array_linhas_conjectura[n]) e
                Não Igual(linha_gerada, nucleo_geom[n]))
21             então
22                array_linhas_conjectura[j] ← linha_gerada
23                j ← j + 1
24             fim-então
25         fim-se
26      fim-enquanto
27      Armazena a linha vaga gerada
28      i ← i + 1
29   fim-enquanto
```

O Algoritmo 4.9 trabalha da seguinte forma. Na linha 01 é executada uma consulta que busca a região base a partir do código de identificação da região base (*id_regiao_base*).

Na linha 04 é gerado um polígono (retângulo, quadrado, triângulo, círculo ou elipse, usando os algoritmos propostos nas seções 4.4.3.1 a 4.4.3.5, respectivamente), o qual é atribuído à variável "*regiao_base_elementos*". As linhas vagas serão geradas dentro nesse polígono. Após a definição da região base para a geração dos elementos de núcleo e conjectura da linha vaga, são gerados os elementos de núcleo e de conjectura que compõem a linha vaga, como descrito detalhadamente a seguir.

Na linha 08 é gerada uma linha de acordo com a porcentagem do comprimento da linha (*porc_elem_nucleo*) e com a região base para a geração dos elementos (*regiao_base_elementos*). A linha gerada é atribuída à variável "*linha_gerada*". Na linha 09, é verificado se a linha gerada é diferente das linhas

geradas armazenadas no vetor "*array_linhas_nucleo[n]*" e se a linha gerada é diferente dos elementos do núcleo e da conjectura de outras linhas vagas já armazenadas em alguma estrutura existente. Em caso afirmativo, a linha é adicionada ao vetor de elementos do núcleo (linha 11) e a quantidade elementos de núcleo é incrementada (linha 12). Os passos das linhas 07 a 15 são executados até que todos os elementos do núcleo da linha vaga sejam gerados.

As operações descritas nas linhas 16 a 26 são semelhantes às operações das linhas 05 a 15. A diferença refere-se ao fato de que nas linhas 05 a 15 são gerados elementos do núcleo da linha vaga, enquanto que nas linhas 16 a 26 são gerados elementos de conjectura da mesma linha vaga.

Por fim, na linha 27 a linha vaga gerada é armazenada em uma tabela de um banco de dados. Uma linha vaga gerada é composta por dois objetos, sendo que um dos objetos contém os elementos (linhas) que formam o núcleo e o outro objeto contém os elementos (linhas) que formam a conjectura da linha vaga. Em particular, as linhas 03 a 29 são executadas até que todas as linhas vagas desejadas sejam geradas.

A Figura 4.9 ilustra a representação da geração de linhas vagas, usando o Algoritmo 4.9, considerando como parâmetros o tamanho (comprimento), quantidade de linhas, quantidade de elementos de núcleo e de conjectura, além da quantidade de pontos de cada linha a ser gerada. Na Figura 4.9 são ilustradas três linhas vagas geradas, onde cada linha vaga gerada possui diferentes quantidades de elementos de núcleo (linhas vermelhas) e conjectura (linhas azuis). Nessa figura também é ilustrada as três regiões base (variável *regiao_base_elementos*) nas quais as três linhas vagas foram geradas (polígonos amarelos).

Figura 4.9 Representação da geração de linhas vagas a partir de regiões exatas geradas em uma região base.

O Algoritmo 4.9 gera linhas vagas considerando apenas as características da VASA, pois o modelo *Egg-Yolk* não possui linha vaga. A seção 4.5.3 descreve a geração de linhas vagas segundo o modelo QMM.

4.5.3 Algoritmo de Geração de Linhas Vagas (Modelo QMM)

Uma linha vaga do modelo QMM é composta por uma parte exata e por uma parte hipotética, sendo que a parte conhecida da linha é formada por linhas exatas e a parte hipotética é formada por regiões exatas. Ademais, o modelo QMM considera diversos tipos de linhas vagas, tais como, linha completamente exata, linha completamente vaga, linha fracamente vaga, linha parcialmente vaga e linha fortemente vaga, conforme descrito na seção 3.3.

O Algoritmo 4.10 detalha o algoritmo para a geração de linhas vagas baseados do modelo QMM, sendo seus parâmetros e variáveis descritos a seguir.

- ✓ **Lista de Parâmetros:**
  - **nome_objeto:** nome do objeto que será representado pela linha vaga (ex.: trilha de animais);
  - **tipo:** nome do tipo de linha vaga que será gerado (ex.: linha fracamente vaga, linha fortemente vaga, linha completamente exata);

- **quant_linhas:** quantidade de linhas vagas a serem geradas;
- **quant_pts_linha:** quantidade pontos que cada linha terá;
- **porcentagem:** porcentagem do comprimento de cada linha em relação à média do comprimento do MBR da região base;
- **giro:** rotação de cada linha vaga (de 0º a 360º);
- **id_regiao_base:** código da região base onde serão geradas as linhas vagas.

✓ **Lista de Variáveis:**

- **regiao_base:** região base (região "mãe"), obtida a partir do *"id_regiao_base*";
- **linha_base_gerada:** recebe a linha base gerada. A partir dessa linha será gerada a linha vaga simples;
- **linha_vaga_gerada:** recebe a linha vaga gerada a partir da linha base gerada.

Algoritmo 4.10 Gera_linhas_vagas_qmm

```
gera_linhas_vagas_qmm(nome_objeto, tipo, quant_linhas,
quant_pts_linha, porcentagem, giro, id_regiao_base)
Saída: Confirmação de sucesso ou falha da geração das linhas vagas
simples geradas.
01  regiao_base ← nucleo_regiao_base(id_regiao_base)
02  i ← 1
03  enquanto (i <= quant_linhas) faça
04    linha_base_gerada ← gera_retorna_linha(quant_pts_linha,
      porcentagem, giro, regiao_base)
05    caso tipo for
06      se "LINHA COMPLETAMENTE EXATA" então
07        linha_vaga_gerada ← coleção_geometria(linha_base_gerada)
08      se "LINHA COMPLETAMENTE VAGA" então
09        linha_vaga_gerada ←
           coleção_geometria(BUFFER(linha_base_gerada))
10      se "LINHA FRACAMENTE VAGA" então
11        linha_vaga_gerada ← coleção_geometria(UNIÃO(
           linha_base_gerada, BUFFER(PontoInicial(linha_base_gerada))))
12      se "LINHA PARCIALMENTE VAGA" então
13        linha_vaga_gerada ← coleção_geometria(UNIÃO(
           linha_base_gerada, (UNIÃO(BUFFER(
           PontoInicial(linha_base_gerada)), BUFFER(
           PontoFinal(linha_base_gerada))))
14      se "LINHA FORTEMENTE VAGA" então
15        linha_vaga_gerada ← coleção_geometria(
           PontoInicial(linha_base_gerada), BUFFER(linha_base_gerada))
16          fim-quando
17    Armazena a linha vaga gerada ("linha_vaga_gerada")
18    i ← i + 1
19  fim-enquanto
```

O Algoritmo 4.10 trabalha da seguinte forma. Na linha 01 é executada uma consulta que busca a região de núcleo da região base, a partir do código de identificação da região base (*id_regiao_base*). Essa consulta retorna a região onde serão geradas as linhas vagas. Na linha 04 é gerada uma linha base, a qual é atribuída à variável "*linha_base_gerada*". A linha vaga final será gerada a partir dessa linha base.

Na sequência, é verificado qual o tipo de linha vaga que será gerada e, para cada possibilidade, é executada uma operação particular. Caso a linha seja do tipo "*linha_ completamente_exata*" (linha 06), a linha base gerada anteriormente é atribuída à variável "*linha_vaga_gerada*" (linha 07). Caso a linha seja do tipo "*linha_completamente_vaga*" (linha 08), é gerada uma região exata a partir do *buffer* da linha base e essa região é atribuída à variável "*linha_vaga_gerada*" (linha 09). Caso a linha seja do tipo "*linha_fracamente_vaga*" (linha 10), é gerada uma região exata a partir do *buffer* do ponto inicial da linha base e essa região exata é unida à linha base (linha 11). Caso a linha seja do tipo "*linha_parcialmente_vaga*" (linha 12), são geradas duas regiões exatas a partir do *buffer* do ponto inicial e do ponto final da linha base, e essas duas regiões são unidas à linha base (linha 13). Por fim, caso a linha seja do tipo "*linha_fortemente_vaga*" (linha 14), a variável "*linha_vaga_gerada*" recebe a região exata gerada a partir do *buffer* da linha base e um ponto aleatório da fronteira dessa região exata gerada. Esse ponto aleatório representará a extensão mínima da linha fortemente vaga.

Por fim, na linha 17 a linha vaga gerada é armazenada em uma tabela de um banco de dados. Em particular, as linhas 03 a 19 são executadas até que todas as linhas vagas desejadas sejam geradas.

A Figura 4.10 ilustra a representação da geração dos cinco tipos de linhas vagas do modelo QMM a partir de uma região base usando o Algoritmo 4.10, considerando os parâmetros tipo de linha vaga, tamanho (comprimento), quantidade, tipo de linhas vagas e quantidade de pontos de cada linha vaga gerada. Note que uma linha vaga do modelo QMM é representada por um objeto que contém linhas (vermelhas) e polígonos (azuis), dependendo do tipo de linha vaga.

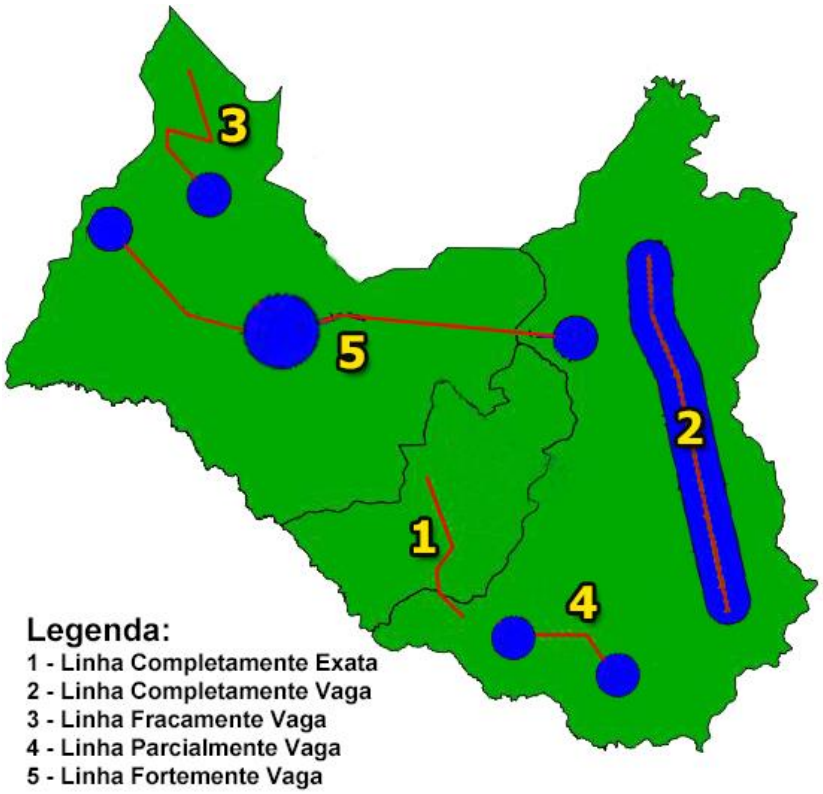

Figura 4.10 Representação da geração dos 5 tipos de linhas vagas do modelo QMM a partir de uma região base.

4.5.4 Algoritmo de Geração de Regiões Vagas Simples

O algoritmo de geração de regiões vagas simples gera regiões vagas simples a partir dos modelos de dados espaciais vagos *Egg-Yolk*, QMM e VASA geram regiões vagas simples contidas em uma região base. O Algoritmo 4.11 detalha o algoritmo para a geração de regiões vagas simples, sendo seus parâmetros e variáveis descritos a seguir.

- ✓ **Lista de Parâmetros:**
  - **modelo:** nome do modelo espacial vago (*Egg-Yolk*, QMM ou VASA);
  - **nome_objeto:** nome do objeto que será representado por uma região vaga simples (ex.: lago, plantação);
  - **quant_regioes:** quantidade de regiões vagas simples a serem geradas;
  - **porc_conj:** porcentagem de cada uma das partes de conjectura em relação à região base;
  - **porc_nucleo:** porcentagem de cada uma das partes de núcleo em relação à sua respectiva conjectura;

- **tipo_geom_conj:** tipo da forma geométrica da conjectura (ex.: retângulo, triângulo, círculo);
- **tipo_geom_nucleo:** tipo da forma geométrica do núcleo;
- **id_regiao_base:** código da região base onde serão geradas as regiões vagas simples. Essa região corresponde à região "mãe" da região vaga a ser gerada. Por exemplo, supondo que as regiões vagas simples representem as plantações, a região base dessas plantações é uma determinada área de cultivo.

✓ **Lista de Variáveis:**

- **regiao_base_conjectura:** armazena a região base para a geração das regiões da conjectura;
- **regiao_base_nucleo:** armazena a região base para a geração das regiões do núcleo;
- **regiao_valida:** verifica se a região vaga simples gerada é válida ou não (verdadeiro ou falso);
- **regiao_conj_valida:** verifica se a conjectura gerada é válida ou não (verdadeiro ou falso);
- **conjectura_gerada:** recebe o polígono gerado referente à conjectura da região vaga;
- **nucleo_gerado:** recebe o polígono gerado referente ao núcleo da região vaga;
- **nucleo_geom[n]:** representa o núcleo das regiões vagas já armazenadas em uma estrutura de dados, como exemplo, armazenado em uma tabela;
- **conjectura_geom[n]:** representa a conjectura das regiões vagas já armazenadas em uma estrutura de dados, como exemplo, armazenada em uma tabela.

Algoritmo 4.11 Gera_regioes_vagas_simples

```
gera_regioes_vagas_simples(modelo, nome_objeto, quant_regioes,
    porc_conj, porc_nucleo, tipo_geom_conj, tipo_geom_nucleo,
    id_regiao_base)
Saída: Confirmação de sucesso ou falha da geração das regiões vagas
simples.
    regiao_base_conjectura ←
01   União(conjectura_regiao_base(id_regiao_base),
      nucleo_regiao_base(id_regiao_base))
02  regiao_base_nucleo ← nucleo_regiao_base(id_regiao_base)
```

```
03  i ← 1
04  enquanto (i <= quant_regioes) faça
05     regiao_valida ← falso
06     enquanto (regiao_valida = falso) faça
07        regiao_conj_valida ← falso
08        enquanto (regiao_conj_valida = falso) faça
09           conjectura_gerada ← gera_poligono(porc_conj,
              regiao_base_conjectura)
10           se(Area(conjectura_gerada) * 0.7) <
              Area(Interseção(conjectura_gerada, regiao_base_nucleo)))
11              então  regiao_conj_valida ← verdadeiro
12        fim-enquanto
13        nucleo_gerado ← gera_poligono(porc_nucleo,
           regiao_base_nucleo)
14        se(modelo = 'VASA')
15           então  conjectura_gerada ← Diferença(
              conjectura_gerada , nucleo_gerado)
          se (Intersecta(conjectura_gerada, nucleo_geom[n]) ou
16         Intersecta(núcleo_gerado, nucleo_geom[n]) ou
           Intersecta(núcleo_gerado, conjectura_geom[n]))
17           então  regiao_valida ← verdadeiro
18     fim-enquanto
19     Armazena a região vaga gerada
20     i ← i + 1
21  fim-enquanto
```

O Algoritmo 4.11 trabalha da seguinte forma. Na linha 01 é executada uma consulta que busca a união da região de núcleo e de conjectura da região base, a partir do código de identificação da região base (*id_regiao_base*). Já a linha 02 executa uma consulta que busca a região de núcleo da região base, a partir do código de identificação da região base (*id_regiao_base*). Essas duas consultas resultam nas regiões nas quais serão geradas as regiões vagas simples, sendo que a região de núcleo da região vaga será gerada dentro do núcleo da região base (variável "*regiao_base_nucleo*") e região de conjectura da região vaga poderá ser gerada dentro do núcleo ou da conjectura da região base (variável "*regiao_base_conjectura*").

Na linha 09 um determinado tipo de polígono (retângulo, quadrado, triângulo, círculo ou elipse) é gerado, de acordo com o tipo especificado no parâmetro "*tipo_geom_conj*". Esse polígono é gerado a partir de funções que geram polígonos exatos. Estas funções também foram implementadas neste projeto e estão descritas nas seções 4.4.3.1 a 4.4.3.5, possuindo os parâmetros "*regiao_base_conjectura*" e "*porc_conj*". O polígono gerado é atribuído à variável "*conjectura_gerada*". Após a geração da conjectura da região vaga, é verificado se pelo menos 70% de sua área está contida na região base do núcleo (linha 10). Em caso afirmativo, a conjectura gerada é aceita (linha 11). Este percentual foi definido para garantir que a maior

parte da região de conjectura gerada está contida no núcleo da região base, pois o núcleo da região vaga a ser gerado deverá estar contida tanto na conjectura da região vaga, quanto no núcleo da região base.

Na linha 13, é gerado um determinado tipo de polígono, de acordo com o tipo especificado no parâmetro "*tipo_geom_nucleo*". Para a geração desse polígono são considerados os parâmetros "*porc_nucleo*" e "*regiao_base_nucleo*". Por fim, o polígono gerado é atribuído à variável "*nucleo_gerado*". Note que esse polígono refere-se ao núcleo, enquanto que o polígono anterior refere-se à conjectura (linha 09).

Na linha 14 é verificado se região vaga simples a ser gerada é baseada na VASA. Em caso afirmativo, então a conjectura é construída pela diferença da conjectura gerada com o núcleo gerado (linha 15). É necessário fazer este teste (linha 14), pois o núcleo e a conjectura na VASA são disjuntos, enquanto nos modelos *Egg-Yolk* e QMM o núcleo está contido na conjectura.

Na linha 16, é verificado se (*i*) a conjectura gerada intersecta o núcleo de outras regiões vagas existentes já armazenadas em alguma estrutura de dados; ou (*ii*) se o núcleo gerado intersecta o núcleo de outras regiões vagas existentes já armazenadas; (*iii*) ou se o núcleo gerado intersecta com a conjectura de outras regiões vagas existentes já armazenadas. Caso a condição seja satisfeita, a região vaga simples gerada é aceita (linha 17).

Por fim, na linha 19 a região vaga simples gerada é armazenada em uma tabela de um banco de dados. Note que uma região vaga simples corresponde a um objeto, formado por duas regiões exatas, na qual a primeira região corresponde à conjectura e a segunda região corresponde ao núcleo da região vaga. Em especial, as linhas 04 a 21 são executadas até que todas as regiões vagas simples sejam geradas.

A Figura 4.11 ilustra a representação da geração de regiões vagas simples a partir de uma região base, utilizando o Algoritmo 4.11, considerando os parâmetros tipo de polígono de núcleo (polígonos vermelhos) e de conjectura (polígonos azuis), o tamanho (porcentagem) do núcleo em relação a conjectura, tamanho (porcentagem) da conjectura em relação a região base e quantidade de regiões vagas.

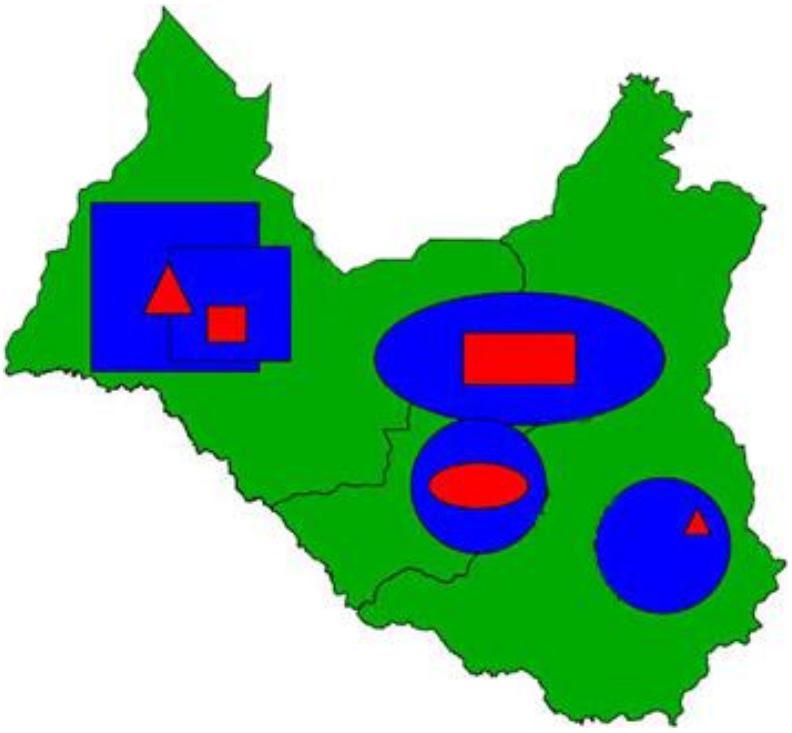

Figura 4.11 Representação da geração de regiões vagas simples a partir de uma região base.

4.5.5 Algoritmo de Geração de Regiões Vagas Distintas Agrupadas

O algoritmo de geração de regiões vagas distintas agrupadas gera regiões vagas simples ou complexas baseadas nas definições de regiões da VASA, pois os modelos *Egg-Yolk* e QMM não abordam regiões vagas distintas. Uma região vaga segundo a VASA pode ser formada por um conjunto de elementos (regiões) que compõem o núcleo e por um conjunto de elementos que compõem a conjectura.

Para a geração das linhas vagas distintas e agrupadas, primeiramente é gerada uma região exata dentro da região base, como por exemplo, um município ou uma área de uma plantação. Esta região gerada é denominada "*regiao_base_elementos*", sendo que esta região possui um tamanho percentual (área) em relação à região base. Na "*regiao_base_elementos*" primeiramente são gerados os elementos de núcleo e posteriormente os elementos de conjectura das regiões vagas.

Os elementos de núcleo e conjectura que formam uma região base possuem seu tamanho em relação à porcentagem da região exata gerada (*regiao_base_elementos*). Ademais, os elementos de núcleo são distintos, pois não possuem nenhum ponto em comum entre si.

O Algoritmo 4.12 detalha o algoritmo para a geração de regiões vagas distintas agrupadas, sendo seus parâmetros e variáveis descritos a seguir.

✓ **Lista de Parâmetros:**

- **nome_objeto:** nome do objeto que será representado por uma região vaga distinta agrupada (ex.: vegetação nativa, área de pragas e/ou parasitas);
- **quant_regioes:** quantidade de regiões vagas a serem geradas;
- **quant_elem_nucleo:** quantidade de elementos (polígonos) que o núcleo de cada região vaga terá;
- **quant_elem_conjectura:** quantidade de elementos (polígonos) que a conjectura de cada região vaga terá;
- **porc_area_elementos:** porcentagem da região exata na qual serão gerados os elementos de núcleo e conjectura de uma região vaga, em relação à região base;
- **porc_elem_nucleo:** porcentagem de cada elemento do núcleo em relação à região base dos elementos;
- **porc_elem_conjectura:** porcentagem de cada elemento da conjectura em relação à região base dos elementos;
- **tipo_geom_area_elementos:** tipo de polígono da área base para a geração dos elementos de núcleo e de conjectura da região vaga (ex.: retângulo, triângulo, círculo);
- **tipo_geom_nucleo:** tipo de polígono dos elementos de núcleo;
- **tipo_geom_conj:** tipo de polígono dos elementos da conjectura;
- **id_regiao_base:** código da região base onde serão geradas as regiões bases (variável "*regiao_base_elementos*") para a geração dos elementos de núcleo e de conjectura de uma região vaga. Essa região corresponde à região "mãe" das regiões vagas a serem geradas. Por exemplo, supondo que as regiões vagas distintas agrupadas a serem geradas representem áreas de pragas/parasitas, então a região base para a geração dessas áreas de pragas/parasitas é uma determinada plantação.

✓ **Lista de Variáveis:**

- **regiao_base:** armazena a região base (região "mãe"), obtida a partir do "*id_regiao_base*";

- **regiao_base_elementos:** gerada dentro da região base, consiste na região onde serão gerados os elementos de núcleo e de conjectura de uma região vaga;
- **regiao_gerada:** recebe o polígono gerado que, após validado, é armazenado no vetor de regiões do núcleo ou da conjectura;
- **array_regioes_nucleo:** armazena os elementos (polígonos) referentes ao núcleo da região vaga a ser gerada;
- **array_regioes_conjectura:** armazena os elementos (polígonos) referentes à conjectura da região vaga a ser gerada;
- **nucleo_geom[n]:** representa os elementos de núcleo das regiões vagas gerados anteriormente e já armazenadas em uma estrutura de dados, como exemplo, armazenados em uma tabela;
- **conjectura_geom[n]:** representa os elementos de conjectura das regiões vagas gerados anteriormente e já armazenadas em uma estrutura de dados.

Algoritmo 4.12 Gera_regioes_vagas_distintas_agrupadas

```
gera_regioes_vagas_distintas_agrupadas(nome_objeto, quant_regioes,
quant_elem_nucleo, quant_elem_conjectura, porc_area_elementos,
porc_elem_nucleo, porc_elem_conjectura, tipo_geom_area_elementos
tipo_geom_nucleo, tipo_geom_conj, id_regiao_base)
Saída: Confirmação de sucesso ou falha da geração das regiões vagas
simples ou complexas distintas agrupadas.
01  regiao_base ← nucleo_regiao_base(id_regiao_base)
02  i ← 1
03  enquanto (i <= quant_regioes) faça
04     regiao_base_elementos ← gera_poligono(porc_area_elementos,
        regiao_base)
05     j ← 1
06     array_regioes_nucleo ← nulo
07     enquanto (j <= quant_elem_nucleo) faça
08        regiao_gerada ← gera_poligono(porc_elem_nucleo,
           regiao_base_elementos)
          se (Não Intersecta(regiao_gerada, array_regioes_nucleo[n]) e
09             Não Intersecta(regiao_gerada, nucleo_geom[n]) e
               Não Intersecta(núcleo_gerado, conjectura_geom[n]))
10           então
11              array_regioes_nucleo[j] ← regiao_gerada
12              j ← j + 1
13           fim-então
14        fim-se
15     fim-enquanto
16     j ← 1
17     array_regioes_conjectura ← nulo
18     enquanto (j <= quant_elem_conjectura) faça
19        regiao_gerada ← gera_poligono(porc_elem_conjectura,
           regiao_base_elementos)
20        se (Não Intersecta(regiao_gerada, array_regioes_nucleo[n]) e
```

```
             Não Intersecta(regiao_gerada, nucleo_geom[n]))
21           então
22              array_regioes_conjectura[j] ← regiao_gerada
23              j ← j + 1
24           fim-então
25        fim-se
26     fim-enquanto
27     Armazena a região vaga gerada
28     i ← i + 1
29  fim-enquanto
```

O Algoritmo 4.12 trabalha da seguinte forma. Na linha 01 é executada uma consulta que busca a região de núcleo da região base, a partir do código de identificação da região base (*id_regiao_base*). Essa consulta retorna a região onde serão geradas as regiões vagas distintas e agrupadas.

Na linha 04 é gerado um polígono (retângulo, quadrado, triângulo, círculo ou elipse) de acordo com os parâmetros *tipo_geom_area_elementos*, "*porc_area_elementos*" e "*regiao_base*". O polígono gerado é atribuído à variável "*regiao_base_elementos*". As regiões vagas distintas e agrupadas serão geradas dentro desse polígono gerado.

Após a definição da região base para a geração dos elementos de núcleo e conjectura da região vaga, são gerados os elementos de núcleo e de conjectura que compõem a região vaga distinta agrupada. Isso é feito da seguinte forma.

Na linha 08 é gerado um polígono de acordo com o tipo especificado no parâmetro "*tipo_geom_nucleo*", a partir de funções que geram formas geométricas, sendo passados os parâmetros "*porc_elem_nucleo*" e "*regiao_base_elementos*". O polígono gerado é atribuído à variável "*regiao_gerada*". Na linha 09 é verificado se a região gerada intersecta com o núcleo de alguma região vaga existente e armazenada em alguma estrutura de dados e se a região gerada intersecta com a conjectura de alguma região vaga existente e já armazenada. Caso não haja intersecção, a região é adicionada ao vetor de elementos de núcleo (linha 11) e a e a quantidade de elementos de núcleo é incrementada (linha 12). As linhas 07 a 15 são executadas até que todos os elementos do núcleo da região vaga distinta agrupada tenham sido gerados.

As linhas 16 a 26 são semelhantes às linhas 05 a 15. A diferença refere-se ao fato de nas linhas 05 a 15 são gerados elementos do núcleo da região vaga, enquanto que nas linhas 16 a 26 são gerados elementos da conjectura da mesma região vaga.

Por fim, na linha 27 a região vaga distinta agrupada gerada é armazenada em uma tabela de um banco de dados. Como destacado anteriormente, uma região vaga distinta agrupada gerada é composta por dois conjuntos, sendo que um dos conjuntos contém os elementos (polígonos) que formam o núcleo e o outro conjunto contém os elementos que formam a conjectura da região vaga distinta agrupada. Em especial, as linhas 03 a 29 são executadas até que todas as regiões vagas distintas agrupadas sejam geradas.

A Figura 4.12 ilustra a representação da geração de regiões vagas distintas agrupadas a partir de regiões exatas geradas dentro de uma região base, utilizando o Algoritmo 4.12, considerando os parâmetros tipo de polígono de núcleo, de conjectura e da região base gerada, o tamanho (porcentagem) do núcleo e da conjectura em relação a região base gerada, o tamanho da região base gerada em relação a região base "mãe", quantidade de regiões vagas a serem geradas, quantidade de elementos de núcleo e de conjectura de cada região vaga gerada. Nesta figura foram geradas três regiões vagas complexas, onde cada região vaga possui diferentes quantidades de elementos de núcleo (polígonos vermelhos) e de conjectura (polígonos azuis). Nessa figura também é ilustrada as três regiões base (polígonos amarelos) nas quais as três regiões vagas foram geradas.

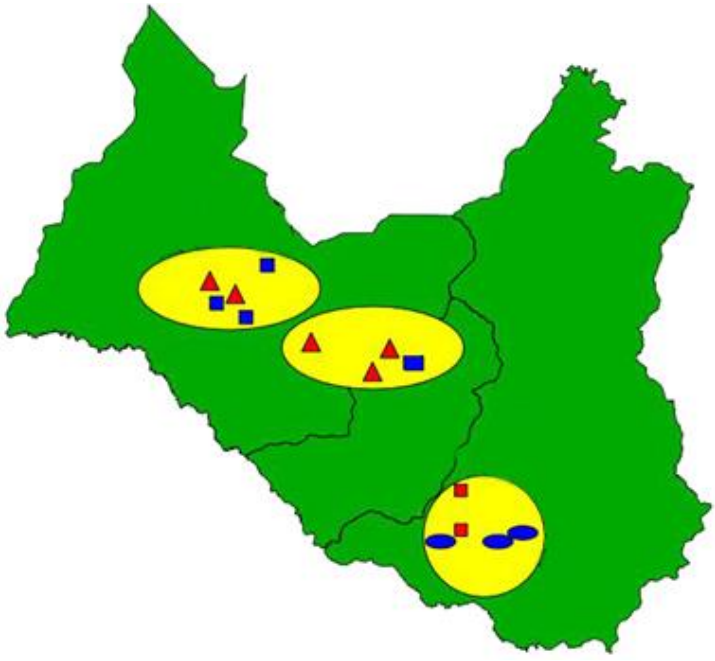

Figura 4.12 Representação da geração de regiões vagas distintas agrupadas a partir de regiões exatas geradas dentro de uma região base.

4.6 Construção da ferramenta *VagueDataGeneration*

A ferramenta de geração de dados espaciais vagos *VagueDataGeneration* foi desenvolvida com o intuito de auxiliar o usuário na geração de dados espaciais vagos. A ferramenta possui diversas funcionalidades que facilitam a parametrização e a definição dos dados a serem gerados a partir de um determinado algoritmo desenvolvido, considerando os conceitos e particularidades de um determinado modelo e de um tipo de dado espacial vago. A partir das opções selecionadas pelos usuários, a ferramenta gera os dados espaciais vagos de acordo com os algoritmos propostos nas seções 4.4 e 4.5.

A ferramenta de geração e o manual de instalação e uso podem ser obtidos no endereço eletrônico http://gbd.dc.ufscar.br/vaguedatageneration. O manual de instalação e uso está disponível também no *Apêndice A* desta monografia.

Na seção 4.6.1 são ilustrados exemplos de telas da ferramenta, enquanto que na seção 4.6.2 descreve e ilustra a modelagem do esquema do banco de dados da ferramenta *VagueDataGeneration*.

4.6.1 Exemplos de telas da Ferramenta *VagueDataGeneration*

A seguir são ilustrados as principais funcionalidades da ferramenta de geração de dados espaciais vagos *VagueDataGeneration*. Todas as funcionalidades da ferramenta são descritos no Apêndice A dessa dissertação. O Apêndice A apresenta o manual de instalação e uso da ferramenta. O manual de instalação e uso da ferramenta e o executável da ferramenta podem ser obtidos no endereço eletrônico http://gbd.dc.ufscar.br/vaguedatageneration.

A Figura 4.13 ilustra a tela inicial da ferramenta, a qual permite que o usuário selecione qual o modelo de dados que ele irá usar como base (ou seja, *Egg-Yolk*, QMM ou VASA) e que tipo de dado ele quer gerar.

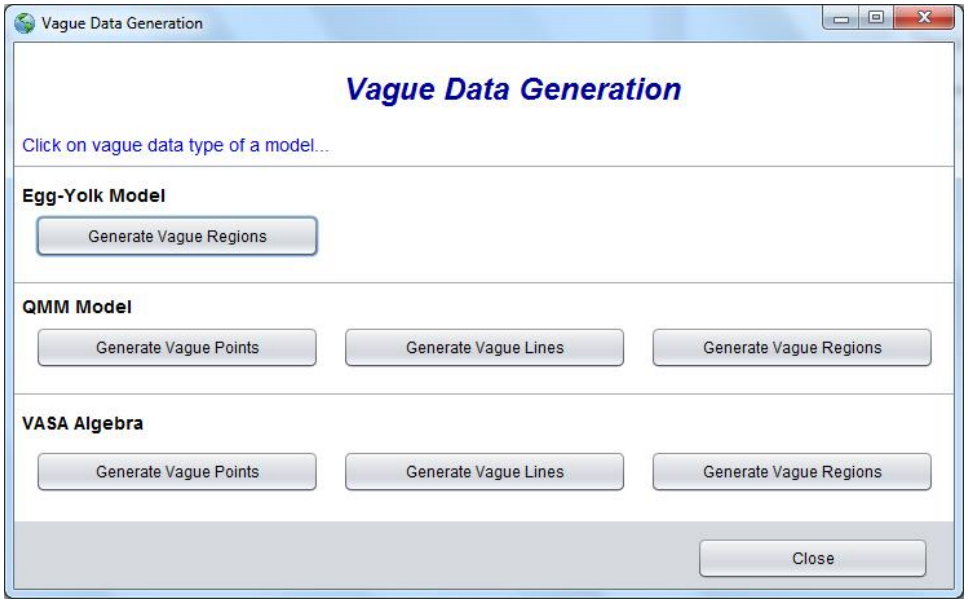

Figura 4.13 Tela Inicial da Ferramenta *VagueDataGeneration*.

Após o escolher o modelo de dado e o tipo de dado desejado, uma nova tela se abrirá. Nas telas de geração dos tipos de dado de acordo com o modelo, o usuário é capaz de selecionar parâmetros como o tamanho, o formato, a quantidade, entre outros, dependendo do tipo de dado/modelo de dado espacial vago selecionado. A Figura 4.14 ilustra uma tela de geração de tipo de dado/modelo, a qual corresponde à geração de regiões vagas (simples ou complexas) segundo as definições da VASA.

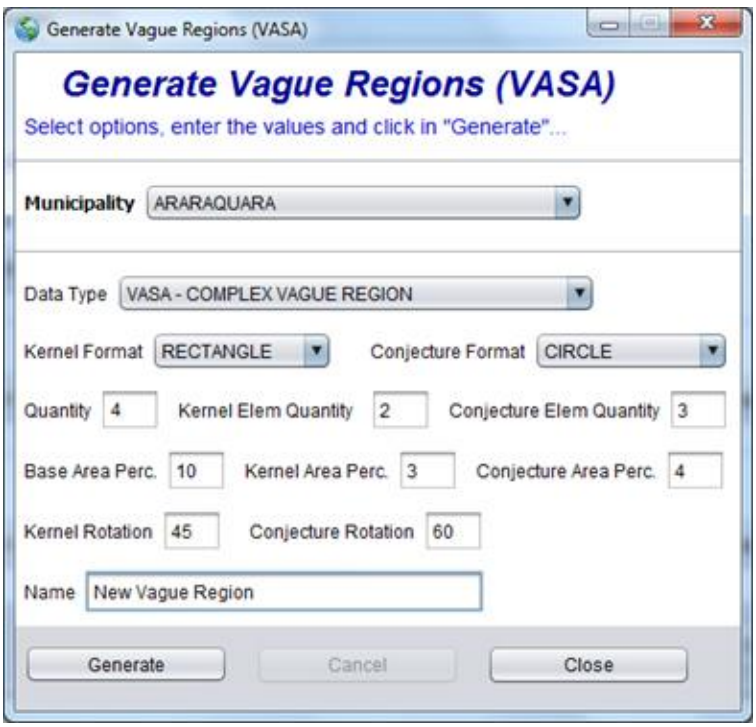

Figura 4.14 Tela de Geração de Regiões Vagas segundo a VASA.

As demais telas da ferramenta possuem o mesmo formato da tela de geração de regiões vagas segundo a VASA ilustrada na Figura 4.14, gerando os dados espaciais vagos a partir das características e particularidades de cada modelo e tipo de dado considerado neste projeto.

Para o desenvolvimento da ferramenta foi utilizada a linguagem de programação Java, e a *IDE Netbeans*. A linguagem Java foi escolhida por se tratar de uma linguagem robusta, estável e *open source* (código aberto).

Na seção 4.6.2 descreve a modelagem do esquema do banco de dados da ferramenta *VagueDataGeneration*.

4.6.2 Modelagem do Esquema do Banco de Dados da Ferramenta *VagueDataGeneration*

Para armazenar os dados espaciais vagos sintéticos gerados pela ferramenta *VagueDataGeneration* proposta, foi modelado e implementado um banco de dados relacional contendo tabelas e colunas geométricas (*geometry*).

Os dados espaciais vagos gerados são armazenados em colunas do tipo *geometry*, onde o núcleo e a conjectura de um dado vago serão armazenados em colunas distintas na tabela.

As regiões vagas geradas a partir do modelo *Egg-Yolk* são armazenadas na tabela "*egg_yolk_vague_region*", onde a clara é armazenada na coluna "*egg_geom*", enquanto a gema é armazenada na coluna "*yolk_geom*".

Na coluna "*the_geom*" são armazenados as duas partes, tanto a clara, quanto a gema. Estamos armazenando duas vezes o mesmo dado vago gerado, para proporcionar diferentes tipos de armazenamentos para o mesmo objeto espacial vago. Esses diferentes tipos de armazenamentos poderão ser importantes para testes técnicas de processamento de consultas e teste de desempenho de índices sobre esses dados gerados e armazenados, por exemplo.

Os pontos vagos gerados a partir do modelo QMM são armazenados na tabela "*qmm_vague_point*", na coluna "*the_geom*" onde é armazenado apenas uma região exata, que representa um ponto vago conforme a definição do ponto vago desse modelo. Já as linhas vagas geradas a partir do modelo QMM são armazenadas na tabela "*qmm_vague_line*", onde as partes exatas da linha vaga

gerada podem ser armazenadas nas colunas "*geom_point*" e "*geom_line*" e as partes hipotéticas são armazenadas na coluna "*geom_region*". As regiões vagas geradas a partir do modelo QMM são armazenadas na tabela "*qmm_vague_region*", onde a parte exata é armazenada na coluna "*min_geom*", enquanto a parte hipotética é armazenada na coluna "*max_geom*". Na coluna "*the_geom*" são armazenados tanto a parte exata, quanto a parte hipotética.

Os pontos vagos gerados a partir da VASA são armazenados na tabela "*vasa_vague_point*", onde os elementos de núcleo são armazenados na coluna "*crisp_geom*" e os elementos de conjectura são armazenados na coluna "*vague_geom*". Já as linhas vagas geradas a partir da VASA são armazenadas na tabela "*vasa_vague_line*", onde os elementos de núcleo são armazenados na coluna "*crisp_geom*" e os elementos de conjectura são armazenados na coluna "*vague_geom*". Por fim, as regiões vagas geradas a partir da VASA são armazenadas na tabela "*vasa_vague_region*", onde os elementos de núcleo são armazenados na coluna "*crisp_geom*" e os elementos de conjectura são armazenados na coluna "*vague_geom*".

A Figura 4.15 ilustra a modelagem lógica relacional do esquema do banco de dados utilizado para armazenamento dos dados espaciais vagos gerados pelos algoritmos de geração executados na ferramenta *VagueDataGeneration*.

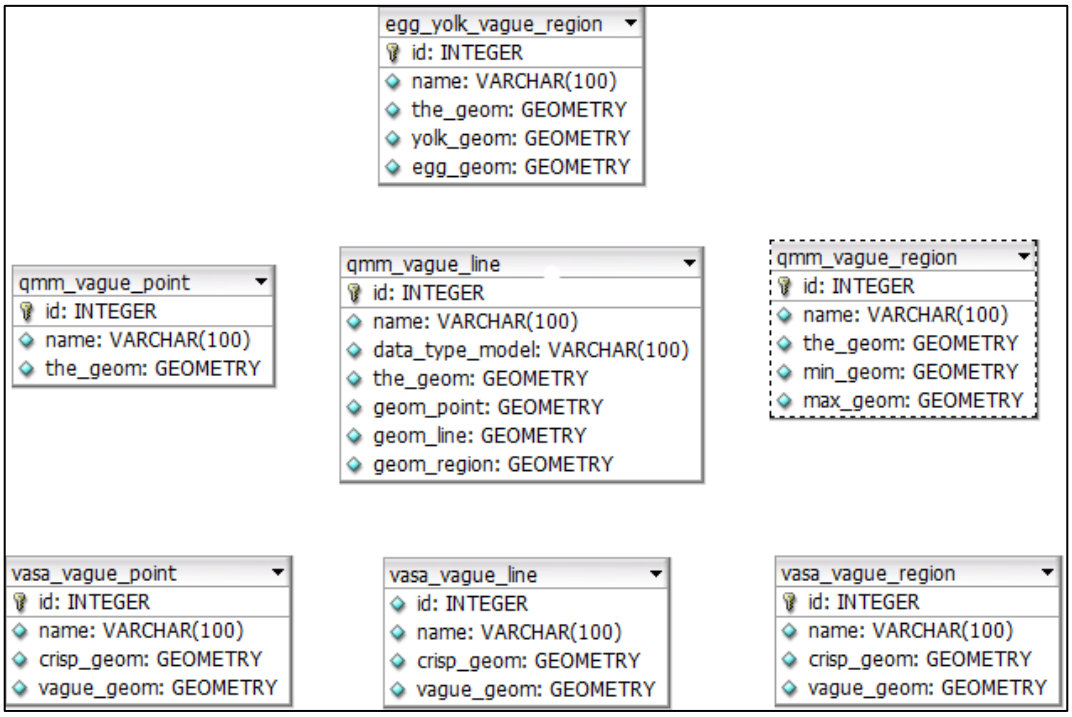

Figura 4.15 Modelagem Lógica Relacional do Esquema do Banco de Dados da Ferramenta *VagueDataGeneration*.

4.7 Considerações Finais

Neste capítulo foram apresentados os algoritmos propostos neste projeto, os quais referem-se aos algoritmos de geração de dados espaciais vagos sintéticos, além dos algoritmos de geração de geometrias exatas. Também foi descrita a ferramenta *VagueDataGeneration*, a qual foi implementada para oferecer ao usuário uma forma mais simples de gerar os dados espaciais vagos, por meio da qual o usuário pode esses dados escolhendo parâmetros e definindo características desejadas. A partir dos algoritmos propostos, é possível gerar dados espaciais vagos sintéticos baseados nos tipos de dados espaciais vagos dos modelos *Egg-Yolk*, QMM e VASA, respeitando as características e limitações de cada modelo.

No capítulo 5 é apresentado um estudo de caso referente à geração de dados espaciais vagos, o qual é usado para validar o trabalho desenvolvido.

Capítulo 5

ESTUDO DE CASO

*Este capítulo descreve um **Estudo de Caso** com o objetivo de mostrar a aplicabilidade da proposta de geração de dados espaciais vagos. Mais especificamente, este capítulo apresenta o **desenvolvimento** de uma **Ferramenta de Geração de Fenômenos Rurais Vagos** e a **Modelagem do Esquema do Banco de Dados desta Ferramenta**. Por fim, são apresentadas as **hipóteses** definidas no início trabalho, juntamente com as suas respectivas respostas aplicadas a este estudo de caso.*

5.1 Geração de Fenômenos Rurais Vagos

Como estudo de caso visando mostrar a aplicabilidade da proposta de mestrado foi desenvolvida uma aplicação para a geração de dados espaciais vagos considerando os fenômenos vagos encontrados em uma zona rural. Como exemplo de fenômenos rurais pode-se citar áreas de cultivo, plantações, vegetação nativa, rios, pragas e parasitas. As fronteiras ou a localização espacial destes fenômenos podem ser incertas e, por isso, tais fenômenos são considerados objetos espaciais vagos.

Neste estudo de caso, primeiramente foram definidos o conjunto de fenômenos existentes em uma zona rural. Logo após a definição dos fenômenos, foram definidos quais tipos de dados e quais as combinações de geometrias poderiam representar um determinado fenômeno. Foram consideradas as seguintes formas geométricas: pontos, linhas e polígonos (subdivididos em retângulo, quadrado, triângulo, círculo e elipse).

A Tabela 5.1 lista os fenômenos considerados neste estudo de caso, quais modelos podem representar um determinado fenômeno e qual a combinação de geometrias formam o fenômeno vago.

Tabela 5.1 Lista de Fenômenos Rurais Vagos.

		Egg-Yolk	QMM	VASA	Formato do Fenômeno
1	Área de cultivo	Região Vaga	Região Exata Região Vaga	Região Exata Região Vaga Simples	Quadrado, retângulo, triângulo, círculo ou elipse
2	Corredor da Área de Cultivo	-	Linha Exata	Linha Exata	Linha Reta
3	Fumaça	Região Vaga	Região Vaga	Região Vaga Simples	Círculo ou Elipse (a partir das queimadas)
4	Irrigação	-	Linha Exata, Linha Vaga, Região Exata	Linha Exata Simples Linha Vaga Simples Linha Exata Complexa Linha Vaga Complexa	Linha ou círculo (buffer da linha)
5	Lago	Região Vaga	Região Exata Região Vaga	Região Exata Região Vaga Simples	Elipse (Contorno) Retângulo, triângulo, círculo ou elipse (Buraco)
6	Município	-	Região Exata	Região Exata Simples Região Exata Complexa	Dados Reais do IBGE para os Municípios de São Carlos, Araraquara e Ibaté
7	Pastagem	Região Vaga	Região Exata Região Vaga	Região Exata Região Vaga Simples Região Vaga Complexa	*Difference* (área não ocupada por nenhum fenômeno gerado)
8	Pesticida	Região Vaga	Região Vaga	Ponto Exato Região Vaga Simples	Ponto, círculo, elipse, quadrado, retângulo ou triângulo
9	Plantação (Talhão)	Região Vaga	Região Exata Região Vaga	Região Exata Região Vaga Simples	Quadrado, retângulo, triângulo, círculo ou elipse (Curvas de nível)
10	Praga e/ou Parasita	Região Vaga	Ponto Vago Região Vaga	Ponto Vago Simples Ponto Vago Complexo Região Vaga Simples Região Vaga Complexa	Pontos, círculos ou elipses
11	Propriedade Rural	-	Região Exata	Região Exata	Quadrado, retângulo ou triângulo
12	Queimada	Região Vaga	Região Vaga	Região Vaga Simples	Círculo, elipse, quadrado, retângulo ou triângulo
13	Rio	-	-	Linha Exata Linha Vaga	Linha com Diversos Pontos (IBGE)
14	Tipo de Solo	Região Vaga	Região Vaga	Região Vaga Simples	Círculo, elipse, quadrado, retângulo ou triângulo
15	Trilha de animais	-	Linha Vaga	Linha Vaga Simples Linha Vaga Complexa	Linha com Diversos Pontos
16	Vegetação nativa	Região Vaga	Região Exata Região Vaga	Região Exata Região Vaga Simples	Quadrado, retângulo, triângulo, círculo ou elipse

Todos os fenômenos serão gerados dentro de uma região base e de forma sintética. A região base consiste em um município pré-definido. Cada um destes fenômenos será gerado individualmente, podendo ser gerados de forma independente ou a partir de outro fenômeno gerado anteriormente, sempre respeitando os limites da região base. Após a definição da região base (município),

uma região exata foi gerada. Esta região representa uma propriedade rural. Diversos fenômenos podem ser gerados dentro de uma propriedade rural. A Figura 5.1 ilustra um exemplo de geração e distribuição de diversos fenômenos gerados dentro de uma determinada propriedade rural.

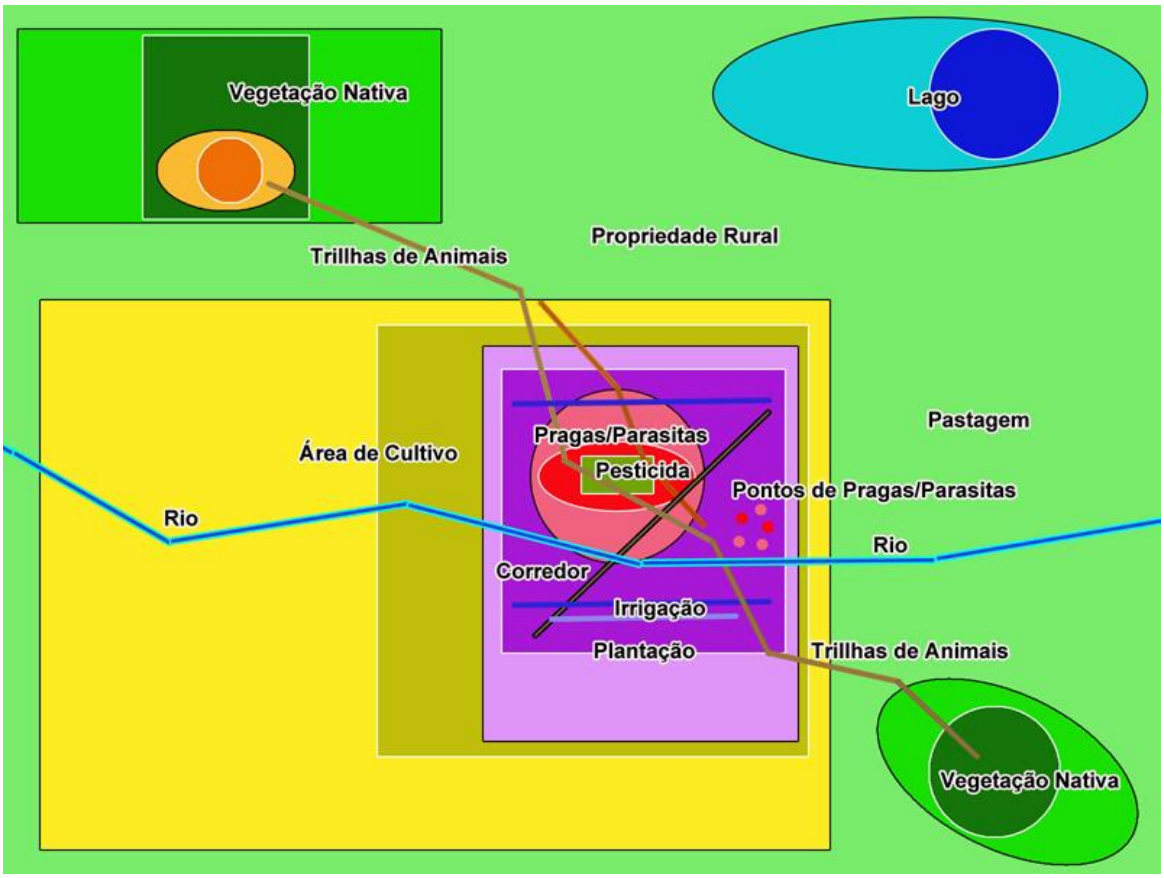

Figura 5.1 Distribuição dos Fenômenos dentro de uma Propriedade Rural.

A partir da Figura 5.1 percebe-se que um determinado fenômeno pode ser gerado dentro (estar contido) de outro fenômeno gerado anteriormente. Por exemplo, para gerar uma região de praga/parasita, primeiramente deve-se gerar uma propriedade rural, uma área de cultivo e uma região de plantação, respectivamente, sendo que a área de praga/parasita está contida na região de plantação, a região de plantação está contida na área de cultivo, a área de cultivo está contida na propriedade rural e por fim, a propriedade rural está contida em um determinado município. Desse modo, tem-se então uma hierarquia de geração de fenômenos. A Figura 5.2 ilustra a hierarquia de geração dos fenômenos rurais considerados neste estudo de caso. Esta hierarquia define a ordem e precedência da geração dos fenômenos rurais. Os fenômenos em branco representam os

fenômenos exatos (*crisp*), enquanto os fenômenos em cinza representam os fenômenos vagos descritos na Tabela 5.1.

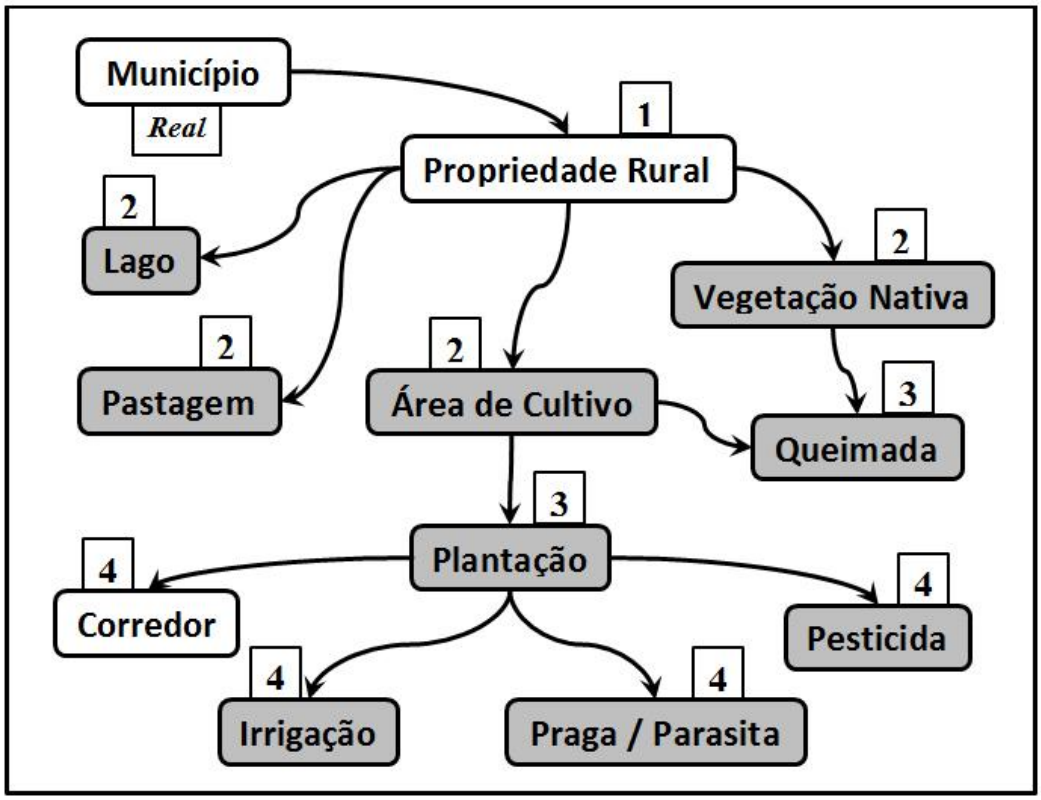

Figura 5.2 Hierarquia de Geração de Fenômenos Rurais.

Alguns dos fenômenos rurais considerados neste estudo de caso não fazem parte da hierarquia de geração, pois não respeitam as fronteiras de um determinado fenômeno gerado. Como exemplo, podemos citar o fenômeno “Rio”, pois um determinado rio não necessariamente está contido em apenas uma propriedade rural. Chamamos estes fenômenos de “Fenômenos Independentes”. Os fenômenos independentes podem ser gerados sem respeitar as fronteiras de uma determinada propriedade rural ou até mesmo, um determinado município. A Figura 5.3 ilustra os “fenômenos independentes”.

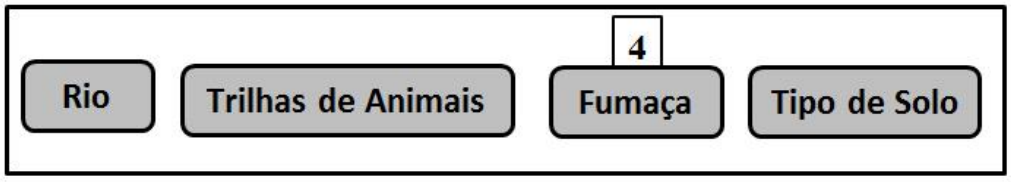

Figura 5.3 Fenômenos Rurais Independentes.

5.1.1 Ferramenta de Geração de Fenômenos Rurais Vagos

A ferramenta de geração de fenômenos rurais vagos denominada "*GeneratorVaguePhenomena*" gera objetos espaciais vagos sintéticos que representam diversos fenômenos existentes em áreas rurais. Esta ferramenta foi desenvolvida na linguagem Java, utilizando a interface de desenvolvimento *Netbeans*, utilizando os algoritmos descritos nas seções 4.4 e 4.5 para a geração dos fenômenos espaciais vagos. Essa ferramenta pode ser obtida no endereço eletrônico http://gbd.dc.ufscar.br/vaguedatageneration.

Na Tela Inicial da ferramenta o usuário deverá clicar no botão correspondente à forma de como se deseja gerar os fenômenos rurais (botões "*Auto Generate Phenomena*" e "*Manually Generate Phenomena*") ou clicar no botão correspondente ao fenômeno específico que deseja gerar (demais botões). A Figura 5.4 ilustra a tela inicial da ferramenta.

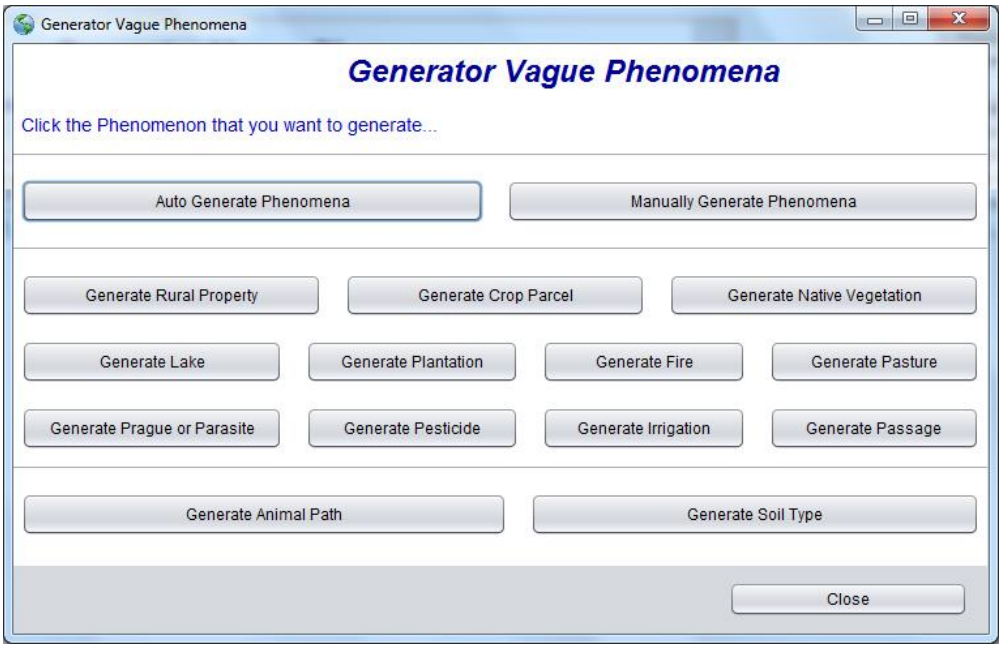

Figura 5.4 Tela Inicial da Ferramenta *GeneratorVaguePhenomena*

Na tela de geração manual de fenômenos são gerados todos os fenômenos considerados no estudo de caso. Para a geração dos fenômenos o usuário deverá selecionar alguns parâmetros, tais como, modelo, tamanho, formato, quantidade, entre outros, dependendo do fenômeno rural, do modelo de dados (por ex.: VASA) e do tipo de dado definido para cada fenômeno (por ex.: pontos vagos).

Primeiramente o usuário seleciona o “município base” onde serão geradas as propriedades rurais. A partir da geração das propriedades rurais são gerados os demais fenômenos, distribuídos dentro desta propriedade rural. Caso o usuário não queira gerar um determinado fenômeno, basta deixar a caixa de texto “Quantidade” em Branco ou colocar o valor “0” no mesmo. A Figura 5.5 ilustra a tela de Geração Manual de Fenômenos.

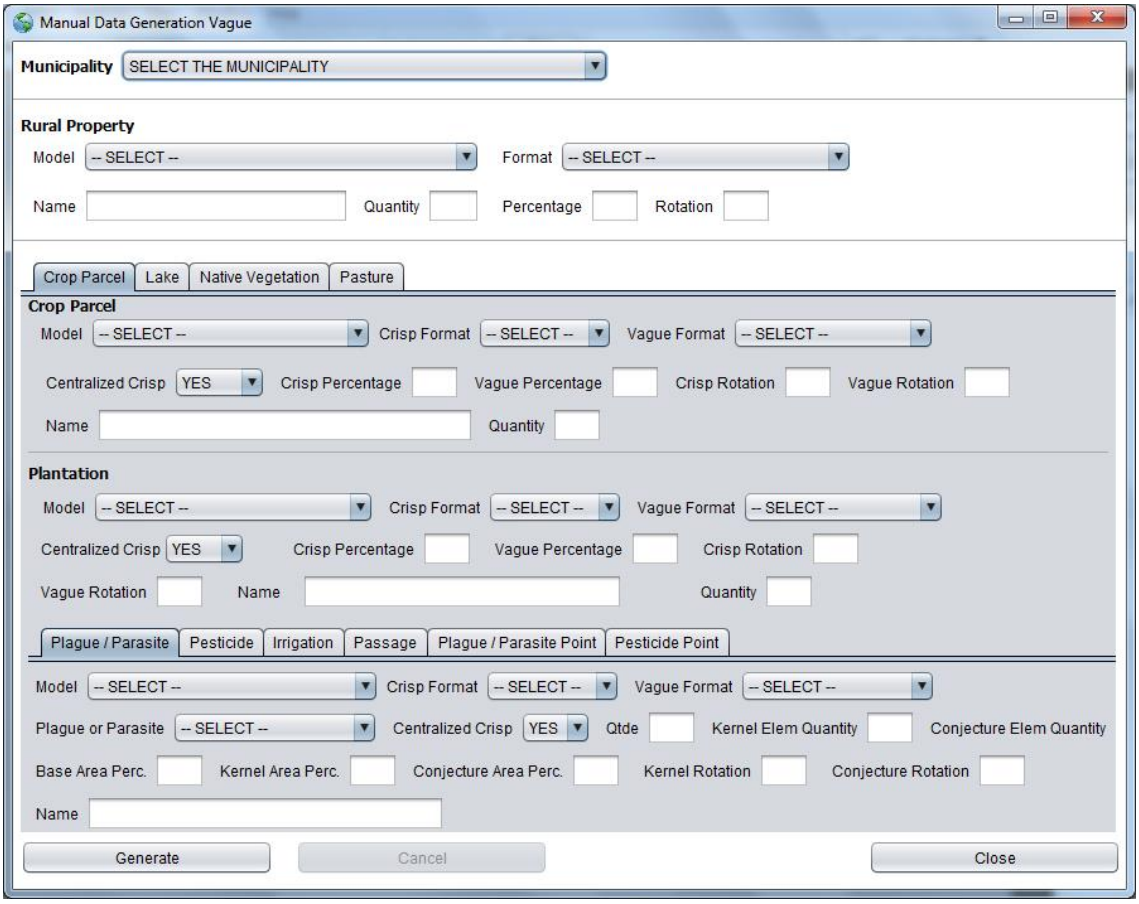

Figura 5.5 Tela de Geração Manual de Fenômenos.

Na tela de geração automática de fenômenos são gerados todos os fenômenos automaticamente a partir de um fator de escala que vai de “0” a “10”.

Nesta tela o usuário seleciona o “município base” e o fator de escala. Após selecionar o município base e o fator de escala o usuário marca ou desmarca os fenômenos que se deseja gerar. Caso o usuário queira gerar um determinado fenômeno, basta marcá-lo e vice e versa. O usuário ainda tem a opção de marcar e desmarcar todos os fenômenos de uma só vez. A Figura 5.6 ilustra a tela de Geração Automática de Fenômenos.

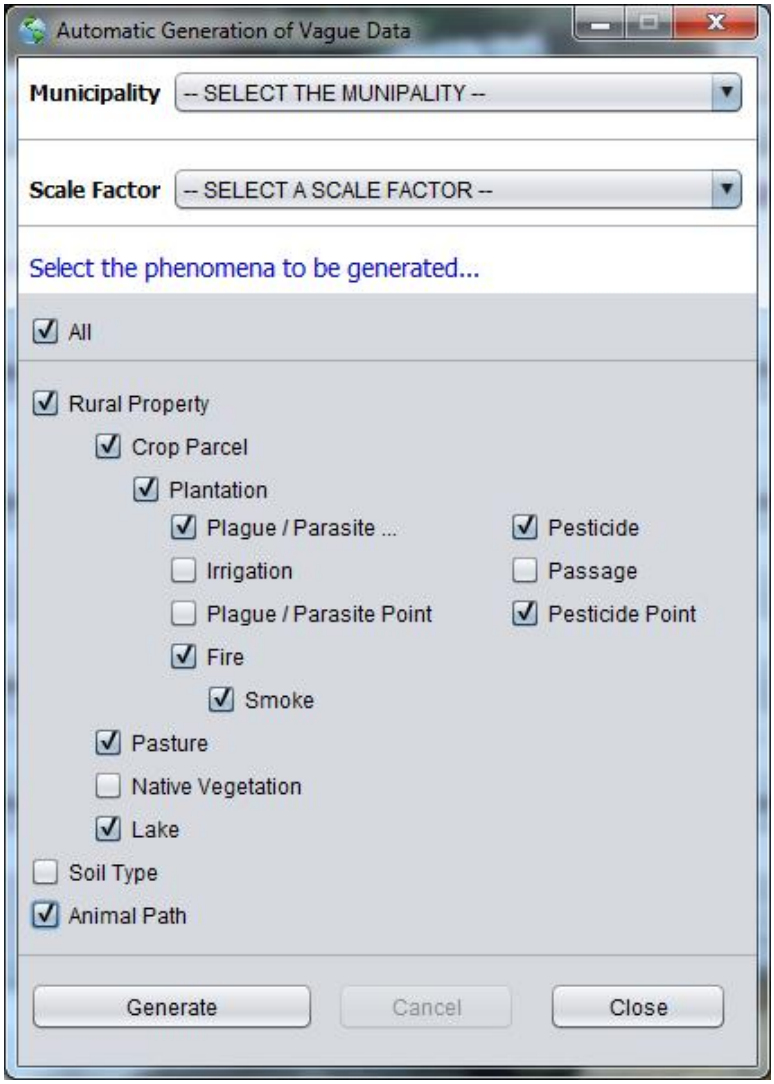

Figura 5.6 Tela de Geração Automática de Fenômenos.

Como vimos na tela inicial (Figura 5.4), o usuário tem a opção de gerar um determinado fenômeno individualmente. Por exemplo, na tela de geração de plantações o usuário seleciona a Área de Cultivo onde deseja gerar uma ou mais plantações. Esta área de cultivo selecionada será a “região base” para a geração das plantações, ou seja, todas as plantações serão geradas dentro dessa área de cultivo. Por fim, o usuário preenche outros parâmetros, tais como, modelo de dados, tipo de dado espacial vago, formato, porcentagem e rotação (giro) da parte exata e da parte vaga, além da quantidade e do nome da plantação. A Figura 5.7 ilustra a tela de geração de plantações.

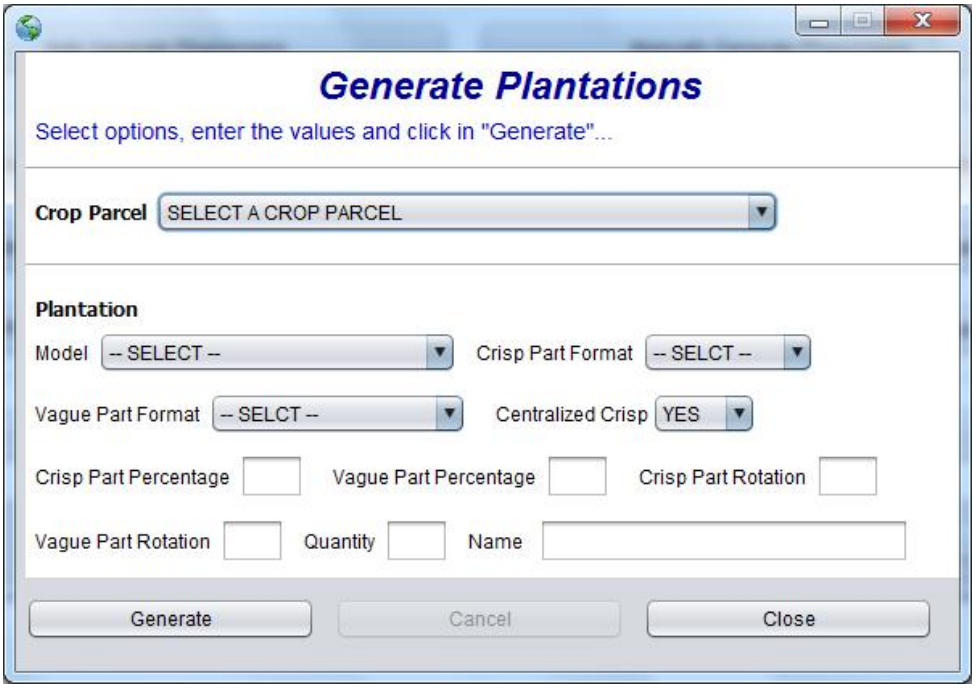

Figura 5.7 Tela de Geração de Plantações.

Devido a grande quantidade de fenômenos considerados em nosso estudo de caso, foi apresentada nessa dissertação apenas uma tela de geração individual de fenômeno vago – a tela de geração de plantações.

Vale ressaltar que foi desenvolvida uma tela específica para cada fenômeno considerado no estudo de caso e que as demais telas possuem o mesmo formato da tela de Geração de Plantações ilustrada na Figura 5.7.

5.1.2 Modelagem do Esquema do Banco de Dados do Estudo de Caso

Para armazenar os fenômenos rurais vagos sintéticos gerados pela ferramenta *GeneratorVaguePhenomena*, foi modelado um banco de dados relacional contendo tabelas, colunas, chaves e relacionamentos entre as tabelas. Esse banco de dados foi implementado no SGBD *PostgreSQL*/*PostGIS*.

A Figura 5.8 ilustra e descreve o esquema do banco de dados modelado para o armazenamento dos fenômenos rurais vagos gerados pela ferramenta *GeneratorVaguePhenomena*.

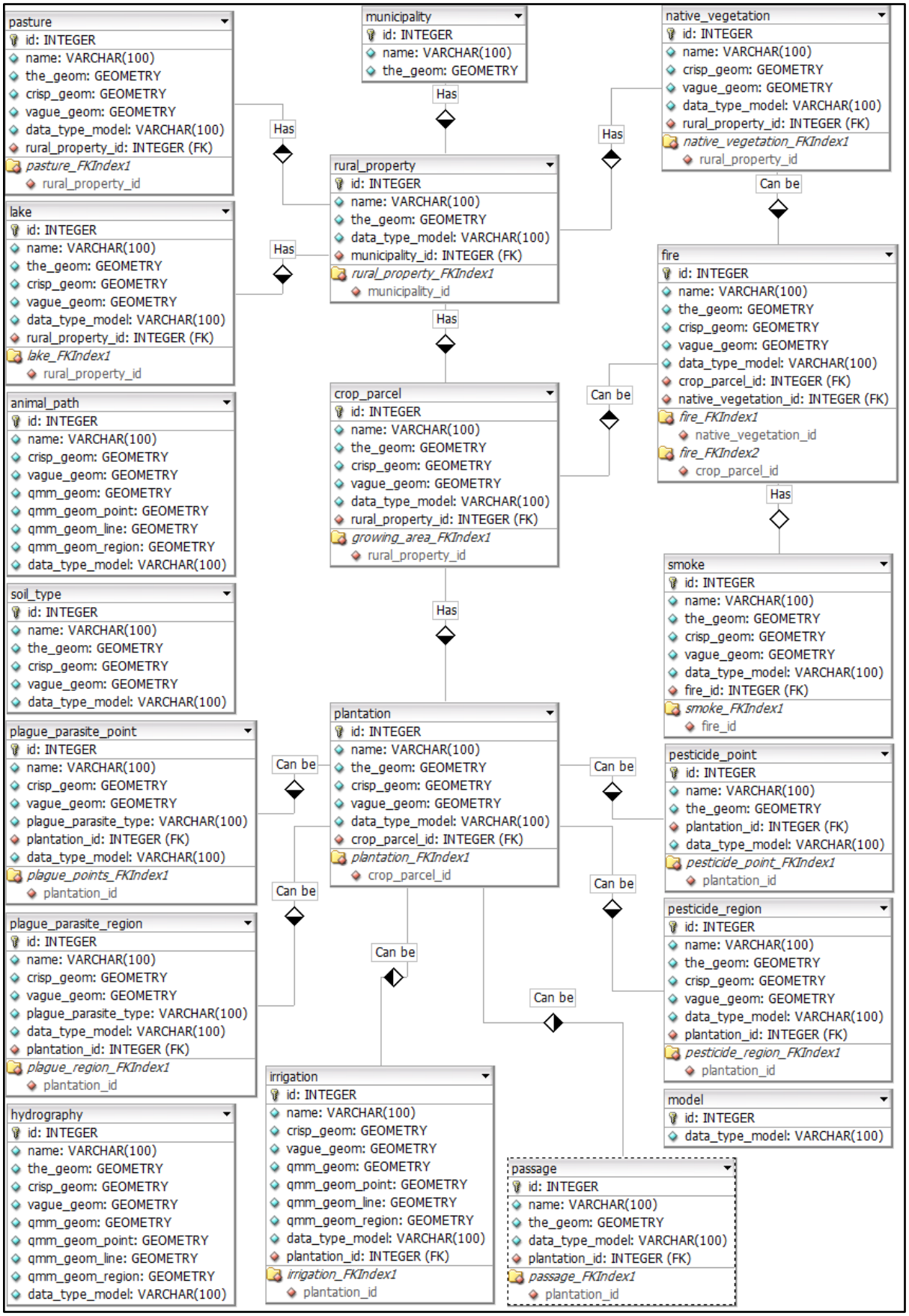

Figura 5.8 Modelagem do Esquema do Banco de Dados da Ferramenta *GeneratorVaguePhenomena*.

O banco de dados da ferramenta *GeneratorVaguePhenomena* foi modelado considerando os três modelos de dados espaciais vagos abordados nesse trabalho (*Egg-Yolk*, QMM e VASA), ou seja, é possível gerar e armazenar fenômenos rurais que representem os três modelos e todos os tipos de dados espaciais vagos de cada modelo, de acordo com a Tabela 5.1. O esquema do banco de dados é ilustrado na Figura 5.8.

Além da modelagem do banco de dados da ferramenta, considerando os três modelos de dados espaciais vagos, foram modelados individualmente cada modelo de dados espaciais vagos considerados neste trabalho.

A Figura 5.9 ilustra e descreve a modelagem do esquema do banco de dados da ferramenta *GeneratorVaguePhenomena*, considerando apenas o modelo *Egg-Yolk*. Como o modelo *Egg-Yolk* aborda apenas regiões vagas simples, não é possível representar todos os fenômenos do estudo de caso. Por exemplo, trilhas de animais e pontos de pragas/parasitas não são representadas, pois são formados por linhas vagas e por pontos vagos, respectivamente.

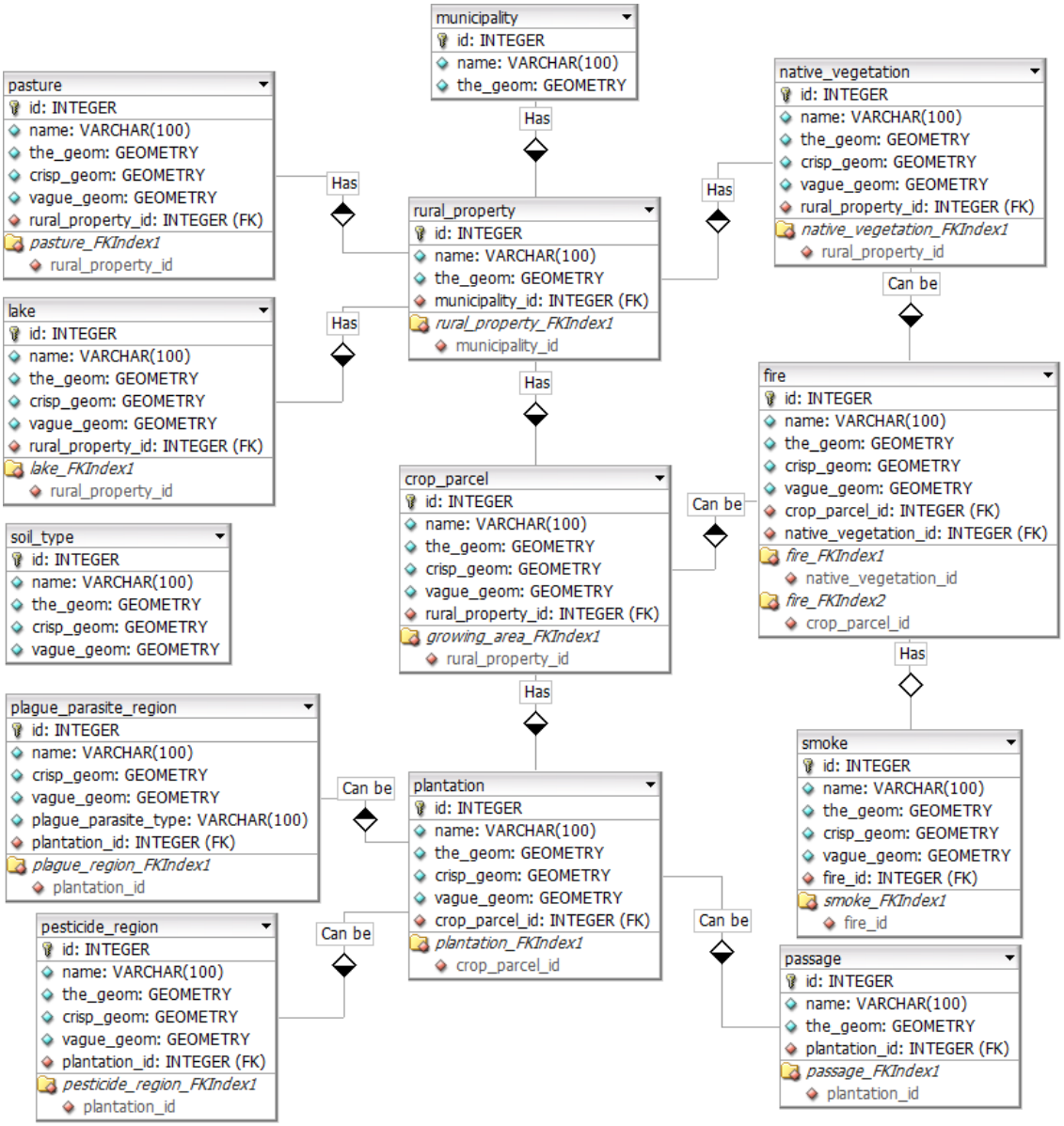

Figura 5.9 Modelagem do Esquema do Banco de Dados da Ferramenta *GeneratorVaguePhenomena*, considerando apenas o Modelo *Egg-Yolk*.

Considerando apenas o modelo QMM foi possível representar todos os fenômenos da ferramenta *GeneratorVaguePhenomena*. Esse modelo aborda tanto as regiões vagas, quanto os pontos vagos e linhas vagas. No entanto, os fenômenos rurais representados por pontos (ex.: pragas/parasitas e pesticidas) não são precisamente representados, pois o modelo QMM representa os pontos vagos como uma região exata. A Figura 5.10 ilustra e descreve a modelagem do esquema do banco de dados da ferramenta do estudo de caso considerando apenas o modelo QMM.

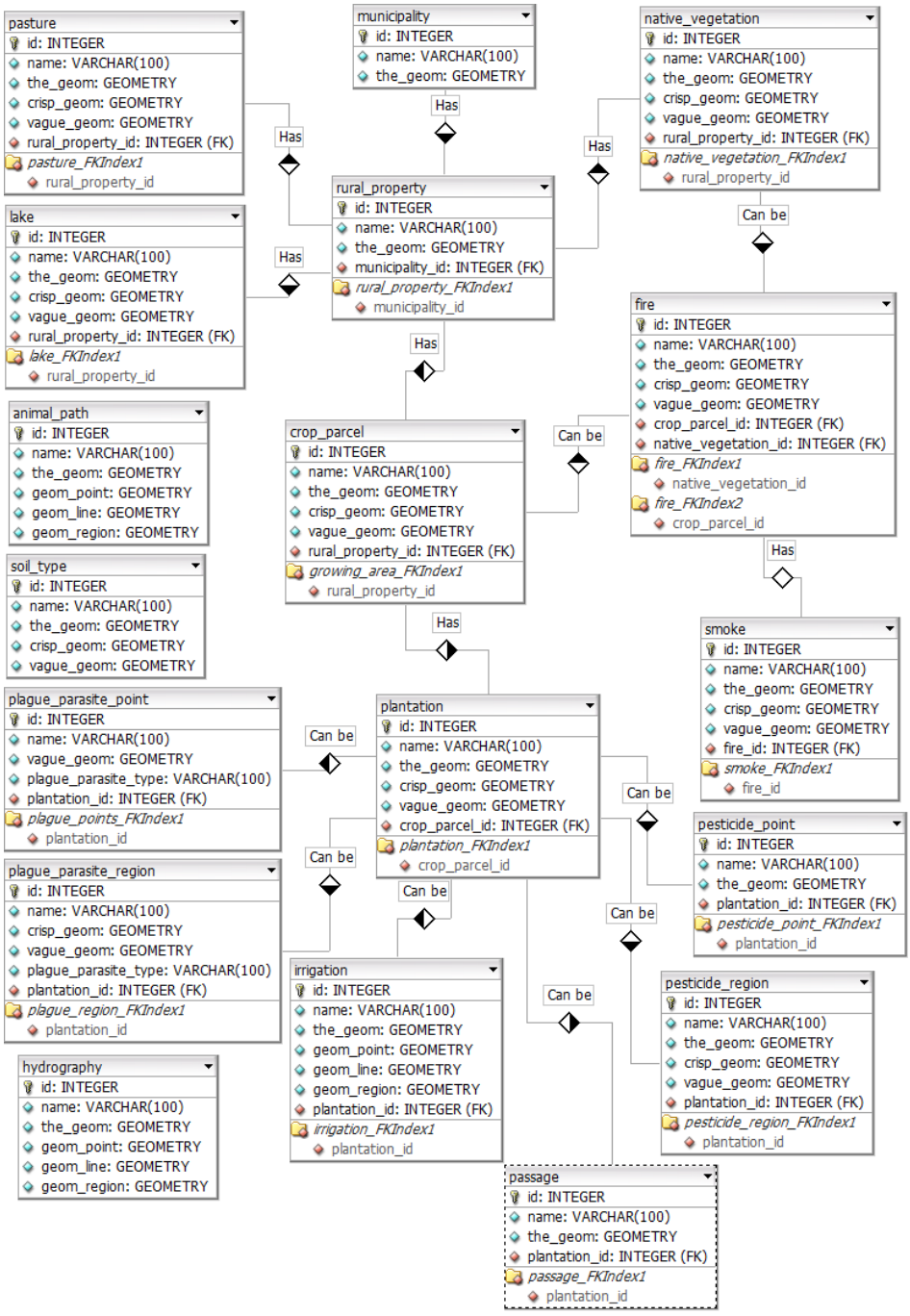

Figura 5.10 Modelagem do Esquema do Banco de Dados da Ferramenta *GeneratorVaguePhenomena*, considerando apenas o Modelo QMM.

Por fim, o esquema do banco de dados da ferramenta foi modelado considerando apenas características da VASA. A partir das características da VASA todos os fenômenos do estudo de caso foram representados. A principal diferença da modelagem utilizando as definições da VASA para a modelagem utilizando as definições do modelo QMM consiste na representação da parte hipotética (parte vaga) das linhas vagas, mais precisamente, das trilhas de animais, pois na VASA a parte hipotética das linhas vagas é representada por linhas exatas e no modelo

QMM é representado por uma região exata. A Figura 5.11 ilustra e descreve a modelagem do esquema do banco de dados da ferramenta do estudo de caso, considerando apenas a VASA.

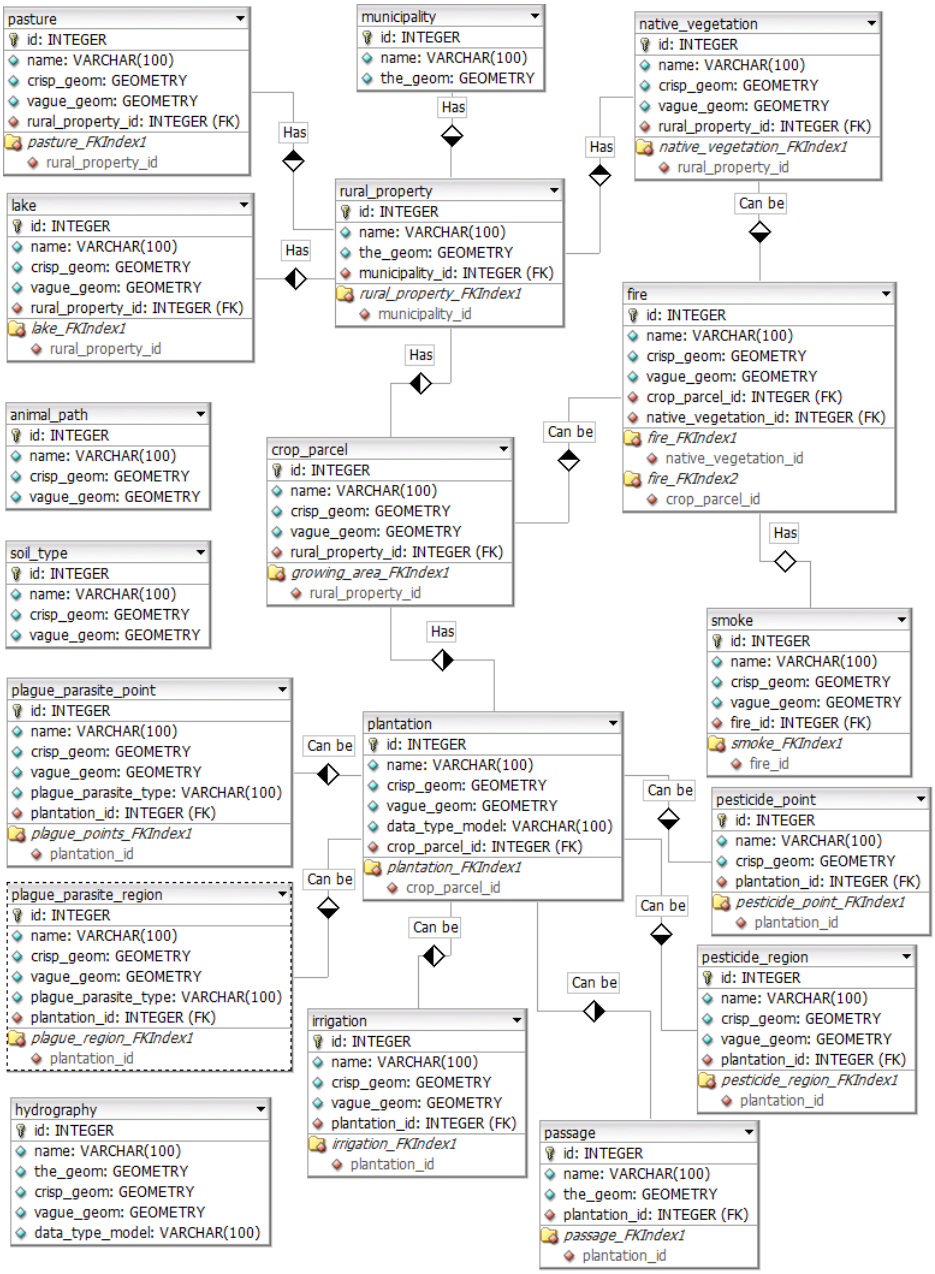

Figura 5.11 Modelagem do Esquema do Banco de Dados da Ferramenta *GeneratorVaguePhenomena*, considerando apenas na VASA.

A modelagem separada para cada modelo pode ser interessante, pois cada modelo possui suas próprias características e operações definidas sobre os seus tipos de dados. No entanto, ao modelar uma aplicação prática percebe-se as limitações de cada modelo.

Apesar dos modelos de dados espaciais vagos possuírem características específicas, uma aplicação real em geral pode necessitar de características de diversos modelos, bem como seus tipos de dados espaciais vagos. Com a utilização de diversos modelos tem-se maior flexibilidade para modelar os objetos do mundo real, bem como o tipo de dado mais apropriado e mais próximo da realidade. Em contrapartida, não existem operadores topológicos que representem os relacionamentos entre tipos de dados de diferentes modelos.

5.2 Resultado das Hipóteses e Validação do Trabalho

O desenvolvimento deste trabalho de pesquisa se baseou em três **hipóteses**, apresentadas na seção 1.4 do capítulo 1.

Com o desenvolvimento dos algoritmos de geração de dados espaciais vagos sintéticos e o desenvolvimento da ferramenta de geração de fenômenos rurais vagos (nosso estudo de caso) concluímos que todas as hipóteses foram confirmadas com sucesso. A seguir são apresentadas essas três hipóteses, juntamente com a resposta e a justificativa de cada uma delas:

1. "*É possível gerar dados espaciais vagos sintéticos considerando e controlando na geração dos dados características tais como formato, tamanho, volume, complexidade, localização e distribuição espacial dos dados espaciais vagos.*" **Resposta:** Sim. Os dados espaciais vagos sintéticos foram gerados a partir de uma região base e distribuídos dentro desta mesma região. Na geração foram considerados todos os parâmetros definidos na primeira hipótese.
2. "*É possível gerar dados espaciais vagos sintéticos a partir das definições dos modelos de dados espaciais vagos Egg-Yolk, QMM e VASA.*" **Resposta:** Sim. Foram desenvolvidos algoritmos para cada tipo de dado espacial vago dos três modelos de dados espaciais vagos considerados neste projeto (modelos *Egg-Yolk*, QMM e VASA)
3. "*É possível posicionar os dados espaciais vagos sintéticos gerados em uma determinada localização do espaço.*" **Resposta:** Sim. Os dados espaciais vagos gerados podem ser posicionados no centro da região

base. Ademais, a parte exata ou extensão mínima de uma região vaga simples poderá ser posicionada no centro da parte hipotética ou extensão máxima dessa região vaga.

Por fim, a **validação** da geração de dados espaciais vagos sintéticos foi realizada para cada modelo, onde os dados gerados foram validados de duas maneiras:

1. *Verificar se os dados gerados atenderam às propriedades e restrições do modelo no qual é baseado:* Foram executados os algoritmos de geração de dados vagos com diferentes combinações de parâmetros de entrada e os resultados retornaram dados espaciais vagos que atenderam as características e particularidades esperadas para cada tipo de dado de cada modelo de dados espaciais vagos;
2. *Verificar o quão útil foram/serão os dados espaciais vagos gerados para os usuários finais:* Para esta validação foi utilizada a ferramenta de geração de fenômenos rurais vagos (estudo de caso). Foram gerados diversos fenômenos rurais vagos que representaram pontos vagos, linhas vagas e regiões vagas referentes aos três modelos considerados neste projeto. Os resultados dessas operações mostraram que os fenômenos rurais vagos gerados atendiam os requisitos e características referentes ao modelo de dados e ao tipo de dado espacial vago corresponde e estavam posicionados dentro do fenômeno rural vago definido na hierarquia de geração dos fenômenos rurais vagos. Ademais, os dados gerados estão sendo atualmente usados nos seguintes trabalhos. O primeiro trabalho, em nível de mestrado e sendo desenvolvido pelo aluno Anderson Chaves Carniel, visa implementar um tipo abstrato de dados para a VASA usando o sistema gerenciador de banco de dados PostgreSQL/PostGIS. O segundo trabalho, em nível de doutorado e sendo desenvolvido pelo aluno Thiago Luís Lopes Siqueira, possui como alguns de seus objetivos a proposta de esquemas lógicos de *data warehouses* com dados espaciais vagos e a proposta de um índice para *data warehouses* com dados espaciais vagos.

Capítulo 6

CONCLUSÕES

*Este capítulo apresenta as **conclusões** deste trabalho de pesquisa em nível de mestrado. Mais especificamente, este capítulo destaca as **contribuições** deste trabalho e as suas **limitações**, além de algumas **lições aprendidas** e de possíveis **trabalhos futuros**.*

6.1 Contribuições e Limitações

Neste tralhado de pesquisa em nível de mestrado foram desenvolvidos algoritmos para a geração de dados espaciais vagos sintéticos a partir dos modelos de dados espaciais vagos *Egg-Yolk*, QMM e VASA. A partir desses algoritmos é possível criar uma base de dados espacial vaga, como por exemplo, um *Data Warehouse* Espacial Vago (DWEV). Esta base de dados criada poderá ser utilizada para a realização de pesquisas, tais como, testar o desempenho de índices, testar técnicas de processamento de consultas de *Data Warehouses* que armazenam dados espaciais vagos, entre outras pesquisas.

A principal contribuição deste trabalho de pesquisa foi o desenvolvimento de algoritmos para a geração de dados espaciais vagos sintéticos baseados nos modelos de dados espaciais vagos, uma vez que não foi encontrado na literatura nenhum trabalho que gerasse esse tipo de dado. Os algoritmos desenvolvidos foram implementados na linguagem de programação *PL/pgSQL* e os dados espaciais vagos sintéticos gerados foram armazenados no banco de dados relacional *PostgreSQL*. No entanto, tanto os algoritmos, quanto a estrutura de armazenamento

são genéricos, podendo ser facilmente implementados em outras linguagens de programação e armazenados em outras estruturas de banco de dados que possuam funcionalidades similares à linguagem de programação *PL/pgSQL* e ao SGBD *PostgreSQL* com a extensão *PostGIS*.

Além disso, foi desenvolvido um estudo de caso para mostrar a aplicabilidade da proposta de geração de dados espaciais vagos. A partir da ferramenta e modelagem do banco de dados do estudo de caso percebe-se a aplicabilidade e relevância dos dados espaciais vagos para a representação de diversos objetos do mundo real.

6.2 Lições aprendidas

Para o desenvolvimento deste trabalho foi necessário um grande estudo teórico para o entendimento das características específicas de cada modelo e de cada tipo de dado de um determinado modelo. A maior dificuldade enfrentada neste trabalho foi à falta de implementações práticas referentes aos modelos, uma vez que até o momento, nenhuma implementação para esses modelos foi encontrada na literatura.

Neste trabalho aprendi a realizar um levantamento bibliográfico, extrair as principais informações das bibliografias, além do desenvolvimento dos algoritmos de geração de geometrias, geração de dados espaciais vagos sintéticos e o desenvolvimento da modelagem e da ferramenta de geração de fenômenos rurais (estudo de caso).

A principal lição que este trabalho de pesquisa me passa é a importância em se buscar novos conhecimentos, trabalhar em parceria com o orientador e com os colegas do grupo de pesquisa.

Por fim, neste trabalho percebi o quanto os modelos de dados espaciais vagos são importantes para uma melhor representação dos objetos do mundo real, pois proporciona uma representação mais próxima a realidade que encontramos. Percebi também que existe uma grande lacuna na literatura a ser preenchida em relação à representação e geração de dados espaciais vagos.

6.3 Trabalhos Futuros

A geração de dados espaciais vagos é uma área pouco explorada na literatura. Como trabalhos futuros, destacamos a necessidade de pesquisas sobre os seguintes assuntos:

1. *Inclusão de Relacionamentos Topológicos na Geração dos Dados Espaciais Vagos Sintéticos*: localização dos dados espaciais vagos gerados no espaço e satisfação de relacionamentos, como por exemplo, o usuário pode querer uma sobreposição de 10% entre a parte vaga de um objeto e a parte vaga de outro objeto ou que 50% dos objetos vagos satisfaçam o relacionamento espacial de interseção entre os seus núcleos;
2. *Geração de Dados Espaço-Temporais Vagos Sintéticos*: gerar dados espaciais vagos sintéticos e movimentá-los periodicamente. A geração destes dados poderão se basear nos algoritmos e funções desenvolvidas neste projeto, acrescentando os aspectos temporais para os dados gerados;
3. *Geração de Dados Espaciais Vagos a partir de dados Reais*: gerar dados espaciais vagos que representem objetos do mundo real, independentemente do domínio em que se encontra este objeto. A geração destes dados pode se basear nos algoritmos e funções desenvolvidas neste projeto, adaptando-os para a geração de dados espaciais vagos a partir de dados espaciais exatos reais;
4. *Criar um Benchmark para DW Espacial Vago*: utilizando os algoritmos de geração de dados espaciais vagos pode-se propor um esquema de um DW Espacial Vago e adicionar uma carga de trabalho para criar um *Benchmark* para este DW; e
5. *Desenvolvimento de uma Ferramenta Gráfica para a visualização dos Dados Espaciais Vagos Gerados neste projeto*.

REFERÊNCIAS

BEJAOUI. L.; PINET, F.; BÉDARD, Y.; SCHNEIDER, M. Qualified topological relations between spatial objects with possible vague shape. **International Journal of Geographical Information Science**, v. 23, n. 7, p. 877-921, 2009.

BEJAOUI, L.; PINET, F.; BÉDARD, Y.; SCHNEIDER, M. OCL for formal modelling of topological constraints involving regions with broad boundaries. **GeoInformatica**, v. 14, n. 3, p. 353-378, 2010.

CÂMARA, G.; CASANOVA, M. A.; HEMERLY, A. S.; MAGALHÃES, G. C.; MEDEIROS, C. M. B. **Anatomia de Sistemas de Informação Geográfica**. 10ª. Escola de Computação, Campinas, 1996. 193 p.

CIFERRI, R. R. **Análise da influência do fator distribuição espacial dos dados no desempenho de métodos de acesso multidimensionais**. 2002. 246f. Tese (Doutorado em Ciência da Computação) – Centro de Informática, Universidade Federal de Pernambuco, Recife, 2002. p. 9-14.

COHN. A. G.; BENNETT, B.; GOODAY, J.; COTTS, N. M. Qualitative Spatial Representation and Reasoning with the Region Connection Calculus. **GeoInformatica**, v. 1, n. 3, p. 275-316, 1997.

COHN, A. G.; GOTTS, N. M. The 'egg-yolk' representation of regions with indeterminate boundaries. In: BURROUGH, P.A.; FRANK, A.U. (Eds.), **Geographic Objects with Indeterminate Boundaries**. London: GISDATA, 1996. p. 171-187.

DAVIS J. C.; QUEIROZ, G. R. Modelagem de dados geográficos. In: CÂMARA, G., CASANOVA, M. A., DAVIS J. C., VINHAS, L., QUEIROZ, G. R. **Banco de Dados Geográficos.** Curitiba, Editora MundoGEO, 2005. p. 43-83. Disponível em: < http://www.dpi.inpe.br/livros/bdados/capitulos.html> Acesso em: 10 de Jan. de 2012.

DILO, A. **Representation of and reasoning with vagueness in spatial information: a system for handling vague objects**. Tese (Doutorado). Wageningen University, 2006.

DILO, A.; BY, R. D.; STEIN, A. A system of types and operators for handling vague spatial objects. **International Journal of Geographical Information Science**, 2007.

DUBOIS, D.; PRADE, H. Fuzzy sets and probability: misunderstandings, bridges and gaps. In: **Fuzzy Systems**. [S.l.: s.n.], 1993. v. 2, p. 1059–1068.

EGENHOFER, M. J.; FRANZOSA, R. D. Point-set topological spatial relations. **International Journal of Geographical Information Science**, v. 5, n. 2, p. 161-174, 1991.

EGENHOFER, M.; HERRING, J. **Categorizing Binary Topological Relationships Between Regions, Lines, and Points in Geographic Databases**. Orono: Department of Surveying Engineering, University of Maine, 1991. 28 p.

GHAZEL, M.; FREEMAN, G.H.; VRSCAY, E.R.. **An effective hybrid fractal-wavelet image coder using quadtree partitioning and pruning**. IEEE, CCECE Halifax, NS, Canada, 2000.

LEHMANN, F.; COHN, A. G. The EGG/YOLK reliability hierarchy: semantic data integration using sorts with prototypes. **Proceedings of the third international conference on Information and knowledge management**. New York, NY, USA: ACM, 1994. p. 272-279.

NASCIMENTO, S. M.; TSURUDA, R. M.; SIQUEIRA, T. L. L.; TIMES, V. C.; CIFERRI, R. R.; CIFERRI, C. D. A. **The Spatial Star Schema Benchmark**. Campos do Jordão, SP, Brazil: GeoInfo, 2011. p. 73-84.

PAULY, A.; SCHNEIDER, M. Vague Spatial Data Types, Set Operations, and Predicates. In: BENCZÚR, A.; DEMETROVICS, J.; GOTTLOB, G. (Org.). **Advances in Databases and Information Systems**. Berlin / Heidelberg: Springer, 2004. p. 379-392.

PAULY, A.; SCHNEIDER, M. Vague Spatial Data Types. In: **Encyclopedia of GIS**. Springer US, 2008. p. 1213-1217.

PAULY, A.; SCHNEIDER, M. VASA: An algebra for vague spatial data in databases. **Information Systems**, v. 35, n. 1, p. 111-138, 2010.

RANDELL, D. A.; COHN, A. G. Modelling Topological and Metrical Properties in Physical Processes. In: LEVESQUE, H.; BRACHMANN, R.; REITER, R. (Eds.), **Principles of Knowledge Representation and Reasoning**. Los Altos, CA, USA: Morgan Kaufmann, 1989. p. 55-66.

RANDELL, D. A.; CUI, Z.; COHN, A. G. A Spatial Logic based on Regions and Connection**.** 3rd International Conference on Principles of Knowledge Representation and Reasoning. **Proceedings…** San Mateo, CA, USA: Morgan Kaufmann, 1992. p.165-176.

RODRÍGUEZ, A. Inconsistency Issues in Spatial Databases. In: BERTOSSI, L.; HUNTER, A.; SCHAUB, T. (Org.). **Inconsistency Tolerance**. Berlin / Heidelberg: Springer, 2005. p. 237-269.

SIQUEIRA, T. L. L.; MATEUS, R. C.; CIFERRI, R. R.; TIMES, V. C.; CIFERRI, C. D. A. Querying Vague Spatial Information in Geographic Data Warehouses. In: GEERTMAN, S.; REINHARDT, W.; TOPPEN, F. (Eds.), **Advancing Geoinformation Science for a Changing World**. Berlin / Heidelberg: Springer, v. 1, n. 18, 2011, p. 379-397.

SIQUEIRA, T. L. L.; CIFERRI, R. R.; TIMES, V. C.; CIFERRI, C. D. A. Benchmarking spatial data warehouses. **Proceedings of the 12th international conference on**

Data warehousing and knowledge discovery. Berlin / Heidelberg: Springer-Verlag, 2010 (DaWaK'10). p. 40-51.

THEODORIDIS, Y.; SILVA, J. R. O.; NASCIMENTO, M. A. On the generation of spatiotemporal datasets. **Proceedings of the 6th International Symposium on Advances in Spatial Databases**. London, UK, UK: Springer-Verlag, 1999 (SSD'99). p. 147–164.

ZADEH, L. Fuzzy sets. **Information and Control**, v. 8, n. 3, p. 338–353, 1965. ISSN 00199958. Disponível em: <http://www.sciencedirect.com/science/article/pii/S001999586590241X>.