Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

GeoDrill: Uso de SQL para integração de Fontes de Dados Espaciais Heterogêneas com ou sem Esquema

José Amilton Moura Acioli Filho

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande – Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Sistemas de Computação

Cláudio de Souza Baptista, Ph.D.
(Orientador)

Cláudio Elízio Calazans Campelo, Ph.D.
(Orientador)

Campina Grande, Paraíba, Brasil

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

A181g Acioli Filho, José Amilton Moura.

GeoDrill: uso de SQL para integração de fontes de dados espaciais heterogêneas com ou sem esquema / José Amilton Moura Acioli Filho. – Campina Grande, 2016.

108f. : il. color.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2016.

"Orientação: Prof. Ph.D. Claúdio de Souza Baptista, Prof. Ph.D. Claudio Elízio Calazans Campelo".

Referências.

1. Ciência da Computação. 2. Dados Espaciais. 3. Dados Espaciais - Integração. 4. GeoDrill (Integração de Dados). 5. Linguagem SQL. I. Baptista, Claúdio de Souza. II. Campelo, Claudio Elízio Calazans. III. Universidade Federal de Campina Grande, Campina Grande (PB). IV. Título.

CDU 004(043)

"GEODRILL: USO DE SQL PARA INTEGRAÇÃO DE FONTES DE DADOS ESPACIAIS HETEROGÊNEAS COM OU SEM ESQUEMA"

JOSE AMILTON MOURA ACIOLI FILHO

DISSERTAÇÃO APROVADA EM 02/09/2016

CLÁUDIO DE SOUZA BAPTISTA, PhD., UFCG
Orientador(a)

CLAUDIO ELIZIO CALAZANS CAMPELO, PhD., UFCG
Orientador(a)

CARLOS EDUARDO SANTOS PIRES, Dr., UFCG
Examinador(a)

HUGO FEITOSA DE FIGUEIRÊDO, Dr., IFPB
Examinador(a)

CAMPINA GRANDE - PB

Resumo

Com a evolução da web e dos sistemas de informação, as organizações têm obtido dados dos mais diversos formatos, estruturas e tipos, podendo-se destacar os espaciais. Devido aos dados apresentarem características distintas, estes acabam sendo mantidos em fontes de dados heterogêneas, sendo assim necessário investir cada vez mais em soluções que possam integrar e analisar estes dados de diferentes fontes. Algumas destas soluções conseguem analisar o componente espacial dos dados, no entanto, essa análise dos dados espaciais é limitada pelo tipo de dados ou funções espaciais suportadas. Neste trabalho, é abordado o problema da integração de dados espaciais de fontes de dados heterogêneas, com ou sem esquema, utilizando linguagem SQL. Este é um problema em aberto na área de integração de dados espaciais, pois as soluções existentes apresentam inúmeras limitações, a exemplo da linguagem de consulta utilizada, os meios para acesso a dados, as tecnologias que podem ser integradas, as funções disponibilizadas e os tipos de dados espaciais suportados. Visando solucionar esse problema, desenvolveu-se a solução GeoDrill, uma extensão do Apache Drill que dá suporte a todas as funções espaciais padronizadas pela OGC (Open Geospatial Consortium), através da linguagem SQL, podendo realizar consultas em dados com ou sem esquema. Para validar a capacidade de integração dos dados no GeoDrill, foi desenvolvido um experimento para analisar as funcionalidades e o desempenho do mesmo. A solução GeoDrill foi capaz de realizar a integração dos dados espaciais de fontes heterogêneas, apresentando-se como uma alternativa para a resolução de parte das limitações existentes na área.

Palavras chaves: Dados Espaciais, Integração de Dados Espaciais, Dataspace, SQL.

Abstract

With the evolution of the web and information systems, organizations have obtained data of various formats, structures and types, specially the spatial one. Due to different characteristics presented in data, such data have been stored in heterogeneous data sources. Therefore, it is needed to increasingly invest in solutions that can integrate and analyze these data from different sources. Some of these solutions can analyze the spatial component of data; however, this analysis of spatial data is limited either by the data type or spatial functions supported. In this work, the problem of spatial data integration from heterogeneous data sources is addressed, either with or without using schemas, using SQL language. This is an open issue in the area of spatial data integration, since existing solutions present many limitations, such as the query language used, the ways to access data, the technologies that can be integrated, the available functions set and the spatial data types supported. Aiming at solving this problem, the GeoDrill solution was developed, which is an extension of the Apache Drill that supports all standard spatial functions provided by the OGC (Open Geospatial Consortium) through the SQL language. The GeoDrill can perform queries on data with or without schema. In order to validate the capacity of GeoDrill to integrate data, an experiment was conducted to analyze its functionalities and performance. The obtained results indicate the GeoDrill solution is able to integrate spatial data from heterogeneous data sources. Hence, it appears to be a suitable alternative for solving part of the existing limitations in this research field.

Keywords: Spatial Data, Spatial Data Integration, Dataspace, SQL.

Agradecimentos

Agradeço a Deus, por me dar discernimento e controle para enfrentar os momentos mais complicados neste processo de dois anos e meio. Agradeço a minha família, minha esposa Tatiany, meus pais e meus irmãos, por me darem apoio e suportarem o estresse ao longo deste mestrado.

Agradeço aos meus orientadores, professores Cláudio Baptista e Cláudio Campelo, pelo ensinamento, conselhos, ajuda e paciência. Por lidarem com este aluno complicado, distraído e bruto.

Aos meus amigos do LSI, pois sem seu apoio e revisões este trabalho não estaria completo. Por me auxiliarem a lapidar o meu trabalho e me ajudarem quando precisei.

Agradeço aos professores convidados para a minha banca da defesa, Hugo Feitosa e Carlos Eduardo Pires, por aceitarem o convite, pelas correções sugeridas que enriqueceram o trabalho e, por fim, pela aprovação.

Sumário

Lista de Abreviaturas e Siglas

API	*Application Programming Interface*
BI	*Business Intelligence*
CLI	*Command Line Interface*
CSV	*Comma-Separated Values*
DFS	*Distributed file System*
DW	*Data Warehouse*
ETC	Extração, Transformação e Carga
EXT	*Extended File System*
GAV	*Global As View*
GLAV	*Global Local As View*
GML	*Geography Markup Language*
HDFS	*Hadoop Distributed File System*
JDBC	*Java Database Connectivity*
JSON	*JavaScript Object Notation*
KML	*Keyhole Markup Language*
LAV	*Local As View*
NTFS	*New Technology File System*
OGC	*Open Geospatial Consortium*
OLAP	*On-line Analytical Processing*
OR	*Objeto Relacional*
RDD	*Resilient Distributed Dataset*
SGBD	Sistema Gerenciador de Banco de Dados
SI	Sistema de Informação
SIG	Sistema de Informação Geográfica
SQL	*Structured Query Language*
SRID	*Spatial Reference System Identifier*
UDF	*User Data Function*
UDT	*User Data Type*
W3C	*World Wide Web Consortium*
WKB	*Well-Known Binary*
WKT	*Well-Known Text*
XML	*eXtensible Markup Language*

Lista de Figuras

Lista de Tabelas

Lista de Quadros

Lista de Códigos Fonte

Capítulo 1 – Introdução

Nas últimas décadas, o custo de aquisição e manutenção de sistemas computacionais diminuiu significativamente [1]. Com isto, organizações das mais diversas áreas têm investido bastante em Tecnologia da Informação. Quase todas as empresas, sejam de grande ou médio porte, utilizam algum Sistema de Informação (SI) para diversas atividades, incluindo a automação de processos, o registro de operações e tramitação de requisições, o gerenciamento de vendas, a realização de controle operacional e a análise de dados.

Dentre os SI utilizados, alguns possuem funcionalidades adicionais, permitindo a manipulação e a visualização de informações espaciais. Tais funcionalidades são muito úteis, tendo em vista que estima-se que cerca de oitenta por cento dos dados possuem algum componente espacial [2]. Sistemas que possuem este tipo de funcionalidade são denominados de Sistemas de Informação Geográfica (SIG).

Com a facilidade de aquisição e utilização dos sistemas computacionais, bem como a rápida evolução dos sistemas de informação, o custo da aquisição de dados também foi reduzido. Assim sendo, as organizações têm investido bastante, tanto em soluções para recuperar e gerenciar dados, quanto em sistemas que analisem estes dados espaciais. Dentre as várias tecnologias utilizadas para análise de informações destacam-se, as OLAP (*On-Line Analytical Processing*)**,** que são tecnologias de BI (*Business Intelligence*) [1] que provêm integração dos dados. Devido à grande competitividade existente no ramo empresarial, faz-se necessário investir continuamente em meios que possam extrair informações úteis dos dados.

Grande parte dos sistemas computacionais existentes confia a gerência dos dados a Sistemas Gerenciadores de Banco de Dados (SGBD), os quais, em sua maioria, utilizam o modelo lógico de dados relacional. Este é um modelo largamente difundido e que mantém a independência dos dados [3].

Por outro lado, com a evolução da Web 2.0, o advento da Internet das Coisas e a produção de dados em grande volume e em alta velocidade (denominados Big Data), tem-se investido em novos SGBD que não seguem o padrão relacional (conhecidos como sistemas NoSQL), mudando a forma como os dados são acessados [4]. Outro fator que motivou a aderência desses sistemas foi o fato de cada solução é aplicada a um domínio de problema específico. As novas tecnologias de armazenamento de dados

saem do paradigma "*one size fits all*" [5] , assim cada solução está melhor adaptada às características e necessidades de alguns cenário de problemas específicos [4].

Outrossim, novos SGBD que seguem o padrão relacional e que possuem escalabilidade e eficiência também estão surgindo no mercado - conhecidos como sistemas NewSQL [6]. Com a proliferação de sistemas heterogêneos, utilizados nas empresas, aumenta-se a demanda por integração destas diversas fontes de dados. Em alguns casos, surge ainda a necessidade de integração *on-the-fly*. Tais sistemas precisam se adequar às tecnologias de dados já utilizadas pelas organizações, sendo necessária a integração dos dados entre estas soluções.

É importante ressaltar, contudo, que mesmo com a utilização dos SI e dos SGBD nas organizações, informações valiosas ainda são mantidas em sistemas legados e em arquivos, como planilhas Excel, arquivos texto com delimitadores, ou outros tipos [1] [7] [8] [9]. Em sua maioria, estes arquivos estão no formato de planilhas, pois na maioria das vezes, é recorrido a esse tipo de solução para a geração de relatórios personalizados.

Deste modo, o cruzamento de dados oriundos de várias fontes de informação é necessário para se obter análises mais precisas e informações com maior valor agregado. No entanto, a integração destas fontes de dados heterogêneas com a possibilidade da realização de consultas em tempo real ainda é um problema em aberto [10], o qual é conhecido como problema de integração de dados [11].

Além disto, considera-se a possibilidade de manipular dados espaciais presentes em fontes distintas, ofertando um meio de acesso adicional para outros sistemas de análise de dados, como um OLAP espacial [12] ou para um Sistema de Informação Geográfica.

Grande parte das tecnologias NoSQL são limitadas no que diz respeito ao suporte a dados espaciais [13]. Dentre as que possuem tal suporte, há limitações relacionadas aos tipos de dados espaciais e às funções implementadas, sejam por restrições de desempenho, limite no armazenamento de dados (não sendo possível armazenar polígonos complexos) ou restrições relacionadas a UDT (*User Data Type*). Ademais, algumas das soluções não apresentam suporte nativo a tais tipos de dados. Outro problema é que muitas das ferramentas de análise de dados não oferecem suporte às soluções NoSQL, pois a maioria das soluções NoSQL não segue um padrão que possibilite a outros sistemas acessarem e consultarem seus dados de forma simplificada.

Muitas das soluções de análise de dados que utilizam estas tecnologias são proprietárias [1].

À medida que os sistemas de apoio à decisão possam acessar várias tecnologias de armazenamentos de dados, é proporcionada uma gama maior de informações a analisar, fazendo com que tanto indicadores quanto relatórios fornecidos sejam mais precisos. Além disso, os meios de mineração de dados também podem contar com mais informações, possibilitando uma extração de conhecimento mais completa.

Atualmente, uma das soluções que tem se apresentado mais promissora na questão de interoperabilidade entre fontes de dados heterogêneas é o Apache Drill[1] [14], que é baseada no Google Dremel [15]. Esta solução está em evolução, com uma comunidade ativa que sempre contribui para melhorar e adicionar novas funcionalidades. Trata-se, pois, de uma solução extensível que possui código aberto. No entanto, observa-se a ausência de soluções que visem estender o Apache Drill para trabalhar com dados espaciais.

Desta forma, percebe-se a falta de soluções que permitam integrar várias fontes de dados heterogêneas e que possibilitem a realização de consultas SQL convencionais e espaciais. Trata-se de uma área de pesquisa bastante promissora conhecida como *Spatial Data Integration*.

Atualmente, existem poucas soluções para a integração de fontes de dados heterogêneas. Alguns trabalhos tiveram como objetivo a integração de bases relacionais ou de arquivos, mas poucos conseguiram reunir uma gama de tecnologias com padrões distintos. Neste sentido, nota-se que, até o momento, o suporte completo a dados espaciais não foi o foco de nenhuma destas soluções.

Este trabalho visa fornecer uma alternativa para integrar dados espaciais de fontes de dados heterogêneas, possibilitando que sejam realizadas consultas espaciais. Propõe-se que a solução seja um middleware para que outros sistemas possam reutilizá-la, permitindo que estes sistemas possam integrar e consultar diversas tecnologias de armazenamento de dados (como SGBD relacionais, sistemas NoSQL, arquivos em Excel, CSV e JSON), utilizando linguagem SQL. Os arquivos podem estar localizados tanto em um sistema de arquivos local quanto em um sistema de arquivos distribuído. Não é necessário que estes sistemas criem implementações diferentes para acessar cada tecnologia e utilizem linguagens de consultas distintas.

[1] Apache Drill - https://drill.apache.org/

1.1 Questões de pesquisa

As seguintes questões de pesquisa foram elaboradas e avaliadas nesta dissertação:

1. É possível executar consultas espaciais em linguagem SQL no GeoDrill permitindo a integração de dados espaciais de fontes heterogêneas?
2. Existe diferença no desempenho das consultas realizadas no GeoDrill, dependendo do tipo de fonte de dados utilizado?
3. O desempenho do GeoDrill é equivalente ao de um SGDB espacial convencional?

1.2 Objetivos

O objetivo geral deste trabalho é desenvolver um middleware que permita a realização de consultas espaciais em bases de dados heterogêneas utilizando linguagem SQL, provendo *Spatial Data Integration*.

Os objetivos específicos são:

- Desenvolver uma solução que possibilite a análise de dados espaciais, caracterizada por um conjunto de funções espaciais que possam interoperar entre fontes de dados heterogêneas, esta solução é denominada GeoDrill.
- Simplificar a criação de funções convencionais e espaciais no GeoDrill. Este objetivo visa diminuir a sobrecarga de funções e consequentemente o tempo de desenvolvimento e manutenção destas.
- Possibilitar que a arquitetura proposta seja reusável por outros trabalhos, possibilite isolar as tecnologias utilizadas, visando reduzir dependências e possibilitando adição de novas funcionalidades.
- Desenvolver uma solução que permita avaliar o desempenho de execução das funções espaciais pelo GeoDrill.

1.3 Contribuições

A principal contribuição deste trabalho foi o desenvolvimento da solução GeoDrill, que integra dados espaciais de fontes heterogêneas utilizando linguagem de consulta SQL. Desta forma é possível analisar, consultar e cruzar os dados espaciais dos mais variados tipos de fontes (desde SGBD convencionais a soluções NoSQL) e acessar

arquivos de dados estruturados (SGBD relacional) e semiestruturados (JSON). O GeoDrill implementa todas as funções espaciais definidas pela OGC (*Open Geospatial Consortium*) e algumas adicionais existentes no PostGIS/PostgreSQL[2].

O GeoDrill foi desenvolvido como uma extensão do Apache Drill, tendo sido desenvolvida uma arquitetura que simplifica a criação e manutenção de funções tanto espaciais quanto convencionais.

A área de integração de dados está atualmente em foco, tanto para dados convencionais quanto espaciais. Referente à integração de dados convencionais, esta possui um desenvolvimento avançado, apresentando várias soluções que possibilitam agregar fontes heterogêneas. Com relação à integração de dados espaciais de fontes de dados heterogêneas, algumas soluções possibilitam fazê-la, porém de forma limitada. Dentre as limitações, está o conjunto de tipos e funções espaciais suportados, além do tipo de linguagem de consulta utilizado, que acaba não adotando os padrões internacionais definidos. Para esta área, o conjunto de soluções que permite explorar dados semiestruturados é limitado. Nem todas as soluções existentes proporcionam uma linguagem de consulta simples para analisar e integrar as diversas fontes, desta forma destacam-se as soluções que possuem linguagem de consulta baseada em SQL. A solução GeoDrill visa a integração de dados espaciais de diversas fontes, por meio da linguagem SQL, possibilitando integrar fontes de dados estruturados e semiestruturados, assim preenchendo uma lacuna existente na área.

1.4 Estrutura da Dissertação

O restante desta dissertação está estruturado da seguinte forma:

Capítulo 2 – Referencial Teórico. Neste capítulo, serão apresentados os conceitos fundamentais e as tecnologias abordados nesta dissertação.

Capítulo 3 – Trabalhos Relacionados. Neste capítulo, serão descritos os trabalhos mais importantes relacionados à temática desta dissertação e que apresentam as soluções mais recentes para interoperabilidade entre fontes de dados heterogêneas.

Capítulo 4 – Desenvolvimento da solução. Neste capítulo, serão abordados todos os pontos que cobrem o desenvolvimento da solução, desde a arquitetura às tecnologias utilizadas, incluindo os problemas e limitações destas. Também serão apresentados as funções implementadas e o contexto de utilização das mesmas.

[2] PostGIS/PostgreSQL - http://PostGIS.net/

Capítulo 5 – Avaliação da solução. Neste capítulo, será realizado um estudo comparativo para avaliar a solução no contexto de execução de várias consultas espaciais distintas utilizando várias fontes de dados, ambientes convencionais e distribuídos. A solução também será comparada ao PostGIS/PostgreSQL no quesito desempenho.

Capítulo 6 – Conclusões e Trabalhos Futuros. Neste capítulo, serão apresentadas as conclusões e discutidas as contribuições desta dissertação, bem como serão apresentadas as limitações deste trabalho e as ideias para a continuação da pesquisa em trabalhos futuros.

Apêndice 1 – Gráficos Detalhados. Neste apêndice, serão apresentados os gráficos detalhados dos resultados colhidos pelos experimentos apresentados no Capítulo 5.

Apêndice 2 – Comparações Estatísticas. Neste apêndice, serão apresentadas as comparações estatísticas detalhadas dos resultados do experimento no Capítulo 5.

Capítulo 2 – Referencial Teórico

Neste capítulo são discutidos os conceitos básicos abordados ao longo desta dissertação. Primeiramente, é apresentado o que são dados semiestruturados. Após, são apresentados e descritos dados espaciais, incluindo seus padrões e funções utilizadas para consulta. Em seguida, são descritas técnicas para a integração de dados de fontes heterogêneas; e, por fim, é descrita a solução do Apache Drill, escolhida para utilização nesta dissertação.

2.1 Dados semiestruturados

Dados podem ser classificados dependendo da estrutura como são apresentados. Como classificações, podem ser definidos dados estruturados, semiestruturados e não estruturados. Dados estruturados possuem uma estrutura fixa, definida como esquema, sendo o formato comum dos dados mantidos nos SGBD.

Dados não estruturados não apresentam estrutura definida, tendo suas informações relevantes dispostas ao longo do conjunto de dados. Exemplos desses tipos de dados são texto como notícias e artigos de jornais, revista ou páginas da web, mensagens de texto, documentos (PDF, Word), vídeos, imagens, áudios, entre outros.

Dados semiestruturados são caracterizados por não possuírem uma estruturação rígida. Assim, esses dados não seguem um esquema definido previamente. Comumente, a representação destes dados envolve a organização em uma estrutura hierárquica (uma árvore), que acaba evoluindo devido ao fato de haver atributos irregulares ou incompletos. Alguns dos atributos, presentes em um conjunto de dados semiestruturados, podem ser fixos; no entanto, cada dado deste conjunto pode vir com alguns atributos distintos. Podem ser citados XML (*eXtensible Markup Language*)[3] e JSON (*JavaScript Object Notation*)[4] como exemplos de modelos para dados semiestruturados [16].

O Código Fonte 1 e o Código Fonte 2 são exemplos contendo as mesmas informações escritas em XML e JSON, respectivamente. Estes dados são referentes a um objeto (menu) com outro objeto interno (popup) que possui uma coleção (menuitem).

[3] XML - https://www.w3.org/XML/
[4] JSON - http://www.json.org/

Estes modelos são amplamente difundidos e acompanham o avanço da internet, pois são utilizados desde a estruturação das páginas web a comunicação de dados entre cliente e servidor. Os dados em XML possuem uma estruturação com *tags*: cada *tag* possui um nome, o qual é repetido sempre que for necessário utilizá-la, o que acaba por torna-la uma estrutura verborrágica. Esta representação é utilizada comumente para criação de layouts como um HTML ou transferência de dados entre cliente e servidor. Por outro lado, os dados em JSON possuem uma estruturação mais simples a qual é bem semelhante a uma definição de uma classe em uma linguagem de programação como Java. Esta é a representação comumente utilizada para transferência de dados entre cliente e servidor e consequente manipulação destes como objeto JavaScript [17].

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()"/>
    <menuitem value="Open" onclick="OpenDoc()"/>
    <menuitem value="Close" onclick="CloseDoc()"/>
  </popup>
</menu>
```

Código Fonte 1: Dados em XML. Adaptados da fonte: http://json.org/example.html

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
}}}
```

Código Fonte 2: Dados em JSON. Adaptados da fonte: http://json.org/example.html

Tanto os dados em XML quanto em JSON podem ser manipulados e transformados em objetos, e também existem linguagens que permitem consultas a estes dados. Para XML, a mais conhecida é a linguagem XQUERY[5], a qual é definida pela W3C. Para JSON, existem várias API que permitem consultas estruturadas como JsonSQL[6] ou JSONPath[7].

Por exemplo, considerando os dados referentes aos Código Fonte 1 e Código Fonte 2, é possível acessar o atributo "popup", a lista de "menuitem" e resgatar o valor do segundo elemento da lista, a com as consultas descritas no Código Fonte 3. Várias dessas linguagens de consultas a dados semiestruturados analisam o esquema e os

[5] XQUERY - https://www.w3.org/TR/xquery/
[6] JsonSQL - http://www.trentrichardson.com/jsonsql/
[7] JSONPath - http://goessner.net/articles/JsonPath/

metadados de forma dinâmica no momento de execução da consulta e, caso não encontrem um parâmetro requisitado (a exemplo de "valor"), não ocorre erro. Nas consultas, os parâmetros não existentes são tratados como nulos.

```
XQUERY: /menu/popup/menuitem[2]/value
JSONPath: menu.popup.menuitem[1].value
```

Código Fonte 3: Exemplos de consulta a dados semiestruturados

A abordagem utilizada pelos SGBD tradicionais determina que primeiro seja definido um esquema, para, em seguida, serem armazenados os dados. Qualquer alteração na estrutura dos dados implica em alterações no esquema. Esta abordagem é conhecida como "*schema first, data later*". Com a rápida evolução da web, os dados estão sendo coletados com maiores frequências, em maiores volumes e de vários meios diferentes. Muitas vezes o formato destes dados sofrerá mudanças, mas, independentemente destas mudanças, é importante que estes dados sejam armazenados para consequente análise. Esta abordagem é conhecida como "*data first, schema later/never*", onde o foco principal é armazenar os dados e analisar seu esquema antes ou no momento da consulta, de forma dinâmica [18] [19] [20].

Vários dos SGBD convencionais possuem estratégias para lidar com algum tipo de dados semiestruturados. Para a questão do "*data first*", como exemplo, é possível armazenar os dados como objetos JSON ou XML em uma coluna, no entanto, a tabela deve ter sido criada previamente e a coluna deve ter definido o tipo semiestruturado ou um tipo textual. Para permitir a análise destes dados, existem funções que recebem como parâmetros estas informações e uma consulta em XQUERY, por exemplo. Porém, nem todo SGBD incorpora a consulta a estes objetos diretamente na sintaxe da sua linguagem SQL. Ao utilizar SQL, torna-se mais simples a utilização e aprendizagem das funções para explorar dados semiestruturados, pois não será necessário aprender uma nova linguagem de consulta.

2.2 Consultas Espaciais

Dados espaciais são dados que contêm a localização geográfica de objetos, podendo ser bidimensional ou tridimensional. Uma das formas de representação de dados espaciais é a vetorial. Estes dados podem ser representados em formas geométricas como pontos, linhas, polígonos, sequências de linhas e coleções de cada um destes tipos. A menor unidade a compor as geometrias é o ponto, que possui coordenadas referentes a longitude (x), latitude (y) e altitude (z), esta última sendo

necessária apenas caso seja uma informação tridimensional. Os dados geográficos, além das coordenadas, possuem também a referência ao sistema de projeção geográfica utilizada. Na Figura 1, são apresentados alguns dos tipos espaciais citados.

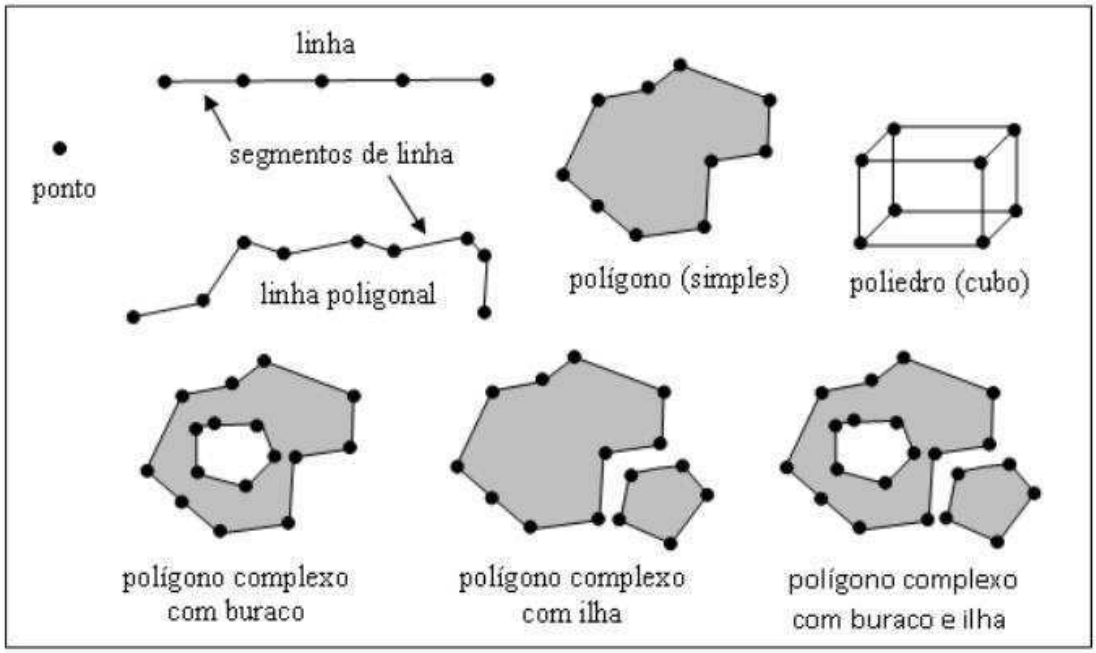


Figura 1: Tipos de dados espaciais. Fonte: [21]

Dados espaciais são coletados e utilizados diariamente por qualquer sistema da internet, de forma direta ou indireta. Considerando quão importantes estes dados são, foram criados vários padrões pela OGC[8]. Estes padrões refletem tanto na forma de representação dos dados espaciais, quanto nas funções que permitem analisar estes dados. Esta padronização objetiva a interoperabilidade entre as tecnologias que analisam informações espaciais.

Considerando a necessidade de transferir dados espaciais entre tecnologias distintas, existem padrões referentes aos formatos de escrita dos dados espaciais, dentre os quais destacam-se WKT[9] (*Well-Known Text*), WKB[10] (*Well-Known Binary*), GML[11] (*Geography Markup Language*) e KML[12] (*Keyhole Markup Language*). Além destes, existe ainda o formato Shapefiles[13] que, embora seja proprietário, é considerado um padrão de fato.

[8] OGC - http://www.opengeospatial.org/
[9] WKT - http://www.opengeospatial.org/standards/wkt-crs
[10] WKB – Capítulo 8 do documento - http://www.opengeospatial.org/standards/sfa
[11] GML - http://www.opengeospatial.org/standards/gml
[12] KML - http://www.opengeospatial.org/standards/kml
[13] Shapefiles - https://doc.arcgis.com/pt-br/arcgis-online/reference/shapefiles.htm

Por exemplo, é possível representar um ponto em WKT através do texto "POINT (10 10)" ou uma linha contendo 3 pontos através do texto "LINESTRING (10 10, 30 30, 50 50)". O mesmo ponto em WKB seria representado por "01010000000000000000002440000000000000002440" (representação em hexadecimal). Embora a representação WKB, quando visualizada em formato textual, aparente utilizar mais espaço de armazenamento, esta forma de representação utiliza menos bytes. Além disso, o sistema de referência espacial (SRID) pode ser embutido na especificação WKB e a interpretação e conversão da geometria também é mais rápida para WKB.

GML e KML são dois dos padrões para notação de dados espaciais que tomam por base o formato XML, podendo ainda ter outros dados que complementam as informações espaciais. O Código Fonte 4 e o Código Fonte 5 são exemplos de um dado espacial, representando uma linha, escrita em GML e KML, respectivamente. É possível visualizar como as notações de dados espaciais são distintas entre si.

```
<gml:LineString>
     <gml:posList>
        10 10 30 30 50 50
     </gml:posList>
</gml:LineString>
```

Código Fonte 4: Exemplo de uma linha em GML

```
<LineString>
     <extrude>1</extrude>
     <tessellate>1</tessellate>
     <altitudeMode>absolute</altitudeMode>
     <coordinates>
        10,10,0 30,30,0 50,50,0
     </coordinates>
</LineString>
```

Código Fonte 5: Exemplo de uma linha em KML

A OGC [22] define e padroniza as funções que os sistemas que possuem suporte a dados espaciais precisam implementar. Tais funções, no padrão SQL, têm o prefixo "ST", que significa *spatial type*. Como exemplo, pode-se citar a função de distância entre duas geometrias, cujo nome é *ST_Distance*.

Segundo a documentação da OGC as funções são divididas em 10 grupos. O primeiro grupo está destinado às funções para construção das geometrias. Para gerar os objetos das geometrias a partir de dados em WKT são definidas as funções *GeomFromText, PointFromText, LineFromText, PolyFromText, MPointFromText, MLineFromText, MPolyFromText* e *GeomCollFromTxt*. Para gerar geometrias de dados em WKB são utilizadas as funções *GeomFromWKB, PointFromWKB, LineFromWKB, PolyFromWKB, MPointFromWKB, MLineFromWKB* e *MPolyFromWKB*.

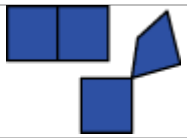
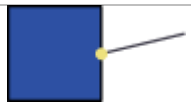
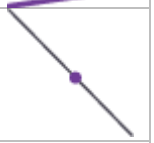
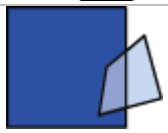
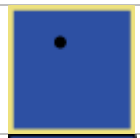
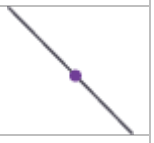
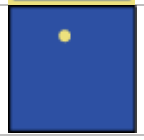
As funções que podem ser executadas no tipo “geometria” com objetivo de recuperar um atributo ou característica são: *Dimension, GeometryType, AsText, AsBinary, SRID, IsEmpty, IsSimple, Boundary* e *Envelope*. Para recuperar a geometria em formato WKT é utilizada a função *AsText*, e para recuperar em formato WKB, é utilizada a função *AsBinary*.

As funções que possibilitam converter geometrias são *ST_AsText* (recebe uma geometria como parâmetro e retorna a mesma em WKT), *ST_GeomFromText* (recebe um WKT como parâmetro e retorna o formato utilizado pelo SGBD), *ST_Point* (recebe as coordenadas e retorna um ponto).

Existem funções específicas para cada tipo de geometria. Para o tipo “ponto”, as funções podem retornar suas coordenadas - os nomes destas funções são X e Y. Para o tipo “cadeia de linhas”, as funções são: *StartPoint, EndPoint, IsClosed, IsRing, Length, NumPoints* e *PointN*. Para polígonos, as funções são: *Centroid, PointOnSurface, Area*, *ExteriorRing, NumInteriorRing* e *InteriorRingN*. Para todos os tipos de coleções de geometrias, as funções são: *NumGeometries* e *GeometryN*. Para a coleção de cadeias de linhas, os métodos são: *IsClosed* e *Length*. Para as coleções de polígonos, as funções são *Centroid, PointOnSurface* e *Area*.

As próximas funções são ditas topológicas e testam se duas geometrias possuem uma relação e retornam um valor booleano. Estas funções são: *Equals, Disjoint, Touches, Within, Overlaps, Crosses, Intersects, Contains* e *Relate*. O Quadro 1 apresenta exemplos destas funções espaciais e como se relacionam com os diferentes tipos espaciais. As células vazias indicam que não é possível realizar a função entre os dois tipos de geometrias.

Quadro 1: Exemplos de funções topológicas. Fonte: http://www.gitta.info/SpatialQueries/en/html/TopoBasedOps_learningObject1.html

	Polígono-Poligono	**Linha-Linha**	**Ponto-Ponto**	**Polígono-Linha**	**Polígono-Ponto**	**Linha-Ponto**
Disjoint						
Touches						
Overlap						
Contains						
Inside						
Covers						
Covered by						
Equal						

As próximas funções recebem duas geometrias e retornam um valor resultante. A primeira função é *Distance*, que retorna a distância entre as duas geometrias. As funções *Intersection, Difference, Union* e *SymDifference* retornam uma geometria resultante da operação entre as duas geometrias fornecidas como parâmetros. As funções *Buffer* e *ConvexHull* podem ser aplicadas a uma única geometria. Na Figura 2 são demonstrados os resultados de execução das funções *Intersection*, *Difference*, *Union* e *SymDifference* em duas geometrias. Na Figura 3 e na Figura 4 são demostradas as funções *ConvexHull* e *Buffer*, respectivamente.

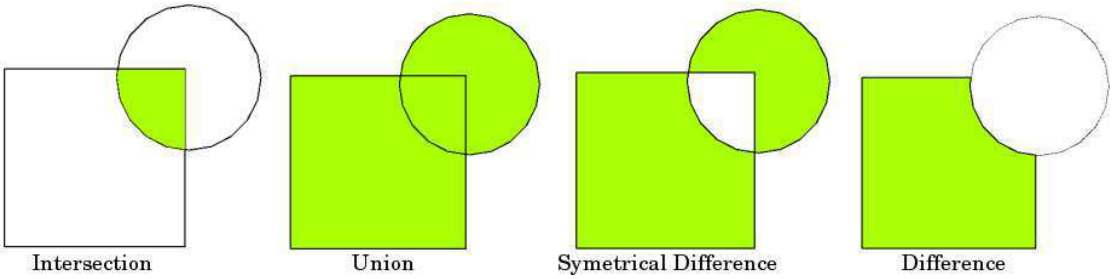


Figura 2: Exemplo de funções que implementam operadores espaciais. Fonte: https://docs.qgis.org/2.8/en/docs/gentle_gis_introduction/vector_spatial_analysis_buffers.html

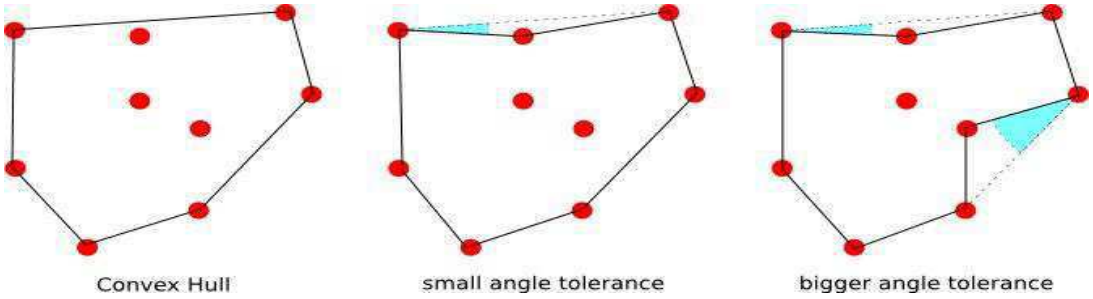


Figura 3: Exemplo de função ConvexHull em uma coleção de pontos. Fonte: http://state.gift/konvexe-hulle_1076692.html

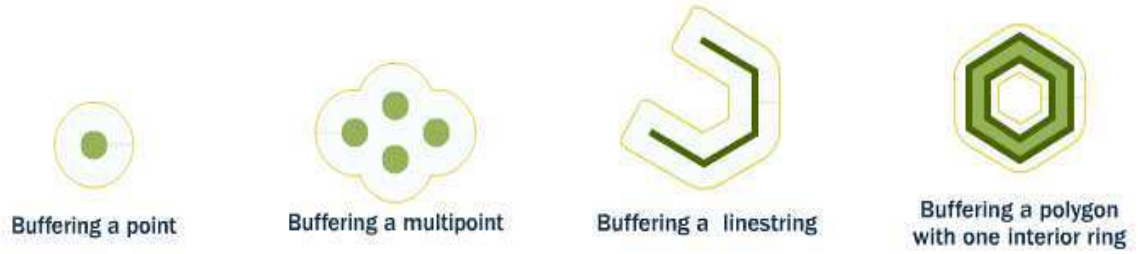


Figura 4: Exemplo da função Buffer. Fonte: http://suite.opengeo.org/docs/latest/dataadmin/pgBasics/generation.html

2.3 Integração de dados

As organizações podem manter seus dados em diversas fontes de dados heterogêneas, as quais podem estar espalhadas pelos departamentos, servidores e estações de trabalhos. Dentre estas fontes de dados, destacam-se os SGBD e arquivos de textos, como planilhas e arquivos no formato CSV. Com a evolução da web, o volume dos dados que alimentam os sistemas organizacionais vem aumentando, assim surgindo sistemas de gerenciamento de dados que manipulam dados semiestruturados e visam escalabilidade. Dentre os novos sistemas, destacam-se aqueles conhecidos como NoSQL.

Como consequência da crescente utilização de dados oriundos de fontes heterogêneas, investe-se cada vez mais em soluções que visam integrar estes dados, objetivando sempre fornecer uma informação mais precisa para os usuários. Nesta seção é abordada a caracterização destes tipos de tecnologias.

2.3.1 Arquitetura de *Data Warehousing* (DW)

Uma das arquiteturas base para integração das informações oriundas de diversas fontes é a de *Data Warehousing* (DW), que representa o ambiente central onde os dados serão integrados, armazenados e analisados. Para carregar os dados no DW, é realizado um processo de ETC (Extração, Transformação e Carga), por meio do qual podem-se extrair os dados de várias tecnologias distintas e carregá-los em um SGBD. Com esta carga, os dados ficam materializados no DW. O DW tem uma arquitetura que objetiva melhor desempenho para consultas do tipo OLAP (*Online Analytical Processing*) [23].

No entanto, o processo de ETC possui algumas limitações: por exemplo, é necessário criar um mapeamento dos esquemas de origem para o destino. Porém, o processo é sensível a este mapeamento, mudanças drásticas no esquema requerem mudanças no processo de ETC. Há uma maior complexidade em trabalhar com a informação em tempo real quando se utiliza desse processo de carregamento. A Figura 5 ilustra a arquitetura do DW.

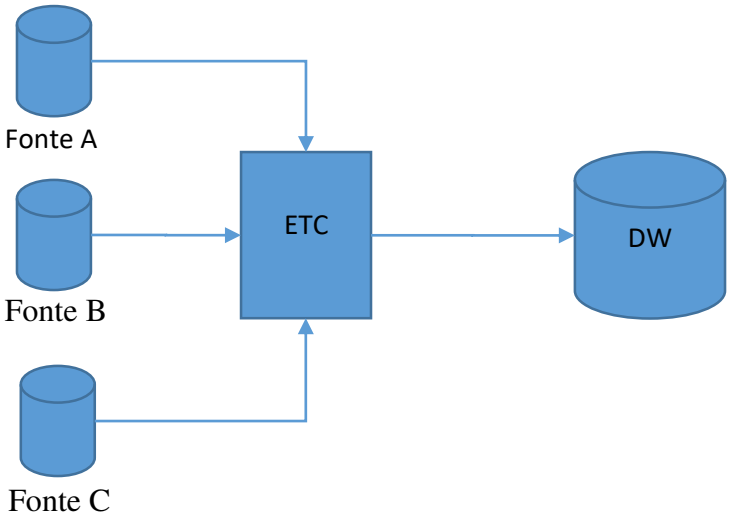


Figura 5: Arquitetura de um DW

2.3.2 Arquitetura baseada em mediadores

Outra arquitetura para a integração de dados utiliza mediadores. Esta arquitetura não materializa os dados, desta forma sempre dispondo da informação mais atual. O

mediador trabalha como um serviço que quebra e traduz as consultas para serem executadas em cada fonte de dados. Esta estratégia utiliza um esquema global para a realização das consultas, no entanto, existem mais problemas a serem resolvidos com essa arquitetura: é necessário que exista um mapeamento entre o esquema global, utilizado para a integração, com os esquemas locais [24]. A Figura 6 mostra a arquitetura de um mediador.

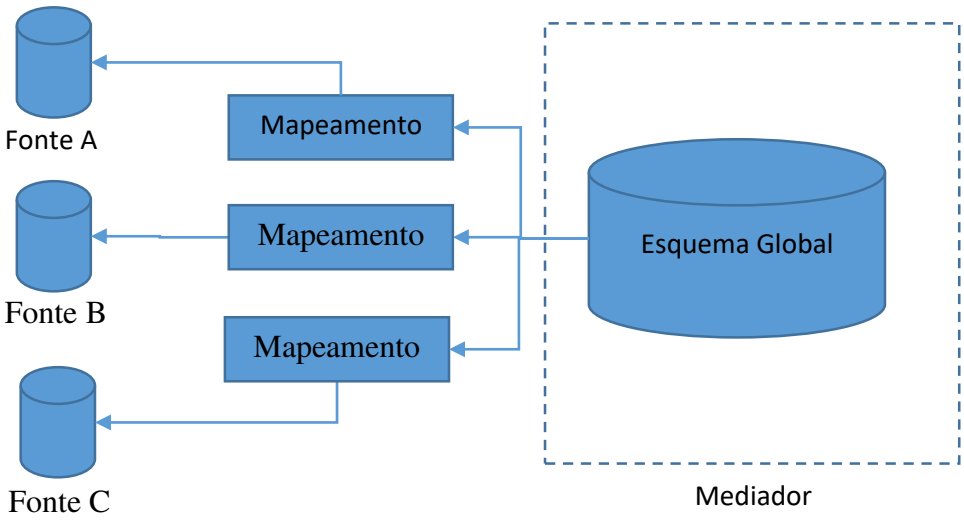


Figura 6: Arquitetura de um mediador

Kosh [24] descreve 3 tipos de mapeamento entre o esquema Global e os esquemas das fontes de dados. O primeiro é denominado de *Global As View* (GAV), no qual o esquema global é definido de acordo com os esquemas das fontes. Neste caso, o mapeamento é do esquema global para o local. Para este mapeamento, é mais simples traduzir as consultas; no entanto, o esquema global é mais sensível às alterações nas fontes. O segundo é denominado de *Local As View* (LAV), no qual o esquema global é previamente definido e o mapeamento é uma visão no esquema local. O sentido deste mapeamento é do esquema local para o global. Este, por outro lado, torna a tradução de consultas mais complexas; no entanto, o esquema global não é dependente das fontes. Por fim, a última abordagem de mapeamento é denominada de *Global Local As View* (GLAV). Neste caso, é possível ter visões refletindo os esquemas locais e um mapeamento global agrupando todas as visões.

2.3.3 Arquitetura baseada em *dataspaces*

O conceito de *dataspaces* foi introduzido por Franklin et al. [10] e objetiva superar os problemas comuns com a integração de dados que ocorrem nas outras arquiteturas. Estes sistemas não mantêm dados (materialização) e também não

trabalham com a definição de um esquema global, sendo desta forma sistemas propícios a analisar arquivos semiestruturados e soluções NoSQL. Estes, por sua vez, trabalham sob demanda de forma a explorar as fontes de dados e seus esquemas. Dependendo das consultas a serem executadas nestes sistemas, é possível utilizar abordagens mais integradas para atender a demanda. A Figura 7 mostra a arquitetura de *dataspaces*.

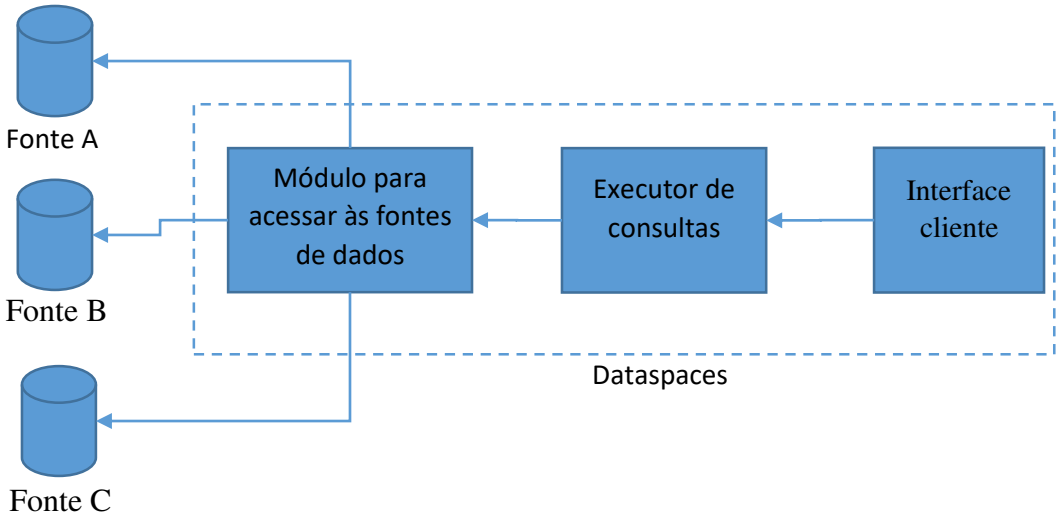


Figura 7: Arquitetura de dataspaces

Os sistemas baseados em *dataspaces* não só analisam e integram informações de fontes distintas, assim como também podem manipular informações nestas fontes, como atualizar ou inserir registros. Estes sistemas mantêm uma linguagem comum para interoperar entre as possíveis fontes de dados.

2.4 Apache Drill

O Apache Drill é uma solução baseada no Google Dremel [15], que pode ser considerada um *dataspace*. Esta solução possibilita integrar várias fontes de dados, não sendo necessário definir um mapeamento ou esquemas globais para as fontes de dados acessíveis, além de explorar os esquemas das fontes sob demanda. O Drill pode ser utilizado como um *middleware* que objetiva a realização de consultas *ad-hoc* em vários tipos de fontes de dados, desde SGBD convencionais a algumas tecnologias NoSQL, e arquivos, como CSV e JSON. Para poder apresentar um bom desempenho nas consultas, o Drill utiliza meios para processar os dados em memória principal e apresenta recursos para ser instalado em um cluster distribuído, além de poder ser utilizado em modo convencional, com uma única máquina. O Drill é uma solução de código aberto desenvolvido em Java para integrar dados convencionais de várias fontes de dados heterogêneas.

Ao utilizar o Drill é possível realizar consulta em várias bases de dados sem a necessidade de materialização dos dados. Dentre as fontes de dados acessíveis, estão quaisquer SGBD que possuam conexão via ODBC/JDBC e possam ser consultados via SQL. É possível acessar vários sistemas de arquivos como o local (NTFS ou EXT), distribuídos como o HDFS, Amazon S3, Azure Blob Storage, Google Cloud Storage, NAS, dentre outros. Dentre os tipos de arquivos que podem ser lidos pode-se elencar JSON, CSV e outros arquivos delimitados. As soluções NoSQL possíveis de conectar e explorar são HBase[14], Hive[15] e MongoDB[16].

Além de poder acessar e consultar os dados de cada fonte de dados isoladamente, o Drill também pode cruzar os dados destas diversas fontes. Para realizar as consultas a qualquer uma das fontes, o Drill utiliza linguagem de consulta convencional, SQL 2003[17]. O Drill pode ser acessado como um SGBD, utilizando um drive ODBC ou JDBC, ou como uma API nativa, *web service* e linha de comando. Por isto, esta solução pode ser utilizada para analisar e cruzar os dados destas várias fontes não convencionais. Excel[18], Tableau[19] e Qlik[20] são exemplos de soluções que acessam SGBD convencionais que podem utilizar o Drill para acessar soluções NoSQL e arquivos semiestruturados.

No Código Fonte 6, são apresentados alguns exemplos de consultas realizadas no Drill, onde cada uma delas acessa uma fonte de dados distinta. A primeira consulta (Q1) acessa arquivos de log no sistema de arquivos. A segunda consulta (Q2) acessa uma coleção de usuários no MongoDB, agrupando a quantidade de usuários por país. Por fim, a terceira consulta (Q3) acessa um arquivo JSON em um sistema de arquivos da Amazon, filtrando os usuários pelo identificador.

```
Q1: SELECT * FROM dfs.root.`/web/logs`;

Q2: SELECT country, count(*)  FROM mongodb.web.users
  GROUP BY country;

Q3: SELECT timestamp  FROM s3.root.`clicks.json`
  WHERE user_id = 'jdoe';
```

Código Fonte 6: Consultas no Apache Drill

[14] HBase - https://hbase.apache.org/
[15] Hive - https://hive.apache.org/
[16] MongoDB - https://www.mongodb.com/
[17] SQL 2003 - http://savage.net.au/SQL/sql-2003-2.bnf.html
[18] Microsoft Excel - https://products.office.com/pt-br/excel
[19] Tableau - http://www.tableau.com/pt-br
[20] Qlik - http://global.qlik.com/br

Para identificar a fonte de dados na consulta, o Drill utiliza uma notação semelhante a vários SGBD na qual o nome de uma tabela é composto pelo esquema e banco de dados a qual pertence, da seguinte forma: <banco>.<esquema>.<tabela>. Desta forma, o Drill pode cruzar várias fontes de dados, mas mantendo o padrão da linguagem SQL. Pode se tomar como exemplo o Código Fonte 6, na primeira consulta (Q1) na clausula "FROM" a fonte de dados é o sistema de arquivos (DFS), o esquema é a raiz do sistema e o arquivo é endereçado. Na segunda consulta (Q2), é utilizado o MongoDB como base de dados. Na última consulta (Q3) os dados são consultados no sistema de arquivos distribuídos da Amazon (S3).

Para realizar consultas a várias fontes de dados, o Drill utiliza técnicas propostas no artigo que introduz o Dremel [15]. A técnica utilizada para dividir a consulta entre as fontes de dados é denominada de *multilevel query execution trees*. Ao executar uma consulta, esta é traduzida em um plano lógico onde são definidos os metadados, as fontes de dados onde cada dado reside e o fluxo dos dados. Em seguida, o plano lógico é otimizado e convertido em um plano físico que descreve como a consulta deve ser executada. Na Figura 8 são ilustradas as etapas de tradução de uma consulta.

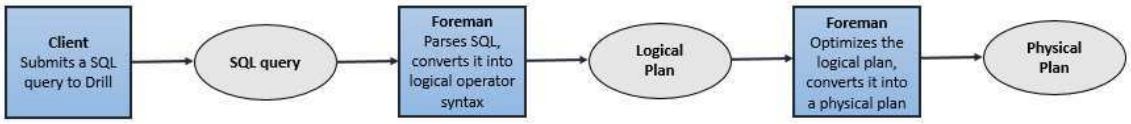


Figura 8: Lógica para tradução das consultas no Drill. Fonte: https://drill.apache.org/docs/drill-query-execution/

A técnica de *multilevel query execution trees* é utilizada para fragmentar o plano físico, possibilitando que porções da consulta sejam paralelizadas. Desta forma, é montada uma árvore onde os fragmentos são os nós e folhas. Um exemplo desta árvore pode ser visualizado na Figura 9, onde cada nível da árvore apresenta um conjunto de fragmentos que pode ser executado de forma independente. É possível paralelizar as consultas executando primeiro os fragmentos folhas, iterando sobre os níveis da árvore até finalmente processar o fragmento representado pela raiz. Assim, são os fragmentos folhas que extraem os dados das tabelas nas fontes de dados, e existirá no mínimo um fragmento folha por tabela.

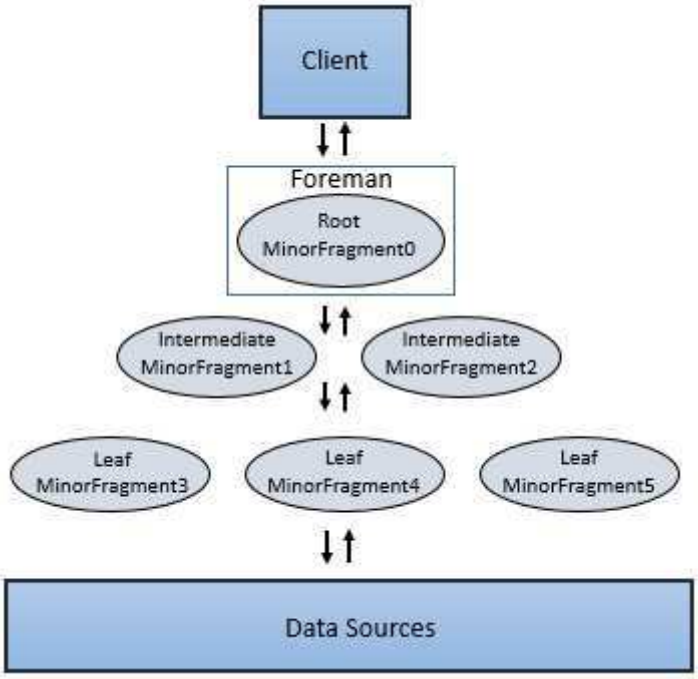


Figura 9: Exemplo de estrutura de fragmentos do plano de execução físico. Fonte: https://drill.apache.org/docs/drill-query-execution/

Para executar um fragmento da consulta em uma fonte de dados, o Drill utiliza um de seus componentes denominado de *Storage plugin interface*. Esta interface define a leitura e escrita das informações nas fontes de dados, além de apresentar regras de otimização e ler metadados das fontes de dados. Assim, é selecionado o *Storage Plugin* específico da fonte e o fragmento em plano físico é traduzido para a linguagem da fonte de dados.

Para explorar dados semiestruturados como JSON, o Drill possibilita uma navegação estrutural, semelhante ao acesso aos atributos de um objeto, além de apresentar várias funções para manipular coleções presentes nestes tipos de dados. Um exemplo da navegação citada pode ser visualizado no Código Fonte 7.

```
SELECT t.level1.level2[n][2] FROM dfs.myfile.json t;
```

Código Fonte 7: Consulta a arquivo JSON, explorando coleção. Adaptado da fonte: https://drill.apache.org/docs/json-data-model/

O Drill permite cruzar os dados de fontes distintas, realizando operações como junção, interseção, união e qualquer outra definida no padrão SQL. No Código Fonte 8 é mostrada uma consulta com a junção entre dois arquivos JSON - a condição é realizada utilizando o identificador dos objetos de cada arquivo (id). Ressalva-se que poderiam ser utilizadas fontes de dados distintas.

```
SELECT tbl1.id, tbl1.type FROM dfs.`donuts.json` as tbl1
JOIN dfs.`moredonuts.json` as tbl2 ON tbl1.id=tbl2.id;
```

Código Fonte 8: Consulta com junção de arquivos JSON. Adaptado da fonte: https://drill.apache.org/docs/using-sql-functions-clauses-and-joins/

Além de permitir explorar várias fontes de dados utilizando consultas em SQL, o Drill ainda apresenta extensibilidade referente a adição de novas UDF (*User Defined Functions*). Para desenvolver funções para o Drill, é necessário criar um projeto na linguagem de programação Java utilizando a API da versão para a qual deseja criar as funções. Cada função é uma classe Java que implementa uma das interfaces do Drill específicas para as funções. Não é necessário alterar o código do Drill para adicionar as novas funções. Todas as instruções podem ser visualizadas na documentação do Apache Drill[21].

O Drill é extensível apenas com funções, não provendo funcionalidade para novos tipos de dados, possuindo capacidade adicional de escrever no sistema de arquivo o resultado de uma consulta. Este arquivo é escrito no formato CSV ou JSON, podendo ser então consultado e atualizado. Esta capacidade de escrita somente está disponível para arquivos.

2.5 Conclusão

Neste capítulo, foi apresentada a fundamentação teórica para este trabalho. Foram abordados os conceitos referentes à definição dos dados semiestruturados e problemas na manipulação destes. Em seguida, foram apresentados conceitos da definição de dados espaciais, os padrões utilizados para prover interoperabilidade entre as soluções que dão suporte a dados espaciais e os tipos de dados espaciais. Além disso, foram apresentados os conceitos e arquiteturas referentes à integração de dados. Por fim, foi feita uma descrição geral da solução que será utilizada neste trabalho, o Apache Drill.

No próximo capítulo, serão abordados os trabalhos relacionados a esta dissertação, citando estudos que visam integrar dados de fontes heterogêneas, e também que manipulam dados semiestruturados e dados espaciais.

[21] Documentação UDF - https://drill.apache.org/docs/develop-custom-functions-introduction/

Capítulo 3 – Trabalhos Relacionados

Neste capítulo, serão abordados trabalhos que possibilitaram a integração de dados de fontes heterogêneas, primeiro serão abordadas soluções para integração de dados convencionais e em seguida para dados espaciais.

3.1 Tecnologias de integração de dados não espaciais

Com a evolução da computação e da web, dados são coletados de forma frequente, sendo apresentados em vários formatos, alguns seguem esquemas rígidos (dados estruturados), outros apresentam esquemas flexíveis (dados semiestruturados), como também outros tipos não estruturados. Visando explorar as informações contidas nestes dados, tem-se investigado a melhor forma de integrá-los considerando que em muitos casos é necessário cruzar os vários tipos distintos de dados.

Considerando esse problema, Melnik et al. [25] desenvolveram o Dremel, que visa, através de uma arquitetura baseada em *dataspaces*, integrar dados de fontes de dados heterogêneas com escalabilidade para milhões de registros. Foi então proposto um modelo de armazenamento de dados colunar que possibilita lidar com dados em diversos formatos, como estruturados e semiestruturado, além de utilizar uma linguagem de consulta baseada em SQL. O Dremel realiza as consultas de forma distribuída sem interferência direta do usuário. O processamento da consulta distribuída executa somente porções específicas da consulta em cada máquina, conceito definido como *multilevel query execution trees*, de modo a simplificar e diminuir o volume de dados transferido.

No entanto, o trabalho de Melnik et al. apresenta algumas limitações. Primeiramente, não é capaz de realizar consultas em dados espaciais. Segundo, não consegue integrar dados de sistemas distintos como arquivos, NoSQL, ou um SGBD, sem replicá-los. Por fim, a utilização desta solução por outro sistema seria mais simples, se fosse utilizada linguagem SQL padrão.

O trabalho de Hausenblas e Nadeau [26] é baseado no Dremel, utilizando tanto o conceito de escalabilidade quanto a ideia de dividir o processamento da consulta entre as máquinas do servidor, a solução deste trabalho é denominada de Apache Drill. O Drill é uma solução para realização de consultas e análise de dados *ad-hoc*. Este possui suporte à linguagem de consulta SQL 2003 e possibilita acesso via API nativa,

JDBC/ODBC e linha de comando (CLI). Com o apache Drill é possível consultar dados em SGBD convencionais além de soluções NoSQL e arquivos com dados aninhados, semiestruturados ou estruturados, como JSON e CSV. A solução permite escalonamento horizontal, em um cluster com várias máquinas. Utilizando este tipo de escalonamento o Drill possibilita dividir a consulta em fragmentos e distribui-los entre as máquinas, desta forma, cada máquina pode processar somente a porção da consulta referente ao conjunto de dados que possui, sendo uma medida para o ganho de desempenho. Outro meio para ganho de desempenho utilizado pelo Drill é o processamento das consultas em memória principal. Por fim, esta solução permite criar novas UDF (*User Defined Function*), desta forma permitindo que suas funcionalidades sejam estendidas sem que os usuários precisem alterar seu código fonte. As limitações da solução estão no suporte a UDT (*User Defined Type*), isto impede que novos tipos e estruturas de dados, que possam existir em uma fonte ou aqueles criados por outros usuários, possam ser utilizados pelo Drill sem que seu código fonte seja alterado. Ademais, não há suporte a dados espaciais.

O trabalho de Zaharia et al. [27] apresenta a solução Apache Spark, que foca na integração de dados utilizando uma linguagem procedural de alto nível para analisar dados de forma distribuída. Este trabalho apresenta um framework para processamento de BigData, o qual é uma alternativa às opções existentes de MR (*MapReduce*), como o Hadoop. O Spark traz um novo modelo de computação distribuída que permite que os usuários possam programar em alto nível utilizando uma API em uma linguagem de programação como Java ou Scala e não somente o paradigma do MR. Além disso, resolve alguns problemas computacionais que não são eficientes com as soluções MR convencionais por abordar processamento em memória. A API do Spark simplifica o paradigma de programação distribuída e MR, para isto utiliza-se uma estrutura de dados denominada *Resilient Distributed Dataset* (RDD).

O modelo de programação apresentado é um pouco restritivo em relação às funções suportadas, no entanto simplifica o paradigma MR, por utilizar um modelo de programação imperativa. A solução apresenta melhorias e ganho de desempenho em relação ao Hadoop, devido, principalmente, às estruturas de dados criadas.

O Apache Spark possui limitações no quesito de usabilidade, pois outras aplicações precisariam alterar seu código fonte para acessarem-no e consultarem-no, devido ao Spark utilizar protocolo para acesso e linguagem de consulta próprias. O Spark possui um modelo de programação restritivo, pois, para uma análise eficiente e

escalável, devem-se utilizar as estruturas de dados e métodos da API fornecida. Devido a este modelo restritivo, o Spark não abstrai totalmente o conceito de MR: para uma análise de dados avançada é necessário utilizar programação em MR. Por fim, outra limitação é a falta de suporte a dados espaciais. Esta é uma solução voltada a um ambiente distribuído, principalmente por necessitar de conceitos de MR.

Com objetivo de abstrair parte de conceitos específicos da API do Apache Spark e simplificar a análise dos dados, Armbrust et al. [28] apresentam uma extensão que permite utilizar a linguagem SQL para auxiliar nas análises dos dados, definida como Spark SQL. Esta extensão acrescenta a capacidade de realizar consultas SQL diretamente no Apache Spark. Destarte, a mesma possibilita acesso via JDBC e adiciona uma nova estrutura de dados a API, com a adição dessa extensão, é possível realizar a integração e consulta a dados de fontes distintas utilizando consultas SQL.

Como contribuições, essa extensão adicionou uma nova estrutura de dados chamada de dataframe, na qual podem ser utilizadas consultas em SQL para analisar os dados. Além disso, foi acrescentada maior extensibilidade à solução com a adição de um compilador, o qual pode ter suas regras modificadas utilizando as linguagens de programação do Apache Spark. Possibilita assim que usuários criem novas funções, tipos e estruturas de dados de uma forma mais simples. No entanto, essa solução não possui suporte a dados espaciais. A linguagem SQL apresentada para analisar os dados não utiliza todos os termos e funções disponíveis do padrão SQL, além de que, para realizar a análise, é necessário criar um código procedural embutindo a consulta SQL. Por se tratar de uma extensão para o Spark, esta objetiva um ambiente distribuído.

3.2 Tecnologias de integração de dados espaciais

Em Aji et al. [29], tanto o Apache Hadoop quanto o Hive foram modificados para possibilitar o processamento de BigData espacial utilizando a linguagem HiveQL, linguagem baseada em SQL. Esta extensão, denominada Hadoop GIS, adiciona a capacidade de indexação de dados em tempo real e vários tipos de consultas e funções espaciais além de estender os tipos de dados suportados pelo Hive. Com essa solução, é possível utilizar o Hive para integrar dados estruturados e semiestruturados. No entanto, essa solução só se torna eficiente à medida que o volume de dados aumenta e exista um cluster com tamanho considerável para prover o poder computacional necessário. Além disto, essa solução é baseada em Hive/Hadoop, que por sua vez são consideradas

soluções para processamento em batch, pois as mesmas não garantem o processamento dos dados em tempo real.

Conforme o que foi discutido por Franklin et al. [30], para um ambiente de Big Data, são necessárias soluções que possam representar, tratar, indexar e analisar dados espaciais. Com este objetivo Eldawy e Mokbel [31] desenvolveram o SpatialHadoop.

O SpatialHadoop é uma extensão do Hadoop que apresenta capacidade de representar dados espaciais, como pontos, linhas e polígonos; de criar índices espaciais; e de realizar consultas topológicas. Na criação de índices espaciais, a ferramenta utiliza o processamento distribuído e o sistema HDFS (*Hadoop Distributed File System*) [32] para calcular, manter e consultar o índice espacial. Além disso, é utilizada uma estratégia de índice espacial em dois níveis para poder dividir o trabalho e melhorar o desempenho das consultas. Em relação às consultas topológicas, todas são implementadas utilizando o paradigma MR. Adicionalmente, foi criada uma extensão da linguagem de consulta do Hadoop, denominada Pigeon [33], para manipulação e consulta dos dados espaciais, possibilitando a utilização de operadores espaciais.

Como limitações, uma aplicação do SpatialHadoop deve utilizar a API nativa para comunicação, além de executar as consultas utilizando MR ou um script na linguagem Pigeon. Desta forma, é necessário alterar tanto como as aplicações se comunicam quanto a forma de realizar as consultas. Também é necessário carregar os dados previamente no sistema HDFS, para assim processá-los, e não foi apresentado se a Pigeon dá suporte ao processamento de dados semiestruturados.

Os autores do SpatialHadoop escreveram outro artigo [34] para apontar as características necessárias a uma solução de análise de Big Data espacial destinada a usuários finais. Neste artigo, foi definido que os usuários finais das soluções não necessitam possuir conhecimento técnico específico da tecnologia de armazenamento dos dados.

Nos casos de uso são apresentadas algumas tecnologias desenvolvidas para tratar problemas de consulta e visualização de dados, de uma forma que as soluções não são totalmente genéricas, mas voltadas aos usuários finais. As próprias soluções usam como base o SpatialHadoop e não são extensíveis nem podem ser utilizadas por outras tecnologias, e a camada de visualização dos dados é acoplada diretamente à solução. Algumas das soluções citadas são Tareeg [35] e Shahed [36].

A solução Tareeg[22], desenvolvida por Alarabi et al. [35], é uma solução web que permite a extração e visualização de dados oriundos do OSM (*Open Street Maps*). Por exemplo, a partir de uma área definida pelo usuário, a solução extraí todas as estradas do OSM mapeadas na área determinada. Esta é uma solução desenvolvida para este propósito específico, visto que o conjunto de dados do OSM é volumoso (aproximadamente 500 GB), dificultando e, sendo computacionalmente custoso, a tarefa de carregar estes dados em um SGBD espacial e, em seguida, realizar a extração. Por ter um propósito específico, não é possível realizar consultas além das definidas. Outrossim, não é uma solução de consulta em tempo real e não existe uma API ou serviço para que esta solução seja utilizada por outras aplicações.

Outra solução baseada no SpatialHadoop é a Shahed. Tal solução, desenvolvida por Eldawy et al. [36], permite consultar, minerar e visualizar dados Big Data espaço temporal providos por satélites. Como funcionalidades é possível selecionar informações e agregá-las, considerando uma área de seleção dos dados e uma faixa de tempo. Esta solução possui propósito específico, pois oferece algumas funções para análise e o conjunto de dados a ser analisado não pode ser modificado. A solução oferece uma interface gráfica objetivando facilitar a manipulação por parte dos usuários. Como limitações, a solução apresenta a característica de não ser tempo real e de não possuir nenhuma outra forma para que uma aplicação possa usufruir do serviço da Shahed.

No artigo de Zhang et al. [37] é demonstrada uma extensão do Hbase para manipular dados espaciais, este trabalho foca na descrição da extensão bem como no trabalho dos índices, mas não explicita as funções criadas, nem descreve o conjunto de dados utilizados no experimento. O tipo de índice criado é de grid. O Hbase trabalha sobre o Hadoop e é um tipo de sistema NoSQL com armazenamento colunar, mas apresenta melhor desempenho na realização de consultas em tempo real.

Para esta extensão, o problema é o tipo de linguagem de consulta a ser utilizada, que é uma linguagem de consulta própria que não consegue integrar dados de outras fontes, mas trabalha com dados semiestruturados. Por se tratar de uma aplicação baseada em MR, esta necessita de cluster para poder apresentar bom desempenho.

Em Zhang, Song e Liu [38] é apresentada uma solução para armazenamento de Big Data espacial baseada em MongoDB. No entanto, existem alguns problemas em

[22] Tareeg - http://siwa-umh.cs.umn.edu/app/webroot/osme/

relação a esse tipo de armazenamento e subsequente consulta. O principal problema é que a solução dá suporte exclusivamente ao tipo espacial “ponto”. Mesmo que houvesse suporte a outros tipos espaciais, o MongoDB possui limitações com a replicação dos dados e armazenamento de BigData espacial, a limitação no tamanho do documento possível a ser adicionado ao MongoDB é atualmente de 16 MB. Este limite impede que certos objetos espaciais, de grande volume, possam ser armazenados nesta solução.

Por fim, esse artigo trata exclusivamente do armazenamento dos dados espaciais no MongoDB e da utilização de sua estrutura de replicação de dados. Não foi definida nenhuma função espacial para consulta aos dados.

Para a análise, consulta e integração de dados espaciais considerando um ambiente de alto nível, foi criada uma extensão para o Apache Spark denominada GeoSpark por Yu, Wu e Sarwat [39]. Esta solução adiciona novos tipos de dados para o Spark, além de adicionar funções espaciais para consultar os dados. É utilizada a linguagem procedural do Spark para realizar consultas, esta não é integrada com Spark SQL, além de ter implementado alguns tipos de funções espaciais. Estas funções são *mínimo bounding box*, união de polígonos, sobreposição/contem (em auto junções), distância espacial (*ST_DWithin*), junção espacial (*ST_Intersects*) e KNN (*K-Nearest Neighbors*) espacial. No entanto, o GeoSpark é limitado no que concerne à disponibilidade de funções espaciais, restringindo assim os tipos de análises a serem realizadas. Também há limitação quanto à linguagem de consulta, pois não é possível utilizar as funções espaciais com a linguagem do Spark SQL.

A solução Simba, desenvolvida por Xie et al. [40], é baseada no Spark SQL e objetiva analisar dados espaciais. Esta beneficia-se de meios de consulta como a linguagem SQL e dos meios de acesso adicionados pelo Spark SQL como acesso via JDBC. São adicionados tipos, índices e funções espaciais, os quais podem ser manipulados em linguagem procedural ou SQL. Como limitações somente o tipo ponto foi adicionado, o conjunto de funções também é limitado, principalmente devido ao tipo ponto, são realizados poucos tipos de consultas espaciais. As funções espaciais disponíveis são KNN espacial, junção com KNN espacial, junção com distância espacial (*ST_DWithin*), distância espacial com círculo e *bounding box*.

3.3 Considerações Finais

Neste capítulo, foram abordados dois conjuntos de aplicações, primeiro foram as aplicações para consulta e integração de dados, que possibilitam analisar dados

semiestruturados. Em seguida foram revisadas soluções que possibilitam analisar o componente espacial nestes dados, com a possibilidade de integração. Em uma visão geral é possível levantar características para serem avaliadas de cada trabalho apresentado, como:

Linguagem de consulta: qual linguagem de consulta utilizada pela solução para analisar os dados. Pode ser SQL, alguma variação de SQL ou uma linguagem própria;

Middleware: Permite o acesso por outras aplicações, este acesso pode ser feito por API nativa, JDBC/ODBC ou CLI. Outras aplicações podem utilizar esta solução como meio para análise de dados.

Extensível: se a solução é extensível, por meio de criação de tipos e funções;

Dados semiestruturados: é possível explorar e analisar dados semiestruturados na solução;

Integra fontes de dados distintas: a solução permite acessar dados de outras fontes distintas;

Suporte espacial: para essa característica são definidos três níveis, não possui, básico quando a solução suporta alguns tipos e funções espaciais ou completo quando a solução suporta todos os tipos espaciais e as funções definidas pela OGC;

Ambiente Desktop: A solução foi desenvolvida para ser utilizada em um ambiente convencional (Desktop), por exemplo, como meio para consultar outras fontes e explorar dados semiestruturados. Não necessita de um cluster para ter bom desempenho. Como observado no Quadro 2, dentre os trabalhos apresentados é possível verificar que existem lacunas em relação a possibilitar manipular dados semiestruturados e dados espaciais em um ambiente convencional ou distribuído utilizando uma linguagem de consulta padrão como SQL. Além disso, nem todas as soluções que apresentam extensões espaciais apresentam um conjunto completo de suporte a tipos e funções espaciais.

Quadro 2: Comparação dos trabalhos

	Dremel	**Apache Drill**	**Apache Spark**	**Spark SQL**	**Hadoop GIS**	**SpatialHadoop**	**Hbasespatial**	**GeoSpark**	**Simba**
Linguagem de consulta	Baseada em SQL	SQL	Própria	SQL	Baseada em SQL	Baseada em SQL	Baseada em SQL	Própria	SQL
Middleware	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Extensível	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Dados semiestruturados	Sim	Sim	Sim	Sim	Não	Não	Sim	Sim	Sim
Integra fontes de dados distintas	Não	Sim	Sim	Sim	Não	Não	Não	Sim	Sim
Suporte espacial	Não	Não	Não	Não	Básico	Completo	Básico	Básico	Básico
Possibilidade de utilizar em ambiente convencional	Não	Sim	Não	Não	Não	Não	Não	Não	Não

No próximo capítulo discorre-se sobre consultas espaciais a fontes heterogêneas livres de esquemas.

Capítulo 4 – GeoDrill: consultas espaciais a fontes heterogêneas livres de esquema

Neste capítulo, é apresentada a solução desenvolvida para possibilitar a realização de consultas espaciais em bases de dados heterogêneas livres de esquema. Esta solução foi implementada na forma de um middleware chamado de GeoDrill, uma vez que este é uma extensão da ferramenta Apache Drill, provendo suporte à interoperabilidade de dados espaciais. Inicialmente, são abordadas questões referentes à extensibilidade da tecnologia utilizada (Apache Drill) e às atividades preparatórias para iniciar o desenvolvimento da solução. Em seguida, descrevem-se os aspectos arquiteturais do GeoDrill, as funções espaciais implementadas e exemplos de consultas espaciais que podem ser executadas através da solução proposta.

4.1 Melhorias na extensibilidade do Apache Drill

Antes de iniciar o desenvolvimento do GeoDrill, foram realizados estudos preliminares de alguns componentes arquiteturais do Apache Drill. Testes utilizando o Drill revelaram algumas limitações em relação à abordagem utilizada para implementação das funções existentes. Esta limitação afeta diretamente o tempo de desenvolvimento de novas funções, bem como o custo de futuras manutenções.

Os problemas identificados são referentes aos tipos de dados que o Drill fornece como parâmetros para suas funções, bem como à hierarquia desses tipos, aos mecanismos de validação dos dados e à sobrecarga na criação das funções. Diante disto, foi desenvolvido um conjunto de classes auxiliares que permitem encapsular os tipos de dados do Drill, visando tão somente resolver estes problemas.

Deste modo, foi criada uma hierarquia de classes que permitiu um mapeamento dos tipos do Drill para os tipos nativos da linguagem utilizada (Java). Esta hierarquia é ilustrada no diagrama de classes na Figura 10, a notação UML adota como referência o livro Learning UML 2.0 [41]. A classe superior da hierarquia é denominada de DataCap, a qual foi definida como uma classe *template*. Este tipo de classe em Java

pode ser implementada utilizando o conceito de *Generics* [23]. Ao utilizar *Generics* é possível desenvolver todos os métodos da classe de forma dinâmica. Assim, ao instanciar ou estender a classe genérica é possível definir o tipo de objeto a ser manipulado e, a partir daí, tanto os métodos quantos os atributos dessa classe serão manipulados como o tipo do objeto definido. Um exemplo comum da utilização do *Generics* é uma lista de elementos em Java. Ao instanciar uma lista é possível informar a classe dos objetos que a mesma deve armazenar; assim os métodos de adicionar objetos à lista podem verificar o tipo a ser adicionado. O retorno dos métodos também pode ser parametrizado com *Generics*, possibilitando retornar o tipo específico e não o tipo mais abrangente (*Object*). Seguindo essa abordagem, a classe DataCap foi definida com dois parâmetros genéricos: Y, relacionado ao atributo data, referente a um objeto da API do Drill; e T, utilizado para definir o tipo de saída do método *getValue* e determinar o tipo do parâmetro de entrada do método *setValue*. Nas implementações, o parâmetro T é referente aos tipos Java, como Integer, Double, String, Boolean e byte[].

A motivação para a criação desta hierarquia surgiu em razão da tecnologia utilizada possuir vários tipos de dados que referenciam um único tipo na linguagem Java. A exemplo disto, o Drill possui os tipos IntHolder e NullableIntHolder, que correspondem a inteiros em Java. Porém, no Drill, o segundo é utilizado para dados que podem ser nulos. Este problema pode ocasionar o uso demasiado de sobrecarga de funções para acomodar os diferentes tipos de parâmetros possíveis. Este uso de sobrecarga pode ser observado, inclusive, nas funções integrantes do Drill. Por exemplo, o Código Fonte 9 foi copiado do repositório do Drill no GitHub[24], este código foi simplificado, removendo-se o corpo da função, pois refere-se aos aspectos da implementação que não são relevantes para esta análise. Neste código fonte, é apresentada a função de concatenação de cadeia de caracteres. No exemplo, são apresentadas duas implementações, no repositório existem quatro, que diferem no que tange os parâmetros de entrada (linhas 4, 5, 22 e 23), pois não há modificação no corpo da função.

[23] Generics - https://docs.oracle.com/javase/tutorial/java/generics/index.html
[24] Exemplo de funções para String desenvolvidas no Apache Drill - https://github.com/apache/drill/blob/master/exec/java-exec/src/main/java/org/apache/drill/exec/expr/fn/impl/StringFunctions.java

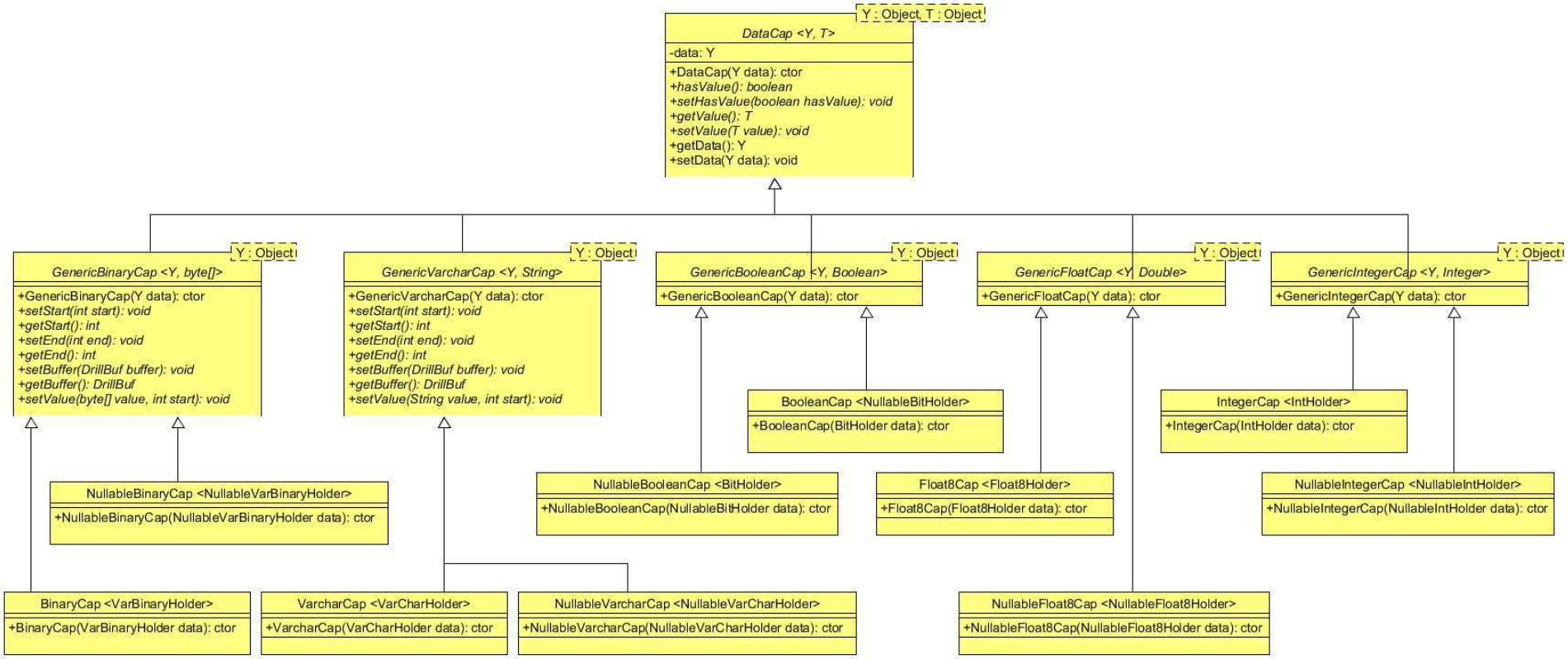


Figura 10: Diagrama da hierarquia de classes DataCap

Código Fonte 9: Exemplo de função de concatenação com sobrecarga.

```
1: @FunctionTemplate(name = "concat", scope = FunctionScope.SIMPLE, nulls =
2: NullHandling.INTERNAL)
3: public static class Concat implements DrillSimpleFunc {
4:    @Param VarCharHolder left;
5:    @Param VarCharHolder right;
6:    @Output VarCharHolder out;
7:    @Inject DrillBuf buffer;
8:
9:    @Override
10:   public void setup() {
11:   }
12:
13:   @Override
14:   public void eval() {
15:         // Corpo da função
16:   }
17: }
18:
19: @FunctionTemplate(name = "concat", scope = FunctionScope.SIMPLE, nulls =
20: NullHandling.INTERNAL)
21: public static class ConcatRightNullInput implements DrillSimpleFunc {
22:   @Param VarCharHolder left;
23:   @Param NullableVarCharHolder right;
24:   @Output VarCharHolder out;
25:   @Inject DrillBuf buffer;
26:
27:   @Override
28:   public void setup() {
29:   }
30:
31:   @Override
32:   public void eval() {
33:         // Corpo da função idêntico ao da função anterior
34:   }
35: }
```

Com a modificação realizada no GeoDrill, foi possível criar assinaturas únicas para determinadas funções, evitando sobrecarga. Além disso, torna-se possível adicionar, caso necessário, métodos específicos para cada tipo. Por exemplo, o tipo GeoDrill GenericVarcharCap possui um método para retornar o *buffer* referente ao tipo VarCharHolder, o qual é encapsulado. As classes instanciáveis foram criadas seguindo os mapeamentos exibidos no Quadro 3. Nesta tabela, a primeira coluna é referente ao tipo do Drill. A segunda coluna representa o tipo de DataCap específico para o qual é mapeado na nova classe DataCap do GeoDrill. Por fim, a terceira coluna representa o tipo de dados na linguagem Java.

Quadro 3: Mapeamento de tipos do Apache Drill para DataCap e Java

Tipo no Apache Drill	Classe GeoDrill DataCap	Tipo no Java
VarBinaryHolder	BinaryCap	byte[]
NullableVarBinaryHolder	NullableBinaryCap	byte[]
BitHolder	BooleanCap	Boolean
NullableBitHolder	NullableBooleanCap	Boolean
Float8Holder	Float8Cap	Double
NullableFloat8Holder	NullableFloat8Cap	Double
IntHolder	IntegerCap	Integer
NullableIntHolder	NullableIntegerCap	Integer
VarCharHolder	VarcharCap	String
NullableVarCharHolder	NullableVarcharCap	String

Antes de iniciar a implementação das funções espaciais no GeoDrill, foi necessário estruturar o processo de encapsulamento dos tipos do Drill. Com este fim, foi criada uma classe que possibilita o encapsulamento do tipo do Drill em uma classe DataCap específica. Esta classe, que implementa o padrão de projeto *factory*, denomina-se DataCapFactory. Esta classe é exibida no diagrama da Figura 11, contendo o método *createDataCap*, que retorna um DataCap caso o objeto de entrada seja um dos tipos do Drill (caso contrário, não retorna nada).

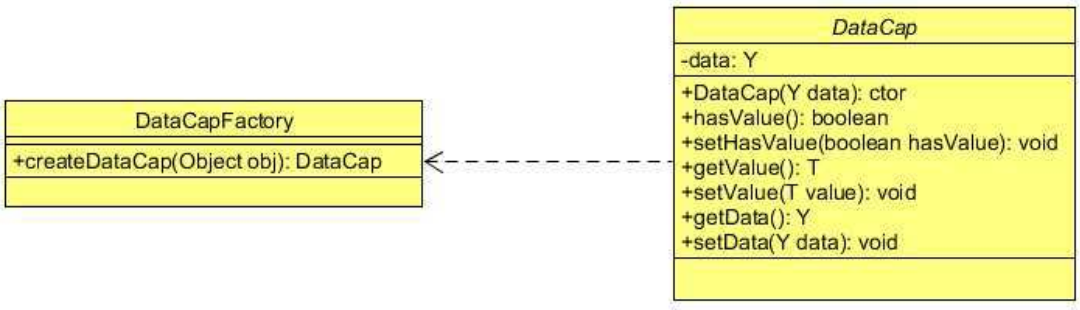


Figura 11: DataCapFactory

A DataCapFactory age como um mecanismo facilitador para o desenvolvimento de funções. Assim, as fachadas das funções tornam-se independentes do tipo de dado esperado como parâmetro, bem como não há dependência em relação ao uso de parâmetros nulos.

4.2 GeoDrill: Representação dos dados espaciais

O Drill permite que seus usuários criem funções e as adicionem ao sistema, desde que alguns padrões de criação sejam seguidos. No entanto, não é possível criar novos tipos de dados, embora isto pudesse ser contornado, por exemplo, com a criação de um mecanismo que permitisse aos usuários mapear um tipo recentemente criado para cada fonte de dados acessível pela ferramenta. Devido a esta limitação, não é possível utilizar uma representação própria para os dados espaciais. Considerando também a possibilidade de leitura dos dados de vários tipos de fontes, como arquivos CSV ou JSON, MongoDB, dentre outras, é necessário definir uma representação de dados espacial que possa ser utilizada como parâmetro das funções e que seja difundida na comunidade.

Diante essas limitações, foram considerados dois tipos de representações espaciais para serem utilizados: *Well-Known Text* (WKT) e *Well-Known Binary* (WKB). Estes tipos de representação são reconhecidos pela ampla maioria dos SGBD espaciais e plataformas de SIG, sendo representados em arquivos textuais. Selecionou-se o tipo WKB para utilização como parâmetro das funções espaciais, pois esta representação requer menos espaço, bem como possui SRID embutido. Por outro lado, o tipo WKT foi utilizado para o armazenamento em arquivos (como CSV ou JSON), pois esta representação é mais simples, possibilitando que o usuário possa interpretar e alterar a informação, se necessário.

Uma vez que as bases de dados espaciais são frequentemente disponibilizadas utilizando-se a representação WKT, implementou-se na ferramenta GeoDrill a função SQL *ST_GeomFromText*, utilizada para converter o dado de entrada em WKT para a representação espacial utilizada internamente.

No Código Fonte 10 foi exemplificada a utilização das funções espaciais no GeoDrill. Nesta consulta, converte-se uma geometria representada em WKT para a representação espacial utilizada pelo GeoDrill e depois aplica-se uma função espacial (*ST_Buffer*) a essa geometria. É importante observar que, neste caso, onde a conversão dos dados não utiliza nenhum SRID, considera-se o dado de entrada como uma geometria plana, ou seja, as operações são realizadas como em um plano cartesiano. A consulta apresenta o mesmo padrão SQL utilizados nos SGBD espaciais.

```
SELECT ST_Buffer(ST_GeomFromText( p.wkt ), 0.01) FROM Pontos p;
```

Código Fonte 10: Consulta para converter um dado em WKT e criar um buffer nesse dado.

4.3 Arquitetura do GeoDrill

Esta seção apresenta os principais componentes arquiteturais do GeoDrill, bem como alguns dos seus aspectos de implementação. Primeiro é informada a arquitetura do Drill com a adição do GeoDrill. Além disso, descrevem-se ainda aspectos relacionados às contribuições adicionais introduzidas na Seção 4.3.1.

A arquitetura do Drill como a extensão GeoDrill é ilustrada na Figura 12. Nesta figura, percebe-se que o GeoDrill tem como foco estender o mecanismo de execução de consultas do Drill (*Execution Engine*). Por meio da figura, observa-se que o fluxo para execução de uma consulta tem início com sua especificação por um usuário ou por uma aplicação cliente. Conforme discutido no Capítulo 2, um usuário ou aplicação cliente pode-se comunicar com o Drill por meio de vários tipos de interfaces, como, por exemplo: CLI, JDBC/ODBC e *web service*. Quando uma consulta em linguagem SQL é submetida ao sistema, o Drill monta o plano da consulta, definindo as fontes de dados utilizadas e quais porções da consulta serão executadas em cada fonte. Uma vez definido o plano de consulta, o mecanismo de execução de consulta (*Execution Engine*) é acionado. Neste momento, é possível executar as funções espaciais incorporadas pelo GeoDrill. A execução de uma função espacial ocorre após os dados estarem na memória. A execução das funções espaciais será detalhada na Seção 4.4.

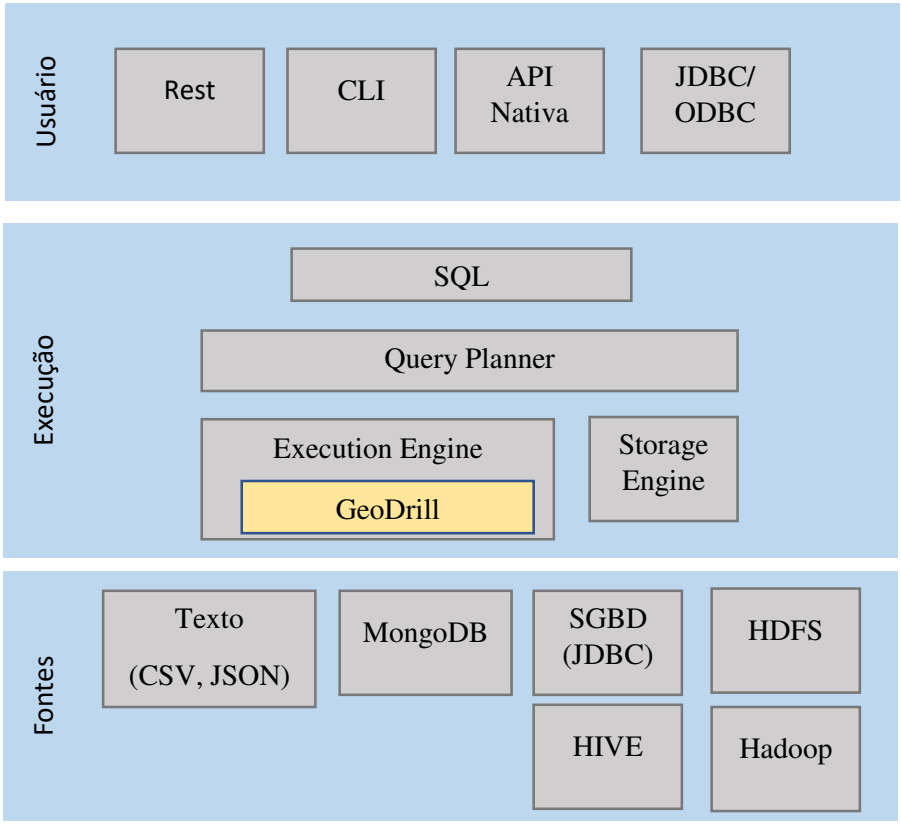


Figura 12: Diagrama da arquitetura do Drill com o GeoDrill

A implementação do GeoDrill é baseada em uma arquitetura em três camadas. A Figura 13 ilustra a interação entre estas camadas. Esta arquitetura contém três camadas: "Fachada da função espacial", "Validação e transformação dos dados" e "Execução da função espacial".

A primeira camada (Fachada da função espacial) é onde estão implementadas as funções espaciais, considerando-se os objetos e assinaturas das interfaces definidas pela API do Drill. Basicamente, a fachada da função espacial define o nome, os parâmetros de entrada e saída da função. Adicionalmente, esta camada se responsabiliza por encapsular os parâmetros definidos na classe GeoDrill DataCap, executar a chamada da implementação da função na segunda camada e tratar o retorno, quando necessário.

A segunda camada (Validação e transformação dos dados) é responsável por receber os parâmetros de entrada encapsulados, converter os dados em geometrias e invocar as funções disponibilizadas pela próxima camada. Esta camada valida os parâmetros recebidos, verificando a consistência de seus tipos e se os mesmos possuem ou não valores. Após as validações, extraem-se os dados destes parâmetros e, caso necessário, convertem-se os dados WKB em objetos do tipo Geometry. Por fim, invoca-se a função espacial da terceira camada e trata-se o retorno dessa função.

A terceira camada (Execução da função espacial), por sua vez, recebe os parâmetros referentes aos dados espaciais convertidos no tipo Geometry, da API espacial selecionada (GeoTools), e utiliza as funções específicas dessa API. No entanto, nem sempre a API espacial utilizada dispõe de métodos com semântica equivalente às funções disponíveis na primeira camada. Com isso, pode ser necessário fazer ajustes ou implementar novas funções nesta última camada, conforme discutido em mais detalhes na Seção 4.3.3.

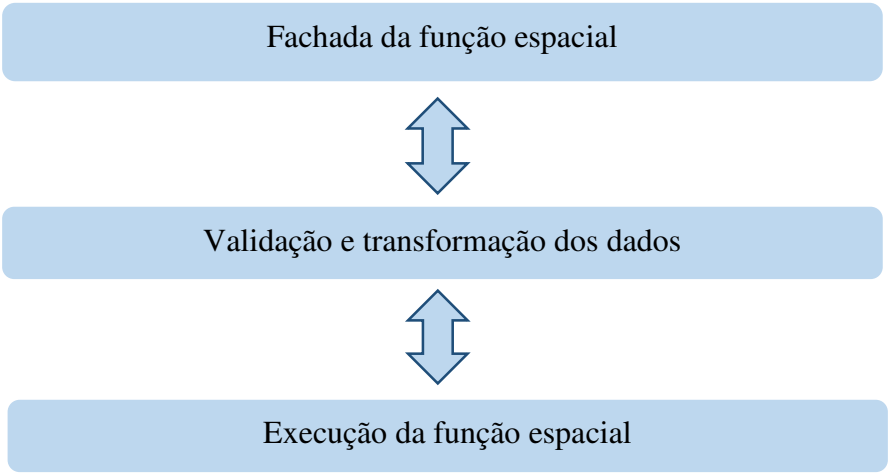


Figura 13: Arquitetura do GeoDrill

No diagrama de sequência da Figura 14, é possível visualizar uma ilustração da interação entre as camadas do GeoDrill, desde o momento da requisição da função pelo usuário ou aplicação cliente. O fluxo é iniciado com a requisição da função espacial. Esta requisição é realizada na sua fachada, camada na qual os parâmetros de entrada e saída são definidos, encapsulados e passados na requisição para a segunda camada. Na segunda camada, os dados são validados e convertidos nos tipos específicos como por exemplo um inteiro ou uma geometria. Por fim, é feita a requisição para a terceira camada, que executa a função espacial.

Nas próximas subseções, estas camadas serão detalhadas em relação aos aspectos de implementação, bem como será descrito em mais detalhes o mecanismo de comunicação entre camadas.

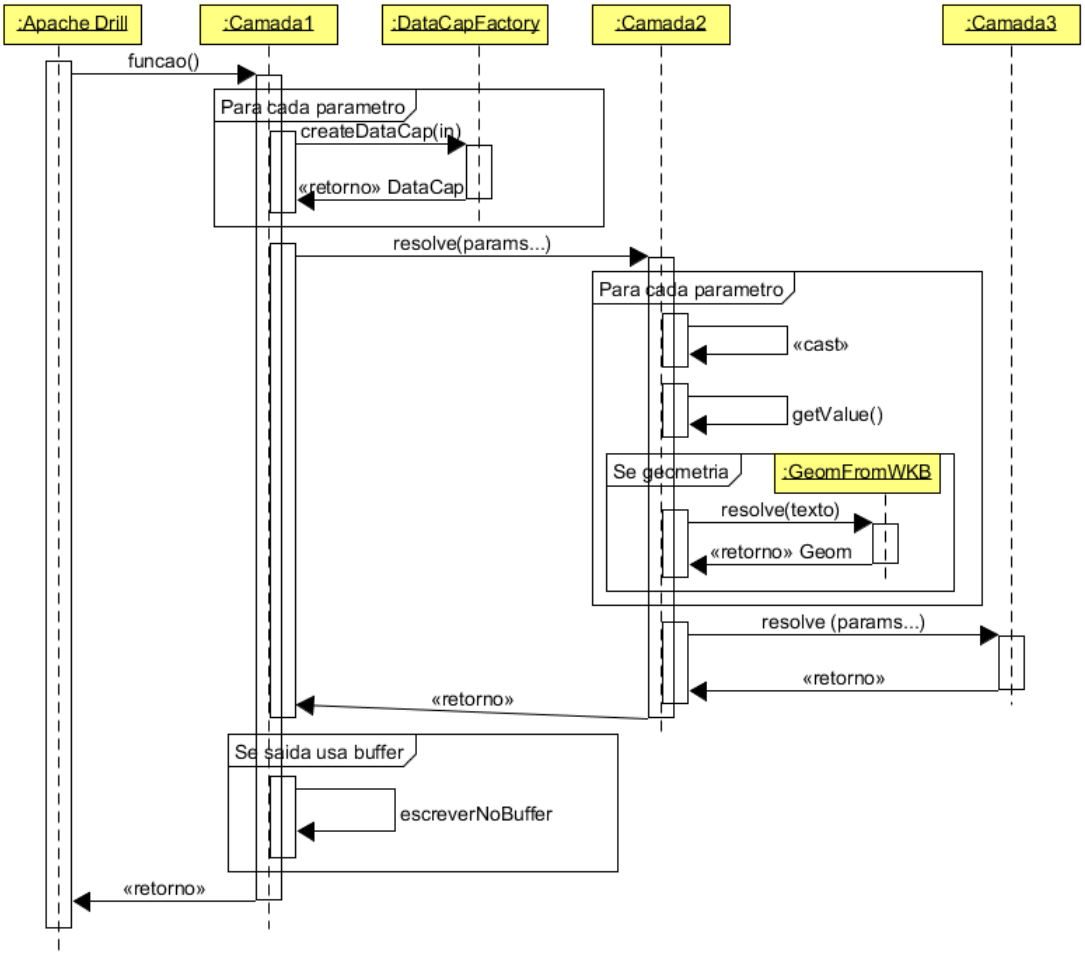


Figura 14: Diagrama de sequências, interação entre as camadas.

4.3.1 Fachada da função espacial

Esta camada define a assinatura das funções, as quais são criadas como classes Java que implementam a interface DrillSimpleFunc da API do Apache Drill, conforme exemplificado no Código Fonte 11 para a função *ST_Buffer*, o nome da função é definido na anotação na primeira linha do código. Os parâmetros da função são os atributos da classe, utilizando anotações específicas para definição do tipo de atributo, como atributo de entrada ou saída. No exemplo dado, é possível verificar os parâmetros da função, os quais são a geometria, o tamanho do *buffer* e o tipo de retorno. Estes parâmetros foram definidos das linhas 6 a 15.

Para os casos de sobrecarga da função, é necessário criar uma nova classe, especificando o novo conjunto de parâmetros utilizados, porém mantendo o mesmo nome da função. Esta classe contém um método *eval()* que representa o corpo da função (linhas 20 a 37). Neste método, inicialmente, cada parâmetro é convertido em um objeto GeoDrill DataCap. No exemplo do Código Fonte 11, ilustra-se a chamada realizada à classe GeoDrill DataCapFactory para encapsular os parâmetros de entrada e saída (linhas 21 a 27).

Depois, é realizada a chamada à classe, na camada seguinte, para resolução da função (linhas 29 e 30). Por fim, se necessário, é realizada a escrita no *buffer* do parâmetro de saída, quando o dado é um WKB ou um texto. No exemplo do Código Fonte 11, é apresentada a chamada para a implementação na segunda camada, passando os parâmetros encapsulados e retornando uma cadeia de bytes, que são gravados no *buffer* do parâmetro de saída (linhas 32 a 36).

Código Fonte 11: Exemplo de implementação da primeira camada

```
1: @FunctionTemplate(name = "ST_Buffer", scope =
2: FunctionTemplate.FunctionScope.SIMPLE, nulls =
3: FunctionTemplate.NullHandling.NULL_IF_NULL)
4: public static class ST_Buffer1 implements DrillSimpleFunc {
5:
6:    @Param
7:    VarBinaryHolder geom;
8:    @Param
9:    Float8Holder distance;
10:
11:   @Output
12:   VarBinaryHolder output;
13:
14:   @Inject
15:   DrillBuf buffer;
16:
17:   public void setup() {
18:   }
19:
20:   public void eval() {
21:         DataCap geomCap = DataCapFactory
22:                           .createDataCap(geom);
23:         DataCap distanceCap = DataCapFactory
24:                           .createDataCap(distance);
25:
26:         DataCap outputCap = DataCapFactory
27:                           .createDataCap(output);
28:
29:         byte[] resolve = new ST_BufferImpl().resolve(geomCap,
30:  distanceCap);
31:
32:         if( resolve != null){
33:               buffer = buffer.reallocIfNeeded(resolve.length);
34:         }
35:
36:         Utils.writeBytesBinaryBuffer(outputCap, buffer, resolve,0);
37:   }
38: }
```

Nos casos em que os dados de retorno são WKB ou texto, é realizada uma realocação do tamanho do *buffer* do parâmetro de saída, de acordo com a quantidade de bytes do retorno. Devido ao modo que o Drill realiza a importação destas funções, a realocação do *buffer* precisa ser realizada nesta camada.

Todas as funções espaciais desenvolvidas foram baseadas nas existentes no padrão SQL *Spatial*. As funções que possibilitam converter geometrias são *ST_AsText, ST_GeomFromText* e *ST_Point*. As funções que permitem avaliar se duas geometrias possuem uma relação são *ST_Contains, ST_ContainsProperly, ST_CoveredBy, ST_Covers, ST_Crosses, ST_Disjoint, ST_DWithin, ST_Equals, ST_Intersects, ST_OrderingEquals, ST_Overlaps, ST_Relate, ST_Touches, ST_Within* e *ST_Distance*. As funções que retornam uma condição ou propriedade da geometria são *ST_IsEmpty, ST_IsRectangle, ST_IsSimple, ST_IsValid, ST_Area, ST_Dimension, ST_GeometryType, ST_Length, ST_M, ST_NumGeometries, ST_NumInteriorRings, ST_NumPoints, ST_SRID, ST_X, ST_Y* e *ST_Z*. Por fim, as funções que retornam novas geometrias a partir de outras são *ST_Boundary, ST_Buffer, ST_Centroid, ST_ConvexHull, ST_Difference, ST_EndPoint, ST_Envelope, ST_ExteriorRing, ST_GeometryN, ST_InteriorRingN, ST_Intersection, ST_PointN, ST_PointOnSurface, ST_Shift_Longitude, ST_StartPoint, ST_SymDifference* e *ST_Union*.

4.3.2 Validação e transformação dos dados

Nesta segunda camada da arquitetura do GeoDrill há uma maior flexibilidade para implementação das funções, uma vez que não há restrições de codificação impostas pela API do Drill. Por exemplo, a referência às classes por seus nomes canônicos (composto pelo nome do pacote mais nome da classe). Além disso, esta camada permite utilizar classes genéricas e interfaces (não implementadas pelo Drill), que permitem modularizar e generalizar a implementação das várias funções espaciais.

O principal objetivo dessa camada é tratar os objetos DataCap. Assim, foram criadas algumas classes genéricas para simplificar o desenvolvimento das funções. Essas classes foram agrupadas de acordo com os tipos das funções e seus retornos:

- **EvaluateGeomToBoolean**: avalia se uma geometria possui uma determinada característica (a qual é definida pela função espacial). Por exemplo, pode-se avaliar se a geometria é retangular (i.e., *ST_IsRectangle*).
- **EvaluateGeomsWithPropertyToBoolean**: avalia se duas geometrias satisfazem uma relação com uma propriedade (esta propriedade é um parâmetro da função). Por exemplo, pode-se avaliar se duas geometrias possuem uma distância relativa mínima uma da outra (i.e., *ST_DWithin*).
- **EvaluateGeomsToBoolean**: avalia se duas geometrias satisfazem uma relação espacial, como, por exemplo, a relação de interseção (i.e., *ST_Intersects*).

- **EvaluateGeomsToGeom**: retorna uma geometria resultante de uma operação espacial entre duas geometrias, como, por exemplo, uma união de duas geometrias (i.e., *ST_Union*).
- **EvaluateGeomToGeom**: retorna uma geometria resultante de uma operação específica implementada pela função, como, por exemplo, recuperar o ponto central da geometria (i.e., *ST_Centroid*).
- **EvaluateGeomToGeomWithProperty**: retorna uma geometria resultante de uma operação específica implementada pela função. Porém, neste caso, a função utiliza um parâmetro adicional não espacial. Por exemplo, pode-se retornar o enésimo ponto de uma geometria (i.e., *ST_PointN*).
- **GetGeomProperty**: retorna uma propriedade da geometria, como, por exemplo, o identificador do sistema de referência espacial adotado (i.e., *ST_SRID*)
- **ResolveGeomsRelation**: retorna uma propriedade não espacial calculada entre duas geometrias, como, por exemplo, a distância entre duas geometrias (i.e., *ST_Distance*).

Com a utilização destas classes não é mais necessário se preocupar com todas as etapas do desenvolvimento, desde a validação até a conversão dos dados, para a construção de uma função espacial. Em linhas gerais, as funções espaciais precisam somente herdar de uma dessas classes genéricas e implementar o método que realiza a chamada à classe da terceira camada, conforme ilustrado no exemplo do Código Fonte 12, pelo método *evaluateMethod* nas linhas 27 e 28.

Para todas as funções, são realizadas validações básicas para verificação de atributos nulos e de tipo nas conversões dos objetos GeoDrill DataCap, estas validações podem ser observadas no Código Fonte 12 nas linhas 4 a 11. Cada classe genérica possui um conjunto de validações padrão, mas é possível adicionar novas validações sem a necessidade de efetuar muitas modificações no código.

Após a etapa de validação, todos os parâmetros referentes às geometrias são convertidos de WKB para o objeto do tipo Geometry, estas conversões podem ser observadas das linhas 13 a 16 do Código Fonte 12. Os demais parâmetros são recuperados de seus respectivos objetos DataCap. Após esta recuperação dos dados, é realizada a chamada da função espacial correspondente na terceira camada.

Com o retorno da função espacial, se o tipo não for Geometry ou texto, é possível gravar a informação diretamente no parâmetro de saída, o qual está encapsulado em um objeto

DataCap, utilizando o método específico para guardar a informação. No caso do tipo geométrico, é realizada a conversão para WKB, esta etapa de conversão pode ser observada no método *evaluate* nas linhas 21 a 25 do Código Fonte 12. Por fim, o WKB ou os dados textuais são retornados para a primeira camada.

Código Fonte 12: Exemplo de classe genérica da segunda camada

```
1: public abstract class EvaluateGeomsToGeom {
2:    public byte[] resolve(DataCap<?,?> inputCap1, DataCap<?,?>
3: inputCap2) {
4:          if(!inputCap1.hasValue() || !inputCap2.hasValue()) {
5:                return null;
6:          }
7:
8:          GenericBinaryCap<?> inputBinary1 = (GenericBinaryCap<?>)
9: inputCap1;
10:         GenericBinaryCap<?> inputBinary2 = (GenericBinaryCap<?>)
11: inputCap2;
12:
13:         Geometry geom1 =
14: ST_GeomFromWKB.resolve(inputBinary1.getValue());
15:         Geometry geom2 =
16: ST_GeomFromWKB.resolve(inputBinary2.getValue());
17:
18:         return evaluate(geom1, geom2);
19:   }
20:
21:   private byte[] evaluate(Geometry geom1, Geometry geom2) {
22:         Geometry geomt = evaluateMethod(geom1, geom2);
23:
24:         return Utils.getGeomBytes(geomt);
25:   }
26:
27:   public abstract Geometry evaluateMethod(Geometry geom1, Geometry
28: geom2);
29: }
```

4.3.3 Execução da função espacial

Esta camada é responsável pela tarefa mais importante, que é a implementação da função espacial propriamente dita. Nesta camada, é realizada a requisição ao método específico da API espacial utilizada. Caso a API espacial não contenha o método necessário, é possível criar essa implementação com a composição de outros métodos espaciais ou criar uma implementação própria nesta camada.

Um exemplo de função espacial composta é a *ST_ContainsProperly*, a qual indica se uma geometria está contida em outra, mas não há interseção entre as bordas das geometrias. Como esta função não existe na API utilizada (GeoTools [42]), esta foi implementada utilizando outras três funções espaciais disponíveis. Para isto, utilizou-se uma função para recuperar a borda da primeira geometria (função getBoundary()); verificou-se se a segunda

geometria fazia interseção com a borda (função intersects()); e, por fim, verificou-se o caso da primeira geometria conter a segunda (função contains()).

As funções espaciais podem ser divididas pelos seus tipos de retorno. Por exemplo, algumas funções retornam geometrias (e.g., *ST_Intersection*), as quais são resultantes da operação da função. Por outro lado, outras funções retornam literais, os quais podem corresponder à propriedade(s) de uma geometria ou à verificação se duas geometrias satisfazem ou não uma determinada relação. Os tipos mais comuns de retorno são Geometry e Boolean.

A implementação desta camada teve como base a API GeoTools [42], que possui um conjunto de funções espaciais bem completo.

Para algumas funções, foi necessário um desenvolvimento adicional. Por exemplo, para implementar a função *ST_PointOnSurface*, foi necessário verificar primeiro o tipo geométrico. Se o tipo é um ponto, a função retorna o próprio ponto diretamente; se o tipo é uma linha, a função retorna um ponto randômico; se o tipo é um polígono a função retorna um ponto aleatório de seu anel exterior. Esse exemplo pode ser visualizado no Código Fonte 13. Por fim, o retorno da função é fornecido diretamente para a segunda camada após o fim da execução do método desta camada.

Código Fonte 13: Exemplo de implementação de função da terceira camada

```
1: public class ST_PointOnSurface{
2:
3:    public static Point resolve(Geometry geom1) {
4:          if(geom1 instanceof Point) {
5:                return (Point)geom1;
6:          }
7:
8:          if(geom1 instanceof LineString) {
9:                return getPointOnSurface((LineString)geom1);
10:         }
11:
12:         if(geom1 instanceof Polygon) {
13:               return getPointOnSurface((Polygon)geom1);
14:         }
15:
16:         return null;
17:   }
18:
19:   public static Point getPointOnSurface(Polygon p) {
20:         LineString exteriorRing = p.getExteriorRing();
21:         return getPointOnSurface(exteriorRing);
22:   }
23:
24:   public static Point getPointOnSurface(LineString l) {
25:         int numPoints = l.getNumPoints();
26:         int np = (int) (Math.random() * numPoints);
27:
28:         return l.getPointN(np);
29:   }
30: }
```

4.4 Exemplos de consultas

Nesta seção, são exemplificadas as sequências de execução de três tipos de funções espaciais:

- *ST_GeomFromText*, que recebe uma geometria representada através de um dado convencional (texto) e retorna um dado espacial;
- *ST_Union*, que recebe duas geometrias e retorna a união destas;
- *ST_Intersects*, que recebe duas geometrias e retorna um valor booleano, caso as geometrias possuam alguma interseção.

Estas funções foram selecionadas pois são frequentemente utilizadas em várias consultas para análise de dados espaciais. Os parâmetros de entrada e saída utilizados por estas funções são similares àqueles utilizados pela maioria das funções implementadas. Enquanto a primeira função é utilizada para converter dados textuais em dados espaciais, a segunda permite unir geometrias, como, por exemplo, para gerar um estado a partir das geometrias de seus municípios. A terceira função, por sua vez, é normalmente utilizada como condição de junção entre tabelas contendo dados espaciais, por retornar um dado booleano.

O diagrama de sequência da Figura 15 ilustra a execução de uma chamada à função *ST_GeomFromText*. A consulta que exemplifica a requisição é apresentada no topo da figura. O fluxo é iniciado quando o GeoDrill requisita a função espacial, passando os dados nos parâmetros de entrada. Esta requisição é realizada na fachada da função (primeira camada). Depois de encapsular os dados de entrada, é realizada a requisição para a segunda camada. Esta função tem um modelo diferente das demais funções desenvolvidas, pois recebe um WKT e retorna um WKB, enquanto as demais funções recebem pelo menos um WKB como parâmetro.

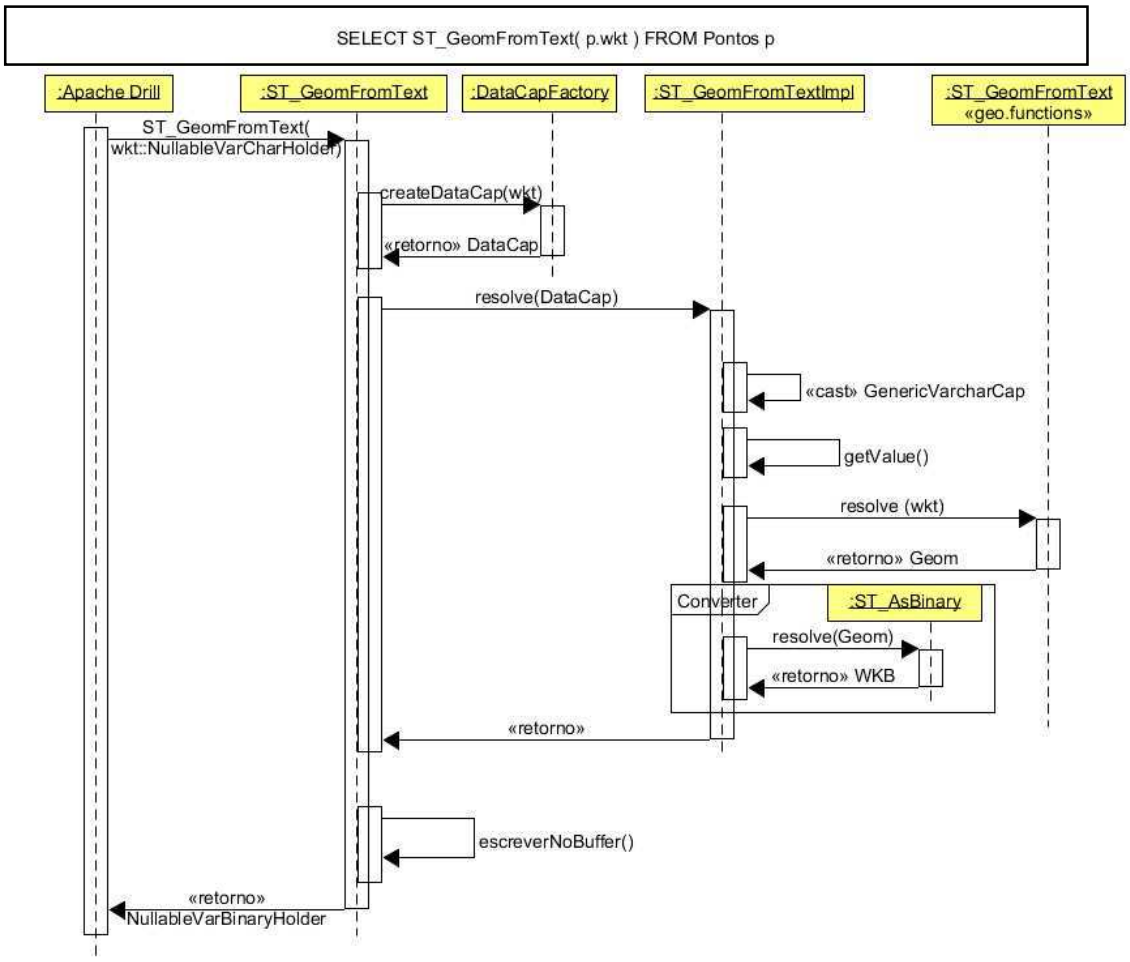


Figura 15: Diagrama de sequência da função ST_GeomFromText

A execução da função *ST_Union* é ilustrada no diagrama de sequência da Figura 16. O exemplo de consulta SQL que dispara a execução desta função é exibido no topo da figura. A diferença principal para essa função, em relação à anterior, é que esta recebe dois parâmetros referentes às geometrias, dados em WKB, os quais são encapsulados em objetos DataCap e posteriormente na segunda camada os dados (WKB) são convertidos em objetos do tipo Geometry, por fim é realizada a operação de união dessas geometrias. Esta função espacial é

uma, dentre as várias, que retorna uma geometria, a qual é convertida em WKB que, por sua vez, é salvo no parâmetro de saída da função. Posteriormente, este parâmetro de saída é resgatado pelo GeoDrill.

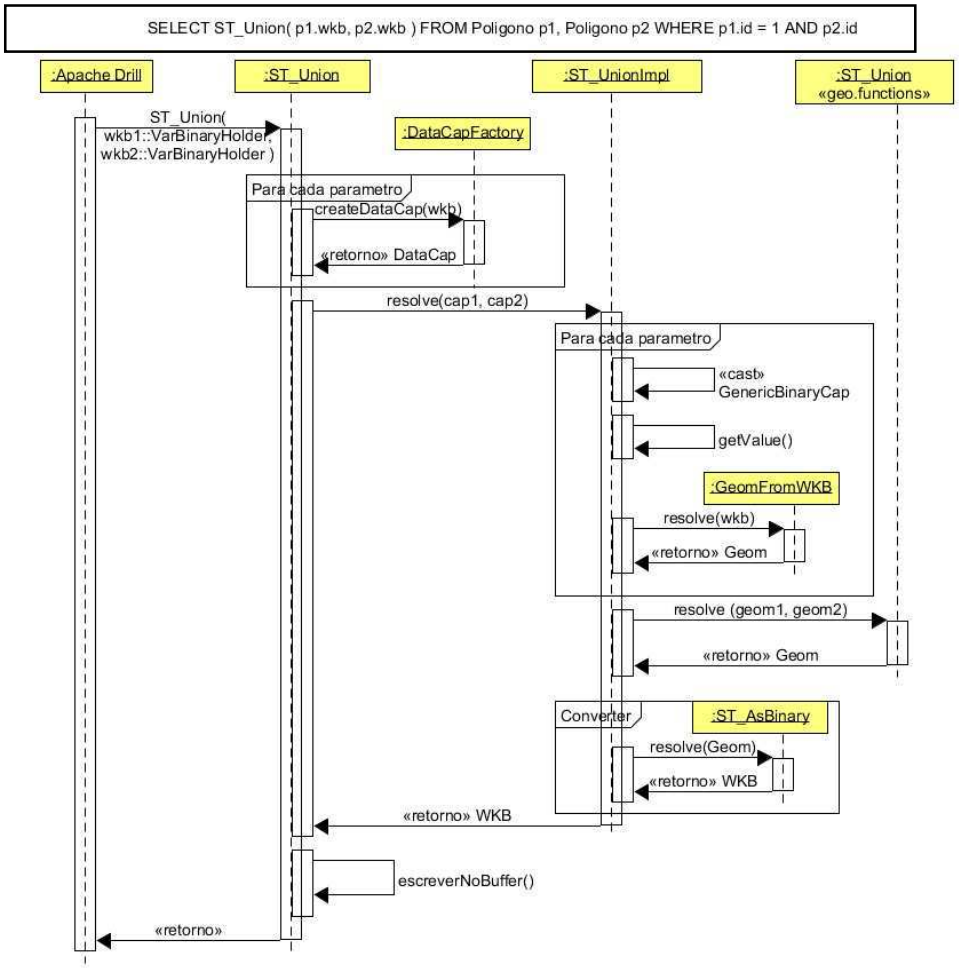


Figura 16: Diagrama de sequência da função ST_Union

A sequência de execução e a consulta que exemplifica a utilização da função espacial *ST_Intersects* podem ser visualizadas na Figura 17. Esta função é frequentemente utilizada como condição de junção em consultas que requerem junções espaciais. Esta função é semelhante à função anterior por requisitar dois parâmetros referentes a dados espaciais, no entanto, retorna um booleano, o qual indica se a duas geometrias intersectam-se espacialmente. Diferente das demais consultas, a requisição realizada entre a "Fachada da função" e a camada de "Validação e transformação dos dados" passa como parâmetros tanto os objetos de entrada quanto os de saída da função. Após a execução da função, o dado booleano de retorno é salvo no parâmetro de saída, o qual está encapsulado em um objeto DataCap.

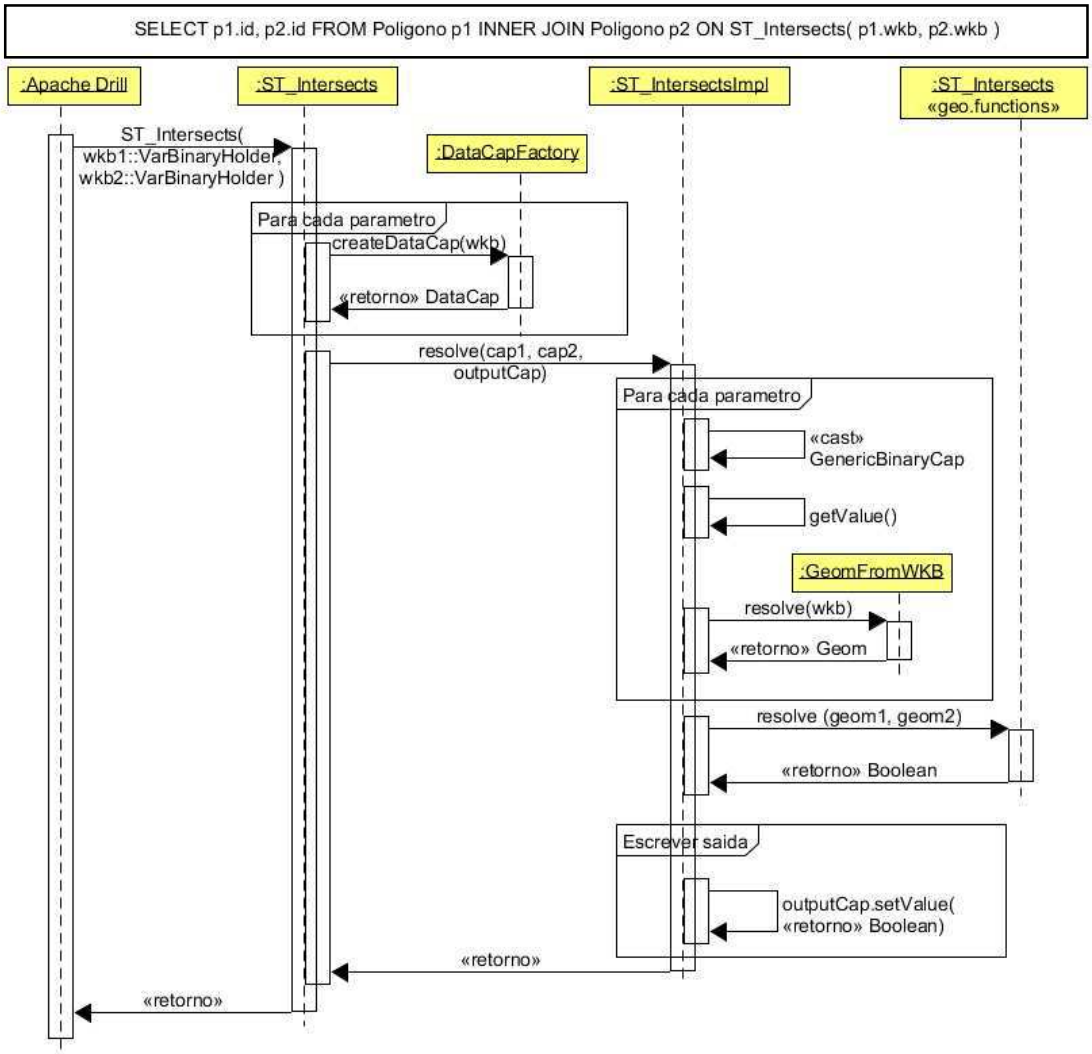


Figura 17: Diagrama de sequência da função ST_Intersects

4.5 Comparativo de soluções existentes

Nesta seção é realizado um comparativo com soluções existentes que lidam com dados espaciais e que possibilitem meios de consultar dados semiestruturados. Essas soluções não permitem necessariamente integrar fontes heterogêneas, mas são soluções extremamente difundidas na comunidade e utilizadas há bastante tempo.

A comparação do GeoDrill com as outras soluções apontadas no Capítulo 3, pode ser visualizada no Quadro 4, este quadro e suas características são as mesmas adotadas na Seção 3.3, atualizado com a adição da coluna GeoDrill.

Quadro 4: Comparativo do GeoDrill com outras soluções de integração de dados

	Dremel	**Apache Drill**	**Apache Spark**	**Spark SQL**	**Hadoop GIS**	**SpatialHadoop**	**Hbasespatial**	**GeoSpark**	**Simba**	**GeoDrill**
Linguagem de consulta	Baseada em SQL	SQL	Própria	SQL	Baseada em SQL	Baseada em SQL	Baseada em SQL	Própria	SQL	SQL
Middleware	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Extensível	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Dados semiestruturados	Sim	Sim	Sim	Sim	Não	Não	Sim	Sim	Sim	Sim
Integra fontes de dados distintas	Não	Sim	Sim	Sim	Não	Não	Não	Sim	Sim	Sim
Suporte espacial	Não	Não	Não	Não	Básico	Completo	Básico	Básico	Básico	Completo
Possibilidade de utilizar em ambiente convencional	Não	Sim	Não	Não	Não	Não	Não	Não	Não	Sim

4.6 Conclusão

O Apache Drill é uma ótima solução no quesito de integração de dados, possuindo meios de explorar e cruzar dados de fontes distintas. O GeoDrill enriquece o conjunto de funcionalidades dessa solução possibilitando a exploração e consulta a dados espaciais. O GeoDrill contribui não só no quesito de trazer novas funcionalidades, mas também meios de melhorar a extensibilidade do Apache Drill.

Dentre as soluções avaliadas, somente SpatialHadoop possui suporte espacial completo, tanto em relação aos dados possíveis de representar quanto ao conjunto de funções espaciais.

No entanto, este não possui suporte a dados semiestruturados. Devido a suas características, o SpatialHadoop é uma solução para processamento de consultas em *batch*. Para integrar dados faz-se necessário utilizar um DW. O SpatialHadoop não possui meios de carregar os dados de outras fontes distintas. A linguagem de consulta é o Pigeon, ou seja, não é utilizado o padrão SQL.

O Hive, por sua vez, é outra solução NoSQL, que funciona utilizando o Hadoop[25], e é voltado para consultas em batch. A linguagem de consulta empregada é muito próxima do SQL padrão. O Hive possui suporte a dados semiestruturados, mas não possui nenhum recurso sintático que simplifique a consulta aos dados semiestruturados. Existe uma API espacial para o Hive, a qual é desenvolvida pela ESRI[26], com suporte completo aos vários tipos espaciais. No entanto, ao utilizar essa API, foi possível verificar erros na conversão de dados, dependendo da projeção espacial utilizada. Além disso, esta tecnologia não provê suporte para utilização de índices espaciais.

Por fim, o Apache Spark é uma solução para consultar e processar informações. Este possui uma extensão que possibilita realizar consultas SQL, acessar fontes externas e dados semiestruturados. É possível realizar consultas utilizando SQL, com algumas limitações, que requerem a necessidade de utilizar uma linguagem procedural junto com a consulta, semelhante à utilização de uma linguagem como PL/SQL. As extensões espaciais existentes não seguem por completo o padrão SQL espacial definido pelo OGC[27]. Além disso, esta tecnologia não provê suporte para utilização de índices espaciais. Outra limitação é em relação à forma de acesso, que se dá apenas via API nativa.

O GeoDrill se destaca entre estas soluções por permitir, por meio de linguagem SQL padrão, explorar dados semiestruturados sem necessidade de se efetuar uma carga destes dados. Além disso, permite cruzar dados de várias fontes distintas, utilizando processamento em memória e, desta forma, tem um bom desempenho para processar arquivos e dados de fontes externas, não necessitando realizar nenhum processo de extração e carga de dados, desta forma trabalhando sempre com a informação atualizada. Por fim, como ponto mais importante, o GeoDrill permite explorar as informações espaciais dentre os dados destas fontes distintas já citadas, podendo, por exemplo, cruzar dados de um CSV contendo pontos e polígonos com uma base de dados armazenada no PostgreSQL ou MongoDB.

[25] Hadoop - http://hadoop.apache.org/
[26] ESRI - http://www.esri.com/
[27] OGC - http://www.opengeospatial.org/

O GeoDrill ainda possui algumas limitações em comparação às demais tecnologias apresentadas. Dentre as limitações estão a criação e utilização dos índices espaciais pelas consultas. A criação de índices espaciais possibilita ganho de desempenho nas consultas espaciais. Pensa-se em estratégias para trabalhos futuros de como integrar a criação de diferentes tipos de índices e sua utilização nas consultas, utilizando as funções espaciais, de modo a utilizar uma criação de índices dinâmicos.

Capítulo 5 – Validação Experimental do GeoDrill

Este capítulo descreve uma avaliação experimental que tem como principal objetivo responder às seguintes questões de pesquisa:

1. É possível executar consultas espaciais em linguagem SQL no GeoDrill que permita a integração de dados espaciais de fontes heterogêneas?
2. Existe diferença no desempenho das consultas realizadas no GeoDrill, dependendo do tipo de fonte de dados utilizado?
3. O desempenho do GeoDrill é equivalente ao de um SGDB espacial convencional?

Para avaliar a primeira questão, considerada a mais importante, verifica-se a capacidade de realizar consultas espaciais cruzando tabelas de fontes de dados heterogêneas. Essa avaliação visa também indicar as diferenças entre o GeoDrill e outros trabalhos relacionados à integração de dados espaciais. Para avaliar a segunda questão, realiza-se um comparativo acerca do desempenho das consultas no GeoDrill de acordo com a fonte de dados utilizada. Por fim, realiza-se um comparativo do desempenho do GeoDrill com uma tecnologia de SGBD espacial para avaliar a terceira questão.

Este capítulo é dividido em três seções: a primeira seção descreve os preparativos do experimento, o ambiente utilizado, os dados, as variáveis, os testes e as etapas da validação experimental; a segunda é destinada às etapas do experimento, apresentando os resultados obtidos nas medições e as comparações realizadas; e, por fim, a terceira seção é destinada às discussões deste capítulo.

5.1 O Experimento

O experimento conduzido foi dividido em três etapas. Na primeira etapa, valida-se a capacidade do GeoDrill de realizar a integração de dados espaciais de fontes de dados heterogêneas, que é considerada a principal questão de pesquisa. É necessário que o GeoDrill seja validado na primeira etapa para que sejam realizadas medições adicionais, a fim de avaliar o desempenho das consultas espaciais. Na segunda etapa dos experimentos, a comparação de desempenho foi realizada entre as fontes CSV e MongoDB do GeoDrill. Foram selecionadas estas fontes de dados por apresentarem características bem distintas entre si. Os arquivos em CSV caracterizam dados estruturados comumente gerados por usuários de

planilhas eletrônicas, armazenados em forma de arquivos. Por sua vez, o MongoDB é um sistema NoSQL, que apresenta escalabilidade para armazenamento de coleções de dados semiestruturados. As medições da segunda etapa foram utilizadas para definir a fonte de dados utilizada na comparação de desempenho da próxima etapa. Por fim, na última etapa, foram comparados os desempenhos do GeoDrill e do PostGIS/PostgreSQL.

Este experimento baseou-se em outros trabalhos, como o de Zhang et al. [43] e o de Ray et al. [44], e em um relatório on-line do Instituto Indiano de Tecnologia [45]. Estes trabalhos usam os dados do senso dos EUA, uma base de dados conhecida como TIGER/Line[28]. Os experimentos destes trabalhos utilizam dados de 2009 ou anos anteriores. Para os experimentos aqui descritos, foi recuperada uma base mais atual dos dados TIGER/Line (com dados até o ano de 2014). Foram aproveitados os conjuntos de consultas realizando modificações, caso necessário, em itens como nome de tabelas e colunas, bem como remoção de colunas não existentes.

Adicionalmente, para um contexto considerando consultas no Brasil, foi realizada uma parte do experimento com a base de dados do IBGE[29], a Base Cartográfica Contínua do Brasil ao Milionésimo (*BCIM)*[30], de 2014. Para o caso da base BCIM, foram desenvolvidas novas consultas, que exploram diversos tipos espaciais, funções e junções, pois não foram encontrados *benchmarks* bem avaliados que utilizam essa base de dados.

Na primeira etapa do experimento, para avaliar a interoperabilidade do GeoDrill, os dados espaciais foram armazenados em arquivos CSV e no MongoDB. Por exemplo, em uma das consultas propostas, a primeira tabela é de um CSV e a segunda é do MongoDB.

Na segunda etapa, com o GeoDrill foi realizado o experimento comparativo entre duas fontes de dados distintas, para indicar se há diferença nos tempos de execução dependendo da fonte de armazenamento dos dados e qual destas fontes traz um melhor desempenho para as consultas espaciais no GeoDrill. Estes resultados foram utilizados na seleção da fonte para o comparativo com o PostGIS/PostgreSQL, em seguida.

Por fim, na última etapa do experimento, foram realizadas duas comparações. A primeira do GeoDrill com o PostGIS/PostgreSQL em um ambiente desktop e a segunda em um ambiente distribuído. Desta forma, seria possível verificar o comportamento das duas tecnologias nestes ambientes, comparando o desempenho das mesmas. É importante verificar

[28] TIGER/Line - https://www.census.gov/geo/maps-data/data/tiger-line.html
[29] IBGE - http://www.ibge.gov.br/home/
[30] BCIM - http://downloads.ibge.gov.br/downloads_geociencias.htm

se há uma mudança drástica de desempenho entre as comparações supracitadas, por causa da mudança de ambiente computacional.

As subseções a seguir descrevem as características das bases de dados utilizadas e o ambiente do experimento. Apresenta-se uma descrição geral dos testes que foram realizados, definindo as variáveis nesses testes, e descrevendo as hipóteses observadas.

5.1.1 Descrição do ambiente de teste

Foram definidos dois ambientes de teste para avaliar o GeoDrill. O primeiro ambiente é um desktop, enquanto que o segundo ambiente é distribuído. Para o ambiente desktop, foi utilizada uma máquina com processador Intel core i7 3770 de 3.4 Ghz, 32 GB de memória RAM, HD de 2 TB e sistema operacional Windows 7.

O ambiente distribuído utiliza uma arquitetura recomendada pela documentação do Apache Drill [47], a qual indica um número ímpar de máquinas para execução em ambiente distribuído. Dessa forma, dada as limitações de recursos computacionais para o experimento, foi possível utilizar três máquinas virtuais compondo o ambiente distribuído. Duas destas máquinas possuem processador com 2 cores de 2.6 Ghz, 4 GB de memória RAM e HD de 60 GB. A terceira máquina possui processador com 6 cores de 2.6 Ghz, 8 GB de RAM e HD de 90 GB. As três máquinas executam sistema operacional Ubuntu 14.04.

Foram utilizadas as seguintes versões de software neste experimento: Apache Drill 1.4 com a extensão GeoDrill, proposta nesta dissertação; e MongoDB 3.2, para avaliar a interoperabilidade da solução. Adicionalmente, para o ambiente distribuído, foi utilizado o Apache Zookeeper 3.4.631, para gerenciar a comunicação entre as instâncias do Apache Drill. Para a etapa comparativa do experimento, foi utilizado o PostgreSQL 9.4 com PostGIS/PostgreSQL 2.1.8.

A instalação dos sistemas e configuração do ambiente distribuído foi realizada segundo o recomendado pelo Apache Drill [46] e PostgreSQL [47]. Esta configuração distribuída é recomendada pelo software de gerenciamento, o Zookeeper, o qual utiliza uma máquina como mestre e as demais como escravos. A máquina mestre tem o papel de gerenciar as consultas distribuindo-as entre as escravas e agregando os resultados das escravas.

5.1.2 Dados

Os dados utilizados foram obtidos do Censo Norte-Americano (TIGER/Line) e da Base Cartográfica Contínua do Brasil ao Milionésimo (BCIM) do IBGE. As projeções das bases

[31] Apache Zookeeper - https://zookeeper.apache.org/

TIGER/Line e BCIM possuem SRID 4326 (WGS 84)[32] e 4674 (SIRGAS 2000)[33], respectivamente.

Foram utilizadas oito tabelas da base de dados TIGER/Line. Cada tabela possui uma coluna geometria (linha, polígonos ou ponto); porém, estas são distintas em relação ao número de registros, o espaço ocupado pela tabela e o tipo da coluna geométrica. As tabelas utilizadas, seus índices, o número de registros e o tamanho destas no PostgreSQL e em CSV são apresentados na Tabela 1.

Tabela 1: Dados das tabelas da base TIGER/Line

Tabela	**Índice**	**Nº de registros**	**Tamanho no PostgreSQL**	**Tamanho em CSV**
arealm	arealm_geom_gist	129179	72 MB	156 MB
areawater	areawater_geom_gist	2292810	1338 MB	2.39 GB
linearwater	linearwater_geom_gist	5708959	4262 MB	6.71 GB
pointlm	pointlm_geom_gist	857617	80 MB	82.7 MB
rails	rails_geom_gist	180958	71 MB	100 MB
roads	roads_geom_gist	19592688	4856 MB	10.1 GB
primaryroads	primaryroads_geom_gist	12396	26 MB	53.1 MB
states	state_geom_gist	56	16 KB	17.4 MB

A tabela arealm é referente à localização de marcos como aeroportos, cemitérios, parques e universidades, possuindo onze colunas e geometria de um polígono. A tabela areawater é referente à localização de áreas aquáticas como lagos, lagoas, pântanos oceanos, geleiras e áreas cobertas, também possui onze colunas e geometria de um polígono. A tabela linearwater é outra tabela sobre hidrografia referente a rios, córregos, valas, canais, aquedutos e canais artificiais, possuindo seis colunas e a geometria é uma cadeia de linhas (*Linestring*). A tabela pointlm, que possui 6 colunas, refere-se à localização de marcos, nos quais as geometrias podem ser referenciadas com pontos, como igrejas e escolas. A tabela rails é

[32] WGS 84 - http://spatialreference.org/ref/epsg/wgs-84/
[33] Sirgas 2000 - http://spatialreference.org/ref/epsg/4674/

referente às linhas de ferro, possui quatro colunas e sua geometria é de uma cadeia de linhas. As tabelas roads e primaryroads são referentes às ruas e estradas primárias, respectivamente. Estas duas tabelas possuem o mesmo conjunto de cinco colunas e geometrias de cadeias de linhas. A tabela states refere-se aos estados dos EUA, além dos cinquenta estados são contabilizadas outras regiões, possuindo quinze colunas e geometria do tipo polígono.

As descrições detalhadas destas tabelas podem ser visualizadas em um dicionário de dados do censo americano [48], com suas respectivas colunas, detalhes e características.

Nas consultas utilizando a base de dados BCIM, foram utilizadas cinco tabelas. O único tipo espacial ausente nesta base é ponto, o qual foi substituído pelo centróide das geometrias de algumas das tabelas. Os nomes destas tabelas, índices, número de registros, o tamanho PostgreSQL e em CSV são apresentados na Tabela 2.

Tabela 2: Dados das tabelas da base BCIM

Tabela	**Índice**	**Nº de registros**	**Tamanho no PostgreSQL**	**Tamanho em CSV**
hid_barragem_l	hid_barragem_l_geom_gist	413	120 KB	168 KB
hid_massa_dagua_a	hid_massa_dagua_a_geom_gist	5117	3544 KB	5.91 MB
hid_trecho_drenagem_l	hid_trecho_drenagem_l_geom_gist	229436	132 MB	174 MB
lim_municipio_a	lim_municipio_a_geom_gist	5570	14 MB	42.6 MB
lim_unidade_federacao_a	lim_unidade_federacao_a_geom_gist	27	8192 bytes	5,32 MB

As tabelas hid_barragem_l,hid_massa_dagua_a e hid_trecho_drenagem_l são referentes às informações hídricas como barragens, lagos, bacias hidrográficas e rios. As duas primeiras tabelas possuem coluna espacial do tipo polígono e a última tabela do tipo cadeia de linhas. Estas tabelas possuem nove, oito e quinze colunas, respectivamente.

As tabelas lim_municipio_a e lim_unidade_federacao_a referem-se às geometrias dos municípios e estados, respectivamente. As duas utilizam polígonos e possuem sete colunas cada.

Os detalhes e informações pertinentes às tabelas e suas colunas podem ser visualizados em um dicionário de dados publicado na página do IBGE [49].

5.1.3 Descrição dos testes

As consultas realizadas na base TIGER/Line seguem padrões do *benchmark* Jackpine [44]. Estas se diversificam por utilizarem vários tipos de junções entre vários tipos de dados espaciais, como polígono-polígono, polígono-ponto, ponto-linha e polígono-linha e utilizarem várias funções espaciais para filtrar os registros, alterando a quantidade de registros entre as consultas.

Para a base BCIM, foram definidas consultas que abrangem outras funções espaciais e outros elementos, como agregação de dados convencionais e espaciais, além dos vários tipos de junções. No caso desta base de dados, as consultas não foram baseadas diretamente de um trabalho, mas são resultantes dos vários trabalhos analisados nesta dissertação, pois nem todas estas consultas espaciais são possíveis de serem realizadas nos trabalhos relacionados que foram analisados.

A base BCIM possui tamanho menor em comparação à base TIGER/Line, logo, o tempo total para execução destas consultas será menor. Por esta razão, foram planejadas consultas com o foco na agregação de dados espaciais e convencionais.

Visando avaliar a interoperabilidade do GeoDrill foram cruzadas as fontes CSV e MongoDB no ambiente desktop. Foram realizadas avaliações nas duas bases de dados, TIGER/Line e BCIM, utilizando seus respectivos conjuntos de consultas. Para avaliar se existem diferenças entre os tempos de execução das consultas em função dos tipos de fontes de dados consultadas, foram avaliadas as fontes de dados CSV e MongoDB nos dois ambientes propostos.

Contudo, devido a limitações do MongoDB em relação ao tamanho máximo do registro/documento (16 MB), não foi possível armazenar parte dos dados TIGER/Line. Desta forma, os testes que utilizam o MongoDB foram realizados utilizando somente os dados da base BCIM.

No caso do ambiente distribuído, também foram analisados somente os dados BCIM, pois o ambiente apresenta recursos limitados de memória e disco.

5.1.4 Descrição das consultas

Para a realização dos testes e subsequente obtenção dos resultados, foram definidas algumas consultas espaciais que abrangem os vários tipos de dados e funções espaciais.

O primeiro conjunto de consultas é referente aos dados TIGER/Line, composto por 23 consultas. Neste conjunto, foi utilizada uma pequena variedade de funções espaciais

(*ST_Touches*, *ST_Intersects* e *ST_ContainsProperly*); no entanto, o foco principal deste conjunto de consultas foi o volume de dados selecionado e as junções espaciais. Estas consultas definidas para os dados TIGER/Line podem ser visualizadas no Quadro 5. As primeiras consultas iniciam com a seleção de poucas linhas das tabelas e, em cada consulta subsequente, aumenta-se gradativamente o número de linhas selecionadas de cada tabela e a quantidade de junções. Estas consultas visam diversificar os tipos espaciais utilizados, possibilitando junções entre linha-linha, linha-polígono, polígono-polígono e ponto-polígono. Para cada consulta, uma função espacial é utilizada e o número de linhas é incrementado. Estas consultas tendem a avaliar o desempenho geral da solução, pois a maioria das análises envolvendo dados espaciais tratam de junções e filtragem de dados espaciais, utilizando, em sua maioria, a função para verificar interseção entre as geometrias.

Não foram propostas novas funções para a base TIGER/Line, pois a base já possui um bom conjunto de funções e, ao serem realizados testes preliminares com funções de agregação, o tempo de execução se apresentou impraticável para um experimento, mesmo em um SGBD espacial convencional.

Quadro 5: Consultas espaciais para os dados TIGER/Line

id	Consulta
c1	select a.fullname from linearwater a, linearwater b where st_touches(a.geom, b.geom) and b.fullname='yukon riv';
c2	select a.fullname from arealm a, linearwater b where st_touches(a.geom, b.geom) and b.fullname='yukon riv';
c3	select a.name from states a, states b where st_touches(a.geom, b.geom) and b.statefp='31';
c4	select a.fullname from areawater as a, states as b where st_containsproperly(b.geom, a.geom) and a.mtfcc='h2030' and b.statefp ='11';
c5	select a.fullname from primaryroads a, states b where st_intersects(a.geom, b.geom) and b.statefp='06';
c6	select a.fullname from linearwater as a, states as b where st_containsproperly(b.geom, a.geom) and b.statefp='11';
c7	select a.fullname from pointlm as a, states as b where st_containsproperly(b.geom, a.geom) and a.mtfcc='k1231' and b.statefp='11';
c8	select distinct a.fullname from arealm as a, states as b where st_containsproperly(b.geom, a.geom) and a.mtfcc='k2361' and b.statefp='06';
c9	select a.fullname from pointlm as a , states as b where st_containsproperly(b.geom, a.geom) and a.mtfcc='c3061' and b.statefp='11';
c10	select distinct a.fullname from arealm as a , states as b where st_containsproperly(b.geom, a.geom) and a.mtfcc='k2540' and b.statefp='06';
c11	select distinct a.fullname from roads a , arealm b where st_intersects(a.geom,b.geom) and a.mtfcc='s1820' and b.mtfcc='k2182';
c12	select a.fullname from pointlm as a , states as b where st_containsproperly(b.geom, a.geom) and a.mtfcc in ('k1231', 'k2165', 'k2190') and b.statefp='11';
c13	select a.fullname from areawater as a, states as b where st_containsproperly(b.geom, a.geom) and a.mtfcc='h3010' and b.statefp ='11';
c14	select distinct a.fullname from arealm as a , states as b where st_containsproperly(b.geom,a.geom) and a.mtfcc='c3023' and b.statefp='06';
c15	select distinct a.fullname from roads a , arealm b where st_intersects(a.geom, b.geom) and a.mtfcc='s1100' and b.mtfcc='k2182';
c16	select a.fullname from pointlm as a , states as b where st_containsproperly(b.geom,a.geom) and a.mtfcc='c3022' and b.statefp='11';
c17	select a.fullname from areawater as a , states as b where st_containsproperly(b.geom, a.geom) and a.mtfcc in ('h3010', 'h2030') and b.statefp ='11';
c18	select distinct a.fullname from arealm as a , states as b where st_containsproperly(b.geom, a.geom) and a.mtfcc='k2180' and b.statefp='06';
c19	select distinct a.fullname from roads a , arealm b where st_intersects(a.geom, b.geom) and a.mtfcc='s1780' and b.mtfcc='k2182';
c20	select a.fullname from pointlm as a , states as b where st_containsproperly(b.geom, a.geom) and a.mtfcc in ('k1231', 'k2165') and b.statefp='11';
c21	select distinct a.fullname from arealm as a , states as b where st_containsproperly(b.geom,a. geom) and a.mtfcc='k2582' and b.statefp='06';
c22	select a.fullname from pointlm as a , states as b where st_containsproperly(b.geom, a.geom) and a.mtfcc='k2451' and b.statefp='11';
c23	select distinct a.fullname from roads a , arealm b where st_intersects(a.geom, b.geom) and a.mtfcc='s1730' and b.mtfcc='k2182';

Para o conjunto de dados BCIM, foram definidas 15 consultas, as quais podem ser visualizadas no Quadro 6. Estas consultas abrangem os vários tipos de dados espaciais e uma diversa gama de funções espaciais, dentre as quais algumas utilizam agregação dos dados espaciais. Foram utilizadas as funções *ST_Buffer, ST_Contains, ST_Intersects, ST_Union, ST_Crosses, ST_length* e *ST_Area*. Os tipos de junções utilizadas foram ponto-polígono, linha-polígono e polígono-polígono.

Estas funções consideram um conjunto de dados um pouco menor em comparação com os dados TIGER/Line. Estes dados são importantes, pois são utilizados em várias aplicações brasileiras de SIG, além de possuírem características distintas aos dados americanos como, por exemplo, a divisão dos estados que usa fronteiras naturais e não políticas, com exceção do Acre. As consultas para este conjunto de dados envolvem operações com agregação de dados espaciais. Estas consultas com elementos de agregação utilizam a função de agregação *ST_Union*. Seu funcionamento é semelhante a uma função SUM ou AVG, no entanto, esta retorna uma geometria que representa a união das geometrias passadas como parâmetro.

Quadro 6: Consultas espaciais para os dados BCIM

id	Consulta
c1	select td.nome from (select * from lim_unidade_federacao_a where sigla = 'pb') uf, hid_trecho_drenagem_l td where st_crosses(uf.geom, td.geom);
c2	select count(*) from (select * from lim_unidade_federacao_a where sigla = 'pb') uf, hid_trecho_drenagem_l td where st_contains(uf.geom, td.geom);
c3	select td.nome from (select st_union(geom) geom from lim_unidade_federacao_a where sigla in ('pb', 'pe')) uf, hid_trecho_drenagem_l td where st_crosses(uf.geom, td.geom);
c4	select mun.nome from (select * from lim_unidade_federacao_a where sigla in ('pe')) uf, lim_municipio_a mun where st_intersects(uf.geom, mun.geom) and not st_contains(uf.geom, mun.geom);
c5	select mada.nome from (select nome, st_buffer(geom, 10000/1609.34) geom from lim_municipio_a mun where mun.nome = 'campina grande') mun inner join hid_massa_dagua_a mada on st_intersects(mun.geom, mada.geom);
c6	select mun.nome from (select nome, st_buffer(st_centroid(geom), 100/1609.34) centro from lim_municipio_a mun) mun where exists(select * from hid_trecho_drenagem_l td where st_intersects(mun.centro, td.geom));
c7	select * from hid_trecho_drenagem_l td where st_length(td.geom) >= (select max(st_length(td.geom)) from hid_trecho_drenagem_l td);
c8	select uf.nome from lim_unidade_federacao_a uf, hid_trecho_drenagem_l td where st_contains(uf.geom, td.geom) group by uf.nome order by count(*) desc limit 10;
c9	select uf.nome from lim_unidade_federacao_a uf, hid_trecho_drenagem_l td where and st_crosses(uf.geom, td.geom) group by uf.nome order by count(*) desc limit 5;
c10	select uf.nome from lim_unidade_federacao_a uf, (select * from hid_barragem_l order by st_area(geom) desc limit 1) td where st_intersects(uf.geom, td.geom) ;
c11	select td.nome, count(*) ct from lim_unidade_federacao_a uf, hid_trecho_drenagem_l td where st_crosses(uf.geom, td.geom) group by td.nome;
c12	select uf.nome, count(*) from lim_unidade_federacao_a uf, hid_trecho_drenagem_l td where st_contains(uf.geom, td.geom) group by uf.nome;
c13	select uf.nome, count(*) from lim_unidade_federacao_a uf, hid_trecho_drenagem_l td where st_overlaps(uf.geom, td.geom) group by uf.nome;
c14	select mun.nome from (select * from lim_municipio_a where (st_area(geom) * power(1609.34, 2)) < 20000) mun where not exists (select * from hid_trecho_drenagem_l td where st_contains(mun.geom, td.geom)) ;
c15	select uf.nome from lim_unidade_federacao_a uf, hid_trecho_drenagem_l td where st_crosses(uf.geom, td.geom) group by uf.nome having count(*) > 5;

Visando avaliar a interoperabilidade do GeoDrill, as consultas do Quadro 6 foram reescritas de forma que cada tabela utilizada seja oriunda de uma fonte distinta. Ademais, foi proposta uma consulta, que é descrita no Quadro 7, para avaliar a integração entre três fontes de dados heterogêneas. Nesta consulta, foram utilizadas três tabelas distintas, referentes aos municípios, estados e barragens, enquanto as fontes de dados utilizadas foram JSON, MongoDB, e CSV, respectivamente. Esta consulta informa o número de barragens que fazem interseção aos municípios vizinhos do estado da Paraíba.

Quadro 7: Consultas espaciais para os dados BCIM, utilizando 3 tabelas para 3 fontes distintas

id	Consulta
c16	with municipios_vizinhos as (select mun.gid, mun.nome, mun.geom from (select * from lim_unidade_federacao_a where sigla = 'pb') uf join lim_municipio_a mun where st_intersects(uf.geom, mun.geom) and not st_contains(uf.geom, mun.geom)), select count(b.gid) from hid_barragem_l b join municipios_vizinhos mun where st_intersects(b.geom, mun.geom);

5.1.5 Variáveis de Teste

Para a primeira e segunda etapas de avaliações, foram consideradas as seguintes variáveis de teste: as bases de dados (TIGER/Line e BCIM), as consultas espaciais, as tecnologias de armazenamento (CSV e MongoDB) e os dois ambientes operacionais (desktop e distribuído). Enquanto na primeira etapa utilizou-se somente o ambiente desktop, na segunda ambos foram utilizados. A variável de resposta coletada em ambas as etapas foi o tempo de execução das consultas.

Para a terceira e última etapa, o GeoDrill foi comparado com o PostGIS/PostgreSQL. Desta forma, foram consideradas as seguintes variáveis: a tecnologia utilizada para consultar os dados (PostGIS/PostgreSQL ou GeoDrill), as bases de dados (TIGER/Line e BCIM), as consultas espaciais e os dois ambientes operacionais (desktop e distribuído). A variável de resposta analisada foi o tempo de execução das consultas, da mesma forma que nas etapas anteriores.

5.1.6 Hipóteses

Os testes realizados tiveram como objetivo avaliar a flexibilidade do GeoDrill na realização de consultas espaciais em diversas fontes de dados; as diferenças de desempenho do GeoDrill para diferentes tipos de fontes de dados; e comparar o desempenho do GeoDrill com o PostGIS/PostgreSQL.

Referente à flexibilidade do GeoDrill na execução de consultas espaciais cruzando fontes heterogêneas (questão de pesquisa 1) tem-se as seguintes hipóteses:

H0: O GeoDrill não realiza consultas espaciais em fontes heterogêneas.

H1: O GeoDrill realiza consultas espaciais em fontes heterogêneas.

Em relação às comparações que avaliaram o desempenho do GeoDrill para as diferentes fontes de dados utilizadas (questão de pesquisa 2), foram definidas as seguintes hipóteses:

H0: Os tempos das consultas entre as duas fontes são iguais.

H1: Consultas em CSV possuem tempo menor de consulta.

H2: Consultas em MongoDB possuem tempo menor de consulta.

Em relação à comparação de desempenho entre o GeoDrill e o PostGIS/PostgreSQL (questão de pesquisa 3), foram definidas as seguintes hipóteses:

H0: Os tempos das consultas entre as duas tecnologias são iguais.

H1: O PostGIS/PostgreSQL possui tempo menor de consulta.

H2: O GeoDrill possui tempo menor de consulta.

5.2 Etapas do experimento

O experimento foi dividido em três etapas para responder às hipóteses. A primeira etapa (Avaliação da interoperabilidade do GeoDrill) foi definida para responder à questão principal da pesquisa em relação à interoperabilidade do GeoDrill. A segunda etapa (Avaliação do desempenho de consultas no GeoDrill em relação à tecnologia de armazenamento de dados utilizada) foi definida para avaliar se existe diferença entre o desempenho na execução das consultas espaciais em relação à fonte na qual os dados estão armazenados. A terceira etapa (Avaliação comparativa entre o GeoDrill e o PostGIS/PostgreSQL) avalia o tempo de execução das consultas espaciais do GeoDrill em relação a um SGBD espacial convencional, o PostGIS/PostgreSQL. Os resultados obtidos nestas etapas do experimento também auxiliam a identificar questões para pesquisas futuras.

Para a geração dos gráficos, comparações e testes estatísticos foi utilizado o software R[34] de computação estatística. Para basear os testes e análise foram utilizadas referências de livros sobre métodos estatísticos e suas aplicações utilizando o R [50] [51] [52]. Os detalhes referentes às comparações estatísticas desta seção podem ser visualizados no Apêndice 1 e Apêndice 2.

[34] The R Project - https://www.r-project.org/

Para todas as comparações foi realizada uma avaliação utilizando *boxplot* e, quando necessário, foi realizada uma análise com testes estatísticos.

5.2.1 Avaliação da interoperabilidade do GeoDrill (Etapa 1)

Esta etapa do experimento foi conduzida com o objetivo de avaliar a capacidade de executar as consultas espaciais em fontes de dados heterogêneas, considerada a principal questão de pesquisa. Para isto, partes dos dados foram armazenadas em CSV e em MongoDB, possibilitando executar consultas cruzando estas fontes de dados. Para as avaliações envolvendo ambas as bases de dados (TIGER/Line e BCIM) foram realizadas consultas cruzando dados de um arquivo CSV e uma coleção MongoDB. Além disso, para as avaliações envolvendo a base BCIM, foi realizada uma consulta adicional (consulta 16) que utiliza 3 fontes de dados: JSON, CSV e MongoDB. Os resultados são exibidos por meio dos gráficos de *boxplot* para os dados TIGER/Line e BCIM em ambiente desktop, que podem ser visualizados na Figura 18 e na Figura 19, respectivamente.

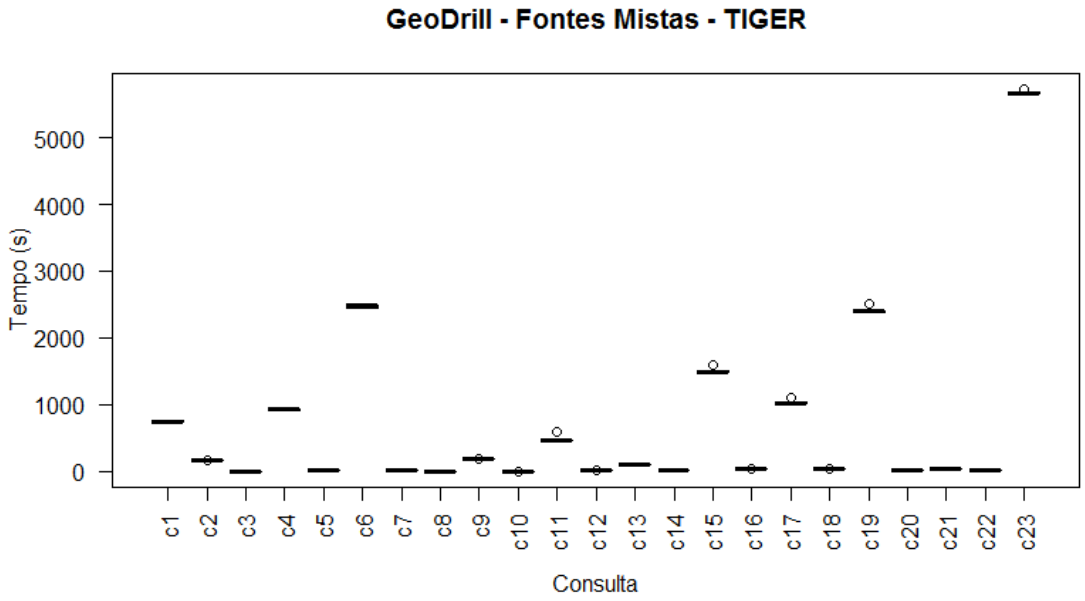


Figura 18: Boxplot dos tempos das consultas à base TIGER/Line em fontes mistas em ambiente desktop

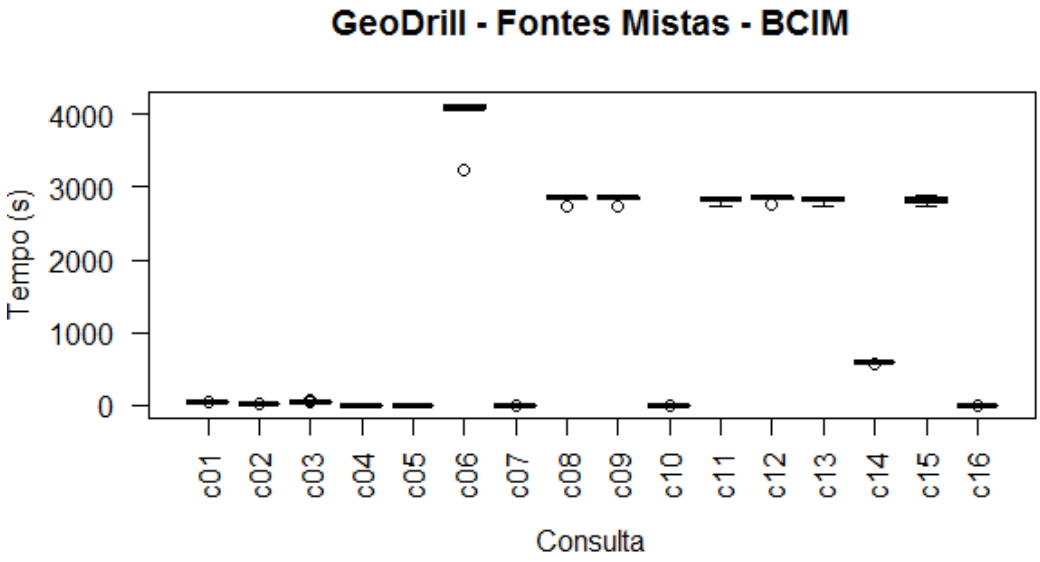


Figura 19: Boxplot dos tempos das consultas à base BCIM em fontes mistas em ambiente desktop

O objetivo aqui foi verificar se o GeoDrill mantém a integração de fontes de dados heterogêneas mesmo ao executar as consultas espaciais. Como o GeoDrill apresentou resultados para todas as consultas propostas, é possível afirmar que este foi bem-sucedido nesta etapa. Desta forma, conclui-se que é possível cruzar os dados espaciais de várias fontes de dados heterogêneas utilizando linguagem de consulta SQL, além disso, é possível executar as consultas espaciais em todas as tecnologias suportadas pelo Apache Drill.

Utilizando-se soluções como Hadoop GIS ou SpatialHadoop, não seria possível realizar as consultas através de uma linguagem como SQL. Neste caso, estas teriam de ser executadas utilizando Pigeon ou um código em linguagem Java escrito para utilizar MR. No Hadoop GIS não seria possível realizar a maioria das consultas espaciais, pois o mesmo oferece suporte a apenas algumas funções espaciais, além de possuir limitações relacionadas à execução de junções espaciais. Além disso, estas soluções não permitem explorar dados semiestruturados diretamente nas consultas, ou seja, não seria possível carregar dados de fontes como JSON ou MongoDB, que foram fontes utilizadas nesta etapa do experimento. Não obstante, nenhuma das duas soluções permite integrar dados diretamente sem replicá-los, embora ambas possam ser usadas como um DW para integrar informações.

As soluções Hbasespatial, GeoSpark Simba podem explorar dados espaciais e semiestruturados, no entanto, todas possuem suporte espacial básico, limitado a apenas algumas funções. O HBaseSpatial não é capaz de integrar fontes de dados heterogêneas, ou seja, faz-se necessário replicar os dados. No entanto, o GeoSpark e o Simba possuem a

capacidade de integrar fontes como um *dataspace*. Porém, destes, somente o Simba permite utilizar linguagem SQL para realização das consultas espaciais. Como o GeoSpark e o Simba possuem um conjunto limitado de funções, consultas que usem as funções como *crosses*, *buffer, contains properly* ou agregação de dados espaciais com união, não podem ser executadas nestas soluções. Como exemplos da utilização das funções *crosses, union* e *buffer*, é possível citar as consultas, c1, c3 e c5 do Quadro 6, respectivamente. Outros exemplos podem ser visualizados no Quadro 5 e no Quadro 6, referentes às consultas às bases TIGER/Line e BCIM, respetivamente.

5.2.2 Avaliação do desempenho de consultas no GeoDrill em relação a tecnologia de armazenamento de dados utilizada (Etapa 2)

A avaliação conduzida nesta etapa pretende determinar se existe diferença no desempenho das consultas espaciais realizadas pelo GeoDrill dependendo da tecnologia que armazena os dados, uma vez que o GeoDrill utiliza processamento em memória. Para responder a esta questão foram feitas medições do tempo de execução das consultas para os dados oriundos da base BCIM, nos ambientes desktop e distribuído, utilizando-se as fontes de dados CSV e MongoDB.

A Figura 20 apresenta o *boxplot* dos resultados obtidos com a execução das consultas à base BCIM em CSV, em ambiente desktop. Os resultados das consultas ficaram divididos em dois grupos, com aproximadamente metade das consultas com tempos superiores a 40 minutos, e outra metade com tempos inferiores a um minuto.

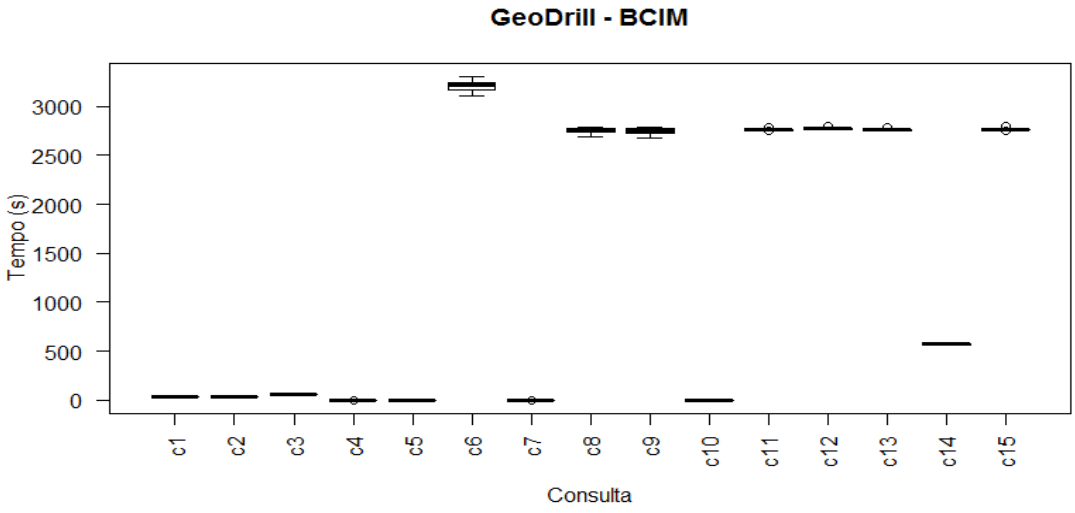


Figura 20: Boxplot dos tempos das consultas à base BCIM, em CSV, utilizando ambiente desktop

Os resultados do *boxplot* da Figura 21 são referentes às consultas executadas na base BCIM, tendo como fonte o MongoDB, em ambiente desktop. Observa-se graficamente que estes resultados são bem próximos dos apresentados anteriormente para os dados em CSV.

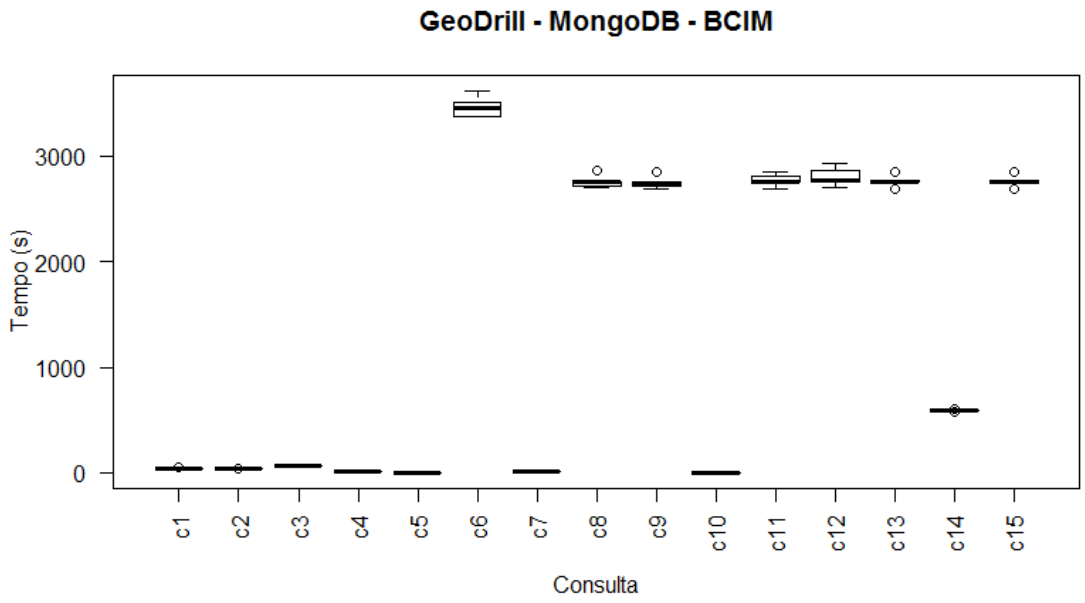


Figura 21: Boxplot dos tempos das consultas à base BCIM, armazenadas em MongoDB, utilizando ambiente desktop

Os resultados apresentados na Figura 22 e na Figura 23 foram obtidos utilizando a arquitetura distribuída do GeoDrill, nas consultas espaciais aos dados BCIM, em arquivos CSV e MongoDB, respectivamente. Visualmente, é possível perceber que o desempenho destas consultas foi inferior ao apresentado na arquitetura desktop. Isto pode ser justificado pelo fato de o ambiente distribuído utilizado dispor de menos recursos computacionais que o ambiente desktop.

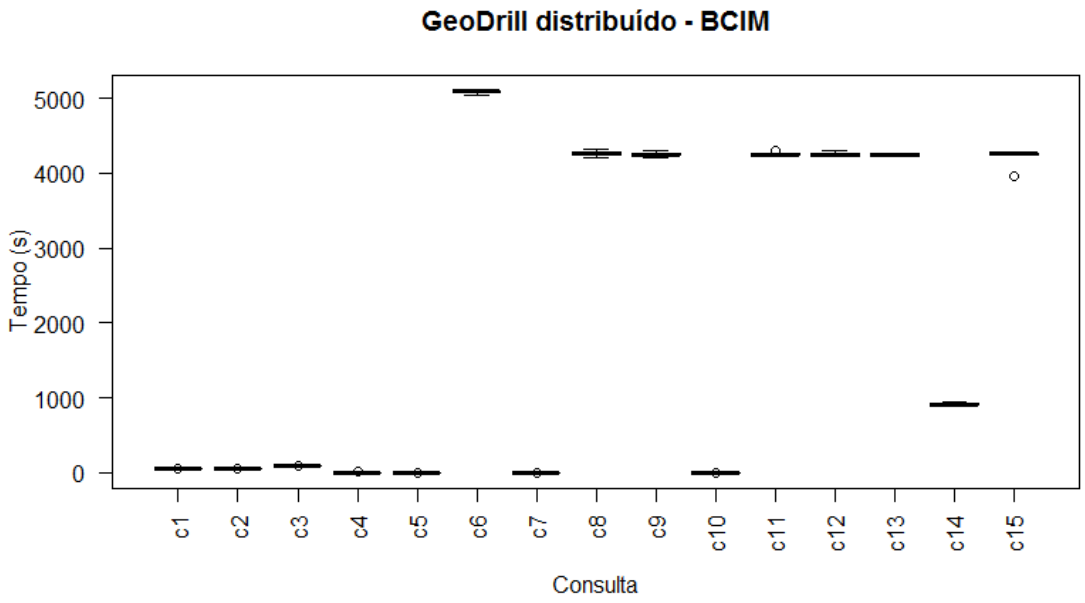


Figura 22: Boxplot dos tempos das consultas à base BCIM, em CSV, utilizando arquitetura distribuída

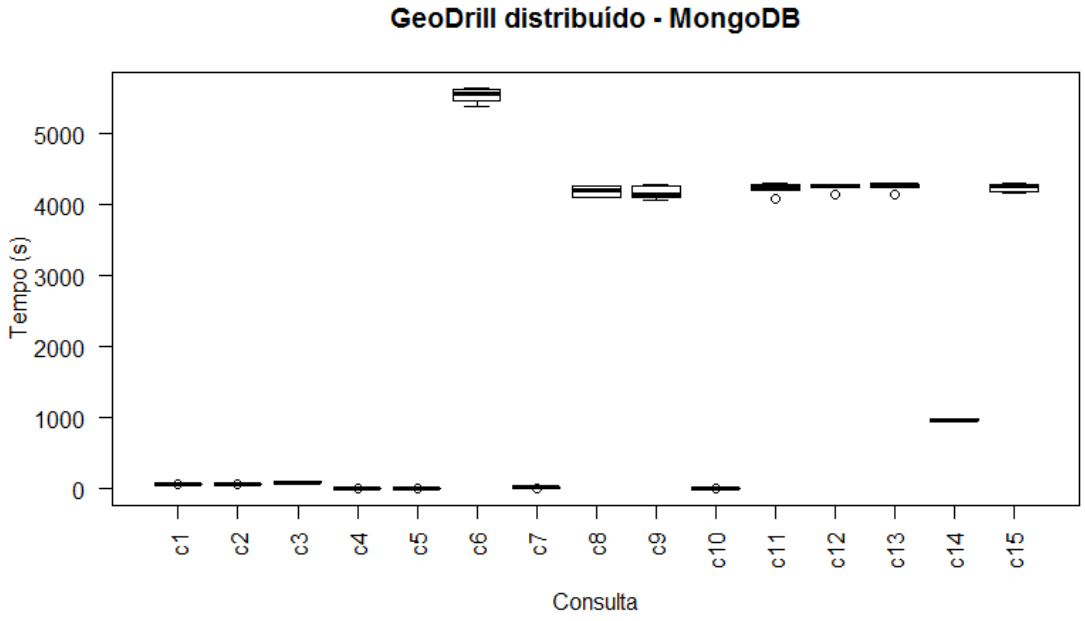


Figura 23: Boxplot dos tempos das consultas à base BCIM, armazenada no MongoDB, utilizando arquitetura distribuída

Após a obtenção dos tempos de cada consulta para cada tecnologia e ambiente, é possível compará-los para indicar se há diferença em relação à tecnologia de armazenamento utilizada.

Para o ambiente desktop, os testes utilizando *boxplot* presentes na Figura 24 indicaram que as consultas de 1 a 7 e 14 obtiveram melhor desempenho para a fonte de dados CSV, enquanto a consulta 10 obteve melhor desempenho para o MongoDB (vide Apêndice 1). Para

as demais consultas que apresentaram interseção, foram realizados testes estatísticos (vide Apêndice 2).

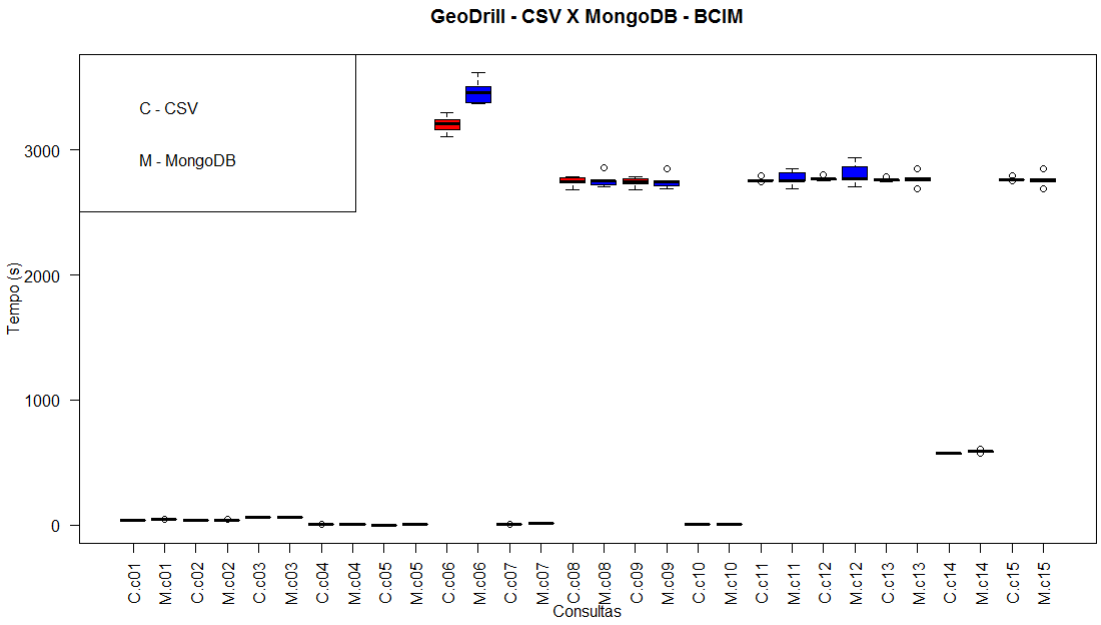


Figura 24: GeoDrill em ambiente desktop – CSV X MongoDB – BCIM

Os testes estatísticos realizados nas consultas com interseção não refutaram a hipótese nula. Assim, é possível considerar os resultados destas consultas como iguais, pois não possuem diferença estatística. Portanto, considera-se que há diferenças entre o tipo de fonte de dados utilizada, pois metade das consultas obtiveram menores tempos em CSV.

Para o ambiente distribuído, cujas comparações são ilustradas nos gráficos de *boxplots* da Figura 25, é possível observar que o GeoDrill obteve melhor desempenho para o tipo CSV no ambiente para execução das consultas de 1 a 7, 10 e 14 (vide Apêndice 1). As demais consultas apresentaram interseção, o que requer uma avaliação baseada em testes estatísticos (vide Apêndice 2).

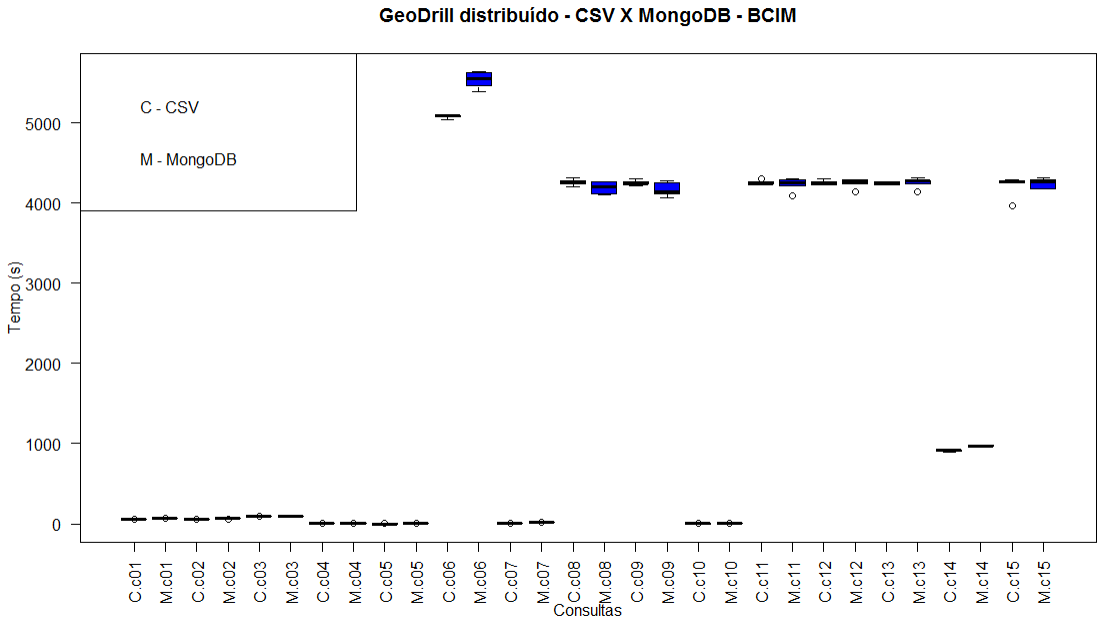


Figura 25: GeoDrill em ambiente distribuído - CSV X MongoDB - BCIM

Não foi possível refutar a hipótese nula em nenhum teste estatístico nos resultados das consultas com interseção. Deste modo, pode-se considerar que os resultados obtidos para estas consultas, utilizando-se as fontes CSV e MongoDB, são estatisticamente iguais. Portanto, para o ambiente distribuído, pode-se considerar que há impacto no desempenho dependendo da fonte de dados utilizada. A fonte de dados CSV também apresentou os menores tempos nesse ambiente.

Como uma avaliação geral para os dois ambientes propostos, foi possível observar que as fontes de dados utilizadas influenciam no tempo das consultas espaciais. A fonte de dados CSV apresentou melhores resultados em ambos os ambientes.

5.2.3 Avaliação comparativa do GeoDrill com o PostGIS/PostgreSQL (Etapa 3)

Nesta etapa, o GeoDrill é comparado com o PostGIS/PostgreSQL para verificar se o GeoDrill apresenta um bom desempenho na realização das consultas espaciais. Para esta comparação, foram utilizados os dados das bases TIGER/Line e BCIM. Devido às limitações do ambiente distribuído, os dados TIGER/Line só foram avaliados em ambiente desktop. Para o GeoDrill, os dados da base BCIM são os informados nas etapas anteriores, em CSV.

A Figura 26 exibe um gráfico do tipo *boxplot* que resume os resultados obtidos no GeoDrill para execução das consultas espaciais utilizando os dados TIGER/Line em arquivos CSV.

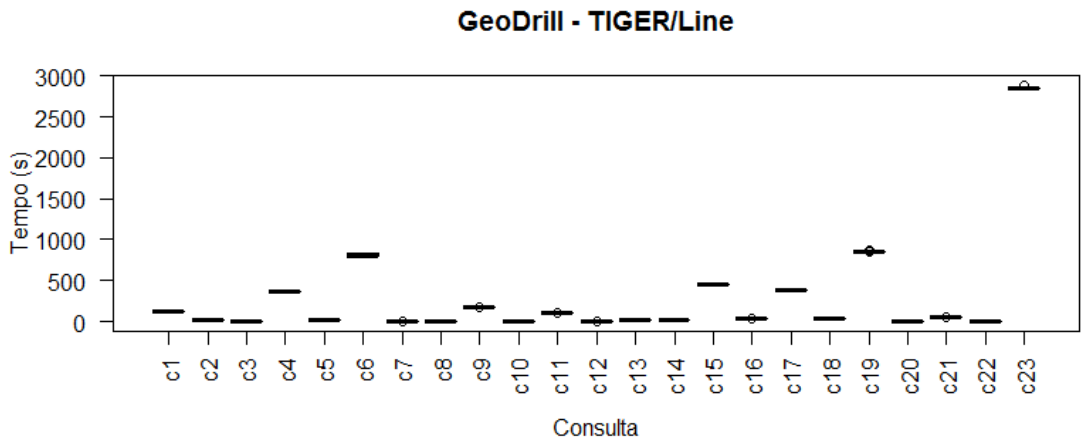


Figura 26: Boxplot dos tempos das consultas à base TIGER/Line (em CSV) com GeoDrill em ambiente desktop

Por sua vez, um gráfico que sintetiza os resultados obtidos para execução das consultas utilizando o PostGIS/PostgreSQL com dados TIGER/Line é apresentado na Figura 27. É possível observar que houve alguns *outliers* em algumas consultas (c1, c11 e c15). Estes *outliers* podem ter ocorrido por diversos motivos, como, por exemplo, se os dados estavam sendo carregados no *cache* do disco rígido ou se ocorreu uma interrupção que influenciou no tempo. Entretanto, de forma geral, a maioria das consultas apresenta tempo de execução menor que um segundo, o que aparenta melhor eficiência para o PostGIS/PostgreSQL.

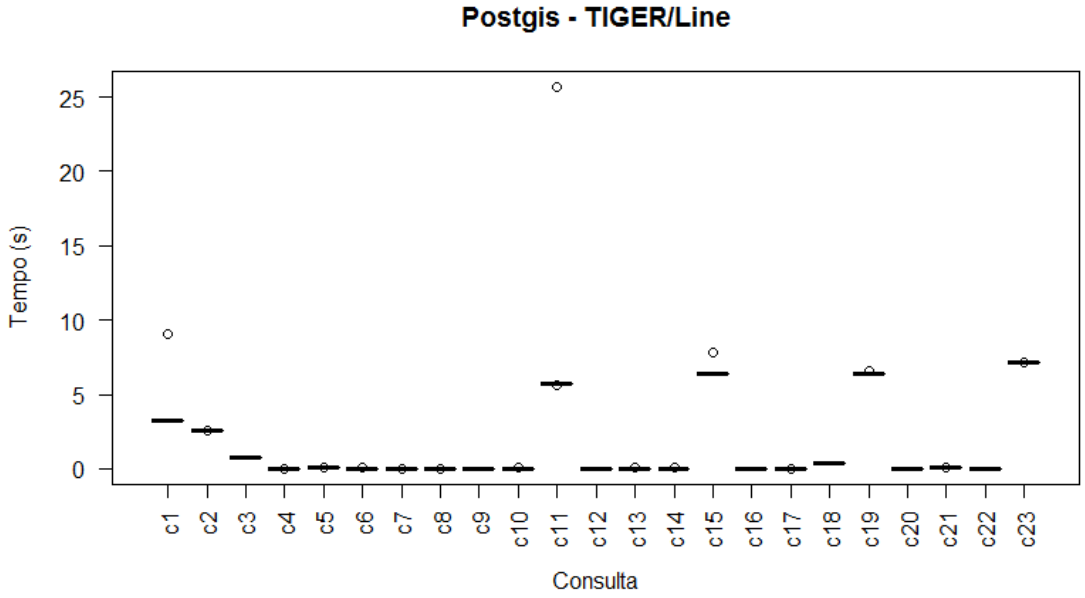


Figura 27: Boxplot dos tempos das consultas à base TIGER/Line com PostGIS/PostgreSQL em ambiente desktop

Na Figura 28, é apresentado o gráfico *boxplot* dos resultados obtidos para execução dos testes com os dados BCIM utilizando o PostGIS/PostgreSQL em ambiente desktop. Estes

resultados são bem distintos em comparação aos anteriores, com a base TIGER/Line. Os tempos entre estas consultas também foram bem distintos: enquanto algumas foram executadas em tempos maiores que uma hora, outras apresentaram tempos inferiores a um segundo. De maneira geral, estes resultados também apresentaram menos *outliers* e um desvio padrão pequeno, sendo possível inferir que estes resultados foram mais concisos.

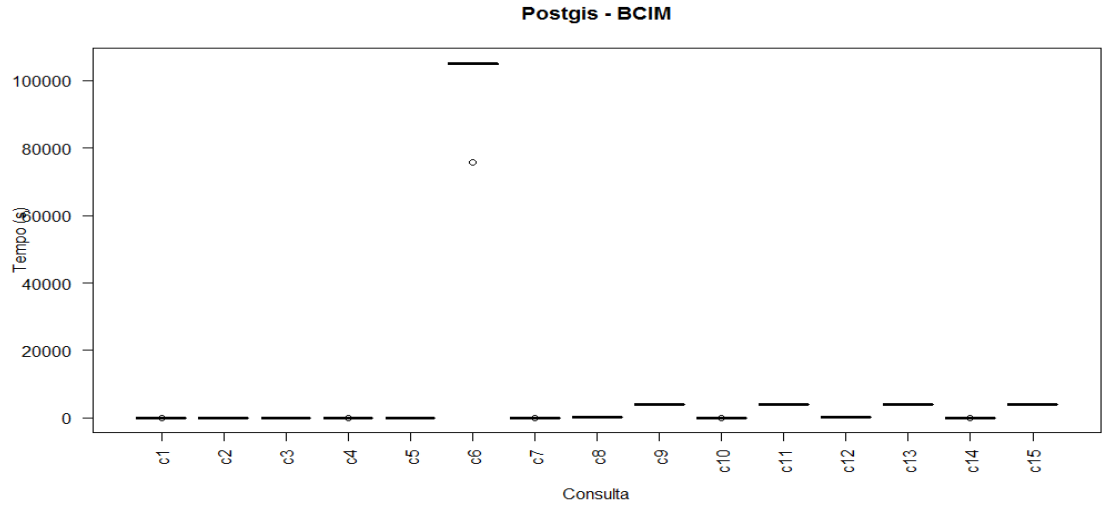


Figura 28: Boxplot dos tempos das consultas à base BCIM com PostGIS/PostgreSQL em ambiente desktop

Aparentemente, a única consulta que apresentou comportamento totalmente distinto às demais foi a consulta 6. Este comportamento pode ser explicado pela utilização da operação de *buffer* e, consequentemente, de realização de uma junção sem indexação da nova geometria. Tanto a função de *buffer*[35] quanto a junção são operações custosas computacionalmente, principalmente com dados não indexados.

O PostGIS/PostgreSQL foi configurado também em ambiente distribuído para execução de consultas à base BCIM. Os resultados obtidos são ilustrados no gráfico *boxplot* apresentado Figura 29. É possível observar que os resultados de seis consultas (c8, c9, c11, c12, c13 e c15) são distintos das demais deste ambiente, com tempos superiores a uma hora e seis minutos. Não é possível visualizar a consulta c6 neste gráfico, pois a mesma não completou sua execução após algumas horas (aproximadamente vinte e quatro horas). Deste modo, os resultados para essa consulta foram desconsiderados.

[35] Função *buffer* - http://PostGIS.net/docs/ST_Buffer.html

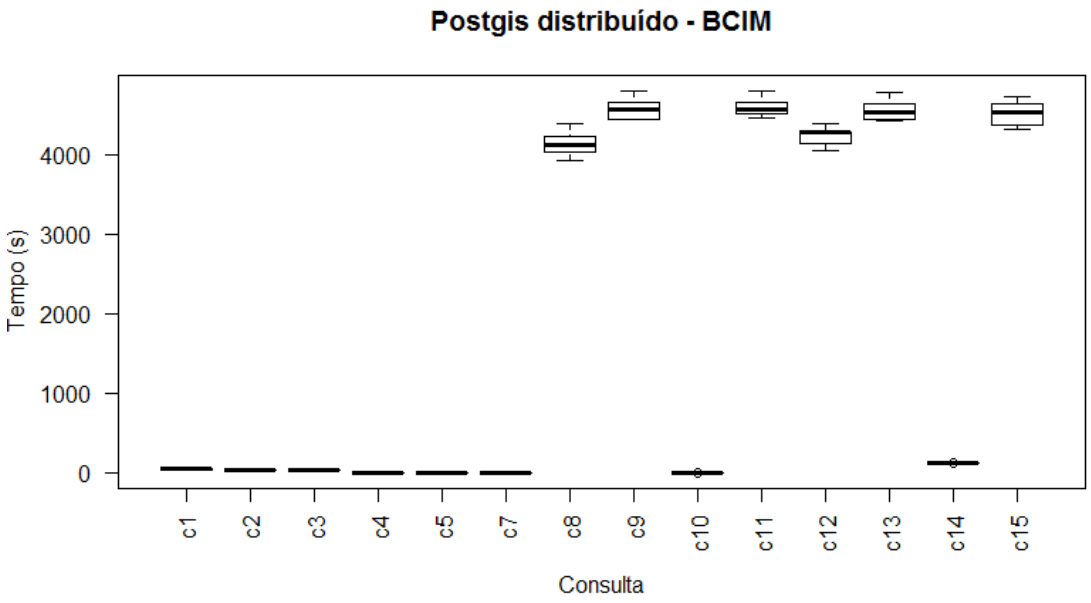


Figura 29: Boxplot dos tempos das consultas na base BCIM utilizando arquitetura distribuída no PostGIS/PostgreSQL

O gráfico apresentado na Figura 30 ilustra a comparação dos resultados obtidos com o PostGIS/PostgreSQL e o GeoDrill, utilizando arquivos CSV com dados TIGER/Line.

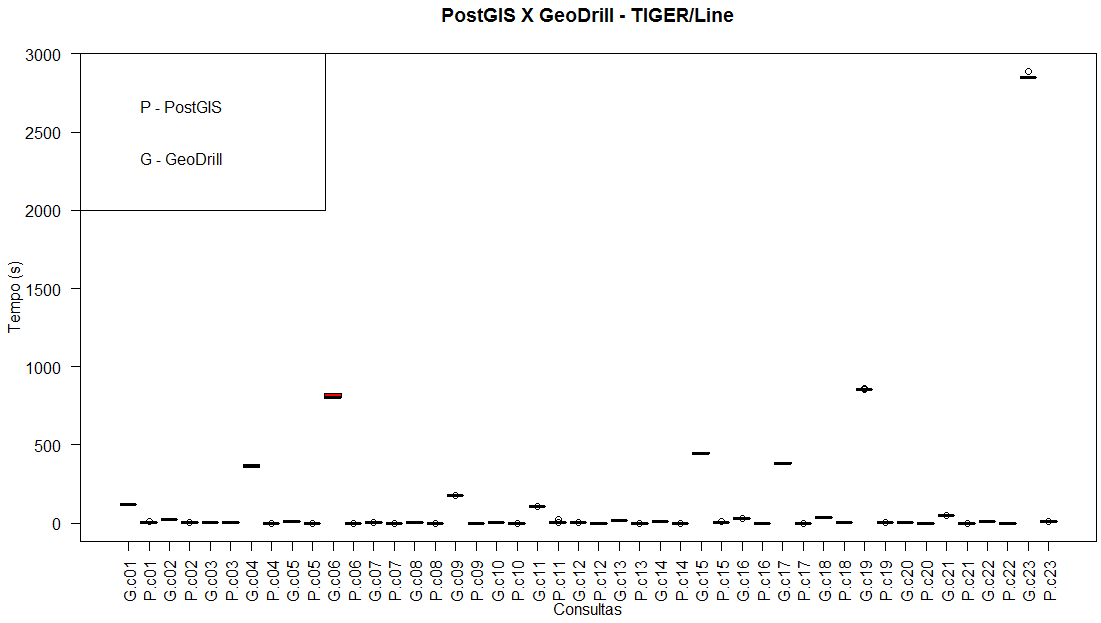


Figura 30: PostGIS/PostgreSQL X GeoDrill – Desktop – Tiger/Line

Ao visualizar as comparações de forma individual, foi possível observar que não há interseção em nenhum gráfico (vide Apêndice 1). Portanto, refuta-se a hipótese nula, a qual presume que as duas soluções possuem o mesmo desempenho. Em todos os gráficos, o GeoDrill apresenta tempo de execução maior do que o do PostGIS/PostgreSQL. No entanto,

deve-se considerar que o GeoDrill é uma solução de integração de dados que não necessita de processo de ETC para analisar os dados de várias fontes heterogêneas. Por outro lado, a análise dos dados utilizando PostGIS/PostgreSQL requer, primeiramente, a realização de um processo de ETC para extrair dados de fontes heterogêneas, como por exemplo, CSV ou MongoDB. Desta forma, o tempo de desenvolvimento e execução do processo de ETC, pode tornar a diferença final entre os tempos do GeoDrill e do PostGIS/PostgreSQL irrelevantes.

Os gráficos exibidos na Figura 31 são referentes à comparação dos resultados obtidos com o PostGIS/PostgreSQL e o GeoDrill, utilizando arquivos CSV com os dados BCIM.

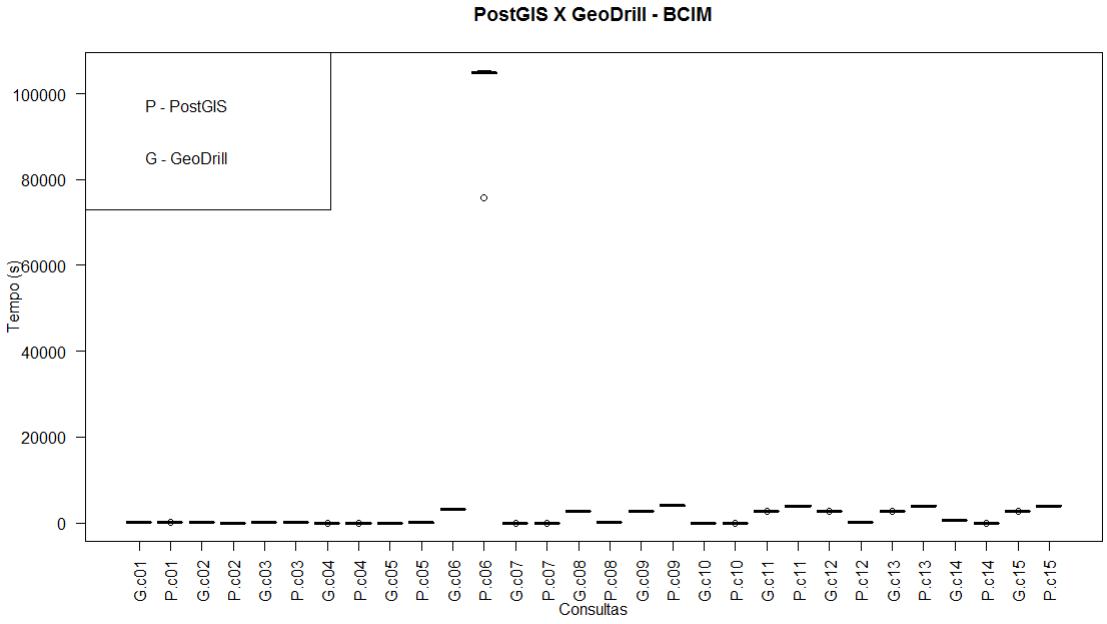


Figura 31: PostGIS/PostgreSQL X GeoDrill – Desktop – BCIM

Esta avaliação apresentou comportamento bem distinto em relação à anterior. Verificou-se que não houve interseção entre os *boxplots* das consultas, assim refutando a hipótese nula para todas as consultas. Nas consultas "c01", "c02", "c04", "c07", "c08", "c10", "c12" e "c14", o PostGIS/PostgreSQL apresentou os menores tempos, enquanto que nas demais consultas o GeoDrill obteve os menores tempos (vide Apêndice 1).

Os resultados ilustrados nos *boxplots* da Figura 32 são referentes às consultas na base BCIM em ambiente distribuído no PostGIS/PostgreSQL e no GeoDrill, utilizando arquivos CSV.

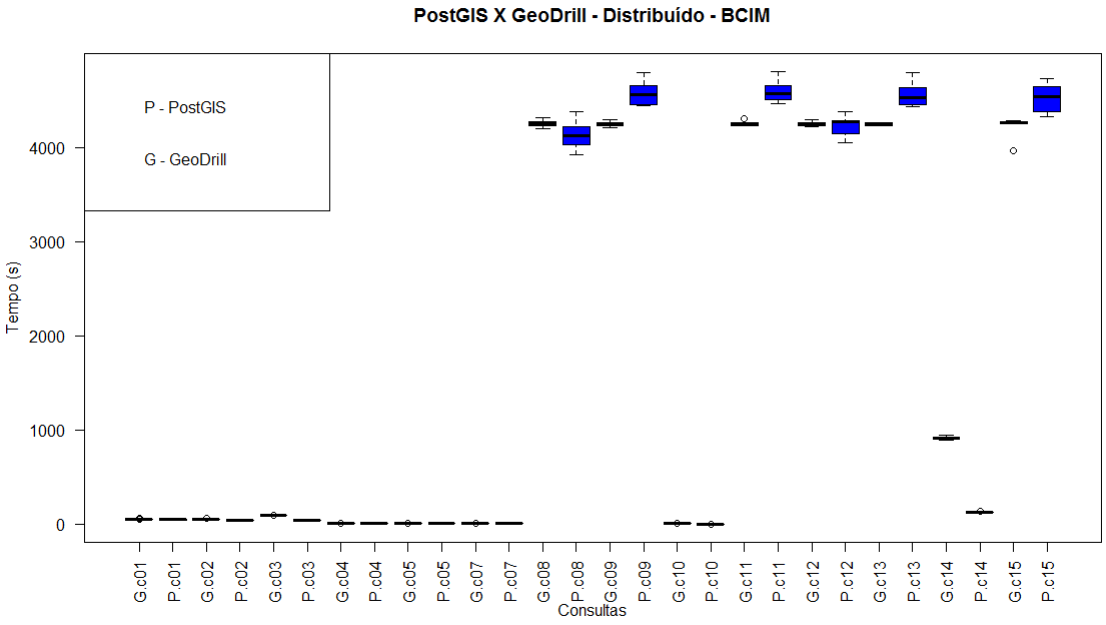


Figura 32: PostGIS/PostgreSQL X GeoDrill - Distribuído - BCIM

Neste comparativo, o GeoDrill obteve melhores resultados em oito das quinze consultas (1, 2, 3, 4, 7, 8, 10, 14), enquanto o PostGIS/PostgreSQL obteve melhores resultados em cinco das quinze consultas (5, 9, 11, 13, 15) (vide Apêndice 1). A consulta 12 foi a única que apresentou interseção, não sendo possível afirmar qual solução foi a melhor. Entretanto, pode-se considerar que o GeoDrill obteve melhores resultados, tanto por ter desempenho melhor em um maior número de consultas, como também pelo fato do PostGIS/PostgreSQL não ter conseguido executar uma das consultas (consulta 6) dentro de um limite de tempo razoável.

5.3 Discussão dos resultados

A capacidade de realização de consultas espaciais em bases heterogêneas pelo GeoDrill foi validada, apresentando um resultado positivo para a solução, uma vez que esta demonstrou ser capaz de executar todas as consultas propostas em tempo razoável. Além de possibilitar cruzar informações de bases distintas, foi possível integrar dados estruturados e semiestruturados, estes últimos armazenados em fontes como MongoDB e JSON.

Como o GeoDrill é uma extensão para o Apache Drill e foi possível verificar que, ao executar as funções espaciais manteve-se a capacidade de interoperar dados entre bases distintas, pode-se afirmar que o GeoDrill mantém as mesmas funcionalidades para as outras fontes de dados suportadas pelo Drill, como SGBD convencionais, HBASE e sistemas de arquivos distribuídos.

Ainda sobre a flexibilidade e interoperabilidade do GeoDrill, foi possível realizar uma consulta aos dados BCIM cruzando três fontes de dados: JSON, CSV e MongoDB. Com esta última validação, ficou mais evidente a capacidade de cruzar dados das fontes heterogêneas. A integração das fontes e a análise dos dados semiestruturados foram feitas utilizando SQL padrão, não sendo necessário utilizar linguagem adicional para explorar os dados.

Na segunda etapa do experimento, considerando a flexibilidade do GeoDrill, foi possível realizar as mesmas consultas espaciais em duas fontes de dados diferentes: CSV e MongoDB. Os tempos observados para execução das consultas não foram muito distintos. A fonte de dados MongoDB obteve melhor resultado em comparação com a fonte de dados CSV, apenas para a consulta 10 dos dados BCIM. Nas demais consultas, o MongoDB obteve desempenho pior ou equivalente às consultas realizadas em CSV. O GeoDrill conseguiu realizar, com bom desempenho, consultas em dados armazenados em arquivos do tipo CSV e no MongoDB utilizando vários operadores topológicos.

Em geral, o GeoDrill simplificou a utilização do MongoDB e a preparação dos dados para a inserção no MongoDB. Foi possível verificar a flexibilidade do GeoDrill, ao se realizar as mesmas consultas entre as fontes de dados CSV e MongoDB. Essa flexibilidade do GeoDrill possibilita a migração entre tecnologias, pois mantém um padrão de linguagem de consulta já conhecido e difundido. Isso também é importante para que outras tecnologias a utilizem como meio para analisar dados de fontes distintas, a exemplo de ferramentas OLAP ou SOLAP (*Spatial* OLAP).

Para a terceira etapa do experimento, em relação à base TIGER/Line, o GeoDrill não conseguiu superar os resultados apresentados pelo PostGIS/PostgreSQL. No entanto, ressalva-se que não foi avaliado e medido o tempo de execução de um processo de ETC para transferir dados de fontes heterogêneas para o PostGIS/PostgreSQL. Assim, pode-se considerar a utilização do GeoDrill como uma alternativa a um SGBD espacial.

Na base de dados BCIM, os resultados apresentados foram bem distintos. As consultas em que o GeoDrill obteve melhores desempenhos foram as que apresentaram alguma transformação nos dados espaciais, como um *buffer*, ou utilizam funções que podem levar os índices do PostGIS/PostgreSQL a não apresentarem bons resultados, como por exemplo, a função "*crosses*".

Por fim, o último teste validou a característica de execução de consultas em ambiente distribuído. Esperava-se que o desempenho apresentado fosse menor devido aos recursos utilizados, entretanto, mesmo com as limitações observadas, o GeoDrill se mostrou capaz de

realizar as consultas nas arquiteturas propostas e apresentou tempos de execução menores para algumas consultas, em comparação com o PostGIS/PostgreSQL.

Capítulo 6 – Considerações finais e trabalhos futuros

Neste capítulo, são apresentadas as conclusões e contribuições desta dissertação, bem como as limitações deste trabalho e as ideias para a continuação da pesquisa em trabalhos futuros.

Com esta pesquisa, foi possível estender o Apache Drill de modo a suportar dados espaciais, permitindo pesquisas, análises e o cruzamento de dados espaciais de várias fontes de dados heterogêneas. Assim, foi proposta uma resolução para o problema de integração de dados espaciais, utilizando uma aplicação de *dataspace*, que realiza consultas *ad-hoc*, não necessitando definir esquema ou replicar os dados.

As modificações existentes na linguagem SQL que permitem acessar e manipular os dados de várias fontes de dados, além de possibilitar explorar dados estruturados e semiestruturados, simplificam a utilização da solução. A adoção da linguagem SQL permite que aplicações que analisam dados, acessando algum SGBD espacial convencional, possam utilizar o GeoDrill.

O GeoDrill oferece a possibilidade de outras aplicações utilizarem-no como meio de acesso às diversas tecnologias de armazenamento de dados. Esta capacidade decorre do GeoDrill ser acessível via JDBC/ODBC para comunicação entre as aplicações e utilizar linguagem SQL para consulta aos dados. Desta forma, a aplicação necessitaria apenas gerenciar a conexão com o GeoDrill.

Foi possível, com o GeoDrill, executar várias funções espaciais, realizando a junção de vários tipos de dados espaciais, a filtragem e transformações dos dados espaciais. Outra contribuição do GeoDrill é a capacidade de criar novas funções, devido à arquitetura em camadas adotada e a um conjunto auxiliar de estruturas de dados criadas. A simplicidade da adição e manutenção das funções torna a solução adaptável a outros cenários. Por exemplo, com a adição de novos tipos de dados, é necessário modificar e adicionar novas funções. É importante que este processo seja simples e rápido.

Foram realizadas três etapas de avaliações experimentais objetivando validar as questões de pesquisa e testar as funcionalidades do GeoDrill. Primeiramente, foi possível verificar que foi mantida a funcionalidade de integração de dados ao realizar testes com o cruzamento de dados de fontes CSV e MongoDB, além de uma consulta cruzando estas duas

fontes citadas com uma terceira fonte contendo dados em JSON. É importante destacar que estas três fontes de dados apresentam características bem distintas, desde a linguagem para consultar os dados, ao modelo e formato de armazenamento dos dados.

Em seguida, foi verificado que há diferenças no tempo execução das consultas espaciais dependendo da fonte de dados utilizada no GeoDrill. Assim, para os experimentos que comparam o desempenho do GeoDrill, é necessário escolher uma fonte específica.

A última etapa consistiu na análise de desempenho do GeoDrill em relação ao PostGIS/PostgreSQL, apresentando resultados variados. O GeoDrill não tem um bom desempenho nas consultas para a base de dados TIGER/Line, mas apresenta melhores desempenhos nas consultas à base de dados BCIM. Nesta etapa, foi possível definir um referencial para avaliar o desempenho das consultas espaciais em desktop e assim extrair objetivos para pesquisas futuras.

O GeoDrill mostrou-se uma alternativa para integração de dados espaciais, podendo ser utilizado também como uma solução para migração entre tecnologias e realização de consultas *ad-hoc*. Também foi possível executar algumas consultas espaciais com bom desempenho em comparação ao PostGIS/PostgreSQL.

6.1 Contribuições

As principais contribuições deste trabalho foram:

- **GeoDrill:** extensão do Apache Drill que permite a realização de consultas espaciais em fontes de dados heterogêneas, possibilitando cruzar os dados destas fontes. O GeoDrill utiliza linguagem SQL padrão para as consultas e suporta as funções espaciais definidas pela OGC (*Open Geospatial Consortium*) e algumas adicionais existentes no PostGIS/PostgreSQL.
- **Arquitetura em camadas e classes DataCap:** foram definidas classes e uma arquitetura em camadas que auxilia o desenvolvimento e a manutenção das funções espaciais existentes. A arquitetura e as classes adicionadas pelo GeoDrill podem ser utilizadas por qualquer trabalho que vise estender o Apache Drill ou o GeoDrill, não sendo restritas ao contexto espacial.
- **Conjunto de consultas BCIM:** foram realizados vários experimentos que visam avaliar as funcionalidades e o desempenho do GeoDrill. Estes experimentos utilizam bases de dados atualizadas e uma gama variada de dados e consultas

espaciais. Este conjunto de experimentos pode ser utilizado para avaliar o GeoDrill após a adição de novas funcionalidades, como também para avaliar os trabalhos futuros que utilizarem o GeoDrill. É possível adaptar os experimentos para avaliar outras soluções que visam integração de dados.

6.2 Trabalhos futuros

Nesta seção são elencados possíveis trabalhos futuros com base nesta dissertação.

- Alguns trabalhos avaliam a possibilidade de utilizar índices dinâmicos em pesquisas espaciais. Como trabalho futuro, objetivando melhorar o desempenho da execução das consultas, propõe-se investigar e desenvolver mecanismos que possibilitem criar índices espaciais convencionais e dinâmicos. Visto a peculiaridade do GeoDrill não armazenar os dados consultados, avalia-se que a criação de índices espaciais dinâmicos possa oferecer grandes ganhos ao desempenho da solução.
- Quanto à integração de um SGBD espacial, faz-se necessário investigar metodologias que viabilizem a utilização de funções e índices espaciais nativos na arquitetura do GeoDrill. Assim, ao executar uma consulta, em um SGBD espacial, seria utilizado um índice local.
- Aliado à criação de índices espaciais dinâmicos, pretende-se pesquisar metodologias de particionamento de dados espaciais, visando melhorar o desempenho das consultas e a escalabilidade da solução.
- Pesquisar e avaliar novas técnicas de otimização de consultas a dados espaciais na plataforma GeoDrill.
- Incorporar novos formatos de dados espaciais como GML, KML e Shapefiles à plataforma GeoDrill.
- Incorporar dados espaciais não estruturados como Raster e Modelos de Elevação de Terreno à plataforma GeoDrill.

Referências Bibliográficas

[1] S. Chaudhuri, U. Dayal e V. Narasayya, “An overview of business intelligence technology.,” *Communications of the ACM 54.8,* pp. 88-98, 2011.

[2] C. Franklin, “An introduction to geographic information systems: linking maps to databases,” *Journal Database,* vol. 15, pp. 15-21, 1992.

[3] H. Garcia-Molina, J. D. Ullman e J. Widom, Database Systems: The Complete Book 2nd edition, Pearson Prentice Hall, 2008.

[4] R. Cattell, “Scalable SQL and NoSQL data stores.,” *ACM SIGMOD Record 39.4,* pp. 12-27, 2011.

[5] M. Stonebraker e U. Cetintemel, “" One size fits all": an idea whose time has come and gone.,” *21st International Conference on Data Engineering (ICDE'05). IEEE,* pp. 2-11, 2005.

[6] M. Stonebraker, “NewSQL: An Alternative to NoSQL and Old SQL for New OLTP Apps,” *Communications of the ACM. Retrieved,* pp. 07-06, 2012.

[7] H. Chen, R. H. Chiang e V. C. Storey, “Business Intelligence and Analytics: From Big Data to Big Impact.,” *MIS quarterly 36.4,* pp. 1165-1188, 2012.

[8] E. Turban, R. Sharda, J. E. Aronson e D. King, Business intelligence: A managerial approach, Prentice Hall, 2008.

[9] H. J. Watson e B. H. Wixom, “The current state of business intelligence.,” *Computer 40.9,* pp. 96-99, 2007.

[10] M. Franklin, A. Halevy e D. Maier, “From databases to dataspaces: a new abstraction for information management.,” *ACM Sigmod Record 34.4,* pp. 27-33, 2005.

[11] C. Koch, *Data integration against multiple evolving autonomous schemata,* Tese de Doutorado, Vienna U, 2001.

[12] S. Rivest, Y. Bédard, M.-J. Proulx, M. Nadeau, F. Hubert e J. Pastor, “SOLAP technology: Merging business intelligence with geospatial technology for interactive spatio-temporal exploration and analysis of data,” *ISPRS Journal of Photogrammetry and Remote Sensing,* vol. 60, pp. 17-33, 2005.

[13] C. de Souza Baptista, C. E. S. Pires, D. F. B. Leite e M. G. de Oliveiraa, “NoSQL Geographic Databases: An Overview,” em *Geographical Information Systems: Trends and Technologies*, CRC Press, 2014.

[14] M. Hausenblas e J. Nadeau, “Apache drill: interactive ad-hoc analysis at scale.,” *Big Data 1.2,* pp. 100-104, 2013.

[15] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton e T. Vassilakis, “Dremel: interactive analysis of web-scale datasets.,” *Proceedings of the VLDB Endowment 3.1-2,* pp. 330-339, 2010.

[16] P. Buneman, “Semistructured data.,” *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART*

symposium on Principles of database systems. ACM, pp. 117-121, 1997.

[17] D. Flanagan, JavaScript: the definitive guide, O'Reilly Media, Inc., 2006.

[18] R. Agrawal, A. Somani e Y. Xu, "Storage and querying of e-commerce data.," *VLDB,* vol. 1, pp. 149-158, 2001.

[19] Z. H. Liu, B. Hammerschmidt e D. McMahon, "JSON data management: supporting schema-less development in RDBMS.," *Proceedings of the 2014 ACM SIGMOD international conference on Management of data. ACM,* pp. 1247-1258, 2014.

[20] Z. H. Liu e D. Gawlick, "Management of Flexible Schema Data in RDBMSs-Opportunities and Limitations for NoSQL-," *CIDR,* 2015.

[21] R. CIFERRI, "Banco de Dados Espaciais," [Online]. Available: http://gbd.dc.ufscar.br/download/files/courses/NonConventionalDB_2010/RicardoCiferri_slidesBDE.pdf. [Acesso em 06 07 2016].

[22] "opengeospatial," [Online]. Available: https://portal.opengeospatial.org/files/?artifact_id=829. [Acesso em 02 07 2016].

[23] S. Chaudhuri e U. Dayal, "An overview of data warehousing and OLAP technology.," *ACM Sigmod record 26.1,* pp. 65-74, 1997.

[24] C. Koch, *Data integration against multiple evolving autonomous schemata,* Tese de Doutorado. Vienna U, 2001.

[25] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton e T. Vassilakis, "Dremel: interactive analysis of web-scale datasets.," *Proceedings of the VLDB Endowment 3.1-2,* pp. 330-339, 2010.

[26] M. Hausenblas e J. Nadeau., "Apache drill: interactive ad-hoc analysis at scale.," *Big Data 1.2,* pp. 100-104, 2013.

[27] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker e I. Stoica, "Spark: Cluster Computing with Working Sets," *HotCloud 10,* pp. 10-10, 2010.

[28] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley e M. Zaharia, "Spark sql: Relational data processing in spark.," *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM,* pp. 1383-1394, 2015.

[29] A. Aji, F. Wang, H. Vo, R. Lee, Q. Liu, X. Zhang e J. Saltz, "Hadoop GIS: a high performance spatial data warehousing system over mapreduce.," *Proceedings of the VLDB Endowment 6.11,* pp. 1009-1020, 2013.

[30] C. Franklin, "An introduction to geographic information systems: linking maps to databases," *Database, 15(2),* p. 12–21, 1992.

[31] A. Eldawy e M. F. Mokbel, "SpatialHadoop: A MapReduce framework for spatial data.," *Data Engineering (ICDE), 2015 IEEE 31st International Conference on. IEEE,* pp. 1352-1363, 2015.

[32] K. Shvachko, H. Kuang, S. Radia e R. Chansler, "The hadoop distributed file system.," *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on. IEEE,* 2010.

[33] A. Eldawy e M. F. Mokbel, “Pigeon: A spatial MapReduce language.,” *Data Engineering (ICDE), 2014 IEEE 30th International Conference on. IEEE,* 2014.

[34] A. Eldawy e M. F. Mokbel, “The era of big spatial data.,” *Data Engineering Workshops (ICDEW), 2015 31st IEEE International Conference on. IEEE,* pp. 42-49, 2015.

[35] L. Alarabi, A. Eldawy, R. Alghamdi e M. F. Mokbel, “TAREEG: A MapReduce-Based Web Service for Extracting Spatial Data from OpenStreetMap.,” *Proceedings of the 2014 ACM SIGMOD international conference on Management of data. ACM,* pp. 897-900, 2014.

[36] A. Eldawy, M. F. Mokbel, S. Alharthi, A. Alzaidy, K. T. Ghani e Sohaib, “SHAHED: A MapReduce-based System for Querying and Visualizing Spatio-temporal Satellite Data,” *Data Engineering (ICDE), 2015 IEEE 31st International Conference on. IEEE,* pp. 1585-1596, 2015.

[37] N. Zhang, G. Zheng, H. Chen, J. Chen e X. Chen, “Hbasespatial: A scalable spatial data storage based on hbase.,” *rust, Security and Privacy in Computing and Communications (TrustCom), 2014 IEEE 13th International Conference on. IEEE,* pp. 644-651, 2014.

[38] X. Zhang, W. Song e L. Liu, “An Implementation approach to store GIS spatial data on NoSQL database.,” *Geoinformatics (GeoInformatics), 2014 22nd International Conference on. IEEE,* pp. 1-5, 2014.

[39] J. Yu, J. Wu e M. Sarwat, “GeoSpark: A Cluster Computing Framework for Processing Large-Scale Spatial Data.,” 2015.

[40] D. Xie, F. Li, B. Yao, G. Li, L. Zhou e M. Guo, “Simba: Efficient In-Memory Spatial Analytics.,” *SIGMOD,* 2016.

[41] R. Miles e K. Hamilton, Learning UML 2.0, O'Reilly Media, Inc., 2006.

[42] “GeoTools,” [Online]. Available: http://www.geotools.org/. [Acesso em 02 05 2016].

[43] S. Zhang, J. Han, Z. Liu, K. Wang e S. Feng, “Spatial queries evaluation with mapreduce.,” *Grid and Cooperative Computing, 2009. GCC'09. Eighth International Conference on. IEEE,* pp. 287-292, 2009.

[44] S. Ray, B. Simion e A. D. Brown, “Jackpine: A benchmark to evaluate spatial database performance.,” *Data Engineering (ICDE), 2011 IEEE 27th International Conference on. IEEE,* pp. 1139-1150, 2011.

[45] “Relatório do instituto indiano de tecnologia,” [Online]. Available: http://www.gise.cse.iitb.ac.in/wiki/images/c/c4/Finalreport.pdf. [Acesso em 02 05 2016].

[46] “Apache Drill,” [Online]. Available: https://drill.apache.org/. [Acesso em 02 05 2016].

[47] “Postgres,” [Online]. Available: http://www.postgresql.org/. [Acesso em 02 05 2016].

[48] “EUA Census,” [Online]. Available: http://www2.census.gov/geo/pdfs/maps-data/data/tiger/tgrshp2014/TGRSHP2014_TechDoc.pdf. [Acesso em 02 05 2016].

[49] “IBGE,” [Online]. Available: ftp://geoftp.ibge.gov.br/mapeamento_sistematico/base_continua_ao_milionesimo/BCIM_V4_DocTecnica_VOL_I.pdf. [Acesso em 02 05 2016].

[50] G. K. Kanji, 100 statistical tests, Sage, 2006.

[51] N. Lewis, 100 Statistical Tests in R, Heather Hills Press, 2013.

[52] G. E. Box, J. S. Hunter e W. G. Hunter., Statistics for experimenters: design, innovation, and discovery, New York: Wiley-Interscience, 2005.

[53] S. Chaudhuri e U. Dayal, "An overview of data warehousing and OLAP technology.," *ACM Sigmod record 26.1,* pp. 65-74, 1997.

[54] S. Amer-Yahia e A. Halevy, "What does Web 2.0 have to do with databases?.," *Proceedings of the 33rd international conference on Very large data bases.,* pp. 1443-1443, 2007.

[55] "opengeospatial," [Online]. Available: https://portal.opengeospatial.org/files/?artifact_id=829.. [Acesso em 02 07 2016].

[56] Y. Wang e S. Wang, "Research and implementation on spatial data storage and operation based on Hadoop platform.," *Geoscience and Remote Sensing (IITA-GRS), 2010 Second IITA International Conference on,* vol. 2, pp. 275-278, 2010.

[57] M. Hadjieleftheriou, Y. Manolopoulos, Y. Theodoridis e V. J. Tsotras, "R-trees–a dynamic index structure for spatial searching.," *Encyclopedia of GIS. Springer US,* pp. 993-1002, 2008.

[58] H. Vo, A. Aji e F. Wang, "Sato: A spatial data partitioning framework for scalable query processing," *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems,* pp. 545-548, 2014.

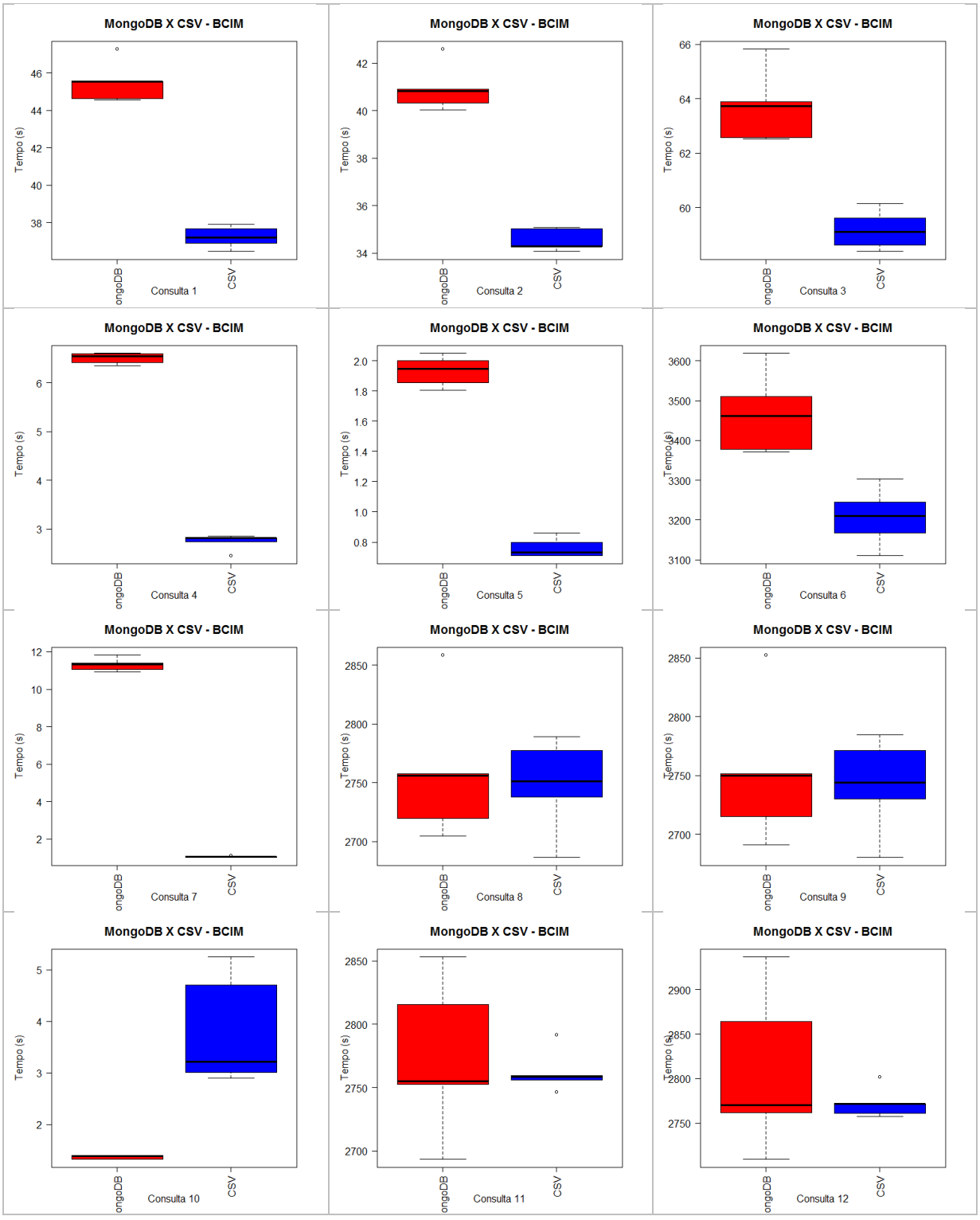
[59] A. Eldawy, L. Alarabi e M. F. Mokbel, "Spatial partitioning techniques in spatial hadoop," *Proceedings of the VLDB Endowment 8.12 ,* pp. 1602-1605, 2015.

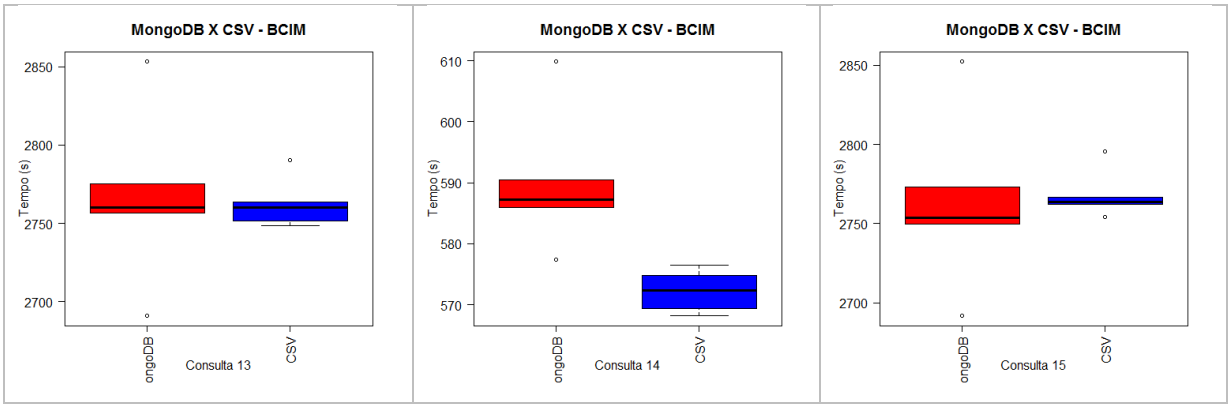
[60] M. Franklin, A. Halevy e D. Maier, "From databases to dataspaces: a new abstraction for information management.," *ACM Sigmod Record 34.4,* pp. 27-33, 2005.

Apêndice 1 – Gráficos Detalhados

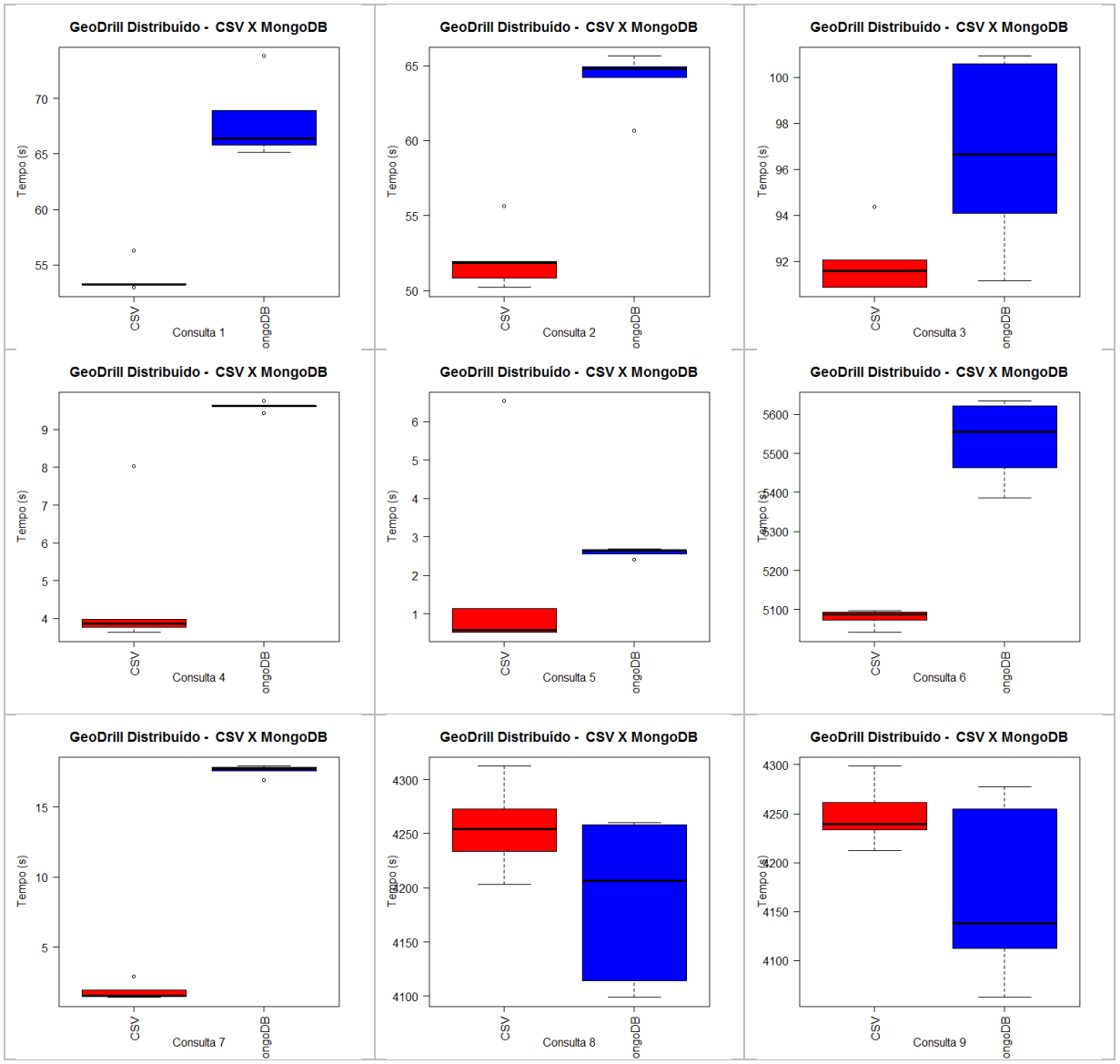
Neste apêndice são apresentados os gráficos de *boxplot* detalhados por consulta. Estes gráficos podem ser utilizados como primeira opção para as comparações entre cada etapa de teste.

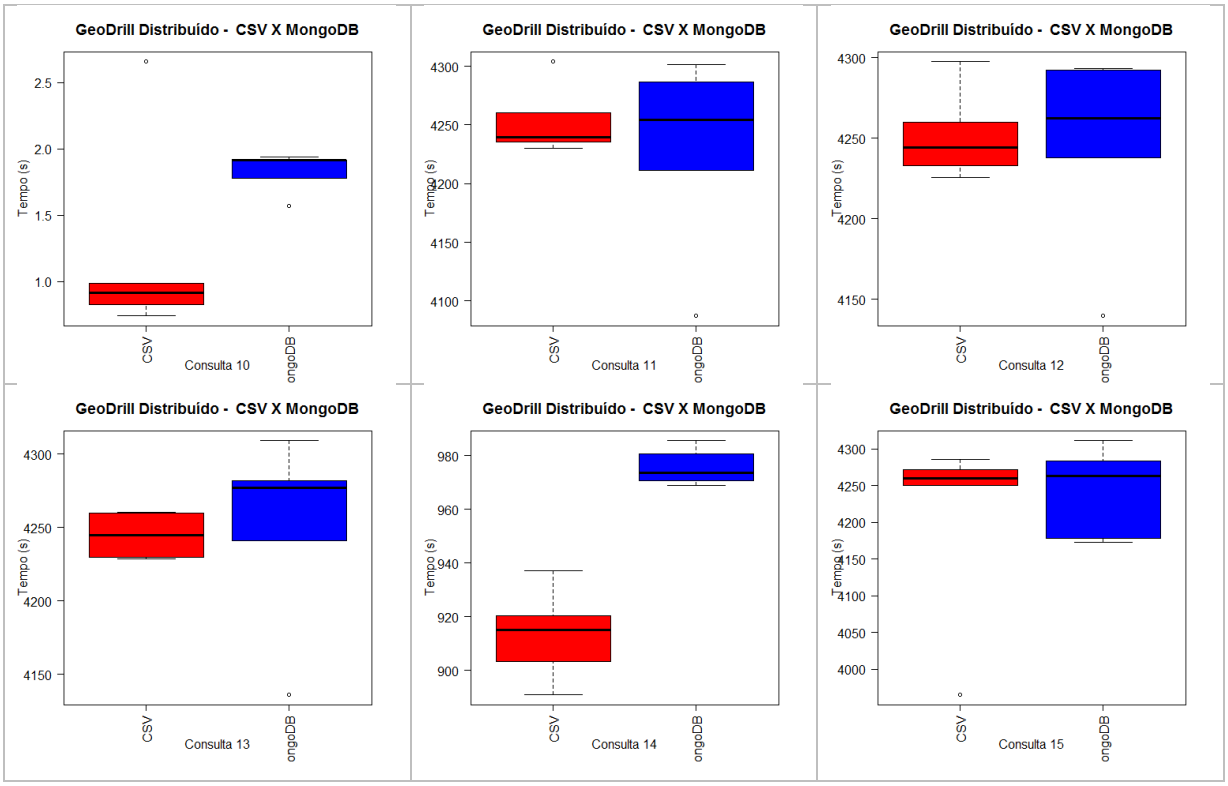
Quadro 8: Boxplot das consultas na base BCIM com GeoDrill utilizando MongoDB e CSV em ambiente Desktop



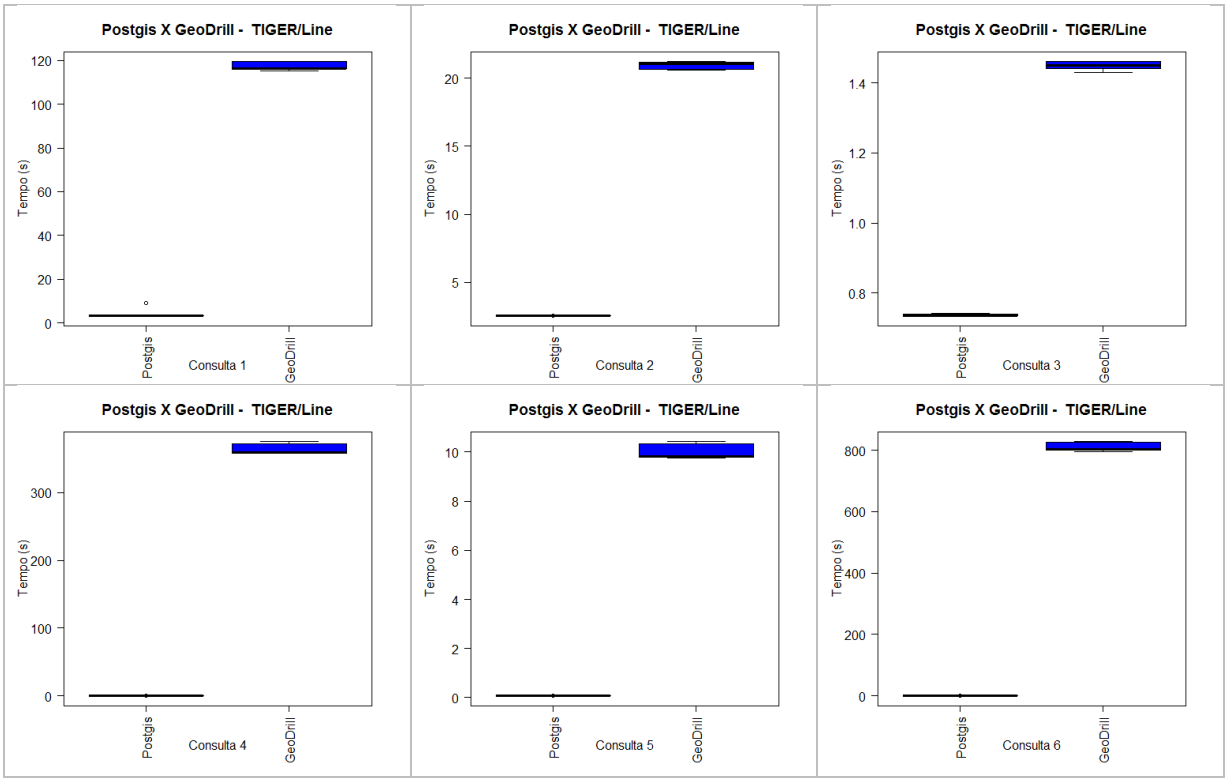


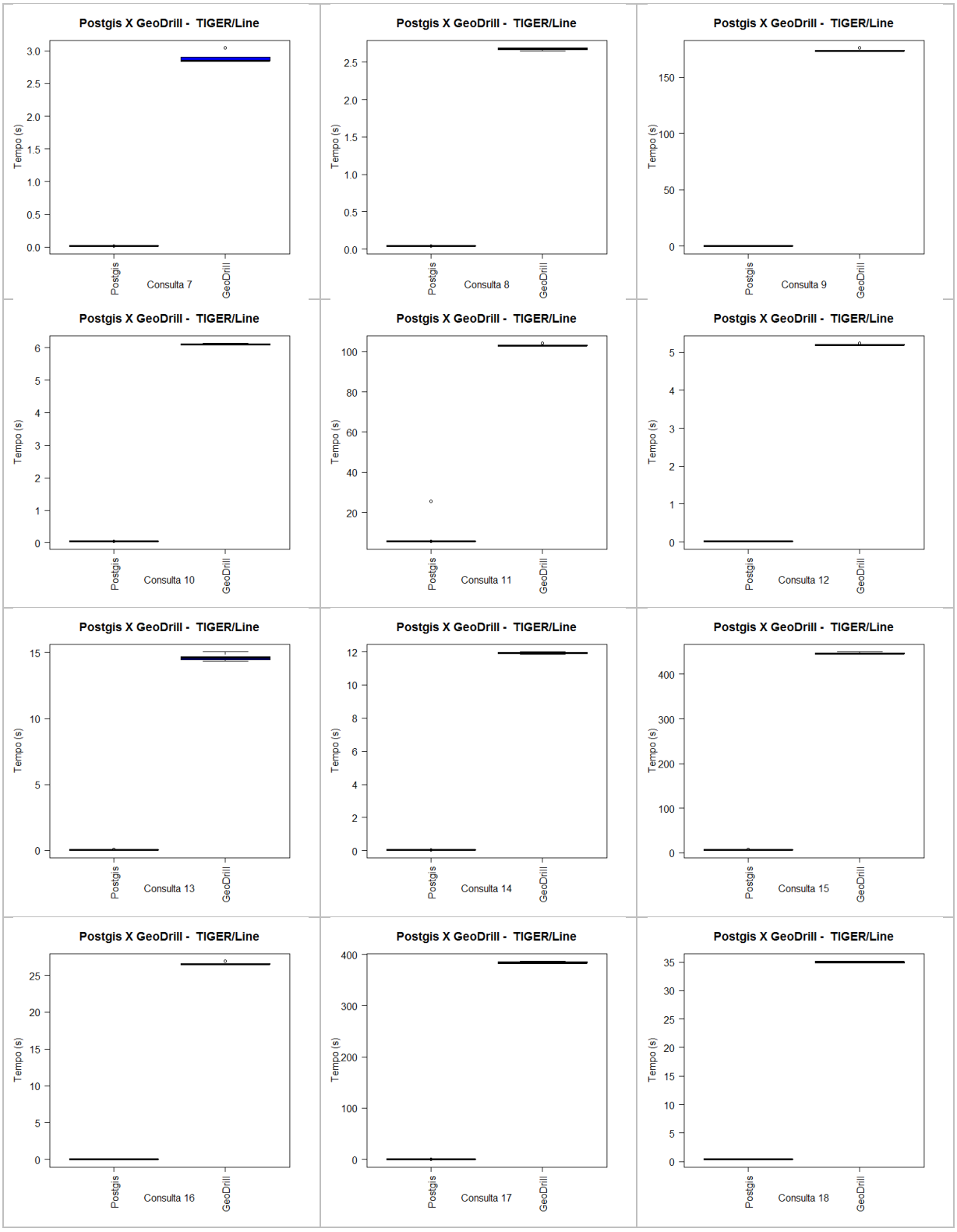
Quadro 9: Boxplot das consultas na base BCIM com GeoDrill distribuído utilizando MongoDB e CSV em ambiente distribuido

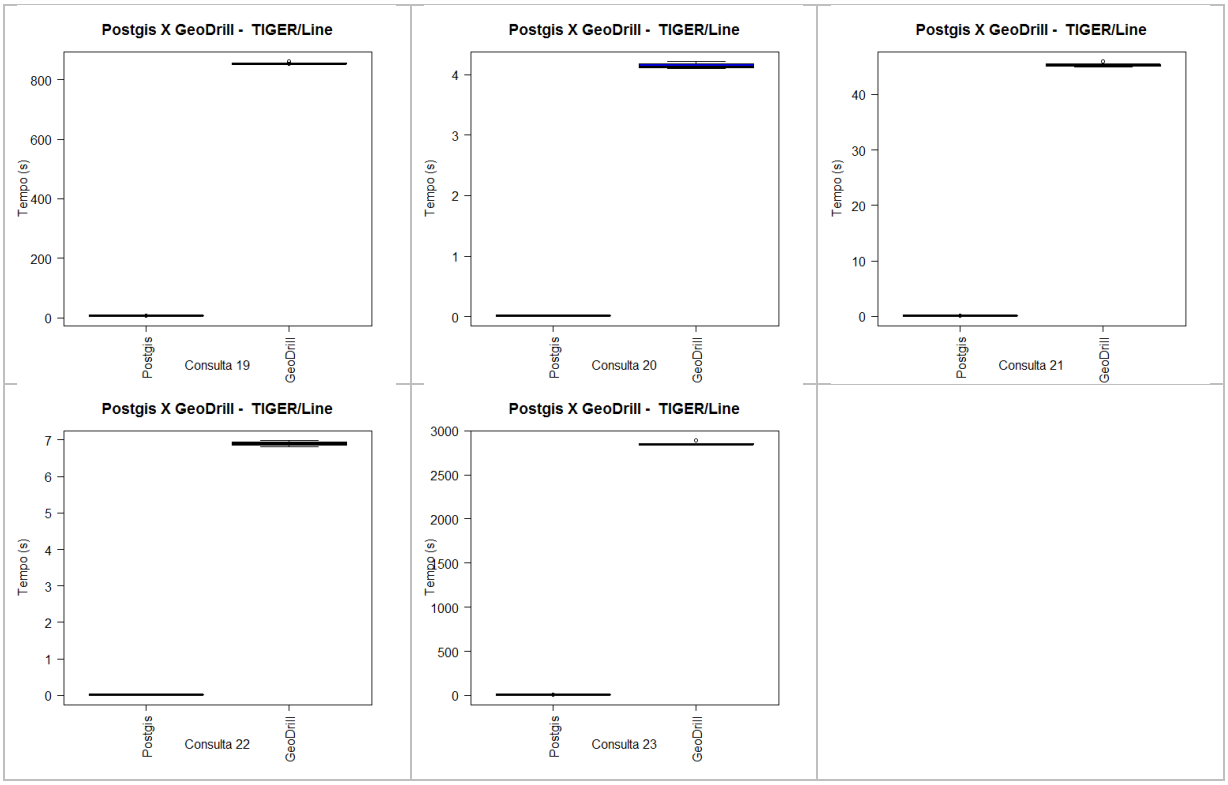




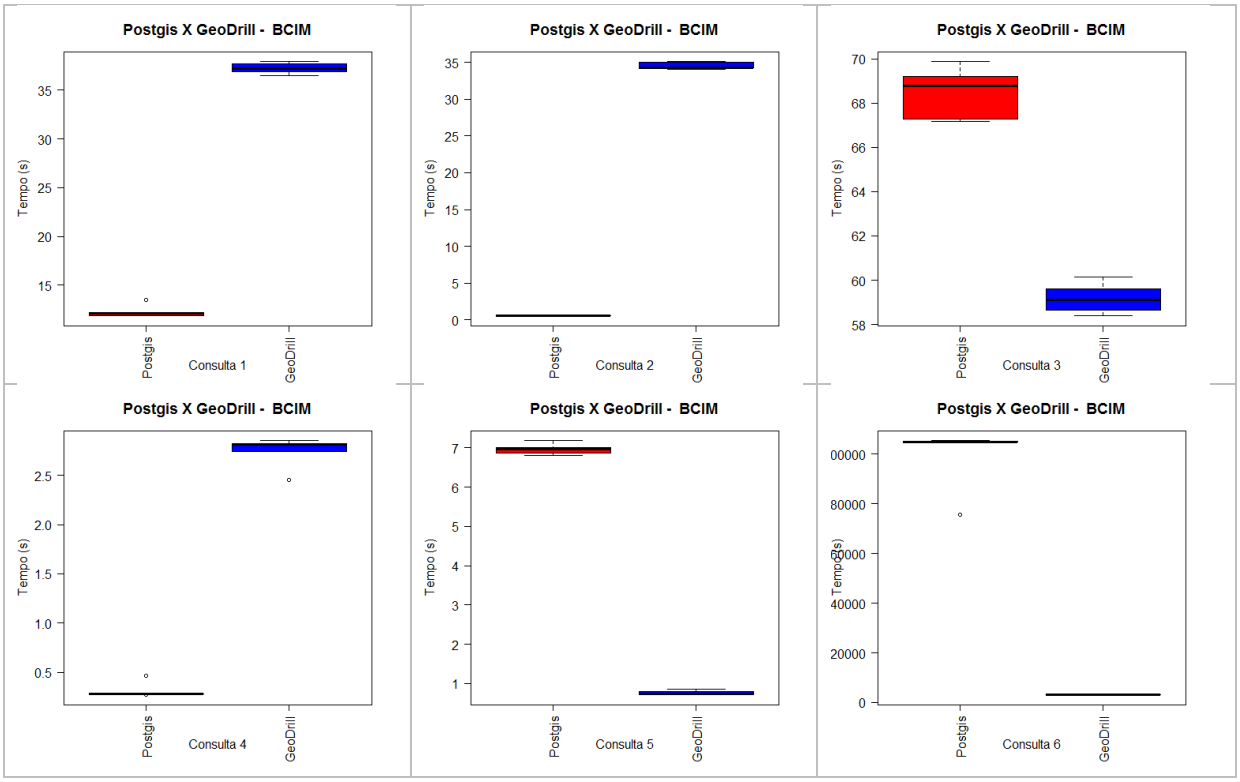
Quadro 10: Boxplot das consultas na base TIGER/Line entre PostGIS/PostgreSQL e GeoDrill em ambiente desktop

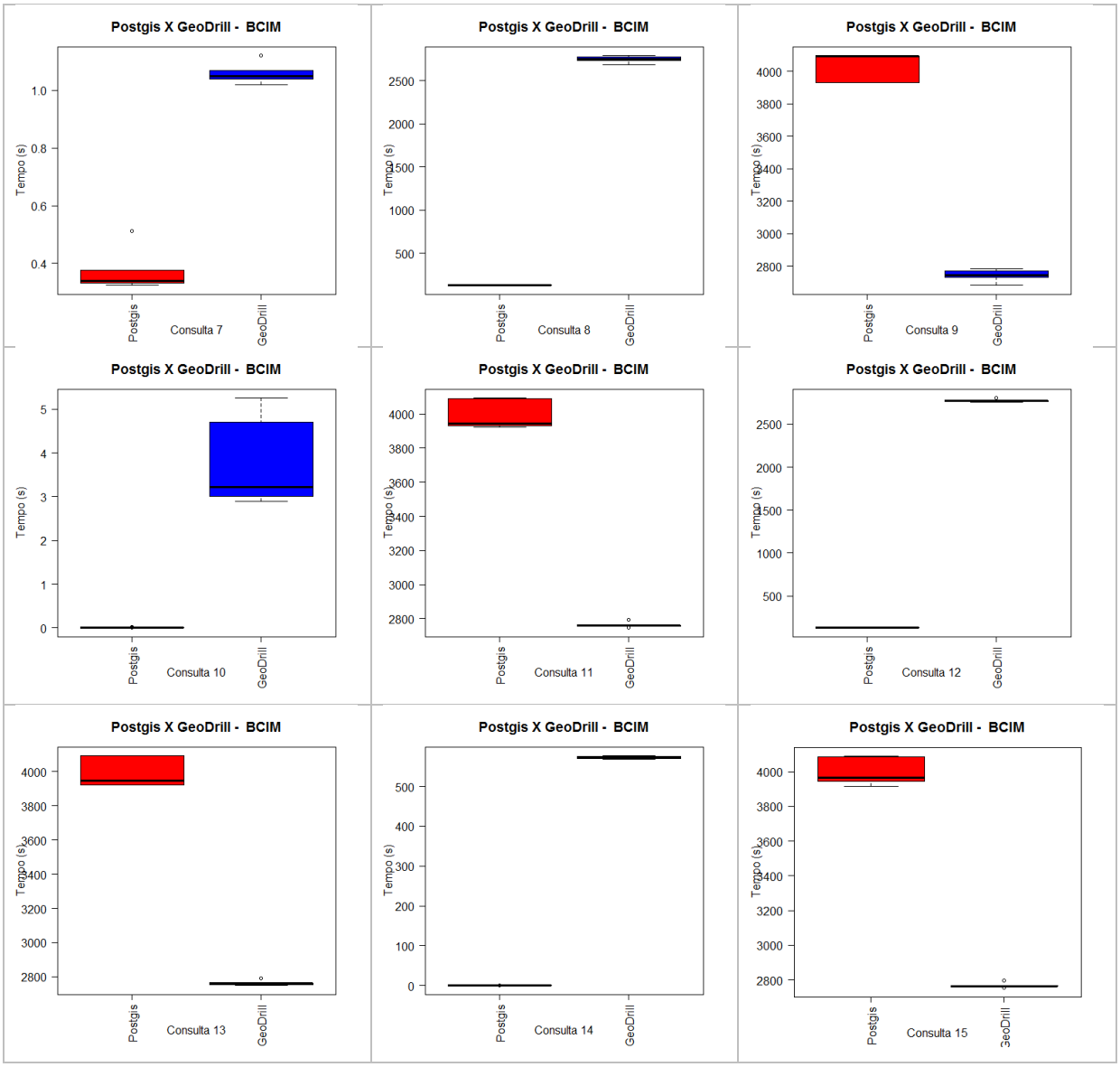



Postgis X GeoDrill - TIGER/Line
Tempo (s)
Postgis
GeoDrill
Consulta 7
Consulta 8
Consulta 9
Consulta 10
Consulta 11
Consulta 12
Consulta 13
Consulta 14
Consulta 15
Consulta 16
Consulta 17
Consulta 18

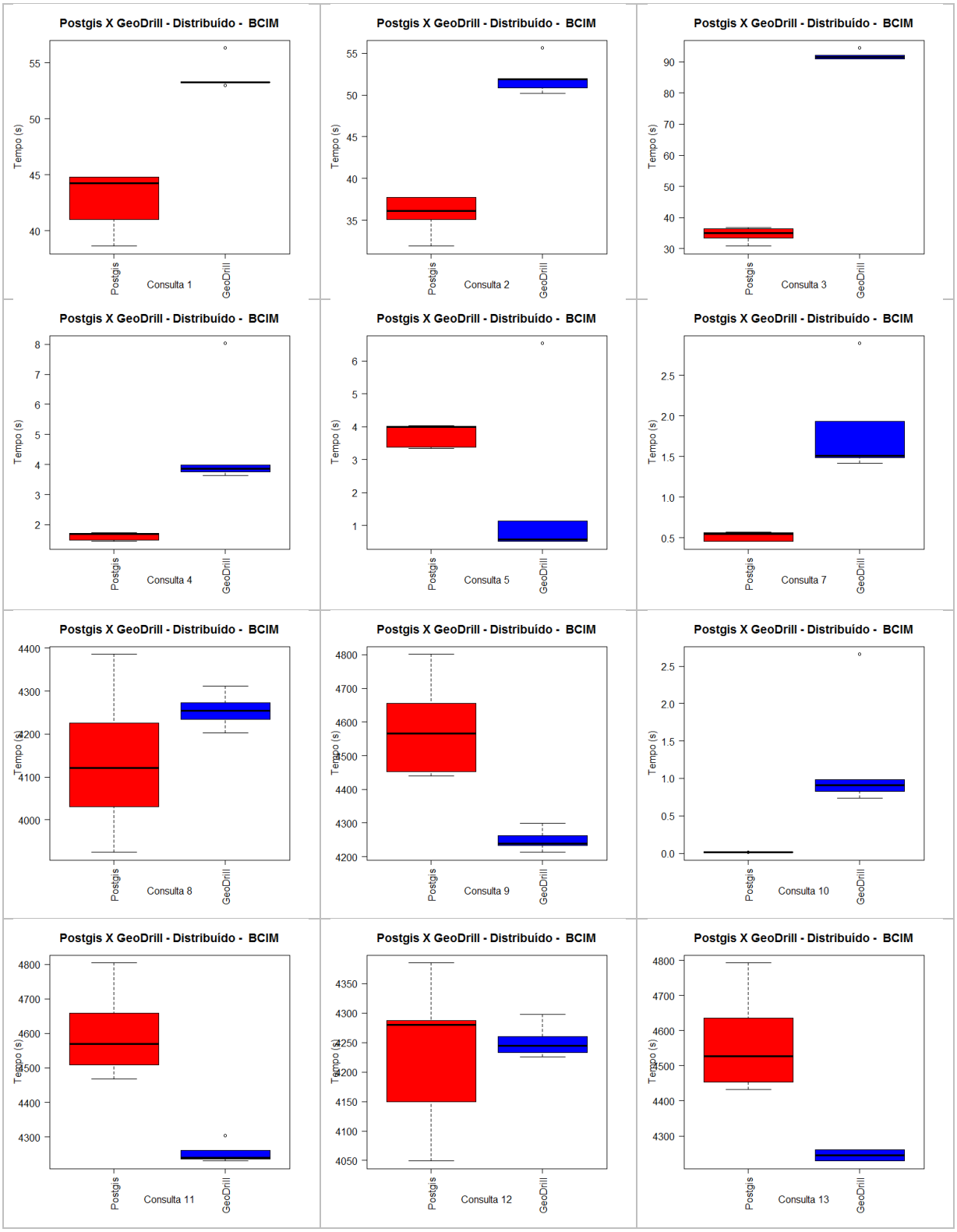


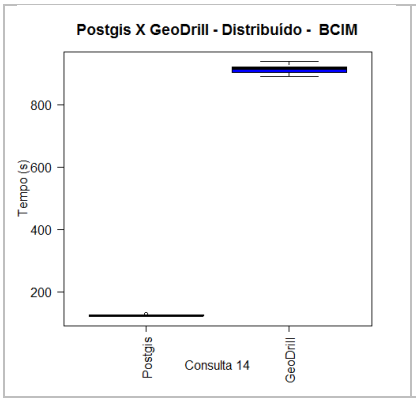
Quadro 11: Boxplot das consultas na base BCIM entre PostGIS/PostgreSQL e GeoDrill em ambiente desktop

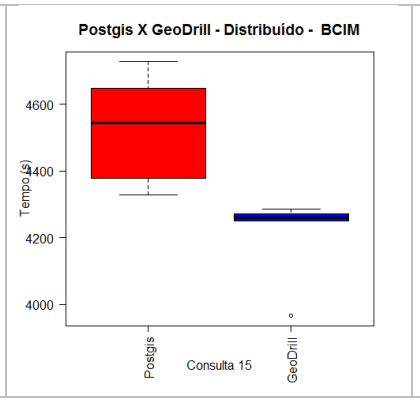



Postgis X GeoDrill - BCIM
Tempo (s)
Postgis
GeoDrill
Consulta 7
Consulta 8
Consulta 9
Consulta 10
Consulta 11
Consulta 12
Consulta 13
Consulta 14
Consulta 15

Quadro 12: Boxplot das consultas na base BCIM com GeoDrill e PostGIS/PostgreSQL em ambiente distribuído




Postgis X GeoDrill - Distribuído - BCIM
Tempo (s)
800
600
400
200
Postgis
GeoDrill
Consulta 14


Postgis X GeoDrill - Distribuído - BCIM
Tempo (s)
4600
4400
4200
4000
Postgis
GeoDrill
Consulta 15

Apêndice 2 – Comparações Estatísticas

As Comparações estatísticas foram realizadas somente com os resultados da segunda etapa dos experimentos, descritos na seção 5.2. Nesta etapa, foi avaliado se existe diferença nos tempos de execução das consultas dependendo da fonte de dados utilizada. Descrevem-se a seguir os testes estatísticos para os resultados obtidos, comparando duas fontes de dados distintas (CSV ou MongoDB) nos dois ambientes propostos.

Os testes estatísticos e os gráficos apresentados neste apêndice foram realizados no software R[36] de computação estatística. As análises adotadas tomam como referência livros sobre métodos estatísticos e também sobre como aplica-los utilizando o R [50] [51] [52].

GeoDrill – CSV X MongoDB em ambiente desktop

As comparações utilizando os gráficos de *boxplot* presentes no Quadro 8 indicam que as consultas de 1 a 7 e 14 obtiveram melhor desempenho utilizando a fonte CSV. Observa-se ainda que consulta 10 obteve melhores resultados no MongoDB. Para as demais consultas que apresentaram intersecção, foram realizados testes estatísticos.

Para as consultas 8, 9, 11, 12, 13, e 15, realizou-se inicialmente um teste de normalidade dos dados (Shapiro-Wilk[37]). Utilizando uma confiança de 95%, sendo α=0,05, nenhum dos resultados apresentou *p-value* menor que o α. Desta forma, não se rejeita a hipótese nula do teste Shapiro-Wilk, podendo assim considerar os dados normais. Os resultados podem ser visualizados na Tabela 3.

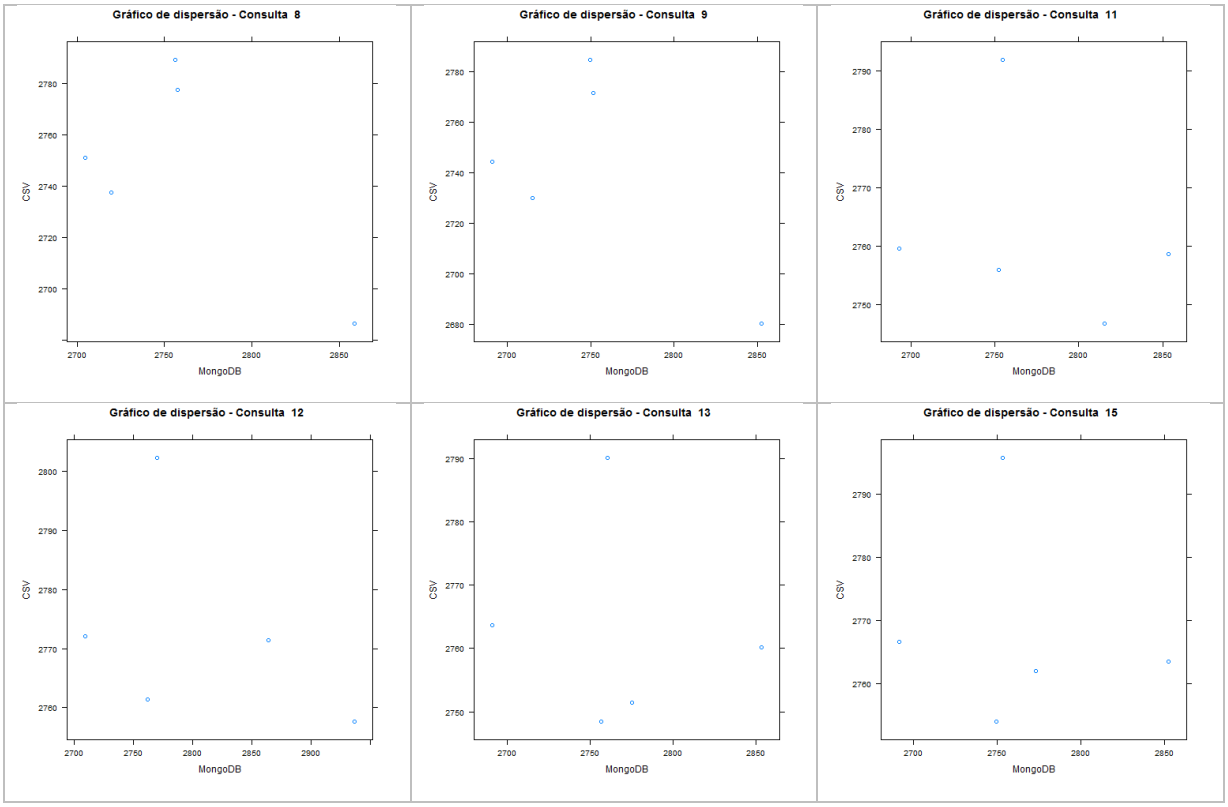
Tabela 3: Resultado do teste de normalidade para os dados das consultas

MongoDB				CSV			
Consulta	*P-value*	Consulta	*P-value*	Consulta	*P-value*	Consulta	*P-value*
c8	0.2288	c12	0.6357	c8	0.6502	c12	0.1769
c9	0.3496	c13	0.6586	c9	0.724	c13	0.2388
c11	0.8254	c15	0.6733	c11	0.09733	c15	0.1036

Após o teste de normalidade, foram gerados gráficos de dispersão dos dados, estes auxiliam na identificação de correlação entre os dados. Os gráficos de dispersão dos dados podem ser visualizados no Quadro 13.

[36] The R Project - https://www.r-project.org/
[37] Shapiro-Wilk - https://stat.ethz.ch/R-manual/R-devel/library/stats/html/shapiro.test.html

Quadro 13: Gráficos de correlação simples para os dados das consultas



Através dos gráficos de dispersão exibidos no Quadro 13, é possível perceber (visualmente) que não existem indícios de que há correlação entre os fatores, conforme esperado. Desta forma, pode-se seguir utilizando um teste T de Student[38]. Antes do teste, foi avaliada se cada amostra possui variância homogênea. É necessário avaliar a variância dos dados é homogênea, pois esta informação é um parâmetro para o teste T. Utilizando confiança de 95%, a hipótese da variância homogênea não foi descartada das consultas 8 e 9. Estes resultados podem ser visualizados na Tabela 4.

Tabela 4: Resultado do teste de variância homogênea

Teste F			
Consulta	P-value	Consulta	P-value
c8	0.4579	c12	0.007604
c9	0.443	c13	0.03226
c11	0.02912	c15	0.02872

Por fim são apresentados os dados do teste T utilizando as hipóteses:

H0: os dados apresentam médias iguais

H1: os dados apresentam médias distintas

[38] Teste T - https://stat.ethz.ch/R-manual/R-devel/library/stats/html/t.test.html

Em nenhuma das consultas a hipótese nula foi refutada. Sendo assim, é considerado que não existe diferença estatística entre os resultados, ou seja, é possível considerar os resultados destas consultas como iguais. O resultado do teste T de Student pode ser visualizado na Tabela 5.

Tabela 5: Resultado do teste T

Teste T					
Consulta	P-value	Teste	Consulta	P-value	Teste
c8	0.7441	Two Sample	c12	0.4362	Welch Two Sample
c9	0.7722	Two Sample	c13	0.8733	Welch Two Sample
c11	0.7145	Welch Two Sample	c15	0.8783	Welch Two Sample

GeoDrill comparação entre Mongo e CSV em ambiente distribuído

Utilizando as comparações proporcionadas pelos gráficos de *boxplot* do Quadro 9, é possível inferir que o GeoDrill obteve melhor desempenho para o tipo CSV no ambiente distribuído executando as consultas de 1 a 7, 10 e 14. As demais consultas apresentaram interseção, o que requer uma avaliação baseada em testes estatísticos.

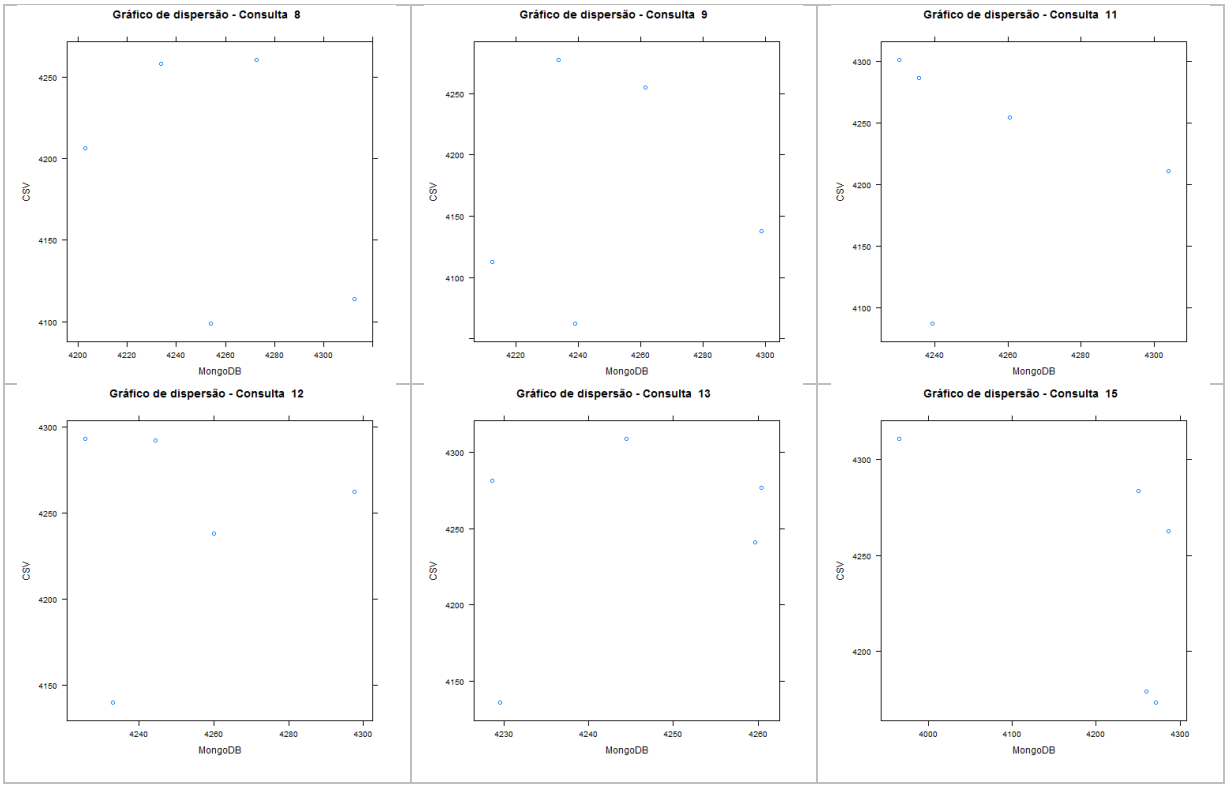
O primeiro teste a ser realizado é de normalidade dos dados (Shapiro-Wilk). Os resultados do teste de normalidade podem ser visualizados na Tabela 6.

Tabela 6: Resultado do teste de normalidade para os dados das consultas

MongoDB				CSV			
Consulta	P-value	Consulta	P-value	Consulta	P-value	Consulta	P-value
c8	0.1796	c12	0.1347	c8	0.997	c12	0.4434
c9	0.45	c13	0.1989	c9	0.7711	c13	0.166
c11	0.2677	c15	0.2785	c11	0.1307	c15	0.0025

Utilizando uma confiança de 95% e tendo alfa como 0,05, somente a consulta 15 apresentou *p-value* menor que o alfa, ou seja, rejeitou a hipótese nula do teste Shapiro-Wilk, o que implica a não normalidade dos dados desta consulta. Deste modo, essa consulta não será considerada no teste de correlação. O teste de correlação foi então realizado com as demais consultas, por meio de gráficos de dispersão que podem ser visualizados no Quadro 14.

Quadro 14: Gráficos de correlação simples para os dados das consultas



Analisando visualmente os gráficos de dispersão no Quadro 14, não foi possível identificar nenhuma relação linear entre os resultados plotados. Assim, é possível considerar que não existe correlação entre os dados. Desse modo, foi realizado um teste de variância dos dados e, em seguida, um teste T de Student.

Para o teste de variância, foi utilizada um fator de confiança de 95%. Somente a variância da consulta 13 não pôde ser considerada homogênea, pois o *p-value* é menor que 5%, assim a hipótese nula foi refutada. Estes resultados podem ser visualizados na Tabela 7.

Tabela 7: Resultado do teste de variância homogênea

Teste F			
Consulta	P-value	Consulta	P-value
c8	0.2482	c12	0.1516
c9	0.06684	c13	0.01437
c11	0.06667		

Por fim, são apresentados os dados do teste T utilizando as hipóteses:

H0: os dados apresentam médias iguais

H1: os dados apresentam médias distintas

Em nenhuma das consultas foi possível refutar a hipótese nula do teste T, que indica que os resultados são iguais, estatisticamente. Deste modo, pode-se considerar que os resultados para estas consultas, tanto para CSV quanto para MongoDB, são iguais estatisticamente. Os resultados dos testes T são apresentados na Tabela 8.

Tabela 8: Resultado do teste T

Teste T					
Consulta	P-value	Teste	Consulta	P-value	Teste
c8	0.1232	Two Sample	c12	0.8267	Two Sample
c9	0.1077	Two Sample	c13	0.8943	Welch Two Sample
c11	0.5458	Two Sample			