ELABORATION D'UN MOTEUR DE TRAITEMENT DES DONNEES SPATIALES MASSIVES VECTORIELLES OPTIMISANT L'INDEXATION SPATIALE

Mémoire

Jonathan Engélinus

Maîtrise en sciences géographiques
Maître en sciences géographiques (M. Sc. géogr.)

Québec, Canada

ELABORATION D'UN MOTEUR DE TRAITEMENT DES DONNEES SPATIALES MASSIVES VECTORIELLES OPTIMISANT L'INDEXATION SPATIALE

Mémoire

Jonathan Engélinus

Sous la direction de:

Thierry Badard, directeur de recherche

Résumé

Les données massives se situent au cœur de beaucoup d'enjeux scientifiques et sociétaux, et leur volume global ne cesse de croître. Il devient donc crucial de disposer de solutions permettant leur traitement et leur analyse. Hélas, alors qu'une majorité de ces données intègrent une composante spatiale vectorielle, peu de systèmes sont à même de gérer cette dernière. En outre, les rares prototypes qui s'y essaient respectent mal les standards ISO et les spécifications OGC et présentent des performances limitées. La présente recherche visait donc à déterminer comment gérer les données spatiales massives vectorielles de façon plus complète et efficiente. Il s'agissait en particulier de trouver une façon de les indexer avec une bonne scalabilité horizontale, d'assurer leur compatibilité avec la norme ISO-19125 et ses extensions, et de les rendre accessibles depuis les logiciels SIG. Il en résulte le système Elcano, une extension spatiale au système de gestion de données massives Spark qui fournit des performances accrues par rapport aux solutions du marché.

Abstract

Big data are in the midst of many scientific and economic issues. Furthermore their volume is continuously increasing. As a result, the need for management and processing solutions has become critical. Unfortunately, while most of these data have a vectorial spatial component, almost none of the current systems are able to manage it. In addition, the few systems who try either do not respect the ISO standards and OGC specifications or show poor performances. The aim of this research was then to determine how to manage the vectorial massive data more completely and efficiently. The objective was to find a scalable way of indexing them, ensuring their compatibility with ISO-19125 and its extensions, and making them accessible from GIS. The result is the Elcano system. It is an extension of the massive data management system Spark which provides increased performance compared to current market solutions.

Table des matières

Liste des tableaux

Liste des figures

Remerciements

Ce projet de recherche a été mené grâce au soutien et au financement du Centre de Recherche en Géomatique, et en particulier du professeur Thierry Badard. Il a aussi bénéficié de la bourse Fond d'héritage Géoïde 2015 et de la bourse Géomatique 2016 offerte par l'Association Québecoise des Sciences Géomatiques. Je tiens donc à remercier ces deux organismes pour leur soutien financier ainsi que pour leur reconnaissance envers mon travail. Je remercie mon directeur, Thierry Badard, pour avoir guidé mes recherches tout au long de cette maîtrise sans jamais se départir de sa sympathie, de son sens de l'humour et d'une expertise conséquente. Je lui suis également reconnaissant d'avoir permis que je participe à trois colloques en septembre et octobre 2016, et d'avoir aidé à la présentation de mes résultats préliminaires à ces occasions. Mes remerciements vont également aux professeurs Sara Irina Fabrikant, Ha Pham et Yvan Bédard, pour avoir répondu à certaines de mes questions et pour m'avoir fourni des conseils clés lors de ces colloques. Merci également aux professeurs Frédéric Hubert et Eric Guilbert pour la bienveillance avec laquelle ils ont accepté d'assurer l'examen et la relecture de ce mémoire. J'aimerais aussi témoigner toute ma reconnaissance à Eveline Bernier du CRG et à Valérie Cloutier et Danielle Goulet du SCG pour leur grande disponibilité et leur professionnalisme. Merci également à Louis-Etienne Guimond, à Jonathan Gagnon, à Mohamed Ali Chouaer, à Teriitutea Quesnot et au professeur Eric Guilbert pour m'avoir accompagné lors de mes premiers pas dans l'enseignement. J'adresse aussi un merci à mes collègues étudiants avec qui j'ai partagé de bons moments, entre autres à Kamal, Hervé et Serges, ainsi qu'aux autres étudiantes et étudiants en géomatique de l'Université Laval. Ma reconnaissance va aussi à mes collègues de l'Université de Cergy : merci à Laurent, Alain, Jean-Pierre, Violaine et Marc pour leur sympathie, leur compréhension et leur soutien tout au long de ce processus de recherche. Merci également à mes amis Pierrot Seban (qui m'a bien aidé à trouver les dernières coquilles de ce mémoire), à Brigitte Dumont, Fabian di Maria, Kevin Blau, Rodrigo Malmsten, Nathalie Gassel, Olivier Vanderaa, Cecilia Invertadi, Yoann Martinez, Anais Fauche, Nina Bonnefoy et Ben Coudert pour leur soutien de chaque instant. La curiosité avec laquelle ils ont suivi mes travaux n'a d'égale que la patience avec laquelle ils ont accepté l'éloignement qui en résultait. J'aimerais aussi m'adresser de façon posthume aux poètes Luc-André Rey, Benjamin Pottel et Benoit Proost pour les remercier d'avoir été de mes amis le temps que la vie l'a permis. Votre joie communicative et vos vers magnifiques ont souvent servi de renfort à mes nuits d'écriture solitaire. Un grand merci également à mes parents Huguette et Claude, pour avoir accepté que je m'expatrie et consacre plus d'une année à ce projet d'études malgré le chagrin que cet éloignement causait. Mon

grand frère Fabrice a également ma reconnaissance pour m'avoir très tôt donné envie d'en apprendre davantage sur les sciences, la programmation et la géographie. Enfin, je tiens à exprimer toute ma reconnaissance à mon amoureuse, Geneviève Marceau Vacchino, pour sa présence continuelle et pour avoir accepté que je sois parfois si peu disponible au cours de cette année. J'ai beaucoup de chance de t'avoir à mes côtés, et le réconfort que tu m'as apporté tout au long de ce projet m'est très précieux. Pour terminer, ma gratitude va à travers les siècles au navigateur Juan Sebastián Elcano, pour avoir trouvé la force et l'audace de terminer le tour du monde commencé par Magellan ainsi que pour m'avoir inspiré le nom du système développé dans le cadre de cette recherche.

Avant-propos

Ce mémoire intègre trois articles, qui sont en cours de soumission auprès de différentes revues de géomatique. Ils constituent le corps des chapitres 2, 3 et 4. Leur auteur principal (Jonathan Engélinus) est également l'auteur de ce mémoire. Le coauteur des articles, Thierry Badard, est le directeur du projet de maîtrise ; il a joué le rôle de conseiller et de relecteur tout au long de leur rédaction.

Le texte des articles est identique à celui proposé pour publication. Ils n'ont subi que quelques adaptations quant à leur mise en forme, afin de s'intégrer harmonieusement au reste du mémoire (numérotation des sections, structure de la bibliographie, etc.).

Voici un récapitulatif des articles insérés :

Article du chapitre 2 :

[43] Engélinus, J. et Badard, T., 2017, « Elcano : un système de gestion des données massives à composante spatiale basé sur Spark SQL », Revue internationale de géomatique (en cours de soumission).

Auteurs : Jonathan Engélinus, Thierry Badard

Résumé : Les données massives se situent au cœur de beaucoup d'enjeux scientifiques et sociétaux, et leur volume global ne cesse de croître. Alors que la plupart d'entre elles intègrent une composante spatiale vectorielle, il existe peu de systèmes de gestion de ces données à même de tenir compte de celle-ci. Ainsi, alors que Spark est peut-être à l'heure actuelle l'environnement de gestion des données massives le plus efficace et le plus facile à étendre, il n'est employé que par cinq systèmes gérant l'analyse de données spatiales vectorielles. Parmi ces solutions, aucune ne respecte entièrement les standards ISO et les spécifications OGC en termes de traitements spatiaux, et plusieurs présentent des performances limitées ou un manque d'extensibilité. Dans cet article, les auteurs cherchent une façon de dépasser ces limitations. Après une étude détaillée des lacunes des solutions existantes, ils définissent donc un système plus respectueux des standards. La solution proposée, Elcano, est une extension de Spark permettant le requêtage de données spatiales et compatible avec la norme ISO-19125.

Article du chapitre 3 :

[45] Engélinus, J. et Badard, T., 2017, « Vers une indexation des données spatiales massives perfor-

mante et efficace », Geomatica (en cours de soumission).

Auteurs : Jonathan Engélinus, Thierry Badard

Résumé : Les données massives se situent au cœur de beaucoup d'enjeux scientifiques et sociétaux, et leur volume global ne cesse de croître. Il devient donc crucial de disposer de solutions permettant leur traitement rapide et leur analyse efficace. Hélas, alors qu'une majorité de ces données intègrent une composante spatiale vectorielle, peu de systèmes sont à même de gérer cette dernière. En outre, les rares prototypes qui s'y essaient présentent des performances limitées en termes d'indexation spatiale. La présente recherche vise à déterminer comment améliorer cette situation. Une revue de littérature portant sur les types d'indexation spatiale et leur utilisation dans un contexte de données massives est donc entrepris dans un premier temps. Ensuite, l'intégration des solutions jugées les plus efficientes au système Elcano est décrite. Finalement, des tests sur la vitesse d'exécution de celui-ci démontrent qu'elle surpasse celles des solutions du marché.

Article du chapitre 4 :

[44] Engélinus, J. et Badard, T., 2017, « Vers un modèle unifié de la composante spatiale vectorielle et son application à Elcano », à traduire en anglais pour soumission dans Journal of GIS.

Auteurs : Jonathan Engélinus, Thierry Badard

Résumé : Les données massives se situent au cœur de beaucoup d'enjeux scientifiques et sociétaux, et leur volume global ne cesse de croître. Mais si beaucoup d'entre elles comportent une composante spatiale, peu de systèmes sont à même de gérer efficacement celle-ci. En outre, aucune solution ne respecte entièrement les standards ISO et les spécifications OGC qui décrivent la façon de gérer la composante spatiale. Le système Elcano conçu par le Centre de Recherche en Géomatique de l'Université Laval s'approche d'un tel objectif, puisqu'il gère de façon efficiente toutes les géométries et fonctions d'analyse spatiale décrites par la norme ISO-19125. Mais il n'en reste pas moins largement perfectible : la norme sur laquelle il repose a plus de dix ans et se limite aux données spatiales vectorielles bidimensionnelles, alors que d'autres structures comme les géométries en 2.5D et les TIN sont aujourd'hui employées couramment. Le présent article définit un modèle unifié de la composante spatiale qui intègre certaines de ces nouvelles structures au système Elcano.

Introduction

Contexte

Enjeux des données massives

Les données informatiques occupent un espace de plus en plus incontournable dans le monde contemporain. Ainsi, Walmart traite quotidiennement plus de 2.4 petabytes (soit $2.4 * 10^{15}$ octets) de données, et Google dix fois plus [83]. Elles prennent des formes variées : vidéos échangées en ligne, forums, e-mails, flux Twitter et Facebook, téléphones intelligents, relevés de capteurs GPS... Lorsque la complexité de ces données est telle qu'il devient impossible de les traiter à partir de bases de données classiques, on parle de « mégadonnées », « Big Data » ou « données massives » [99]. C'est la dernière appellation qui sera retenue ici.

L'impossibilité de gérer les données massives de façon classique est due à leurs spécificités, que Laney résumait en 2001 par les « 3V » [77] : Volume, Vélocité et Variété. Autrement dit, ces données occupent un espace de stockage trop important, demandent un traitement trop rapide ou sont de natures trop diverses pour être gérées comme des données ordinaires. Cette définition est citée dans plus de 300 travaux, et certains auteurs y ajoutent deux autres V : Valeur et Véracité [30, 90]. Ils veulent alors signifier que ces données peuvent aussi être tellement précieuses qu'elles nécessitent un mode de collecte spécifique, ou tellement douteuses qu'elles demandent des vérifications adaptées.

L'amélioration de la gestion des données massives au cours des dernières années a permis des progrès spectaculaires. Par exemple, alors qu'il fallait plusieurs années en 2003 pour décrypter le génome humain, cela ne prenait plus que quelques jours en 2012 [35]. Aujourd'hui, les données massives commencent même à jouer un rôle prédictif quant à la survenue de certaines maladies [24]. Elles connaissent également des applications nombreuses dans des domaines aussi variés que le commerce [84], la géographie [57], le journalisme [58] et la physique [79]. Pour plusieurs auteurs, cette évolution marquerait l'émergence d'un nouveau paradigme scientifique, dans lequel les découvertes sont guidées par les données plutôt que par la théorie [85, 4]. Sans aller jusque-là, il apparait comme crucial de pouvoir gérer les données massives de façon optimale, ce qui implique de disposer de systèmes informatiques adaptés à leur complexité.

L'écosystème Hadoop maintenu par la Fondation Apache est aujourd'hui un standard de fait en matière de gestion des données massives. Il est adopté par des géants de l'informatique comme Oracle et IBM, tout en continuant à participer au succès de plusieurs startups [48]. Open source, il rassemble une vaste communauté de développeurs et s'est vu ajouter au cours du temps un grand nombre d'extensions plus spécialisées qui constituent son écosystème. Le succès d'Hadoop peut s'expliquer par son architecture, qui permet le traitement rapide d'énormes volumes de données. Celle-ci repose en effet sur une « grappe » de serveurs (ou cluster) qui se répartissent le stockage et le traitement des données. De cette façon, les données sont disséminées sur l'ensemble de la grappe, dans des fichiers HDFS [1] auxquels sont appliqués des traitements en parallèle. Une telle découpe permet une bonne

1. Acronyme d'Hadoop Distributed File System, qui est un système de gestion distribuée des fichiers fourni nativement avec Hadoop.

scalabilité horizontale des systèmes reposant sur Hadoop, c'est-à-dire qu'il est possible d'y adapter les performances en fonction des besoins sans pour autant augmenter la puissance individuelle des machines. Il suffit en effet d'ajouter des serveurs au cluster pour que le parallélisme des traitements soit renforcé et permette une plus grande puissance de calcul, la vitesse d'exécution étant idéalement réduite de moitié lorsque le nombre de machines double [120].

Bien que dominant sur le marché de la gestion des données massives, Hadoop commence à rencontrer une certaine concurrence. Ainsi en 2011, la société LexisNexis a lancé le système HPCC, qui se révèle parfois plus rapide qu'Hadoop [106] même si c'est au prix d'une incompatibilité avec celui-ci. L'Université de Berkeley a par contre proposé en 2014 un environnement entièrement compatible avec Hadoop mais pouvant également s'utiliser sans lui. Il s'agit de Spark, qui est depuis devenu un des projets les plus importants de la Fondation Apache. Spark privilégie le recours à la mémoire vive pour stocker les données de ses traitements, ce qui limite les accès disque et le rend jusque dix fois plus rapide qu'Hadoop [128] dans les cas classiques et jusqu'sept fois plus rapide pour le traitement de données spatiales [124] [2].

Néogéographie, localisation et données massives

Apparus à partir des années 1970, de nombreux systèmes d'information géographiques (SIG) permettent aujourd'hui de produire, de consulter et d'analyser des cartes à partir de données géographiques. Chacun d'eux gère cinq étapes essentielles du cycle de vie de l'information géographique, qui sont appelées les « cinq A » par Denègre et Salgé [31] : son abstraction, son acquisition, son archivage, son analyse et son affichage. Ils facilitent chaque étape de la production de cartes, ainsi que leur traitement et leur superposition sous forme de couches. Leur émergence va de pair avec celle des systèmes de gestion de bases de données spatiales [61], qui sont capables d'ajouter aux données classiques une composante indiquant leur forme et leur position sur un territoire géographique.

L'informatisation et la démocratisation de la cartographie induite par les SIG a encore été renforcée par son arrivée sur Internet. Pour certains auteurs, cet événement correspond à l'émergence d'un nouveau paradigme : la « néogéographie ». Celui-ci serait marqué par « une grande interactivité et des contenus géolocalisés générés par les utilisateurs » [89]. D'après Schmidt et Gartner [107], il implique également le recours progressif à des technologies open source, qui ont conduit à une démocratisation de la cartographie et à sa dissémination sur le web. Le projet Openstreetmap (OSM) constitue un exemple significatif de cette mouvance. Entièrement open source, il permet à n'importe quel internaute de contribuer à une cartographie de l'ensemble des régions terrestres. Actuellement, OSM regroupe des milliers d'internautes actifs, qui y ajoutent 46 millions de nœuds chaque mois [64].

Ce succès considérable de la néogéographie entraine une inflation de la production et de la consultation de données spatiales. Par exemple, Franklin indiquait dès 1992 que 80% des données d'entreprise comportaient une composante de localisation [51]. Cette situation s'est encore renforcée avec l'arrivée

2. Comme il repose sur Hadoop, cet outil en constitue moins comme un concurrent de celui-ci qu'un complément intéressant, dont le potentiel est détaillée dans un chapitre ultérieur [1.1.1.2].

sur le marché de nouveaux capteurs comme les puces GPS des téléphones intelligents [8]. Le succès et le renouveau de la cartographie qui en résultent rend aussi l'organisation de celle-ci plus complexe. Il devient ainsi de plus en plus difficile de gérer ces données géolocalisées de façon classique [46], car elles deviennent progressivement des données massives tant leur volume global croît de façon incontrôlée. Les systèmes de gestion de données spatiales classiques ne sont en effet pas conçus pour répartir les traitement ou les données entre plusieurs machines, si bien que leur scalabilité horizontale est faible. Pourtant, il semble crucial de pouvoir analyser la localisation afin de maximiser la compréhension des phénomènes associés aux données et de faciliter la prise de décision qui en résulte. Dans un article de 2011 souvent cité [84], le cabinet McKinsey indique ainsi qu'une meilleure utilisation des données de la géolocalisation pourrait apporter un gain de 100 milliards de dollars aux fournisseurs de services et de l'ordre de 700 milliards aux utilisateurs finaux.

La possibilité de gérer la localisation des données apparaît donc comme un enjeu scientifique et sociétal important, même et surtout lorsqu'il s'agit de données massives. L'étude des solutions actuelles en ce domaine ainsi que des avenues possibles pour les améliorer sera donc au centre de la présente recherche, et la section suivante en effectue un premier parcours.

Gestion des données massives géolocalisées

Cette section compare des systèmes de gestion des données spatiales massives existants en fonction de plusieurs notions, qui demandent d'abord à être brièvement définies dans le contexte de données massives. En plus de reposer sur Hadoop comme évoqué précédemment, un système de gestion des données massives idéal devrait donc être aussi complet, efficient et rapide que possible. La **complétude** désigne ici la prise en compte par un système de tous les types de données spatiales élémentaires et des opérations de base sur celles-ci, avec aussi éventuellement la prise en compte d'extensions à celles-ci comme l'altitude. L'**efficience** désigne pour sa part la capacité du système a agir efficacement avec un minimum de moyens. Par exemple, celui-ci aura recours à une indexation des données afin de limiter la quantités d'opérations nécessaires sur celles-ci et donc d'accélérer leur traitement. Enfin, la **rapidité** de traitement résulte naturellement de l'efficience, mais ne doit pas s'entendre au premier degré. En effet, une accélération brut des traitements d'un système de gestion des données massives présentant un haut degré de scalabilité horizontale peut s'obtenir trivialement en augmentant le nombre de serveurs du cluster. Pour comparer équitablement la rapidité de deux systèmes de ce genre, il faut donc plutôt les tester à nombre de serveurs égal, de façon a estimer un débit de traitement pour un nombre de serveurs donné.

Quelques systèmes de gestion des données massives géolocalisées reposant sur Hadoop [3] sont apparus au cours des deux dernières années : Hadoop GIS, Geomesa, Pigeon [2, 67, 40]... Mais il ne s'agit

3. Parmi les solutions de gestion de données massives ne reposant pas sur Hadoop, certaines bases de donnes NoSQL ont dépassé le niveau de simples prototypes pour atteindre celui d'applications commerciales. Mais leurs capacités de traitement sont intrinsèquement plus limitées car contrairement a Hadoop qui est axé sur l'optimisation des traitements distribués, elles sont surtout conçues pour optimiser la répartition des données, comme détaillé dans un chapitre ultérieur [1.1].

généralement que de prototypes [8], même si certains comme Hadoop GIS peuvent s'avérer plutôt rapides [2]. Parmi ceux-ci, seulement cinq solutions proposent une gestion des données spatiales à partir de Spark [4]. La première, Magellan [104], permet d'analyser tous les types de données spatiales à l'aide de requêtes SQL mais ne dispose d'aucun système d'indexation spécifique pour celles-ci. Les deux suivantes, Spatial Spark [124] et GeoSpark [127], permettent des analyses plus rapides grâce à des méthodes d'indexation spatiale intégrées, mais sans interface permettant l'exécution de requêtes SQL sur ces données. Simba [121] est également doté d'un système d'indexation, mais sans permettre de gérer tous les types de géométries vectorielles existants, ce qui limite la portée de ses applications possibles. Enfin, l'extension Spark de Geomesa est limitée quant aux requêtes qu'elle permet d'exécuter ; et oblige à recourir à une base de données externe qui bride ses performances.

Outre ces limitations, Cortés et al. [20] indiquent en 2015 que les systèmes de gestion des données spatiales massives reposant sur Hadoop et Spark présentent un défaut récurrent : bien que répartie quant à ses traitements, l'indexation y est dirigée de façon centralisée, à partir d'un serveur maître qui répartit les tâches qu'elle implique entre les autres serveurs. Une telle hiérarchisation nuit en effet selon eux à la scalabilité horizontale de l'indexation, réduisant la rapidité d'exécution permise. Pour y remédier, ils définissent un mode de traitement moins centralisé mais qui renonce à utiliser l'environnement Hadoop pour le remplacer par une solution exotique.

Dans un tel contexte, il peut être intéressant de se demander si un système similaire à l'approche de Cortés et al. mais ne renonçant pas aux possibilités de l'écosystème Hadoop peut être défini. Il serait en effet dommage de renoncer à Hadoop sans motif sérieux, étant donné son statut de standard de fait spécialisé dans le traitement des donnes massives, l'absence de solutions alternatives matures et le grand nombre de composants (comme Spark) qu'il permet de combiner.

Problématique

En résumé, la possibilité de gérer les données massives apparaît comme un enjeu de plus en plus important à mesure que leur volume augmente. Mais alors que beaucoup de systèmes reposant sur Hadoop permettent une gestion généraliste de celles-ci, le traitement de la localisation qui leur est souvent associée n'en est qu'à ses débuts. Ainsi, les quelques prototypes permettant actuellement d'analyser des données massives comportant une référence spatiale manquent soit de rapidité, soit de complétude quant aux types spatiaux qu'ils gèrent, soit d'une integration a l'écosystème Hadoop malgré la richesse de celui-ci.

Autrement dit, **il n'existe aucun système de gestion des données spatiales massives complet et efficient reposant sur l'écosystème Hadoop.**

4. Plus récentes et reposant comme évoqué sur un environnement permettant des traitements plus rapides, ces solutions surclassent [124, 127, 121] celles basées uniquement sur Hadoop en terme de rapidité, si bien que ces dernières ne seront pas détaillées ici. Les qualités et limites d'une solution ne reposant que sur Hadoop (ISP) sont cependant détaillées dans un chapitre ultérieur [2.3.3].

Objectifs et questions de recherche

Objectif principal

L'objectif principal **(OP)** de cette recherche est de **concevoir un outil de gestion des données spatiales massives vectorielles efficient et compatible avec l'écosystème Hadoop**. Il s'agit en d'autres termes de proposer un système permettant de dépasser les limitations inhérentes aux systèmes de gestion des données massives actuels. Plus spécifiquement, la recherche vise à décrire et à valider une façon plus complète et rapide de traiter les composantes spatiales des données massives analysées. De plus, comme détaillé plus haut, le moteur devra être compatible avec Hadoop, car celui-ci constitue un standard de fait aujourd'hui difficilement contournable, performant, et aux extensions riches.

Objectifs spécifiques

L'objectif principal se décline en cinq sous-objectifs, qui approfondissent respectivement les notions de modélisation de la composante spatiale, d'indexation spatiale, de validation par prototypage et de connectivité avec les logiciels SIG.

A ces cinq objectifs répondent cinq questions de recherche, qui en précisent les modalités.

Un premier sous-objectif **(O1)** est de **créer un modèle unifié de la composante spatiale**, c'est-à-dire réunissant les aspects utiles des normes et des recommandations qui décrivent celle-ci. L'idée sous-jacente est d'adapter le système visé par l'objectif principal à tous les types de géométries vectorielles utilisées en géomatique, de façon à le rendre aussi complet que possible. Il devra donc permettre d'analyser une plus grande variété de phénomènes que les systèmes actuels. La norme ISO-19125 de 2004, qui décrit les géométries élémentaires ainsi qu'une façon de les interroger par des requêtes, semble constituer l'élément de base nécessaire à l'élaboration d'un tel modèle. Cependant, cette norme n'est pas sans lacunes, puisqu'elle ne permet pas par exemple d'associer une altitude aux objets spatiaux qu'elle décrit. Les apports de recommandations plus récentes décrivant les objets spatiaux et les fonctions d'analyse qui leur sont applicables, comme celles définies par l'OGC, doivent également être étudiées et intégrées lorsque cela s'avère pertinent. Une première question de recherche peut donc s'énoncer ainsi **(Q1)** : **comment unifier la norme ISO et les recommandations qui l'ont suivie de façon à permettre une gestion élargie de la composante spatiale ?**

Un second sous-objectif **(O2)** est de **concevoir un système à partir de ce modèle à même de gérer des données massives à référence spatiale et de les indexer efficacement**. Il s'agit donc de parvenir à ce que le système conçu permette d'analyser rapidement tous les types de données spatiales vectorielles pertinents dans un contexte géographique. Un tel objectif a été approché par plusieurs prototypes actuels [2, 67, 40, 124, 127, 121], mais avec systématiquement des lacunes quant à la vitesse ou à la généralité des analyses spatiales qu'ils permettent. La seule exception semble être le modèle proposé par Cortés et al. [20], qui prétend optimiser la façon dont les données spatiales massives sont indexées et donc la vitesse à laquelle elles peuvent être analysées ensuite. Mais c'est au prix d'un re-

noncement à Hadoop, ce qui éloigne leur solution du cadre de référence choisi pour ce mémoire. Une seconde question de recherche peut donc s'énoncer ainsi **(Q2)** : **comment optimiser l'indexation des données spatiales d'une façon compatible avec l'environnement Hadoop ?**

Un troisième sous-objectif **(O3)** est de **produire un prototype du système envisagé**. Essentiellement technique, cet objectif vise à implémenter la solution définie par les objectifs précédents, afin d'en démontrer la faisabilité et de faciliter sa mise en test. Pour cela, certaines bibliothèques de l'écosystème Hadoop pourront probablement être utilisées. La vitalité de cette communauté de développeurs, la gratuité de son code source et son statut de standard de fait en font en effet un choix naturel. D'où cette troisième question de recherche **(Q3)** : **les concepts et bibliothèques liés à l'écosystème Hadoop suffisent-ils à implémenter un prototype du système envisagé ?**

Un quatrième sous-objectif **(O4)** est de **soumettre le prototype à un banc de tests exhaustif**. Il s'agit en particulier de démontrer qu'il permet une meilleure scalabilité horizontale que les solutions d'indexation des données spatiales existants sur le marché. Le maintien de ses performances face à une montée en charge du volume de données seront donc comparées à celles d'un outil classique (PostGIS), mais aussi à celle d'un autre outil de gestion des données spatiales massives doté d'un système d'indexation (Spatial Spark). Une quatrième question de recherche peut donc s'énoncer ainsi **(Q4)** : **comment produire une solution de gestion des données spatiales plus performante que PostGIS et Spatial Spark au-delà de N clusters et d'un certain volume de données ?**

Un cinquième sous-objectif **(O5)**, de portée plus concrète, est de **rendre le système envisagé accessible depuis les logiciels SIG**. Plus spécifiquement, il s'agit de le doter de l'interopérabilité nécessaire pour qu'il soit consultable depuis des solutions classiques de visualisation des données. L'idée sous-jacente est donc de faciliter l'intégration de l'outil proposé aux dispositifs existants de visualisation des données géographiques. D'où cette cinquième question de recherche **(Q5)** : **comment rendre le système de gestion des données massives envisagé consultable depuis un logiciel SIG ?**

Méthodologie

La recherche s'est déroulée en trois étapes, comme illustré par la figure 0.1.

Une première étape consistait en une **revue de la littérature**, accompagnée d'une **veille technologique**. La revue commençait par un inventaire des principales représentations de la composante spatiale, afin de déterminer leurs possibilités d'unification. Elle s'est également intéressée aux avancées concernant la notion d'indexation spatiale, afin de les appliquer au contexte des données massives. La veille technologique consistait quant à elle à étudier l'évolution des solutions de gestion des données massives depuis l'écosystème Hadoop. Elle s'intéressait en particulier à l'élément Spark de cet écosystème et à sa composante Spark SQL permettant d'interroger les données à l'aide de requêtes, en raison de la rapidité et de la profondeur d'analyse qu'il semblait permettre. Cette étape devait initialement durer de janvier à avril 2016, mais s'est prolongée pendant une partie de l'été.

Une seconde étape consistait à **concevoir** le système d'indexation spatiale des données visé. Il s'agissait de définir l'architecture nécessaire à celui-ci ainsi que les processus le constituant, en réutilisant autant que possible des bibliothèques existantes lorsque celles-ci s'avéraient adaptées aux objectifs. Elle a été complétée par de nouvelles lectures et des compléments de veille technologique, afin de lever les incertitudes rencontrées. En raison de la souplesse et de l'efficacité qu'ils permettent même dans un contexte d'incertitude, certains principes de la **méthode agile** ont été appliqués : courtes étapes validées par l'implémentation de fonctionnalités, souplesse des méthodes en fonction des objectifs, amélioration continue des procédures et recherche de l'excellence technique. Cette étape a duré de mai à septembre 2016.

Une troisième étape consistait d'abord à implémenter un **prototype** du système conçu, avec de brefs retours à l'étape de conception afin de remédier aux blocages rencontrés. En fait, étant donné les développements précoces que la **méthode agile** implique, plusieurs composants de ce prototype existaient déjà en début d'étape afin de vérifier les hypothèses de conception. Ces éléments ont donc été réunis et adaptés en vue de respecter le sous-objectif **O3**. Le prototype a donc été conçu de façon à s'insérer dans l'écosystème Hadoop et à rencontrer les concepts associés à celui-ci. Sa réalisation a été suivie d'une période de **tests**, afin de **vérifier** qu'il validait tous les sous-objectifs de la recherche. Il s'agissait en particulier de vérifier ses performances en termes de rapidité d'exécution et de scalabilité horizontale, à l'aide d'un banc de test suffisamment exhaustif pour rencontrer le sous-objectif **O4**. Cette vérification a également entraîné de brefs retours aux étapes de conception et d'implémentation, afin de corriger certains bogues ou d'affiner l'outil. Elle s'est terminée en octobre 2016.

La rédaction du mémoire s'est faite sous la forme de trois articles, de façon à raccourcir sa durée d'écriture tout en valorisant la diffusion des résultats de la recherche. Chacun de ces articles est en cours de soumission.

Structure du document

Ce document suit le format d'un mémoire par articles. Il est constitué de trois articles en cours de soumission. Si leur contenu est identique à celui soumis en vue de leur publication, leur format et numérotations de chapitre ont été adaptés pour faciliter leur intégration dans ce mémoire.

Le chapitre 1 présente un état de l'art approfondi des différents concepts et systèmes reliés à la présente recherche. Il aborde et relie en particulier les notions de données massives, de composante spatiale et d'indexation spatiale.

Le chapitre 2 reprend l'article « Elcano : un système de gestion des données massives à composante spatiale basé sur Spark SQL ». Il explore les limites des systèmes d'analyse spatiale vectorielle des données massives actuels quant à leur capacité à représenter les différents types de géométries spatiales et à en permettre l'analyse. Il résout ensuite l'essentiel de ces limitations, en proposant un système de gestion des données spatiales massives entièrement compatible avec la norme ISO-19125-2 :

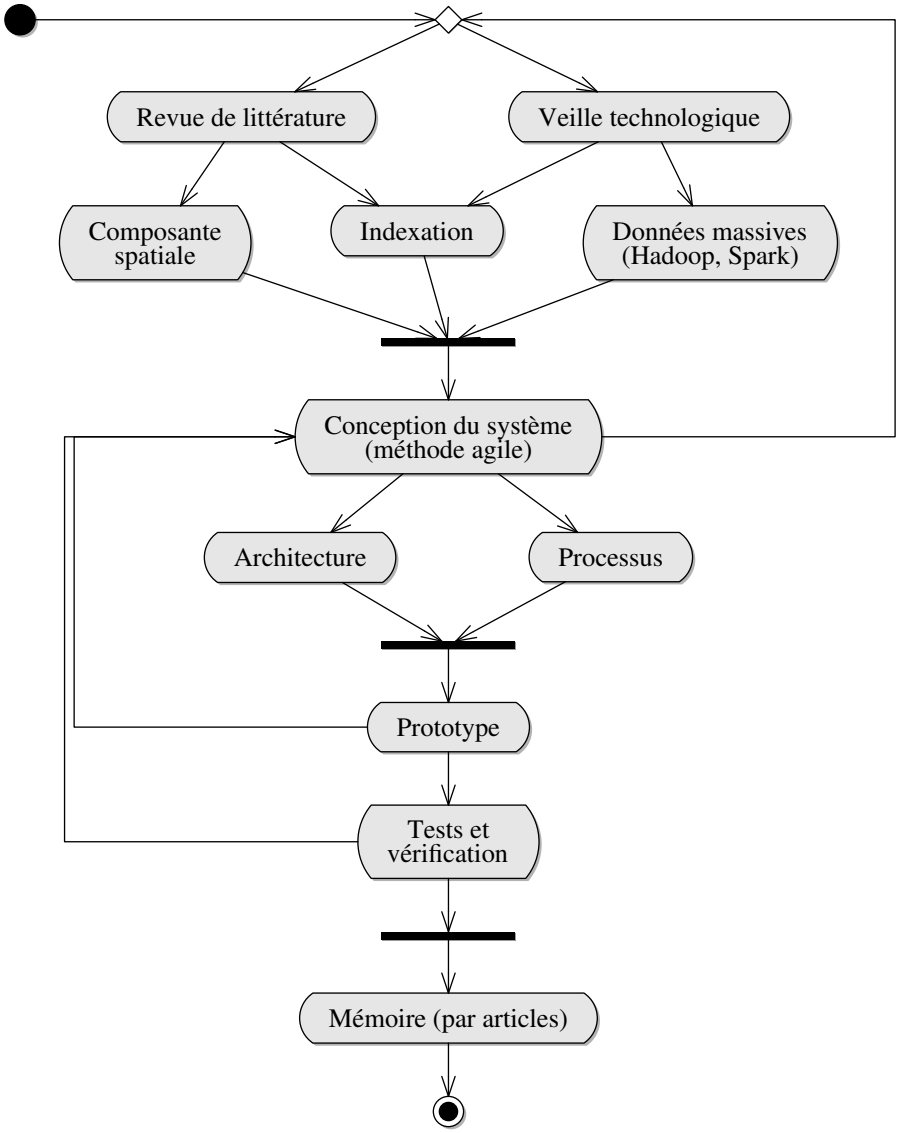

FIGURE 0.1 – Étapes principales de la recherche

Elcano.

Le chapitre 3 contient l'article « Vers une indexation des données spatiales massives performante et efficace ». Il complète la visée de l'article précédent, par une revue de littérature sur l'indexation spatiale puis par une extension du système Elcano à celle-ci. Pour terminer, un banc de tests estime la scalabilité horizontale et les performances du système ainsi conçu.

Le chapitre 4 présente l'article « Vers un modèle unifié de la composante spatiale vectorielle et son application à Elcano ». Celui-ci adopte une perspective plus large, en passant en revue les principales évolutions et extensions de la norme ISO-19125. Il propose ensuite un nouveau modèle pour le système Elcano, qui intègre quelques-unes de ces améliorations.

Finalement, la conclusion synthétise l'apport dû à ces différentes publications et évoque les perspectives de recherche qu'elles ouvrent.

Chapitre 1

État de l’art

La présente recherche vise à concevoir un système d'analyse des données spatiales complet, efficace et adapté aux données massives. Afin d'y aider, cet état de de l'art explore successivement les concepts de données massives, de composante spatiale et d'indexation spatiale.

1.1 Données massives

Qu'est-ce qui distingue fondamentalement les données massives des données classiques ? Il pourrait être tentant de les caractériser comme dépassant un certain volume, mais dans la pratique cette frontière ne cesse d'être repoussée par les avancées technologiques et ne saurait constituer un référentiel pérenne. Ainsi d'après IDC [1], la quantité globale de données en circulation croît de 90% tous les deux ans et ce rythme se maintiendra jusqu'en 2020 [52]. Il semble donc bien qu'une définition des données massives nécessite de recourir à un autre critère que simplement le volume qu'elles impliquent. C'est l'incapacité de traiter ces données à l'aide de solutions ordinaires qui sera retenue ici. Selon ce critère, il s'agit de données impossibles à stocker et à manipuler à l'aide d'un système de gestion des données classique.

Cette définition pose immédiatement la question des moyens à mettre en œuvre afin de traiter les données massives efficacement.

Une première approche consiste à proposer des bases de données qui renoncent à certaines des contraintes qu'impliquent les bases de données classiques, pour offrir en contrepartie de meilleures performances ou une meilleure résistance aux pannes. Elle s'illustre par l'apparition de bases d'un genre nouveau, souvent qualifiées de NoSQL [111]. Plus flexibles que les bases de données classiques, celles-ci reposent souvent sur une distribution des données (et parfois de leurs traitements) entre plusieurs machines. Une seconde approche, complémentaire, prend comme point de départ l'évolution des capacités de traitements distribués des processeurs et des logiciels, d'une façon cette fois agnostique par rapport à la façon dont les données à traiter sont structurées. C'est surtout cette seconde approche qui sera détaillée ici, car elle ne s'impose pas d'emblée un format de données spécifique, visant plutôt à optimiser la façon dont les traitements sont distribués. Autrement dit, elle vise à offrir un environnement de traitement des donnés massives plutôt qu'une simple base de données adaptée à celles-ci, ce qui la rend plus généraliste et extensible. Les environnements Hadoop et Spark ainsi que les concepts qu'ils impliquent seront donc décrits de façon précise. Mais les concepts liés aux bases de données NoSQL seront également évoqués et illustrés en fin de section.

1.1.1 Environnements de gestion de données massives

Selon You [123], le traitement efficace des données massives a été rendu possible grâce à deux principales innovations. D'une part, alors que les traitements parallèles performants nécessitaient initialement de disposer de processeurs haute capacité (HPC), ces traitements ont été rendus accessibles

1. Acronyme de « International Data Corporation », qui désigne une société américaine spécialisée dans les études de marché technologiques.

au grand public par l'arrivée sur le marché des processeurs multi-cœurs et par la simplicité d'accès aux ressources d'un cluster que permet le Cloud Computing. D'autre part, des méthodes logicielles généralisant et simplifiant les traitements parallèles sont apparues, dont le plus connu est sans doute l'algorithme Map Reduce [28]. Deux environnements tirant tous les bénéfices de ces évolutions, Hadoop et Spark, sont présentés ci-dessous.

1.1.1.1 Hadoop

L'écosystème Hadoop est aujourd'hui un standard de fait en matière de gestion des données massives. Il est né de la rencontre entre un algorithme mis au point par Google et un projet de Cutting rattaché à la Fondation Apache en 2001. Ce dernier, Lucène, était un outil destiné à permettre des recherches rapides dans un corpus textuel. Cutting l'a employé en 2003 comme base d'un projet de moteur de recherche web généraliste, Nutch, qui a également rejoint la Fondation Apache en 2004. En parallèle, Google a rendu public en 2004 l'algorithme Map Reduce [28]. Celui-ci permet d'accélérer les traitements de données volumineux, en les distribuant sur une grappe de serveurs afin de profiter de leur parallélisme. Une intégration de Map Reduce sur une branche du projet Nutch a été tentée à partir de 2005, afin d'y faciliter la gestion des données massives. A partir de 2009, cette branche est devenue un projet central de la Fondation Apache : Hadoop [120]. Celui-ci a depuis été adopté par des géants de l'informatique comme Oracle et IBM, tout en participant au succès de plusieurs startups [48]. Il accueille une série d'applications open source plus spécialisées, réparties en couches, qui constituent son écosystème. La figure 1.1 en donne une vue synthétique, forcément partielle car il s'agit d'un projet qui évolue sans cesse et implique un grand nombre de développeurs.

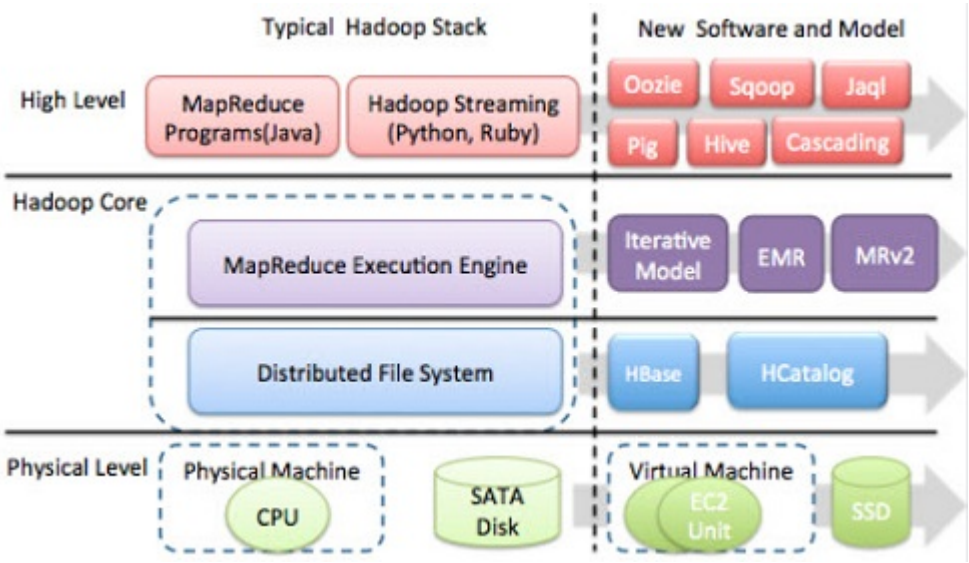

FIGURE 1.1 – L'écosystème Hadoop [33]

Le succès d'Hadoop peut s'expliquer par le fait qu'il permet de traiter rapidement une grande quantité de données. Son architecture repose en effet sur une « grappe » de serveurs (ou cluster) qui se répar-

tissent les tâches de stockage et de traitement. Les données sont donc disséminées sur l'ensemble de la grappe, dans des fichiers distribués (HDFS) auxquels sont appliqués des traitements en parallèle. Cette répartition se fait en fonction des deux étapes de l'algorithme Map Reduce, qui divise entre plusieurs machines les traitements à réaliser (étape « map »), puis collecte et combine les résultats intermédiaires ainsi produits (étape « reduce »).

Une telle découpe permet une bonne **scalabilité horizontale** des systèmes reposant sur Hadoop, ce qui signifie que l'augmentation linéaire des performances peut être obtenue simplement en ajoutant des serveurs au cluster, sans augmenter la puissance des processeurs utilisés [2]. L'ajout de machines suffit en effet à renforcer les capacités de traitement en parallèle, et donc la puissance de calcul [120].

Afin de faciliter l'échange et le stockage de fichiers de façon distribuée, Hadoop intègre le système « Hadoop Distributed File System » (HDFS). Ce système de fichiers permet l'accès cohérent en lecture et en écriture aux documents et paquetages qu'il contient depuis tous les nœuds du cluster. Il garantit donc que des traitements en parallèle pourront être appliqués sans conflit d'accès, ce qui joue un rôle clef dans la scalabilité horizontale d'Hadoop.

1.1.1.2 Spark

Spark a été développé à partir de 2014 par l'AMPLab de l'Université de Berkeley, puis repris par la fondation Apache. Il peut fonctionner en tant qu'élément de l'environnement d'Hadoop [3] ou de façon autonome. Il propose une alternative puissante à Map Reduce, puisqu'il remplace les nombreux accès disque impliqués par cette méthode par la diffusion de structures de données spécifiques sur la mémoire vive du cluster. Celles-ci sont appelées « Resilient Distributed Datasets » (RDD), et consistent en des blocs combinant des données et du code exécutable à même de les transformer. Ce choix architectural rend Spark jusqu'à dix fois plus rapide qu'Hadoop [128], même si ce peut être au prix d'une charge mémoire plus importante [59].

Le très récent système de gestion de fichiers distribués Alluxio [4] stocke ses fichiers en mémoire vive et est directement compatible avec Spark, ce qui semble à même d'améliorer encore les performances de ce dernier. Mais une telle généralisation de l'emploi de la mémoire vive pose la question de la préservation des données en cas de panne électrique sur l'ensemble du cluster.

Le noyau de Spark a été initialement conçu en Scala [95], un langage qui gère à la fois les paradigmes fonctionnels et orientés objet tout en restant interopérable avec toutes les bibliothèques Java [3]. Or, ce langage permet généralement des codages plus concis et fiables grâce à ses aspects fonctionnels et son typage implicite. Il gère en outre l'héritage multiple, l'initialisation paresseuse et la surcharge d'opérateurs, et intègre un moteur d'analyse syntaxique. Ces éléments font de Scala un langage syn-

2. Par opposition, la scalabilité verticale désigne une augmentation des performances obtenue par amélioration matérielle d'une seule machine, ce qui peut s'avérer plus coûteux.

3. Il se situe alors dans la couche de haut niveau de la figure 1.1, mais remplace l'algorithme MapReduce du noyau d'Hadoop par ses propres fonctionnalités.

4. `http://www.alluxio.org`

taxiquement plus puissant et concis que Java, tout en lui permettant d'être entièrement compatible avec ce dernier.

Comme synthétisé par la figure 1.2, Spark gère aussi les langages Python, R et Java, et comporte quatre extensions de base : Spark SQL, Spark Streaming, MLLib et GraphX. La première extension permet de questionner et de transformer les données à l'aide de requêtes SQL. La seconde permet de traiter des données reçues en continu, tandis que les deux dernières gèrent respectivement le Machine Learning et la représentation visuelle de données. Plus précisément, MLLib est une extension dédiée à l'intelligence artificielle qui permet depuis Spark de lancer des apprentissages à partir de larges jeux de données en exploitant pleinement le parallélisme de celui-ci [88], puis d'appliquer ces apprentissages à d'autres jeux de données. Il peut par exemple servir à reconnaître un motif parmi des photographies. GraphX a une visée plus analytique, en rendant visible les données et leurs interconnections sous la forme de graphes. La figure 1.3 fournit ainsi un exemple minimaliste de relation réciproque entre deux données symbolisant l'amitié de deux personnes. Malgré l'intérêt de toutes ces extensions, c'est principalement Spark SQL qui sera abordé dans le cadre de cette recherche, en raison des possibilités d'adaptations aux normes ISO et OGC qu'il offre.

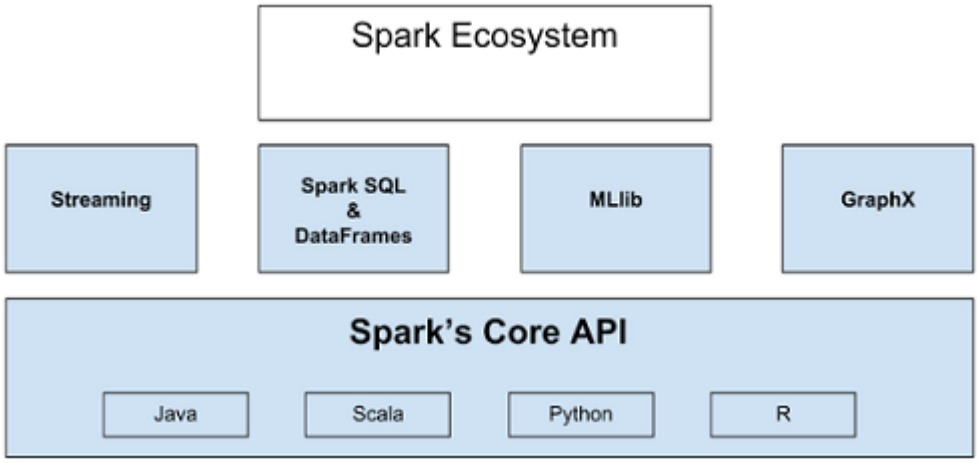

FIGURE 1.2 – Principales composantes de Spark SQL [80]

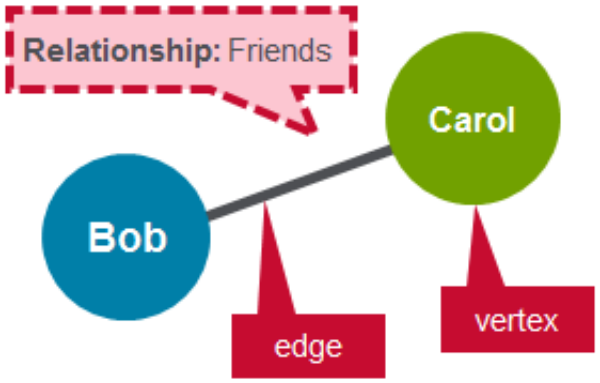

FIGURE 1.3 – Représentation d'une relation réciproque sous GraphX [87].

La composante Spark SQL [6] permet d'enrober les RDD de Spark avec des « DataFrames », ce qui renforce leurs possibilités d'interaction et leurs performances. Les DataFrames peuvent en effet être interrogés de la même façon que les tables d'une base de données relationnelle classique, c'est-à-dire à l'aide de requêtes respectant le format SQL. Cette composante permet donc de combiner une partie des avantages d'un système de gestion des données classique avec une architecture dédiée aux données massives. Spark SQL aide aussi à augmenter les performances, en sérialisant les données puis en optimisant stratégiquement les requêtes de transformation sur celles-ci avant leur exécution. Il autorise aussi la définition de types de données personnalisés et l'ajout de fonctions SQL. De plus, son optimiseur Catalyst est ouvert à l'ajout de points d'extension et de règles de traitement spécifiques.

Spark SQL n'est pas le seul élément de l'écosystème Hadoop à reposer essentiellement sur la mémoire vive et à permettre l'utilisation de requêtes SQL. C'est aussi le cas d'Impala, qui est sorti un an avant Spark. Mais Spark présente quelques avantages décisifs par rapport à cet outil :

— **Autonomie** : Spark peut aussi être utilisé indépendamment d'Hadoop, alors qu'Impala en reste tributaire ;

— **Polyvalence** : Spark permet de traiter aussi bien les données reçues sous forme de flux (via Spark Streaming) que celles reçues « d'un bloc » (en batch), tandis qu'Impala se limite aux données reçues en batch.

— **Généralité** : Alors qu'Impala est spécifiquement dédié au lancement rapide de requêtes SQL, Spark peut aussi être interrogé ou étendu en Scala et peut être directement appliqué à beaucoup d'autres contextes comme le machine learning.

— **Performance** : Spark serait jusqu'à sept fois plus rapide qu'Impala pour le traitement de données spatiales [124].

Format Parquet Autant Spark qu'Impala permettent de charger et de sauver des tables de données (typiquement dans HDFS) au format Parquet [119]. Celui-ci a été défini originellement par Twitter, et est actuellement géré par la fondation Apache. Il applique une compression aux données tabulaires,

ce qui les rend beaucoup plus compactes que des fichiers HDFS ordinaires à quantité d'informations égale. Par conséquent, leur lecture représente une plus faible charge mémoire pour le cluster et s'accompagne d'un gain en termes de rapidité. De plus, le format gère les types et les noms des colonnes de la table persistée, ce qui permet sous Spark sa récupération directe sous forme de DataFrame. Par contre, l'écriture vers un fichier Parquet peut s'avérer coûteuse, en raison du temps de compression qu'elle implique.

Malgré la richesse et la maturité de systèmes comme Hadoop et Spark, ceux-ci sont incapables de gérer nativement les données à composante spatiale [cf. 1.2], et ne sont étendus par aucun système gérant ce type de données de façon entièrement satisfaisante.

1.1.2 Systèmes de gestion de bases de données NoSQL

Si nous avons privilégié ici les solutions Hadoop et Spark en raison de leur indépendance vis à vis de la structure de données, il existe d'autres systèmes à même de gérer les données massives. Parmi ceux-ci, il nous a semblé intéressant de détailler les systèmes de gestion de bases de données NoSQL, en raison de leur complémentarité par rapport aux solutions évoquées. En outre, ces systèmes de stockage sont souvent reliables à Hadoop et Spark, et peuvent par conséquent constituer des alternatives intéressantes au système de fichiers HDFS.

L'idée sous-jacente de cette autre approche est de renoncer à certaines des contraintes qu'impliquent le mode transactionnel des bases de données classiques. Ces contraintes sont souvent résumées par l'acronyme ACID [63] : atomicité des opérations, consistance des informations, isolation des transactions jusqu'à leur réalisation, durabilité des résultats. L'assouplissement des contraintes ACID a conduit à la conception de nouveaux systèmes de gestion des données, qui suivent des approches variées mais sont généralement regroupées sous l'appellation « NoSQL » [111] (dans le sens de « not only SQL ») par la littérature. Ceux-ci se veulent plus rapides et/ou plus tolérants aux pannes que les systèmes classiques, même si c'est au prix de données moins structurées.

Il existe essentiellement quatre types de bases de données NoSQL, suivant le format interne qu'elles emploient pour stocker les données : par clés, par colonnes, par documents et en réseau (ou graphe) [98]. Le format par clé est le plus simple mais aussi le moins structuré : il consiste juste à stocker les données par identifiant unique. Le format par colonne est un peu plus structuré, puisqu'il permet de regrouper les données en colonnes et en familles de colonnes. Le format par document est encore plus structuré : il stocke les données dans des arbres dont chaque branche peut être interrogée. Le format en réseau est le plus complexe, puisqu'il situe les données dans un graphe ou un multi-graphe plutôt que dans un arbre à racine unique. Adapté pour modéliser des distances, ce format semble le plus adapté à un contexte géographique. Mais il induit aussi une certaine fragmentation, qui rend difficile de représenter ensuite les données dans un format plus hiérarchique.

Les systèmes NoSQL sont au cœur de la réactivité des moteurs de recherche actuels, en permettant de répondre immédiatement à des requêtes portant sur d'énormes volumes de données. A titre d'illustra-

tion, trois systèmes NoSQL sont brièvement décrits ici [5].

HBase [54] est un système de gestion de base de données NoSQL par colonne. Il s'agit d'une extension d'Hadoop développée par la Fondation Apache, qui se présente comme une surcouche optionnelle d'HDFS. Celle-ci permet le stockage structuré de tables de très grande dimension, dans un format non-relationnel. Inspirée du système BigTable conçu par Google, elle est utilisée par la messagerie de Facebook depuis 2010 [1]. Plus récemment, un connecteur entre HBASE et Spark a été rendu disponible par HortonWorks [129].

MongoDB [17] est un système open source de gestion de base de données NoSQL par document mis sur le marché par MongoDB Inc. en 2009. Il emploie un format de stockage et d'échange spécifique appelé BSON (contraction de binary et de JSON), ce qui permet d'interroger celles-ci à l'aide de requêtes JSON [6]. Depuis sa version 2.4, il gère le format GeoJSON [cf. 1.2.2], qui intègre les sept types de géométries spatiales définis par la norme ISO-19125. Il permet nativement d'effectuer sur celles-ci des recherches spatiales (inclusion, intersection, proximité), en s'aidant aux besoins d'index spatiaux 2D reposant sur GeoHash [cf. 1.3.1]. Ses concepteurs l'ont doté d'extensions permettant sa connexion avec Hadoop et avec Spark [68], ce qui illustre sa complémentarité avec ceux-ci. Même s'il permet une répartition distribuée des données, MongoDB est en effet moins performant qu'Hadoop pour effectuer rapidement des traitements massifs sur celles-ci, car il n'a pas été conçu initialement dans ce but [27].

ElasticSearch [56] est un autre système open source de gestion de base de données NoSQL par document. Il a été conçu puis distribué à partir de 2010 par Shay Bannon, qui a créé une entreprise du même nom en 2010. Celle-ci avait quatre ans plus tard une valeur estimée de 700 millions de dollars, ce qui témoigne du succès rapide de l'outil [66]. Celui-ci intègre le système d'indexation inversée Lucène (qui fut également un des piliers d'Hadoop), et repose sur une répartition distribuée des données. Interrogeable via le format JSON depuis un service web, ElasticSearch peut aussi être connecté à Hadoop et à Spark. Comme MongoDB, il gère les sept types de géométries spatiales de GeoJSON [cf. 1.2.2], mais en y ajoutant l'enveloppe et le cercle [66]. Nativement, un système d'indexation des données spatiales selon les méthodes GéoHash et QuadTree est fourni [74].

Ces exemples illustrent la diversité des bases de données NoSQL, et leur richesse y compris en ce qui concerne le traitement et l'indexation de données spatiales. Néanmoins, celles-ci imposent aux données des formats d'accès et de stockage spécifiques : par colonne pour HBASE et par document JSON pour MongoDB et ElasticSearch. De plus, elles sont davantage conçues pour répartir des données massives de façon distribuée que pour optimiser les traitements distribués effectués sur celles-ci. Hadoop et Spark ont par contre été spécifiquement conçus pour optimiser ce dernier aspect, sans imposer de format particulier aux données qu'ils permettent de traiter. Ils semblent donc plus à même

5. Un inventaire plus précis des systèmes de gestion de données NoSQL peut être trouvé dans la revue de littérature établie par Cattel [15].

6. Acronyme de « JavaScript Object Notation », qui désigne un format léger d'échange de données [12].

de servir de base à un système de traitement des données spatiales massives généraliste et optimal, ce qui explique qu'ils aient été retenues comme support dans le cadre de la présente recherche.

1.2 Composante spatiale

Après le parcours des deux grandes familles de solutions disponibles pour le traitement des données massives, qui a permis de choisir celle qui semblait la plus adaptée pour servir de support au système envisagé, la présente section détaille un autre concept essentiel à sa conception : celle de composante spatiale.

Une des premières tentatives de formalisation de la composante spatiale se retrouve chez Bertin, qui classifie les informations spatiales représentées sur une carte en points, en lignes et en surfaces [10]. D'après Palsky, cette recherche fondatrice s'intéresse surtout au signifiant des cartes et à leur langage visuel, dans une visée « structuraliste » [97]. Il est vrai que l'ouvrage de Bertin décrit aussi par exemple six « variables rétiniennes » (ou « variables visuelles »), qu'il importe d'adapter aux informations affichées afin de leur donner sens : la forme, l'orientation, la couleur, le grain, la valeur et la taille.

Depuis, la composante spatiale a connu une série de normalisations et de spécifications qui l'éloignent d'une définition purement graphique et la rendent plus opératoire. Ce point est détaillé ci-dessous.

1.2.1 Normes et spécifications décrivant la composante spatiale

Les principaux organismes qui normalisent et spécifient la composante spatiale sont brièvement décrits au début de cette section. Quelques normes essentielles émises par l'ISO et quelques recommandations de l'OGC portant sur la composante spatiale sont ensuite parcourues, ainsi que des extensions et adaptations résultant de leur mise en pratique. L'emphase est mise sur la modélisation des données spatiales vectorielles, car leur sémantique plus forte les rend plus adaptéees à l'analyse de phénomènes spatiaux. Mais les données matricielles sont également abordées, en raison de la généricité de leur format et du très grand nombre d'applications qu'elles permettent.

1.2.1.1 Organismes de normalisation

Afin d'améliorer la gestion des données à référence spatiale, un consortium des principaux acteurs du domaine s'est formé à partir de 1994 : l'Open Geospatial Consortium (OGC). Cette organisation privée, qui comportait 258 membres actifs en 2006, a ainsi progressivement établi des recommandations techniques conceptualisant les objets spatiaux, leur traitement, et l'interopérabilité des processus qui s'y appliquent. Les recommandations publiées par l'OGC dans le domaine géospatial inspirent souvent fortement les standards établis ensuite par l'ISO.

De caractère plus officiel, l'Organisation Internationale de Normalisation (ISO) est une organisation non gouvernementale fondée en 1947 et présente dans plus de 150 pays, qui émet des documents jouant un rôle de standard *de jure* quant aux formalismes à adopter dans un grand nombre de contextes.

Elle rédige généralement ces documents en coopération avec des acteurs clés des domaines concernés, ce qui peut parfois entraîner une certaine partialité [53] mais assure également une bonne adéquation avec la réalité du terrain. Par exemple, le comité TC 211 de l'ISO constitué en 1994 émet des standards concernant l'information géographique et la façon de la gérer.

1.2.1.2 Norme ISO-19125

L'ISO a publié en 2004 la norme ISO-19125, qui décrit les géométries élémentaires pouvant servir à représenter et localiser les objets spatiaux 2D liés à une zone géographique ainsi que les traitements SQL minimaux que doit offrir un système gérant ces données. Elle est ventilée selon deux volets. Le premier (ISO-19125-1) décrit les types d'objets spatiaux élémentaires, ainsi que les méthodes qui peuvent s'y appliquer. Plus spécifique, le second (ISO-19125-2) décrit les conventions que la norme devrait respecter en cas d'implémentation dans un système de gestion de bases de données relationnelles utilisant le langage SQL.

Le volet ISO-19125-1 de la norme définit sept sortes de géométries 2D constructibles par interpolation linéaire sur un plan en deux dimensions. Celles-ci se répartissent en trois types simples proches de ceux définis par Bertin [10] (le point, la polyligne et le polygone) et en quatre types composés construits par le rassemblement de géométries simples (le multi-point, la multi-polyligne, le multi-polygone et la collection de géométries). Le tableau 1.1 décrit ces différents types de géométries, tout en fournissant des exemples d'objets spatiaux qu'elles pourraient servir à représenter. A ces géométries peuvent s'appliquer différentes fonctions spatiales, qui se déclinent en méthodes, en relations et en opérations ainsi qu'en une fonction métrique. Les méthodes fournissent une information à propos de la géométrie à laquelle elles sont appliquées, tandis que les relations renvoient un booléen qui indique si deux géométries sont liées par une des relations spatiales issues du modèle DE 9IM défini par Clementini et Laurini [19] à partir des travaux de Egenhofer et al. [39, 37]. Les opérations spatiales permettent quant à elles de créer une nouvelle géométrie à partir de deux géométries reçues en paramètres, en leur appliquant une opération ensembliste. Enfin, la fonction métrique disponible permet de connaitre la distance en mètres entre deux géométries.

Nom	Description	Exemple	Type
Point	Vecteur représentant une position géographique ponctuelle.	Un arbre, une maison	Simple
Polyligne	Suite de points reliés deux à deux, représentant une structure géographique linéaire.	Une route, une rivière	Simple
Polygone	Multi-ligne(s) fermée(s) représentant une structure géographique polygonale comportant éventuellement des trous.	Une région, un lac.	Simple
Multi-point	Groupe de points.	Les arbres d'une forêt.	Composé
Multi-polyligne	Groupe de polylignes.	Les voies d'un réseau routier.	Composé
Multi-polyligne	Groupe de polygones.	Les zones commerciales d'une ville.	Composé
Collection de géométries	Groupe de géométries simples de plusieurs types.	Les éléments d'un aéroport.	Composé

TABLE 1.1 – Géométries vectorielles définies par la norme ISO-19125-1

Le volet ISO-19125-2 de la norme traite pour sa part de l'implémentation SQL de ces différentes fonctions, et fournit des règles de nommage pour celles-ci. Par exemple, la fonction ST_Intersects vérifie si deux géométries ont une relation d'intersection tandis que ST_X retourne la première coordonnée d'une géométrie de type point et que la fonction ST_Union réunit deux géométries en une. Cette norme ne se veut par contre pas prescriptive quant aux technologies à employer, si bien qu'il est possible de l'appliquer aussi bien à un système de gestion des données classique qu'à un système de gestion des données massives du moment que la référence spatiale y est gérée. Ce volet de la norme décrit également les formats WKT [7] et WKB [8], qui permettent respectivement de représenter les géométries 2D sous forme textuelle (WKT) et de les sérialiser dans un format binaire (WKB).

1.2.1.3 Norme ISO-19107

Cette norme a été publiée par l'ISO en 2003, puis en 2005 dans sa version définitive. Elle décrit les différentes géométries vectorielles pouvant évoluer dans un espace en trois dimensions ainsi que les relations susceptibles de les lier. Décrivant aussi bien des géométries constructibles par interpolation linéaire que des courbes, elle s'intéresse aussi aux TIN [9] et aux transformations du système de coordonnées, ce qui la rend bien plus générale qu'ISO-19125. Cette norme joue un rôle fondateur, puisque

7. WKT est un acronyme de « well-know text ».
8. WKB est un acronyme de « well know-binary ».
9. TIN est un acronyme de « Triangulated Irregular Network « . La section suivante en fournit une description détaillée.

les autres standards gérant la composante spatiale vectorielle se bornent à décrire une sélection des objets qu'elle propose. ISO-19125 constitue la partie 2D de cette norme, compatible avec celle-ci.

1.2.1.4 Norme ISO-13249-3

La norme ISO-13249-3 a donné lieu à une publication de l'ISO en 2003, 2006, 2011 et 2016. Plus générale qu'ISO-19125 mais plus concrète et spécifique qu'ISO-19107, elle décrit des géométries tridimensionnelles ou constructibles à l'aide de courbes. Elle permet en particulier d'ajouter un troisième paramètre marquant les coordonnées spatiales d'un point, de passer d'un datum géographique à un autre, et d'utiliser des géométries tridimensionnelles. Cette norme étend aussi le format WKT initialement décrit par la norme ISO-19125, pour lui permettre de représenter des géométries en 2.5D, en 3D, ou constituées de courbes.

1.2.1.5 Recommandations Simple Features et Simple Features SQL

Les recommandations **Simple Features** et **Simple Features SQL** établies par l'OGC correspondent en grande partie aux normes ISO-19125-1 et ISO-19125-2 [69]. Leur volet SQL a cependant connu davantage de versions : 1999, 2005, 2006 et 2010. A partir de 2006, les fonctions décrites ont une signature identique à celles de la norme ISO-19125-2, mais avec une portée un peu plus générale. En effet, la recommandation Simple Features SQL s'inspire de la norme ISO-13249-3 pour permettre l'ajout d'une troisième coordonnée permettant d'indiquer l'élévation (2.5D). Elle évoque également l'ajout de géométries comme les polyèdres [10] et les TIN.

1.2.1.6 Norme ISO-19123

La norme ISO-19123 définie par l'ISO en 2005 décrit des géométries spatiales telles qu'elles couvrent entièrement la surface considérée. Il ne s'agit donc plus seulement de polygones, de points ou de multilignes, mais bien d'un maillage qui recouvre l'ensemble d'une zone géographique. Dans le cas d'un recouvrement régulier, le même motif est reproduit sur l'ensemble de la surface, tandis qu'il varie dans le cas d'un maillage irrégulier. Cette section décrit l'implémentation le plus courante pour chacune de ces possibilités.

Grille La grille appartient à la catégorie des recouvrements réguliers. Elle consiste à diviser la zone couverte en carrés de même dimension appelés pixels, qui partagent entre eux leurs arrêtes. Dès lors, il s'agit d'une représentation matricielle du territoire et non d'une représentation vectorielle, car les géométries utilisées sont une découpe du territoire tout à fait indépendante des objets qui s'y trouvent. Ce mode de découpe simplifie le traitement des données géographiques, et connaît des applications dans des domaines aussi variés que la sismographie [7] et le traitement d'images satellite [34]. Mais il ne permet pas de localiser ou d'interroger directement les objets représentés, ce qui le rend moins adapté à des fins d'analyse spatiale que le format vectoriel.

10. Il s'agit de géométries en trois dimensions dont chaque face est un polygone.

TIN La méthode du réseau de triangles irréguliers (TIN) consiste à couvrir une surface tridimensionnelle à l'aide de triangles de tailles différentes partageant leurs arrêtes. Ces triangles sont souvent construits par triangulation de Delaunay [29], c'est-à-dire en s'interdisant de situer un sommet à l'intérieur du cercle inscrit d'un autre triangle. Il s'agit en effet d'une méthode qui évite de donner aux triangles des formes trop allongées qui nuiraient à la précision des représentations. Contrairement aux grilles, les TIN sont des structures vectorielles, car la position et la hauteur des sommets des triangles varient en fonction de données du terrain comme l'élévation. Ces points peuvent par exemple servir à représenter les sommets et les puits d'une zone montagneuse. Ils jouent donc un rôle clé dans l'élaboration de modèles numériques de terrain à même de représenter précisément les ruptures de pentes [109].

1.2.2 Extensions et adaptations existantes

S'il est exceptionnel de les voir gérées par un système de gestion des données massives, les normes évoquées précédemment ont connu un grand nombre d'implémentations en ce qui concerne la gestion de données ordinaires. La présente section présente quelques-unes de ces implémentations, ainsi que des extensions ou adaptations qu'elles proposent par rapport au standard initial.

JTS [11] est une solution open source développée à partir de la fin des années 1990 pour les besoins du gouvernement de la Colombie Britannique, repris par Vivid Solutions puis devenue plus récemment un projet Eclipse. Ses classes fournissent des fonctionnalités rencontrant la norme ISO-19125, mais sont également capables de gérer l'élévation, les TIN, l'indexation et le changement de système de coordonnées. Sa gestion du WKT est cependant plus proche de la version ISO-19125-2 que de la version ISO-13249-3, ce qui peut nuire à la représentation de certaines géométries. Par contre, le format WKB employé est plus proche des formats définis par ISO-13249-3 et par PostGIS.

PostGIS [94] s'appuie sur un portage de JTS en langage C++ appelé GEOS [12] (Geometry Engine Open Source). Il intègre ses méthodes au contexte d'une base de données relationnelle classique, en rendant disponibles dans PostgreSQL chacune des fonctions spatiales définies par la norme ISO-19125, ainsi que la prise en compte d'une troisième dimension indiquant l'élévation (2.5D). Il permet également de modifier la projection de ces données. A partir de sa version 2, il gère également les surfaces polyédriques et les TIN [100] décrites par les normes ISO-13249-3 et ISO-19123. PostGIS est également fourni avec un grand nombre de fonctions d'analyse spatiale vectorielle et matricielle, comme l'extraction du polygone de couverture correspondant à un groupe de points, la recherche des N géométries les plus proches d'une géométrie et le calcul de profil d'élévation [101]. Il permet également de gérer les formats EWKT et EWKB, qui constituent des extensions artisanales des formats WKB et WKT d'ISO-19125. Le format EWKB permet de stocker le système de projection employé

11. https://github.com/locationtech/jts
12. https://trac.osgeo.org/geos/

par une géométrie. Le format EWKT tend quant à lui à converger avec la dernière version de WKT telle que définie par la norme ISO-13249-3.

GeoJSON [13] est une extension du langage JSON [12] aux données géographiques, qui permet de représenter celles-ci d'une façon concise et facilement interopérable. La recommandation RFC 7946 [14] décrit précisément ce format, qui permet de déclarer les sept types de géométries vectorielles décrits par la norme ISO-19125, ainsi que deux structures propres appelées « Feature » et « FeatureCollection ». La première réunit une géométrie et des données attributaires dans un même objet JSON, tandis que la seconde rassemble une liste de « Features ». GeoJSON offre de plus la possibilité d'attribuer une élévation aux données spatiales, ce qui permet de représenter des TIN même si ce type n'y est pas directement défini [102].

TopoJSON [11] est une extension de GeoJSON qui représente les données sous une forme purement topologique. Assez ancien [38, 18], ce mode de représentation a permis aux premiers systèmes d'information géographiques de réduire l'utilisation de l'espace disque en sauvant les points, les arrêtes et les polygones constituant les géométries dans des tables différentes de façon à en limiter la redondance. C'est donc un format qui limite par factorisation les répétitions d'arcs d'une géométrie à l'autre, si bien qu'il réduit l'espace de stockage nécessaire. Par exemple, il permet de n'utiliser que six points au lieu de huit pour représenter deux carrés partageant une arrête. Actuellement, il reste intéressant pour certaines applications comme la diffusion d'informations géographiques sur le web, puisque son format permet des messages plus compacts que GeoJSON.

1.2.3 Normes spatiales décrivant spécifiquement l'interopérabilité

Par les normes décrivant la composante spatiale et les fonctions qu'il est possible d'y appliquer, l'OGC et l'ISO font un premier pas vers l'interopérabilité de celle-ci. Étant donné que le système envisagé dans le cadre de la présente recherche vise également à être interopérable afin de pouvoir facilement communiquer avec les systèmes d'information géographiques, deux autres jeux de normes qui visent plus spécifiquement à renforcer l'interopérabilité des systèmes gérant les données spatiales sont décrits ici : la spécification de services web permettant l'accès à l'information spatiale et la gestion de méta-données.

Décrites par les normes ISO-19115 et ISO-19119 les **méta-données** permettent de connaitre le format des données géographiques partagées, ainsi que l'historique de leur modification et une indication de leur niveau de qualité. La norme ISO-19139 décrit pour sa part les méta-données concernant les services géographiques eux-mêmes. Malgré l'importance de telles informations pour l'intégration de données spatiales hétérogènes et pour l'élaboration de cartes de qualité, elles ne feront pas ici l'objet d'un développement, car elles n'ont pas d'impact direct sur la structure interne des jeux de données vectorielles.

Les **services web** définis en vue d'un usage spatial sont quant à eux résumés dans la table 1.2, à partir de l'inventaire qu'en a fait Krimbacher en 2014 [73]. Les normes ISO et OGC associées sont également évoquées. Parmi ceux-ci c'est principalement le service WFS qui est susceptible d'intéresser la présente recherche, car il permet de sélectionner et de modifier des données spatiales vectorielles en interrogeant le service web à l'aide de simples requêtes HTTP.

Service web	Abréviation	Finalité	Principale norme associée
Web Map Service	WMS	Rendre disponibles des cartes depuis le Web.	ISO-19128, OGC Web Map Server Implementation Specification
Web Map Tile Service	WMTS	Rendre disponibles des cartes de type raster depuis le Web.	OGC Web Map Tile Service Implementation Standard
Web Feature Service	WFS	Rendre disponibles des données vectorielles depuis le Web.	OGC Web Feature Service 2.0 Interface Standard
Web Processing Service	WPS	Fournir une interface générique pour le traitement des données géospatiales.	OGC WPS 2.0 Interface Standard

TABLE 1.2 – Types de services web définis par l'ISO et l'OGC

1.2.4 Systèmes d'information géographique

A présent que les normes impliquées par l'interopérabilité de la composante spatiale ont été étudiées, il reste à présenter les systèmes d'information géographique qui les emploient pour représenter et diffuser des données spatiales. Dans le cadre de la présente recherche, il s'agira en effet de définir comment les relier avec le système de gestion des données spatiales massives envisagé. Les SIG se situent en effet à l'intersection des bases de données, de la cartographie et des analyses spatiales, si bien que leur emploi semble promettre des impact sur la cartographie et les analyses spatiales, impliquant de nouvelles recherches et réflexions.

Au sens large, le terme « système d'information géographique » (SIG) s'entend parfois comme une infrastructure humaine, matérielle et technique visant à produire, consulter et analyser des cartes à partir de données géographiques. Le terme SIG sera employé ici dans le sens plus restreint d'une suite logicielle à même de permettre ou de faciliter ces opérations. Classiquement, celle-ci gère cinq étapes du cycle de vie de l'information géographique, que Denègre et Salgé appellent les « cinq A » [31] : son abstraction, son acquisition, son archivage, son analyse et son affichage.

Cette section présente trois SIG qui restent largement utilisés en 2017. Elle évoque ensuite brièvement

leurs limites par rapport à ce qu'une gestion efficiente des données spatiales massives nécessiterait.

1.2.4.1 Présentation de quelques SIG

MapInfo Apparu en 1986, MapInfo [13] était à l'époque le premier SIG utilisable depuis un ordinateur personnel. Depuis, l'outil a été continuellement enrichi et constitue aujourd'hui une solution propriétaire mature pour le traitement de données spatiales aussi bien vectorielles que matricielles. Il dispose de son propre format de stockage (TAB) et de d'échange (MID/MIF) de fichiers contenant des données spatiales, mais permet aussi d'échanger ceux-ci avec d'autres logiciels. MapInfo est en effet interopérable avec PostGIS, Oracle, et les formats WFS et WMS définis par l'OGC.

ArcGIS [14] est une suite propriétaire à visée géomatique mise sur le marché en 1999 par ESRI. Née de l'intégration d'outils antérieurs en une solution unique, elle représentait en 2004 un coût de production de plus de 100 millions de dollars, et permettait de gérer plusieurs centaines de formats de données [105]. Depuis sa version 9, la composante ArcGIS Server permet son installation et son emploi depuis un serveur comme alternative à une utilisation purement bureautique. Malgré sa puissance et l'ambition qui la guide, la solution proposée par ESRI ne respecte qu'en partie les normes OGC et ISO, par exemple en ce qui concerne la dénomination et le fonctionnement des opérateurs spatiaux.

QGIS Disponible depuis 2002, QGIS [15] permet comme les solutions précédentes de traiter, d'afficher et d'échanger des données à référence spatiale. Mais il s'agit cette fois d'un logiciel libre et gratuit, distribué sous licence GNU GPL par la Fondation Open Source Geospatial (OSGeo). L'OSGeo a signé en 2008 un accord de coopération avec l'OGC en vue de rendre les technologies conformes aux normes OGC, ce qui témoigne de la bonne compatibilité de QGIS avec ces standards. Permettant dès sa première version de gérer des données vectorielles et d'accéder à PostGIS, l'outil permet également depuis 2004 de gérer le format matriciel. Plus récemment, son extension QGIS Mobile permet de le relier directement à la situation de terrain à l'aide de téléphones intelligents [75].

1.2.4.2 Limites des SIG actuels

S'ils sont adaptés au traitement de données spatiales ordinaires, les SIG du marché semblent mal préparés à gérer les données massives. En effet, leur architecture est généralement trop centralisée pour permettre la distribution des traitements entre plusieurs serveurs, si bien qu'elle n'autorise qu'un faible degré de scalabilité horizontale [9].

Afin de contourner cet état de fait, la connectivité des SIG classiques a été étendue à des systèmes à même de gérer les données massives, comme la base de données NoSQL par documents MongoDB [16], et l'écosystème Hadoop [65]. Mais ces adaptations n'augmentent pas la puissance de traitement du

13. http://mapinfo.com
14. http://resources.arcgis.com
15. http://qgis.org
16. https://plugins.qgis.org/plugins/qgis-mongodb-loader/

SIG lui-même, qui nécessite des découpes et des généralisations dans les données massives avant de pouvoir les représenter. L'affichage même de volumes élevés de données représente en effet un défi que ces logiciels ne peuvent adresser.

Il resterait donc à définir une architecture offrant une interaction directe avec les données spatiales massives, de façon à permettre au décideur de les gérer et de les afficher efficacement. Mais un tel projet nécessiterait aussi sans doute d'inventer de nouvelles méthodes cartographiques [10], puisque celles définies par Bertin et encore largement utilisées aujourd'hui ont été conçues à une époque où les cartes étaient essentiellement statiques et concernaient des jeux de données limités.

La possibilité de rendre un SIG à même de traiter les données spatiales massives reste donc à plus d'un titre un problème ouvert. Malgré son intérêt et son importance, une telle entreprise dépasserait cependant le cadre de la présente recherche. Il s'agira seulement ici d'y proposer une base technique stable, à travers un système capable de manipuler rapidement d'importants volumes de données spatiales. C'est dans cet esprit que la notion d'indexation spatiale sera abordée en détail dans la partie suivante, car elle pourrait constituer une façon d'accélérer les traitements du système de gestion des données spatiales massives envisagé.

1.3 Indexation spatiale

Indexer des données peut s'entendre comme le fait de leur appliquer une transformation qui facilite leur accès en lecture et/ou en écriture. Lorsqu'il s'agit de données spatiales, des techniques particulières peuvent être employées, puisqu'il devient possible de connaître les positions associées aux phénomènes représentés par les données ainsi que les relations de proximité qui les relient [41]. Dans ce contexte, les méthodes d'indexation spécifiques à la composante spatiale consistent généralement à exploiter ces informations supplémentaires pour réduire le volume de données à traiter en les regroupant par proximité. Elles permettent donc d'accélérer considérablement des opérations coûteuses comme la jointure de données spatiales, qui nécessiterait sinon de recourir à un produit cartésien entre les données à joindre.

1.3.1 Méthodes d'indexation spatiale

Eldawi et Mokbel [41] indiquent qu'il existe deux grandes catégories d'indexation spatiale : l'indexation plate et l'indexation hiérarchique. La première découpe simplement les données géographiques en secteurs, tandis que la seconde les classe à l'aide d'un arbre. Cette sous-section décrit quelques méthodes d'indexation appartenant à ces catégories.

Parmi les méthodes d'indexation spatiale plane, Grid-File semble la plus simple et la plus générique et sera d'abord abordée. Plus complexe, la méthode plane du diagramme de Voronoï sera ensuite brièvement décrite. En ce qui concerne les méthodes d'indexation spatiale hiérarchique, Kothuri et al. [72] indiquent dans une revue de la question l'existence de Quadtree, R-tree, hB-tree, TV-tree,

SS-tree et SR-tree. A des fins de concision, seules les deux premières seront détaillées ici, car SR-Tree, SS-Tree et TV-Tree sont de simples variantes de R-Tree [23] tandis que hB-tree semble surtout adaptée à un contexte multidimensionnel qui s'éloigne du cadre de notre recherche [78]. L'indexation plane à l'aide de GeoHash sera ensuite développée, car elle offre un compromis intéressant entre R-Tree et Quadtree. Son emploi par le récent modèle de système de gestion des données spatiales massives proposé par Cortés et al.[20] 1.3.4 incite par ailleurs à la considérer avec attention dans le cadre de cette recherche. Enfin, une méthode récente combinant GeoHash et R-Tree dans un contexte de données massives sera présentée.

Grid-file est une méthode d'indexation plate qui consiste à simplement découper l'espace en cellules identiques à l'aide d'un motif régulier (généralement le carré). Elle a été été proposée en 1984 par Nievergelt et al. [92] en tant que moyen de parcourir des fichiers (typiquement des images) à l'aide de plusieurs clés. Très facile à implémenter, elle permet un haut degré de scalabilité. Par contre, dans un contexte géographique, elle mène à un grand nombre de découpes inutiles lorsque les géométries à considérer n'occupent qu'une part restreinte du territoire ou lorsque la taille des cellules est mal paramétrée [123].

Diagramme de Voronoi Décrit dès 1994 par Okabe et al. [96], la méthode du diagramme de Voronoi consiste à découper l'espace en zones inter-connectées irrégulières, de façon à ce que les géométries de chaque zone aient le même plus proche voisin. Ce mode d'indexation plane a été originellement conçu pour faciliter un algorithme de recherche du plus proche voisin appliqué à des points situés dans un espace en deux dimensions, mais il a été depuis étendu et appliqué à bien d'autres situations : géolocalisation continue, lutte contre les virus informatiques, travail sur des données incertaines [16]. Il a aussi été récemment appliqué à un contexte de données spatiales massives par GeoSpark [127].

Quadtree Quadtree est une méthode d'indexation hiérarchique qui connait un grand nombre d'applications, celles-ci ne se limitant pas aux données à référence spatiale. Elle est par exemple employée pour optimiser les réseaux de peer-to-peer [114], pour faciliter la transformation d'images et de vidéos [112] et pour la planification d'itinéraires en robotique [93]. Cette richesse d'utilisation peut s'expliquer par l'ancienneté de la méthode, puisqu'elle fut proposée par Finkel et Bentley dès 1974 [49] comme moyen d'accéder à des données via une clé composée. Elle consiste en un arbre dont chaque nœud non-terminal a exactement quatre fils. Dans le cas d'une utilisation géographique, les descendants d'un même nœud sont associés à des zones géographiques voisines de même taille, formant autant de subdivisions que les données en nécessitent pour être accédées efficacement. Selon Yu et al. [127], cette méthode présente des performances proches de celles de R-Tree pour des tâches de jointure spatiale sous Spark. Si elle semble permettre une moins bonne scalabilité horizontale que que Grid-file, car les données y sont d'avantages inter-reliées, Quadtree a pour avantage d'entrainer moins de découpes inutiles de l'espace que ce dernier [123], surtout lorsque les données ne sont pas localisées de façon homogène dans le territoire considéré. La figure 1.4 présente ainsi une découpe de

l'espace en 64 cases sous Grid-File qui ne nécessite que 28 cases sous Quadtree pour une répartition équivalente des données.

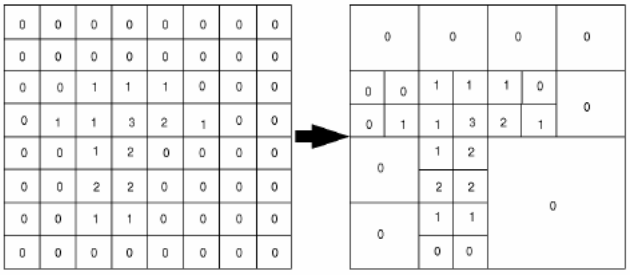

FIGURE 1.4 – Économie permise avec Quadtree (à droite) par rapport à Grid-file (à gauche) en termes de découpes de l'espace [70].

R-Tree a été proposée en 1984 par Guttman comme une alternative dynamique à Quadtree [62]. Elle emploie une structure en arbre dans laquelle chaque nœud contient une enveloppe qui englobe les géométries de tous les nœuds fils. Contrairement à ce qui se passe avec Quadtree, un nœud non-terminal n'a donc pas nécessairement quatre fils, et ses fils ne couvrent pas toujours des surfaces identiques. Sa découpe de l'espace s'avère donc moins rigide qu'avec Quadtree, pour des performances équivalentes sous GeoSpark [127]. Ceci explique peut-être que malgré son ancienneté, R-Tree soit implémentée et étendue par beaucoup de systèmes d'indexation spatiale récents. Dotée d'une moins bonne scalabilité horizontale que Grid-File et Quadtree en raison de sa structure plus hiérarchique, elle est par contre très efficace pour éviter des découpes inutiles de l'espace [123].

La figure 1.5 illustre les modes d'indexation Grid-File, Quadtree et R-Tree. La position intermédiaire de Quadtree par rapport à l'approche plane de la première méthode et l'organisation en arbre de la troisième y apparaît clairement.

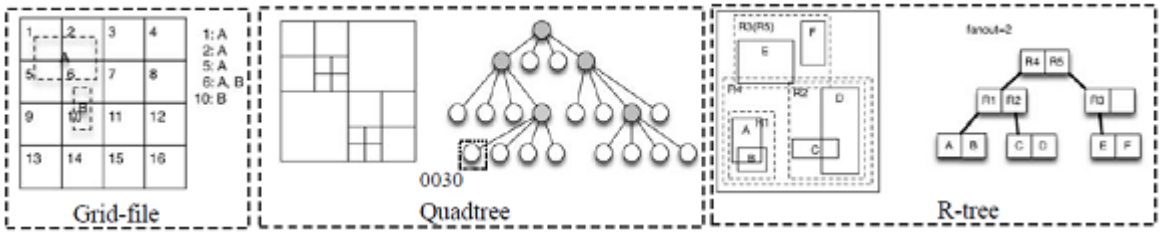

FIGURE 1.5 – Fonctionnement de Grid-File, Quadtree et R-Tree [123].

GeoHash n'était au départ qu'une façon compacte de représenter les coordonnées d'une zone géographique, définie par Niemeyer en 2008 [91]. Bien que ce ne soit pas sa visée initiale, GeoHash a ensuite été employée à des fins d'indexation spatiale des données massives par des solutions aussi variées que Geomesa [67], ElasticSearch et BIG-LHT [20]. La raison de cette popularité peut s'expliquer par sa souplesse et sa simplicité de mise en œuvre : l'index s'y réduit à une clé numérique, à partir de laquelle il est possible de déduire la latitude et la longitude d'une géométrie. A chaque chiffre de cette

clé correspond une zone géographique contenue dans celle exprimée par la grille précédente, si bien que plus la clé est longue plus elle délimite précisément une zone. La figure 1.6 illustre ce fonctionnement. Il y apparaît que GeoHash permet comme Quadtree de découper l'espace selon différentes échelles, mais en représentant cette fois ces découpes par un simple nombre au lieu d'employer une organisation en arbre. Cette simplicité d'utilisation fait de GeoHash un outil facile à intégrer dans des bases de données NoSQL organisées par clé [50]. Si sa version initiale permet seulement de découper l'espace 2D, il a aussi connu plusieurs tentatives d'adaptation à la composante temporelle [20, 50].

FIGURE 1.6 – Représentation des découpes de l'espace permises par GeoHash [86].

GeoHash + R-Tree Takasu et al. [113] ont appliqué une combinaison de GeoHash et R-Tree aux architectures distribuées. Dans leur modèle, GeoHash est utilisé pour découper hiérarchiquement l'espace en grille, dont les données sont ensuite indexées par autant d'index R-Tree qu'il existe de cases dans la première grille. Ces auteurs ont montré l'efficacité de leur solution dans un environnement distribué, avec une réactivité proche du temps réel dans certains cas. Cependant ces performances sont dues à la faible taille des index R-Tree construits, si bien qu'un maintien de l'efficacité lorsque la densité des données par case croît ne semble pas assuré. De plus, ce modèle implique l'utilisation d'une base de données externe, ce qui ne le rend pas agnostique par rapport au format des données et implique des accès disque nombreux.

1.3.2 Application à la jointure spatiale

Les différents types d'indexation spatiales qui viennent d'être parcourus ont pour objectif commun d'améliorer la vitesse de traitement des données spatiales. Une application essentielle de l'indexation

spatiale, à savoir l'amélioration du temps d'exécution et de la charge mémoire liée aux opérations de **jointure spatiale** sera détaillée ici à titre d'illustration.

Une jointure spatiale consiste à sélectionner, parmi les éléments de deux jeux de données, les couples qui respectent une relation spatiale donnée (généralement en rapport avec leur proximité). Il s'agit d'une opération fort coûteuse lorsqu'elle est mise en œuvre de façon naïve, car elle implique de tester chaque couple possible et donc d'établir un produit cartésien entre les jeux de données [20]. Elle pourrait donc constituer un critère d'évaluation pertinent quant à l'efficacité du système d'indexation envisagé dans le cadre de la présente recherche.

Une stratégie générale appelée **partitionnement** consiste à créer plusieurs groupes de géométries, en découpant l'espace par secteur ou simplement en regroupant les géométries voisines. Ensuite, seules les géométries situées dans une zone commune sont comparées. A première vue, cette méthode devrait entrainer une assez bonne scalabilité horizontale, puisqu'elle découpe les données en plusieurs blocs accessibles ensuite à un traitement en parallèle. Mais dans la pratique, cela dépend du type de découpe envisagé : la scalabilité est meilleure si le partitionnement repose sur un index plat plutôt que sur un index hiérarchique, car les zones définies hiérarchiquement forment des arbres dont certaines branches continuent d'être interdépendantes après découpe [123]. Donc, les méthodes d'indexation Grid-File, GeoHash et par Voronoï seraient à priori de meilleurs candidats pour la création d'index spatiaux des données massives que les méthodes hiérarchiques comme R-Tree et Quadtree, puisque ce besoin de scalabilité horizontale implique une répartition indépendante des données. Dans la pratique cependant, ce n'est pas toujours le cas, car les structures plates gèrent plus mal les données spatiales réparties de façon non-homogène que les index hiérarchiques. C'est peut-être ce qui explique la bonne réactivité permise par une solution qui combine ces deux types d'index, comme celle de Takasu et al. [113].

Une stratégie complémentaire visant à faciliter la jointure spatiale consiste à construire pour chaque géométrie une représentation homogène et simplifiée : le rectangle minimal orthogonal dont les côtés sont parallèles aux axes de coordonnées et qui entoure la bordure extérieure de la géométrie (avec éventuellement une marge de tolérance, définie en fonction des besoins). Ces rectangles sont alors appelés **enveloppes**, ou « Minimum Bounding Rectangles ». Une fois ceux-ci construits, une première jointure approximative peut être effectuée en les utilisant, de façon à simplifier ensuite le calcul d'une jointure exacte pour les enveloppes jugées proches [123]. Si l'emploi d'enveloppe peut être associé à tous les types d'indexation, il s'accorde particulièrement bien avec R-Tree, étant donné que chaque nœud de l'arbre employé par cette méthode contient une enveloppe englobant celles de ses fils. Cependant, l'emploi de cette stratégie pose question lorsque les phénomènes du territoire étudié sont majoritairement localisés à l'aide de points, puisqu'elle consiste dans ce cas particulier à remplacer par quatre coordonnées identiques des géométries qui n'en contiennent qu'une.

Les deux sections qui suivent ont une portée plus pratique, puisqu'elles détaillent l'implémentation des différents index spatiaux par les systèmes existants. Étant donné leur intérêt relativement à la présente recherche, l'accent sera mis sur les systèmes dédiés aux données massives, mais deux systèmes

classiques seront d'abord brièvement abordés.

1.3.3 Quelques implémentations classiques

Les systèmes classiques de gestion des données spatiales JTS et PostGIS 1.2.2 ont été conçus pour l'essentiel à une époque où le traitement des données massives n'apparaissait pas comme une nécessité, si bien que leur implémentation n'intègre pas directement la notion de scalabilité. Les classes de JTS permettent cependant de gérer différents types d'indexation : Voronoï, R-Tree et QuadTree. PostGIS est fourni quant à lui avec un système d'indexation spatiale propre appelé GiST, qui repose sur R-Tree. Aucun de ces systèmes ne propose par contre une implémentation de GeoHash, sans doute en raison de son apparition beaucoup plus récente.

1.3.4 Quelques implémentations pour données massives

BIG-LHT est un système de gestion des données spatiales massives proposé par Cortés et al. [20], qui permet des traitements spatiaux entièrement décentralisés et tenant compte de la composante temporelle. Les données reçues y sont traitées selon deux phases : un jeu de serveurs de premier niveau les redirige suivant leur localisation spatiale en employant leur code GeoHash vers un jeu de serveurs de second niveau, plus nombreux, qui les stocke en fonction de leur date de réception. Le nombre de serveurs appartenant à chaque niveau évolue dynamiquement en fonction des besoins rencontrés et en particulier de la quantité de données associée à chaque zone géographique, ce qui semble permettre une bonne scalabilité horizontale. Ce système a cependant le défaut d'avoir été conçu sans le support d'un outil de traitement des données massives existant, ce qui semble impliquer un prototype difficile à maintenir et à étendre. En particulier, BIG-LHT exclue d'emblée toute compatibilité avec Hadoop et Spark, sous le prétexte qu'ils présentent un défaut structurel : bien que répartie quant à ses traitements, l'indexation spatiale n'y peut être dirigée que de façon centralisée, à partir d'un serveur maître qui répartit les traitements entre les autres serveurs. Une telle hiérarchisation nuirait selon ses auteurs à la scalabilité horizontale de l'indexation. Pour y remédier, ils définissent donc un mode de traitement moins centralisé, mais qui renonce à l'environnement Hadoop pour se baser plutôt sur le système de peer-to-peer Free Pastry [17]. Dans un tel contexte, il peut être intéressant de se demander si la rigidité de l'architecture centralisée d'Hadoop n'est pas compensée par sa maturité, la diversité des éléments qui constituent son écosystème et l'évolutivité qu'il permet. Assez curieusement, BIG-LHT reproduit d'ailleurs implicitement une telle hiérarchie, en impliquant deux niveaux de serveurs dont le premier gère le second. Il ne semble donc pas établi qu'un système ayant une aussi bonne scalabilité horizontale que le leur mais ne renonçant pas aux possibilités de l'écosystème Hadoop soit inconcevable.

Geomesa est un système de gestion des données spatiales massives conçu en 2015, qui est compatible avec Hadoop et depuis peu avec Spark [67]. Il repose sur la base de données par clé Accumulo [18].

17. http://www.freepastry.org/
18. https://accumulo.apache.org/

Il s'agit du seul prototype permettant à la fois de lancer des requêtes dans un format proche de SQL (ECQL), et de visualiser les données. Il est cependant limité dans les requêtes spatiales qu'il permet, ayant initialement été conçu pour détecter l'inclusion de points dans une enveloppe. De plus, les tests de Xie et al. (2015) montrent que la durée de construction de son index spatial sous-performe Spatial Spark, GeoSpark, Simba et Hadoop GIS [121].

Spatial Spark fut historiquement le premier système de gestion des données spatiales massives reposant sur Spark à proposer une indexation spatiale. Conçu par des chercheurs de la City University de New York en 2015 [124], il repose in fine sur l'implémentation de R-tree de JTS, qu'il applique à des données chargées depuis un fichier CSV et transitant dans des Resilient Data Sets (RDD). Il n'offre hélas aucune interface avec Spark SQL, n'étant au contraire pilotable que par les paramètres reçus en ligne de commande. Ce dernier point rend difficile l'usage de Spatial Spark pour des tâches d'analyses spatiales, puisqu'il ne permet pas d'interroger les données à l'aide de requêtes SQL.

GeoSpark a été proposé en 2015 par des chercheurs de la State University d'Arizona [127]. Ce système ajoute à Spark une sous-classe de RDD qui lui permet de gérer la composante spatiale dans un espace partitionné. De même que Spatial Spark, le prototype repose sur JTS, dont il intègre trois méthodes d'indexation : Voronoi, Quadtree et R-tree. Plus proche de Spark que Spatial Spark dans son implémentation, il semble garantir de meilleures performances. Malheureusement, cette approche rend également plus difficile le portage vers des versions ultérieures de Spark, et rend problématique l'intégration de la solution à Spark SQL. Ainsi, d'après Xie et al., GeoSpark « is a library running on top of and outside of Spark without a query engine » [121]. Là encore, un tel système semble donc peu adapté à des tâches d'analyse spatiale.

Simba est un système proposé conjointement par les universités de l'Utah et de Shangaï en 2015 [121]. L'outil intègre Spark mais remplace Spark SQL par sa propre implémentation. Celle-ci consiste à organiser les données en tableaux à deux dimensions plutôt que sous forme de séquences, de façon à faciliter leur partitionnement spatial et donc leur indexation. De même que GeoSpark, cette solution propose plusieurs types d'indexation, mais celles-ci sont cette fois entièrement pilotées par des requêtes SQL. Si l'approche est séduisante, elle se heurte dans la pratique à plusieurs écueils. Premièrement, elle empêche de travailler avec une version standard de Spark, puisqu'elle nécessite qu'il soit recompilé avec le code de Simba en remplacement de celui de Spark SQL. Elle semble donc difficilement transportable à une nouvelle version de Spark, même si les concepteurs ont veillé à ne pas retoucher son noyau. Deuxièmement, les seules géométries qu'elle permet de traiter sont le point et en partie les enveloppes, alors que la norme ISO-19125 implique aussi de gérer la polyligne, le polygone et les géométries composées. Troisièmement, nos tests ont révélé des incohérences dans son analyseur syntaxique SQL, qui par exemple oblige à écrire « IN » après l'expression « POINT(x, y) » même quand aucun test d'inclusion n'est envisagé. Ces éléments ne contribuent pas à faire de Simba une solution mature et pérenne, même s'il a le mérite d'être la première tentative d'indexation spatiale intégrant Spark SQL.

1.4 Conclusion

Afin de faciliter la rencontre de l'objectif de cette recherche, qui consiste à concevoir un moteur de gestion des données spatiales vectorielles complet et efficace dans un contexte de données spatiales massives, l'état de l'art a exploré les trois principaux concepts impliqués par cette fin.

Le concept de données massives a d'abord été défini, et deux grandes catégories de solutions permettant de gérer celles-ci ont été détaillées : les systèmes de gestion de données NoSQL et les environnements de gestion des données massives comme Hadoop. Les seconds se sont avérés plus généraux et plus adaptés à des traitements massifs que les premiers, ce qui en fait de meilleurs candidats pour servir de base à l'élaboration d'un moteur spatial efficace.

Le concept de composante spatiale a ensuite été défini, puis détaillé en fonction des différentes normes et recommandations qui décrivent celle-ci. Il est apparu que si la norme ISO-19125 peut servir de socle à une description de la composante spatiale adaptée à un contexte géographique, il existe des extensions et applications de celle-ci, comme la gestion de l'élévation, qui gagneraient à être intégrées par un moteur de gestion de la composante spatiale véritablement complet.

Le concept d'indexation spatiale a finalement été défini, et les méthodes qui permettent de la mettre en œuvre ont été détaillées et comparées. Il est apparu à cette occasion que les moins hiérarchiques de ces méthodes, si elles sont les moins économiques en terme de découpes de l'espace, présentent aussi souvent une meilleure scalabilité horizontale dans un contexte distribué. Dans un second temps, des systèmes de gestion des données spatiales implémentant ces méthodes d'indexation ont été étudiés, avec un fort accent mis sur ceux qui permettent de traiter les données massives. Ces derniers se sont avérés à chaque fois limités quant à leurs performances ou aux possibilités d'analyse spatiale qu'ils offraient. Proposer un système de gestion des données spatiales massives plus complet et efficace s'est donc imposé comme une nécessité.

Chapitre 2

Gestion des données massives à composante spatiale sous Spark SQL

2.1 Introduction

Ce chapitre décrit un premier modèle du moteur de gestion des données spatiales massives vectorielles reposant sur l'écosystème Hadoop envisagé. Seuls les deux volets de la norme ISO-19125 lui sont appliqués à cette étape, car ils suffisent à le doter de l'autonomie nécessaire pour gérer toutes les géométries élémentaires 2D et pour répondre à des requêtes relationnelles. L'ajout d'un module d'indexation spatiale et d'extensions issues d'autre normes et recommandations sont donc laissés respectivement aux chapitres 3 et 4.

Le corps de ce chapitre est constitué d'un article, qui commence [2.2.1] par évoquer l'importance des données massives à référence spatiale [cf. 1.1], et le rôle d'Hadoop et de Spark [cf. 1.1.1.2]. Dans une seconde partie [2.2.2], il détaille les lacunes de quelques systèmes de gestion des données spatiales massives reposant sur ces environnements par rapport au respect de la norme ISO-19125 [cf. 1.2.1.2]. Le modèle d'un système rencontrant cette norme est ensuite présenté et justifié et quelques détails sont fournis quant à son implémentation [2.2.3.2]. Finalement, [2.2.5], la façon dont ce modèle pourrait être rendu plus performant par l'ajout d'une prise en compte de l'indexation spatiale [cf. 1.3.4] est brièvement traitée [2.2.4].

2.1.1 Contributions

L'article qui constitue le corps de ce chapitre est référencé ci-dessous.

[43] Engélinus, J. et Badard, T., 2017, « Elcano : un système de gestion des données massives à composante spatiale basé sur Spark SQL », Revue internationale de géomatique (en cours de soumission).

2.2 Corps de l'article

Titre : Elcano : un système de gestion des données massives à composante spatiale basé sur Spark SQL

Auteurs : Jonathan Engélinus, Thierry Badard

Résumé: Les données massives se situent au cœur de beaucoup d'enjeux scientifiques et sociétaux, et leur volume global ne cesse de croître. Alors que la plupart d'entre elles intègrent une composante spatiale vectorielle, il existe peu de systèmes de gestion de ces données à même de tenir compte de celle-ci. Ainsi, alors que Spark est peut-être à l'heure actuelle l'environnement de gestion des données massives le plus efficace et le plus facile à étendre, il n'est employé que par cinq systèmes gérant l'analyse de données spatiales vectorielles. Parmi ces solutions, aucune ne respecte entièrement les standards ISO et les spécifications OGC en termes de traitements spatiaux, et plusieurs présentent des performances limitées ou un manque d'extensibilité. Dans cet article, les auteurs cherchent une façon de dépasser ces limitations. Après une étude détaillée des lacunes des solutions existantes, ils

définissent donc un système plus respectueux des standards. La solution proposée, Elcano, est une extension de Spark permettant le requêtage de données spatiales et compatible avec la norme ISO-19125.

Abstract: Big data are in the midst of many scientific and economic issues. Furthermore their volume is continuously increasing. As a result, the need for management and processing solutions has become critical. Unfortunately, while most of these data have a vectorial spatial component, almost none of the current systems are able to manage it. For example, while Spark may be the most efficient environment for managing big data, it is only used by five spatial data management systems. None of these solutions fully complies with ISO standards and OGC specifications in terms of spatial processing, and many of them are neither performant enough nor extensible. The authors seek a way to overcome these limitations. Therefore, after a detailed study of the limitations of the existing systems, they define a system in greater accordance with the ISO-19125 standard. The proposed solution, Elcano, is an extension of Spark complying with this standard and allowing the SQL querying of spatial data.

Mots-clés : Elcano, ISO-19125, Magellan, Spark SQL, données massives

Keywords: Elcano, ISO-19125, Magellan, Spark SQL, big data

2.2.1 Introduction

Aujourd'hui, les besoins en termes de systèmes capables de traiter efficacement de grandes quantités de données spatiales sont devenus très importants. En effet, la rencontre de la cartographie et d'Internet a fait entrer celle-ci dans un nouveau paradigme que la littérature qualifie de néogéographie. Celui-ci est marqué par « une grande interactivité et des contenus géolocalisés générés par les utilisateurs » [89]. Le succès considérable de la néogéographie a entrainé l'inflation de la production et de la consultation de données géoréférencées. Celle-ci est encore renforcée par l'arrivée sur le marché de nouveaux capteurs comme les puces GPS des téléphones intelligents [8]. Cette situation entraîne un regain d'intérêt important envers la cartographie, mais elle en rend aussi l'organisation plus complexe, puisqu'il devient de plus en plus difficile de gérer et de représenter de telles quantités de données à l'aide d'outils classiques [46].

L'environnement Hadoop [120], qui est actuellement le projet le plus important de la fondation Apache, constitue un standard de fait pour le traitement et la gestion de données massives. Cet outil, qui est très populaire et impliqué dans le succès de nombreuses startups [48], implémente l'algorithme MapReduce [28], qui permet de distribuer des traitements entre les serveurs d'un cluster puis d'en réunir les résultats de façon cohérente. Les données à traiter sont elles-mêmes réparties entre les serveurs par le système de fichiers distribués HDFS [1], qui est fourni nativement avec Hadoop. Il en résulte un haut degré de scalabilité horizontale, celle-ci pouvant être définie comme la possibilité d'augmenter linéairement les performances d'un système reposant sur plusieurs serveurs en fonction des besoins. Un véritable écosystème d'éléments interopérables s'est constitué autour d'Hadoop, permettant de gérer des aspects aussi variés que le streaming (Storm [2]), la sérialisation (Avro [3]) et l'analyse de données (Hive [4]).

A partir de 2014, l'AMPLab de l'Université de Berkeley a développé un nouvel élément de l'écosystème Hadoop, qui a depuis été repris par la fondation Apache. Il s'agit de Spark, qui propose une alternative intéressante à HDFS et à MapReduce. Dans Spark en effet, les données et le code des traitements à leur appliquer sont répartis par petits blocs appelés RDD [5] sur l'ensemble de la mémoire vive du cluster. Ce choix architectural, qui limite fortement les accès disque, rend Spark jusqu'à dix fois plus rapide qu'une utilisation classique d'Hadoop dans certains cas [128], même si c'est au prix d'une charge mémoire plus importante [59]. En outre, un composant appelé Spark SQL [6] enrobe les RDD Spark d'une surcouche appelée « DataFrames » qui permet d'organiser les données reçues par Spark en tables temporaires et de les interroger à l'aide de requêtes SQL. Spark SQL réduit aussi les temps de traitement de Spark, par une optimisation stratégique des requêtes exécutées et la sérialisation des données manipulées. Il autorise enfin la définition de types de données personnalisés (UDT [6]) et la

1. HDFS est un acronyme de « Hadoop Distributed File System »
2. https://storm.apache.org/
3. https://avro.apache.org/
4. https://hive.apache.org
5. RDD est un acronyme de « Resilient Distributed Dataset »
6. UDT est un acronyme de « User Defined Type »

création de fonctions personnalisées (UDF[7]), qui permettent respectivement de rendre de nouveaux types de données et de traitements accessibles depuis SQL.

Cette possibilité d'interroger les données massives à l'aide de requêtes SQL pourrait s'avérer un support utile à leur analyse, afin de mieux comprendre les phénomènes qu'elles illustrent. Il serait en particulier intéressant de pouvoir l'appliquer à leur composante de localisation spatiale, qui selon Franklin est présente dans 80% des données d'entreprise [51]. Selon une étude du cabinet McKinsey [84] souvent citée, une meilleure utilisation de la localisation spatiale des données massives pourrait d'ailleurs rapporter 100 milliards de dollars pour les fournisseurs de services et de l'ordre de 700 milliards pour les utilisateurs finaux. La gestion des données massives à référence spatiale apparait donc comme un enjeu économique, scientifique et sociétal important. Spark semble en ce domaine aussi une solution prometteuse, puisqu'il traite les données spatiales massives jusqu'à sept fois plus vite qu'un autre élément d'Hadoop gérant le SQL (Impala[8]). [124].

Il existe actuellement quelques systèmes permettant de gérer des données spatiales massives à partir d'Hadoop, comme Hadoop GIS [2], Geomesa [67] et Pigeon [40]. Mais il s'agit généralement davantage de prototypes que de technologies matures [8].

Parmi ces systèmes, seulement cinq proposent une gestion des données spatiales à partir de Spark. Les deux premiers, Spatial Spark [124] et GeoSpark [127], se contentent d'ajouter une gestion de la composante spatiale à la version de base de Spark, malgré les gains de performance et les requêtages SQL qu'aurait permis la prise en compte de son module SQL. Au lieu de gérer des requêtes SQL, l'outil n'est donc pilotable sous sa forme actuelle qu'à l'aide de paramètres passés en ligne de commande. GeoSpark repose quant à lui sur sa propre extension spatiale du type RDD de Spark, qui ne le rend pas directement compatible avec Spark SQL [126]. Le troisième, Magellan [104], définit des types de données spatiales directement utilisables en Spark SQL, mais sans gérer correctement certaines opérations spatiales comme l'union de polygones disjoints, les différences symétriques impliquant plus d'une géométrie et la création d'enveloppe. Le quatrième, Simba [121], autorise des requêtes SQL mais sans que celles-ci permettent de gérer d'autres géométries que le point et d'exécuter des fonctions spatiales standards sur celles-ci. Enfin, le cinquième prototype est une extension de Geomesa, qui permet son utilisation depuis Spark. Le système est cependant limité dans les requêtes spatiales qu'il permet, n'ayant été initialement conçu que pour la recherche des points inclus dans une enveloppe. Il présente de plus des performances limitées [121] par rapport aux autres outils, ce qui s'explique peut-être par le fait qu'il impose l'emploi d'une base de données par clé (Accumulo[9]) pour stocker les données spatiales à traiter.

Il n'existe donc à l'heure actuelle aucun système de gestion des données spatiales gérant tous les types de géométries 2D et permettant des requêtes SQL sur des données spatiales à partir de Spark. Tous les modèles qui approchent un tel objectif présentent des capacités limitées. Celles-ci portent aussi

7. UDF est un acronyme de « User Defined Function »
8. https://impala.apache.org/
9. https://accumulo.apache.org/

bien sur les types de géométries 2D disponibles que sur les traitements spatiaux qu'il est possible de leur appliquer. Cette situation est malencontreuse, étant donné l'intérêt économique, scientifique et sociétal qu'aurait une amélioration du traitement des données spatiales massives et l'apparent potentiel de Spark et de son module Spark SQL en ce domaine.

Pour y remédier, le présent article étudie dans une première section les limites des solutions Spark actuelles. Sur base de cette étude, il propose ensuite un nouveau modèle qui les surpasse. Celui-ci permet en effet de gérer tous les types de géométries 2D et toutes les fonctions spatiales élémentaires, telles qu'elles sont décrites par la norme ISO-19125. Enfin, une troisième section évoque brièvement comment le modèle pourrait être accéléré par l'ajout d'un module d'indexation spatiale.

2.2.2 Limites des solutions existantes

Afin d'évaluer les capacités des systèmes de gestion des données spatiales massives reposant actuellement sur Spark à gérer entièrement la composante spatiale 2D, la norme ISO-19125 peut utilement être employée en tant que référentiel. En effet, les deux volets de celle-ci décrivent respectivement les types de géométries 2D (volet ISO-19125-1) et les fonctions SQL que doit rendre accessible un système permettant la manipulation de ces données à l'aide de requêtes SQL (volet ISO-19125-2) pour être considéré comme complet et facilement compatible avec d'autres systèmes. Dans cette optique, il sera d'abord question des types de géométries couverts par les différents systèmes. Puis nous verrons à quelles fonctions spatiales ils donnent accès et s'il serait possible de les étendre à d'autres.

2.2.2.1 Limites quant aux types de géométries 2D couverts

Un système respectant la norme ISO-19125-1 est censé gérer les sept principaux types de géométries 2D constructibles par interpolation linéaire. Ceux-ci peuvent être répartis en trois types simples (le point, la polyligne et le polygone) et en quatre types composés (le multipoint, la multi-polyligne, le multipolygone et la collection de géométries). Voici une étude de la façon dont les différents systèmes étudiés rencontrent cette norme.

Spatial Spark et GeoSpark intègrent tous ces types de géométries, car leur modèle repose sur l'emploi de la bibliothèque JTS [10] qui a été conçue de façon à rencontrer les standards de l'ISO et les recommandations de l'OGC [26]. Geomesa permet aussi de gérer toutes les géométries dans sa version actuelle [103], tandis que Simba ne gère que le point.

Le cas du système Magellan d'HortonWorks est plus mitigé et va être détaillé. Il ne permet en effet de traiter que le point, la polyligne et le polygone. Cela peut sembler suffisant si on suppose, comme le fait un des concepteurs du système [110], que les géométries composées sont réductibles à des tableaux de géométries. Mais en réalité, une telle approche ne peut guère mener qu'à un système dysfonctionnel. En effet, n'étant pas explicitement des géométries, de tels tableaux ne sont pas

10. https://github.com/locationtech/jts

recevables comme opérandes d'une fonction spatiale et leur renvoi comme résultat d'une opération spatiale comme l'union de polygones disjoints provoque une erreur de type.

Outre les choix conceptuels des développeurs, les limitations de Magellan sont également dues à l'emploi de la bibliothèque ESRI Tools en tant que moteur de traitement des opérations spatiales. Celle-ci ne permet en effet pas de traiter toutes les géométries 2D définis par la norme ISO-19125. Il y manque ainsi le type collection de géométries, tandis le type multi-polygone n'est que partiellement implémenté. De plus, le code WKT [11] produit par ESRI Tools est rarement identique à ce que les normes de l'ISO et les recommandations de l'OGC permettraient d'espérer.

Les incomplétudes des différentes solutions étudiées par rapport aux exigences de la norme ISO-19125-1 sont résumées dans le tableau 2.1. Celles d'ESRI Tools y ont été ajoutées afin de donner une idée des limites qu'il impose à l'évolution de Magellan.

Système	Point	Polyligne	Polygone	Multipoint	Multi-polyligne	Multipolygone	Collection
GeoSpark	Oui	Oui	Oui	Oui	Oui	Oui	Oui
Spatial Spark	Oui	Oui	Oui	Oui	Oui	Oui	Oui
Simba	Oui	Non	Non	Non	Non	Non	Non
Geomesa	Oui	Oui	Oui	Oui	Oui	Oui	Oui
Magellan	Oui	Oui	Oui	Non	Non	Non	Non
ESRI	Oui	Oui	Oui	Oui	Oui	Partiellement	Non

TABLE 2.1 – Couverture des géométries 2D définies par ISO-19125-1

2.2.2.2 Limites générales quant aux fonctions spatiales couvertes

Alors que le volet ISO-19125-1 de la norme ISO-19125 décrit comme on vient de le voir les géométries 2D élémentaires, son volet ISO-19125-2 décrit les signatures des fonctions spatiales (qui se ventilent en relations, opérations, fonction métrique et méthodes) qu'un système de gestion de base de données spatiales devrait rendre accessibles depuis ses requêtes SQL. Par contre, elle laisse libre la façon dont ces méthodes sont implémentées, ce qui rend concevable de l'appliquer à un contexte de données massives. L'application de cette norme ISO-19125-2 suppose par contre à minima que le système auquel elle est appliquée permette des requêtes SQL et rende possible de leur associer des fonctions personnalisées SQL (UDF). Cette section explore la façon dont les cinq systèmes actuels rencontrent la norme, et indique s'il est possible de les y étendre le cas échéant.

Spatial Spark repose uniquement sur le noyau de Spark et n'est interrogeable que par des paramètres passés en ligne de commande. Il manipule directement les type RDD de cette version de base dans ses traitements, si bien que les entourer de DataFrames de façon à les rendre gérables par Spark SQL ne serait possible qu'au prix d'une révision complète de son implémentation. L'application de la norme ISO-19125-2 à Spatial Spark semble donc rendue difficile par les choix qui ont guidé son implémentation, même si elle ne semble pas interdite par son modèle.

11. WKT est un acronyme de « well-know text ». Il désigne un langage défini par l'OGC afin de représenter les géométries dans un format textuel.

GeoSpark, quant à lui, étend comme on l'a vu le type RDD de Spark et n'est de ce fait pas directement compatible avec Spark SQL. Un de ses concepteurs indique cependant que l'intégration de ce point est prévue dans une version ultérieure du système, et qu'il existerait en attendant une façon indirecte de changer ces RDD en DataFrames [126]. Mais il ne détaille pas la procédure générale à employer à cette fin ni la façon d'y appliquer ensuite des requêtes SQL. Le modèle actuel de GeoSpark ne semble donc pas compatible avec la norme ISO-19125-2, qui implique de pouvoir interroger tous les types de géométries à l'aide de requêtes SQL.

Simba, en ce qui le concerne, repose sur sa propre adaptation de Spark SQL, ce qui semble permettre la requêtage SQL et la création d'UDF. Dans la pratique cependant, son analyseur syntaxique présente de telles incohérences qu'il ne permet pas de rendre la seule géométrie qu'il gère accessible à de telles fonctions. Il est par exemple obligatoire d'écrire « IN » à la suite de l'expression « POINT(x, y) » même quand aucun test d'inclusion n'est envisagé pour le point déclaré. Ces éléments ne font pas de Simba une solution mature et pérenne, et rend impossible d'y intégrer la norme ISO-19125-2.

L'extension Spark de Geomesa, pour sa part, ne reposait jusque-là que sur la version de base de Spark mais connait récemment une tentative d'intégration à Spark SQL. L'extension reste cependant bridée en amont par l'emploi du format CQL et de la base de données Accumulo qu'impose le cœur de Geomesa [103], si bien qu'elle ne semble pas à même de permettre une application d'ISO-19125-2 autonome et agnostique par rapport au format de données à traiter.

Magellan, enfin, ne permet pas en tant que tel le requêtage SQL, mais définit des types de données personnalisés (UDT) pour les trois types de géométries qu'il gère. Il serait donc tentant de considérer que l'ajout de fonctions SQL personnalisées (UDF) à son modèle suffirait à ce qu'il rencontre la norme ISO-19215-2.

En résumé, dans leur version actuelle aucun des systèmes étudiés n'intègre la norme ISO-19125-2, mais Magellan semble le seul à même d'y être adapté. Les sections suivantes vérifient donc s'il est possible d'étendre celui-ci aux relations spatiales, aux opérations spatiales, à la fonction métrique et aux méthodes spatiales.

2.2.2.3 Limites quant aux relations spatiales couvertes

L'ajout de fonctions SQL personnalisées à Magellan ne permet de gérer que les deux tiers des relations spatiales décrites par ISO-19125-2 : il ne prend en charge ni ST_Crosses, ni ST_Overlaps, ni ST_Relate. Cette limitation est uniquement due à son implémentation, puisque la bibliothèque ESRI Tools sur laquelle il repose permet ces opérations. La situation est résumée dans le tableau 2.2.

ST_... :	**Equals**	**Intersects**	**Disjoint**	**Within**	**Contains**	**Touches**	**Crosses**	**Overlaps**	**Relate**
Magellan	Oui	Oui	Oui	Oui	Oui	Oui	Non	Non	Non
ESRI	Oui	Oui	Oui	Oui	Oui	Oui	Oui	Oui	Oui

TABLE 2.2 – Couverture des relations spatiales ISO-19125-2

2.2.2.4 Limites quant aux opérations spatiales couvertes

L'ajout de fonctions SQL personnalisées à Magellan ne permet de gérer que la moitié des opérations spatiales décrites par ISO-19125-2. Il n'implémente en effet ni ST_Union, ni ST_SymDifference. Ses implémentations de ST_Intersection et ST_Difference, quant à elles, ne fonctionnent que lorsque leur résultat n'est pas un multipolygone ou une collection de géométries. Ces limitations sont en partie dues à l'implémentation de Magellan, puisque la bibliothèque spatiale ESRI Tools sur laquelle il repose permet de gérer ST_Union et ST_Difference. Cependant, il arrive que même ESRI Tools seule renvoie des résultats inexacts lorsque le résultat est une géométrie composée. La situation est résumée dans le tableau 2.3.

	ST_Intersection ST_Difference	**ST_Union ST_SymDifference**	**ST_ConvexHull ST_Buffer**	**ST_Transform**
Magellan	Partiellement	Non	Oui	Non
ESRI	Partiellement	Partiellement	Oui	Non

TABLE 2.3 – Couverture des opérations spatiales ISO-19125-2

2.2.2.5 Limites quant à la fonction métrique couverte

L'ajout d'une fonction SQL personnalisée à Magellan ne permet pas de gérer la fonction métrique de distance décrite par ISO-19125-2. Cette limitation est uniquement due à son implémentation, puisqu'ESRI Tools permet de la gérer. La situation est résumée dans le tableau 2.4.

	ST_Distance
Magellan	Non
ESRI	Oui

TABLE 2.4 – Couverture de la fonction métrique ISO-19125-2

2.2.2.6 Limites quant aux méthodes spatiales couvertes

L'ajout de fonctions SQL personnalisées à Magellan ne permet de gérer qu'une petite partie des méthodes décrite par ISO-19125-2 : deux méthodes communes à toutes les géométries (ST_GeometryType et ST_IsEmpty) ainsi que les deux méthodes spécifiques au point (ST_X et ST_Y). Ces limitations sont en partie dues à l'implémentation de Magellan, puisqu'ESRI Tools gère environ la moitié des méthodes décrites par la norme. La situation est résumée dans les tableaux 2.5 à 2.11.

	ST_Dimension **ST_Boundary**	**ST_GeometryType** **ST_IsEmpty**	**ST_AsText**	**ST_AsBinary** **ST_Envelope** **ST_SRID** **ST_IsSimple** **ST_Centroid**
Magellan	Non	Oui	Non	Non
ESRI	Oui	Oui	Partiellement	Non

TABLE 2.5 – Couverture des méthodes ISO-19125-2 comme à toutes les géométries

	ST_X	**ST_Y**
Magellan	Oui	Oui
ESRI	Oui	Oui

TABLE 2.6 – Couverture des méthodes ISO-19125-2 pour le point

	ST_StartPoint	**ST_IsClosed**	**ST_Length**	**ST_NumPoints**
	ST_EndPoint	**ST_IsRing**		**ST_PointN**
Magellan	Non	Non	Non	Non
ESRI	Oui	Oui	Oui	Oui

TABLE 2.7 – Couverture des méthodes ISO-19125-2 pour la polyligne

	ST_PointOnSurface	**ST_ExteriorRing**	**ST_NumInteriorRings**
		ST_Area	**ST_InteriorRingN**
Magellan	Non	Non	Non
ESRI	Non	Oui	Oui

TABLE 2.8 – Couverture des méthodes ISO-19125-2 pour le polygone

	ST_NumGeometries	**ST_GeometryN**
Magellan	Non	Non
ESRI	Non	Non

TABLE 2.9 – Couverture des méthodes ISO-19125-2 pour toute géométrie composée

	ST_IsClosed	**ST_Length**
Magellan	Non	Non
ESRI	Oui	Oui

TABLE 2.10 – Couverture des méthodes ISO-19125-2 pour la multi-polyligne

	ST_PointOnSurface	ST_Area
Magellan	Non	Non
ESRI	Non	Non

TABLE 2.11 – Couverture des méthodes ISO-19125-2 pour le multipolygone

Il apparaît donc que Magellan, qui semblait parmi les systèmes étudiés le seul à pouvoir être étendu à la norme ISO-19125-2, présente en fait des limitations qui empêchent de réaliser pleinement cette extension. Ces limites sont dues à la fois à des erreurs d'implémentation et au choix de la bibliothèque ESRI Tools comme support, car celle-ci ne permet de rencontrer que partiellement la norme ISO-19125-2.

2.2.2.7 Synthèse des limitations

Le tableau 2.12 présente une synthèse des principales limitations des systèmes étudiés, telles qu'évoquées dans les sous-sections précédentes. Il commence par rappeler leurs limitations les plus gênantes. Puis il rappelle les types de géométries qu'ils permettent de gérer et par voie de conséquence leur adéquation avec la norme ISO-19125. Il indique ensuite s'ils gèrent le SQL et s'ils rencontrent la norme ISO-19125-2.

	Magellan	Spatial Spark	GeoSpark	Simba	Geomesa
Principale limitation	Repose une bibliothèque spatiale limitée	Difficilement extensible au SQL	Non extensible au SQL dans son modèle actuel	Syntaxiquement bogué et non extensible	Nécessite une base NoSQL
Types de géométries	Simples seulement	Toutes	Toutes	Point	Toutes
Respect d'ISO-19125-1	Partiel	Oui	Oui	Très partiel	Oui
Gestion du SQL	Non, mais y est extensible	Non	Non	Oui, remplace Spark SQL	Oui, bridé par CQL
Respect d'ISO-19125-2	Non, mais peut être approché	Non	Non	Non	Non

TABLE 2.12 – Limitations des systèmes spatiaux Spark actuels

La section suivante présente le modèle d'un nouveau système de gestion des données massives spatiales sous Spark : Elcano, qui vise à surmonter les limitations présentées ici.

2.2.3 Présentation du système Elcano

Cette section indique les objectifs qui ont guidé la conception du système Elcano, présente et justifie le modèle puis donne un aperçu de son implémentation.

2.2.3.1 Objectifs de conception

L'objectif principal guidant la conception d'Elcano est qu'il constitue un système de gestion de données spatiales vectorielles qui s'abstrait des systèmes étudiés jusqu'ici. Il doit donc intégrer tous les types de géométries 2D élémentaires définis par la norme ISO-19125-1. Il doit aussi permettre d'accéder aux fonctions spatiales associées, afin d'améliorer l'analyse des phénomènes géolocalisés que concernent les données massives qu'il traite. Toutes les relations, opérations et méthodes spatiales définies par la norme ISO-19125-2 doivent donc être rendues disponibles par Elcano. Par exemple, l'appel à la fonction SQL ST_Intersects doit indiquer si deux géométries quelconques sont en intersection et l'appel à la fonction ST_Union doit renvoyer leur union. Le système doit également permettre de charger des données spatiales d'une façon simple et générique, afin d'être facile à alimenter et à étendre vers d'autres formats. Il doit aussi permettre la persistance des données en mémoire dans un format compact, de façon à les traiter plus rapidement et plus économiquement. Enfin, il doit être extensible au besoin à d'autres géométries que celles définies par la norme ISO-19125 [12].

Le modèle visant à rencontrer ces objectifs est présenté et justifié ci-dessous.

2.2.3.2 Architecture

La figure 2.1 présente le modèle d'Elcano.

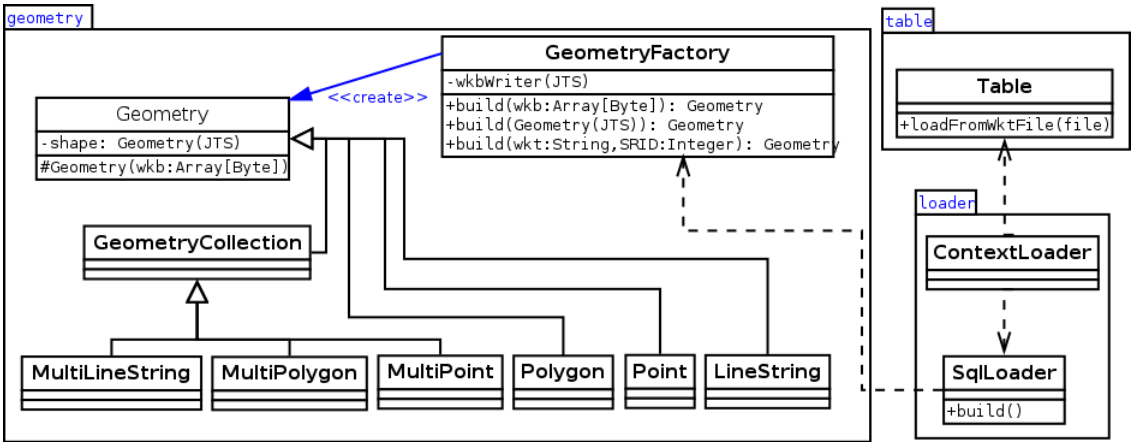

FIGURE 2.1 – Modèle d'Elcano

Dans ce modèle, les classes du paquetage « geometry » permettent de rencontrer l'objectif visant à intégrer les géométries élémentaires et les fonctions spatiales liées à Elcano. Le paquetage « loader » permet quant à lui de leur associer des fonctions spatiales SQL disponibles au requêtage. Le chargement générique des données à traiter et leur persistance en mémoire sont pour leur part gérés par la classe « Table » avec le support des méthodes de conversion de la classe GeometryFactory.

La façon dont sont atteints ces différents objectifs est décrite et justifiée ci-dessous.

12. Une des extensions les plus évidentes consisterait à gérer les géométries 2.5D, afin d'intégrer la notion d'élévation.

Gestion de géométries 2D et de leurs fonctions spatiales Le paquetage « geometry » d'Elcano contient une classe concrète par géométrie élémentaire 2D spécifiée par la norme ISO-19125. Il repose sur la bibliothèque spatiale JTS, car celle-ci a été spécifiquement conçue de façon à respecter les normes ISO (y compris ISO-19125-1) et les recommandations de l'OGC [26]. Ce choix permet donc d'éviter qu'Elcano rencontre les mêmes aléas que Magellan, dont les possibilités d'extension sont bridées par l'emploi d'une bibliothèque spatiale moins adéquate. Le système aurait pu employer directement les classes de JTS, comme le font Spatial Spark et GeoSpark, mais à des fins d'optimisation il semblait intéressant de ne pas être contraint par l'implémentation de la bibliothèque spatiale choisie. A cette fin, le paquetage « geometry » d'Elcano se constitue d'une hiérarchie de classes indépendante de JTS, qui l'entourent et lui appliquent le pattern de conception « proxy » [36]. Comme abordé plus en détail dans la section dédiée à l'implémentation, ce choix de conception a permis de rendre certaines opérations plus rapides dans Elcano que dans JTS et d'y ajouter des méthodes inaccessibles depuis JTS.

Gestion de fonctions spatiales SQL Il restait à rendre les méthodes spatiales fournies par la bibliothèque choisie disponibles sous la forme de fonctions SQL aux signatures adéquates. Cela est géré par la classe SqlLoader du paquetage « loader » d'Elcano, dont la méthode build() est initialisée au démarrage de l'application. Cette méthode déclare en effet des fonctions SQL personnaliées (UDF) qui font appel aux méthodes spatiales fournies par le paquetage « geometry » d'Elcano pour rencontrer la norme visée.

Chargement d'objets spatiaux Afin de pouvoir traiter les données de façon générique, Elcano reçoit les géométries de celles-ci au format WKT, un format décrit par la norme ISO-19125 qui permet de représenter les géométries sous une forme textuelle plutôt concise. Il permet au système d'acquérir selon une même procédure tous les types des géométries qui utilisent ce format répandu et de les rendre consultables à l'aide de requêtes SQL. Elcano permet ainsi de charger un jeu de données tabulaire (employant par exemple le format CSV [13]) sous la forme d'une table SQL temporaire à l'aide de sa classe Table. La gestion de formats plus spécifiques comme JSON [14] ou Parquet [119] pourrait aussi facilement être ajoutée au système par héritage de la classe Table.

Persistance d'objets spatiaux Afin d'occuper encore moins d'espace en mémoire, les données spatiales reçues sous forme textuelle (WKT) sont sérialisées dans le format WKB étendu [15] lors de leur chargement par la classe Table. Ce format binaire défini par l'OGC permet en effet de stocker sous une forme très compacte une géométrie ainsi que l'identifiant de la projection spatiale qu'elle emploie. Afin de fournir une interface unique de chargement et d'export de toutes les géométries vers WKB étendu et WKT, le pattern de conception « fabrique abstraite » [118] est appliqué via la classe

13. CSV est l'acronyme de « Comma-separated values ». Il désigne un format de fichier dans lequel les données sont présentées sous forme de listes séparées par des virgules.
14. Acronyme de « JavaScript Object Notation », qui désigne un format léger d'échange de données [12].
15. WKB est un acronyme de « Well-know binary ».

GeometryFactory d'Elcano, qui permet de produire toute classe héritant de Geometry à partir de ces formats.

Extensibilité quant aux types spatiaux traités En centralisant la création des géométries, le pattern de conception « fabrique abstraite » [118] rend aussi Elcano extensible par héritage de sa classe Geometry à d'autres types de géométries que celles définies par la norme ISO-19215.

La sous-section suivante donne quelques éléments de l'implémentation de ce modèle.

2.2.3.3 Éléments d'implémentation

Cette section traite quelques points de détail sur l'implémentation d'Elcano, en complément à la description de son modèle.

Accélération des fonctions spatiales dans Elcano Afin d'accélérer le fonctionnement de certaines fonctions spatiales dans Elcano par rapport à la façon dont elles sont implémentées dans JTS, les classes JTS des géométries ne sont créées que si nécessaire. C'est plutôt leur code WKB étendu qui est utilisé lorsqu'il suffit à appliquer la fonction spatiale visée. Par exemple, le type de géométrie est directement extrait d'un octet du code WKB et l'intersection de deux objets n'est testée via JTS que si leurs formes binaires ne sont pas identiques. Comme le format WKB est directement géré par Spark SQL, il a également permis d'éviter d'employer des types définis personnalisés (UDT) pour les rendre consultables en SQL. Là aussi, un gain d'efficacité en résulte, car ces types se sont avérés un facteur de ralentissement lors de nos tests et ne sont pas gérables de la même façon depuis toutes les versions de Spark. Cette approche s'inspire de celle employée par le système de gestion des données spatiales PostGIS [94].

Exemple de chargement d'objets spatiaux Afin d'illustrer la simplicité et la généricité de cette procédure, la figure 2.2 présente la requête de chargement d'une table CSV contenant des données spatiales depuis Elcano. Elle commence par construire un objet de type Table en lui passant en paramètres le nom de table à créer dans Spark SQL, le nom de la colonne contenant un identifiant unique, le nom de la colonne contenant une géométrie, et un SRID. Ensuite, la méthode loadFromWktFile() est appelée afin de charger cette table depuis un fichier CSV doté d'un header. Suite à ce chargement, les données sont placées dans des RDD Spark afin de pouvoir être traitées de façon distribuée par la mémoire vive du cluster, puis elles sont entourées d'un DataFrame qui permet de les consulter à l'aide de requêtes SQL.

```
import org.elcano.table.Table

val t1 = new Table("province", "gid", "the_geom",  4326)
t1.loadFromWktFile("hdfs://regard-cluster.scg.ulaval.ca:8020/labo/province.csv")
```

FIGURE 2.2 – Chargement d'une table CSV dans Spark SQL à l'aide d'Elcano, en Scala.

Création de fonctions spatiales SQL La figure 2.3 fournit un exemple de déclaration des fonctions SQL (UDF) associées à des relations spatiales, telles qu'elles sont déclarées au démarrage d'Elcano par sa classe SQLLoader.

```
udf.register("ST_Equals", (g1: Array[Byte], g2: Array[Byte]) => build(g1).equals(build(g2)))
udf.register("ST_Disjoint", (g1: Array[Byte], g2: Array[Byte]) => build(g1).disjoint(build(g2)))
udf.register("ST_Intersects", (g1: Array[Byte], g2: Array[Byte]) => build(g1).intersects(build(g2)))
udf.register("ST_Crosses", (g1: Array[Byte], g2: Array[Byte]) => build(g1).crosses(build(g2)))
udf.register("ST_Within", (g1: Array[Byte], g2: Array[Byte]) => build(g1).within(build(g2)))
udf.register("ST_Contains", (g1: Array[Byte], g2: Array[Byte]) => build(g1).contains(build(g2)))
udf.register("ST_Overlaps", (g1: Array[Byte], g2: Array[Byte]) => build(g1).overlaps(build(g2)))
udf.register("ST_Touches", (g1: Array[Byte], g2: Array[Byte]) => build(g1).touches(build(g2)))
udf.register("ST_Relate", (g1: Array[Byte], g2: Array[Byte], expr: String) => build(g1).relate(build(g2), expr))
```

FIGURE 2.3 – Exemple d'implémentation des relations spatiales dans Elcano, en Scala.

Des tests unitaires systématiques vérifient que ces fonctions sont définies selon une dénomination identique à celle documentée par ce standard.

Gestion du type GeometryCollection depuis JTS Lors du développement d'Elcano, le respect de la norme a pu être atteint par encapsulation des méthodes de JTS pour la plupart des géométries. Mais il s'est avéré plus délicat pour la classe GeometryCollection, car aucune des versions de JTS testées (jusqu'à 1.14) n'autorise l'utilisation de la plupart des fonctions spatiales pour le type « collection de géométries », bien qu'elles soient implémentées en interne par JTS. D'après Martin Davis [25], cette désactivation est un choix implémentatif délibéré visant à limiter l'usage du type GeometryCollection, car c'est la seule géométrie à pouvoir contenir des collections de géométries simples superposées. Afin de respecter au plus proche la norme ISO-19125-2, nous avons réactivé ces fonctionnalités de JTS dans Elcano, car le risque d'incohérence induit semblait trop faible et localisé pour renoncer à celles-ci. Parmi ces méthodes, seule ST_Boundary manquait tout à fait dans JTS pour ce type de géométrie, si bien qu'elle a été complètement implémentée dans Elcano.

Déploiement et compatibilité Elcano peut être ajouté en tant que jeu de classes comme en tant qu'extension autonome à n'importe quelle version de Spark à partir de la 1.4.

A présent que les objectifs et le modèle d'Elcano ont été présentés et complétés par quelques précisions sur son implémentation, la section suivante étudie la possibilité future d'y ajouter un système d'indexation spatiale en vue d'augmenter sa rapidité de traitement.

2.2.4 Vers une gestion efficiente de l'indexation spatiale

Malgré les apports conceptuels d'Elcano par rapport aux solutions existantes, qui en fait le premier système de gestion des données spatiales massives vectorielles à intégrer tous les aspects de la norme ISO-19125 à Spark, celui-ci manque dans son modèle actuel d'un système à même d'optimiser la vitesse de traitement des données spatiales. Autrement dit, il ne permet pas de gérer l'indexation spatiale des données à traiter. Celle-ci se définit en effet comme une réorganisation des données spatiales,

notamment en exploitant leurs relations de proximité, en vue d'accélérer les traitements qui leur sont appliqués [41].

Parmi les systèmes étudiés, quatre proposent par contre des modules d'indexation spatiale, mais aucune qui serait à la fois efficace et extensible. L'indexation proposée par Spatial Spark hérite ainsi directement de JTS, alors que celui-ci n'a pas été conçu pour gérer des donnés massives réparties dans un environnement distribué. GeoSpark propose une gestion de l'indexation spatiale plus intégrée et performante [126], mais au prix d'une extension du type RDD et donc d'un blocage de son intégration à Spark SQL. L'indexation proposée par Simba serait plus performante [121], mais est actuellement bridée par les incohérences de son analyseur syntaxique. Elle impose de plus un remplacement de Spark SQL par Simba pour fonctionner, ce qui la rend peu évolutive. Geomesa, enfin, se limite aux indexations permises par Accumulo, qu'il utilise pour stocker les données spatiales à traiter. Il en résulte de moins bonnes performances que pour toutes les autres solutions disposant d'une indexation spatiale [121].

En 2015, Cortés et al. [20] citent par ailleurs un défaut récurrent des solutions d'indexation des données massives actuelles : l'architecture maître-esclave selon laquelle les traitements sont répartis dans de l'écosystème Hadoop leur impose au moins conceptuellement l'emploi d'un index centralisé, ce qui réduirait leur scalabilité dans un environnement multi-serveurs. Pour y remédier, ils ont défini et testé un mode d'indexation décentralisé pour les données à références spatiales, qui fonctionne même lorsqu'elles sont reçues sous forme de streaming. Leur solution, appelée BIG-LHT, ajoute un système d'indexation temporel à l'algorithme GeoHash défini par Niemeyer [91].

Malgré son côté novateur, l'étude de Cortés et al. se détourne sans doute trop rapidement d'Hadoop et de Spark, qu'elle remplace par sa propre solution basée sur FreePastry. Il importerait donc de déterminer si un système de traitement des données massives à référence spatiale gérant efficacement l'indexation mais ne renonçant pas à la richesse de l'environnement Hadoop et aux possibilités de Spark peut être défini.

2.2.5 Conclusion

En conclusion, il est étonnant de constater à quel point les systèmes gérant les données massives à référence spatiale dans l'environnement Hadoop sont peu nombreuses et limités. Le potentiel scientifique, économique et sociétal lié au traitement de telles données est pourtant immense, et les qualités d'Hadoop en tant que standard de fait pour leur traitement n'est plus à démontrer. Malgré cela, il n'existe que cinq solutions gérant les données spatiales à partir de Spark, alors que celui-ci est peut-être le module d'Hadoop le plus prometteur pour ce type de traitements.

Chacun de ces systèmes présente des limitations quant aux types de géométries 2D et aux fonctions spatiales qu'ils peuvent gérer. Après un inventaire de ceux-ci, Magellan est apparu comme le seul à même d'être étendu à la norme ISO-19125-2, mais son modèle empêche dans les faits de l'y adapter entièrement. Pour y remédier, cet article propose un nouveau système de gestion des données spatiales

massives sous Spark, Elcano. Celui-ci intègre au module SQL de Spark la bibliothèque de traitements spatiaux JTS, choisie en raison de sa conformité avec les standards de l'ISO. Cette approche permet de rendre accessibles toutes les fonctions SQL définies par la norme ISO-19125-2 depuis Elcano, mais ajouter un système d'indexation spatiale efficace à celui-ci reste un problème ouvert. Une façon d'y parvenir serait peut-être d'adapter la solution définie par Cortés et al. [20] à l'environnement Spark, mais d'autres approches sont à considérer en s'inspirant des méthodes d'indexation spatiale usuelles dans le domaine géospatial. Ceci fera l'objet de nos travaux futurs.

2.2.5.1 Références

[2] A. AJI et al. “Hadoop GIS : a high performance spatial data warehousing system over mapreduce”. In : *Proceedings of the VLDB Endowment* 6.11 (2013), p. 1009–1020.

[6] M. ARMBRUST et al. “Spark sql : Relational data processing in spark”. In : *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 2015, p. 1383–1394.

[8] T. BADARD. “Mettre le Big Data sur la carte : défis et avenues relatifs à l’exploitation de la localisation”. In : *Colloque ITIS - Big Data et Open Data au coeur de la ville intelligente*. (29 avr. 2014). Québec : Centre de recherche en géomatique, 2014.

[12] T. BRAY. *The javascript object notation (json) data interchange format (No. RFC 7158)*. 2014.

[20] R. CORTÉS et al. “A Scalable Architecture for Spatio-Temporal Range Queries over Big Location Data”. In : *Network Computing and Applications (NCA), 2015 IEEE 14th International Symposium on*. 2015, p. 159–166.

[25] M. DAVIS. *JTS Topology Suite’s Forum*. https://sourceforge.net/p/jts-topo-suite/mailman/message/27654158/. 2011.

[26] M. DAVIS et J. AQUINO. *Jts topology suite technical specifications*. 2003.

[28] J. DEAN et S. GHEMAWAT. “MapReduce : simplified data processing on large clusters”. In : *Communications of the ACM* 51.1 (2008), p. 107–113.

[36] Gamma E. et al. *Design Patterns : Elements of Reusable Object-Oriented Software*. 1994.

[40] A. ELDAWY et M. F. MOKBEL. “Pigeon : A spatial mapreduce language”. In : *Data Engineering, 2014 30th International Conference on IEEE*. 2014, p. 1242–1245.

[41] A. ELDAWY et Mohamed F MOKBEL. “The Era of Big Spatial Data : A Survey”. In : *Information and Media Technologies* 10.2 (2015), p. 305–316.

[43] J. ENGÉLINUS et T. BADARD. “Elcano : un système de gestion des données massives à composante spatiale basé sur Spark SQL », Revue internationale de géomatique”. In : *Revue internationale de géomatique (en cours de soumission)*. 2016.

[46] M. R. EVANS et al. “Spatial big data”. In : *Big Data : Techniques and Technologies in Geoinformatics* (2014), p. 149.

[48] S. FERMIGIER. *Big data et open source : une convergence inévitable ?* 2011. URL : http://projet-plume.org.

[51] C. FRANKLIN et P. HANE. “An Introduction to Geographic Information Systems : Linking Maps to Databases [and] Maps for the Rest of Us : Affordable and Fun.” In : *Database* 15.2 (1992), p. 12–15.

[59] L. GU et H. LI. “Memory or time : Performance evaluation for iterative operation on hadoop and spark”. In : *High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 10th International Conference on IEEE*. 2013, p. 721–727.

[67] J. N. HUGHES et al. “GeoMesa : a distributed architecture for spatio-temporal fusion”. In : *SPIE Defense + Security*. International Society for Optics et Photonics. 2015, 94730F.

[84] J. MANYIKA et al. “Big data : The next frontier for innovation, competition, and productivity”. In : *The McKinsey Global Institute* (2011).

[89] B. MERICSKAY et S. ROCHE. “Cartographie numérique en ligne nouvelle génération : impacts de la néogéographie et de l’information géographique volontaire sur la gestion urbaine participative”. In : *Nouvelles cartographie, nouvelles villes, HyperUrbain* (2010).

[91] G. NIEMEYER. *Geohash*. 2008.

[94] R. O. OBE et L. S. HSU. *PostGIS in action*. Manning Publications Co., 2015.

[103] Commonwealth Computer RESEARCH. *Apache Spark Analysis*. http://www.geomesa.org/documentation/tutorials/spark.html. Accessed : 2017-02-26.

[104] Ram S. *Magellan : Geospatial Analytics on Spark*. 2015. URL : http://hortonworks.com/blog/magellan-geospatial-analytics-in-spark/.

[110] R. SRIHARSHA. *Magellan’s Github - issue 30*. https://github.com/harsha2010/magellan/issues/30. Accessed : 2016-05-05.

[118] J. VLISSIDES et al. “Design patterns : Elements of reusable object-oriented software”. In : *Reading : Addison-Wesley* 49.120 (1995), p. 11.

[119] D. VOHRA. “Apache Parquet”. In : *Practical Hadoop Ecosystem*. Springer, 2016, p. 325–335.

[120] T. WHITE. *Hadoop : The definitive guide*. O’Reilly Media, Inc., 2012.

[121] Dong XIE et al. *Simba : Efficient In-Memory Spatial Analytics*. URL : https://www.cs.utah.edu/~lifeifei/papers/simba.pdf.

[124] S. YOU, J. ZHANG et L. GRUENWALD. “Large-scale spatial join query processing in cloud”. In : *Data Engineering Workshops (ICDEW), 2015 31st IEEE International Conference on*. IEEE. 2015, p. 34–41.

[126] J. YU. *GeoSpark’s Github - issue 33*. https://github.com/DataSystemsLab/GeoSpark/issues/33. Accessed : 2017-02-26.

[127] J. YU, J. WU et M. SARWAT. “Geospark : A cluster computing framework for processing large-scale spatial data”. In : *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM. 2015, p. 70.

[128] M. ZAHARIA et al. “Spark : cluster computing with working sets”. In : *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*. T. 10. 2010, p. 10.

2.3 Compléments à l'article

2.3.1 Limites de la norme ISO-19125

Malgré son rôle fondamental, la norme ISO-19125 manque sur certains points d'exhaustivité. Ainsi, la transformation du système de coordonnées employé par une géométrie n'est pas décrite par ce standard, alors même que son ajout faciliterait l'intégration de données provenant de sources différentes. Une telle méthode est pourtant décrite par des normes plus générales comme ISO-19107 et ISO-13249-3 et employée couramment par des solutions comme PostGIS et pourrait tout à fait être implémentée à l'aide des classes de conversion fournies par JTS. Au vu de ces possibilités et de leur intérêt, l'ajout d'une méthode de transformation du système de coordonnées au modèle d'Elcano a été rendue effective dans un article ultérieur [4.2.4.3].

2.3.2 Évolution du mode de sérialisation d'Elcano

Au début de la conception du système Elcano, la version la plus récente de Spark était la 1.6. Or, la Fondation Apache a depuis rendu disponible Spark 2.x. Cette nouvelle branche constitue une évolution majeure, dans laquelle certaines fonctionnalités ont été abandonnées et d'autres réimplémentées. Malheureusement, il suit que la rétrocompatibilité avec les anciennes versions de Spark n'est pas systématique. Ainsi, la possibilité pour le développeur de gérer la persistance des données dans Spark SQL à l'aide de types de données personnalisés (UDT [16]) consultables depuis SQL a été retirée de la version 2.0 [17] et remplacée par une solution totalement différente dans les versions ultérieures.

Une telle rupture dans la continuité de Spark a obligé à repenser l'architecture d'Elcano, et en particulier la façon dont les géométries y sont persistées dans Spark SQL en vue de leur consultation depuis des requêtes SQL. Afin qu'un compte rendu subsiste de cette évolution, cette section détaille les caractéristiques de la version d'origine, ainsi que les avantages de celle qui l'a remplacée.

La première version d'Elcano gérait la sérialisation des données spatiales en vue de leur lecture depuis des requêtes SQL en implémentant un UDT [18] Spark spécifique appelé GeometryUDT. Celui-ci sérialisait les géométries dans le format binaire WKB [19], afin de réduire leur volume en mémoire lors de leur stockage par Spark SQL. Lorsqu'ensuite les géométries étaient lues en tant que résultat d'une requête SQL, l'UDT les désérialisait afin qu'elles redeviennent des géométries.

Afin que Spark applique automatiquement les méthodes de sérialisation et de désérialisation de GeometryUDT lors du traitement de données spatiales, cette classe héritait de la classe UserDefinedType de Spark SQL. Réciproquement, la classe Geometry était définie par annotation comme un type de données traitable par Spark SQL. Une telle mise en œuvre était rapide à implémenter, mais comportait

16. UDT est un acronyme de « User Defined Type ».
17. Ce retrait a été commenté par un concepteur de Spark le 23 août 2016 sur `http://stackoverflow.com/questions/39096268/how-to-use-user-defined-types-in-spark-2-0`.
18. UDT est un acronyme de « User Defined Type »
19. WKB est un acronyme de « Well-know binary ».

un revers : elle imposait de déclarer Geometry comme une classe concrète alors même que celle-ci n'était jamais instanciée. La figure 2.4 présente le modèle résultant de cet ancien mode de sérialisation, tandis que la figure 2.5 fournit un exemple d'implémentation en Scala de la classe GeometryUDT.

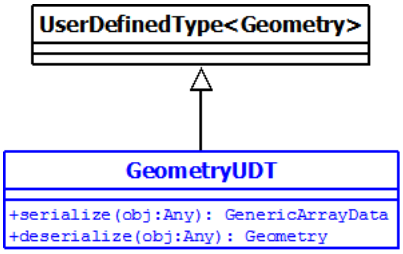

FIGURE 2.4 – Ancien mode de gestion de la persistance dans Elcano

```scala
import org.apache.spark.sql.catalyst.util._

private[elcano] class GeometryUDT extends UserDefinedType[Geometry] {

  override def sqlType: DataType = ArrayType(ByteType, false)
  override def userClass: Class[Geometry] = classOf[Geometry]

  override def serialize(obj: Any): GenericArrayData = {
    obj match {
      case p: Geometry =>
        val wkb = p.asBinary.
          map(x => x.asInstanceOf[Any])
        val row = new GenericArrayData(wkb)
        row
    }
  }

  override def deserialize(obj: Any): Geometry = {
    obj match {
      case values: ArrayData =>
        val wkb = values.toByteArray()
        GeometryFactory.build(wkb)
    }
  }
```

FIGURE 2.5 – Implémentation Scala de la classe GeometryUDT

Pour sa part, le nouveau mode de sérialisation d'Elcano n'a plus recours aux UDT. Ce renoncement lui permet de rester compatible aussi bien avec la récente branche 2.x de Spark qu'avec des versions plus anciennes. Il continue par contre d'utiliser le format WKB pour stocker les géométries dans Spark SQL sous forme binaire, mais ces WKB ne sont traduits en géométries que lorsque des fonctions spatiales SQL leur sont appliquées plutôt qu'à chaque lecture SQL. Le nouveau mode de sérialisation réduit donc le volume de mémoire vive et le temps processeur utilisés pour traiter des géométries, ce qui rend le système plus performant. Le modèle résultant est également amélioré : comme la classe

GeometryUDT en disparaît, la classe Geometry ne doit plus être indiquée comme un type de données traitable par cet UDT et peut devenir abstraite. Par contre, un effet secondaire du nouveau mode de sérialisation est l'impossibilité d'obtenir directement une géométrie comme retour d'une requête SQL. Cependant, cet aspect est facilement contourné par l'extraction du WKB puis sa conversion en géométrie via la « fabrique abstraite » GeometryFactory d'Elcano.

2.3.3 Autre application de la norme ISO-19125-2

Si Elcano constitue le premier moteur de traitement spatial sous Spark à respecter la norme ISO-19125-2, il existe un précédent dans l'environnement Hadoop. En effet You et al. ont conçu en 2015 ISP [122], un système reposant sur Impala qui permet l'intégration de données massives à partir de données spatiales. Les performances de la première version d'ISP (ISP-MC) sont décrites par ses concepteurs comme inférieures à celles de Spatial Spark [124]. Mais ce n'est plus vrai d'une version plus récente (ISP-GPU), qui semble dépasser Spatial Spark en vitesse d'exécution [125].

Cependant ISP-GPU est conçu spécifiquement pour être exécuté sur des processeurs multi-cœurs de type GPU [20], ce qui rend l'emploi et l'évaluation de cette solution problématique. Son efficacité est en effet conditionnée par l'emploi d'un type de matériel dont peu de clusters disposent et que les autres prototypes de gestion des données spatiales massives ne sont pas à même de gérer. En outre, puisque You et al. ont initialement constaté la supériorité de Spatial Spark sur ISP à processeurs égaux [124], il semble légitime de supposer qu'un extension spatiale de Spark adaptée aux GPU surclasserait ISP-GPU. Les limitations d'Impala détaillées dans la section [1.1.1.2] plaident aussi en ce sens.

2.4 Conclusion

Dans ce chapitre, à la suite d'une étude détaillée des limites des cinq systèmes de gestion des données massives vectorielles reposant actuellement sur Spark [2.2.2], le modèle d'un système plus complet a été proposé et justifié [2.2.3.2] : gestion de toutes les géométries 2D et des fonctions spatiales SQL associées, chargement générique des données et persistante de celles-ci. L'opportunité d'ajouter un module d'indexation spatiale efficace et extensible à Elcano est finalement évoquée [2.2.4].

En complément, le chapitre montre comment le manque de compatibilité entre les différentes versions de Spark a influencé le choix d'un mode de persistance qui n'emploie pas les types de données personnalisés (UDT) proposés par Spark 2.3.2. Il mentionne également le manque d'exhaustivité de la norme ISO-19125, à laquelle il manque par exemple une fonction permettant de transformer le système de référence spatiale d'une géométrie, alors qu'une telle méthode serait aussi simple à implémenter qu'utile [2.3.1]. Finalement, un autre système implémentant la norme ISO-19125-2 sous Hadoop est brièvement décrit [2.3.3], même s'il repose sur un environnement à priori moins efficient que Spark et semble surtout devoir ses performances à l'emploi d'un type de processeur spécifique.

20. GPU est un acronyme de « Graphics Processing Unit ». Il désigne un type de processeur utilisé généralement pour équiper les cartes graphiques, et qui permet des traitements massivement parallèles.

Le chapitre suivant porte plus loin la réflexion sur l'indexation spatiale entamée dans la section [2.2.4].

Chapitre 3

Indexation des données spatiales massives

3.1 Introduction

Ce chapitre vise à enrichir le moteur de traitement des données spatiales massives vectorielles Elcano, dont le modèle a été présenté dans le chapitre précédent [2.2.3.2]. Il s'agit ici plus spécifiquement de lui ajouter un module d'indexation spatiale rapide et efficace, de façon à ce qu'il permette un traitement plus rapide des données spatiales.

Le corps du chapitre est constitué d'un article, qui commence [3.2.1] par évoquer le manque d'un système de gestion des données massives gérant l'indexation spatiale [cf. 1.3] d'une façon à la fois efficace pour les données massives et compatible avec Hadoop et Spark [cf. 1.1.1.2], malgré les apports du modèle Elcano [cf. 2.2.3.2]. Dans une seconde partie [3.2.2], l'article décrit l'indexation dans le contexte de données ordinaires mais aussi dans celui de données massives, puis s'intéresse à la scalabilité horizontale de quelques types d'indexation [cf. 1.3.1]. Puis, une revue critique de différents systèmes gérant l'indexation spatiale des données massives est effectuée et se termine par une évocation des difficultés qu'implique le partitionnement [cf. 1.3.2] des données spatiales sous Spark SQL [3.2.3]. Un système gérant l'indexation spatiale de façon souple est ensuite défini [3.2.4.2] en fonction de ces éléments, en tant qu'extension d'Elcano. Une implémentation du nouveau système est finalement soumise à un banc de tests [3.2.5], de façon à mesurer sa scalabilité et à comparer ses performances à celles de Spatial Spark [cf. 1.3.4] et PostGIS [cf. 1.3.3].

3.1.1 Contributions

[45] Engélinus, J. et Badard, T., 2016, « Vers une indexation des données spatiales massives performante et efficace », Geomatica (en cours de soumission).

3.2 Corps de l'article

Titre : Vers une indexation des données spatiales massives performante et efficace

Auteurs : Jonathan Engélinus, Thierry Badard

Résumé : Les données massives se situent au cœur de beaucoup d'enjeux scientifiques et sociétaux, et leur volume global ne cesse de croître. Il devient donc crucial de disposer de solutions permettant leur traitement rapide et leur analyse efficace. Hélas, alors qu'une majorité de ces données intègrent une composante spatiale vectorielle, peu de systèmes sont à même de gérer cette dernière. En outre, les rares prototypes qui s'y essaient présentent des performances limitées en termes d'indexation spatiale. La présente recherche vise à déterminer comment améliorer cette situation. Une revue de littérature portant sur les types d'indexation spatiale et leur utilisation dans un contexte de données massives est donc entrepris dans un premier temps. Ensuite, l'intégration des solutions jugées les plus efficientes au système Elcano est décrite. Finalement, des tests sur la vitesse d'exécution de celui-ci démontrent qu'elle surpasse celles des solutions du marché.

Abstract: Big data are in the midst of many scientific and economic issues. Furthermore their volume is continuously increasing. As a result, the need for management and processing solutions has become critical. Unfortunately, while most of these data have a vectorial spatial component, almost none of the current systems are able to manage it. In addition, the few systems who try show poor performances. The aim of this research was then to determine how to improve this. We first endeavor to review the literature on the types of spatial indexation and their use in a context of massive data. In a second time, the most efficient solutions are implemented into the Elcano system. Finally, the tests show that the resulting system surpasses the current solutions of the market.

Mots-clés : indexation, geohash, R-Tree, scalabilité horizontale, Spark SQL

Keywords: indexation, geohash, R-Tree, horizontal scalability, Spark SQL

3.2.1 Introduction

La possibilité de traiter d'importants volumes de données informatiques est une nécessité de plus en plus présente dans le monde contemporain. Walmart traite ainsi quotidiennement plus de 2.4 petabytes (soit 2.4 * 10^{15} octets) de données, et Google dix fois plus [83]. Cette situation ira en s'intensifiant, puisque d'après IDC le volume total de données en circulation croît de 90% tous les deux ans à un rythme qui se maintiendra jusqu'en 2020 [52]. Souvent, ces données sont trop volumineuses pour être traitées par des bases de données classiques, et sont alors qualifiées de « données massives » [99].

L'écosystème Hadoop [120] constitue un standard de fait pour le traitement et la gestion de ces données massives. Son noyau implémente l'algorithme Map Reduce [28], qui permet de distribuer des traitements entre les serveurs d'un cluster puis d'en réunir les résultats de façon cohérente. Les données à traiter sont elles-mêmes réparties entre les serveurs par le système de fichiers distribués HDFS [1], qui est fourni nativement avec Hadoop. Il en résulte un haut degré de scalabilité horizontale, celle-ci pouvant être définie comme la possibilité d'augmenter linéairement les performances d'un système reposant sur plusieurs serveurs en fonction des besoins.

Malheureusement, Hadoop ne permet pas de gérer la localisation spatiale associée aux données massives, alors même que 80% des données d'entreprise en comportent une [51]. Une étude du cabinet McKinsey [84] indique pourtant qu'une meilleure utilisation de la localisation spatiale des données massives pourrait rapporter 100 milliards de dollars pour les fournisseurs de services et de l'ordre de 700 milliards pour les utilisateurs finaux.

Afin de commencer à répondre à ce besoin important, nous présentons dans une autre article [43] le modèle (reproduit en figure 3.1) d'un système de gestion des données spatiales massives : Elcano.

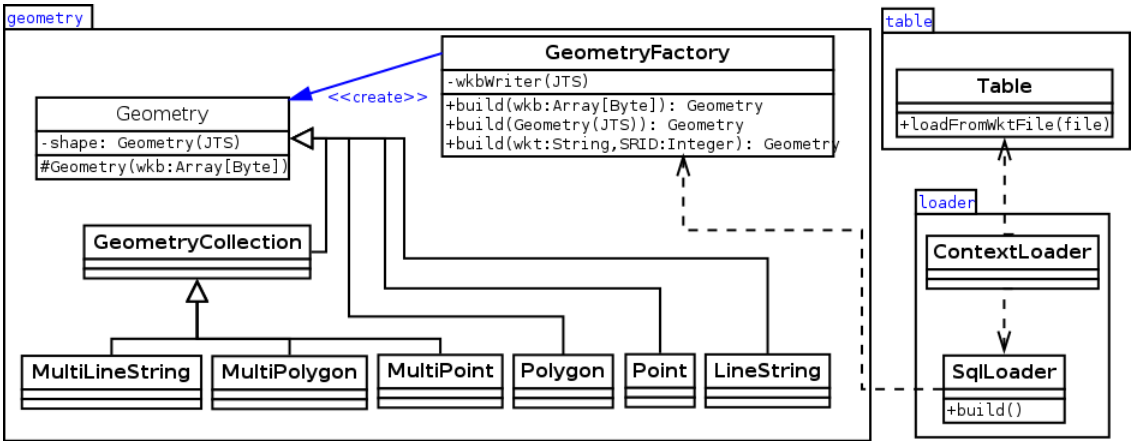

FIGURE 3.1 – Modèle d'Elcano étendu à l'indexation

Celui-ci a été conçu afin d'atteindre quatre objectifs : gérer tous les types de données spatiales 2D

1. HDFS est un acronyme de « Hadoop Distributed File System »

élémentaires (cf. package « geometry » du modèle, qui repose sur la bibliothèque spatiale JTS [2]), permettre de leur appliquer les fonctions spatiales décrites par la norme ISO-19125 à l'aide de requêtes SQL (cf. package loader du modèle, qui charge les fonctions au démarrage du système), gérer leur importation de façon générique et les persister efficacement (cf. classe Table, qui charge des données au format WKT [3] et les persiste au format WKB [4]).

Elcano repose sur Spark [5], un élément récent de l'écosystème Hadoop qui permet d'effectuer des traitements depuis la mémoire vive du cluster. Cette approche limite les accès disque et permet de traiter des données massives jusqu'à dix fois plus vite qu'une utilisation classique d'Hadoop [128]. Cinq autres systèmes de gestion des données spatiales reposant sur Spark existent [124, 126, 104, 121, 67], mais Elcano est le seul à permettre une gestion de tous les types de données spatiales massives 2D et des fonctions spatiales associées à l'aide de requêtes SQL d'une façon qui (grâce à son intégration du module Spark SQL de Spark [6, 43]) optimise stratégiquement les traitements exécutés, évite la création de tables inutiles et sérialise les données manipulées.

Elcano sous sa forme actuelle présente cependant une lacune importante. Il y manque un module gérant l'indexation spatiale des données, de façon à accélérer leur traitement. Parmi les autres systèmes de gestion des données spatiales reposant sur Spark, quatre comportent un module d'indexation spatiale mais sans parvenir jamais à un modèle à la fois efficace et compatible avec l'utilisation de requêtes SQL depuis Spark.

Une piste d'amélioration semble apportée par Cortés et al. [20], qui indiquent que les modules d'indexation spatiale au cœur des systèmes de gestion de données spatiales massives existants ont un défaut récurrent : ces solutions indexent les données de façon centralisée, ce qui nuit à leur scalabilité dans un environnement multi-serveurs. Pour y remédier, ces auteurs proposent un système de gestion des données spatiales massives appelé BIG-LHT, qui gère l'indexation spatiale de manière décentralisée et semble permettre une excellente scalabilité horizontale. Par exemple, la latence moyenne de chargement de données spatiales reçues passe d'environ 1.5 secondes pour un cluster de 6 serveur à 0.5 secondes pour un cluster de 15 serveurs [6], ce qui suggère par approximation linéaire une diminution du temps d'exécution d'environ 84% lorsque le nombre de machines est multiplié par 10.

Mais pour parvenir à un tel résultat, l'équipe de Cortés fait le choix radical de se détourner de l'écosystème Hadoop, classiquement utilisé pour gérer les données massives, pour concevoir plutôt une solution atypique qui repose sur le système de peer-to-peer FreePastry [7]. Ils justifient ce renoncement par le fait que les solutions reposant sur Hadoop restent tributaires de l'architecture maître-esclave uti-

2. https://github.com/locationtech/jts
3. WKT est un acronyme de « well-know text ». Il désigne un langage défini par l'OGC afin de représenter les géométries dans un format textuel.
4. WKB est un acronyme de « Well-know binary ». Ce format binaire défini par l'OGC permet de stocker sous une forme très compacte une géométrie ainsi que l'identifiant de la projection spatiale qu'elle emploie.
5. http://spark.apache.org/
6. Le débit lors de ce test était 100 000 données reçues par secondes, sur des machines équipées de 8 Go de RAM.
7. http://www.freepastry.org/

lisée par celui-ci pour distribuer les traitements, qui empêcherait de véritablement répartir l'indexation spatiale entre les serveurs. Mais la rigidité induite par la distribution centralisée des traitements par Hadoop n'est-elle pas compensée par sa maturité, la diversité de son environnement et l'évolutivité qu'il permet ?

Dans ce contexte, il semble intéressant de déterminer si une solution d'indexation spatiale performante mais compatible avec Hadoop peut aussi être définie. C'est l'objectif principal du présent article, qui vise à proposer une extension d'Elcano dotée d'un moteur d'indexation des données spatiales massives efficace et généraliste mais ne renonçant pas à la richesse de l'écosystème Hadoop.

La première section de cet article présente une revue de littérature de la notion d'indexation spatiale. Les limitations de quelques outils existants à même d'indexer des données spatiales sont ensuite évoquées. Une troisième partie décrit l'architecture de l'extension d'Elcano mise en place pour dépasser ces limitations. Finalement, les performances d'une implémentation de ce nouveau système sont comparées avec celles de solutions souvent utilisées sur le marché.

3.2.2 Indexation spatiale

De façon très générale, l'indexation de données peut s'entendre comme une transformation qui leur est appliquée de façon à faciliter leur accès en lecture et/ou en écriture ensuite. Lorsqu'il s'agit de données spatiales, des techniques particulières peuvent être employées, puisqu'il devient possible de connaître la position de chaque donnée ainsi que les relations de proximité qui les lient [41]. Les méthodes d'indexation spécifiques à la composante spatiale consistent donc souvent à exploiter cette interconnexion plus forte entre les données pour réduire le volume d'information à traiter lors des requêtes. Elles permettent de la sorte d'accélérer considérablement des opérations coûteuses (comme la jointure de données spatiales, qui nécessiterait sinon un produit cartésien entre les données).

Dans le contexte d'un système de traitement des données massives, l'indexation spatiale se heurte à une difficulté supplémentaire. Dans un tel système, les données sont en effet dispersées entre différentes machines, ce qui nuit à leur comparaison. Au lieu d'affronter cette fragmentation, une solution viable semble être d'y associer un partitionnement par zone des données. Plus des données sont proches géographiquement, et plus elles ont alors de chances d'être situées sur un même serveur. Cette approche, qui est celle retenue par Cortés et al. [20], s'avère cependant inapplicable directement depuis Spark SQL, comme expliqué plus loin.

3.2.2.1 Méthodes d'indexation spatiale

Eldawi et Mokbel [41] indiquent qu'il existe deux grandes catégories d'indexation spatiale : l'indexation plate et l'indexation hiérarchique. La première découpe simplement les données géographiques en secteurs, tandis que la seconde les classifie sous forme d'arbre. Cette sous-section décrit quelques méthodes d'indexation appartenant à chacune d'entre elles.

Parmi les méthodes d'indexation spatiale plane, Grid-File est peut-être la plus simple et la plus générique et sera d'abord abordée. Plus complexe, la méthode plane par diagramme Voronoï sera ensuite brièvement décrite. En ce qui concerne les méthodes d'indexation spatiale hiérarchique, Kothuri et al. [72] indiquent dans une revue de la question l'existence de Quadtree, R-tree, hB-tree, TV-tree, SS-tree et SR-tree. A des fins de concision, seules les deux premières seront détaillées ici, car SR-Tree, SS-Tree et TV-Tree sont de simples variantes de R-Tree [23] tandis que hB-tree semble surtout adaptée à un contexte multidimensionnel qui s'éloigne du cadre de notre recherche [78]. L'indexation plane à l'aide de GeoHash sera ensuite développée, car elle offre un compromis intéressant entre R-Tree et Quadtree. Son emploi par le modèle de Cortés et al.[20] évoqué dans l'introduction incite par ailleurs à la considérer avec attention dans le cadre de cette recherche. Enfin, une méthode récente combinant GeoHash et R-Tree dans un contexte de données massives sera présentée.

Grid-file [92] est une méthode d'indexation plate, qui consiste à simplement découper l'espace en cellules identiques à l'aide d'un motif régulier (généralement le carré). Très facile à implémenter, elle permet un haut degré de scalabilité. Par contre, dans un contexte géographique, elle mène à un grand nombre de découpes inutiles lorsque les géométries à considérer n'occupent qu'une part restreinte du territoire ou lorsque la taille des cellules est mal paramétrée [123]. De plus, lorsqu'une géométrie se trouve dans plus d'une case de la grille, elle est dupliquée lors de l'indexation. Cet aspect rend surtout la méthode efficace pour de petites géométries comme le point, car elles débordent plus rarement des cases et évitent donc les duplications.

Diagramme Voronoi Ce mode d'indexation [96] qui connaît un très grand nombre d'applications [16] et est implémenté par le système de gestion des données spatiales classique JTS, consiste à découper l'espace en zones inter-connectées, de façon à ce que les géométries de chaque zone aient le même plus proche voisin. Yu et al. indiquent qu'il présente des performances inférieures à celles de Quadtree et R-tree pour des tâches de jointure spatiale sous Spark [127].

Quadtree [49] est une méthode d'indexation hiérarchique qui existe depuis plus de 30 ans et connait des applications fort diverses [114, 112, 93], et est implémentée par le système de gestion des données spatiales classique JTS [8]. Elle consiste en un arbre dont chaque nœud non-terminal a exactement quatre fils. Dans le cas d'une utilisation géographique, les descendants d'un même nœud sont associés à des zones géographiques voisines de même taille, formant autant de subdivisions que les données en nécessitent pour être accédées efficacement. Selon Yu et al. [127], cette méthode présente des performances proches de celles de R-Tree pour des tâches de jointure spatiale dans un contexte de données massives. S'il présente une scalabilité horizontale un peu moins bonne que celle de Grid-File, Quadtree a pour avantage d'entrainer moins de découpes inutiles de l'espace que ce dernier [123]. La figure 3.2 présente ainsi une découpe de l'espace en 64 cases sous Grid-File qui ne nécessite que 28 cases sous Quadtree pour une répartition équivalente des données.

8. https://github.com/locationtech/jts

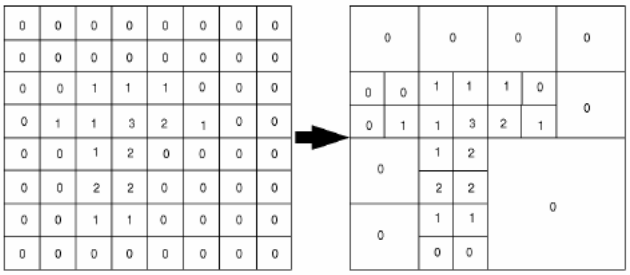

FIGURE 3.2 – Économie permise avec Quadtree (à droite) par rapport à Grid-file (à gauche) en termes de découpes de l'espace [70].

R-Tree emploie une structure en arbre dans laquelle chaque nœud contient une enveloppe qui englobe les géométries de tous les nœuds fils. Contrairement à ce qui se passe avec Quadtree, un nœud non-terminal n'a donc pas nécessairement quatre fils, et ses fils ne couvrent pas toujours des surfaces identiques. Naturellement dotée d'une moins bonne scalabilité que Grid-File et Quadtree en raison de sa structure plus hiérarchique, R-Tree est PAR contre très efficace pour éviter des découpes inutiles de l'espace [123], ce qui la rend très performante pour l'indexation de larges géométries comme des polygones correspondant à des lacs. Cela explique peut-être qu'outre son utilisation par des systèmes classiques comme JTS et PostGIS [9], R-Tree soit utilisée aussi par des systèmes de gestion des données spatiales massives plus récents, dans lesquels elle affiche des performances semblables à celles de QuadTree [127].

La figure 3.3 illustre les modes d'indexation Grid-File, Quadtree et R-Tree. La position intermédiaire de Quadtree par rapport à l'approche plane de la première méthode et l'organisation en arbre de la troisième y apparaît clairement.

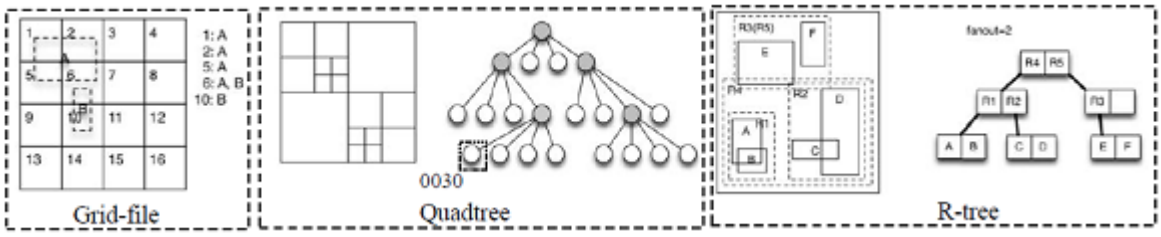

FIGURE 3.3 – Fonctionnement de Grid-File, Quadtree et R-Tree [123].

GeoHash [91] s'entendait pour son concepteur comme une simple façon de représenter des coordonnées géographiques de façon compacte. Elle a cependant été employée rapidement en tant que méthode d'indexation spatiale plane, y compris pour les données massives [67, 20]. La raison de ce succès peut s'expliquer par sa souplesse et sa simplicité de mise en œuvre : l'index s'y réduit à une clé numérique, à partir de laquelle il est possible de déduire la latitude et la longitude d'une géométrie. A chaque chiffre de cette clé correspond une zone géographique contenue dans celle exprimée par les

9. http://postgis.net

chiffres précédents, si bien que plus la clé est longue plus elle délimite précisément une zone. La figure 3.4 illustre ce fonctionnement. Il y apparaît que GeoHash permet comme Quadtree de découper l'espace selon différentes échelles, mais en représentant cette fois ces découpes par un simple nombre au lieu de nécessiter la construction d'un arbre. Comme Grid-File, GeoHash est particulièrement adapté à de petites géométries comme le point, car elles débordent plus rarement des cases de la grille utilisée.

FIGURE 3.4 – Représentation des découpes de l'espace permises par GeoHash [86].

GeoHash + R-Tree Takasu et al. [113] ont appliqué une combinaison de GeoHash et R-Tree aux architectures distribuées. Dans leur modèle, GeoHash est utilisé pour découper hiérarchiquement l'espace en grille, dont les données sont ensuite indexées par autant d'arbres R-Tree qu'il existe de cases dans la première grille. Ces auteurs ont montré l'efficacité de leur solution dans un environnement multi-serveurs, avec une réactivité proche du temps réel dans certains cas. Cependant, ces performances sont dues à la faible taille des index R-Tree construits, si bien qu'un maintien de l'efficacité lorsque la densité des données par case croît ne semble pas assuré. De plus, ce modèle implique l'utilisation d'une base de données NoSQL, ce qui ne le rend pas agnostique par rapport au format des données et risque d'impliquer des accès disque nombreux.

En résumé, il apparait que les méthodes d'indexation spatiale étudiées ont des forces et faiblesses distinctes qui les rendent parfois complémentaires. Les méthodes d'indexation planes exigent peu de dépendance entre les données et semblent donc plus à même d'offrir une bonne scalabilité horizontale que les méthodes hiérarchiques. Elles semblent donc à priori plus adaptées à l'indexation efficace de données dans un contexte distribué. Mais les méthodes d'indexation hiérarchiques comme R-Tree semblent malgré tout plus appropriées pour indexer des géométries larges ou réparties de façon irrégulière, car elles diminuent le nombre de découpes et de duplications nécessaires par rapport aux

méthodes d'indexation planes. La solution hybride de Takasu et al. [113], qui imbrique une indexation plane et une indexation hiérarchique, serait peut-être aussi à considérer mais force sous sa forme actuelle le recours à une base de données tierce.

Afin de mieux discerner comment certaines de ces méthodes pourraient être adaptées à un contexte de données spatiales massives, le chapitre suivant étudie la façon dont quelques systèmes de gestion des données spatiales massives existants gèrent l'indexation spatiale.

3.2.3 Description et limites de solutions existantes

Eldawy et Mokbel [41] classent les solutions traitant les données spatiales massives en trois catégories : « top-on », « built-in », et « from scratch ». Les premières s'ajoutent simplement à un système existant en tant que sur-couche, les secondes modifient le fonctionnement de ce système afin de l'adapter aux données spatiales, et les troisièmes construisent leur propre système de A à Z. Le tableau 3.1 synthétise les apports et faiblesses de ces différentes approches, en fonction des performances et de l'évolutivité qu'elles permettent. L'approche « top-on » y apparaît comme la plus simple, « from scratch » comme la plus performante, et « built-in » comme un compromis entre ces extrêmes.

	Top-on	**Built-in**	**From scratch**
Performances	Limitées par le système de base	Moins limitées	Optimales
Maintenance	Simple	Moyennement simple	Difficile

TABLE 3.1 – Typologie des solutions traitant les données spatiales massives.

Les paragraphes suivants discutent la façon dont plusieurs solutions de gestion des données spatiales massives gérant l'indexation spatiale se situent par rapport à cette typologie, de façon à estimer leurs forces et leurs faiblesses.

BIG-LHT [20] permet des traitements spatiaux entièrement décentralisés et tenant compte de la composante temporelle. Les données reçues y sont traitées selon deux phases : un premier niveau de serveurs les redirige suivant leur localisation spatiale en employant la clé GeoHash correspondant à leur localisation, et un second niveau de serveurs les stocke en fonction de leur date de réception. Le nombre de serveurs employés à chaque niveau évolue dynamiquement en fonction des besoins rencontrés, ce qui rend l'approche apte à minimiser les goulots d'étranglement en cas de montée en charge. Mais elle a été développée sans le support d'un environnement de traitement des données massives existant, sous prétexte de ne pas reproduire leurs défauts structurels. Ceci la positionne dans la catégorie « from scratch » de la typologie d'Eldawy et Mokbel, qui implique un prototype difficile à maintenir et à étendre.

Geomesa est une solution conçue en 2015, qui est compatible avec Hadoop et depuis peu avec Spark [67]. Historiquement conçue pour reposer sur la base de données par clé Accumulo, elle permet aussi de s'interfacer avec Cassandra depuis Spark. Cette dépendance forte avec des bases de données

NoSQL la situe clairement comme une approche top-on selon la typologie d'Eldawy et Mokbel. Sous Spark, Geomesa est le seul prototype permettant à la fois de lancer des requêtes SQL [10], et de visualiser les données. Il s'avère cependant limité dans les requêtes spatiales qu'il permet, par le fait qu'il n'a été conçu initialement que pour gérer la recherche des points inclus dans une enveloppe [41]. De plus, les tests de Xie et al. [121] montrent une durée de construction des index spatiaux plus longue que pour Spatial Spark, GeoSpark, Simba et Hadoop GIS.

Spatial Spark [124] est à notre connaissance le premier système d'indexation spatiale reposant sur Spark. Il emploie l'implémentation JTS de l'index R-tree, et l'applique à des données chargées depuis un fichier CSV et chargées dans Spark. Eldawy et Mokbel classent cette solution dans la catégorie top-on de leur typologie, mais elle nous semble plutôt relever de la catégorie built-in. En effet, une fois l'index R-Tree construit, cette solution emploie la fonctionnalité broadcast de Spark pour le transporter entre tables, ce qui s'avère bien moins coûteux qu'une jointure explicite. Spatial Spark n'offre de plus aucune interface avec Spark SQL, se contentant de permettre un pilotage depuis les paramètres de la ligne de commande. Il ne semble cependant pas y avoir d'obstacle majeur à une adaptation de son modèle à Spark SQL, à condition de revoir son implémentation [43].

GeoSpark [127] ajoute à Spark une sous-classe de RDD qui lui permet de gérer la composante spatiale dans un espace partitionné. Il s'agit donc clairement d'une approche built-in dans la typologie d'Eldawy et Mokbel : le système originel est étendu à la donnée spatiale. De même que Spatial Spark, le prototype repose sur JTS, dont il intègre trois méthodes d'indexation : Voronoi, Quadtree et R-tree. Son modèle modifie davantage la structure interne de Spark que le fait Spatial Spark, ce qui semble garantir de meilleures performances. Malheureusement, cette approche rend également plus difficile le portage vers des versions ultérieures de Spark, et empêche l'intégration de la solution à Spark SQL. Ainsi, d'après Xie et al., GeoSpark « is a library running on top of and outside of Spark without a query engine » [121].

Simba [121] pourrait être situé entre les catégories built-in et from scratch de la typologie d'Eldawy et Mokbel, puisqu'il réécrit Spark SQL d'une façon adaptée à la composante spatiale. Les DataFrames sont ainsi redéfinis en tableaux à deux dimensions plutôt qu'en séquences, ce qui facilite leur partitionnement spatial et leur indexation. De même que GeoSpark, cette solution propose plusieurs méthodes d'indexation, mais elles sont cette fois entièrement pilotées par des requêtes SQL. Si l'approche est ambitieuse, elle se heurte dans la pratique à plusieurs écueils. Premièrement, elle empêche de travailler avec une version standard de Spark, puisqu'elle nécessite qu'il soit recompilé avec sa version de Spark SQL. Elle est donc aussi difficilement transportable à une nouvelle version de celui-ci. Deuxièmement, la seule géométrie qu'elle traite est le point, alors que la norme ISO-19125 en implique six autres. Troisièmement, nos tests ont révélé des incohérences dans son analyseur syntaxique SQL, qui par exemple oblige à écrire « IN » après l'expression « POINT(x, y) » même quand aucun

10. Geomesa gère plus précisément le format ECQL, qui constitue un sous-ensemble limité de SQL.

test d'inclusion n'est envisagé. Ces éléments ne font pas de Simba une solution mature et pérenne, même si elle a le mérite d'être la première tentative d'indexation spatiale intégrant Spark SQL.

3.2.3.1 Gestion du partitionnement géographique par ces solutions

Parmi les solutions de gestion des données massives gérant l'indexation spatiales étudiées, seules BIG-LHT, Simba et GeoSpark proposent un partitionnement des données par zones avant leur traitement de façon à accélérer celui-ci. Spatial Spark propose quant à lui un partitionnement optionnel en cours de traitement, mais celui-ci s'avère moins performant que d'autres approches [121, 127] tout en nécessitant un paramétrage complexe.

Parmi ces solutions, ni BIG-LHT ni GeoSpark ne peuvent être étendues à une gestion des requêtes SQL dans leur version actuelle. En effet, la première repose sur une organisation en arbre des données assez éloignée de leur organisation tabulaire et est peu évolutive en raison de son rejet de l'écosystème Hadoop, tandis que le modèle de la seconde est inextensible au module SQL de Spark [43, 20]. Simba permet pour sa part un traitement SQL des données, mais au prix du remplacement de Spark SQL par sa propre implémentation qui introduit des erreurs de syntaxe. Autrement dit, aucun de ces prototypes n'est en mesure de lier le partitionnement qu'il propose à une gestion des requêtes SQL.

Spatial Spark s'avère donc la seule solution qui pourrait être étendue à Spark SQL, au prix certes d'une profonde réimplémentation [43]. Mais dans la pratique, la technique de partitionnement sur laquelle celle-ci repose est peu efficace, car elle n'a lieu que dynamiquement au cours de l'exécution. Elle consiste à grouper explicitement les données par zone géographique, dans des secteurs distincts de la mémoire du clusteur. Or, une telle réorganisation des données impose un coûteux reclassement de celles-ci lors de chaque traitement, car Spark SQL répartit par défaut les données de façon aléatoire sur la mémoire du cluster lors de leur chargement. De plus, le code de Spatial Spark traite les partitions à l'aide d'une boucle séquentielle [124], si bien qu'il ne permet aucune amélioration du temps d'exécution par un renforcement du parallélisme. Ce mode de partitionnement vise donc juste à diviser la construction de l'index R-Tree en plusieurs étapes dans les cas extrêmes où cela s'avère nécessaire pour éviter un dépassement de la mémoire. Pour peu que le volume de données permette de l'utiliser, la version sans partitionnement de Spatial Spark s'avère donc plus performante.

A la lumière de ce parcours des limites actuelles des systèmes de gestion des données massives en terme de gestion de l'indexation spatiale, la section suivante décrit une extension du système Elcano qui permet de la gérer plus efficacement.

3.2.4 Extension d'Elcano à l'indexation spatiale

Cette section indique les objectifs qui ont guidé l'intégration d'un module gérant l'indexation spatiale au modèle du système Elcano. Elle présente et justifie ensuite le modèle résultant et donne un aperçu de son implémentation.

3.2.4.1 Objectifs de conception

L'architecture proposée vise essentiellement à étendre le modèle du système de gestion des données spatiales massives Elcano d'une façon qui permette de mieux gérer l'indexation spatiale dans un contexte de données massives que ne le font les systèmes actuels. A cette fin, un premier objectif consiste à rendre le module d'indexation visé aussi complet que possible, de façon à ce qu'il reste efficace face à différents jeux de données. Par exemple, il devra permettre d'accélérer aussi bien le traitement de points que celui de géométries plus larges, et de gérer aussi bien une répartition homogène des objets spatiaux sur le territoire qu'une répartition très dispersée. Un second objectif vise à doter le système d'une gestion du partitionnement efficace, de façon à présenter une bonne scalabilité horizontale et à rester efficace même face à de très gros volumes de données spatiales. Un troisième objectif est de rendre ce système extensible, de façon à ce que de nouvelles méthodes d'indexation puissent y être ajoutées par l'utilisateur lorsque celles de base ne suffisent pas à traiter certaines situations particulières.

Le modèle étendu en vue de rencontrer ces objectifs est présenté et justifié ci-dessous.

3.2.4.2 Architecture

La figure 3.5 présente le nouveau modèle d'Elcano une fois étendu à l'indexation spatiale.

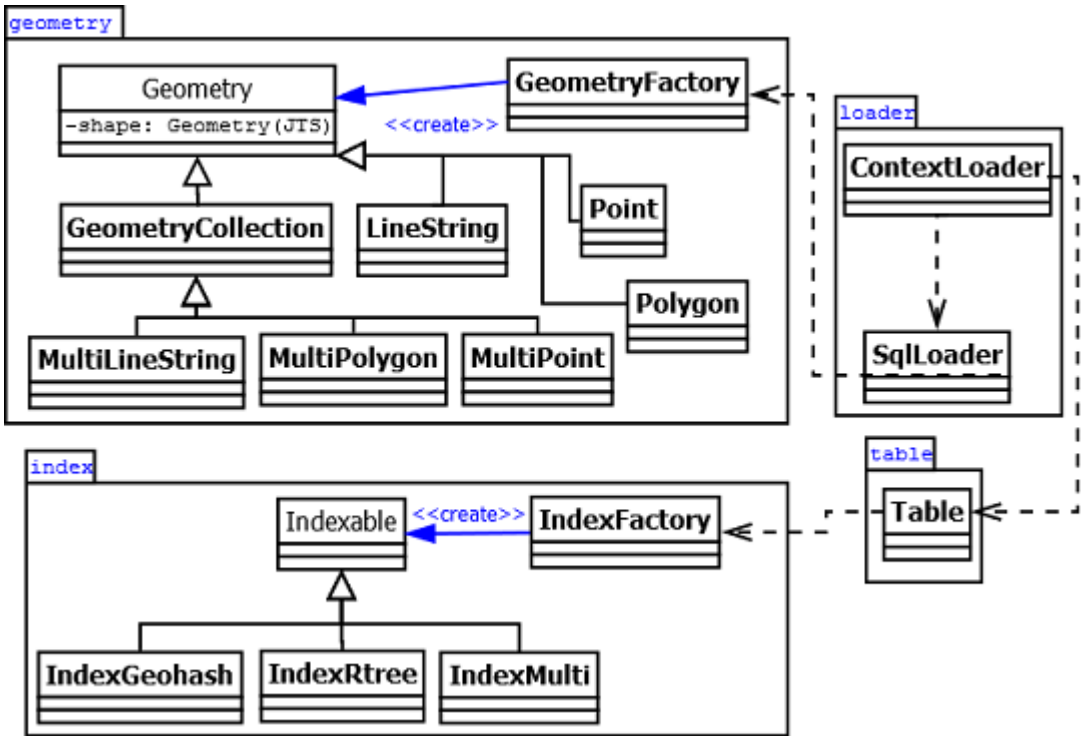

FIGURE 3.5 – Modèle général de la nouvelle version d'Elcano

Les paquetages « geometry », « table « et « library » de la partie haute du modèle correspondent à

la version de base d'Elcano, telle que décrite de façon détaillée dans un article précédent [43] et rappelée en introduction. Le paquetage « index » de la partie basse du modèle correspond au module d'indexation spatiale envisagé. Afin de bénéficier des fonctionnalités de la version de base d'Elcano, il y est relié par la classe Table de celui-ci. Lorsque cette dernière charge un jeu de données dans une table SQL, elle la rend en effet directement indexable par la classe IndexFactory du nouveau module.

La figure 3.6 montre plus spécifiquement les classes et une partie des méthodes de ce module d'indexation.

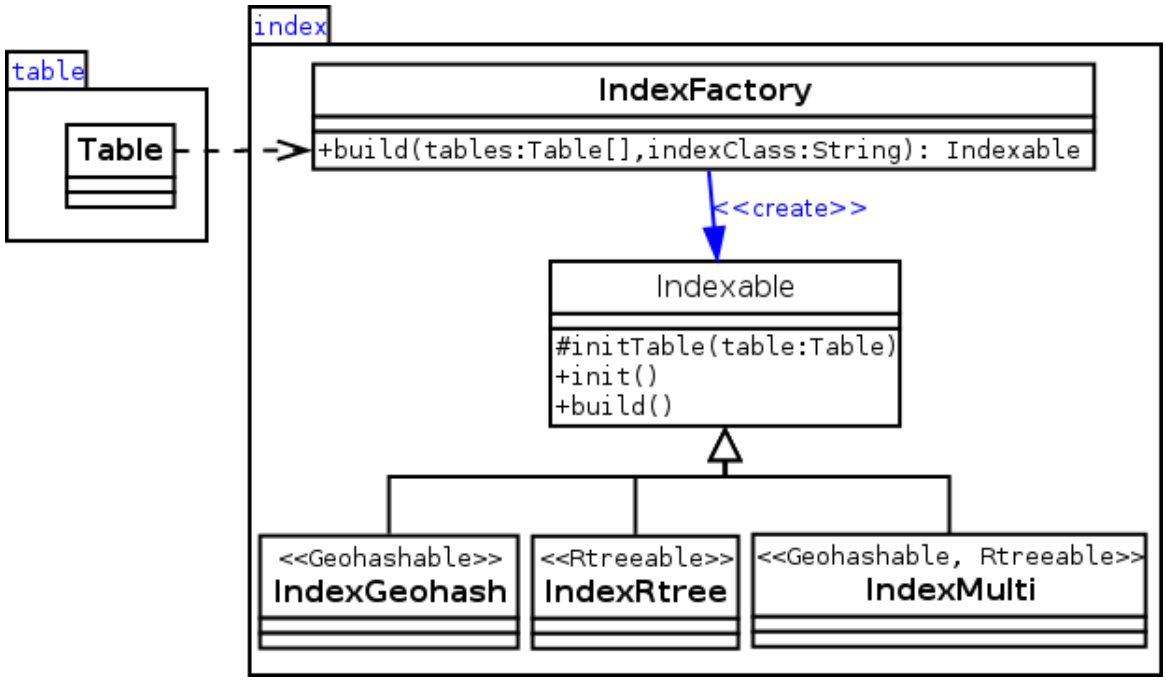

FIGURE 3.6 – Modèle de la composante d'indexation spatiale d'Elcano

Dans ce modèle, les classes IndexGeohash, IndexRTree et IndexMulti permettent de rendre le module complet quant à ses possibilités d'indexation. Les interfaces RTreeable et Geohashable ainsi que la classe Indexable participent à l'amélioration de la scalabilité horizontale du système. Cette classe Indexable permet également, en association avec IndexFactory, d'étendre l'extensibilité à d'autres types d'indexation comme envisagé.

La façon dont sont atteints ces différents objectifs est décrite et justifiée ci-dessous.

Complétude Afin de rendre le système efficace à gérer toutes les tailles et toutes les distributions de géométries, laisser l'utilisateur choisir entre plusieurs modes d'indexation en fonction de ses connaissances de la distribution et la nature des données à traiter semble plus réaliste que l'emploi d'un seul mode d'indexation qui serait optimal pour tous les cas [11]. Les méthodes d'indexation étudiées

11. Une autre approche consisterait cependant à automatiser le choix du mode indexation en fonction de l'analyse statistique d'un échantillon de données.

plus haut s'avèrent d'ailleurs adaptées à des situations distinctes en fonction en particulier de leur nature plane ou hiérarchique. Cette extension d'Elcano permet donc à l'utilisateur de choisir entre trois modes d'indexation : une méthode plane (classe GeohashIndex) permet surtout d'indexer les petites géométries (comme des points indiquant les arbres d'une forêt dense), une méthode hiérarchique (classe RTreeIndex) permet d'indexer de larges géométries (comme des zones d'activité), et une méthode qui combine les deux (classe MultiIndex) gère les cas intermédiaires. Des détails concernant le choix et le fonctionnement de ces méthodes sont fournis dans la sous-section suivante.

Gestion du partitionnement Nos essais n'ont pas permis de mettre en œuvre sous Spark SQL un partitionnement par zone géographique explicite plus performant que celui proposé par Spatial Spark, qui est comme on l'a vu la seule autre solution à tenter de définir un système de partitionnement compatible avec Spark SQL. Par contre, l'ajout aux tables indexées d'une colonne contenant une clé numérique indiquant leur zone géographique s'est avéré un mode de partitionnement implicite efficace. Spark SQL emploie en effet nativement une indexation par clé de hachage lorsque les éléments de deux tables sont joints à l'aide d'une simple égalité [60]. Ce mode de jonction s'avère donc une solution performante, qui évite autant que possible de provoquer un reclassement des données explicite [12]. Les méthodes d'indexation GeoHashIndex et MultiIndex d'Elcano emploient la clé GeoHash comme étiquette marquant les différentes zones de découpe afin d'appliquer ce principe.

Extensibilité Afin d'être facile à étendre à des cas qui ne seraient pas gérables par ces types d'indexation, le modèle repose sur le pattern de conception « fabrique abstraite » [118], ce qui permet son extension par l'ajout de classes héritant de la classe abstraite Indexable. Des détails sur le fonctionnement d'Indexable sont fournis dans la sous-section traitant de l'implémentation.

Le tableau 3.2 résume les apports du nouveau modèle d'Elcano par rapport aux quatre autres systèmes de gestion des données massives proposant un système d'indexation sous Spark. Il est le seul à respecter la norme ISO-19125 quant aux types de géométries qu'elle permet de traiter et aux fonctions SQL qu'elle rend disponibles. Comme GeoSpark et Simba, il propose un système d'indexation spatiale qui peut comporter plusieurs étapes [13].

	Spatial Spark	**GeoSpark**	**Simba**	**Geomesa**	**Elcano**
Indexation	1 étape	2 étapes	2 étapes	via Accumulo	n étapes
Syntaxe proche de SQL	Non, et difficile à étendre à Spark SQL	Non	Oui, sans extensibilité et avec quelques bogues	Oui, en CQL	Oui
Types de géométries	Toutes	Toutes	Point	Point	Toutes

TABLE 3.2 – Comparatif d'Elcano et des prototypes concurrents

12. Par contre, le regroupement d'éléments à l'aide d'une fonction SQL définie par l'utilisateur, d'une inégalité ou d'un « LIKE » est bien plus lente sous Spark SQL, car elle implique un tri des données et donc leur reclassement.
13. La gestion de l'indexation est décrite plus en détail dans la section suivante.

La sous-section suivante revient plus en détail sur les modes d'indexation proposés par le modèle.

3.2.4.3 Méthodes d'indexation proposées

Comme évoqué ci-dessus, trois méthodes d'indexation ont été ajoutées à Elcano afin de permettre la gestion de tous les types de géométries. Il s'agissait plus précisément d'intégrer une méthode d'indexation plane afin de traiter efficacement les géométries de petite taille ou réparties de façon homogène, une méthode d'indexation hiérarchique afin de traiter efficacement les géométries de grandes dimensions ou dispersées, et une méthode hybride afin de traiter les cas intermédiaires. Cette sous-section justifie et décrit plus précisément les méthodes proposées.

Méthode d'indexation hiérarchique R-Tree a été choisie en tant que méthode hiérarchique car elle affiche des performances équivalentes à Quadtree dans un contexte de données massives [127] tout en entrainant moins de découpes de l'espace. La méthode commence par extraire les identifiants et les enveloppes de toutes les données et s'en sert pour construire l'arbre R-Tree correspondant à leurs rapports de proximité. Elle diffuse ensuite cet index sur tous les serveurs du cluster, de façon à ce qu'il soit disponible pour des traitements parallèles. Par sa nature hiérarchique, cette méthode a une moins bonne scalabilité horizontale que les deux autres, mais nos tests indiquent qu'elle est plus performante que GeoHash lorsque le volume de données à indexer est faible ou moyen.

Méthode d'indexation plane GeoHash a été choisie parmi les méthodes planes car elle permet une scalabilité horizontale comparable à celle de R-Tree tout en étant plus structurée. En regroupant les zones par indice, elle permet de recourir au partitionnement implicite évoqué plus haut et donc de traiter de très gros volumes de données efficacement. Elle est surtout performante pour indexer des points répartis de façon homogène sur le territoire, mais afin de l'adapter au cas de géométries de plus grande taille, il est possible de paramétrer l'index lors de sa création de façon à ce que les clés de sa découpe GeoHash soient moins précises. La taille des carrés de la grille de découpe dépend en effet de la taille de la clé GeoHash utilisée, celle-ci pouvant aller de 0 (aucune découpe) à 63 (découpe la plus fine). Un défaut de cette approche est si une géométrie est si large qu'elle n'appartient pas à une case de la grille, elle est ignorée (une duplication s'avérant, lors de nos tests, trop couteuse en temps d'exécution). La méthode d'indexation suivante surmonte cet écueil, ce qui la rend plus adéquate pour traiter des géométries aux tailles variées.

Méthode d'indexation combinée L'index combiné étend le fonctionnement de l'index GeoHash. Afin de gérer aussi le cas de géométries aux tailles variées, l'index GeoHash peut en effet être paramétré pour reposer sur plusieurs grilles. Il fonctionne alors à la façon d'un tamis inversé, qui commencerait par capturer les grains de sable les plus petits et terminerait par les plus gros galets. Les grilles sont donc appliquées successivement lors de l'indexation pour réunir les objets proches, en commençant par la plus précise. Un index R-Tree récupère ensuite les géométries qui n'ont été capturées par aucune grille et leur applique une indexation R-Tree, de façon à assurer qu'aucune géométrie n'est

ignorée. Cette approche permet ainsi de traiter les données qui le peuvent à l'aide de GeoHash puis de profiter des performances supérieures de R-Tree pour le traitement de petits volumes de données pour traiter celles qui restent. Elle évoque donc le modèle hybride de Takasu et al. [113], mais sans obliger à l'emploi d'une base de données externe. Elle constitue donc une nouvelle façon efficace d'indexer des géométries de tailles variées. L'opération supplémentaire qu'elle nécessite la rend cependant plus lente que GeoHash, qui sera préférée pour indexer de petites géométries comme les points. R-Tree sera quant à lui préféré si le jeu de données ne contient que des géométries de grande taille comme les frontières de pays, un passage préliminaire par GeoHash s'avérant alors de peu d'intérêt.

Les avantages et inconvénients de ces trois modes d'indexation sont synthétisés dans le tableau 3.3.

	Avantages	**Inconvénients**
GeoHash	Reste performant pour de gros volume de données	Limité à de petites géométries comme le point, plus lent que R-Tree pour de petits volumes de données.
R-Tree	Indexe efficacement de larges géométries comme les régions	De construction plus lente et plus vite dépassé par le volume des données que GeoHash.
GeoHash + R-Tree	Indexe efficacement des géométries de tailles variées	Plus lent que les autres types d'indexation si la taille des géométries est homogène.

TABLE 3.3 – Modes d'indexation nativement disponibles dans Elcano.

La sous-section suivante fourni quelques détails plus annexes concernant l'implémentation d'Elcano et de ses modes d'indexation.

3.2.4.4 Implémentation

Après la présentation du nouveau modèle d'Elcano et des modes d'indexation qu'il permet, cette section évoque quelques points de détails liés à son implémentation.

Bibliothèques utilisées La méthode d'indexation R-Tree proposée repose, comme c'est le cas pour Spatial Spark, sur la version JTS de cet index, car elle est à notre connaissance sa seule implémentation mature et open source. Contrairement à ce qui se passe avec Spatial Spark, l'index est cependant intégré à Spark SQL, ce qui permet de profiter des optimisations propres à cette composante et d'augmenter d'autant la distribution des opérations. De plus, la construction JTS de l'index [14] n'a lieu qu'après sa diffusion sur les différents serveurs, de façon à réduire la charge mémoire utilisée lors de celle-ci. Nos tests préliminaires indiquent en effet que l'index demande un temps de transfert deux fois moindre s'il est diffusé avant sa construction par JTS plutôt qu'ensuite. Non géré par JTS, l'in-

14. Cette construction est une étape nécessaire de la production d'un index R-Tree dans JTS. Elle intervient après que toutes les géométries y ont été ajoutées, et avant qu'il soit consulté pour la première fois.

dex GeoHash emploie quant à lui la bibliothèque « GeoHash Java » [15], car l'intégration de celle-ci est immédiate et fournit des méthodes variées comme la recherche de la liste des GeoHash pouvant être associés à une enveloppe.

Gestion de l'extensibilité Afin de rendre la création de nouvelles méthodes d'indexation spatiale dans Elcano simple et générique, les classes qui en héritent ne sont contraintes que de fournir deux paramètres obligatoires lors de leur construction (tableau des tables à indexer et chaîne de caractère indiquant la méthode à employer) et d'implémenter deux méthodes (transformations à appliquer à chaque table avant la construction de l'index, et construction de celui-ci de façon à relier les différentes tables).

Factorisation Afin de limiter la duplication de code lors de l'implémentation, celle-ci est gérée par des traits Scala. Il s'agit de structures permettent à la fois l'héritage multiple et l'implémentation des méthodes, de façon à réunir les avantages des interfaces et des classes en Java. Cette facilité propre à Scala a facilité de beaucoup l'implémentation de la classe IndexMulti, qui combine les traits correspondant aux indexations GeoHash et R-Tree.

A présent que le modèle d'Elcano a été étendu à l'indexation spatiale et que la façon dont il gère différents types d'indexation et quelques éléments de son implémentation ont été parcourus, la section suivante décrit le banc de test qui lui a été appliqué, de façon à évaluer les performances qu'il permet par rapport à d'autres solutions.

3.2.5 Banc de tests comparatifs

Cette section compare les performances d'Elcano à un autre système de gestion des données spatiales massives sous Spark (Spatial Spark) et à un système de gestion des données spatiales classique (PostGIS). Spatial Spark a été choisi parmi les systèmes étudiés car c'est le seul dont le modèle semble adaptable au format SQL (au prix certes d'une réimplémentation) [43], et PostGIS car il est à notre connaissance l'unique système classique permettant des requêtes SQL spatiales entièrement conformes à la norme ISO-19125 et gérant l'indexation spatiale. Une autre raison est que ces trois systèmes gèrent une méthode d'indexation commune (R-Tree), ce qui facilite une juste comparaison de leurs performances.

Pour les besoins de ce banc de tests, Elcano et Spatial Spark ont été installés sur un cluster de serveurs constitué d'un serveur maître doté de 8 Go mémoire vive et de neuf serveurs esclaves dotés de 4Go de mémoire vive. Chaque machine du cluster emploie le système d'exploitation CentOS 6.5 et comporte huit processeurs Intel Xeon de 2.33 GHz. PostGIS a été optimisé à l'aide de pgTune [16] et testé dans des conditions comparables.

15. https://mvnrepository.com/artifact/ch.hsr/geohash
16. http://pgtune.leopard.in.ua/

Chaque test de cette section consiste en un décompte du résultat d'une jointure spatiale entre deux tables. Il s'agit donc de sélectionner parmi les éléments de ces tables les couples qui respectent une relation spatiale donnée (généralement en rapport avec leur proximité). Le choix de cette opération vient du fait qu'elle serait très lourde si elle était effectuée sans indexation, puisqu'elle impliquerait un calcul d'intersection entre les géométries de chaque ligne résultant du produit cartésien des deux tables. La jointure employée ne retient que les géométries qui se rencontrent précisément [17]. Le retour est un simple décompte, afin que la tâche soit gérable par Spatial Spark bien que ses limites l'empêchent de répondre à des requêtes SQL. Finalement, les données considérées sont uniquement des tables statiques, c'est-à-dire que leur contenu n'évolue pas avec le temps [18].

Le test 1 vise à comparer la vitesse d'exécution de PostGIS, de Spatial Spark et d'Elcano (en mode R-Tree) face à une montée en volume des données. Il s'agit d'un comptage des intersections entre l'enveloppe englobant le Québec et sept jeux de points positionnés aléatoirement dans l'enveloppe du Canada. Ces sept listes consistent en des fichiers CSV générés à l'aide d'un script Python puis chargés dans HDFS. Elles comptent respectivement 1000, 10 000, 100 000, un million, cent millions et un milliard de points [19]. Leur intersection avec une enveloppe entourant la région du Québec a été testée pour chacun des systèmes mesurés.

Le tableau 3.4 présente une synthèse des résultats du premier test. Afin de faciliter une première comparaison entre les différents systèmes, la durée cumule les temps nécessaires pour l'indexation et pour une première requête sur les données. Il apparaît que les performances d'Elcano dépassent toujours celles de PostGIS et de Spatial Spark au-delà d'un million de points. Pour les plus petits volumes, PostGIS dépasse les deux autres solutions, mais il présente un ralentissement de plus en plus marqué ensuite. Il met ainsi plusieurs heures à traiter 100 millions de points, contre moins de cinq minutes pour Elcano. La différence entre Spatial Spark et Elcano est plus ténue, mais elle s'accentue au profit d'Elcano à mesure que la charge augmente.

Le moins bon résultat de PostGIS lorsque le nombre de serveurs augmente s'explique sans doute par la faible scalabilité horizontale qu'il permet, n'étant pas un système conçu pour le traitement des données massives. Le dépassement de Spatial Spark par Elcano peut s'expliquer quant à lui par le fait que le second repose sur Spark SQL et bénéficie donc des optimisations de requêtes et des mises en cache de ce dernier [43, 6], tandis que Spatial Spark n'emploie que la version de base de Spark. Par contre, pour de petits volumes de données (moins d'un million de points), la solution classique PostGIS l'emporte, sans doute parce que son architecture est plus simple implique moins de traitements qu'une solution

17. L'emploi d'une distance de tolérance, bien que techniquement possible, n'aurait fait que renforcer le nombre de correspondances entre les tables et nous a donc pas semblé indispensable a une comparaison élémentaire des systèmes.

18. En cas de dépassement de cette limitation et donc d'extension de la solution à un contexte de données reçues en flux continu, il faudrait également prendre en compte les délais de mise à jour et de suppression de données comme le font Cortés et al. [20] pour l'évaluation de BIG-LHT

19. Les tests suivants ont tous été effectués en utilisant des points, car Elcano comme Spatial Spark transforment tous les types de géométries en enveloppes avant de les indexer, si bien qu'il semble légitime d'attendre un résultat similaire quelle que soit la géometrie. Il serait cependant intéressant dans un travail futur d'appliquer des tests similaires avec d'autres géométries, afin d'étudier l'impact précis de la forme et de la taille des données spatiales sur les performances d'indexation.

distribuée alors que cette dernière n'est pas nécessaire pour de tels volumes. De même, le dépassement d'Elcano par Spatial Spark en dessous de dix millions de points peut sans doute s'expliquer par la surcouche de traitements qu'implique l'emploi du module Spark SQL d'Elcano alors que ce module montre toute son efficacité pour de plus gros volumes.

Nombre de points	**PostGIS (ms)**	**Spatial Spark (ms)**	**Elcano (ms)**
1000	234	6 543	9 516
10 000	326	6 622	9 714
100 000	3 783	8 301	9 030
1 000 000	29 898	8 301	10 747
10 000 000	269 257	20 487	17 099
100 000 000	5 752 821	55 017	37 378
1 000 000 000	plus de 10h	399 100	273 074

TABLE 3.4 – Test 1 - Durée de comptage d'intersections en fonction d'une montée en volume.

Le test 2 compare la scalabilité d'Elcano et de Spatial Spark pour un nombre de serveurs variant de 1 à 9, pour le décompte des intersections entre l'enveloppe du Québec et un milliard de points aléatoirement situés dans l'enveloppe du Canada. PostGIS n'a pas été intégré à ce test, car d'une part le test précédent établit clairement qu'il sous-performe les deux autres systèmes pour cette tâche, et d'autre part aucune extension ne permet de diviser ses traitements entre plusieurs serveurs.

Le tableau 3.5 présente une synthèse des résultats du second test. Spatial Spark et Elcano y présentent dans tous les cas de bons résultats en termes de scalabilité horizontale. En outre, lors du passage de 1 à 9 serveurs, les temps d'exécution d'Elcano et de Spatial Spark connaissent des diminutions similaires : 87,4% pour Spatial Spark et 87,2% pour Elcano. Par contre, Elcano s'avère environ une fois et demi plus rapide que Spatial Spark quel que soit le nombre de serveurs employés.

La plus grande rapidité d'Elcano lors de ce second test peut sans doute s'expliquer à nouveau par son emploi de Spark SQL, ainsi que par les optimisations (détaillées plus haut) que son intégration de R-Tree apporte. Le taux de scalabilité des deux systèmes est par contre très proche, ce qui peut s'expliquer par le fait qu'ils emploient tous les deux la bibliothèque spatiale JTS et la même méthode d'indexation.

Nombre de serveurs	Spatial Spark (ms)	Elcano (ms)
1	3 349 414	2 196 344
2	1 718 672	1 123 153
3	1 143 790	762 536
4	875 284	588 401
5	696 195	473 635
6	586 211	391 297
7	511 111	340 784
8	456 446	314 796
9	423 647	280 761

TABLE 3.5 – Test 2 - Scalabilité en fonction du nombre de serveurs employés.

Le test 3 fournit un comparatif plus détaillé des performances de PostGIS, Spatial Spark et Elcano (en mode R-Tree puis en mode GeoHash). Il consiste à dénombrer les intersections entre tous les points d'un jeu d'un million de points situés dans l'enveloppe du Canada et tous les points d'une copie de ce jeu (évaluant ainsi 100 milliards d'intersections). La durée d'exécution est cette fois divisée en temps nécessaire pour l'indexation, pour une première requête et pour une seconde requête. De cette façon, le résultat est plus proche des performances en situation réelle et permet de comparer plus précisément les solutions. En effet, alors que l'indexation n'est nécessaire qu'une fois pour deux tables données, une requête SQL doit être lancée pour chaque opération de jointure spatiale qui leur est appliquée.

Le tableau 3.6 présente une synthèse des résultats du troisième test. L'indexation des données par PostGIS y apparaît comme un peu plus rapide qu'avec Spatial Spark, mais au prix d'une durée d'exécution des requêtes SQL trois fois plus élevée. Elcano présente dans tous les autres cas les meilleures performances. Ainsi, il indexe les données jusqu'à cinq fois plus rapidement que PostGIS et exécute la première requête sur celle-ci deux fois plus vite que Spatial Spark. Elcano est en outre la seule des trois solutions à rendre une seconde requête sur des données identiques plus performante que la première. Ainsi, Elcano exécute la seconde requête plus de 26 fois plus vite que Spatial Spark lorsqu'il est en mode R-Tree, et jusque six fois plus vite lorsqu'il est en mode GeoHash. La moindre performance du mode GeoHash d'Elcano par rapport au mode R-Tree pour cette seconde requête est compensée par la construction deux fois plus rapide de son index, ce qui illustre la complémentarité des différents modes d'indexation choisis.

Les meilleures performances d'Elcano lors d'une seconde requête pourrait s'expliquer par la mise en cache performante et la gestion de tables temporaires qu'autorise Spark SQL. Le temps de création plus long pour l'index R-Tree s'explique sans doute quant à lui par la nécessité de le diffuser intégralement sur tous les serveurs, alors que GeoHash ne nécessite que d'adjoindre une clé numérique aux données des différentes tables. Le meilleur temps d'exécution de R-Tree par rapport à GeoHash à partir de la seconde requête s'explique sans doute quant à lui par le peu d'échanges entre les serveurs que nécessite R-Tree une fois calculé (une copie de l'index est en effet présente en mémoire sur tous les serveurs et peut être directement consultée par chacun d'eux).

Solution	Indexation	Première requête (ms)	Seconde requête (ms)
PostGIS	29 756	100 742	100 742
Spatial Spark	36 824	36 824	36 824
Elcano (mode R-Tree)	13 578	15 754	1 393
Elcano (mode Geo-hash)	5 518	13 492	6 862

TABLE 3.6 – Test 3 - Durée de création d'index, puis de première et de seconde requête.

Au-delà d'un certain volume de données, Elcano surpasse donc PostGIS et Spatial Spark en termes de vitesse d'exécution (test 1) et de gestion des requêtes multiples (test 2). Il présente une scalabilité comparable à Spatial Spark, mais maintient une vitesse d'exécution supérieure à celui-ci lorsque le nombre de serveurs augmente.

3.2.6 Conclusion

Le présent article a commencé par donner un aperçu des possibilités et des solutions actuelles en terme d'indexation spatiale des données massives. L'intégration à un système de gestion des données spatiales massives (Elcano [43]) a ensuite été proposée. A la fois complet, doté d'une gestion efficace de la scalabilité horizontale et extensible, le modèle qui en résulte permet d'indexer les données spatiales à l'aide de R-Tree, de GeoHash ou d'une combinaison des deux. Comme la version de base d'Elcano, il repose sur Spark SQL, ce qui ouvre d'intéressantes possibilités en termes de performances et d'extensibilité. Les tests d'une implémentation de la solution montrent ainsi qu'une seconde requête lancée par Elcano sur des données indexées est plus rapide que la première grâce aux mises en cache que son utilisation de Spark SQL permet, ce qui n'est pas le cas pour d'autres solutions comme Spatial Spark. Elcano affiche également d'aussi bons résultats que Spatial Spark en terme de scalabilité horizontale, et s'avère plus rapide que PostGIS et Spatial Spark à partir des jointures spatiales de 10 millions de points ou plus.

Par contre, les possibilités d'évolution d'une architecture reposant sur un traitement en mémoire vive des données massives comme Spark reste une question ouverte. Par exemple, Elcano ne propose pas actuellement de solution pour le traitement de données évoluant en fonction du temps, bien que Spark permette de traiter des données reçues en continu. Une façon d'y parvenir pourrait être de compléter notre modèle d'une stratégie de détection de modèles d'événements spatio-temporels inspirée de celles décrites par Andrienko et al. [5] et par Thom et al. [115], ainsi que d'un système de détection des « points chauds » dans un flux d'événements [82].

3.2.6.1 Références

[5] Natalia ANDRIENKO et al. “Detection, tracking, and visualization of spatial event clusters for real time monitoring”. In : *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*. IEEE. 2015, p. 1–10.

[6] M. ARMBRUST et al. “Spark sql : Relational data processing in spark”. In : *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 2015, p. 1383–1394.

[16] R. CHENG et al. “Uv-diagram : A voronoi diagram for uncertain data”. In : *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*. 2010, p. 796–807.

[20] R. CORTÉS et al. “A Scalable Architecture for Spatio-Temporal Range Queries over Big Location Data”. In : *Network Computing and Applications (NCA), 2015 IEEE 14th International Symposium on*. 2015, p. 159–166.

[23] T. K. DANG, J. KÜNG et R. WAGNER. “The sh-tree : A super hybrid index structure for multidimensional data”. In : *International Conference on Database and Expert Systems Applications*. Springer. 2001, p. 340–349.

[28] J. DEAN et S. GHEMAWAT. “MapReduce : simplified data processing on large clusters”. In : *Communications of the ACM* 51.1 (2008), p. 107–113.

[41] A. ELDAWY et Mohamed F MOKBEL. “The Era of Big Spatial Data : A Survey”. In : *Information and Media Technologies* 10.2 (2015), p. 305–316.

[43] J. ENGÉLINUS et T. BADARD. “Elcano : un système de gestion des données massives à composante spatiale basé sur Spark SQL », Revue internationale de géomatique”. In : *Revue internationale de géomatique (en cours de soumission)*. 2016.

[45] J. ENGÉLINUS et T. BADARD. “Vers une indexation des données spatiales massives performante et efficace”. In : *Geomatica (en cours de soumission)*. 2016.

[49] R. A. FINKEL et J. L. BENTLEY. “Quad trees a data structure for retrieval on composite keys”. In : *Acta informatica* 4.1 (1974), p. 1–9.

[51] C. FRANKLIN et P. HANE. “An Introduction to Geographic Information Systems : Linking Maps to Databases [and] Maps for the Rest of Us : Affordable and Fun.” In : *Database* 15.2 (1992), p. 12–15.

[52] J. GANTZ et D. REINSEL. “The digital universe in 2020 : Big data, bigger digital shadows, and biggest growth in the far east”. In : *IDC iView : IDC Analyze the future* 2007 (2012), p. 1–16.

[60] K. GULIYEV et A. SHAIKHHA. *Query Synthesis for Big Data*. GRIN Verlag, 2015.

[67] J. N. HUGHES et al. “GeoMesa : a distributed architecture for spatio-temporal fusion”. In : *SPIE Defense + Security*. International Society for Optics et Photonics. 2015, 94730F.

[70] A. KNOLL et al. “Interactive isosurface ray tracing of large octree volumes”. In : *Interactive Ray Tracing 2006, IEEE Symposium on*. IEEE. 2006, p. 115–124.

[72] R. K. V. KOTHURI, S. RAVADA et D. ABUGOV. “Quadtree and R-tree indexes in oracle spatial : a comparison using GIS data”. In : *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*. ACM. 2002, p. 546–557.

[78] D. B. LOMET et B. SALZBERG. “The hB-tree : A multiattribute indexing method with good guaranteed performance”. In : *ACM Transactions on Database Systems (TODS)* 15.4 (1990), p. 625–658.

[82] R. MACIEJEWSKI et al. “A visual analytics approach to understanding spatiotemporal hotspots”. In : *IEEE Transactions on Visualization and Computer Graphics* 16.2 (2010), p. 205–220.

[83] M. MAIER, A. SEREBRENIK et I. VANDERFEESTEN. *Towards a Big Data Reference Architecture*. 2013.

[84] J. MANYIKA et al. “Big data : The next frontier for innovation, competition, and productivity”. In : *The McKinsey Global Institute* (2011).

[86] Y. MATSUDA. *Geo Library for Amazon DynamoDB – Part 1 : Table Structure*. 2013. URL : https://aws.amazon.com/fr/blogs/mobile/geo-library-for-amazon-dynamodb-part-1-table-structure/.

[91] G. NIEMEYER. *Geohash*. 2008.

[92] J. NIEVERGELT, H. HINTERBERGER et K. C. SEVCIK. “The grid file : An adaptable, symmetric multikey file structure”. In : *ACM Transactions on Database Systems (TODS)* 9.1 (1984), p. 38–71.

[93] H. NOBORIO, T. NANIWA et S. ARIMOTO. “A quadtree-based path-planning algorithm for a mobile robot”. In : *Journal of Robotic Systems* 7.4 (1990), p. 555–574.

[96] A. OKABE, B. BOOTS et K. SUGIHARA. “Nearest neighbourhood operations with generalized Voronoi diagrams : a review”. In : *International Journal of Geographical Information Systems* 8.1 (1994), p. 43–71.

[99] R. PICOT CLEMENTE, C. BOTHOREL et P. LENCA. *Une brève introduction aux Données Massives-Challenges et perspectives*. Rapp. tech. HAL, 2015.

[104] Ram S. *Magellan : Geospatial Analytics on Spark*. 2015. URL : http://hortonworks.com/blog/magellan-geospatial-analytics-in-spark/.

[112] G. J. SULLIVAN et R. L. BAKER. “Efficient quadtree coding of images and video”. In : *IEEE Transactions on Image Processing* 3.3 (1994), p. 327–331.

[113] A. TAKASU et al. “An Efficient Distributed Index for Geospatial Databases”. In : *International Conference on Database and Expert Systems Applications*. Springer. 2015, p. 28–42.

[114] E. TANIN, A. HARWOOD et H. SAMET. “Using a distributed quadtree index in peer-to-peer networks”. In : *The VLDB Journal* 16.2 (2007), p. 165–178.

[115] D. THOM et al. “Spatiotemporal anomaly detection through visual analysis of geolocated twitter messages”. In : *Visualization Symposium (PacificVis), 2012 IEEE Pacific*. IEEE. 2012, p. 41–48.

[118] J. VLISSIDES et al. “Design patterns : Elements of reusable object-oriented software”. In : *Reading : Addison-Wesley* 49.120 (1995), p. 11.

[120] T. WHITE. *Hadoop : The definitive guide*. O’Reilly Media, Inc., 2012.

[121] Dong XIE et al. *Simba : Efficient In-Memory Spatial Analytics*. URL : https://www.cs.utah.edu/~lifeifei/papers/simba.pdf.

[123] S. YOU. “Large-scale spatial data management on moderne parallel and distributed platforms”. In : 2015.

[124] S. YOU, J. ZHANG et L. GRUENWALD. “Large-scale spatial join query processing in cloud”. In : *Data Engineering Workshops (ICDEW), 2015 31st IEEE International Conference on*. IEEE. 2015, p. 34–41.

[126] J. YU. *GeoSpark’s Github - issue 33*. https://github.com/DataSystemsLab/GeoSpark/issues/33. Accessed : 2017-02-26.

[127] J. YU, J. WU et M. SARWAT. “Geospark : A cluster computing framework for processing large-scale spatial data”. In : *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM. 2015, p. 70.

[128] M. ZAHARIA et al. “Spark : cluster computing with working sets”. In : *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*. T. 10. 2010, p. 10.

3.3 Compléments à l’article

3.4 Impact de la persistance sur les performances

Par manque de place, l’article ne traite pas des conséquences de la persistance des données sur la durée d’indexation. Cette section y remédie, en comparant une indexation reposant entièrement sur la mémoire vive avec deux variantes. La première variante sauve les données d’une table sur HDFS dans un fichier Parquet [119] après leur chargement sous forme de DataFrame et avant leur indexation, tandis que la seconde variante les sauve dans un tel fichier une fois indexées. Dans les deux cas, les données sont ensuite chargées depuis le fichier Parquet créé plutôt que depuis leur version d’origine. Ceci évite que Spark SQL, en raison de sa gestion paresseuse des actions, répète des opérations liées à la construction de leur DataFrame et à leur indexation.

Les tableaux 3.7 et 3.8 présentent les résultats de ces trois modes de persistance. Il consiste pour tous ces cas à dénombrer les intersections entre un jeu de points situés aléatoirement dans l’enveloppe du

Canada et une copie de ce jeu. Le premier tableau montre que le recours à la seule mémoire vive est dans tous les cas plus rapide que le recours à Parquet s'il s'agit juste d'effectuer un seul comptage. Cependant, le second tableau montre qu'une persistance après indexation accélère de beaucoup une seconde requête sur des données indexées, tandis qu'une persistance avant indexation est généralement sans influence sur les performances. De plus, le gain d'une persistance après indexation est plus marqué pour les index GeoHash et combiné que pour l'index R-Tree.

TABLE 3.7 – Durée de comptage d'intersections pour différents modes de persistance.

Nombre de points	**Indexation**	**Si sauvegarde avant indexation (ms)**	**Si sauvegarde après indexation (ms)**	**Si aucune sauvegarde (ms)**
1000	R-Tree	13580	11671	8299
1000	GeoHash	18694	18527	13973
1000	Combiné	21344	25764	15297
10000	R-Tree	14309	13150	9181
10000	GeoHash	18771	17988	14622
10000	Combiné	20370	21275	15063
100000	R-Tree	16149	17318	11261
100000	GeoHash	18957	18838	14090
100000	Combiné	24085	21964	16599
1000000	R-Tree	36152	37608	28497
1000000	GeoHash	21852	24054	19010
1000000	Combiné	25000	26976	20150

TABLE 3.8 – Durée de second comptage d'intersections pour différents modes de persistance.

Nombre de points	**Indexation**	**Si sauvegarde avant indexation (ms)**	**Si sauvegarde après indexation (ms)**	**Si aucune sauvegarde (ms)**
1000	R-Tree	657	596	660
1000	GeoHash	3969	586	3492
1000	Combiné	3811	721	3657
10000	R-Tree	536	806	706
10000	GeoHash	3996	514	3613
10000	Combiné	3829	670	3499
100000	R-Tree	1061	769	1073
100000	GeoHash	4502	556	4039
100000	Combiné	3988	554	3932
1000000	R-Tree	1314	954	1486
1000000	GeoHash	5399	583	6862
1000000	Combiné	5575	684	6100

Ces résultats suggèrent qu'une persistance de données spatiales indexées n'améliore véritablement les performances d'Elcano que si une réutilisation importante est visée. De plus, le recours à des

sauvegardes intermédiaires n'améliore pas les performances, ce qui semble indiquer que Spark SQL utilise la mémoire de façon très efficiente.

3.5 Opportunité d'une alternative à HDFS

Il resterait à déterminer si l'impact de la persistance sur la performance évolue lorsque celle-ci est gérée par un système de gestion des données NoSQL plutôt que via HDFS. Le système de fichiers HDFS ne permet en effet pas nativement de gérer des données typées ou classées en colonnes. Le format Parquet semble permettre de dépasser cette limitation, mais c'est au prix d'un temps en écriture parfois rébarbatif. Il serait donc intéressant de déterminer si parmi les nombreux systèmes de bases de données NoSQL existants, certains sont à même de faciliter la persistance des données spatiales massives sous Spark.

Une première tentative en ce sens a consisté à relier le système de gestion de base de données NoSQL par colonnes Cassandra [76] à Elcano à l'aide du connecteur DataStax [20]. Il semble en résulter un temps de sauvegarde et de récupération au moins aussi bon qu'avec Parquet, mais ce point demanderait des tests plus précis pour être étayé. Bien que ne gérant pas directement les données spatiales, Cassandra permet le stockage de données de type binaire, ce qui a permis d'y stocker des géométries au format WKB. Malheureusement, il s'est par contre avéré impossible de créer ou modifier des tables Cassandra depuis Spark, car ce n'était pas géré par le connecteur employé. Il a donc été nécessaire de créer et typer les tables extérieurement à Spark avant de les utiliser, ce qui illustre les limites actuelles d'un tel couplage.

Si même ce couplage avec une base de données NoSQL ne gérant pas le type spatial et reliée à Spark via un connecteur assez limité semble aller dans le sens d'une amélioration d'Elcano, il serait intéressant de tester son association à des systèmes NoSQL à priori mieux adaptés. Il conviendrait par exemple de déterminer dans quelle mesure l'utilisation d'une base de données NoSQL par documents intégrant une référence spatiale, comme MongoDB [17] et ElasticSearch [56], améliorerait les performances. Il serait également intéressant de mesurer l'impact d'une intégration de la base de données par colonnes HBase [54], qui repose directement sur HDFS.

3.6 Limites du banc de test appliqué

En l'état, les tests intégrés à l'article portent principalement sur des géométries de type Point. Cela peut sembler une lacune par rapport à d'autres types de géométries, car qu'est-ce qui indique que leur indexation ne s'avérerait pas plus lente ? En fait, cette inquiétude disparaît si on sait que quel que soit le type de géométrie utilisé, l'indexation se construit à partir de l'enveloppe et non de la géométrie elle-même. Quel que soit la géométrie de base, l'enveloppe se présente en effet sous la forme d'un polygone rectangulaire parallèle aux axes du système de coordonnées.

20. https ://github.com/datastax/spark-cassandra-connector

Par contre, la largeur des géométries pourrait peut-être avoir un impact sur les performances : les géométries larges ont de plus grosses enveloppes, ce qui augmente potentiellement la quantité d'intersections à tester en cas de jointure. Il serait donc intéressant d'établir d'autres séries de tests à même d'estimer le comportement des différents types d'indexation, face à des géométries de tailles et de proportions variées. Cela n'a cependant pas été nécessaire dans le cadre de la présente recherche, car les résultats du banc de test [3.2.5] suffisent à établir qu'Elcano est plus rapide que Spatial Spark et PostGIS à enveloppes équivalentes.

3.7 Conclusion

Ce chapitre présente les possibilités de quelques méthodes d'indexation spatiale dans un contexte classique ainsi que dans celui des données massives [3.2.2.1]. Il étudie aussi la façon dont quelques systèmes gérant les données spatiales massives intègrent ces méthodes [3.2.3]. Il présente ensuite un modèle qui ajoute un module d'extension spatiale à Elcano et les méthodes d'indexation spatiales qu'il propose [3.2.4.2] et compare ses performances à celles d'autres solutions du marché [3.2.5].

Le chapitre indique aussi qu'un partitionnement spatial explicite sous Spark SQL a jusqu'à présent posé plus de difficultés qu'il n'en résolvait [3.2.3.1], si bien qu'une approche plus implicite est envisagée par le modèle proposé [3.2.4.2]. Il montre aussi qu'une gestion fine de la persistance depuis Parquet aurait pu encore diminuer les durées d'exécution des requêtes spatiales indexées par GeoHash [3.4]. Enfin, il évoque à la fois la complétude relative et les limites du banc de test employé et indique comment il pourrait être étendu avec fruit [3.6].

Le chapitre suivant se situe dans une optique plus généraliste, puisqu'il vise à ajouter au système Elcano les extensions de la norme ISO-19125 qui semblent les plus pertinentes.

Chapitre 4

Vers un modèle unifié de la composante spatiale vectorielle

4.1 Introduction

Ce chapitre vise à élargir la gamme des géométries traitées par le moteur de gestion des données spatiales massives vectorielles Elcano, dont le modèle a été présenté dans le chapitre 2 [2.2.3.2] et enrichi d'un système d'indexation spatiale dans le chapitre 3 [3.2.4.2]. Il s'agit ici plus spécifiquement d'étendre son modèle au-delà de la norme ISO-19125, de façon à l'ouvrir à d'autres traitements que celui des géométries 2D constructibles par interpolation linéaire.

Le corps de ce chapitre est constitué d'un article, qui commence [4.2.1] par évoquer l'importance des données massives à référence spatiale [cf. 1.1] et par souligner le manque de conformité des systèmes de traitement spatial existants [cf. 1.3.4] quant au respect des standards établis par l'ISO et l'OGC pour décrire la composante spatiale. Il parcourt ensuite [4.2.2] ces standards en insistant sur les formats vectoriels linéaires qui étendent ISO-19125 [cf. 1.2.1.2]. Une troisième section [4.2.3] vise à isoler les pratiques les plus courantes en étudiant comment ces normes sont adaptées et implémentées par quelques systèmes et formats dédiés aux données spatiales [cf. 1.2.2]. Dans une quatrième section [4.2.3.1], les principales extensions de la norme ISO-19125 identifiées lors de ce parcours sont discutées et certaines d'entre elles sont choisies pour intégrer un modèle unifié de la composante spatiale, en fonction notamment de leur utilité dans un contexte géographique. Un nouveau modèle du gestionnaire de géométries d'Elcano [cf. 2.2.3, 3.2.4.2] intégrant ces extensions est finalement présenté [4.2.4].

4.1.1 Contributions

[44] Engélinus, J. et Badard, T., 2017, « Vers un modèle unifié de la composante spatiale vectorielle et son application à Elcano », à traduire en anglais pour soumission dans Journal of GIS.

4.2 Corps de l'article

Titre : Vers un modèle unifié de la composante spatiale vectorielle et son application à Elcano

Auteurs : Jonathan Engélinus, Thierry Badard

Résumé : Les données massives se situent au cœur de beaucoup d'enjeux scientifiques et sociétaux, et leur volume global ne cesse de croître. Mais si beaucoup d'entre elles comportent une composante spatiale, peu de systèmes sont à même de gérer efficacement celle-ci. En outre, aucune solution ne respecte entièrement les standards ISO et les spécifications OGC qui décrivent la façon de gérer la composante spatiale. Le système Elcano conçu par le Centre de Recherche en Géomatique de l'Université Laval s'approche d'un tel objectif, puisqu'il gère de façon efficiente toutes les géométries et fonctions d'analyse spatiale décrites par la norme ISO-19125. Mais il n'en reste pas moins largement perfectible : la norme sur laquelle il repose a plus de dix ans et se limite aux données spatiales vectorielles bidimensionnelles, alors que d'autres structures comme les géométries en 2.5D et les TIN

sont aujourd'hui employées couramment. Le présent article définit un modèle unifié de la composante spatiale qui intègre certaines de ces nouvelles structures au système Elcano.

Abstract: Big data are in the midst of many scientific and economic issues. Furthermore their volume is continuously increasing. As a result, the need for management and processing solutions has become critical. Unfortunately, while most of these data have a vectorial spatial component, almost none of the current systems are able to manage it. In addition, the few systems who try do not respect the ISO standards and OGC specifications. The Elcano system developed by the Laval University Geomatics Research Center is approaching this objective. Indeed, it efficiently manages all geometries and spatial analysis functions described in ISO-19125. But this ten-years-old standard is limited to two-dimensional vectorial spatial data whereas other structures like 2.5D geometries and TIN are today widely used. This paper defines a unified model of the spatial component that integrates many of these new structures into Elcano.

Mots-clés : Elcano, ISO, OGC, 2.5D, TIN

Keywords: Elcano, ISO, OGC, 2.5D, TIN

4.2.1 Introduction

La rencontre de la cartographie et d'Internet a entrainé celle-ci vers un nouveau paradigme : la néogéographie. Celle-ci est marquée par « une interactivité de plus en plus importante des outils cartographiques, ainsi que par la génération collaborative de contenus géolocalisés par les utilisateurs d'Internet » [89]. L'arrivée récente sur le marché de nouveaux capteurs comme les puces GPS des téléphones intelligents est encore venu renforcer ce changement profond [8], qui s'accompagne d'une inflation de la production et de la consultation de données géoréférencées. Il s'en suit un regain d'intérêt envers la cartographie, mais aussi une complexification importante de celle-ci.

La gestion d'une telle quantité de données entraine en effet des difficultés conceptuelles et techniques importantes, qui amènent à abandonner peu à peu les systèmes de gestion classiques [46] pour se tourner vers des systèmes spécialisés dans la gestion des données massives. Parmi ceux-ci, l'écosystème Hadoop [120] maintenu par la Fondation Apache est devenu au fil des années un standard de fait doté de nombreuses extensions. Le plus actif des éléments de cet écosystème est actuellement Spark, qui présente d'excellentes performances en raison de sa limitation des accès disque [128, 124]. Cependant, bien que 80% des données d'entreprise [51] et plus de 50% des données échangées sur Twitter soient associés à une localisation, il n'existe que très peu de prototypes reposant sur Hadoop ou Spark à même de traiter efficacement celle-ci [8, 9].

L'inflation de la quantité de données spatiales échangées sur Internet rend également difficile leur normalisation. Les nombreux outils cartographiques disponibles en ligne ne seront pourtant à même d'échanger facilement des informations que si des conventions universelles leur permettent d'employer un langage commun. Afin de combler les besoins de standards cartographiques, un consortium des principaux acteurs du domaine s'est formé à partir de 1994 : l'Open Geospatial Consortium (OGC). Cette organisation a progressivement établi des recommandations techniques permettant de conceptualiser les entités géographiques, d'effectuer sur eux des traitements spatiaux, et de les échanger de façon interopérable. A un niveau plus officiel, cette tendance a été accompagnée par l'Organisation Internationale de Normalisation (ISO), qui a constitué la même année le comité ISO/TC 211 en charge de standardiser l'information géographique. Malgré l'ancienneté et le grand nombre d'applications qu'ont connu les standards de ces organismes, ils peinent aujourd'hui à être adaptés aux environnements de gestion des données massives. Ainsi, la norme ISO-19125, malgré sa simplicité, n'était jusqu'à nos travaux intégrée à aucun système reposant sur Spark.

Afin d'apporter un début de solution à cette double problématique, nous décrivons dans d'autres articles le modèle du système Elcano [42, 43, 45], qui permet de traiter des données massives à référence spatiale (voir figure 4.1). Il a été conçu afin d'atteindre quatre objectifs : gérer tous les types de données spatiales 2D élémentaires (cf. package « geometry » du modèle, qui repose sur la bibliothèque spatiale JTS [1]), permettre de leur appliquer les fonctions spatiales décrites par la norme ISO-19125 à l'aide de requêtes SQL (cf. package loader du modèle, qui charge les fonctions au démarrage du

1. https://github.com/locationtech/jts

système), gérer leur importation de façon générique et les persister efficacement (cf. classe Table, qui charge des données au format WKT [2] et les persiste au format WKB [3]). Nous complétons le modèle d'Elcano par un module d'indexation spatiale dédié aux données massives efficace pour tous les types de géométries et gérant le partitionnement [45], mais cette extension ne sera pas détaillée car elle présente peu de liens avec la problématique évoquée ici et s'ajoute en tant que surcouche au modèle de base sans l'altérer. Un système qui repose sur la composante Spark SQL [6] de Spark et permet la gestion performante de données spatiales massives a été implémenté afin de tester ces modèles.

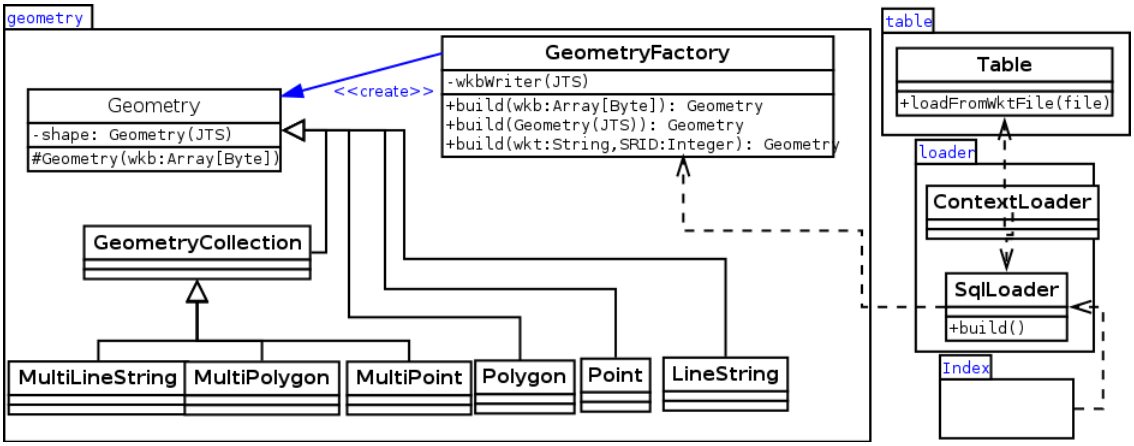

FIGURE 4.1 – Modèle d'Elcano étendu à la gestion de la 2.5, des TIN et du datum.

L'application du standard ISO-19125 à Elcano permet qu'il gère tous les types de géométries 2D élémentaires et toutes les fonctions SQL de traitement des données à références spatiales, ce qui suffit à le rendre plus complet que les solutions existantes [43]. Cependant, cette norme n'est pas sans présenter certaines limitations. Elle ne permet ainsi que la prise en compte de géométries 2D, ce qui ne lui permet pas de représenter l'élévation. Celle-ci constitue pourtant un paramètre dont le potentiel économique se chiffre en milliards de dollars [22], et sans lequel la modélisation d'environnements urbains tiendrait de l'utopie [32]. A fortiori, la norme ISO-19125 ne permet pas de modéliser un relief en le couvrant d'un réseau de triangles irréguliers (TIN) [116], bien que cette technique soit à la base de la construction des modèles numériques de terrain vectoriels utilisés en modélisation. Cette norme ne permet pas non plus de gérer la topologie, le format matriciel ou le changement de datum, malgré leur potentiel qui sera détaillé dans une section ultérieure.

L'objectif du présent article est de proposer un nouveau modèle d'Elcano qui dépasse ces limitations. Il vise donc d'abord à établir une synthèse cohérente des approches actuelles en termes de normalisation et de spécification de la composante spatiale. A cette fin, il commence par évoquer les principaux apports de la norme ISO-19125 et d'autres spécifications OGC et normes ISO décrivant des données

2. WKT est un acronyme de « well-know text ». Il permet représenter les géométries dans un format textuel.
3. WKB est un acronyme de « Well-know binary ». Ce format binaire permet de stocker sous une forme très compacte une géométrie ainsi que l'identifiant de la projection spatiale qu'elle emploie.

à composante spatiale constructibles par interpolation linéaire [4]. Puis il s'intéresse aux implémentations de ces normes et spécifications par différents systèmes, aussi qu'aux fonctionnalités que ceux-ci ajoutent par rapport aux standards ISO et OGC. Finalement, une troisième section décrit l'intégration d'un modèle unifié de la composante spatiale à Elcano.

4.2.2 Standards et spécifications de la composante spatiale

Cette section présente les normes les plus importantes définies par l'ISO et l'OGC en vue de représenter des géométries constructibles par interpolation linéaire. L'emphase est mise sur la modélisation des données spatiales vectorielles, car la sémantique qu'elle associe aux phénomène spatiaux la rend plus adaptée à leur analyse. Mais les données matricielles sont également abordées, en raison de la généricité de leur format et du grand nombre d'applications qu'elles permettent.

4.2.2.1 Normes ISO-19125

L'ISO a publié en 2004 la norme ISO-19125, qui décrit les géométries élémentaires pouvant servir à représenter et localiser les objets spatiaux liés à une zone géographique ainsi que les traitement SQL minimaux que doit offrir un système gérant ces données. Elle est constituée de deux volets. Le premier (ISO-19125-1) décrit les types d'objets spatiaux élémentaires, ainsi que les méthodes qui peuvent s'y appliquer. Plus spécifique, le second (ISO-19125-2) décrit les conventions que la norme devrait respecter en cas d'implémentation dans un système de gestion de base de données relationnelles utilisant la langage SQL.

Plus précisément, le volet ISO-19125-1 de la norme définit sept sortes de géométries 2D constructibles par interpolation linéaire sur un plan en deux dimensions. Celles-ci se répartissent en trois types simples (le point, la polyligne et le polygone) et en quatre types composés construits par le rassemblement de géométries simples (le multi-point, la multi-polyligne, le multi-polygone et la collection de géométrie). A ces géométries peuvent s'appliquer différentes fonctions spatiales, qui se déclinent en méthodes, en relations et en opérations ainsi qu'en une fonction métrique. Les méthodes fournissent une information à propos de la géométrie à laquelle elles s'appliquent, tandis que les relations renvoient un booléen qui indique si deux géométries sont liées par une des relations spatiales décrites par le modèle DE 9-IM établi par Clementini et Laurini [19] à partir des travaux de Egenhofer et al. [39, 37]. Les opérations spatiales permettent quant à elles de créer une nouvelle géométrie à partir de deux géométries reçues en paramètres, en leur appliquant une opération ensembliste. Enfin, la fonction métrique disponible permet de connaitre la distance en mètres entre deux géométries.

Le volet ISO-19125-2 de la norme traite pour sa part de l'implémentation SQL de ces différentes fonctions et fournit les règles de nommage à appliquer dans ce contexte. Ce second volet ne se veut

4. Les géométries reposant sur d'autres principes que l'interpolation linéaire (vraie 3D, représentations courbes, prise en compte du temps...) ne seront pas détaillées ici, mais le lecteur intéressé par cette problématique pourra consulter avec fruit la revue établie par Van Oosterom et Stoter [117]. S'il s'intéresse à la vraie 3D, il pourra également se documenter sur la norme OGC CityGML [71], qui permet de représenter des objets 3D usuels dans un environnement urbain.

par contre pas prescriptif quant aux technologies à employer, si bien qu'il est possible de l'appliquer aussi bien à un système de gestion des données classique qu'à un système de gestion des données massives du moment que la référence spatiale y est gérée. Il décrit également les formats WKT et WKB, qui permettent respectivement de représenter les géométries 2D sous forme textuelle (WKT) et de les sérialiser dans un format binaire (WKB).

4.2.2.2 Norme ISO-19107

Cette norme a été publiée par l'ISO en 2003, puis en 2005 dans sa version définitive. Elle décrit les différentes géométries vectorielles pouvant évoluer dans un espace en trois dimensions ainsi que les relations susceptibles de les lier. Décrivant aussi bien des géométries constructibles par interpolation linéaire que des courbes, elle s'intéresse aussi aux TIN [5] et aux transformations du système de coordonnées, ce qui la rend bien plus générale qu'ISO-19125. Cette norme joue un rôle fondateur, puisque les autres standards gérant la composante spatiale vectorielle se bornent à décrire une sélection des objets qu'elle propose. ISO-19125 constitue la partie 2D de cette norme, compatible avec celle-ci.

4.2.2.3 Norme ISO-13249-3

La norme ISO-13249-3 a donné lieu à une publication de l'ISO en 2003, 2006, 2011 et 2016. Plus générale qu'ISO-19125 mais plus concrète et spécifique qu'ISO-19107, elle décrit des géométries tridimensionnelles ou constructibles à l'aide de courbes. Elle permet en particulier d'ajouter un troisième paramètre marquant les coordonnées spatiales d'un point, de passer d'un datum géographique à un autre, et d'utiliser des géométries tridimensionnelles. Cette norme étend aussi le format WKT initialement décrit par la norme ISO-19125, pour lui permettre de représenter des géométries 2.5D, 3D, ou constituées de courbes.

4.2.2.4 Recommandations Simple Features et Simple Features SQL

Les recommandations **Simple Features** et **Simple Features SQL** établies par l'OGC correspondent en grande partie aux normes ISO-19125-1 et ISO-19125-2 [69]. Leur volet SQL a cependant connu davantage de versions : 1999, 2005, 2006 et 2010. A partir de 2006, les fonctions décrites ont une signature identique à celles de la norme ISO-19125-2, mais avec une portée un peu plus générale. En effet, la recommandation Simple Features SQL s'inspire de la norme ISO-13249-3 pour permettre l'ajout d'une troisième coordonnée permettant d'indiquer l'élévation (2.5D). Elle évoque également l'ajout de géométries comme les polyèdres [6] et les TIN [7].

4.2.2.5 Norme ISO-19123

La norme ISO-19123 définie par l'ISO en 2005 décrit des géométries spatiales telles qu'elles couvrent entièrement la surface considérée. Il ne s'agit donc plus seulement de polygones, de points ou de multi-

5. TIN est un acronyme de « Triangulated Irregular Network ». La section suivante en fournit une description détaillée.
6. Il s'agit de géométries en trois dimensions dont chaque face est un polygone.
7. La section suivante fournit une description détaillée des TIN

lignes, mais bien d'un maillage qui recouvre l'ensemble d'une zone géographique. Dans le cas d'un recouvrement régulier, le même motif est reproduit sur l'ensemble de la surface, tandis qu'il varie dans le cas d'un maillage irrégulier. Cette section décrit l'implémentation le plus courante pour chacune de ces possibilités.

Grille La grille appartient à la catégorie des recouvrements réguliers. Elle consiste à diviser la zone couverte en carrés de même dimension appelés pixels, qui partagent entre eux leurs arrêtes. Dès lors, il s'agit d'une représentation matricielle du territoire et non d'une représentation vectorielle, car les géométries utilisées ne correspondent en aucune façon aux entités géographiques du territoire. Ce mode de découpe simplifie le traitement des données géographiques, et connaît des applications dans des domaines aussi variés que la sismographie [7] et le traitement d'images satellite [34]. Mais il ne permet pas de localiser ou d'interroger directement les objets représentés, ce qui le rend moins adapté à des fins d'analyse spatiale que le format vectoriel.

TIN La méthode du réseau de triangles irréguliers (TIN) consiste à couvrir une surface tridimensionnelle à l'aide de triangles de tailles différentes partageant leurs arrêtes. Ces triangles sont souvent construits par triangulation de Delaunay [29], c'est-à-dire en s'interdisant de situer un sommet à l'intérieur du cercle inscrit d'un autre triangle. Il s'agit en effet d'une méthode qui évite de donner aux triangles des formes trop allongées qui nuiraient à la précision des représentations. Contrairement aux grilles, les TIN sont des structures vectorielles, car la position et la hauteur des sommets des triangles varient en fonction de données du terrain comme l'élévation. Ces points peuvent par exemple servir à représenter les sommets et les puits d'une zone montagneuse. Ils jouent donc un rôle clé dans l'élaboration de modèles numériques de terrain à même de représenter précisément les ruptures de pentes [109].

4.2.3 Extensions et adaptations existantes

S'il est exceptionnel de les voir intégrées par un système de gestion des données massives, les normes évoquées précédemment ont connu un grand nombre d'applications à la gestion de données ordinaires. La présente section présente quelques-unes de ces implémentations, ainsi que des extensions ou adaptations qu'elles proposent par rapport au standard initial.

JTS [8] est une solution open source développée à partir de la fin des années 1990 pour les besoins du gouvernement de la Colombie Britannique, repris par Vivid Solutions puis devenue plus récemment un projet Eclipse. Ses classes fournissent des fonctionnalités rencontrant la norme ISO-19125, mais sont également capables de gérer l'élévation, les TIN, l'indexation et le changement de système de coordonnées. Sa gestion du WKT est cependant plus proche de la norme ISO-19125-2 que de la norme ISO-13249-3, ce qui peut nuire à la représentation de certaines géométries et n'en fait pas toujours un outil standard. Le tableau 4.1 montre ainsi que JTS suit une syntaxe WKT différente de celle du

8. https://github.com/locationtech/jts

standard ISO-13249-3 pour gérer les géométries 2.5D 4.1. Par contre, le format WKB employé est plus proche des formats définis par ISO-13249-3.

Géométrie	Format(s)	Syntaxe
Point 2D	Tous	POINT(0 1)
Point 2.5D	JTS, PostGIS	POINT(0 1 10)
Point 2.5D	ISO-13249-3	POINT Z (0 1 10)
Point 2.5D	ISO-19125-2	Impossible.

TABLE 4.1 – Déclaration d'un point 2D et d'un point 2.5D selon différents formats

PostGIS [94] s'appuie sur un portage de JTS en langage C++ appelé GEOS [9] (Geometry Engine Open Source). Il intègre ses méthodes au contexte d'une base de données relationnelle classique, en rendant disponibles dans PostgreSQL chacune des fonctions spatiales définies par la norme ISO-19125, ainsi que la prise en compte d'une troisième dimension indiquant l'élévation (2.5D). Il permet également de modifier la projection de ces données. A partir de sa version 2, il gère également les surfaces polyédriques et les TIN [100] décrites par les normes ISO-13249-3 et ISO-19123. PostGIS est également fourni avec un grand nombre de fonctions d'analyse spatiale vectorielle et matricielle, comme l'extraction du polygone de couverture correspondant à un groupe de points, la recherche des N géométries les plus proches d'une géométrie et le calcul de profil d'élévation [10] [101]. Il permet également de gérer les formats EWKT et EWKB, qui constituent des extensions artisanales des formats WKB et WKT d'ISO-19125. Ceux-ci peuvent gérer le système de projection employé par une géométrie (EWKT le reçoit en paramètre de ses fonctions, tandis qu'EWKB le persiste avec la géométrie). Le tableau 4.2 fournit quelques exemples de géométries que le format EWKT permet de déclarer alors que le format WKT défini par la norme ISO-19125 ne le permet pas [11].

Géométrie	Expression EWKT
Point 2.5 D	POINT(0 1 2)
Point 2D avec un troisième paramètre M indiquant une mesure	POINTM(0 0 0)
Triangle 2D	TRIANGLE((0 0, 0 9, 9 0, 0 0))
Réseau TIN, constitué de triangles 2.5D	TIN(((0 0 0, 0 0 1, 0 1 0, 0 0 0)), ((0 0 0, 0 1 0, 1 1 0, 0 0 0)))

TABLE 4.2 – Exemple d'expressions EWKT.

GeoJSON [13] est une extension du langage JSON [12] aux données géographiques, qui permet de représenter celles-ci d'une façon concise et facilement interopérable. La recommandation RFC 7946

9. https://trac.osgeo.org/geos/

10. Une étude plus détaillée de ces fonctions dépasserait la visée du présent article, étant donné leur richesse, leur variété et leur format spécifique à PostGIS.

11. Ces exemples sont tirés de la documentation en ligne de PostGIS : http://postgis.refractions.net/docs/using_postgis_dbmanagement.html

décrit précisément [14] ce format, qui permet de déclarer les sept types de géométries vectorielles décrits par la norme ISO-19125, ainsi que deux structures propres appelées « Feature » et « FeatureCollection ». La première réunit une géométrie et des données attributaires dans un même objet JSON, tandis que la seconde rassemble une liste de « Features ». GeoJSON offre de plus la possibilité d'attribuer une élévation aux données spatiales, ce qui pourrait permettre d'y représenter des TIN même si ce type n'y est pas directement défini [102].

TopoJSON [11] est une extension de GeoJSON qui représente les données sous une forme purement topologique. Assez ancien [38, 18], ce mode de représentation a permis aux premiers systèmes d'information géographiques de réduire l'utilisation de l'espace disque en sauvant les points, les arrêtes et les polygones constituant les géométries dans des tables différentes de façon à en limiter la redondance. C'est donc un format qui limite par factorisation les répétitions d'arcs d'une géométrie à l'autre, si bien qu'il réduit l'espace de stockage nécessaire. Par exemple, il permet de n'utiliser que six points au lieu de huit pour représenter deux carrés partageant une arrête. Actuellement, il reste intéressant pour certaines applications comme la diffusion d'informations géographiques sur le web, puisque son format permet des messages plus compacts que GeoJSON.

La section suivante récapitule les extensions identifiées dans les deux sections précédentes, en vue de sélectionner celles qui seront retenues pour la conception du modèle unifié de la composante spatiale envisagé. Comme annoncé dans l'introduction, elle se limite cependant aux extensions qui impliquent des données constructibles par interpolation linéaire, ce qui exclue les géométries courbes et les fonctions d'analyse spatiale avancées bien qu'elles soient permises par certaines des normes et spécifications évoquées.

4.2.3.1 Choix des extensions à intégrer au modèle

Le tableau 4.3 présente les sept principales extensions que les normes, spécifications et applications étudiées jusqu'à présent ont apporté comparativement à la norme ISO-19125. Ces extensions sont ensuite discutées afin de déterminer lesquelles intégreront l'extension d'Elcano envisagée.

Extension	**Apportée par**	**Reprise par**
Vraie topologie	premiers logiciels SIG	TopoJSON
3D polyédrique	ISO-13249-3, Simple Features et Simple Features SQL	PostGIS
Format matriciel	ISO-19123	PostGIS, JTS
2.5D et **paramètre M**	ISO-13249-3, Simple Features et Simple Features SQL	PostGIS, JTS, GeoJSON, TopoJSON
TIN	ISO-19123, Simple Features et Simple Features SQL	PostGIS, JTS
Gestion du datum	ISO-13249-3	JTS, PostGIS

TABLE 4.3 – Extensions potentielles identifiées pour Elcano.

Vraie topologie A première vue, une organisation topologique des données spatiales prendrait tout son sens dans un contexte de données massives, en raison de l'économie de mémoire et de la réduction des temps de transfert qu'elle semble permettre. Mais elle nous a semblé difficilement intégrable à un gestionnaire de données spatiales massives qui comme Elcano organise les données spatiales sous forme de tables SQL. En effet, la concision due à l'approche topologique vient du fait qu'elle fragmente les données spatiales dans trois tables (les points, les arcs et les polygones) afin d'en limiter les redondances. Il faut donc ensuite relier ces tables pour reconstituer les géométries auxquelles des requêtes d'analyse spatiale sont appliquées. Or, une jointure entre tables dans un contexte de données massives peut s'avérer fort coûteuse, car les données jointes sont généralement dispersées sur plusieurs serveurs. Il ne semble donc pas tenable de provoquer la jonction de trois tables massives lors de la consultation SQL de chaque polygone dans Elcano, si bien que l'approche topologique ne sera pas incluse dans l'extension envisagée.

3D polyédrique En lui permettant de manipuler des polyèdres, il serait possible d'ouvrir Elcano à un début de gestion de la 3D. Ceci pourrait entrainer des applications intéressantes, dans des domaines comme la modélisation d'environnements urbains[108] ou l'océanographie [21]. Mais la complexité conceptuelle et technique induite serait telle qu'elle impliquerait une redéfinition complète du modèle actuel. Par exemple, le module d'indexation spatiale d'Elcano [45] cesserait d'être valide car il n'a été conçu qu'en vue d'un contexte géographique standard. La 3D ne sera donc pas incluse dans l'extension envisagée, mais son intégration pourrait être envisagée dans le cadre de travaux plus approfondis.

Format matriciel Il serait techniquement possible d'intégrer également le format matriciel à Elcano, par exemple à l'aide de JTS [12] ou d'une bibliothèque plus spécifique comme JAITools [13]. Cependant ce format est également d'une complexité telle qu'il demanderait une étude spécifique, tout en s'avérant moins approprié à des fins d'analyse spatiale que le format matriciel en raison de sa sémantique plus pauvre. Il ne sera donc pas non plus intégré à l'extension envisagée.

2.5D Le format 2.5D sera par contre inclus dans l'extension d'Elcano, étant donné qu'il permet de gérer l'élévation sans induire les mêmes complexités que la 3D et sans quitter le cadre d'une situation géographique standard. Chacun des points des géométries 2.5D restent en effet liés au sol, ce qui les rend conceptuellement plus proches de géométries 2D que de véritables polyèdres. Cette possibilité de traiter des données d'élévation depuis Elcano ouvre celui-ci à de nombreuses applications [22, 32], ainsi qu'à la possibilité de gérer des TIN.

Paramètre M La gestion des géométries intégrant un paramètre M indiquant une mesure, telle qu'elle est permise par le format EWKT, semble structurellement proche d'une gestion de la 2.5D et pourrait également être intégrée au modèle. Mais elle semble peu utile dans le cas d'Elcano : comme c'est une information qui n'aide pas directement à la géolocalisation, elle pourrait tout autant

12. http://docs.geotools.org/latest/userguide/extension/grid.html
13. http://jaitools.org/

se trouver dans une colonne attributaire de la table reçue par le système. Les géométries accompagnées d'une mesure ne seront donc pas ajoutées à l'extension envisagée. Leur emploi pourrait par contre aider à une gestion de la composante temporelle par un modèle plus étendu, dans laquelle la mesure transportée par la géométrie indiquerait par exemple une date d'acquisition ou une durée de vie.

TIN Les TIN seront également intégrés à l'extension du modèle de base d'Eclano envisagée, étant donné leur simplicité structurelle et leurs puissantes applications. Ils viennent en effet compléter la gestion de l'élévation évoquée ci-dessus, en permettant d'élaboration de véritables modèles numériques de terrain vectoriels.

Gestion du datum La possibilité d'indiquer le système de coordonnées d'une géométrie et d'éventuellement le transformer sera également retenue en tant qu'extension d'Elcano. Elle peut en effet faciliter l'intégration de données hétérogènes en les ramenant à un système de coordonnées commun avant leur traitement.

Les trois extensions retenues pour étendre le modèle d'Elcano sont donc la gestion de l'élévation (2.5D), celle des TIN, et celle des transformations de systèmes de coordonnées. Le nouveau modèle d'Elcano résultant de cette intégration est détaillé dans la section suivante.

4.2.4 Extension d'Elcano vers un modèle unifié de la composante spatiale

Cette section indique les objectifs qui ont guidé l'extension du modèle du système de gestion des données spatiales massives Elcano à d'autres cas que ceux permis par la norme ISO-19125. Elle présente et justifie ensuite le modèle résultant et donne un aperçu de son implémentation.

4.2.4.1 Objectifs de conception

Un premier objectif consiste à intégrer à Elcano les extensions à la norme ISO-19125 qui ont été sélectionnées dans la section précédente : une gestion de la 2.5D, des TIN, et du datum des géométries. Un second objectif consiste à rendre ces éléments interrogeables à l'aide de requêtes SQL, afin de faciliter leur emploi à des fins d'analyse spatiale et de s'inscrire en prolongement des objectifs qui guidant la conception du premier modèle d'Elcano.

Le modèle étendu en vue de rencontrer ces objectifs est présenté et justifié ci-dessous.

4.2.4.2 Architecture

La figure 4.2 présente le modèle nouveau modèle d'Elcano une fois étendu par-delà la norme ISO-19125.

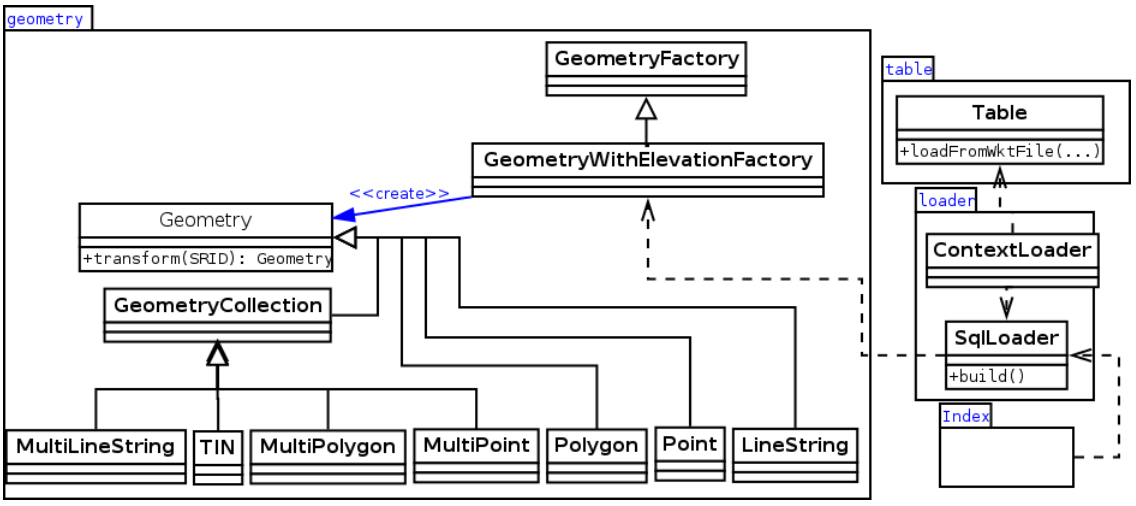

FIGURE 4.2 – Extension du modèle d'Elcano à d'autres types de géométries.

Dans ce modèle, la classe GeometryWithElevationFactory étend Elcano aux géométries 2.5D, la classe TIN l'étend aux géométries éponymes et la méthode transform() de la classe Geometry permet les changements de systèmes de coordonnées. Une version modifiée de la classe SqlLoader permet quant à elle de rendre ces extensions consultables en SQL.

La façon dont sont atteints ces différents objectifs est décrite et justifiée ci-dessous.

Gestion de l'élévation Une façon d'étendre le modèle à une gestion de l'élévation aurait pu consister à dupliquer toutes ses géométries afin de leur adjoindre un équivalent 2.5D. Mais une technique plus économique consiste comme ici à étendre la « fabrique abstraite » GeometryFactory [118] utilisée pour instancier toutes les géométries, par une héritière capable de créer aussi des géométries 2.5D. Les mêmes classes sont donc utilisées pour gérer des géométries 2.5D et la 2D, suivant le paramétrage reçu à leur instanciation.

Gestion des TIN Étant donné qu'ils sont représentables par des listes de triangles ou de points, les TIN sont ici déclarés en tant que nouveau type qui hérite de GeometryCollection. Ils auraient pu aussi hériter de la classe MultiPolygon gérant les collections de polygone, ou de la classe MultiPoint gérant les collections de points, mais étant donné qu'ils peuvent être représentés par chacun de ces formats sans perte d'information, il semblait plus approprié qu'ils héritent d'un type plus général susceptible de gérer aussi bien les points que les polygones.

Gestion du datum Afin d'intégrer une gestion des systèmes de coordonnées géographiques à Elcano, l'approche choisie ici consiste à permettre la transformation des données d'un système de coordonnées vers un autre (via la méthode transform() de la classe Geometry), afin de pouvoir facilement relier des données de sources différentes. Une autre approche consisterait à ajouter aux fonctions spatiales un paramètre indiquant le SRID, comme le fait le format EWKT de PostGIS. Mais celle-

ci permettrait seulement d'indiquer le datum actuel des données, pas de les convertir à un nouveau. Contrairement à l'approche proposée ici, elle ne suffirait donc pas en tant que telle à préparer l'export des données acquises depuis des sources différentes vers une même table ou un même fichier.

Accessibilité depuis SQL Afin de rendre les trois extensions d'Elcano définies ci-dessus lisibles depuis des requêtes SQL, trois fonctions SQL sont ajoutées au système en complément de celles définies par la norme ISO-19125-2 (voir tableau 4.4). La dénomination de celles-ci respecte le standard ISO-13249-3, étant donné qu'il s'agit d'une des normes spatiales les plus générales et les plus employées parmi celles étudiées ici. Elle permet en outre de conserver une cohérence par rapport à la version de base d'Elcano, puisqu'elle intègre les fonctions spatiales définies par ISO-19125-2. Les méthodes SQL permettant de créer des géométries en SQL depuis Elcano ont quant à elles été étendues à un troisième paramètre optionnel, de façon à pouvoir gérer la 2.5D lors de leur création.

Fonction	**Paramètre(s)**	**Retour**
ST_Z	Point en 2.5D	Accès à la troisième coordonnée du point.
ST_TIN	Multi-point en 2.5D	Collection des triangles de Delaunay (TIN) correspondant au multi-point.
ST_Transform	Géométrie, SRID cible	Géométrie convertie dans le datum indiqué par le SRID cible.

TABLE 4.4 – Méthodes SQL ajoutées à Elcano.

De portée plus pratique, la section suivante évoque l'implémentation mise en œuvre pour intégrer les extensions évoquées.

4.2.4.3 Implémentation

Cette section apporte quelques précisions sur la façon dont les nouveaux types géométriques choisis ont été implémentés dans Elcano. Elles ont quelquefois consisté en des simplifications par rapport au modèle conceptuel décrit.

Bibliothèque spatiale employée La version de base d'Elcano repose sur la bibliothèque spatiale JTS, car celle-ci s'avère à notre connaissance la seule à appliquer efficacement les standards de l'ISO [43, 45]. L'ajout d'une troisième coordonnée, d'une gestion des TIN et d'un système de transformation du datum dans Elcano s'inscrit dans cette continuité en étant également gérable à l'aide de cette bibliothèque.

Gestion de l'élévation par Elcano Comme indiqué plus haut, l'implémentation des différentes extensions par Elcano repose sur JTS. La gestion de l'élévation passe dans celui-ci par l'emploi d'un paramètre indiquant le nombre de dimensions du système de coordonnées (ici 3) aux classes utilisées pour gérer des géométries en WKT et en WKB, comme le montre l'exemple de code de la figure 4.3. Le code Scala présenté en figure 4.4 montre quant à lui qu'employer un tel paramètre n'empêche pas

ensuite une gestion correcte de géométries à deux dimension. Il se comporte donc comme un majorant quant au nombre maximal de dimensions gérable par les classes de JTS plutôt que comme une contrainte. L'implémentation d'une gestion de l'élévation dans Elcano peut donc se limiter au passage d'un paramètre de valeur 3 aux classes adéquates de JTS et à l'ajout à la classe Point d'une méthode accédant à la troisième coordonnée de cette géométrie, sans nécessiter d'extension de la fabrique abstraite employée. Le modèle d'implantation qui en résulte est présenté par la figure 4.5.

```
import com.vividsolutions.jts.io._
val dimCoordinate = 3
val readerWkt = new WKTReader()
val writerWkt = new WKTWriter(dimCoordinate)
val readerWkb = new WKBReader()
val writerWkb = new WKBWriter(dimCoordinate)
val point = readerWkt.read("POINT (1 0 2)")
println(point.getCoordinate().z) // output : 2
println(writerWkt.write(point)) // output : POINT (1 0 2)
val wkb = writerWkb.write(point)
val point2 = readerWkb.read(wkb)
println(writerWkt.write(point2)) // output : POINT (1 0 2)
```

FIGURE 4.3 – Gestion de l'élévation depuis les formats WKT et WKB de JTS.

```
import com.vividsolutions.jts.io._
val dimCoordinate = 3
val readerWkt = new WKTReader()
val writerWkt = new WKTWriter(dimCoordinate)
val readerWkb = new WKBReader()
val writerWkb = new WKBWriter(dimCoordinate)
val point = readerWkt.read("POINT (1 0)")
println(point.getCoordinate().z) // output : 2NaN
println(writerWkt.write(point)) // output : POINT (1 0)
val wkb = writerWkb.write(point)
val point2 = readerWkb.read(wkb)
println(writerWkt.write(point2)) // output : POINT (1 0)
```

FIGURE 4.4 – Gestion préservée d'une géométrie en 2 dimensions par JTS.

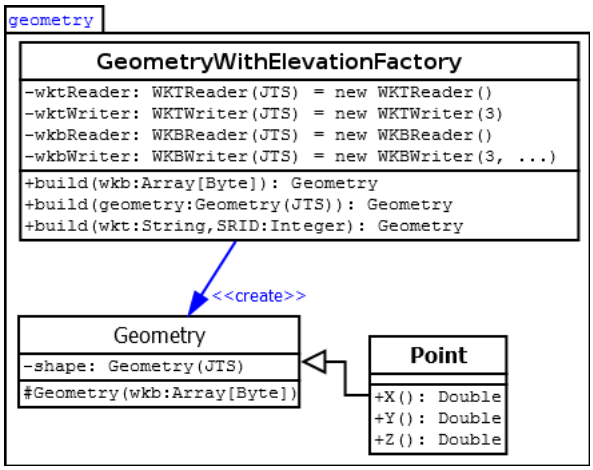

FIGURE 4.5 – Implantation de l'ajout d'une troisième coordonnée au modèle d'Elcano.

Gestion des TIN S'il permet de gérer une troisième coordonnée comme illustré précédemment, le format WKT défini par JTS ne gère pas les TIN. Néanmoins, JTS permet de gérer programmatiquement ce format à partir de la classe DelaunayTriangulationBuilder [14]. Celui-ci prend comme entrée un multi-point en 2.5D, et renvoie en sortie une collection de triangles respectant les principes de la triangulation de Delaunay. La création d'une classe TIN spécifique comme prévu par le modèle n'est donc pas nécessaire dans la pratique : il suffit d'ajouter à la classe MultiPoint une méthode qui appelle les méthodes JTS adéquates pour renvoyer la collection de triangles souhaitée. Le modèle d'implantation qui en résulte est présenté par la figure 4.6.

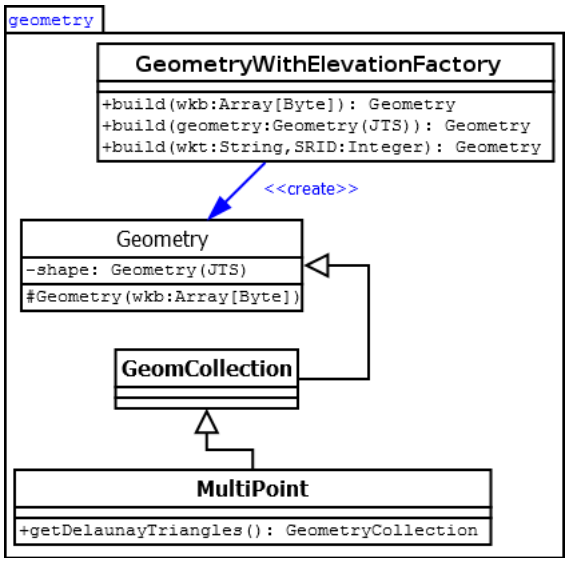

FIGURE 4.6 – Implantation d'une gestion des TIN dans le système Elcano.

14. http://tsusiatsoftware.net/jts/javadoc/com/vividsolutions/jts/triangulate/DelaunayTriangulationBuilder.html

Gestion des transformation du système de coordonnées Afin de pouvoir être interrogée depuis une requête SQL, la gestion du changement de datum a été ajoutée à la classe Geometry d'Elcano et rendue accessible depuis sa fonction SQL « ST_Transform » [15]. La figure 4.7 présente le modèle d'implantation qui en résulte.

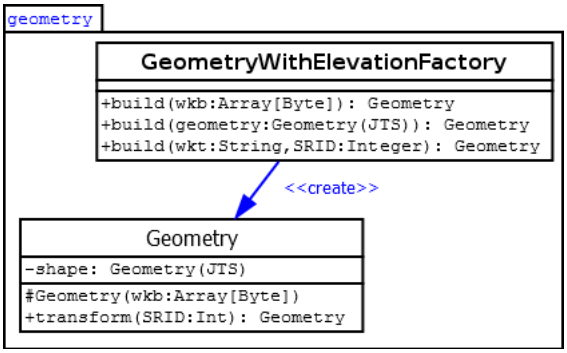

FIGURE 4.7 – Implantation d'une gestion des transformations de datum au modèle d'Elcano.

4.2.5 Conclusion

Cet article a commencé par un inventaire des normes ISO et OGC décrivant les données à référence spatiale constructibles par interpolation linéaire, puis les a intégrées à une ébauche de modèle unifié de la composante spatiale vectorielle gérant la 2.5D, les TIN et les transformations du système de coordonnées. Celui-ci consiste en une extension du système de gestion des données massives à référence spatiale Elcano, de façon à affranchir ce dernier des contraintes liées à la norme ISO-19125.

S'il atteint bien l'objectif visé, le modèle unifié de la composante spatiale qui en résulte reste largement extensible. Il pourrait ainsi être intéressant d'étudier son extension à une véritable gestion de la troisième dimension, qui ne se limiterait pas comme ici à la prise en compte d'une coordonnée d'élévation. Une base de travail intéressante pourrait alors être la revue établie de Van Oosterom et Stoter sur le sujet [117]. L'ajout d'une gestion du format matriciel serait également à considérer, car elle élargirait de beaucoup la portée du modèle et permettrait de l'appliquer à de nouvelles situations comme le traitement d'images satellites ou la modélisation numérique de terrain. A plus long terme, l'ajout d'une période d'existence aux géométries et aux fonctions spatiales gagnerait aussi à être envisagé, car une telle extension permettrait de gérer des données massives sous la forme d'événements spatio-temporels, et faciliterait peut-être la détection dynamique de « points chauds » [82] en leur sein.

15. Cette signature correspond à celle employée par ISO-13249-3 et PostGIS

4.2.5.1 Références

[6] M. ARMBRUST et al. “Spark sql : Relational data processing in spark”. In : *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 2015, p. 1383–1394.

[7] G. AYDIN et al. “Streaming Data Services to Support Archival and Real-Time Geographical Information System Grids”. In : *Sixth Annual NASA Earth Science Technology Conference*. 2006.

[8] T. BADARD. “Mettre le Big Data sur la carte : défis et avenues relatifs à l’exploitation de la localisation”. In : *Colloque ITIS - Big Data et Open Data au coeur de la ville intelligente*. (29 avr. 2014). Québec : Centre de recherche en géomatique, 2014.

[9] T. BADARD et J. ENGÉLINUS. “Elcano : un prototype de moteur Big Data spatial basé sur Spark”. In : *Colloque ITIS - Innovation et données massives : un panorama*. (31 oct. 2016). Québec : Centre de recherche en géomatique, 2016.

[11] M. BOSTOCK. *How to infer topology*. http://bost.ocks.org/mike/topology. Accessed : 2016-11-13. 2013.

[12] T. BRAY. *The javascript object notation (json) data interchange format (No. RFC 7158)*. 2014.

[13] H. BUTLER et al. “Geojson specification”. In : *Geojson.org* (2008).

[14] H. BUTLER et al. *RFC 7946 - The GeoJSON Format*. Rapp. tech. 2016.

[18] E. CLEMENTINI et P. DI FELICE. “A model for representing topological relationships between complex geometric features in spatial databases”. In : *Information sciences* 90.1 (1996), p. 121–136.

[19] E. CLEMENTINI et R. LAURINI. “Un cadre conceptuel pour modéliser les relations spatiales”. In : *Revue des Nouvelles Technologies de l’Information (RNTI)* 14 (2008), p. 1–17.

[21] P. CUGIER et P. LE HIR. “Development of a 3D hydrodynamic model for coastal ecosystem modelling. Application to the plume of the Seine River (France)”. In : *Estuarine, Coastal and Shelf Science* 55.5 (2002), p. 673–695.

[22] Belanger D. “La stratégie d’élévation nationale”. In : *Colloque Géomatique 2016*. (19 oct. 2016). Québec : ACSG, 2016.

[29] B. DELAUNAY. “Sur la sphere vide”. In : *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk* 7.793-800 (1934), p. 1–2.

[32] E. DEVYS et G. GESQUIÈRE. *Intégration des données et modèles urbains : standards, normes et tendances pour les SIG*.

[34] A. DREWS et al. “Monitoring and remote failure detection of grid-connected PV systems based on satellite observations”. In : *Solar Energy* 81.4 (2007), p. 548–564.

[37] M. J. EGENHOFER et R. D. FRANZOSA. “On the equivalence of topological relations”. In : *International journal of geographical information systems* 9.2 (1995), p. 133–152.

[38] M. J. EGENHOFER et Robert D FRANZOSA. “Point-set topological spatial relations”. In : *International Journal of Geographical Information System* 5.2 (1991), p. 161–174.

[39] M. J. EGENHOFER et J. HERRING. “Categorizing binary topological relations between regions, lines, and points in geographic databases”. In : *The* 9.94-1 (1990), p. 76.

[42] J. ENGELINUS et T. BADARD. “Towards a Real-Time Thematic Mapping System for Streaming Big Data”. In : *GIScience September 2016, Montreal*. 2016.

[43] J. ENGÉLINUS et T. BADARD. “Elcano : un système de gestion des données massives à composante spatiale basé sur Spark SQL », Revue internationale de géomatique”. In : *Revue internationale de géomatique (en cours de soumission)*. 2016.

[44] J. ENGÉLINUS et T. BADARD. “Vers un modèle unifié de la composante spatiale vectorielle et son application à Elcano”. In : *à traduire en anglais pour soumission dans Journal of GIS*. 2016.

[45] J. ENGÉLINUS et T. BADARD. “Vers une indexation des données spatiales massives performante et efficace”. In : *Geomatica (en cours de soumission)*. 2016.

[46] M. R. EVANS et al. “Spatial big data”. In : *Big Data : Techniques and Technologies in Geoinformatics* (2014), p. 149.

[51] C. FRANKLIN et P. HANE. “An Introduction to Geographic Information Systems : Linking Maps to Databases [and] Maps for the Rest of Us : Affordable and Fun.” In : *Database* 15.2 (1992), p. 12–15.

[69] J KERHERVÉ. “L’interopérabilité des systémes d’information géographique”. Thèse de doct. Haute école de gestion de Genève, 2008.

[71] Thomas H KOLBE. “Representing and exchanging 3D city models with CityGML”. In : *3D geo-information sciences*. Springer, 2009, p. 15–31.

[82] R. MACIEJEWSKI et al. “A visual analytics approach to understanding spatiotemporal hotspots”. In : *IEEE Transactions on Visualization and Computer Graphics* 16.2 (2010), p. 205–220.

[89] B. MERICSKAY et S. ROCHE. “Cartographie numérique en ligne nouvelle génération : impacts de la néogéographie et de l’information géographique volontaire sur la gestion urbaine participative”. In : *Nouvelles cartographie, nouvelles villes, HyperUrbain* (2010).

[94] R. O. OBE et L. S. HSU. *PostGIS in action*. Manning Publications Co., 2015.

[100] *PostGIS Special Functions Index*. http://postgis.net/docs/PostGIS_Special_Functions_Index.html. Accessed : 2016-11-13.

[101] P. RACINE. *Advanced spatial analysis with PostGIS*. 2015. URL : https://2015.foss4g-na.org.

[102] *Representing a TIN in GeoJSON*. hhttp://gis.stackexchange.com/questions/22809/representing-a-tin-in-geojson. Accessed : 2016-11-13.

[108] N. SHIODE. “3D urban models : recent developments in the digital modelling of urban environments in three-dimensions”. In : *GeoJournal* 52.3 (2000), p. 263–269.

[109] K. SHIRE et al. “Regulation of the EBNA1 Epstein-Barr virus protein by serine phosphorylation and arginine methylation”. In : *Journal of virology* 80.11 (2006), p. 5261–5272.

[116] V. J. D. TSAI. “Delaunay triangulations in TIN creation : an overview and a linear-time algorithm”. In : *International Journal of Geographical Information Science* 7.6 (1993), p. 501–524.

[117] P. VAN OOSTEROM et J. STOTER. “5D data modelling : full integration of 2D/3D space, time and scale dimensions”. In : *International Conference on Geographic Information Science*. Springer. 2010, p. 310–324.

[118] J. VLISSIDES et al. “Design patterns : Elements of reusable object-oriented software”. In : *Reading : Addison-Wesley* 49.120 (1995), p. 11.

[120] T. WHITE. *Hadoop : The definitive guide*. O’Reilly Media, Inc., 2012.

[124] S. YOU, J. ZHANG et L. GRUENWALD. “Large-scale spatial join query processing in cloud”. In : *Data Engineering Workshops (ICDEW), 2015 31st IEEE International Conference on*. IEEE. 2015, p. 34–41.

[128] M. ZAHARIA et al. “Spark : cluster computing with working sets”. In : *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*. T. 10. 2010, p. 10.

4.3 Compléments à l’article

4.4 Connectivité avec les systèmes d’information géographiques

Par manque de place dans l’article, il n’a pas été possible de détailler les solutions mises en œuvre afin d’assurer l’interopérabilité d’Elcano avec les principaux logiciels SIG [cf. 1.2.4], de façon à permettre la consultation, l’analyse et la visualisation de données spatiales massives depuis ceux-ci. Par son recours aux logiciels SIG, qui occupent une place charnière entre les bases de données spatiales, les représentations cartographiques et les analyses spatiales, une telle approche semble pourtant promettre à terme une extension des questionnements et des impacts de la recherche au contexte de la représentation cartographique de données spatiales massives.

Afin de pouvoir relier Elcano aux différents logiciels SIG, il convenait donc d’étendre son interopérabilité. L’établissement d’un tel pont semble une problématique fortement connexe avec celle de l’unification des standards portant sur la composante spatiale, puisque l’interopérabilité des systèmes commence par l’emploi de protocoles de description des données communs. L’ISO et l’OGC ont plus spécifiquement publié plusieurs normes et spécifications techniques qui décrivent la gestion de l’interopérabilité de données géographiques, en particulier à travers les notions de service web et de méta-données [cf. 1.2.3].

La recommandation « Web Feature Service 2.0 Interface Standard » publiée en 2010 par l'OGC décrit ainsi le format WFS, qui semble particulièrement adapté à l'échange de données géographiques. Celui-ci permet en effet de sélectionner et de modifier des données spatiales vectorielles simplement en envoyant des requêtes HTTP vers un service web. Si cette recommandation a servi d'inspiration, il s'est cependant avéré délicat de l'implémenter en tant que telle pour permettre l'interopérabilité d'Elcano avec les logiciels SIG, si bien qu'une solution plus directe et plus flexible a du être trouvée. La suite de cette section décrit plus en détail celle-ci ainsi que les choix qu'elle a impliqués.

4.4.1 Choix d'un service web adapté

Un premier défi semblait donc de rendre Elcano directement consultable depuis un logiciel SIG. Pour cela, sa transformation en un service web gérant un format interopérable avec ces systèmes, comme WFS, semblait une voie toute indiquée.

Mais cela impliquait de contourner deux importantes difficultés. Premièrement, alors qu'un service web nécessite d'être maintenu dans le temps, les processus Spark effectuant des traitements par lots sont généralement conçus de façon à expirer à la fin du traitement des données. Mais cette impermanence assure que la mémoire et les processeurs du cluster utilisé soient libérés après exécution, si bien qu'y contrevenir par l'ajout d'un service web au noyau d'Elcano aurait nui aux performances et qu'il a fallu trouver une autre façon de procéder. Deuxièmement, l'usage d'Elcano nécessite de pouvoir exécuter directement des requêtes SQL, quelquefois accompagnées de code Scala effectuant certaines tâches annexes comme l'initialisation. Or, un service WFS n'autorise pas directement des requêtes en ces formats.

Ces difficultés ont été contournées par l'application open source Livy. Celle-ci consiste en un service web de type REST qui n'est pas une composante de Spark mais peut y faire appel, si bien qu'elle survit à l'extinction de processus Spark tout en permettant de communiquer avec lui. Elle permet de plus de passer n'importe quel code Scala en tant que paramètre au service web. Cependant, l'interface de Livy s'est avérée plutôt rudimentaire, puisqu'elle se limite à des opérations passées en ligne de commande. De plus ; l'intégration des dépendances extérieures d'Elcano à Livy posait des problèmes de configuration.

Pour y remédier, Livy a été encapsulée dans le gestionnaire de « notebook » Zeppelin [16]. De configuration plus aisée, celui-ci offre une interface web qui permet de construire des pages de script appelées « notes », elles-mêmes constituées de blocs de code permutables appelés « paragraphes ». Ces derniers sont destinés par défaut à contenir du code Scala, mais peuvent aussi gérer du code shell préfixé par l'expression « %shell% »ou des requêtes structurées pour Spark SQL en les préfixant par « %sql% ». Qui plus est, le résultat des requêtes SQL peut y être affiché dans un format graphique (par exemple un graphique à barres ou un camembert).

La figure 4.8 présente un court exemple de script Elcano sous la forme d'une note Zeppelin. Le

16. https://zeppelin.apache.org/

premier paragraphe initialise Elcano et charge la table « province » depuis un fichier CSV, tandis que la seconde sélectionne le champ attributaire « name » et le code WKT des enveloppes entourant les géométries de cette table. Dans le bas de l'image, une partie du résultat de la requête affichée sous forme de tableau apparait.

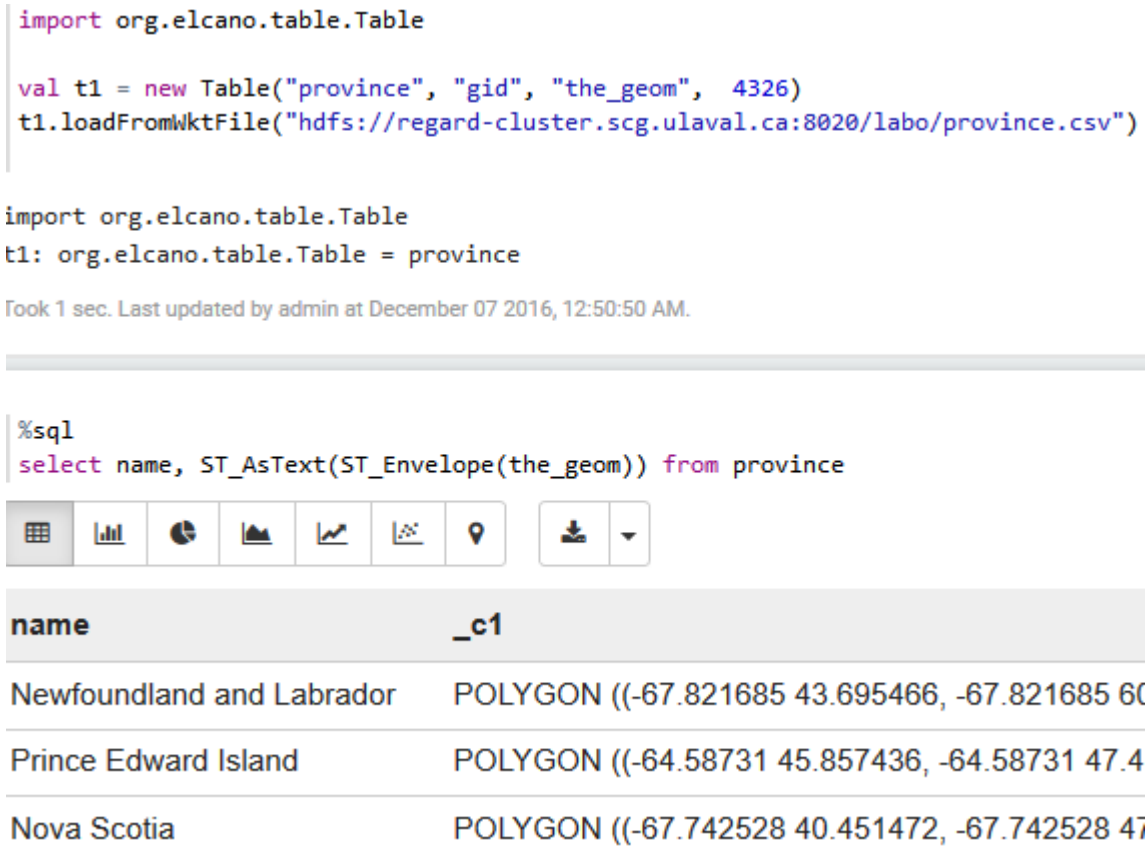

FIGURE 4.8 – Exemple d'utilisation d'Elcano depuis Zeppelin.

Chaque opération possible depuis l'interface du notebook Zeppelin peut également être déclenchée depuis un service web de type REST, ce qui donne à cette application la même puissance que Livy. L'ergonomie et la souplesse de Zeppelin sont par contre bien meilleures, puisque le recours à son interface peut permettre de préparer ou de compléter les requêtes échangées avec son service web. Ceci pourrait permettre par exemple de créer une table Spark SQL depuis l'interface, puis d'y appliquer des requêtes SQL à distance à partir du service.

4.4.2 Connectivité avec les SIG

Il restait à établir une connectivité entre le service web de Zeppelin intégrant Elcano et les principaux logiciels SIG. Comme Zeppelin, contrairement à WFS, n'est pas directement géré depuis ceux-ci, l'approche la plus simple semblait d'ajouter aux SIG un module permettant d'interroger à l'aide de requêtes SQL les données accessibles via le service web.

L'architecture définie pour intégrer un tel connecteur à un logiciel SIG est résumée par la figure 4.9. Elle a pour l'instant été conçue pour QGIS, mais pourrait aisément être étendue à tout logiciel SIG gérant un langage de script assez puissant pour interroger un service web de type REST.

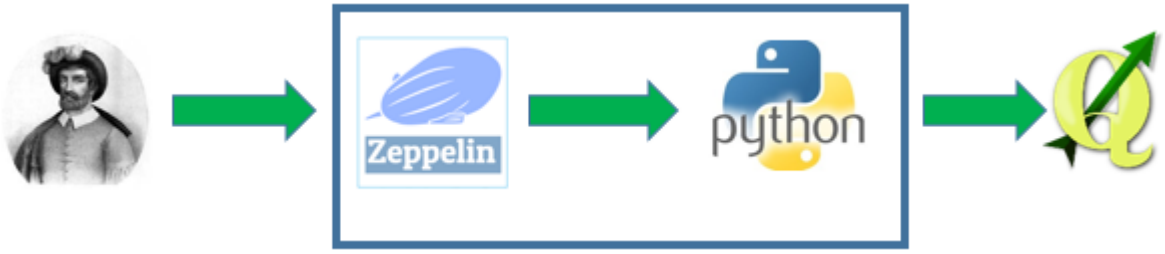

FIGURE 4.9 – Architecture du connecteur Elcano-SIG.

Afin de permettre leur affichage dans le logiciel SIG, les données massives gérées par Elcano (à gauche de l'image) sont rendues consultables depuis Zeppelin par la création d'un notebook adapté. Le service REST de Zeppelin est ensuite accédé par un module de connexion (ici écrit en Python) qui lui envoie une requête SQL à exécuter sur le notebook. Zeppelin interroge alors Elcano à l'aide de cette requête et communique le résultat au module, qui s'en sert pour créer et afficher une carte à la volée dans l'interface de visualisation du logiciel SIG (à droite de l'image).

4.4.3 Cas d'utilisation

Afin d'illustrer ce principe de fonctionnement, la figure 4.10 donne un exemple d'appel à l'interface du connecteur Elcano-SIG.

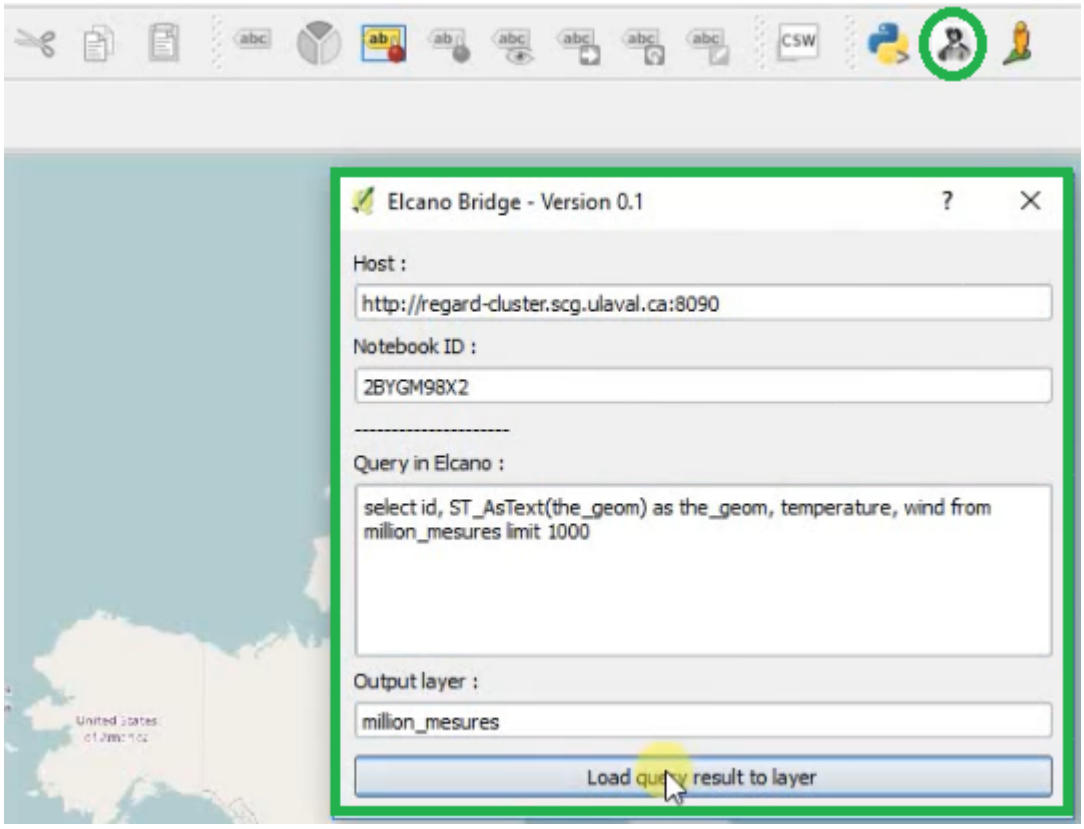

FIGURE 4.10 – Interface du connecteur SIG d'Elcano.

L'icône utilisée pour lancer le module est entourée d'un cercle vert et le menu qu'elle ouvre d'un rectangle vert. Ce menu permet d'indiquer l'adresse HTTP du notebook Zeppelin à consulter ainsi qu'une requête de sélection à y appliquer. Le résultat est alors immédiatement affiché dans une carte créée à la volée présentée par la figure 4.11. Il s'agit ici d'un nuage de mille points appartenant à l'enveloppe du Canada.

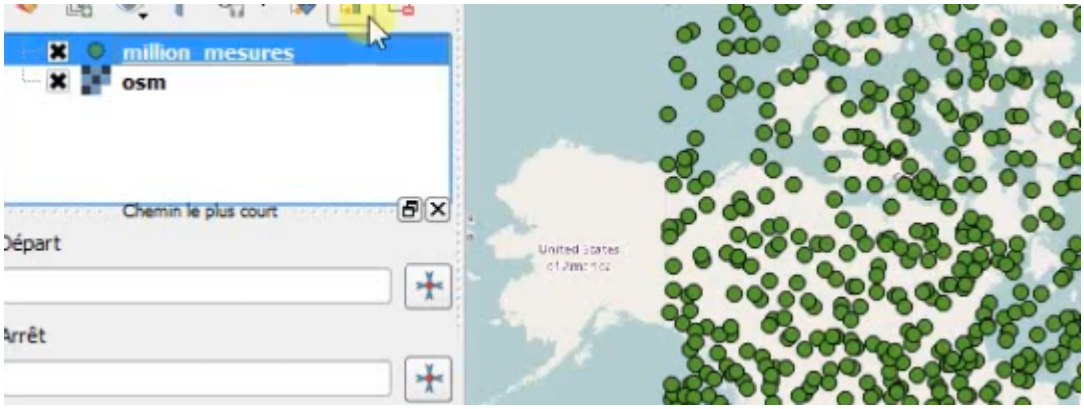

FIGURE 4.11 – Exemple de retour du connecteur SIG d'Elcano.

Il est à noter que le volume de données affiché ici a été intentionnellement réduit de façon à rester adapté au logiciel SIG utilisé, car celui-ci n'a pas été conçu pour gérer directement des quantités massives de données. Les implications de cette limitation vont être détaillées en conclusion de ce chapitre.

4.5 Conclusion

Dans ce chapitre, à la suite d'une étude détaillée des normes et spécifications décrivant la composante spatiale constructible par interpolation linéaire [4.2.2] ainsi que de leur adaptation par différentes solutions [4.2.3], certaines des extensions qu'elles proposent par rapport à la norme ISO-19125 sont choisies [4.2.3.1] afin d'intégrer une ébauche de modèle unifié de la composante spatiale [4.2.4]. Celui-ci prend la forme d'une extension apportée au système de gestion des données massives Elcano [43, 45], auquel sont ajoutées une gestion de l'élévation, des TIN et du datum.

L'article choisit par contre de ne pas gérer le format matriciel, la 3D et la topologie au nouveau modèle d'Elcano 4.2.3.1. Les deux premiers sont en effet très complexes et dépassent la visée initiale de la recherche, tandis que le troisième fragmente les géométries entre plusieurs tables d'une façon qui implique dans un contexte SQL de joindre ces tables à la lecture de chaque géométrie. Dans l'article, nous montrons qu'une telle approche s'avérerait couteuse dans un contexte de requêtes SQL distribuées comme celui Elcano. Mais il resterait intéressant de rechercher d'autres façons d'intégrer la topologie à un contexte de données massives, étant donné le gain d'espace mémoire qu'elle permet par sa compacité. L'implémentation du format matriciel gagnerait également à être envisagée dans une version ultérieure du modèle, étant donné sa généricité, son grand nombre d'applications et la possibilité technique de l'implémenter en utilisant JTS [17], GDAL [18], ou une bibliothèque spécifiquement dédiée au format comme JAITools [19].

En complément à l'article, le chapitre présente un connecteur Elcano-SIG [4.4] mis au point pour les besoins de cette recherche. Celui-ci se compose dans son modèle actuel d'un service Web de type REST porté par Zeppelin [4.4.1] et d'un client QGIS artisanal permettant d'interroger le service à l'aide d'une requête SQL et d'afficher sa réponse à l'aide d'une carte construite à la volée dans QGIS [4.4.2].

Dans une recherche ultérieure, il serait intéressant de déterminer s'il est possible d'éviter ce recours à un plugin interne au logiciel SIG pour visualiser les données d'Elcano, par exemple en tentant de modéliser un service web qui intègre la puissance de Zeppelin mais repose sur un format directement interopérable avec les logiciels SIG [cf. 1.2.3]. Cependant, même une telle amélioration se heurterait aux limites des logiciels SIG en terme de capacité d'affichage, qui empêchent de les employer pour visualiser des volumes de données très importants. S'ils sont adaptés au traitement de données spatiales

17. http://docs.geotools.org/latest/userguide/extension/grid.html
18. http://www.gdal.org/
19. http://jaitools.org/

ordinaires, les logiciels SIG du marché sont en effet mal préparés aux données massives, en raison de la faible scalabilité horizontale permise par leur architecture mono-serveur [9].

Le logiciel SIG pour données spatiales massives reste donc à inventer.

Conclusion et perspectives

Conclusion

La première section de cette conclusion synthétise les contributions apportées par cette recherche, tandis que la seconde évoque les perspectives qu'elle ouvre.

Contributions

Au terme de cette recherche, un système de gestion des données spatiales massives efficient et complet a pu être conçu (**OP**). Celui-ci offre un modèle unifié de la composante spatiale qui applique et étend ISO-19125 (**O1**) [2.2.3.2, 3.2.4.2, 4.2.4]. Il propose un système d'indexation spatiale efficace et paramétrable (**O2**) [3.2.4.2], et peut être directement relié à QGIS selon une méthode facilement extensible à d'autres SIG (**O5**)) [4.4]. Au fil des trois articles constituant le cœur de ce mémoire, un prototype de ce système conçu à partir d'outils issus de l'écosystème Hadoop a été implémenté et amélioré (**O3**). Il a la forme d'une extension spatiale au gestionnaire de données massives Spark SQL. Les tests réalisés sur un cluster de neuf serveurs montrent que ce prototype surpasse les systèmes PostGIS et Spatial Spark à partir d'un certain volume de données, tout en présentant une bonne scalabilité horizontale **(O4)** [3.2.5].

Cette recherche a visé à optimiser et à synthétiser les systèmes existants, au lieu de produire un système d'un genre nouveau. L'objectif était donc essentiellement d'améliorer l'interopérabilité et les performances en termes de gestion des données massives à référence spatiale. C'est dans ce contexte qu'il a été comparé à des outils existants. Malgré tout, il s'entend que le prototype réalisé est davantage une validation par l'exemple qu'une solution logicielle définitive. En outre, le projet ne considère le traitement des données spatiales massives que d'un point de vue généraliste, sans tenir compte des problèmes d'intégration ou de validation qu'ils impliquent en réalité. Mais ces limites sont fidèles aux objectifs de la recherche, qui visait avant tout à montrer la possibilité d'une meilleure approche que les prototypes actuels. Les adaptations possibles ainsi que les optimisations spécifiques à certaines situations sont donc laissées à des études plus appliquées.

Perspectives

Dans sa version actuelle, le système proposé ne traite que des données reçues en batch, et n'offre qu'un système de visualisation rudimentaire et statique. Or, il serait intéressant d'y inclure la possibilité de traiter et de visualiser des données reçues en continu [42]. Une telle extension améliorerait en effet de beaucoup la réactivité du système ainsi que son ergonomie, ce qui faciliterait la prise de décision à partir de celui-ci. De plus, elle pourrait permettre de réduire à la source le volume de données à traiter, par exemple en excluant celles qui se situent en dehors d'une fenêtre temporelle définie [55].

L'extension du système à des données reçues en continu serait techniquement possible à partir de Spark, puisque celui-ci comporte nativement un module de gestion du streaming. Mais cette évolution pose quelques problèmes conceptuels et techniques. Elle ne saurait aller sans une redéfinition en pro-

fondeur des solutions d'indexation fournies dans le second article. Le périmètre applicatif de celles-ci se limite en effet aux données reçues d'un bloc (en batch). De plus, l'implémentation JTS de R-Tree employée dans la version actuelle d'Elcano construit un index en lecture seule, ce qui rend difficile son utilisation directe sur des données fluctuantes.

Dans le cas de données de streaming associées a des phénomènes en mouvement, comme les coordonnées GPS de véhicules, la possibilité de visualiser des données massives reçues en continu pose des problèmes encore plus ardus [20]. En premier lieu, quelles seraient les variables visuelles pertinentes pour la représentation de telles données ? Malgré leur succès et leur longévité, celles définies par Bertin en 1967 [10] s'avèrent impropres à la représentation de phénomènes spatio-temporels. Bertin a en effet exclu d'emblée le mouvement des représentations cartographiques, en considérant qu'il en modifierait trop les règles et que son étude relevait plutôt de la cinématographie. Si des travaux plus récents tentent d'ajouter au modèle de Bertin des variables visuelles liées au mouvement [81, 47], tout reste à faire en ce qui concerne leur application à la représentation de données massives à référence spatiale.

Une fois résolues ces problématiques liées au traitement et à la représentation de données spatiales reçues en continu, il resterait à définir un système effectivement à même de gérer de telles données. Celui-ci ne saurait se limiter à une extension de Spark reliée à un Zeppelin et à QGIS, comme c'est le cas pour la version actuelle d'Elcano. Zeppelin est en effet très limité dans les représentations géographiques qu'il permet, tandis que QGIS n'est efficace que pour le traitement de faibles volumes de données. Une étape majeure dans l'évolution d'Elcano serait donc de lui adjoindre un nouveau type de notebook ou de système d'information géographique, à même de représenter efficacement de gros volumes de données reçus en continu et reposant sur les mêmes principes de scalabilité horizontale de Hadoop. Étant donné la richesse des problématiques qu'elle implique, la définition d'un tel système pourrait s'inscrire au cœur d'une recherche doctorale.

Dans une visée encore plus large, l'extension d'Elcano au traitement des formats matricielles ou 3D, qui ont été rejetés dans le chapitre 4 en raison de leur complexité, serait également une voie de recherche pleine de défis et de promesses, en raison des difficultés de représentation particulières et des volumes de données encore plus importants qu'ils impliqueraient.

20. Il serait aussi possible d'avoir du streaming sur des données spatiales « statiques », comme des stations météo ou tout autre capteurs fixes, dont seules les mesures sont l'objet de changements. Dans un tel dernier cas, la problématique évoquée n'est pas à considérer.

Bibliographie

[1] A. S. AIYER et al. “Storage Infrastructure Behind Facebook Messages : Using HBase at Scale.” In : *IEEE Data Eng. Bull.* 35.2 (2012), p. 4–13.

[2] A. AJI et al. “Hadoop GIS : a high performance spatial data warehousing system over mapreduce”. In : *Proceedings of the VLDB Endowment* 6.11 (2013), p. 1009–1020.

[3] Thomas ALEXANDRE. *Scala for Java Developers*. Packt Publishing Ltd, 2014.

[4] C. ANDERSON. “The end of theory : The data deluge makes the scientific method obsolete”. In : *Wired magazine* 16.7 (2008), p. 16–07.

[5] Natalia ANDRIENKO et al. “Detection, tracking, and visualization of spatial event clusters for real time monitoring”. In : *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*. IEEE. 2015, p. 1–10.

[6] M. ARMBRUST et al. “Spark sql : Relational data processing in spark”. In : *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 2015, p. 1383–1394.

[7] G. AYDIN et al. “Streaming Data Services to Support Archival and Real-Time Geographical Information System Grids”. In : *Sixth Annual NASA Earth Science Technology Conference*. 2006.

[8] T. BADARD. “Mettre le Big Data sur la carte : défis et avenues relatifs à l’exploitation de la localisation”. In : *Colloque ITIS - Big Data et Open Data au coeur de la ville intelligente*. (29 avr. 2014). Québec : Centre de recherche en géomatique, 2014.

[9] T. BADARD et J. ENGÉLINUS. “Elcano : un prototype de moteur Big Data spatial basé sur Spark”. In : *Colloque ITIS - Innovation et données massives : un panorama*. (31 oct. 2016). Québec : Centre de recherche en géomatique, 2016.

[10] J. BERTIN. “Semiologie Graphique : Les Diagrammes”. In : *Les Reseaux, Les Cartes* (1967).

[11] M. BOSTOCK. *How to infer topology*. `http://bost.ocks.org/mike/topology`. Accessed : 2016-11-13. 2013.

[12] T. BRAY. *The javascript object notation (json) data interchange format (No. RFC 7158)*. 2014.

[13] H. BUTLER et al. “Geojson specification”. In : *Geojson.org* (2008).

[14] H. BUTLER et al. *RFC 7946 - The GeoJSON Format*. Rapp. tech. 2016.

[15] R. CATTELL. “Scalable SQL and NoSQL data stores”. In : *Acm Sigmod Record* 39.4 (2011), p. 12–27.

[16] R. CHENG et al. “Uv-diagram : A voronoi diagram for uncertain data”. In : *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*. 2010, p. 796–807.

[17] K. CHODOROW. *MongoDB : the definitive guide*. " O’Reilly Media, Inc.", 2013.

[18] E. CLEMENTINI et P. DI FELICE. “A model for representing topological relationships between complex geometric features in spatial databases”. In : *Information sciences* 90.1 (1996), p. 121–136.

[19] E. CLEMENTINI et R. LAURINI. “Un cadre conceptuel pour modéliser les relations spatiales”. In : *Revue des Nouvelles Technologies de l’Information (RNTI)* 14 (2008), p. 1–17.

[20] R. CORTÉS et al. “A Scalable Architecture for Spatio-Temporal Range Queries over Big Location Data”. In : *Network Computing and Applications (NCA), 2015 IEEE 14th International Symposium on*. 2015, p. 159–166.

[21] P. CUGIER et P. LE HIR. “Development of a 3D hydrodynamic model for coastal ecosystem modelling. Application to the plume of the Seine River (France)”. In : *Estuarine, Coastal and Shelf Science* 55.5 (2002), p. 673–695.

[22] Belanger D. “La stratégie d’élévation nationale”. In : *Colloque Géomatique 2016*. (19 oct. 2016). Québec : ACSG, 2016.

[23] T. K. DANG, J. KÜNG et R. WAGNER. “The sh-tree : A super hybrid index structure for multidimensional data”. In : *International Conference on Database and Expert Systems Applications*. Springer. 2001, p. 340–349.

[24] T. H. DAVENPORT, P. BARTH et R. BEAN. “How ’big data’ is different”. In : *MIT Sloan Management Review* 54.1 (2013).

[25] M. DAVIS. *JTS Topology Suite’s Forum*. https://sourceforge.net/p/jts-topo-suite/mailman/message/27654158/. 2011.

[26] M. DAVIS et J. AQUINO. *Jts topology suite technical specifications*. 2003.

[27] G. DE ROSA. *Hadoop Vs. MongoDB : Which Platform is Better for Handling Big Data ?* 2014. URL : http://www.aptude.com/blog/entry/hadoop-vs-mongodb-which-platform-is-better-for-handling-big-data.

[28] J. DEAN et S. GHEMAWAT. “MapReduce : simplified data processing on large clusters”. In : *Communications of the ACM* 51.1 (2008), p. 107–113.

[29] B. DELAUNAY. “Sur la sphere vide”. In : *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk* 7.793-800 (1934), p. 1–2.

[30] Y. DEMCHENKO et al. “Addressing big data issues in scientific data infrastructure”. In : *Collaboration Technologies and Systems (CTS), 2013 International Conference on*. IEEE. 2013, p. 48–55.

[31] J. DENÈGRE et F. SALGÉ. *Les systèmes d'information géographique*. Presses universitaires de France, 1996.

[32] E. DEVYS et G. GESQUIÈRE. *Intégration des données et modèles urbains : standards, normes et tendances pour les SIG*.

[33] F. DONG. "Extending starfish to support the growing hadoop ecosystem". Thèse de doct. Duke University, 2012.

[34] A. DREWS et al. "Monitoring and remote failure detection of grid-connected PV systems based on satellite observations". In : *Solar Energy* 81.4 (2007), p. 548–564.

[35] A. DROIT. "Les nouveaux défis pour l'intégration et l'analyse des données génomiques". In : *Conference ITIS - Le défi du traitement des données*. (29 avr. 2015). Québec : Centre de recherche du CHU de Québec, 2015.

[36] Gamma E. et al. *Design Patterns : Elements of Reusable Object-Oriented Software*. 1994.

[37] M. J. EGENHOFER et R. D. FRANZOSA. "On the equivalence of topological relations". In : *International journal of geographical information systems* 9.2 (1995), p. 133–152.

[38] M. J. EGENHOFER et Robert D FRANZOSA. "Point-set topological spatial relations". In : *International Journal of Geographical Information System* 5.2 (1991), p. 161–174.

[39] M. J. EGENHOFER et J. HERRING. "Categorizing binary topological relations between regions, lines, and points in geographic databases". In : *The* 9.94-1 (1990), p. 76.

[40] A. ELDAWY et M. F. MOKBEL. "Pigeon : A spatial mapreduce language". In : *Data Engineering, 2014 30th International Conference on IEEE*. 2014, p. 1242–1245.

[41] A. ELDAWY et Mohamed F MOKBEL. "The Era of Big Spatial Data : A Survey". In : *Information and Media Technologies* 10.2 (2015), p. 305–316.

[42] J. ENGELINUS et T. BADARD. "Towards a Real-Time Thematic Mapping System for Streaming Big Data". In : *GIScience September 2016, Montreal*. 2016.

[43] J. ENGÉLINUS et T. BADARD. "Elcano : un système de gestion des données massives à composante spatiale basé sur Spark SQL », Revue internationale de géomatique". In : *Revue internationale de géomatique (en cours de soumission)*. 2016.

[44] J. ENGÉLINUS et T. BADARD. "Vers un modèle unifié de la composante spatiale vectorielle et son application à Elcano". In : *à traduire en anglais pour soumission dans Journal of GIS*. 2016.

[45] J. ENGÉLINUS et T. BADARD. "Vers une indexation des données spatiales massives performante et efficace". In : *Geomatica (en cours de soumission)*. 2016.

[46] M. R. EVANS et al. "Spatial big data". In : *Big Data : Techniques and Technologies in Geoinformatics* (2014), p. 149.

[47] S. I. FABRIKANT et K. GOLDSBERRY. “Thematic relevance and perceptual salience of dynamic geovisualization displays”. In : *Proceedings, 22th ICA/ACI International Cartographic Conference, Coruna*. 2005.

[48] S. FERMIGIER. *Big data et open source : une convergence inévitable ?* 2011. URL : http://projet-plume.org.

[49] R. A. FINKEL et J. L. BENTLEY. “Quad trees a data structure for retrieval on composite keys”. In : *Acta informatica* 4.1 (1974), p. 1–9.

[50] A. FOX et al. “Spatio-temporal indexing in non-relational distributed databases”. In : *Big Data, 2013 IEEE International Conference on*. 2013, p. 291–299.

[51] C. FRANKLIN et P. HANE. “An Introduction to Geographic Information Systems : Linking Maps to Databases [and] Maps for the Rest of Us : Affordable and Fun.” In : *Database* 15.2 (1992), p. 12–15.

[52] J. GANTZ et D. REINSEL. “The digital universe in 2020 : Big data, bigger digital shadows, and biggest growth in the far east”. In : *IDC iView : IDC Analyze the future* 2007 (2012), p. 1–16.

[53] V. GAUTRAIS. “Données massives et droit”. In : *Colloque ITIS - Données massives et droit*. (31 oct. 2016). Québec : Chaire L.R. Wilson, Université de Montréal, 2016.

[54] L. GEORGE. *HBase : the definitive guide*. " O’Reilly Media, Inc.", 2011.

[55] L. GOLAB. “Sliding window query processing over data streams”. Thèse de doct. University of Waterloo, 2006.

[56] C. GORMLEY et Z. TONG. *Elasticsearch : The Definitive Guide*. "O’Reilly Media, Inc.", 2015.

[57] M. GRAHAM et T. SHELTON. “Geography and the future of big data, big data and the future of geography”. In : *Dialogues in Human Geography* 3.3 (2013), p. 255–261.

[58] J. GRAY, L. CHAMBERS et L. BOUNEGRU. *The data journalism handbook*. " O’Reilly Media, Inc.", 2012.

[59] L. GU et H. LI. “Memory or time : Performance evaluation for iterative operation on hadoop and spark”. In : *High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 10th International Conference on IEEE*. 2013, p. 721–727.

[60] K. GULIYEV et A. SHAIKHHA. *Query Synthesis for Big Data*. GRIN Verlag, 2015.

[61] Ralf Hartmut GÜTING. “An introduction to spatial database systems”. In : *The VLDB Journal—The International Journal on Very Large Data Bases* 3.4 (1994), p. 357–399.

[62] A. GUTTMAN. *R-trees : a dynamic index structure for spatial searching*. T. 14. 2. ACM, 1984.

[63] T. HAERDER et A. REUTER. “Principles of transaction-oriented database recovery”. In : *ACM Computing Surveys (CSUR)* 15.4 (1983), p. 287–317.

[64] P. HASHEMI et al. *OpenStreetMap in GIScience, Experiences, Research, and Applications (LNCS)*. 2015.

[65] E. HOEL et M. PARK. "Big Data : Using ArcGIS with Apache Hadoop". In : *Esri International Developer Summit* (2014).

[66] T. HUDDLESTON. "Latest funding round values Elasticsearch at 700 million USD". In : *Fortune* (2014). URL : http://fortune.com/2014/06/06/latest-funding-round-values-elasticsearch-at-700-million/.

[67] J. N. HUGHES et al. "GeoMesa : a distributed architecture for spatio-temporal fusion". In : *SPIE Defense + Security*. International Society for Optics et Photonics. 2015, 94730F.

[68] MongoDB INC. *The MongoDB 3.4 Manual*. 2016. URL : https://docs.mongodb.com/v3.4.

[69] J KERHERVÉ. "L'interopérabilité des systémes d'information géographique". Thèse de doct. Haute école de gestion de Genève, 2008.

[70] A. KNOLL et al. "Interactive isosurface ray tracing of large octree volumes". In : *Interactive Ray Tracing 2006, IEEE Symposium on*. IEEE. 2006, p. 115–124.

[71] Thomas H KOLBE. "Representing and exchanging 3D city models with CityGML". In : *3D geo-information sciences*. Springer, 2009, p. 15–31.

[72] R. K. V. KOTHURI, S. RAVADA et D. ABUGOV. "Quadtree and R-tree indexes in oracle spatial : a comparison using GIS data". In : *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*. ACM. 2002, p. 546–557.

[73] A. KRIMBACHER et al. *Service-oriented Architecture for Thematic Cartography on the Web*. 2014. URL : http://ika.ethz.ch.

[74] R. KUĆ et M. ROGOZINSKI. *ElasticSearch server*. Packt Publishing Ltd, 2016.

[75] R. LAHAYE et S. LADET. "Les concepts de base des SIG nomades". In : *Le Cahier des Techniques de l'INRA* (2014).

[76] A. LAKSHMAN et P. MALIK. "Cassandra : a decentralized structured storage system". In : *ACM SIGOPS Operating Systems Review* 44.2 (2010), p. 35–40.

[77] D. LANEY. "3D data management : Controlling data volume, velocity and variety". In : *META Group Research Note* 6 (2001), p. 70.

[78] D. B. LOMET et B. SALZBERG. "The hB-tree : A multiattribute indexing method with good guaranteed performance". In : *ACM Transactions on Database Systems (TODS)* 15.4 (1990), p. 625–658.

[79] C. LYNCH. "Big data : How do your data grow ?" In : *Nature* 455.7209 (2008), p. 28–29.

[80] F. MAALOULI et R. HIGHTOWER. *Introduction to Apache Spark*. http://www.mammatustech.com/introduction-to-apache-spark. Accessed : 2016-11-21.

[81] A. M. MACEACHREN. “An evolving cognitive-semiotic approach to geographic visualization and knowledge construction”. In : *Information Design Journal* 10.1 (2001), p. 26–36.

[82] R. MACIEJEWSKI et al. “A visual analytics approach to understanding spatiotemporal hotspots”. In : *IEEE Transactions on Visualization and Computer Graphics* 16.2 (2010), p. 205–220.

[83] M. MAIER, A. SEREBRENIK et I. VANDERFEESTEN. *Towards a Big Data Reference Architecture*. 2013.

[84] J. MANYIKA et al. “Big data : The next frontier for innovation, competition, and productivity”. In : *The McKinsey Global Institute* (2011).

[85] B. MARK. “Big Data Challenges for Geoinformatics”. In : *Geoinformatics & Geostatistics : An Overview* (2012).

[86] Y. MATSUDA. *Geo Library for Amazon DynamoDB – Part 1 : Table Structure*. 2013. URL : https://aws.amazon.com/fr/blogs/mobile/geo-library-for-amazon-dynamodb-part-1-table-structure/.

[87] C. MCDONALD. *How to get started using Apache Spark GraphX in Scala ?* 2016. URL : https://mapr.com/blog/how-get-started-using-apache-spark-graphx-scala/.

[88] Xiangrui MENG et al. “Mllib : Machine learning in apache spark”. In : *Journal of Machine Learning Research* 17.34 (2016), p. 1–7.

[89] B. MERICSKAY et S. ROCHE. “Cartographie numérique en ligne nouvelle génération : impacts de la néogéographie et de l’information géographique volontaire sur la gestion urbaine participative”. In : *Nouvelles cartographie, nouvelles villes, HyperUrbain* (2010).

[90] R. MISHRA et R. SHARMA. “Big data : opportunities and challenges”. In : *International Journal of Computer Science and Mobile Computing* 4.6 (2015), p. 27–35.

[91] G. NIEMEYER. *Geohash*. 2008.

[92] J. NIEVERGELT, H. HINTERBERGER et K. C. SEVCIK. “The grid file : An adaptable, symmetric multikey file structure”. In : *ACM Transactions on Database Systems (TODS)* 9.1 (1984), p. 38–71.

[93] H. NOBORIO, T. NANIWA et S. ARIMOTO. “A quadtree-based path-planning algorithm for a mobile robot”. In : *Journal of Robotic Systems* 7.4 (1990), p. 555–574.

[94] R. O. OBE et L. S. HSU. *PostGIS in action*. Manning Publications Co., 2015.

[95] M. ODERSKY et al. “The Scala language specification”. In : *Programming Methods Laboratory, EPFL, Switzerland* (2004).

[96] A. OKABE, B. BOOTS et K. SUGIHARA. “Nearest neighbourhood operations with generalized Voronoi diagrams : a review”. In : *International Journal of Geographical Information Systems* 8.1 (1994), p. 43–71.

[97] G. PALSKY. "MAP DESIGN VS SÉMIOLOGIE GRAPHIQUE : Réflexions sur deux courants de la cartographie théorique". In : *Cartes & géomatique* 212 (2012), p. 7–12.

[98] A. G. PIAZZA. "NoSQL". Thèse de doct. Haute école de gestion de Genève, 2013.

[99] R. PICOT CLEMENTE, C. BOTHOREL et P. LENCA. *Une brève introduction aux Données Massives-Challenges et perspectives*. Rapp. tech. HAL, 2015.

[100] *PostGIS Special Functions Index*. http://postgis.net/docs/PostGIS_Special_Functions_Index.html. Accessed : 2016-11-13.

[101] P. RACINE. *Advanced spatial analysis with PostGIS*. 2015. URL : https://2015.foss4g-na.org.

[102] *Representing a TIN in GeoJSON*. hhttp://gis.stackexchange.com/questions/22809/representing-a-tin-in-geojson. Accessed : 2016-11-13.

[103] Commonwealth Computer RESEARCH. *Apache Spark Analysis*. http://www.geomesa.org/documentation/tutorials/spark.html. Accessed : 2017-02-26.

[104] Ram S. *Magellan : Geospatial Analytics on Spark*. 2015. URL : http://hortonworks.com/blog/magellan-geospatial-analytics-in-spark/.

[105] Smith S. *Dr. David Maguire on the ArcGIS 9.0 Product Family Release*. 2004. URL : http://www10.giscafe.com/nbc/articles/view_weekly.php?articleid=208790.

[106] S. SAGIROGLU et D. SINANC. "Big data : A review". In : *Collaboration Technologies and Systems (CTS), 2013 International Conference on*. IEEE. 2013, p. 42–47.

[107] M. SCHMIDT et G. GARTNER. "Decision Support Tool for Web-Based Thematic Mapping". In : *Junior Scientist Conference 2010*. Citeseer. 2010, p. 293.

[108] N. SHIODE. "3D urban models : recent developments in the digital modelling of urban environments in three-dimensions". In : *GeoJournal* 52.3 (2000), p. 263–269.

[109] K. SHIRE et al. "Regulation of the EBNA1 Epstein-Barr virus protein by serine phosphorylation and arginine methylation". In : *Journal of virology* 80.11 (2006), p. 5261–5272.

[110] R. SRIHARSHA. *Magellan's Github - issue 30*. https://github.com/harsha2010/magellan/issues/30. Accessed : 2016-05-05.

[111] M. STONEBRAKER. "SQL databases v. NoSQL databases". In : *Communications of the ACM* 53.4 (2010), p. 10–11.

[112] G. J. SULLIVAN et R. L. BAKER. "Efficient quadtree coding of images and video". In : *IEEE Transactions on Image Processing* 3.3 (1994), p. 327–331.

[113] A. TAKASU et al. "An Efficient Distributed Index for Geospatial Databases". In : *International Conference on Database and Expert Systems Applications*. Springer. 2015, p. 28–42.

[114] E. TANIN, A. HARWOOD et H. SAMET. "Using a distributed quadtree index in peer-to-peer networks". In : *The VLDB Journal* 16.2 (2007), p. 165–178.

[115] D. THOM et al. “Spatiotemporal anomaly detection through visual analysis of geolocated twitter messages”. In : *Visualization Symposium (PacificVis), 2012 IEEE Pacific*. IEEE. 2012, p. 41–48.

[116] V. J. D. TSAI. “Delaunay triangulations in TIN creation : an overview and a linear-time algorithm”. In : *International Journal of Geographical Information Science* 7.6 (1993), p. 501–524.

[117] P. VAN OOSTEROM et J. STOTER. “5D data modelling : full integration of 2D/3D space, time and scale dimensions”. In : *International Conference on Geographic Information Science*. Springer. 2010, p. 310–324.

[118] J. VLISSIDES et al. “Design patterns : Elements of reusable object-oriented software”. In : *Reading : Addison-Wesley* 49.120 (1995), p. 11.

[119] D. VOHRA. “Apache Parquet”. In : *Practical Hadoop Ecosystem*. Springer, 2016, p. 325–335.

[120] T. WHITE. *Hadoop : The definitive guide*. O’Reilly Media, Inc., 2012.

[121] Dong XIE et al. *Simba : Efficient In-Memory Spatial Analytics*. URL : https://www.cs.utah.edu/~lifeifei/papers/simba.pdf.

[122] J. YOU S. Zhang et L. GRUENWALD. “ISP : Large-Scale In-memory Spatial Data Processing System”. In : 2015.

[123] S. YOU. “Large-scale spatial data management on moderne parallel and distributed platforms”. In : 2015.

[124] S. YOU, J. ZHANG et L. GRUENWALD. “Large-scale spatial join query processing in cloud”. In : *Data Engineering Workshops (ICDEW), 2015 31st IEEE International Conference on*. IEEE. 2015, p. 34–41.

[125] S. YOU, J. ZHANG et L. GRUENWALD. “Scalable and efficient spatial data management on multi-core CPU and GPU clusters : A preliminary implementation based on Impala”. In : *Data Engineering Workshops (ICDEW), 2015 31st IEEE International Conference on*. IEEE. 2015, p. 143–148.

[126] J. YU. *GeoSpark’s Github - issue 33*. https://github.com/DataSystemsLab/GeoSpark/issues/33. Accessed : 2017-02-26.

[127] J. YU, J. WU et M. SARWAT. “Geospark : A cluster computing framework for processing large-scale spatial data”. In : *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM. 2015, p. 70.

[128] M. ZAHARIA et al. “Spark : cluster computing with working sets”. In : *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*. T. 10. 2010, p. 10.

[129] Z. ZHANG. *Spark-on-HBase : dataframe based HBase connector*. 2016. URL : http://hortonworks.com/blog/spark-hbase-dataframe-based-hbase-connector/.