Titre: Title:	Évaluation d'infrastructures sémantiques à large échelle pour la détection d'intrusion dans un centre d'opérations et de sécurité de l'information
Auteur: Author:	Jean-Yves Sami Ouattara
Date:	2021
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Ouattara, J.-Y. S. (2021). Évaluation d'infrastructures sémantiques à large échelle pour la détection d'intrusion dans un centre d'opérations et de sécurité de l'information [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/6298/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/6298/
Directeurs de recherche: Advisors:	Michel Gagnon, & Jose Manuel Fernandez
Programme: Program:	Génie informatique

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Évaluation d'infrastructures sémantiques à large échelle pour la détection d'intrusion dans un centre d'opérations et de sécurité de l'information

JEAN-YVES SAMI OUATTARA
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Avril 2021

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Évaluation d'infrastructures sémantiques à large échelle pour la détection d'intrusion dans un centre d'opérations et de sécurité de l'information

présenté par **Jean-Yves Sami OUATTARA**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Frédéric CUPPENS, président
José FERNANDEZ, membre et directeur de recherche
Michel GAGNON, membre et codirecteur de recherche
Amal ZOUAQ, membre

DÉDICACE

À ma famille, à mes amis qui ont toujours cru en moi...

REMERCIEMENTS

Sans le soutien de plusieurs personnes, je n'aurai pas pu compléter cette maîtrise. Je tiens à exprimer toute ma reconnaissance à mon directeur de recherche José Manuel Fernandez qui m'a accepté dans son laboratoire de recherche. Ton soutien et tes conseils m'ont guidé durant ces années de maîtrise. Je remercie mon codirecteur Michel Gagnon qui a accepté de me codiriger et dont les conseils pertinents m'ont permis d'améliorer ce mémoire.

Je remercie toute l'équipe du laboratoire, professionnels de recherche et collègues, pour avoir créé un environnement d'entraide, de bonne humeur et d'apprentissage continu. En particulier, à Militza, merci pour avoir continuellement défendu nos intérêts et nous avoir permis de mener à bien notre recherche. Merci à Marielba qui m'a soutenu avec patience et abnégation durant la rédaction et la correction de mon mémoire.

À mes parents, vous avez tout donné pour m'offrir une éducation de qualité. Je vous suis éternellement reconnaissant. Léa, Évariste, Sandrine, Armande, vous avez toujours veillé sur moi et avez été un exemple pour moi. Merci d'être toujours présents pour moi. Kalouna, tu as été un soutien constant du début à la fin de cette maîtrise. Merci de m'avoir donné la force de continuer.

RÉSUMÉ

Pour répondre à la croissance de la menace en cybersécurité, les entreprises, particulièrement les institutions financières, centralisent les ressources humaines et technologiques de sécurité dans un *Security Operations Center* (SOC).

Cependant, le volume important d'informations à traiter par les analystes ne fait que retarder le temps de réponse en cas d'incidents. Pour accompagner les analystes de sécurité dans leur investigation, de nombreux outils analytiques ont été développés. Néanmoins, ces outils n'offrent pas les meilleures performances quand il s'agit de nouvelles menaces.

Une des solutions étudiées est l'utilisation d'ontologies pour renforcer les capacités analytiques des analystes de sécurité. En créant un système expert s'appuyant sur l'ontologie et ses propriétés sémantiques, on pourrait améliorer la mise en relation d'événements et d'entités dans le SOC et ainsi améliorer la corrélation d'événements.

Cette approche serait prometteuse si les bases de données RDF (*Resource Description Framework*) ont la capacité de traiter en temps raisonnable les volumes de données transitant dans le SOC. C'est le sujet de notre recherche. Nous avons d'abord répertorié les caractéristiques pertinentes des bases de données de graphe dans le contexte d'un SOC. En faisant le lien entre ces caractéristiques et les besoins des différents tiers d'analystes de sécurité du SOC, nous proposons un guide de décision dans le choix du magasin RDF en fonction du scénario. Nous allons plus loin en comparant de manière expérimentale des magasins RDF à savoir Anzograph, GraphDB, Stardog et Virtuoso. Cette comparaison s'est faite sur les temps de chargements de données et les temps de réponses à des requêtes, avec une augmentation progressive de la taille des données.

Les résultats de notre recherche montrent que les technologies sémantiques actuelles ont suffisamment évolué pour répondre aux besoins des entreprises. Il faut cependant souligner qu'aucun outil était supérieur sur toutes les requêtes. Cela montre l'impact de la modélisation des ontologies et des requêtes sur les performances selon l'outil utilisé.

ABSTRACT

To respond to the growing cyber threat, companies, particularly financial institutions, are centralising human and technological security resources in a *Security Operations Center* (SOC).

However, the large amount of information to be processed by the analysts only delays the incident response time. To assist security analysts in their investigation, many analytical tools have been developed. However, these tools do not offer the best performance when it comes to new threats.

One of the solutions studied is the use of ontologies to reinforce the analytical capabilities of security analysts. By creating an expert system based on ontology and its semantic properties, one could improve the relationships between events and entities in the SOC and thus improve event correlation.

This approach would be promising if RDF (*Resource Description Framework*) stores have the capacity to handle the volumes of data passing through the SOC in a reasonable time. This is the subject of our research. We first identified the relevant characteristics of graph databases in the context of a SOC. By relating these characteristics to the needs of different levels of security analysts in the SOC, we propose a decision guide for the choice of the RDF store according to the scenario. We go further by experimentally comparing RDF stores namely Anzograph, GraphDB, Stardog and Virtuoso. This comparison was done on data load times and query response times, with a progressive increase in data size.

The results of our research show that current semantic technologies have evolved sufficiently to meet the needs of companies. However, it should be noted that no single tool was superior on all queries. This shows the impact of ontology and query modelling on performance depending on the tool used.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

ATOM	Abstractions Translation Ontology Method
BGP	Basic Graph Pattern
HIDS	Host Intrusion Detection System
IDES	Intrusion Detection Expert System
IDS	Intrusion Detection System
IoC	Indicators of Compromise
IPS	Intrusion Prevention System
MTTR	Mean-Time-To-Response
NIDS	Network Intrusion Detection System
OLAP	Online Analytical Processing
OLTP	Online Transactional Processing
QMpH	Query mixes per hour
QpS	Queries per second
RDF	Resource Description Framework
SIEM	Security Information and Event Management
SOAR	Security Orchestration and Automation Response
SOC	Security Operations Center
SUT	Systems under Test
SaaS	SOC as a Service
W3C	World Wide Web Consortium

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

Avec la croissante transformation digitale dans les entreprises, tout secteur confondu, de nombreux défis portent sur la protection des systèmes informatiques contre une exploitation mal intentionnée. Ces défis sont encore plus importants en ce qui concerne les grandes institutions, notamment les institutions financières, qui ont des systèmes informatiques complexes et surtout qui gèrent des informations très sensibles. En effet, en raison de la complexité des systèmes et du grand nombre d'utilisateurs, leur surface d'attaque est assez grande et donne ainsi une plus grande opportunité aux cybercriminels. Au Canada, nous pouvons donc noter une large augmentation des cyberattaques sur les dix dernières années, ce qui entraîne des coûts économiques. Selon le rapport annuel de l'Institut Ponemon, publié par IBM Security, le coût d'une brèche de sécurité est passé de 6,26 millions de dollars en 2019 à 6,35 millions de dollars en 2020 [3]. La rentabilité du cybercrime donne plus de motivation aux attaquants, ce qui augmente les défis des défenseurs. Il s'en suit donc un jeu du chat et de la souris, avec attaquants et défenseurs modernisant leur artillerie au fil des ans.

Le principal rôle des défenseurs demeure de protéger les actifs des systèmes qu'ils sécurisent. Les menaces peuvent être externes, mais sont aussi internes, provenant d'erreurs ou même d'acteurs malveillants au sein de l'institution. Au-delà des compétences des défenseurs, de nombreux mécanismes de sécurité sont alors mis en place pour les épauler dans leur tâche, et ainsi assurer la protection contre une intrusion dans le système, et de manière générale contre une utilisation illicite du système.

Au fil des années, ces mécanismes ont évolué continuellement pour s'adapter aux menaces et au contexte des utilisateurs. Dans les années 1960, les principales mesures de sécurité étaient de la sécurité physique pour protéger les données, et des mots de passe pour assurer la confidentialité des données. Puis, avec l'éclatement de la bulle Internet, de nouveaux usages ont ouvert la voie à de nouvelles menaces. Pour les contrer, de nouvelles familles de technologies sont nées : les antivirus et les pare-feux. Le rôle des antivirus était de détecter et corriger les menaces sur les hôtes, tandis que les pare-feux filtraient le trafic entrant et sortant, pour empêcher les menaces de s'infiltrer par le réseau. Ces mesures ont suffi quelque temps à réduire drastiquement les menaces sur les systèmes informatiques. Cependant, la rapide expansion des environnements informatiques a rendu le périmètre à sécuriser plus large et plus complexe. Avec plus d'opportunités et de motivation, l'activité des cybercriminels n'a cessé de progresser. Pour enrayer cette progression, de nouvelles couches de protection étaient nécessaires : les systèmes de détection d'intrusion (SDI).

Les systèmes de détection d'intrusion, en anglais Intrusion Detection System (IDS), sont utilisés pour détecter les activités malveillantes dans un système informatique. Cette détection se fait généralement soit par une analyse comportementale, soit par l'utilisation de règles (signatures par exemple). Parmi les IDS, on distingue deux catégories : ceux qui surveillent l'état de sécurité du réseau, appelés en anglais Network Intrusion Detection System (NIDS), et ceux qui surveillent l'état de sécurité des hôtes, ou Host Intrusion Detection System (HIDS) en anglais.

Ces senseurs sont rapidement devenus insuffisants face au volume croissant d'informations transitant dans le réseau, à la surface d'attaques toujours grandissante et aux cybercriminels plus motivés et outillés. Les senseurs et toute la pléthore d'outils de sécurisation des systèmes ne pouvaient à eux seuls couvrir efficacement l'analyse de toutes les données à surveiller. Rapidement, de nouveaux problèmes viendront remettre en question l'efficacité de ces outils : les faux positifs et les faux négatifs. Les faux positifs sont les alertes provenant de sources non malveillantes et catégorisées comme étant des attaques. La réponse aux incidents sur ces alertes est une perte énorme de ressources pour le personnel de sécurité et réduit le temps passé sur les incidents légitimes. D'autre part, les faux négatifs sont quant à eux l'absence d'alertes sur des activités malveillantes. Ces attaques passent donc inaperçues et menacent directement la confidentialité et l'intégrité des données, ainsi que la disponibilité des services.

Une des raisons menant à ces performances, parfois désastreuses, est l'absence de contexte des informations récupérées par les senseurs. Le traitement individuel des événements ne permet pas de détecter des attaques sophistiquées, se faisant en plusieurs phases, ou sur plusieurs fronts. Pour faire face à ces insuffisances des IDS, les systèmes de gestion des événements et de l'information liés à la sécurité, Security Information and Event Management (SIEM) en anglais, ont été mis en place.

Les SIEM font généralement partie des centres d'opérations et de sécurité de l'information, ou SOC en anglais. Selon Oracle, "le SOC est une plate-forme permettant la supervision et l'administration de la sécurité du système d'information au travers d'outils de collecte, de corrélation d'événements et d'intervention à distance" [4]. C'est donc l'ensemble comprenant le personnel de sécurité et les différents outils et ressources à leur disposition pour gérer la sécurité informatique d'une entreprise. Pour assurer la surveillance en temps réel des incidents de sécurité, le personnel du SOC assure entre autres fonctionnalités le triage des incidents,le *threat hunting* et la réponse aux incidents, en anglais *incident response*. Le triage des incidents consiste à distinguer les alertes légitimes des faux positives et à classer ces incidents suivant leur impact. Ceci permet de prioriser la résolution des incidents. Quant au *threat hunting*, c'est la recherche de menaces qui auraient échappé aux différents outils de détection en place.

Enfin, la réponse aux incidents regroupe les différents étapes de résolution d'un incident, dont les investigations forensiques pour déterminer les causes et les auteurs de l'incident.

Le rôle des SIEM dans le SOC est de centraliser les données venant des différents senseurs du système informatique (systèmes de détection et de prévention d'intrusions, pare-feu, etc.) et les traiter pour séparer le trafic légitime des activités malveillantes. Afin d'améliorer le traitement des données venant des différents senseurs du réseau, des travaux ont été menés pour harmoniser les formats. Les SIEM ont amélioré les performances des analystes de sécurité pendant plusieurs années, mais les attaques devenant de plus en plus sophistiquées, ces dispositifs ont depuis longtemps atteint leurs limites. En effet, il y a un important gouffre entre le volume important de données agrégées dans le SIEM et l'insuffisance de capacités analytiques pour automatiser l'investigation [5]. De plus, il est difficile de configurer et de maintenir les règles de détection dans les SIEM. La tentative d'harmonisation des formats a amélioré la corrélation entre les événements capturés par les différents senseurs et capteurs, mais le manque d'expressivité limite les performances des mécanismes de corrélation existants. La séparation du trafic légitime et des incidents de sécurité dépend grandement des analystes SOC. Dans un SOC, les analystes de sécurité sont chargés d'analyser les données provenant des différents outils de sécurité du réseau, de mener le triage des événements pour prioriser les incidents potentiels, de rediriger les incidents plus importants vers le personnel compétent et de participer à la correction et la récupération des systèmes affectés [6]. Ils sont généralement répartis suivant l'ampleur de l'incident et les compétences nécessaires pour remédier à cet incident.

Selon l'Institut Ponemon, les analystes de sécurité inspectent seulement 29% des alertes soulevées par les outils de sécurité. Parmi ces alertes inspectées, 40% constituent des faux positifs [7]. Face aux besoins des analystes, de nombreuses approches ont été explorées en vue d'améliorer les capacités analytiques dans les SIEM. Parmi celles-ci, nous avons l'intelligence artificielle qui augmente la capacité analytique des analystes de sécurité et permet de détecter plus rapidement les menaces. Parmi les techniques d'intelligence artificielle utilisées pour la sécurité, l'analytique de graphes (*graph analytics*) constitue, selon Gartner [8], une tendance pour le futur, qui devrait croître jusqu'à 100% d'ici à l'horizon 2022. L'analytique de graphes consiste en l'analyse et au traitement de données sous forme de graphes. L'analyse de l'information sous la forme de graphes permet de bénéficier des avantages des graphes, notamment en matière de flexibilité et de performances pour la découverte de connections entre entités dans le modèle étudié [9]. Dans un contexte de sécurité informatique, l'intérêt pour le traitement de graphes et pour les bases de données de graphes vient de l'hétérogénéité des données qui sont de moins en moins structurées. Dans ce contexte, il devient difficile de modéliser et exploiter efficacement une base de données relationnelle englobant les

différentes données. Parmi les modèles de graphes utilisés, nous nous intéressons aux graphes de connaissance. Un graphe de connaissances peut être défini comme un réseau d'entités, leurs propriétés sémantiques et les relations entre ces entités [10]. Ainsi, par sa structure, un graphe de connaissances offre la flexibilité nécessaire à la couverture des données structurées et non structurées. Les graphes de connaissances privilégiés dans notre recherche sont les ontologies. Une ontologie est une représentation de l'information qui, en plus de la flexibilité offerte en tant que graphes, permet également de créer une couche sémantique qui décrit les différents concepts et propriétés d'un domaine, et les relations qui les lient. Le développement de systèmes experts basés sur les ontologies dans un contexte de cybersécurité permettrait donc de répondre aux insuffisances des SIEM.

1.1 Éléments de la problématique

Notre recherche est centrée sur l'amélioration de la détection d'intrusion et du *threat hunting* dans les SOC d'institutions financières. Dans ce contexte de cybersécurité, les graphes de connaissance sont à la fois utiles pour représenter les concepts et processus dans le SOC, mais aussi de détecter des relations entre plusieurs entités. Dans le cadre par exemple de la détection d'anomalies, l'utilisation de graphes accélère le processus de comparaison du comportement de deux individus ayant les mêmes fonctions pour déceler une déviation du comportement. Pour se faire, il est nécessaire de relier les différentes activités menées par chaque individu à travers les fichiers de journalisation, les documents administratifs et tout autre fichier dans des formats hétérogènes. L'utilisation d'une ontologie est donc appropriée pour parcourir tous ces éléments hétérogènes et appliquer des règles d'inférence pour la détection d'anomalie.

Une première phase du projet a consisté au développement d'un système expert basé sur une ontologie pour la détection d'intrusion [1]. Bien que l'utilisation d'ontologies en sécurité informatique reste prometteuse et mentionnée dans plusieurs études [22–25], un des défis très peu étudiés demeure son application dans un environnement réel. En effet, dans un SOC par exemple, le volume quotidien d'informations à stocker et analyser peut facilement avoisiner les pétaoctets (10^{15} octets). Il est donc indispensable, avant de rechercher une solution applicable dans un environnement réel, d'étudier la viabilité de la technologie sous-jacente dans cet environnement. Peu de recherches portent sur l'utilisation d'ontologies pour le traitement de larges quantités d'informations. Pour pouvoir profiter des avantages des technologies sémantiques dans l'amélioration de l'analytique en sécurité informatique, il faut au préalable trouver les bases de données et de traitement ontologiques existantes, qui sont adaptées au contexte en entreprise. Ce sont ces outils qui permettront d'héberger les ontologies et d'ef-

fectuer différentes opérations sur les données hébergées. Ainsi, nous avons étudié ces outils dans l'optique de trouver les meilleures offres sur le marché pour notre contexte d'utilisation dans un SOC. Nous nous sommes rendu compte de la difficulté d'évaluer ces outils de manière objective. En effet, les performances de chaque base de données ontologique sont dépendantes de la configuration, du cas d'utilisation, mais aussi du design du modèle ontologique hébergé et interrogé. Toutefois, cette étude permet dans une certaine mesure d'orienter l'implémentation du système expert, selon le cas d'utilisation et les performances que nous pouvons espérer des outils supportant le système expert.

1.2 Objectifs de recherche

Notre recherche est la suite des travaux de maîtrise de Simon Malenfant-Corriveau et d'Étienne Ducharme [1, 11]. Leurs travaux de recherche ont produit le Abstractions Translation Ontology Method (ATOM), une méthode de développement d'une ontologie utilisable pour un système expert et d'une preuve de concept appliquée à la détection d'intrusion dans un SOC.

Les graphes sont déjà utilisés dans les SIEM pour améliorer la corrélation d'événements. Pour optimiser cette corrélation, nous avons choisi d'utiliser des ontologies pour ajouter une couche sémantique au dessus des données brutes et créer des systèmes experts. Pour étendre l'utilisation des SIEM à un système expert, nous devons transformer les données du SIEM dans un format compatible aux bases de données ontologiques. Les règles de détection du SIEM seront ensuite traduites sous forme de requête SPARQL, le langage de requête sur les bases de données RDF. Cependant, pour atteindre cet objectif, nous devons répondre à des questions liées aux performances des requêtes SPARQL à grande échelle, dans un contexte d'utilisation dans un SOC, à la flexibilité des bases de données ontologiques et à la courbe d'apprentissage dans la création du système expert et dans l'utilisation de ce dernier.

Notre recherche a donc pour objectif de valider l'utilisation des ontologies comme méthode pour améliorer la détection d'intrusion et le *threat hunting* dans un SOC. Pour développer un modèle de détection autour d'une ontologie, nous étudions les performances que les bases de données et de traitements ontologiques peuvent fournir dans un environnement traitant de larges quantités de données. Pour atteindre notre objectif de recherche, nous l'avons subdivisé en quatre principales questions (Q) de recherche :

- Q1 : Quelles sont les bases de données et de traitements ontologiques à grande échelle en termes de caractéristiques pertinentes pour une utilisation dans le contexte d'un SOC ?
- Q2 : Quelles sont les performances relatives des magasins RDF dans le traitement d'importants volumes de données à l'échelle des données transitant dans un SOC ?

— Q3 : Comment évalue-t-on dans le contexte d'un SOC les bases de données et de traitements ontologiques à grande échelle en termes de flexibilité de création de requêtes ?
— Q4 : Quelles sont les performances des outils de cueillette et de traduction d'information des senseurs d'un SOC vers une base de données ontologiques ?

La modélisation de l'ontologie ne fait pas partie du domaine de la recherche. Les modèles ontologiques sont recyclés de travaux antérieurs. De plus, l'évaluation de performance ne tiendra pas compte de la duplication de données et de la gestion efficiente de la base de données ontologiques. La détection, la modélisation des attaques et les faux positifs/négatifs ne font pas partie des travaux.

1.3 Plan du mémoire

Notre mémoire est organisé comme suit. Le chapitre 2 porte sur les différents concepts importants pour la compréhension de nos travaux. Il s'agit de la détection d'intrusion dans le SOC, mais aussi des ontologies en sécurité informatique. Nous présentons ensuite les méthodes d'évaluation de performance d'une base de données sémantiques. Ensuite, dans le chapitre 3, nous présentons les résultats de notre étude comparative des bases de données sémantiques adaptées à un contexte de SOC dans une entreprise. Cette comparaison est qualitative et nous permet de fournir des éléments d'aide à la décision pour le choix d'un outil en fonction des besoins. Par la suite, le chapitre 4 porte sur l'évaluation de performances des bases de données et de traitements ontologiques, l'architecture de notre plateforme d'évaluation de performances et notre méthodologie d'évaluation. Finalement, nous présentons et discutons les résultats dans le chapitre 5, avant de conclure dans le chapitre 6.

CHAPITRE 2 BASE DE CONNAISSANCES ET TRAVAUX ANTÉRIEURS

La détection d'intrusion dans les systèmes informatiques est un sujet très développé, autant dans le milieu académique qu'en industrie. Pour contrer les menaces en cybersécurité, de nouvelles techniques et stratégies de détection sont pensées et implémentées. Ainsi, au fil des années, plusieurs générations d'outils de détection se sont succédées. La nouvelle génération de technologies penche de plus en plus vers l'utilisation de techniques analytiques avancées et de l'intelligence artificielle pour toujours plus d'automatisation. Cette appétence pour l'automatisation est consécutive du volume énorme de données à traiter par les analystes de sécurité. Il est donc primordial de déléguer certaines tâches à un outil intelligent, permettant aux analystes de se concentrer sur les tâches qui nécessitent le plus leur expertise et leur individualité. Dans notre recherche, nous nous intéressons à l'utilisation d'ontologies pour créer des systèmes experts ayant pour rôle d'aider l'analyste de sécurité dans la recherche de menaces dans le système informatique qu'il maintient.

Ce chapitre porte sur les dernières avancées dans la détection d'intrusion dans les *Security Operations Center* (SOC), avec un intérêt particulier pour les techniques analytiques avancées et l'utilisation de l'intelligence artificielle. Ensuite, nous analysons les limites actuelles dans les capacités défensives des SOC. Puis, nous présentons, comme solution étudiée, le concept d'ontologie et les ontologies en sécurité informatique. Enfin, nous abordons l'évaluation de performance des bases de données et de traitement sémantiques, notamment pour l'application dans un environnement ayant un fort trafic de données.

2.1 Détection d'intrusion

Dans son rapport annuel sur le coût des brèches de données, en anglais *data breaches*, IBM indique que 42% des fuites de données au Canada ont été causées par des attaques contre les systèmes informatiques [3]. Le secteur financier est le secteur le plus touché par les attaques.

Pour se protéger des cyberattaques, des mécanismes de sécurité sont mis en place. La détection d'intrusion est un point central des pratiques de sécurité informatique pour détecter les menaces et activer les mesures de sécurité idoines [12]. Ce domaine de la sécurité des systèmes d'information est en constante évolution depuis quelques décennies, en raison de la croissance des cyberattaques. Les origines de la détection d'intrusion remontent à une étude menée par James P. Anderson, en 1980, sur l'audit de fichiers pour détecter des accès non autorisés [13]. Cette étude inspira les travaux de Dorothy Denning et Peter Neumann qui

développèrent le premier prototype de système de détection d'intrusion, en anglais *Intrusion Detection Expert System* (IDES).

On peut classer les types de *Intrusion Detection System* (IDS) selon la plate-forme surveillée ou encore selon la méthode de détection [14].

Selon la plate-forme surveillée, les *Host Intrusion Detection System* (HIDS) qui détectent les attaques se produisant localement sur une machine surveillée et les *Network Intrusion Detection System* (NIDS) qui surveillent le trafic pour détecter les intrusions provenant d'une connexion au réseau.

Quant à la méthode de détection, la plus répandue est la détection par signatures. Cette méthode se base sur les motifs d'une attaque (signature) pour la détecter. Ces signatures sont stockées dans une base de données de signatures. Pour que cette méthode soit efficace, les attaques détectées doivent être connues et leurs signatures déjà importées dans la base de données. Il est donc difficile d'appliquer cette méthode de détection dans un environnement plus large et complexe [15]. En effet, la surface d'attaques devient grande et la précision des techniques de détection par signature baisse alors, avec un plus grand nombre de faux-positifs et de faux-négatifs. Un fort taux de faux-positifs résulte en un gaspillage des ressources de réponse aux incidents. La seconde méthode de détection est la détection d'anomalies. Elle s'appuie sur une base de référence créée en représentant le comportement normal de l'environnement. Ce comportement normal est l'ensemble d'activités légitimes pour lesquelles le système a été construit. La base de référence est comparée à l'état du système au cours du temps pour déterminer des déviations. La plus grande difficulté avec cette technique de détection est la modélisation de la base de référence, surtout dans un environnement large comme celui d'une institution financière. Le nombre d'utilisateurs et d'applications étant très élevé, il est peu probable de réussir à capturer tous les comportements légitimes. Ceci peut également résulter en un grand nombre de mauvaises classifications. Cependant, l'avantage de cette méthode sur la détection par signatures est la possibilité de détecter des attaques inconnues. En effet, ces attaques dévient du comportement normal attendu dans le système.

Comme on peut le voir, les systèmes de détection d'intrusion possèdent des limitations. Pour sécuriser un réseau interne, en plus des IDS, il existe d'autres outils de sécurité tels que les filtres antivirus et les pare-feux. Cependant, dans les entreprises plus larges possédant de nombreux employés et un réseau informatique plus étendu, il est primordial d'avoir une plate-forme centralisée et une équipe dédiée à la sécurité informatique. Le SOC est un ensemble d'outils de sécurité et de personnel qualifié qui ont pour rôle d'examiner les données transitant dans le réseau interne pour déterminer les menaces. En fonction des besoins de l'entreprise, il existe plusieurs modèles de SOC, selon Gartner.

Premièrement, il y a les SOC virtuels, notamment adaptés aux petites à moyennes organisations. Ces SOC sont réactifs et mis en route en cas d'alertes critiques ou d'incidents. Ils ne bénéficient pas de locaux et le personnel est limité et souvent éloigné géographiquement. Adopter un SOC virtuel permet de bénéficier de la centralisation des événements, permettant une meilleure priorisation des incidents, tout en limitant le coût des locaux. L'administration du SOC se fait à distance. Un deuxième modèle de SOC est le SOC multifonction. Il est composé de locaux, matériels et d'une équipe dédiée non seulement à la sécurité mais à d'autres fonctions critiques. En combinant les ressources pour répondre à différents besoins, ce modèle de SOC permet de réduire les coûts de fonctionnement. Il existe aussi les SOC dédiés, en anglais *dedicated* SOC. Ces SOC sont réservés le plus souvent aux grandes entreprises ou aux organisations à haut risque. Ils comprennent des locaux dédiés, une équipe dédiée et sont fonctionnels 24/7. En plus de ces principaux modèles, les SOC en tant que service, en anglais *SOC as a Service* (SaaS), prennent de l'essor sur le marché des SOC. Ce sont des SOC externes qui sont payés pour surveiller et répondre aux incidents d'une organisation ou d'une entreprise. L'entreprise peut limiter la gestion du personnel qualifié nécessaire à la gestion d'incidents, ainsi que les locaux, le matériel et les logiciels nécessaires pour la sécurité.

Indépendamment du modèle choisi, les activités de sécurité dans un SOC s'appuient sur une pléthore d'outils de sécurité. En plus des outils de surveillance des systèmes, de nombreux senseurs et capteurs rapportent sur l'état des réseaux et des systèmes et sur leur utilisation. L'ensemble de ces informations sont agrégés en un seul endroit, le *Security Information and Event Management* (SIEM). Il a été conçu pour permettre la corrélation des informations collectées par les différents senseurs. En corrélant les informations provenant de sources différentes, les taux de faux positifs et de faux négatifs peuvent être réduits. Les analystes de sécurité se servent des fonctionnalités analytiques avancées du SIEM pour analyser de gros volumes de données et déceler des menaces manuellement difficiles à détecter. Il est donc le "couteau suisse" de la réponse aux incidents [16]. Selon Gartner, les trois principaux cas d'utilisation des technologies SIEM sont la détection de menace avancée (*advanced threat detection*), la surveillance et enfin l'investigation et la réponse aux incidents.

En plus du SIEM, une des dernières technologies dans l'écosystème des SOC est l'introduction des *Security Orchestration and Automation Response* (SOAR). Cette solution est destinée aux entreprises les plus matures en termes de sécurisation de leurs systèmes informatiques et qui désirent intégrer quatre groupes d'outils de sécurité en une seule plate-forme : l'orchestration et l'automatisation de la sécurité, la réponse aux incidents et le renseignement, en anglais *threat intelligence*. Le principal avantage d'adopter un SOAR est la réduction du temps de réponse aux incidents, en anglais *Mean-Time-To-Response* (MTTR), grâce à l'automatisation d'une grande partie des processus et à la centralisation des commandes. Cette automatisa-

tion induit aussi une réduction des coûts d'exploitation tout en augmentant l'efficacité des analystes. Ces cas d'utilisation permettent de distinguer les analystes de sécurité suivant leur rôle.

2.1.1 Analystes de sécurité dans les SOC

Dans un SOC, les analystes de sécurité s'aident d'outils de sécurité et d'outils analytiques pour prévenir, détecter et corriger les menaces sur le réseau de l'organisation. Les principales fonctions liées à la réponse aux incidents dans un SOC sont :

— La classification et le triage d'événements. Les différents senseurs et capteurs dans le réseau lèvent des alertes suivant un ensemble de règles configurées. Ces alertes n'indiquent pas nécessairement des attaques. Pour séparer les incidents des faux positifs, des indicateurs de compromission, en anglais *Indicators of Compromise* (IoC), sont analysés dans les activités des utilisateurs, les événements systèmes, les journaux du pare-feu et des routeurs, etc. En plus de ces IoC, des combinaisons d'événements connues peuvent être un signe d'intrusion. Pour réaliser cette fonction, les équipes de sécurité définissent un *playbook* de réponse à l'incident qui représente l'ensemble des étapes que l'analyste doit suivre en fonction de scénarios pertinents pour la compagnie. Pour réaliser ce *playbook*, les analystes peuvent utiliser le framework Mitre ATT&CK. ATT&CK est une base de connaissances constituée de tactiques et stratégies offensives basées sur des scénarios réels. En se basant sur les approches connues d'adversaires potentiels, les analystes peuvent améliorer le *playbook* de l'entreprise et adapter la sécurité globale de l'entreprise.

— La priorisation et l'analyse. Pour assurer la continuité d'affaires, il est important de s'attaquer aux incidents suivant leur impact sur le fonctionnement de l'entreprise. La priorisation consiste donc à analyser les alertes venant des actifs les plus importants en premier. Il est donc nécessaire d'avoir au préalable réaliser un inventaire des actifs et une étude de risque. Ceci permet de compléter le *playbook* de réponse aux incidents avec la priorité de résolution des incidents.

— La remédiation et la récupération. Lorsqu'une alerte est classée en tant qu'incident, suivant la gravité de cet incident, un ensemble de mesures sont prises. Le principal objectif des analystes est de contenir la menace et relancer les systèmes interrompus. Le processus d'éradication de la menace et de correction constitue la remédiation. En parallèle, une investigation sur l'incident (forensique) se fait soit pour évaluer les dommages ou à des fins de poursuites judiciaires. Après avoir contenu l'incident, éradiquer la menace et corriger les vulnérabilités causant la menace, s'en vient la récupération des systèmes endommagés.

— Le *threat hunting*. Les attaques les plus sophistiquées ou les menaces internes peuvent passer outre les règles de détection des senseurs et des capteurs. Cependant, les événements qui transitent dans le réseau sont enregistrés. Ces événements sont donc analysés à l'aide d'outils de sécurité et analytiques pour détecter les menaces cachées. Pour aiguiller l'investigation, le *threat intelligence* rassemble des informations sur les menaces venant de différentes sources comme des forums sur le *dark web*. Ces informations sont analysées et traitées pour comprendre le fonctionnement d'acteurs cybercriminels, des schémas d'attaques et surtout des nouvelles menaces sur l'entreprise. Au-delà de la recherche d'intrusions, le *threat intelligence* permet d'améliorer la posture de sécurité de l'entreprise en s'assurant de détecter ces nouvelles menaces.

Suivant la taille du SOC et la compagnie concernée, il y a trois principaux *Tiers* d'analystes de sécurité qui occupent ces fonctions [17].

Les analystes *Tiers 1* sont souvent des postes junior. Ce sont des spécialistes du triage d'événements. Ces analystes parcourent les différents journaux dans le SIEM à la recherche de schémas qui correspondent à un incident. Ils constituent la première ligne de défense pour l'analyse d'une attaque. Pour répondre aux besoins constants de surveillance du système, ce rôle doit être rempli 24h/24. Des roulements sont alors mis en place pour maintenir un certain nombre d'analystes en poste. Le volume d'informations que ces analystes doivent parcourir peut devenir rapidement tellement consistant qu'ils ne sont pas en mesure de tout traiter [7]. De plus, le triage doit se faire rapidement pour détecter et corriger les incidents le plus rapidement possible et ainsi limiter les potentiels dégâts. Quand un incident se trouve dans leur *playbook* de réponse à l'incident, ils procèdent à l'élimination de la menace suivant la démarche explicitée dans le *playbook*. Les incidents au delà du domaine de leur *playbook*, de leurs compétences ou nécessitant plus d'investigation sont redirigés vers les analystes *Tiers 2*. En plus du triage et de la première réponse à l'incident, ces analystes sont également chargés de la gestion et de la configuration des outils de surveillance et de sécurité.

Les analystes *Tiers 2* sont les spécialistes de la réponse à l'incident. Les incidents remontés par les analystes *Tiers 1* sont examinés à ce niveau. Leur rôle est de procéder à une analyse approfondie de l'incident pour déterminer l'ampleur de la menace et l'étendu de l'infection en vue d'éliminer la cause et entamer la récupération des systèmes altérés. Ils s'aident pour cela des renseignements sur les menaces ciblant leur organisation. De ce fait, ils font également de l'analyse forensique pour déterminer le mode opératoire de l'attaquant, en vue d'éventuelles poursuites judiciaires ou pour mieux empêcher qu'une telle attaque se reproduise. Le volume d'informations traitées par ces analystes dépendra donc de l'ampleur de la menace. En effet, ils vont suivre les différents symptômes de l'infection dans le temps pour déterminer le moment de l'infection et avoir une vue plus complète sur le mode d'infection et l'impact de

l'incident. Cependant, leur analyse est plus ciblée et le volume d'informations traitées sera généralement inférieur à celui des analystes *Tiers 1*. Pour permettre cette analyse, ils s'aideront de nombreux outils analytiques pour visualiser et établir les liens entre les événements dans le réseau et leur investigation.

Les analystes *Tiers 3* sont les spécialistes de la recherche de menace. La principale responsabilité de ces analystes est de trouver les menaces qui ont échappées aux outils de détection. Pour se faire, ils vont analyser les audits de vulnérabilités et s'assurer d'avoir une compréhension à jour des actifs du réseau d'entreprise. Ces audits permettent de circonscrire les points vulnérables (configurations, processus, etc.) pour mieux cibler la recherche. En s'aidant des renseignements sur les menaces avancés, en anglais *advanced cyber threat intelligence*, ils recherchent des indicateurs de compromission du système informatique. Ils participent également à l'optimisation des outils de surveillance en spécialisant les outils selon les menaces déjà trouvées sur le réseau de l'entreprise. N'ayant pas nécessairement d'indices pour délimiter la quantité d'informations à analyser, le domaine de leur recherche est sur la majorité des données enregistrées sur plusieurs mois. Dans son rapport [3] annuel sur les fuites de données publié en 2020, IBM indique que le temps moyen pour détecter une intrusion était de 207 jours. En fonction de la taille du réseau et en fonction de la taille de l'équipe allouée constamment à la recherche de menaces, on peut estimer que ces analystes analyseront les données sur au moins trois mois. La quantité de données concernées, le domaine étendu de la recherche et surtout le manque d'orientations vers une menace spécifique à chercher font que des outils performants d'analyse et de visualisation sont indispensables aux analystes *Tiers 3*. C'est une des raisons pour lesquelles les SIEM offrent déjà une visualisation des relations entre entités et événements sous forme de graphes.

Bien que les SIEM évoluent constamment pour répondre aux besoins des analystes de sécurité, ils possèdent des limites. La principale limite réside en leur capacité à corréler des événements. Les sources de données des SIEM sont les différents outils de sécurité dans le SOC, entièrement dédiés à la sécurité ou possédant des informations importantes d'un point de vue sécurité (pare-feu, routeurs, serveurs, IDS/IPS, etc.). Cependant, des informations primordiales peuvent manquer. En fonction du fournisseur, il peut manquer par exemple les informations contextuelles sur l'état et la topologie du réseau interne. Cette information pourrait aider à diminuer les erreurs de classification des événements, en éliminant les alertes ne pouvant provenir de vraies attaques compte tenu de la configuration du réseau au moment de l'événement. De plus, les SIEM manquent de flexibilité. Il n'existe pas de standardisation dans le développement de SIEM. Par conséquent, l'analyste de sécurité doit apprendre le fonctionnement du SIEM, d'un vendeur à l'autre. Ceci augmente la courbe d'apprentissage et demande une plus grande expertise de la part des analystes. Une autre raison vient du fait

que chaque vendeur se spécialise dans le développement de sa solution. Pour accompagner les entreprises dans le choix d'un SIEM adapté à leurs besoins, Gartner a classé les produits SIEM existants, en fonction de leur efficacité et de leur vision.

2.1.2 Produits existants

Pour présenter un aperçu des technologies SIEM sur le marché, nous nous sommes basé sur les résultats du rapport 2020 de Gartner. L'entreprise Gartner est une compagnie principalement orientée vers la recherche et le conseil. Elle étudie les technologies sur le marché et donne des recommandations aux entreprises pour une meilleure priorisation des processus [18]. Dans son rapport sur les technologies SIEM sur le marché appelé *Magic Quadrant for Security Information and Event Management*, Gartner évalue les vendeurs et leurs produits selon deux principaux critères : la capacité à exécuter et la vision. La capacité à exécuter (*Ability to execute*) est la performance du produit à remplir les fonctions qu'on attend d'un SIEM, à savoir : la détection de menaces, la surveillance, l'investigation et la réponse aux incidents. L'expérience utilisateur et le support sont des éléments essentiels dans l'évaluation de ce critère. Le second critère d'évaluation est la vision de l'entreprise (*Completeness of vision*). Ce critère englobe l'habilité à comprendre l'évolution des menaces et du besoin en sécurité, et à proposer des produits innovants qui répondront aux attentes présentes et même futures. L'importance de ce critère réside en la durée et l'effort de développement d'un produit de sécurité. Sans prédire les futurs besoins des consommateurs pour les intégrer à l'avance dans la solution, il est très probable que le produit soit rapidement à la traîne face aux compétiteurs ou pire face à de nouvelles menaces. Ainsi, Gartner évalue les vendeurs suivant ces deux critères pour ensuite les classer dans quatre grandes familles : les leaders, les challengers, les visionnaires, et les niches. Les "leaders" regroupent les vendeurs qui réussissent à la fois à prédire les besoins du marché, mais aussi à remplir au mieux les requis d'un SIEM. Les "challengers" sont des vendeurs ayant des technologies SIEM robustes, mais encore incomplètes, sans une bonne vision du marché futur. Les "visionnaires" sont des vendeurs qui fournissent de vaste éventails de fonctionnalités dans leurs SIEM, en tenant compte de l'évolution des besoins, mais qui ne possèdent pas la maturité pour toucher une plus large clientèle. Les "niche players" sont composés de vendeurs spécialisés vers un cas d'utilisation de leurs technologies, avec des capacités limités pour le reste des cas d'utilisation.

Dans cette section, nous allons présenter un représentant de chacune des catégories de vendeurs classés dans le rapport 2020, ainsi que l'évolution opérée par ces représentants depuis le rapport Gartner 2018 [19].

Fireye Dans la catégorie des “niches”, Fireye est une compagnie de sécurité fournissant *FireEye Helix*, le principal élément de sa technologie SIEM. Il s’intègre avec d’autres produits de Fireye pour la sécurité du réseau, des points d’extrémité, en anglais *endpoints*. Cette intégration offre des capacités analytiques additionnelles lors d’une investigation. Fireye fournit un support avec sa solution sous forme de guides pour les investigations, mais aussi d’une expertise à la demande pour assister ses clients dans l’investigation d’alertes. Le faible nombre de fonctionnalités et sa faible adoption sur le marché des SIEM classe cet outil dans la catégorie des “niches”. Il marque sa première apparition dans le classement de Gartner.

IBM Dans la catégorie “leader”, IBM a développé une plate-forme de sécurité et d’intelligence autour de son SIEM Qradar. Dans cette plateforme, IBM propose sous forme de licences, un gestionnaire de vulnérabilité, une application pour l’inspection du contenu de paquets réseaux, un gestionnaire de risque, une solution d’analytiques du comportement utilisateur (*User Behavior Analytics* ou UBA), un support d’investigation forensique et une solution d’analytiques avancées appelée *IBM Qradar Advisor with Watson*. Cet écosystème SIEM touche aux différents besoins d’une entreprise. Il est donc placé par Gartner dans la catégorie “leader”. Depuis le dernier rapport, les changements opérés sur ce SIEM sont principalement sur l’efficacité d’alertes et aussi une intégration du framework MITRE ATT&CK à son composant d’analytiques avancées *Qradar Advisor with Watson*. Malgré tous ces points positifs, IBM Qradar est limité sur les options de collecte de données venant des *endpoints* et des machines hôtes. De plus, l’expérience utilisateur est le principal retour négatif des utilisateurs.

LogPoint Dans la catégorie “visionnaire”, la compagnie LogPoint propose une solution SIEM en quatre modules : le SIEM de base, un module UEBA, un module d’analytiques avancées, et un module de gestion des différents systèmes LogPoint pour les installations réparties. La principale force de ce SIEM est son offre de fonctionnalités concernant la vie privée (*privacy*). En étant le seul SIEM à obtenir la certification sur la vie privée *Common Criteria EAL 3+*, LogPoint est passé d’une solution “niche” en 2018 à une solution “visionnaire” dans ce nouveau classement. Les améliorations majeures depuis le rapport 2018 concernent l’investigation des incidents, avec de meilleures capacités de visualisations et d’exploration de données (*data mining*). Ce SIEM manque cependant de flexibilité quant à la collecte et à la traduction depuis des sources de données personnalisées. Certaines intégrations ne sont pas supportées et demanderont un effort de développement au consommateur.

Dans le rapport 2020 de Gartner, il ne figurait aucun “challengers”. Ceci peut s’expliquer par l’évolution rapide des menaces qui donne beaucoup de poids à la capacité à innover

et à s'adapter d'une entreprise. En résumé, les différentes technologies SIEM sur le marché ont deux principales limites. Premièrement, elles manquent de flexibilité. L'intégration avec d'autres solutions de sécurité ou avec les différentes technologies déjà existantes dans un réseau est souvent difficile ou impossible. De plus, l'augmentation des capacités d'une offre SIEM est également limitée quand cela est possible. Les acteurs malveillants augmentent rapidement leurs capacités offensives, imposant aux défenseurs de pouvoir modifier et faire évoluer les outils de sécurité. Un manque de flexibilité rend difficile cet objectif. La deuxième limite est l'expérience utilisateur. Dans un SIEM, d'importants volumes de données, venant de sources diverses, vont être stockés. Les analystes de sécurité ont besoin d'un outil à la fois simple, précis et puissant pour parcourir et mener une investigation sur ces données, dans un temps raisonnable. La difficulté et le temps nécessaire pour prendre en main et exploiter les différentes fonctionnalités du SIEM sont donc des facteurs décisifs. Selon les besoins d'un SOC, le fournisseur de SIEM peut changer, conduisant à un nouvel apprentissage des spécificités du nouveau SIEM. Cette courbe d'apprentissage existe toujours et est un des éléments qui est souvent ressorti dans les avis des consommateurs. Il faut noter qu'au delà de la courbe d'apprentissage, l'absence de fonctionnalités pour simplifier ou améliorer l'analytique est également un des principaux axes de développement des SIEM.

Face à ces insuffisances, nous avons choisi d'étudier l'intégration des ontologies dans le développement d'outils de détection d'intrusion dans les SOC.

2.2 Les ontologies

La prévision des tendances technologiques jusqu'en 2022 selon Gartner place en 4^{e} position l'analytique des graphes, en anglais *graph analytics*. L'utilisation de graphes, en particulier de graphes de connaissances, permet "d'accélérer la préparation des données et d'appliquer une science de données plus complexe et adaptée". En effet, les graphes sont particulièrement efficaces pour traiter des données hétérogènes, à la fois structurées et non structurées [8]. Dans notre recherche, nous nous intéressons aux ontologies pour faire face aux limites des SIEM.

L'ontologie est une forme de représentation de la connaissance, généralement sous forme de graphes orientés. Elle fait partie des technologies sémantiques qui ont pour objectif de dériver du sens des données, de sorte à permettre à une machine d'interpréter cette information. Parmi les différentes définitions de l'ontologie, Gruninger (1996) la définit comme "une compréhension commune de certains domaines d'intérêt qui peuvent être utilisés comme un cadre unificateur pour résoudre les problèmes" [20]. Une ontologie revient donc en la représentation du monde (ou d'une partie) comme étant un ensemble de concepts, leurs définitions

et leurs relations. Cette “conceptualisation” du monde permet d’améliorer la compréhension du domaine et la communication d’informations, mais surtout crée une couche sémantique compréhensible par une machine. Cette dernière caractéristique de l’ontologie forme la base du web sémantique, défini dans les standards de la *World Wide Web Consortium* (W3C).

Selon Gartner, la nouvelle vague d’utilisation de l’ontologie est orientée vers l’automatisation d’une tâche en utilisant les propriétés de l’ontologie pour créer des systèmes experts. Ceci vient des capacités de raisonnement associées aux ontologies. Il est donc possible de modéliser un ensemble de règles qui permettra d’automatiser des tâches répétitives. En plus, en créant une couche sémantique compréhensible par une machine, on est capable de créer des systèmes experts.

Nous avons quatre principaux types d’ontologies [21]. Les ontologies *top-level* qui définissent des concepts généraux comme la localisation. Les ontologies de *domaine* sont des ontologies spécialisées vers un domaine en particulier. Elles décrivent les différentes entités, les attributs et les relations dans ce domaine. Les ontologies de *tâche* (*task ontology*), qui décrivent une tâche donnée. Les ontologies *applicatives* (*Application ontology*), qui décrivent des vocabulaires liés à un domaine et à une tâche. En combinant des vocabulaires de plusieurs domaines, ou du même domaine mais de sources de données différentes, l’ontologie permet de créer un pont entre domaines et facilite l’utilisation de données hétérogènes.

Indépendamment du type d’ontologies, les informations stockées dans le graphe de connaissances sont sous forme de triplets sujet-propriété-objet. L’ensemble de triplets peut être enregistré dans un fichier RDF sous plusieurs syntaxes telles que RDF/XML, N-triples, Turtle, Notation 3, RDFa et JSON-LD. Le Resource Description Framework (RDF) est un ensemble de standards du World Wide Web Consortium (W3C) utilisé dans le Web sémantique. Les différentes syntaxes représentent différents formats de fichiers. Cette diversité de formatage permet la compatibilité avec différentes applications.

2.2.1 Les ontologies en sécurité

Les avantages des ontologies dans la gestion de données à la fois structurées et non structurées ont motivé la recherche vers l’exploitation de modèles ontologiques en sécurité informatique. Dans cette section, nous présentons des ontologies appliquées à des problématiques de sécurité.

Coppolino *et al.* (2009) augmentent la détection d’intrusion en y ajoutant un système de diagnostic basé sur l’ontologie [22]. Leur approche, au delà de la détection d’une menace en corrélant les événements de plusieurs sources, vise à expliquer les causes de l’attaque et

aussi d'en évaluer l'impact. En trouvant les vulnérabilités et les événements associés à une attaque, les analystes de sécurité pourront plus efficacement mitiger la menace et expliquer les conséquences de l'attaque sur le système. Pour mener le diagnostic, ils développent un framework pour comparer les événements d'une possible intrusion à des schémas d'événements stockés dans leur ontologie. Leur évaluation sur un banc de test leur permettrait d'augmenter la précision de la détection dans le scénario choisi et aussi d'avoir une explication de haut niveau des vulnérabilités associées.

Granadillo *et al.* (2012) proposent une modélisation des structures de données et opérations d'un SIEM pour apporter une dimension plus sémantique aux informations et assurer l'interopérabilité de formats existants [23]. Ils ont divisé leur base de connaissances en deux classes : une classe "Information" englobant les informations contextuelles et les événements, et une classe "Opération" traitant des actions à mener (corrélation, simulation) pour détecter et adresser une intrusion. Leur framework est ensuite évalué contre une attaque par un *botnet* (infection, contrôle de la machine, exploitation de la machine). Cette évaluation et les contre-mesures sont toutes faites de manière théorique, en présentant les règles logiques de détection et de mitigation de l'attaque.

Shenbagam et Salini (2014) ont développé une ontologie pour classer les attaques sur les applications Web dans le but de prédire les attaques sur un système et suggérer des correctifs [24]. Leur approche a été d'importer et compléter des vocabulaires sur les biens, menaces, vulnérabilités, méthodes de prévention et contre-mesures. En créant des contraintes définissant les différents scénarios à détecter et les actions à prendre en fonction du scénario, ils ont proposé une architecture pour prédire des attaques. Ils n'ont cependant pas fourni une évaluation de leur architecture dans des conditions réelles.

Huang et *al.* (2014) proposent une méthode d'analyse comportementale des logiciels malveillants, basée sur une ontologie [25]. En créant un bac à sable pour exécuter des logiciels malveillants, ils sont capables de récupérer les principales informations englobant le comportement des logiciels infectés testés. Ils s'intéressent aux modifications du registre, aux modifications des fichiers et aux connections réseau. Leur approche est intéressante pour classer les comportements, mais ils n'ont pas proposé d'utilisation concrète de leur système.

Ducharme (2017) et Malenfant-Corriveau (2017) ont développé DIOSE (détection d'intrusion à l'aide d'un système expert basé sur l'ontologie) et ATOM (*Abstractions Translation Ontology Method*), une méthode de développement de système expert basé sur l'ontologie [1, 11]. Leurs travaux se sont basés sur les travaux de Sadighian (2015) [26]. Nos travaux sont subséquents à leurs travaux. DIOSE est un système de détection d'intrusion, en simulant le comportement d'un SIEM à l'aide d'une ontologie. La base de connaissances joue donc le

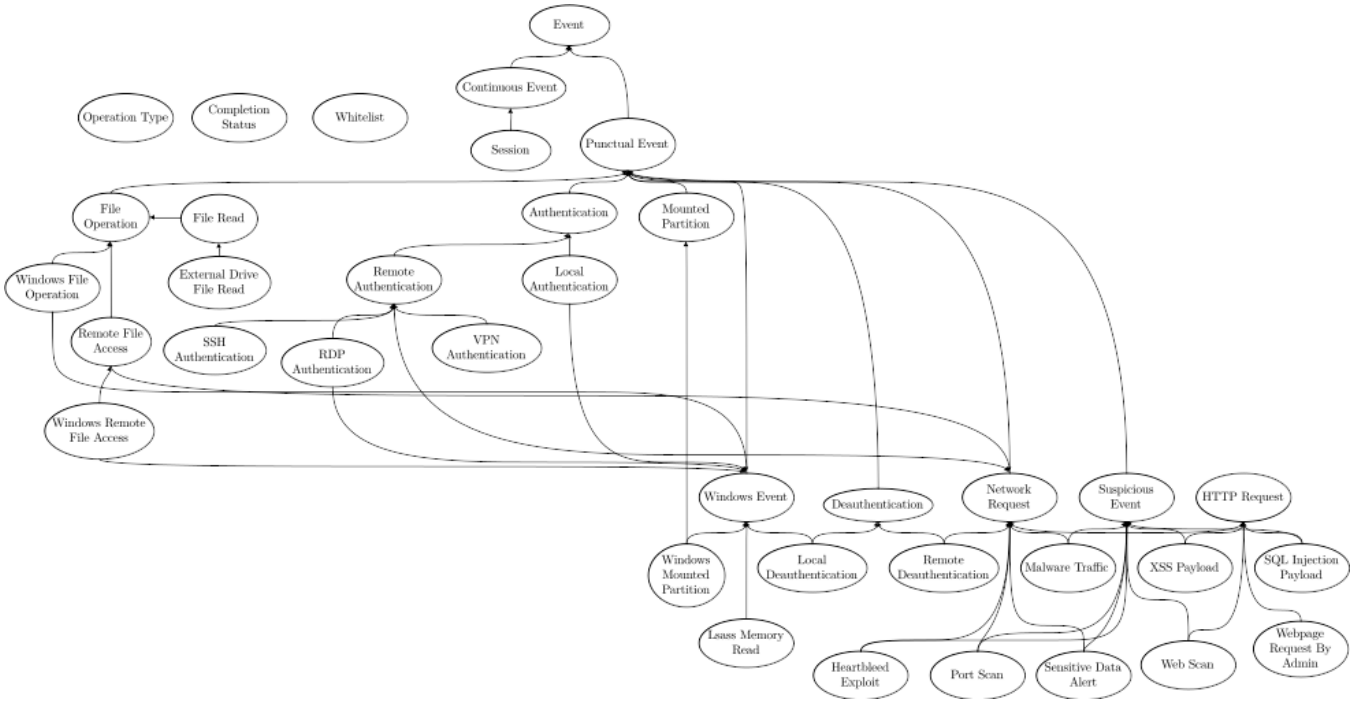


FIGURE 2.1 Classes liées au concept d'événement [1]

rôle d'agrégateur des flux d'informations des senseurs et aussi, à l'aide des capacités d'inférence et de raisonnement sémantiques, assume la corrélation d'événements. Ils s'attaquent avec leur système expert au manque d'expressivité et de flexibilité des SIEM traditionnels. Les trois principaux concepts de la base de connaissances sont les événements, le contexte et les vulnérabilités. À la figure 2.1, nous pouvons observer les classes liées au concept d'événement. En créant des machines à états constituées d'un ensemble de règles (requêtes SPARQL, raisonnement), Ducharme présente des scénarios de détection d'attaque. À la figure 2.2, nous avons une requête SPARQL entrant dans le processus de détection d'une attaque Web. Néanmoins, leurs travaux ne sont pas testés dans un environnement de production demandant des capacités d'ingestion de données et de requêtes vers la base de données similaires au SIEM d'un SOC. Nous allons donc étudier les méthodes d'évaluation de performance des bases de données ontologiques pour les comparer aux requis de performance dans un SOC.

Onwubiko (2018) propose CoCoa, *Cybersecurity Operations Centre Ontology for Analysis*, une ontologie développée selon le framework de cybersecurité du *National Institute of Standards and Technology* (NIST) [2]. Leur framework a pour objectif de combiner les journaux des senseurs traditionnels aux renseignements (threat intelligence) de plusieurs types de flux d'informations différents. Une nouveauté par rapport aux autres ontologies étudiées dans la littérature est l'intégration de données non structurées telles que des photographies et

```
SELECT * WHERE {
        ?event1 rdf:type :XSSPayload ;
                :hasDestination ?webpage ;
                :hasTimestamp ?time1 ;
                :hasSource ?machine .
        ?event2 rdf:type :WebpageRequestByAdmin ;
                :hasDestination ?webpage ;
                :hasTimestamp ?time2 .
        ?webpage rdf:type :Webpage .
        ?website :hasWebpage ?webpage
        FILTER (?time1 < ?time2)
}
```

FIGURE 2.2 Requête SPARQL pour la détection d'une attaque XSS [1]

des journaux de contrôle d'accès physique. Ils diversifient ainsi les sources de données pour augmenter les capacités d'analyse et de détection. Enfin, ils développent un guide analytique (*playbook*) pour la détection et la mitigation de menaces. Les principales étapes sont : la collecte d'informations des différentes sources, la surveillance à travers les différents senseurs et les procédures et processus de réponses à l'incident. Leur ontologie contient non seulement les informations pour la détection, mais également les processus de l'entreprise pour répondre aux menaces. À la figure 2.3, nous pouvons observer quelques classes et relations de cette ontologie. Les concepts définis sont de très haut niveau. En fonction du cas d'utilisation, des sous-classes doivent être créées pour couvrir les concepts plus proches des données. Ainsi, à la figure 2.4, le graphe de connaissances permet de catégoriser des informations venant de sources à la fois structurées et non structurées.

Narayanan *et al.* (2018) s'attaquent à la détection anticipée d'événements de sécurité [27]. Ils proposent un framework basé sur l'ingestion d'informations des différents senseurs d'un système, mais aussi des renseignements sur les menaces, en anglais *threat intelligence*. Leur proposition est construite autour d'ontologies de sécurité existantes telles que *Unified Cy-*

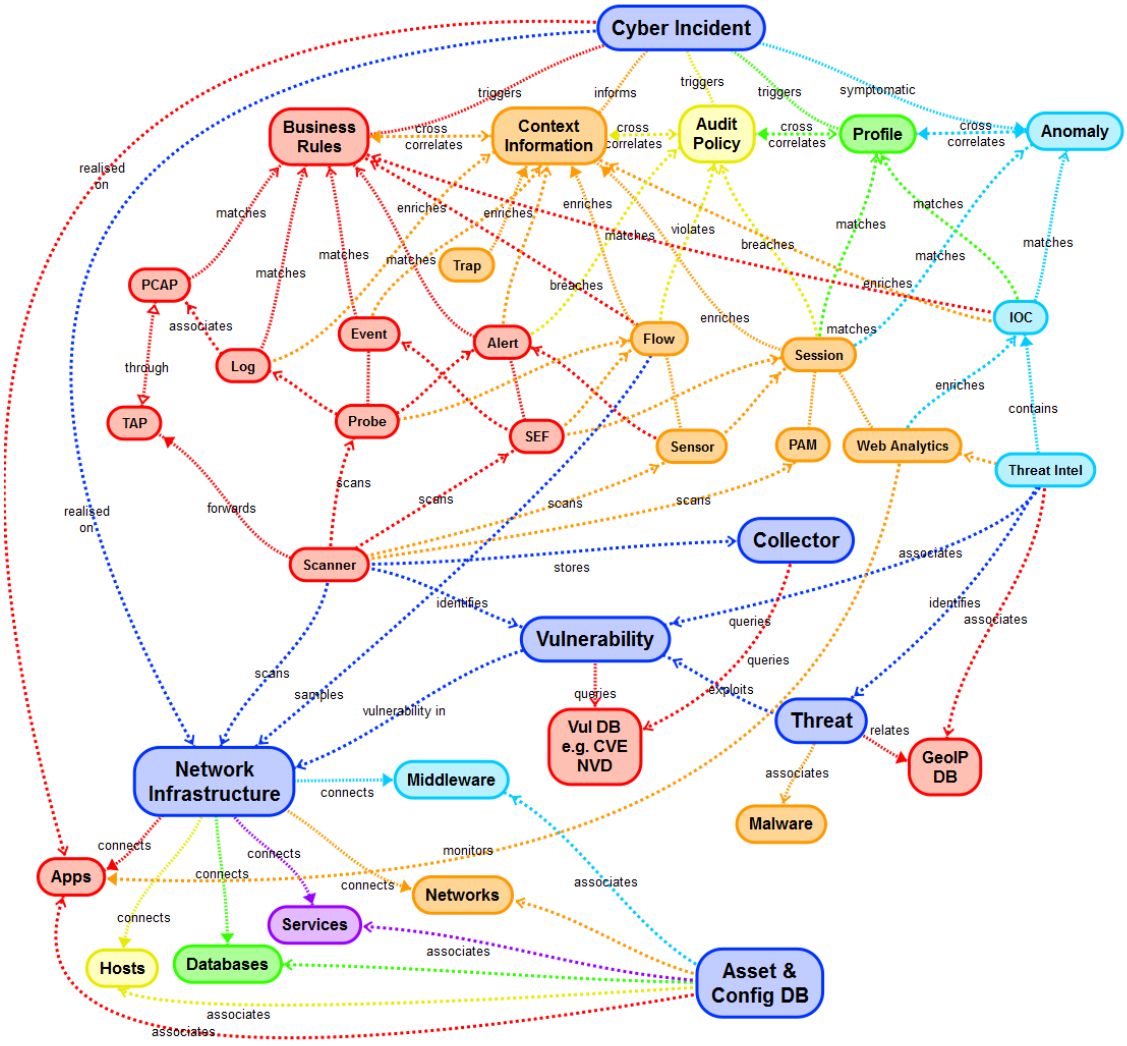


FIGURE 2.3 Graphe de connaissances sur la réponse aux incidents dans un SOC [2]

bersecurity Ontology (UCO). UCO est une ontologie représentant les différents formats de communication de renseignements sur les menaces informatiques. Les chercheurs veulent à travers ce framework aider les analystes de sécurité en intégrant dans une même interface les données permettant de détecter les attaques et les variations d'attaques connues. Pour démontrer l'efficacité de leur module cognitif de cybersécurité, en anglais *cognitive cybersecurity module*, ils déploient une architecture simulant un système réel, qu'ils infectent avec un rançongiciel. Leur scénario de détection inclut les étapes de renseignements sur le rançongiciel depuis des sources textuelles, à la population de la base de connaissances ontologiques avec les indicateurs d'attaques.

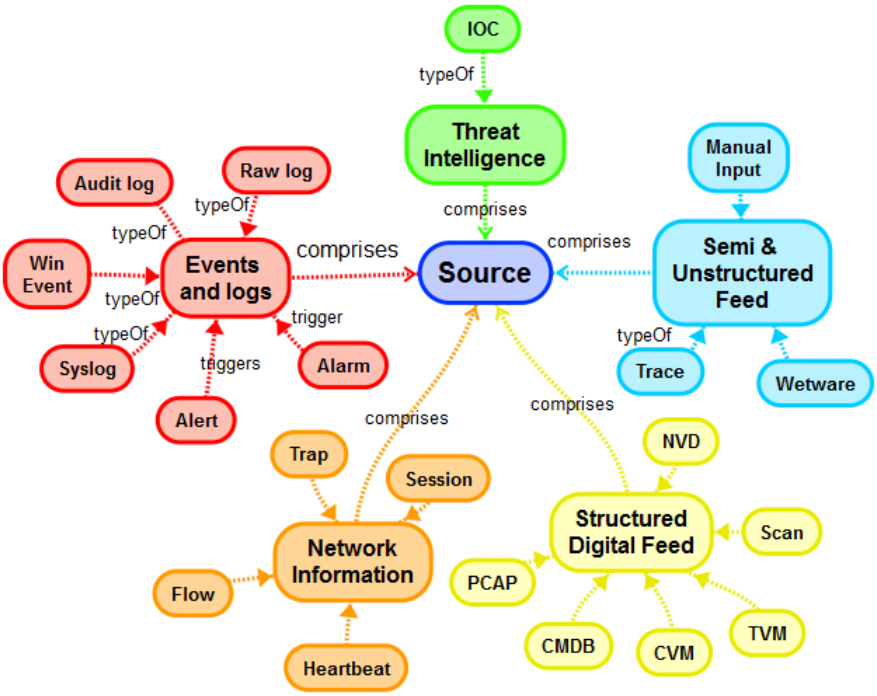


FIGURE 2.4 Ontologie de catégorisation de sources de journalisation [2]

2.2.2 Évaluation de performance de bases de données ontologiques

L'adoption massive des graphes sémantiques est ralentie par la barrière technique, nécessaire à leur développement, selon Gartner. Au delà de ce gap dans le développement d'ontologies, il y a également les limites de performance des graphes de connaissance face à un volume important de données. Pour étudier la viabilité de ces solutions dans un environnement de production, l'évaluation des performances, en anglais *benchmark*, des bases de données sémantiques est primordiale. Dans cette section, nous présentons quelques notions et recherches qui nous ont été utiles dans la compréhension et l'évaluation de performances des magasins RDF, en anglais *RDF stores*.

Plusieurs facteurs entrent en jeu dans l'évaluation de performance des bases de données RDF. Le modèle ontologique, les requêtes SPARQL et l'architecture de la base de données RDF en sont des exemples. Dans une étude des composants clés des bases de données RDF [28], Ali *et al.* (2020) présentent les techniques de stockage, d'indexation, les langages supportés et les stratégies d'optimisation de génération de requêtes. Ces différents facteurs ont un rôle important dans les performances des bases de données.

Le stockage des données joue un rôle majeur dans n'importe quelle système de gestion de bases de données. Les mécanismes de stockage de données répertoriés sont les tables de triplets, les tables de propriétés, le partitionnement vertical, le stockage de graphes et le stockage varié. Les tables de triplets consistent en un stockage des données dans une seule table contenant les sujets, les prédicats et les objets des triplets RDF. Cette stratégie devient rapidement coûteuse avec l'évolutivité des données. Le stockage dans des tables de propriétés consiste à stocker les données dans des tables différentes correspondant aux propriétés des triplets RDF. Chaque table comprend donc une colonne des sujets et n colonnes représentant les propriétés correspondant au sujet en question. Quoique performant pour les jointures sujet-sujet, cette stratégie est moins efficace quand il s'agit des autres jointures sujet-objet, objet-objet et de combinaisons de jointures différentes. Dans le partitionnement vertical, pour chaque propriété, une table est créée avec deux colonnes représentant les sujets et objets de cette propriété. Cette stratégie est efficace pour les jeux de données RDF volumineux.

L'indexation a un rôle dans l'efficacité d'extraction des triplets solutions d'une requête SPARQL. Parmi les stratégies existantes, nous avons l'indexation par prédicat qui correspond au stockage par partitionnement vertical. Chaque table partitionné est indexée par le prédicat associé. Ensuite, nous avons l'indexation par permutation de triplets qui consiste à indexer sur les différentes permutations de triplets. Puis, l'indexation en utilisant un système de gestion de bases de données existant.

En plus de ces éléments détaillés, les chercheurs présentent un état de l'art de framework d'évaluation de performance de bases de données RDF. Leur article permet de comprendre les différentes architectures des bases de données RDF et les méthodes d'évaluation des bases de données RDF. À la fin de leur article, ils constatent le besoin d'une évaluation de performance sur les nouvelles bases de données RDF.

Aluç *et al.* présentent un framework d'évaluation de performance des bases de données RDF [29]. Ils observent que les *benchmarks* existants ne sont pas adaptés car ils ne comprenent pas des caractéristiques ou des requêtes assez variées. Pour appuyer leurs arguments, ils modélisent deux classes de caractéristiques de requêtes qui permettraient une meilleure évaluation des bases de données, et testent quelques *benchmarks* populaires suivant ces classes. Les principales caractéristiques sont la structure des requêtes et les caractéristiques liées au jeu de données. La structure d'une requête SPARQL peut prendre plusieurs formes, indépendamment des résultats de la requête. La forme de la requête est liée au schéma du graphe, en anglais *Basic Graph Pattern* (BGP), un ensemble de schéma de triplets qui constitue la base du standard SPARQL [30]. Les résultats de la requête SPARQL sont trouvés en faisant correspondre les schémas de triplets du BGP au graphe de connaissances RDF en entier. La

manière dont cette correspondance se fait et est optimisée d'une technologie à une autre, impacte la performance des requêtes SPARQL. Les chercheurs considèrent quatre principales caractéristiques. En premier lieu, le nombre de schémas de triplets dans une requête (un ensemble de BGP) sépare les requêtes simples des requêtes complexes. Les sommets de jointure étant les variables et termes dans un BGP, la seconde caractéristique est le nombre de sommets de jointure, en anglais *join vertex count*. Ensuite, nous avons le degré des sommets de jointure, en anglais *join vertex degree* qui a une incidence sur la complexité de traitement de la requête par la base de données RDF. Enfin, le type de sommets de jointure (sujet-sujet, objet-objet ou sujet-objet) est le dernier facteur structural.

Les facteurs liés au jeu de données tournent autour de la cardinalité des entités et de la manière dont ces entités seront indexées et traitées dans une requête. En effet, l'ordre dans lequel les jointures des BGP se font dépendent de plusieurs paramètres d'optimisation inhérents à la base de données testée. C'est donc un facteur important à considérer quand on veut tester de grosses quantités de données. Les chercheurs ont testé quatre benchmarks SPARQL sur ces caractéristiques : le *Lehigh University Benchmark* (LUBM), le *Berlin SPARQL Benchmark* (BSBM), le *SP2Bench* et le *Dbpedia SPARQL Benchmark* (DBSB). Ils concluent que les différents *benchmarks* testés permettent d'évaluer chacun une partie des facteurs retenus, mais aucun de ces *benchmarks* n'est assez complet pour couvrir tous les facteurs simultanément. Pour y remédier, ils proposent un ensemble d'outil constituant leur *benchmark*, *Waterloo SPARQL Diversity Test Suite* (WatDiv). Ce framework a pour but de générer des *benchmarks* plus diverses dans les requêtes mais aussi dans la charge. Il est constitué d'un générateur de jeu de données, d'un générateur de modèles de requêtes, d'un générateur de requêtes suivant les modèles générés et d'un extracteur de caractéristiques qui évalue des bases de données suivant les facteurs présentés ci-dessus. Pour conclure leur recherche, ils ont évalué quelques bases de données RDF avec WatDiv. Cette évaluation leur a permis de conclure qu'aucun des systèmes évalués est le plus performant face à tous les types de requêtes et de charges. Cette conclusion renforce notre motivation à faire un état de l'art des bases de données RDF à grande échelle, pour permettre un choix de la base de données adéquate dans un scénario précis.

Les travaux de Morsey *et al.* [31] portent sur l'évaluation de performance des magasins RDF dans le but de soutenir le développement du Web sémantique. Cette évaluation de performance aiderait à détecter des insuffisances des magasins RDF. Le *benchmark* qu'ils proposent est basé sur les journaux (*logs*) de DBpedia [1] qui contiennent les requêtes d'utilisateurs. À partir de l'historique d'utilisation de DBpedia, les chercheurs proposent une évaluation avec

1. DBpedia est une base de données RDF regroupant les informations trouvées sur Wikipedia, sous forme de graphes de connaissance orientés.

des requêtes réelles pour simuler une utilisation se rapprochant du contexte réel. Ils mettent à disposition des outils pour générer un jeu de données de taille variable tiré de DBpedia, ainsi qu'un ensemble de requêtes basées sur 25 modèles de requêtes. Ils testent leur *benchmark* sur Virtuoso, Sesame, Jena-TDB et BigOWLIM (maintenant GraphDB). Les métriques de performance sont le nombre de requêtes par heure, en anglais *Query mixes per hour* (QMpH) et les requêtes par seconde, en anglais *Queries per second* (QpS). Ils trouvent que Virtuoso a la meilleure performance d'ensemble, suivi de GraphDB.

Salehpour et Davis comparent les *benchmarks* de graphes de connaissance [32]. Ils s'intéressent à la performance des systèmes de gestions de données (SGM) en fonction de la politique de stockage de données (en lignes, en colonnes, en graphes ou sous forme de documents). Ils comparent Virtuoso, Blazegraph et MongoDB en utilisant comme *benchmark* WatDiv, Fish-Mark, BowlognaBench et BioBench-Allie. Dans leur recherche, ils définissent de manière claire et concise des concepts importants pour la compréhension des mécanismes de requêtes SPARQL. Contrairement aux autres recherches étudiées, les chercheurs analysent avec une granularité plus fine les différents facteurs pouvant influencer la performance d'un SGM face à un type de requêtes. Ils arrivent à la conclusion qu'aucun des SGM est le plus performant suivant chaque type de jointure (sujet-sujet, sujet-objet, jointure en arbre). Leurs conclusions soutiennent les recherches précédentes sur la supériorité en performance d'un SGM en fonction du type de requête et de la taille du jeu de données (évolutivité).

Framework d'évaluation

Dans cette section, nous comparons les frameworks de benchmarks que nous avons trouvés dans la littérature. Il y a deux types d'outils : les outils automatisés qui permettent de mener un test de bout en bout et les outils qui génèrent seulement des requêtes et des données. Pour cette dernière catégorie, les tests sont manuels.

L'utilisation de benchmarks existant permet de traiter différents aspects liées aux données (taille de l'ontologie, complexité de l'ontologie, complexité des requêtes), ainsi que la manière d'interagir avec le SGBD.

SP2Bench SP2Bench teste les performances des magasins RDF à répondre aux requêtes SPARQL, en se basant sur des scénarios issus de la base de données DBLP. La base de données DBLP est un graphe RDF d'informations bibliographiques issues de journaux en informatique. SP2Bench comprend un générateur de données de taille arbitraire et des requêtes spécifiquement construites pour couvrir différentes caractéristiques de l'évaluation. Il ne permet pas d'automatiser la phase de test.

Le générateur de données permet de créer jusqu'à trois milliards de triplets, soit environ 300 Go de données. Lors d'un test, les métriques mesurées sont le taux de succès, le temps de chargement des données dans le magasin RDF, les performances par requête, les performances d'ensemble et la consommation de mémoire.

SP2Bench a été publié en 2008. Il n'évalue pas les performances liées à l'inférence. En plus du manque d'automatisation, ce framework est assez vieux et n'est pas représentatif des cas d'utilisation actuels.

IGUANA IGUANA est un projet de recherche développé pour permettre l'évaluation de bases de données de graphe. L'objectif de ce projet était de créer un benchmark suffisamment flexible pour être compatible avec les *endpoints* HTTP mais aussi les opérations en ligne de commande. Contrairement aux autres frameworks analysés, celui-ci a la particularité d'évaluer également les performances de lecture et d'écriture.

Ce framework ne comprend pas de jeux de données ni de requêtes développés pour le test des magasins RDF. C'est un outil d'automatisation qui permet d'effectuer des tests hautement paramétrables. Il comprend les métriques classiques d'évaluation et permet d'étendre les métriques disponibles assez facilement.

Waterloo SPARQL Diversity Test Suite (WatDiv) WatDiv est un framework d'évaluation des bases de données RDF. Il a été développé en 2014. Ce benchmark est composé d'un générateur de données et d'un générateur de requêtes à partir de modèles. Le principal apport de ce benchmark est l'introduction d'une plus grande variété de requêtes SPARQL, en considérant les caractéristiques structurales et les classes de sélectivité des requêtes. De plus, il est suffisamment flexible pour permettre l'utilisation d'un jeu de données quelconque. En ajustant le facteur de taille de WatDiv, on peut choisir le nombre de triplets et surtout la complexité du jeu de données de test à générer.

WatDiv permet de réaliser deux types de tests : les tests basiques et les tests de conditions limites (en anglais *stress testing*). Lors de tests basiques, la base de données est interrogée avec quatre catégories de requêtes : des requêtes linéaires (L), des requêtes étoilées (S), des requêtes en flocon (F) et des requêtes complexes (C) (voir figure 2.5). Suivant la catégorie de requêtes, le type d'opérations et le nombre d'opérations pour retrouver les triplets correspondant varient en fonction de l'architecture du magasin RDF. Ainsi, en variant le type de requêtes et le nombre de triplets durant les tests basiques, WatDiv permet de tester l'aptitude des magasins RDF à traiter plusieurs scénarios différents et les performances des magasins RDF suivant ces cas. Le *stress testing* vise à déterminer le comportement des bases de données dans des conditions extrêmes et mettre en évidence des limites dans la modélisation des magasins RDF.

WatDiv est un des frameworks les plus récents dans la recherche. Il s'est construit en partant des limites des benchmarks précédents et permettrait de tester une plus grande variété de caractéristiques. Nous avons donc choisi ce framework pour la génération des données et la génération de requêtes pour nos tests.

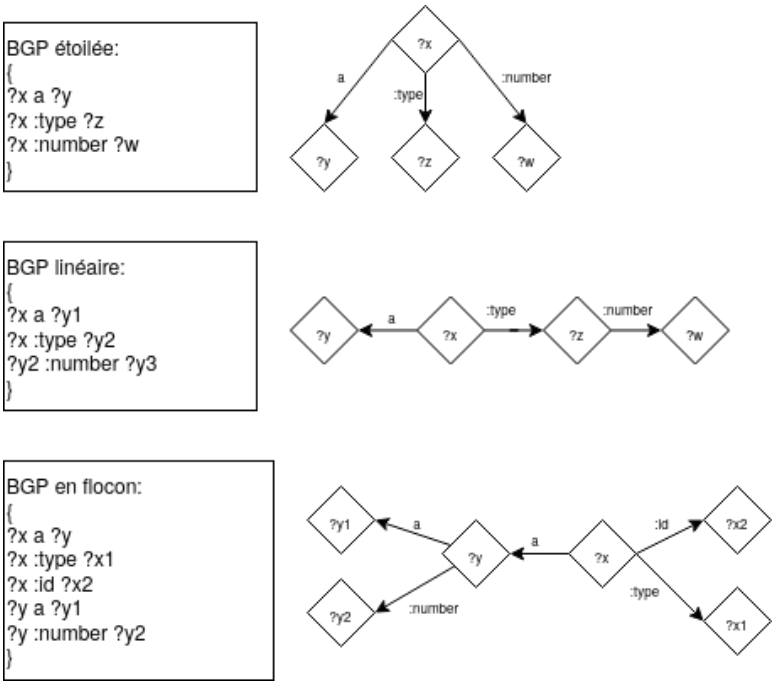


FIGURE 2.5 Catégories de requêtes dans WatDiv

SPARQL Query Benchmarker Ce framework [2] a été écrit en Java pour supporter le standard SPARQL 1.1 à travers des requêtes HTTP. Il est ainsi compatible avec tous les magasins RDF supportant le standard SPARQL 1.1 et supportant les requêtes SPARQL à travers le protocole HTTP. Tout comme IGUANA, ce benchmark est un framework d'automatisation des opérations sur des magasins RDF. Les données sémantiques et les requêtes doivent être générées au préalable pour pouvoir l'utiliser. Plusieurs types d'opérations peut être testées, incluant les requêtes et les mises à jour. La paramétrisation des requêtes permet de générer des requêtes différentes à partir d'un modèle de requêtes et de paramètres. L'objectif est de pouvoir évaluer les performances d'une catégorie de requête sans que les mêmes données ne soient gardées en mémoire et renvoyées à chaque interrogation.

Pour évaluer les magasins RDF, la première étape consiste au choix des opérations à rouler sur le SGBD. Cet ensemble d'opérations est mélangé et roulé de manière aléatoire pour limiter la rétention en mémoire des résultats. Pour simuler un environnement de production,

2. https ://sourceforge.net/projects/sparql-query-bm/

il est possible de lancer des opérations d'échauffement. Par défaut, l'ensemble de requêtes sur la base de données est lancé 25 fois pour avoir des résultats plus précis. Si l'objectif est d'évaluer le comportement du SUT dans des conditions extrêmes de trafic, le *stress testing* permet de répéter un ensemble de requêtes pendant une durée fixée, en choisissant le nombre de connexions en parallèle. En augmentant au fur et à mesure le nombre de connexions en parallèle, on teste l'utilisation simultanée du SUT par un nombre variable d'agents.

À la fin du benchmark, plusieurs statistiques sont relevées. Nous pouvons citer par exemple : le temps total du benchmark, les QpS, les QMpH, les erreurs et les résultats par opérations. Il est donc possible de vérifier la précision des résultats en comparant les résultats reportés et une liste de contrôle prédéfinie.

Pour sa simplicité et sa couverture des différents types de tests voulus, nous avons choisi ce framework comme principal outil pour l'automatisation de notre évaluation de performance.

2.3 Conclusion

Dans ce chapitre, nous avons présenté les principaux concepts liés à notre recherche. Notre recherche se faisant dans le contexte du SOC d'une institution financière, l'évolution des techniques de détection d'intrusion et la ségrégation du rôle des analystes dans le SOC nous permettent de mieux situer notre étude. Nous avons pu rapidement introduire les besoins des analystes SOC en fonction de leur rôle. De ces besoins découle notre motivation d'utiliser une ontologie pour augmenter les capacités analytiques et réduire la complexité d'une analyse même sur d'importants volumes de données. La littérature sur les ontologies en sécurité nous confirme la pertinence d'utilisation des ontologies pour résoudre un problème de sécurité. Cependant, nous devions également considérer le volume des données et aussi l'ergonomie d'une solution basée sur les ontologies. Notre dernier point de littérature s'est donc concentré sur les méthodes d'évaluation de performances des bases de données ontologiques. Néanmoins, la littérature sur les méthodes d'évaluation de performances de magasins RDF manquait de diversité sur les outils pouvant supporter des cas d'utilisation à large échelle. De plus, les caractéristiques des magasins RDF autres que leurs performances n'étaient pas toujours prises en compte. Dans notre recherche, d'une part, nous avons répertorié et analysé les outils en terme de pertinence dans le contexte d'un SOC, d'autre part, nous avons réalisé des expériences sur des magasins RDF modernes et performants. Ces expériences ont ainsi l'avantage d'être récentes et de faire figurer de nouveaux outils développés spécialement pour répondre aux précédentes limites des magasins RDF.

CHAPITRE 3 FRAMEWORK D'AIDE À LA DÉCISION DANS UN CONTEXTE DE SOC

Ce chapitre étudie les outils sémantiques supportant des graphes RDF volumineux. Nous analysons leur architecture et leurs caractéristiques, puis nous les comparons pour donner des critères d'aide à la décision dans le choix de magasins RDF adéquats pour un scénario donné. Lors du développement de modèles ontologiques pour la détection d'intrusion dans un SOC, nous allons être confronté à la quantité de données qui transitent dans le réseau d'entreprise. Les limites de performance du langage de requête SPARQL sur des graphes RDF quand le nombre de triplets augmentent nous motivent à rechercher les outils développés répondant à cette problématique.

Pour notre analyse, nous avons construit une méthodologie pour la description et la comparaison des technologies sémantiques. Cette méthodologie a été créée à partir de la littérature consultée, pour mieux englober les caractéristiques les plus pertinentes des outils sémantiques. En plus de la recherche, nous avons exploré les comparateurs de bases de données de graphes ainsi que les papiers blancs de vendeurs de technologies sémantiques. La première étape a consisté à trouver les éléments importants dans le choix d'une base de données RDF. Nous pouvons les regrouper en trois grandes catégories : l'identité de la base de données de graphes, l'architecture et le positionnement dans l'écosystème des bases de données de graphes.

L'identité de la base de données de graphes regroupe les éléments superficiels qui décrivent la base de données. Tout d'abord, une base de données de graphes n'est pas toujours sémantique. Elle peut être un graphe de propriétés, en anglais *property graph*. Il faut donc distinguer les graphes de propriétés qui n'ont pas les capacités sémantiques que nous recherchons dans un graphe RDF. Certaines bases de données sont multi-modèles. Elles supportent les graphes comme une manière de visualiser la base de données, mais stockent les données de plusieurs manières différentes. Cette distinction est importante car elle a un impact sur les fonctionnalités et les performances de la base de données choisie. Ensuite, nous nous intéressons aux modèles de licences qui détermineront à la fois le degré de contrôle que nous pouvons avoir sur l'application, mais aussi le coût d'utilisation de la base de données de graphes. Enfin, le langage de programmation et les langages compatibles pour les API sont importantes dans l'intégration de la base de données dans un environnement d'entreprise et les capacités d'interaction. Selon leur licence de distribution, les outils sémantiques que nous avons analysés peuvent être répartis en deux catégories : les outils *open source* dont le code est disponible et

les caractéristiques plus détaillées, et les outils commerciaux dont les caractéristiques ont une description de haut niveau, souvent sans entrer dans les détails techniques d'implémentation. Dans le deuxième cas, il est difficile de donner une présentation complète de la technologie. Dans cette partie, nous pouvons déjà identifier un certain nombre de caractéristiques qui rendront le magasin RDF plus flexible d'utilisation, autant pour le développement que pour l'utilisateur final. En effet, nous faisons notre recherche dans le contexte d'un SOC. Les analystes SOC ont pour habitude de créer leurs propres outils et d'optimiser les outils existants pour mieux correspondre aux besoins de l'entreprise. La compatibilité avec plusieurs langages donne plus de latitude sur la manière d'interagir avec les modèles hébergés dans le magasin RDF.

L'architecture du magasin de triplets regroupe l'ensemble des éléments techniques à considérer dans le choix de la base de données, selon le contexte d'utilisation voulu. C'est ici que nous étudions en profondeur les caractéristiques du magasin RDF. Pour le développement d'ontologies, il est primordial que le magasin RDF soit conforme aux standards du Web sémantique. Les principaux standards qui nous intéressent sont la conformité avec le standard SPARQL version 1.1 pour bénéficier des dernières fonctionnalités implémentées. L'inférence est essentielle pour générer ou prouver de nouveaux énoncés. La stratégie d'inférence choisie par le vendeur déterminera le cas d'utilisation et les performances le cas échéant. Les deux stratégies d'inférence retrouvées sont le *forward chaining* et le *backward chaining*. Dans le *forward chaining*, de nouveaux triplets sont déduits à partir des triplets existants. Cette stratégie induit un coût calculatoire initial car l'inférence se fait automatiquement et de manière récursive jusqu'à ce qu'il n'y ait plus de nouveaux triplets créés. En contrepartie, les requêtes suivantes sont plus rapides après la synchronisation de la base de données avec les nouveaux énoncés créés. Une autre stratégie d'inférence possible est le *backward chaining*. Dans cette approche, les règles d'inférence sont appliquées durant les requêtes. Ceci permet d'économiser l'espace de stockage car juste les triplets ciblés sont sujets à l'inférence. Les ressources nécessaires pour raisonner sur toute la base de données ontologique sont également réduites. Pour une très large base de données, ces ressources peuvent être conséquentes. La contrepartie réside en la répétition du processus d'inférence pour chaque requête. Sur le long terme, les ressources nécessaires peuvent être supérieures à une unique inférence sur tout le graphe RDF. L'inférence est une fonctionnalité indispensable qui permet aux analystes de prouver des faits et d'automatiser certaines tâches. En plus de l'inférence, les capacités de visualisation de graphes d'un magasin RDF sont primordiales pour l'utilisateur final, à la fois connaisseur du Web sémantique, mais surtout celui sans connaissances en ontologies. La visualisation dans un plan de travail ergonomique facilite les tâches d'analyse.

Le magasin RDF est intégré dans un environnement d'entreprises comprenant plusieurs autres technologies. Des capacités d'intégration et d'extensibilité d'un graphe RDF permettent de le connecter avec des API et des outils utiles pour augmenter les fonctionnalités de la base de données de graphe. En effet, la possibilité de connecter un moteur de traitement de texte ou un moteur de recherche à la base de données ontologique aide à l'automatisation de l'importation et du traitement de données dans les magasins RDF. L'extensibilité et la possibilité d'intégrer d'autres outils, particulièrement des outils analytiques plus avancés, donnent une fois de plus de la flexibilité à l'utilisateur final. En plus des caractéristiques sémantiques, les bases de données peuvent être spécialisées vers le traitement de données en ligne. Les Online Transactional Processing (OLTP) sont des systèmes orientés vers le traitement rapide de transactions. Les transactions représentent les changements dans la base de données. Ces opérations de mises à jour, insertion et suppression dans la base de données sont optimisées. Les Online Analytical Processing (OLAP) sont des systèmes de traitement analytique de données. Ce sont des systèmes orientés pour l'analyse avancée de données et le traitement de requêtes complexes depuis de larges bases de données.

Une des raisons d'utiliser une base de données RDF dans une entreprise est l'augmentation des performances d'un service ou d'une tâche. La maturité et les performances du magasin RDF sont donc un des principaux critères d'analyse. Parmi ces performances, nous avons l'évolutivité, en anglais *scalability*, de la solution. Cette évolutivité comprend le nombre de triplets que le magasin RDF peut stocker, mais aussi sa résilience. Il est important de noter si le graphe peut être déployé en clusters pour une duplication du stockage ou encore être déployé dans le nuage. La duplication du stockage permet de paralléliser la lecture des données stockées et d'agir comme un équilibreur de charge, en anglais *load balancer*. Les performances et la disponibilité en sont accrues. Les rapports de *benchmark* constituent un critère supplémentaire de performance.

Nous avons trouvé les caractéristiques liées à l'architecture de la base de données de triplets RDF dans la documentation de l'outil concerné, les papiers blancs et les articles de recherche sur le graphe RDF. Pour les solutions propriétaires, les caractéristiques de haut niveau sont décrites sans entrer dans les détails d'implémentation.

La dernière catégorie de notre méthodologie est l'étude du positionnement de la base de données de graphe dans l'écosystème des SGBD de graphes. Nous nous intéressons aux classements et aux recherches menés par des organisations étudiant les bases de données de graphe. En plus de ces classements, les secteurs d'activités et les profils des clients des vendeurs donnent des informations sur le positionnement des vendeurs. En effet, un vendeur

desservant de nombreux clients dans le secteur des finances, avec des cas d'utilisation dans ce secteur, aura tendance à optimiser ses produits pour une telle utilisation.

3.1 Étude des technologies sémantiques à grande échelle

Nous avons choisi les bases de données sémantiques en fonction de leurs apparitions et de leurs performances dans les articles de recherche sur les magasins RDF à large échelle, mais aussi selon leur classement par les organisations *Bloor Research*[1] et *DB-Engines*[2]. DB-Engines classe les bases de données, notamment les magasins RDF, en fonction du modèle de données et de leur popularité. Bloor Research est une compagnie de recherche qui étudie les technologies disponibles dans l'industrie pour guider les organisations dans leur choix. Nous pouvons voir sur la figure 3.1 certains des magasins RDF évalués : *Amazon Neptune*, *Cambridge Semantics*, *Ontotext GraphDB* et *Stardog*. *Apache Jena*, *Blazegraph* et *Virtuoso* ne figurent pas sur ce classement, mais sont souvent mentionnés dans la recherche et le classement *DB-Engines* (voir figure 3.2).

3.1.1 Apache Jena

Apache Jena est un framework gratuit et *open source* codé en Java et spécialisé dans le développement et le déploiement d'applications du Web Sémantique. Il est distribué sous la licence *Apache License*, Version 2.0. Ce framework est composé d'API pour l'écriture et l'interrogation de bases de données RDF, ainsi que pour l'inférence. *Jena* est une des solutions les plus adoptées pour la recherche en raison de sa flexibilité et sa facilité d'utilisation [36]. DB-Engines la classe 2^{e} magasin RDF [37]. La figure 3.3 présente l'architecture de Jena et l'interaction entre les différents composants du framework.

Il existe deux principales manières d'interagir avec la base de données sémantique. D'une part, on peut configurer un serveur HTTP pour la gestion et les requêtes à travers un navigateur. *Apache Jena Fuseki* est le serveur HTTP qui sert d'interface avec le magasin RDF, permettant de faire des requêtes en SPARQL sur l'ontologie existante ou d'importer des fichiers RDF. *Fuseki* est compatible avec le protocole SPARQL 1.1 [38]. D'autre part, *Jena* utilise des API pour faire les requêtes directement sur la base de données. L'API RDF est l'API de base pour les interrogations. Elle englobe une API pour la prise en charge de l'ontologie et une API, *ARQ*, pour le traitement des requêtes SPARQL. L'API de *Jena* pour l'ontologie est compatible avec les langages OWL-full, OWL-Lite, OWL-DL et RDFS. L'indexation

1. https ://www.bloorresearch.com
2. https ://db-engines.com/en/ranking

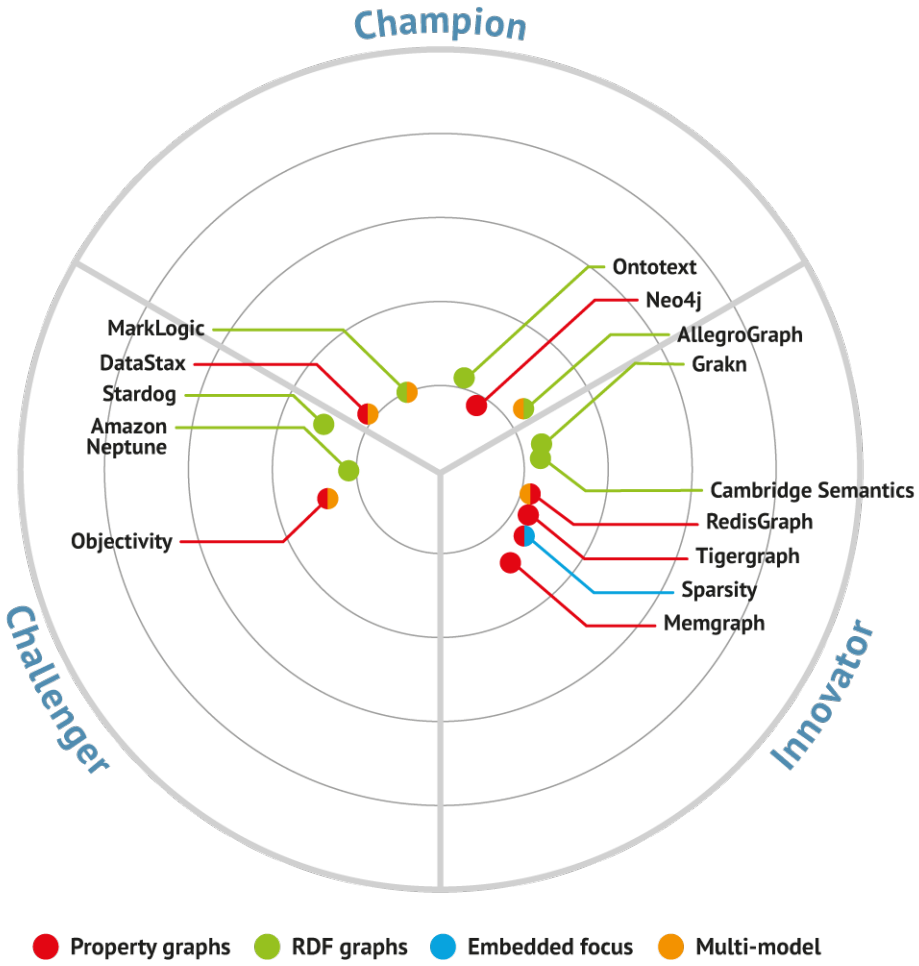


FIGURE 3.1 Paysages des bases de données de graphes par Bloor Research [33]. Nous nous intéressons uniquement aux graphes RDF. Les SGBD *multi-model* combinent au moins deux modèles de données (graphe RDF, graphe de propriété, relationnelle, etc.)

des triplets utilisée est une table de triplets. Le modèle de stockage de l'ontologie en cours d'utilisation est un stockage en mémoire. Lors d'une requête, les sous-graphes nécessaires pour la traiter sont chargés en mémoire le temps de la requête. L'API *ARQ* est utilisée dans plusieurs autres projets de recherche et est souvent associée à d'autres bases de données en tant que méthode de requête [39]. Cette API supporte le standard SPARQL version 1.1. Elle permet de faire des requêtes SPARQL à distance à travers le protocole HTTP, des requêtes fédérées vers un autre point d'extrémité (*endpoint*) SPARQL et des requêtes locales directement sur la base de données ontologique. *ARQ* permet aussi de faire des requêtes de mise à jour [30], permettant de créer des graphes RDF et charger ces nouvelles entités dans le graphe RDF existant. *ARQ* peut être étendue à *Lucene* ou *ElasticSearch*[3] pour combiner les

3. Moteurs de recherche par texte intégral

Rank Nov 2020	Rank Oct 2020	Rank Nov 2019	DBMS	Database Model	Score Nov 2020	Score Oct 2020	Score Nov 2019
1.	1.	1.	MarkLogic	Multi-model	11.10	-0.63	-1.72
2.	2.	↑ 3.	Apache Jena - TDB	RDF	2.94	+0.01	+0.37
3.	3.	↓ 2.	Virtuoso	Multi-model	2.54	-0.03	-0.10
4.	4.	4.	Amazon Neptune	Multi-model	2.43	-0.05	+0.83
5.	5.	5.	GraphDB	Multi-model	2.11	+0.01	+0.97
6.	6.	↑ 7.	Stardog	Multi-model	1.46	-0.01	+0.70
7.	7.	↓ 6.	AllegroGraph	Multi-model	1.04	+0.01	+0.17
8.	8.	8.	Blazegraph	Multi-model	0.80	-0.02	+0.22
9.	9.	↑ 10.	RDF4J	RDF	0.58	+0.01	+0.20
10.	↑ 11.	↓ 9.	Redland	RDF	0.47	+0.03	-0.06
11.	↓ 10.	11.	4store	RDF	0.45	+0.01	+0.10
12.	12.	↑ 14.	CubicWeb	RDF	0.21	0.00	+0.07
13.	↑ 14.	↑ 16.	AnzoGraph DB	Multi-model	0.18	+0.02	+0.07
14.	↑ 15.	↑ 15.	Mulgara	RDF	0.16	+0.01	+0.03
15.	↓ 13.	↑ 17.	Strabon	RDF	0.16	0.00	+0.07
16.	↑ 17.	↑ 19.	Dydra	RDF	0.13	+0.01	+0.06
17.	↓ 16.	↑ 18.	BrightstarDB	RDF	0.12	0.00	+0.04
18.	18.	↓ 13.	RedStore	RDF	0.06	-0.03	-0.13
19.	19.	↑ 20.	SparkleDB	RDF	0.01	+0.01	-0.04

FIGURE 3.2 Classement des magasins RDF par DB-Engines [34]

requêtes SPARQL à une recherche en texte intégral (*full text search*). Cette extension permet par exemple d'utiliser des expressions régulières pour filtrer les résultats d'une requête, et donc augmenter l'expressivité de la requête et la précision des résultats. Après la couche RDF, il y a une couche optionnelle pour l'inférence et le raisonnement. Cette API est flexible, permettant de créer des raisonneurs, en plus de ceux inclus. Dans le cadre d'un projet de recherche, cette flexibilité est un point fort de *Jena*. Finalement, il y a la couche de stockage des données. *Jena* est compatible avec le stockage de données sous forme de graphes RDF (*Jena TDB*) et sous forme relationnelle (*Jena SDB*). Des traducteurs (*parsers*) font le pont d'un formalisme à l'autre, permettant par exemple de traiter des requêtes SPARQL sur une base de données relationnelle, en transformant la requête SPARQL en requête SQL (*SPARQL2SQL*). *Jena TDB* ne possède pas de méthode de partitionnement des données ou de duplication des données [37]. Les requêtes vers la base de données peuvent être parallélisées. En cas de parallélisation et d'accès concurrents, des mécanismes anti-corruption de données peuvent être employés.

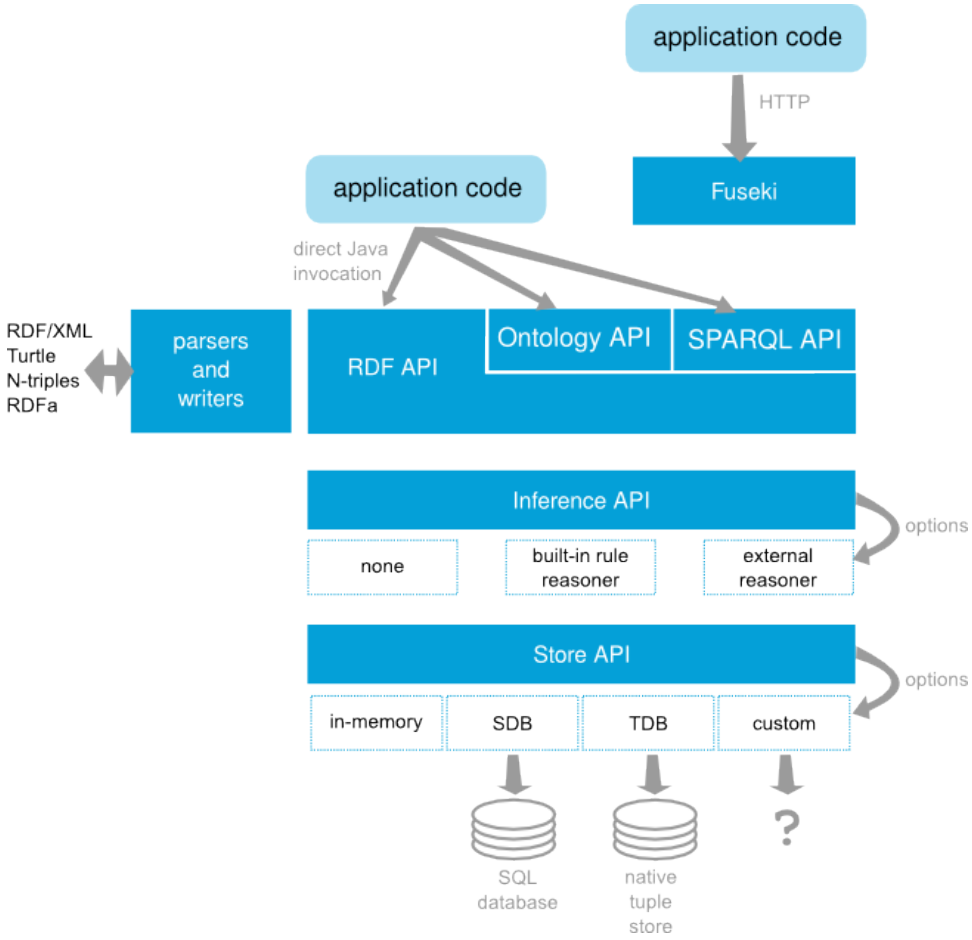


FIGURE 3.3 Architecture de Jena [35]

Pour sa flexibilité, sa modularité et sa licence *open source*, *Jena* est un des magasins RDF les plus utilisés dans la recherche académique. D'après un *benchmark*, *Jena TDB* permettrait de charger sur une même machine 1,7 milliards de triplets [40]. La documentation de *Jena TDB* ne mentionne pas une limite dans la taille de la base de données pouvant être chargée, mais considérant que *Jena* adresse en mémoire les entrées-sorties, en anglais *Memory-Mapped I/O* (MMIO) [41] et que *Jena TDB* n'est pas modélisée pour un déploiement distribué du stockage, la taille sera donc limitée par les ressources de la machine hôte.

3.1.2 GraphDB de Ontotext

GraphDB, anciennement *OWLIM*, est un SGBD sémantique développé par la compagnie Ontotext. Il existe trois licences de *GraphDB* : *GraphDB Free* qui est la version libre du logiciel, avec des capacités limitées ; *GraphDB Standard Edition* (SE) et *GraphDB Enterprise Edition* (EE) qui sont des licences commerciales. *GraphDB* est classée 5^{e} magasin RDF et

8e SGBD graphique sur DB-Engines. Ce SGBD est une des solutions les plus utilisées en entreprise et une des solutions ayant les meilleures performances pour de l'inférence à grande échelle [40,42]. Il est entièrement développé en Java mais supporte également les langages de programmation .Net, C#, Node.js, PHP, Python, Ruby et Scala.

GraphDB est développée à partir du framework RDF *Eclipse RDF4J* [43]. Les API pour le stockage et l'interrogation de la base de données sont fournies par RDF4J. La figure 3.4 montre l'architecture haut niveau de *GraphDB*. Ses trois principaux composants sont le moteur (*engine*), les connecteurs (*connectors*) et le plan de travail (*workbench*).

L'*engine* est le cœur de la base de données RDF. Un optimiseur des requêtes SPARQL permet d'améliorer la performance des requêtes en donnant la possibilité de créer des plans de requêtes. Pour une requête donnée, il est possible d'avoir des informations sur la manière dont les requêtes sont exécutées par *GraphDB*. Ceci permet d'améliorer les performances de ces requêtes en augmentant par exemple la proximité des données (avec un cache) formant les sous-graphes impliqués dans la requête. Le moteur d'inférence et de raisonnement employé est *TRREE* (*Triple Reasoning and Rule Entailment Engine*). Des règles d'inférence prédéfinies sont fournies : RDFS, RDFS plus, OWL-HORST, OWL-max, OWL2-QL, OWL2-RL. Il est toutefois possible d'importer des règles personnalisées. *GraphDB* adopte la stratégie d'inférence de *forward chaining*. Le dernier composant de l'*engine* est la pile de stockage des données. Les données sont stockées sous forme de fichiers. L'indexation se fait également en mémoire (MMIO) et est optimisée pour permettre à *GraphDB* de traiter plus de 17 milliards de triplets [40], sous réserve de la mémoire disponible. Le stockage et l'indexation sont des combinaisons de stratégies existantes. Pour l'indexation, GraphDB utilisent deux principaux index : prédicats, sujets, objets et prédicats, objets, sujets. Toutes les données sont stockées dans le même répertoire.

Les connecteurs permettent d'étendre les caractéristiques de *GraphDB* avec des composants ou logiciels extérieurs. Le connecteur Lucene permet de faire de la recherche par texte intégral sur le magasin RDF. L'intégration de MongoDB [4] permet de générer des requêtes SPARQL sur des bases de données de documents MongoDB. Les requêtes sont traduites en SQL avant d'interroger la base de données relationnelle.

Le *workbench* est une plate-forme d'administration web, permettant d'accéder depuis le navigateur aux différentes fonctionnalités de *GraphDB*. Il permet ainsi de gérer, importer et visualiser les données RDF hébergées dans des répertoires, interroger le magasin RDF et surveiller l'utilisation de ressources. C'est un des points forts de *GraphDB* qui permet aux développeurs d'en étendre les capacités.

4. spécialisé dans le stockage de documents

Les versions *SE* et *Free* possèdent pratiquement les mêmes caractéristiques. La principale différence est une capacité à gérer des requêtes concurrentes illimitées pour la version *SE*, contre un maximum de deux requêtes en parallèle pour la version *Free*. La version *EE* est une version améliorée et adaptée pour une utilisation à très grande échelle, notamment en entreprises. Elle permet le déploiement des répertoires sous forme de *clusters* pour la duplication de données, mais surtout des requêtes concurrentes sur les différents *clusters*. En distribuant la charge dans différents clusters, *GraphDB EE* permet de lever la limitation de la taille du magasin RDF. Les données peuvent être réparties sur des machines différentes et la mémoire disponible lors de la lecture ou de l'écriture devient le cumul de la mémoire des différents clusters. Il faut tout de même noter que le coût des transactions réseaux nécessaires à l'utilisation d'un déploiement distribué peut dépasser le gain lié à la distribution des serveurs [44].

Les trois versions donnent l'option d'un support technique de la part de Ontotext pour une aide lors de la résolution de problèmes liés au déploiement ou pour déterminer une utilisation de *GraphDB* dans un contexte spécifique. *GraphDB* est un des magasins RDF les plus utilisés en entreprises avec comme cas d'utilisation notables : l'inférence à grande échelle, le stockage de documents, la gestion de métadonnées [42]. Dans son classement des bases de données de graphes de 2020, Bloor classe Ontotext dans la catégorie des "champions" [33].

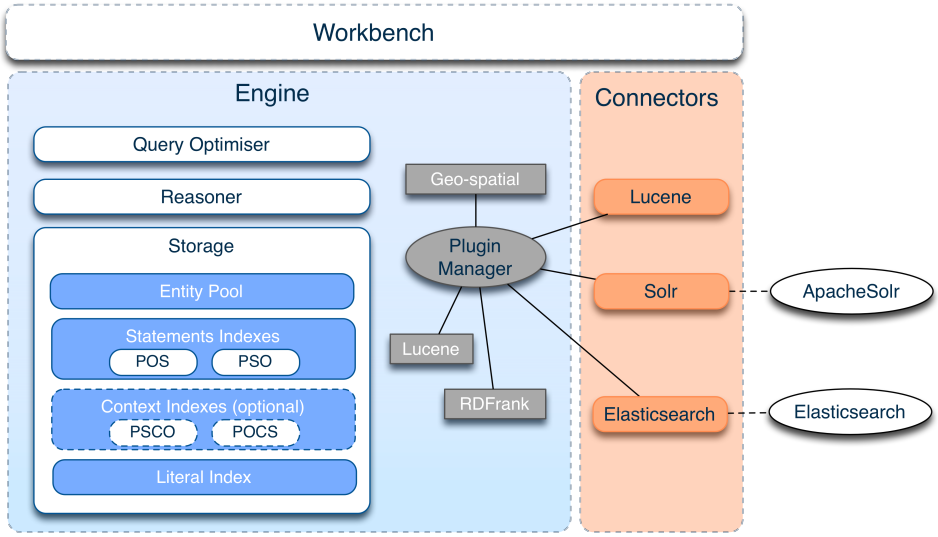


FIGURE 3.4 Architecture de GraphDB [43]

3.1.3 Virtuoso

Virtuoso est une solution multi-modèle permettant de représenter des bases de données de graphes et des bases de données relationnelles. Ce SGBD est développé par la compagnie Openlink qui le distribue sous une licence *open source* et une licence commerciale. Catégorisé comme un SGBD multi-modèle, Virtuoso est classé 3^{e} magasin RDF et 5^{e} SGBD graphique par DB-Engines [45]. Virtuoso est une solution adaptée pour être flexible et avoir des performances à large échelle. Il est connu pour servir de magasin RDF et d'*endpoint* SPARQL à de larges bases de données RDF telles que DBpedia et BioRDF. Virtuoso est développé en C et supporte plusieurs langages de programmation .Net, C, C#, C++, Java, JavaScript, Perl, PHP, Python et Ruby.

La stratégie de stockage de Virtuoso est sous forme d'une table de quatre colonnes : les graphes, les propriétés, les sujets et les objets. Des méthodes de compression permettent de réduire considérablement la taille des données. Même si la représentation visuelle des données est sous forme de graphe, Virtuoso a adopté un stockage de la table RDF dans une base de données SQL. Les données sont stockées dans une table de triplets avec une quatrième colonne correspondant aux graphes. L'indexation se fait sur la table de triplets. Erling *et al.* justifient ce choix de modélisation à la difficulté et au coût des jointures lorsque que de larges bases de données sont hébergées dans une unique table [46]. Dans ce cas de figure, la traduction des requêtes SPARQL en SQL se fait au moment de l'optimisation de la requête SPARQL. La requête SQL est ensuite compilée pour interroger la base de données. Virtuoso est compatible avec le standard SPARQL version 1.1. Virtuoso supporte des extensions comme Jena ARQ pour interroger l'ontologie. Pour un déploiement à large échelle, Virtuoso dispose d'une version distribuée. Il est possible de déployer des clusters hébergeant les bases de données partitionnées à travers la ferme de serveurs. En terme de performance, Virtuoso a permis de charger près de 59 milliards de triplets [40].

Comme cas d'utilisation, Virtuoso est adopté dans l'analytique et la virtualisation de données dans plusieurs secteurs d'activités comme les finances et l'industrie biomédicale [45]. Un *benchmark* a permis de charger et de faire des requêtes sur 70 milliards de triplets [40].

3.1.4 Stardog

Stardog est une base de données RDF développée entièrement en Java. Elle supporte les langages de programmation : .Net, Clojure, Groovy, Java, JavaScript, Python et Ruby. Stardog est uniquement distribué sous une licence commerciale visant les entreprises. DB-Engines classe Stardog 6^{e} magasin RDF [47].

La stratégie d'indexation utilisée est l'indexation par permutation de triplets. Stardog dispose de connecteurs permettant d'interroger des bases de données de graphes et relationnelles. Elle dispose également de connecteurs pour améliorer les capacités analytiques telles que des serveurs BI/SQL (*Business Intelligence*) pour connecter la base de données à des plateformes analytiques. Stardog Studio est l'interface web d'administration. En plus de cette interface, Stardog possède des API pour exécuter des requêtes directement sur les magasins RDF. Il supporte le standard SPARQL 1.1 et le régime d'inférence OWL2. La stratégie d'inférence adoptée est l'approche *backward chaining*. En plus de SPARQL, les requêtes peuvent être faites avec le langage de requête GraphQL. Pour une plus grande disponibilité, Stardog peut être déployée de manière distribuée. Elle met également à disposition des API pour réaliser de l'apprentissage machine sur les graphes de connaissance.

Le principal cas d'utilisation dans ce contexte est la prédiction analytique. Un *benchmark* a permis d'importer jusqu'à 50 milliards de triplets sur une machine [40]. Stardog délivre un support technique 24/7 pour compléter son offre. En résumé, Stardog est une solution complète pour une entreprise, avec des fonctionnalités traditionnelles aux graphes de connaissance, mais aussi des fonctionnalités innovantes, pour s'adapter aux nouveaux besoins analytiques des entreprises. Stardog est positionnée en tant que "challenger" par Bloor dans son classement des bases de données de graphes de 2020 [33].

3.1.5 Blazegraph

Blazegraph (ancien Bigdata) est une base de données de graphe *open source*, développée en Java. Elle a été conçue avec pour cible des graphes larges et complexes et une compatibilité avec l'apprentissage machine. Elle est classée 8^{e} magasin RDF sur *DB-Engines* [48]. En plus du Java, Blazegraph supporte les langages de programmation .Net, C, C++, JavaScript, PHP, Python et Ruby. Blazegraph est déployée sous une licence gratuite (Community Edition). La licence commerciale pour les entreprises n'est plus disponible. Durant un *benchmark* fait sur une seule machine, jusqu'à 12,7 milliards de triplets ont pu être chargés dans la base de données.

La base de données de Blazegraph peut contenir jusqu'à 50 milliards de triplets sur une même machine. Blazegraph est compatible avec la duplication pour offrir une plus grande disponibilité. Blazegraph supporte le standard SPARQL version 1.1 et la parallélisation des requêtes. L'indexation se fait par la création d'arbres d'index. Au delà du nombre de triplets que le magasin RDF peut charger, les principaux atouts de Blazegraph étaient l'utilisation de cartes graphiques pour l'accélération graphique (*GPU acceleration*), c'est-à-dire l'utilisation de cartes graphiques pour l'optimisation des opérations. Elle devait augmenter les perfor-

mances pour un coût réduit, ainsi que d'autres fonctionnalités disponibles dans la version entreprise.

3.1.6 Amazon Neptune

Amazon Neptune est une base de données de graphe développée par Amazon. Elle est entièrement hébergée dans le nuage AWS et est déployée sous une licence commerciale à la demande. Neptune est classée 4e magasin RDF et 6e SGBD graphique par le comparateur DB-Engines [49].

Neptune est entièrement hébergée dans le nuage de AWS. Les données sont stockées dans des clusters dédiés, avec la possibilité de répliquer les données dans plusieurs zones géographiques pour une meilleure disponibilité. La taille maximum de ces clusters de stockage est de 64 teraoctets, qui est également la taille limite d'un graphe dans le cluster. Neptune est compatible avec les langages de requête SPARQL et Gremlin.

Comme plusieurs autres outils sémantiques, il est possible de connecter le *workbench* de RDF4J à une base de données Neptune. Cependant, la manière recommandée d'interagir avec la base de données est à travers des *notebooks*. L'avantage d'une telle approche est la possibilité d'utiliser d'autres langages de programmation dans un seul environnement et de créer des *pipelines* d'exécution faisant intervenir plusieurs technologies. Les possibilités de visualisation sont également augmentées par l'utilisation de *notebooks*. Neptune ne possède pas de modules d'inférence RDF. L'extension de Neptune avec des connecteurs peut combler cette absence, mais aurait également un impact à déterminer sur les performances. Neptune utilise trois index de permutations de triplets.

Ainsi, Neptune se positionne comme un service infonuagique géré par Amazon AWS et spécialisé dans sa capacité à gérer de grands volumes de données et une grande disponibilité de service. Bloor le classe dans la catégorie "challenger" de son classement des SGBD de graphe de 2020 [33].

3.1.7 Anzograph

Anzograph est une base de données de graphe développée par la compagnie Cambridge Semantics. Ce SGBD de graphe est disponible sous une licence gratuite, limitée dans l'utilisation de la mémoire, et une licence commerciale. Elle est classée 15e magasin RDF sur *DB-Engines* [50]. Anzograph supporte les langages de programmation C++, Java et Python.

Anzograph est un SGBD natif de graphe pour le traitement massif en parallèle, en anglais *massively parallel processing* (MPP), plus spécifiquement une base de données OLAP. C'est

à dire un logiciel dédié spécifiquement au traitement et à l'analyse en parallèle d'importants volumes de données [51]. Elle s'intègre avec les *notebooks* Jupyter et Zeppelin qui favorisent la création de pipelines pour l'exploration, l'analyse et la visualisation des graphes RDF. Spécialisée dans l'analytique de graphes, Anzograph comprend des algorithmes graphiques pour l'exploration de graphes et le calcul de métriques sur les ontologies chargées dans la base de données. Elle supporte également l'intégration avec d'autres outils analytiques comme R et Tableau.

Anzograph est compatible avec les standards W3C RDF et SPARQL 1.1. Pour atteindre de meilleures performances, les données sont stockées en mémoire et sur le disque dans un format de graphe natif et le modèle d'inférence adopté est le *forward chaining*. Aucun index n'est créé et la recherche de triplets se fait en parcourant le graphe stocké. Le modèle d'inférence lui permet d'atteindre de meilleures résultats, en consommant toutefois plus de mémoire que le modèle *backward chaining*. De plus, les requêtes SPARQL sont compilées en C++. Ainsi, la première interrogation de la base de données avec une requête donnée est plus longue. Cependant, le temps de traitement de la même requête les fois suivantes est nettement réduit.

Anzograph peut être déployée sur des plates-formes infonuagiques ou sur du matériel. Pour assurer la disponibilité et la sauvegarde des données, elle est compatible avec la réplication des données dans des clusters.

Anzograph permet de traiter de grands volumes de données, comme on peut le voir dans un *benchmark* au cours duquel jusqu'à un trillion de triplets ont été chargés en moins de deux heures [40]. C'est ainsi un SGBD de graphe avec de nombreuses fonctionnalités analytiques, de bonnes performances quand les données augmentent et une bonne intégration en entreprise. Elle est utilisée dans les secteurs financier et pharmaceutique. Dans son rapport de 2020 sur les bases de données de graphes, Bloor la positionne comme un "innovateur" [33].

3.2 Comparaison qualitative des solutions

Dans cette section, nous allons comparer les outils présentés, selon un ensemble de caractéristiques. Pour cette comparaison, nous avons retenu les caractéristiques les plus importantes pour mener un projet à la fois académique, mais également viable dans un environnement réel.

La première caractéristique est le modèle de licences. Il y a trois principaux modèles à distinguer : les licences *open source*, les licences gratuites et les licences commerciales. Les licences *open source* permettent l'accès au code source du produit et permettent de mieux comprendre le fonctionnement de l'outil. Il est aussi possible d'étendre les fonctionnalités. Les

TABLEAU 3.1 Comparaison de magasins RDF suivant un ensemble de caractéristiques

SGBD	**Licence**	**Requête**	**Langage**	**Inférence**	**Déploiement**	**Performances**	**Classement**	**Sécurité**	**Points forts**
Anzograph	Propriétaire : Entreprise, Gratuite	SPARQL 1.1	C++	RDFS OWL	Réplication Cluster Nuage serveur autonome	1700 milliards (200 nœuds, 208 GB de RAM) [40]	DBE : 15 [34] Bloor : Innovateur [33]	Oui	Analytique Performance
Blazegraph	*Open source*	SPARQL 1.1	Java	RDFS OWL	Réplication Cluster Nuage serveur autonome	12,7 milliards [40]	DBE : 8 [34] Bloor : Innovateur [52]	Oui	Recherche
GraphDB	Propriétaire : Entreprise, Standard, Gratuite	SPARQL 1.1	Java	RDFS OWL	Réplication Cluster Nuage serveur autonome	13 milliards (512 GB de RAM) [40]	DBE : 5 [34] Bloor : Innovateur [33]	Oui	Inférence à large échelle Sémantique Polyvalent
Jena TDB	*Open source*	SPARQL 1.1	Java	RDFS OWL	Serveur autonome	1,7 milliards [40]	DBE : 2 [34]	Oui	Recherche
Neptune	Propriétaire : À la demande	SPARQL 1.1	Non divulgué	Aucune inférence	Uniquement nuage Réplication Cluster	Aucun	DBE : 3 [34] Bloor : Challenger [33]	Oui	Évolutivité, disponibilité
Stardog	Propriétaire : Entreprise, Gratuite	SPARQL 1.1 SQL	Java	RDFS OWL	Réplication Cluster Nuage serveur autonome	50 milliards (256 GB de RAM) [40]	DBE : 6 [34] Bloor : Challenger [33]	Oui	données hétérogènes, virtualisation graphique
Virtuoso	Entreprise, *Open source*	SPARQL 1.1 SQL	C	RDFS OWL	Réplication Cluster Nuage serveur autonome	150 milliards (8 nœuds, 256 GB de RAM)	DBE : 4 [34] Bloor : Challenger [52]	Oui	Unification relationnelle-graphique

licences libres ne donnent pas accès au code source mais permettent une utilisation du produit sans frais. Le fonctionnement interne et l'implémentation du standard RDF ne sont pas toujours bien documentés. Les possibilités d'extension sont également restreintes. Enfin, les licences commerciales donnent accès au produit ou à un ensemble additionnel de fonctionnalités suivant une tarification précise. Dans un projet de recherche, il est important de pouvoir connaître et comprendre les rouages internes des outils, ou au moins ceux qui sont directement liés au sujet de recherche. Les solutions possédant des licences *open source* sont donc à considérer malgré la contrepartie sur les performances ou les limitations en fonctionnalités. Dans cette catégorie, Blazegraph, Jena et Virtuoso sont les seules solutions ayant une licence *open source* de notre analyse. GraphDB a été développée à partir du framework *open source* RDF4J. Une partie des fonctionnalités sont donc à code ouvert et peuvent être modifiées. Le reste des solutions étant commerciales, l'utilisation et la modification des technologies ne peuvent se faire qu'à travers les moyens publiés dans le cas échéant par le vendeur.

Suivant le besoin, les fonctionnalités analytiques du SGBD de graphe constituent un des principaux critères de choix. En effet, l'optimisation et la disponibilité d'opérations analytiques peuvent permettre d'atteindre de meilleures performances que d'avoir recours à un logiciel tiers, mais aussi augmenter l'expérience utilisateur. Certains types de bases de données sont spécialisées dans l'analytique, comme les OLAP. Pour des cas d'utilisation impliquant une analyse intensive de gros volumes de données, ces bases de données ont un avantage par leur modèle. Par exemple, la possibilité d'appliquer directement des algorithmes de parcours aux graphes chargés dans le SGBD permet d'augmenter la précision des résultats voulus tout en s'assurant de le faire dans un temps raisonnable. Anzograph est la seule OLAP de notre liste. Elle est la plus spécialisée pour les scénarios analytiques tels que l'analyse de données à la fois structurées et non structurées, l'application du traitement de langage naturel ou l'exploration et le calcul de métriques d'un graphe RDF. En plus des fonctions de base, la facilité d'intégration avec des technologies utilisées dans un environnement d'entreprise est essentielle dans la comparaison des magasins RDF. Les données traitées peuvent nécessiter du traitement de texte ou l'ingestion depuis différentes sources en temps réel. La visualisation des données dans des outils plus spécialisés depuis l'interface du SGBD permet une fois de plus d'avoir une meilleure expérience utilisateur et de combler les limites du magasin RDF. Ces intégrations sont généralement présentes dans les versions commerciales des logiciels.

La stratégie de déploiement de la solution est également importante car l'infrastructure hébergeant le projet devra être compatible. Nous avons le déploiement sur une seule machine et le déploiement dans des clusters. Certains outils ne peuvent être retrouvés que dans le nuage spécifique à un fournisseur tel que Amazon AWS ou Microsoft Azure. Il faut donc considérer cette caractéristique dans le choix de la solution. En termes de performance, ces deux dé-

ploiements ont chacun leurs avantages et contreparties. Le déploiement sur une seule machine requiert moins d'administration et se fait généralement plus rapidement. Les performances sont intrinsèquement liées à la puissance de la machine sur laquelle la base de données est déployée. Pour de très larges graphes, les besoins en mémoire, en processeur et en espace de stockage peuvent rapidement devenir énormes et surtout limitant. Le déploiement en clusters permet de répartir les ressources sur plusieurs serveurs. De plus, on a accès à d'autres services comme la duplication du stockage pour une meilleure disponibilité. Cependant, le coût en communication réseau est grand. Beaucoup de bases de données RDF distribuées reposent sur un framework tel que Hadoop ou Apache Spark. Ces frameworks sont spécialisés pour l'analytique de larges quantités d'informations et permettent d'adresser l'évolutivité des graphes de connaissances. Toutefois, ils induisent un coût initial lié au lancement et au fonctionnement du framework. Seule Neptune n'est uniquement déployable que dans le nuage AWS. L'architecture de Jena n'est pas compatible au déploiement dans des clusters, limitant son utilisation à une seule machine.

Ensuite, nous avons la flexibilité. La flexibilité est un mélange de plusieurs critères. C'est la facilité d'extension d'un outil et son intégration avec d'autres technologies existantes. Ce critère est l'un des plus importants pour l'utilisateur final. En effet, il est difficile de développer une solution qui soit universellement complète. En fonction du scénario d'utilisation, du contexte de l'entreprise ou des analystes, il est nécessaire de pouvoir adapter la solution développée. Au delà de la flexibilité du modèle, le magasin RDF est le principal facteur de flexibilité. Par exemple, les langages de programmations pour développer des logiciels interagissant avec le modèle hébergé dépendent du SGBD graphique choisi. Pour des analystes qui ont des compétences dans différents langages de programmation, il est important de leur permettre d'utiliser l'outil avec un langage dans lequel ils sont le plus à l'aise. Ensuite, l'intégration d'autres outils. Avec le développement du nuage et du DevOps comme combinaison développement logiciel et des opérations sur le système informatique, l'utilisation de *notebooks* et autres outils pour l'administration de plusieurs programmes en un seul endroit s'est répandue. La manière dont un magasin RDF hébergeant le modèle de détection d'intrusion s'intègre avec le reste de l'infrastructure informatique et des pratiques des analystes marquera la courbe d'apprentissage et le degré de difficulté à utiliser et à s'adapter à cette nouvelle manière de chercher les menaces dans les réseaux. Pour ce critère, Anzograph et Stardog sont des solutions spécialisées dans l'analytique en entreprise. De ce fait, de nombreux outils analytiques utilisés en entreprise sont facilement intégrables. Virtuoso est le magasin RDF compatible avec le plus d'API et de méthodes d'accès différentes. Il a également le plus de langages de programmation supportés. En terme de flexibilité, Neptune a le moins de méthodes d'accès différentes. De plus, étant uniquement déployable sur le nuage d'Amazon AWS, c'est

une limite à prendre en compte. GraphDB, notamment à travers la partie *open source* de son code, RDF4J, est très extensible. Ceci lui donne une flexibilité d'utilisation dans différents contextes différents.

Le dernier critère de comparaison que nous considérons est la performance globale du SGBD. Cette performance se traduit par le temps de chargement des données dans le magasin RDF, le temps de requêtes vers le magasin RDF et le temps de mise à jour de données déjà chargées. Les *benchmarks* disponibles menés par les vendeurs ou par des chercheurs ainsi que les rapports de Bloor sur les magasins RDF placent Anzograph en tête de notre liste. C'est une solution qui permet de charger de très importants volumes de données. La compilation des requêtes permet de réduire considérable le temps de requêtes et de mises à jour des données. Elle est suivie par Stardog et Neptune. Pour les performances d'inférence à grande échelle, GraphDB fait partie des meilleurs SGBD. Neptune nécessite l'intégration d'un *endpoint* externe pour l'inférence. Ses performances en matière d'inférence sont donc limitées.

Dans le tableau 3.1, nous avons rassemblé les différentes caractéristiques que nous jugeons importantes pour le choix d'un magasin RDF. Ces caractéristiques nous permettent de distinguer des outils plus adaptés pour la recherche, d'outils plus adaptés dans un environnement de production.

Pour la recherche, nous retenons Apache Jena et Blazegraph comme principaux magasins RDF. Ceci est dû à leur flexibilité, leur modularité et la disponibilité de leur code source pour d'éventuelles extensions. En particulier, Jena est déjà très présent dans la littérature ce qui lui a valu son classement sur DB-Engines.

GraphDB est un outil polyvalent, très performant pour les opérations sémantiques, en particulier l'inférence. Il est également très extensible ce qui en fait un outil autant adapté à un projet de recherche qu'à une application en production.

Anzograph est un magasin RDF hautement performant, particulièrement pour les opérations analytiques qu'on peut retrouver dans un environnement de production. Il est de ce fait la meilleure solution que nous avons pu analyser en matière de performance et d'analytique.

Neptune permet de bénéficier des avantages de AWS en matière d'évolutivité et de disponibilité, mais aussi en termes de facilité de déploiement.

3.3 Moteur de contrainte : vers une compilation des requêtes SPARQL

L'évaluation de requêtes complexes en SPARQL induit une baisse de performance même pour des bases de données RDF matures. Comme nous l'avons présenté dans la revue de littérature, en fonction du type de requêtes et des caractéristiques des requêtes, les magasins RDF ont des

performances différentes. Une alternative à l'interrogation des bases de données de graphes en SPARQL est la conversion de la requête en contraintes compilables dans un langage plus performant que SPARQL. Nous analysons dans cette section cette option proposée par Babar (2020) [53].

Babar présente dans son mémoire une méthode de transformation des requêtes SPARQL en contraintes pour un IDS. Cette méthode étend la détection d'anomalies basée sur la validation de contraintes réseau à la validation de contraintes au niveau applicatif. Cette nouvelle couche de contrôle permet d'augmenter la fiabilité des alertes et de renforcer la défense du système. Cet IDS a été appliqué à un système de contrôle de trafic aérien, en anglais *Air Traffic Control* (ATC).

Le système de détection d'intrusion augmenté par Babar est composé de deux principaux éléments : un IDS à partir de contraintes réseau et une ontologie de domaine ATC constituée de contraintes applicatives.

Les contraintes réseau sont liées aux protocoles de communication au niveau de la couche réseau, comme par exemple l'*IP de l'émetteur*. Les paquets transitant dans le réseau sont d'abord traduits dans un format adapté au moteur de contraintes de l'IDS. Puis, ils sont validés contre des règles prédéfinies dans l'IDS. Pour introduire à cette routine les contraintes applicatives, Babar s'est servie d'ontologie de domaine sur l'ATC développée par Morel (2017) [54]. Cette ontologie fournit la spécification technique des anomalies dans le système, ainsi que les requêtes et les règles pour évaluer les événements transitant dans l'ATC contre cette spécification. Elle a été construite en suivant la méthodologie ATOM, développée par Malenfant-Corriveau [11]. En partant de questions de compétence, les anomalies et le comportement *normal* sont définis en se servant de concepts de haut niveau proches du langage de l'expert. Ces informations sont directement extraites des paquets échangés par l'ATC. Nous avons par exemple la *vitesse d'un avion*. Pour extraire ces données des paquets réseau bruts, un traducteur RDF a été développé. Ensuite, en lieu et place des requêtes SPARQL pour la détection d'anomalies, Babar transforme les requêtes SPARQL en contraintes écrites dans un langage spécifique à l'IDS, en anglais *Domain Specific Language* (DSL). Ensuite, depuis les contraintes DSL, Babar génère des contraintes écrites en langage C. C'est ce code C qui valide l'intégrité des paquets au niveau applicatif. Le framework global de l'IDS permettra donc de valider l'intégrité des paquets en temps réel à la fois au niveau de la couche réseau et aussi de la couche applicative.

Dans ce modèle d'utilisation de l'ontologie pour la détection d'anomalies, l'ontologie est utilisée pour créer la logique abstraite de détection. Cependant, la détection est faite à partir d'un code compilé, produit à partir de cette logique. L'utilisation des requêtes sur la base

de données ontologique peut être utile pour le diagnostique et l'explication des alertes levées par l'IDS. Mais il n'est plus nécessaire de pouvoir faire des requêtes en temps réel ou proches du temps réel. Dans le contexte d'un SOC, cette approche peut également être bénéfique. En prenant par exemple la recherche de menaces dans un SOC, le framework ATT&CK de MITRE peut être utilisé pour concevoir un système expert avec la méthodologie ATOM. La traduction et la compilation des requêtes SPARQL en contraintes pour un IDS permettront d'automatiser le processus analytique de l'expert. Les IDS étant optimisés pour traiter la quantité de données transitant dans le réseau, nous pouvons bénéficier à la fois des avantages sémantiques d'une ontologie sans perdre en performance dans la détection d'intrusion. Pour la forensique post-incident, la possibilité d'investiguer les alertes à l'aide de requêtes de haut niveau augmentera les capacités analytiques des analystes.

3.4 Conclusion

Dans ce chapitre, nous avons pu déterminer des caractéristiques importantes dans le choix d'un magasin RDF. Ensuite, nous avons pu analyser et comparer les SGBD sémantiques les plus populaires selon ces caractéristiques. Au vu des différentes caractéristiques de ces magasins RDF et leurs classements, nous avons pu choisir quelques SGBD sur lesquels des expériences seront menées. Ces expériences nous permettront de comparer leurs performances dans un scénario et avec des ressources similaires. Nous pourrons ainsi compléter les analyses dans ce chapitre par les expériences que nous mèneront.

CHAPITRE 4 ÉVALUATION DE PERFORMANCES

Notre recherche se fait dans le contexte d'utilisation des technologies sémantiques (RDF et SPARQL) pour explorer les relations entre entités dans de larges volumes de données. Pour mettre en œuvre ce projet, nous avons besoin de magasins RDF adaptés. Dans le chapitre précédent, nous avons présenté quelques bases de données RDF. Dans ce chapitre, nous présentons la méthodologie adoptée durant notre recherche pour évaluer et comparer les performances de quelques magasins RDF choisis parmi ceux présentés. Dans le cadre de cette évaluation, nous avons recherché les jeux de données disponibles en ligne pour la réalisation de projets de sécurité et pour notre évaluation plus spécifiquement.

Pour mieux présenter notre évaluation de performances, quelques concepts additionnels ont besoin d'être décrits.

4.1 Méthodologie d'évaluation de performance

Nous mesurons l'aptitude des magasins RDF à répondre en temps raisonnable à une requête SPARQL, en fonction des besoins d'évolutivité. Déterminer ce qui est considéré comme raisonnable dépend fortement du cas d'utilisation et de la comparaison avec les méthodes existantes. Pour cette raison, nous concentrons nos expériences sur la comparaison des temps de réponses des magasins RDF entre eux.

Dans le chapitre 3, nous avons pu cerner des magasins RDF qui ont en principe des performances satisfaisantes, mais aussi des capacités analytiques adaptées à un environnement de production. Parmi ces bases de données sémantiques, nous avons retenu : Anzograph pour son évolutivité, Stardog pour les capacités analytiques qu'elle fournit et enfin Openlink Virtuoso et GraphDB pour leurs performances citées dans les articles de recherche. Ces systèmes dans notre évaluation sont les systèmes sous test, en anglais *Systems under Test* (SUT). Nous avons opté pour des tests paramétriques, simulant des situations qu'on pourrait trouver dans un environnement de production. Initialement, nous avions pour cible un autre magasin RDF : Apache Jena. Cependant, des expériences préliminaires ont montré une grande différence de performances avec les autres magasins RDF. Nous l'avons donc retiré de nos expériences. Amazon Neptune a également été retiré des expériences car le seul déploiement possible est sur le nuage d'Amazon AWS. Il était incompatible avec les environnements de test à notre disposition.

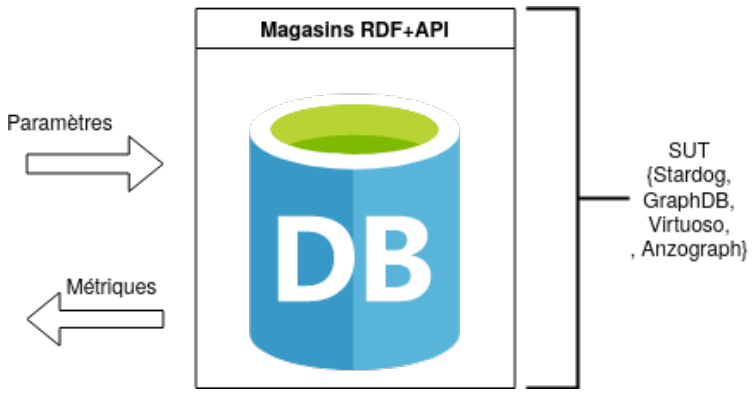


FIGURE 4.1 Systèmes sous test

Les deux catégories de variables de contrôle que nous comparons sont l'évolutivité (*scalability*) des magasins RDF et leur comportement face à une augmentation de la charge de travail. L'évolutivité peut être mesurée en faisant varier la taille du jeu de données. La complexité de l'ontologie permet également de tester l'évolutivité du SUT car le nombre d'opérations nécessaires pour répondre à une requête augmente avec la complexité de celle-ci et de l'ontologie ciblée. Quant à la charge de travail, nous la simulons en augmentant le nombre de requêtes par unité de temps, ainsi que le nombre de requêtes en parallèle. Durant notre évaluation, nous testons certaines fonctions des SUT.

Nous nous intéressons au temps de chargement des données dans la base de données sémantique. Dans un environnement réel, les données à charger peuvent atteindre des pétaoctets. Le temps de chargement est un élément essentiel dans la comparaison des solutions. L'indexation des entités peut se faire au moment de l'importation des données, de manière sélective. En fonction de la stratégie d'indexation, le temps de chargement peut donc varier. Nous avons choisi de ne pas distinguer les magasins RDF selon le moment où l'indexation se fait et de procéder à une comparaison globale du temps d'importation de données. La deuxième fonction est la réponse à des requêtes SPARQL.

Nous mesurons et comparons le temps de réponse aux requêtes SPARQL pour les différents magasins RDF. Pour qu'une application sémantique soit adoptée en production, nous devons d'abord nous assurer que le support de cette application, à savoir la base de données graphique, est adaptée à la tâche implémentée. Pour simuler un environnement de production, nous considérons différents types de requêtes ainsi que des interrogations en parallèle. Nous avons les requêtes d'agrégation qui consistent en l'interrogation du magasin RDF pour retrouver des informations. Ce sont les requêtes les plus fréquentes. De plus, nous avons les requêtes de création, de mise à jour et de suppression d'informations dans la base de données.

4.1.1 Variables de contrôle

Nous pouvons répartir les variables de contrôle en trois catégories. D'abord, nous avons les paramètres liés au SUT, constitués du magasin RDF et des différentes API. La configuration du magasin RDF fait aussi partie de ces paramètres. En fonction du SGBD et de la licence, certains paramètres comme la mémoire maximale allouée sont limitées. Pour faire la comparaison, il est donc nécessaire de bien choisir et spécifier la version et la configuration des magasins RDF. Ensuite, nous avons l'environnement de test constitué de la machine sur laquelle nous allons mener l'évaluation et de ses caractéristiques. Les paramètres environnementaux ont une incidence directe sur les performances que peuvent atteindre les différentes solutions. Les plus importants paramètres sont le processeur, la mémoire vive, la vitesse du disque de stockage et la vitesse des transactions réseau. Le processeur intervient surtout dans la parallélisation des opérations. Plusieurs solutions adoptent un stockage en mémoire des données en cours d'utilisation. La quantité de mémoire vive devient décisive quand la quantité de données à mettre en mémoire augmente. Que ce soit pour le chargement initial des données, des mises à jour ou l'inférence, des opérations d'écriture et de lecture se font sur le disque de stockage. Le disque de stockage choisi a un impact sur la vitesse de ces opérations. Dans notre cas, la vitesse des transactions réseau aura un impact négligeable car nous avons opté pour un déploiement sur un unique hôte avec une instance par SGBD. Enfin, nous avons les paramètres liés aux données et aux requêtes sur ces données. Ces paramètres sont liés aux scénarios d'utilisation du magasin RDF. Ce sont la taille et la complexité de l'ontologie, les types de requêtes et le nombre de requêtes en parallèle. Pour évaluer l'impact de la variation de ces paramètres, nous relevons des métriques. La comparaison et l'interprétation de ces métriques pour chaque SGBD nous donnera une meilleure idée des points forts et des limites de la technologie, dans les conditions de test.

4.1.2 Métriques

Quelques métriques qui peuvent être considérées pour une évaluation de performance sont le temps de réponse aux opérations (chargement, mise à jour, requête), la précision des résultats et le rendement, en anglais *throughput*.

Le temps de réponse (ou latence) nous permet de comparer pour un ensemble de paramètres le temps écoulé entre le lancement d'une opération et la réception de la réponse. Généralement, pour les solutions natives, le temps de latence entre le lancement de la requête et le début du traitement est négligeable. Cependant, pour les frameworks, il faut compter un délai pour le lancement des différents processus de traitement. La précision des résultats mesure l'habilité du système à renvoyer des résultats corrects après une requête. Le rendement est le nombre

de requêtes exécutables par unité de temps. Cette métrique donne une idée sur la charge de travail que peut supporter le système. Deux unités de rendement sont couramment utilisées dans la littérature : Queries per second (QpS) qui représente le nombre d'exécutions d'une requête spécifique par seconde et Query mixes per hour (QMpH) qui est le nombre de requêtes aléatoires par heure.

Nous avons choisi d'évaluer quatre magasins RDF : Anzograph, GraphDB, Virtuoso et Stardog. Pour automatiser le processus et couvrir les différentes variations des paramètres, nous nous servons de frameworks de benchmark. De plus, nous menons toutes les expériences sur une seule machine, ce qui implique que nous ne testerons pas les fonctions de réplication et la création de clusters.

4.2 Jeux de données

Pour évaluer les performances des requêtes SPARQL sur des magasins RDF, nous avons besoin d'un modèle ontologique, de données pour peupler ce modèle et des requêtes SPARQL à évaluer. Un des défis de notre recherche a été de trouver des données adaptées. Avec d'autres membres du laboratoire de recherche, nous avons énuméré les jeux de données bruts liés à la sécurité informatique disponibles sur Internet. Dans cette section, nous allons présenter les données utilisées dans notre évaluation, ainsi que notre tentative de créer notre propre jeu de données sémantique à partir de données brutes.

Nous pouvons distinguer trois catégories de données utilisées dans la création d'ontologies : les données issues de graphes de connaissance du Web sémantique, les données générées par des outils en fonction du modèle d'ontologie souhaitée et les données brutes. Chaque catégorie a ses avantages en matière d'utilisation et un degré d'effort différent pour une adaptation à un scénario de recherche donné.

Données brutes

Pour notre recherche, nous devions avoir accès aux données de sécurité collectées par le SIEM de notre partenaire industriel (Desjardins). À partir de ces données, nous aurions pu créer une ontologie et un jeu de données sémantique pour nos expériences. Pour automatiser l'importation d'informations des données brutes vers l'ontologie, nous avons besoin d'étudier les outils et mécanismes de traduction des formats du SIEM et des senseurs vers un format d'ontologie tels que RDF/XML ou RDF/JSON. Ceci correspond à notre question de recherche Q4. Nous n'avons cependant pas pu accéder aux données de Desjardins. Pour pallier ce manque

de données, nous avons recherché des jeux de données bruts liés à la sécurité disponibles librement.

Plusieurs éléments sont ressortis de cette recherche. Il existe plusieurs jeux de données publics liés à la sécurité informatique. Pour un premier filtrage, nous nous sommes intéressé aux jeux de données assez récents, publiés après 2010. Un jeu de données récent a une plus grande probabilité d'avoir les caractéristiques des réseaux modernes sur lesquels notre recherche porte. Ensuite, le format des données était considéré. Pour reproduire les mêmes étapes qu'on aurait voulu sur le jeu de données initial, nous devons avoir un jeu de données brut, avec le moins d'altérations possibles. Les données les plus brutes que nous avons trouvées étaient sous la forme de captures du trafic réseau. Ces données demandent le plus de traitements, mais ont potentiellement le plus d'informations pertinentes. La taille du jeu de données nous donne une idée de la quantité d'événements et d'informations auxquels on peut s'attendre. Enfin, la méthode de création du jeu de données et les citations dans des articles de recherche ont motivé une analyse plus détaillée du jeu de données.

Après avoir sélectionné les données qui semblaient les plus appropriées, nous avons réalisé une analyse de ces données pour déterminer les scénarios en sécurité autour desquels nous pourrions développer un modèle ontologique. Le principal jeu de données utilisé a été le *CSE-CIC-IDS2018* [55]. Il est le résultat d'un projet collaboratif entre le Centre de la Sécurité des Télécommunications (CST) et le *Canadian Institute for Cybersécurity* (CIC), à l'Université du Nouveau Brunswick. Les données générées sont utilisées pour entraîner des algorithmes de détection d'anomalies. Pour générer ce jeu de données, une infrastructure informatique a été mise en place. Des attaques ont ensuite été réalisées sur cette infrastructure. Le trafic d'événements a été capturé et mis à disposition. Ce projet nous a permis de découvrir la difficulté de trouver des données de bonne qualité et en quantité. En effet, si ce jeu avait été adapté pour de l'apprentissage machine, il n'était cependant pas adapté pour modéliser une ontologie. Nous nous sommes heurtés au manque de sémantique dans les informations enregistrées et à l'absence d'informations contextuelles cruciales pour notre étude. En effet, nous n'avions pas les *logs* de chaque section importante du réseau, soit dû à l'absence de senseurs et de capteurs ou à cause de leur configuration. Par les *logs* d'administration système sur les machines hôtes Windows (*syslog*) n'étaient pas configurés pour récupérer en détails les différents événements qui passaient par les machines en question. Nous avions donc des *logs* pauvres en informations pertinentes. De plus, l'architecture et les configurations des senseurs et capteurs étaient lacunaires. Nous n'avons pas accès aux différentes règles de pare-feu ou aux règles de détection de l'IDS. Tous ces éléments de contexte manquants limitent la mise en place d'un scénario de détection réaliste. Nous avons quand même mis en place tout un environnement pour traduire les informations des données brutes vers une ontologie.

TABLEAU 4.1 Comparaison de jeux de données publics

Jeu de données	Source	Format	Taille (Go)	Description courte
Splunk BOTS v1 (2017)	Splunk	JSON CSV	6.1	Captures de trafic réseau et de journaux d'application durant une compétition de sécurité. Il comprend des attaques et remédiations. Les données sont structurées et étiquetées
Splunk BOTS v2 (2018)	Splunk	Pré-indexé	16.4	Captures de trafic réseau et de journaux d'application durant une compétition de sécurité. Il comprend des attaques et remédiations. Les données sont structurées et étiquetées
Unified Host and Network Dataset (2017)	Los Alamos National Laboratory Enterprise	CSV	185	Constituées de journaux d'hôtes Windows et de routeurs sur le réseau d'entreprise pendant 90 jours. Les champs sont décrits et les données sont pré-traitées. Les données sont structurées mais pas étiquetées.
Comprehensive, Multi-Source Cyber Security Events (2015)	Los Alamos National Laboratory Enterprise	CSV	12	Jeu de données structurées, non étiquetées, correspondant au trafic d'authentification dans le réseau d'entreprise pendant 58 jours.
MACCDC (2012)	Raytheon Technologies	PCAP	5.2	Durant la compétition de sécurité MACCDC, les participants s'introduisent dans des systèmes dédiés. Le trafic généré durant la compétition est capturé pour former ce jeu de données
CSE-CIC-IDS2018 (2018)	CIC et CSE	PCAP CSV Syslog	220	Une architecture réseau complète a été recréée sur AWS pour simuler des attaques sur un réseau d'entreprise. Le trafic réseau généré et les journaux d'événements sur les machines ont été capturés. Pour l'apprentissage machine, des fichiers CSV contiennent les valeurs des caractéristiques du jeu de données.
CICDDoS2019 (2019)	CIC	PCAP CSV	3	Simulation du comportement de 25 utilisateurs et captures des journaux d'utilisation. Les fichiers CSV contiennent des caractéristiques pour l'apprentissage machine. Les données avant le traitement sont disponibles sous forme de captures réseau.
The VERIS Community Database (2018)	Verizon	JSON	8554 entrées	Projet développé pour promouvoir le partage entre organisations de renseignements sur les incidents de sécurité.
SMIA (2013)	Swedish Defense Research Agency	PCAP Syslog CSV	414	Ce projet simule un réseau réel pour générer du trafic réseau et des journaux d'événements sur les machines. Les journaux de plusieurs outils de sécurité sont capturés.
Mordor (2020)	Chercheurs en sécurité	JSON	< 1	Événements de sécurité générés par des techniques d'attaques simulés en se basant sur le framework Mitre ATT&CK. Les événements suivent les cycles de vie d'attaques. Les données sont structurées et étiquetées
ISTS 12 (2015)	Information Security Talent Search	PCAP	383	Ensemble de captures réseau durant une compétition de sécurité. L'IDS SNORT est utilisé pour classer les événements suivant des règles. Les données sont structurées
WRCCD (2012-2020)	Western Regional Collegiate Cyber Defense Competition	PCAP	+1To	Captures de trafic réseau durant une compétition de sécurité. La compétition se déroule chaque année.
Inter-Service Academy Cyber Defense Competition (2011)	NSA	texte	12	Ensemble de journaux, donc les journaux d'IDS tels que SNORT. Ces journaux ont été capturés durant une compétition de sécurité. Les données sont structurées.

Par exemple, nous avons rejoué les captures réseau dans un IDS SNORT avec des règles de détection pour récupérer des journaux similaires à ceux qu'on trouverait dans un SOC. Nous avons fait de même pour les journaux systèmes des machines hôtes. Ces tentatives nous ont permis de reproduire quelques informations contextuelles, mais pas suffisamment pour créer une ontologie riche.

La qualité et la diversité sémantique des jeux données n'étaient pas suffisantes pour notre projet. Face à ces limites, nous avons donc concentré nos efforts sur les jeux de données sémantiques existants, même si ces données ne sont pas liées à la sécurité informatique.

Données issues du Web sémantique

Ces données utilisées dans l'évaluation de performance des magasins RDF proviennent directement de graphes de connaissance réels, disposant d'un *endpoint* SPARQL en ligne. Le principal avantage de ces données est qu'elles sont déjà sous forme de graphes dans une syntaxe RDF. Le modèle ontologique est également connu. Quant aux requêtes associées, elles sont souvent issues de journaux d'utilisation de la base de données par des utilisateurs, ce qui en fait des requêtes plus proches d'une utilisation des magasins RDF dans un cas réaliste. DBpedia [1] en est un exemple. C'est une base de connaissance construite à partir d'informations collectées dans les articles de Wikipedia. Ces informations sont structurées sous la forme d'une ontologie avec plus de 4 millions d'entités décrites, 685 classes, 2795 propriétés et plusieurs milliards de triplets au total. DBpedia met à disposition un *endpoint* SPARQL [2] pour interroger directement les données hébergées. Les requêtes sur cet *endpoint* sont journalisées. Dans un benchmark, elles peuvent être réutilisées pour simuler une utilisation réelle. Tout ceci permet d'avoir de la diversité dans le processus d'évaluation. Cependant, cette approche souffre de quelques limites. Le fait de choisir des requêtes dans les journaux d'utilisation ne permet pas de cibler un scénario en particulier. Il se pourrait que certains comportements des magasins RDF soient omis soit parce que les requêtes correspondantes n'ont pas été choisies, soit parce qu'elles sont tout simplement rarement utilisées ou inexistantes. C'est pour cette raison que des benchmarks ont été créés avec des générateurs de données et des générateurs de requêtes, basés sur les principales caractéristiques qu'on retrouve et qu'on souhaite tester dans un SGDB sémantique.

1. https ://wiki.dbpedia.org/
2. http ://dbpedia.org/sparql/

Données générées

Les jeux de données peuvent également être générés suivant une ontologie qui décrira les entités et les relations entre entités, une liste de mots qui permettra de nommer les instances d'entités et le nombre de triplets voulus. Ce dernier élément est vital pour notre évaluation car nous pouvons ainsi faire varier la taille de la base de données ontologique sans pour autant risquer de changer les propriétés globales de l'ontologie, c-à-d., les classes, la profondeur des classes et les propriétés du graphe créé. De plus, les générateurs de données construits spécialement dans le but de réaliser un benchmark ont l'avantage de produire des données dont les caractéristiques sont personnalisées pour faire ressortir les limites des magasins RDF. Ces générateurs de données sont généralement couplés de générateurs de requêtes qui génèrent des requêtes différentes suivant des modèles.

WatDiv fait partie de cette catégorie d'outils. Les générateurs de données et de requêtes de WatDiv ont été pensés pour combler les insuffisances des benchmarks qui l'ont précédé. Nous les avons utilisés pour générer des données de tailles différentes et des requêtes appartenant à des catégories différentes.

En plus de WatDiv, nous avons également utilisé une ontologie sur le contrôle de trafic aérien construite par de Miceli [56]. Cette ontologie a été majoritairement développée en utilisant la méthodologie ATOM. Le modèle ontologique contient des concepts avioniques tels que la position des appareils, le plan de vol et les modèles d'appareil. de Miceli a créé ATC-Sense pour implémenter la détection d'anomalies. ATC-Sense est une plateforme simulant l'environnement du trafic aérien comprenant les avions, les vols, les différents radars et le logiciel utilisé dans le contrôle aérien. Elle est couplée avec le modèle ontologique. Les différentes informations traversant la plateforme sont converties en instances du modèle ontologique. Ce sont ces instances qui constituent le jeu de données ATC que nous utilisons dans nos expériences. De nombreux projets de recherche de notre laboratoire s'appuyant sur ATOM pour la modélisation d'ontologies, l'utilisation de cette ontologie dans notre évaluation nous permettra d'estimer le comportement des magasins RDF face à des ontologies développées avec ATOM. Les données qui ont servies à peupler le modèle ontologique proviennent d'une émulation d'un système cyberphysique.

4.3 Framework d'évaluation

Pour notre évaluation, nous avons combiné WatDiv pour la génération du jeu de données RDF et des requêtes SPARQL et *SPARQL Query Benchmarker* pour l'automatisation du processus d'évaluation. Nous avons décrit ces deux framework dans la revue de littérature.

4.4 Scénarios

Dans cette section, nous présentons nos tests. Après avoir essayé plusieurs configurations d'évaluation, nous avons retenu celles qui étaient compatibles avec l'ensemble des magasins RDF évalués et qui nous permettaient de récupérer les métriques désirées.

Nous avons présenté plus haut les variables de contrôle pertinentes pour des tests : les systèmes, l'environnement et l'utilisation. Nous allons maintenant décrire plus en détail les choix que nous avons faits pour chaque catégorie de paramètres, ainsi que le déroulement de nos tests. Le principal objectif était de réaliser des tests entièrement reproductibles.

4.4.1 Magasins RDF

Les SUT comprennent le magasin RDF, la configuration et la version du magasin RDF et les API utilisées dans la réalisation des tests. Dans le tableau 4.2, nous avons les magasins RDF, leur version et le diminutif utilisé pour simplifier les prochaines mentions. Nous avons privilégié la configuration par défaut de chaque SGBD. Ceci dit, il est tout à fait possible d'optimiser en fonction du besoin les performances des magasins RDF. GraphDB et Stardog étant écrits en Java, nous avons fixé à 20 Go la taille du tas de mémoire (*heap memory*). Virtuoso et Anzograph quant à eux sont restés avec les configurations par défaut, ce qui leur permet d'utiliser toute la mémoire disponible. Les API des outils ont été utilisées seulement pour le chargement des données dans la base de données. Nous reviendrons en détail sur leur utilisation et leurs spécificités.

Tableau 4.2 Systèmes sous test

SGBD	Licence	Chargement de données (API)
Anzograph 2.2.1	Entreprise	*azgi*
GraphDB 9.5.0	Standard (SE)	*loadrdf*
Stardog 7.4.4	Académique	*stardog-admin*
Virtuoso 8.03.3319	Entreprise	*isql*

4.4.2 Environnement

Pour nos expériences, nous avons utilisé un ordinateur roulant sur un processeur Intel Core i7-7700K, doté de quatre cœurs, avec une vitesse de 4,20 GHz. La mémoire vive était de 32 Go. Suivant les recommandations de benchmark des SGBD, nous avons choisi comme disque de stockage un SSD NVMe de 1 To. Le système d'exploitation sur cette machine était Ubuntu 20.04.1. Pour réduire les ressources utilisées par d'autres processus, la machine était lancée

en mode *commande* et contrôlée à distance. Pour favoriser la reproductibilité et la gestion de notre environnement de test, tous les magasins RDF étaient déployés dans des conteneurs Docker. Ce choix de déploiement nous a permis de mieux surveiller la consommation en ressources des outils. Avec Docker, il est également plus simple de s'assurer que chaque magasin RDF est déployé dans des conditions similaires. Quant à la praticabilité, nous avons pu mieux segmenter nos tests et limiter les interactions entre différents tests grâce à l'utilisation de conteneurs. Cela nous a également permis de nous assurer que notre environnement puisse être reproductible car les images utilisées sont disponibles publiquement sur Docker Hub [3].

4.4.3 Tests

Tableau 4.3 Informations importantes dans les résultats recueillis des benchmark

Champs	**Valeurs**
Échauffement	1
Exécutions	10 (basique) ou pendant 180s (*stress test*)
Processus en parallèle	maximum (uniquement durant un *stress test*)
Temps total d'exécution	(secondes), erreur : -1
Temps total de réponse	(secondes), erreur : -1
Opérations par seconde	QpS
Opérations par heure	QMpH
Correction des requêtes	correcte ou incorrecte
Nombre d'erreurs	(nombre)

Pour que nos tests soient reproductibles indépendamment des jeux de données des magasins RDF évalués, nous avons mis en place une routine d'évaluation. L'importance de cette routine est également de s'assurer que nous limitons les biais liés à notre utilisation dans la comparaison des différents SGBD.

La première étape consiste en une initialisation de la base de données. Après avoir lancé l'instance Docker correspondant au magasin RDF à évaluer, nous chargeons les données dans le SGDB. Pour mesurer les performances dans des conditions optimales, nous nous servons des API fournies par les magasins RDF (voir le tableau 4.2). Toutes les API utilisées disposaient d'une option pour recueillir le temps de chargement des données. Ce chargement de données permet d'évaluer le chargement initial d'une ontologie dans un contexte similaire à un scénario réel. Dans un cas d'utilisation en production, les données initiales à importer sont de tailles variées et peuvent devenir volumineuses. Après ce chargement initial volumineux, il n'est pas rare d'avoir des mises à jour de taille moindre. Dans un SOC, les données initiales

3. https ://hub.docker.com/

peuvent être composées d'informations contextuelles sur l'infrastructure informatique, d'une banque de vulnérabilités et des événements sur une période temporelle fixée. En fonction de la plage temporelle, la quantité de données peut grandement augmenter. Pour une investigation, surtout s'il y a une suspicion d'intrusion, il est primordial que le chargement se fasse le plus rapidement possible. Pendant le chargement de l'ontologie, les ressources consommées par l'instance Docker sont mesurées. Ces ressources sont l'évolution de la consommation de la mémoire et du processeur, les données échangées à travers l'interface réseau de l'instance et les données lues et écrites sur le disque. Ces informations nous ont permis d'avoir une idée des ressources les plus importantes en fonction du magasin RDF. En effet, en identifiant suivant le magasin RDF les ressources qui sont le plus consommées, nous pouvons déterminer les avenues d'optimisation de performance par l'augmentation de ces ressources.

Après l'initialisation, nous procédons à l'interrogation des magasins RDF à l'aide de requêtes SPARQL à travers le protocole HTTP. Nous avons choisi de réaliser deux types de tests. Une première évaluation basique a consisté à mesurer le temps de réponse des magasins RDF à un ensemble de requêtes prédéterminées. Les requêtes choisies permettent d'évaluer la capacité du SGBD à traiter certaines fonctionnalités SPARQL. Ce test a pour objectif de faire une comparaison des SGBD pour chaque catégorie de requête. En gardant notre scénario d'application dans un SOC, après avoir chargé les données liées à une investigation, un analyste voudrait passer en revue les différents événements dans le SGBD à l'aide de requêtes SPARQL. Le temps de réponse à chaque requête a une incidence directe sur la productivité de l'analyste. La seconde évaluation est un *stress test*. Nous évaluons les performances du magasin RDF quand le nombre de requêtes en parallèle augmente. Sur une investigation, plusieurs analystes peuvent avoir besoin d'accéder simultanément aux mêmes ressources. Le magasin RDF doit être en mesure de répondre à chacune des interrogations dans un temps raisonnable.

Pour réaliser ces tests, nous avons utilisé le framework d'automatisation SPARQL Query Benchmarker. Les requêtes SPARQL se font à travers HTTP, ce qui rend ce framework compatible avec les magasins RDF que nous testons. L'authentification est également supportée et les identifiants sont transmis au framework à travers des paramètres. Le résumé des résultats est enregistré dans un fichier CSV. Ces informations comprennent le nombre d'opérations combinées par heure, le nombre de requêtes par heure et par seconde, le temps moyen de réponse, le temps total de l'expérience et d'autres statistiques. Cependant, ces informations n'étant pas suffisantes pour vérifier l'exactitude des réponses aux requêtes SPARQL, nous avons également relevé la sortie détaillée du framework. Dans cette sortie, nous pouvons comparer les résultats de chaque requête SPARQL aux résultats attendus. Cette comparai-

son permet de déterminer quand une erreur s'est produite ou si un magasin n'a pas pu trouver certains résultats d'un BGP.

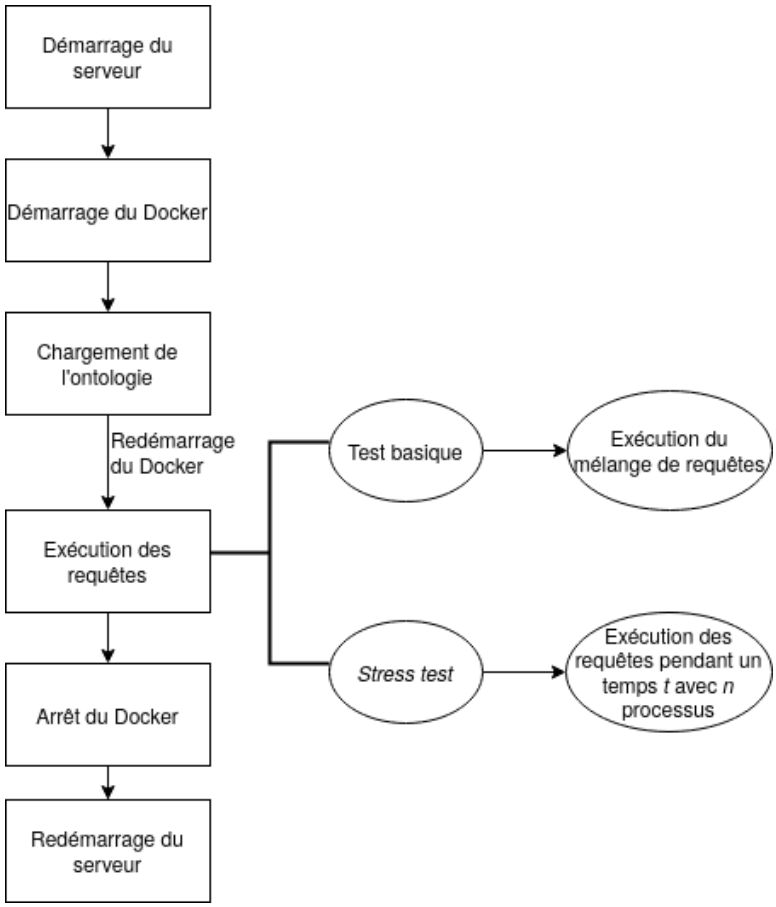


FIGURE 4.2 Flux d'exécution des expériences

Pour limiter les interférences entre les expériences sur chaque magasin RDF et entre les différentes étapes de ces expériences, nous redémarrons l'instance Docker après chaque étape. Nous vérifions également que la mémoire et le processeur alloués aux processus précédents ont été libérés le cas échéant. L'échauffement au début de chaque test permet de s'assurer que le magasin RDF est dans les conditions optimales d'utilisation. Dans la figure 4.2, nous illustrons le flux d'exécution des expériences pour chaque SUT. Pour chaque étape, nous avons scripté les opérations à exécuter, une fois de plus pour limiter les différences d'utilisation entre les différents magasins RDF.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX wsdbm: <http://db.uwaterloo.ca/~galuc/wsdbm/>
INSERT DATA{
    graph <http://db.uwaterloo.ca/~galuc/wsdbm/Context/C#id>
    {
wsdbm:ProductC rdf:type wsdbm:ProductCategory13838;
        wsdbm:hasGenre wsdbm:SubGenre92;
        <http://ogp.me/ns#title> "Nouveau produit C";
        <http://schema.org/contentRating> "18";
        <http://ogp.me/ns#tag> wsdbm:Topic211;
         <http://schema.org/actor> wsdbm:User48575.
wsdbm:PurchaseC wsdbm:purchaseFor wsdbm:ProductC.
wsdbm:UserC wsdbm:likes wsdbm:ProductC.
    }
}
```

Listing 4.1 Requête SPARQL pour l'insertion d'un graphe contenant: création d'un nouveau produit et de métadonnées sur le produit, ajout d'un énoncé d'achat du produit, ajout d'un énoncé de préférence du produit par un utilisateur

Données et requêtes

Nous avons réalisé nos expériences sur deux jeux de données : les données générées par WatDiv et l'ontologie ATC. WatDiv nous a permis d'étudier l'impact de la taille du jeu de données sur les performances des requêtes SPARQL dans les SUT.

Grâce au générateur de données de WatDiv et au modèle ontologique fourni, nous avons généré trois jeux de données de tailles 10, 100 et 500 millions de triplets. Ces données ont occupé sur le disque de stockage respectivement 1,5 Go, 15 Go et 74 Go. Ensuite, à partir des modèles de requêtes, nous avons généré des requêtes associées au modèle ontologique utilisé. Il existe au total vingt modèles de requêtes sur quatre BGP : trois modèles de requêtes linéaires, sept modèles de requêtes en étoile, cinq modèles de requêtes en flocon et trois modèles de requêtes combinant les trois BGP précédant. Pour limiter le nombre de tests à réaliser, nous avons pris un échantillon dans chaque BGP pour un total de 10 requêtes. À ces requêtes d'agrégation, nous avons ajouté 6 requêtes de mise à jour de la base de données. Ces dernières requêtes permettent de tester la mise à jour d'un énoncé (*statement*) existant, l'insertion d'un nouvel énoncé et la suppression d'un énoncé de l'ontologie.

Quant à l'ontologie ATC, elle comprend environ 5 millions de triplets. À partir des scénarios de détection initialement modélisés dans le mémoire de de Miceli, nous avons construit cinq requêtes, comprenant une requête de mise à jour d'énoncés dans l'ontologie.

Ces requêtes constituent les mélanges d'opérations qui seront lancées durant le benchmark. Le mélange d'opérations consiste à interroger les SUT avec les différentes permutations des requêtes. En changeant l'ordre des requêtes et en faisant la moyenne par requête, on réduit l'impact de résultats issus d'une mise en cache des statistiques précédentes.

4.5 Conclusion

Dans ce chapitre, nous avons détaillé les différentes expériences que nous avons réalisées sur les magasins RDF. Ces expériences nous ont permis de déterminer les points forts de chaque magasin testé dans un scénario précis. Dans le prochain chapitre, nous analyserons les résultats de nos expériences.

CHAPITRE 5 RÉSULTATS

5.1 Résultats d'expérience

Nos expériences nous ont permis de collecter des statistiques sur les performances des magasins RDF suivant les configurations présentées au chapitre précédent. Dans ce chapitre, nous analysons ces résultats dans l'optique de mieux discerner les forces et les faiblesses des SUT.

Tableau 5.1 Conventions de nommage des jeux de données

Diminutif	**Description**
ATC	Jeux de données comprenant 2 millions de triplets et 2 millions de triplets inférés
WatDiv10M	Jeux de données comprenant 10 millions de triplets
WatDiv100M	Jeux de données comprenant 100 millions de triplets
WatDiv500M	Jeux de données comprenant 500 millions de triplets

5.1.1 Temps de chargement

Les coûts de chargement des jeux de données dans les magasins RDF ont été mesurés sur des systèmes dans leur configuration par défaut. En fonction du cas d'utilisation, il est possible d'optimiser ces performances en demandant au vendeur les paramètres optimaux. À la figure 5.1, nous avons une comparaison sur une échelle logarithmique des temps de chargement en fonction des deux ontologies utilisées. Pour avoir une idée de la valeur numérique de ces performances, le tableau 5.2 présente les temps en secondes de chargement des jeux de données.

Tableau 5.2 Temps de chargement des ontologies dans les magasins RDF

	Temps de chargement (s)			
Ontologies	Anzograph	GraphDB	Stardog	Virtuoso
ATC	5	25	14	12
WatDiv10M	20	63	38	38.5
WatDiv100M	199	933	394	419
WatDiv500M	1057	6272	2884	2212

On remarque que Anzograph a les meilleures performances sur tous les jeux de données. Ses temps de chargement sont plus de deux fois inférieurs au second SUT le plus performant. Ceci s'explique par le fait que Anzograph est le seul OLAP de notre étude. Elle est donc modélisée spécialement pour charger et analyser d'importantes quantités de données. Stardog et

Virtuoso suivent avec Virtuoso légèrement plus performante que Stardog. Par défaut, Anzograph, Virtuoso et Stardog n'infèrent pas au chargement de l'ontologie. GraphDB pratique le *forward chaining* et procède donc à l'inférence de nouveaux triplets durant l'importation d'une ontologie. Pour les jeux de données générés par WatDiv, la stratégie d'inférence n'a pas d'impact sur le nombre de triplets. Cependant, sur l'ontologie ATC, GraphDB importe, du fait de l'inférence, deux fois plus de triplets que les autres magasins RDF. En désactivant l'inférence, le chargement de l'ontologie ATC passe de 25 s à 13 s.

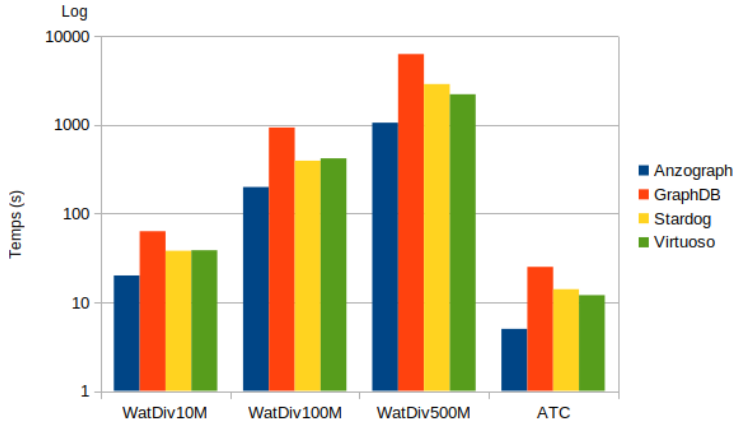


FIGURE 5.1 Temps de chargement des magasins RDF

En analysant le temps d'importation d'un jeu de données en fonction du nombre de triplets, on remarque que Anzograph, Stardog et Virtuoso ont des performances presque linéaires. En fixant le matériel utilisé, il est donc possible, pour un même modèle ontologique, d'estimer le temps de chargement quand le nombre de triplets augmente. Ceci ne marche que si aucune autre opération telle que l'inférence n'est exécutée.

Concernant la méthode d'importation des données, GraphDB nécessite que la base de données soit hors ligne pour charger les données. Pour ne pas avoir à arrêter la base de données, il faut dupliquer le matériel. Ceci est un inconvénient surtout quand on vise de larges jeux de données. Stardog et GraphDB étant écrits en Java, nous avons pu constater une amélioration des performances en augmentant la taille du tas de 8 Go à 20 Go. La documentation nous informe cependant qu'à partir de 64 Go, il est préférable d'augmenter horizontalement les ressources plutôt que verticalement. Anzograph est un SGBD de graphe en mémoire. De ce fait, la majorité des opérations se font en mémoire. Avoir les données et le graphe entier en mémoire est un avantage d'un point de vue analytique car l'accès aux données est plus rapide. Cependant, le matériel, notamment la mémoire vive disponible limite la taille de l'ontologie à importer. Nous avons par exemple tenté sans succès d'importer un jeu de données de 1

milliard de triplets sur la machine hôte dotée de 32 Go de mémoire. Le maximum que nous avons pu importer avec 32 Go de mémoire était 700 millions de triplets. De plus, vu que le graphe reste en mémoire, il faut également prévoir suffisamment de mémoire pour toutes autres opérations à rouler sur les données graphiques. Dans Virtuoso, nous avons pu importer le jeu de données de 1 milliard de triplets en 4827 s. On a toujours un coût quasi-linéaire quand le jeu de données passe de 10 millions de triplets à 100 millions et 1000 millions de triplets.

Cette différence entre Anzograph et Virtuoso s'explique par la nature différente des deux magasins RDF. Virtuoso stocke les données sur le disque et n'importe en mémoire que les triplets nécessaires. De plus, contrairement à Anzograph, Virtuoso n'est pas un OLAP. Les performances analytiques ne sont pas le principal objectif de ce SGBD. Ainsi, Anzograph est plus performante au chargement que toutes les bases de données graphiques de notre évaluation, mais demande aussi le plus de ressources en mémoire.

Au-delà des paramètres optimisés que le vendeur peut proposer, une manière d'améliorer les performances serait de diviser le jeu de données en plusieurs fichiers contenant un nombre fixe de triplets. Ce nombre dépendrait du SGBD. Ensuite, en parallélisant l'importation des différents fichiers, on pourrait s'attendre à de meilleures performances. Ceci est encore plus marqué lorsqu'on distribue la charge sur plusieurs serveurs.

5.1.2 Temps de réponse et d'exécution des requêtes

Dans cette section, nous analysons les résultats des benchmarks sur le temps de réponse et d'exécution aux requêtes SPARQL. Le temps de réponse est le temps entre l'envoi de la requête et le début de réponse à celle-ci. Quant au temps d'exécution, c'est le temps total qui s'écoule entre l'envoi de la requête et la réception de la réponse. Il comprend en plus le temps total de réception des réponses. Il est donc supérieur ou égal au temps de réponse. La comparaison de ces temps permet de déterminer des facteurs influençant le temps d'exécution des requêtes et de cibler les points forts des SUT. En effet, pour un même temps d'exécution, un magasin RDF avec un temps de réponse plus élevé peut indiquer un temps supérieur du lancement de son framework, mais un traitement plus rapide de la requête. Quand la quantité d'informations à analyser augmente, ce magasin pourrait bénéficier de meilleures performances.

Les temps de réponse et d'exécution ont été évalués suivant une évolution de la taille du jeux de données, mais aussi en fonction d'une augmentation de la charge sur le SUT. Cette dernière évaluation nous a permis de déterminer, dans le contexte de notre expérience, les magasins RDF plus résilients aux contraintes.

Pour une évaluation juste des SUT, nous avons considéré l'exactitude des résultats trouvés par les magasins RDF. En effet, le temps d'exécution dépend de la profondeur de la recherche. De plus, la précision est également un facteur important de performances. Pour intégrer ce facteur, nous avons comparé le nombre de résultats de chaque requête à une référence. Pour l'ensemble des expériences sur tous les jeux de données, les magasins RDF GraphDB, Stardog et Virtuoso ont renvoyé 100% des résultats. Quant à Anzograph, sur une requête précise, le nombre de résultats est inférieur au nombre attendu. Il s'avère que le nombre de résultats qui peut être récupéré en utilisant l'API REST [1] est limité à 1000. En interrogeant le magasin RDF avec l'API propriétaire, nous trouvons le nombre de résultats attendu dans un temps similaire.

Anzograph a également la particularité parmi les SUT de compiler les requêtes SPARQL. Ainsi, à la première interrogation de la base de données avec une requête spécifique, celle-ci est d'abord compilée. Ceci induit un coût supplémentaire qui est largement rattrapé lors des requêtes suivantes. Pour ne pas pénaliser ce SGBD mais aussi pour tenir compte d'aberrations dans les résultats, les plus petits et plus grands temps recueillis ont été écartés avant de faire la moyenne temporelle.

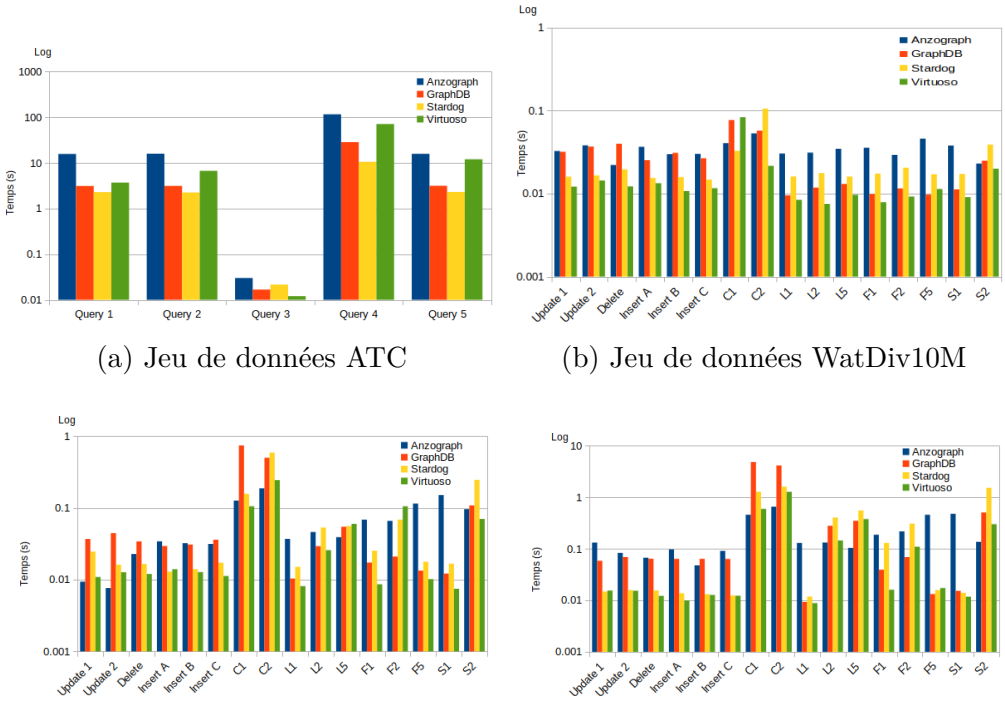
Évaluation avec un unique client

Nous évaluons l'évolutivité des SUT en faisant varier la taille du jeu de données WatDiv à 10 M, 100 M et 500 M (millions). Les requêtes utilisées dans l'évaluation sur les jeux de données WatDiv sont créées suivant les BGP (linéaire :L, flocon :F, étoile :S, complexe :C) décrits au chapitre 4. Nous avons construit quelques instances de requêtes par BGP. Nous analysons également les performances sur l'ontologie ATC, construite à partir de la méthodologie ATOM. Les instances de l'ontologie ATC ont été créés à partir du système cyberphysique émulé par de Miceli. Pour cette première évaluation, les requêtes ont été faites avec un unique client. Dans la figure 5.2, nous pouvons observer les temps moyens d'exécution des requêtes selon le SUT, pour chaque jeu de données utilisé.

Pour le jeu de données ATC, Stardog et GraphDB ont les meilleurs performance d'ensemble, avec Stardog sensiblement supérieures. Sur DB-Engines, GraphDB est classée comme étant un des meilleurs magasins RDF pour l'inférence à large échelle. En gardant les paramètres par défaut, GraphDB a inféré sur les données initiales pour former un jeu de données deux fois plus larges que les autres SUT. Ceci a eu un impact sur ses performances. Anzograph a les pires performances sur ce jeu de données. En tant qu'OLAP, elle est spécialisée dans l'extraction d'informations dans des volumes importants de données et l'analyse de ces informations. Les

1. https ://en.wikipedia.org/wiki/Representational_state_transfer

FIGURE 5.2 Comparaison des temps d'exécution aux requêtes des SUT sur les différents jeux de données. En abscisse, nous avons les différentes requêtes.



(a) Jeu de données ATC

(b) Jeu de données WatDiv10M

(c) Jeu de données WatDiv100M

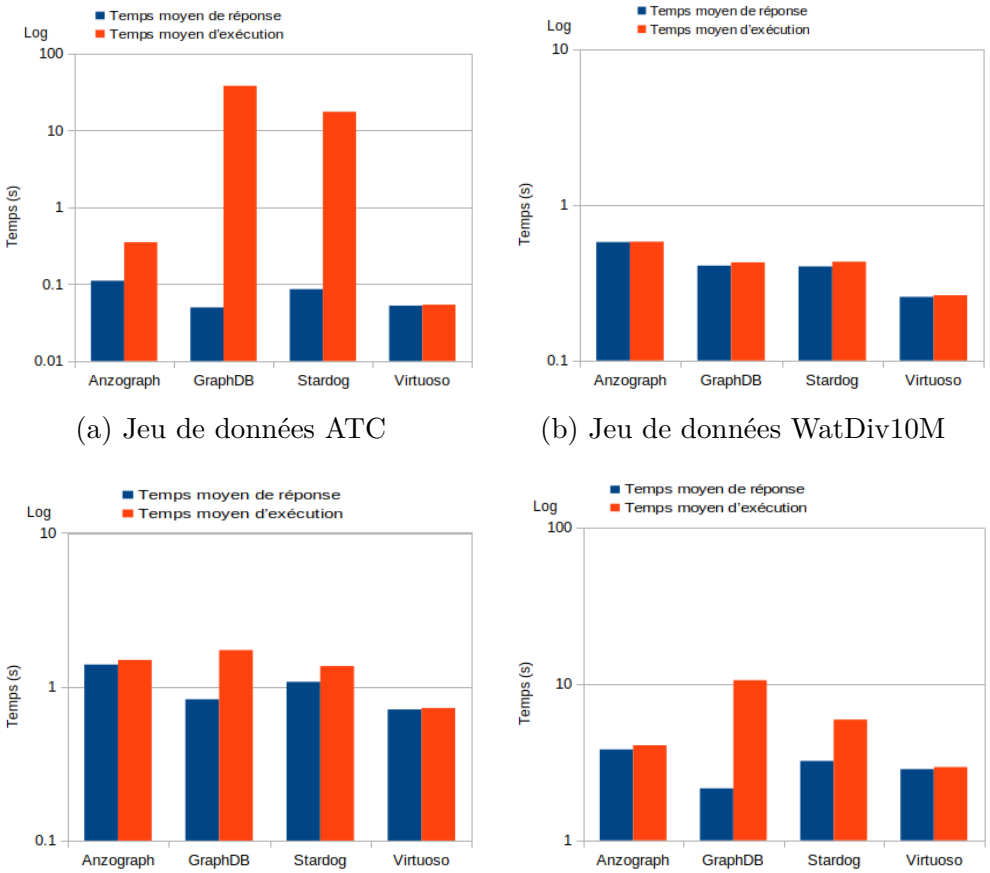
(d) Jeu de données WatDiv500M

requêtes utilisées pour cette tâche sont complexes et comprennent des fonctions analytiques. Cependant, pour le jeu de données ATC, les requêtes inspirées de la détection d'anomalies sont des requêtes courtes et simples, sans fonctions analytiques. Cette différence peut expliquer les performances d'Anzograph face aux autres SUT.

Quant aux jeux de données WatDiv, nous pouvons observer à la figure 5.2 les temps d'exécution moyens par requête, pour chaque magasin RDF. Ces temps ont été calculés sur 10 mélanges d'opérations, en retirant les temps aux extrémités pour réduire un biais lié à des comportements erratiques. Dans WatDiv10M, Virtuoso a les meilleurs temps d'exécution, avec un pire cas pour la requête C1. Stardog a la deuxième meilleure performance pour les requêtes de mise à jour, mais est moins performante que GraphDB pour le reste des requêtes. Anzograph n'est bien classée que pour les requêtes C1, C2 et S2. Dans WatDiv100M, Virtuoso a une fois encore les meilleures performances d'ensemble. Les performances relatives d'Anzograph se sont améliorées. Mais, elle garde le plus de pires cas pour les requêtes L1, F1, F5 et S1. Dans le dernier jeu de données, WatDiv500M, Virtuoso a une fois de plus les meilleurs temps d'ensemble. Stardog a une performance similaire à Virtuoso pour les

requêtes de mise à jour. Malgré trois pires cas, Anzograph relève tout de même cinq des meilleurs temps d'exécution.

FIGURE 5.3 Comparaison des temps moyens de réponses et d'exécution des SUT sur les différents jeux de données : une moyenne sur l'ensemble des requêtes a été calculée



(a) Jeu de données ATC

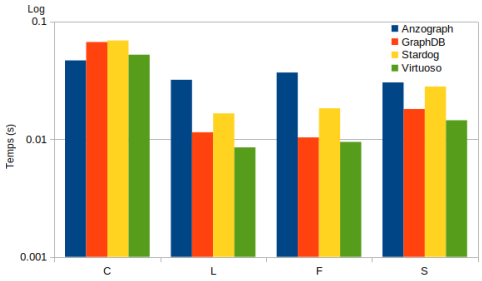
(b) Jeu de données WatDiv10M

(c) Jeu de données WatDiv100M

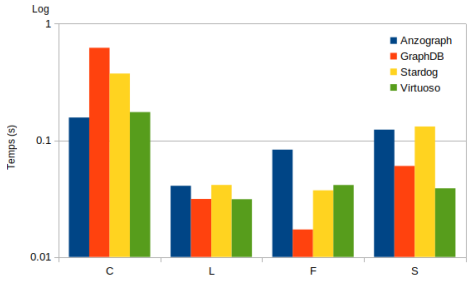
(d) Jeu de données WatDiv500M

En plus des temps d'exécution, à la figure 5.3, nous pouvons comparer les temps moyens de réponse aux temps moyens d'exécution. Pour tous les jeux de données, Anzograph et Virtuoso ont la plus petite différence entre les temps de réponse et d'exécution. Ceci peut être dû à une meilleure indexation lors de l'importation des données qui permet de retrouver plus rapidement les triplets ciblés. Pour les jeux de données WatDiv, on peut observer comment l'écart se creuse quand la taille des données augmente. Les performances de GraphDB et Stardog se dégradent quand la taille des données augmentent. On peut donc choisir une de ces bases de données en fonction du volume de données qu'on souhaite traiter, en plus de la complexité de ces données.

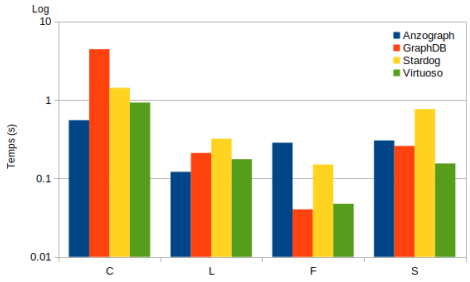
FIGURE 5.4 Comparaison des SUT en fonction des BGP. Pour chaque catégorie de requêtes, le temps moyen de réponses est représenté.



(a) Jeu de données WatDiv10M



(b) Jeu de données WatDiv100M



(c) Jeu de données WatDiv500M

Pour avoir une meilleure idée des points forts et des faiblesses des magasins RDF, nous avons rassemblé les résultats par BGP, à la figure 5.4. Anzograph a les meilleures performances pour les requêtes complexes (C). Quand la taille du jeu de données augmente, on observe une amélioration significative des performances relatives sur les requêtes linéaires (L) pour Anzograph. Ses performances relatives sur les requêtes en flocon (F) restent les pires cas tout au long de l'expérience. Virtuoso a de meilleures performances pour les requêtes en étoile (S). GraphDB a les meilleures performances pour les requêtes en flocon (F). Ainsi, on remarque qu'aucun SUT n'est le meilleur sur tous les types de jointure.

Évaluation avec des clients simultanés

Dans cette sous-section, nous évaluons la résilience des SUT face à un nombre croissant de connexions simultanées. Nous avons fait varier les connexions simultanées de 1 à 32 clients, avec un pas de 2. Nous avons retenu les résultats sur WatDiv100M car ce sont les résultats les plus significatifs. Sur WatDiv500M, la mémoire disponible après le chargement du jeu de données n'était pas suffisante pour faire des tests sur Anzograph.

Dans le tableau 5.3, nous pouvons observer les résultats du *stress test*. Pour chaque SUT et pour chaque requête, nous avons relevé le QpS, le QpH, ainsi que le ratio entre le QpS pour un client et le QpS pour plusieurs clients simultanées. Pour vérifier l'exactitude des résultats renvoyés pour chaque requête, nous avons une colonne *Résultats* qui correspond au nombre de triplets renvoyés à la suite de la requête. Ces résultats correspondent à la somme des réponses pour chaque client. GraphDB, Stardog et Virtuoso ont toutes renvoyé les résultats attendus pour chaque requête. Quant à Anzograph, comme expliqué précédemment, les requêtes à travers l'API REST sont limitées à 1 000 réponses, ce qui a impacté les résultats de la requête S2.

Les métriques QpS et QpH nous permettent de classer les SUT suivant leur capacité à supporter une charge croissante. Virtuoso est largement devant sur ce plan, suivie par Stardog. Anzograph a les pires performances. Ceci peut s'expliquer par le fait qu'Anzograph est optimisée pour de l'analytique mais surtout pour fonctionner de manière distribuée. En comparant les rapports entre le QpS sur un client et le QpS sur plusieurs clients simultanés, nous cherchons à déterminer la stabilité des SUT. Virtuoso a un rapport proche de 1 pour toutes les requêtes. L'augmentation de la charge a peu impacté ses performances. Stardog a connu des améliorations et des baisses de performances. L'écart-type est plus élevé. On peut observer la baisse de performance d'Anzograph avec ce ratio. C'est donc le SUT qui est le moins résilient à l'augmentation de la charge.

Tableau 5.3 Performances des SUT sous stress sur WatDiv100M : le ratio est le rapport entre l'utilisation d'un seul client et de plusieurs clients en parallèle

SUT	Anzograph				GraphDB				Stardog				Virtuoso			
Requêtes	QpS	QpH	Ratio	Résultats	QpS	QpH	Ratio	Résultats	QpS	QpH	Ratio	Résultats	QpS	QpH	Ratio	Résultats
C1	1.8	6375.3	3.8	12663	0.7	2551.3	1.9	12663	4.5	16239.1	1.4	12663	8.5	30589.4	1.1	12663
C2	1.0	3729.2	3.9	1386	1.1	3848.9	1.9	1386	1.2	4420.8	1.4	1386	3.7	13167.5	1.1	1386
L1	5.3	18926.0	3.1	378	106.3	382785.4	0.9	378	83.9	302048.0	0.8	378	141.1	507985.3	0.9	378
L2	3.7	13330.5	5.3	39690	20.3	73145.9	1.7	39690	16.0	57542.4	1.2	39690	35.5	127843.4	1.1	39690
L5	4.3	15637.4	3.7	52266	18.6	66826.8	1.0	53109	14.2	51241.7	1.3	53109	24.7	88755.6	0.7	53109
F1	3.0	10651.5	4.6	189	57.4	206780.6	1.0	189	34.0	122254.6	1.2	189	106.5	383567.5	1.1	189
F2	2.7	9767.2	5.6	5481	36.0	129488.0	1.3	5481	10.7	38698.6	1.4	5481	9.0	32362.4	1.1	5481
F5	1.9	6744.4	3.8	1323	146.0	525693.7	0.5	1323	89.2	320989.8	0.6	1323	145.5	523903.0	0.7	1323
S1	1.7	6251.6	4.1	252	93.3	335933.4	0.9	252	96.7	348045.5	0.6	252	178.9	644112.7	0.8	252
S2	4.6	16520.0	1.9	63000	5.0	18058.9	1.8	169407	2.8	10034.7	1.5	169407	13.4	48212.1	1.1	169407

5.2 Synthèse des comparaisons

Dans cette section, nous vous présentons une synthèse des résultats comparatifs selon les critères qualitatifs déroulés dans le chapitre 3 et selon les résultats d'expériences que nous avons réalisées.

En termes de facilité d'utilisation, Stardog et GraphDB sont les solutions les plus faciles d'utilisation, avec Stardog sensiblement devant GraphDB. Parmi les magasins RDF, ce sont

les SGBD ayant le plus d'outils intégrés à l'installation. Les outils d'administration natifs permettent de gérer toutes les étapes d'ingestion, de manipulation et de visualisation des graphes de connaissance, dans une interface uniformisée. Pour un développeur d'ontologies, ces solutions sont les plus faciles à prendre en main. Anzograph est également facile d'utilisation, manque cependant de fonctionnalités présentes nativement dans les précédentes solutions. Virtuoso est particulièrement difficile à prendre en main car c'est un SGBD multi-modèle (relationnel). Les ressources documentaires ne sont pas aussi bien expliquées et souvent difficiles à trouver.

En termes de capacités analytiques, Anzograph est le meilleur outil évalué. Pour rappel, c'est une base de données OLAP spécialisée pour le traitement analytique en parallèle. Les points forts sont la présence d'algorithmes graphiques et autres fonctions analytiques pré-compilés. Pour pallier le manque de fonctionnalités natives de visualisation, ce magasin RDF intègre des *notebooks* Jupyter et Zeppelin. Stardog a également de nombreuses fonctions analytiques intéressantes en entreprises. Il s'intègre aussi bien avec les *notebooks* Jupyter et Zeppelin et d'autres outils d'analytiques avancés. GraphDB est la meilleure solution pour l'inférence à grande échelle. Virtuoso est un OLTP, ce qui explique le peu de fonctionnalités analytiques.

Concernant les performances, nous devons distinguer les performances d'ingestion de données et les performances de gestion des requêtes SPARQL. En termes d'ingestion de données, Anzograph est la meilleure solution. Nous avons pu le confirmer avec les résultats expérimentaux. Ensuite, nous avons Virtuoso, puis Stardog et GraphDB. Quant aux requêtes, Virtuoso a été la solution la plus constante. Pour traiter des requêtes, avec ou sans augmentation de charges, en considérant des agents en parallèle, Virtuoso a eu les meilleures performances d'ensemble sur les scénarios de nos expériences. Stardog et GraphDB ont eu des performances assez similaires. Anzograph a la particularité de compiler les requêtes en C++ lors de la première occurrence.

En termes d'évolutivité, Anzograph est une base de données conçues pour le traitement massif en parallèle. Avec Virtuoso, elle a les meilleures performances évolutives. Pour de gros volumes de données, ces magasins RDF l'emportent. Stardog a de meilleures performances que GraphDB quand on augmente le volume de données et qu'on distribue la base de données dans plusieurs nœuds.

5.3 Grille de décisions selon les métiers d'un SOC

Les différents magasins RDF que nous avons étudiés ont des caractéristiques et des avantages différents. Dans cette section, nous faisons le lien avec les besoins dans un SOC. Dans notre revue de littérature, nous avons défini quelques métiers dans un SOC.

Les analystes, selon leur rôle et selon l'organisation, auront des besoins différents. Cependant, nous tentons d'estimer ces besoins en fonction d'une définition de leurs responsabilités et avec des statistiques sur le traitement moyen d'une intrusion.

Les analystes *Tiers 1* mènent le moins d'analyses avancées sur les données. En effet, le triage d'événements se fait majoritairement suivant les alertes remontées dans le SIEM et suivant le *playbook* de réponse aux incidents de l'organisation. Le volume de données concerné par l'analyse est la totalité des données qui transitent dans le réseau d'entreprise. La détection et l'analyse se fait en temps presque réel. Donc, le principal besoin des analystes *Tiers 1* est d'avoir des outils rapides. En ce sens, Anzograph permet d'ingérer le plus rapidement d'importants volumes de données. Si le volume est moins important, Virtuoso peut être une meilleure solution car l'accès en parallèle par plusieurs analystes sera mieux géré. Dans ce contexte, la stratégie d'inférence de GraphDB induira une perte de performances à cause du volume de données. Une inférence constante sur un important volume de données est peu efficient.

Les analystes *Tiers 2* font de la réponse aux incidents. Le volume de données concerné par l'investigation dépendra de l'incident mais sera relativement inférieur à celui d'un analyste *Tiers 1*. De plus, la plage de temps concernée sera plus longue pour remonter jusqu'à la cause de l'incident. Pour mener une investigation sur une alerte donnée, ils s'aident de plusieurs outils analytiques. Leur principal besoin est la présence de fonctionnalités analytiques pour déterminer les relations dans d'importants volumes de données, venant de différentes sources. En ce sens, Anzograph et Stardog sont les meilleures solutions pour bénéficier des fonctionnalités analytiques. Leur intégration avec les outils analytiques qu'on peut retrouver dans le nuage d'une entreprise facilitera la prise en main par les analystes. Virtuoso n'étant pas une solution analytique, elle est la moins recommandée pour ce cas d'utilisation.

Les analystes *Tiers 3* recherchent des menaces non détectées par les outils classiques. Pour se faire, ils développent leurs outils pour suivre une méthodologie propre. La plage de données est la plus longue pour traquer des menaces infiltrées depuis plusieurs mois. Le volume de données concerné par leur investigation est l'ensemble des données emmagasinées sur la période d'investigation, ce qui pourrait s'avérer énorme. Un des intérêts à élargir la plage temporelle est d'augmenter la probabilité de trouver des événements corrélés qui sont les

symptômes d'une intrusion. Ils ont donc besoin d'outils à la fois flexibles pour s'adapter à leur méthodologie d'investigation et ayant de bonnes performances analytiques. La vitesse de réponse aux requêtes ou d'ingestion des données ne sont pas la priorité, mais reste importante. Pour cela, Anzograph, Stardog ou GraphDB peuvent être utiles. Anzograph et Stardog permettront d'offrir de la flexibilité à travers les *notebooks* tout en garantissant des performances et des fonctionnalités analytiques. Virtuoso est compatible avec plusieurs langages de programmation, donc il est plus facile de créer des outils qui interagissent directement avec le magasin RDF. GraphDB a de très bonnes performances en termes d'inférence. Cela permettra d'automatiser le raisonnement de l'analyste.

De manière générale, pour faire un choix, il faut aussi choisir entre la performance et le coût de l'infrastructure. En effet, nous avons Anzograph qui a de bonnes performances mais qui est un SGBD entièrement en mémoire. Pour charger le volume de données nécessaires dans le cas d'un analyste *Tiers 3* par exemple, la quantité de mémoire nécessaire peut être importante. Il faudra donc choisir entre charger les données rapidement mais devoir installer une infrastructure plus avancée ou charger les données plus lentement avec une infrastructure limitée.

5.4 Conclusions

En définitive, aucune solution n'est la meilleure sur tous les plans et pour tous les cas d'utilisation. Nous recommandons donc de coupler les magasins RDF en fonction du besoin pour combler les lacunes de chaque outil. Ces résultats nous donnent une idée des performances et du comportement d'un magasin RDF dans un scénario précis. Mais, la variation de comportements quand les paramètres changent nous montrent que pour avoir de meilleurs résultats, nous devons refaire nos expériences dans les conditions réelles du scénario étudié. Nous pourrons nous servir de nos résultats actuels pour mieux préparer et analyser des expériences dans les conditions réelles.

CHAPITRE 6 CONCLUSION

Dans ce chapitre, nous discutons les résultats de notre travaux et leur lien avec notre objectif de recherche. Ensuite, nous présentons les limitations de notre approche. Nous terminons sur des améliorations futures à nos travaux.

6.1 Discussion

L'objectif de nos travaux était de valider l'utilisation des ontologies dans un SOC traitant d'importantes quantités de données. Nous devions donc valider la maturité des technologies sémantiques en matière de performance pour une utilisation à grande échelle dans le contexte d'un SOC. Ceci nous a amené à subdiviser notre recherche en quatre questions de recherche. Cependant, nous n'avons pas pu avoir accès à l'infrastructure SOC et aux données du SOC de notre partenaire industriel (Desjardins). Pour cette raison, nous avons mené une analyse générique en nous assurant que les données utilisées donnent des résultats pertinents pour notre besoin dans un SOC.

6.1.1 Q1 : Quelles sont les bases de données et de traitements ontologiques à grande échelle en termes de caractéristiques pertinentes pour une utilisation dans le contexte d'un SOC ?

Pour notre première question de recherche, nous avons analysé les magasins RDF capables de traiter de l'information à grande échelle. À partir de la littérature, nous avons identifié les caractéristiques pertinentes à considérer dans un SGBD sémantique.

Dans le chapitre 3, nous avons présenté ces caractéristiques et notre méthodologie pour choisir les magasins RDF compatibles avec nos besoins. Parmi les différents SGBD sémantiques présents dans l'écosystème, nous en avons retenu quelques uns. Dans la section 3.1, l'architecture de ces outils est présentée de sorte à mieux comprendre leurs forces et faiblesses, mais aussi à pouvoir les comparer. Nous avons pu montrer qu'il existe bien des magasins RDF conçus pour le déploiement à grande échelle de graphes sémantiques.

Pour aider à choisir parmi les solutions existantes, dans la section 3.2, nous avons fait une comparaison des magasins RDF retenus par rapport aux fonctionnalités clé. Un tableau comparatif de ces principales caractéristiques permettra de choisir la solution adéquate au contexte d'utilisation voulue. Nous avons été particulièrement sensible à l'importance de la flexibilité des outils autant pour le développement du système expert que pour l'utilisation

et l'extension par l'utilisateur final, à savoir l'analyste de sécurité. Ceci nous a permis de retenir les solutions les plus prometteuses pour une évaluation expérimentale.

6.1.2 Q2 : Quelles sont les performances relatives des magasins RDF dans le traitement d'importants volumes de données à l'échelle des données transitant dans un SOC ?

Pour répondre à cette seconde question, nous nous sommes d'abord servi des informations recueillies dans le chapitre 3. Les performances relevées par des benchmarks nous ont orienté vers les magasins RDF adéquats pour un traitement d'importants volumes de données.

Pour confirmer ces données, nous avons établi une méthodologie expérimentale dans le chapitre 4. Un des principaux buts visés lors de la conception de cette méthodologie était la reproductibilité des expériences. Nous nous sommes inspiré de la littérature et d'évaluations précédentes pour mettre en œuvre un plan expérimental. Pour mener ces expériences, nous avons exploré les jeux de données sémantiques disponibles et avons même tenté de créer un jeu de données sémantiques à partir de données brutes. Les expériences menées nous ont permis à la fois de comparer les performances des magasins RDF suivant une augmentation de la taille des données et de la charge, mais aussi de montrer que les temps de chargement des données et les temps de réponses aux requêtes peuvent être bas. En effet, nous avons pu observer que même pour les plus larges jeux de données étudiés, les temps de réponses étaient inférieurs à la seconde pour la majorité des magasins RDF.

Nous pouvons donc supposer qu'en optimisant la configuration de ces magasins, les performances des requêtes SPARQL sont suffisantes pour soutenir une utilisation dans le contexte d'un SOC.

6.1.3 Q3 : Comment évalue-t-on dans le contexte d'un SOC les bases de données et de traitements ontologiques à grande échelle en termes de flexibilité de création de requêtes ?

Un aspect de notre recherche portait sur la flexibilité d'utilisation des magasins RDF pour le développement de systèmes experts, mais aussi leur flexibilité auprès de l'utilisateur final. Nous avons pu relever dans les magasins RDF étudiés les caractéristiques importantes en terme de flexibilité. D'après notre analyse des métiers d'analystes de sécurité dans un SOC et des différents besoins suivant le métier, nous avons conclu que les principaux besoins pour l'utilisateur final sont la présence dans les SGBD d'outils et de capacités analytiques avancés, mais aussi la possibilité d'étendre le système ou de le coupler avec des outils existants.

Ainsi, l'intégration avec les autres technologies particulièrement les technologies analytiques fréquemment retrouvés dans les nuages d'entreprises et la facilité de modifier, étendre ou automatiser le comportement du magasin RDF à travers la programmation de scripts sont des exemples de caractéristiques qui ont un impact sur les performances d'utilisation de la solution finale.

6.1.4 Q4 :Quelles sont les performances des outils de cueillette et de traduction d'information des senseurs d'un SOC vers la base de données ontologiques ?

Cette question de recherche n'a pas été traitée. Nous n'avons pas pu avoir accès à l'infrastructure et aux données issues du SOC de notre partenaire industriel. Nous avons donc choisi de nous concentrer sur les trois autres questions de recherche.

Grâce à nos interactions avec des analystes de sécurité et notre revu de littérature, nous avons exploité les résultats des questions de recherche pour mettre en relation les besoins des analystes de sécurité dans un SOC aux différents outils que nous avons évalués. Cette grille de décision permet d'orienter vers le choix d'un outil en fonction du besoin et des caractéristiques disponibles.

6.1.5 Résultats

Les résultats de nos expériences sont prometteurs. Pour le chargement d'importants volumes de données, Anzograph est la solution la mieux adaptée. C'est un OLAP spécialisé dans le traitement distribué des données en mémoire. Virtuoso est ressortie comme un outil polyvalent, occupant la première ou la deuxième place pour toutes nos expériences. Sa principale force est dans la résilience face à une augmentation de la charge. Les performances de GraphDB se sont dégradées quand la taille du jeu de données a augmenté. Compte tenu de ses performances pour l'inférence et de sa facilité d'utilisation, pour des scénarios sur des jeux de données moins importants, GraphDB est un outil adapté. Stardog est un outil très orienté vers l'analytique avec plusieurs optimisations disponibles. Pour un déploiement dans un contexte d'entreprise, il est particulièrement intéressant quant à sa compatibilité avec les autres outils analytiques d'un environnement infonuagique.

En couplant ces résultats à l'analyse des caractéristiques des outils dans le chapitre 3, nous recommandons l'utilisation de Anzograph comme outil final en production. Ses performances répondent aux différents besoins des analystes de sécurité. Anzograph est un spécialisé dans le traitement massif en parallèle. Les traitements se faisant en mémoire, pour limiter le coût de

l'infrastructure, Anzograph peut être couplé avec un SGBD performant sur les transactions comme Virtuoso. On fait ainsi un compromis entre le coût d'exploitation et les performances. Stardog est un outil orienté vers l'analytique comme Anzograph. Il est une alternative moins performante à Anzograph. GraphDB est recommandée pour des projets de recherche ou des projets en phase de test. C'est un outil polyvalent et extensible. Ses performances se dégradent quand le volume et la complexité des données augmentent mais sa polyvalence et sa flexibilité en font un bon outil avant de passer en production.

6.2 Limitations

Une des principales limitations de notre recherche est l'absence d'étude des outils sémantiques basés sur des frameworks de traitement distribué tels que Apache Spark et Hadoop. Nous avons pu voir dans la littérature qu'une voie de recherche de méthode d'optimisation du traitement de larges volumes de données sémantiques était l'utilisation de frameworks distribués. Pour réduire la surface de notre étude, nous avons volontairement retiré ces outils de notre état de l'art. Cependant, une comparaison de solutions natives et de solutions basées sur des frameworks de traitement distribué pourrait donner une meilleure vue d'ensemble des performances des outils sémantiques de manière générale.

Une autre limitation est l'évaluation des magasins RDF sur une seule machine hôte. En effet, des outils comme Anzograph sont spécialisés dans le déploiement distribué. Faire une évaluation sur une seule machine hôte ne nous permet pas de déterminer l'étendu des performances atteignables avec ces outils. Dans la même lancée, le fait de déployer sur une machine hôte nous a empêché de tester des solutions uniquement disponibles dans le nuage telles que Amazon Neptune.

Ensuite, les expériences que nous avons réalisées nous donnent de nombreuses informations, mais ne nous ont pas permis de déterminer les cas extrêmes d'utilisation des magasins RDF. Nous aurions par exemple voulu augmenter la quantité de mémoire et de stockage du matériel utilisé pour pouvoir faire des tests sur de plus gros jeux de données. Se faisant, nous aurions pu déterminer les limites des magasins RDF et mieux délimiter une grille de décision. De plus, les requêtes que nous avons utilisées permettaient de tester les différents types de jointure, mais manquaient de fonctionnalités analytiques. Par exemple, Anzograph et Stardog n'ont pas eu les meilleures performances sur les requêtes, mais sont optimisées pour le traitement de fonctions analytiques dans les requêtes.

Enfin, nous n'avons pas réalisé de tests avec de vraies données et de vraies requêtes issues d'un SOC. Ce point est important car nous avons pu déterminer que les performances des

magasins RDF sont sensibles au type de données et de requêtes. Ce qui nous amène aux travaux futurs.

6.3 Travaux futurs

Dans cette section, nous présentons des travaux futurs qui répondraient directement aux limitations soulevées dans la section 6.2.

Tout d'abord, nous suggérons l'étude d'outils intégrant des frameworks de traitement distribué de données. Notre objectif de recherche se pose dans le contexte d'un SOC. Ces frameworks de traitement distribué sont présents dans les environnements d'entreprises pour plusieurs cas d'utilisation. Au-delà de l'évolutivité que les outils intégrant ces frameworks promettent, on pourrait avoir une meilleure intégration avec le reste de l'environnement informatique. Le gain de performances n'est donc plus restreint à l'utilisation d'applications sémantiques, mais aussi à la préparation en amont.

Ensuite, la mise en place des expériences dans un environnement distribué avec des capacités matérielles élevées augmenterait le réalisme et la pertinence des résultats. Cet environnement peut être dans le nuage pour tester les solutions exclusivement chez un fournisseur de services, ou dans un centre de données. Cette approche permettra de mieux déterminer les limites des magasins RDF.

Enfin, pour des résultats plus pertinents dans le contexte d'un SOC, il est nécessaire d'avoir des données plus proches de celles d'un SOC. Nous avons pu remarquer que les performances des requêtes SPARQL ont baissé sur le jeu de données ATC qui est pourtant le moins volumineux. Un jeu de données générique ne suffit donc pas à évaluer les outils que nous voulons utiliser.

RÉFÉRENCES

[1] É. Ducharme, "Détection d'intrusion à l'aide d'un système expert basé sur l'ontologie," mémoire de maîtrise, Dép. de génie informatique et génie logiciel, École Polytechnique de Montréal, Montréal, QC, 2017. [En ligne]. Disponible : http://publications.polymtl.ca/2923/

[2] C. Onwubiko, "CoCoa : An ontology for cybersecurity operations centre analysis process," dans *2018 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)*, 2018, p. 1–8.

[3] "Cost of a data breach report," IBM, Rapport technique, 2020. [En ligne]. Disponible : https://www.ibm.com/security/digital-assets/cost-data-breach-report

[4] Oracle. Security operations center. [En ligne]. Disponible : https://www.oracle.com/fr/cloud/soc-security-operations-center.html

[5] C. Zhong, J. Yen, P. Liu et R. F. Erbacher, "Learning from experts' experience : Toward automated cyber security data triage," *IEEE Systems Journal*, vol. 13, n°. 1, p. 603–614, 2019.

[6] A. D'Amico et K. Whitley, *The Real Work of Computer Network Defense Analysts*. Springer Berlin Heidelberg, 2008, p. 19–37. [En ligne]. Disponible : https://doi.org/10.1007/978-3-540-78243-8_2

[7] "The state of malware detection and prevention," Ponemon Institute, Rapport technique, 2016. [En ligne]. Disponible : http://go.cyphort.com/Ponemon-Report-Page.html

[8] "Top 10 data and analytics technology trends that will change your business," Gartner, Rapport technique, 2019. [En ligne]. Disponible : https://www.gartner.com/document/3906812

[9] Oracle. Qu'est-ce qu'un graphe de données ? [En ligne]. Disponible : https://www.oracle.com/big-data/what-is-graph-database/

[10] H. Paulheim, "Knowledge graph refinement : A survey of approaches and evaluation methods," *Semantic Web*, vol. 8, p. 489–508, 2017.

[11] S. Malenfant-Corriveau, "Proposition d'une méthode de développement d'ontologie pour un système expert en sécurité," mémoire de maîtrise, Dép. de génie informatique et génie logiciel, École Polytechnique de Montréal, Montréal, QC, 2017. [En ligne]. Disponible : http://publications.polymtl.ca/2922/

[12] N. A. Premathilaka, A. C. Aponso et N. Krishnarajah, "Review on state of art intrusion detection systems designed for the cloud computing paradigm," 2013.

[13] J. Anderson, “Computer security threat monitoring and surveillance,” 1980.

[14] A. Milenkoski, M. Vieira, S. Kounev, A. Avritzer et B. D. Payne, “Evaluating computer intrusion detection systems : A survey of common practices,” *ACM Computing Surveys*, vol. 48, n°. 1, 2015. [En ligne]. Disponible : https://doi.org/10.1145/2808691

[15] R. Koch, “Towards next-generation Intrusion Detection,” *2011 3rd International Conference on Cyber Conflict (ICCC)*, 2011.

[16] MCAFEE. What is SIEM. [En ligne]. Disponible : https://www.mcafee.com/enterprise/en-ca/security-awareness/operations/what-is-siem.html

[17] F. B. Kokulu, A. Soneji, T. Bao, Y. Shoshitaishvili, Z. Zhao, A. Doupé et G.-J. Ahn, “Matched and mismatched SOCs : A qualitative study on security operations center issues,” dans *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. Association for Computing Machinery, 2019, p. 1955–1970.

[18] “Magic quadrant for security information and event management,” Gartner, Rapport technique, 2020. [En ligne]. Disponible : https://www.gartner.com/document/3981040

[19] “Magic quadrant for security information and event management,” Gartner, Rapport technique, 2018. [En ligne]. Disponible : https://www.gartner.com/document/3894573

[20] M. Uschold et M. Gruninger, “Ontologies : principles, methods and applications,” *The Knowledge Engineering Review*, vol. 11, n°. 2, p. 93–136, 1996.

[21] N. Guarino, “Semantic matching : Formal ontological distinctions for information organization, extraction, and integration,” dans *Information Extraction A Multidisciplinary Approach to an Emerging Information Technology*. Springer Berlin Heidelberg, 1997, p. 139–170.

[22] L. Coppolino, S. D’Antonio, I. A. Elia et L. Romano, “From intrusion detection to intrusion detection and diagnosis : An ontology-based approach,” dans *Software Technologies for Embedded and Ubiquitous Systems*. Springer Berlin Heidelberg, 2009, p. 192–202.

[23] G. G. Granadillo, Y. B. Mustapha, N. Hachem et H. Debar, “An ontology-driven approach to model SIEM information and operations using the SWRL formalism,” *International Journal of Electronic Security and Digital Forensics*, vol. 4, n°. 2/3, p. 104–123, 2012. [En ligne]. Disponible : https://doi.org/10.1504/IJESDF.2012.048412

[24] P. Salini et J. Shenbagam, “Article : Prediction and classification of web application attacks using vulnerability ontology,” *International Journal of Computer Applications*, vol. 116, n°. 21, p. 42–47, 2015.

[25] H. Huang, T. Chuang, Y. Tsai et C. Lee, “Ontology-based intelligent system for malware behavioral analysis,” dans *International Conference on Fuzzy Systems*, 2010.

[26] A. Sadighian, “Intrusion detection from heterogenous sensors,” thèse de doctorat, Dép. de génie informatique et génie logiciel, École Polytechnique de Montréal, Montréal, QC, 2015. [En ligne]. Disponible : http://publications.polymtl.ca/1702/

[27] S. N. Narayanan, A. Ganesan, K. Joshi, T. Oates, A. Joshi et T. Finin, “Early detection of cybersecurity threats using collaborative cognition,” dans *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, 2018, p. 354–363.

[28] W. Ali, M. Saleem, B. Yao, A. Hogan et A.-C. N. Ngomo, “Storage, indexing, query processing, and benchmarking in centralized and distributed rdf engines : A survey,” 2020.

[29] G. Aluç, O. Hartig, M. T. Özsu et K. Daudjee, “Diversified stress testing of RDF data management systems.” Springer International Publishing, 2014, p. 197–212.

[30] *W3C SPARQL 1.1 Query Language*, Norme, 2013. [En ligne]. Disponible : https://www.w3.org/TR/sparql11-query/

[31] M. Morsey, J. Lehmann, S. Auer et A.-C. Ngonga Ngomo, “Dbpedia SPARQL benchmark – performance assessment with real queries on real data,” dans *International Semantic Web Conference*. Springer Berlin Heidelberg, 2011, p. 454–469.

[32] M. Salehpour et J. G. Davis, “A comparative analysis of knowledge graph query performance,” *ArXiv*, vol. abs/2004.05648, 2020. [En ligne]. Disponible : http://arxiv.org/abs/2004.05648

[33] Bloor Research. Bases de données graphiques en 2020. [En ligne]. Disponible : https://www.bloorresearch.com/research/graph-database-2020/

[34] DB Engines. Classement des bases de données. [En ligne]. Disponible : https://db-engines.com/en/ranking/rdf+store

[35] Apache Jena. Documentation de Jena. [En ligne]. Disponible : https://jena.apache.org/getting_started/

[36] W. Huang, S. A. Raza, O. Mirzov et L. Harrie, “Assessment and benchmarking of spatially enabled RDF stores for the next generation of spatial data infrastructure,” *ISPRS International Journal of Geo-Information*, vol. 8, nº. 7, p. 310, 2019. [En ligne]. Disponible : http://dx.doi.org/10.3390/ijgi8070310

[37] DB Engines. Évaluation de Jena TDB. [En ligne]. Disponible : https://db-engines.com/en/system/Apache+Jena+-+TDB

[38] Apache Jena. Documentation de Jena Fuseki. [En ligne]. Disponible : https://jena.apache.org/documentation/fuseki2/

[39] ——. Documentation des requête avec Jena. [En ligne]. Disponible : https://jena.apache.org/documentation/query/

[40] W3C. Large triple stores. [En ligne]. Disponible : https://www.w3.org/wiki/LargeTripleStores

[41] Apache Jena. Documentation de Jena TDB. [En ligne]. Disponible : https://jena.apache.org/documentation/tdb/

[42] DB Engines. Évaluation de GraphDB. [En ligne]. Disponible : https://db-engines.com/en/system/GraphDB

[43] Ontotext. Documentation de GraphDB. [En ligne]. Disponible : https://graphdb.ontotext.com/documentation/

[44] A. Schätzle, M. Przyjaciel-Zablocki, S. Skilevic et G. Lausen, "S2RDF : RDF querying with SPARQL on spark," *CoRR*, vol. abs/1512.07021, 2015. [En ligne]. Disponible : http://arxiv.org/abs/1512.07021

[45] DB Engines. Évaluation de Virtuoso. [En ligne]. Disponible : https://db-engines.com/en/system/Virtuoso

[46] O. Erling et I. Mikhailov, *RDF Support in the Virtuoso DBMS*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2009, p. 7–24. [En ligne]. Disponible : https://doi.org/10.1007/978-3-642-02184-8_2

[47] DB Engines. Évaluation de Stardog. [En ligne]. Disponible : https://db-engines.com/en/system/Stardog

[48] ——. Évaluation de Blazegraph. [En ligne]. Disponible : https://db-engines.com/en/system/Blazegraph

[49] ——. Évaluation de Neptune. [En ligne]. Disponible : https://db-engines.com/en/system/Amazon+Neptune

[50] ——. Évaluation de Anzograph. [En ligne]. Disponible : https://db-engines.com/en/system/AnzoGraph+DB

[51] IBM. OLAP. [En ligne]. Disponible : https://www.ibm.com/cloud/learn/olap

[52] "Graph and RDF databases 2016," Bloor, Rapport technique, 2020. [En ligne]. Disponible : https://www.bloorresearch.com/research/graph-and-rdf-databases-2016/download-pdf/

[53] A. Babar, "An approach to represent and transform application specific constraints for an intrusion detection system," mémoire de maîtrise, Queen's University, Kingston, ON, 2020. [En ligne]. Disponible : http://hdl.handle.net/1974/27685

[54] L.-P. Morel, “Using ontologies to detect anomalies in the sky,” Masters thesis, Dép. de génie informatique et génie logiciel, École Polytechnique de Montréal, Montréal, QC, 2017. [En ligne]. Disponible : http://publications.polymtl.ca/2818/

[55] Canadian Institute for Cybersecurity. Jeu de données CSE-CIC-IDS2018. [En ligne]. Disponible : https://www.unb.ca/cic/datasets/ids-2018.html

[56] J.-Y. de Miceli, “Proposition d’une solution pour la détection d’attaques informatiques sophistiquées contre les communications ADS-B,” mémoire de maîtrise, Dép. de génie informatique et génie logiciel, École Polytechnique de Montréal, Montréal, QC, 2020. [En ligne]. Disponible : http://publications.polymtl.ca/

[57] O. Erling, “Virtuoso, a hybrid RDBMS/graph column store,” *IEEE Data Engineering Bulletin*, vol. 35, p. 3–8, 2012.

[58] M. F. Husain, L. Khan, M. Kantarcioglu et B. Thuraisingham, “Data Intensive Query Processing for Large RDF Graphs Using Cloud Computing Tools,” dans *2010 IEEE 3rd International Conference on Cloud Computing*, 2010, p. 1–10, iSSN : 2159-6190.

[59] M. Saleem, G. Szárnyas, F. Conrads, S. A. C. Bukhari, Q. Mehmood et A.-C. Ngonga Ngomo, “How Representative Is a SPARQL Benchmark ? An Analysis of RDF Triplestore Benchmarks.” ACM Press, 2019.

[60] I. Abdelaziz, R. Harbi, Z. Khayyat et P. Kalnis, “A survey and experimental comparison of distributed SPARQL engines for very large RDF data,” *Proceedings of the VLDB Endowment*, vol. 10, nº. 13, 2017. [En ligne]. Disponible : http://dl.acm.org/citation.cfm?doid=3151106.3151109

[61] *The Semantic Web – ISWC 2011*. Springer Berlin Heidelberg, 2011, vol. 7031. [En ligne]. Disponible : https://link.springer.com/10.1007/978-3-642-25073-6

[62] G. A. Atemezing et F. Amardeilh, “Benchmarking Commercial RDF Stores with Publications Office Dataset.” Springer International Publishing, 2018. [En ligne]. Disponible : http://link.springer.com/10.1007/978-3-319-98192-5_54

[63] M. Schmidt, T. Hornung, G. Lausen et C. Pinkel, “SP2Bench : A SPARQL Performance Benchmark,” *ArXiv*, 2008. [En ligne]. Disponible : http://arxiv.org/abs/0806.4627

[64] M. Saleem, Q. Mehmood et A.-C. Ngonga Ngomo, “FEASIBLE : A Feature-Based SPARQL Benchmark Generation Framework.” Springer International Publishing, 2015, series Title : Lecture Notes in Computer Science.

[65] J. Huang, D. J. Abadi et K. Ren, “Scalable SPARQL querying of large RDF graphs,” *Proceedings of the VLDB Endowment*, 2011.

[66] “Amazon neptune,” Bloor, Rapport technique, 2020. [En ligne]. Disponible : https://www.bloorresearch.com/research/amazon-neptune/

[67] “Cambridge semantics anzograph,” Bloor, Rapport technique, 2020. [En ligne]. Disponible : https://www.bloorresearch.com/research/cambridge-semantics-anzograph-2020/

[68] “Ontotext GraphDB,” Bloor, Rapport technique, 2020. [En ligne]. Disponible : https://www.bloorresearch.com/research/ontotext-graphdb-and-the-ontotext-platform/

[69] “Stardog,” Bloor, Rapport technique, 2020. [En ligne]. Disponible : https://www.bloorresearch.com/research/stardog-2020/

[70] Eclipse RDF4J. Documentation de Eclipse RDF4J. [En ligne]. Disponible : https://rdf4j.org/about/

[71] W3C. SPARQL endpoints. [En ligne]. Disponible : https://www.w3.org/wiki/SparqlEndpoints

[72] Stardog. Documentation de Stardog. [En ligne]. Disponible : https://www.stardog.com/docs/

ANNEXE A REQUÊTES SPARQL POUR LE JEU DE DONNÉES WATDIV

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX wsdbm: <http://db.uwaterloo.ca/~galuc/wsdbm/>
PREFIX gr: <http://purl.org/goodrelations/>

CLEAR graph <http://db.uwaterloo.ca/~galuc/wsdbm/Context/A#id>;
INSERT DATA{
    graph <http://db.uwaterloo.ca/~galuc/wsdbm/Context/A#id>
    {
wsdbm:ProductA rdf:type wsdbm:ProductCategoryA;
wsdbm:hasGenre wsdbm:SubGenre92;
<http://ogp.me/ns#title> "Nouveau produit A - Updated";
<http://schema.org/contentRating> "18";
<http://ogp.me/ns#tag> wsdbm:Topic211;
<http://schema.org/actor> wsdbm:User48575.
wsdbm:OfferA gr:includes wsdbm:ProductA.
wsdbm:PurchaseA wsdbm:purchaseFor wsdbm:ProductA.
wsdbm:UserA wsdbm:likes wsdbm:ProductA.
    }
}
```

Listing A.1 Requête SPARQL Update 1

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX wsdbm: <http://db.uwaterloo.ca/~galuc/wsdbm/>
PREFIX gr: <http://purl.org/goodrelations/>
CLEAR graph <http://db.uwaterloo.ca/~galuc/wsdbm/Context/A#id>;
INSERT DATA{
    graph <http://db.uwaterloo.ca/~galuc/wsdbm/Context/A#id>
    {
wsdbm:ProductA2 rdf:type wsdbm:ProductCategoryA2;
wsdbm:hasGenre wsdbm:SubGenre923;
 <http://ogp.me/ns#title> "Nouveau produit A2 - Updated";
 <http://schema.org/contentRating> "19";
 <http://ogp.me/ns#tag> wsdbm:Topic2110;
 <http://schema.org/actor> wsdbm:User40575.
wsdbm:OfferA2 gr:includes wsdbm:ProductA2.
wsdbm:PurchaseA2 wsdbm:purchaseFor wsdbm:ProductA2.
wsdbm:UserA2 wsdbm:likes wsdbm:ProductA2.
    }
}
```

Listing A.2 Requête SPARQL Update 2

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
CLEAR graph <http://db.uwaterloo.ca/~galuc/wsdbm/Context/1#id>;
CLEAR graph <http://db.uwaterloo.ca/~galuc/wsdbm/Context/B#id>;
CLEAR graph <http://db.uwaterloo.ca/~galuc/wsdbm/Context/C#id>;
```

Listing A.3 Requête SPARQL Delete

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX wsdbm: <http://db.uwaterloo.ca/~galuc/wsdbm/>
PREFIX gr: <http://purl.org/goodrelations/>
INSERT DATA{
    graph <http://db.uwaterloo.ca/~galuc/wsdbm/Context/1#id>
    {
wsdbm:Product1 rdf:type wsdbm:ProductCategory2;
wsdbm:hasGenre wsdbm:SubGenre92;
<http://ogp.me/ns#title> "Nouveau produit";
<http://schema.org/contentRating> "17";
<http://ogp.me/ns#tag> wsdbm:Topic245;
<http://schema.org/actor> wsdbm:User32989.
wsdbm:Offer10532 gr:includes wsdbm:Product1.
wsdbm:Purchase112021 wsdbm:purchaseFor wsdbm:Product1.
wsdbm:User48575 wsdbm:likes wsdbm:Product1.
    }
}
```

Listing A.4 Requête SPARQL Insert 1

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX wsdbm: <http://db.uwaterloo.ca/~galuc/wsdbm/>
PREFIX gr: <http://purl.org/goodrelations/>
INSERT DATA{
    graph <http://db.uwaterloo.ca/~galuc/wsdbm/Context/B#id>
    {
wsdbm:ProductB rdf:type wsdbm:ProductCategory13838;
wsdbm:hasGenre wsdbm:SubGenre92;
 <http://ogp.me/ns#title> "Nouveau produit B";
 <http://schema.org/contentRating> "18";
 <http://ogp.me/ns#tag> wsdbm:Topic211;
 <http://schema.org/actor> wsdbm:User48575.
wsdbm:OfferB gr:includes wsdbm:ProductB.
wsdbm:PurchaseB wsdbm:purchaseFor wsdbm:ProductB.
wsdbm:UserB wsdbm:likes wsdbm:ProductB.
    }
}
```

Listing A.5 Requête SPARQL Update B

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX wsdbm: <http://db.uwaterloo.ca/~galuc/wsdbm/>
PREFIX gr: <http://purl.org/goodrelations/>
INSERT DATA{
    graph <http://db.uwaterloo.ca/~galuc/wsdbm/Context/C#id>
    {
wsdbm:ProductC rdf:type wsdbm:ProductCategory13838;
wsdbm:hasGenre wsdbm:SubGenre92;
 <http://ogp.me/ns#title> "Nouveau produit C";
 <http://schema.org/contentRating> "18";
 <http://ogp.me/ns#tag> wsdbm:Topic211;
 <http://schema.org/actor> wsdbm:User48575.
wsdbm:OfferC gr:includes wsdbm:ProductC.
wsdbm:PurchaseC wsdbm:purchaseFor wsdbm:ProductC.
wsdbm:UserC wsdbm:likes wsdbm:ProductC.
    }
}
```

Listing A.6 Requête SPARQL Insert C

```
SELECT ?v0 ?v4 ?v6 ?v7  WHERE {
        ?v0 <http://schema.org/caption> ?v1 .
        ?v0 <http://schema.org/text> ?v2 .
        ?v0 <http://schema.org/contentRating> ?v3 .
        ?v0 <http://purl.org/stuff/rev#hasReview> ?v4 .
        ?v4 <http://purl.org/stuff/rev#title> ?v5 .
        ?v4 <http://purl.org/stuff/rev#reviewer> ?v6 .
        ?v7 <http://schema.org/actor> ?v6 .
        ?v7 <http://schema.org/language> ?v8 .
}
```

Listing A.7 Requête SPARQL C1

```
SELECT ?v0 ?v3 ?v4 ?v8 WHERE {
        ?v0 <http://schema.org/legalName> ?v1 .
        ?v0 <http://purl.org/goodrelations/offers> ?v2 .
        ?v2 <http://schema.org/eligibleRegion> wsdbm:Country5 .
        ?v2 <http://purl.org/goodrelations/includes> ?v3 .
        ?v4 <http://schema.org/jobTitle> ?v5 .
        ?v4 <http://xmlns.com/foaf/homepage> ?v6 .
        ?v4 wsdbm:makesPurchase ?v7 .
        ?v7 wsdbm:purchaseFor ?v3 .
        ?v3 <http://purl.org/stuff/rev#hasReview> ?v8 .
        ?v8 <http://purl.org/stuff/rev#totalVotes> ?v9 .
}
```

Listing A.8 Requête SPARQL C2

```
SELECT ?v0 ?v2 ?v3 WHERE {
        ?v0 wsdbm:subscribes wsdbm:Website3276 .
        ?v2 <http://schema.org/caption> ?v3 .
        ?v0 wsdbm:likes ?v2 .
}
```

Listing A.9 Requête SPARQL L1

```
PREFIX gn:<http://www.geonames.org/ontology#>
SELECT ?v1 ?v2 WHERE {
        wsdbm:City3 gn:parentCountry ?v1 .
        ?v2 wsdbm:likes wsdbm:Product0 .
        ?v2 <http://schema.org/nationality> ?v1 .
}
```

Listing A.10 Requête SPARQL L2

```
PREFIX gn:<http://www.geonames.org/ontology#>
SELECT ?v0 ?v1 ?v3 WHERE {
        ?v0 <http://schema.org/jobTitle> ?v1 .
        wsdbm:City3 gn:parentCountry ?v3 .
        ?v0 <http://schema.org/nationality> ?v3 .
}
```

Listing A.11 Requête SPARQL L5

```
SELECT ?v0 ?v2 ?v3 ?v4 ?v5 WHERE {
        ?v0 <http://ogp.me/ns#tag> wsdbm:Topic164.
        ?v0 rdf:type ?v2 .
        ?v3 <http://schema.org/trailer> ?v4 .
        ?v3 <http://schema.org/keywords> ?v5 .
        ?v3 wsdbm:hasGenre ?v0 .
        ?v3 rdf:type wsdbm:ProductCategory2 .
}
```

Listing A.12 Requête SPARQL F1

```
SELECT ?v0 ?v1 ?v2 ?v4 ?v5 ?v6 ?v7 WHERE {
        ?v0 <http://xmlns.com/foaf/homepage> ?v1 .
        ?v0 <http://ogp.me/ns#title> ?v2 .
        ?v0 rdf:type ?v3 .
        ?v0 <http://schema.org/caption> ?v4 .
        ?v0 <http://schema.org/description> ?v5 .
        ?v1 <http://schema.org/url> ?v6 .
        ?v1 wsdbm:hits ?v7 .
        ?v0 wsdbm:hasGenre wsdbm:SubGenre95 .
}
```

Listing A.13 Requête SPARQL F2

```
PREFIX gr: <http://purl.org/goodrelations/>
SELECT ?v0 ?v1 ?v3 ?v4 ?v5 ?v6 WHERE {
        ?v0 <http://purl.org/goodrelations/includes> ?v1 .
        wsdbm:Retailer786 gr:offers ?v0 .
        ?v0 <http://purl.org/goodrelations/price> ?v3 .
        ?v0 <http://purl.org/goodrelations/validThrough> ?v4 .
        ?v1 <http://ogp.me/ns#title> ?v5 .
        ?v1 rdf:type ?v6 .
}
```

Listing A.14 Requête SPARQL F5

```
PREFIX gr: <http://purl.org/goodrelations/>
SELECT ?v0 ?v1 ?v3 ?v4 ?v5 ?v6 ?v7 ?v8 ?v9 WHERE {
        ?v0 gr:includes ?v1 .
        wsdbm:Retailer15 gr:offers ?v0 .
        ?v0 gr:price ?v3 .
        ?v0 gr:serialNumber ?v4 .
        ?v0 gr:validFrom ?v5 .
        ?v0 gr:validThrough ?v6 .
        ?v0 <http://schema.org/eligibleQuantity> ?v7 .
        ?v0 <http://schema.org/eligibleRegion> ?v8 .
        ?v0 <http://schema.org/priceValidUntil> ?v9 .
}
```

Listing A.15 Requête SPARQL S1

```
PREFIX dcterms: <http://purl.org/dc/terms/Location>
SELECT ?v0 ?v1 ?v3 WHERE {
        ?v0 dcterms:Location ?v1 .
        ?v0 <http://schema.org/nationality> wsdbm:Country0 .
        ?v0 wsdbm:gender ?v3 .
        ?v0 rdf:ype wsdbm:Role2 .
}
```

Listing A.16 Requête SPARQL S2

ANNEXE B REQUÊTES SPARQL POUR LE JEU DE DONNÉES ATC

```
PREFIX atc-core:
<http://pyxis.ece.queensu.ca/graph/atc/ontologies/atc-core#>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?plane ?icao
WHERE {
    ?plane rdf:type atc-core:Aircraft;
       atc-core:hasICAOAddress ?icao
       }
```

Listing B.1 Requête SPARQL Query 1

```
PREFIX atc-core:
<http://pyxis.ece.queensu.ca/graph/atc/ontologies/atc-core#>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?plane ?id
WHERE {
    ?plane rdf:type atc-core:Aircraft;
       atc-core:hasRegisterID ?id;
       }
}
```

Listing B.2 Requête SPARQL Query 2

```
PREFIX :
<http://pyxis.ece.queensu.ca/graph/atc/ontologies/dds-topics/flight-plan#>

PREFIX atc-core:
<http://pyxis.ece.queensu.ca/graph/atc/ontologies/atc-core#>

PREFIX atc-data:
<http://pyxis.ece.queensu.ca/graph/atc/ontologies/atc-data#>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?proprio
WHERE {
        ?proprio rdf:type atc-core:AircraftOwner;
             atc-core:ownsAircraft ?plane.
    ?plane atc-core:hasAircraftType atc-data:AN2.
}
```

Listing B.3 Requête SPARQL Query 3

```
select * where {
        ?s ?p ?o .
}
```

Listing B.4 Requête SPARQL Query 4

```
PREFIX :
<http://pyxis.ece.queensu.ca/graph/atc/ontologies/dds-topics/flight-plan#>

PREFIX atc-core:
<http://pyxis.ece.queensu.ca/graph/atc/ontologies/atc-core#>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT *
WHERE {
    {
    SELECT * WHERE{
    ?plane rdf:type atc-core:Aircraft;
           atc-core:hasRegisterICAO ?icao.}
    }

    UNION
       {
    SELECT * WHERE{
    ?plane rdf:type atc-core:Aircraft;
           atc-core:hasRegisterID ?id.}
    }
}
```

Listing B.5 Requête SPARQL Query 5