Les bases de données graphes pour la science des données : implantation d'entrepôts des données et prédiction d'interactions protéine-protéine

Hajer Akid

UNIVERSITÉ DE STRASBOURG
en cotutelle avec
l'UNIVERSITÉ DE SFAX (Tunisie)

Mathématiques, Sciences de l'Information et de l'Ingénieur (MSII)

Laboratoire des sciences de l'ingénieur, de l'informatique

et de l'imagerie (ICube)

THÈSE présentée par :

Hajer AKID

soutenue le : **12 décembre 2022**

pour obtenir le grade de : **Docteur de l'université de Strasbourg**

Discipline/ Spécialité : Informatique

Les bases de données graphes pour la science des données : Implantation d'entrepôts de données et prédiction d'interactions protéine-protéine

THÈSE dirigée par :

M. BEN AYED Mounir	Professeur, université de Sfax
M. LACHICHE Nicolas	Maître de conférences, université de Strasbourg

RAPPORTEURS :

Mme SMAÏL-TABBONE Malika	Maître de conférences, université de Lorraine
M. Jamel FEKI	Professeur, université de Sfax

AUTRES MEMBRES DU JURY :

M. Mohamed JMAIEL	Professeur, université de Sfax
M. Omar BOUSSAÏD	Professeur, université Lumière Lyon 2

INVITÉ :

M. Gabriel FREY	Maître de conférences, université de Strasbourg

Université de Sfax
École Nationale d'Ingénieurs de Sfax
École doctorale des Sciences et de la Technologie

Université de Strasbourg
École doctorale n° 269 : Mathématiques, Sciences de l'Information et de l'Ingénieur

THÈSE

pour obtenir le grade de :

Docteur de l'université de Sfax et de l'université de Strasbourg

Discipline / spécialité :

Ingénierie des systèmes informatiques

présentée par :

Hajer AKID

Les bases de données graphes pour la science des données : Implantation d'entrepôts de données et prédiction d'interactions protéine-protéine

Soutenue le 12 décembre 2022 devant le jury composé de :

M. JMAIEL Mohamed	Professeur, Université de Sfax	Président
M. BOUSSAÏD Omar	Professeur, Université Lumière Lyon 2	Examinateur
M. FEKI Jamel	Professeur, Université de Sfax	Rapporteur
Mme SMAÏL-TABBONE Malika	Maître de conférences, Université de Lorraine	Rapporteur
M. BEN AYED Mounir	Professeur, Université de Sfax	Directeur de thèse
M. LACHICHE Nicolas	Maître de conférences, Université de Strasbourg	Directeur de thèse
M. Gabriel Frey	Maître de conférences, Université de Strasbourg	Co-encadrant

Dédicaces

À mon défunt père,
pour son amour inconditionnel, sa grande confiance et ses immenses sacrifices tout au long de mon parcours. Sans son encouragement, je ne serais pas là où je suis aujourd'hui.
Qu'il repose en paix.

À ma mère,
pour son amour inestimable, son soutien indéfectible et sa patience sans failles pendant toutes ces années d'études. Je l'aime et je lui suis éternellement reconnaissante.

À mon mari,
pour son grand amour qui illumine mon quotidien et ses encouragements durant les épreuves les plus difficiles. Sa patience, sa compréhension et son soutien sont les éléments clés de ma réussite.

À mes sœurs,
pour l'amour qu'elles me réservent et leurs encouragements inestimables au long de mes études.

À mes proches, mes amis et mes collègues,
pour leur confiance en moi et leur appui.

Qu'ils trouvent ici l'expression de ma profonde gratitude, mon grand amour et mon immense respect.

Remerciements

Je remercie toutes les personnes ayant participé à l'élaboration de ce travail, sans qui cette thèse n'aurait jamais vu le jour.

J'exprime tout d'abord mes plus respectueux remerciements et ma profonde gratitude à mes encadrants *M. Mounir BEN AYED*, professeur à l'université de Sfax, et *M. Nicolas LACHICHE*, maître de conférences à l'université de Strasbourg, pour leur temps précieux mis généreusement à ma disposition, leurs multiples conseils et leurs nombreuses remarques constructives qui ont largement contribué à l'amélioration des travaux de recherche présentés dans ce mémoire.

Outre mes directeurs de thèse, je tiens à remercier chaleureusement *M. Gabriel FREY*, maître de conférences à l'université de Strasbourg, pour ses judicieux conseils et ses fructueuses discussions qui ont grandement contribué à l'enrichissement de ce travail.

Mes sincères remerciements vont également à *Mme. Julie THOMPSON*, directrice de recherches CNRS à l'université de Strasbourg et *M. Kirsley CHENNEN*, Chercheur post-doctoral à l'université de Strasbourg, qui m'ont permis de bénéficier de leurs compétences et expertise dans le domaine de la bio-informatique.

Je remercie également *M. Mohamed Jmaiel*, professeur à l'université de Sfax, de m'avoir fait l'honneur de présider le jury de ma thèse.

Je remercie aussi vivement *Mme. Malika SMAÏL-TABBONE*, maître de conférences à l'université de Lorraine et *M. Jamel FEKI*, professeur à l'université de Sfax, qui m'ont

permis de bénéficier de leurs compétences en acceptant de rapporter et évaluer ma thèse.

Je tiens également à exprimer ma plus sincère reconnaissance à *M. Omar BOUSSAÏD* qui m'a honoré en acceptant d'examiner ce travail.

Enfin, je tiens à exprimer ma gratitude envers mes collègues des laboratoires ICube et REGIM-Lab pour leurs encouragements et leur soutien dans les moments difficiles.

Table des matières

Table des figures

Liste des tableaux

Acronymes

IPP	:	Interaction Protéine-Protéine
MDM	:	Modèle de Données Multidimensionnelles
MDM2G	:	Multidimensional Data Model to Graph
NoSQL	:	Système de Gestion de Base de Données Relationnelles
OLAP	:	OnLine Analytical Processing
OLTP	:	OnLine Transaction Processing
RDF	:	Resource Description Framework
R-OLAP	:	Relational OnLine Analytical Processing
STRING	:	Search Tool for the Retrieval of Interacting Genes
SGBD	:	Système de Gestion de Base de Données
SGBDR	:	Système de Gestion de Base de Données Relationnelles
TPC-DS	:	Transaction processing Performance Council for Decision Support

Chapitre 1

Introduction générale

Sommaire

1.1 Contexte et motivations

L'avènement des objets connectés et l'émergence des applications produisant des données volumineuses ont créé de nouvelles exigences qui défient considérablement les technologies traditionnelles de stockage, de gestion et d'analyse des données. C'est dans ce contexte qu'est apparue la discipline de la « science des données », qui combine différents domaines et utilise un large éventail de techniques pour extraire des connaissances à partir des données Vicario and Coleman (2020). Bien que le terme science des données ne soit pas nouveau, sa signification et les techniques associées ont évolué à la suite de l'émergence du « Big Data » , qui rend la partie « données » de plus en plus volumineuse, hétérogène, variée et difficilement exploitable. Dans ce contexte, le rôle de la science des données est devenu de plus en plus essentiel afin de comprendre les données massives générées par différentes sources, extraire les connaissances précieuses cachées dans les données et en tirer profit pour améliorer la prise de décision dans divers domaines tels que la bioinformatique, la santé et l'industrie.

Cette thèse s'inscrit pleinement dans la continuité des travaux de la communauté de la recherche scientifique sur l'exploitation efficace des données massives pour une meilleure prise de la décision. Nous nous intéressons particulièrement à l'utilisation des bases de données graphes pour résoudre des problématiques de science des données. En effet, les bases de données orientées graphes sont apparues depuis plusieurs décennies, mais leur popularité a progressivement diminué à cause de l'émergence d'autres types de bases de données telles que les bases de données hiérarchiques (XML) et spatiales Angles and Gutierrez (2008). Récemment, la nécessité d'analyser des données complexes a ravivé l'intérêt des chercheurs et des organisations pour l'utilisation des graphes. En effet, de nos jours, les données produites et collectées par les entreprises sont non seulement très volumineuses mais aussi très complexes Majidian and Raeesi Vanani (2019). Cette complexité se traduit par le fait que la valeur des données ne provient pas seulement des informations qu'elles contiennent directement et explicitement, mais aussi des informations implicites cachées dans les liens qui les relient Akid and Ayed (2016). Par exemple, dans un réseau social, l'analyse des noms et prénoms des utilisateurs est moins pertinente que l'analyse de leurs liens d'amitié qui permet de recommander de nouveaux amis ou produits. De même, dans le cas des réseaux d'Interaction Protéine-Protéine (IPP), l'analyse des noms des protéines apporte très peu de connaissances comparée à l'analyse de leurs liens qui permet d'identifier au mieux leurs fonctions biologiques et de développer de nouvelles stratégies thérapeutiques.

Classiquement, pour concevoir un système d'aide à la décision, les données à analyser sont souvent stockées dans des entrepôts de données relationnels. Ces entrepôts

utilisent généralement les bases de données relationnelles pour centraliser le stockage et offrir une vue globale des données circulant au sein de l'entreprise. En effet, ces entrepôts utilisent des bases de données relationnelles pour stocker les données de manière structurée et organisée en tables. Néanmoins, au cours de la dernière décennie, plusieurs recherches ont soulevé l'insuffisance des bases de données relationnelles dans le cas de la prise de la décision à partir des données massives Ajah and Nweke (2019); Golfarelli and Rizzi (2018). En effet, bien qu'elles soient matures et fondées sur des bases mathématiques solides, les bases de données relationnelles possèdent un schéma de données très rigide. Ainsi, l'adaptation de leurs schémas aux données non structurées est très difficile Corbellini et al. (2017); Jaiswal and Agrawal (2013). De plus, le stockage des données volumineuses dans les bases de données relationnelles coûte très cher en raison de leur évolutivité verticale Pokorny (2013), le partitionnement des données pouvant impacté fortement les performances. En ce qui concerne la capacité de gérer et d'analyser des données complexes, les Système de Gestion de Base de Données (SGBDR) utilisent des opérations de jointure, qui sont également très coûteuses en termes de temps de calcul Agrawal and Patel (2016). Par conséquent, pour répondre à ces nouveaux besoins, une nouvelle alternative de bases de données dites Not Only SQL (NoSQL) Mostajabi et al. (2021) est apparue. Les principaux avantages de ces bases de données sont la grande flexibilité de leurs schémas et la haute évolutivité de leurs modèles Davoudian et al. (2018). Les bases de données NoSQL sont généralement classées en quatre familles en fonction de leur modèle logique : les bases de données orientées clés-valeurs, les bases de données orientées colonnes, les bases de données orientées documents et les bases de données orientées graphes. Chaque famille de bases de données NoSQL est conçue pour des applications spécifiques. Dans cette thèse, notre intérêt se porte sur l'utilisation des bases de données orientées graphes, catégorie associée habituellement à des applications qui gèrent des données complexes.

Les bases de données orientées graphes sont conçues suivant un modèle orienté graphes qui s'appuie sur deux concepts : les noeuds et les arcs. Les noeuds modélisent les entités ou les objets tels que les utilisateurs d'un réseau social, les auteurs d'un réseau collaboratif ou les protéines d'un réseau d'IPP. En ce qui concerne les arcs, ils modélisent les liens entre les entités comme les liens d'amitié, de co-publications ou d'interactions. Les noeuds et les arcs possèdent des étiquettes qui indiquent leurs types (sémantiques). Dans le cas des graphes pondérés, les arcs portent des valeurs qui fournissent des informations supplémentaires sur les relations. Contrairement aux bases de données relationnelles qui reposent sur le mécanisme de jointure pour interroger les données complexes, les bases de données orientées graphes stockent physiquement les liens entre les données. Ainsi, les jointures classiques qui sont effectuées au moment de l'in-

terrogation à l'aide des clés étrangères, sont remplacées par des parcours (traversées) du graphe. En conséquence, les bases de données orientées graphes promettent des performances supérieures par rapport aux bases de données relationnelles, particulièrement lorsque les données sont très complexes et que plusieurs entités sont impliquées dans les requêtes analytiques.

1.2 Contributions

Dans ce travail, nous étudions l'utilisabilité des bases de données orientées graphes dans le domaine de la science des données suivant deux aspects qui constituent les contributions principales de cette thèse, à savoir : l'implantation d'entrepôts de données et la prédiction des valeurs manquantes sur les arcs d'un réseau d'IPP à l'aide des bases de données orientées graphes. Ces deux aspects sont actuellement des problématiques importantes en science des données.

1.2.1 Implantation d'entrepôts de données basée sur les graphes

La première contribution de cette thèse concerne la modélisation et l'implantation des entrepôts de données en utilisant des bases de données orientées graphes, dans le but de démontrer leur capacité à analyser des données complexes. Notre intérêt pour l'entrepôt de données découle de son importance au sein de l'architecture des systèmes d'information décisionnels. Un entrepôt de données permet de centraliser le stockage et l'historisation des données à vocation analytique, en vue de leur utilisation ultérieure dans la prise de décisions stratégiques Bentayeb (2011). La modélisation d'un entrepôt se fait souvent à l'aide d'un Modèle de Données Multidimensionnelles (MDM) qui fournit une représentation multidimensionnelle des données en se basant sur les concepts de fait et de dimension. Le fait est le sujet d'analyse et les dimensions sont les axes d'analyse. Classiquement, l'implantation d'un entrepôt de données se fait à l'aide d'une approche relationnelle d'analyse en ligne Relational OnLine Analytical Processing (R-OLAP). Cette approche consiste à traduire les concepts de fait et de dimension en tables relationnelles et à utiliser des requêtes SQL pour les interroger. Par conséquent, les requêtes analytiques complexes nécessitent souvent plusieurs jointures entre la table de fait et les tables de dimension interrogées. Ces jointures ralentissent considérablement le temps de réponse. De plus, le stockage des données volumineuses sous la forme de bases de données relationnelles dans ces entrepôts est coûteux en raison du besoin de scalabilité verticale. Pour pallier cet inconvénient, plusieurs approches récentes ont proposé l'utilisation des bases de données NoSQL orientées colonnes ou documents pour l'im-

plantation des entrepôts de données Chevalier et al. (2015a); Boussahoua et al. (2017); Dehdouh et al. (2020). Bien qu'elles permettent l'entreposage des données très volumineuses, ces approches présentent un inconvénient majeur puisqu'il faut adapter leurs modèles pour répondre à des requêtes analytiques complexes. Quant aux bases de données graphes, leur efficacité lors de l'entreposage de données demeure relativement peu étudiée Castelltort and Laurent (2014); Sellami et al. (2020). Dans cette thèse, nous proposons l'utilisation de systèmes NoSQL orientés graphes pour répondre aux questions complexes exprimées par les décideurs. Pour ce faire, nous introduisons un nouveau formalisme intitulé Multidimensional Data Model to Graph (MDM2G) qui regroupe plusieurs règles de passage traduisant chaque concept du MDM en un concept équivalent dans les graphes. Les deux schémas multidimensionnels classiques d'un entrepôt de données, à savoir le schéma en étoile et celui en flocon de neige sont pris en considération dans nos règles. Ces règles permettent donc de définir deux variantes d'un entrepôt de données NoSQL orienté graphes, en étoile et en flocon de neige. Dans le cas des approches relationnelles, le recours à un schéma en flocon de neige vise à réduire l'espace disque occupé, mais entraîne une augmentation du temps de réponse. Pour identifier le schéma le plus performant dans le cadre de notre approche basée sur les graphes, nous implantons et comparons la performance de nos deux entrepôts de données quand les requêtes deviennent plus complexes et quand le volume de données interrogées augmente (comparaison inter-modèles). En outre, nous comparons les deux entrepôts de données orientés graphes que nous avons proposés à des entrepôts de données similaires implantés en se basant sur l'approche relationnelle classique R-OLAP. Cette deuxième comparaison a pour objectif de déterminer l'entrepôt de données le plus adapté au stockage des données volumineuses et complexes (comparaison intra-modèles). Pour ces deux comparaisons, nous utilisons les données du banc d'essais décisionnel reconnu Transaction processing Performance Council for Decision Support (TPC-DS) Nambiar and Poess (2006) pour générer plusieurs volumes de données et plusieurs catégories de requêtes de complexité différente. Comme critères de comparaison, nous nous appuyons sur deux métriques qui sont le temps de chargement et le temps d'interrogation de données.

1.2.2 Prédiction de valeurs manquantes sur les arcs des réseaux d'IPP

La deuxième contribution de cette thèse porte sur l'utilisation des bases de données orientées graphes pour la prédiction des valeurs manquantes sur les arcs d'un graphe pondéré. Nous nous intéressons tout particulièrement aux réseaux d'IPP pondérés qui peuvent être modélisés intuitivement à l'aide des graphes. Dans ce cas, les noeuds sont

les protéines, les arcs représentent les interactions entre les protéines et les valeurs portées par les arcs indiquent la probabilité d'existence d'une similarité fonctionnelle entre ces protéines. La détection d'IPP encore non évaluées dans les laboratoires en utilisant les techniques expérimentales est très lente et coûteuse Kovács et al. (2019). Par conséquent, le recours à la science des données pour réduire l'ensemble des paires de protéines à étudier expérimentalement est d'un grand intérêt. La problématique principale revient à chercher les informations les plus pertinentes pour représenter les caractéristiques du graphe et identifier les plus fortes similarités parmi les couples de protéines non-encore évalués. En fait, dans la littérature, la majorité des approches de prédiction proposées permettent de prédire uniquement l'existence ou non d'un lien manquant sans tenir compte des valeurs qui peuvent être portées par les arcs. En d'autres termes, ces approches utilisent les informations topologiques d'existence ou non d'un chemin intermédiaire entre les paires de nœuds non directement liés pour calculer une mesure de similarité pour ces paires comme le nombre de voisins en commun Barzel and Barabási (2013), l'indice d'Adamic Adar Adamic and Adar (2003) et l'indice d'allocation de ressource Liu and Lü (2010). Cette mesure est ensuite triée par ordre décroissant en vue d'identifier et proposer une liste de liens potentiels. Malheureusement, certaines mesures de similarité topologiques largement utilisées dans d'autres domaines comme les réseaux sociaux, échouent lors de la recherche de nouvelles interactions potentielles dans les réseaux d'IPP Kovács et al. (2019). Cet échec est dû au fait que les informations liées à la topologie ne parviennent pas à capturer les forces structurelles et évolutives qui régissent les réseaux d'IPP. Pour pallier cet inconvénient, nous proposons dans ce travail une nouvelle approche qui tient compte non seulement de la topologie du graphe mais aussi des valeurs portées par les arcs intermédiaires entre les paires de noeuds non directement liées. Pour ce faire, nous proposons de nouvelles représentations par agrégation qui agrègent les valeurs portées par les chemins intermédiaires. De plus, pour profiter pleinement des mesures topologiques classiques et de nos mesures basées sur l'agrégation, nous proposons une approche originale d'apprentissage automatique qui combine ces deux types de mesures afin de prédire les liens manquants et leurs valeurs. Notre algorithme prend en entrée les différentes mesures afin de prédire les interactions pertinentes à suggérer aux biologistes et à vérifier expérimentalement. Afin de valider notre proposition, nous commençons par implanter et évaluer nos représentations par agrégation. Ensuite, nous testons l'efficacité de notre approche d'apprentissage automatique basée sur l'agrégation en utilisant les données de l'organisme levure (S. Cerevisiae) du banc d'essai reconnu Search Tool for the Retrieval of Interacting Genes (STRING) Szklarczyk et al. (2021). Enfin, nous comparons leurs performances aux approches classiques en utilisant le même jeu de données du point de vue de différentes métriques :

le temps de calcul, la précision, l'erreur absolue moyenne, l'erreur relative moyenne et l'erreur quadratique moyenne.

1.3 Organisation du mémoire

Ce manuscrit est organisé comme suit :

Dans le deuxième chapitre, nous présentons les motivations de l'utilisation des bases de données orientées graphes pour la gestion et l'analyse des données complexes en science des données. Ainsi, nous commençons par introduire les bases de données relationnelles et montrer leurs limitations dans le contexte des données complexes. Ensuite, nous présentons le mouvement NoSQL et les différents modèles logiques non-relationnels pour montrer l'intérêt du modèle orienté graphes quand les données sont fortement connectées. Après, nous passons en revue les différentes études comparatives menées pour évaluer la performance des bases de données orientées graphes et les comparer aux autres types de bases de données en vue de justifier l'efficacité de leur utilisation dans une solution basée sur les graphes en science des données. Enfin, un état de l'art sur les approches et les méthodologies de migration vers des bases de données orientées graphes est exposé dans ce chapitre.

Le troisième chapitre concerne la première contribution de cette thèse qui porte sur l'utilisation de bases de données orientées graphes pour l'implantation des entrepôts de données. Ainsi, nous introduisons les concepts clés des entrepôts et nous décrivons les différentes approches proposées dans la littérature en vue de transformer le modèle conceptuel multidimensionnel d'un entrepôt à l'un des modèles logiques des bases de données NoSQL. Nous regroupons ces approches en trois catégories selon le modèle logique NoSQL cible choisi. Ensuite, nous introduisons notre formalisme MDM2G permettant la mise en œuvre d'un entrepôt de données NoSQL orienté graphes. Nous décrivons nos règles formelles de transformations MDM2G. Après, nous détaillons l'étude comparative menée pour évaluer la performance de nos entrepôts de données NoSQL orientés graphes et la comparer aux approches relationnelles classiques en se basant sur les données du banc d'essai TPC-DS dédié à l'évaluation des systèmes décisionnels. Enfin, nous concluons par la détermination de l'entrepôt de données le plus adapté aux requêtes analytiques complexes.

Le quatrième chapitre porte sur la deuxième contribution de cette thèse dans laquelle nous proposons d'utiliser les systèmes NoSQL orientés graphes pour la prédiction de valeurs dans le contexte des réseaux d'IPPpondérés. Un état de l'art sur les approches classiques de prédiction de liens est dressé dans ce chapitre. Ces approches peuvent être regroupées en trois familles : approches topologiques, approches probabilistes et

approches basées sur l'apprentissage. Ensuite, nous détaillons nos représentations basées sur l'apprentissage et notre approche d'apprentissage automatique basée sur l'agrégation. Après, nous présentons les résultats obtenus sur les données du banc d'essai STRING. Enfin, nous exposons les apports de notre modèle par rapport aux approches traditionnelles.

Dans le cinquième chapitre, nous clôturons ce manuscrit par une conclusion générale dans laquelle nous dressons le bilan de nos contributions et nous présentons l'ensemble de nos perspectives.

1.4 Liste des publications

Les différents travaux de recherche décrits dans cette thèse ont fait l'objet des publications suivantes :

— Akid H. and Ben Ayed M., « Towards nosql graph data warehouse for big social data analysis », 16th International Conference on Intelligent Systems Design and Applications (ISDA), Porto, Portugal, 14-12 December, 2016. Springer, Cham., pp. 965-973.

— Akid H., Ben Ayed M., G. Frey, and Lachiche N., « Entrepôts de données NoSQL orientés graphes », 35ème Conférence sur la Gestion de Données Principes Technologies et Applications (BDA), Lyon, France, 15-18 Octobre, 2019. Hal-03176580 pp.51-52.

— Akid H., Frey G., Ben Ayed M., and Lachiche N., « Performance of NoSQL graph implementations of star vs. snowflake schemas », IEEE acess. ISSN : 21693536, 10.1109/ACCESS.2022.3171256.

— Akid H., Chennen K, Frey G., Thompson J., Ben Ayed M., and Lachiche N., « Apprentissage automatique basé sur l'agrégation pour la prédiction de liens dans les réseaux d'interaction protéine-protéine », 22ème Conférence sur l'Extraction et la Gestion des connaissances (EGC), Blois, France, 24-28 Janvier, 2022, vol. RNTI-E-38, pp.481-482.

Chapitre 2

Les bases de données au service de la science des données

Sommaire

2.1 Introduction

Les graphes sont actuellement au coeur de plusieurs types d'applications en sciences des données. Qu'il s'agisse de réseaux d'interactions protéine-protéine, de réseaux de documents, de réseaux sociaux ou de réseaux routiers, les données de ces applications peuvent être modélisées intuitivement par le biais des graphes. Trouver des informations pertinentes et extraire des connaissances intéressantes à partir de ces données fortement connectées se fait généralement par des requêtes très complexes. L'utilisation des bases de données relationnelles traditionnelles pour répondre à ces requêtes, consiste souvent à faire de multiples jointures pour lier les différentes tables. Étant donné le coût élevé de l'opération de jointure dans les bases de données relationnelles, de nombreux travaux de recherche ont fait appel aux bases de données orientées graphes comme un outil simple et intuitif pour le stockage et la gestion des liens entre les données. Par conséquent, ces dernières années, l'utilisation des bases de données graphes en science des données a envahi plusieurs domaines tels que la biologie, la chimie, et la sociologie.

L'objectif de ce chapitre est de présenter le contexte et les motivations d'utilisation des bases de données graphes au service de la science des données. Ainsi, nous introduisons dans la section 2.2 les concepts clés des bases de données relationnelles prédominantes et nous détaillons leurs limitations qui exigeaient la recherche de nouvelles solutiofns pour la gestion des données massives et complexes. Nous présentons ensuite dans la section 2.3 les principes des bases de données NoSQL qui ont émergé en alternative aux bases de données relationnelles. La section 2.4 est dédiée plus spécifiquement à l'introduction des bases de données orientées graphes qui sont au centre de nos recherches. Un résumé de la littérature concernant les approches existantes pour la migration vers les bases de données graphes est présenté dans la section 2.5. Dans la section 2.6, nous présentons les travaux qui ont porté sur l'utilisation des graphes en science des données. Enfin, dans la section 2.7 nous faisons le bilan de ce chapitre.

2.2 Modèle relationnel

Parmi les nombreuses catégories de bases de données existantes, les bases de données relationnelles sont les plus dominantes depuis les années 80. Les bases de données relationnelles sont basées sur le modèle relationnel qui a été introduit par E.F. Codd dans les années 70 Codd (2002). Dans son papier, E.F. Codd a défini les fondements de ce modèle. En particulier, le concept de relation qui est utilisé ici dans son sens mathématique correspond dans ce modèle aux tables. Les n-uplets (ou tuples) de la relation sont les lignes (ou enregistrements) de la table. Chaque ligne est identifiée d'une manière

unique à l'aide de sa clé primaire. Les colonnes de la table représentent les attributs des données. Ainsi, chaque tuple contient une valeur pour chaque attribut.

2.2.1 Opération de jointure

Pour combiner les tuples de deux relations ou plus dans une base de données relationnelle, une opération de jointure est requise. Cette opération doit respecter une ou plusieurs conditions de jointure. L'utilisation des opérations de jointure suit en général la définition des clés étrangères, ce qui impose une surcharge importante à la base de données Aboutorabi[a] et al. (2015). Par exemple, pour relier la table $Store\_Sales$ aux tables $Date\_Dim$ et $Customer$, deux clés étrangères vers les clés primaires de ces deux tables ont été ajoutées à la table $Store\_Sales$ comme illustré dans la Figure 2.1. Dans le cadre des bases de données relationnelles, les liens qui relient les tables sont des liens logiques qui se font à travers la vérification des valeurs des attributs communs. Ainsi, ces liens ne sont pas stockés dans la base et sont vérifiés uniquement au moment de l'interrogation des données. Dans le cas de la requête présentée à la Figure 2.1, la jointure de la table $Store\_Sales$ à la table $Date\_Dim$ consiste à vérifier si l'achat a été effectué en 2002. En ce qui concerne la jointure de la table $Store\_Sales$ à la table $Customer$, elle permet de récupérer le prénom du client.

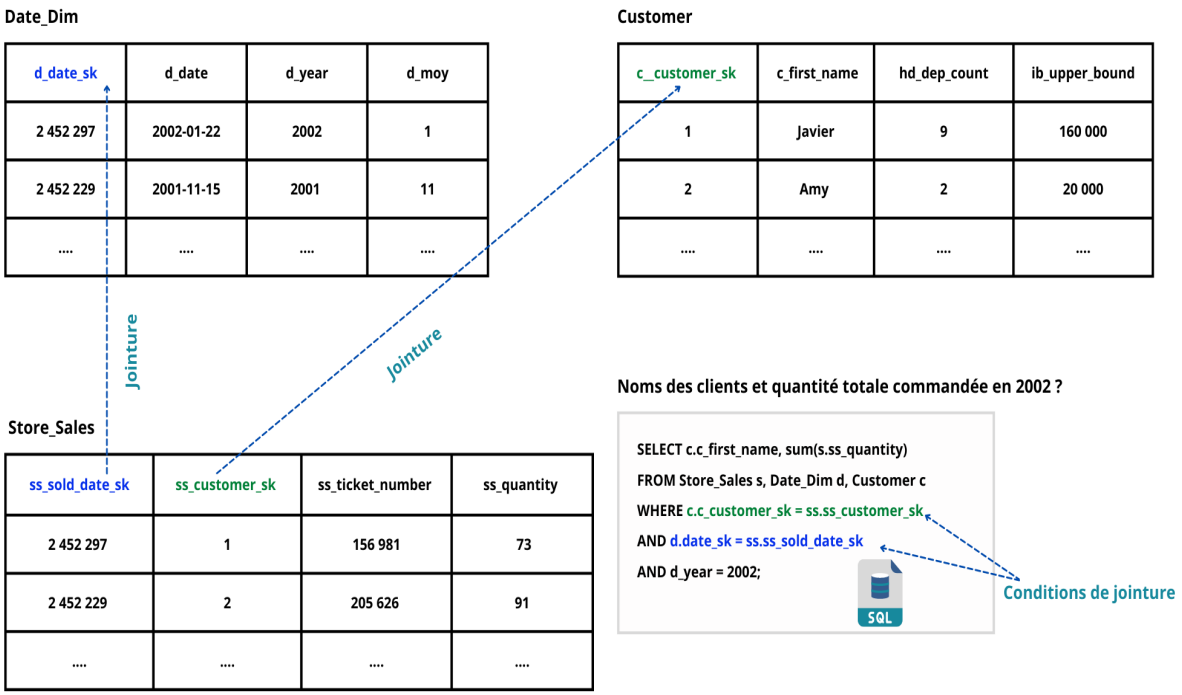


FIGURE 2.1 – Gestion des données complexes à l'aide des bases de données relationnelles

Dans la mise en oeuvre la plus intuitive de l'opération de jointure, chaque tuple dans une relation doit être comparé à chaque tuple dans l'autre relation Mishra and Eich (1992) . Par conséquent, les requêtes complexes qui interrogent simultanément plusieurs tables ayant un nombre important de lignes prennent beaucoup de temps pour répondre aux requêtes des utilisateurs. Pour remédier à cet inconvénient, plusieurs travaux de recherche ont visé l'optimisation des algorithmes de jointure pour réduire le temps de réponse Halstead et al. (2013); Schuh et al. (2016). En particulier, certains travaux Valduriez (1987); Mishra and Eich (1992) ont mis en avant l'efficacité de l'utilisation des index lors de la récupération des données à partir de plusieurs tables. Les index sont créés souvent automatiquement sur les clés primaires afin de retrouver plus rapidement la ligne correspondante à la valeur d'une clé étrangère. Malgré les efforts d'optimisation des algorithmes de jointure, la performance des bases de données relationnelles lors du stockage des données complexes diminue lorsque le nombre de relation interrogées augmente Almabdy (2018). De plus, la capacité des bases de données relationnelles à stocker des données volumineuses est limitée à cause des propriétés ACID. Ces propriétés constituent un atout majeur pour les systèmes de traitement des transactions en ligne (Online Transaction Processing, OnLine Transaction Processing (OLTP). Toutefois, ces mêmes propriétés représentent une limitation fondamentale lors de la distribution des données sur plusieurs serveurs.

2.2.2 Propriétés ACID

L'une des caractéristiques principales associée aux bases de données relationnelles est la nécessité de satisfaire les quatre propriétés Atomicité, Cohérence, Isolement et Durabilité (ACID) suivantes pour garantir le traitement des transactions d'une façon fiable Kanwar et al. (2013) :

— Atomicité : Les mises à jour de la base de données doivent être réalisées totalement ou pas du tout. Dans le cas où une partie de la transaction échoue, les données doivent être remises dans l'état initial.

— Cohérence : Chaque transaction doit préserver la cohérence des données tout en respectant les contraintes d'intégrité de la base. En d'autres termes, une transaction valide doit nécessairement mener la base de données d'un état cohérent à un autre état cohérent.

— Isolation : Les transactions simultanées se comportent comme des unités isolées dont les phases intermédiaires ne sont pas visibles par les autres transactions.

— Durabilité : Une fois qu'une transaction a été validée, les changements qu'elle a apportés persistent même en cas de défaillance du système.

Le maintien des différentes propriétés ACID est primordial dans le cadre des bases de données relationnelles. Généralement, ces bases stockent les données volumineuses en appliquant le principe de la scalabilité verticale qui consiste à améliorer les capacités d'un système existant en lui ajoutant plus de ressources. Il est par exemple possible d'augmenter la taille de la mémoire ou d'ajouter des processeurs. Dans ce cas, les données sont souvent centralisées sur le même serveur ce qui rend relativement facile la satisfaction des quatre propriétés. Malheureusement, dans le contexte du Big Data, la scalabilité verticale est une solution très coûteuse à cause de la quantité énorme des données à stocker. Il est donc plus intéressant de suivre le principe de la scalabilité horizontale qui consiste à rajouter au besoin un ou plusieurs serveurs à la base afin de ne pas avoir un seul point de contention (en anglais single point of failure). Les données dans ce cas sont divisées sur plusieurs machines et certaines données sont copiées sur différents serveurs pour accélérer l'accès, ou renforcer la robustesse si un serveur est défaillant. Ainsi, pour maintenir les propriétés ACID, les différentes copies doivent être cohérentes. De ce fait, des opérations de mises à jour doivent être effectuée régulièrement. Le coût de ces opérations de mises à jour s'ajoute à celui des opérations d'insertion, modification et suppression puisqu'une transaction ne peut être validée que si elle a été effectuée sur tous les serveurs. Dans ce cas, le système fait patienter l'utilisateur durant ce temps ce qui influe considérablement sur le temps de réponse dès l'augmentation du volume de données stockées dans la base.

2.2.3 Insuffisance des bases de données relationnelles

Il est indéniable que les bases de données relationnelles ont réussi à s'imposer comme un outil efficace pour la gestion des données structurées durant plusieurs décennies. Plusieurs cas d'utilisation favorisent le recours aux bases relationnelles comme la gestion des stocks, ventes, comptes bancaires, etc. Étant donné la maturité du modèle relationnel et du langage SQL, le choix des SGBDR demeure intéressant pour ces types d'applications. En revanche, avec l'avènement des données massives et l'émergence des applications qui nécessitent le stockage des données de grandes tailles et de haute complexité comme Facebook, LinkedIn et Google, les caractéristiques des données collectées sont devenues peu adaptées aux bases relationnelles. Le besoin de répartir ces données dans des nouvelles structures de stockage plus adaptées à la distribution est primordial. En plus du besoin de scalabilité horizontale, il est nécessaire de choisir une base de données dont le schéma est flexible. En fait, les données du Big Data sont souvent variées et contiennent des informations manquantes. Dans ce contexte, il est extrêmement difficile d'utiliser les bases de données relationnelles ayant un schéma statique ce qui exige de

définir a priori l'ensemble des champs de la base. De plus, le mécanisme de jointure entre les tables est nécessaire pour construire des requêtes complexes. Toutefois, cela engendre un coût considérable surtout quand le volume de données devient gigantesque. Imaginons une table contenant tous les internautes qui ont acheté des livres à partir d'un site web et une autre stockant l'ensemble des bouquins disponibles dans ce site. Si on souhaite trouver tous les livres achetés par un internaute particulier à une date précise, il faudra effectuer le produit cartésien de ces deux tables qui peuvent contenir des millions d'enregistrements puis filtrer le résultat. L'espace mémoire et le temps requis pour exécuter cette simple requête sont énormes et révèlent l'insuffisance du modèle relationnel.

Toutes les raisons précédemment mentionnées ont poussé plusieurs entreprises, notamment les grands acteurs du web, à chercher des alternatives pour stocker et gérer les données volumineuses, variées et complexes. C'est dans ce contexte que les systèmes de gestion de bases de données NoSQL ont vu le jour. Cette nouvelle génération de bases de données est certes moins mature que les bases de données relationnelles mais ses caractéristiques sont plus prometteuses pour les applications qui doivent collecter, stocker et analyser des données massives. Nous détaillons les principes de ces systèmes, leurs catégories et leurs capacités à gérer les données complexes dans la section suivante.

2.3 Spécificités des bases de données NoSQL

Le NoSQL est une stratégie de stockage relativement nouvelle permettant de stocker et manipuler des données massives. Cette catégorie a émergé en 2009 et a occupé une place importante dans le monde académique et dans les entreprises. Le besoin fondamental auquel répond ce nouveau paradigme est d'offrir une solution qui s'adapte aux données volumineuses et hétérogènes. En effet, afin de résoudre les problèmes liés à l'explosion du volume de données, les grands acteurs du web ont procédé à des compromis sur les propriétés ACID des SGBD traditionnels. Ces compromis sur la notion du relationnel ont permis de libérer les systèmes de gestion de bases de données relationnels de leurs freins à la scalabilité. Toutefois, le NoSQL n'a pas pour vocation de remplacer le modèle relationnel mais plutôt de combler ses faiblesses dans des cas d'utilisation bien déterminés.

Le terme NoSQL désigne une nouvelle génération de bases de données dont les objectifs, les caractéristiques, et les modèles sont différents des bases relationnelles classiques. Dans le cas du NoSQL, l'unité logique n'est plus la table, et les données ne sont en général pas manipulées avec SQL Chevalier et al. (2015a). Ces systèmes partagent plusieurs caractéristiques communes telles que :

— L'absence des schémas stricts : Le schéma et les types des données stockées peuvent

être changés à tout moment sans influencer l'application. Les données ne possèdent pas forcément le même schéma. Certains champs peuvent être omis sans avoir le besoin de les stocker avec des valeurs nulles.

— Partitionnement horizontal : Pour stocker des données massives, les bases de données NoSQL utilisent la technique du partitionnement horizontal qui consiste à diviser les données en plusieurs partitions horizontales, chacune étant stockée sur un ou plusieurs noeuds (serveurs) d'un cluster.

— Absence de notion ACID : Les propriétés ACID des bases relationnelles sont remplacées dans les bases de données NoSQL par d'autres propriétés nommés BASE (Basically Available, Soft state, Eventually consistent). Ces propriétés BASE indiquent que toutes les mises à jour effectuées sur un système distribué NoSQL ne seront cohérentes que dans les périodes sans mises à jour. En effet, il est impossible de conserver réellement un système dans un état cohérent en permanence, ce qui aurait pour effet d'empêcher la disponibilité des données. Les systèmes NoSQL favorisent donc la disponibilité au lieu de la cohérence afin de répondre aux requêtes des utilisateurs.

2.3.1 Modèles non-relationnels

Les bases de données NoSQL visent à satisfaire les nouvelles exigences des applications actuelles dont les données sont très volumineuses et / ou complexes. Cela est possible grâce aux caractéristiques précédemment citées, notamment la haute disponibilité des données, la tolérance aux pannes et la grande flexibilité du schéma de la base de données. Pour tirer profit pleinement des bases de données NoSQL et de leurs caractéristiques, plusieurs modèles ont été définis en vue de répondre à plusieurs types d'applications qui gèrent des données de volumes et complexités différentes. Dans l'analyse des flux d'événements par exemple, il est plus intéressant de privilégier la taille de la base que la complexité. En revanche, dans le cas des applications biologiques, les données sont très connectées ce qui favorise le choix d'une base capable de modéliser et stocker les liens. Des modèles de bases NoSQL dédiés à des applications de tailles et complexités différentes ont donc vu le jour.

Les bases de données NoSQL sont généralement classées en quatre catégories principales : orientée clé-valeur, orientée colonnes, orientée documents, et orientée graphes. Le modèle orienté clé-valeur est le plus simple et permet ainsi de stocker le plus de données. Chaque valeur est associée dans ce modèle à une clé qui permet de la retrouver. La simplicité de ce modèle lui permet de stocker facilement de gros volumes de données variées. Toutefois, pour chercher des informations à partir de ces données, de

grands efforts sont requis. Les modèles orientés colonnes et orientés documents offrent également la possibilité de stocker des données volumineuses et non structurées respectivement dans des familles de colonnes qui regroupent un ensemble de colonnes et dans des collections de documents qui regroupent un ensemble de documents. Quant au modèle orienté graphes basé sur les concepts de noeud et d'arc, la différence principale par rapport aux autres modèles de bases de données NoSQL est le stockage physique de liens entre les données.

Plusieurs travaux de recherche ont comparé ces catégories entre elles et aux bases de données relationnelles sur la base des caractéristiques fonctionnelles (comme la dénormalisation et la jointure) et non fonctionnelles (comme la flexibilité et la complexité) Nayak et al. (2013); Gupta et al. (2017); Sahatqija et al. (2018). De point de vue gestion de données complexes, les bases de données NoSQL orientées clé-valeur sont les moins adaptées à modéliser et stocker les données interconnectées. En effet, la simplicité de ce modèle qui constitue son grand atout pour certaines applications, limite sa capacité à gérer les liens complexes comme illustré à la Figure 2.2. Le modèle orienté colonnes et le modèle orienté documents ne sont guère plus destinés au stockage de données complexes. Cependant, dans la littérature, des approches ont été proposées pour adapter ces modèles à la représentation et au stockage des données connectées. En particulier, dans Li (2010); Chevalier et al. (2015a); Dehdouh et al. (2015), les auteurs ont proposé de stocker les données corrélées dans la même famille de colonnes. D'autres travaux ont suggéré l'utilisation de la notion de documents imbriqués pour une représentation hiérarchique des données connectées Hanine et al. (2015); Hamouda and Zainol (2017); Singh (2019). Bien qu'il soit possible grâce aux approches précédemment citées de gérer les données complexes en utilisant les modèles orientés colonnes et documents, l'étude de la performance de ces approches montre qu'elles sont d'une part coûteuses en termes de temps de calcul Lee and Zheng (2015); Boussahoua et al. (2017) et d'autre part limitées en termes de profondeur des liens entre les données Gómez et al. (2016) comme illustré à la Figure 2.2 Chandra (2015).

La difficulté de gérer les données complexes d'une part en utilisant les bases de données relationnelles et d'autre part en se servant des bases de données NoSQL orientées clé-valeur, colonnes et documents ont largement contribué à l'augmentation de la popularité des bases de données NoSQL orientées graphes. Ces dernières sont principalement dédiées au stockage non seulement des données mais des liens entre elles permettant ainsi plus d'efficacité lors de l'interrogation des données complexes. Un autre avantage des bases de données graphes est qu'elles sont dotées d'une interface graphique naturelle et adaptée à une visualisation, voire une exploration interactive des liens entre les données.

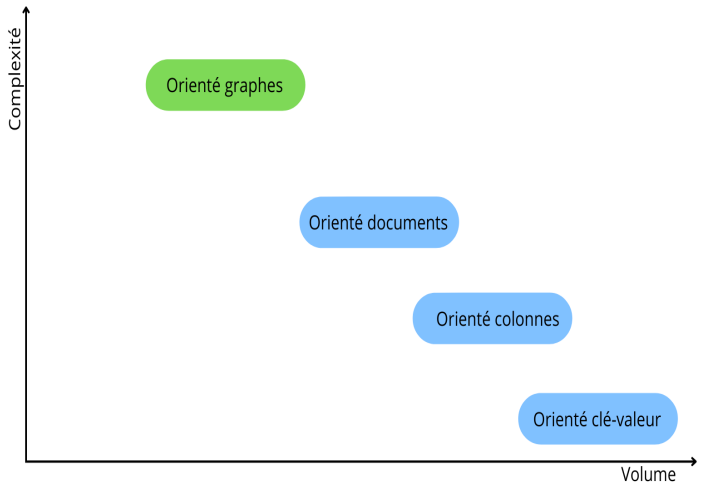


FIGURE 2.2 – Gestion des données complexes à l'aide des bases de données NoSQL

2.3.2 Théorème CAP

Contrairement aux bases de données relationnelles qui reposent sur les propriétés ACID pour garantir la cohérence et l'intégrité des données, les bases de données NoSQL se focalisent sur des propriétés facilitant la distribution des données. Ces propriétés sont définies dans le théorème Consistency, Availability and Partition tolerance (CAP), également connu sous le nom de théorème de Brewer. Le théorème de CAP affirme qu'il est impossible d'obtenir simultanément trois propriétés dans un système distribué. Par conséquent, seules deux des trois propriétés Consistency, Availability and Partition tolerance peuvent être satisfaites en même temps : AC, AP ou CP. Les bases de données relationnelles implémentent les propriétés de Cohérence (Consistency) et de Disponibilité (Availability), ce qui correspond au système AC. Cependant, pour les grands acteurs du web tels que Google, Facebook et Amazon, la disponibilité et la tolérance au partitionnement sont plus importantes que la cohérence des données. Par exemple, il est moins préoccupant que deux internautes obtiennent des résultats de recherche différents sur Google que de ne pas obtenir de résultat du tout. Les trois propriétés sont définies par le théorème CAP de la manièrte suivante :

— Cohérence (Consistency) : Tous les noeuds d'un système distribué voient exactement les mêmes données au même moment. Cela signifie que lorsque les données sont mises à jour dans un serveur du cluster, ces mêmes données sont également mises à jour dans tous les autres serveurs.

— Disponibilité (Availability) : Les données doivent être disponibles à tout moment.

Ainsi, les utilisateurs peuvent accéder aux données et effectuer des opérations à tout moment, sans interruption de service.

— Tolérance au partitionnement (Partition tolerance) : Pour garantir la disponibilité des données à tout moment, le système doit être capable de continuer à fonctionner correctement même dans le cas ou certains des noeuds sont en pannes.

2.3.3 Avantages des bases de données NoSQL

Le NoSQL présente certains avantages par rapport aux SGBD classiques, à savoir :

— Flexibilité du schéma : Chaque enregistrement de la base NoSQL est libre de suivre sa propre structure. On peut ajouter ou parfois retirer des champs sans utiliser des valeurs « NULL ».

— Scalabilité horizontale : Les bases NoSQL sont basés sur le principe de scalabilité horizontale qui consiste à ajouter de nouveaux noeuds dans le cluster quand le volume des données augmente. L'équilibre du volume de donnée entre les différents noeuds est assuré souvent automatiquement.

— Tolérance aux pannes : Les bases NoSQL sont conçues pour éliminer le risque de panne. Ainsi, si un noeud tombe, un autre prend la charge et par conséquent on peut accéder à la donnée sur un autre noeud vu que les données sont répliquées.

Les bases de données NoSQL sont une option prometteuse pour stocker de grandes quantités de données, mais elles doivent faire face à plusieurs défis importants. . En particulier, contrairement aux bases de données relationnelles pour lesquelles le langage d'interrogation SQL est mature et standardisé pour tous les systèmes SGBDR, les langages d'interrogation des bases de données NoSQL sont encore peu matures et variables d'un système NoSQL à l'autre. Par conséquent, lorsqu'on migre vers un système NoSQL, il est nécessaire de prévoir un temps d'adaptation et de faire plus d'investigations au niveau des requêtes. En outre, de nombreuses approches traditionnelles s'appuient sur les bases de données relationnelles, telles que les approches d'implantation d'entrepôts de données et les approches classiques de fouille de données. Par conséquent, l'utilisation de ces approches dans le contexte des systèmes NoSQL nécessite leur adaptation à ce nouvel environnement.

2.4 Bases de données graphes

Au cours des dernières années, il y a eu un regain d'intérêt pour le stockage et la gestion des données dans des bases de données orientées graphes. Ces dernières sont basées

sur la théorie des graphes introduite par Leonard Euler. Les concepts fondamentaux de ce modèle de base de données sont les noeuds (sommets du graphe) et les arcs (arêtes du graphe). Les arcs permettent de lier les données pour faciliter la gestion des données complexes en remplaçant les jointures par des traversées du graphe comme illustré à la Figure 2.3.

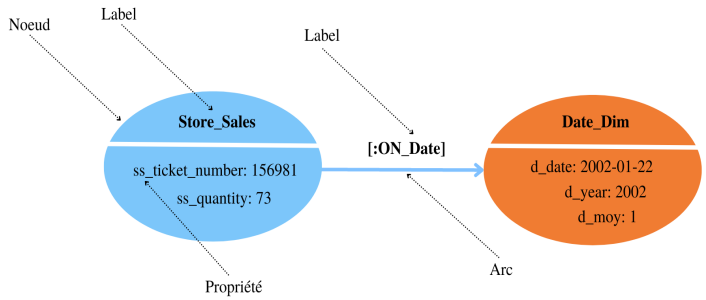


FIGURE 2.3 – Gestion des données complexes à l'aide des bases de données orientées graphes

Ce type de base de données était populaire au début des années 1990 mais s'est éteint avec l'apparition de XML et de l'internet Buerli and Obispo (2012). Grâce au mouvement NoSQL, les bases de données orientées graphes ont attiré de nouveau l'attention des chercheurs et des entreprises pour pallier les inconvénients des bases de données relationnelles lors de la gestion des données complexes dans plusieurs domaines. Dans Angles et al. (2013), une étude a été menée pour tester la performance des bases de données lors de l'interrogation des réseaux sociaux dont les données sont très connectées. Les résultats de ce travail ont montré la capacité des graphes à répondre aux requêtes complexes dans un temps raisonnable et l'échec des bases de données relationnelles quand la profondeur des liens dépasse quatre. Dans Fabregat et al. (2018), une base de données graphe a été proposée pour le stockage et l'interrogation rapide des données relatives aux voies biologiques qui contiennent de nombreuses tables de sorte que les requêtes nécessitent généralement des opérations de traversée entraînant une dégradation de performances et de temps de réponse longs. Les travaux Yoon et al. (2017); Lysenko et al. (2016); Mullen et al. (2016) ont utilisé une base de données graphes pour le stockage des liens entre les entités biologiques et ont montré que le choix de cette base permet aux experts une meilleure compréhension des données et une rapidité d'accès.

Le Resource Description Framework (RDF) et le modèle des graphes à propriétés sont les deux modèles les plus connus pour la représentation des données sous la forme d'un graphe. Bien qu'elles soient basées sur les notions du graphe, ces deux représentations sont différentes en termes de modèle de données, de schéma, de langage de requête,

de sens et de contenu.

2.4.1 Resource Description Framework (RDF)

Le modèle RDF est un standard de modélisation introduit par le World Wide Web Consortium (W3C) et qui est fortement utilisé dans le domaine du web sémantique. Ce modèle représente les données sous la forme d'un graphe orienté et étiqueté. Le graphe RDF est fondé sur trois catégories de noeuds : des ressources nommées, des ressources anonymes et des littéraux Lasolle et al. (2021). Une ressource nommée possède un identifiant nommé Internationalized Resource Identifier (IRI) et vise à décrire une catégorie de noeud (organisme, protéine, médicament, etc). Elle contient des propriétés et peut avoir des instances. Les ressources anonymes possèdent un nom mais qui n'a pas de valeur (noeud vide). Quant au littéral, il s'agit d'une valeur constante de type prédéfini. Les noeuds du modèle RDF peuvent être reliés à l'aide des liens qui se caractérisent par le triplet (sujet, prédicat, objet). Le sujet correspond à la ressource décrite par le prédicat et à laquelle une valeur peut être associée (objet). Pour interroger les données stockées suivant un modèle RDF, le W3C a encouragé l'utilisation du langage SPARQL. Malgré la grande utilisation de RDF dans le web sémantique, notamment lors de la définition des ontologies, le modèle le plus utilisé demeure le modèle des graphes à propriétés Robinson et al. (2015).

2.4.2 Modèle des graphes à propriétés

La plupart des systèmes de base de données de graphes actuels ont été conçus pour prendre en charge le modèle de données des graphes à propriétés. Un graphe à propriétés désigne un graphe dirigé étiqueté avec la caractéristique spéciale que chaque noeud ou arête peut maintenir un ensemble (éventuellement vide) de propriétés. Une propriété est formée d'un nom et d'une valeur. Les langages les plus connus pour l'interrogation des données stockées suivant un modèle des graphes à propriétés sont Cypher et PGQL Angles et al. (2019). Un modèle de données en graphes à propriétés, noté G, est formellement défini Angles (2018) par (N^G, E^G, ρ^G, λ^G, σ^G) où :

— $N^G=\{n_1,...,n_j\}$ est un ensemble fini de noeuds,

— $E^G=\{e_1,...,e_k\}$ est un ensemble fini d'arcs,

— $\rho^G : E^G \longrightarrow (N^G \times N^G)$ est une fonction totale qui associe chaque arc de E^G à une paire de noeuds (noeuds source et cible) dans N^G,

— $\lambda^G : (N^G \cup E^G) \longrightarrow L$ est une fonction partielle qui associe des noeuds et des arcs à un ensemble d'étiquettes de L,

— $\sigma : (N^G \cup E^G) \times P \longrightarrow V$ est une fonction partielle qui associe des noeuds et des arcs à des propriétés, et pour chaque propriété P elle attribue une valeur de V.

2.4.3 Panorama des SGBD orientés graphes

Au cours des dernières années, la nécessité de gérer des données très connectées et les limites des bases de données traditionnelles pour couvrir les exigences des applications actuelles ont incité la recherche de nouvelles solutions permettant de stocker et interroger rapidement et efficacement les liens entre les données complexes. En conséquence, un certain nombre des SGBD orientés graphes sont apparus et ont été mis à la disposition des entreprises et des chercheurs. Ces SGBD modélisent les données du graphe à l'aide d'un ensemble fini de sommets et un ensemble fini d'arêtes. La plupart des solutions proposées obéissent au modèle des graphes à propriétés. Ainsi, en plus des sommets et des arêtes, les données sont souvent décrites à l'aide d'étiquettes au niveau des sommets et des arêtes.

Les principaux avantages des SGBD orientés graphes sont les suivants Fernandes and Bernardino (2018); Guia et al. (2017) :

— Flexibilité du schéma permettant de stocker des données structurées, semi-structurées et non structurées.

— Représentation de l'information d'une manière très intuitive et très proche des concepts métier.

— Facilité du stockage, de la gestion et de l'interrogation des liens entre les données, généralement impliqués dans des cas réels.

— Haute performance pour les requêtes complexes grâce à la matérialisation des arcs ce qui permet d'éviter les opérations des jointures lors de l'interrogation.

— optimisation de certaines requêtes grâce aux algorithmes de la théorie des graphes tels que la recherche du plus court chemin entre deux noeuds.

Les SGBD orientés graphes les plus populaires sont Neo4j, ArangoDB, orientDB et GraphDB. Comme illustré dans la Figure 2.4, Neo4j est le SGBD le plus dominant. Le score de popularité illustré dans la Figure 2.4 est calculé à l'aide de plusieurs critères de classement comme :

— Nombre de mentions du SGBD sur les sites Web : mesuré en nombre de résultats dans les requêtes des moteurs de recherche,

— Fréquence des discussion techniques sur le système : calculée en utilisant le nombre de questions connexes et le nombre d'utilisateurs intéressés sur les sites de questions-réponses,

— Nombre d'offres d'emploi : défini comme le nombre d'offres publiées sur les principaux moteurs de recherche d'emploi dans lesquelles le système est mentionné.

Les différents critères de classement ci-dessus sont normalisés pour calculer un seul score de popularité pour chaque SGBD.

Dans la littérature, plusieurs travaux de recherche se sont intéressés à la comparaison de la performance de certaines implantations. Dans Alm and Imeri (2021), une étude de performance de bases de données graphes Neo4j, GraphDB et OrientDB a été menée. Les quatre opérations de base CRUD (Create, Read, Update et Delete) permettant de créer, lire, mettre à jour et supprimer des données sont prises en considération dans cette comparaison. Les résultats de ce travail montrent que la performance de la base de données Neo4j surpassent largement GraphDB et OrientDB. Dans Beis et al. (2015), les auteurs ont comparé la performance des trois bases de données graphes Neo4j, Titan et OrientDB. Les résultats expérimentaux de ce travail ont démontré que la performance de ces bases est similaire lors de la gestion de données de petites tailles. Néanmoins, quand la quantité de données devient très volumineuse, Neo4j possède la meilleure performance. Ces bonnes performances ont été confirmées par d'autres travaux Jouili and Vansteenberghe (2013); MAKAME (2018).

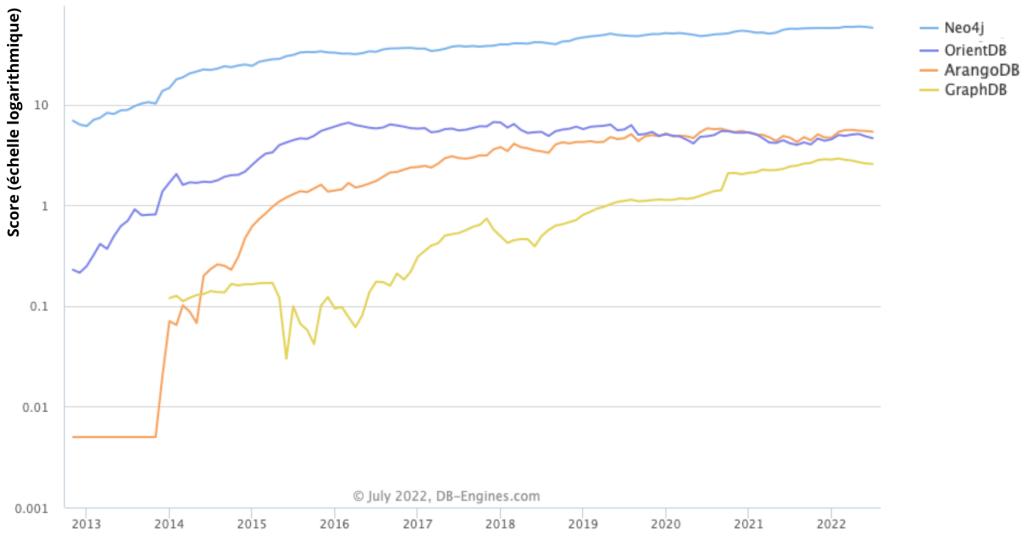


FIGURE 2.4 – Classement de la popularité de différents SGBD orientés graphes dans DB-ENGINES en juillet 2022

En plus des études qui ont comparé les bases de données orientées graphes entre elles, certains travaux de recherche se sont intéressés à la comparaison de la performance des bases de données orientées graphes aux bases de données relationnelles. Dans Yoon et al. (2017), la performance de Neo4j a été comparée à celle de MySQL, un SGBDR

et très connu. Ce travail a démontré que Neo4j surpasse toujours MySQL en termes de temps d'exécution. En effet, plus les relations interrogées étaient complexes, plus la différence de temps entre les deux systèmes était grande. De plus, pour des dizaines de millions de relations ou des relations avec une profondeur supérieure à cinq (traditionnellement cinq opérations de jointure), MySQL n'a pas pu terminer la requête. De même, dans Miller (2013), les auteurs ont comparé la performance de Neo4j à celle de MySQL dans le domaine des réseaux sociaux. Pour des données ayant un petit volume, le SGBD MySQL est meilleur. Cependant, dans le cas des données volumineuses et nécessitant plusieurs jointures, Neo4j est considéré comme le SGBD orienté graphes le plus performant.

En se basant sur les travaux de recherche publiés et les résultats des études de performance publiés, nous constatons que Neo4j est le SGBD orienté graphes le plus efficace lors de la gestion et l'interrogation des données massives et très complexes. Comparé aux SGBD relationnels, Neo4j a l'avantage de gérer plus efficacement des données très connectées. Un critère important lors du choix de cet SGBD est la profondeur des requêtes. Plus que la requête nécessite un nombre important de jointures, plus la performance est à l'avantage de Neo4j. Ainsi, dans nos travaux de recherche nous utilisons Neo4j comme SGBD orienté graphes.

Neo4j est un SGBD open source écrit en Java. Il est développé depuis plus de dix ans. Il est devenu rapidement le principal SGBD orienté graphes. Neo4j s'appuie sur le modèle des graphes à propriétés. Ainsi, contrairement aux SGBD relationnels où les données sont stockées dans les tables, en Neo4j les données sont stockées dans les noeuds et les arcs, et les jointures sont remplacées par des parcours du graphe. Le langage d'interrogation de Neo4j est intitulé Cypher. Les requêtes SQL des SGBD relationnels qui sont de la forme $SELECT\ (colonne)\ FROM\ (table)\ WHERE\ (condition)$ sont remplacés par des requêtes cypher $MATCH(Noeud)\ WHERE\ (condition)\ RETURN\ (propriété)$. Les requêtes complexes nécessitant plusieurs jointures en SQL, sont écrites en précisant les relations entre les données dans Cypher.

Les deux requêtes ci-dessous cf. Tableau 2.1 sont équivalentes et permettent de calculer la moyenne de la quantité vendue par catégorie dans le magasin numéro 1 pendant l'année 2000. En SQL, l'interrogation de plusieurs tables nécessite forcément plusieurs jointures. Néanmoins en graphe, cela se traduit par l'utilisation explicite des liens définis entre données.

SQL	Cypher
SELECT AVG(ss.ss_quantity) avg_qty, i.i_category, **FROM** Date_dim dt, Store_Sales ss, Item i, Store s **WHERE** dt.d_date_sk = ss.ss_sold_date_sk **AND** ss.ss_item_sk = i.i_item_sk **AND** ss.ss_store_sk = s.s_store_sk **AND** dt.d_year = 2000 **AND** s.store_sk = 1 **GROUP BY** i.i_category ;	**MATCH** (d :Date_Dim{d_year :2000}) **WITH** d **MATCH** (ss :Store_Sales) **WITH** ss, d **MATCH** (d)<-[:ON_Date]-(ss) **WITH** ss, d **MATCH** (i :Item) WITH i, ss, d **MATCH** (ss)-[:OF_Item]->(i) **WITH** i, ss, d **MATCH** (s :store{Store_sk :1}) **WITH** i, ss, d, s **MATCH** (ss)-[:AT_Store]->(s) **WITH** i, ss, d, s **RETURN** AVG(ss.ss_quantity) avg_qty, i.i_category ;

TABLE 2.1 – Requête SQL versus Cypher

2.5 Migration vers les bases de données graphes

Comme les bases de données relationnelles sont les plus matures et les plus utilisées depuis plusieurs années, de nombreux systèmes d'information sont déployés à l'aide de ces bases de données. Ainsi, pour pouvoir utiliser ces systèmes dans le contexte des données volumineuses et complexes, il est nécessaire de disposer d'une solution de migration des données d'une base relationnelle à une base NoSQL. Par conséquent, récemment différentes approches de migration ont été proposées. Dans cette section, nous présentons principalement les approches qui permettent la migration vers une base de données orientée graphes.

2.5.1 Méthodologie

L'implantation d'une base de données est un processus complexe qui comprend plusieurs niveaux (étapes) en commençant par l'analyse des besoins jusqu'à la mise en place de la base. D'une manière générale, trois niveaux d'abstraction ont été définis pour les bases de données, à savoir Connolly and Begg (2005) :

— Niveau conceptuel : Modélise les données à stocker dans la base de données indépendamment de tous les aspects techniques. La modélisation conceptuelle repose sur l'analyse des besoins et de l'existant afin de répondre aux attentes des utilisateurs finaux.

— Niveau logique : Consiste à choisir au préalable un modèle de stockage (relation-

nel, orienté colonnes, orienté documents, etc.). Les données sont ensuite représentées en utilisant les concepts de ce modèle (tables, familles de colonnes, collections de documents, etc.).

— Niveau physique : Permet d'implanter concrètement la base de données en utilisant un système de gestion de base de données bien déterminé (MariaDB, Cassandra, MongoDB, etc.).

Le processus de passage d'un niveau d'abstraction à un autre s'appelle correspondance ou « mapping » et se fait à l'aide d'un ensemble de règles de transformation. Par exemple, pour implanter une base de données relationnelle conçue selon le modèle entité-association, des règles doivent être appliquées pour traduire les concepts d'entités et d'associations du modèle conceptuel en concepts équivalents dans le modèle logique relationnel. En plus de l'implantation d'une base de données, il est possible d'effectuer la migration d'une base de données. La migration est un processus qui vise à transformer la structure source vers une structure de données cible alors que les deux structures diffèrent suivant le niveau conceptuel et/ou physique. Matthes et al. (2011). Dans ce cas, il est crucial de définir des règles de transformation qui permettent de trouver la correspondance entre les différents éléments de la structure source et de la structure cible afin de garantir une migration efficace et sans perte de données. Par exemple, pour passer d'une base relationnelle qui stocke les données dans des tables constituées d'enregistrements et de colonnes à une base de données orientée documents qui stocke les données dans des collections de documents, il est crucial de définir des règles de transformation. Ces règles permettent de trouver la correspondance entre les concepts du modèle relationnel source (table, ligne, colonne, etc.) et les concepts du modèle orienté documents cible (collection de documents, document, document imbriqué, etc.).

2.5.2 Règles de transformation

Dans la littérature, différents travaux ont étudié la migration vers les graphes. Des règles de transformation ont été proposées pour migrer d'une base relationnelle à une base orientée graphes. Les approches proposées regroupent généralement deux étapes principales, à savoir Bordoloi and Kalita (2013) :

— L'analyse de métadonnées (MA) : Consiste à inspecter le schéma et les contraintes de la base de données relationnelle source et construit le modèle orienté graphes correspondant à la base de données cible.

— La traduction (mapping) des données (DM) : Permet d'assurer le stockage dans la base de données orientées graphes en appliquant des règles de transformation qui

cherchent la correspondance entre les métadonnées du schéma source et le schéma cible.

La version la plus simple et la plus naïve des approches de conversion d'une base relationnelle en une base orientée graphes De Virgilio et al. (2013); Bordoloi and Kalita (2013) suivent généralement une méthode qui transforme fondamentalement les tuples en noeuds et les clés étrangères en arcs qui relient les noeuds. Dans De Virgilio et al. (2013), une approche appelée Relational To Graph (R2G) a été proposée et implémentée en langage Java pour convertir automatiquement une base relationnelle en graphes. L'idée de base de cette approche est de stocker dans le même noeud des valeurs de données susceptibles d'être récupérées ensemble lors de l'interrogation de la base à l'aide des requêtes. Un algorithme universel qui convertit un modèle relationnel en un modèle orienté graphes a été aussi proposée dans Orel et al. (2016). Cet algorithme considère différents aspects du processus de migration comme le choix des tables à stocker dans des noeuds et l'identification des clés étrangères à utiliser pour les relations. L'algorithme a été implémenté en tant que procédure stockée dans le SGBD Informix et les données ont été converties en graphes et stockées dans Neo4j. Dans Megid et al. (2018), un algorithme Functional Dependency To Graph (FD2G) a été proposé et qui profite de l'existence d'une dépendance fonctionnelle entre les informations existantes dans la base de données relationnelle. Ces dépendances sont utilisées pour effectuer automatiquement la conversion en graphes. Dans Unal and Oguztuzun (2018), les auteurs détaillent les différentes étapes permettant de créer une base de données orientées graphes à partir d'une base de données relationnelle conçue selon le modèle entité-association, à savoir :

— Chaque table d'entités est représentée dans le modèle orienté graphes par une étiquette sur les noeuds.
— Chaque ligne d'une table d'entités est un noeud dans le graphe.
— Les colonnes de ces tables deviennent des propriétés stockées dans les noeuds.
— Les clés primaires techniques doivent être supprimées et uniquement les clés primaires métier sont conservées. Des index sont ajoutés pour les attributs de recherche fréquents.
— Les clés étrangères se traduisent par des arcs entre les noeuds. Ces clés sont à retirer après la création des liens.
— Les tables de jointure sont transformées en noeuds, les colonnes de ces tables deviennent des propriétés dans les noeuds.

Les approches de migration proposées ont permis de montrer la faisabilité de convertir les concepts du modèle relationnel en concepts équivalents dans les bases orientées

graphes. Certains travaux De Virgilio et al. (2014); Megid et al. (2018) ont implémenté et testé ces approches. Les résultats ont montré une meilleure efficacité face à des requêtes complexes comparée aux bases relationnelles.

2.6 Les graphes au coeur de la science des données

La révolution numérique a entraîné une augmentation exponentielle du volume et de la diversité des données disponibles. Les réseaux sociaux, les objets connectés et les données ouvertes ne sont que quelques exemples de sources de données émergentes qui peuvent être exploitées pour améliorer les processus décisionnels des entreprises. En effet, transformer ces données en connaissances exploitables peut permettre de mieux comprendre les besoins des clients, d'optimiser les offres de produits et de services, de renforcer les stratégies de marketing et de prévoir les tendances du marché. C'est dans ce contexte que la science des données est apparue, offrant des méthodes et des outils pour extraire des connaissances pertinentes à partir des données massives. La science des données est donc devenue une discipline cruciale pour les entreprises, leur permettant d'analyser les données de manière efficace et efficiente, afin de mieux comprendre leur marché et de prendre des décisions éclairées.

Les techniques de la science des données incluent l'exploration et la visualisation des données, l'apprentissage automatique, la fouille de données, la statistique et l'analyse prédictive. La discipline de la science des données repose sur plusieurs techniques utilisées pour comprendre, modéliser et prédire des données, ainsi que pour extraire des informations pertinentes à partir des liens entre ces données. Cette discipline est étroitement liée à des domaines connexes tels que la gestion de bases de données, la Business Intelligence (BI) et la visualisation de données Kotu and Deshpande (2018). Aujourd'hui, la science des données est au centre de nombreux travaux de recherche qui visent à résoudre des problèmes concrets dans divers domaines tels que la maintenance prédictive pour éviter les pannes de machines dans l'industrie 4.0 Diallo et al. (2021), la prédiction de l'efficacité d'un médicament pour le traitement de nouvelles maladies Himmelstein et al. (2017), la recommandation de nouveaux amis sur les réseaux sociaux Kumari et al. (2022) ou de nouveaux collaborateurs dans les réseaux scientifiques Lande et al. (2020); Fernandes and Bernardino (2018).

2.7 Conclusion

Il est indéniable que les données pouvant être représentées intuitivement à l'aide de graphes sont au cœur de nombreux types d'applications. Cependant, l'utilisation de

bases de données relationnelles pour stocker et modéliser ces données peut présenter plusieurs inconvénients, tels que des coûts élevés en termes de jointures. Grâce à l'émergence de bases de données NoSQL, et en particulier de modèles orientés graphes, de nombreux travaux de recherche ont proposé d'exploiter ce modèle pour une gestion plus efficace de données complexes. Les travaux de migration du relationnel vers le modèle orienté graphes ont montré la faisabilité et l'efficacité de l'utilisation de bases de données graphes pour stocker des données hautement connectées dans divers domaines scientifiques tels que la sociologie, la biologie et la chimie. Dans le chapitre suivant, nous nous intéressons à l'utilisation des graphes pour la mise en place de systèmes décisionnels. Nous introduisons une nouvelle méthodologie appelée MDM2G, qui permet d'implémenter un entrepôt de données orienté graphes. Cette méthodologie s'appuie sur la modélisation des données en utilisant des graphes et vise à fournir des solutions efficaces pour la gestion de données complexes. L'efficacité de cet entrepôt face à des requêtes analytiques complexes est également étudiée en détail dans le chapitre suivant.

Chapitre 3

Implantation d'entrepôts de données basés sur les graphes

Sommaire

3.1 Introduction

L'entrepôt de données est l'un des outils les plus utilisés pour historiser les données analytiques afin d'aider les décideurs lors de la prise de la décision dans une multitude de domaines tels que la santé Hamoud et al. (2018), l'agriculture Ngo et al. (2019) et la biologie Bernasconi et al. (2017). Pendant plusieurs années, les SGBDR ont été considérés comme la meilleure technologie pour l'implantation des entrepôts de données. L'approche R-OLAP basée sur les bases de données relationnelles a été la plus dominante. Étant donné les limites des bases de données relationnelles détaillées dans le chapitre précédent dans la section 2.2.3, plusieurs travaux de recherche ont étudié la possibilité d'utiliser les bases de données NoSQL lors de l'implantation des entrepôts de données massives Dehdouh (2016); Chevalier et al. (2015b); Scabora et al. (2016); Santos and Costa (2016); Boussahoua et al. (2017); Chavalier et al. (2016); Soransso and Cavalcanti (2018); Gallinucci et al. (2019). Les approches proposées dans ces travaux peuvent être regroupées en trois principales catégories, à savoir : approche basée sur le modèle orienté colonnes Dehdouh (2016); Chevalier et al. (2015b); Scabora et al. (2016); Santos and Costa (2016); Boussahoua et al. (2017), approche basée sur le modèle orienté documents Chevalier et al. (2015b); Chavalier et al. (2016); Soransso and Cavalcanti (2018); Gallinucci et al. (2019) et approche basée sur le modèle orienté graphes Castelltort and Laurent (2014); Sellami et al. (2018). Certains des travaux précédemment cités ont étudié la performance des entrepôts NoSQL orientés colonnes et documents en se basant sur des métriques telles que le temps de chargement de données dans l'entrepôt Chavalier et al. (2016) et le temps de réponse à des requêtes utilisateurs Dehdouh et al. (2015); Chavalier et al. (2016).

Dans ce chapitre, nous présentons dans la section 3.2 les concepts clés relatifs aux entrepôts de données et nous passons en revue les travaux existants dans la littérature et qui permettent de migrer vers un entrepôt de données NoSQL. Ensuite, nous détaillons dans la troisième section 3.3 notre approche d'implantation d'entrepôt de données à l'aide d'une base NoSQL orientée graphes. Plus précisément, nous présentons nos règles de transformation que nous avons nommées MDM2G et qui permettent de convertir un schéma multidimensionnel en étoile et un schéma en flocon de neige en un modèle logique NoSQL orienté graphes. Après, dans la quatrième section 3.4 nous détaillons les expérimentations que nous avons menées pour comparer la performance des entrepôts de données orientés graphes aux entrepôts relationnels. Dans la section 3.5 nous présentons les résultats de nos expérimentations que nous discutons dans la section 3.6. Enfin, nous clôturons dans la cinquième section 3.7 ce chapitre par le choix du modèle le plus pertinent dans le cas de l'interrogation des données analytiques avec des requêtes

complexes.

3.2 État de l'art

L'avènement du Big Data a créé plusieurs défis pour la gestion des données massives ; parmi ceux-ci, nous trouvons l'intégration des données analytiques dans les entrepôts. Un entrepôt de données est un type de base de données dédié principalement au stockage de gros volumes de données utilisées à des fins analytiques. La définition donnée par Bill Inmon énonce qu'un entrepôt de données est un ensemble de données intégrées, orientées vers un sujet, non volatiles, historisées, résumées et accessibles pour l'interrogation et l'analyse. Les différents attributs mentionnés dans cette définition, tels que l'intégration, l'orientation sujet, la non-volatilité, l'historisation et le résumé, sont expliqués en détail dans Teste (2000) :

— Intégrées : Les données stockées dans un entrepôt proviennent souvent de sources multiples et potentiellement hétérogènes. Afin de garantir un état cohérent, il est nécessaire de les intégrer en les homogénéisant grâce à un codage et une description unique ;

— Orientées sujet : Les données de l'entrepôt sont structurées selon une organisation thématique autour des sujets majeurs de l'entreprise, contrairement aux systèmes transactionnels qui sont structurés par processus fonctionnel. Cette approche orientée sujet permet de disposer de l'ensemble des informations nécessaires au processus analytique ;

— Historisées : Les données stockées dans l'entrepôt sont destinées à être consultées et analysées plutôt que modifiées, afin de préserver la traçabilité de l'information et des décisions prises. En conséquence, les opérations de mise à jour et de suppression ne sont pas autorisées. La fonction d'historisation permet de suivre l'évolution des données au fil du temps et d'en tirer des connaissances pour faciliter la prise de décisions ;

— Résumées : Les données provenant des sources doivent être agrégées et réorganisées pour faciliter le processus de prise de décision. Cela implique la création de vues agrégées de données pour fournir une vue synthétique et cohérente des informations importantes pour l'analyse ;

— Disponibles pour l'interrogation et l'analyse : Les données de l'entrepôt doivent être accessibles pour les utilisateurs qui ont des droits d'accès spécifiques, ce qui permet de répondre à leurs besoins d'analyse et de prise de décision. Pour cela,

l'entrepôt doit disposer d'un module de traitement de requêtes qui permet aux utilisateurs d'interroger les données de manière efficace. Ce module doit être capable de comprendre un langage de requête riche en opérateurs et d'exploiter la richesse du modèle de données pour fournir des réponses précises et rapides.

L'architecture générale des systèmes décisionnels repose sur quatre éléments clés (voir Figure 3.1) :

— Les sources de données : Les données qui alimentent l'entrepôt proviennent de différentes sources, sont de différents types et sont souvent dispersées dans plusieurs systèmes. Il y a les données internes (provenant des bases de production) et les données externes (fournies par les partenaires tels que les fournisseurs, clients, documents juridiques, etc.) à l'entreprise ;

— L'entrepôt de données : C'est l'espace de stockage de l'ensemble des données pertinentes pour la prise de décision. Les données sont organisées selon le modèle de l'entrepôt de données et sont historisées pour permettre une analyse temporelle ;

— Les magasins de données : Ce sont des sous-ensembles extraits à partir de l'entrepôt de données pour répondre aux besoins d'analyse rapides et spécifiques à une catégorie de décideurs ;

— Les outils d'analyse : Ils permettent de manipuler les données selon plusieurs axes d'analyse. La visualisation de l'information s'effectue de manière claire et lisible pour aider les décideurs qui ne sont généralement pas des experts en informatique.

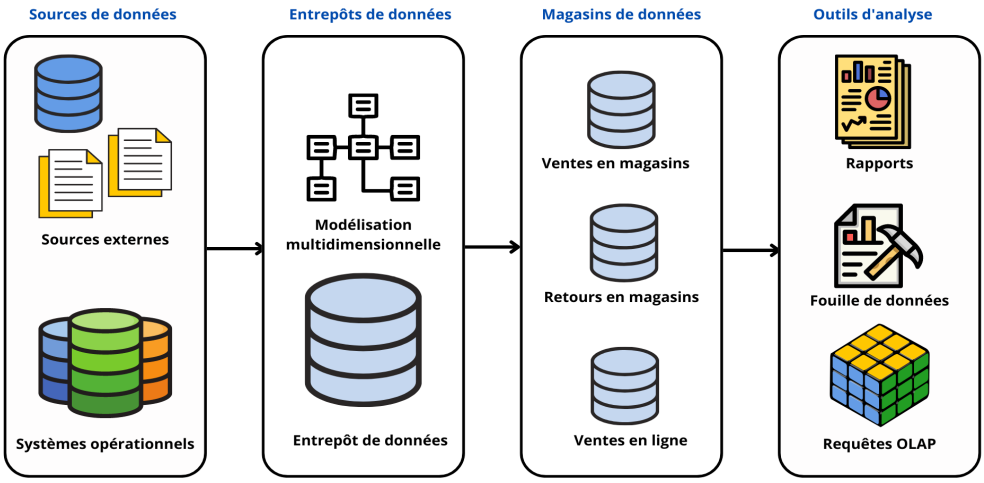


FIGURE 3.1 – Architecture classique d'un système décisionnel

Les entrepôts de données permettent aux décideurs d'avoir une vision globale de l'information circulant dans leurs entreprises en organisant les données. Généralement, un entrepôt de données est conçu selon un modèle multidimensionnel Kimball (1996).

3.2.1 Modélisation conceptuelle multidimensionnelle

a modélisation conceptuelle multidimensionnelle est une approche pour décrire les données d'un entrepôt en utilisant un niveau d'abstraction élevé. Les données d'un entrepôt sont ainsi décrites comme si elles étaient placées dans un espace à plusieurs dimensions Gosain and Singh (2015). La modélisation multidimensionnelle considère le sujet analysé comme un point dans un espace à plusieurs dimensions avec diverses perspectives d'analyse, ce qui la rend plus proche de la façon de penser des analystes de données et des utilisateurs finaux. Elle se base sur trois concepts fondamentaux : le fait, la dimension et la hiérarchie.

— Concept de fait : Il modélise le sujet à analyser. Le fait est décrit à l'aide d'un ou plusieurs attributs appelés mesures et qui possèdent souvent des valeurs numériques. Dans la Figure 3.2, le fait $Store\_Sales$ modélise les ventes des produits de l'entreprise effectuées dans les magasins. La mesure $ss\_quantity$ représente la quantité de produits vendus. Généralement, le fait est analysé en associant ses mesures à des fonctions d'agrégation telles que la somme (SUM), la moyenne (AVG) et le minimum (MIN) ;

— Concept de dimension : Il représente l'axe suivant lequel le fait est analysé. Chaque dimension comporte un ensemble d'attributs utilisés pour varier les valeurs des mesures. Il existe deux types d'attributs. Le premier type d'attributs appelés paramètres sert à définir les niveaux de granularité ou de détail. Le deuxième type d'attributs dits faibles ou informationnels sert à décrire les paramètres. Par exemple, dans la Figure 3.2, les ventes de la table de faits $Store\_Sales$ sont analysées suivant la dimension $Customer$. $Customer\_Demographics$ est un paramètre qui peut être utilisé pour varier la mesure $ss\_quantity$ et analyser les ventes par la démographie du client. c_first_name est un attribut faible permettant de décrire le client par son prénom. Généralement, chaque modèle conceptuel multidimensionnel contient une dimension Date permettant d'historiser les données afin d'observer leurs évolutions et aider les décideurs dans la prise de la décision ;

— Concept de hiérarchie : Elle permet d'organiser les paramètres d'une dimension suivant leurs niveaux de granularité en commençant par la granularité la plus fine et en allant à la granularité la plus grossière. Par exemple, dans la Figure 3.2, les attributs successifs $Customer\_Demographics$ et $Income\_Band$ de la dimension $Customer$ sont organisés suivant leurs niveaux de granularité.

L'implantation d'un entrepôt de données nécessite de transformer son schéma conceptuel multidimensionnel en un schéma logique (relationnel, NoSQL, etc.). Le schéma

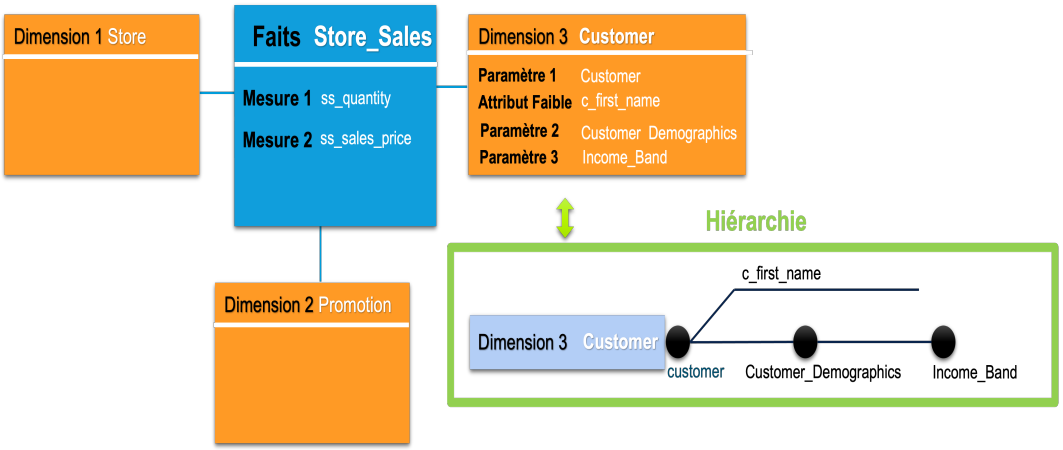


FIGURE 3.2 – Modèle conceptuel multidimensionnel

logique est ensuite traduit en un schéma physique en prenant en considération les technologies à utiliser pour la mise en oeuvre de l'entrepôt (MariaDB, Oracle, Cassandra, MongoDB, etc.). Dans la littérature, plusieurs approches ont été proposées pour définir un schéma logique multidimensionnel d'un entrepôt de données.

3.2.2 Approche relationnelle

En vue de construire un modèle logique approprié pour un entrepôt de données, nous pouvons suivre l'approche Relational OnLine Analytical Processing (R-OLAP), Hybrid OnLine Analytical Processing (H-OLAP) ou Multidimensional OnLine Analytical Processing (M-OLAP). R-OLAP est la stratégie d'entreposage la plus ancienne. Elle permet de transformer les concepts de fait et de dimension d'un modèle conceptuel multidimensionnel en tables relationnelles. Ainsi, deux principaux schémas relationnels pour le décisionnel ont été définis dans le cadre de cette approche en vue de simuler une structure multidimensionnelle dans un SGBDR, à savoir : le schéma en étoile et le schéma en flocon de neige.

— Schéma en étoile : Un schéma en étoile comprend une table de faits centrale et des tables de dimensions (Figure 3.3). Ce modèle représente les dimensions d'une manière dénormalisée. Chaque table de dimension est liée à la table de faits à l'aide de sa clé primaire, transformée en clé étrangère dans la table de fait. Néanmoins les dimensions ne sont pas liées entre elles ;

— Schéma en flocon de neige : Un schéma en flocon de neige est une extension du modèle en étoile dans laquelle certaines dimensions sont hiérarchisées (Figure 3.4). Elle consiste à garder la même table de faits et à normaliser les tables de dimension en vue de permettre une représentation plus explicite de la hiérarchie.

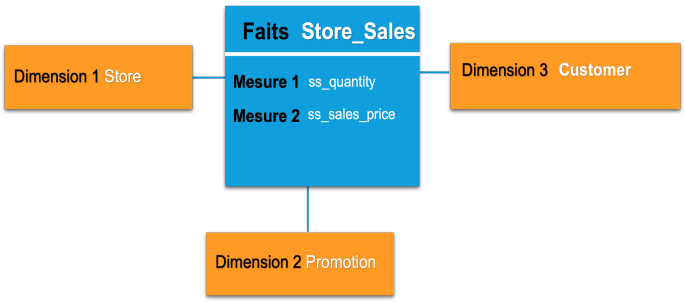


FIGURE 3.3 – Schéma en étoile

La normalisation est l'une des caractéristiques fondamentales des bases de données relationnelles. L'objectif de la normalisation est d'éliminer les redondances afin d'éviter les problèmes de mise à jour. Dans le cas du schéma en flocons, les dimensions sont décrites à travers une succession de tables reliées par des clés étrangères.

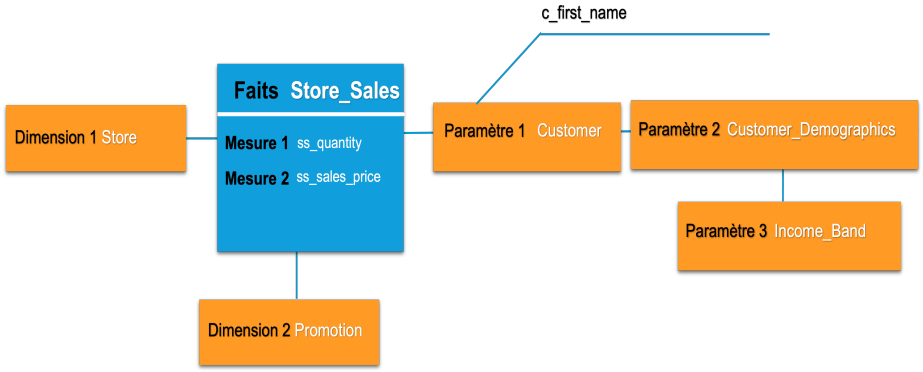


FIGURE 3.4 – Schéma en flocon de neige

Avec l'émergence des systèmes NoSQL et la remise en cause des bases de données relationnelles, plusieurs chercheurs ont proposé de nouvelles approches permettant d'implanter des entrepôts de données NoSQL à l'aide du modèle NoSQL orienté colonnes Serrano et al. (2015), documentsJia et al. (2016); Chouder et al. (2019) ou graphes De Virgilio et al. (2014); Lee et al. (2015).

3.2.3 Approche orientée colonnes

Cette approche permet aux entrepôts de données d'être implantés sous des systèmes NoSQL orientés colonnes. Dans Chevalier et al. (2015b); Boussahoua et al. (2017) les auteurs ont proposé un ensemble de règles de transformation pour convertir les faits, mesures, dimensions et attributs en concepts analogues dans le modèle NoSQL orienté colonnes. Plus précisément, les faits et les dimensions sont transformés en familles de colonnes où les mesures et les attributs sont stockés dans des colonnes. Dans Dehdouh

et al. (2015), les auteurs ont proposé trois méthodes pour permettre la mise en oeuvre de l'entrepôt de données orienté colonnes. La première méthode permet de stocker des faits et des dimensions dans la même famille de colonnes. La deuxième méthode stocke les faits et la dimension séparément. Chaque table de faits est transformée en une famille de colonnes contenant des mesures sous la forme de colonnes. Les dimensions sont également transformées en familles de colonnes ayant des attributs comme colonnes. Cette méthode modélise et stocke un schéma en étoile. La troisième méthode prend en compte les hiérarchies. Chaque attribut d'une dimension est stocké dans une famille de colonnes distincte. Les résultats de ce travail ont montré que l'interrogation des hiérarchies dans des entrepôts de données orientés colonnes est très coûteuse en termes de temps de calcul. De même, les résultats de Dehdouh et al. (2015) montrent que le fractionnement des attributs des dimensions dans différentes familles de colonnes affecte les performances de l'entrepôt de données orienté colonnes.

3.2.4 Approche orientée documents

Dans la littérature, de nombreuses méthodes ont été proposé afin de permettre la transformation du modèle multidimensionnel en un modèle orienté documents. Dans Chevalier et al. (2015b), les auteurs ont proposé de convertir chaque fait en une collection de documents contenant ses mesures. La valeur de chaque mesure est associée à une clé qui porte le nom de la mesure. En ce qui concerne les dimensions, elles sont transformées en collections de documents contenant les attributs. Chaque valeur d'un attribut est associée à une clé qui contient son nom. Dans Chevalier et al. (2015a), les auteurs ont proposé trois méthodes d'implantation d'entrepôts de données basées sur le modèle orienté documents. Dans la première méthode, les faits et les dimensions sont stockés ensemble dans la même collection de documents. Dans la seconde méthode, les faits et les dimensions sont stockés dans différentes collections de documents. En d'autres termes, chaque fait et les mesures associées sont stockés dans une collection de documents. De même, chaque dimension et ses attributs associés sont également stockés dans une collection de documents. Le dernier ensemble reprend la deuxième méthode tout en normalisant les dimensions pour modéliser et créer explicitement des hiérarchies. Ainsi, les dimensions sont normalisées dans différentes collections de documents ayant des attributs en tant que des sous documents. Cette étude a révélé que la modélisation et le stockage de hiérarchies à l'aide du concept de sous documents réduisent considérablement les performances des requêtes analytiques complexes qui nécessitent des jointures multiples. Indépendamment du contexte d'implantation des entrepôts de données, une autre étude Gómez et al. (2016) a rapporté l'impact de la structuration des données sous

la forme de plusieurs documents imbriqués. Les résultats de ce travail ont démontré que l'interrogation de données stockées à différents niveaux dans une collection de documents nécessite une manipulation complexe et plus de temps pour être exécutée.

3.2.5 Approche orientée graphes

Le modèle orienté graphes est composé de noeuds et d'arcs étiquetés. Les noeuds et les arcs peuvent stocker des propriétés sous forme de paires clé-valeurs. Afin de mettre en oeuvre un entrepôt de données orienté graphes, Castelltort and Laurent (2014) a proposé de transformer les faits en noeuds. Les mesures de chaque fait sont stockées en tant que propriétés dans le même noeud. De plus, les dimensions sont transformées en noeuds. Il existe deux types de relations entre les noeuds. Le premier type de relation est appelé $FACT$ qui lie le fait aux dimensions. Le deuxième type est étiqueté $HIER$ qui lie les attributs des dimensions. Ce travail s'est concentré sur l'adaptation du langage Cypher d'interrogation de la base de données orientée graphes Neo4j pour prendre en charge les opérateurs OLAP, principalement les Slice, Dice et Roll up. Cependant, dans ce travail, les auteurs n'ont pas étudié l'efficacité de l'entrepôt de données orienté graphes, notamment lorsque les requêtes deviennent plus complexes ou quand le volume de données augmente. Dans Sellami et al. (2018), les auteurs fournissent des règles de transformation formelles pour convertir un modèle conceptuel multidimensionnel en modèle NoSQL orienté graphes. Cependant, les performances de l'entrepôt de données proposé n'ont pas été évaluées.

En l'absence d'évaluation des performances des entrepôts de données orientés graphes et avec l'intérêt croissant pour les graphes dans le domaine de la science des données, nous proposons dans ce chapitre une approche d'entreposage orientée graphes que nous évaluons à travers deux métriques : le temps de chargement des données et le temps de réponse à un jeu de requêtes analytiques.

3.3 Règles de transformation « Mutidimensional Data Model to Graph » : MDM2G

Pour implanter un entrepôt NoSQL orienté graphes, des règles de transformation sont nécessaires pour traduire les concepts du modèle conceptuel multidimensionnel en concepts équivalents dans le modèle logique orienté graphes. Ce dernier est un graphe étiqueté où les données sont représentées à travers les concepts de noeuds, d'arcs et de propriétés définies au niveau des noeuds ou / et des arcs (paires clé-valeur). Les noeuds représentent des entités. Les arcs représentent les relations entre les entités. Les noeuds et

les arcs peuvent avoir une ou plusieurs étiquettes et contenir une ou plusieurs propriétés.

Traditionnellement, l'approche R-OLAP transforme chaque fait du modèle conceptuel multidimensionnel en une table de fait. La table de faits contient des mesures sous forme de colonnes. Chaque dimension est aussi convertie en une table de dimensions qui contient des attributs sous forme de colonnes. Chaque instance de tables de faits et de dimensions est stockée dans une ligne spécifique. D'une manière analogue à l'approche R-OLAP, nous proposons nos règles de transformation. Pour ce faire nous définissons d'abord les concepts du modèle de données source qui est le modèle conceptuel multidimensionnel.

3.3.1 Formalisme du modèle multidimensionnel

Définition 1. Un modèle multidimensionnel dénoté MDM, est formellement défini Ravat et al. (2008); Chevalier et al. (2015b) par le triplet (F^{MDM}, D^{MDM}, $Star^{MDM}$) où :

— $F^{MDM}=\{f_1,...,f_n\}$ est un ensemble fini de faits,

— $D^{MDM}=\{d_1,...,d_m\}$ est un ensemble fini de dimensions,

— $Star^{MDM} : F^{MDM} \longrightarrow 2^{D^{MDM}}$ est une fonction qui associe chaque fait $F_i \in F^{MDM}$ à un ensemble de dimensions $D_i \in D^{MDM}$.

Définition 2. Un fait $F_i \in F^{MDM}$, est défini par (Nom^{F_i}, M^{F_i}) où :

— Nom^{F_i} est le nom du fait,

— $M^{F_i}=\{m_1,...,m_{p_i}\}$ est un ensemble de mesures.

Définition 3. Une dimension $D_i \in D^{MDM}$ est définie par (Nom^{D_i}, A^{D_i}, H^{D_i}) où :

— Nom^{D_i} est le nom de la dimension,

— $A^{D_i}=\{a_1,...,a_{r_i}\}$ est un ensemble d'attributs,

— $H^{D_i}=\{h_1,...,h_{s_i}\}$ est un ensemble de hiérarchies.

Définition 4. Une hiérarchie d'une dimension D_i, notée $H_j \in H^{D_i}$, est définie par (Nom^{H_j}, $Param^{H_j}$, $Faible^{H_j}$) où :

— Nom^{H_j} est le nom de la hiérarchie,

— $Param^{H_j}=\{param_1{}^{H_j},...,param_{q_j}{}^{H_j}\}$ est un ensemble d'attributs appelés paramètres de hiérarchie,

— $Faible^{H_j}$ est une fonction associant à chaque paramètre zéro ou plusieurs attributs faibles.

3.3.2 Modèle des graphes à propriétés

Le modèle cible de nos règles de transformation est un modèle des graphes à propriétés. Des modèles de données orienté graphes sont apparus depuis les années 1980, mais leur popularité a progressivement diminué avec l'émergence d'autres modèles telles que le modèle spatial, semi-structuré et XML Angles and Gutierrez (2008). Récemment, les bases de données orientées graphes ont regagné l'attention en raison de la nécessité croissante de stocker, de traiter, de gérer et d'analyser des données qui peuvent être représentées intuitivement à l'aide graphes comme les réseaux sociaux Akid and Ayed (2016); Moosavi et al. (2017), les réseaux biologiques Himmelstein et al. (2017); Fabregat et al. (2018); Timón-Reina et al. (2021), et les réseaux de documents Mezzanzanica et al. (2018); Mercorio et al. (2019). En effet, les bases de données orientées graphes sont considérées comme l'une des structures les plus efficaces et naturelles pour modéliser les interactions entre les objets d'un réseau Vicknair et al. (2010). De nombreux systèmes de gestion de bases de données orientées graphes sont disponibles aujourd'hui, tels que Neo4j Robinson et al. (2015) et GraphDB Güting (1994). Un schéma de base de données ainsi que des instances dans ce modèle sont sous la forme d'un graphe dirigé et étiqueté, où les noeuds représentent les objets et les arcs représentent les connexions entre eux. Alors que les bases de données relationnelles nécessitent des opérations de jointure coûteuses pour répondre à des requêtes complexes, les bases de données orientées graphes considèrent les relations entre les entités aussi importantes que les entités elles-mêmes Batra and Tyagi (2012) ce qui facilite la navigation entre les entités. Comme détaillé dans le chapitre précédent, il existe deux modèles de données orientés graphes : le graphe à propriétés (GP) permettant aux noeuds (sommets) et aux arcs d'avoir un certain nombre de propriétés arbitraires et le cadre de description des ressources (RDF) initialement conçu pour représenter des informations sur les ressources sur le World Wide Web Raman et al. (2017). Le modèle le plus utilisé est le modèle des graphes à propriétés Robinson et al. (2015). De façon informelle, un GP est un graphe étiqueté dirigé où les données sont représentées au moyen de noeuds, d'arcs et de propriétés (paires clé-valeur). Les noeuds représentent les entités et les arcs représentent les relations entre elles. Les noeuds et les arcs peuvent être étiquetés avec une ou plusieurs étiquettes et contiennent des propriétés qui représentent leurs caractéristiques. Définissons L, P et V comme :

— $L=\{l_1,...,l_a\}$ est un ensemble infini d'étiquettes,

— $P=\{p_1,...,p_b\}$ est un ensemble infini de noms de propriétés,

— $V=\{v_1,...,v_c\}$ est un ensemble fini de valeurs atomiques.

Définition 5. Un modèle de données en graphe à propriétés, noté G, est formellement défini Angles (2018) par (N^G, E^G, ρ^G, λ^G, σ^G) où :

— N^G=$\{n_1,...,n_j\}$ est un ensemble fini de noeuds,

— E^G=$\{e_1,...,e_k\}$ est un ensemble fini d'arcs,

— $\rho^G : E^G \longrightarrow (N^G \times N^G)$ est une fonction totale qui associe chaque arc de E^G à une paire de noeuds (noeuds source et cible) dans N^G,

— $\lambda^G : (N^G \cup E^G) \longrightarrow L$ est une fonction partielle qui associe des noeuds et des arcs à un ensemble d'étiquettes de L,

— $\sigma : (N^G \cup E^G) \times P \longrightarrow V$ est une fonction partielle qui associe des noeuds et des arcs à des propriétés, et elle attribue une valeur de V à chaque propriété.

3.3.3 Cas d'un schéma en étoile

Dans le contexte des bases de données relationnelles, la conception de l'étoile transforme chaque fait du modèle conceptuel multidimensionnel en une table de faits relationnels. La table de faits contient des mesures sous forme de colonnes. De plus, chaque dimension est convertie en une table de dimension dénormalisée qui contient tous les attributs (paramètres et attributs faibles) sous forme de colonnes. Chaque instance de tables de faits et de dimensions est stockée dans une ligne spécifique. De la même manière, nous utilisons les définitions mentionnées précédemment des concepts de modèle multidimensionnel et de graphe à propriétés pour proposer nos règles de transformation qui définissent un schéma de graphe en forme d'étoile « Star Graph ».

Transformation 1. Chaque modèle de données multidimensionnel MDM(F^{MDM}, D^{MDM}, $Star^{MDM}$) est transformé en un modèle de données en graphe multidimensionnel MGD(N^{MGD}, E^{MGD}, ρ^{MGD}, λ^{MGD}, σ^{MGD}) où :

— N^{MGD}=$\{n_1,...,n_j\}$ est un ensemble fini de noeuds de faits et de dimensions,

— $\rho^{MGD} : E^{MGD} \longrightarrow (N^{MGD} \times N^{MGD})$ est une fonction totale qui associe chaque arc dans E^{MGD} à un noeud de fait source et un noeud de dimension cible dans N^{MGD},

— $\lambda^{MGD} : (N^{MGD} \cup E^{MGD}) \longrightarrow L$ est une fonction partielle qui associe à l'ensemble des noeuds représentant les faits et les dimensions et aux arcs un ensemble d'étiquettes de L,

— $\sigma : (N^{MGD} \cup E^{MGD}) \times P \longrightarrow V$ est une fonction partielle qui associe à l'ensemble des noeuds représentant les faits et les dimensions et aux arcs un ensemble de propriétés, et pour chaque propriété elle attribue une valeur de V.

Transformation 2. Chaque fait $F_i(Nom^{F_i}, M^{F_i}) \in F^{MDM}$ se transforme en un ensemble de noeuds de faits $N_{F_i} \in N^{MGD}$ définis par ($Nom^{F_i^{MGD}}$, $M^{F_i^{MGD}}$) où :

— $Nom^{F_i^{MGD}}$ est le nom du fait F_i associé à la fonction λ^{MGD} comme étiquette pour les noeuds de faits N_{F_i},

— $M^{F_i^{MGD}}$ est un ensemble de mesures du fait F_i associé à la fonction σ^{MGD} aux noeuds de fait N_{F_i} comme propriétés. La valeur de la mesure est stockée en tant que valeur de la propriété.

Cette règle crée autant de noeuds de faits que d'instances du fait. La Figure 3.5 illustre cette règle de transformation. Dans notre exemple, le fait « Store_Sales » se transforme en un ensemble de noeuds avec la même étiquette de fait « Store_Sales » ayant « ss_ticket_number » et « ss_quantity » comme propriétés de mesure.

Transformation 3. Chaque dimension $D_i(Nom^{D_i}, A^{D_i}, H^{D_i}) \in D^{MDM}$ est transformée en un ensemble de noeuds de dimension $N_{D_i} \in N^{MGD}$ défini par $(Nom^{D_i^{MGD}}, A^{D_i^{MGD}})$ où :

— $Nom^{D_i^{MGD}}$ est le nom de la dimension D_i associé à la fonction λ^{MGD} comme étiquette de la dimension noeuds N_{D_i},

— $A^{D_i^{MGD}}$ est un ensemble d'attributs (paramètres et attributs faibles) de la dimension D_i associé à la fonction σ^{MGD} en tant que propriétés dans le noeud de dimension N_{D_i}. Ainsi, les hiérarchies ne sont pas prises en compte,

— Un arc est défini entre chaque noeud de fait source N_{F_i} et le noeud de dimension cible N_{D_i} à l'aide de la fonction ρ^{MGD} .

Cette règle permet de créer pour chaque dimension autant de noeuds que ses instances. La Figure 3.5 illustre la transformation des dimensions et de leurs attributs. Dans notre exemple, la dimension « Customer » est transformée en un ensemble de noeuds ayant la même étiquette « Customer ». Tous les attributs qui donnent des détails sur les clients sont transformés en propriétés dans les noeuds Customer.

Dans cette transformation, tous les noeuds des dimensions sont directement liés au fait par des arcs. Par conséquent, le schéma en forme d'étoile permet d'interroger le modèle de données orienté graphes multidimensionnel en traversant le graphe d'un niveau. Dans ce cas, la profondeur, qui est le nombre de chemins entre un noeud de fait et un noeud de dimension, est égale à un. La Figure 3.5 montre la transformation des jointures entre la table de faits « $Store\_Sales$ » et la table des dimensions « $Customer$ » en un ensemble d'arcs étiquetés « : $BY\_Customer$ ».

3.3.4 Cas d'un schéma en flocon

Contrairement au modèle de données en étoile où tous les paramètres sont regroupés dans une seule table de dimension dénormalisée, le modèle de données en flocon de neige permet de représenter des hiérarchies en utilisant plusieurs sous-dimensions

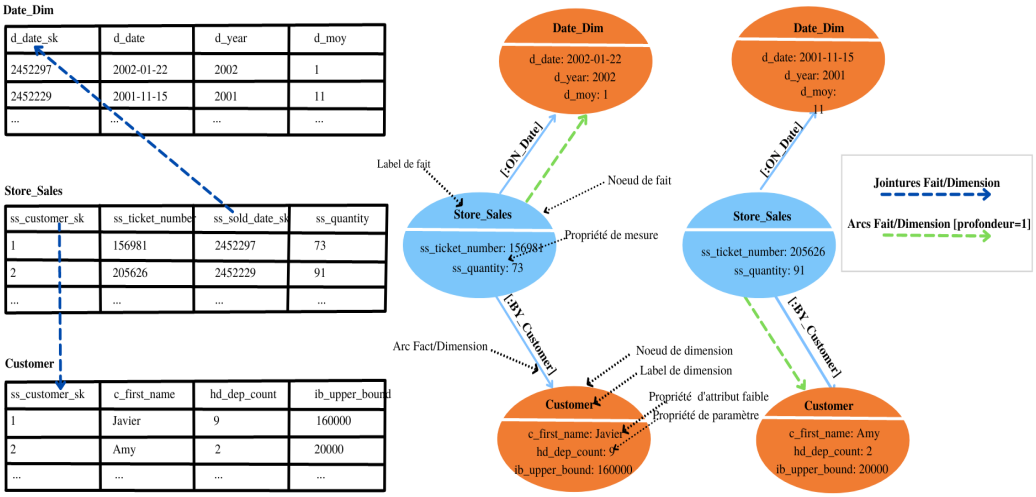


FIGURE 3.5 – MDM2G : Schéma en étoile

qui sont des tables relationnelles plus petites et normalisées. La normalisation en graphe consiste à créer plus de noeuds « normalisés » afin d'éviter la redondance de certaines données. Les noeuds normalisés sont reliés par des arcs. Nous proposons donc un schéma multidimensionnel en forme de flocon de neige basé sur des bases de données en graphes « Snow Graph ». Nous conservons les deux premières règles de transformation mentionnées ci-dessus et nous ajoutons deux autres règles qui permettent de représenter les hiérarchies dans les bases de données en graphe comme suit :

Transformation 4. Chaque dimension $D_i(Nom^{D_i}, A^{D_i}, H^{D_i}) \in D^{MDM}$ est transformée en un ensemble de noeud définis par ($Nom^{D_i{}^{MGD}}$, $A^{D_i{}^{MGD}}$, $H^{D_i{}^{MGD}}$) où :

— $Nom^{D_i{}^{MGD}}$ est le nom de la dimension D_i associé par la fonction λ^{MGD} comme étiquette aux différents noeuds de dimension N_{D_i},

— $A^{D_i{}^{MGD}}$ est un ensemble de paramètres et d'attributs faibles de dimension D_i. Chaque paramètre est transformé en un ensemble de noeuds qui définissent une hiérarchie. Chaque attribut faible d'un paramètre est transformé en une propriété du noeud.

— $H^{D_i{}^{MGD}}$ est un ensemble de noeuds qui représente la hiérarchie D_i.

Transformation 5. Chaque hiérarchie (Nom^{H_j}, $Param^{H_j}$, $Faible^{H_j}$) est transformée en un ensemble de noeuds liés ($Param^{H_j{}^{MGD}}$, $Faible^{H_j{}^{MGD}}$) où :

— $Param^{H_j{}^{MGD}}$ est un ensemble de noeuds de paramètres. La fonction λ^{MGD} associe à ces noeuds paramètres le nom de $Param^{H_j}$ comme étiquette. Par exemple, dans la Figure 3.6, le « customer », « Household_Demographics » et « Income_Band » se transforment en noeuds distincts.

— $Faible^{H_j{}^{MGD}}$ est un ensemble de propriétés associées aux noeuds de paramètres utilisant la fonction σ^{MGD}.

— Un arc est défini entre les noeuds de fait N_{F_i} et le paramètre connexe le plus bas $Param_k$ de chaque dimension à l'aide de la fonction ρ^{MGD}.

— Les arcs sont définis entre les paramètres voisins d'une même hiérarchie à l'aide de la fonction ρ^{MGD}. Par exemple, « Customer » et « Household_Demographics » sont liés en utilisant la relation « Current_HDemo », et « Household_Demographics » est connecté à « Income_Band » en utilisant le lien « Has ».

Dans ce cas, la profondeur, qui est le nombre d'arcs reliant le noeud de fait aux noeuds de paramètres, est supérieure à 2.

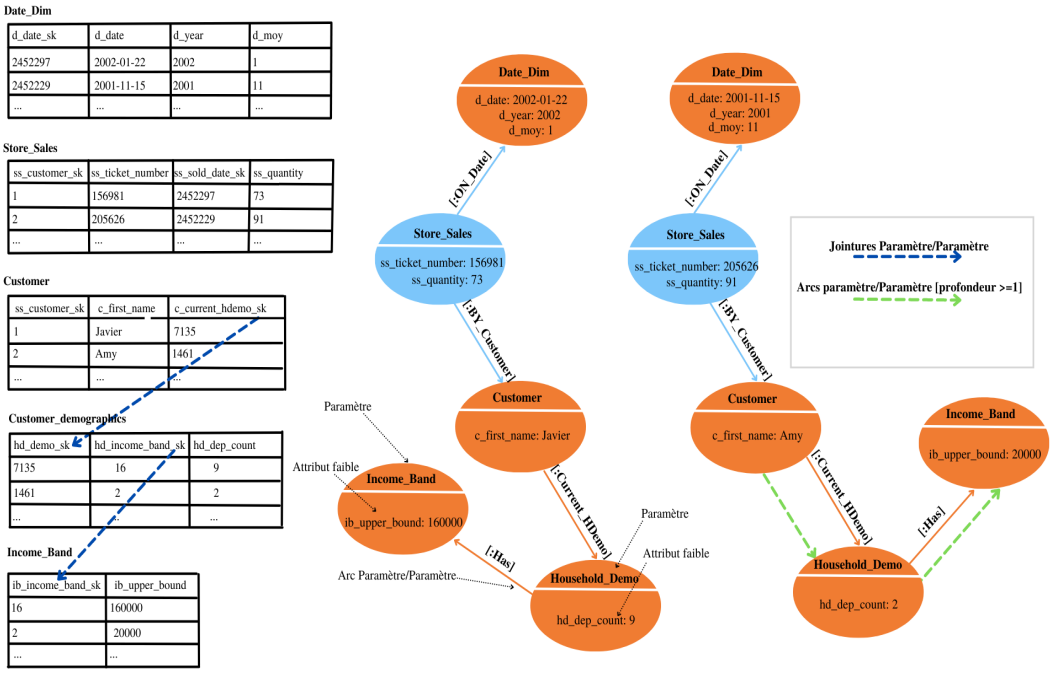


FIGURE 3.6 – MDM2G : Schéma en flocon

3.4 Expérimentations

Nos expériences ont trois objectifs principaux. Le premier consiste à valider notre approche en appliquant les règles de transformation proposées pour implanter des entrepôts de données graphes en forme d'étoile « Star Graph » et de flocon de neige « Snow Graph ». Le deuxième objectif est de comparer les performances des entrepôts de données en graphes proposés à des entrepôts de données relationnelles analogues en forme d'étoile « Star Relational » et « Snow Relational ». Les entrepôts relationnels sont mis en oeuvre en utilisant l'approche R-OLAP traditionnelle : comparaison intra-modèle. Le troisième objectif est d'évaluer l'efficacité des conceptions de données en étoile et en flocon de neige dans le contexte d'entrepôts en graphes pour déterminer si un modèle de données en graphe de type flocon de neige serait moins efficace qu'un modèle de

données en étoile : comparaison inter-modèles. Notre comparaison est faite en tenant compte du modèle de données, de la complexité des requêtes et de la taille des données.

Pour atteindre les objectifs mentionnés ci-dessus, nous utilisons Neo4j (version 3.5.0), une base de données orientée graphes écrite en Java. Elle est interrogée via le langage de requête Cypher. Nous utilisons Neo4j pour écrire nos règles de transformation et implémenter les entrepôts de données orientés graphes en forme d'étoile et de flocon de neige. Pour comparer ces derniers aux entrepôts de données relationnelles, nous utilisons MariaDB (version 10.1.38) comme base de données relationnelle. Ces entrepôts de données ont été déployés sous une machine virtuelle avec 32 Go de RAM et 8 To de disque. La machine virtuelle fonctionne sous le système d'exploitation 64 bits Ubuntu-18.04.01 LTS. Aucun index n'a été ajouté, dans aucun SGBD, car nous supposons que le filtrage peut concerner toutes les colonnes dans un contexte OLAP, où les utilisateurs font régulièrement de nouvelles requêtes. Les caches ont été vidés avant chaque requête afin de s'assurer que le temps d'exécution correspond à la première fois qu'une requête est posée. Dans un contexte OLAP, les utilisateurs exécutent de nouvelles requêtes plutôt que de répéter les mêmes. Le processus expérimental est résumé par la Figure 3.7.

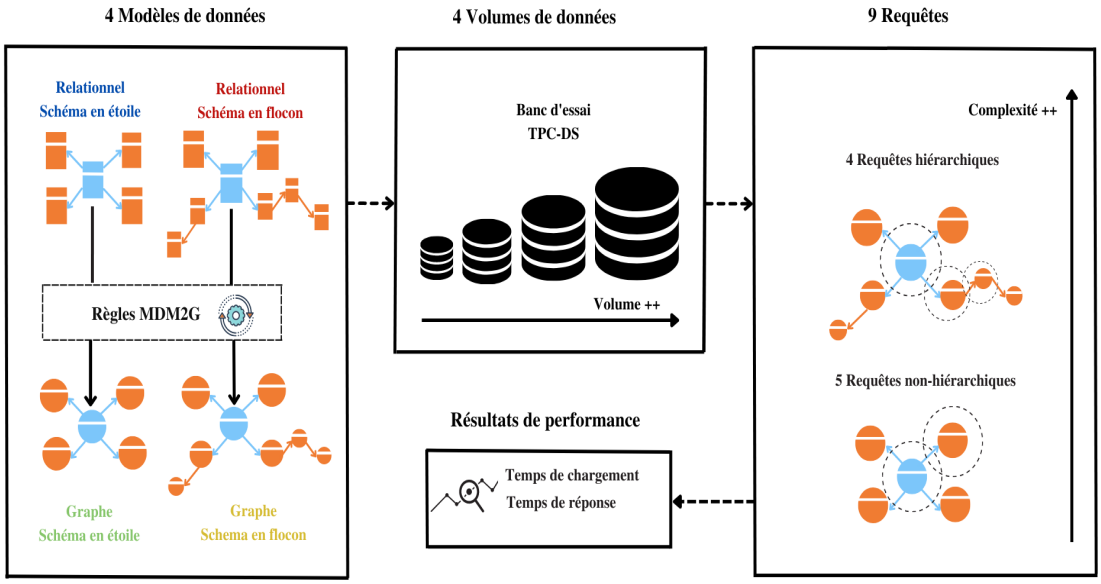


FIGURE 3.7 – Processus expérimental

3.4.1 Protocole expérimental

L'évaluation entre les entrepôts de données en graphes et les entrepôts de données relationnelles est basée principalement sur deux critères qui sont le temps de chargement et le temps de réponse. Ces critères ont été choisis pour décider objectivement quel SGBD est le plus efficace lorsque les données deviennent plus volumineuses ou que les requêtes deviennent plus complexes.

3.4.2 Banc d'essais TPC-DS

Les données ont été générées à partir du jeu de données de référence TPC-DS qui a été proposé pour évaluer les performances de systèmes d'aide à la décision Poess et al. (2007). TPC-DS englobe plusieurs schémas en flocon de neige qui modélisent les activités d'un fournisseur de produits vendant des marchandises via trois canaux de distribution : magasin, catalogue et Internet Nambiar and Poess (2006). Le modèle de données TPC-DS est composé de 7 tables de faits et de 17 tables de dimensions partagées. Chaque table de faits a un schéma en flocon de neige. Une caractéristique distinctive du modèle de données TPC-DS est le nombre de colonnes dans chaque table. Le nombre moyen de colonnes est de 18 Nambiar and Poess (2006), ce qui permet de générer des requêtes complexes avec des prédicats appliqués sur de nombreuses colonnes.

Dans ce travail, nous nous concentrons sur le schéma en flocon de neige le plus utilisé Chevalier et al. (2015a); Qu and Dessloch (2017) qui fait intervenir la table de faits $Store\_Sales$ du canal magasin et ses 10 dimensions : date, heure, magasin, promotion, article, client, démographie client, les données du ménage, la tranche de revenu et l'adresse du client. Le générateur de données TPC-DS nommé DSDGEN génère pour chaque entité (fait ou dimension) un fichier de données distinct. Ces fichiers de données sont mis à l'échelle au moyen de facteurs d'échelle (SF) qui représentent la taille des données en gigaoctets. Dans ce travail, nous avons généré des données selon quatre facteurs d'échelle différents SF1, SF3, SF5 et SF7 qui sont respectivement de 1 Go, 3 Go, 5 Go et 7 Go. La Table 3.1 montre le nombre de lignes générées pour chaque table du schéma en flocon choisi. Alors que la table de faits est mise à l'échelle linéairement avec le facteur d'échelle, les tables de dimension non statiques sont mises à l'échelle de manière sous-linéaire. Cependant, les données des tables de dimensions statiques telles que les dimensions de date et d'heure sont chargées une seule fois et ne sont pas mises à jour pendant la phase de maintenance des données Nambiar and Poess (2006).

Comme mentionné précédemment, TPC-DS implique plusieurs schémas en flocon de neige. Dans le schéma en flocon choisi, les données relatives aux clients sont décomposées hiérarchiquement en différentes tables liées par des relations un-à-plusieurs.

Tables	SF1	SF3	SF5	SF7
Store_Sales	2 880 404	8 639 377	14 400 052	20 159 325
Customer	100 000	188 000	277 000	366 000
Customer_Demographics	94 215	171 3197	249 626	322 762
Customer_Address	43 282	81 261	119 432	158 478
Date_Dim	73 049	73 049	73 049	73 049
Household_Demographics	7 200	7 200	7 200	7 200
Income_Band	20	20	20	20
Item	18 000	36 000	54 000	74 000
Promotion	300	344	388	433
Store	12	32	52	72
Time_Dim	86 400	86 400	86 400	86 400

TABLE 3.1 – Nombre de lignes par facteur d'échelle

Afin de comparer les performances de la conception en flocon de neige à la conception en étoile dans le contexte d'un entrepôt de données en graphe, nous avons dénormalisé la dimension client et ses tables associées en utilisant de nombreuses jointures gauches pour obtenir une grande table nommée « $Customer\_Details$ » qui contient tous les détails sur les clients (données démographiques des clients, tranche de revenu et adresse du client et les données des ménages). De plus, nous avons légèrement modifié le modèle de données de TPC-DS pour obtenir des structurations strictement en flocon de neige et d'étoiles respectivement, comme le montre la Figure 3.8. De fait, nous avons supprimé les colonnes qui référencent l'adresse du client, les données démographiques des clients et les données des ménages dans la table $Store\_Sales$. Par exemple, nous avons supprimé l'adresse du client au moment des transactions de vente, et nous ne gardons que l'adresse actuelle. De plus, nous avons supprimé les colonnes qui font référence à la dimension $Date\_Dim$ dans les tables $Store$, $Promotion$ et $Customer$. Nous avons également supprimé la référence de $Item$ dans le Tableau $Promotion$.

3.4.3 Requêtes

Le générateur de requêtes TPC-DSQGEN permet de générer des requêtes selon différents modèles. Dans nos expériences, nous avons sélectionné neuf requêtes différentes qui appartiennent au flocon de neige choisi (canal de vente en magasin). Ces neuf requêtes peuvent être regroupées en trois catégories principales telles que présentées dans le Tableau 3.2. La première catégorie est composée de cinq requêtes qui n'impliquent pas de hiérarchies. Il est donc évident que la table $Customer$ n'est pas utilisée dans ces requêtes. Plus précisément, toutes les tables interrogées sont directement liées à la table de faits $Store\_Sales$. Par exemple, la requête Q3, cf. le Tableau 3.3 , calcule le montant

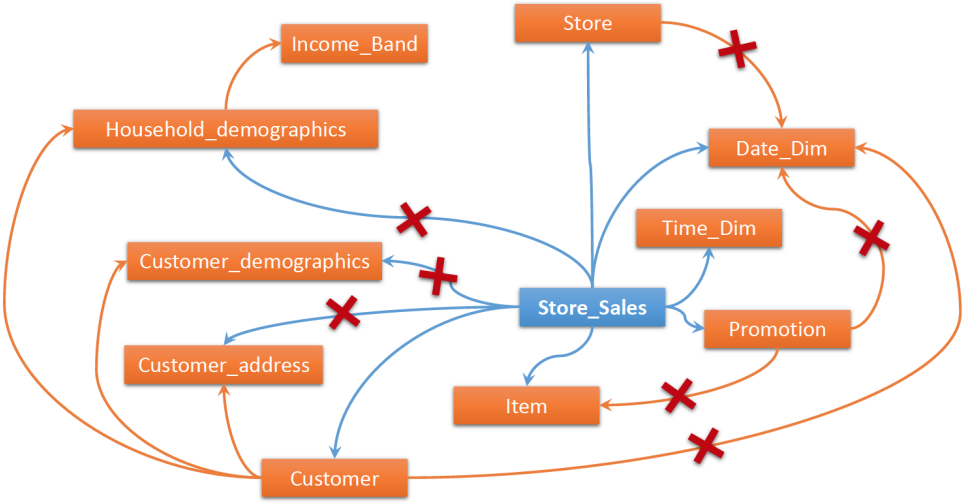


FIGURE 3.8 – Modèle de TPC-DS

Type	Nom de requête	Dimensions	Hiérarchies	Lignes retournées
Non-hiérarchiques	Q3	2	non	61
	Q28	0	non	1
	Q42	2	non	10
	Q52	2	non	100
	Q55	2	non	69
Hiérarchiques	Q7	7	oui	100
	Q27	5	oui	100
Hiérarchiques et cumulatives	Q13	6	oui	1
	Q48	5	oui	1

TABLE 3.2 – Caractéristiques des requêtes

total de la remise par marque d'article du fabricant 427 pour toutes les ventes qui ont eu lieu en novembre. Du point de vue de la base de données relationnelle, cette requête nécessite différentes jointures pour obtenir les données des tables $Item$, $Store\_Sales$ et $Date\_Dim$. Cependant, dans une base de données orientée graphes, toutes ces jointures sont remplacées par des arcs.

SQL	Cypher
SELECT dt.d_year,	
item.i_brand_id brand_id,	**MATCH** (d :date_dim{d_moy :11})
item.i_brand brand,	**WITH** d
SUM(ss_ext_discount_amt) sum_agg	**MATCH** (d)<-[:ON_Date]-(ss :Store_Sales)
FROM Date_dim dt,	**WITH** ss, d
Item i,	**MATCH** (i :item{i_manufact_id :427})
Store_Sales ss	**WITH** i, ss, d
WHERE dt.d_date_sk = ss.ss_sold_date_sk	**MATCH** (ss)-[:OF_Item]->(i)
AND ss.ss_item_sk = i.i_item_sk	**WITH** i, ss, d
AND i.i_manufact_id = 427	**RETURN** d.d_year,
AND dt.d_moy = 11	i.i_brand_id AS brand_id,
GROUP BY dt.d_year,	i.i_brand AS brand,
i.i_brand,	**SUM**(ss.ss_ext_discount_amt) as sum_agg
ii_brand_id	**ORDER BY** d.d_year,
ORDER BY dt.d_year,	sum_agg **DESC**,
sum_agg **DESC**,	brand_id
brand_id	**LIMIT** 100 ;
LIMIT 100 ;	

TABLE 3.3 – Q3 - Requête non-hiérarchique

Dans la même catégorie, on distingue également la requête Q28 qui est non seulement non hiérarchique mais aussi plate puisque seule la table de faits est utilisée dans cette requête et aucune dimension n'est interrogée. Comme le montre le Tableau 3.4, Q28 calcule le prix catalogue moyen, le nombre de prix catalogue et le nombre de prix catalogue distincts de six compartiments de vente différents du canal de vente en magasin. Chaque compartiment est défini par une gamme d'articles distincts et des informations sur le prix catalogue, le montant du coupon et le coût de gros. La catégorie de requêtes *non hiérarchique* est utilisée pour comparer les performances du modèle en graphe par rapport au modèle relationnel puisque leurs variantes en étoile et en flocon de neige sont identiques.

La deuxième catégorie se compose de 2 *requêtes hiérarchiques* qui sont exécutées pour répondre à des questions complexes. Ces requêtes passent par plus de cinq dimensions, jusqu'à une profondeur de 2, et incluent des hiérarchies et des agrégats. Par exemple, la requête Q7, détaillée dans le Tableau 3.5, calcule la quantité moyenne, le prix catalogue, la remise et le prix de vente des articles promotionnels vendus dans les magasins où la promotion n'est pas offerte par courrier ou lors d'un événement spé-

cial. Les résultats sont limités à un sexe, un statut matrimonial et un niveau d'éducation spécifiques.

SQL	Cypher
SELECT i_item_id, **AVG**(ss_quantity) agg1, **AVG**(ss_list_price) agg2, **AVG**(ss_coupon_amt) agg3, **AVG**(ss_sales_price) agg4 **FROM** Store_Sales, Customer, Customer_Demographics, Date_Dim, Item, Promotion **WHERE** ss_sold_date_sk = d_date_sk **AND** ss_item_sk = i_item_sk **AND** ss_customer_sk= c_customer_sk **AND** c_current_cdemo_sk= cd_demo_sk **AND** ss_promo_sk = p_promo_sk **AND** cd_gender = 'F' **AND** cd_marital_status = 'W' **AND** cd_education_status = '2 yr Degree' **AND** (p_channel_email = 'N' **OR** p_channel_event = 'N') **AND** d_year = 1998 **GROUP BY** i_item_id **ORDER BY** i_item_id **LIMIT** 100;	**MATCH** (cd :customer_demographics) **WHERE** cd_gender='F' **AND** cd_marital_status='W' **AND** cd_education_statu='2 yr Degree' **WITH** cd **MATCH** (cd)<-[:Current_CDemo]-(c :customer) **WITH** cd, c **MATCH** (c)<-[:BY_Customer]-(ss) **WITH** ss **MATCH** (p :promotion)<-[:ON_Promo]-(ss) **WHERE** p.p_channel_email='N' **OR** p.p_channel_event='N' **WITH** ss **MATCH** (d :date_dimd_year :1998)<-[:ON_Date]-(ss) **WITH** ss **MATCH** (ss)-[:OF_Item]->(i :item) **WITH** ss, i **RETURN** i.i_item_id, **AVG**(ss.ss_quantity) as agg1, **AVG**(ss.ss_list_price) as agg2, **AVG**(ss.ss_coupon_amt) as agg3, **AVG**(ss.ss_sales_price) as agg4 **ORDER BY** i.i_item_id **LIMIT** 100;

TABLE 3.5 – Q7 - Requête hiérarchique

La troisième catégorie contient des **requêtes hiérarchiques et cumulatives** qui sont non seulement très complexes mais aussi cumulatives. Ces requêtes renvoient une seule ligne agrégeant toutes les lignes ou tous les noeuds sélectionnés. Par exemple, la requête Q13, présentée dans le Tableau 3.6, calcule la quantité moyenne des articles vendus, le prix moyen des ventes, le coût moyen et le coût total pour les ventes en gros en magasin. Les valeurs des mesures sont calculées pour les différents types de clients, y compris la démographie de leur ménage, le prix de vente et différentes combinaisons d'états et de bénéfices des ventes pour une année donnée.

SQL	Cypher
SELECT AVG(ss_quantity),	**MATCH** (d :date_dim{d_year :2001})
AVG(ss_ext_sales_price),	**WITH** d
AVG(ss_ext_wholesale_cost),	**MATCH** (d)<-[:ON_Date]-(ss :Store_Sales)
SUM(ss_ext_wholesale_cost)	**WITH** ss
FROM Store_Sales,	**MATCH** (s :store)<-[:AT_Store]-(ss :Store_Sales)
Store,	**WITH** ss
Customer_Demographics,	**MATCH** (ss :Store_Sales)-[:BY_Customer]->(c)
Customer,	**WITH** ss, c
Household_Demographics,	**MATCH** (ca :customer_address) **WITH** ss, c, ca
Customer_Address,	**MATCH** (c)-[:Current_Addr]->(ca)
Date_Dim	**WITH** ss, c, ca
WHERE s_store_sk = ss_store_sk	**MATCH** (hd :household_demographics)
AND ss_sold_date_sk = d_date_sk	**WITH** hd, ss, c, ca
AND d_year = 2001	**MATCH** (c)-[:Current_HDemo]->(hd)
AND ((ss_customer_sk = c_customer_sk	**WITH** ss, ca, c, hd
AND c_current_cdemo_sk = cd_demo_sk	**MATCH** (cd :customer_demographics)
AND c_current_hdemo_sk = hd_demo_sk	**WITH** ss, ca, c, hd, cd
AND cd_marital_status = 'U'	**MATCH** (c)-[:Current_CDemo]->(cd)
AND cd_education_status = 'Advanced Degree'	**WITH** ss, ca, c, hd, cd
AND ss_sales_price **BETWEEN** 100.00 **AND** 150.00	**WHERE** ((cd.cd_marita_status = 'U'
AND hd_dep_count = 3)	**AND** cd.cd_education_status = 'Advanced Degree'
OR (c_current_hdemo_sk = hd_demo_sk	**AND** ss.ss_sales_price>=100.00
AND ss_customer_sk = c_customer_sk	**AND** ss.ss_sales_price<=150.00
AND c_current_cdemo_sk = cd_demo_sk	**AND** hd.hd_dep_count = 3)
AND cd_marital_status = 'M'	**OR** (cd.cd_marita_status = 'M'
AND cd_education_status = 'Primary'	**AND** cd.cd_education_status = 'Primary'
AND ss_sales_price BETWEEN 50.00 AND 100.00	**AND** ss.ss_sales_price>=50.00
AND hd_dep_count = 1)	**AND** ss.ss_sales_price<=100.00
OR (ss_customer_sk = c_customer_sk	**AND** hd.hd_dep_count = 1)
AND c_current_hdemo_sk = hd_demo_sk	OR (cd.cd_marita_status = 'D'
AND c_current_cdemo_sk = cd_demo_sk	**AND** cd.cd_education_status = 'Secondary'
AND cd_marital_status = 'D'	**AND** ss.ss_sales_price>= 150.00
AND cd_education_status = 'Secondary'	**AND** ss.ss_sales_price<=200.00
AND ss_sales_price **BETWEEN** 150.00 **AND** 200.00	**AND** hd.hd_dep_count = 1))
AND hd_dep_count = 1))	**AND** ((ca.ca_country = 'United States'
AND ((ca_address_sk = c_current_add_sk	**AND** ca.ca_state **IN** ['AZ','NE','IA']
AND ss_customer_sk = c_customer_sk	**AND** ss.ss_net_profit>=100
AND ca_country = 'United States'	**AND** ss.ss_net_profit<=200)
AND ca_state **IN** ('AZ','NE','IA')	**OR** (ca.ca_country ='United States'
AND ss_net_profit BETWEEN 100 **AND** 200)	**AND** ca.ca_state **IN** ['MS','CA','NV']
OR (ss_customer_sk = c_customer_sk	**AND** ss.ss_net_profit>=150
AND c_current_addr_sk = ca_address_sk	**AND** ss.ss_net_profit<=300)
AND ca_country = 'United States'	**OR** (ca.ca_country = 'United States'
AND ca_state **IN** ('MS','CA','NV')	**AND** ca.ca_state **IN** ['GA','TX','NJ']
AND ss_net_profit **BETWEEN** 150 **AND** 300)	**AND** ss.ss_net_profit>=50
OR (ca_address_sk = c_current_addr_sk	**AND** ss.ss_net_profit<=250))
AND ss_customer_sk = c_customer_sk	**RETURN AVG**(ss.ss_quantity),
AND ca_country = 'United States'	**AVG**(ss.ss_ext_sales_price),
AND ca_state **IN** ('GA','TX','NJ')	**AVG**(ss.ss_ext_wholesale_cost),
AND ss_net_profit **BETWEEN** 50 **AND** 250)) ;	**SUM**(ss.ss_ext_wholesale_cost) ;

TABLE 3.6 – Q13 - Requête hiérarchique et cumulative

Le but des expériences suivantes est de montrer que nous pouvons implanter un entrepôt de données orienté graphes en utilisant notre approche et lui appliquer une variété de requêtes. Nous évaluons les performances de chaque approche en fonction du temps d'exécution de l'ensemble des requêtes du TPC-DS que nous avons listées précédemment. Nous rapportons le temps d'exécution des requêtes adaptées pour le schéma en étoile et le schéma en flocon de neige. Comme Neo4j possède son propre langage de requête, les requêtes sont traduites dans le langage de requête Cypher.

3.5 Résultats

Dans cette section, nous rapportons les résultats de l'évaluation des performances des entrepôts de données relationnelles et les entrepôts orientés graphes du point de vue de deux métriques : le temps de chargement de données et le temps de réponse face à des requêtes analytiques.

3.5.1 Temps de chargement

Dans les bases de données orientées graphes, les relations entre les noeuds ont la même importance que les arcs Robinson et al. (2015). Alors que les bases de données relationnelles reposent sur des jointures pour répondre à des requêtes complexes, les bases de données orientées graphes stockent physiquement les liens entre les noeuds. Par conséquent, le temps de chargement de données dans des entrepôts orientées graphes est nettement plus long que le temps de chargement dans les entrepôts relationnels en raison du temps nécessaire pour créer des arcs entre les noeuds. Cela a été vérifié sur les expériences que nous avons menées. Comme le montre la Figure 3.9, le temps de chargement de l'entrepôt de données relationnelles est jusqu'à quatorze fois plus rapide que celui de l'entrepôt de données orienté graphes. De plus, contrairement aux bases de données relationnelles où le temps de création des entrepôts de données en étoile et en flocon est la même, la création d'un entrepôt de données en graphe avec un schéma en forme de flocon de neige prend plus de temps qu'un entrepôt de données en graphe avec un schéma en étoile. En effet, l'entrepôt de données en graphe en forme de flocon de neige nécessite la création de plus d'arcs, ce qui nécessite plus de temps.

3.5.2 Temps de réponse

On distingue les trois types de requêtes détaillés dans le Tableau 3.2.

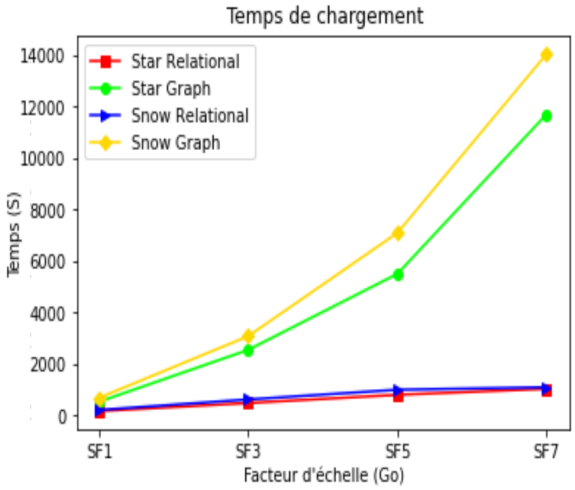


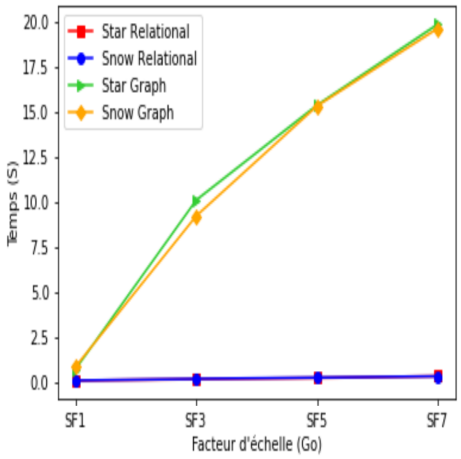
FIGURE 3.9 – Temps de chargement

Requêtes non hiérarchiques

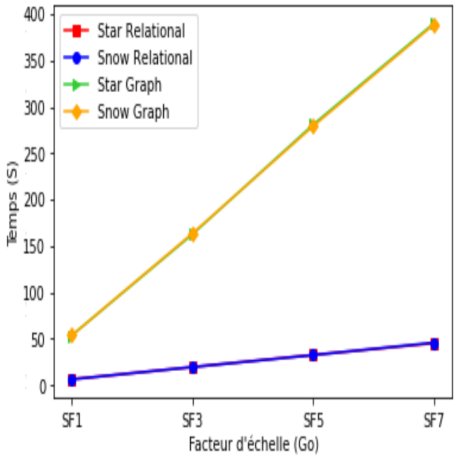
Les temps d'exécution des requêtes non hiérarchiques Q3, Q28, Q42, Q52 et Q55 sont représentés respectivement dans les Figures 3.10a, 3.10b, 3.10c, 3.10d et 3.10e. Les requêtes Q3, Q42, Q52 et Q55 sont très similaires. La requête Q3 ne fait pas de sélection sur l'année, contrairement aux autres requêtes, donc son temps d'exécution est plus long. Cependant, la croissance du temps d'exécution en fonction de la quantité de données à traiter reste similaire. La requête Q28 a une structure différente. Elle nécessite de lire plusieurs fois la table de fait, de garder en mémoire une grande quantité d'informations, d'où un temps d'exécution plus long. Comme mentionné précédemment dans la section 3.4.3, ces requêtes n'impliquent pas de hiérarchies. Plus précisément, la dimension client et toutes ses hiérarchies associées ne sont pas présentes dans ces requêtes. Donc, la requête est la même pour les schémas en étoile et en flocon de neige, et le temps de réponse des entrepôts de données normalisés et dénormalisés est le même. Ainsi, les courbes des entrepôts de données normalisées et dénormalisées se superposent. Ces expérimentations montrent que pour ces requêtes peu complexes, ne nécessitant pas de lier un grand nombre de données différentes qui nécessiteraient plusieurs jointures dans le modèle relationnel, les bases de données relationnelles sont plus performantes que les bases de données graphes (jusqu'à dix fois plus rapides).

Requêtes hiérarchiques

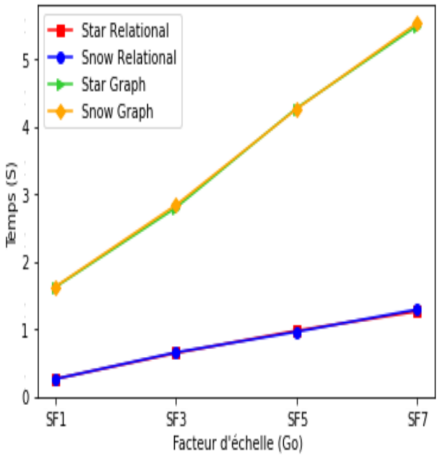
Les temps d'exécution des requêtes Q7 et Q27 sont représentés respectivement dans la Figure 3.11a et la Figure 3.11b. Les requêtes Q7 et Q27 sont similaires. Ce sont des requêtes hiérarchiques, d'une profondeur maximale de 2, et comprenant le calcul



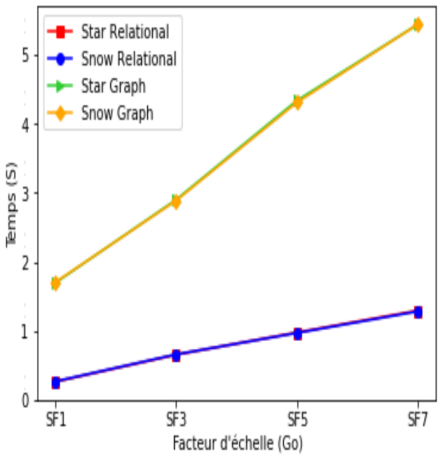
(a) Q3



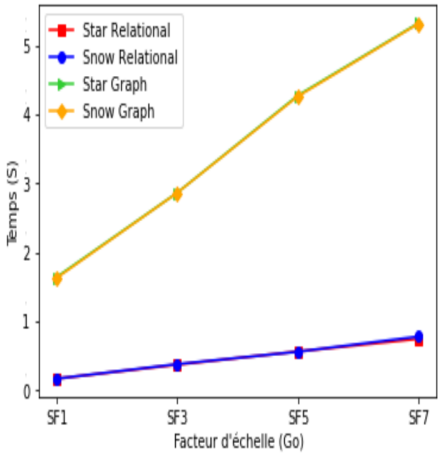
(b) Q28



(c) Q42



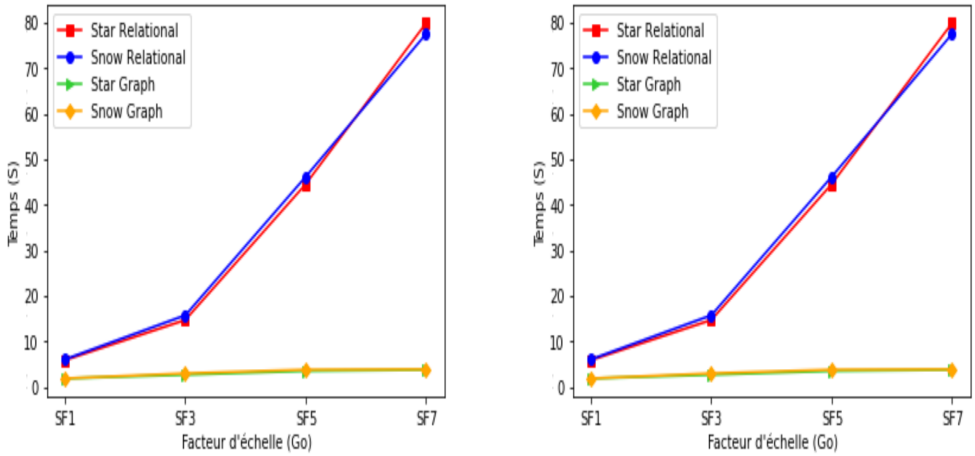
(d) Q52



(e) Q55

FIGURE 3.10 – Temps de réponse à des requêtes non-hiérarchiques

d'agrégats et un tri des résultats. Les délais d'exécution sont donc proches. Pour le deuxième type de requêtes, nos résultats montrent que le schéma en flocon nécessite plus de temps pour répondre aux requêtes, comparé au schéma en étoile dans le cas des entrepôts de données relationnelles. Cependant, pour les bases de données orientées graphes, les courbes des entrepôts de données orientés graphes en étoile et en flocon sont superposées. Étonnamment, ils ont les mêmes performances. De plus, l'entrepôt de données en graphes est nettement plus efficace que l'entrepôt de données relationnelles (jusqu'à plus de vingt fois plus rapide). Enfin, lorsque la taille des données augmente, l'entrepôt de données en graphe est plus efficace que le relationnel.

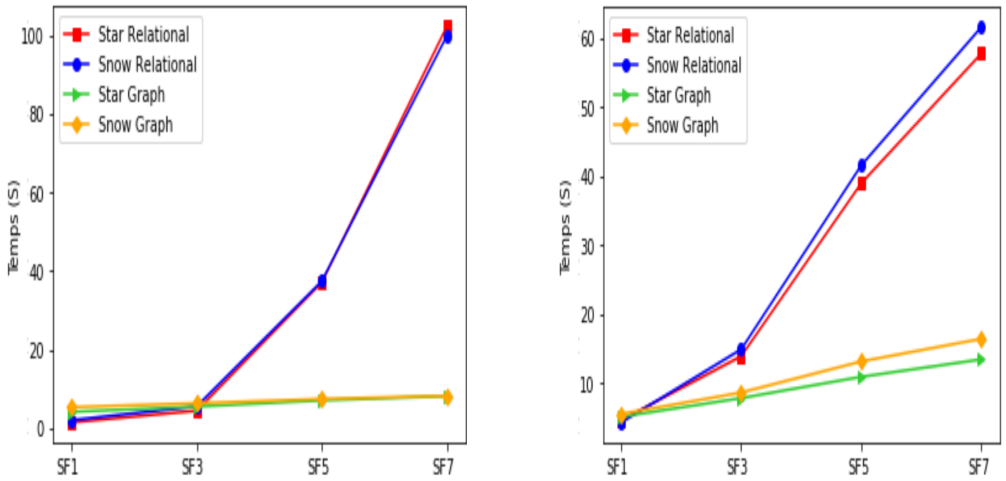


(a) Q7 (b) Q27

FIGURE 3.11 – Temps de réponse à des requêtes hiérarchiques

Requêtes hiérarchiques et cumulatives

Les temps d'exécution des requêtes Q13 et Q48 sont présentés respectivement aux Figures 3.12a et 3.12b. Les deux requêtes effectuent des calculs de quelques agrégats (moyenne et somme) sur les ventes pour une année donnée et pour une disjonction de conditions. Les conditions portent sur différentes sous-dimensions de hiérarchie Customer_details : Customer_Demographics, Household_Demographics et Customer_Address. Pour cette catégorie, les performances des entrepôts de données relationnelles et en graphes sont similaires lorsque la taille des données est petite. Cependant, pour des tailles de données plus importantes, les entrepôts de données en graphes répondent aux requêtes plus rapidement que les requêtes relationnelles (jusqu'à dix fois plus rapidement).



(a) Q13 (b) Q48

FIGURE 3.12 – Temps de réponse à des requêtes hiérarchiques et cumulatives

3.6 Discussion

Dans une implantation relationnelle d'un entrepôt de données, le principal inconvénient d'un schéma en flocon de neige est que les requêtes complexes exigent une navigation multidimensionnelle plus profonde dans les dimensions de l'entrepôt. Par conséquent, plusieurs jointures sont requises afin de répondre aux requêtes qui interrogent plusieurs tables de dimensions. Ces jointures augmentent le temps de réponse par rapport au schéma en étoile dans lequel les dimensions ne sont pas normalisées. De plus, les expérimentations que nous avons menées pour tester l'efficacité des entrepôts relationnelles et orientés graphes face à la première catégorie des requêtes non hiérarchiques, montrent qu'une implantation d'un schéma en étoile à l'aide d'une base de données relationnelle est plus efficace qu'une implantation similaire à l'aide d'une base de données orientée graphes. En ce qui concerne la deuxième et la troisième catégories de requêtes, celles impliquant de nombreuses dimensions ou hiérarchies, les entrepôts de données orientés graphes se sont avérés efficaces. En effet, les relations entre le fait et la dimension, et entre les attributs d'une même dimension (hiérarchies) sont implantées physiquement. Ainsi, en cas d'augmentation du volume à traiter, lorsque les requêtes parcourent plus de dimensions, les entrepôts de données orientés graphes sont plus efficaces, et plus robustes à l'augmentation de la complexité. Tandis que les bases de données relationnelles parcourent toutes les tables jusqu'à ce que les données correspondant aux critères de recherche soient trouvées, les bases de données en graphes parcourent uniquement les noeuds qui répondent aux critères. Les entrepôts de données orientés graphes en étoile et en flocon de neige sont efficaces selon le cas d'utilisation (données normalisées ou non

normalisées). De plus, les résultats montrent qu'il est possible d'envisager le schéma en flocon de neige pour les entrepôts de données orientés graphes afin d'ajouter facilement des données supplémentaires liées aux dimensions sans impact significatif sur le temps de réponse des requêtes complexes.

3.7 Conclusion

Dans ce chapitre, nous avons détaillé la première contribution de cette thèse qui concerne l'implantation des entrepôts de données en utilisant les bases de données NoSQL orientées graphes. Le but de cette étude est de déterminer si un entrepôt de données relationnelles ou un entrepôt de données orientées graphes serait plus efficace. Pour ce faire, nous avons proposé un ensemble de règles de transformation appelé MDM2G permettant de convertir un modèle de données multidimensionnel d'un entrepôt en un modèle des graphes à propriétés sans perte d'informations. Les deux schémas multidimensionnels en étoile et en flocon de neige sont pris en compte par nos règles pour permettre l'implantation de deux variantes d'entrepôts NoSQL orientés graphes : en étoile et en flocon de neige. Pour valider notre approche, nous avons mené des expérimentations en se basant sur les données générées à partir du banc d'essais décisionnel TPC-DS. Plusieurs volumes de données ont été générées : 1 Go, 3 Go, 5 Go et 7 Go. Notre processus expérimental a montré la faisabilité de mise en place des systèmes OLAP avec des bases de données en graphes en utilisant le SGBD Neo4j. Ce processus comprend la transformation des données, le chargement des données et l'exécution de requêtes analytiques complexes. L'ensemble du processus nous a permis de comparer les différentes approches entre elles. De plus, nous avons comparé les performances des entrepôts de données orientées graphes à des entrepôts de données relationnelles similaires. Nos résultats montrent que nos deux entrepôts de données orientés graphes proposés sont performants. Le schéma dénormalisé est plus efficace pour certaines requêtes. Les résultats des expérimentations montrent l'avantage de l'utilisation de la catégorie orientée graphes des bases données NoSQL pour implémenter des systèmes OLAP et répondre à des requêtes analytiques complexes.

Dans le chapitre suivant, nous détaillons la deuxième contribution de la thèse qui concerne l'utilisation des bases de données orientées graphes pour la prédiction des liens manquants et de leurs valeurs dans le cas des réseaux d'Interaction Protéine-Protéine IPP pondérés.

SQL	Cypher
	MATCH (ss :Store_Sales) **WHERE** ss.ss_quantity>=0
	AND ss.ss_quantity<=5 **AND** ((ss.ss_list_price>=18
	AND ss.ss_list_price<=18 + 10)
	OR (ss.ss_coupon_amt>=1939
	AND ss.ss_coupon_amt<=1939 + 1000)
SELECT *	**OR** (ss.ss_wholesale_cost>=34
FROM (**SELECT AVG**(ss_list_price) B1_LP,	**AND** ss.ss_wholesale_cost<=34 + 20))
Count(ss_list_price) B1_CNT,	**WITH AVG**(ss.ss_list_price) as B1_LP,
Count(**DISTINCT** ss_list_price) B1_CNTD	**Count**(ss.ss_list_price) as B1_CNT,
FROM Store_Sales	**Count**(**DISTINCT** ss.ss_list_price) as B1_CNTD
WHERE ss_quantity **BETWEEN** 0 **AND** 5	**MATCH** (ss :Store_Sales) **WHERE** ss.ss_quantity>=6
AND (ss_list_price **BETWEEN** 18	**AND** ss.ss_quantity<=10 **AND** (ss.ss_list_price>=1
AND 18 + 10	**AND** ss.ss_list_price<=1 + 10
OR ss_coupon_amt **BETWEEN** 1939	**OR** ss.ss_coupon_amt>=35
AND 1939 + 1000	**AND** ss.ss_coupon_amt<=35+1000
OR ss_wholesale_cost **BETWEEN** 34	**OR** ss.ss_wholesale_cost>=50
AND 34 + 20)) B1,	**AND** ss.ss_wholesale_cost<= 50+20)
(**SELECT AVG**(ss_list_price) B2_LP,	**WITH AVG**(ss.ss_list_price) as B2_LP,
Count(ss_list_price) B2_CNT,	**Count**(ss.ss_list_price) as B2_CNT,
Count(**DISTINCT** ss_list_price) B2_CNTD	**Count**(**DISTINCT** ss.ss_list_price) AS B2_CNTD,
FROM Store_Sales	B1_LP, B1_CNT, B1_CNTD
WHERE ss_quantity **BETWEEN** 6 **AND** 10	**MATCH** (ss :Store_Sales)
AND (ss_list_price **BETWEEN** 1 **AND** 1 + 10	**WHERE** ss.ss_quantity>=11 **AND** ss.ss_quantity<=15
OR ss_coupon_amt **BETWEEN** 35	**AND** (ss.ss_list_price>=91
AND 35 + 1000	**AND** ss.ss_list_price<=91 + 10
OR ss_wholesale_cost **BETWEEN** 50	**OR** ss.ss_coupon_amt>=1412
AND 50 + 20)) B2,	**AND** ss.ss_coupon_amt<=1412 + 1000
(**SELECT AVG**(ss_list_price) B3_LP,	**OR** ss.ss_wholesale_cost>=17
Count(ss_list_price) B3_CNT,	**AND** ss.ss_wholesale_cost<= 17 +20)
Count(**DISTINCT** ss_list_price) B3_CNTD	**WITH AVG**(ss.ss_list_price) as B3_LP,
FROM Store_Sales	**Count**(ss.ss_list_price) AS B3_CNT,
WHERE ss_quantity **BETWEEN** 11 **AND** 15	**Count**(**DISTINCT** ss.ss_list_price) AS B3_CNTD,
AND (ss_list_price **BETWEEN** 91	B2_LP, B2_CNT, B2_CNTD, B1_LP,
AND 91 + 10	B1_CNT, B1_CNTD
OR ss_coupon_amt **BETWEEN** 1412	**MATCH** (ss :Store_Sales) **WHERE** ss.ss_quantity>=16
AND 1412 + 1000	**AND** ss.ss_quantity<=20 **AND** (ss.ss_list_price>=9
OR ss_wholesale_cost **BETWEEN** 17	**AND** ss.ss_list_pric<=9 +10
AND 17 + 20)) B3,	**OR** ss.ss_coupon_amt>=5270
(**SELECT AVG**(ss_list_price) B4_LP,	**AND** ss.ss_coupon_amt<=5270 +1000
Count (ss_list_price) B4_CNT,	**OR** ss.ss_wholesale_cost>=29
Count(DISTINCT ss_list_price) B4_CNTD	**AND** ss.ss_wholesale_cost<=29+20)
FROM Store_Sales	**WITH AVG**(ss.ss_list_price) as B4_LP,
WHERE ss_quantity **BETWEEN** 16 **AND** 20	**Count** (ss.ss_list_price) as B4_CNT,
AND (ss_list_price **BETWEEN** 9	**Count** (DISTINCT ss.ss_list_price) as B4_CNTD,
AND 9 + 10	B3_LP, B3_CNT, B3_CNTD,
OR ss_coupon_amt **BETWEEN** 527 0	B2_LP, B2_CNT, B2_CNTD,
AND 5270 + 1000	B1_LP, B1_CNT, B1_CNTD
OR ss_wholesale_cost **BETWEEN** 29	**MATCH** (ss :Store_Sales)
AND 29 + 20)) B4,	**WHERE** ss.ss_quantity>=21 **AND** ss.ss_quantity<=25
(**SELECT AVG**(ss_list_price) B5_LP,	**AND** (ss.ss_list_price>=45 **AND** ss.ss_list_price<=45 +10
Count(ss_list_price) B5_CNT,	**OR** ss.ss_coupon_amt>=826
Count(DISTINCT ss_list_price) B5_CNTD	**AND** ss.ss_coupon_amt<=826 +1000
FROM Store_Sales ss	**OR** ss.ss_wholesale_cost>=5
WHERE ss_quantity **BETWEEN** 21 **AND** 25	**AND** ss.ss_wholesale_cost<=5+20)
AND (ss_list_price **BETWEEN** 45	**WITH AVG**(ss.ss_list_price) as B5_LP,
AND 45 + 10	**Count** (ss.ss_list_price) as B5_CNT,
OR ss_coupon_amt **BETWEEN** 826	**Count**(**DISTINCT** ss.ss_list_price) as B5_CNTD,
AND 826 + 1000	B4_LP, B4_CNT, B4_CNTD,
OR ss_wholesale_cost **BETWEEN** 5	B3_LP, B3_CNT, B3_CNTD, B2_LP, B2_CNT,
AND 5 + 20)) B5,	B2_CNTD, B1_LP, B1_CNT, B1_CNTD
(**SELECT AVG**(ss_list_price) B6_LP,	**MATCH** (ss :Store_Sales)
Count(ss_list_price) B6_CNT,	**WHERE** ss.ss_quantity>=26 **AND** ss.ss_quantity<=30
Count(**DISTINCT** ss_list_price) B6_CNTD	**AND** ((ss.ss_list_price>=174
FROM Store_Sales	**AND** ss.ss_list_price<=174 + 10)
WHERE ss_quantity **BETWEEN** 26 **AND** 30	**OR** (ss.ss_coupon_amt>=5548
AND (ss_list_price **BETWEEN** 174	**AND** ss.ss_coupon_amt<=5548 + 1000)
AND 174 + 10	**OR** (ss.ss_wholesale_cost>=42
OR ss_coupon_amt **BETWEEN** 5548	**AND** ss.ss_wholesale_cost<=42 + 20))
AND 5548 + 1000	**WITH AVG**(ss.ss_list_price) AS B6_LP,
OR ss_wholesale_cost **BETWEEN** 42	**Count** (ss.ss_list_price) AS B6_CNT,
AND 42 + 20)) B6	**Count**(**DISTINCT** ss.ss_list_price) as B6_CNTD,
LIMIT 100 ;	B5_LP, B5_CNT,
	B5_CNTD, B4_LP, B4_CNT,
	B4_CNTD, B3_LP, B3_CNT, B3_CNTD,
	B2_LP, B2_CNT, B2_CNTD,
	B1_LP, B1_CNT, B1_CNTD
	RETURN * **LIMIT** 100 ;

TABLE 3.4 – Q28 - Requête non-hiérarchique

Chapitre 4

Apprentissage automatique pour la prédiction des interactions protéine-protéine

Sommaire

4.1 Introduction

Les protéines sont des molécules essentielles qui remplissent une variété de fonctions vitales dans les organismes vivants, notamment en interagissant avec d'autres molécules telles que des protéines ou des acides nucléiques Kamada et al. (2014). Une interaction protéine-protéine (IPP) se produit lorsque deux protéines se lient physiquement pour accomplir une fonction biologique, telles que la catalyse de réactions métaboliques, la transcription de l'ADN, la réponse aux stimuli et le transport de molécules d'un endroit à un autre. L'étude de ces interactions constitue une source potentielle pour approfondir notre compréhension du fonctionnement cellulaire et élaborer de nouvelles approches thérapeutiques. Bien que de nombreuses techniques expérimentales à haut débit aient été développées pour détecter les IPPs dans les laboratoires, le nombre d'interactions protéiques inconnues dépasse le nombre d'interactions documentées expérimentalement Kovács et al. (2019); Chen et al. (2020). La détection expérimentale des IPPs est souvent coûteuse, difficile à mettre en place et prend beaucoup de temps. Afin de surmonter ce défi, les chercheurs ont opté à des outils informatiques pour restreindre le nombre de paires de protéines potentielles qui pourraient établir une future interaction. Les paires de protéines candidates identifiées à l'aide des outils informatiques sont ensuite vérifiées expérimentalement dans le laboratoire.

Un réseau d'IPP peut être modélisé intuitivement à l'aide d'un graphe non-orienté et non-étiqueté (non-pondéré). Dans ce cas, les nœuds représentent les protéines et les arcs reliant les nœuds représentent la présence d'une interaction physique entre les deux protéines correspondantes Mistry et al. (2017). Pour répondre à la problématique de prédiction d'interactions inconnues, les approches proposées dans la littérature utilisent souvent les informations relatives au graphe en vue de détecter l'existence ou non d'un futur lien potentiel. Dans ce contexte, plusieurs mesures de similarité (représentations) basées sur la topologie du graphe du réseau d'IPP ont été largement utilisées comme par exemple l'indice des voisins communs (CN) Barzel and Barabási (2013), l'indice Adamic-Adar (AA) Adamic and Adar (2003), l'indice d'allocation des ressources (RA) Liu and Lü (2010) et l'indice d'attachement préférentiel (PA) Lichtenwalter et al. (2010). Malgré la maturité de ces mesures, elles sont principalement définies pour des graphes non-étiquetés où tous les arcs possèdent la même importance (poids) dans le graphe. En d'autres termes, les mesures topologiques se basent sur les informations relatives à l'existence d'un chemin intermédiaire de longueur 2 entre les protéines non directement reliées pour identifier l'existence ou non d'un lien direct potentiel pour ces paires. Une mesure de valeur 0 indique que les deux nœuds ne sont pas proches, tandis que des valeurs plus élevées indiquent que les nœuds sont plus proches. Par exemple dans la

Figure 4.1, les deux protéines $CT3$ et $Q0142$ sont reliées par un seul chemin intermédiaire de longueur 2. L'ensemble de voisins en commun de $CT3$ et $Q0142$ contient uniquement la protéine $OYE2$. Ainsi la mesure CN est égale à 1.

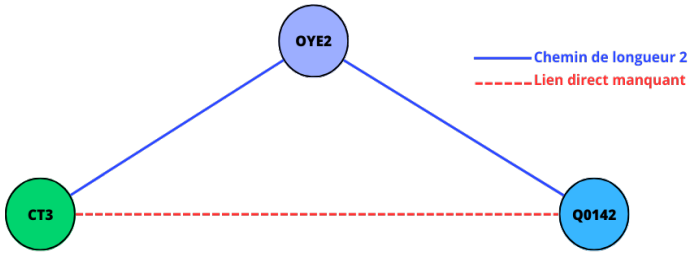


FIGURE 4.1 – Représentations pour la prédiction de l'existence d'un lien

La mesure CN repose sur le principe que deux individus qui partagent un ami en commun ont plus de probabilités d'être amis que ceux qui n'ont aucun ami en commun. Plusieurs mesures de similarité topologiques s'appuient sur cette même idée et attribuent une faible mesure aux nœuds reliés par un nombre limité de voisins communs. Toutefois, bien que la mesure CN soit efficace dans de nombreux domaines tels que les réseaux sociaux et les réseaux collaboratifs, elle ne fonctionne pas de manière optimale dans le contexte des données biologiques Kovács et al. (2019). En biologie, un nombre faible de connexions en commun entre deux protéines ne garantit pas l'absence d'interaction. Par exemple, la Figure 4.2 illustre des données provenant de la base de données biologique STRING version 11.0 Szklarczyk et al. (2021), qui contient des informations sur les interactions protéine-protéine. Les protéines $MMS4$ et $POP3$ n'ont aucun voisin en commun dans cette version, ce qui donne une valeur de CN égale à zéro et suggère qu'il n'existe pas de lien potentiel entre elles. Cependant, ces deux protéines sont reliées dans la même version par 720 chemins intermédiaires de longueur 3. Dans la version 11.5 de la base de données STRING, un lien direct entre $MMS4$ et $POP3$ a été ajouté.

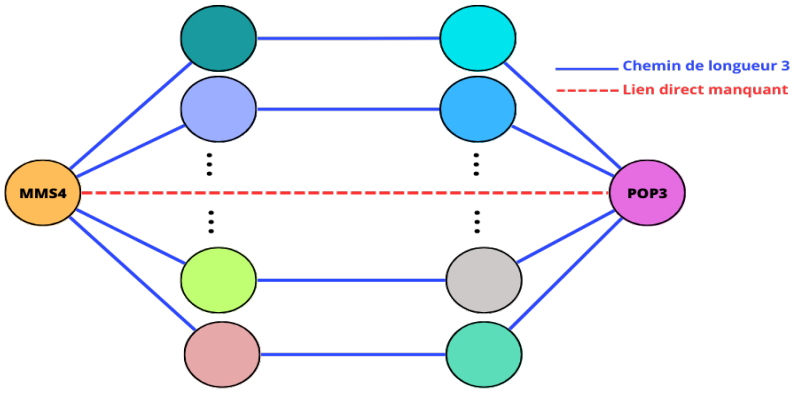


FIGURE 4.2 – Extrait de la base de données STRING (V11.0)

Comme la plupart des mesures de similarité topologiques sont basées sur l'existence de voisins en commun (chemins de longueur 2) pour identifier la possibilité d'interagir, elles ne sont malheureusement pas appropriées au contexte biologique. Récemment, une nouvelle mesure nommée L3 Kovács et al. (2019) a été proposée pour tenir compte des chemins de longueurs supérieures ou égales à 3. L3 est calculée en évaluant la présence de plusieurs chemins de longueur 3 entre les protéines non directement liées (Figure 4.3). Cette mesure a été testée sur diverses bases de données biologiques et s'est avérée plus précise que les mesures de similarité classiques. Selon L3, deux protéines sans voisins communs peuvent interagir si elles ont plusieurs chemins de longueur 3 qui les relient. Cependant, L3 n'est pas adaptée pour les graphes étiquetés.

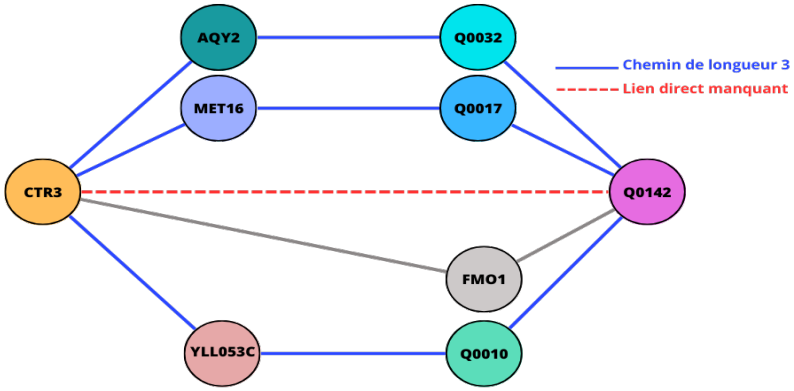


FIGURE 4.3 – Algorithme L3

À l'heure actuelle, les IPPs peuvent être collectées à partir de plusieurs sources en ligne ainsi qu'à partir d'expérimentations menées dans les laboratoires. BioGRID Oughtred et al. (2021), HI-II-14Rolland et al. (2014), UniProt Consortium (2015) et STRING Szklarczyk et al. (2021) sont des exemples de bases de données en ligne qui contiennent des IPPs. Malgré la diversité des sources de données des IPPs, ces interactions ne sont pas toutes fiables. En effait, les erreurs dues aux expérimentations sont inévitables et peuvent induire la détection d'interactions faussement positives, c'est-à-dire qui ne se produisent pas réellement dans la cellule. Ainsi, l'utilisation de toutes les interactions prédites avec le même niveau de confiance peut influer sur les résultats Paul and Anand (2020). Dans ce contexte, l'ajout d'étiquettes (scores) sur les arcs d'un graphe IPP qui représentent la confiance dans ces interactions est de grand intérêt. L'utilisation de ces étiquettes dans l'apprentissage peut aider à prédire les valeurs portées par les arcs donc la confiance en ces arcs.

Dans cette thèse, nous proposons une nouvelle approche qui utilise les étiquettes des graphes IPP pondérés pour prédire de nouvelles interactions pertinentes ainsi que leurs étiquettes. Pour capturer les informations portées par les arcs du réseau IPP, nous

proposons différentes représentations basées sur l'agrégation qui capturent et agrègent les informations de confiance portées par les étiquettes sur les chemins intermédiaires de longueur 3 entre les protéines non directement liées. En outre, nous proposons une nouvelle approche d'apprentissage automatique qui permet de combiner au mieux les informations apportées par les valeurs des mesures liées à la topologie, la mesure L3 et nos représentations basées sur l'agrégation afin de prédire de nouvelles interactions et leurs poids avec le minimum d'erreur.

Le reste de ce chapitre est organisé comme suit : dans la section 4.2, nous détaillons les mesures les plus utilisées dans la littérature pour la prédiction de l'existence ou non d'un lien potentiel entre les paires reliées par des chemins intermédiaires et qui ne sont pas directement liées. Dans la section 4.3, nous introduisons nos modélisations basées sur les graphes que nous avons proposées pour les IPPs pondérés. La section 4.4 a pour objet d'introduire nos différentes représentations qui s'appuient sur l'agrégation et notre approche de prédiction basée sur l'apprentissage automatique. La section 4.5 détaille les expérimentations menées pour valider nos propositions. Dans la section 4.6, nous détaillons les résultats de nos expérimentations. Dans la section 4.8, nous clôturons ce chapitre et dressons le bilan de nos principales contributions.

4.2 État de l'art

La prédiction d'un lien entre deux nœuds non directement liés consiste à prédire si un lien est manquant ou susceptible d'apparaître dans le futur. La prédiction de liens est un sujet préoccupant dans différentes disciplines pour lesquelles l'analyse des arcs est non triviale et peut apporter de la connaissance. Par exemple, en médecine, la prédiction de liens entre les symptômes d'une maladie et les médicaments capables de la traiter s'est avérée d'un grand intérêt pour réutiliser certains médicaments pour le traitement de nouvelles maladies. Himmelstein et al. (2017). Dans le cas des réseaux sociaux, la prédiction de liens entre amis est au cœur des applications de recommandation pour identifier les produits susceptibles d'intéresser un utilisateur. En bioinformatique, depuis plusieurs années, la prédiction de liens est le sujet de plusieurs travaux de recherche afin de proposer aux biologistes les protéines qui peuvent interagir afin de minimiser le coût et le temps des expérimentations faites en laboratoires.

Dans ce travail, nous nous intéressons à la prédiction des liens dans le domaine de la bioinformatique. Différentes approches ont été proposées dans la littérature pour résoudre le problème de prédiction de liens potentiels et suggérer aux biologistes de nouvelles interactions à vérifier expérimentalement. Ces approches peuvent être classées principalement en trois catégories : approches basées sur la similarité qui étudient un

ensemble de caractéristiques pour trouver des nœuds similaires, approches statistiques probabilistes qui optimisent un ensemble de paramètres pour établir un modèle qui peut calculer la probabilité de former un lien et approches qui combinent certaines mesures de similarité pour construire un modèle d'apprentissage.

4.2.1 Méthodes de détection expérimentale

Les protéines sont des macromolécules formées par des chaînes d'acides aminés et qui remplissent plusieurs fonctions vitales au sein des organismes. Pour accomplir leurs fonctions, plus de 80% des protéines interagissent avec d'autres protéines Berggård et al. (2007). Les IPPs correspondent à des liens physiques établis au sein de deux ou plusieurs molécules. L'ensemble les IPPs établies dans un organisme, une cellule ou un tissu est appelé interactome. En général, un interactome est modélisé intuitivement sous la forme d'un graphe non-orienté où les nœuds représentent les protéines et les arêtes reliant les nœuds représentent l'existence d'une interaction entre les protéines. L'analyse des réseaux d'IPP fournit des connaissances pertinentes sur le mécanisme moléculaire des organismes et les fonctions des protéines. Au fil des années, grâce au développement des techniques à haut débit, différentes méthodes expérimentales de détection d'interactions protéine-protéine ont été développées comme le criblage à deux hybrides connu sous le nom « Yeast-two hybrid », la purification par affinité en tandem connu sous le nom « Y TAP-tag », le transfert d'énergie entre molécules fluorescentes appelée « FRET », ou l'identification des complexes protéiques par spectrométrie de masse nommée « MS-PCI » Yang et al. (2020).

Les méthodes expérimentales qui permettent d'identifier de nouvelles IPPs peuvent être regroupées en fonction de deux caractéristiques principales : le type de méthode utilisée et le nombre d'interactions détectées. Les méthodes de détection de type binaire, telles que la méthode Yeast-two hybrid et la méthode FRET, permettent de calculer une interaction directe entre deux protéines. D'autres méthodes, telles que la cristallographie aux rayons X, permettent d'identifier des complexes protéiques. En ce qui concerne le nombre d'interactions détectées, certaines méthodes sont capables de trouver un petit nombre d'interactions, appelées « méthodes à petite échelle », tandis que d'autres sont capables de détecter un grand nombre d'interactions, appelées « méthodes à grande échelle »Brouard (2013).

Malgré la multiplicité des techniques expérimentales qui permettent d'identifier les liens manquants, ces techniques sont très coûteuses et nécessitent beaucoup du temps. Dans ce cadre, plusieurs travaux de recherche ont proposé des approches permettant de chercher les protéines les plus susceptibles d'interagir afin de réduire l'ensemble des

paires de protéines à tester expérimentalement.

4.2.2 Mesures de similarité topologiques

Les réseaux d'IPP sont caractérisés par différentes propriétés topologiques relatives au graphe des IPPs. Chaque nœud du graphe possède un degré X qui correspond au nombre de nœuds adjacents à ce nœud. Les approches qui reposent sur l'analyse de la topologie des réseaux d'IPP représentent en premier lieu l'information fournie par le graphe, puis cette représentation est utilisée pour identifier les liens manquants les plus probables.

Les approches basées sur la similarité peuvent être classées en deux catégories : approches de calcul de similarité basées sur les attributs de nœuds Wang et al. (2016) et approches de calcul de similarité basées sur la topologie du réseau Srilatha and Manjula (2016). Généralement, les approches de similarité basées sur la structure du réseau sont de trois types, à savoir les approches locales, les approches globales et les approches quasi-locales Mutlu and Oghaz (2019). Ici, nous prêtons principalement attention aux approches locales qui calculent les scores de similarité de deux nœuds n'ayant pas un lien direct en fonction des caractéristiques structurelles des nœuds voisins telles que les mesures de l'indice des voisins communs (CN), l'indice Adamic-Adar (AA), l'indice d'allocation des ressources (RA) et l'indice d'attachement préférentiel (PA).

Voisins communs (CN)

CN est l'un des indices de calcul de similarité les plus répandus dans le domaine de prédiction de liens grâce à sa simplicité et sa grande efficacité pour certains domaines tels que les graphes de collaborations scientifiques Newman (2001) et les réseaux sociaux Yao et al. (2016). L'idée derrière CN est que la probabilité d'être lié pour deux nœuds à l'avenir est affectée par le nombre de leurs nœuds communs, c'est-à-dire que deux nœuds établiront très probablement un lien s'ils ont plus de nœuds partagés. Cette mesure est définie comme suit 4.1 :

$$CN(X,Y) = |N(X) \cap N(Y)| \tag{4.1}$$

où N(x) est l'ensemble des nœuds adjacents au nœud X, et N(Y) est l'ensemble des nœuds adjacents au nœud Y.

Coefficient de Jaccard (CJ)

Contrairement à l'indice CN non normalisé, CJ ne prend pas seulement en compte le nombre de nœuds en commun, mais le normalise également en considérant l'ensemble total du nombre de voisins partagés et non partagés. La formulation de cette mesure proposée par Jaccard (1901) est la suivante 4.2 :

$$CJ(X,Y) = \frac{CN(X,Y)}{TN(X,Y)} \tag{4.2}$$

où TN(X,Y) est le nombre total de voisins de X et de Y défini comme suit 4.3 :

$$TN(X,Y) = |N(X) \cup N(Y)| \tag{4.3}$$

Adamic Adar (AA)

L'indice AA proposé par Adamic and Adar (2003) donne plus de poids aux voisins communs relativement moins nombreux. Dans la formule de AA, les voisins partagés de deux nœuds sont pénalisés par le logarithme de leurs degrés comme suit 4.4 :

$$AA(X,Y) = \sum_{u \in N(X) \cap N(Y)} \frac{1}{\log|N(u)|} \tag{4.4}$$

Attachement préférentiel (PA)

Il a été montré par Barabási and Albert (1999) que les nouveaux nœuds rejoignant le réseau sont très probablement liés à un nœud existant qui a des degrés plus élevés plutôt qu'à un nœud qui a des degrés inférieurs. Ainsi, l'indice PA a été proposé 4.5 :

$$PA(X,Y) = |N(X)| * |N(Y)| \tag{4.5}$$

Ressource Allocation (RA)

Un indice RA, très similaire à AA, a été proposé par Zhou et al. (2009). Certaines études montrent que les performances de RA et AA sont très proches lorsque le degré moyen du réseau est faible. Cependant, PA est de meilleure performance lorsque le degré moyen du réseau est élevé Liu et al. (2019); Wang et al. (2015). La mesure RA est définie comme suit 4.6 :

$$RA(X,Y) = \sum_{u \in N(X) \cap N(Y)} \frac{1}{|N(u)|} \tag{4.6}$$

Certains travaux de recherche ont proposé des approches qui permettent de prédire les liens manquants en combinant plusieurs mesures basées sur la topologie du graphe. Ces approches utilisent les valeurs des mesures topologiques calculées en se basant sur les informations relatives à l'existence ou non d'un chemin intermédiaire de longueur 2. Ces valeurs sont fournies par la suite à un algorithme d'apprentissage supervisé pour prédire l'existence ou non d'un lien potentiel manquant Al Hasan et al. (2006); Gao et al. (2015).

Chemins de longueur trois (L3)

L3 est une nouvelle mesure de similarité introduite par Kovács et al. (2019) pour prédire les liens manquants dans les réseaux d'IPP. Dans cette étude, les auteurs montrent que, contrairement à sa performance dans le domaine des réseaux sociaux, le coefficient de jaccard échoue dans le cas de la détection de nouvelles interactions protéiques. En effet, les protéines interagissent non pas si elles sont similaires les unes aux autres, mais si l'une d'entre elles est similaire aux protéines voisines de l'autre. La formule de la mesure L3 est la suivante 4.7 :

$$P(X,Y) = \sum_{U,V} \frac{a_{XU} * a_{UV} * a_{VY}}{\sqrt{|N(U)| * |N(V)|}} \tag{4.7}$$

où $a_{XU} = 1$ si les protéines X et U interagissent, et zéro sinon.

Par exemple, les deux protéines $CTR3$ et $Q0142$ (Figure 4.3) sont reliées par un seul chemin intermédiaire de longueur 2 et 9 chemins intermédiaires de longueur 3. L3 permet de représenter l'existence de ces 9 chemins. L3 a été testé sur plusieurs bases de données telles que HI-II-14 Rolland et al. (2014) et la partie consacrée à l'organisme levure dans la base de données STRING Szklarczyk et al. (2021). Puisqu'une mesure de similarité donne une valeur pour une paire de protéines non connectées par un lien direct, cette valeur a été utilisée directement pour faire des prédictions Kovács et al. (2019), typiquement en considérant les k meilleurs scores prédits. Remarquons que dans ce cas il n'y a pas de construction d'un modèle à l'aide d'un jeu d'entraînement, ne serait-ce que le réglage d'un seuil. Les résultats montrent que L3 surpasse toutes les approches locales de calcul de similarité en matière de précision.

4.2.3 Modèles probabilistes

Afin d'apporter une solution à la problématique préoccupante de la prédiction de liens, certains chercheurs se sont intéressés à l'utilisation de modèles probabilistes qui s'appuient principalement sur des données statistiques pour la prédiction de liens. Les

approches probabilistes construisent un modèle qui s'adapte à la structure du réseau. La probabilité d'existence d'un lien potentiel manquant entre deux nœuds est estimée à l'aide d'une probabilité conditionnelle. Au fil des années, de nombreux modèles ont été proposés, notamment le modèle probabiliste basé sur les graphes, qui considère les nœuds du graphe comme des variables aléatoires et les liens manquants à prédire comme des indépendances conditionnelles. Les approches probabilistes peuvent être regroupées en deux principales catégories : les réseaux bayésiens et les réseaux de Markov. Dans le domaine de la biologie, des classifieurs bayésiens naïfs ont été utilisés pour prédire les interactions protéine-protéine manquantes dans l'interactome de l'organisme levure Park and Bader (2011) et l'interactome de l'homme Wißing et al. (2019). Dans Birlutiu et al. (2012), une approche basée sur les réseaux bayésiens a été proposée pour pour la prédiction des IPPs. Cette approche combine les informations relative à la topologie du réseau et les informations relatives aux protéines Birlutiu et al. (2012). Les réseaux bayésiens ont été aussi utilisés dans le cadre non supervisé pour inférer des réseaux de régulation en cherchant à détecter des relations de causalité entre des données d'expression de gènes Brouard (2013).

4.3 Modélisation des réseaux d'IPP pondérés

Avant de s'intéresser à nos principales contributions qui concernent la prédiction des scores des liens manquants dans un IPP pondéré, nous avons porté un grand intérêt à la modélisation de ces réseaux. Motivés par nos résultats du chapitre précédent 3.1 et qui montrent que la normalisation de la base de données n'est pas coûteuse lors de l'interrogation des bases de données orientées graphes, nous avons proposé deux types de modélisations d'un réseau IPP : modélisation dénormalisée et modélisation normalisée. Dans les deux cas, le graphe des IPPs est non-orienté et présente des pondérations (étiquettes sur les arcs). Les modélisations que nous avons proposées sont basées sur celui de graphes à propriétés. Dans cette section, nous introduisons ces deux modèles.

4.3.1 Modélisation dénormalisée

La version dénormalisée des IPPs est un graphe étiqueté et pondéré. Les nœuds représentent les protéines et portent comme propriétés les caractéristiques des protéines sous la forme d'un couple clé-valeur. En ce qui concerne les arcs, ils possèdent la même étiquette « INTERACTS_**WITH** ». Comme l'illustre la Figure 4.4, chaque arc contient toutes les caractéristiques de l'interaction sous la forme de couples clé-valeur. Ces caractéristiques peuvent être collectées de différentes sources comme les expérimenta-

tions menées aux laboratoires, les données ajoutées par un expert ou les données qui proviennent des bases de données en ligne. La clé indique la source ou le type de caractéristique et la valeur indique son poids qui représente la confiance en cette interaction suivant cette source.

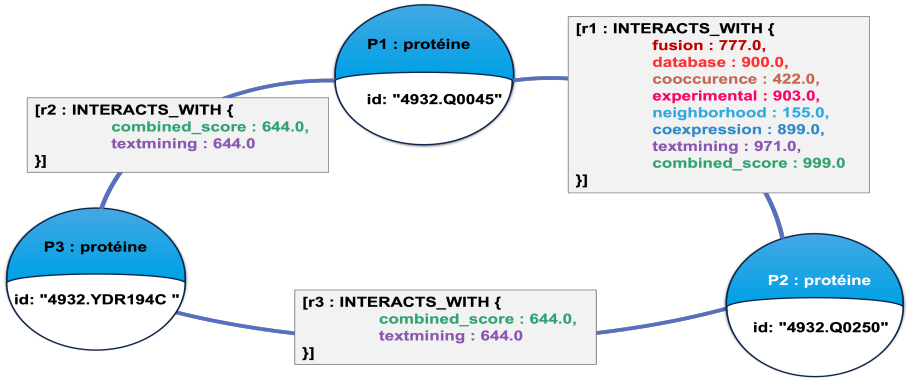


FIGURE 4.4 – Modélisation dénormalisée

4.3.2 Modélisation normalisée

Contrairement à la version dénormalisée des réseaux d'IPP, la version normalisée est un graphe ayant des arcs hétérogènes. Ainsi, les arcs possèdent des étiquettes différentes qui représentent la source des caractéristiques stockées sur l'arc comme l'illustre la Figure 4.5. Les arcs avec une étiquette « EXPERIMENTAL » concernent les informations collectées à travers les expérimentations effectuées dans les laboratoires. Les arcs avec une étiquette « DATABASE » modélisent les informations ajoutées par un expert. Bien que ce graphe soit complexe et puisse engendrer l'occupation de plus d'espace mémoire, il pourrait offrir une meilleure performance dans le cas de l'intérêt à une source particulière d'information comme celles des expérimentations.

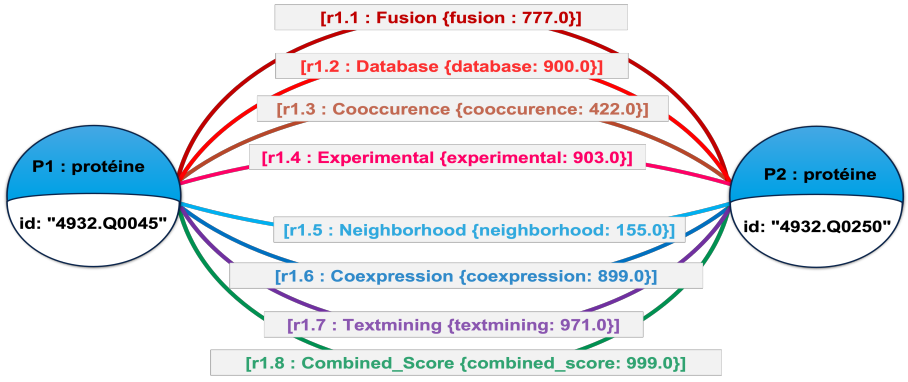


FIGURE 4.5 – Modélisation normalisée

Les deux modélisations proposées dans cette section seront validées expérimenta-

lement et testées face à différentes requêtes qui portent sur différentes longueurs de chemins.

4.4 Apprentissage automatique basé sur l'agrégation des poids

4.4.1 Représentation d'une paire de protéines par agrégation des valeurs portées sur les chemins intermédiaires

La majorité des mesures de similarité proposées dans littérature sont définies pour des graphes non-pondérés. En outre, les mesures proposées sont souvent définies pour des chemins de longueur 2. Pour répondre aux besoins émergent de faire des prédictions à partir d'un IPP pondéré et en prenant en considération les chemins de longueur 3, nous proposons dans ce travail une nouvelle représentation basée sur l'agrégation. Comme son nom l'indique, notre représentation par agrégation est basée sur des fonctions d'agrégation. En particulier, nous utilisons les fonctions MIN, AVG et MAX pour résumer les valeurs portées par les chemins intermédiaires de longueur L entre chaque couple de protéines. L'agrégation des chemins intermédiaires se fait en deux étapes :

— Agrégation des valeurs de chaque chemin intermédiaire : première agrégation pour résumer toutes les valeurs d'un chemin intermédiaire en une seule valeur. Par exemple, la fonction MIN permet de résumer toutes les valeurs sur chaque chemin intermédiaire de longueur L en un seul score. Ce score représente la valeur minimale sur le chemin intermédiaire. Le résultat de cette étape est une liste qui contient les minimums des scores portés par tous les chemins intermédiaires de longueur L.

— Agrégation des valeurs obtenues par la première agrégation : deuxième agrégation pour résumer les valeurs obtenues pour chaque chemin intermédiaire. L'utilisation de la fonction AVG par exemple permet de calculer la moyenne de toutes les valeurs minimales obtenues avec MIN. Dans ce cas, les données des différents chemins intermédiaires de longueur L sont résumées à l'aide de la représentation $AVGMIN$ en un seul score α comme illustré dans la Figure 4.6.
La formulation de la mesure AVGMIN est la suivante 4.8 :

$$AVGMIN(X,Y) = AVG_{\forall \text{chemin } c \text{ de longueur } =3 \text{ entre X et Y}}(MIN_{\forall \text{arc } a \text{ du chemin } c}(score(a)) \tag{4.8}$$

Notre approche de représentation basée sur l'agrégation permet de définir 9 combi-

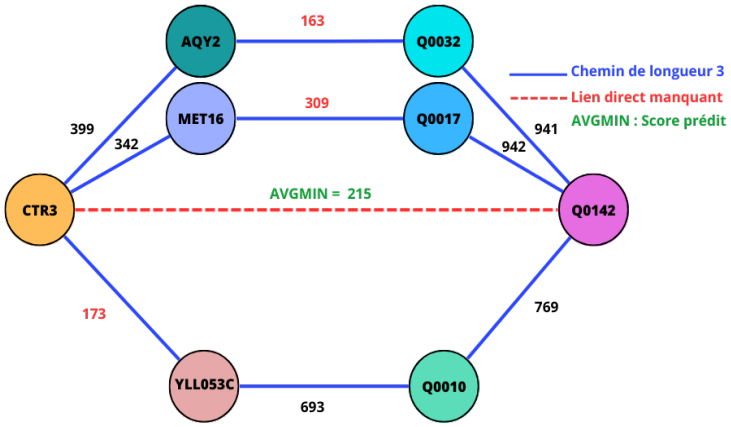


FIGURE 4.6 – Représentation basée sur l'agrégation des chemins de longueur 3

naisons possibles (représentations) qui peuvent être fournies à un algorithme d'apprentissage pour prédire les interactions manquantes et leurs valeurs.

4.4.2 Apprentissage automatique d'un modèle combinant les différentes représentations

Nous proposons d'utiliser les différentes représentations obtenues par agrégation et les différentes mesures de similarité topologiques comme entrée d'un algorithme d'apprentissage afin d'apprendre, sur un jeu de données, à combiner les informations apportées par ces différents descripteurs pour faire de meilleures prédictions. Notre approche par apprentissage, schématisée dans la Figure 4.7, commence par calculer l'ensemble des mesures de similarité topologiques AA, CN, PA, RA, TN et L3 pour chaque couple de protéines non directement connectées et ayant au moins un chemin intermédiaire de longueur 3. Les arcs directs qui relient ces protéines sont « calculables » par notre approche puisqu'il est possible de calculer nos représentations par agrégation. L'étape suivante consiste à calculer nos représentations pour tous les arcs calculables. Après, nous ajoutons le nombre de liens intermédiaires de longueur 3 entre les couples de protéines concernées. Nous donnons toutes ces caractéristiques en entrée d'un algorithme d'apprentissage supervisé tel que l'arbre de modèle Jolly (2018). Nous avons choisi un arbre de modèle pour son efficacité, en temps de calcul et en précision, sa lisibilité et sa capacité à gérer des attributs éventuellement corrélés. Le but de ce travail n'est pas d'identifier le meilleur algorithme d'apprentissage, en en comparant plusieurs, mais de montrer comment l'apprentissage automatique, et en particulier un arbre de modèles, peut combiner les apports des différentes mesures de similarité et de notre agrégation pour mieux prédire les interactions non-encore étudiées et leurs valeurs. Enfin, nos testons notre approche pour les couples de protéines reliés par au moins un chemin de

longueur 3 et pour lesquels il n'y a pas un lien direct. Les arcs manquants sont dans ce cas des arcs « prédictibles »car nous pouvons les prédire grâce à notre approche.

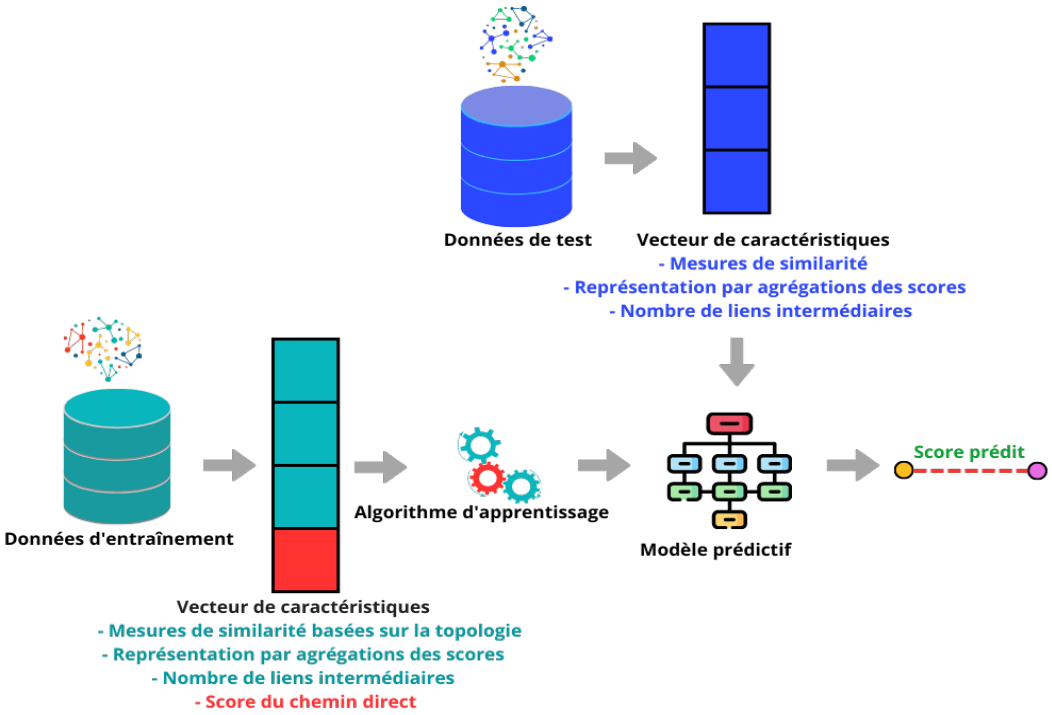


FIGURE 4.7 – Prédiction par apprentissage

Dans la section suivante, nous discutons de nos résultats et nous évaluons la performance de notre représentation par agrégation et de notre méthode de prédiction par apprentissage automatique.

4.5 Expérimentations

Les expérimentations menées dans ce travail ont trois objectifs principaux :

— Validation et évaluation de nos modélisations des IPPs : les modélisations dénormalisée et normalisée que nous avons proposées pour modéliser un réseau IPP ont été validées par une implantation et testées grâce à un jeu de requêtes complexes. Comme notre approche considère des chemins de longueur 3, nous cherchons la modélisation qui permet de répondre rapidement aux requêtes complexes dont la profondeur est égale à 3.

— Construction de notre modèle d'apprentissage automatique basé sur l'agrégation : nous calculons les mesures de similarité toplogiques et nos représentations par agrégation afin de les fournir à un algorithme de régression et construire un modèle permettant de prédire les scores manquants dans un IPP pondéré.

— Évaluation de nos représentations et de notre approche d'apprentissage : Nos représentations basées sur l'agrégation et notre algorithme d'apprentissage automatique ont été validées expérimentalement. Afin de tester l'efficacité de nos propositions, nous avons comparé leurs performances aux mesures de similarité classiques basées sur la topologie du graphe en se basant sur différentes métriques.

4.5.1 Métriques d'évaluation

Pour valider les résultats obtenus par les différentes représentations AA, CN, PA, RA, TN, L3, nos représentations par agrégation et notre modèle d'apprentissage, il est important de vérifier si les scores prédits ont été bien ajoutés dans la V11.5 de la base de données STRING et de comparer la valeur prédite à la valeur ajoutée. En outre, pour comparer toutes ces méthodes, il est nécessaire de disposer d'un ensemble de métriques.

Temps de calcul

Pour cette première métrique, nous cherchons le temps nécessaire pour calculer les différentes mesures AA, CN, PA, RA, TN, L3 et les représentations par agrégation en utilisant les données de la version version V11.0 de STRING. Cette métrique nous permet de voir si notre représentation est coûteuse en termes du temps de calcul. Quant à l'arbre de modèles, il n'est pas concerné par cette métrique puisqu'il est construit après le calcul de toutes ces mesures pour toutes les interactions connues dans V11.0.

Précision du Top 500

Afin de comparer l'information apportée par les représentations par agrégation et par l'apprentissage d'un arbre de modèles aux différentes mesures de similarité topologiques, nous utilisons une deuxième métrique qui calcule la précision des 500 meilleures prédictions (Top 500) Li et al. (2018). Comme les mesures de similarité ne fournissent pas un score directement comparable avec celui renseigné dans les canaux, l'état de l'art Kovács et al. (2019) consiste à évaluer la précision définie comme le rapport entre le nombre de liens Lp qui figurent dans le Top 500 et le nombre de liens ajoutés dans V11.5. Ainsi, si Lp interactions parmi les top 500 ont été ajoutées dans version V11.5 alors la précision est :

$$Précision = \frac{Lp}{500} \tag{4.9}$$

MAE, $MRAE$ **et** *$RMSE$*

Les trois dernières métriques sont l'erreur absolue moyenne (MAE) , l'erreur relative absolue moyenne ($MRAE$) et la racine carrée de l'erreur quadratique moyenne ($RMSE$). L'utilisation de MAE et $RMSE$ permet de calculer la moyenne de l'écart entre la valeur du score prédite et sa valeur réelle Quant à la métrique $MRAE$, elle mesure la valeur moyenne du quotient de l'erreur absolue entre le score attendu et le score prédit à la valeur du score attendu. $MRAE$ est généralement exprimée en pourcentage. Pour ces trois métriques, nous considérons l'arbre de modèles puisque les autres mesures de similarité ne calculent pas une valeur comparable directement avec le score enregistré dans la base. De plus, nous effectuons une étude d'ablation pour évaluer l'apport d'utilisation de nos représentations par agrégation lors de la prédiction des valeurs manquantes. Ainsi, nous comparons les valeurs de MAE, $MRAE$ et $RMSE$ des arbres de modèles construits avec toutes les mesures (ALL), avec toutes les mesures sauf L3 (SANS_L3), avec toutes les mesures sauf nos représentations par agrégation (SANS_AG), avec toutes les mesures sauf L3 et nos représentations par agrégation (SANS_AG_L3). L'objectif de cette étude est de vérifier si l'utilisation de nos représentations par agrégation améliore la performance de l'apprentissage.

Soit A la valeur du score manquant dans le graphe des IPPs et E la valeur du score prédit en utilisant les données extraites à partir du graphe. Les formules des différentes métriques sont les suivantes :

$$MAE(A, E) = \frac{1}{n} \sum_{i=1}^{n} |A_i - E_i| \tag{4.10}$$

$$MRAE(A, E) = \sum_{i=1}^{n} \frac{|A_i - E_i|}{E_i} \tag{4.11}$$

$$RMSE(A, E) = \sqrt{\frac{\sum_{i=1}^{n} (A_i - E_i)^2}{n}} \tag{4.12}$$

4.5.2 Banc d'essais STRING

Afin d'évaluer la capacité de notre approche à prédire les valeurs manquantes sur les liens d'un réseau IPP pondéré, nous avons utilisé STRING (Search Tool for the Retrieval of Interacting Genes) qui stocke les données d'un IPP pondéré. STRING est une base de données largement utilisée dans le domaine de la biologie. Les données de STRING sont dynamiques. Au fil des années, la base a évolué en termes de protéines intégrées et d'in-

teractions ajoutées, voire supprimées. Plusieurs versions de STRING sont disponibles en ligne. Les deux versions les plus récentes sont V11.0 et V11.5 qui ont été mises en ligne respectivement en 2020 et 2021. Dans ce travail, ces deux versions ont été utilisées pour tester l'efficacité de notre approche lors de la prédiction des liens manquants et de leurs étiquettes en passant d'une version à une autre. Les interactions dans STRING sont de deux types : interactions connues (interactions physiques détectées expérimentalement ou affirmés par un expert humain) et interactions fonctionnelles (interactions prédites par contexte génomique ou fouille d'articles scientifiques) Franceschini et al. (2015). La version V11.5 la plus récente de STRING contient 14 094 organismes (homme, levure, etc.) avec 20 052 394 042 interactions. Ces interactions proviennent de plusieurs sources, appelées canaux Szklarczyk et al. (2021). Ces canaux sont *Experimental*, *Database*, *Neighborhood*, *Fusion*, *Cooccurrence*, *Coexpression* et *Textmining*. Pour chaque canal, un score est calculé pour indiquer la fiabilité de l'interaction suivant les données collectées à travers ce canal.

— Expérimental : assigne un score de confiance à l'interaction en se basant sur les expériences réelles établies dans les laboratoires.

— Database : contient les données qui ont été affirmées par un expert humain. Par conséquent, ce canal contient dans la plupart des cas des scores très élevés (expériences biochimiques, biophysiques et génétiques).

— Textmining : attribue un score aux paires de protéines mentionnées ensemble dans des publications scientifiques.

— Coexpression : calcule le score d'une interaction protéique par la comparaison du modèle d'expression des protéines.

— Neighborhood : comprend les données de voisinage génomique.

— Fusion : assigne aux paires de protéines un score d'association lorsqu'il existe au moins un organisme où leurs orthologues respectifs ont fusionné en un seul gène.

— Cooccurrence : évalue la distribution phylogénétique des orthologues de toutes les protéines d'un organisme donné.

— Combiné : le canal combiné combine les données qui proviennent des sept canaux précédemment cités.

Le réseau des IPPs de STRING peut être représenté sous la forme d'un graphe pondéré et non-orienté. Les nœuds sont les protéines qui sont caractérisées par leurs identifiants. La présence d'un arc étiqueté avec un poids (score) p issue d'un canal c signifie que les deux protéines correspondantes interagissent avec une probabilité p pour le ca-

nal c. Ainsi, les différentes interactions de STRING sont étiquetées par des scores qui varient entre 0 (pas d'interaction pour le canal concerné) et 1000 (le score le plus fort).

STRING spécifie des seuils de confiance :

— Scores faibles (Low) : les arcs qui portent des scores

— Scores moyens (Medium) : les arcs qui portent des scores >=400 et <700 sont considérés comme des interactions de fiabilité moyenne. Ils représentent 15.91% de l'ensemble des interactions de STRING.

— Scores élevés (High) : les arcs qui portent des scores >=700 et <900 sont considérés comme des interactions de fiabilité élevée. Ils représentent 5.33% de l'ensemble des interactions de STRING.

— Scores les plus élevés (Highest) : les arcs qui portent des scores >=900 sont considérés comme des interactions de très grande fiabilité. Ils représentent 7.58% l'ensemble des interactions de STRING

Dans ce travail, nous nous intéressons aux données de l'organisme levure (S. Cerevisiae) de STRING. Nous utilisons les données de la version V11.0 et V11.5 de STRING relatives à cet organisme. En effet, les différentes versions de STRING correspondent aux connaissances des réseaux d'IPP à une date donnée. Passer d'une version à une autre se traduit par l'ajout mais aussi la suppression de certaines interactions, voire de protéines, et parfois la modification des scores d'interactions existantes. Les MAE et $RMSE$ présentés dans la Figure 4.8 montrent la modification des scores entre les deux versions. Si l'on considère tous les arcs qui ont une valeur pour le score combiné dans les versions V11.0 et V11.5 et que l'on utilise la valeur de la version V11.0 comme prédiction et que l'on les compare avec les valeurs présentes dans V11.5, on mesure un MAE de 60.99 et un $RMSE$ de 109.36. En outre, les scores Medium et High possèdent les valeurs de MAE et $RMSE$ les plus élevées. Le choix des deux versions V11.0 et V11.5 de STRING a pour objectif de vérifier si on arrive à prédire les scores des interactions qui vont apparaître dans une nouvelle version de STRING en se basant sur une version antérieure. Le tableau 4.1 résume le nombre de protéines et le nombre d'interactions connues pour chaque canal de l'organisme levure dans les deux versions V11.0 et V11.5 de STRING.

4.5.3 Données d'apprentissage

Dans la phase d'apprentissage, nous nous appuyons sur une partie des données de V11.0 de STRING. En effet, nous considérons l'ensemble des couples de protéines P^{APP} qui interagissent directement et qui sont reliées par au moins un chemin intermédiaire de

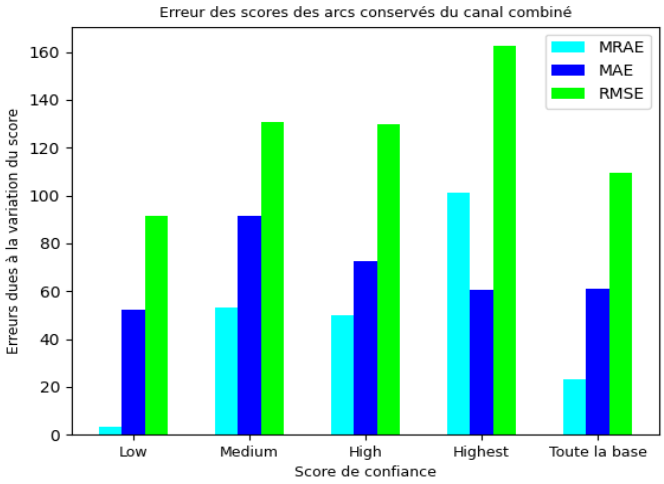


FIGURE 4.8 – Variation des scores de STRING

Caractéristique	V11.0	V11.5
Nombre de protéines	6 574	6 394
Nombre d'arcs dans le canal combiné	922 983	994 296
Nombre d'arcs dans le canal coexpression	555 166	602 624
Nombre d'arcs dans le canal cooccurence	4 681	4 689
Nombre d'arcs dans le canal database	46 816	78 317
Nombre d'arcs dans le canal experimental	394 274	482 448
Nombre d'arcs dans le canal fusion	3 916	4 094
Nombre d'arcs dans le canal neighborhood	121 000	120 550
Nombre d'arcs dans le canal textmining	697 466	714 101

TABLE 4.1 – Nombre d'interactions directes par canal dans l'organisme levure dans V11.0 et V11.5 de STRING

longueur 3. Comme il existe un chemin de longueur 3 entre ces couples, nous pouvons utiliser notre approche d'apprentissage basée sur l'agrégation pour calculer un score pour l'interaction directe. Nous appelons l'interaction directe dans ce cas « Interaction calculable ». Pour chaque couple du P^{APP}, nous calculons un vecteur qui comprend les valeurs des mesures de similarité topologiques (AA, CN, PA, RA,TN et L3), les valeurs des 9 représentations par agrégations (MINMIN, MINMAX, AVGMIN, AVGMAX, AVGAVG, MAXMIN, MAXAVG, MAXMAX), le nombre des chemins intermédiaires de longueur 3 et la valeur portée par l'arc direct entre les deux protéines considérées. Nous calculons ce vecteur pour chaque canal de STRING. Nous obtenons huit jeux d'apprentissage qui concernent les huit canaux de la base de données STRING. Pour les mesures de similarité topologiques, seuls les chemins de longueur 2 entre les couples du P^{APP} sont pris en considération. De plus, comme ces mesures sont adaptées aux IPP non-pondérés, les scores des interactions ne sont pas utilisés. Les valeurs des mesures sont ainsi calcu-

Canal	Nombre d'arcs calculables
Canal combiné	919 914
canal coexpression	552 315
Canal cooccurence	3 539
Canal database	43 856
Canal experimental	394 274
Canal fusion	3 916
Canal neighborhood	119 195
Canal textmining	693 904

TABLE 4.2 – Jeux d'apprentissage : interactions calculables par canal dans V11.0

lées en se basant sur l'existence ou non d'un chemin intermédiaire de longueur 2. Étant donné que 71.18% des scores dans STRING sont faibles et peuvent donc être des faux positifs, nous avons testé l'impact de l'utilisation de ces scores sur la capacité des mesures topologiques à identifier les arcs potentiels dans STRING. À la suite de ces tests, nous avons conclu que les scores faibles empêchent les mesures topologiques de trouver les protéines les plus susceptibles à interagir. En d'autres termes, deux protéines reliées par plusieurs chemins intermédiaires de scores faibles ne représentent pas forcément un bon candidat. Par conséquent, pour tirer profit pleinement des mesures de similarité topologiques, nous les avons calculés pour les scores >=400 du canal combiné. En ce qui concerne nos représentations par agrégation, nous souhaitons tester notre capacité à prédire les scores avec le minimum d'erreur qu'ils soient faibles, moyens, élevés ou très élevés. Ainsi, aucun seuil n'a été fixé et tous les arcs de la base de données STRING ont été pris en considération. Tous les calculs des vecteurs d'apprentissage ont été faits entièrement dans le SGBD orienté graphes Neo4j en utilisant son langage d'interrogation Cypher. Ces vecteurs représentent notre jeu d'apprentissage. Ce jeu d'apprentissage est fourni à un arbre de régression afin de construire un modèle qui permettra de prédire les étiquettes de certains arcs manquants dans V11.0 de STRING. Le tableau 4.2 résume la taille de notre base d'apprentissage pour chaque canal de la base de données STRING.

Pour évaluer le modèle construit à l'aide des données d'apprentissage, nous nous basons sur des données différentes dédiées pour la phase de test.

4.5.4 Données de test

Dans la phase de test, nous utilisons également les données de V11.0 de STRING. Néanmoins, dans ce cas, nous considérant d'autres couples de protéines P^{TEST} pour lesquelles il existe au moins un chemin intermédiaire de longueur 3 mais il n'y a aucun arc direct reliant ces protéines. Les interactions entre les protéines P^{TEST} sont « prédic-

Canal	Ajoutées dans V11.5	Prédictibles et ajoutées dans V11.5
Canal combiné	234 172	234 120
Canal coexpression	118 733	64 064
Canal cooccurence	29	8
Canal database	78 317	42 221
Canal experimental	162 880	162 406
Canal fusion	545	212
Canal neighborhood	11 489	10 321
Canal textmining	177 847	119 195

TABLE 4.3 – Jeux de test : interactions prédictibles par canal dans V11.0 et ajoutées dans V11.5

tibles »à l'aide de notre modèle. Le vecteur du test contient les valeurs des mesures de similarité topologiques, les valeurs des9 représentations par agrégations et le nombre des chemins intermédiaires de longueur 3 entre les couples de protéines P^{TEST} . Les vecteurs sont calculés pour les huit canaux et permettent d'obtenir huit jeux de test. Les données de test sont fournies au modèle construit sur les données du jeu d'apprentissage pour prédire la valeur sur les arcs manquants pour les protéines P^{TEST} non directement liées. Grâce à notre approche, nous pouvons prédire le score de tous les arcs prédictibles et qui ont été ajoutés dans la version V11.5 de la base de données STRING. Certains scores ajoutés dans V11.5 sont non-prédictibles et donc ne seront pas prédits par notre approche. Le tableau 4.3 illustre le nombre d'interactions prédictibles et ajoutées dans V11.5 pour tous les canaux de la base de données STRING.

4.6 Résultats

4.6.1 Évaluation des modélisations des réseaux d'IPP pondérés

Pour évaluer les deux modélisations que nous avons proposées pour les IPPs pondérés, nous avons stocké les données de V11.0 de STRING dans deux variantes de bases données orientées graphes : G_IPP dénormalisée et G_IPP normalisée. La base de données G_IPP dénormalisée s'appuie sur la modélisation dénormalisée. Les arcs qui relient les protéines portent dans ce cas la même étiquette « INTERACTS_**WITH** ». Chaque arc contient les sept scores qui proviennent des sept canaux et un huitième score combiné sous la forme de huit paires clé-valeur. En ce qui concerne la base de données G_IPP normalisée, elle est créée suivant une modélisation normalisée. Ainsi, un arc est créé pour chaque canal et contient le score de ce canal sous la forme d'une paire clé-valeur. Par conséquent, huit types d'arcs peuvent exister entre deux protéines

si les scores de tous les canaux sont renseignés. Les arcs portent dans ce cas différentes étiquettes qui désignent le nom du canal : « DATABASE », « EXPERIMENTAL », « NEIGHBORHOOD », « TEXTMINING », « FUSION », « COEXPRESSION », « CO-OCCURENCE » et « COMBINED ». Pour comparer la performance des deux bases de données orientées graphes créées, nous les avons implantées sur deux machines virtuelles différentes ayant les mêmes caractéristiques. Les deux machines fonctionnent sous le système d'exploitation 64 bits Ubuntu-18.04.01 LTS avec 32 Go de RAM et 8 To de disque. Nous avons utilisé Neo4j (version 4.4.2) comme SGBD orienté graphes. En outre, nous avons utilisé MariaDB (version 10.3.31) pour implanter des bases de données relationnelles similaires R_IPP dénormalisée et R_IPP normalisée afin de comparer la performance des modélisations orientées graphes des IPPs pondérés aux modélisations relationnelles. Nous avons évalué les quatre bases de données G_IPP dénormalisée, G_IPP normalisée, R_IPP dénormalisée et R_IPP normalisée en termes de temps de chargement de données de la base de données STRING, l'espace mémoire occupé et le temps de réponse face à des requêtes de profondeur inférieur ou égale à 3. Dans ce travail, pour optimiser le temps de calcul des vecteurs d'apprentissage et de test, nous choisirons la modélisation qui minimise le temps de recherche et l'agrégation des scores sur les chemins de longueur 3. Le tableau 4.4 illustre un exemple de requête complexe de longueur 3. Cette requête cherche tous les chemins intermédiaires de longueur 3 entre les protéines $YBL045C$ et $YKL192C$. Chaque arc du chemin intermédiaire doit avoir un score combiné très élevé et un score textmining supérieur à 500.

Dans les deux variantes des bases de données orientées graphes G_IPP dénormalisée et normalisée, les relations sont créées et stockées physiquement dans la base. Par conséquent, le temps de chargement des données des IPPs dans des bases est nettement plus long que dans les bases de données relationnelles en raison du temps nécessaire pour créer des arcs entre les nœuds. Comme la version G_IPP normalisée permet de créer un arc pour chaque canal, le temps de chargement et le volume occupé dépassent ceux de la base de données G_IPP dénormalisée. Cela a été validé par les expériences que nous avons menées.

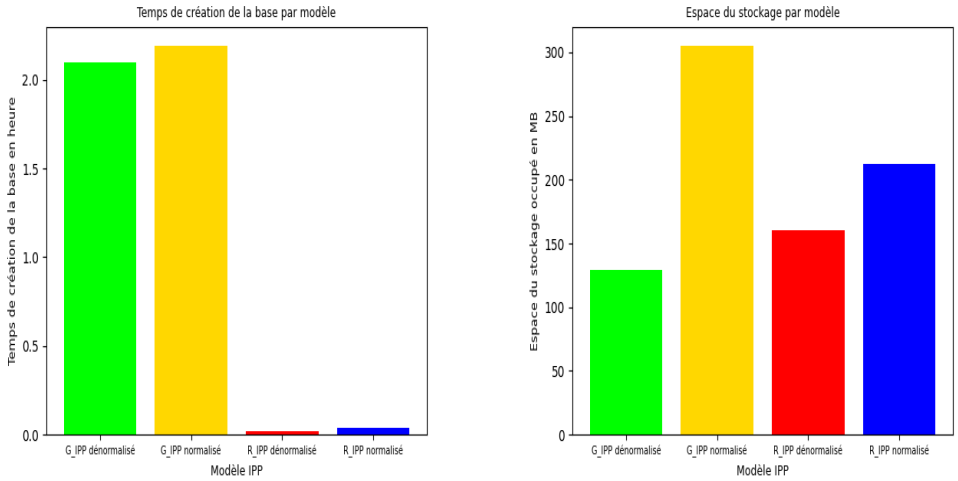
Comme le montre la Figure 4.9a, les deux versions orientées graphes des IPPs sont beaucoup plus coûteuses que les deux versions relationnelles des IPPs en termes du temps de chargement de la base. En outre, la création des IPPs en utilisant une modélisation orientée graphes normalisée est légèrement plus coûteuse que la modélisation dénormalisée. En termes de volume occupé, comme l'illustre la Figure 4.9b, la version normalisée de la base de données orientée graphes des IPPs est plus coûteuse à cause de la création d'un arc pour chaque canal.

En termes de temps de réponse aux requêtes, nous avons calculé le temps de réponse

G_IPP Dénormalisé	G_IPP Normalisé
	MATCH(p1 :PROTEIN{id :"4932.YBL045C"})
MATCH (p1 :PROTEIN{id :"4932.YBL045C"})	**WITH** p1
WITH p1	**MATCH**(p2 :PROTEIN{id :"4932.YKL192C"})
MATCH (p2 :PROTEIN{id :"4932.YKL192C"})	**WITH** p1, p2
WITH p1, p2	**MATCH** p=(p1)-[:INTERACT_**WITH***3..3]->(p2)
MATCH p=(p1)-[:INTERACT_**WITH***3..3]->(p2)	**WITH** p1, p2, p
WITH p1, p2, p	**MATCH**(p1)-[r :COMBINED]->(p2)
MATCH (p1)-[r :INTERACT_**WITH**]->(p2)	**WHERE** r.combined_score>=900
WHERE r.combined_score>=900	**WITH** p1, p2, p, r, relationships(p) AS rels
AND r.textmining>=500	**MATCH**(p1)-[k :TEXTMINING]->(p2)
WITH p1, p2, p, r, relationships(p) AS rels	**WHERE** k.textmining>=500
UNWIND rels AS x	**WITH** p1, p2, p, r, relationships(p) AS rels
WITH p1, p2, r, x	**UNWIND** rels AS x
WITH p1, p2, r,	**WITH** p1, p2, r, x
MIN(x.combined_score) AS MIN,	**WITH** p1, p2, r,
AVG(x.combined_score) AS AVG,	**MIN**(x.combined_score) AS MIN,
MAX(x.combined_score) AS MAX	**AVG**(x.combined_score) AS AVG,
WITH p1, p2, r,	**MAX**(x.combined_score) AS MAX
MIN(**MIN**) AS MINMIN,	**WITH** p1, p2, r,
AVG(**MIN**) AS AVGMIN,	**MIN**(**MIN**) AS MINMIN,
MAX(**MIN**) AS MAXMIN,	**AVG**(**MIN**) AS AVGMIN,
MIN(**AVG**) AS MINAVG,	**MAX**(**MIN**) AS MAXMIN,
AVG(**AVG**) AS AVGAVG,	**MIN**(**AVG**) AS MINAVG,
MAX(**AVG**) AS MAXAVG,	**AVG**(**AVG**) AS AVGAVG,
MIN(**MAX**) AS MINMAX,	**MAX**(**AVG**) AS MAXAVG,
AVG(**MAX**) AS AVGMAX,	**MIN**(**MAX**) AS MINMAX,
MAX(**MAX**) AS MAXMAX	**AVG**(**MAX**) AS AVGMAX,
RETURN p1.id AS p1, p2.id AS p2,	**MAX**(**MAX**) AS MAXMAX
MINMIN, MINAVG,	**RETURN** p1.id AS p1, p2.id AS p2,
MINMAX, AVGMIN, AVGAVG,	MINMIN, MINAVG,
AVGMAX, MAXMIN, MAXAVG,	MINMAX, AVGMIN, AVGAVG,
MAXMAX, r.combined_score as combined;	AVGMAX, MAXMIN, MAXAVG,
	MAXMAX, r.combined_score as combined;

TABLE 4.4 – Requête complexe de profondeur 3

des différentes bases de données face à des requêtes de profondeur 1, 2 et 3. Comme l'illustre la Figure 4.10, la version G_IPP normalisée est plus performante lors de la réponse à des requêtes complexes dont la longueur de chemin est égale à 3. Pour les requêtes de profondeur 1 qui interrogent les interactions directes, les bases relationnelles présentent les meilleures performances. Néanmoins, leurs temps de réponse sont proches de celui des bases de données orientées graphes. À partir des chemins de longueur 2, les bases de données orientées graphes présentent de meilleures performances. La différence entre les versions G_IPP normalisée et dénormalisée n'est pas notable. Cependant, la version R_IPP normalisée est plus coûteuse à cause du temps nécessaire pour établir les



(a) Temps de chargement (b) Volume occupé

FIGURE 4.9 – Temps de chargement et volume occupé par modèle

jointures.

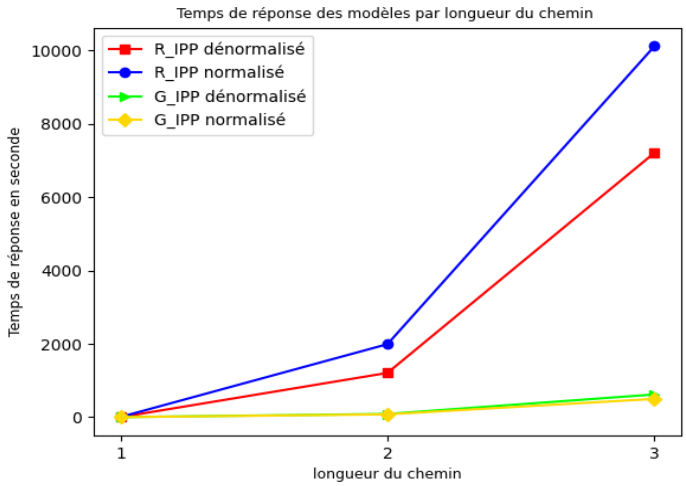


FIGURE 4.10 – Temps de réponse à des requêtes complexes par modèle

4.6.2 Comparaison des représentations par agrégations aux mesures topologiques

En termes de temps de calcul, nos résultats présentés dans la Figure 4.11 montre que nos 9 représentations basées sur l'agrégation ne sont pas coûteuses comparées à L3. Pour certaines représentations basées sur l'agrégation comme AVGMIN, le temps de calcul a été inférieur au temps de calcul de L3. Le calcul des mesures topologiques qui n'utilisent que les chemins de longueur 2 est plus rapide que le temps de calcul de L3 et

nos représentations par agrégation basés sur des chemins de longueur 3.

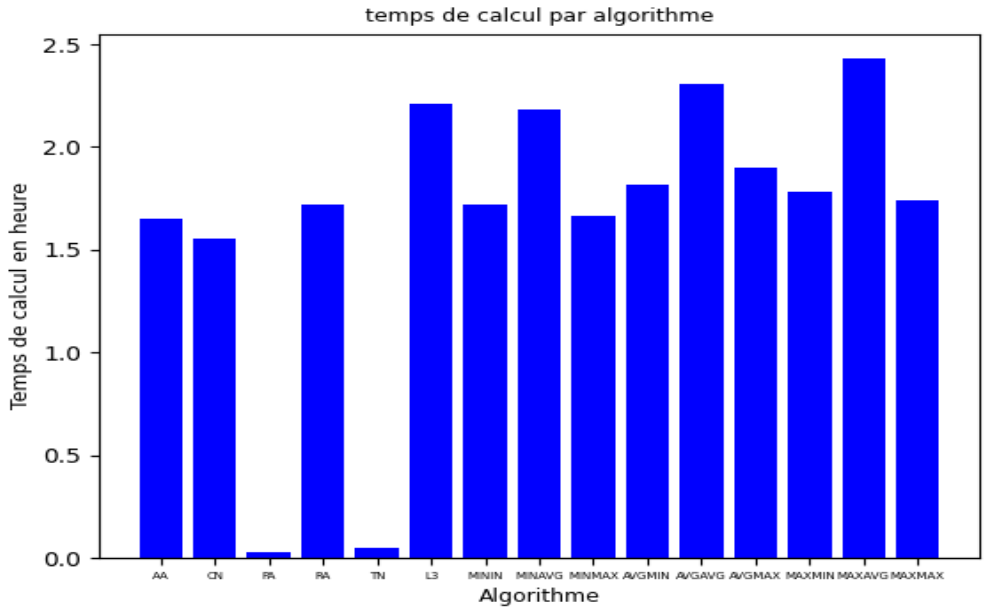


FIGURE 4.11 – Temps de calcul des représentations pour le canal combiné

En ce qui concerne la précision, nos résultats illustrés dans la Figure 4.12 montrent que notre arbre de modèles surpasse toutes les mesures de similarité classiques en termes de précision du Top 500. En outre, la mesure L3 est plus performante que les mesures topologiques basées sur des chemins de longueur 2. L'utilisation de ces deux représentations, en plus des mesures topologiques, pourrait contribuer à la prédiction des liens IPPs manquants avec le minimum d'erreur.

4.6.3 Évaluation de l'apprentissage automatique basé sur l'agrégation

Pour évaluer l'apport de l'utilisation de nos représentations par agrégation dans l'apprentissage automatique, nous avons mené une étude d'ablation. L'ablation consiste à éliminer certains indicateurs fournis à l'algorithme d'apprentissage pour comprendre leurs contributions. Tout d'abord, nous avons calculé MAE, $MRAE$ et $RMSE$ pour notre modèle construit à l'aide de tous les indicateurs. Ensuite, nous avons calculé MAE, $MRAE$ et $RMSE$ en utilisant tous les indicateurs sauf L3. Après, nous avons cherché les valeurs de MAE, $MRAE$ et $RMSE$ suite à la suppression de toutes les représentations par agrégation. Enfin, nous avons calculé le MAE, $MRAE$ et $RMSE$ en utilisant tous les indicateurs sauf L3 et les représentations par agrégation. Comme le montre la Figure 4.13, la suppression de L3 augmente le MAE de 15.3% et le $RMSE$ de 25.2%. Tou-

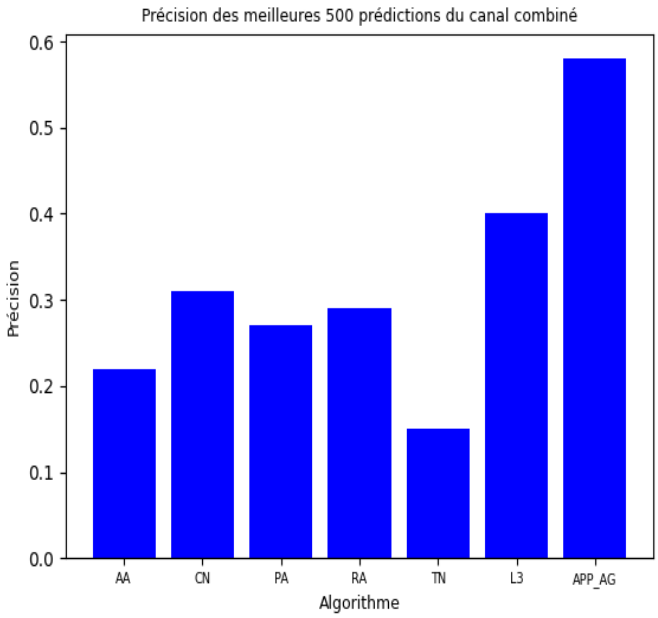


FIGURE 4.12 – Précision des représentations pour le canal combiné

tefois, la suppression des représentations par agrégation augmente le MAE de 32.9% et le $RMSE$ de 52.1%. De plus, la suppression de L3 et les différentes représentations par agrégation augmente le MAE de 64.2% et le $RMSE$ de 85.4%. De plus, les résultats de la métrique $MRAE$ présentés dans 4.13 montrent que l'écart de la valeur prédite par rapport à la valeur réelle augmente considérablement lors de la suppression de nos représentations par agrégation. Ainsi, bien que l'apport des représentations par agrégations à l'apprentissage soit plus important que celui de L3, l'utilisation de L3 améliore la performance de notre modèle.

La contribution de l'utilisation de nos mesures basées sur l'agrégation dans l'apprentissage a été confirmée pour tous les canaux prédictibles dans V11.0 et ajoutés dans V11.5 de la base de données STRING. Les résultats de nos expérimentations illustrés dans le tableau 4.5 montrent que des résultats similaires sur tous les canaux. La suppression de nos mesures basées sur l'agrégation augmente l'erreur pour tous les canaux.

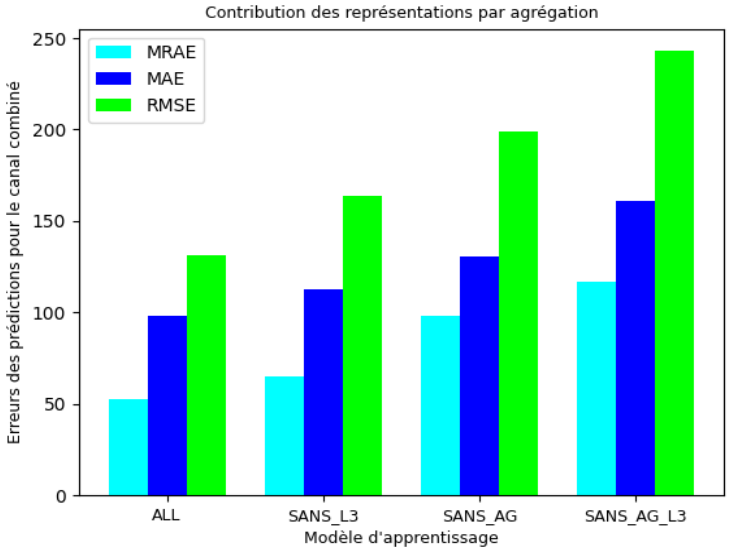


FIGURE 4.13 – Contribution des représentations par agrégation

Canal	SANS_AG_L3	SANS_AG	SANS_L3	ALL
Combiné	117.01	98.00	65.19	52.88
Coexpression	88.30	61.98	44.29	33.12
Cooccurence	89.14	69.30	57.11	48.10
Database	132.76	119.21	89.51	77.31
Expérimental	114.56	89.14	63.99	54.16
Fusion	44.91	34.99	24.83	12.55
Neighborhood	79.10	54.6	33.29	23.44
Textmining	137.10	111.21	99.29	86.49

TABLE 4.5 – MRAE des arcs prédictibles et ajoutés de tous les canaux de la base de données STRING

4.7 Discussion

Les expérimentations menées dans ce travail confirment l'hypothèse que l'utilisation des chemins de longueur 3 pour la prédiction des liens manquants en biologie est plus pertinente que l'utilisation des chemins de longueur 2. En termes de précision, l'utilisation de L3 permet de prédire l'existence d'un lien potentiel avec une précision qui dépasse les mesures topologiques basées sur des chemins de longueur 2. En outre, l'utilisation des chemins de longueur 3 et de leurs étiquettes dans notre approche d'apprentissage basée sur l'agrégation permet d'obtenir une précision qui dépasse celle de L3. Par conséquent, l'utilisation des étiquettes portées par les chemins de longueur 3

permet d'améliorer la précision.

En termes d'erreur du score prédit par rapport au score ajouté, nos résultats montrent que notre approche d'apprentissage basée sur l'agrégation permet de prédire des scores aux interactions manquantes avec des valeurs relativement proches des valeurs ajoutées dans la version V11.5. Étant donné que la base de données STRING est dynamique et ses scores changent d'une version à une autre, notre approche peut être impactée par la présence des faux positifs dans la version V11.0 et qui ont été mis à jour dans la version V11.5. De plus, grâce aux résultats de notre étude d'ablation, nous pouvons constater que toutes les représentations sont informatives. Néanmoins, la présence de L3 et de nos représentations par agrégation contribuent nettement à la prédiction des scores avec le minimum d'erreur.

Dans nos expérimentations, nous avons utilisé la version normalisée des IPPs pour calculer les différents vecteurs d'apprentissage et de test. L'avantage de l'utilisation de la version normalisée est qu'elle permet d'accéder directement aux informations relatives à un canal particulier.

4.8 Conclusion

Le recours à des mesures de similarité topologiques est sans doute d'un grand intérêt dans plusieurs cas d'utilisation et dans une multitude de domaines. Toutefois, dans le domaine de la biologie, se limiter aux mesures de similarité topologiques basées sur des chemins de longueur 2 présente plusieurs inconvénients. D'une part, les chemins de longueur 2 sont moins représentatifs que les chemins de longueur 3 dans les réseaux d'IPP. Par exemple, deux protéines reliées par un seul chemin intermédiaire de longueur 2 ont tendance à interagir s'il existe plusieurs chemins de longueur 3 qui les relient. En outre, les mesures basées sur la topologie sont généralement proposées pour des graphes non-pondérés. Dans ce cas, le calcul de ces mesures utilise l'information relative à l'existence ou non d'un lien. Le score calculé et trié par ordre décroissant permet d'identifier une liste d'arc potentiel suivant ce score qui ne comprend aucune information sur les valeurs portées par les arcs. Pour pallier cet inconvénient, nous avons proposé dans ce travail une nouvelle représentation par agrégation. L'avantage de cette représentation est qu'elle permet de capturer toutes les informations apportées par les chemins intermédiaires entre les couples de protéines non seulement en termes d'existence d'un lien mais aussi du score qu'il porte. Grâce à nos différentes représentations, nous avons construit un modèle d'apprentissage qui prend en entrée les 9 combinaisons des fonctions MIN, AVG et MAX, les mesures basées sur la topologie, le nombre de chemins de longueur 3 entre deux protéines pour prédire les arcs manquants et leurs valeurs.

Nos résultats expérimentaux montrent que nos représentations basées sur l'agrégation contribuent nettement à l'amélioration de la performance du modèle en termes d'erreur absolue moyenne (MAE), erreur relative absolue moyenne ($MRAE$) et racine carrée de l'erreur quadratique moyenne ($RMSE$).

Chapitre 5

Conclusion générale

Sommaire

5.1 Bilan des contributions

La quantité de données générées et collectées ne cesse d'augmenter à un rythme sans précédent, provenant de diverses sources telles que les réseaux sociaux, les capteurs IoT et les données ouvertes. Dans un monde de plus en plus concurrentiel, cette diversité des sources de données est une véritable opportunité pour les entreprises, mais constitue également un défi majeur en raison des limites des technologies classiques de stockage, de traitement et d'analyse de données. De nombreuses recherches sont donc menées pour découvrir de nouvelles technologies capables d'extraire des connaissances pertinentes à partir de ces données massives et hétérogènes. Parmi ces technologies, les bases de données graphes attirent le monde académique et professionnel lors de l'analyse des données non seulement massives et variées mais également très complexes, où les relations entre les données sont non triviales et apportent des informations pertinentes. Cette thèse s'inscrit pleinement dans le contexte de l'utilisation des bases de données graphes pour répondre à des problématiques scientifiques préoccupantes en science des données. D'une part, nous avons exploré les bases de données graphes pour la mise en place d'un entrepôt de données capable de répondre rapidement et efficacement à des requêtes analytiques très complexes. D'autre part, nous avons utilisé cette catégorie de bases de données pour effectuer des prédictions de valeurs manquantes sur les arêtes d'un réseau d'interaction protéine-protéine (IPP).

En ce qui concerne les entrepôts de données, les approches classiques impliquent généralement l'utilisation de bases de données relationnelles. Ce choix s'explique en grande partie par la maturité des systèmes de gestion de bases de données relationnelles et leur capacité à stocker et gérer efficacement des données structurées. Néanmoins, avec l'avènement du Big Data, plusieurs travaux de recherche ont mis en évidence les limites de ces systèmes et ont proposé de nouvelles solutions alternatives pour implanter des entrepôts de données massives. Dans ce contexte, les systèmes Not Only SQL (NoSQL) sont considérés comme une voie intéressante pour améliorer les capacités de stockage et la performance des entrepôts relationnels traditionnels. Les avantages des systèmes NoSQL par rapport aux bases de données relationnelles sont leur grande évolutivité, flexibilité accrue et capacité à gérer des données semi-structurées ou non structurées. Cependant, migrer un entrepôt classique basé sur une base de données relationnelles vers un entrepôt qui s'appuie sur une base NoSQL est une tâche complexe. En effet, les bases NoSQL sont de différentes catégories (orientées clés-valeurs, colonnes, documents et graphes) et ont des modèles de données différents de celui des bases relationnelles. Il est donc important de choisir la base NoSQL qui répond aux besoins de l'entreprise

tout en permettant une migration efficace de l'entrepôt de données. Pour répondre à cette problématique, nous avons proposé et évalué dans cette thèse une approche qui permet de migrer à un entrepôt de données orienté graphes. Pour assurer cette migration, nous avons proposé le formalisme MDM2G qui regroupe plusieurs règles de migration tout en prenant en compte les cas de schéma en étoile et de schéma en flocons de neige. Afin de valider notre approche et de la comparer à d'autres travaux existants dans la littérature, nous avons utilisé le jeu de données du banc d'essais TPC-DS. En outre, pour évaluer les performances de nos entrepôts orientés graphes, nous avons généré plusieurs volumes de données de TPC-DS ainsi qu'un ensemble de requêtes analytiques ayant différentes complexités. Nos résultats ont montré des différences significatives de performance entre les implantations relationnelles et orientées graphes d'un entrepôt. En effet, une implantation relationnelle fonctionne mieux pour les requêtes qui interrogent un petit nombre de tables, tandis qu'une implantation orientée graphes est plus performante pour des requêtes complexes impliquant plusieurs tables. En outre, contrairement aux bases de données relationnelles où l'utilisation d'un schéma en flocons de neige est souvent très coûteuse, l'utilisation de ce type de schéma pour la mise en oeuvre d'un entrepôt de données orienté graphes n'a pas d'impact notable sur la performance. Il convient de noter que les bases de données orientées graphes nécessitent plus d'espace de stockage que les bases relationnelles pour stocker les liens entre les données. Cependant, opter pour la migration aux graphes peut être justifié par les gains de performance lors de la réponse aux requêtes analytiques complexes.

En plus de l'implantation des entrepôts de données basés sur les graphes, nous avons proposé dans cette thèse d'utiliser les bases de données NoSQL orientées graphes pour stocker et analyser les informations portées par les arcs d'un réseau d'IPP pondéré. Dans ce cas, les noeuds du réseau représentent les protéines et les liens entre eux indiquent la présence d'une interaction physique. De plus, un poids est attribué à chaque lien pour quantifier la confiance accordée à cette interaction. Le poids est déterminé à partir de plusieurs sources d'information telles que les expérimentations effectuées en laboratoire et les données fournies par des experts. Dans notre approche, nous nous appuyons sur l'ensemble des interactions existantes entre les protéines ainsi que sur leurs poids pour prédire des futures interactions potentielles. Les données du réseau d'IPP pondéré sont alors stockées dans une base de données graphes pour exploiter les poids portés par les arcs et en extraire des connaissances utiles pour la prise de la décision. En effet, les approches traditionnelles se basent sur l'existence ou non d'un chemin intermédiaire de longueur 2 entre deux protéines non directement reliées pour identifier un futur lien direct entre ces deux protéines. Ces approches utilisent des mesures de similarité basées

sur la topologie du graphe telles que le nombre de voisins communs et Adamic Adar. Récemment, la mesure L3 a été introduite pour identifier un futur lien en se basant sur l'existence ou non des chemins intermédiaires de longueur 3. Malgré la pertinence des différentes mesures proposées, les poids portés par les arcs n'est pas considéré par ces approches. Afin de remédier à cette problématique, nous avons proposé dans ce travail de nouvelles représentations basées sur des fonctions d'agrégation permettant d'agréger les valeurs portées par les arcs. Nos représentations combinent les fonctions AVG, MIN et MAX pour agréger les poids portés par les chemins intermédiaires entre deux noeuds reliés par des chemins intermédiaires sans être reliés directement. De plus, nous avons proposé de combiner les mesures classiques basées sur la topologie du graphe et nos représentations basées sur l'agrégation dans une approche d'apprentissage automatique qui prédit des futures interactions et leurs scores. Pour valider et évaluer notre approche, nous avons utilisé les données de l'organisme levure du banc d'essais reconnu STRING. Nos résultats expérimentaux ont montré que nos représentations basées sur l'agrégation ne sont pas coûteuses en termes du temps de calcul. De plus, notre approche basée sur l'apprentissage qui combine ces représentations avec les mesures de similarité basées sur la topologie permet de prédire des liens pertinents manquants avec une précision qui dépasse la précision des mesures basées sur la topologie. En outre, nous avons réalisé une étude basée sur l'ablation qui montre la pertinence de nos représentations basées sur l'agrégation qui permettent de prédire les poids des futures interactions avec le minimum d'erreur.

L'ensemble des travaux de recherche décrits dans cette thèse ont fait l'objet des publications suivantes :

— *Akid H. and Ben Ayed M., « Towards nosql graph data warehouse for big social data analysis », 16th International Conference on Intelligent Systems Design and Applications (ISDA), Porto, Portugal, 14-12 December, 2016. Springer, Cham., pp. 965-973.*

— *Akid H., Ben Ayed M., G. Frey, and Lachiche N., « Entrepôts de données NoSQL orientés graphe », 35ème Conférence sur la Gestion de Données Principes Technologies et Applications (BDA), Lyon, France, 15-18 Octobre, 2019. Hal-03176580 pp.51-52.*

— *Akid H., Frey G., Ben Ayed M., and Lachiche N., « Performance of NoSQL graph implementations of star vs. snowflake schemas », IEEE acess. ISSN : 21693536, 10.1109/ACCESS.2022.3171256.*

— *Akid H., Chennen K, Frey G., Thompson J., Ben Ayed M., and Lachiche N., « Apprentissage automatique basé sur l'agrégation pour la prédiction de liens dans*

les réseaux d'interaction protéine-protéine », 22ème Conférence sur l'Extraction et la Gestion des connaissances (EGC), Blois, France, 24-28 Janvier, 2022, vol. RNTI-E-38, pp.481-482.

5.2 Perspectives

L'utilisation des bases de données graphes est actuellement un sujet d'actualité en science des données. Malgré les opportunités offertes par cette catégorie de bases de données, les défis scientifiques à relever sont nombreux. Au cours de nos recherches, nous avons identifié différents verrous scientifiques auxquels nous souhaitons apporter des réponses dans nos futurs travaux :

— Implantation et évaluation des opérateurs OLAP dans le cas des entrepôts orientés graphes : Bien que les opérateurs OLAP soient largement utilisés dans les bases de données relationnelles (R-OLAP), peu de travaux ont porté sur leur extension et utilisation dans le cas des bases de données graphes. Il serait donc intéressant de compléter les travaux existants en évaluant les performances des opérateurs OLAP pour les entrepôts basés sur les graphes (G-OLAP). Nous proposons d'évaluer les performances de ces opérateurs en considérant différentes tailles de données et différents nombres de dimensions interrogées pour augmenter la complexité de la requête ;

— Migration vers un entrepôt NoSQL orienté graphes : Dans la littérature, plusieurs approches pour l'implantation d'entrepôts NoSQL orientés colonnes et documents ont été proposées. Par conséquent, il serait pertinent de proposer des règles de transformation permettant la migration à partir de ces entrepôts de données vers un entrepôt orienté graphes sans perte d'informations ;

— Étude approfondie de la performance d'une approche d'apprentissage basée les agrégations : Les bases de données orientées graphes permettent une navigation intuitive et rapide entre les données. Les liens entre les données classiquement cherchées à l'aide des jointures sont stockés dans la base de données graphes et utilisés pour une meilleure prise de décision. Dans notre étude, agréger les valeurs portées par les arcs s'est avéré pertinent lors de l'apprentissage. Toutefois, le temps de calcul et les ressources utilisées peuvent augmenter considérablement lorsque les longueurs de chemins augmentent. Ainsi, une étude de performance détaillée est pertinente pour quantifier le gain tout en considérant le temps requis et les ressources utilisées ;

— Prise en considération d'autres métriques d'évaluation : Les métriques utilisées

pour valider et évaluer nos différentes propositions sont les plus utilisées dans la littérature, tels que le temps de calcul, la racine carrée de l'erreur quadratique moyenne et la précision. Néanmoins, d'autres métriques pourraient être utilisées afin de mesurer l'impact environnemental de nos approches, comme par exemple la consommation d'énergie lors de l'apprentissage.

Bibliographie

Aboutorabi[a], S. H., Rezapour, M., Moradi, M., and Ghadiri, N. (2015). Performance evaluation of sql and mongodb databases for big e-commerce data. In *2015 International Symposium on Computer Science and Software Engineering (CSSE)*, pages 1–7. IEEE.

Adamic, L. A. and Adar, E. (2003). Friends and neighbors on the web. *Social networks*, 25(3) :211–230.

Agrawal, S. and Patel, A. (2016). Astudy on graph storage database of nosql. *International Journal on Soft Computing, Artificial Intelligence and Applications (IJSCAI)*, 5(1) :33–39.

Ajah, I. A. and Nweke, H. F. (2019). Big data and business analytics : Trends, platforms, success factors and applications. *Big Data and Cognitive Computing*, 3(2) :32.

Akid, H. and Ayed, M. B. (2016). Towards nosql graph data warehouse for big social data analysis. In *International Conference on Intelligent Systems Design and Applications*, pages 965–973. Springer.

Al Hasan, M., Chaoji, V., Salem, S., and Zaki, M. (2006). Link prediction using supervised learning. In *SDM06 : workshop on link analysis, counter-terrorism and security*, volume 30, pages 798–805.

Alm, R. and Imeri, L. (2021). A performance comparison between graph databases : Degree project about the comparisonbetween neo4j, graphdb and orientdb on different operations.

Almabdy, S. (2018). Comparative analysis of relational and graph databases for social networks. In *2018 1st International Conference on Computer Applications & Information Security (ICCAIS)*, pages 1–4. IEEE.

Angles, R. (2018). The property graph database model. In *Proceedings of the 12th Alberto Mendelzon International Workshop Foundations of Data Management*, volume 2100.

Angles, R. and Gutierrez, C. (2008). Survey of graph database models. *ACM Computing Surveys (CSUR)*, 40(1) :1–39.

Angles, R., Prat-Pérez, A., Dominguez-Sal, D., and Larriba-Pey, J.-L. (2013). Benchmarking database systems for social network applications. In *First International Workshop on Graph Data Management Experiences and Systems*, pages 1–7.

Angles, R., Thakkar, H., and Tomaszuk, D. (2019). Rdf and property graphs interoperability : Status and issues. *AMW*, 2369.

Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *science*, 286(5439) :509–512.

Barzel, B. and Barabási, A.-L. (2013). Network link prediction by global silencing of indirect correlations. *Nature biotechnology*, 31(8) :720–725.

Batra, S. and Tyagi, C. (2012). Comparative analysis of relational and graph databases. *International Journal of Soft Computing and Engineering*, 2(2) :509–512.

Beis, S., Papadopoulos, S., and Kompatsiaris, Y. (2015). Benchmarking graph databases on the problem of community detection. In *New Trends in Database and Information Systems II*, pages 3–14. Springer.

Bentayeb, F. (2011). *Entrepôts et analyse en ligne de données complexes centrés utilisateur : un nouveau défi*. PhD thesis, Université Lumière-Lyon II.

Berggård, T., Linse, S., and James, P. (2007). Methods for the detection and analysis of protein–protein interactions. *Proteomics*, 7(16) :2833–2842.

Bernasconi, A., Ceri, S., Campi, A., and Masseroli, M. (2017). Conceptual modeling for genomics : building an integrated repository of open data. In *International Conference on Conceptual Modeling*, pages 325–339. Springer.

Birlutiu, A., d'Alché Buc, F., and Heskes, T. (2012). https ://doi.org/combining protein and network topology information for predicting protein-protein interactions : a bayesian approach.

Bordoloi, S. and Kalita, B. (2013). Designing graph database models from existing relational databases. *International Journal of Computer Applications*, 74(1).

Boussahoua, M., Boussaid, O., and Bentayeb, F. (2017). Logical schema for data warehouse on column-oriented nosql databases. In *International Conference on Database and Expert Systems Applications*, pages 247–256. Springer.

Brouard, C. (2013). *Inférence de réseaux d'interaction protéine-protéine par apprentissage statistique*. PhD thesis, Université d'Évry-Val-d'Essonne.

Buerli, M. and Obispo, C. (2012). The current state of graph databases. *Department of Computer Science, Cal Poly San Luis Obispo, mbuerli@ calpoly. edu*, 32(3) :67–83.

Castelltort, A. and Laurent, A. (2014). Nosql graph-based olap analysis. In *KDIR : Knowledge Discovery and Information Retrieva*, pages 217–224.

Chandra, D. G. (2015). Base analysis of nosql database. *Future Generation Computer Systems*, 52 :13–21.

Chavalier, M., El Malki, M., Kopliku, A., Teste, O., and Tournier, R. (2016). Document-oriented data warehouses : Models and extended cuboids, extended cuboids in oriented document. In *2016 IEEE Tenth International Conference on Research Challenges in Information Science*, pages 1–11. IEEE.

Chen, Y., Wang, W., Liu, J., Feng, J., and Gong, X. (2020). Protein interface complementarity and gene duplication improve link prediction of protein-protein interaction network. *Frontiers in genetics*, 11 :291.

Chevalier, M., El Malki, M., Kopliku, A., Teste, O., and Tournier, R. (2015a). Implementation of multidimensional databases with document-oriented nosql. In *International Conference on Big Data Analytics and Knowledge Discovery*, pages 379–390. Springer.

Chevalier, M., El Malki, M., Kopliku, A., Teste, O., and Tournier, R. (2015b). Implementing multidimensional data warehouses into nosql. *Proceedings of the 17th International Conference on Enterprise Information Systems*, pages 172–183.

Chouder, M. L., Rizzi, S., and Chalal, R. (2019). Exodus : Exploratory olap over document stores. *Information Systems*, 79 :44–57.

Codd, E. F. (2002). A relational model of data for large shared data banks. In *Software pioneers*, pages 263–294. Springer.

Connolly, T. M. and Begg, C. E. (2005). *Database systems : a practical approach to design, implementation, and management*. Pearson Education.

Consortium, U. (2015). Uniprot : a hub for protein information. *Nucleic acids research*, 43(D1) :D204–D212.

Corbellini, A., Mateos, C., Zunino, A., Godoy, D., and Schiaffino, S. (2017). Persisting big-data : The nosql landscape. *Information Systems*, 63 :1–23.

Davoudian, A., Chen, L., and Liu, M. (2018). A survey on nosql stores. *ACM Computing Surveys (CSUR)*, 51(2) :1–43.

De Virgilio, R., Maccioni, A., and Torlone, R. (2013). Converting relational to graph databases. In *First International Workshop on Graph Data Management Experiences and Systems*, pages 1–6.

De Virgilio, R., Maccioni, A., and Torlone, R. (2014). R2g : a tool for migrating relations to graphs. In *Proceedings of the 17th International Conference on Extending Database Technology*, pages 640–643.

Dehdouh, K. (2016). Building olap cubes from columnar nosql data warehouses. In *International Conference on Model and Data Engineering*, pages 166–179. Springer.

Dehdouh, K., Bentayeb, F., Boussaid, O., and Kabachi, N. (2015). Using the column oriented nosql model for implementing big data warehouses. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, page 469. The Steering Committee of The World Congress in Computer Science, Computer,Computer Engineering and Applied Computing (WorldComp).

Dehdouh, K., Boussaid, O., and Bentayeb, F. (2020). Big data warehouse : Building columnar nosql olap cubes. *International Journal of Decision Support System Technology (IJDSST)*, 12(1) :1–24.

Diallo, M. S., Mokeddem, S. A., Braud, A., Frey, G., and Lachiche, N. (2021). Identifying benchmarks for failure prediction in industry 4.0. In *Informatics*, volume 8, page 68. Multidisciplinary Digital Publishing Institute.

Fabregat, A., Korninger, F., Viteri, G., Sidiropoulos, K., Marin-Garcia, P., Ping, P., Wu, G., Stein, L., D'Eustachio, P., and Hermjakob, H. (2018). Reactome graph database : Efficient access to complex pathway data. *PLoS computational biology*, 14(1) :e1005968.

Fernandes, D. and Bernardino, J. (2018). Graph databases comparison : Allegrograph, arangodb, infinitegraph, neo4j, and orientdb. In *Data*, pages 373–380. [En ligne ; Consulté le 20-octobre-2022].

Franceschini, A., Szklarczyk, M. D., RUnit, S., and biocViews Network, B. (2015). Package 'stringdb'. http://www.mabs.ups-tlse.fr/site/images/c/c4/RSTRINGdb.man.pdf. [En ligne ; Consulté le 19-octobre-2022].

Gallinucci, E., Golfarelli, M., and Rizzi, S. (2019). Approximate olap of document-oriented databases : A variety-aware approach. *Information Systems*.

Gao, F., Musial, K., Cooper, C., and Tsoka, S. (2015). Link prediction methods and their accuracy for different social networks and network metrics. *Scientific programming*, 2015.

Golfarelli, M. and Rizzi, S. (2018). From star schemas to big data : 20+ years of data warehouse research. In *A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years*, pages 93–107. Springer.

Gómez, P., Casallas, R., and Roncancio, C. (2016). Data schema does matter, even in nosql systems ! In *2016 IEEE Tenth International Conference on Research Challenges in Information Science*, pages 1–6. IEEE.

Gosain, A. and Singh, J. (2015). Conceptual multidimensional modeling for data warehouses : a survey. In *Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing : Theory and Applications (FICTA) 2014*, pages 305–316. Springer.

Guia, J., Soares, V. G., and Bernardino, J. (2017). Graph databases : Neo4j analysis. In *ICEIS (1)*, pages 351–356.

Gupta, A., Tyagi, S., Panwar, N., Sachdeva, S., and Saxena, U. (2017). Nosql databases : Critical analysis and comparison. In *2017 International conference on computing and communication technologies for smart nation (IC3TSN)*, pages 293–299. IEEE.

Güting, R. H. (1994). Graphdb : Modeling and querying graphs in databases. In *the 20th International Conference on Very Large Data Bases*, volume 94, pages 12–15. Citeseer.

Halstead, R. J., Sukhwani, B., Min, H., Thoennes, M., Dube, P., Asaad, S., and Iyer, B. (2013). Accelerating join operation for relational databases with fpgas. In *2013 IEEE 21st Annual International Symposium on Field-Programmable Custom Computing Machines*, pages 17–20. IEEE.

Hamoud, A., Hashim, A. S., and Awadh, W. A. (2018). Clinical data warehouse : a review. *Iraqi Journal for Computers and Informatics*, 44(2).

Hamouda, S. and Zainol, Z. (2017). Document-oriented data schema for relational database migration to nosql. In *2017 International conference on big data innovations and applications (innovate-data)*, pages 43–50. IEEE.

Hanine, M., Bendarag, A., and Boutkhoum, O. (2015). Data migration methodology from relational to nosql databases. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 9(12) :2566–2570.

Himmelstein, D. S., Lizee, A., Hessler, C., Brueggeman, L., Chen, S. L., Hadley, D., Green, A., Khankhanian, P., and Baranzini, S. E. (2017). Systematic integration of biomedical knowledge prioritizes drugs for repurposing. *Elife*, 6 :e26726.

Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37 :547–579.

Jaiswal, G. and Agrawal, A. P. (2013). Comparative analysis of relational and graph databases. *IOSR Journal of Engineering (IOSRJEN)*, 3(8) :25–27.

Jia, T., Zhao, X., Wang, Z., Gong, D., and Ding, G. (2016). Model transformation and data migration from relational database to mongodb. In *2016 IEEE International Congress on Big Data (BigData Congress)*, pages 60–67. IEEE.

Jolly, K. (2018). *Machine Learning with scikit-learn Quick Start Guide : Classification, regression, and clustering techniques in Python*. Packt Publishing Ltd.

Jouili, S. and Vansteenberghe, V. (2013). An empirical comparison of graph databases. In *2013 International Conference on Social Computing*, pages 708–715. IEEE.

Kamada, M., Sakuma, Y., Hayashida, M., and Akutsu, T. (2014). Prediction of protein-protein interaction strength using domain features with supervised regression. *The Scientific World Journal*, 2014.

Kanwar, R., Trivedi, P., and Singh, K. (2013). Nosql, a solution for distributed database management system. *International journal of computer applications*, 67(2).

Kimball, R. (1996). *The data warehouse toolkit : practical techniques for building dimensional data warehouses*. John Wiley & Sons, Inc.

Kotu, V. and Deshpande, B. (2018). *Data science : concepts and practice*. Morgan Kaufmann.

Kovács, I. A., Luck, K., Spirohn, K., Wang, Y., Pollis, C., Schlabach, S., Bian, W., Kim, D.-K., Kishore, N., Hao, T., et al. (2019). Network-based prediction of protein interactions. *Nature communications*, 10(1) :1–8.

Kumari, A., Behera, R. K., Sahoo, K. S., Nayyar, A., Kumar Luhach, A., and Prakash Sahoo, S. (2022). Supervised link prediction using structured-based feature extraction in social network. *Concurrency and Computation : practice and Experience*, 34(13) :e5839.

Lande, D., Fu, M., Guo, W., Balagura, I., Gorbov, I., and Yang, H. (2020). Link prediction of scientific collaboration networks based on information retrieval. *World Wide Web*, 23(4) :2239–2257.

Lasolle, N., Bruneau, O., Lieber, J., Nauer, E., and Pavlova, S. (2021). Assister l'édition manuelle de données rdf à l'aide du raisonnement à partir de cas. In *IC 2021-32ème Journées Francophones d'Ingénierie des Connaissances*.

Lee, C.-H. and Zheng, Y.-L. (2015). Sql-to-nosql schema denormalization and migration : a study on content management systems. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 2022–2026. IEEE.

Lee, S., Park, B. H., Lim, S.-H., and Shankar, M. (2015). Table2graph : A scalable graph construction from relational tables using map-reduce. In *2015 IEEE First International Conference on Big Data Computing Service and Applications*, pages 294–301. IEEE.

Li, C. (2010). Transforming relational database into hbase : A case study. In *2010 IEEE international conference on software engineering and service sciences*, pages 683–687. IEEE.

Li, S., Huang, J., Zhang, Z., Liu, J., Huang, T., and Chen, H. (2018). Similarity-based future common neighbors model for link prediction in complex networks. *Scientific reports*, 8(1) :1–11.

Lichtenwalter, R. N., Lussier, J. T., and Chawla, N. V. (2010). New perspectives and methods in link prediction. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 243–252.

Liu, H., Kou, H., Yan, C., and Qi, L. (2019). Link prediction in paper citation network to construct paper correlation graph. *EURASIP Journal on Wireless Communications and Networking*, 2019(1) :1–12.

Liu, W. and Lü, L. (2010). Link prediction based on local random walk. *EPL (Europhysics Letters)*, 89(5) :58007.

Lysenko, A., Roznovăţ, I. A., Saqi, M., Mazein, A., Rawlings, C. J., and Auffray, C. (2016). Representing and querying disease networks using graph databases. *BioData mining*, 9(1) :1–19.

Majidian, S. and Raeesi Vanani, I. (2019). *Literature Review on Big Data Analytics Methods*, pages 1–17.

MAKAME, A. A. (2018). *Evaluation and Analysis on Neo4J and OrientDB Graph Database*. PhD thesis, Universitas Gadjah Mada.

Matthes, F., Schulz, C., and Haller, K. (2011). Testing & quality assurance in data migration projects. In *2011 27th IEEE international conference on software maintenance (ICSM)*, pages 438–447. IEEE.

Megid, Y. A., El-Tazi, N., and Fahmy, A. (2018). Using functional dependencies in conversion of relational databases to graph databases. In *International Conference on Database and Expert Systems Applications*, pages 350–357. Springer.

Mercorio, F., Mezzanzanica, M., Moscato, V., Picariello, A., and Sperlì, G. (2019). A tool for researchers : Querying big scholarly data through graph databases. In *ECML/PKDD (3)*, pages 760–763.

Mezzanzanica, M., Mercorio, F., Cesarini, M., Moscato, V., and Picariello, A. (2018). Graphdblp : a system for analysing networks of computer scientists through graph databases. *Multimedia Tools and Applications*, 77(14) :18657–18688.

Miller, J. J. (2013). Graph database applications and concepts with neo4j. In *Proceedings of the southern association for information systems conference, Atlanta, GA, USA*, volume 2324.

Mishra, P. and Eich, M. H. (1992). Join processing in relational databases. *ACM Computing Surveys (CSUR)*, 24(1) :63–113.

Mistry, D., Wise, R. P., and Dickerson, J. A. (2017). Diffslc : A graph centrality method to detect essential proteins of a protein-protein interaction network. *PloS one*, 12(11) :e0187091.

Moosavi, S. A., Jalali, M., Misaghian, N., Shamshirband, S., and Anisi, M. H. (2017). Community detection in social networks using user frequent pattern mining. *Knowledge and Information Systems*, 51(1) :159–186.

Mostajabi, F., Safaei, A. A., and Sahafi, A. (2021). A systematic review of data models for the big data problems. *IEEE Access*.

Mullen, J., Cockell, S. J., Woollard, P., and Wipat, A. (2016). An integrated data driven approach to drug repositioning using gene-disease associations. *PloS one*, 11(5) :e0155811.

Mutlu, E. C. and Oghaz, T. A. (2019). Review on graph feature learning and feature extraction techniques for link prediction. *arXiv preprint arXiv :1901.03425*.

Nambiar, R. O. and Poess, M. (2006). The making of tpc-ds. In *Proceedings of the 32nd international conference on Very large data bases*, pages 1049–1058. VLDB Endowment.

Nayak, A., Poriya, A., and Poojary, D. (2013). Type of nosql databases and its comparison with relational databases. *International Journal of Applied Information Systems*, 5(4) :16–19.

Newman, M. E. (2001). Clustering and preferential attachment in growing networks. *Physical review E*, 64(2) :025102.

Ngo, V. M., Le-Khac, N.-A., Kechadi, M., et al. (2019). Designing and implementing data warehouse for agricultural big data. In *International Conference on Big Data*, pages 1–17. Springer.

Orel, O., Zakošek, S., and Baranovič, M. (2016). Property oriented relational-to-graph database conversion. *automatika*, 57(3) :836–845.

Oughtred, R., Rust, J., Chang, C., Breitkreutz, B.-J., Stark, C., Willems, A., Boucher, L., Leung, G., Kolas, N., Zhang, F., et al. (2021). The biogrid database : A comprehensive biomedical resource of curated protein, genetic, and chemical interactions. *Protein Science*, 30(1) :187–200.

Park, Y. and Bader, J. S. (2011). Resolving the structure of interactomes with hierarchical agglomerative clustering. *BMC bioinformatics*, 12(1) :1–10.

Paul, M. and Anand, A. (2020). Impact of low-confidence interactions on computational identification of protein complexes. *Journal of Bioinformatics and Computational Biology*, 18(04) :2050025.

Poess, M., Nambiar, R. O., and Walrath, D. (2007). Why you should run tpc-ds : a workload analysis. In *Proceedings of the 33rd international conference on Very large data bases*, pages 1138–1149. VLDB Endowment.

Pokorny, J. (2013). Nosql databases : a step to database scalability in web environment. *International Journal of Web Information Systems*, 9(1) :69–82.

Qu, W. and Dessloch, S. (2017). Distributed snapshot maintenance in wide-column nosql databases using partitioned incremental etl pipelines. *Information Systems*, 70 :48–58.

Raman, R., Hong, S., and Chafi, H. (2017). Graph data processing system that supports automatic data model conversion from resource description framework to property graph. US Patent App. 14/812,819.

Ravat, F., Teste, O., Tournier, R., and Zurfluh, G. (2008). Algebraic and graphic languages for olap manipulations. *International Journal of Data Warehousing and Mining (IJDWM)*, 4(1) :17–46.

Robinson, I., Webber, J., and Eifrem, E. (2015). *Graph databases : new opportunities for connected data*. " O'Reilly Media, Inc.".

Rolland, T., Taşan, M., Charloteaux, B., Pevzner, S. J., Zhong, Q., Sahni, N., Yi, S., Lemmens, I., Fontanillo, C., Mosca, R., et al. (2014). A proteome-scale map of the human interactome network. *Cell*, 159(5) :1212–1226.

Sahatqija, K., Ajdari, J., Zenuni, X., Raufi, B., and Ismaili, F. (2018). Comparison between relational and nosql databases. In *2018 41st international convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 0216–0221. IEEE.

Santos, M. Y. and Costa, C. (2016). Data models in nosql databases for big data contexts. In *International Conference on Data Mining and Big Data*, pages 475–485. Springer.

Scabora, L. C., Brito, J. J., Ciferri, R. R., and Ciferri, C. D. d. A. (2016). Physical data warehouse design on nosql databases. In *Proceedings of the 18th International Conference on Enterprise Information Systems*, pages 111–118. SCITEPRESS-Science and Technology Publications, Lda.

Schuh, S., Chen, X., and Dittrich, J. (2016). An experimental comparison of thirteen relational equi-joins in main memory. In *Proceedings of the 2016 International Conference on Management of Data*, pages 1961–1976.

Sellami, A., Nabli, A., and Gargouri, F. (2018). Transformation of data warehouse schema to nosql graph data base. In *International Conference on Intelligent Systems Design and Applications*, pages 410–420. Springer.

Sellami, A., Nabli, A., and Gargouri, F. (2020). Graph nosql data warehouse creation. In *Proceedings of the 22nd International Conference on Information Integration and Web-based Applications & Services*, pages 34–38.

Serrano, D., Han, D., and Stroulia, E. (2015). From relations to multi-dimensional maps : towards an sql-to-hbase transformation methodology. In *2015 IEEE 8th International Conference on Cloud Computing*, pages 81–89. IEEE.

Singh, A. (2019). Data migration from relational database to mongodb. *Global Journal of Computer Science and Technology*.

Soransso, R. and Cavalcanti, M. C. (2018). Data modeling for analytical queries on document-oriented dbms. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pages 541–548. ACM.

Srilatha, P. and Manjula, R. (2016). Similarity index based link prediction algorithms in social networks : A survey. *Journal of Telecommunications and Information Technology*.

Szklarczyk, D., Gable, A. L., Nastou, K. C., Lyon, D., Kirsch, R., Pyysalo, S., Doncheva, N. T., Legeay, M., Fang, T., Bork, P., et al. (2021). The string database in 2021 : customizable protein–protein networks, and functional characterization of user-uploaded gene/measurement sets. *Nucleic acids research*, 49(D1) :D605–D612.

Teste, O. (2000). *Modélisation et manipulation d'entrepôts de données complexes et historisées*. PhD thesis, Université Paul Sabatier-Toulouse III.

Timón-Reina, S., Rincón, M., and Martínez-Tomás, R. (2021). An overview of graph databases and their applications in the biomedical domain. *Database*, 2021.

Unal, Y. and Oguztuzun, H. (2018). Migration of data from relational database to graph database. In *Proceedings of the 8th International Conference on Information Systems and Technologies*, pages 1–5.

Valduriez, P. (1987). Join indices. *ACM Transactions on Database Systems (TODS)*, 12(2) :218–246.

Vicario, G. and Coleman, S. (2020). A review of data science in business and industry and a future view. *Applied Stochastic Models in Business and Industry*, 36(1) :6–18.

Vicknair, C., Macias, M., Zhao, Z., Nan, X., Chen, Y., and Wilkins, D. (2010). A comparison of a graph database and a relational database : a data provenance perspective. In *Proceedings of the 48th annual Southeast regional conference*, page 42. ACM.

Wang, P., Xu, B., Wu, Y., and Zhou, X. (2015). Link prediction in social networks : the state-of-the-art. *Science China Information Sciences*, 58(1) :1–38.

Wang, Z., Liang, J., Li, R., and Qian, Y. (2016). An approach to cold-start link prediction : Establishing connections between non-topological and topological information. *IEEE Transactions on Knowledge and Data Engineering*, 28(11) :2857–2870.

Wißing, C., Rougier, H., Baumann, C., Comeyne, A., Crevecoeur, I., Drucker, D. G., Gaudzinski-Windheuser, S., Germonpré, M., Gómez-Olivencia, A., Krause, J., et al. (2019). Stable isotopes reveal patterns of diet and mobility in the last neandertals and first modern humans in europe. *Scientific reports*, 9(1) :1–12.

Yang, F., Fan, K., Song, D., and Lin, H. (2020). Graph-based prediction of protein-protein interactions with attributed signed graph embedding. *BMC bioinformatics*, 21(1) :1–16.

Yao, L., Wang, L., Pan, L., and Yao, K. (2016). Link prediction based on common-neighbors for dynamic social network. *Procedia Computer Science*, 83 :82–89.

Yoon, B.-H., Kim, S.-K., and Kim, S.-Y. (2017). Use of graph database for the integration of heterogeneous biological data. *Genomics & informatics*, 15(1) :19.

Zhou, T., Lü, L., and Zhang, Y.-C. (2009). Predicting missing links via local information. *The European Physical Journal B*, 71(4) :623–630.

Université de Sfax
École Nationale d'Ingénieurs de Sfax
École doctorale des Sciences et de la Technologie

Université de Strasbourg
École doctorale n° 269 : Mathématiques, Sciences de l'Information et de l'Ingénieur

Résumé

Avec l'essor du big data, le recours à la science des données est devenu crucial pour extraire les connaissances à partir des données massives. Ces dernières années, plusieurs catégories de bases de données ont été utilisées en science des données pour stocker, gérer et analyser efficacement les mégadonnées. En particulier, les bases de données NoSQL (Not Only SQL) ont émergé comme une alternative aux bases de données relationnelles. Dans cette thèse, nous nous intéressons à la catégorie orientée graphes des bases de données NoSQL qui remplace les jointures relationnelles coûteuses par des parcours du graphe. Nous les utilisons pour répondre à deux problématiques en science des données et qui constituent nos principales contributions : implanter des entrepôts de données capables de répondre rapidement à des requêtes analytiques complexes et construire des descripteurs afin de les utiliser pour prédire les valeurs manquantes sur les arcs d'un réseau d'interaction protéine-protéine.

Mots clés : Base de données relationnelles, bases de données graphes, entrepôts de données, construction de descripteurs, interaction protéine-protéine

Abstract

With the rise of big data, the use of data science has become critically important to extract insights from a huge amount of data. In recent years, several categories of databases have been used in data science to efficiently store, manage and analyze huge amounts of data. In particular, NoSQL (Not Only SQL) databases have emerged as an alternative to relational databases. In this thesis, we are interested in the graph-oriented category of NoSQL databases that replaces expensive relational joins with graph traversals. We use graph databases to address two data science issues of concern that are our main contributions: Implementing data warehouses capable of responding quickly to complex analytical queries and constructing descriptors to be used to predict missing values on the edges of a protein-protein interaction network.

Keywords: Relational databases, graph databases, data warehouses, feature construction, protein-protein Interaction

Université de Strasbourg

Hajer AKID

Université de Sfax

Les bases de données graphes pour a science des données : Implantation d'entrepôts de données et prédiction d'interactions protéine-protéine

Résumé

Avec l'essor du big data, le recours à la science des données est devenu crucial pour extraire les connaissances à partir des données massives. Ces dernières années, plusieurs catégories de bases de données ont été utilisées en science des données pour stocker, gérer et analyser efficacement les mégadonnées. En particulier, les bases de données NoSQL (Not Only SQL) ont émergé comme une alternative aux bases de données relationnelles. Dans cette thèse, nous nous intéressons à la catégorie orientée graphes des bases de données NoSQL qui remplace les jointures relationnelles coûteuses par des parcours du graphe. Nous les utilisons pour répondre à deux problématiques en science des données et qui constituent nos principales contributions : implanter des entrepôts de données capables de répondre rapidement à des requêtes analytiques complexes et construire des descripteurs afin de les utiliser pour prédire les valeurs manquantes sur les arcs d'un réseau d'interaction protéine-protéine.

Mots clés

Base de données relationnelles, bases de données graphes, entrepôts de données, construction de descripteurs, interaction protéine-protéine

Résumé en anglais

With the rise of big data, the use of data science has become critically important to extract insights from a huge amount of data. In recent years, several categories of databases have been used in data science to efficiently store, manage and analyze huge amounts of data. In particular, NoSQL (Not Only SQL) databases have emerged as an alternative to relational databases. In this thesis, we are interested in the graph-oriented category of NoSQL databases that replaces expensive relational joins with graph traversals. We use graph databases to address two data science issues of concern that are our main contributions: Implementing data warehouses capable of responding quickly to complex analytical queries and constructing descriptors to be used to predict missing values on the edges of a protein-protein interaction network.

Keywords

Relational databases, graph databases, data warehouses, feature construction, protein-protein Interaction