Anonymisation dans le contexte des graphes de connaissances

Maxime Thouvenot

Thèse de doctorat

Anonymisation dans le contexte des graphes de connaissances

présentée par THOUVENOT Maxime

soutenue le 6 décembre 2022

Myriam Lamolle	Professeure à l'Université de Paris 8	Rapporteur / Présidente
Benjamin Nguyen	Professeur à l'INSA Centre Val de Loire	Rapporteur
Philippe Calvez	Directeur du laboratoire CSAI CRIGEN Engie	Co-encadrant de thèse
Olivier Curé	Maître de conférences à l'Université Gustave Eiffel	Directeur de thèse

Remerciements

Je tiens tout d'abord à remercier Olivier CURÉ et Philippe CALVEZ de m'avoir accordé leur confiance en encadrant mon travail durant ces trois ans. Je salue la patience d'Olivier qui a toujours su se montrer disponible pour répondre à mes questions et m'aiguiller dans certains moments de doute. Philippe a également suivi avec attention l'avancée de mes travaux malgré qu'il ait parfois été difficile de se rencontrer du fait de la condition sanitaire. Je garde avec les deux d'excellents souvenirs de moments passés au sein du laboratoire mais également à l'extérieur.

Mes remerciements vont aussi à Myriam LAMOLLE et Benjamin NGUYEN pour avoir accepté d'être membres de mon comité de thèse. J'ai pleinement conscience du temps et du travail que le rôle de rapporteur représente et pour cela, je tiens à leur témoigner ma plus profonde gratitude.

Je souhaite remercier mes collègues chercheurs du Laboratoire d'Informatique Gaspard-Monge avec lesquels j'ai pu échanger tout au long de cette thèse, notamment dans le cadre de mes fonctions en tant que moniteur pour les étudiants de Licence. Je remercie particulièrement Claire DAVID, Nadime FRANCIS, Fabian REITER, Olivier BOUILLOT, Antoine MEYER et bien d'autres de m'avoir accompagné alors que je découvrais le métier de l'enseignement. J'étends également ces remerciements à mes collègues de bureau, Weiqin XU, Christophe CALLÉ et Joffrey DE OLIVEIRA avec qui j'ai pu partager des moments conviviaux au bureau comme à l'extérieur.

Enfin, je tiens à terminer cette partie en remerciant ma famille pour le soutien qu'elle m'a apporté pendant cette période, notamment au plus haut de la pandémie de Covid-19. Je réalise la chance que j'ai eu de pouvoir passer ce moment pénible en leur présence. Je leur suis reconnaissant d'avoir essayé de s'intéresser au de sujet de thèse alors même qu'ils n'y comprenaient rien (et qu'ils n'y comprennent toujours rien d'ailleurs !).

Table des matières

Table des figures

Liste des tableaux

Résumé

L'avènement d'Internet tel qu'il est apparu dans les années 1990 a permis la mise en place d'échanges de données dans des proportions comme jamais auparavant. Cette évolution en termes de volume, de vitesse et de diversité a atteint son paroxysme avec l'émergence du phénomène Big Data il y a un peu plus d'une quinzaine d'années. L'information est aujourd'hui une ressource extrêmement précieuse, stockée principalement à l'intérieur de centre de données afin d'être analysée dans le but de générer de nouvelles connaissances. Parallèlement à ce déluge de données, la dernière décennie a également vu naître le mouvement Open Data plaidant pour davantage de transparence et de partage des données de la part des acteurs publics et privés. Cela s'est traduit par la publication d'un large nombre de jeux de données ainsi que la mise en ligne de plusieurs plate-formes permettant de rendre plus accessibles des ressources pouvant relever de l'intérêt général, e.g. data.gouv.

Néanmoins, un problème de taille est qu'une partie non négligeable de ces données peuvent concerner des individus et correspondre à ce que l'on appelle des informations "sensibles" telles que le salaire, l'orientation sexuelle, la religion, etc. Face à ce défi, l'anonymisation s'est vite imposée comme un processus essentiel. D'abord un sujet abordé par la communauté des statistiques mathématiques dans les années 80/90, les informaticiens s'en sont finalement emparés au tournant des années 2000, leur travail aboutissant à la création de nombreux principes de confidentialité e.g. k-anonymity, l-diversity, differential privacy, etc.

Pour chacune de ces méthodes se pose la difficulté de trouver un compromis entre la confidentialité des données et leur utilité. D'une part, le jeu de données doit être transformé de manière à empêcher l'établissement de tout lien entre une information sensible et un individu ou une organisation mais de l'autre, cela ne doit pas se faire au détriment de la capacité à analyser le jeu de données a posteriori.

Plusieurs approches d'anonymisation ont déjà été proposées cependant la majorité d'entre elles tentent de traiter le problème dans le contexte de bases de données suivant le modèle relationnel. Les modèles fondés sur les graphes ont en comparaison fait l'objet de beaucoup moins de travaux, en particulier ceux concernant le modèle de données RDF (Resource Description Framework) du Web des données. Dans le cadre de cette thèse, nos recherches ont principalement porté sur l'anonymisation de graphe des connaissances (*knowledge graphs*) et notamment l'adaptation de techniques développées à l'origine pour le modèle

relationnel.

Dans un premier temps, nous avons repris une technique nommée anatomie, dont le principe consiste à supprimer le lien direct entre une entité et son attribut sensible, et l'avons étendu en y intégrant des aspects sémantiques.

Ce travail a par la suite été approfondi avec l'intégration d'un autre principe de confidentialité, à savoir k-anonymity. Nous défendons le bien-fondé d'utiliser les deux approches simultanément en expliquant de quelle manière cela peut permettre de prévenir un certain nombre d'attaques. Nous proposons également deux algorithmes pour appliquer k-anonymity sur un graphe RDF.

Pour finir, nous présentons un système placé au-dessus d'un système de gestion de base de données (SGBD) RDF afin d'anonymiser des graphes de connaissances. Conformément aux principes du *privacy-by-design*, nous prenons le parti que l'anonymisation est une tâche fastidieuse et difficile qui ne devrait pas relever de la responsabilité des ingénieurs mais être intégrée directement au sein d'un SGBD. En nous reposant sur une stratégie de partitionnement couplée à un mécanisme de contrôle d'accès, nous établissons une distinction entre d'un côté des données sensibles non-anonymisées, accessibles uniquement à des utilisateurs privilégiés, et de l'autre, leur équivalent anonymisé publié publiquement. L'anonymisation est déléguée à un autre composant capable de suivre les différentes modifications appliquées aux données au fur et à mesure afin de les anonymiser en conséquence.

Abstract

The advent of the Internet as it appeared in the 1990s allowed an increase in terms of data exchanges in proportions never seen before. This evolution in terms of volume, velocity and variety has reached its climax with the emergence of the Big Data phenomenon a little over fifteen years ago. Information is now, more than ever, an extremely valuable resource, stored mainly inside data centers in order to be analyzed so as to produce even more new knowledge. Alongside this data deluge, the last decade has also seen the birth of the Open Data movement advocating for more transparency and data sharing from public and private actors. This resulted in more publications and the setting up of various platforms to help make resources more easily accessible to the general public, e.g. data.gouv.

However, a major problem is that a significant part of this data may relate to individuals and correspond to "sensitive" information such as salary, sexual orientation, religion, etc. Faced with this challenge, anonymization quickly established itself as an essential computational process. First a subject tackled by the mathematical statistics community in the 80s/90s, computer scientists eventually took over at the turn of the 2000s, their work resulting in the creation of many privacy principles *e.g.*, k-anonymity, l-diversity, differential privacy and so on.

No matter the method one chooses, they must always address the issue of finding a trade-off between what is called data privacy and data utility. On the one hand, the dataset must be transformed in such a way as to prevent the linking of any sensitive information to an individual or an organization, but on the other hand, this must not be done to the detriment of the analyzes which could be run on top of the data.

Several anonymization approaches have already been proposed, however, the majority of them only considers the problem in the context of databases following the relational data model. Graph-based models have in comparison been the subject of much less research, more specifically those concerning the RDF (Resource Description Framework) data model. During the time of this thesis, our work has mainly focused on the anonymization of knowledge graphs and in particular the adaptation of techniques originally developed for the relational model.

At first, we adapted a technique called anatomy, the principle of which consists in removing the direct link between an entity and its sensitive attri-

bute, and we extended it by integrating semantic aspects.

This work was subsequently expanded upon with the integration of another privacy principle, namely k-anonymity. We defend the validity of using both approaches simultaneously by explaining how this can be used to prevent some attacks. We also propose two algorithms to apply k-anonymity over an RDF graph.

Finally, we present a system designed on top of an RDF database management system (DBMS) in order to anonymize knowledge graphs. In accordance with the principles of privacy-by-design, we argue that anonymization is a tedious and difficult task that should not be the responsibility of engineers but be integrated directly into a DBMS. By relying on a partitioning strategy coupled with an access control mechanism, we make a distinction between non-anonymized sensitive data, accessible only to privileged users, and their anonymized counterpart which is publicly released. The anonymization is delegated to another component capable of tracking the various updates which occured in the data in order to anonymize the graph accordingly.

Chapitre 1

Introduction

1.1 Contexte

L'avènement du Web tel que nous le connaissons aujourd'hui a marqué une révolution dans la manière dont les êtres humains peuvent communiquer entre eux. En premier lieu un système utilisé presque uniquement afin de consulter des pages statiques, le Web a finalement muté au tournant des années 2000 en un espace interactif dans lequel il est devenu commun de non seulement consulter des données mais également de participer à leur création et évolution. Dans ce que certains ont nommé le web 2.0, les utilisateurs sont ainsi passés d'un état de simple consommateur de contenu à celui de producteur actif, les fers de lance de cette nouvelle vision étant bien entendu les réseaux sociaux. Plus récemment, le phénomène du *Big Data*, rendu possible notamment par les progrès techniques

en matière d'infrastructures de stockage (e.g., data centers), a vu la production de données évoluer de manière inédite, que ce soit par rapport à son volume, sa vélocité ou bien même sa variété (textes, images, musiques, vidéos, etc).

Néanmoins, alors que l'on peine encore à mesurer toute l'étendue des possibilités offertes par cette transformation fulgurante en seulement 20 ans, la transition vers un nouveau Web, qualifié de "sémantique", se profile depuis un certain temps. Évoquée pour la première fois au début des années 90 par Tim Berners-Lee, il s'agit d'un projet visant à rendre l'information présente sur internet compréhensible par des machines et permettre par la suite à ces dernières d'effectuer des raisonnements. Pour réaliser cette ambition, de nouvelles structures de données et standards ont vu le jour, la pierre angulaire étant les graphes de connaissances fondés sur le formalisme RDF (Resource Description Framework). Les premiers cas d'utilisation pratique de ces technologies sont apparus il y a environ une quinzaine d'années avec la création d'immenses bases de connaissances telles que DBPedia dont le contenu est utilisé, entre autres, par le moteur de recherche Google afin d'affiner ses recherches. C'est dans le cadre de cette nouvelle extension du Web que nous nous plaçons pour effectuer nos travaux.

1.2 Problématique

Parallèlement à cette abondance de données s'est développé le mouvement de l'*Open Data* dont le but est de militer en faveur de plus de transparence et de distribution des données détenues par les différents acteurs (publics et éventuellement privés) de l'industrie. À l'instar de l'initiative *Open-Source* dont les revendications concernaient le libre accès ainsi que la distribution de code, les tenants de l'*Open-Data* considèrent les données comme étant une ressource précieuse dont la diffusion auprès d'agents extérieurs pourraient bénéficier au plus grand nombre. En ce sens, plusieurs actions ont été entreprises, on notera par exemple la mise en place par le gouvernement français d'une plateforme à l'adresse https://www.data.gouv.fr/ mettant à disposition les données publiques françaises. Des plateformes analogues pour les données européennes [1] ou américaines [2] existent également.

Toutefois, le problème est qu'une partie non négligeable de ces données concerne des individus ou organisations et pourrait représenter un danger pour la confidentialité de ces derniers. On parle dans ce cas de *données sensibles*, celles-ci pouvant apparaître sous diverses formes : salaire, orientation sexuelle, religion, état de santé ou encore opinions politiques. Afin de continuer à tirer des bénéfices du Big Data tout en protégeant les entités contenues dans les données, l'anonymisation s'est vite imposée comme une solution indispensable. L'objectif de ce processus s'articule autour de deux concepts : la confidentialité des données (*data privacy)* ainsi que leur utilité (*data utility*). D'un côté, nous voulons

1. https://data.europa.eu/fr
2. https://data.gov/

transformer les jeux de données de manière à empêcher tout attaquant de découvrir une quelconque information compromettante. De l'autre, nous voulons préserver au maximum l'intégrité de ces jeux en vue de poursuivre nos analyses et d'en extraire de nouvelles informations pertinentes.

D'abord un domaine exploré majoritairement par la communauté des mathématiques statistiques, l'anonymisation est finalement devenue, à partir du début des années 2000, un thème de recherche particulièrement intéressant pour les informaticiens. Cela a débouché entre autres sur la création d'un ensemble de principes de confidentialité dédiés au modèle de données relationnel. Par ailleurs, la prise de conscience par certaines autorités publiques du danger que représentent les données sensibles a permis l'élaboration, puis la mise en vigueur, de plusieurs textes de loi et régulations. Pour les pays membres de l'Union européenne, le cadre légal vis-à-vis de l'exploitation des données personnelles est fixé par le *Règlement général sur la protection des données*[3] (RGPD) promulgué en 2016 et entré en vigueur en mai 2018. Ce texte a notamment servi d'inspiration pour la rédaction du *California Consumer Privacy Act*[4] (CCPA) sur la protection des données dans l'état de Californie aux États-Unis.

Tout au long de cette thèse, nous nous concentrons sur le problème de l'anonymisation dans le cadre particulier des graphes de connaissances fondés sur le modèle de données RDF. Partant de ce point, nous dégageons trois grandes questions qui structureront le reste de cette thèse de doctorat.

1.3 Questions de recherche

1.3.1 Question 1 : adapter une technique existante pour le modèle de données relationnel au modèle RDF et intégrer les aspects sémantiques propres aux graphes de connaissances à cette technique d'anonymisation

L'essentiel des études sur l'anonymisation produit depuis environ une vingtaine d'années a été effectué en considérant le modèle de données relationnel. Si des travaux s'intéressant à ce problème pour les graphes existent, ils restent minoritaires et s'appliquent principalement à des graphes sociaux (*i.e.*, avec un unique type de propriété entre les noeuds, *e.g.*, *likes* d'un réseau tel que Facebook). Les articles de recherche se plaçant dans le cadre spécifique des graphes de connaissances représentent quant à eux une part encore plus infime.

Néanmoins nous avons été surpris de remarquer que ces approches, bien qu'abordant le sujet de manière différente, avaient généralement pour point commun de négliger les aspects sémantiques qui font la particularité du Web sémantique. Dès lors, nous avons commencé à nous interroger sur les avantages que l'intégration de tels mécanismes (qui font la singularité des graphes de connaissances) pourrait apporter à une solution d'anonymisation. Partant de ce constat, nous nous sommes fixé trois objectifs.

3. https://gdpr-info.eu/
4. https://oag.ca.gov/privacy/ccpa

Concernant l'adaptation d'une approche déjà connue pour le modèle relationnel aux graphes de connaissances, ceci impliquerait d'opérer plusieurs modifications, notamment pour la prise en compte de la structure des données en elles-mêmes et des moyens de les requêter (le langage SPARQL adapté à RDF, bien qu'il s'inspire en partie de la syntaxe de SQL, reste fondamentalement différent).

Cette première étape accomplie, il est nécessaire de déterminer de quelle manière la technique en question pourrait être étendue par l'incorporation d'aspects sémantiques. Nous observons généralement que dans un souci de préserver l'utilité des données, les approches de généralisation classiques accordent un soin particulier à essayer de regrouper des entités similaires entre elles. L'intégration du raisonnement pourrait-elle aider à guider ce type de processus ?

Enfin, le dernier point qui nous intéresse concerne l'évaluation de notre approche à l'égard de l'utilité des données. Étant donné un jeu anonymisé, dans quelle mesure pouvons-nous le requêter et espérer récupérer des résultats pertinents ?

1.3.2 Question 2 : comment pouvons-nous étendre nos précédents travaux de manière à répondre à différents types d'attaque ?

Les attaques à l'encontre d'un jeu de données peuvent être de natures très variées. En fonction de l'angle qu'il choisit, un adversaire peut ainsi cibler différentes sections d'un jeu de façon à découvrir les données personnelles d'une entité. Les techniques d'anonymisation classiques font généralement le choix de se focaliser sur un seul type d'attaque et de ne pas considérer les autres.

En reprenant l'approche développée lors de la question précédente qui s'attelle à dissimuler les valeurs des attributs sensibles, nous nous sommes demandés s'il serait intéressant de l'associer à une autre méthode, conçue pour faire face à un autre type d'attaque. Se faisant, serait-il possible de combler certaines des faiblesses de chacune des approches afin d'améliorer la sécurité fournie aux entités présentes dans le graphe ?

De nouveau se pose également le sujet de l'utilité des données. En privilégiant davantage la confidentialité des données dans cette question, est-on encore capable d'exploiter les données de façon satisfaisante ?

1.3.3 Question 3 : comment peut-on intégrer l'anonymisation au coeur même d'un SGBD, au plus proche des données ?

La protection de la vie privée dès la conception, *privacy by design* en anglais, est une approche de l'ingénierie des systèmes dont les principes concernent l'imbrication de contrôles de protection des données au cœur des systèmes traitant de données personnelles. L'anonymisation étant un processus lourd, certains supposent qu'en déléguer la responsabilité à des systèmes dédiés pourrait faciliter l'adoption de ces pratiques par l'industrie. C'est conformément à ces principes

que nous avons souhaité créer un framework pour l'anonymisation de données RDF pouvant s'intégrer facilement avec un SGBD quelconque.

Dans les deux questions précédentes, nous nous étions situés dans un contexte "statique" en ne considérant qu'une seule itération d'un jeu de données fixe. Or, en situation réelle, les données sont généralement dynamiques et évoluent dans le temps aux travers de suppressions, d'ajouts ou de modifications. Cette différence ouvre la voie à de multiples attaques que les techniques classiques sont incapables de gérer. On s'intéresse donc ici à des propositions conçues dans le but de prendre en compte plusieurs itérations d'un même jeu et de garantir la sécurité des entités qui y apparaissent. Comme pour la question de recherche 1, nous souhaitons nous appuyer sur une approche existante, il en existe quelques-unes, mais un travail d'adaptation pour le modèle RDF reste quoi qu'il en soit indispensable.

Par ailleurs, le fait de travailler avec un SGBD nous place dans un environnement contrôlé dans lequel nous pouvons proposer de nouvelles fonctionnalités, par exemple, la restriction des droits d'un utilisateur en fonction de son rôle dans la base. L'anonymisation entraînant forcément une dégradation de l'intégrité des données, il serait intéressant qu'un utilisateur privilégié puisse accéder aux données intactes tandis que l'utilisateur moins privilégié ne pourrait accéder qu'à leur version anonymisée. Une telle distinction impliquerait de fait plusieurs choses. Tout d'abord, il faudrait que les données soient partitionnées en deux ensembles séparés : l'un privé, l'autre public. En outre, l'accès à chacune de ces parties devrait être soumis à la vérification du rôle de l'utilisateur. Cela présuppose donc l'existence d'un composant de contrôle d'accès. Enfin, si deux partitions coexistent, de quelle manière leur synchronisation doit-elle être gérée ? Est-il possible qu'elles puissent être synchronisées en permanence ? Le cas échéant, à quelle fréquence devrait-elle l'être ? Comment garder une trace des incohérences entre les deux partitions à un instant T ?

Indépendamment du schéma de partitionnement et des restrictions d'accès, un objectif majeur pour l'ergonomie de notre système doit rester la transparence de ces mécanismes, particulièrement en ce qui concerne le requêtage des données. Ces derniers doivent être aussi invisibles que possibles vis-à-vis de l'écriture des requêtes mais également de leurs performances lorsque celles-ci sont exécutés dans le système.

1.4 Contributions

Pour répondre à la question de recherche 1, nous avons repris une technique nommée anatomie, conçue à l'origine pour le modèle de données relationnel et dont le principe consiste à briser les liens entre les entités et leurs attributs sensibles via la création de groupes d'attributs. Nous expliquons en détail les modifications qu'il a été nécessaire d'effectuer afin d'adapter cette approche au modèle RDF et comment nous l'avons finalement étendu en y incorporant des aspects de raisonnement. En nous appuyant sur une ontologie dans laquelle sont définis les attributs sensibles, nous sommes en mesure de les comparer entre eux,

de calculer leurs similitudes et ainsi de créer des groupes plus cohérents d'un point de vue sémantique d'où le nom que nous avons choisi pour cette proposition : anatomie sémantique. Le chapitre se conclue sur une série d'évaluations réalisées sur plusieurs graphes de tailles différentes et portant sur l'utilité des données à l'égard des requêtes de comptage.

Pour la question de recherche 2, nous avons d'abord commencé par présenter deux différents types d'attaque que nous souhaitions traiter. Cela étant fait, nous avons repris l'anatomie sémantique et avons introduit une adaptation de k-anonymity pour les graphes de connaissances. Deux algorithmes pour appliquer cette adaptation (l'un s'appliquant une seule fois sur la globalité des entités et l'autre plusieurs fois sur chaque groupe formé par anatomie) sont proposés et font l'objet d'une batterie d'expérimentations portant encore une fois sur l'utilité des données. Celles-ci nous ont entre autres permis de confirmer que la version dite "globale" était bien plus performante et permettait d'obtenir des résultats plus précis.

Enfin pour la dernière question de recherche, nous avons mis au point Kgastor, un framework pensé pour être placé au-dessus d'un RDF-store afin d'anonymiser des graphes de connaissances dans un contexte dynamique. Par l'utilisation de graphes nommés, notre système est capable de partitionner les données en deux ensembles distincts : l'un destiné à des utilisateurs privilégiés et contenant les données sensibles non-anonymisées, le second accessible à l'intégralité des utilisateurs et contenant les données anonymisées. Le lien entre les deux graphes est assuré par la présence de nœuds vides. Dans une volonté de rendre ce partitionnement transparent pour les utilisateurs, nous avons également développé une solution de réécriture de requêtes qui, en fonction du rôle d'un utilisateur, modifie sa requête afin que celle-ci s'exécute sur le graphe adéquat. Concernant l'anonymisation, nous avons choisi de reprendre en l'ajustant la technique m-invariance conçue pour traiter des données relationnelles dans des contextes dynamiques. Notre approche est par ailleurs accompagnée d'un journal capable de traquer les modifications effectuées sur un graphe depuis la dernière publication de façon à les anonymiser en conséquence.

1.5 Publications

Durant cette thèse de doctorat, réalisée dans le cadre d'un financement du type CIFRE avec Engie, les contributions mentionnées précédemment ont fait l'objet de plusieurs publications.

- **Knowledge graph publishing with anatomy, toward a new privacy and utility trade-off**. Maxime Thouvenot, Olivier Curé, Lynda Temal, Sarra Ben Abbès, Philippe Calvez. Advances in Knowledge Discovery and Management Vol. 10 (AKDM-10). Accepté, à paraître.
- **Knowledge Graph Anonymization using Semantic Anatomization**. Maxime Thouvenot, Olivier Curé, Philippe Calvez. ISWC (Demos/Industry) 2020 : 129-134
- **Knowledge Graph Anonymization using Semantic Anatomization**. Maxime

Thouvenot, Olivier Curé, Philippe Calvez. IEEE International Conference on BigData (IEEE Big Data 2020) : 4065-4074

— **Preventing Attribute and Entity Disclosures : Combining k-anonymity and Anatomy over RDF Graphs**. Maxime Thouvenot, Olivier Curé, Philippe Calvez. IEEE International Conference on BigData (IEEE Big Data 2021) : 5460-5469

— **Kgastor : a Privacy By Design Knowledge Graph anonymized Store**. Maxime Thouvenot, Olivier Curé, Philippe Calvez. IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom 2022). A paraître après la tenue de la conférence en novembre 2022

1.6 Organisation de la thèse

La suite de ce document est organisée en six chapitres.

Le chapitre 2 présente un certain nombre de connaissances de base nécessaires à la compréhension des concepts que nous explorerons dans cette thèse. Nous commençons d'abord par une introduction des notions et du vocabulaire basiques utilisés dans tout processus d'anonymisation. Nous effectuons par la suite une courte rétrospective de l'histoire du Web avant d'introduire le concept de Web sémantique. Une partie de l'écosystème relatif à ce domaine est passée en revue, à savoir : formalisme RDF, langages d'ontologie, langage de requête SPARQL, etc. Par ailleurs, nous présentons également une première contribution concernant le *Lehigh University Benchmark* que nous utilisons pour évaluer toutes nos contributions. Cette partie n'étant pas suffisamment conséquente, nous avons fait le choix de l'intégrer ici plutôt que d'en faire un chapitre à part entière.

Le chapitre 3 revient sur l'état de l'art en matière d'anonymisation pertinent par rapport à nos travaux. Nous évoquons un certain nombre de méthodes conçues pour le modèle relationnel mais également certaines des approches réalisées spécifiquement pour les graphes RDF.

Le chapitre 4 constitue notre première contribution et présente l'anatomie sémantique évoquée dans la section précédente (question de recherche 1). Nous y rappelons le fonctionnement de l'approche d'origine dont nous nous sommes inspirés ainsi que les modifications que nous y avons apportées pour le traitement d'aspect sémantique.

Le chapitre 5 concerne la question de recherche 2 et présente notre deuxième contribution dans laquelle nous utilisons de manière simultanée l'anatomie sémantique et notre propre implantation de k-anonymity pour les graphes RDF. Pour cette adaptation, deux algorithmes différents sont proposés puis évalués.

Le chapitre 6 se charge quant à lui de présenter notre système Kgastor (question de recherche 3). Nous commençons par un rappel des attaques rendues possibles par un contexte dynamique et passons ensuite en revue les différents composants de notre framework : partitionnement, contrôle d'accès et anonymisation.

Enfin, dans le dernier chapitre nous concluons ce travail de recherche et discutons des pistes de recherche qu'il serait intéressant d'explorer à l'avenir.

Chapitre 2

Connaissances de base

2.1 L'anonymisation au sens large

Avant d'entrer dans les détails des techniques d'anonymisation, nous devons d'abord définir certaines notions que nous reprendrons tout au long de cette thèse.

L'anonymisation de données consiste à modifier le contenu ou la structure de celles-ci afin de rendre difficile, voire dans le meilleur des cas, impossible l'association d'un individu à une information sensible. De manière générale, nous appellerons "entités d'intérêt" (EI) les cibles d'une technique d'anonymisation. Chaque EI est caractérisée par un ensemble de valeurs que l'on appelle des attributs et qui correspondent aux colonnes dans un modèle de données relationnel ou aux objets des triplets dans un graphe de connaissances.

En outre, ces attributs peuvent être divisées en quatre catégories différentes [12] :

— **les identifiants explicites** (IDE) permettent d'identifier explicitement une entité, par exemple, le numéro de sécurité sociale.

Explicit Identifier (EID)	Quasi-identifiers (QID)			Sensitive attribute (SA)
Social Security Number	Zipcode	Age	Sex	Religion
190070128414280	77420	29	M	Catholicism
294027511702822	69001	52	F	Islam
164046911809822	75002	34	M	Protestantism
182107709426680	13008	22	M	Islam
182107709426680	77340	44	F	Judaism

FIGURE 2.1 – Catégorisation des attributs dans l'anonymisation

— **les quasi-identifiants** [7] (*QID*) sont un ensemble d'attributs qui, lorsqu'ils sont utilisés conjointement, peuvent rendre possible la ré-identification d'une entité. Exemple : l'âge, le code postal, etc...
— **les attributs sensibles** (*AS*) correspondent à des informations sensibles à propos d'une entité que l'on voudrait exploiter lors de nos analyses. Exemple : maladie, salaire, opinion politique, etc...
— **les attributs non-sensibles** (*ANS*) ne correspondent à aucune des définitions précédentes et ne permettent en rien d'identifier une entité.

Parmi ces catégories, seules les trois premières correspondent à des attributs à risque qu'il est nécessaire de prendre en compte dans une démarche de protection des données. Les IDE, qui permettent une ré-identification sans ambiguïté, doivent d'office être supprimés du jeu de données tandis que les quasi-identifiants et attributs sensibles doivent être d'une manière ou d'une autre anonymisés. Néanmoins, la pertinence des analyses dépendant grandement des AS, ces derniers font par conséquent rarement l'objet de modification et doivent être manipulés avec prudence. Nous fournissons dans la figure 2.1 un exemple de cette catégorisation avec un court exemple où nous retrouvons un IDE, trois QIDs ainsi qu'un AS.

Plusieurs opérations sont possibles pour anonymiser un jeu de données, parmi lesquelles les plus courantes sont :

— **la suppression** : une opération fondée sur la suppression ou le remplacement d'une valeur dans le jeu de données.
— **la généralisation** : une opération transformant une valeur en une version plus générale. Exemple : l'âge d'un individu peut être remplacé par un intervalle de valeurs.
— **la perturbation** : une opération introduisant du bruit ou des données synthétiques dans le jeu de données tout en préservant certaines relations statistiques (comme la variance ou la moyenne). Exemple : modifier l'âge d'un individu de 22 à 23 ans.

2.2 Graphes de connaissances

2.2.1 Le Web sémantique

Présentation

Naissant à la fin des années 1980, le web s'est d'abord présenté comme un espace essentiellement consultatif dans lequel la principale activité consistait à "consommer" du contenu plutôt qu'à en produire. Il s'est par la suite développé en adoptant une dimension participative permettant aux utilisateurs d'interagir et de collaborer entre eux via la création de contenu (messages, etc.). Considérée par certains comme un "Web 2.0" [11], cette technologie a connu la consécration au milieu des années 2000 avec l'apparition des premiers réseaux sociaux.

Cependant, dès la création du Web 1.0, Tim Berners-Lee, considéré comme le concepteur du World Wide Web, avait déjà la vision d'un Web sémantique où les informations ne se limiteraient plus à être stockées mais pourraient également être interprétées par les machines. Pour se faire, il est nécessaire d'étendre le Web syntaxique (Web 1.0) constitué de balises HTML (Hypertext Markup Language) à un Web sémantique où des annotations fournissent des métadonnées "compréhensibles" par la machine. Ces dernières permettent ainsi à des agents (des programmes autonomes) d'interpréter le Web plus efficacement et d'effectuer des tâches pour le compte d'utilisateurs ou d'autres agents.

Actuellement, cette vision n'est toujours pas totalement réalisée mais plusieurs standards ont vu le jour sous l'impulsion du Wide Web Consortium [1] et ceux-ci sont utilisés par des acteurs majeurs de l'industrie.

Applications concrètes

L'une des premières grandes utilisations pratiques de cette technologie remonte au mois de Mai 2012 lorsque Google prit la décision d'intégrer un graphe de connaissances développé en partie à partir du graphe de connaissances Freebase à son moteur de recherche. Le but recherché était de fournir aux internautes un accès plus pertinent aux informations concernant des faits, des personnes ou tout autre type d'entités nommées dans ses pages de résultats. Cela s'est traduit par l'apparition d'un encadré sur la première page dans lequel un court résumé et un ensemble d'informations encyclopédiques concernant l'entité étaient regroupés.

Nous donnons un exemple de cet encadré après avoir recherché "Charles III" sur le moteur de recherche de Google dans la figure 2.2. Par l'utilisation du graphe de connaissances, le moteur est en mesure de récupérer les entités relatives à cette personne ainsi que les relations les unissant puis de les représenter à l'aide d'hyperliens dans l'encadré, permettant ainsi de naviguer d'une entité à l'autre très rapidement.

L'encadré remplit ainsi deux fonctions : d'une part il présente un condensé d'informations sur l'objet de la recherche et de l'autre, il offre la possibilité à

1. https://www.w3.org/

FIGURE 2.2 – Informations sur Charles III obtenues grâce au graphes de connaissances de Google

l'utilisateur d'approfondir lesdites recherches en parcourant les liens proposés.

La plupart des informations manipulées par le graphe sont issues de deux sources :

— *Wikipédia*, dont les résumés introductifs en début de page sont généralement repris.

— une base interne à Google construite originalement à l'aide de *Freebase*[2] qui était une base de connaissances libre éditée de manière collaborative et dont les données provenaient majoritairement de Wikipédia. Maintenu par Google jusqu'en 2015, Freebase a finalement été clôturé et son contenu a été transféré sur une nouvelle plateforme collaborative nommée Wikidata[3].

Dans un autre registre, les technologies du Web sémantique peuvent également servir à des fins de partage et d'intégration d'information. Au sein d'une

2. https://en.wikipedia.org/wiki/Freebase
3. https://www.wikidata.org/wiki/Wikidata:Main_Page

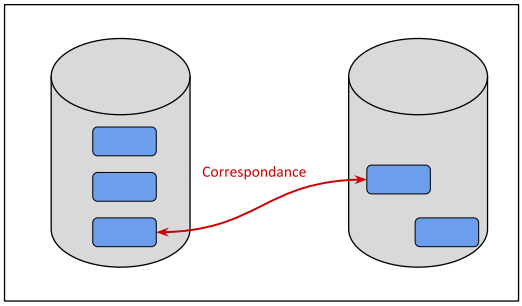


FIGURE 2.3 – Silos de données : trouver les correspondance pour améliorer l'intégration

même organisation, il est courant d'observer différents services développer leur propre système de données indépendamment de celui des autres. Ainsi on appelle un silo de données un ensemble de données maintenu sous le contrôle d'un service déterminé de l'entreprise, et qui se trouve isolé du reste de cette dernière, à la manière d'un silo agricole qui isole le grain des éléments extérieurs. On parle également de "cloisonnement des données".

A l'heure actuelle où la fouille de données représente un atout essentiel pour aider à la prise de décision et se démarquer de la concurrence, les silos se posent comme un problème majeur dans le sens où ils empêchent la bonne circulation de l'information en la contraignant à rester confinée au sein d'un service et, de fait, peuvent avoir une incidence sur l'intégrité des données.

Dans les entreprises, ces silos ont tendance à se développer naturellement, chaque unité organisationnelle possédant ses propres priorités et objectifs. En effet, lorsque dans une entreprise, plusieurs silos contiennent les mêmes données, leurs contenus sont susceptibles de suivre des voies divergentes. Le même constat peut être fait dans le cas d'une fusion entre deux organisations : des données redondantes pourraient exister dans leurs silos respectifs. En ayant recours à des annotations sémantiques, il est possible d'établir des correspondances entre les données similaires afin de faciliter leur intégration.

Afin de mettre en place cette extension du Web, aussi qualifié de "Web des données", le W3C a décidé d'adopter de nouveaux standards : tandis que le modèle de données RDF a été choisi comme formalisme pour représenter les graphes de connaissance, des langages d'ontologie ont été proposés afin d'attacher du sens à ces structures. Dans les sections suivantes, nous présentons chacune de ces technologies.

2.2.2 Modèle de données RDF

Le Resource Description Framework (RDF) est un modèle de graphe destiné à décrire de façon formelle les ressources du Web et leurs métadonnées, de façon à permettre le traitement automatique de telles descriptions. Développé par le

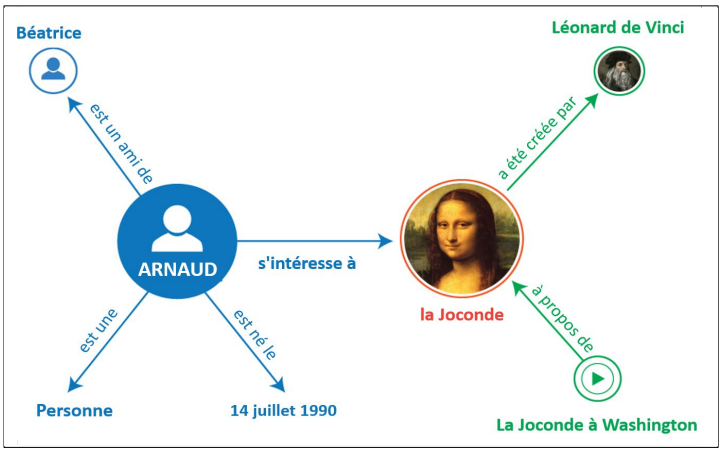


FIGURE 2.4 – Exemple schématique d'un graphe RDF basique

W3C, RDF est le modèle de base sur lequel s'appuie le Web sémantique pour représenter ses données.

Un document structuré en RDF est constitué d'un ensemble de triplets qui sont une association (sujet, prédicat, objet) :

- **le sujet** représente la ressource à décrire ;
- **le prédicat** représente un type de propriété applicable à cette ressource ;
- **l'objet** représente une donnée ou une autre ressource : c'est la valeur de la propriété.

Un document RDF ainsi formé correspond à un multigraphe orienté et étiqueté. Chaque triplet correspond alors à une arête orientée dont l'étiquette est le prédicat, le nœud source est le sujet et le nœud cible est l'objet.

On dénombre trois types de nœuds dans un graphe RDF :

- **les URI et IRI**[4], respectivement "Uniform Resource Identifier" et "Internationalized Resource Identifier", sont utilisées pour identifier des ressources sur le Web. Elles peuvent être retrouvées à n'importe quelle position dans un triplet, que ce soit le sujet, le prédicat ou l'objet. Dans l'exemple de la figure 2.4[5], les termes "Arnaud" et "La Joconde" seraient représentées par leur IRI respectif.
- **les littéraux typés** peuvent uniquement se trouver à la position de l'objet et sont utilisés pour affecter une valeur typée à une propriété : un entier, une chaîne de caractères, une date, etc. Dans l'exemple 2.4, la date de naissance serait exprimée avec un littéral.
- **les nœuds vides** (ou anonymes) désignent des ressources qui ne sont pas identifiées par une IRI. Ce type de nœud est principalement utilisé pour déclarer l'existence d'une ressource sans la nommer ou pour attacher

4. Extension des URI, supporte les caractères unicodes
5. https://www.emse.fr/~zimmermann/W3C/RDF1.1Primer/

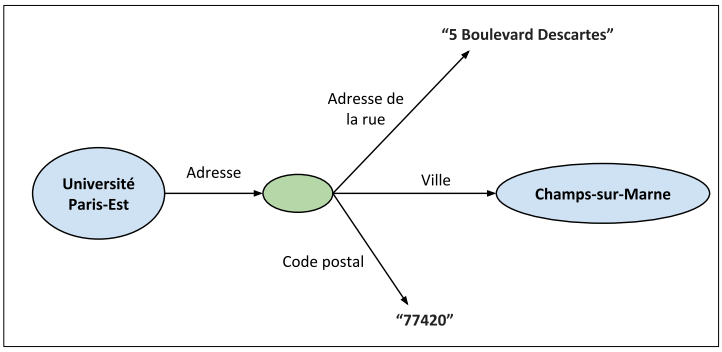


FIGURE 2.5 – Utilisation d'un noeud vide : l'exemple de l'adresse comme donnée complexe

de façon indirecte à une ressource un ensemble de connaissances. Les connaissances en question sont représentées comme des propriétés attachées au nœud anonyme. Dans la figure 2.5, nous montrons comment un nœud vide peut être utilisé afin de réifier les différentes informations d'une adresse.

La syntaxe (ou sérialisation) utilisée à l'origine pour écrire des documents RDF est le RDF/XML mais celle-ci s'avère être très verbeuse. Par la suite, d'autres syntaxes ont été proposées en vue de rendre la lecture plus compréhensible et l'écriture moins lourde : on pourra citer comme exemples Notation3 (N3) ou Turtle (TTL) qui offrent la possibilité, entre autres choses, de définir des préfixes afin de faciliter l'écriture des IRIs.

2.2.3 Langages d'ontologie et graphes de connaissances

Dans la section précédente, nous avons expliqué comment les données pouvaient être représentées à l'aide de triplets dans le modèle RDF. Cependant, à aucun moment nous ne nous sommes intéressés au sens que pouvaient porter les éléments composant ces triplets. Plusieurs vocabulaires ou langages ont ainsi été créés, en prenant comme base le modèle de données RDF, afin de définir le sens attaché aux ressources contenues dans un graphe. Dans cette partie, nous tournons notre attention vers les langages d'ontologie et présentons deux standards du W3C, à savoir RDF Schema (RDFS) et le Web Ontology Langage (OWL).

En informatique et en science de l'information, une ontologie est l'ensemble structuré des termes et concepts représentant le sens d'un champ d'informations ([19]). Les langages d'ontologie sont donc utilisés afin de définir les concepts caractérisant un domaine ainsi que les relations entre ces derniers.

RDFS est le langage d'ontologie le moins expressif du Web sémantique, il permet notamment de définir :

— des classes (l'équivalent des concepts).

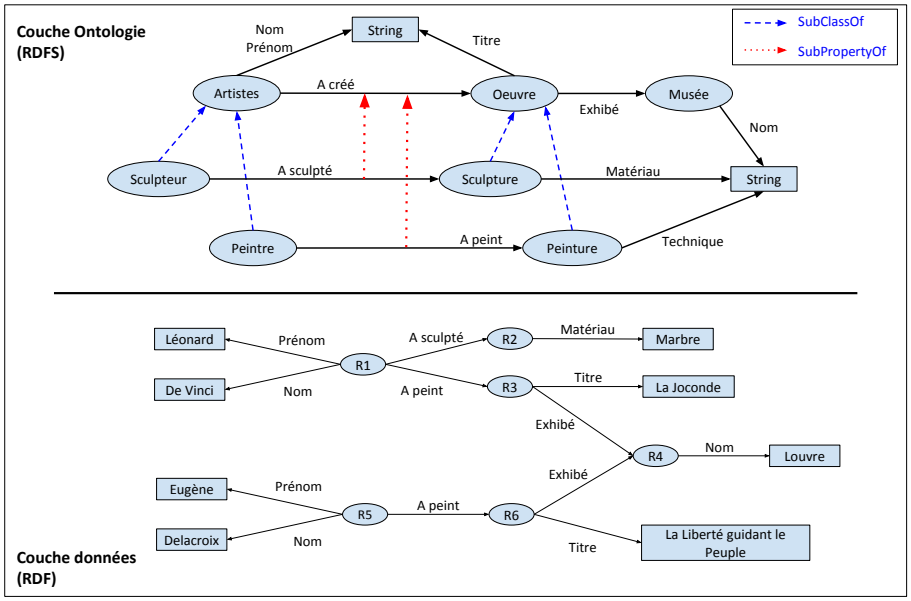


FIGURE 2.6 – Exemple de graphe de connaissances : artistes et œuvres d'art

— des sous-classes et, par conséquent, des hiérarchies de concepts.
— des propriétés (l'équivalent des relations).
— des sous-propriétés et, par conséquent, des hiérarchies de propriétés.
— les ensembles de départ (la classe des sujets) et d'arrivée (la classe des objets) des propriétés.

Un graphe de connaissances est constitué de deux parties : d'une part, nous avons l'ontologie qui est communément appelée la *Terminological Box* (ou TBox) et d'autre part, une base de faits contenant un ensemble d'assertions se reposant sur le vocabulaire décrit par l'ontologie, on parle aussi d'Assertional Box (ou ABox).

Dans la figure 2.6, nous donnons l'exemple d'un graphe de connaissances représentant des artistes et des œuvres. Nous mettons en évidence la distinction entre la TBox (partie supérieure) et la ABox (partie inférieure). Plusieurs hiérarchies peuvent y être observées :

— au niveau des concepts avec "Sculpteur" et "Peintre" qui sont tous deux des sous-concepts de "Artiste", tout comme "Peinture" et "Sculpture" sont eux-mêmes des sous-concepts de "Oeuvre"
— au niveau des relations avec "A créé" qui a pour sous-propriétés "A peint" et "A sculpté".

On notera aussi des restrictions sur les ensembles de départ et d'arrivée de certaines propriétés : par exemple, le prédicat "A créé" et ses sous-propriétés peuvent uniquement partir d'un artiste et pointer vers une œuvre.

La différence majeure entre un graphe de connaissances et tout autre type de base de données (relationnelle ou fondée sur les graphes), est la capacité de pouvoir raisonner sur les données. À partir de faits explicites (*i.e.*, des triplets existants dans le graphe), des raisonneurs sont en mesure de déduire des faits implicites (*i.e.*, qui n'existent pas), on parle dans ce cas **d'inférences**. Ainsi, dans la figure 2.6, nous pouvons affirmer que toute ressource R_i de type "Sculpteur" est également un artiste même dans l'éventualité où le triplet (R_i, type, Artiste) n'existerait pas.

Il existe d'autres langages d'ontologie plus puissants que RDFS permettant d'exprimer des relations plus complexes. Parmi ceux-là, OWL et ses fragments sont sans doute les plus connus et offrent un certain nombre de définitions supplémentaires pour les classes et les propriétés :

- des intersection de classes : une classe "Jument" pourrait être définie comme l'intersection des concepts "Cheval" et "Femelle".
- des propriétés inverses : "est l'enfant de" serait l'inverse de "est le parent de" par exemple. L'existence d'un triplet entre deux nœuds A et B impliquant l'une de ces propriétés impliqueraient donc l'existence d'un second triplet entre B et A impliquant l'autre propriété
- des propriétés transitives : si A est le frère de B et que B est le frère de C, on en déduit que A et C sont également frères.
- des chaînes de propriétés : une propriété peut être définie comme l'enchaînement de plusieurs autres propriétés. Par exemple, si A est le fils de B et B est le fils de C alors A est le grand-parent de C.
- etc.

Étant donné que nous ne faisons appel à aucune de ces constructions dans nos travaux, nous n'entrerons pas plus dans les détails. Il faut néanmoins noter que plus un langage d'ontologie est expressif, plus le coût de calcul nécessaire pour inférer des faits est élevé. C'est pourquoi les profils les plus expressifs de OWL sont indécidables et ne sont par conséquent jamais utilisés en situation pratique. Le travail d'une partie des chercheurs dans ce domaine gravite donc autour de l'étude des compromis pouvant être trouvés entre expressivité et complexité des raisonnements.

2.3 SPARQL, un langage de requête pour le modèle RDF

Après avoir présenté les langages permettant de représenter des données sur le web et de leur attacher une sémantique précise, nous introduisons maintenant le langage permettant de récupérer et de modifier ces données.

SPARQL [6] est un ensemble de spécifications concernant entre autres les langages de requête et les protocoles permettant de manipuler des données RDF. Dans la suite de cette partie, nous ne nous intéresserons qu'à l'aspect relatif au

6. SPARQL Protocol and RDF Query Language

requêtage et nous utiliserons le graphe présenté dans la figure 2.7 pour étayer nos explications quant au fonctionnement général du langage.

Adaptées pour les graphes RDF, SPARQL partage un certain nombre de similarités avec SQL, notamment au niveau de la syntaxe de ses requêtes où il reprend les célèbres clauses **SELECT WHERE**.

Les deux langages restent pourtant très différents, SPARQL adoptant une stratégie dite de "navigation" pour la récupération des données. L'idée est de partir d'un nœud de départ puis de sauter de nœud en nœud par le biais des arêtes jusqu'à atteindre un nœud d'arrivée. Cela diffère des méthodes d'exécution de SQL qui se basent plutôt sur la jointure de tables.

Pour cette raison, les contenus des clause **WHERE** sont également très différents : celles de SPARQL reposent essentiellement sur ce que l'on appelle des triple patterns, autrement dit, des triplets dans lesquels chaque élément (sujet, propriété ou objet) peut être remplacé par une variable. L'ensemble de ces triple patterns forment un *Basic Graph Pattern* (BGP) représentant les sous-graphes que nous recherchons dans la base de données afin de construire une réponse à la requête. Une correspondance (*matching*) entre le BGP et un sous-graphe est établie si les termes du sous-graphe peuvent être utilisés pour remplacer les variables dans le BGP.

Les requêtes de manipulation des données de SPARQL peuvent prendre plusieurs formes, *e.g.*, SELECT, ASK. Dans la suite de cette section, nous nous concentrons sur les requêtes du type SELECT ainsi que les requêtes de mise-à-jour.

2.3.1 Requêtes SELECT

Dans la figure 2.7, nous présentons un exemple de requête $SELECT$ pour récupérer les noms et prénoms des différentes personnes dans le graphe.

On remarquera en en-tête la présence de préfixes, ces derniers sont particulièrement utiles pour attribuer des alias à des espaces de nom et ainsi alléger l'écriture des requêtes. Les variables sont identifiées par le symbole " ?" suivi de leur nom, nous en comptons trois dans notre figure : *?person*, *?n* et *?fn*. Comme en SQL, la clause **SELECT** est utilisée pour spécifier les variables devant apparaître dans l'ensemble des résultats de la requête, on parle dans ce cas de "variables distinguées". Les variables apparaissant dans la clause **WHERE** mais pas dans la **SELECT** ont quant à elle pour fonction de connecter les différents triple patterns du BGP (à l'image d'une jointure en SQL). Dans notre exemple, cette jointure est réalisée par la co-occurrence de la variable *?person* dans les deux triples patterns en tant que sujet.

Une fois que tous les sous-graphes ont été trouvés, les valeurs des variables distinguées sont utilisées afin de construire l'ensemble de réponses. Le résultat de notre requête est fourni dans le Tableau 2.1.

Par ailleurs, SPARQL incorpore des arguments classiques de SQL tels que **LIMIT** qui permet de limiter le nombre de réponses de la requête, **DISTINCT** qui permet de supprimer les doublons dans le résultat, **GROUP BY** qui permet

?n	?fn
Léonard	De Vinci
Eugène	Delacroix

TABLE 2.1 – Ensemble de résultats pour la requête 2.7

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?n ?fn
WHERE {
    ?person  foaf:name  ?n .
    ?person  foaf:familyName  ?fn
}
```

FIGURE 2.7 – Obtenir le nom et le prénom de tous les artistes

d'appliquer des fonctions d'agrégation (*e.g.*, **COUNT()**, **MIN()**, **MAX()**, ...) pour ne citer que ceux-ci.

Le langage intègre également d'autres mots-clés pouvant apparaître dans la clause **WHERE** parmi lesquels **FILTER**, **OPTIONAL** ou encore **UNION**. **FILTER** permet de vérifier qu'une une variable vérifie une certaine condition (*e.g.*, des expressions régulières sont supportées et permettent la vérification de chaînes de caractères). Le mot-clé **OPTIONAL** permet quant à lui de récupérer des résultats même dans le cas où un triple pattern ne trouverait pas de correspondance avec le sous-graphe testé. Les triples patterns exprimées dans cette clause sont donc optionnels, il n'est pas nécessaire qu'ils aient des correspondances (le comportement est similaire au **OUTER JOIN** dans SQL). **UNION** est utilisé pour obtenir des résultats intermédiaires à partir de sous-requêtes, tous les résultats étant ensuite combinés afin de produire un résultat global.

Enfin, SPARQL propose aux utilisateurs un certain nombre de raccourcis syntaxiques permettant de décrire de manière concise des chemins plus en moins complexes entre deux nœuds : les property paths [7]. Nous fournissons une liste exhaustive de ces opérateurs dans le tableau 2.2 et présentons deux d'entre eux que nous utilisons dans nos approches : "*****" et "^".

Un astérisque (*) placé après un élément signifie que ce dernier peut apparaître 0 fois ou plus dans un chemin. Ainsi, le triple pattern suivant :

```
?x ex:childOf ?n.
```

peut correspondre, entre autres, aux triple patterns suivants (ou à aucun d'entre eux si le nombre d'occurrences de *childOf* est de 0) :

```
?x ex:childOf ?n.
```

7. https://www.w3.org/TR/sparql11-property-paths/

Forme syntaxique	Signification
^élément	chemin inversé (de l'objet vers le sujet)
(élément)	groupe de chemin (les parenthèses contrôlent la précédence)
élément1 / élément2	chemin commençant par élement1 suivi de élément
élément1 ^ élément2	raccourci pour (élément1 / ^élément2)
élément1 | élément2	chemins alternatifs (soit élément1 soit élément2)
élément*	chemin constitué de 0 ou plus occurrences de élément
élément+	chemin constitué de 1 ou plus occurrences de élément
élément?	chemin constitué de 0 ou 1 occurrence de élément
élément{n,m}	chemin constitué de n à m occurrence de élément
élément{n}	exactement n occurrences de élément (chemin de taille fixe)
élément{n,}	n occurrences ou plus de élément
élément{,n}	entre 0 et n occurrences de élément

TABLE 2.2 – Ensemble des opérateurs des property paths

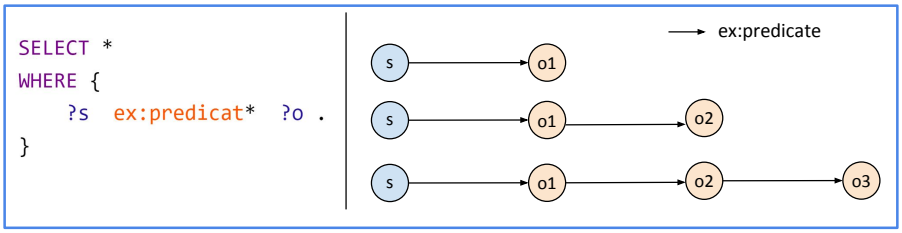


FIGURE 2.8 – Représentation de chemins grâce aux property paths.

```
?x ex:childOf ?temp1.
?temp1 ex:childOf ?n.

?x ex:childOf ?temp1.
?temp1 ex:childOf ?temp2.
?temp2 ex:childOf ?n.
```

Un schéma est également affiché dans la Figure 2.8.

Le second opérateur que nous souhaitons présenter, ^, permet de décrire un chemin inverse. Le triple pattern "*ex*:Maxime *^foaf*:*knows* ?*y*" revient à écrire "?*y* *foaf*:*knows* *ex*:*Maxime*" et donc à chercher toutes les entités connaissant Maxime. Ce property path est notamment utilisé pour éviter d'avoir recours à des variables temporaires dans les requêtes.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

DELETE { ?person foaf:name "Pablo"  }
INSERT { ?person foaf:name "Marcel" }
WHERE  { ?person foaf:name ?name    }
```

FIGURE 2.9 – Requêtes de mise à jour

```
PREFIX ex: <http://example_onto.owl#>
PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>
INSERT DATA {
    ex:Maxime ex:age "26"^^xsd:integer.
    ex:Maxime ex:zipcode "75000".
}
```

(a) Requête d'insertion

```
PREFIX ex: <http://example_onto.owl#>
PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>
DELETE DATA {
    ex:Maxime ex:knows ex:Olivier.
    ex:Martin ex:age 32.
}
```

(b) Requête de suppression

FIGURE 2.10 – Utilisation du mot-clé **DATA** dans les requêtes de mise à jour

2.3.2 Mise à jour de graphes RDF avec SPARQL Update

SPARQL Update est un langage de requête conçu spécifiquement pour les graphes RDF et pensé pour être utilisé conjointement avec la version classique du langage. Il reprend le système de triple patterns et de BGPs présenté précédemment et introduit deux nouvelles clauses : **INSERT** et **DELETE** Un exemple de requête de mise à jour est donné dans la Figure 2.9. Le fonctionnement de la clause **WHERE** est identique à ce qui été décrit plus haut, à savoir la définition d'un BGP correspondant aux sous-graphes qui nous intéressent dans les données.

Les clauses **INSERT** et **DELETE** partagent un comportement similaire : étant donné un sous-graphe récupéré grâce à **WHERE**, elles contiennent chacune un BGP devant être respectivement ajouté ou supprimé relativement à ce sous-graphe. Notons qu'il est nécessaire qu'au moins une variable présente dans **DELETE** et **INSERT** apparaisse également dans **WHERE** puisque c'est de cette façon que les insertions (respectivement suppressions) peuvent être intégrées avec les données présentes dans le graphe.

Car il est possible dans certaines situations de vouloir insérer ou supprimer des triplets sans s'appuyer sur des données déjà existantes, SPARQL Update fournit à cet effet le mot-clé **DATA**. Un exemple est fourni pour chacune des opérations dans la figure 2.10.

2.4 Un benchmark pour les données RDF

2.4.1 Lehigh University Benchmark

De manière à évaluer nos approches, nous avons fait le choix d'utiliser des jeux de données synthétiques générés à partir du Lehigh University Benchmark (LUBM) [20]. Ce benchmark se base sur une ontologie, *Univ-Bench*, construite avec le langage OWL et décrivant le domaine des universités, notamment les départements qui les composent ainsi que les activités s'y déroulant.

Un point majeur dans la conception de LUBM est sa capacité de passage à l'échelle (scalability) : en fournissant un simple paramètre, l'utilisateur peut indiquer au programme le nombre d'universités qu'il souhaite créer, il peut ainsi produire une multitude de jeux de données de tailles variées. LUBM utilise pour se faire un générateur de données dont l'unité minimale de génération est l'université et qui, pour chacune d'entre elles, génère une collection de fichiers OWL permettant de décrire ses départements. Dans un souci de rendre les données plus réalistes, un certain nombre de restrictions sont opérées au moment de l'écriture des fichiers, par exemple : une université doit comporter entre 15 et 25 départements, chaque étudiant diplômé doit suivre au moins un cours mais pas plus de trois, etc.

L'ontologie Univ-Bench définit un total de 43 classes et 32 propriétés (dont 25 object properties et 7 datatype properties) et emploie par ailleurs plusieurs fonctionnalités de raisonnement propres à OWL, à savoir des restrictions (par exemple pour la définition de certains concepts tels que *GraduateStudent*) ou encore des propriétés transitives.

Néanmoins, après avoir étudié quelque temps cette ontologie, nous avons réalisé que cette dernière ne se prêtait pas totalement au travail que nous voulions mené avec nos approches, l'un des principaux problèmes étant l'absence, de notre point de vue, d'attributs pouvant servir de données sensibles à anonymiser. Dans la partie suivante, nous élaborons plus en détail sur les difficultés que nous avons rencontrées.

2.4.2 Extension de LUBM

Cette section constitue une de nos premières contributions. Étant de nature plus mineure, nous avons décidé de l'inclure ici plutôt que d'en faire un chapitre à part entière.

Motivations

Premièrement, rappelons que nos approches, en particulier dans [44] et [45], ont besoin pour fonctionner de s'effectuer sur des attributs sensibles adossées à des taxonomies relativement complexes. Parmi les hiérarchies de concepts définies dans LUBM, deux d'entre elles, *Publication* et *Employee*, ont retenu notre attention avant d'être finalement écartées. La première permet de représenter les publications rédigées par des professeurs et s'étend sur trois niveaux. Malheureusement, les publications ne sont les objets d'aucune propriété dans les jeux de données générés. Par ailleurs, il s'avère que seul le concept de "Publication" est instancié, les autres concepts de la hiérarchie n'apparaissant jamais

dans la ABox. L'utilisation des publications requerrait donc un certain nombre de modifications au sein de LUBM.

La taxonomie des employés est de son côté plus intéressante : étalée sur quatre niveaux, quatre de ses concepts sont instanciés et sont utilisés en tant qu'objet par plusieurs propriétés. Le problème ici concerne le passage à l'échelle de nos approches. Comme nous l'avons expliqué plus tôt, les jeux de données sont produits de manière répétée en fonction du nombre d'universités désiré. Par conséquent, le nombre d'employés augmente de manière linéaire avec la taille des graphes, ce qui signifie également que le nombre de valeurs distinctes pour nos attributs sensibles évoluerait de la même manière. La complexité de nos algorithmes étant quadratique, nous avons observé que l'explosion des temps d'exécution rendait l'évaluation de nos travaux très difficile.

De plus, il faut noter qu'en situation réelle, les attributs sensibles appartiennent à des domaines/ensembles de définition finis ou du moins qui évoluent très peu dans le temps. À titre d'exemple, dans une base de données médicales, les attributs sensibles pourraient être la maladie des patients ou bien les molécules leur étant prescrites, quoi qu'il en soit, il s'agit de deux domaines relativement figés. Même dans les cas où un attribut serait "continu", tel que le salaire d'un individu représenté par un nombre réel, le nombre potentiellement infini de valeurs reste généralement contenu dans un intervalle connu et relativement fixe. Par conséquent, avoir recours à la hiérarchie d'un concept défini dans LUBM est non seulement contraignant du point de vue des performances de nos évaluations mais cela ne permet pas non plus de représenter la majorité des cas pratiques.

Du reste, créer nos propres attributs nous permet également de contrôler leur distribution au sein des jeux de données, là où les attributs dans LUBM sont répartis uniformément entre les différentes entités d'intérêt. Nous pouvons donc générer des graphes aux caractéristiques variées afin de tester le comportement de nos approches dans différents scénarios.

Enfin, et ce point concerne davantage nos articles de recherche [45] et Kgastor dans lequel nous considérons la généralisation de QIDs, nous étions à la recherche d'entités pourvu d'un certain nombre d'attributs, préféremment de type littéral. Or, comme nous l'avons déjà évoqué, la majorité des propriétés présentes dans LUBM (25 sur un total de 32) se trouvent être des object-properties. En plus de soulever une nouvelle fois la question de la scalabilité de nos approches, choisir des concepts propres au benchmark afin de s'en servir en tant que QIDs posaient également problème notamment pour définir comment ces valeurs devaient être généralisées. Les datatype-properties ne sont pas non plus montrées très satisfaisantes car leurs valeurs correspondaient principalement à des chaînes de caractères semi-aléatoires et qui n'étaient pas partagées par les instances de même type, cela les rendant par conséquent difficilement généralisables. En définitive, cette extension a d'abord pour but de répondre à une absence d'attributs conformes à nos besoins mais également à un manque de variété au niveau des attributs déjà présents.

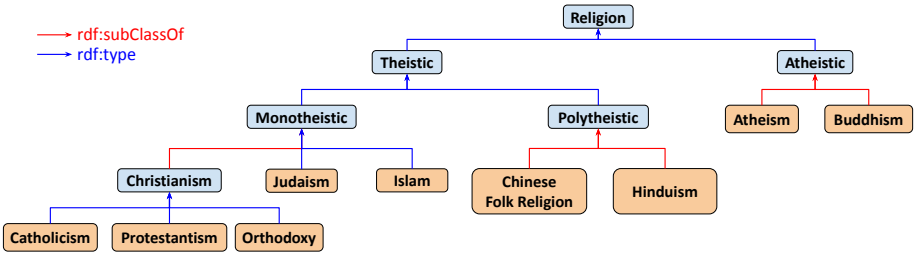


FIGURE 2.11 – Ontologie des croyances religieuses

Ajout d'attributs sensibles : croyances religieuses et opinions politiques

Pour servir d'entités d'intérêt pour tous les attributs de notre extension, nous avons fait le choix d'utiliser les instances du concept "Professor", ou plus précisément des instances de ses sous-concepts "FullProfessor", "AssociateProfessor" et "AssistantProfessor", le concept principal n'étant jamais directement instancié.

En ce qui concerne les attributs sensibles, une sous-ontologie traitant du domaine des croyances religieuses a été mise au point puis intégrée à l'ontologie principale de LUBM. Celle-ci est présentée dans la figure 2.11 et comporte 15 concepts organisés sur quatre niveaux. Les religions sont attribuées aux professeurs par le biais d'une object-property nommée *hasReligion*. Par ailleurs, dans l'optique de réaliser certaines évaluations dans un scénario impliquant deux attributs sensibles, nous avons ajouté une autre propriété, *politics*, permettant cette fois de lier un professeur à une opinion politique. Ces opinions n'appartiennent à aucune hiérarchie et sont organisées comme un simple ensemble de termes : *Socialism*, *Progressivism*, *Conservatism*, *Ecologism*, *Communism* et *Liberalism*.

Dans le fonctionnement initial de notre extension, les valeurs de ces propriétés étaient partagées de manière uniforme entre toutes les entités. Nous avons par la suite modifié ce comportement afin d'obtenir des répartitions biaisées et produire ainsi des graphes aux caractéristiques différentes pour nos évaluations. Les détails de ces probabilités sont fournis dans le tableaux 2.3. Les proportions pour les religions sont vaguement tirées de données réelles que nous avons pu récupérer tandis que celles pour les opinions politiques ont été établies arbitrairement.

Ajout d'attributs complémentaires : attributs génériques et quasi-identifiants

De façon à mettre sous pression nos approches lors de nos évaluations sur l'utilité des données, nous avons tendance à augmenter la sélectivité des requêtes utilisées. Pour se faire, nous avons commencé par générer des attributs aux valeurs génériques : étant donné un nom de propriété et un entier θ servant de limite maximale, nous attribuons à chaque noeud de type *Professor* une

Religion	**Proportion**
Islam	25%
Catholicism	16%
Protestantism	13%
Orthodoxy	3%
Atheism	15%
Hinduism	15%
Chinese Folk Religion	6%
Buddhism	8%

Politics	**Proportion**
Progressivism	28%
Liberalism	24%
Conservatism	19%
Socialism	15%
Communism	9%
Ecologism	5%

TABLE 2.3 – Proportions biaisées pour nos SAs

valeur contenue dans l'intervalle $[0, \theta[$ (la distribution des valeurs était uniforme). Nous pouvons donc facilement créer de nouveaux attributs tout en contrôlant la sélectivité de ces derniers.

Nous avons par la suite étendu le benchmark en ajoutant aux professeurs trois quasi-identifiants couramment rencontrés, à savoir :

— un sexe pouvant prendre deux valeurs, "*male*" ou "*female*"
— un âge dont la valeur est choisie aléatoirement entre 20 et 100 ans.
— un code postal produit de manière générique : étant donné une valeur de départ "s" et une plage "r", une valeur est choisie dans l'intervalle $[s, r))$

Pour chacun de ces trois attributs, les valeurs sont encore une fois réparties uniformément entre les entités.

Chapitre 3

État de l'art au niveau de l'anonymisation

L'abondance des données, couplée à une volonté grandissante de les diffuser, a permis de mettre en lumière le caractère potentiellement délétère d'une partie d'entre elles. Partant de cet état de fait, les premiers travaux sur l'anonymisation, placés dans le contexte des bases de données statistiques, ont été entrepris au cours des années 1990 dans la communauté des mathématiques statistiques. Une dizaine d'années plus tard, les informaticiens se sont emparés du problème, cette fois dans le contexte des bases de données relationnelles, et ont produit les principes de confidentialité les plus connus et les plus employés encore aujourd'hui.

C'est une partie de ces techniques que nous présentons en premier lieu dans ce chapitre, en particulier celles nous ayant le plus influencés durant nos recherches. Nous faisons ensuite de même pour les approches apparues ultérieurement en vue d'être appliquées sur des graphes. Nous commençons brièvement par des propositions pensées pour les graphes sociaux avant de nous intéresser plus spécifiquement au cas des graphes de connaissances.

	IDE	**QID**			**AS**
#	Numéro SS	Code postal	Age	Sexe	Maladie
1	190070128414280	77420	29	Homme	Maladie cardiaque
2	294027511702822	77341	25	Femme	Maladie cardiaque
3	292011301205580	77340	27	Femme	Infection virale
4	195104422020025	77321	23	Homme	Infection virale
5	160042531111418	69012	59	Homme	Diabète
6	167084217227780	69004	52	Homme	Maladie Cardiaque
7	163027740010225	69002	59	Homme	Infection virale
8	164046911809822	69004	55	Homme	Infection virale
9	288055502815322	77423	31	Femme	Diabète
10	286033717108418	77425	33	Femme	Diabète
11	182107709426680	77448	36	Homme	Diabète
12	184109505712822	77345	34	Homme	Diabète

TABLE 3.1 – Jeu de données original

3.1 L'anonymisation pour le modèle relationnel

Afin d'illustrer de manière concrète le fonctionnement de certaines des techniques présentées, nous utiliserons comme exemple la relation affichée dans la table 3.1. On y trouve cinq attributs différents que l'on peut distinguer comme suit :

— le numéro de sécurité sociale (Numéro SS) sert ici d'identifiant explicite (IDE)
— le code postal, l'âge et le sexe correspondent à un ensemble de quasi-identifiants (QIDs)
— enfin, l'état de santé d'un individu correspond à un attribut sensible (AS)

Dans la partie qui va suivre, nous traiterons surtout de méthodes dites de généralisation dont le principe consiste à protéger les données personnelles des entités en modifiant les valeurs de leurs QIDs sans toucher à leur AS. De par leur nature, les IDE sont systématiquement supprimés.

3.1.1 K-anonymity

K-anonymity par Sweeney *et al.* [42] fait partie des premières approches d'anonymisation qui furent proposées pour le modèle relationnel et certainement la plus populaire, si bien qu'il s'agit encore aujourd'hui d'une référence dans le domaine.

L'article de recherche présente notamment le concept de *linking attacks* et expose le rôle principal joué par les QIDs dans leur réalisation. Le principe est plutôt simple : un QID considéré seul ne représente pas de risque pour la

confidentialité d'un individu car sa valeur ne peut pas être associée à une unique entité. En revanche, l'association des valeurs de plusieurs attributs peut s'avérer unique dans un jeu de données et donc permettre d'identifier un individu en particulier.

L'argument de Sweeney est que plusieurs sources de données différentes, concernant globalement le même groupe d'individus, peuvent avoir en commun un certain nombre d'attributs et qu'il est ainsi possible d'établir des correspondances entre certaines entités des deux sources : des *liens* (*links)* peuvent donc être tissés entre les entités. Un problème survient lorsque l'un des deux jeux de données est anonymisé tandis que l'autre contient des identifiants explicites, permettant à un attaquant de lier la véritable identité d'un individu à son enregistrement anonymisé. Se référant à un travail portant sur des données de recensement de 1990 aux États-Unis [18], Sweeney avance qu'il était possible à cette époque d'identifier 87% de la population américaine en se basant sur trois attributs : le code postal, le sexe et l'année de naissance.

Afin de répondre à ce type d'attaque, Sweeney propose un principe de confidentialité pour le modèle relationnel : k-anonymity On considère qu'une table est "k-anonymisée" si un enregistrement donné (l'équivalent d'une ligne) de la table est impossible à distinguer d'au moins $k-1$ autres enregistrements par rapport aux valeurs de ses quasi-identifiants. Ces groupes d'enregistrements sont surnommés des classes d'équivalence.

Le but recherché est qu'un attaquant ait en théorie au mieux une chance sur k d'identifier une entité dans les données, quand bien même il connaîtrait certaines informations concernant sa cible.

Plusieurs opérations d'anonymisation (des suppressions et des généralisations) peuvent être exécutées sur les QIDs de façon à regrouper les enregistrements en des classes d'équivalence d'au moins k éléments partageant tous la même valeur pour chacun de leur quasi-identifiant.

La table 3.2 expose une version k-anonymisée du jeu d'origine avec $k = 4$. En effet, cette nouvelle version de la table compte trois groupes de quatre éléments indiscernables les uns des autres : {1,2,3,4}, {5,6,7,8} et {9,10,11,12}. Nous pouvons remarquer que la création de cette table a nécessité deux types d'opérations :

- les codes postaux et les âges ont été généralisés : les derniers caractères des codes postaux ont été remplacés par des "*" de manière à ne garder que le numéro de département tandis que les âges ont été remplacés par des intervalles.
- les informations sur le sexe ont été entièrement supprimées.

De manière générale, l'enjeu lorsque l'on cherche à produire ce genre de table consiste à former les classes d'équivalence demandant le moins de généralisation et permettant de ce fait de préserver au maximum l'utilité des données anonymisées. Cependant, la recherche de classes optimales étant un problème NP-difficile lorsque $k \geq 3$ ([34] et [1]), de multiples algorithmes ont été proposés afin de trouver des solutions approximative : [34], [1], [26], [27], [4], etc.

	QID			AS
#	Code postal	Age	Sexe	Maladie
1	77***	< 30	***	Maladie cardiaque
2	77***	< 30	***	Maladie cardiaque
3	77***	< 30	***	Infection virale
4	77***	< 30	***	Infection virale
5	690**	≥ 50	***	Diabète
6	690**	≥ 50	***	Maladie cardiaque
7	690**	≥ 50	***	Infection virale
8	690**	≥ 50	***	Infection virale
9	77***	3*	***	Diabète
10	77***	3*	***	Diabète
11	77***	3*	***	Diabète
12	77***	3*	***	Diabète

TABLE 3.2 – Version 4-anonymity du jeu de données

3.1.2 *L*-diversity

Dans l'article de recherche [29], les auteurs mettent en lumière certaines faiblesses de k-anonymity et identifient deux types d'attaque pouvant être utilisées à son encontre : les attaques par homogénéité et celles par connaissances externes.

Pour illustrer **l'attaque par homogénéité**, reprenons les deux tableaux précédents, 3.1 et 3.2, et supposons qu'un adversaire connaissant les QIDs de l'enregistrement #1 cherche à découvrir la valeur de son attribut sensible. Partant de ces informations et du tableau 3.2, il est possible de déterminer que l'enregistrement appartient à la première classe d'équivalence (#1 à #4).

Néanmoins, si la classe compte effectivement quatre enregistrements, elle ne compte que deux attributs sensibles différents : "Maladie cardiaque" et "Infection virale". Dès lors, quand bien même l'adversaire aurait dans l'absolu une chance sur quatre de trouver le bon enregistrement, ses chances sont en réalité d'une sur deux de trouver son AS. Pire encore, dans le cas d'une cible qui serait dans la dernière classe d'équivalence (#9 à #12), l'AS est directement révélé car toutes les entités y sont diabétiques.

Les **attaques par connaissances externes** reposent quant à elles sur des informations complémentaires permettant de briser le principe de confidentialité. Pour reprendre un exemple exposé dans l'article, imaginons un attaquant dont la cible serait une personne de moins de 30 ans et d'origine japonaise. Cet individu se trouverait donc parmi les enregistrement #1 à #4 et souffrirait soit d'une maladie cardiaque soit d'une infection virale. Or, en se basant sur le fait que statistiquement les japonais souffrent très rarement de problèmes cardiaques, il

	QID			**AS**
#	Code postal	Age	Sexe	Maladie
1	77***	≤ 40	***	Maladie cardiaque
4	77***	≤ 40	***	Infection virale
9	77***	≤ 40	***	Diabète
10	77***	≤ 40	***	Diabète
5	690**	≥ 50	***	Diabète
6	690**	≥ 50	***	Maladie cardiaque
7	690**	≥ 50	***	Infection virale
8	690**	≥ 50	***	Infection virale
2	77***	≤ 40	***	Maladie cardiaque
3	77***	≤ 40	***	Infection virale
11	77***	≤ 40	***	Diabète
12	77***	≤ 40	***	Diabète

TABLE 3.3 – Version 3-diversity du jeu de données

serait légitime d'assumer que la cible correspond à l'enregistrement #3 ou #4 et souffre donc d'une infection virale.

Dans les deux cas, k-anonymity pâtit du fait que les valeurs des attributs sensibles ne sont pas considérées au moment de la formation des classes d'équivalence. L-diversity se veut être une extension de k-anonymity dont le principe consiste à introduire de la diversité dans les classes en assurant une bonne distribution des attributs sensibles dans chacune d'entre elles. Ainsi, il est exigé dans ce contexte que chaque classe d'équivalence contienne au moins l attributs sensibles *bien représentés*. Concernant le concept de *bonne représentation*, les auteurs proposent plusieurs définitions fondées sur le nombre d'occurrences des ASs apparaissant dans une classe d'équivalence donnée : distinct l-diversity, entropy l-diversity, recursive (c, l)-diversity, etc. Nous donnons dans la table 3.3 une version "l-diversifiée" de notre jeu de données d'origine avec $l = 3$.

Les opérations utilisées sont encore une fois des suppressions et des généralisations, à ceci près que puisque les ASs sont maintenant considérés dans la création des groupes, plus de modifications sont généralement nécessaires par rapport à k-anonymity. L-diversity permet donc d'obtenir des tables plus sécurisées mais souvent aux dépens de l'utilité des données : dans la table 3.3, il est par exemple devenu impossible d'analyser aussi précisément les âges des individus.

3.1.3 T-closeness

À l'image de l-diversity qui étendait déjà le principe de k-anonymity, t-closeness [28] est elle-même une extension de l-diversity. Ce travail de recherche

introduit le concept de *skewness attack* pouvant être utilisé contre un jeu "l-divers" notamment lorsque certains attributs sensibles apparaissent plus rarement que d'autres.

T-closeness reprend à son compte l'idée d'incorporer de la diversité au niveau des ASs mais stipule que, si une classe d'équivalence contient un AS en particulier, alors la distance entre la distribution de cet attribut sensible dans cette classe et la distribution de l'attribut dans l'ensemble du jeu de données ne doit pas être supérieure à un seuil t. On dit qu'une table respecte t-closeness dès lors que toutes les classes d'équivalence respectent ce principe. En outre, les auteurs utilisent dans leur travail la mesure d'*Earth mover's distance* [40] pour cette tâche.

3.1.4 Anatomie

L'article de Xiao et Tao [48] soulève un problème commun aux trois approches précédentes, et plus généralement à toutes les approches fondées sur la généralisation, à savoir que ces dernières ont tendance à entraîner de grandes pertes d'information qui ont pour conséquence la dégradation de la corrélation entre les QIDs et les ASs. Le point défendu par les auteurs est qu'il devient alors difficile d'analyser précisément les données à l'aide de requêtes impliquant les deux types d'attributs et d'en tirer des résultats pertinents.

Pour reprendre un exemple de l'article avec nos tables 3.1 et 3.3, imaginons que nous ayons une requête de comptage de la forme :

```
SELECT COUNT(*) FROM Unknown-Data
WHERE
    Maladie="Maladie cardiaque" AND 20 ≤ Age ≤ 30 AND
    Code_postal IN [77300, 77500[
```

Sur le jeu de données original, cette requête renvoie un compte de 2 (#1 et #2) cependant son exécution sur la table 3.3 est moins triviale en raison des intervalles utilisés pour représenter les âges et les codes postaux. En premier lieu, il faut déterminer à quelles classes d'équivalence peuvent appartenir les enregistrements d'intérêts : dans notre cas, il s'agirait de la première et de la troisième.

Ensuite, étant donné que les intervalles pour les QIDs dans la requête ne correspondent pas à ceux dans les classes, il faut calculer la probabilité p que l'un de leurs enregistrements soit concerné. Sans information supplémentaire, nous sommes obligés d'assumer que les valeurs de chaque QID sont réparties de manière uniforme dans ces intervalles. Ainsi, pour les codes postaux, nous avons 1000 valeurs possibles (toutes celles entre 77000 et 77999) dont seulement 200 sont concernés par notre requête (de 77300 à 77499). Pour les âges, nous disposons de 40 valeurs (de 1 à 40 ans) dont 10 nous intéressent. La probabilité qu'un enregistrement dans l'une ou l'autre des classes d'équivalence possèdes des valeurs pour ses QIDs correspondants aux intervalles dans la requête est donc $p = \frac{200}{1000} \times \frac{10}{40} = \frac{1}{20}$

Puisque les classe d'équivalence #1 et #3 ne comptent qu'un seul enregistrement souffrant de maladie cardiaque, nous pouvons inférer un résultat approximatif pour la requête sur la table 3.3 $2 \times p = \frac{1}{10}$ qui est plutôt éloigné de la réalité. Cela s'explique par le fait que la généralisation nous empêche de comprendre la distribution réelle des valeurs dans chacune des classes d'équivalence et nous force, faute de mieux, à supposer une distribution uniforme.

Pour répondre à ce problème, Xiao et Tao proposent l'anatomie, une technique dont le principe consiste à séparer les QIDs et les attributs sensibles en deux tables distinctes, en conservant telles quelles les valeurs des QIDs. De plus, avant même la création des deux tables, les enregistrements sont partitionnés suivant une certaine stratégie en différents groupes.

La première table, nommée *QIT* (pour *quasi-identifier table*) regroupe l'ensemble des QIDs non modifiés et compte également une nouvelle colonne permettant d'indiquer l'appartenance d'un enregistrement à un groupe précis. La seconde, dénommée *ST* (pour *sensitive table)* contient quant à elle les statistiques des attributs sensibles pour chaque groupe. Dans la table 3.4, nous fournissons un exemple de deux tables anatomisées comptant au total trois groupes qui correspondent aux classes d'équivalence présentes dans la table 3.3.

L'idée est que l'anatomie protège la confidentialité des entités en supprimant les liens directs entre ces dernières et leur AS. Pour arriver à ses fins, un adversaire doit donc effectuer une supposition probabiliste en se basant sur les données dans la table *ST*. Imaginons que sa cible soit l'un des quatre enregistrements du groupe 1 dans 3.4, il peut seulement supposer que celle-ci a 25% de chance de souffrir d'une maladie cardiaque, 25% d'une maladie virale et 50% du diabète.

Concernant l'utilité des données, la table *QIT*, en gardant intacts les valeurs des QIDs, permet d'effectuer des analyses plus efficaces que les méthodes fondées sur la généralisation. Reprenons la requête précédente et appliquons là à nos tables anatomisées 3.4. En nous référant à la table *ST*, nous pouvons constater que deux individus souffrent d'une maladie cardiaque, l'un dans le groupe 1, l'autre dans le groupe 3. En consultant *QIT*, nous n'avons plus besoin de calculer la probabilité qu'un enregistrement soit concerné par la requête car les valeurs des QIDs sont clairement affichées. Pour chaque groupe, nous comptons quatre enregistrements potentiels ayant 25% de chance de souffrir d'une maladie cardiaque. On peut donc produire le résultat approximatif $2 \times \frac{1}{4} \times 4 = 2$ qui correspond à la réponse attendue.

3.1.5 ϵ-differential privacy

Dans un autre registre, l'article par Dwork [13] aborde la question de manière plus rigoureuse et formelle en introduisant le concept d'ϵ-differential privacy dans le cadre des bases de données statistiques.

Étant donné deux bases de données voisines B_1 et B_2 (*i.e.*, qui diffèrent d'un seul élément), un processus randomisé s'appliquant à une base de données est différentiellement confidentiel si son comportement est sensiblement le même pour les deux bases. Autrement dit, les probabilités d'obtenir l'un ou l'autre des

#	Code postal	Age	Sexe	Groupe-ID
1	77420	29	Homme	1
4	77421	23	Homme	1
9	77423	31	Femme	1
10	77425	33	Femme	1
5	69012	59	Homme	2
6	69004	52	Homme	2
7	69002	59	Homme	2
8	69004	55	Homme	2
2	77341	25	Femme	3
3	77340	27	Femme	3
11	77448	36	Homme	3
12	77345	34	Homme	3

Groupe-ID	Maladie	Cardinalité
1	Maladie cardiaque	1
1	Infection virale	1
1	Diabète	2
2	Maladie cardiaque	1
2	Infection virale	2
2	Diabète	1
3	Maladie cardiaque	1
3	Infection virale	1
3	Diabète	2

TABLE 3.4 – Anatomie sur notre jeu

résultats pour B_1 ou B_2 sont très similaires. Le niveau de similarité en question dépend de la valeur choisie pour ϵ. Du point de vue de la confidentialité, cela signifie intuitivement que la présence ou non d'un individu dans la base ne peut affecter significativement le résultat du processus.

Remarquons que cette définition ne s'applique pas aux données mais aux processus randomisés. En effet, la stratégie décrite dans [13] ne vise pas à modifier directement les données comme les approches précédentes mais plutôt à *perturber* les résultats renvoyés par des requêtes d'agrégation (comptage, moyenne, etc) exécutées sur elles à l'aide de mécanismes ajoutant du bruit. Pour ce faire, [13] s'appuie sur un mécanisme qui se repose sur l'addition d'un bruit provenant de la distribution de Laplace. Dans [14], les mêmes auteurs offrent une formalisation de la quantité de bruit nécessitant d'être ajoutée et proposent un

mécanisme généralisé à cet effet.

En outre, la particularité de differential privacy repose dans les fondements théoriques de sa définition qui permettent d'assurer un certain nombre de garanties par rapport aux autres approches :

- la modélisation d'attaques n'est plus nécessaire : le mécanisme protège tout type d'information, que cela concerne la présence ou non d'un individu, la valeur d'un attribut sensible, etc. De plus, le niveau de confidentialité des données est indépendant des connaissances externes à disposition d'un attaquant.
- le niveau de confidentialité peut être formellement quantifié à l'aide du paramètre ϵ.
- plusieurs mécanismes peuvent être composés et le niveau de confidentialité de cette composition est quantifiable. Ce point est notamment utile pour traiter le cas où deux attaquants ayant accès aux données chercheraient à travailler ensemble.

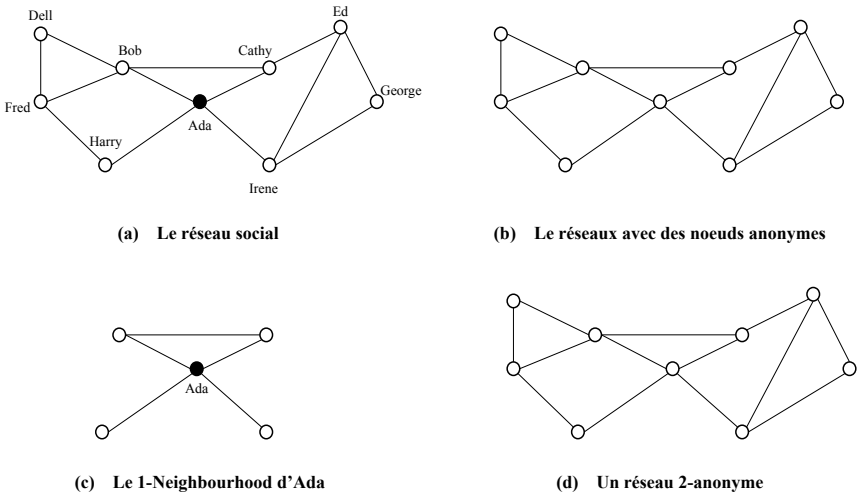
3.2 L'anonymisation pour les graphes

3.2.1 Dans le contexte des graphes sociaux

De nouveaux problèmes de confidentialité peuvent se poser lorsque l'on vient à manipuler des graphes plutôt que de simples tables. Comme l'expliquent Zhou *et al.* dans leur article [53], un adversaire qui posséderait des informations locales sur un nœud du graphe pourrait menacer l'anonymité de certains individus. Nous fournissons un exemple dans la figure 3.1(a) : chaque nœud représente ici une personne et une arête entre deux individus signifie qu'ils sont amis.

La première idée pour anonymiser un tel graphe serait bien entendu de supprimer les noms attribués à chaque nœud, *e.g.*, figure 3.1(b), cependant ce n'est pas suffisant. Imaginons qu'un attaquant sache qu'Ada possède deux amis qui se connaissent mutuellement et deux autres amis qui ne se connaissent pas, autrement dit, qu'il connaisse son voisinage direct dans le graphe, *i.e.*, 1-hop neighbourhood (figure 3.1(c)). Dans ce cas, Ada peut être identifiée puisqu'il s'agit de l'unique nœud avec une telle configuration. De la même manière, Bob peut également être identifié dans le graphe anonymisé si l'adversaire connaît son voisinage direct. Pire encore, à partir de ces informations, l'attaquant pourrait même savoir que Bob et Ada sont amis et ont un ami en commun. On constate donc un effet boule de neige : plus un attaquant possède des informations sur la topologie du graphe et plus il est capable de déduire de nouvelles choses.

Plusieurs approches ont été proposées dans ce contexte, nous avons déjà évoqué [53] mais nous pourrions également citer [21]. Les deux articles de recherche abordent la question de manière similaire et tentent d'anonymiser des graphes en modifiant leur structure via l'insertion ou la suppression d'arêtes (figure 3.1(d)). Dans le même esprit que [43], l'objectif est de faire en sorte qu'un nœud soit indiscernable d'un certain nombre d'autres nœuds par rapport à son degré, son



(a) Le réseau social **(b) Le réseaux avec des noeuds anonymes**

(c) Le 1-Neighbourhood d'Ada **(d) Un réseau 2-anonyme**

FIGURE 3.1 – Attaque par les voisinages dans un graphe.

voisinage, etc.

De son côté, l'anonymisation de graphes de connaissances est un problème actuel qui a déjà fait l'objet de quelques études au sein de la communauté du Web sémantique. Plusieurs techniques ont été proposées, certaines s'inspirent de celles déjà appliquées dans le modèle relationel, d'autres sont plus originales. Nous présentons en détail plusieurs d'entre elles dans les sections qui suivent.

3.2.2 k-RDF-Neighborhood Anonymity

K-RDF-Neighborhood Anonymity est une approche proposée par Heitmann *et al.* [22] qui adapte k-anonymity [42] au format des graphes de connaissances. Elle repose sur des travaux similaires qui ont été réalisés pour les graphes homogènes, notamment par Zhou *et al.* ([52],[53]) ou Hay *et al.* ([21]) mais aussi sur les graphes RDF par Radulovic *et al.* [39].

L'idée principale est la suivante: le voisinage direct (1-hop neighborhood) d'une entité à anonymiser doit être indiscernable du voisinage de $k-1$ autres ressources. Étant donné que l'article se concentre davantage sur les graphes sociaux RDF, l'anonymisation ne prend en compte que le voisinage direct des entités de type *foaf:Person* [1].

Nous donnons un exemple de ces nœuds dans la figure 3.2. l'arête verte représente l'âge de la personne (*foaf:age*) et celle orange sa position géographique (*foaf:based_near*). Les arêtes rouges correspondent aux projets sur lesquels la personne est en train de travailler (*foaf:current_project*) tandis que les arêtes

1. Friend Of A Friend: vocabulaire RDF utilisé pour décrire des personnes et les relations qu'elles entretiennent

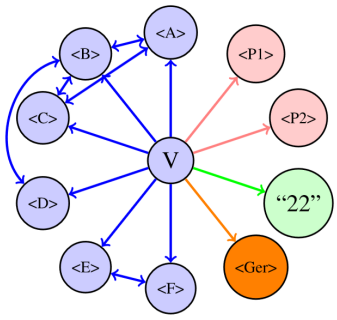


FIGURE 3.2 – Noeud de type *foaf:Person* et son voisinage (tiré de [22])

bleues représentent les personnes que l'entité connaît (*foaf:knows*).

L'algorithme présenté dans l'article peut se diviser en plusieurs étapes :

1. le calcul d'une chaîne de caractères minimale pour chaque entité d'intérêt représentant son 1-hop neighbourhood. L'algorithme est fréquemment amené à comparer des voisinages or, il n'existe pas de méthode efficace permettant de résoudre le problème de l'isomorphisme de graphes [52]. Ces chaînes sont donc utilisées afin de comparer rapidement les voisinages des entités.
2. la recherche de k chaînes similaires dans le but de minimiser la perte d'informations qui consiste elle-même en différentes tâches :
   (a) prendre le voisinage du noeud non-anonymisé ayant le degré le plus élevé.
   (b) calculer la similarité entre ce voisinage et tous les autres non-anonymisés.
   (c) sélectionner les $k-1$ voisinages les plus proches
3. mettre à jour le graphe en modifiant les k 1-hop neighborhood de manière à ce qu'ils deviennent indiscernables les uns des autres.
4. répéter les étapes 2 et 3 jusqu'à ce que toutes les entités soient anonymisées

Les opérations de modification correspondent à des suppressions d'arêtes (contrairement à [52] qui en rajoutaient de nouvelles), cela permet d'éviter d'introduire de fausses informations dans le graphes. Les auteurs de [22] font également remarquer que ce choix est conforme à l'hypothèse du monde ouvert [2] qui suppose qu'une information que l'on ne connaît pas n'est pas forcément fausse.

3.2.3 Anonymisation fondée sur des requêtes : politiques de confidentialité et d'utilité

Dans un autre registre, l'approche proposée par Delanaux *et al.* [8] s'émancipe des travaux réalisées précédemment et présente une stratégie fondée sur

2. Open World Assumption

```
# Privacy query P1
SELECT ?ad
WHERE {
  ?u  a   tcl:User.
  ?u  vcard:hasAddress  ?ad.
}

# Privacy query P2
SELECT ?u ?lat ?long
WHERE {
  ?c  a   tcl:Journey.
  ?c  tcl:user       ?u.
  ?c  geo:latitude   ?lat.
  ?c  geo:longitude  ?long.
}

# Utility query U1
SELECT ?u ?age
WHERE {
  ?u  a  tcl:User.
  ?u  foaf:age  ?age.
}

# Utility query U2
SELECT ?c ?lat ?long
WHERE {
  ?c  a   tcl:Journey.
  ?c  geo:latitude   ?lat.
  ?c  geo:longitude  ?long.
}
```

FIGURE 3.3 – Politiques de confidentialité et d'utilité pour des données sur le transport public.

l'utilisation de requêtes SPARQL.

Elle s'articule autour de deux notions importantes :

— **la politique de confidentialité** (Privacy policy) : il s'agit d'un ensemble de requêtes SPARQL pour lesquelles on ne souhaiterait obtenir aucun résultat après les avoir executées sur un graphe anonymisé. Elles représentent par conséquent les informations sensibles d'un graphe qui devraient être cachées. Cette idée est en partie repris de [6] qui présentait une étude théorique de l'anonymisation des jeux de données RDF dans le contexte des Linked Data et proposait déjà d'exprimer la confidentialité sous la forme de requêtes.

— **la politique d'utilité** (Utility policy) : à l'opposé, cette politique regroupe les requêtes dont le résultat ne devrait pas être modifié après que le graphe a été anonymisé. Elles permettent de protéger les informations importantes que l'on souhaiterait conserver pour des analyses futures.

Dans la figure 3.3, nous reprenons les exemples de politiques fournis dans l'article : on y voit une politique de confidentialité (les requêtes P1 et P2) ainsi qu'une politique d'utilité (U1 et U2) portant sur un jeu de données concernant les transports publics.

Concrètement, *P1* et *P2* peuvent se traduire par l'interdiction d'afficher l'identifiant d'un utilisateur avec son adresse ou ses informations de géolocalisation. *U1* et *U2* permettent quant à elles d'établir que l'âge d'un utilisateur et les informations de géolocalisation liées à un voyage doivent être préservés.

Leur algorithme prend en entrée ces deux politiques et produit un ensemble d'opérations d'anonymisation les satisfaisant. Il ne modifie donc pas physiquement le graphe mais recommande les manipulations à effectuer afin de l'anonymiser conformément aux attentes de l'utilisateur.

```
DELETE  { ?u  vcard:hasAddress   ?ad. }
WHERE   { ?u  a tcl:User.
          ?u  vcard:hasAddress   ?ad.}
```

(a) Exemple 1 : suppression des adresses des utilisateurs

```
DELETE  { ?c  tcl:user   ?u. }
INSERT  { ?c  tcl:user   []. }
WHERE   { ?c  a tcl:Journey.
          ?c  tcl:user       ?u.
          ?c  geo:latitude   ?lat.
          ?c  geo:longitude  ?long. }
```

(b) Exemple 2 : remplacement des identifiants des utilisateurs reliés à un ticket de transport par des noeuds vides

FIGURE 3.4 – Opérations d'anonymisation fondée sur les politiques

Il parcourt l'ensemble des triplets des requêtes contenues dans la politique de confidentialité et vérifie qu'il n'existe pas de conflits avec la politique d'utilité, autrement dit, que ce triplet n'apparaît dans aucune des requêtes d'utilité. Si tel est le cas, l'algorithme peut alors suggérer trois types d'opérations :

- dans tous les cas, il propose la suppression pure et simple du triplet en question.
- si le sujet apparaît comme le sujet ou l'objet d'un autre triplet de la privacy policy, on propose de le remplacer par un nœud vide et de briser le lien entre les deux triplets. La même opération est réalisée si le sujet apparaît parmi les variables projetées (dans la clause SELECT) de la requête.
- respectivement si l'objet du triplet est une IRI et apparaît comme le sujet ou l'objet d'un autre triplet ou qu'il fait partie des variables projetées, on propose de le remplacer par un nœud vide.

Nous offrons un exemple de ces opérations dans la figure 3.4, tiré une nouvelle fois de [8] sur les données de transport.

Finalement, cette approche est intéressante tout particulièrement de par son aspect déclaratif : en laissant la possibilité à un utilisateur de définir ses propres politiques de confidentialité et d'utilité, ce dernier peut alors exprimer ce qu'il est important pour lui de cacher et surtout de garder.

Le processus d'anonymisation n'est pas arbitraire mais modulable et peut être adapté aux besoins véritables de l'utilisateur. Cette notion était brièvement abordée dans k-RDF Neigbourhood Anonymity [22] où il était possible de spécifier un "pourcentage d'anonymisation" : l'algorithme se déroulait alors partiellement sur le graphe mais sans permettre de viser des portions particulières.

Un autre point fort de [8] réside dans la manière dont il produit les opérations d'anonymisation en analysant statiquement les requêtes au sein des politiques. De ce fait, les performances sont complètement indépendantes de la taille du

graphe à anonymiser et ne sont affectées que par le nombre de requêtes à analyser. Il s'agit d'une assurance non négligeable quand on sait que la taille d'un graphe de connaissances peut être de l'ordre de millions voire milliards de nœuds et tout autant d'arêtes.

L'article [9] est une suite des travaux présentés dans [8]. Dans celui-ci, seule une politique de confidentialité est considérée et un nouvel algorithme est présenté pour traiter les termes critiques devant être anonymisés. La procédure d'anonymisation consiste encore une fois à remplacer les ressources par des nœuds vides cependant les suppressions de triplets ne sont plus envisagées.

L'objectif de [9] est d'introduire une approche plus robuste au couplage de données dans le contexte des connaissances externes présentes dans les Linked Open Data. Par ailleurs, un autre ajout par rapport à la proposition précédente est la prise en charge de propriétés d'ontologie spécifiques telles que $owl{:}sameAs$, les propriétés fonctionnelles et fonctionnelles inverses.

De manière similaire, les algorithmes décrits dans cet article opèrent en analysant les triplets contenus dans la politique de confidentialité afin de produire les requêtes de mise à jour adéquates. Leurs performances ne sont sont donc pas influencées par la taille totale du graphe.

Chapitre 4

Anonymisation du graphe de connaissances par l'anatomie sémantique

4.1 Introduction

Comme nous l'avons évoqué lors du chapitre précédent, parmi la multitude d'approches proposées depuis une vingtaine d'années pour résoudre le problème de l'anonymisation, seule une petite minorité se place dans le contexte des données représentées à l'aide du modèle RDF. Cependant, force est de constater que

la plupart d'entre elles font généralement abstraction des aspects sémantiques pourtant propres aux graphes de connaissances.

Suivant une idée avancée dans [39], nous reprenons l'anatomie [48] pour l'adapter au modèle RDF en conservant le principe de l'approche : briser le lien direct entre les QIDs d'une entité et son AS afin de ne pas avoir besoin de généraliser les QIDs et ainsi dégrader la corrélation entre les attributs. Dans l'article originel, ce découplage entre les deux types d'attributs était rendu possible par la séparation du jeu de données en deux tables et l'utilisation de clés étrangères. Dans notre cas, cela est réalisé par l'ajout de nœuds vides intermédiaires.

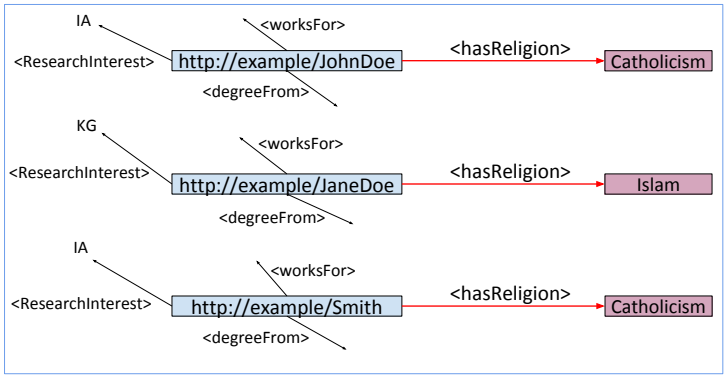
En outre, nous proposons une extension de l'anatomie prenant en compte certains aspects sémantiques des graphes RDF pour guider la création des groupes d'attributs sensibles. Plus concrètement, nous développons un algorithme se reposant sur les types de ces attributs ainsi que l'ontologie à laquelle ils appartiennent afin de créer des groupes où les éléments sont les plus semblables possibles. À cet effet, nous faisons usage d'une mesure de similarité nous permettant de comparer des concepts en fonction de leur position respective dans une même ontologie.

Une évaluation de notre prototype sur de grands graphes synthétiques générés à partir de notre extension du Lehigh University Benchmark (LUBM)[20] figure en fin de chapitre. Nous y présentons un ensemble de requêtes de comptage, impliquant un nombre variable d'attributs, que nous exécutons sur nos jeux de données afin d'en déterminer l'utilité après anonymisation. Par ailleurs, nous dressons également une comparaison entre notre proposition et une autre méthode conçue pour le modèle RDF [8] dans le but de montrer l'efficacité de notre approche à l'égard de l'utilité des données.

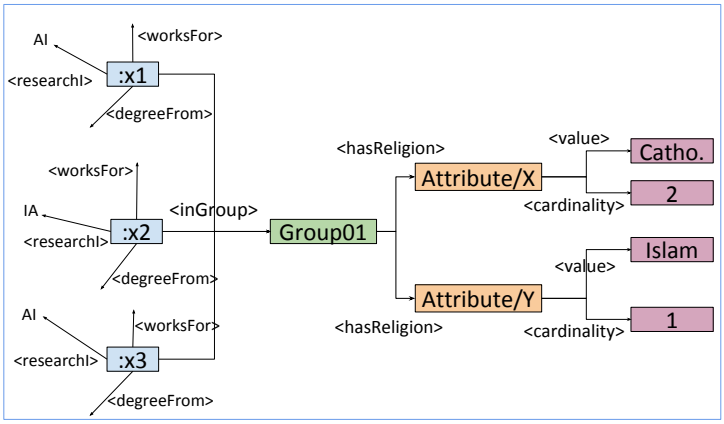
4.2 Adaptation de l'anatomie au modèle RDF

Dans un contexte de graphe de connaissances, l'anatomie implique que les relations entre les QIDs et les ASs soient supprimées par l'insertion de nœuds intermédiaires qui permettront de rassembler les différents ASs en groupes. Contrairement à d'autres méthodes d'anonymisation qui modifient ou suppriment des valeurs dans les données (par exemple celles fondées sur la généralisation), notre conception d'anatomie modifie le graphe au niveau structurel en ajoutant de nouveaux nœuds et arêtes. Le fait que les valeurs des QIDs ne soient pas transformées permet de préserver la corrélation entre les attributs et ainsi de favoriser une analyse de qualité des données une fois que celles-ci sont publiées.

Nous fournissons un exemple complet dans les figures 4.1(a, b) s'appuyant sur les concepts de religion de notre extension LUBM. La figure 4.1(a) présente un extrait de graphe représentant trois professeurs d'université et certains de leurs QIDs. Dans la figure 4.1(b), nous présentons une version « anatomisée » de cet extrait. On peut y voir que les valeurs des QIDs n'ont pas été altérées, que les EIDs (les URIs de John et Jane) ont été remplacés par des nœuds vides et que la propriété *hasReligion* ne pointe plus sur un concept de religion mais sur un nœud vide permettant de réifier plusieurs attributs sensibles. Des propriétés



(a) Extrait d'un graphe d'origine



(b) Application de l'approche

FIGURE 4.1 – Anatomie : suppression des liens directs entre les entités et leur attribut sensible

spécifiques à notre approche ont également été ajoutées par rapport au graphe d'origine, à savoir *inGroup*, *value* et *cardinality*. Parmi celles-ci, la cardinalité correspond au nombre d'occurrences d'un attribut sensible dans le graphe et permet de conserver certaines informations statistiques concernant les groupes, ce qui s'avère primordial à la préservation de l'utilité des données.

Selon la stratégie de partitionnement qui est adoptée, le contenu des groupes peut changer drastiquement. Dans l'article d'origine [48], les auteurs s'inspirent de l-diversity. Néanmoins, différents algorithmes pour l'anatomie peuvent être conçus selon la définition choisie pour ce principe de confidentialité. Dans le cas d'une l-diversity "distincte" par exemple, les groupes devraient seulement se contenter de contenir au moins l valeurs d'ASs différentes tandis que dans celui d'une l-diversity "partition", l'algorithme de clusterisation se devrait de regrouper les ASs en assurant une forme d'équilibre par rapport à leurs cardinalités.

4.3 Services de raisonnement : Realization et Least Common Ancestor

Dans cette partie et celle qui suivra, nous nous servirons de l'ontologie des croyances religieuses présentes dans la figure 2.11 en guise d'exemple afin d'illustrer les services de raisonnement et la mesure de similarité que nous utilisons pour étendre le principe de l'anatomie avec des aspects sémantiques.

Le service de raisonnement *realization* (REA) [2] d'un individu (ou d'un fait) correspond au concept le plus spécifique dont il est une instance. Par exemple, dans la figure 2.11 :

— $REA(Catholicism) = Christianism$
— $REA(Judaism) = Monotheistic$

D'autre part, le *least common ancestor* (LCA) de deux concepts A et B correspond au concept le plus spécifique qui est un ancêtre à la fois de A et de B . Nous étendons ce principe aux instances en considérant que le LCA d'une instance correspond au LCA de son REA. Encore une fois avec la figure 2.11 :

— $LCA(Islam, Judaism) = Monotheistic$
— $LCA(Judaism, Hinduism) = Theistic$
— $LCA(Catholicism, Hinduism) = Theistic$

4.4 Mesures de similarité dans un contexte sémantique

Notre adaptation nécessite de pouvoir évaluer la similarité entre deux entités dans un contexte sémantique. Plusieurs solutions ont été proposées pour supporter cette opération (*e.g.*, [47]). Dans ce travail, nous nous reposons sur Maedche et al. [31] qui introduit la notion de **taxonomy similarity** pour calculer la similarité entre deux instances en fonction de leur concept respectif et de leur position dans l'ontologie. Cette mesure repose sur plusieurs notions de base

que nous présentons maintenant. Tout d'abord, [31] définit une hiérarchie de concepts H comme une relation prenant comme arguments deux concepts C_1 et C_2 : $H(C_1, C_2)$ signifie que "C_1 *est un sous-concept de* C_2". Il présente ensuite la notion de **upward cotopy** (UC), soit C_i un concept quelconque appartenant à un ensemble de concepts C et H la hiérarchie des concepts associés :
$UC(C_i, H) = \{C_j \in C \mid H(C_i, C_j) \vee C_j = C_i\}$

Plus simplement, cela correspond à l'ensemble des super-concepts de C_i ainsi qu'à C_i lui-même. Par exemple, avec les concepts de 2.11 :

— $UC(Judaism, H) = \{Monotheistic, Theistic, Religion\}$
— $UC(Hinduism, H) = \{Polytheistic, Theistic, Religion\}$

Concept Match.

Étant donné deux concepts, C_1 et C_2, et la hiérarchie H, le **concept match** (CM) est défini comme suit : $CM(C_1, C_2, H) = \frac{|(UC(C_1,H) \cap UC(C_2,H))|}{|(UC(C_1,H) \cup UC(C_2,H))|}$

On a ainsi :

$$CM(Judaism, Hinduism, H) = \frac{|(UC(Judaism, H) \cap UC(Hinduism, H))|}{|(UC(Judaism, H) \cup UC(Hinduism, H))|}$$

$$= \frac{|\{Monotheistic, Theistic, Religion\} \cap \{Polytheistic, Theistic, Religion\})|}{|\{Monotheistic, Theistic, Religion\} \cup \{Polytheistic, Theistic, Religion\})|}$$

$$= \frac{|\{Theistic, Religion\}|}{|\{Monotheistic, Polytheistic, Theistic, Religion\}|} = \frac{2}{4} = 0.5 \quad (4.1)$$

Taxonomy Similarity.

Enfin, étant donné deux concepts C_1 et C_2, une hiérarchie H, la taxonomy similarity est définie comme : $TS(C_1, C_2) = \begin{cases} 1 & \text{if } C_1 = C_2 \\ \frac{CM(C_1,C_2)}{2} & \text{otherwise} \end{cases}$

Avec l'exemple précédent, cela signifie que $TS(Judaism, Hinduism) = 0.25$.

4.5 Anatomie sémantique

4.5.1 Introduction d'aspects sémantiques

L'objectif d'une approche anatomique fondée sur les aspects sémantiques est de s'assurer que les individus soient liés à des groupes d'attributs sensibles sémantiquement proches, le but étant de garantir une augmentation de l'utilité des jeux de données publiés par rapport à une approche d'anatomie standard. L'anatomie sémantique repose sur l'utilisation des ontologies associées au jeu de données à anonymiser. En particulier, nous utilisons intensivement la hiérarchie des concepts de la TBox. En tirant parti de ces structures et des services de

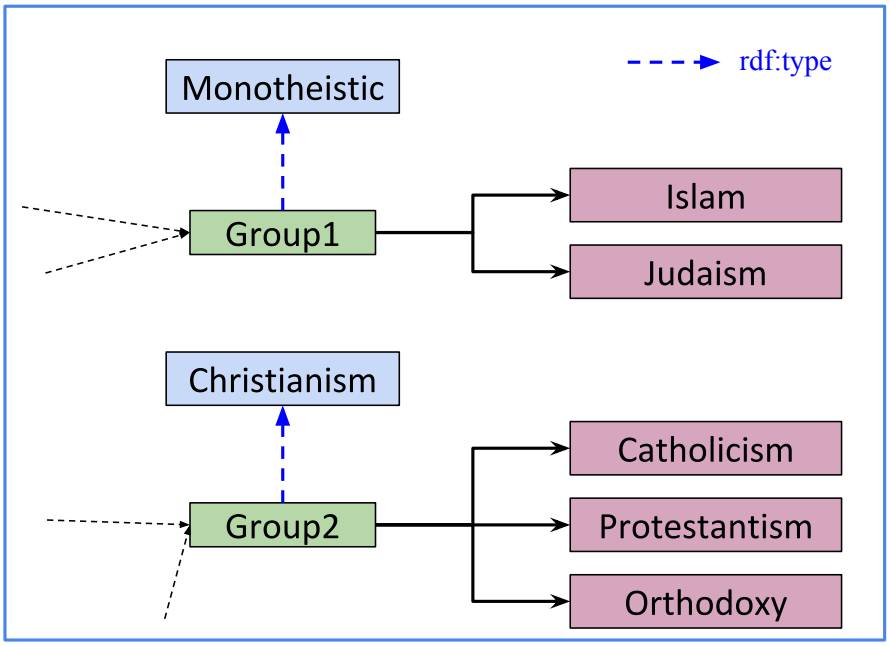


FIGURE 4.2 – Création de groupes spécifiques pour différentes religions

raisonnement qui leur sont associés, nous sommes en mesure de comparer les différents concepts de notre graphe de connaissances.

Nous fournissons dans la figure 4.2 un exemple d'anatomie fondé sur la hiérarchie de classes proposée dans la figure 2.11. Notez que les groupes possèdent désormais un type correspondant au LCA des concepts issus du raisonnement REA sur les religions.

L'algorithme d'anatomie sémantique (Algo 1) que nous présentons dans cette section a pour objectif de former des groupes d'attributs sensibles. Son résultat est ensuite utilisé pour créer des requêtes de mise à jour SPARQL (présentées dans la section suivante) afin de modifier le jeu de données original et de produire une version anonymisée.

4.5.2 Étape de prétraitement

Récupération de tous les concepts et calcul de l'upward cotopy. Avant d'anonymiser le graphe, nous récupérons tous les concepts présents dans l'ontologie et calculons leur UC respectif à l'aide des requêtes SPARQL présentées dans les figures 4.3 et 4.4.

Pour cela, nous sélectionnons simplement toutes les ressources dont le type est *rdfs*:*Class* ou *owl*:*Class*.

La requête pour récupérer les ancêtres d'un concept est quant à elle plus

```
1  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
2  PREFIX owl: <http://www.w3.org/2002/07/owl#>
3
4  SELECT DISTINCT ?concept
5  WHERE {
6      {?concept rdfs:type rdfs:Class.}
7      UNION
8      {?concept rdfs:type owl:Class .}
9  }
```

FIGURE 4.3 – Requête SPARQL utilisée pour les concepts de l'ontologie

```
1  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
2  PREFIX owl: <http://www.w3.org/2002/07/owl#>
3  PREFIX ex: <http://example.org/examples#>
4
5  SELECT ?ancestor
6  WHERE {
7      ex:concept  rdfs:subClassOf*  ?ancestor .
8      FILTER(?ancestor  !=  owl:Thing) .
9      FILTER(?ancestor  !=  rdfs:Resource)
10 }
```

FIGURE 4.4 – Requête SPARQL pour le calcul de l'upward cotopy.

complexe car elle utilise des property paths pour le prédicat "*subClassOf*" à la ligne 7. Grâce à l'opérateur "*", nous pouvons parcourir récursivement les super-concepts d'un concept initial jusqu'à atteindre le sommet de la hiérarchie.

Étant donné que *rdfs*:*Resource* et *owl*:*Thing* sont tous deux des concepts "racines", c'est-à-dire que tout individu dans le vocabulaire OWL (respectivement RDFS) est un descendant de *owl*:*Thing* (resp. *rdfs*:*Resource*), nous considérons qu'ils ne détiennent aucune valeur sémantique, ce qui explique pourquoi nous les filtrons dans la requête.

Matrice des similarités. Au cours du processus d'anatomisation, nous devons calculer plusieurs fois la similarité entre les mêmes concepts. Nous visons donc à optimiser cette partie en évitant les calculs redondants. Pour ce faire, nous calculons au préalable les mesures de similarité entre tous les concepts et réutilisons ces valeurs chaque fois que nous en avons besoin.

Les mesures de similarité sont stockées dans une structure que nous appelons la matrice des similarités et qui garantit un accès rapide à chaque valeur. Sa construction se décompose en deux étapes :

i. dans un premier temps, nous récupérons tous les concepts de l'ontologie du graphe et calculons leur UCs grâce aux requêtes des figures 4.3 et 4.4.

ii. ensuite, nous calculons la *taxonomy similarity* entre toutes les paires de concepts et stockons les valeurs dans la matrice de similarité.

Nous fournissons un exemple de cette structure pour l'ontologie des religions dans le tableau 4.1.

	Religion	**Theistic**	**Judaism**	**Hinduism**	**Atheism**
Religion	1				
Theistic	0.25	1			
Judaism	0.125	0.25	1		
Hinduism	0.125	0.25	0.166	1	
Atheism	0.125	0.125	0.083	0.083	1

TABLE 4.1 – Matrice symétrique des similarités entre les concepts de l'ontologie de la figure 2.11

4.5.3 Algorithme de clusterisation

Dans cette section, nous faisons référence à de nouveaux objets que nous nommons des *clusters*. Ces derniers correspondent à un ensemble d'attributs sensibles ainsi qu'au LCA de cet ensemble.

Nous avons décidé de nous inspirer de l'algorithme bottom-up pour le clustering hiérarchique proposé par [32]. Nous commençons d'abord par récupérer des clusters initiaux (*i.e.*, des clusters contenant un seul AS) que nous fusionnons par la suite en nous basant sur les mesures abordées précédemment. Cette étape se termine une fois que nous obtenons les clusters finaux qui seront utilisés pour produire les requêtes de mise à jour. Le pseudo-code est fourni dans l'algorithme 1.

Partons du principe que les premiers clusters aient été récupérés et soient stockés dans une liste dénotée L_1. Nous les regroupons ensuite itérativement à l'aide de la *taxonomy similarity* que nous appliquons sur les concepts des clusters. Nous répétons les étapes suivantes jusqu'à ce que L_1 soit vide :

i. nous recherchons le premier cluster dans L_1 et le supprimons de la liste (lignes 4 et 5).

ii. nous calculons sa similarité avec tous les autres clusters (lignes 7 à 12), y compris ceux déjà issus d'une fusion (lignes 13 à 18).

iii. à la ligne 19, nous fusionnons les deux clusters les plus proches, ce qui revient à :

   (a) faire l'union de leur ensemble d'attributs sensibles.

   (b) calculer le LCA de leur concept respectif afin d'obtenir le type du nouveau cluster.

   (c) le deuxième groupe est supprimé de la liste à laquelle il appartient, qu'il s'agisse de L_1 ou de L_2 (lignes 20 à 24).

Notons que l'algorithme prend principalement en entrée l'ontologie et non le jeu de données d'origine (sauf pour récupérer les clusters initiaux). Ceci est important lorsque l'on considère la complexité de traitement d'un algorithme puisque les TBox sont réputées pour être plus petites de plusieurs ordres de

Algorithm 1: Algorithme de clustering pour l'anatomie sémantique

Input : Knowledge graph G
Sensitive predicate p
Output: A list L_2 containing the clusters to be used to apply anatomization

```
1  L_1 ← retrieve the initial clusters from G using p;
2  L_2 ← empty list;
3  while L_1 not empty do
4  |  first ←retrieve the first cluster from L_1;
5  |  L_1 ← L_1/{first};
6  |  best_similarity ← 0;
7  |  foreach c in L_1 do
8  |  |  similarity ← compute taxo. similarity between first and c;
9  |  |  if similarity > best_similarity then
10 |  |  |  best_similarity ← similarity;
11 |  |  |  closest ← c;
12 |  |  |  in_L_1 ← True;
13 |  |  foreach c in L_2 do
14 |  |  |  similarity ← compute taxo. similarity between first and c;
15 |  |  |  if similarity > best_similarity then
16 |  |  |  |  best_similarity ← similarity;
17 |  |  |  |  closest ← c;
18 |  |  |  |  in_L_1 ← False;
19 |  |  new_cluster ← merge(first, closest);
20 |  |  if in_L_1 then
21 |  |  |  L_1 ← L_1/{closest};
22 |  |  else
23 |  |  |  L_2 ← L_2/{closest};
24 |  |  L_2 ← L_2 ∪ {new_cluster};
25 Return L_2
```

grandeur que les ABox. Plus précisément, la complexité d'Algo. 1 est en $O(n^2)$, n étant le nombre de clusters initiaux calculés à partir de l'ontologie.

4.5.4 l-diversité sémantique

Dans le cadre de notre approche, nous avons conceptualisé une définition alternative du principe de confidentialité l-diversity. Intuitivement, celle-ci est fondée sur le nombre de sous-concepts directs qu'un concept donné possède dans une TBox. Par exemple, dans la hiérarchie des concepts de la figure 2.11, *Christianism* et *Polytheistic* ont respectivement trois (*Catholicism*, *Protestantism* et *Orthodoxy*) et deux ($Hinduism$ et *ChineseFolk*) sous-concepts directs. Ainsi, pour chaque groupe créé à l'aide de notre algorithme de clustering, il est possible de calculer une valeur locale de l équivalant au nombre de sous-concepts

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX ex: <http://example.org/examples#>

DELETE {?s  ex:predicate   ex:attributeValue.}
INSERT {
    ?s   ex:inGroup  ex:groupX  .
    ex:groupX  rdf:type  ex:concept .
    ex:groupX  ex:predicate   ex:attributeX .
    ex:attributeX  ex:value  ex:attributeValue .
    ex:attributeX  ex:cardinality  ex:attributeCardinality .
}
WHERE {?s  ex:predicate  ex:attributeValue . }
```

FIGURE 4.5 – Application de l'anatomie via une requête SPARQL

directs instanciés dans l'ensemble de données d'origine. La l-diversity sémantique globale du jeu de données anatomisé correspond à la l-diversity locale minimale de tous les groupes. Algo. 1 assure une diversité minimale avec l=2. Néanmoins, selon la structure de la TBox, il serait tout à fait possible d'introduire un seuil minimum pour l dans notre algorithme et ainsi de forcer la création de groupes comptant un plus grand nombre d'attributs.

4.5.5 Génération des requêtes de mise à jour

Une fois les clusters finaux calculés, nous pouvons produire les opérations de mise à jour à effectuer pour appliquer l'anatomie au graphe : nous itérons sur chacun des clusters et sur chacun des attributs qu'ils contiennent afin de créer les requêtes SPARQL appropriées (un modèle générique de requête est exposé dans la figure 4.5).

La clause DELETE supprime le lien direct entre un individu et son AS tandis que la clause INSERT relie un individu à un groupe d'attributs. L'exécution de l'ensemble des requêtes amène finalement au graphe présenté dans la figure 4.6. Les valeurs de *GroupX* et *AttributeX* correspondent à de simples index que nous incrémentons au fur et à mesure du traitement itératif des clusters et des attributs.

4.6 Évaluation

L'objectif principal de cette évaluation est de vérifier si la corrélation des données entre QIDs et AS est bien préservée dans le cadre de requêtes d'agrégation,

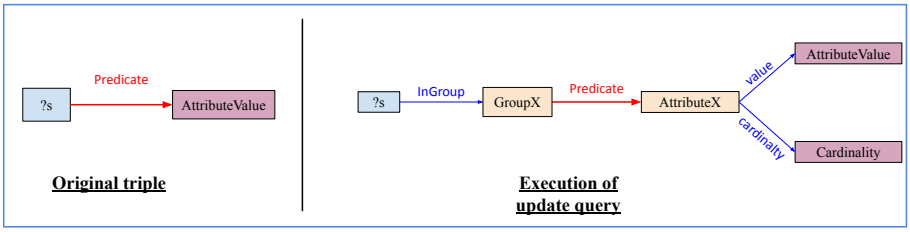


FIGURE 4.6 – Impact de notre requête de mise à jour SPARQL sur un seul triplet

e.g., exécution de requêtes de comptage impliquant les deux types d'attributs. Notre objectif est de produire une approximation du résultat de cette requête à partir du jeu de données anonymisé et que celle-ci soit assez proche du résultat réel dans le jeu d'origine.

4.6.1 Paramètres d'expérimentation

Nous avons réalisé l'évaluation sur un Dell Precision 5820 équipé d'un processeur Intel Xeon W 2255 à 10 cœurs, 96Go 2933 MHz DDR4, SSD 1 To M.2 PCIe NVMe. Le système exécute une distribution Debian 10 Linux. Notre solution d'anonymisation a été implémentée en Java, en utilisant le JDK 8 et le framework Web sémantique Apache Jena [1](version 3.15). Afin de gérer les données RDF et d'exécuter les différentes requêtes SPARQL, nous utilisons le triple store de Jena TDB2.

4.6.2 Jeux de données et requêtes

Nous avons évalué notre système sur 4 jeux de données générés à partir de l'extension LUBM présentée dans la section 2.4. Ils correspondent aux jeux de données LUBM de 100, 1000, 5000 et 10000 universités.

Les entitées d'intérêt ici sont les nœuds correspondant aux professeurs, leur nombre variant d'environ 60 000 à plus de 6 millions selon la taille du graphe. Pour chaque professeur, nous avons décidé de nous concentrer sur quatre attributs distincts. Nous considérons pour ces entités deux attributs sensibles correspondant à notre extension LUBM 2.4.2. Les deux autres attributs sont *Research Interest* (un attribut présent dans LUBM) et un prédicat génériques personnalisé possèdant jusqu'à 30 valeurs distinctes. Afin d'étudier l'efficacité de notre approche dans des scénarios multi-AS (une situation qui n'est pas couverte par [48]), nous divisons nos tests en deux catégories :

i. la religion est le seul AS (et l'opinion politique est un QID régulier)

ii. la religion et l'opinion politique sont toutes les deux considérées comme des ASs.

1. https://jena.apache.org/

Parameter	Values
Taille des jeux de données en millions de triplets	13.5, 135.3, 676.3, 1352
Taille des jeux de données en GB	2.3, 23.8, 120.2, 240.7
Nombre d'entités d'intérêts en milliers	60.2, 600.3 2998.8 5999.5
Nombre de QIDs	1, 2, 3
Nombre d'ASs	1, 2
Cardinalité des requêtes	2, 3, 4
Distribution des valeurs pour SA_1 et SA_2	Skewed/Unskewed , Skewed/Skewed Unskewed/Unskewed, Unskewed/Skewed

TABLE 4.2 – Résumé des paramètres de l'expérimentation

Nous avons d'abord commencé par tester notre approche sur un jeu de données où les valeurs des attributs *hasReligion* et *politics* étaient réparties de manière égale entre l'ensemble des professeurs. Après avoir observé des résultats assez positifs de cette évaluation préliminaire (*i.e.*, les résultats approximatifs pour nos requêtes étaient relativement proches des résultats réels), nous avons voulu vérifier si nous pouvions obtenir des résultats similaires avec différents types de distribution pour nos valeurs d'ASs.

À cet effet, nous avons modifié les jeux de données originaux et créé trois configurations supplémentaires : une distribution uniforme pour les deux attributs, une distribution uniforme pour les religions accompagnée d'une distribution biaisée pour les opinions politiques et vice-versa. Pour évaluer ces quatre configurations, nous considèrons les requêtes de la forme :
SELECT (COUNT(*) AS ?count) WHERE {
$?s\ \ Attr_1\ \ X_1.$ ... $?s\ \ Attr_i\ \ X_i.$
$?s\ \ Attr_{s1}\ \ Y_1.$ ... $?s\ \ Attr_{sj}\ \ Y_j.$ } (1)

Ainsi, une requête implique un total de i QIDs et j ASs (j étant limité à 2 dans notre évaluation). Le tableau 4.2 résume les paramètres impliqués dans notre expérimentation ainsi que les différentes valeurs utilisées pour chacun d'eux.

De manière à d'établir des comparaisons entre un jeu de données d'origine et son homologue anonymisé, nous utilisons la mesure d'*erreur relative* calculée à partir de deux valeurs : **une valeur attendue**, c'est-à-dire le résultat réel obtenu depuis le jeu de données d'origine grâce à l'exécution d'une requête de comptage, et **une valeur estimée** dont le calcul se fait à partir du jeu anonymisé. Soit Exp la valeur attendue obtenue après l'exécution d'une requête COUNT sur le graphe d'origine et soit Est la valeur estimée calculée depuis le graphe anonymisé. L'*erreur relative* est alors définie comme $\frac{|Exp-Est|}{Exp}$.

En raison de l'insertion de nouveaux nœuds (groupes et attributs) dans le jeu de données publié, les liens directs entre les entités et leurs ASs sont supprimés.

Par conséquent, les requêtes suivant le modèle (1) ne sont plus utilisables en tant que telles. Néanmoins, il est possible, par le biais d'un certain nombre de requêtes SELECT intermédiaires, de récupérer des informations intéressantes, notamment concernant les cardinalités des différents ASs. Ces informations peuvent ensuite être utilisés afin de produire des résultats approximatifs.

Prenons par exemple le graphe de la figure 4.1(b) et la requête suivante :
SELECT (COUNT(*) as ?count) WHERE {
?s researchIntérêt "AI". ?s aReligion catholicisme}

En premier lieu, nous récupérons le groupe contenant l'attribut "*catholicisme*" ainsi que la cardinalité de cet AS : $SA\_group = Group1$ et $SA\_card = 2$. Nous récupérons ensuite la cardinalité totale du groupe (c'est-à-dire la somme des cardinalités de tous les attributs) : $total\_Card = 3$. Nous pouvons alors calculer la probabilité, Pr, qu'un individu du groupe suive une certaine religion : $proba\_SA = \frac{SA\_card}{total\_card} = \frac{2}{3}$. Enfin, nous comptons le nombre d'individus appartenant au groupe SA_group et dont l'intérêt de recherche est l'IA. Ceci est possible dans un jeu anatomisé car les QIDs sont publiés directement, sans modification : $nb\_ind = 2$. Notre estimation est finalement calculée comme $Est = proba\_SA * nb\_ind = \frac{4}{3} \approx 1,333...$ dans cet exemple.

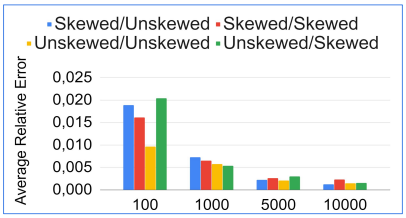
Le même raisonnement peut être appliqué à des requêtes impliquant un plus grand nombre d'attributs, le calcul de nb_ind devant simplement être modifié afin de les prendre en compte. En outre, dans les scénarios multi-AS, nous devons calculer une probabilité pour chaque attribut sensible et adapter la façon dont nous calculons nb_ind, en considérant les différents groupes : $proba\_SA_1 * proba\_SA_2 * \ldots * proba\_SA_n * nb\_ind$.

4.6.3 Résultats et analyses

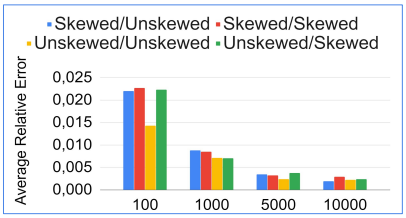
Étant donné un jeu de données et un ensemble de prédicats sensibles, nous sommes en mesure de produire un jeu anonymisé. Nous avons par ailleurs conçu un ensemble de requêtes impliquant de deux à quatre des attributs évoquées dans la section précédente.

Pour une cardinalité de requête donnée (*i.e.*, le nombre d'attributs concernés), nous calculons toutes les combinaisons possibles de valeurs pour ces attributs afin de générer les requêtes correspondantes. Ensuite, pour chaque requête, nous récupérons la valeur réelle du jeu de données d'origine, nous calculons la valeur estimée à partir du graphe anonymisé et nous comparons les deux résultats en utilisant l'erreur relative.

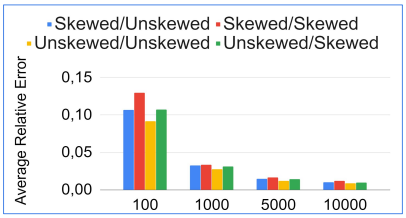
La figure 4.7 présente les erreurs relatives moyennes pour toutes les requêtes d'une cardinalité donnée : le premier niveau (a et b) concerne les requêtes de cardinalité 2 avec les attributs *hasReligion* et *politics*, le second (c et d) celles de cardinalité 3 avec *researchInterest* en attribut supplémentaire et enfin le dernier niveau (e et f) concerne les requêtes de cardinalité 4 avec tous les attributs. Pour chaque niveau, nous dessinons deux graphes : un où la religion est le seul attribut anonymisé et le second où l'opinion politique est également anonymisée (*i.e.*, scenario multi-AS).



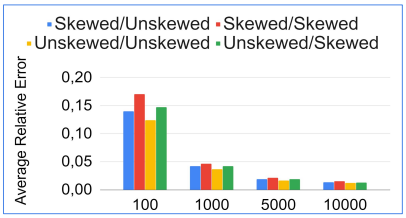
(a) Deux attributs - un AS



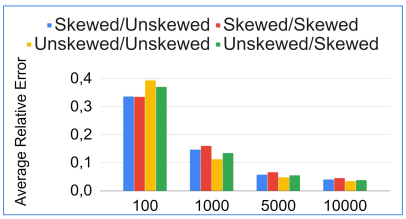
(b) Deux attributs - deux ASs



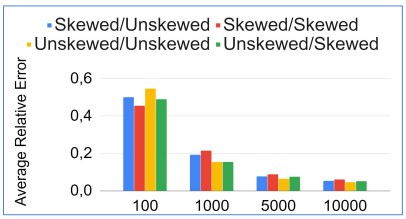
(c) Trois attributs - un AS



(d) Trois attributs - deux ASs



(e) Quatre attributs - un AS



(f) Quatre attributs - deux ASs

FIGURE 4.7 – Évaluation de l'erreur relative moyenne

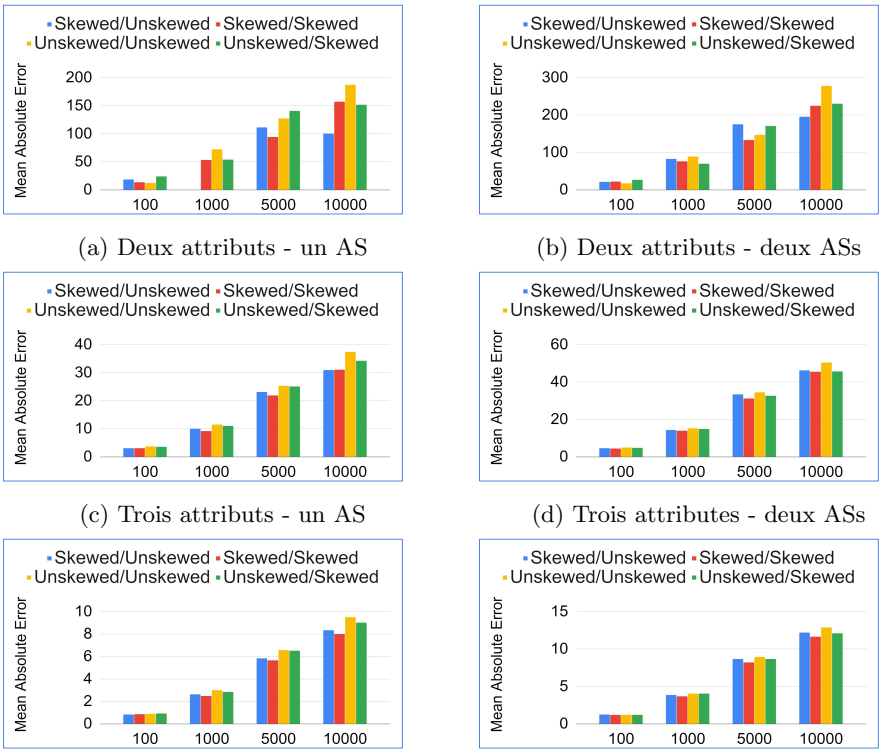
Un fait commun à toutes les évaluations est observé : plus le jeu de données est grand (et *a fortiori* plus nous avons d'entités), plus les résultats estimés dans le graphe anonymisé sont précis. Cela s'explique par la sélectivité réduite des requêtes sur des jeux de données plus grands, c'est-à-dire que nous calculons nos estimations sur des sous-ensembles plus significatifs des entités.

Commençons par nous concentrer sur le premier niveau (a,b). Nous constatons que nos résultats sont conformes à ceux généralement attendus en utilisant l'anatomie. Autrement dit, nous assistons à une erreur relative moyenne inférieure à 2% pour le pire des cas et autour de 0,1% pour le meilleur des cas . De plus, nous constatons que les différentes distributions des valeurs d'ASs n'ont pas d'impact significatif sur l'erreur relative moyenne : une différence d'environ 1% sur le plus petit ensemble de données est observée (elle devient d'autant moins perceptible que la taille du graphique augmente). Quand il s'agit de comparer la précision de notre approche dans un scénario à plusieurs attributs sensibles, nous remarquons des résultats légèrement plus faibles : une différence d'environ 0,6% dans les pires cas. Cela s'explique par le fait qu'une probabilité doit être calculée pour les deux ASs (par opposition à une seule probabilité dans un scénario où un seul attribut est considéré).

Les deux niveaux restants mettent en évidence l'impact de la cardinalité des requêtes sur l'erreur relative. Avec chaque nouvel attribut, nous observons une nette augmentation de l'erreur pour tous les jeux de données, bien que cette augmentation soit plus importante pour les plus petits graphes et qu'elle soit d'autant plus exacerbée lorsque l'on considère plusieurs ASs. Les résultats pour le deuxième niveau sont néanmoins satisfaisants avec une erreur maximale pour LUBM 100 d'environ 12% dans un scénario à un seul AS et de 17% avec deux ASs. En revanche, au troisième niveau, on observe une erreur relative élevée pour LUBM 100 (plus de 30% sur le jeu de la figure 4.7(e)) et même, dans une certaine mesure, pour LUBM 1000 avec un score d'environ 20% dans le pire des cas sur la figure 4.7(f).

Une conclusion qui pourrait être tirée est que notre approche ne s'adapte pas bien à la cardinalité des requêtes, mais l'erreur pour LUBM 5000 et 10000 reste faible (moins de 5%) quelle que soit la configuration. Le problème réside dans le fait qu'à partir de 3 attributs, les requêtes deviennent très sélectives sur les plus petits jeux de données, c'est-à-dire qu'elles renvoient des valeurs de COUNT très basses. Lorsqu'il s'agit de faibles nombres, nous observons des variations beaucoup plus importantes du point de vue de l'erreur relative, même lorsque les valeurs brutes sont relativement proches. Prenons par exemple deux paires de valeurs attendues et estimées ($Exp_1 = 2, Est_1 = 4$) et ($Exp_2 = 40, Est_2 = 50$). Nous calculons l'erreur relative pour les deux paires : nous obtenons un résultat de 100% pour la première et de 25% pour la seconde alors même que l'on pourrait soutenir que la première paire semble plus pertinente si l'on regarde la différence réelle, absolue, entre les valeurs.

Afin d'approfondir la question, nous avons décidé de calculer l'erreur absolue moyenne pour chacune de nos requêtes en plus de l'erreur relative moyenne, les résultats correspondants sont affichés dans la figure 4.8. Nous conduisons les mêmes évaluations par rapport à l'impact des distributions, à la présence de



(a) Deux attributs - un AS

(b) Deux attributs - deux ASs

(c) Trois attributs - un AS

(d) Trois attributes - deux ASs

(e) Quatre attributs - un AS

(f) Quatre attributs - deux ASs

FIGURE 4.8 – Évaluation de l'erreur absolue moyenne

plusieurs ASs ou à la taille du jeu de données. Nous remarquons qu'à mesure que les requêtes deviennent plus sélectives, l'écart entre les valeurs attendues et nos estimations se réduit. Soit C la cardinalité de requête, les écarts vont de 12 à 277 pour $C = 2$, de 3 à 50 pour $C = 3$ et de $0,8$ à 13 pour $C = 4$.

Ainsi, la cardinalité d'une requête n'est pas en soi une limitation pour notre approche. Les performances inférieures s'expliquent par le fait que les requêtes de plus grande cardinalité (*i.e.*, plus sélectives) concernent moins d'entités et que l'erreur relative n'est pas nécessairement la mesure la plus appropriée pour interpréter les résultats dans ces cas de figure.

Nous avons également effectué une évaluation sur le temps requis par notre approche pour se terminer (tableau 4.3). Notre calcul inclut le temps nécessaire pour créer les requêtes de mise à jour (la création des groupes, etc.) ainsi que le temps nécessaire pour les exécuter afin d'anonymiser le graphe.

Nous observons des temps très intéressants malgré la taille des jeux de données. Plus important encore, nous avons mesuré que la majorité du temps était consacrée à interroger le graphe (requêtes SELECT et UPDATE) plutôt qu'à calculer les groupes. En effet, puisque le nombre de valeurs distinctes pour nos

Dataset	Single SA	Two SAs
LUBM 100	3.923	5.183
LUBM 1.000	32.63	63.465
LUBM 5.000	245.555	313.79
LUBM 10.000	585.945	677.032

TABLE 4.3 – Évaluation du temps d'exécution du processus d'anonymisation (en secondes)

ASs reste le même quel que soit le jeu de données (à savoir 9 religions et 6 opinions politiques), l'algorithme de clustering se déroule toujours de la même manière. Pour ces raisons, on observe une durée constante de moins de $0,1$ seconde.

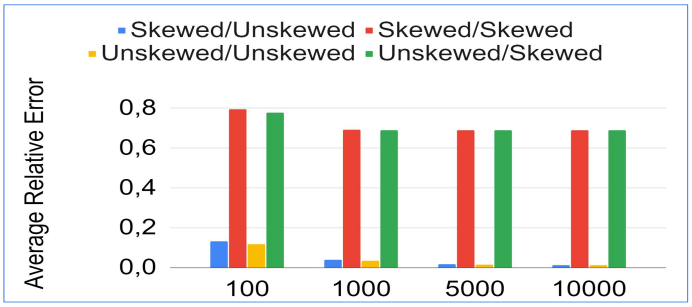
Nous avons également essayé de tester le temps d'exécution de notre approche avec des prédicats autres que *hasReligion* afin d'avoir des valeurs plus distinctes. Cela s'est avéré difficile dans LUBM car les seuls prédicats pertinents que nous avons pu trouver avaient pour objet des professeurs. Autrement dit, en choisissant ces attributs, le nombre de valeurs distinctes pour les ASs augmente linéairement avec la taille du jeu de données. Nous avons réussi à collecter des résultats pour LUBM 100 et 1000 sur le prédicat *advisor* mais les temps devenaient trop élevés même avec LUBM 5000 à tel point que nous avons décidé d'interrompre l'évaluation préemptivement. Nous avons compté 60160 (resp. 599101) valeurs distinctes et avons mesuré un temps d'exécution de 174,267 secondes (resp. 10534 secondes) dont 67,5 (resp. 9861,554 secondes) allouées à l'étape de clustering pour LUBM 100 (resp. LUBM 1000).

En conclusion, nous avons démontré qu'avec plusieurs requêtes intermédiaires, il est toujours possible de récupérer des informations précieuses comme dans le jeu de données d'origine. Par ailleurs, l'anatomie n'est pas grandement affectée par la distribution des valeurs des attributs sensibles et devient d'autant plus efficace à mesure que le nombre d'entités augmente. Nous observons des résultats légèrement pires lorsque nous traitons des requêtes impliquant plusieurs ASs, mais cela est attendu compte tenu de la méthode utilisée pour notre approximation. Enfin, nous avons montré que la cardinalité des requêtes n'était pas non plus une limitation en soit.

4.6.4 Comparaison avec une approche de référence

Dans cette partie, nous comparons notre système à [8] qui est, à notre connaissance, l'une des rares implémentations disponibles d'un logiciel d'anonymisation pour les graphes de connaissances. Notre objectif ici est d'évaluer la capacité d'analyse de requête d'agrégation de [8]. Pour cette comparaison, nous réutilisons des requêtes de cardinalité allant de 2 à 4 impliquant les attributs suivants : *hasReligion*, *politics*, *researchInterest* et un attribut générique personnalisé. Comme précédemment, la religion joue encore le rôle de l'attribut

(a) Deux attributs



(b) Trois attributs

(c) Quatre attributs

FIGURE 4.9 – Évaluation de l'erreur relative de [8]

sensible.

Afin de réaliser cette évaluation, nous avons conçu des politiques de confidentialité et d'utilité adaptées à l'approche de [8]. Intuitivement, après anonymisation, il ne devrait plus être possible de lier une entité à 2 (resp. 3 ou 4) attributs. D'un autre côté, nous visons à mener une analyse sur les religions, elles ne doivent donc pas être anonymisées. Compte tenu de ces politiques, [8] propose 3 alternatives possibles pour mettre à jour les graphes :

1. remplacer les entités par des nœuds vides, garantissant ainsi l'aspect utilitaire des données mais exposant le système à une potentielle *ré-identification par liaison* (*linking attack*) tel que décrit dans [43] ;
2. remplacer un ou plusieurs des QIDs par un nœud vide ;
3. supprimer certains des triplets impliquant les entités et leurs QIDs.

La première approche est capable de fournir des réponses parfaitement précises au détriment de la vie privée, il n'y a donc aucun intérêt à tester son aspect utilitaire. Les deux autres approches ont en revanche un impact sur les QIDs et sont assez similaires à cet égard puisque la valeur d'origine est perdue. Une distinction qui pourrait être notée est que dans (iii), il n'est en théorie même plus possible de savoir que ces attributs existaient en premier lieu. Cette subtilité ne fait pas de différence pour nous du point de vue de l'utilité des données, c'est pourquoi nous avons choisi (ii) afin de mener l'évaluation présentée dans la figure 4.9.

Pour toutes les requêtes, nous désignons la religion comme étant un AS dans la politique d'utilité et incluons le reste des attributs dans la politique de confidentialité. Parmi les requêtes de mise à jour proposées, nous sélectionnons systématiquement l'option d'anonymiser uniquement l'opinion politique, ce qui signifie que toutes les valeurs du prédicat *politics* sont remplacées par des nœuds vides. Puisque nous contrôlons la distribution des attributs dans nos graphes, nous sommes en mesure de tester le comportement de [8] dans différentes configurations.

Sans connaissance externe, la seule hypothèse qu'un attaquant pourrait émettre vis-à-vis de la distribution des opinions politiques est que les valeurs sont réparties de manière égale entre tous les entités. Notez que connaître le nombre de valeurs distinctes pour un attribut qui a été remplacé par un nœud vide n'est pas forcément possible : si l'attribut se trouve appartenir à une hiérarchie de concepts, il serait possible de s'appuyer sur la TBox pour récupérer cette information. Dans une situation où l'attribut serait un littéral (par exemple un nombre ou une chaîne de caractères), la tâche peut se révéler bien plus ardue. Ainsi, pour les besoins de cette évaluation, nous prétendons connaître au préalable le nombre d'opinions politiques distinctes dans l'ensemble de données, noté ($nb_{valeurs}$), sans avoir à le calculer et plus largement, le nombre de valeurs distinctes pour chaque QID (*i.e.*, nous connaissons les intervalles dans lesquels sont contenues les valeurs).

Afin de calculer une valeur approximative, nous commençons par exécuter une requête de comptage impliquant tous les attributs non sensibles, puis nous divisons le résultat par $nb_{valeurs}$. Dans la figure 4.9(a), nous divisons le nombre

d'individus suivant une religion spécifique par $nb_{valeurs}$. Nous procédons de la même manière pour la figure 4.9(b) et (c) mais nous considérons des attributs supplémentaires lors du comptage du nombre d'entités, à savoir *researchInterest* et un attribut générique.

Comme prévu, [8] fonctionne de manière satisfaisante lorsque que la distribution des idées politiques n'est pas biaisée dans la figure 4.9(a) avec une erreur moyenne allant de 2% à $0,2\%$ pour le plus grand jeu de données. La méthode présente par contre des résultats très inférieurs lorsque ce n'est pas le cas, avec plus de 65% d'erreur. Il en va de même pour les figures 4.9(b) et (c) avec une erreur moyenne allant de 1% à 13% (resp. 5% à 48%) lorsque la distribution est uniforme mais culmine à 60% autrement dans le cas contraire. Une fois de plus, nous observons une diminution constante de la précision à mesure que nous considérons plus d'attributs en raison de la sélectivité des requêtes.

Notons que même dans les meilleurs cas, les résultats obtenus par [8] sont légèrement moins bons que les nôtres : moins de $0,5\%$ d'écart dans le pire des cas pour les requêtes de cardinalité 2, autour de 2% pour les requêtes de cardinalité égale à 3 et supérieures à 10% pour les requêtes impliquant quatre attributs. Cela peut s'expliquer par la méthode que nous utilisons pour calculer les estimations avec l'anatomie sémantique : nous nous appuyons sur les cardinalités des ASs tandis que pour [8], nous faisons une hypothèse sur la distribution d'un attribut qui peut ne pas correspondre strictement à la réalité. Quand bien même la distribution serait uniforme, les valeurs peuvent ne pas être réparties parfaitement de manière équitable.

Pour finir sur cette étude comparée, [8] ne garantit pas une analyse efficace pour les requêtes d'agrégat. Nous avons montré que les choix proposés à l'anonymiseur se révélaient être limités.

D'une part, ils suggèrent le remplacement des entités par des nœuds vides, ce qui aurait pour résultat de rendre le graphe vulnérable aux attaques fondées sur l'utilisation des QIDs. De l'autre, ils proposent la suppression d'un ou plusieurs QIDs, ce qui pourrait garantir la confidentialité des entités au détriment de l'utilité du jeu de données car la précision des requêtes d'agrégat deviendrait dès lors trop dépendante de la distribution des valeurs de ces attributs. Cette partie met également en évidence le fait que le choix des opérations d'anonymisation doit être de la responsabilité d'un expert du domaine. En effet, s'il est intéressant de disposer de plusieurs options, il peut être difficile de savoir exactement ce qu'implique une opération vis-à-vis du compromis confidentialité/utilité.

4.7 Conclusion

Nous avons présenté l'anatomie sémantique comme étant une nouvelle approche d'anonymisation adaptée aux graphes de connaissances et prenant en considération certains des aspects sémantiques propres à ces structures. Elle comporte un algorithme se reposant sur des hiérarchies de concepts ainsi que des services de raisonnement pour créer des groupes d'attributs sensibles similaires.

Cette technique se distingue de celles fondées sur la généralisation car elle conserve les valeurs exactes des QIDs tout en supprimant les liens directs entre les entités et leur attribut sensible. De ce fait, elle permet de conserver la corrélation entre les QIDs et les ASs et garantie par conséquent des analyses de données d'une qualité difficilement atteignable.

Cependant, les méthodes reposant sur la généralisation ne sont pas pour autant rendues obsolètes et offrent de leur côté des garanties de confidentialité supérieures à celles d'anatomie notamment car conserver les valeurs des QIDs peut ouvrir la voie à un certain nombre d'attaques.

Fondamentalement, ces deux propositions ont des objectifs contraires : la généralisation accorde une plus grande importance à la confidentialité en anonymisant les QIDs et en conservant les valeurs des ASs tandis que l'anatomie met l'accent sur l'utilité des données en effectuant l'inverse. En conséquence, elle sont efficaces (et vulnérables) face à des types d'attaques également opposés. Dans le prochain chapitre, nous considérons dans quelle mesure il serait possible d'utiliser ensemble les deux approches afin de combler les lacunes de chacune. Le but est ainsi d'intégrer une forme de généralisation des QIDs dans notre anatomie et de protéger davantage les informations personnelles des entités sans pour autant sacrifier l'utilité des données.

Chapitre 5

Empêcher la divulgation des attributs et des entités : combiner k-anonymity et anatomie sur des graphes RDF

5.1 Introduction

Dans ce chapitre, nous revenons sur le compromis confidentialité/utilité dans le contexte du PPDP (*Privacy-Preserving Data publishing*) pour les données RDF en portant cette fois une attention plus particulière au volet confidentialité.

Deux principales formes de divulgation d'informations sont généralement envisagées dans le PPDP :

— la divulgation d'entité (*entity-disclosure*) intervient lorsqu'un individu peut-être explicitement associé à un enregistrement particulier dans les données publiées

— la divulgation d'attributs (*attribute-disclosure*) se produit lorsque la valeur d'un attribut sensible est clairement associable à un individu.

Dans le tableau 5.1, nous présentons un extrait de données non anonymisées. Du point de vue de la divulgation d'entité, si un adversaire connaît le numéro de sécurité sociale (SSN) d'une personne donnée, il retrouvera immédiatement l'enregistrement associé à cette personne. Supprimer les SSNs avant de publier ce jeu de données n'est toutefois pas suffisant car, comme nous l'avons vu dans [43], la combinaison du code postal, de l'âge et du sexe peut représenter suffisamment d'informations pour identifier une personne précise dans la table. À titre d'exemple, dans 5.1, on ne compte qu'un seul homme de 29 ans résidant dans la ville dont le code postal est 80068. La divulgation des attributs est quant à elle particulièrement problématique lorsque l'identité d'un individu est également divulguée. Admettons qu'un adversaire connaisse l'homme de 29 ans habitant à 80068 et qu'il sache par ailleurs que ce dernier a effectué un séjour à l'hôpital. Il peut alors en déduire que cet homme souffre d'asthme.

#	SSN	Zip code	Age	Gender	Condition
1	160-05-4220	80053	28	Male	Asthma
2	160-16-12160	80068	29	Male	Asthma
3	160-05-4228	80068	21	Female	Flu
4	191-12-8800	80053	23	Male	Flu
5	275-80-1220	80053	29	Female	Diabetes
6	223-81-1920	80053	35	Male	Asthma
7	187-15-4229	80068	27	Male	Flu
8	160-17-4228	80068	28	Male	Flu
9	172-09-4242	80053	31	Male	Diabetes
10	175-25-4212	80053	37	Female	Diabetes
11	275-23-8145	80053	36	Female	Diabetes
12	191-51-2353	80053	28	Male	Diabetes

TABLE 5.1 – Jeu de données original

Dans le chapitre précédent, notre travail s'est principalement axé sur l'utilité des données et la protection contre la divulgation d'attributs avec la proposition d'une anatomie sémantique adaptée aux données RDF. Nous introduisons maintenant dans notre framework la technique populaire d'anonymisation k-anonymity afin d'empêcher la divulgation d'entités. Notre objectif ici est donc de renforcer notre approche vis-à-vis de la protection des données personnelles sans pour autant négliger l'utilité des données en général.

Pour appliquer k-anonymity, nous nous appuyons sur les implémentations

disponibles dans l'outil open-source ARX [1]. Considérant l'interaction entre cette technique et notre anatomie sémantique, nous présentons deux algorithmes : l'un où k-anonymity est lancé sur l'ensemble du jeu de données anatomisé et l'autre où k-anonymity est appliqué individuellement sur chaque groupe d'individus généré par anatomie. Notre expérimentation menée sur de grands jeux de données avec différentes valeurs de k met en évidence que le premier algorithme (nommé global) surpasse le second (noté groupe) lorsqu'il s'agit de limiter la perte d'information. Étant donné qu'ARX a été développé pour gérer des données relationnelles, certains ajustements doivent être mis en œuvre de façon à anonymiser des graphes RDF.

5.2 ARX : un outil open-source d'anonymisation

ARX [38] est un outil d'anonymisation open-source permettant à un utilisateur de transformer des données personnelles structurées (*i.e.*, de forme tabulaire) en utilisant la technique de son choix parmi un large éventail. Un grand nombre de principes de confidentialité sont en effet supportés, parmi lesquelles :

— des principes fondés sur la généralisation des données : k-anonymity, l-diversity, t-closeness, etc.
— des principes statistiques tels que k-map.
— des principes dits "sémantiques" [2] tels que (ϵ, δ)-differential privacy

Dans le cadre de notre travail, nous nous sommes principalement intéressés aux fonctionnalités de généralisation présentées par cet outil, en particulier celles relatives au principe de confidentialité k-anonymity. ARX fournit deux types d'algorithmes pour implémenter k-anonymity, à savoir le recodage global et le recodage local. Dans le premier cas, un niveau de généralisation pour chaque attribut est choisi et est appliqué à toutes les classes d'équivalence. Le recodage local permet quant à lui de généraliser les classes différemment les unes des autres. Dans l'intention de préserver au maximum l'utilité des données, nous employons le recodage local car nous avons constaté qu'il permettait de limiter plus efficacement la perte d'informations sans pour autant provoquer d'allongement des temps d'exécution par rapport au recodage global sur nos jeux de données.

En revanche, puisqu'ARX ne prend pas en charge le modèle RDF mais supporte en revanche le format CSV, nous avons dû réaliser un travail de préparation préliminaire des données. Nous extrayons nous-même les valeurs des QIDs des entités depuis notre graphes RDF avant de les soumettre à notre programme Java qui utilise l'API ARX pour produire une version k-anonymisée. Les tuples généralisés sont ensuite utilisés comme base pour construire les requêtes de mise à jour qui anonymiseront le graphe.

Nous fournissons dans la figure 5.1 un diagramme de classes pour l'API ARX. Bien que nous fassions surtout usage des classes définissant les différents principes de confidentialité et les hiérarchies de généralisation (coin inférieur

1. https://arx.deidentifier.org/
2. https://arx.deidentifier.org/overview/

gauche), on peut constater que l'outil ne se limite pas qu'à cela et offre une panoplie de fonctionnalités concernant notamment l'analyse de l'utilité des données.

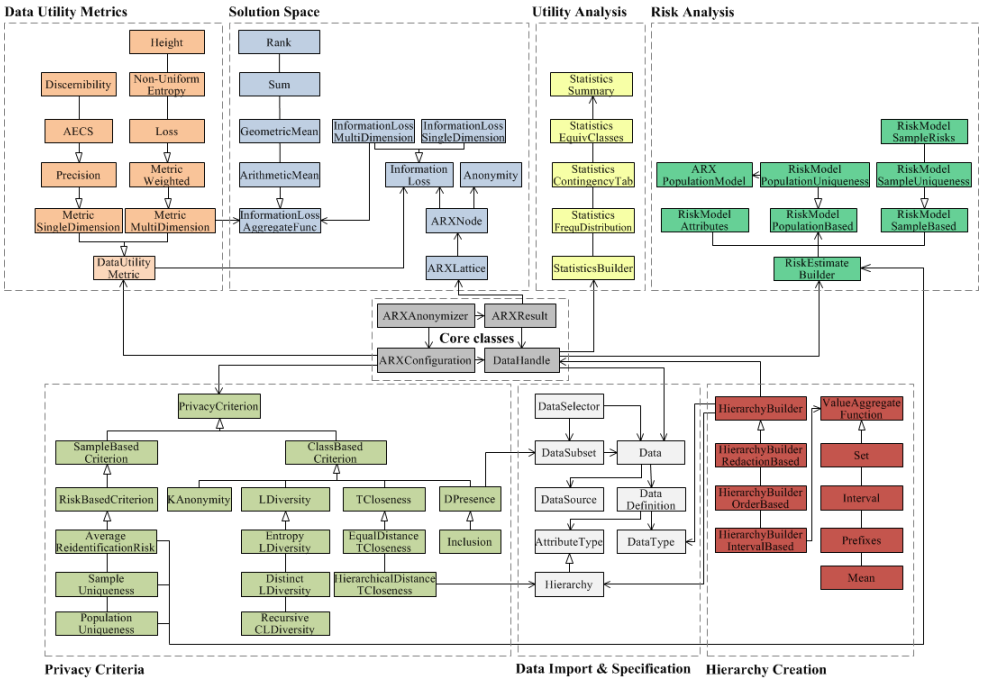


FIGURE 5.1 – Diagramme de classes UML de l'API ARX

5.3 Appliquer k-anonymity conjointement avec l'anatomie sémantique

5.3.1 Justification de notre approche

À première vue, l'utilisation simultanée de l'anatomie et de k-anonymity paraît contre-intuitive puisque les deux techniques ont des philosophies diamétralement opposées. En effet, la première vise à empêcher la divulgation d'attributs en remplaçant les ASs des entités par des groupes d'ASs afin de créer de l'incertitude. Au cours de ce processus, les QIDs ne sont pas modifiés. À l'inverse, l'objectif de k-anonymity consiste à généraliser les QIDs pour prévenir la divulgation d'entités, les ASs restant quant à eux inchangés. Ainsi, si nous appliquons les deux méthodes simultanément, nous prenons le risque de perdre en efficacité sur les deux fronts. Néanmoins, notre intuition est que cela pourrait également nous permettre d'atténuer certaines de leurs faiblesses respectives.

Du point de vue de k-anonymity par exemple, nous pouvons répondre à l'attaque connue sous le nom de *skewness attack* [30] qui se produit lorsqu'une classe d'équivalence contient des enregistrements partageant tous la même valeur pour leur AS. Dans cette situation, si un adversaire sait que sa cible est présente dans la classe en question, il peut découvrir son attribut sensible. La technique l-diversity propose de faire face à ce type d'attaques en imposant un certain degré de *diversité*, autrement dit d'incertitude, sur les valeurs des ASs dans chaque classe d'équivalence. Lorsque nous créons les groupes pour l'anatomie, nous apportons également cette notion d'incertitude dans le sens où si une entité appartient à un groupe alors il peut être potentiellement associé à chacun des attributs sensibles dans le groupe selon une certaine probabilité. Dans notre implantation actuelle, contrairement à l-diversity qui a été conçue dans ce but, nous n'imposons aucune contrainte sur le nombre minimum de valeurs distinctes dans un groupe d'ASs. Actuellement, la seule condition que doit respecter un groupe est de contenir au moins deux ASs mais il ne serait pas difficile de fournir un seuil sur la taille minimale à respecter au moment où nous les créons. D'ailleurs, l'algorithme proposé dans l'article original décrivant anatomie [48] était lui-même construit sur la définition de la l-diversity. En effet, il y était expliqué que l'approche cherchait à calculer une "*partition l-diverse* "[30] du jeu de données qui était ensuite utilisée pour produire les deux tableaux anatomisés.

Concernant l'anatomie, le choix de ne pas modifier les QIDs permet la ré-identification des entités présentes dans le jeu de données (comme cela a été montré dans [43]). Dans certains modèles de confidentialité, le fait de savoir qu'un individu se trouve dans un jeu de données peut, en soi, constituer une violation de son droit à la vie privée. La question du "refus plausible" d'inclusion ou d'exclusion (*the plausible denial of inclusion or exclusion*) des participants à un jeu de données est notamment abordée dans les travaux relatifs à la differential privacy [13]. En d'autres termes, qu'un individu appartienne ou non au jeu de données d'origine, cela ne devrait avoir qu'une influence négligeable sur le jeu anonymisé qui sera produit. Sa présence ou son absence peuvent donc toutes deux être rationnellement réfutées. En ajoutant la généralisation des QIDs à notre proposition d'anatomie, nous souhaitons compliquer la ré-identification et permettre dans une certaine mesure ce type de réfutation.

Dans ce chapitre, nous espérons montrer que malgré la perte de précision inhérente à l'utilisation conjointe de ces deux approches, il est encore possible d'obtenir de bonnes performances au niveau de l'utilité des données vis-à-vis de certains types de requêtes et, en même temps, de combler certaines de leurs lacunes respectives au niveau de la protection des données.

5.3.2 Modélisation des attaques contre notre approche

Dans cette section, nous nous plaçons dans le même contexte que celui présenté dans [43]. Imaginons un attaquant ayant réussi, à l'aide d'une source de données externe, à récupérer les QIDs de sa cible. Dans le cas où k-anonymity aurait été appliqué, il doit supposer l'identité de celle-ci parmi au moins k enregistrements possibles. Toutefois, il n'y a aucune garantie sur la diversité des

ASs possédés par ces enregistrements, le risque de divulgation des attributs est par conséquent difficile à évaluer. Pour notre approche, nous considérons deux scénarios possibles : i) un scénario dans lequel les enregistrements possibles appartiennent tous au même groupe anatomisé. ii) un second dans lequel les enregistrements sont répartis entre plusieurs groupes.

Prenons la figure 5.2 comme exemple : pour des raisons de clarté, nous nous sommes limités à un seul QID (un code postal). Considérons qu'un adversaire recherche les informations d'une personne habitant la zone dont le code postal commence par "*805*" : les enregistrements possibles pour la cible sont donc *x1* et *x2*. Par un calcul probabiliste, nous trouvons que ces individus ont 10 chances sur 15 d'avoir la grippe et 5 sur 15 de souffrir d'asthme. Dans notre implantation, nous pouvons garantir une protection équivalente à 2-diversity distincte mais nous rappelons que la manière dont les groupes sont formés pourrait être ajustée afin d'avoir des garanties plus strictes.

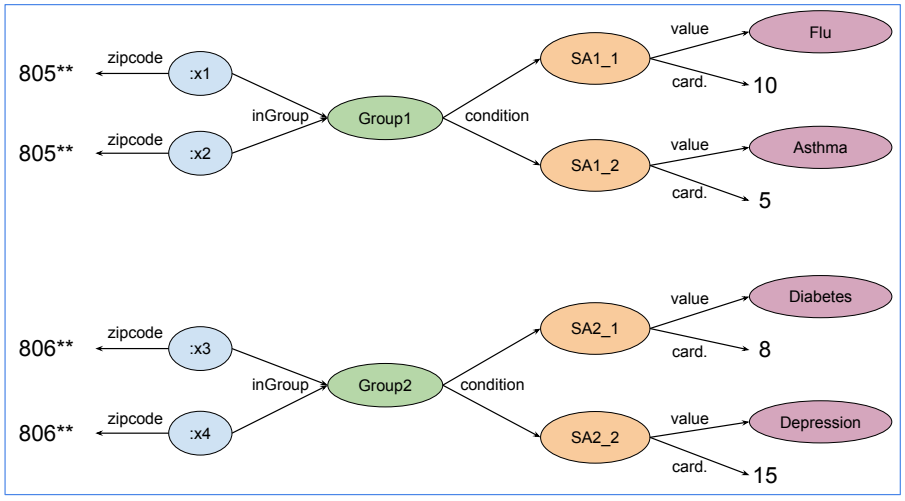


FIGURE 5.2 – Application de l'anatomie et de k-anonymity

Quant au deuxième scénario, en plus de l'incertitude liée à l'anatomie, il faut aussi tenir compte de l'incertitude qui accompagne le choix du groupe.

Soit G l'ensemble des groupes contenant au moins un enregistrement et N le nombre d'enregistrements possibles. Pour chaque groupe G_i dans G, nous nommons $N(G_i)$ le nombre d'enregistrements susceptibles d'être la cible de l'attaquant dans G_i. Soit r un enregistrement probable appartenant à G_i, pour chaque AS contenue dans G_i, la probabilité que cet attribut soit réellement lié à r est donnée par : $(N(G_i)/N) * (card\_SA/card\_group)$

Toujours avec la figure 5.2, imaginons cette fois un attaquant cherchant quelqu'un vivant dans une zone dont le code postal commence par "*80*" : $x1$, $x2$, $x3$ et $x4$ sont tous des enregistrements possibles. L'attaquant doit d'abord choisir un groupe, ici $Group1$ ou $Group2$. Puisque les deux groupes contiennent

le même nombre d'enregistrements susceptibles d'être la cible, ce dernier a donc 50% de chances d'avoir raison. Une fois ce choix réalisé, il ne reste qu'à calculer la probabilité pour les ASs de la même manière que nous l'avons fait dans le premier scénario. En conséquence, la probabilité, par exemple, que la cible souffre de la grippe est égale à $2/4 * 10/15 = 1/3$. Le même raisonnement est ensuite appliqué à tous les autres ASs pour obtenir l'intégralité des probabilités.

5.3.3 K-anonymity global

Rappelons une dernière fois qu'ARX ne supporte pas le modèle RDF et qu'une étape de préparation des données est nécessaire. En supposant que nous disposons d'assertions définissant les quasi-identifiants dans le jeu de données, nous proposons un algorithme dont le but consiste à extraire les valeurs de QID d'un graphe, à les convertir au formalisme CSV supporté par ARX puis à les généraliser.

En outre, ARX attend des utilisateurs qu'ils définissent eux-mêmes des hiérarchies de généralisation pour les attributs nécessitant d'être anonymisés (*i.e.*, QIDs). Ces hiérarchies sont fournies en tant que paramètres à ARX et permettent d'effectuer un certain nombre de transformations sur les données, allant de leur simple suppression jusqu'à leur généralisation dans des intervalles prédéfinis.

Dans la suite de cette section, nous présentons une explication étape par étape de l'algorithme dit "global" affiché dans Algo. 2. Avant toute chose, plusieurs paramètres doivent être fournis :

- le graphe à anonymiser.
- la liste des prédicats correspondant aux QIDs.
- les chemins vers les fichiers CSV dans lesquels sont décrites les hiérarchies de généralisation des QIDs. Elles permettent de définir comment une valeur en particulier est censée être généralisée.
- une valeur de k pour la généralisation.

À la ligne 1, nous récupérons les tuples de QIDs (c'est-à-dire les tuples constitués d'un identifiant pour l'entité et des valeurs de ses QIDs) depuis le graphe G grâce à une requête SPARQL de type SELECT. Celle-ci s'appuie sur la liste P qui contient les prédicats des QIDs.

L'algorithme est appelé "global" car la requête (et par extension l'application de k-anonymity) s'applique à toutes les entités du graphe : aucun filtre sur la base des groupes formés par l'anatomie n'est utilisé. Par le biais de ces tuples de QIDs, nous produisons un fichier CSV qui est ensuite chargé dans l'API ARX en même temps que les hiérarchies pour les QIDs (lignes 2 à 4). Nous continuons en initialisant un objet ARX Anonymizer dont la tâche est d'exécuter k-anonymity sur les données en employant un algorithme de recodage local (lignes 5,6). Nous parcourons chaque tuple généralisé afin de produire une requête SPARQL Update à exécuter (lignes 8 à 11). Finalement, le graphe anonymé peut être conservé sur disque pour une utilisation ultérieure. (ligne 12)

Pour étayer notre propos, nous présentons dans la figure 5.3 un exemple des requêtes de mise à jour générées à partir des données transformées que

Algorithm 2: Algorithme "global" pour l'application de k-anonymity à un graphe de connaissances

Input : Knowledge graph : G, QID predicates : P, QID hierarchies : H, K

1 $data \leftarrow retrieve\ the\ original\ data\ from\ G\ using\ P$;
2 $Flush$ data $to\ csv\ format$;
3 $Load\ data\ from\ csv\ to\ ARX$;
4 $Load$ H $to\ ARX$;
5 $anonymizer \leftarrow initializeKAnonymity(K)$;
6 $transformedData \leftarrow anonyizer.anonymize(data)$;
7 $queries \leftarrow empty$;
8 **foreach** row in $transformedData$ **do**
9 $\quad queries \leftarrow queries \cup \{computeQuery(row)\}$;
10 **foreach** q in $queries$ **do**
11 $\quad Update\ G\ by\ executing\ q$;
12 $Flush\ updated\ graph\ to\ disk$;

produit ARX. Imaginons que nous ayons transformé l'âge et le code postal d'un individu quelconque. L'âge a été généralisé dans un intervalle, sa valeur a donc été remplacée par un nœud vide permettant de réifier les bornes inférieure et supérieure de l'intervalle de généralisation. Le code postal a quant à lui été remplacé par une valeur où le dernier chiffre a été tronqué.

5.3.4 K-anonymity par groupe

Cet algorithme fonctionne essentiellement de la même manière que le précédent, la principale différence étant qu'il exécute plusieurs procédures de k-anonymity, une pour chaque groupe formé par l'anatomie. Alors que la requête utilisée dans Algo. 2 à la ligne 2 permet de récupérer les tuples de toutes les entités du jeu de données, ici nous les récupérons progressivement en nous concentrant sur les entités appartenant à un groupe particulier G. Pour ce faire, nous ajoutons un triplet supplémentaire "?s http://inGroup G " à la requête. La même procédure que précédemment est ensuite réalisée, à savoir la conversion au format CSV et l'anonymisation du graphe par l'intermédiaire de requête SPARQL. Nous répétons le processus pour chaque groupe présent dans le jeu de données.

5.4 Évaluation

L'évaluation pour ce chapitre a été effectuée avec le même contexte computationnel que celui de la section 4.6.1.

```
PREFIX ex: <http://example#>
DELETE {
    ex:Maxime ex:age ?a .
    ex:Maxime ex:zipcode ?z .
}
INSERT {
    ex:Maxime ex:age <http://Blank_X> .
    <http://Blank_X> ex:min 20 .
    <http://Blank_X> ex:max 29 .
    ex:Maxime ex:zipcode "7742*".
}
WHERE {
    ex:Maxime ex:age ?a .
    ex:Maxime ex:zipcode ?z .
}
```

FIGURE 5.3 – Exemple de requête de mise à jour pour la généralisation de quasi-identifiants

5.4.1 Jeux de données et requêtes

Nous réutilisons les jeux de données LUBM 100, 1000 et 5000 introduits dans la section 4.6.2 (un récapitulatif spécifique à ces trois jeux est disponible dans la table 5.2). Encore une fois, les entités d'intérêt correspondent aux nœuds de type "Professor" et les AS aux concepts issus de notre ontologie des religions. Nous ajoutons aux graphes les trois QIDs que nous avons définis dans la section 2.4.2, à savoir le sexe, l'âge et le code postal, et attribuons à chaque professeur une valeur aléatoirement choisie pour chacun de ces attributs.

De plus, chaque QID se voit associé à une hiérarchie de généralisation (définie dans un fichier CSV). Pour les âges, nous faisons usage de plusieurs intervalles : des intervalles de 10 ans, *e.g.*, $[20, 30]$, etc. , des intervalles de 20 ans ainsi qu'une généralisation totale (équivalent plus ou moins à une suppression). Concernant les codes postaux, nous permettons le troncage d'un certain nombre des chiffres les plus à droite ("80642" peut par exemple devenir "806**"). Enfin pour le sexe, la valeur est soit conservée telle quelle, soit supprimée.

Ainsi, pour produire les jeux de données anonymisés, nous appliquons l'anatomie sur les religions et k-anonymity sur les trois QIDs. Notre objectif est de nouveau d'évaluer l'aspect utilitaire de notre approche du point de vue de requêtes d'agrégat. En ce sens, nous considérons dans cette évaluation des requêtes de comptage impliquant les âges, les codes postaux et les religions. Il s'avère qu'ARX n'a pas besoin d'anonymiser le sexe pour former des classes d'équivalence, c'est pourquoi nous ne l'utilisons pas.

Quoi qu'il en soit, en raison des transformations que nous réalisons sur les données, deux difficultés se posent vis-à-vis de l'utilisation de telles requêtes. Tout d'abord, nous rencontrons le même problème relatif à l'anatomie qu'au chapitre précédent : puisque les liens directs entre les entités et les ASs ont été

supprimés, nous devons avoir recours à plusieurs requêtes intermédiaires pour compter les ASs.

La seconde difficulté est liée à l'utilisation de k-anonymity : il a été montré dans [48] que les techniques fondées sur la généralisation n'étaient pas adaptées aux requêtes d'agrégat. Afin d'atténuer ce phénomène, nous exprimons les clauses WHERE de nos requêtes SPARQL avec des intervalles similaires à ceux que nous fournissons à ARX pour généraliser les données. De ce fait, dans toutes les requêtes utilisées pour produire les figures de cette partie, les âges des professeurs ne sont pas requêtés directement mais par le biais d'intervalles de 10 ans *i.e.*, $[20, 30), ..., [90, 100)$. Nous ne recherchons pas non plus les codes postaux complets (car ils pourraient avoir été tronqués) mais des préfixes de codes postaux.

Un point important à mentionner est que l'évaluation présentée ici est reprise directement de notre publication [46] dans laquelle nous utilisions le rappel (*recall*) pour mesurer l'utilité de jeux de données anonymisés plutôt que *l'erreur relative* définie au chapitre précédent 4.6.2 . Nous tenons à préciser que les deux mesures sont en réalité très similaires. En effet, étant donné une valeur attendue Exp et une valeur estimée Est, le rappel est défini comme $\frac{Est}{Exp}$. L'erreur relative est quant à elle définie de la manière suivante :
$\frac{|Exp-Est|}{Exp} = \frac{Exp}{Exp} - \frac{Est}{Exp} = 1 - \frac{Est}{Exp}$
On obtient ainsi l'égalité $recall = 1 - relativeError$.

	Taille (GB)	# de triplets (en millions)	# de professeurs
LUBM 100	2,3	13.5	60268
LUBM 1000	23,8	135.3	600356
LUBM 5000	120,2	676.3	2998826

TABLE 5.2 – Caractéristiques des jeux de données

5.4.2 Durée du calcul des généralisations

Dans cette section, nous mesurons la durée de calcul des généralisations pour les trois jeux de données LUBM. Nous considérons nos deux approches de généralisation (globale et par groupe) et cinq valeurs de k (de 2 à 15). Les résultats sont présentés dans la figure 5.4. Premièrement, cette évaluation souligne que la durée de nos deux approches de généralisation est similaire peu importe le jeu de données (notez comme les lignes pleines et pointillées coïncident presque). Pour les deux approches, nous sommes capables de généraliser à un taux d'environ 1700 triplets/seconde. De plus, la durée de généralisation pour chaque jeu est presque constante peu importe la valeur de k.

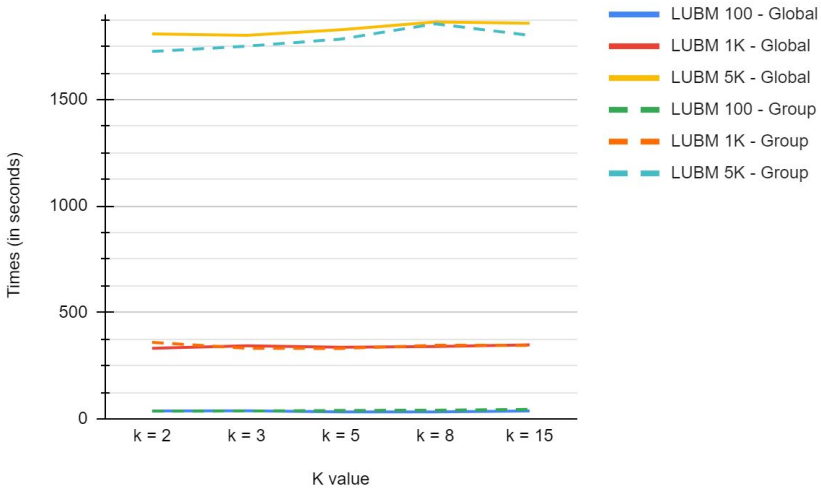


FIGURE 5.4 – Temps d'exécution de la généralisation pour différentes valeurs de k

5.4.3 Requête de comptage avec deux QIDs et un AS - 300 codes postaux

Dans l'évaluation affichée dans la figure. 5.5, nous nous intéressons à l'utilité de LUBM 100 et LUBM 1000 après anonymisation dans un contexte où les valeurs des codes postaux sont comprises entre 80000 et 80300 (non inclus), pour un total de 300 valeurs réparties uniformément entre tous les professeurs

Le rappel est calculé pour les deux algorithmes (global et par groupe) avec différentes valeurs de k.

Pour les deux algorithmes, nous observons qu'à mesure que la taille du graphe augmente, le nombre de tuples parmi lesquels choisir pour la généralisation augmente également, ce qui conduit à une amélioration du rappel. À l'inverse, lorsque k augmente, plus de généralisations peuvent être nécessaires pour former les classes d'équivalence et cela a un impact négatif sur le rappel. Cet effet délétère reste cependant moins visible avec une plus grande taille de graphe.

Avec Algo. 2, les rappels sur LUBM 1000 sont constamment au-dessus de la barre des 90% même avec des valeurs plus grandes de k, ce qui n'est pas nécessairement le cas sur LUBM 100 où le rappel tombe à 36% dans le pire des cas avec $k = 15$).

Concernant l'algorithme par groupe, bien qu'il obtienne des résultats nettement moins bons sur LUBM 100 (en particulier lorsque la valeur de k augmente) avec un rappel de 5% lorsque $k = 15$, il présente des résultats comparables ceux de l'autre algorithme sur LUBM 1 000. La raison à cela est que le nombre d'entités par groupe anatomisé est trop bas dans LUBM 100 pour des valeurs

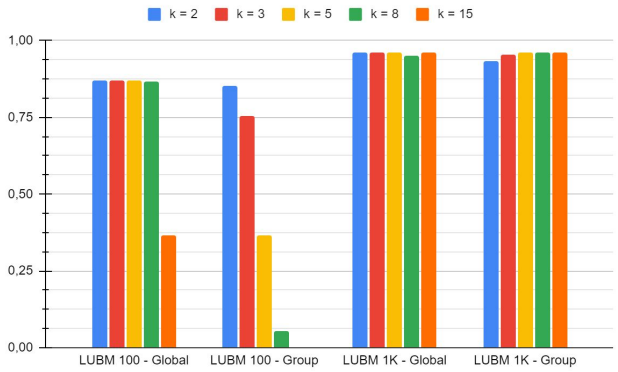


FIGURE 5.5 – Rappel moyen pour le requêtage des âges (par intervalles de 10 ans), des codes postaux (300 valeurs, un chiffre supprimé) et des religions

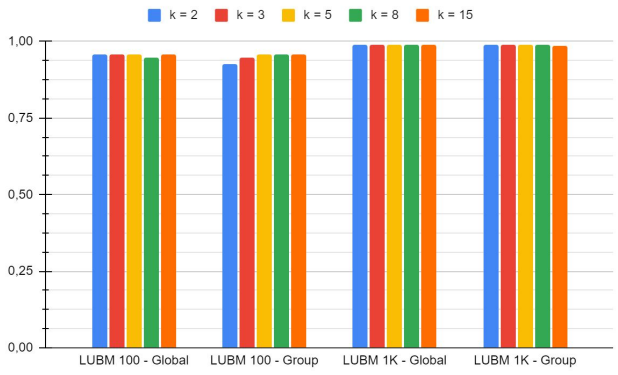


FIGURE 5.6 – Rappel moyen pour le requêtage des âges (par intervalles de 10 ans), des codes postaux (300 valeurs, deux chiffres supprimés) et des religions

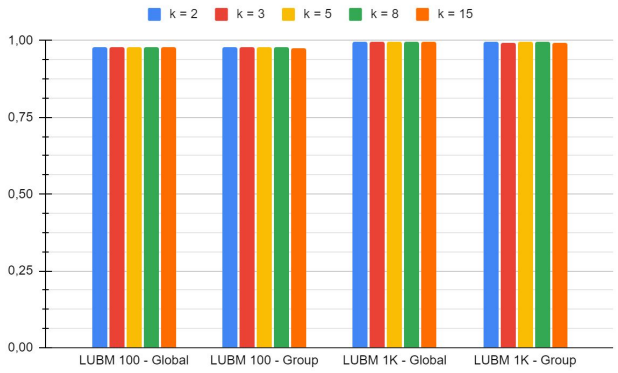


FIGURE 5.7 – Rappel moyen pour le requêtage des âges (par intervalles de 10 ans), des codes postaux (300 valeurs, trois chiffres supprimés) et des religions

supérieures de k, ce qui oblige ARX à généraliser les tuples plus que dans les autres cas. Étant donné que la version globale est appliquée à un plus grand ensemble d'entités, moins de modifications des données sont nécessaires, ce qui se traduit par plus d'utilité à tous les niveaux.

Les figures 5.6 et 5.7 confirment nos observations : puisque nous recherchons des codes postaux plus généralisés, il n'y a plus de baisses de performances notables pour des valeurs supérieures de k après l'application de l'algorithme par groupe. En effet, les rappels dépassent tous la barre des 95% dans toutes les configurations avec des pics à 99% sur la figure 5.13.

Néanmoins, un inconvénient de cette évaluation est que nous utilisons un nombre relativement faible de valeurs distinctes pour les codes postaux (300) par rapport au nombre total d'entités (voir le nombre de professeurs dans le tableau 5.2, *e.g.*, pour LUBM 100 , en moyenne 200 professeurs partagent le même code postal. Avec de si petites valeurs de k (≤ 15), il n'est par conséquent pas nécessaire pour ARX de généraliser cet attribut pour créer des classes d'équivalence.

5.4.4 Requête de comptage avec deux QIDs et un AS - 3000 codes postaux

De façon à mettre à l'épreuve notre approche, nous avons décidé d'utiliser un total de 3000 codes postaux dans la plage $[80,000, 83,000)$. En outre, nous incluons maintenant les résultats d'un jeu de données supplémentaire, LUBM 5000.

Dans la figure 5.8, les rappels sont plus faibles pour l'algorithme global par rapport à la figure 5.5 car nous avons beaucoup plus de diversité dans les valeurs de code postal. Cela oblige ARX à entreprendre de plus grandes transformations des données pour garantir l'anonymat des entités, de sorte qu'il devient plus

difficile de trouver des codes postaux tronqués d'un seul chiffre. Les requêtes sur LUBM 100 sont les plus affectées : nous pouvons voir le rappel tomber en dessous de 50% pour de faibles valeurs de k (3 et 5) et même atteindre 0% à partir de $k = 8$. Bien que l'on constate une légère baisse des performances pour LUBM 1000 (le rappel étant d'environ 86% alors qu'il était supérieur à 95% auparavant), les résultats restent élevés sauf pour le cas de $k = 15$ où les généralisations des codes postaux deviennent trop importantes et conduisent à un score de l'ordre de 36%.

La sur-généralisation provoquée par l'algorithme par groupe en comparaison avec la version globale est davantage mise en évidence ici avec des différences de performances évidentes sur LUBM 100 et 1 000, notamment lorsque k augmente, les résultats allant de 57% à 0% pour LUBM 100 et de 85% à 5% pour LUBM 1000.

Du côté de LUBM 5000, les valeurs de rappel pour l'algorithme global restent constamment supérieures à 90%, ce qui est prévisible car peu de généralisations sont nécesaires tant le nombre d'entités est élevé. Nous ne montrons aucun résultat pour l'algorithme par groupe sur ce jeu en raison de contraintes de temps, le processus d'évaluation prenant environ 20 heures à se finir.

La diversité croissante des valeurs de code postal a eu pour conséquence de rendre les requêtes plus sélectives sur nos jeux de données (en particulier sur les plus petits), autrement dit, elles renvoient des valeurs de comptage moins élevées. À titre d'exemple, dans la figure 5.5, les valeurs récupérées étaient en moyenne égales à 30 (LUBM 100) et 310 (LUBM 1000) alors que ces moyennes tombent à 3 et 30 pour la figure 5.8 pour les mêmes jeux de données.

Lorsqu'il s'agit de faibles nombres, nous avons vu au cours du chapitre précédent que les mesures telles que l'erreur relative ou le rappel n'étaient pas très appropriées pour obtenir des résultats pertinents. C'est pourquoi nous avons décidé de calculer l'erreur absolue en plus du rappel, les résultats sont affichés dans la figure 5.9. Nous voyons que, dans le pire des cas, la différence moyenne entre la valeur attendue et la valeur estimée pour LUBM 100 est d'environ 3 et d'environ 30 pour LUBM 1000 alors même que le nombre total d'entités dans ces jeux de données est égal à 60268 et 600356 respectivement.

Les figures 5.10 et 5.11 présentent de bien meilleurs résultats même sur des graphes plus petits avec un k plus élevé. On observe des rappels supérieurs à 85% dans le pire cas (LUBM 100) et jusqu'à 98% dans le meilleur (LUBM 5000). Une autre chose à noter est qu'il n'y a pratiquement aucune différence entre les algorithmes globaux et par groupe à tous les niveaux : il y a suffisamment d'entités dans chaque groupe anatomisé (même pour LUBM 100) pour que la majorité des codes postaux n'aient eu que deux chiffres tronqués au maximum. Pour cette même raison, la valeur de k n'a pas d'impact significatif sur les résultats : $k = 15$ n'est pas suffisant pour pousser ARX à généraliser davantage.

Puisque nous recherchons des codes postaux plus généralisés dans nos requêtes, nos estimations sont calculées sur de plus grands sous-ensembles des graphes, comme le corroborent les résultats affichés dans la figure 5.11. En effet, bien que les erreurs absolues soient toujours dans les mêmes ordres de grandeur

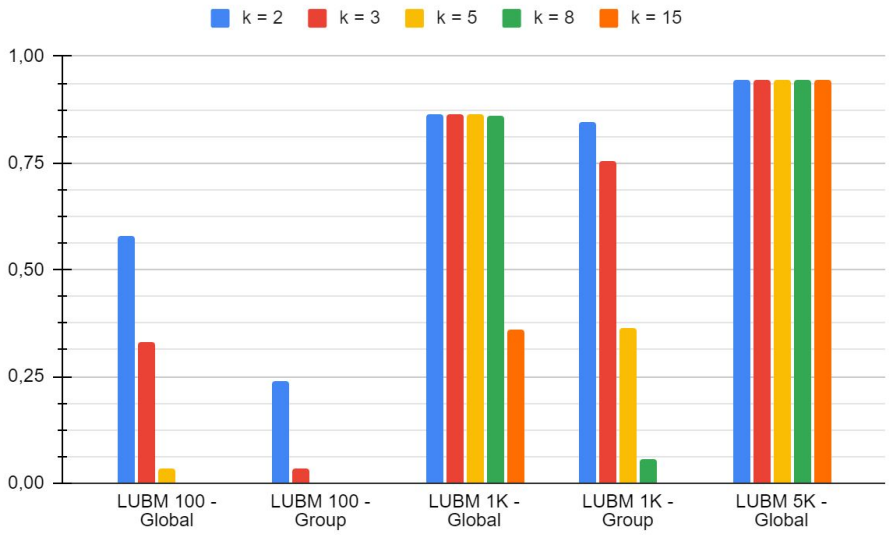


FIGURE 5.8 – Rappel moyen pour le requêtage des âges (intervalles de 10 ans), des codes postaux (3000 valeurs, un chiffre supprimé) et des religions

que 5.9, les rappels ont quant à eux augmenté.

Dans la figure 5.12, nous constatons une augmentation notable des rappels pour LUBM 100 qui dépassent maintenant 96%. Les résultats déjà élevés de LUBM 1000 tournent désormais autour de 98% et ceux de LUBM 5000 culminent à plus de 99% dans certains cas. De même, nous observons dans la figure 5.13 des erreurs moyennes absolues bien inférieures au nombre total d'entités dans chaque jeu de données.

Ces derniers graphiques montrent que nous atteignons les limites de ce qui peut être évalué avec ces seules requêtes puisque les rappels frôlent les 100% presque partout. Pour aller plus loin, il serait nécessaire d'augmenter leur sélectivité et dans ce cas, plusieurs pistes peuvent être explorées :

— imposer plus de diversité dans les QIDs (par exemple en créant plus de codes postaux).
— utiliser des requêtes impliquant davantage d'attributs.
— augmenter la valeur de k afin de provoquer plus de généralisations.

Sur ce dernier point, notons qu'il n'existe pas de conventions ou de régulation spécifiant quelles valeurs de k sont les plus appropriées. Dans le milieu de la santé où les données médicales sont généralement échangées entre un petit nombre de personnes, la valeur de k est souvent comprise entre 5 et 15 [15]. Quoi qu'il en soit, le choix de la valeur de k reste une question compliquée tant il est difficile de quantifier le niveau de protection réellement apporté par une plus grande valeur.

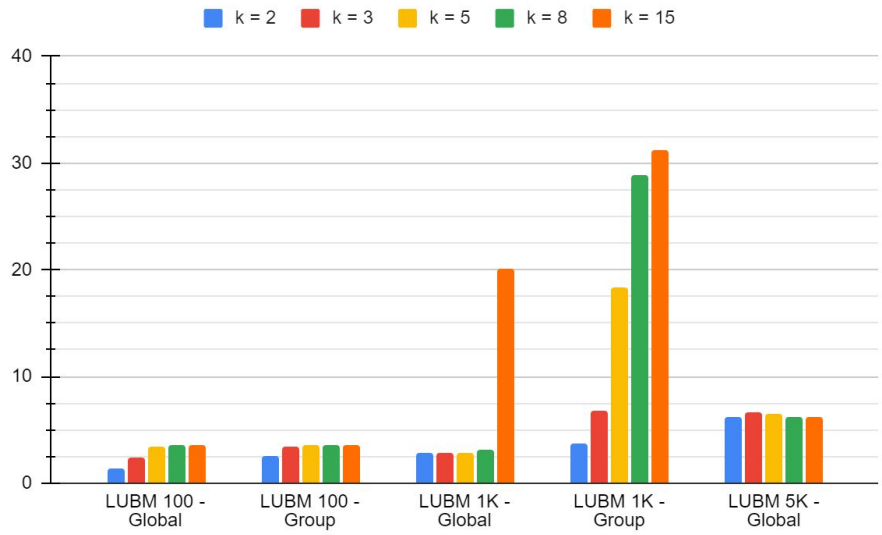


FIGURE 5.9 – Erreur absolue moyenne pour le requêtage des âges (intervalles de 10 ans), des codes postaux (3000 valeurs, un chiffre supprimé) et des religions

5.5 Conclusion

Nous avons considéré dans ce chapitre le compromis confidentialité/utilité pour les données représentées avec le modèle de données RDF. Sur la base de nos travaux précédents sur l'anatomie qui traitaient la divulgation d'attributs, nous avons introduit l'approche k-anonymity afin de répondre cette fois à la divulgation d'entités. Nous avons proposé deux méthodes (notées globales et par groupe) afin d'anonymiser dans un deuxième temps des graphes RDF déjà anatomisés. Nous avons par ailleurs identifié, via des évaluations approfondies, que l'algorithme global fournissait de meilleures mesures de rappel et d'erreur absolue. Ainsi, nous observons une amélioration de la confidentialité sans pour autant sacrifier totalement l'utilité des données au regard de requêtes analytiques.

Toutefois, comme pour toute solution d'anonymisation, trouver les paramètres optimaux, en l'occurrence ici, la valeur de k pour k-anonymity reste un problème difficile. De toute évidence, ces solutions ne suivent pas une philosophie unique (*one-size-fits-all*) et une compréhension fine des données contenues dans un jeu de données est primordiale à la publication de données sûres, capables de résister à différents types d'attaque.

Cette affirmation est non seulement vraie en ce qui concerne la confidentialité, mais également pour l'utilité des données, dans le sens que les hiérarchies de généralisation utilisées pour anonymiser un jeu influencent directement la ma-

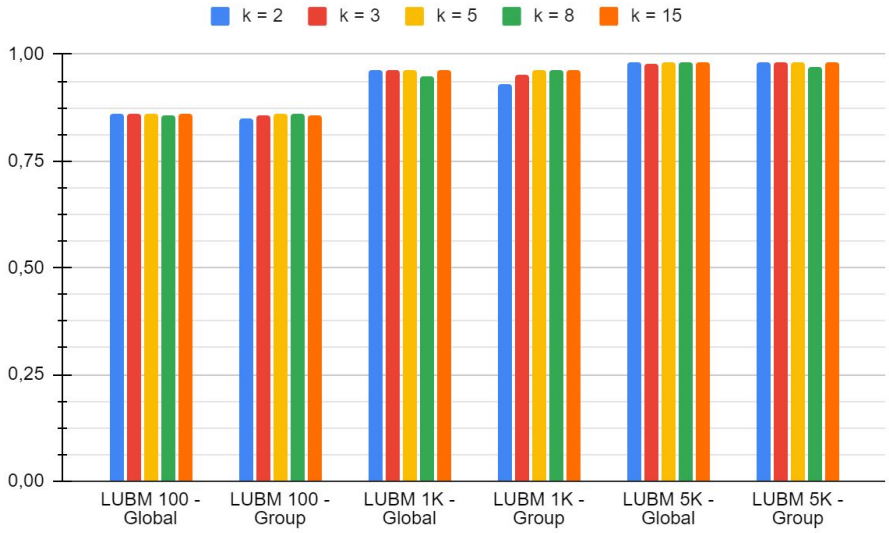


FIGURE 5.10 – Rappel moyen pour le requêtage des âges (intervalles de 10 ans), des codes postaux (3000 valeurs, deux chiffres supprimés) et des religions

nière dont les données pourront être interrogées par la suite et vice-versa (*i.e.*, les volontés d'analyse doivent motiver le choix d'une technique par rapport à une autre).

Jusqu'à présent, nos travaux se sont essentiellement tournés vers l'anonymisation de jeux de données "figés dans le temps" et publiés une unique fois. Dans la réalité cependant, les jeux de données sont dynamiques et les données évoluent au fur et à mesure du temps notamment par le biais de suppressions ou d'insertions. Considérer l'anonymisation dans un contexte dynamique est difficile car cela ouvre la voie à toute sorte de nouvelles attaques que les techniques classiques ne sont pas capables de gérer.

C'est l'objet du chapitre suivant dans lequel nous présentons Kgastor, un framework d'anonymisation conçu pour être intégré directement avec un système de gestion de base de données RDF.

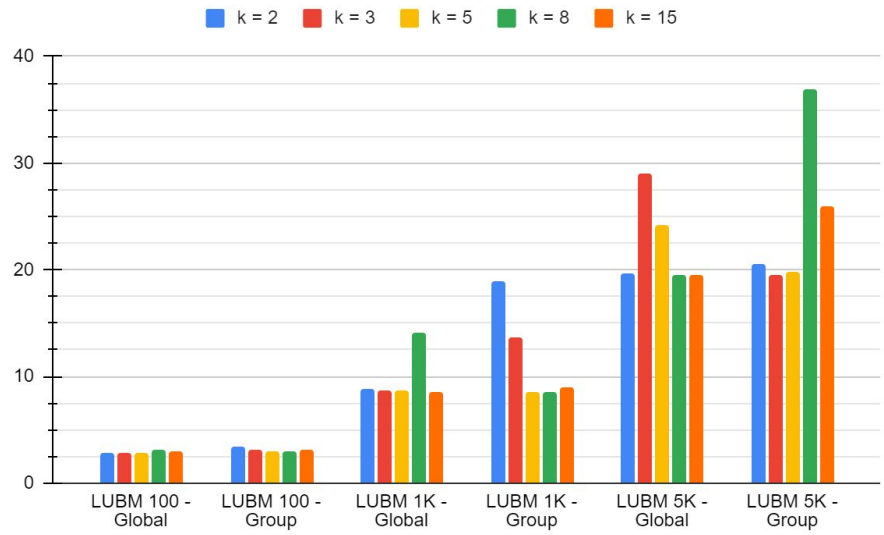


FIGURE 5.11 – Erreur absolue moyenne pour le requêtage des âges (intervalles de 10 ans), des codes postaux (3000 valeurs, deux chiffres supprimés) et des religions

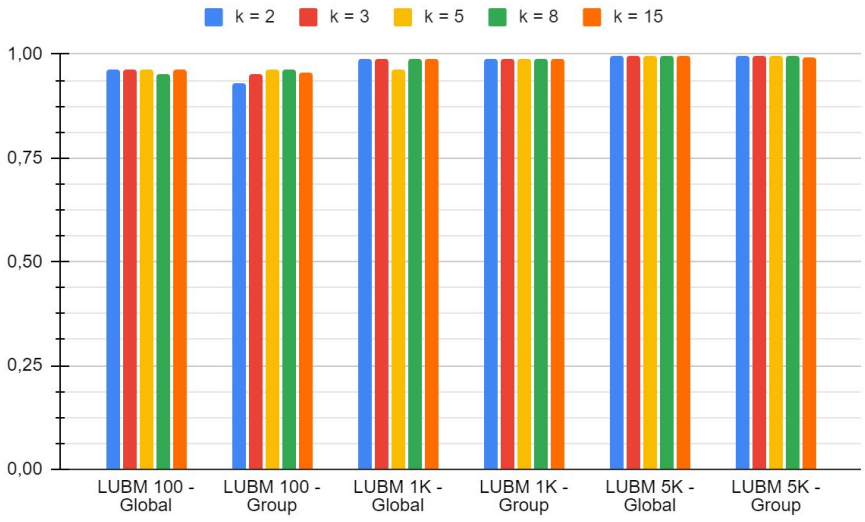


FIGURE 5.12 – Rappel moyen pour le requêtage des âges (intervalles de 10 ans), des codes postaux (3000 valeurs, trois chiffres supprimés) et des religions

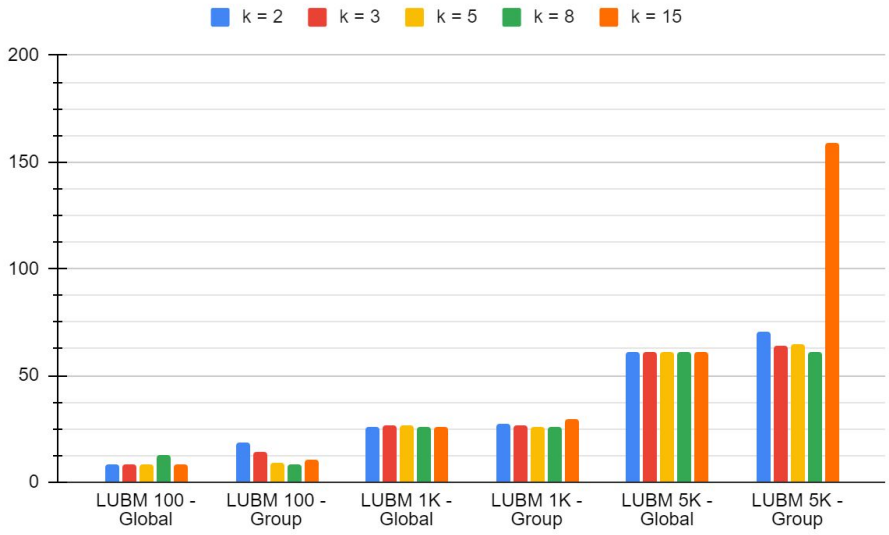


FIGURE 5.13 – Erreur absolue moyenne pour le requêtage des âges (intervalles de 10 ans), des codes postaux (3000 valeurs, trois chiffres supprimés) et des religions

Chapitre 6

Kgastor : un système de gestion de graphes de connaissances anonymisé

6.1 Introduction

L'émergence d'applications reposant sur les graphes de connaissances nécessite de relever de nouveaux défis dans la conception de systèmes de gestion de bases de données (SGBD) pour le modèle de données RDF. Parmi ceux-ci, la

préservation de la confidentialité des données dans le contexte de leur publication (traduit littéralement de "Privacy-Preserving Data publishing") est une question particulièrement délicate pour les organisations confrontées à des réglementations telles que le RGPD [1] (Règlement général sur la protection des données) et le CCAP [2] (California Consumer Privacy Act). Cela ne se limite toutefois pas seulement à donner un accès indirect aux données à des personnes extérieures à l'organisation, par exemple via une interface Web, une application ou un dump de base de données. Des problèmes internes aux organisations peuvent effectivement entrer en ligne de compte dans le sens où un utilisateur malveillant d'une base de données pourrait directement extraire et partager des données sensibles avec le monde extérieur. À titre d'exemple, [24] a pu mettre en lumière que les attaques par atteinte à la vie privée étaient souvent le fait d'utilisateurs légitimes tels que les analystes de données, ces derniers bénéficiant d'un accès illimité à une base de données.

Conformément aux principes de confidentialité dès la conception (en anglais, *privacy-by-design*)[5], nous considérons qu'une base de données RDF est le composant informatique où l'anonymisation des données doit être prise en charge afin de garantir que les considérations de confidentialité soient traitées le plus tôt possible. Du point de vue d'une approche PPDP plus sécurisée, cela doit se traduire par l'intégration d'une stratégie d'anonymisation au cœur du système de gestion des données et le support d'un mécanisme de contrôle d'accès adapté. Récemment, ces questions ont attiré l'attention des fournisseurs de systèmes de gestion de bases de données et ont commencé à être prises en compte dans un certain nombre d'entre eux, principalement relationnels, tels que SAP HANA [25], Microsoft [33], Oracle via ses capacités de rédaction et de masquage de données [36] ou bien les SGBD open-source, *e.g.*, PostgreSQL avec son extension Anonymizer [3].

Néanmoins, parmi l'étendue des approches d'anonymisation historiques qui pourraient être intégrées à un SGBD, un point important est constamment négligé : la nature dynamique des données. En effet, dans les cas d'usage pratique, les données sont constamment sujettes à des changements (*e.g.*, insertions, suppressions). Les publications successives de versions d'une même base qui est mise à jour représentent d'autant plus d'informations pouvant être utilisées par un attaquant et posent donc un réel problème en ce qui concerne la protection des données personnelles, le risque étant qu'une version antérieure de la base permette d'identifier des entités dans une version plus récente (et vice-versa). En raison de leur incapacité à considérer les données dans le temps, les méthodes d'anonymisation classiques (*e.g.*, k-anonymity [43], l-diversity [30] ou t-closeness [35]) ne sont par conséquent pas adaptées à la situation.

De leur côté, certaines techniques, à l'image de m-invariance [49], ont été mises au point spécifiquement en tenant compte de ce contexte dynamique. Étant donné une base de données ayant subi des modifications telles que des insertions, des modifications ou encore des suppressions de données, m-invariance

1. https://bit.ly/2zTZWgk
2. https://bit.ly/3SQB3GW
3. https://bit.ly/3QKtIXA

produit un plan d'anonymisation en accord avec les précédentes publications (anonymisées) qui ont pu être faites de cette base. Intuitivement, elle se charge de conserver une forme de régularité au niveau des classes d'équivalence au travers des diverses publications et introduit, lorsque c'est nécessaire, des données contrefaites afin de conserver certaines propriétés globales du jeu de données.

Dans ce chapitre, nous considérons des SGBDs fondés sur le modèle de données RDF. Ces systèmes attirent de plus en plus l'attention et sont également confrontés aux défis du PPDP. Tout comme avec les SGBDR, nous pensons que pour prévenir les atteintes à la vie privée, les SGBD RDF doivent appliquer des méthodes de protection de la vie privée dès l'ingestion des données.

Nous présentons un framework appelé Kgastor (*KG anonymized store*) conçu pour être placé au-dessus d'un système de gestion de base de données RDF afin d'anonymiser des graphes de connaissances.

Partant de métadonnées fournies par la personne en charge de l'anonymisation, notre système adopte une stratégie de partitionnement afin de scinder les données en deux graphes distincts : d'une part un graphe dit "privé", de l'autre un graphe "par défaut". Le premier, réservé à des utilisateurs privilégiés, contient les données sensibles non-anonymisées des entités tandis que le second, accessible à l'intégralité des utilisateurs, contient ces mêmes données anonymisées. Le lien entre les deux graphes est assuré par la présence de nœuds vides gérés par Kgastor.

La distinction entre les différents types d'utilisateur est laissée à un composant de contrôle d'accès fonctionnant au niveau du graphe (par opposition à d'autres approches qui placent le curseur au niveau des triplets RDF). Associé à notre processeur de requêtes, il empêche les utilisateurs non privilégiés d'obtenir les résultats de certaines de leurs requêtes. Par ailleurs, dans une volonté de rendre le partitionnement des données transparent pour les utilisateurs, le processeur de requêtes SPARQL de Kgastor est équipé d'une solution de réécriture qui, en fonction du rôle d'un utilisateur, modifie sa requête afin que celle-ci s'exécute sur le graphe adéquat.

Enfin, l'anonymisation est déléguée à un autre composant capable de traquer les modifications effectuées sur le jeu de données depuis la dernière publication. Ces informations sont consignées dans un journal puis utilisées pour produire une nouvelle version du jeu. Dans le cadre de ce travail, nous avons eu recours à une adaptation de m-invariance pour anonymiser les données cependant nous tenons à souligner que Kgastor a été développé dans l'idée de n'être dépendant d'aucune technique en particulier. En réalité, à partir du moment où une approche peut garantir la protection de données personnelles dans un contexte dynamique, elle peut tout à fait être utilisée dans notre système.

6.2 Anonymisation de données dynamiques

Le problème entre les techniques de généralisation classiques et les données dynamiques réside dans le fait que ces méthodes ne sont pas capables de garder une trace des anonymisations précédentes pour un enregistrement donné. Fon-

ID	Age	Zip code	Medical condition
Bob	[25, 35]	805**	Flu
Steve	[25, 35]	805**	Gastritis
Gary	[40, 50]	8022*	Gastritis
Linda	[40, 50]	8022*	Dyspepsia

Release 1

ID	Age	Zip code	Medical condition
Bob	[35, 45]	805**	Flu
Jane	[35, 45]	805**	Gastritis
John	[40, 50]	804**	Gastritis
Mary	[40, 50]	804**	Dyspepsia

Release 2

(a) Intersections de QIDs

ID	Age	Zip code	Medical condition
Bob	[25, 35]	805**	Flu
Steve	[25, 35]	805**	Dyspepsia
Gary	[40, 50]	8022*	Gastritis
Linda	[40, 50]	8022*	Dyspepsia

Release 1

ID	Age	Zip code	Medical condition
Bob	[25, 35]	805**	Flu
Jane	[25, 35]	805**	Gastritis
John	[30, 50]	804**	Gastritis
Mary	[30, 50]	804**	Dyspepsia

Release 2

(b) Intersections d'attributs sensibles

FIGURE 6.1 – Attaques par intersection sur deux publications de données

damentalement, un risque peut survenir lorsqu'un enregistrement est présent dans au moins deux publications distinctes et est affecté à différentes classes d'équivalence dans chacune d'entre elles. Le terme "différent" peut ici signifier deux choses. Premièrement, que les classes d'équivalence incluent des enregistrements dont les QIDs ne peuvent être contenus dans les mêmes intervalles et doivent donc être généralisés différemment, c'est-à-dire que les intervalles utilisés pour anonymiser les QIDs d'un enregistrement ne sont pas cohérents d'une version à l'autre. Deuxièmement, les classes peuvent différer selon les valeurs d'attributs sensibles qu'elles contiennent : une valeur peut apparaître dans l'une mais pas dans l'autre et vice-versa. En croisant plusieurs versions, il devient possible d'inférer de nouvelles informations liées aux QIDs et aux ASs de certaines entités, ce qui permet ainsi à un attaquant de restreindre les enregistrements pouvant correspondre à sa cible. Finalement, le risque est que le nombre de possibilités devienne suffisamment faible au point de compromettre le principe de confidentialité appliqué au jeu de données.

Un exemple de ces attaques est fourni dans la figure 6.1 : nous y considérons deux QIDs (l'âge et le code postal) et un seul AS (un état de santé), la colonne ID n'est présente que par souci de clarté et n'apparaîtrait pas dans un cas d'usage réel. Chaque jeu est k-anonymisé (avec $k = 2$) et publié deux fois. La figure 6.1a) décrit les inférences qui peuvent être faites sur la base de différentes généralisations des QIDs : supposons que l'attaquant sache que Bob apparaît dans les deux publications et que son code postal est 80502, il sait par conséquent qu'il ne peut appartenir qu'à la première classe dans les deux versions. En réalisant l'intersection des intervalles d'âge, l'attaquant peut en déduire que Bob a 35 ans. De même, sur la figure 6.1b), un attaquant muni des mêmes in-

formations pourrait traquer les classes d'équivalence dans lesquelles il apparaît ainsi que les ASs qui leur sont associées ({Flu, Dyspepsia} et {Flu, Gastritis}). L'intersection des deux ensembles donne le véritable état de santé de Bob, à savoir Flu (la grippe).

6.3 M-invariance

À partir de ce point, nous utiliserons indifféremment les termes "tuples" et "enregistrements" pour désigner les entités d'intérêt et leurs attributs (QIDs et attribut sensible).

Dans [49], les auteurs présentent m-invariance comme un principe de généralisation pour le modèle de données relationnel dont la satisfaction assure la protection d'informations personnelles dans un contexte dynamique. Dans les faits, l'approche qui est proposée se démarque de celles qui la précèdent (*e.g.*, k-anonymity ou l-diversity) car elle tient compte des mises à jour (*e.g.*, insertions ou suppressions) ayant pu être effectuées sur un jeu de données avant de l'anonymiser.

Elle cherche ainsi à former des classes d'équivalence qui doivent respecter les règles suivantes :

i. chaque classe doit compter au moins m tuples.

ii. aucune classe d'équivalence ne doit contenir deux tuples avec la même valeur pour leur attribut sensible.

iii. si un tuple est présent dans plusieurs publications anonymisées, les classes d'équivalence auxquelles appartient ce dernier doivent impérativement contenir les mêmes valeurs d'attributs sensibles : cet ensemble est défini comme étant la *signature* de la classe d'équivalence. Par exemple dans la figure 6.1b, Bob appartient à deux classes possédant des signatures différentes : $\{Flu, Dyspepsia\}$ dans la première publication et $\{Flu, Gastritis\}$ dans la seconde d'où la possibilité d'attaquer les données de Bob en intersectant ces deux ensembles.

Concrètement, cela signifie que durant son "temps de vie" dans le jeu de données et à travers les différentes publications, un tuple doit systématiquement être associé au même ensemble d'attributs sensibles, autrement dit, cet ensemble de valeurs doit rester **invariant**.

Par ailleurs, une contribution essentielle de cette technique pour gérer les suppression de données, qui sont susceptibles dans certains cas d'entraîner l'absence d'une valeur d'AS et ainsi de mettre en péril cette notion d'invariance, est d'introduire des tuples factices. Ces derniers sont indiscernables des autres tuples présents dans le jeu de données et sont utiles pour compléter certaines classes d'équivalence lorsque cela est nécessaire.

L'application de m-invariance se déroule en 4 phases distinctes que nous détaillons maintenant avec un exemple où $m = 2$.

Division

Nom	**Religion**
Emily	Protestant
Mary	Muslim
Ray	Catholic
Tom	Muslim
Vince	Protestant

TABLE 6.1 – Exemple de tuples nouvellement insérés

Cette phase consiste à récupérer les tuples présents dans le jeu de données et qui étaient déjà présents lors de la publication précédente. Ces tuples sont ensuite utilisés pour construire ce que l'on appelle des buckets, tels que chaque bucket ne contienne que les tuples partageant la même signature (par extension, la signature d'un tuple correspond à celle de la classe d'équivalence à laquelle il appartient). Dans la figure 6.2, nous donnons l'exemple de quatre buckets obtenus après cette phase depuis un jeu de données fictif dans lequel les attributs sensibles seraient des religions.

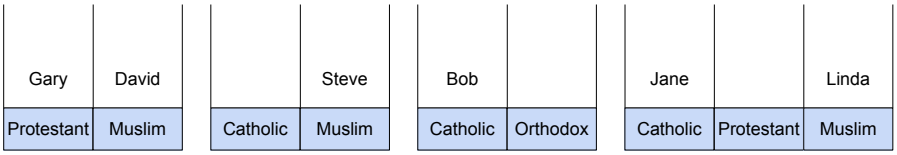


FIGURE 6.2 – Exemple de buckets après la phase de division

Balancing

Un bucket est dit équilibré si chacun de ses attributs sensibles est associée au même nombre de tuples. Pour équilibrer les buckets obtenus après la division, on puise dans l'ensemble des tuples nouvellement insérés dans le jeu de données (que nous nommons *newTuples*). Néanmoins, les nouveaux tuples ne peuvent être déplacés que tant que *newTuples* reste *m-eligible* c'est-à-dire que la proportion de l'AS le plus représenté doit être égale ou inférieure au ratio $1/m$. Le maintien de cette propriété est fondamental car il garantit que l'algorithme puisse effectivement se terminer. S'il s'avère impossible d'utiliser *newTuples* pour compléter un bucket, un tuple factice est créé. Ce scénario peut se produire dans le cas où aucun nouvel enregistrement ne possède l'attribut sensible désiré ou dans celui où la suppression d'un tuple entraînerait la violation de la m-éligibilité.

Nous proposons dans la table 6.1 un ensemble d'enregistrements nouvellement insérés et montrons dans la figure 6.3 le contenu des buckets après la phase de balancing. Notez la présence des deux tuples factices : *F1* et *F2*.

Assignment

Dans cette phase, les tuples restants à l'intérieur de *newTuples* sont affectés

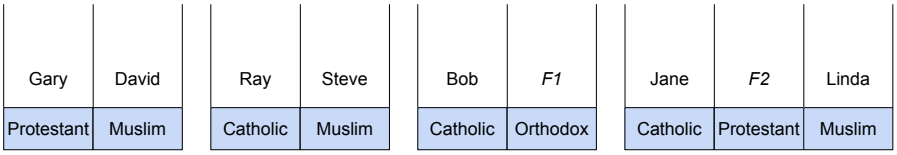


FIGURE 6.3 – Contenu des buckets après balancing

à un bucket dont la signature contient leur attribut sensible (si aucun bucket ne correspond, un nouveau est créé à la volée). Au terme de ce processus, tous les tuples sont affectés à un bucket et tous les buckets sont équilibrés. Nous fournissons un exemple dans la figure 6.4.

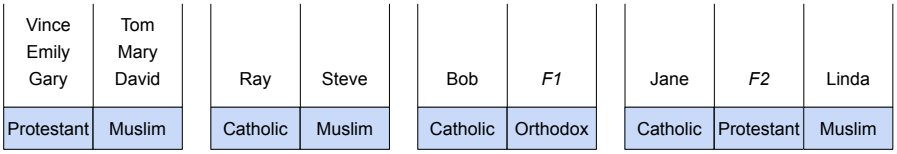


FIGURE 6.4 – Contenu des buckets après assignment

Split

Enfin, les buckets peuvent être divisés en classes d'équivalence, c'est-à-dire en buckets équilibrés contenant au moins m valeurs d'AS, chacune étant associée à exactement un tuple. En vue de minimiser la quantité d'informations perdues par la généralisation, cette phase a pour objectif de regrouper les tuples dont les valeurs de QID peuvent être incluses dans le plus petit intervalle possible.

Chaque bucket peut ainsi être considéré comme un ensemble de $s \geq m$ attributs sensibles associés chacun à une liste d'enregistrements. Étant donné que les buckets sont équilibrés à ce stade, chaque liste contient le même nombre N d'éléments. L'enjeu consiste alors à trouver un index, tel que $index \in [1, n-1]$, permettant de scinder ces listes pour créer deux buckets équilibrés BUC_1 et BUC_2. Dans chaque liste, nous prenons les i premiers tuples de manière à former BUC_1, les tuples restants sont quant à eux utiliser pour remplir BUC_2. Le choix de i s'appuie sur une mesure définie comme la somme du périmètre de BUC_1 et BUC_2, un périmètre correspondant à la taille des intervalles minimums requis pour contenir les valeurs des QIDs d'un bucket.

Nous présentons dans la figure 6.5 les deux plans de fractionnement possibles pour le bucket le plus à gauche dans nos exemples précédents. Dans le premier, l'index est égal à 1, ce qui signifie que les premiers tuples pour chaque attribut sensible seront réunis pour former un nouveau bucket tandis que les enregistrements restants serviront à en créer un autre. Le même raisonnement peut être fait avec le second plan où cette fois ce sont les deux premiers tuples de chaque

colonne qui sont utilisés pour former BUC_1.

Ces opérations de scission sont effectuées jusqu'à l'obtention de toutes les classes d'équivalence.

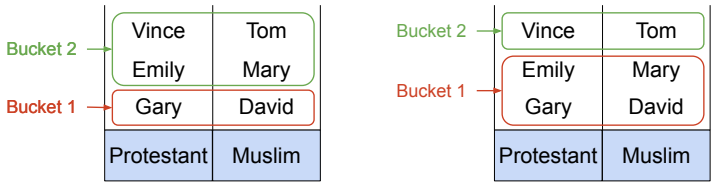


FIGURE 6.5 – Plan de scission pour le premier bucket de 6.4

6.4 Présentation de Kgastor

Un aperçu de l'architecture de Kgastor est fourni dans la Figure 6.6. Il présente les différents composants et leurs interactions.

Le composant de contrôle d'accès accorde des privilèges distincts à différents rôles d'utilisateur. Nous distinguons les utilisateurs privilégiés, *e.g.*, DPO (*Data protection officer*), des utilisateurs non privilégiés, *e.g.*, , les développeurs d'applications ayant besoin d'un accès à la base de données. Selon les cas, DBAs (*Database administrators*) comme analystes de données peuvent appartenir à l'une ou l'autre de ces catégories. La granularité des privilèges se situe au niveau du graphe, c'est-à-dire un état intermédiaire entre un accès fin au niveau des triplets et un accès plus générique s'appliquant au jeu de données dans sa globalité. Ceci est principalement dû à notre approche de partitionnement de graphes qui distribue les triplets dans deux graphes nommés différents.

Les DPO et DBA privilégiés sont chargés de préciser de manière déclarative le contexte d'anonymisation. Autrement dit, ils doivent définir des correspondances entre des requêtes SPARQL et ce qu'ils ont identifié comme ASs, EIDs et QIDs. Ces assertions de correspondance sont stockées dans le composant Mapping Description.

Ensuite, les DBA privilégiés peuvent soumettre le jeu de données au composant Anonymizer qui, à l'aide de m-invariance et de l'API ARX, traite l'anonymisation. Le résultat de cette anonymisation ainsi que les assertions de correspondance sont utilisés pour partitionner le graphe d'origine, ce qui se traduit en dernier lieu par la création de deux graphes nommés respectivement Default et Private graphs.

En ce qui concerne le composant de traitement des requêtes, tout utilisateur autorisé peut soumettre des requêtes de type `select` à Kgastor. Les utilisateurs non privilégiés peuvent uniquement récupérer les données du graphe par défaut tandis que les utilisateurs privilégiés peuvent soumettre des requêtes `select` indépendamment du graphe questionné. Afin de rendre le partitionnement

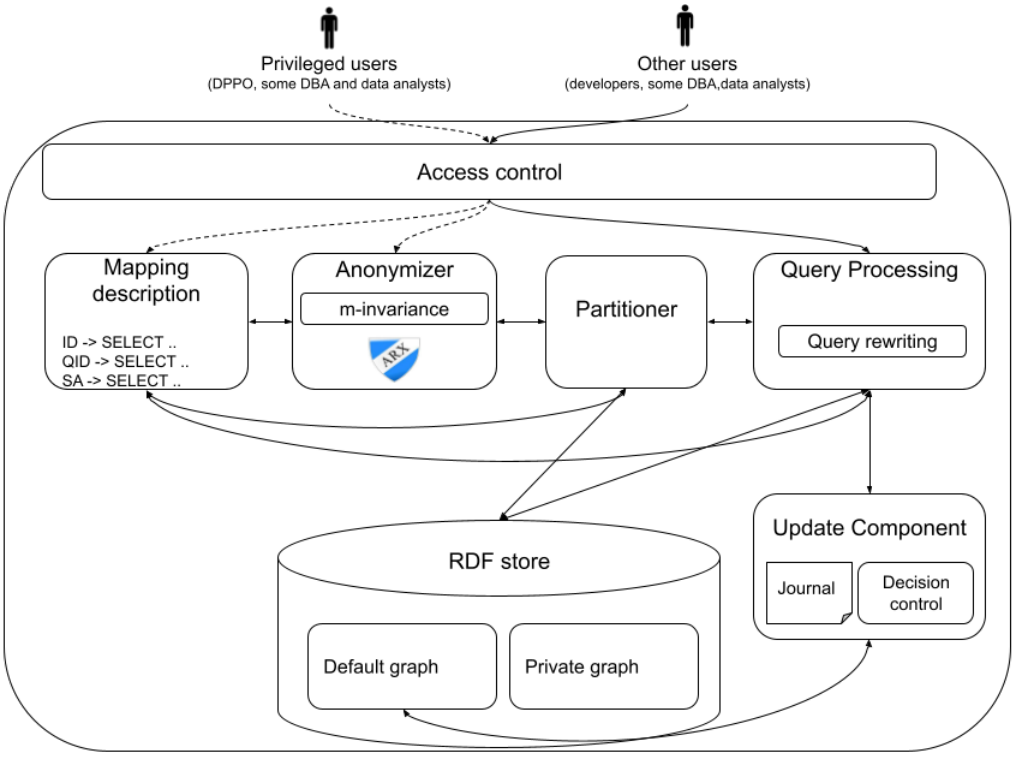


FIGURE 6.6 – Présentation de Kgastor

transparent pour les utilisateurs privilégiés, une solution de réécriture des requêtes a été mise en place pour récupérer automatiquement les données des deux graphes pour les requêtes qui le nécessitent. La détection de telles requêtes est également traitée automatiquement.

Par ailleurs, seuls les utilisateurs privilégiés peuvent exécuter des requêtes `update`. Lorsque celles-ci n'impactent que le graphe privé, la mise à jour est effectuée immédiatement, c'est-à-dire au moment de l'exécution de la requête. Pour les requêtes `update` affectant le graphe par défaut, les opérations sont différées et ces dernières sont temporairement stockées dans le journal du composant Update. Le déclenchement de l'exécution de ces requêtes est de la responsabilité du composant Decision Control. Ce dernier peut être configuré de manière à déclencher la mise à jour du graphe par défaut périodiquement ou en fonction du nombre de requêtes contenues dans le journal de mise à jour.

Plusieurs composantes de Kgastor, telles que le graphe privé et le journal de mise à jour, sont protégées en étant stockées sur des serveurs accessibles uniquement à un nombre restreint d'utilisateurs privilégiés.

6.5 Une intégration de m-invariance au modèle de données RDF

Dans cette partie, nous nous concentrons sur le composant Anonymizer. Nous y présentons notre adaptation de m-invariance au modèle de données RDF ainsi que ses interactions avec l'outil d'anonymisation ARX. Plus précisément, pour chacune des phases de l'algorithme, nous proposons notre méthode pour intégrer celle-ci dans notre système. Notez d'ailleurs que puisque la phase d'assignment n'a pas nécessité d'ajustements particuliers pour être implémentée, cette dernière ne fera pas l'objet d'une section ici.

6.5.1 Division

Comme nous l'avons expliqué précédemment, l'objectif de cette phase consiste à récupérer les tuples qui étaient déjà présents dans le jeu de données lors de la dernière publication afin de faire en sorte que leur signature reste invariante.

De manière à retrouver plus facilement les classes d'équivalence de la publication précédente, nous avons fait le choix d'ajouter un nouvel attribut aux enregistrements, noté *classID*, servant à mémoriser la classe d'appartenance d'un tuple. Avec une simple opération `group by` sur cet attribut, les classes peuvent ainsi être très facilement recréées.

Cependant, étant donné que notre système fonctionne de manière asynchrone en ce qui concerne les données par défaut et les données privées, deux problèmes se posent. Tout d'abord, il faut tenir compte du fait qu'un certain nombre de tuples dans le graphe privé puissent avoir été supprimés entre deux publications. Dans ce cas, retrouver la signature d'une classe qui a perdu un de ses membres devient impossible. Pour résoudre cela, nous utilisons un journal de requêtes dont la présentation sera faîte dans une section suivante : nous filtrons chaque entrée correspondant à une opération de suppression ou de mise à jour afin de retrouver les tuples manquants (nous considérons les mises à jour comme une double opération constituée d'une suppression suivie d'une insertion). Deuxièmement, nous devons être prudents vis-à-vis des tuples nouvellement insérés car ils n'ont aucune incidence sur les classes d'équivalence précédentes. Pour ce faire, nous nous appuyons de nouveau sur l'attribut *classID* : s'il n'y a pas de valeur définie, cela signifie que l'enregistrement apparaît pour la première fois, il est donc ignoré.

6.5.2 Balancing

Afin d'équilibrer les buckets, nous utilisons, si possible, les tuples nouvellement insérés contenus dans l'ensemble *newTuples*. En nous basant sur l'indexation de l'attribut *classID*, la récupération de cet ensemble est rapide.

Dans le cas où l'on ne serait pas en mesure d'équilibrer l'un des buckets, un tuple factice, possédant l'attribut sensible manquant et des valeurs de QIDs copiés sur l'un des tuples dans le bucket, est produit. Cette situation entraîne immédiatement la création d'une nouvelle entrée de suppression dans le journal

des requêtes afin d'indiquer que ce tuple devra être supprimé au moment de publier la prochaine version.

6.5.3 Split

Bien que le principe de cette phase soit expliquée en détail dans [49], aucune implémentation n'est fournie pour la recherche de l'index i alors même que cette opération peut se révéler être critique du point de vue des performances. Pour cette raison, nous présentons ici notre propre algorithme pour la réalisation de cette tâche.

Algorithm 3: Récupérer la liste des valeurs extrêmes des QIDs dans l'ordre inverse

```
   Input  : Bucket : B
            QID Attributes : QID
 1 L ← retrieve the tuples lists from B;
 2 N ← number of tuples in each list;
 3 minMax ← stores min and max values for each QID;
 4 listMinMax ← Empty list;
 5 for i = 0 ; i ≤ N do
 6 |   foreach list in L do
 7 |   |   tuple ← list.get(N − 1 − i);
 8 |   |   foreach qi in QID do
 9 |   |   |   if tuple.qi > MinMax.qi.max then
10 |   |   |   |   Update MinMax.qi.max;
11 |   |   |   if tuple.QI < minMax.qi.min then
12 |   |   |   |   Update minMax.qi.min;
13 |   listMinMax.add(MinMax)
```

Schématiquement, si nous reprenons la figure 6.5 en guise d'exemple, étant donné un bucket B à scinder, l'algorithme 3 permet de parcourir dans l'ordre inverse les tuples contenus dans chaque colonnes de B afin de récupérer les valeurs extrêmes (minimum et maximum) de chacun des QIDs de ces tuples. Dans l'algorithme, L représente par conséquent une liste en deux dimensions où chaque sous-liste correspond à une colonne du bucket.

Concernant la notion de "sens inverse", nous reprenons la terminologie définie dans split où les tuples sont indexés et l'enjeu consiste à trouver le bon index afin de couper un bucket en deux. Dans 6.5, nous avons choisi comme convention de faire évoluer les indices du bas vers le haut : les tuples au plus bas sont d'indice 0, ceux d'au-dessus d'indice 1, etc.

Pour résumer, Algo. 3 renvoie une liste d'objets contenant, pour chaque QID, les valeurs minimale et maximale rencontrées parmi les i derniers tuples dans chaque colonne. Concrètement, le premier élément dans $listMinMax$ correspond aux valeurs extrêmes pour les derniers éléments de chaque colonne ($\{Vince, Tom\}$ dans 6.5), le deuxième élément aux valeurs extrêmes pour les

Algorithm 4: Recherche de l'indice de scission optimal

```
Input : Bucket : B
        QID Attributes : QID
1  L ← retrieve the tuples lists from B;
2  N ← number of tuples in each list;
3  minMax ← stores min and max values for each QID;
4  listReverse ← Algo1(B, QID);
5  minPerimeterSum ← ∞;
6  indexSplit ← 1
7  for i = 0 ; i ≤ N do
8      foreach list in L do
9          tuple ← list.get(N − 1 − i);
10         foreach qi in QID do
11             if tuple.qi > MinMax.qi.max then
12                 Update MinMax.qi.max;
13             if tuple.qi < MinMax.qi.min then
14                 Update minMax.QI.min;
15         reverse← listReverse.get(N - i - 1);
16         ps← perimeterSum(minMax,reverse);
17         if ps < PerimeterSum then
18             PerimeterSum← ps;
19             indexSplit ← i;
```

deux derniers éléments de chaque colonne ($\{Vince, Tom, Emily, Mary\}$ dans 6.5), etc.

L'algoritme 4 exploite cette structure lorsqu'il parcourt à son tour les sous-listes de L (*i.e.*, les colonnes) dans le sens "naturel" cette fois afin de déterminer l'indice de scission optimal pour le bucket. En effet, rappelons que le but de cette phase consiste à minimiser les périmètres des buckets qui seraient créés si B était scindé à l'indice i, le calcul des périmètres est donc une opération critique. Rappelons également que la valeur d'un périmètre est dépendante de la taille des intervalles minimums requis pour contenir les valeurs des QIDs d'un bucket. Ainsi, au moment d'évaluer la validité d'un indice de scission i, le périmètre du second bucket (destiné à contenir les N - i derniers tuples de chaque colonne) a déjà été calculé dans Algorithm 3 et peut être réutilisé rapidement

Il en résulte que chaque colonne du bucket a finalement été parcourue deux fois : une première fois dans le sens inverse et une seconde dans le sens naturel pour rechercher l'indice de scission. Dans la suite de cette section, nous expliquons en détail les algorithmes présentés dans Algorithm 3 et Algorithm 4.

On commence par parcourir les listes L dans l'ordre inverse de façon à créer une liste *reverse* permettant de répertorier les valeurs extrêmes de chaque QID : le i-ème élément de *reverse* contient donc, pour chaque QID, les valeurs maximum et minimum présentes dans le bucket s'il devait contenir les i derniers tuples des listes L, *i.e.*, les éléments $[n-i, ..., n-1]$. *Reverse* est construite de

manière incrémentielle en s'appuyant sur les valeurs extrêmes des QIDs qui ont déjà été stockées jusqu'ici. Si un tuple présente une valeur pour l'un de ses QIDs qui soit inférieure ou supérieure à l'un ou l'autre des extrêmes actuels, l'extrême en question est mis à jour.

Une fois *reverse* créé, on parcourt les listes L normalement afin de relever les valeurs extrêmes, pour chaque QID, des i-ièmes premiers tuples de chaque liste. Ce faisant, nous calculons la somme du périmètre des buckets qui seraient créés si nous devions choisir i comme indice de scission. À la fin de cette passe, on retourne la valeur de *index* pour laquelle la somme du périmètre est la plus petite.

Une fois cette étape terminée, les classes d'équivalence ont été créées, on utilise alors l'outil ARX pour les k-anonymiser individuellement, *i.e.*, nous fixons k à la taille de la classe afin d'obtenir la même généralisation pour tous les tuples.

6.6 Partitionnement des données

Après l'étape d'anonymisation, le composant Partitioner reçoit un graphe RDF, noté $\mathcal{D}$ et contenant les données d'origine (non anonymisées). Le rôle du partitionneur consiste à décomposer ce graphe en 2 graphes distincts, notés $\mathcal{D}_d$ et $\mathcal{D}_p$, respectivement les graphes par défaut et privé tels que : $\mathcal{D} \subsetneq \mathcal{D}_d \cup \mathcal{D}_p$.

Le graphe $\mathcal{D}_p$ stocke les valeurs non anonymisées des attributs EID et QID tandis que $\mathcal{D}_d$ stocke les valeurs anonymisées des QIDs. Afin de permettre la reconstruction du graphe d'origine $\mathcal{D}$ et d'éviter la perte de données vis-à-vis des requêtes privilégiées, des nœuds vides sont introduits dans les deux graphes.

En fait, aux IRIs de chaque EID dans $\mathcal{D}_p$, nous associons un nouveau nœud vide à l'aide de la propriété `owl:sameAs`. Dans $\mathcal{D}_d$, chaque EID est remplacé par le nœud vide correspondant. En conséquence, les graphes $\mathcal{D}_d$ et $\mathcal{D}_p$ peuvent être joints par l'intermédiaire de ces nœuds, rendant ainsi possible, pour les utilisateurs autorisés uniquement, d'interroger les deux graphes simultanément.

En plus de permettre la récupération des valeurs d'origine pour une entité précise de la base de données, cette approche permet d'évaluer la cohérence des données généralisées par rapport à celles d'origine. Dans des cas d'utilisation réels, nous avons découvert que cette tâche était généralement attendue par les DPPOs et les DBAs.

La figure 6.7 présente un exemple de ce partitionnement. Sur le côté gauche, le graphe d'origine (avant anonymisation) contient une IRI "http://.../patient1" correspondant à un EID, un attribut "gastrite" qui est un AS et trois QIDs, à savoir l'âge, le sexe et le code postal. Sur le côté droit de la figure, les valeurs de l'EID et des QIDs non anonymisées ne sont présentes que dans le graphe privé. L'IRI d'identification de l'entité est associée à un nœud vide _:x1, ce dernier servant de sujet à plusieurs triplets dans le graphe par défaut : celui dont l'objet est la maladie ainsi que ceux reliant l'entité aux QID anonymisés (certains chiffres du code postal y sont supprimés et l'âge généralisé grâce à un intervalle de valeurs).

De par la méthodologie employée, nous pouvons considérer Kgastor comme

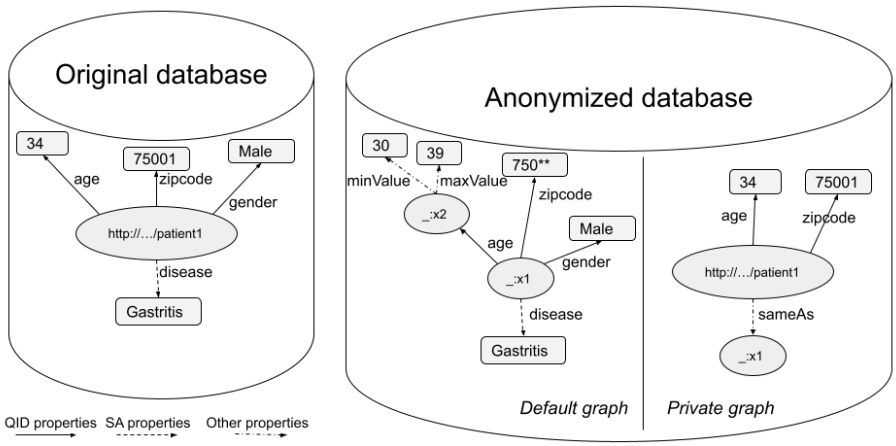


FIGURE 6.7 – Transformation de graphe dans Kgastor

un système de pseudonymisation [23] dans le sens où les données d'origine sont toujours accessibles et par conséquent, il est possible de recréer le jeu de départ à tout moment.

6.7 Traitement des requêtes

6.7.1 Requêtes du type SELECT

Tout utilisateur autorisé peut soumettre une requête à Kgastor mais, en fonction de son rôle et de la nature de la requête, seules certaines requêtes recevront une réponse. Par exemple, seul l'utilisateur autorisé à accéder au graphe privé peut interroger les deux graphes de la base de données. Tous les autres rôles sont limités aux requêtes `select` sur $\mathcal{D}_d$.

De manière à simplifier la soumission des requêtes aux utilisateurs autorisés à interroger le graphe privé, une certaine transparence au niveau du partitionnement est requise afin que ces utilisateurs n'aient pas besoin de mentionner la jointure sur les nœuds vides reliant $\mathcal{D}_d$ et $\mathcal{D}_p$. Dans le contexte de l'exemple de la Figure 6.7, un utilisateur privilégié pourrait soumettre la requête suivante :

```
SELECT ?age ?min ?max
WHERE {
GRAPH <http://private> {<http://../patient1> age ?age}
<http://../patient1> age ?b; minValue ?min;
maxValue ?max}
```

Cette requête serait alors automatiquement réécrite de la manière suivante :

```
SELECT ?age ?min ?max
WHERE {
```

```
GRAPH <http://private>
{<http://../patient1> age ?age; owl:sameAs ?x}
?x age ?b; minValue ?min; maxValue ?max}
```

Une telle réécriture est entièrement effectuée par Kgastor à l'aide du composant de contrôle d'accès permettant de déterminer l'autorisation, des métadonnées stockées par le composant Partitioner et d'une analyse de la requête soumise. Il suffit qu'un utilisateur privilégié émette une requête relative aux deux graphes nommés pour déclencher le mécanisme de réécriture. L'analyse des requêtes SPARQL repose sur le SGBD RDF sous-jacent, c'est-à-dire le module ARQ d'Apache Jena [4] dans le cas de notre prototype. En ce qui concerne d'autres systèmes avec lesquels Kgastor pourrait être employé, tels que GraphDB [5], Virtuoso [6] et Blazegraph [7], l'analyse des données est généralement déléguée à l'API de RDF4J [8].

6.7.2 Requêtes de mise à jour

Dans cette section, nous considérons les différentes requêtes (SPARQL) de mise à jour pouvant être exécutées sur notre modèle de données partitionné. La table 6.2 souligne qu'en fonction de l'opération de mise à jour mais également de sa cible (qu'il s'agisse d'EIDs, de QIDs, d'ASs ou d'ANSs), les formes de traitement peuvent être très différentes. Par exemple, une certaine opération pourrait requérir de modifier le graphe par défaut et/ou privé tandis qu'une autre pourrait nécessiter l'enregistrement d'une entrée dans le journal de mise à jour en vue d'être exécutée plus tard avec d'autres opérations (mode d'exécution sous forme de lots) afin d'anonymiser le graphe par défaut.

L'approche du journal rappelle l'approche Change Data Capture qui est utilisée pour permettre la réplication des données dans les systèmes de gestion de bases de données distribuées. Dans le cadre de notre pseudonymisation, un traitement s'appuyant sur le journal prend en charge la maintenance des graphes privés et par défaut. Le graphe privé est toujours mis à jour au moment où une transaction est soumise au système. En revanche, les opérations de mise à jour sur le graphe par défaut sont effectuées comme un lot de requêtes, à l'exception des opérations de mise à jour sur les ANSs (voir les trois lignes correspondantes dans la table 6.2) qui sont uniquement stockées dans le graphe par défaut et n'ont aucune incidence pour les classes d'équivalence générées par notre implantation de m-invariance.

Le modèle d'une entrée de journal correspond à un 5-uplet (b, nv, ov, p, t) où b correspond à un nœud vide identifiant une entité de la base de données dans $\mathcal{D}_p$, nv et ov sont respectivement les valeurs insérées et supprimées, p est la propriété de l'ontologie qui est associée à l'insertion et à la suppression de valeur

4. https://jena.apache.org/
5. https://www.ontotext.com/products/graphdb/
6. https://virtuoso.openlinksw.com/
7. https://blazegraph.com/
8. https://rdf4j.org/

Query	Private write	Default write	Logging Logging	Anonymizing Anonymizing
Insert EID	Yes	No	No	No
Insert QID	Yes	Yes	Yes	Yes
Insert SA	No	Yes	Yes	No
Insert NSA	No	Yes	Yes	No
Delete EID	Yes	Yes	Yes	Yes
Delete QID	Yes	Yes	Yes	Yes
Delete SA	No	Yes	Yes	No
Delete NSA	No	Yes	Yes	No
Update EID	Yes	No	No	No
Update QID	Yes	Yes	Yes	Yes
Update SA	No	Yes	Yes	Yes
Update NSA	No	Yes	Yes	No

TABLE 6.2 – Requêtes de mise à jour et leur impact sur le processus d'anonymisation

et t est l'horodatage de l'opération de mise à jour. Dans le cas d'une requête d'insertion (respectivement suppression), l'entrée ov (resp. nv) est affectée à une valeur nulle. Ces entrées de journal permettent de créer les requêtes de mise à jour lorsqu'il est temps de mettre à jour le graphe par défaut.

6.8 Travaux connexes

Dans cette partie, nous passons en revue un ensemble de travaux portant sur l'intégration de mécanismes d'anonymisation au sein de systèmes de base de données, le contrôle d'accès ou l'anonymisation de données RDF dans un contexte dynamique. Ces sujets étant très spécifiques à Kgastor, nous avons jugé qu'il était plus pertinent d'évoquer ces articles de recherche ici plutôt que dans le chapitre dédié à l'état de l'art.

[16] présente une solution de contrôle d'accès relativement fine (*i.e.*, au niveau des triplets) pour les graphes RDF. Ses autorisations d'accès se limitent néanmoins aux requêtes en lecture seule. Les caractéristiques de cette proposition n'ont pas été utilisées dans Kgastor car cette dernière ne prend pas en charge les requêtes de mise à jour mais également car nous avons considéré que la granularité de contrôle était trop stricte, là où un contrôle au niveau des graphes était suffisant pour notre approche partitionnée.

Plus récemment, plusieurs articles se sont intéressés à l'anonymisation dans un contexte de données relationnelles. Parmi eux, PINQ (Privacy Integrated Queries) [33] est un langage qui fournit des garanties de confidentialité par le biais du principe de confidentialité différentielle (differential privacy)[13]. Il comprend un langage de programmation déclaratif ainsi qu'un accès aux enregistrements de données via Microsoft LINQ (Language Integrated Queries).

PINQ est destiné aux grands jeux de données gérés sur Dryad, un middleware de calcul parallèle de données. PINQ correspond à une fine couche d'implémentation de confidentialité différentielle au-dessus d'un moteur de requête capable de prendre en charge de nouveaux opérateurs généralement inexistants en SQL standard.

SAP HANA Data Anonymization (DA) [25] prend en charge des capacités de traitement renforcées par la confidentialité. En son cœur, SAP HANA est un système hybride de traitement analytique des transactions fondé sur le modèle de données relationnel. L'extension DA utilise des techniques d'anonymisation telles que k-anonymity, la confidentialité différentielle locale ou l-diversité dans sa version la plus récente.

Sypse [10] est également un travail récent, et toujours en cours, qui adopte les principes de privacy-by-design ainsi que ceux de la pseudonymisation par le biais du partitionnement des données pour le modèle de données relationnel. La solution de confidentialité repose principalement sur le cryptage des données et ne considère aucune approche standard d'anonymisation. Il supporte les opérations de mise à jour sur la base de données mais n'intègre pas de mécanisme d'anonymisation permettant aux utilisateurs d'accéder aux données d'origine sans risquer de compromettre leur confidentialité.

Enfin, un certain nombre d'articles ont déjà abordé le problème de l'anonymisation dans le cadre des données dynamiques et ont pointé les nouvelles attaques rendues possibles lorsque plusieurs versions d'un jeu de données sont impliquées. Certains d'entre eux ont proposé des méthodes pour résoudre le problème dans des scénarios d'insertions uniquement, tels que Pei et al. [37] ou encore Fung et al. [17]. D'autres comme Xiao et Tao [49], Barbosa et al. [3] ou Salas et Torra [41] ont proposé des techniques dans le but de gérer à la fois les insertions et les suppressions.

6.9 Expérimentation

Les évaluations pour cette contribution ont toutes été réalisées sur la machine décrite dans la section 4.6.1.

6.9.1 Jeux de données et requêtes

Comme au chapitre précédent, nous menons nos expériences sur les jeux de données LUBM 100, 1000 et 5000 dont les informations sont répertoriées dans la table 6.3. Nous reprenons les religions introduites dans la section 2.4.2 en guise d'attributs sensibles et reprenons également deux des QIDs (les âges et les codes postaux) présentés dans la section 2.4.2.

De manière à tester l'impact de notre stratégie de partitionnement sur l'exécution de requêtes `select`, nous avons mis au point quatre requêtes `group by` et quatre requêtes `select` standards qui diffèrent par leur sélectivité et leur cardinalité. Les requêtes `group by` calculent principalement des agrégations sur les FullProfessor.

	Taille (GB)	# de triplets (en millions)	# de FullProfessors	# de professors
LUBM 100	2,3	14,24	17144	60628
LUBM 1000	23,8	141,92	170210	600356
LUBM 5000	120,2	706,07	850215	2998826

TABLE 6.3 – Caractéristiques des jeux de données LUBM

Concernant l'évaluation de l'utilité des données, nous nous concentrons de nouveau sur les requêtes de comptage en nous limitant à deux QIDs (l'âge et le code postal) et un attribut sensible correspondant à une religion. Étant donné que m-invariance est une méthode fondée sur la généralisation au même titre que k-anonymity, nous devons modifier ces requêtes de la même manière que dans le chapitre précédent afin de tenir compte du fait que les valeurs des QIDs puissent avoir été transformées. Ainsi, les âges sont requêtés selon des intervalles de 10 ans et nous recherchons des préfixes de codes postaux de taille deux (*i.e.*, les deux premiers chiffres).

Afin d'établir des comparaisons entre les données d'origines et celles anonymisées, nous réutilisons la mesure d'erreur relative présentée dans la section 4.6.2.

6.9.2 Évaluation

Shuffling et partitionnement des données

Dans cette section, nous nous intéressons au coût de notre approche de partitionnement par rapport à une solution non partitionnée. Cette dernière adopte une approche d'anonymisation standard où toutes les valeurs des QIDs ont été supprimées. Dans ce cas, le shuffling correspond à cette suppression. Comparativement, notre approche partitionnée déplace les QIDs vers un graphe privé et introduit des nœuds vides dans les deux graphes pour permettre les opérations de jointure.

La figure 6.8 présente les résultats de notre évaluation sur nos trois jeux de données. Elle met en évidence que la suppression/shuffling occupe une place prépondérante dans le coût de calcul global (qui englobe à la fois le chargement des données et leur shuffling). En outre, les coûts des approches de partitionnement et de non-partitionnement sont relativement similaires. Cela souligne le fait que le coût de la solution de partitionnement de Kgastor est négligeable par rapport à une approche simple et naïve d'anonymisation qui supprimerait des QIDs.

Requêtes du type SELECT

Ici, nous nous focalisons sur l'évaluation du surcoût éventuel de Kgastor lié au traitement des requêtes SELECT. Du point de vue de notre système, il est

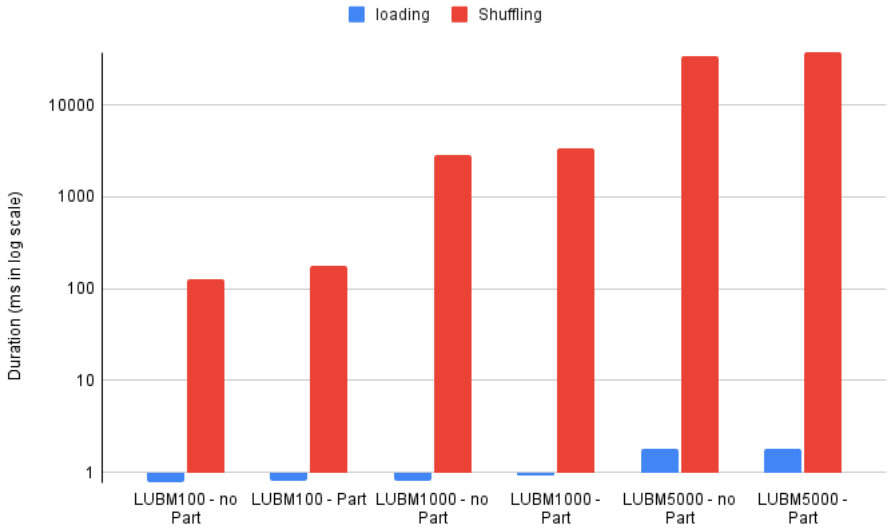


FIGURE 6.8 – Shuffling des données pour les approches sans partitionnement et partitionnement

logique de considérer trois situations :

— l'interrogation du graphe par défaut ;
— l'interrogation du graphe privé ;
— l'interrogation des deux graphes simultanément.

Dans le cadre de requêtes exprimées sur les données contenues dans le graphe par défaut, nous comparons Kgastor à une base de données similaire anonymisée mais non partitionnée. Concernant les requêtes chargées de récupérer des données à partir du graphe privé seul ou des graphes privé et par défaut, nous évaluons Kgastor par rapport à une base de données similaire non anonymisée.

La Figure 6.9 souligne que le coût de requêtage du graphe par défaut sur une implantation partitionnée ou non partitionnée est similaire pour les bases de données allant de LUBM100 à LUBM5000 et ce pour les huit requêtes. Cela souligne le fait que la réécriture des requêtes pour l'approche partitionnée est négligeable.

La figure 6.10 présente l'évolution des performances de traitement de ces dernières. Etant donné que la taille du jeu de données de LUBM 1000 est dix fois plus grande que LUBM 100 et que LUBM 5000 est cinq fois plus grande que LUBM 1000, nous pouvons considérer que le temps de traitement des requêtes est linéaire avec la taille de la base de données.

Dans la Figure 6.11, nous montrons que les performances de Kgastor sur les requêtes privées sont similaires aux performances des requêtes dans un contexte non anonymisé, voire même qu'elles peuvent être légèrement meilleures pour certaines requêtes. Nous supposons que cela est lié à l'approche d'indexation

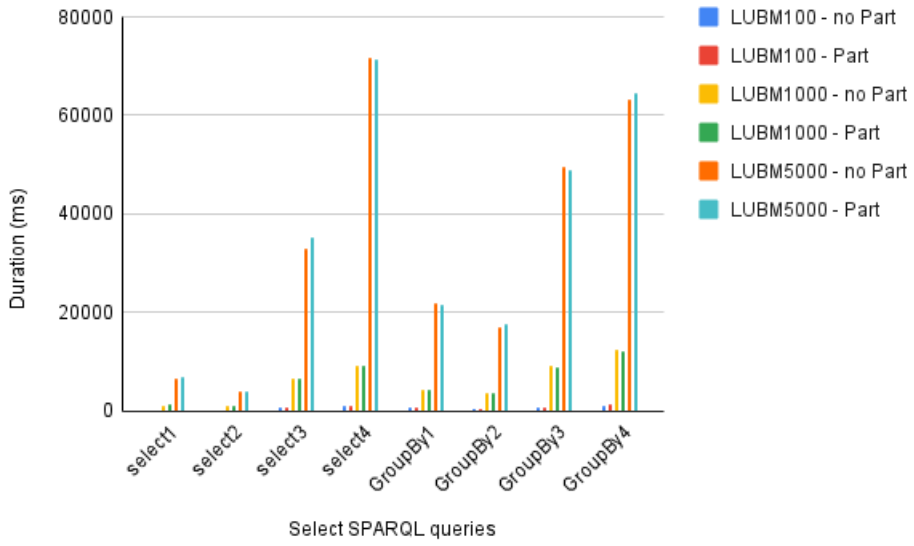


FIGURE 6.9 – Requêtes SELECT sur LUBM 100, 1000 et 5000 pour les approches partitionnée et non partitionnée

que TDB2 utilise lorsqu'il traite des graphes nommés, notamment en usant d'index partitionnés.

Performances de m-invariance

Dans cette expérimentation, nous considérons deux QIDs, à savoir un âge (80 valeurs distinctes) et un code postal (3000 valeurs distinctes), ainsi qu'un AS sous la forme d'une religion (9 valeurs distinctes), sur nos trois jeux de données. Afin de simuler un contexte dynamique, nous divisons les entités d'intérêt (rappelons qu'il s'agit des nœuds de type "Professor") de manière égale en deux ensembles. Le premier est utilisé pour créer la première publication du jeu de données tandis que le second agit comme un vivier dans lequel nous puisons des entités afin de les insérer progressivement dans notre système. Nous jouons avec trois facteurs :

i. la valeur de m qui influence la taille minimale des buckets pouvant être créés.

ii. le rythme de parution de mise à jour R qui contrôle à la fois le nombre de publications que nous voulons produire et...

iii. le volume de mise à jour (*i.e.*, le nombre d'entités supprimées et insérées) entre chaque publication.

Le volume dont nous parlons est défini comme la division entre la taille du vivier d'origine et le rythme de mise à jour. Ainsi, un R plus petit entraînera

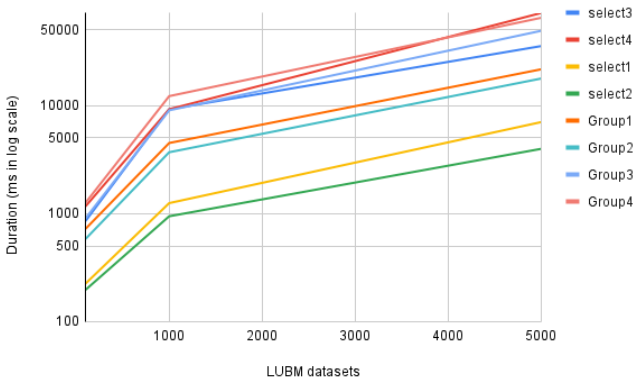


FIGURE 6.10 – Évolution des performances des requêtes SELECT sur LUBM 100, 1000 et 5000 pour l' approche partitionnée

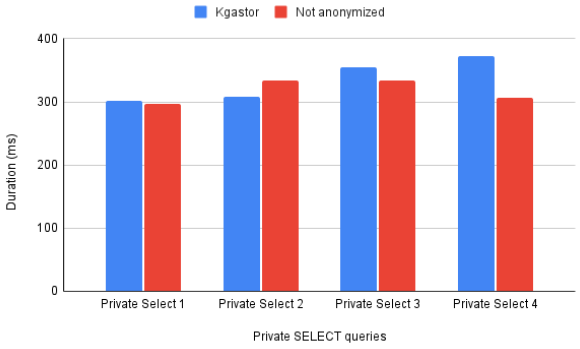
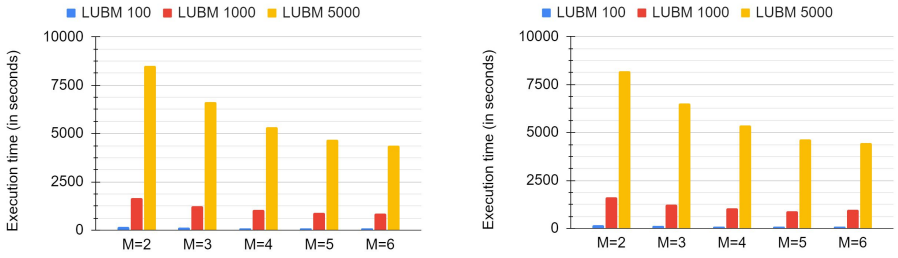


FIGURE 6.11 – Requêtes SELECT privées

un plus petit nombre de mises à jour qui concerneront un nombre plus grand de tuples. *A contrario*, un R plus grand impliquera des mises à jour fréquentes concernant beaucoup moins de tuples. Pour produire une nouvelle publication, nous sélectionnons au hasard $\frac{|vivier|}{R}$ des entités dans le jeu de données (ce qui inclut également leurs attributs) avant de les supprimer et de les remplacer par le même nombre d'entités prélevées depuis le vivier. Indépendamment de la valeur R, à la fin de notre évaluation, tous les tuples à l'intérieur du vivier auront été insérés dans le jeu de données.

Dans ce qui suit, nous nous intéressons à l'impact de m et de R sur plusieurs dimensions telles que le temps moyen nécessaire pour produire une publication (la première publication étant incluse), le temps total nécessaire pour produire toutes les $R+1$ publications ou encore le temps nécessaire pour produire seulement la première publication.



(a) Temps moyen pour une seule publication (b) Temps d'exécution pour la première publication

FIGURE 6.12 – Temps d'exécution : valeurs variables pour m - R = 10

Notez que nous traitons la première version différemment des autres car celle-ci a la particularité de ne dépendre d'aucune version antérieure mais également de prendre en charge l'insertion de beaucoup plus de tuples que toute version ultérieure (la moitié des entités du jeu de données d'origine pour être précis).

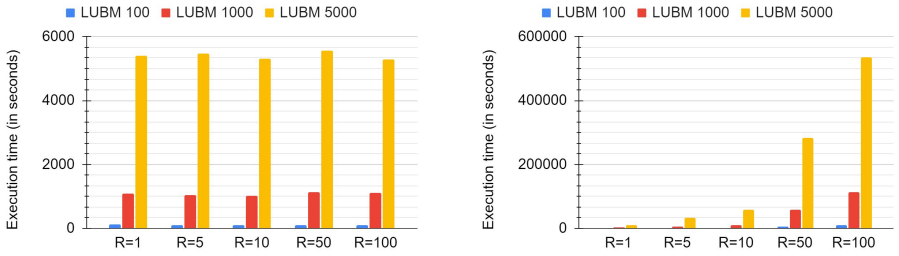
Commençons par énoncer une évidence : plus il y a d'entités dans un jeu de données, plus il faut de temps pour l'anonymiser. Ainsi, comme le montre la Figure 6.12, dans le pire des cas, il faut moins de 200 secondes pour produire une version pour LUBM 100, moins de 1200 secondes pour LUBM 1000 et environ 8500 secondes pour LUBM 5000.

Évolution de *m*. Pour cette expérience, nous fixons le nombre de mise à jour à 10 *i.e.*, 11 versions au total si on compte la première. Comme le montre la Figure 6.12a, à mesure que la valeur de m augmente, le temps moyen nécessaire pour calculer les nouvelles versions diminue considérablement. Il faut environ deux fois plus de temps pour anonymiser un ensemble de données avec $m = 2$ qu'avec $m = 6$. Si l'on prend comme exemple LUBM 5000, le temps d'anonymisation passe de 8500 secondes à environ 4300 secondes. Ce résultat est corroboré par la Figure 6.12b et s'explique par le fait que plus la valeur de m est grande, plus le nombre de tuples contenus dans les buckets est important. En définitive, on a moins de buckets et par conséquent moins d'opérations du type *split* à exécuter (sachant que cette phase est l'une des plus coûteuses, comme nous le verrons juste après).

Évolution du nombre de publications *R* - $m = 4$.

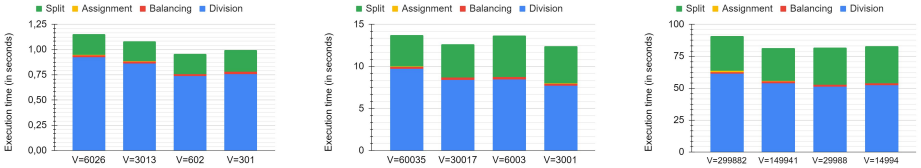
En revanche, sur la Figure 6.13 (où m est fixé à 4) on remarque que le nombre de publications, et par conséquent la taille du volume de mise à jour, n'a pas d'impact significatif sur le temps moyen nécessaire pour produire une nouvelle version, c'est-à-dire qu'indépendamment du volume de mise à jour, il faut à peu près le même temps pour que l'algorithme se termine. Il faut un peu plus de 100 secondes pour produire une version de LUBM 100, plus de 1000 secondes pour LUBM 1000 et plus de 5000 secondes pour LUBM 5000.

En réalité, la majeure partie du temps de calcul est consacrée à l'exécution des requêtes qui mettent à jour les attributs des entités dans les deux jeux



(a) Temps moyen pour une seule publication(b) Temps total pour toutes les publications

FIGURE 6.13 – Temps d'exécution : nombre de mises à jour variables



(a) LUBM 100 (b) LUBM 1000 (c) LUBM 5000

FIGURE 6.14 – Temps d'exécution : volumes de mise à jour variables (V) - m = 4 - Trois publications

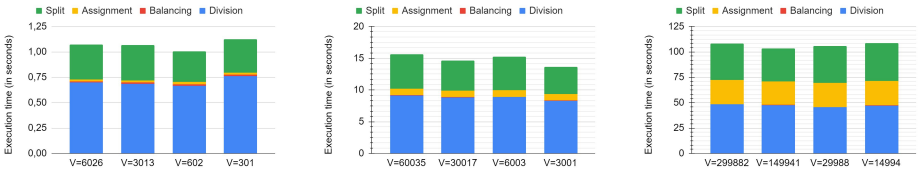
de données (privé et par défaut). Nous y reviendrons plus en détail dans la prochaine évaluation.

Du fait du protocole expérimental que nous suivons, à savoir supprimer et ajouter le même nombre d'entités entre chaque publication, l'algorithme s'applique toujours sur le même nombre total d'entités dans le jeu de données, ce qui explique pourquoi les temps moyens restent constants quelle que soit la valeur de R.

Évolution du volume de mise à jour V.

Comme nous l'avons dit, la majeure partie du temps de calcul est dédiée à l'exécution des requêtes de mise à jour sur nos graphes. Par conséquent, il est difficile d'évaluer avec précision l'impact du volume de mise à jour sur le comportement de notre implantation de m-invariance. Pour résoudre ce problème, nous avons observé le temps d'exécution de chacune des quatre phases de l'algorithme et avons calculé leurs moyennes sur un ensemble de trois versions, *i.e.*, nous avons fixé le volume de mise à jour à une certaine valeur (nous nous appuyons sur notre évaluation précédente pour choisir les volumes) et nous produisons trois versions pour un jeu de données donné. Les résultats sont présentés dans la Figure 6.14.

Tout d'abord, nous remarquons à quel point le temps alloué à l'exécution de m-invariance est en réalité insignifiant par rapport au temps total nécessaire



(a) LUBM 100 (b) LUBM 1000 (c) LUBM 5000

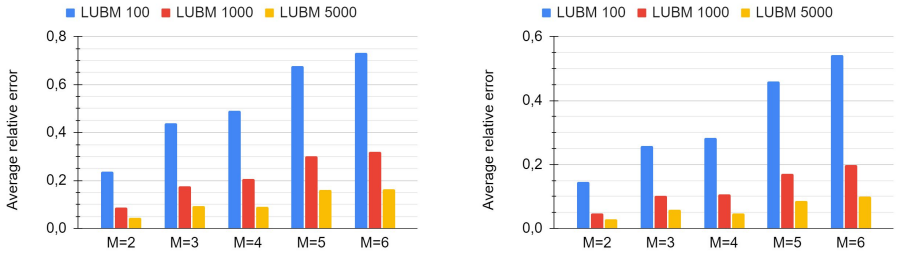
FIGURE 6.15 – Temps d'exécution pour la publication initiale : volumes de mise à jour variables (valeur V) - $m = 4$ - Trois publications

pour produire une version. Pour LUBM 5000, l'algorithme prend moins de 100 secondes à finir sur un total d'environ 5000 secondes pour toute l'exécution.

Deuxièmement, nous observons, pour tous les jeux de données, que le volume de mise à jour a peu d'impact sur la durée globale de l'algorithme : quel que soit le volume, il faut environ 1 seconde pour que l'algorithme se termine sur LUBM 100, 13 secondes sur LUBM 1000 et un peu plus de 80 secondes sur LUBM 5000. *Division* et *split* sont les étapes les plus coûteuses alors que les étapes *balancing* et *assignment* sont relativement rapides voire négligeables.

La phase *Division* est coûteuse car elle doit commencer par interroger le graphe pour récupérer tous les tuples afin de déterminer lesquels d'entre eux étaient déjà présents dans la version précédente. Nous insistons encore une fois sur le fait qu'interroger le graphe est plus coûteux que l'exécution de l'algorithme en lui-même. En ce qui concerne la phase *split*, son objectif est de trouver un schéma de découpage efficace afin de séparer rapidement toutes les entités du jeu de données en classes d'équivalence efficaces. Cependant, en raison encore une fois du protocole, le nombre d'entités est constant d'une version à l'autre quel que soit le volume de mise à jour. Finalement, on observe que les performances des deux étapes sont fortement dépendantes de la taille du graphe ou du nombre d'entités qu'il contient, c'est pourquoi elles peuvent être plus longues et que l'on observe ces temps constants car la taille du jeu de données n'évolue pas au fil des publications.

Néanmoins, l'algorithme se comporte un peu différemment en ce qui concerne la publication initiale, comme le montre la Figure 6.15. En effet, nous remarquons que la phase *assignment* est beaucoup plus coûteuse, en particulier sur des jeux de données plus volumineux tels que LUBM 5000 où son coût devient même comparable à celui de *split* (environ 25 secondes). Pour comprendre cela, nous devons nous rappeler comment la version initiale est produite : nous prenons toutes les entités du jeu de données d'origine et en utilisons la moitié comme base pour la première version. Ainsi, la quantité de mises à jour est beaucoup plus élevée que pour toute autre version. De plus, comme il n'existe aucun bucket à ce stade, la phase *balancing* n'est pas en mesure de traiter les entités nouvellement insérées dans le système, la responsabilité de créer les premiers buckets et d'y ranger les entités revient donc entièrement à *assignment*.



(a) Erreur relative moyenne pour une seule publication

(b) Erreur relative moyenne pour la première publication

FIGURE 6.16 – Utilité des données : valeurs variables de m - R = 10

Utilité des données anonymisées

Dans cette partie, les codes postaux que nous utilisons sont composés de 3000 valeurs continues entre 80000 et 83000, les âges sont compris entre 20 et 100 ans et nous comptons toujours neuf religions. Les valeurs des trois attributs sont réparties uniformément entre chaque nœud de professeur dans chaque jeu de données LUBM.

Évolution de m. Nous observons sur la Figure 6.16 que plus la valeur de m augmente, moins les données deviennent précises. En effet, l'algorithme se retrouve obligé de former des classes d'équivalence plus grandes et donc de regrouper des tuples dont les QIDq sont peut-être trop éloignés, engendrant ainsi un plus haut niveau de généralisation. Sur LUBM 5000 par exemple, alors que l'erreur moyenne avec $m = 2$ est d'environ 5% pour toutes les versions, elle monte à 16% lorsque $m = 6$. Notez également que l'erreur relative moyenne est généralement plus faible pour la version initiale (Figure 6.16) que pour les versions ultérieures. Ce point sera développé dans l'expérimentation suivante.

Évolution de R.

Par rapport au rythme publication, nous remarquons sur la Figure 6.17 une diminution de l'utilité des données lorsque la stratégie de mise à jour implique beaucoup de versions, autrement dit moins de modifications entre les publications. Avec moins d'entités nouvellement insérées, l'algorithme se retrouve avec de possibilités pour créer des buckets ou compléter ceux qui existent déjà. De plus, en produisant plus de publications, le système court un risque plus élevé de créer des buckets contenant peu d'entités en fin de compte. Moins il y a de données avec lesquelles travailler, plus la possibilité de devoir appliquer une généralisation lourde est élevée. Encore une fois, si nous prenons LUBM 5000 comme exemple, l'erreur relative moyenne moyenne pour 5 publications est d'environ 7% alors qu'elle est d'un peu plus de 15% pour une centaine de publications.

Notez qu'à la fin d'une évaluation, quelle que soit la valeur de R, le même nombre de suppressions/insertions a été effectué au total. Ainsi, nous voyons

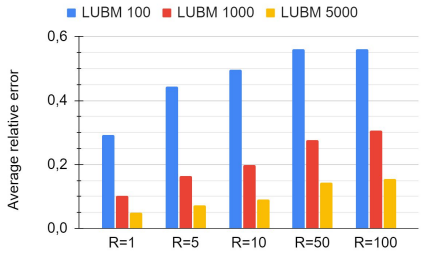


FIGURE 6.17 – Utilité des données : valeurs variables de R - $m = 4$

qu'il est plus efficace d'opter pour une implémentation par lots de notre système, c'est-à-dire de prioriser un faible nombre de versions mais chacune comptant une grande quantité de modifications. En contrepartie, cela a un impact négatif sur la fraîcheur des données dans le graphe par défaut, les mises à jour étant plus éparses. Cependant, un utilisateur privilégié a toujours la possibilité d'utiliser le graphe privé afin d'obtenir des analyses à jour. Du reste, raisonner en termes de lots garantit en fin de compte une plus grande utilité pour le jeu de données. C'est donc un choix plus intéressant sur le long terme.

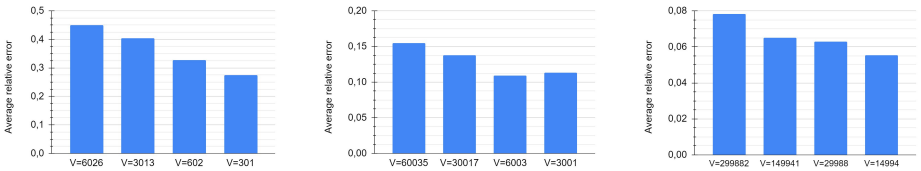
Évolution de V. Nous observons que de grands volumes de mises à jour ont tendance à réduire davantage l'utilité des données du jeu qu'un plus petit nombre de mises à jour. Nous pouvions nous y attendre car moins de mises à jour affectent moins de buckets/classes d'équivalence et ont donc peu d'impact sur l'utilité du jeu de données dans sa globalité.

Comme le montre la Figure 6.18, bien que les écarts d'erreur relative soient beaucoup plus perceptibles pour LUBM 100 (l'erreur est d'environ 45% pour le plus gros volume contre 27% pour le plus petit), ils deviennent plus négligeables sur des jeux de données plus importants tels que LUBM 1000 et 5000 où la différence d'erreur relative entre le volume le plus élevé et le plus petit est inférieure à 5%.

Néanmoins, nous avons vu précédemment que sur une longue période de temps, pour un nombre fixe de mises à jour, il est préférable d'utiliser des stratégies de type batch pour traiter les mises à jour afin de limiter la perte d'informations. Il est en général préférable de traiter des lots de grandes tailles avec une faible fréquence que de publier de plus petits lots avec une fréquence plus élevée.

6.10 Conclusion

À notre connaissance, Kgastor est le premier framework RDF qui adopte les principes de privacy-by-design. Le système est fondé sur le partitionnement des données et prend en charge les opérations de mise à jour de deux manières distinctes : au moment de la transaction pour le graphe privé et avec un délai configurable pour le graphe par défaut. Concernant les mises à jour et l'anony-



(a) LUBM 100 (b) LUBM 1000 (c) LUBM 5000

FIGURE 6.18 – Utilité des données : volumes de mise à jour variables - $m = 4$ - Trois publications

misation, une des approches de l'état de l'art, à savoir m-invariance qui était à l'origine conçue pour le modèle de données relationnel, a été adaptée au modèle de données RDF. Rappelons que le système se veut être "indépendant" vis-à-vis de la technique employée du moment que celle-ci garantit la confidentialité des entités présentes dans le jeu de données dans un contexte dynamique. Finalement, Kgastor propose une solution de pseudonymisation en offrant un compromis précieux entre confidentialité, utilité et fraîcheur des données accessibles.

Parmi les évolutions potentielles de ce système, nous aimerions tourner notre attention vers les services de raisonnement dans le contexte de Kgastor, tout d'abord en commençant par quelques extensions RDFS (par exemple `owl:sameAs`) puis en considérant plus tard OWLRL. Une autre piste de recherche intéressante concernerait les mécanismes mis en place pour déclencher la publication d'une nouvelle version : doit-elle être périodique (une version toutes les quelques semaines ou tous les mois ?) ou doit-elle être publiée dès que le lot de mises à jour le permet ? Le premier choix, qui implique des mises à jours plus lourdes, permettrait aux utilisateurs de bénéficier de données moins généralisées et par extension plus utiles. Le second en revanche permettrait de rendre accessible des données plus récentes mais peut-être peut-être au prix d'une certaine utilité des données, car elle impliquerait plus de généralisations. Le compromis entre les deux reste à ce jour une question ouverte.

Chapitre 7

Conclusion

7.1 Synthèse des contributions

Dans cette section, nous revenons sur les apports majeurs de chacune de nos contributions.

7.1.1 Anatomie sémantique

Une technique d'anonymisation adaptée au modèle RDF

Comme nous l'avons déjà expliqué, la majorité de la littérature se concentre autour de l'anonymisation de données relationnelles, les travaux sur les graphes de connaissances ne représentant quant à eux qu'une infime minorité. Ainsi, nous avons développé une adaptation d'anatomie dont le principe consiste à supprimer les liens directs entre les entités et leurs attributs sensibles en faisant usage de clés étrangères afin de créer des groupes d'attributs. Dans notre implantation, cela est réalisé par l'insertion de nœuds vides intermédiaires.

Incorporation d'aspects sémantiques dans le processus d'anonymisation

Un point surprenant que nous avions remarqué dans les travaux traitant de l'anonymisation pour les graphes de connaissances était leur manque de considération pour les aspects sémantiques pourtant inhérents à ces structures. Par l'utilisation de services de raisonnement simples tels que la subsomption de concepts, nous avons étendu l'approche de base au niveau de la création des groupes en faisant en sorte que ces derniers soient cohérents d'un point de vue sémantique. Plus précisément, plutôt que de rassembler arbitrairement des attributs, notre approche regroupe ceux étant les plus proches du point de vue de sémantique, ce qui implique nécessairement un moyen de les comparer.

Pour cela, nous utilisons une mesure nommée *taxonomy similarity* (elle-même se basant sur deux autres notions, à savoir l'*upward cotopy* et le *concept match*) qui, étant donné deux concepts et une ontologie, est capable de calculer une distance entre ces concepts en fonction de leur position respective dans la hiérarchie.

Afin de regrouper les attributs similaires, nous avons développé un algorithme de type bottom-up dans lequel nous partons d'une situation de départ où chaque attribut correspond à un groupe individuel. Au fur et à mesure de l'exécution de cet algorithme, la taxonomy similarity est employée afin de déterminer les groupes les plus proches (au sens de cette mesure) de manière à les fusionner. Néanmoins, la mesure ayant été définie pour s'appliquer sur un couple de concepts, un problème se pose puisque nous souhaitons comparer des groupes potentiellement composés de plusieurs attributs. Pour remédier à cela, nous avons choisi de faire appel à un autre service de raisonnement, le *least common ancestor* (LCA), nous permettant de déterminer le super-concept le plus précis d'un ensemble de concepts donné. Finalement, chaque groupe est ainsi représenté par son LCA et c'est sur cette base que la taxonomy similarity s'appuie pour les comparer avant de les fusionner.

7.1.2 Utilisation combinée de l'anatomie sémantique et de k-anonymity

Combler les faiblesses des deux approches

Le fait d'appliquer deux méthodes d'anonymisation sur un même jeu de données peut paraître contre-intuitif de prime abord car cela occasionne d'autant plus de pertes d'information et conduit par conséquent à une détérioration accrue de l'utilité des données. En dépit de ce constat, notre intuition est que cette utilisation simultanée peut également servir à répondre à certaines faiblesses de chacune des techniques.

Rappelons que l'anatomie et k-anonymity ont des objectifs diamétralement opposés : la première effectue des changements au niveau des attributs sensibles et laissent intacts les quasi-identifiants, la seconde fait l'inverse. De fait, elles s'exposent chacune à un type spécifique d'attaques tout en se protégeant face à un autre. Parmi ces attaques, nous en distinguons deux, à savoir la divulgation

d'entités et la divulgation d'attributs. Nous expliquons de quelle façon elles peuvent être utilisées à l'encontre de l'une ou l'autre des deux approches mais aussi dans quelle mesure elles peuvent être contrecarrées.

La proposition de deux algorithmes pour l'application de k-anonymity sur des graphes RDF

Dans le cadre de ce travail, nous avons repris l'anatomie sémantique introduite précédemment et développé une adaptation de k-anonymity pour les graphes de connaissances. Pour cette adaptation, nous proposons deux algorithmes dont le fonctionnement diffère sur la manière dont sont créées les classes d'équivalence. Le premier algorithme, que nous avons nommé "global", s'applique, comme la méthode classique, à l'ensemble des entités présentes dans le graphe. Le second, nommé "par groupe", présuppose que le jeu de données a déjà été préalablement anonymisé via l'anatomie et applique k-anonymity de manière indépendante sur chacun des groupes définis par l'anatomie. Plus précisément, là où "global" ne s'exécute qu'une seule fois en considérant toutes les entités à la fois, "par groupe" ne considère que les entités au sein d'un groupe donné et s'exécute successivement sur tous les groupes dans le graphe.

Plusieurs évaluations ont été menées afin de comparer les deux algorithmes vis-à-vis de leur impact sur l'utilité des données et il en est ressorti que la version globale était bien plus performante à cet égard car elle requiert moins de généralisation pour créer les classes d'équivalence. Cela s'explique notamment par le fait que plus le nombre d'entités à disposition de k-anonymity est élevé, plus elle a de possibilités pour les regrouper et produire des classes d'équivalence où les valeurs des quasi-identifiants sont similaires et nécessitent moins de transformations. En se limitant à un sous-ensemble du graphe, l'algorithme par groupe se retrouve quant à lui contraint de généraliser à outrance les données.

7.1.3 Kgastor : un système de gestion de graphe de connaissances anonymisé

Un framework RDF suivant les principes du privacy-by-design

La mise en place d'une stratégie d'anonymisation des données a posteriori est une tâche généralement très lourde pour une organisation. Sans compter les connaissances théoriques essentielles dans ce domaine, il s'avère le plus souvent nécessaire d'opérer une refonte plus ou moins importante de la manière dont sont gérées les données en interne.

En réponse à la fébrilité de ces organisations à s'attaquer au problème, les défenseurs de l'approche du *privacy by design* défendent l'idée que l'anonymisation devrait être déléguée à des systèmes dédiés, disposant d'outils prévus à cet effet. En outre, ils soutiennent que cela permettrait de faciliter l'adoption de bonnes pratiques de gestion des données à grande échelle.

À notre connaissance, Kgastor est le seul système de ce type conçu pour le modèle de données RDF. Par l'utilisation d'une stratégie de partitionnement as-

sociée à une solution de contrôle d'accès ainsi qu'une approche d'anonymisation adaptée au contexte dynamique, nous sommes en mesure de gérer un graphe de connaissances tout en protégeant les données personnelles des entités qu'il contient.

Le partitionnement de données privées et publiques

Nous établissons une distinction entre d'un côté des utilisateurs privilégiés (administrateurs, etc), et de l'autre les utilisateurs ordinaires (développeurs d'applications). Étant donné que le premier type d'utilisateur est généralement chargé de la gestion de la base données, nous considérons qu'il est nécessaire pour ces derniers d'avoir accès à des données non-anonymisées.

Dans cette optique, notre système partitionne le jeu de données d'origine en deux sous-graphes : un premier (privé) destiné aux utilisateurs privilégiés et un second (public) accessible par l'ensemble des utilisateurs et contenant les données anonymisées. Le lien entre les graphes est par la suite assuré grâce à la présence de nœud vides.

Un contrôle d'accès transparent permis par la réécriture de requêtes

De façon à rendre notre stratégie de partitionnement effective, notre système se doit de pouvoir refuser l'accès du graphe privé aux utilisateurs qui ne posséderaient pas l'un des rôles adéquats. Pour ce faire, nous avons développé une solution de réécriture de requête permettant, en fonction des autorisations d'un utilisateur, de modifier une requête soumise au système afin que celle-ci soit exécutée sur le bon graphe. De même, nous proposons de réécrire les requêtes s'appliquant aux deux graphes qu'un utilisateur privilégié pourrait soumettre afin qu'il ne soit pas nécessaire de constamment mentionner la jointure sur les nœuds vides qui assure la liaison. Dans l'intention de ne pas alourdir l'écriture des requêtes pour les utilisateurs, ces procédures sont réalisées de manière tout à fait transparente par le système.

Une solution d'anonymisation pour le modèle RDF dans un contexte dynamique

Comme nous l'avons déjà mentionné à plusieurs reprises, la mise à jour et publication répétée d'un jeu de données permet d'ouvrir la voie à une pléthore de nouvelles attaques auxquelles les techniques d'anonymisation classiques ne peuvent tout simplement pas répondre.

Pour notre propre stratégie d'anonymisation, nous avons adapté m-invariance, une approche conçue à l'origine pour le modèle relationnel et capable de gérer des insertions ainsi que des suppressions. Un point important à noter est que l'anonymisation d'un jeu de données (ici en l'occurrence, celle du sous-graphe par défaut) ne peut pas forcément s'effectuer à la moindre modification. Certaines méthodes peuvent en effet requérir qu'un certain nombre de conditions soient remplies avant de permettre la publication d'une nouvelle version, ce qui est le cas par exemple de m-invariance.

Plus généralement, le fait de disposer d'une plus grande quantité de données est généralement bénéfique car cela offre plus de liberté dans la manière dont elles peuvent être généralisées et permet de minimiser la perte d'informations. Nos évaluations nous ont notamment montré qu'il était plus judicieux, dans le cadre de notre système, de privilégier un nombre restreint de publications avec un volume de mises à jour élevé entre chacune d'entre elles afin de préserver l'utilité des données.

Dans notre système cependant, les deux sous-graphes gagnent à fonctionner de manière asynchrone, en suivant des temporalités différentes. Tandis que le premier exige qu'idéalement les données soient à jour en permanence, le second gagne plutôt à attendre un certain laps de temps avant d'être actualisé. Pour permettre cela, nous avons développé un journal chargé de garder une trace de toutes les modifications effectuées entre chaque mise-à-jour du graphe public afin que celles-ci puissent être facilement réutilisées par le composant d'anonymisation lorsqu'il convient de publier une nouvelle version.

Le graphe privé est quant à lui actualisé dès qu'une requête est soumise au système (à condition que l'utilisateur y soit autorisé bien entendu).

Un système indépendant de l'approche d'anonymisation

Bien que nous utilisions m-invariance pour anonymiser le graphe, nous tenons à insister sur le fait que notre système n'est pas strictement couplé à cette seule approche et offre, au contraire, une certaine liberté au niveau de l'anonymisation. Du moment qu'une technique est capable de traiter des données dans un contexte dynamique, celle-ci peut effectivement remplacer m-invariance. Si certains ajustements peuvent bien entendu être nécessaires, notamment au niveau de la gestion du journal, cela ne concernerait que le composant chargé de l'anonymisation mais n'affecterait en rien le fonctionnement du reste de notre système (*e.g.*, le partitionnement, la réécriture de requête, etc).

7.2 Pistes de recherche pour l'avenir

Dans cette dernière partie, nous tentons de prendre de la hauteur sur ces trois années de recherche : nous revenons sur certaines des questions laissées en suspens dans nos contributions et abordons des perspectives de recherche que nous n'avons pas abordées mais que nous considérons importantes.

7.2.1 L'impact du raisonnement sur l'anonymisation

Les aspects sémantiques ainsi que le raisonnement sont des concepts propres aux graphes de connaissances qui ont jusqu'à maintenant été relativement négligés par les travaux portant sur l'anonymisation. En ce sens, nous pensons qu'il est primordial de leur accorder davantage d'attention tant ils peuvent avoir un impact, aussi positif que négatif, sur ces processus.

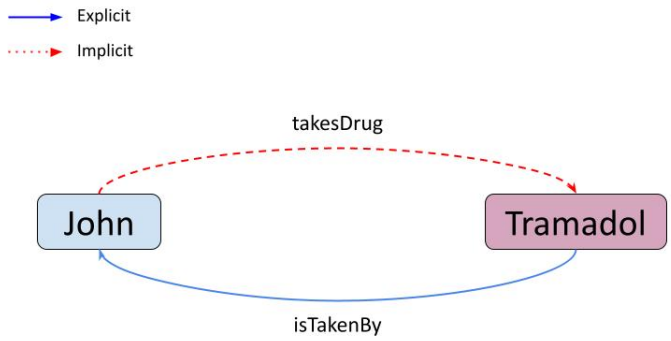


FIGURE 7.1 – Exemple d'une interaction avec des propriétés inverses

Un volet positif

À l'instar de notre article [44] dans lequel nous abordions le sujet en nous limitant à des aspects sémantiques simples, il pourrait être intéressant d'incorporer plus de ces services de raisonnement au sein de différentes approches afin de les rendre potentiellement plus performantes. L'un des enjeux principaux de nombreuses solutions d'anonymisation consistant très souvent à regrouper des données préféremment similaires, le raisonnement peut notamment aider dans ces situations à comparer des entités, des attributs, etc.

Un volet négatif

À contrario, le raisonnement peut également être utilisé par un attaquant afin de découvrir des données personnelles. Pour utiliser une technique d'anonymisation, l'utilisateur doit préalablement désigner des cibles à anonymiser : les quasi-identifiants des entités, leur attribut sensible, etc. Un problème se pose lorsque des données supprimées, généralisées ou tout du moins masquées peuvent être générer de nouveau par inférence sur la base d'autres triplets.

Ainsi, l'interaction entre le système d'anonymisation et le moteur d'inférence devient un point crucial sur l'efficacité de l'approche de protection des données personnelles. Nous présentons dans la figure 7.1 un extrait de jeu de données médical décrivant la prise d'un médicament par un individu. Nous comptons deux propriétés inverses, *takesDrug* et *isTakenBy*, et considérons que les triplets impliquant *takesDrug* sont problématiques. Ainsi, dans le cas où il existe un triplet (X, *isTakenBy*, Y) explicite, notre méthode se retrouverait inefficace car la relation sensible pourrait toujours être inférée. Il convient donc d'aborder les mécanismes de matérialisation de triplets implicites lors de la définition des QIDs et ASs.

Cette démonstration pourrait être étendue en considérant des ontologies plus expressives, *e.g.*, exprimées avec des langages tels que OWLRL qui exploitent

des règles de type Datalog. Quoi qu'il en soit, nous estimons que la communauté manque clairement de recul sur l'étendue de l'impact potentiel du raisonnement sur des approches d'anonymisation et sur les manières dont ils peuvent être utilisés à leur encontre.

7.2.2 Déclenchement d'une nouvelle mise à jour dans Kgastor

Dans notre système Kgastor, les données sont partitionnées en deux sous-graphes évoluant de façon asynchrone. De son côté, le graphe privé est réservé aux utilisateurs privilégiés qui ont intérêt à ce que les mises à jour prennent effet immédiatement afin que les données soient fraîches.

En revanche, le second graphe a vocation à être accessible publiquement, c'est pourquoi il doit être anonymisé. Or, nous avons rappelé à plusieurs reprises au cours de ce document que les approches d'anonymisation gagnent généralement à être appliquées sur de plus grands jeux de données car cela leur offre plus de liberté au niveau des transformations qu'elles peuvent réaliser et leur permet ainsi de limiter la perte d'informations. Nous avons notamment montré dans la section 6.9.2 que notre adaptation de m-invariance était moins performante sur le long terme du point de vue de l'utilité des données lorsqu'elle devait fréquemment traiter des petits volumes de mise à jour.

Pour ces raisons, nous défendons l'idée que, contrairement au graphe privée, les mises à jour de ce graphe ne devraient pas être immédiates (*i.e.*, dès qu'une requête est soumise au système) mais plutôt espacées, après qu'une certaine quantité de modifications aient été effectuées. Il convient maintenant de discuter des mécanismes qu'il serait possible de mettre en place pour permettre ce type de fonctionnement.

Une première proposition pourrait être de rendre ces mises à jour périodiques. Ce choix impliquerait potentiellement un volume de mise à jour plus lourd et permettrait aux utilisateurs de bénéficier de données moins généralisées et donc, plus utiles. La difficulté ici consisterait à déterminer un rythme de publication du graphe public adapté au rythme des modifications apportées au jeu de données (*e.g.*, un jeu où le volume de modifications est élevé pourrait justifier une mise à jour hebdomadaire).

La seconde alternative que nous proposons serait de déclencher la publication des données anonymisées dès que le lot de mises à jour le permettrait. Cette méthode aurait pour avantage de rendre accessible plus rapidement des données mais pourrait nécessiter plus de généralisation et donc une plus grande perte d'information.

Nous présentons ces deux solutions séparément mais il serait intéressant d'imaginer une solution hybride impliquant un mélange des deux approches. Dans le cas présent, nous laissons ouverte la question du compromis entre d'un côté l'utilité des données et de l'autre, sa disponibilité.

7.2.3 Intégration de l'anonymisation au niveau du *Edge*

L'émergence fulgurante de tout un ensemble d'appareils autonomes (*IoT devices*) et l'explosion de la quantité de données produites par ces terminaux a fortement remis en cause l'hégémonie du modèle de traitement centralisé traditionnel (*e.g.*, *cloud computing*).

L'une des réponses proposées ces dernières années pour faire face à ce défi est l'*Edge Computing*, un paradigme selon lequel le calcul et le stockage des données devraient être effectués, du moins en partie, directement sur ces appareils en périphérie (*edge* en anglais) du réseau, là où les données sont produites.

Cependant, un certain nombre de ces capteurs collectent des données que l'on pourrait considérer sensibles. L'exemple le plus actuel aujourd'hui serait sans doute la position (et par extension les déplacements) d'une personne sur une période donnée notamment par le biais d'applications munies de fonctions de géolocalisation. Dans ce genre de scénario, des mesures doivent donc être mises en place pour anonymiser/masquer ce type d'information, la difficulté étant qu'il faut composer avec des machines aux ressources relativement limitées en ce qui concerne la puissance de calcul, la mémoire voire l'autonomie.

Plusieurs travaux ont déjà abordé ces questions. C'est le cas par exemple de [51] qui fournit une analyse globale des risques et des défis relatifs à la sécurité des données dans le contexte précis du *Edge*. L'article présente par la même les technologies en vigueur fondées sur la cryptographie pour protéger les données et dresse la liste des solutions de l'état de l'art.

Enfin nous souhaitons terminer en mentionnant les passerelles qui peuvent être établies entre l'*Edge Computing* et le Web sémantique. En effet, il pourrait être intéressant pour ces appareils situés à la périphérie du réseau d'être capables d'interpréter les données et de prendre des décisions autonomes en conséquence sans avoir besoin de faire remonter les données dans le réseau. Malgré cela, cette interface entre les deux domaines n'a pour l'instant pas beaucoup été étudiée : on pourra citer [50] qui présente un système de gestion de base de données RDF spécifiquement conçu pour de petits appareils. À notre connaissance, l'anonymisation dans ce type de contexte reste toutefois un sujet inexploré.

Bibliographie

[1] Gagan Aggarwal, Tomás Feder, Krishnaram Kenthapadi, Rajeev Motwani, Rina Panigrahy, Dilys Thomas, and An Zhu. Anonymizing tables. In ICDT, 2005.

[2] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. The Description Logic Handbook : Theory, Implementation, and Applications. Cambridge University Press, 2003.

[3] Manuel Barbosa, Alexandre Pinto, and Bruno Gomes. Generically extending anonymization algorithms to deal with successive queries. In Xuewen Chen, Guy Lebanon, Haixun Wang, and Mohammed J. Zaki, editors, 21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012, pages 1362–1371. ACM, 2012.

[4] R.J. Bayardo and Rakesh Agrawal. Data privacy through optimal k-anonymization. In 21st International Conference on Data Engineering (ICDE'05), pages 217–228, 2005.

[5] Ann Cavoukian. Privacy by design, the 7 foundational principles. https ://-bit.ly/3GnOxlT, 2011.

[6] Bernardo Cuenca Grau and Egor V. Kostylev. Logical foundations of privacy-preserving publishing of linked data. In Dale Schuurmans and Michael P. Wellman, editors, AAAI, pages 943–949, 2016.

[7] Tore Dalenius. Finding a needle in a haystack or identifying anonymous census records. Journal of official statistics, 2(3) :329, 1986.

[8] Remy Delanaux, Angela Bonifati, Marie-Christine Rousset, and Romuald Thion. Query-based linked data anonymization. In International Semantic Web Conference (1), volume 11136 of Lecture Notes in Computer Science, pages 530–546, 2018.

[9] Remy Delanaux, Angela Bonifati, Marie-Christine Rousset, and Romuald Thion. RDF graph anonymization robust to data linkage. In WISE 2019, pages 491–506, 2019.

[10] Amol Deshpande. Sypse : Privacy-first data management through pseudonymization and partitioning. In CIDR 2021. www.cidrdb.org, 2021.

[11] Darcy DiNucci. Fragmented future. Print, 53(4) :32, 1999.

[12] Josep Domingo-Ferrer, David Sánchez, and Jordi Soria-Comas. Database Anonymization : Privacy Models, Data Utility, and Microaggregation-Based Inter-Model Connections. Morgan amp ; Claypool Publishers, 2016.

[13] Cynthia Dwork. Differential privacy. In Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II, pages 1–12, 2006.

[14] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Proceedings of the Third Conference on Theory of Cryptography, TCC'06, page 265–284, Berlin, Heidelberg, 2006. Springer-Verlag.

[15] Khaled Emam and Fida Dankar. Protecting privacy using k-anonymity. Journal of the American Medical Informatics Association : JAMIA, 15 :627–37, 07 2008.

[16] Giorgos Flouris, Irini Fundulaki, Maria Michou, and Grigoris Antoniou. Controlling access to RDF graphs. In Future Internet Conference on Future Internet, FIS'10, page 107–117, Berlin, Heidelberg, 2010. Springer-Verlag.

[17] Benjamin C. M. Fung, Ke Wang, Ada Wai-Chee Fu, and Jian Pei. Anonymity for continuous data publishing. In Alfons Kemper, Patrick Valduriez, Noureddine Mouaddib, Jens Teubner, Mokrane Bouzeghoub, Volker Markl, Laurent Amsaleg, and Ioana Manolescu, editors, EDBT 2008, 11th International Conference on Extending Database Technology, Nantes, France, March 25-29, 2008, Proceedings, volume 261 of ACM International Conference Proceeding Series, pages 264–275. ACM, 2008.

[18] Philippe Golle. Revisiting the uniqueness of simple demographics in the us population. In Proceedings of the 5th ACM Workshop on Privacy in Electronic Society, WPES '06, page 77–80, New York, NY, USA, 2006.

[19] Thomas R. Gruber. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2) :199–221, 1993.

[20] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. LUBM : A benchmark for OWL knowledge base systems. J. Web Semant., 3(2-3) :158–182, 2005.

[21] Michael Hay, Gerome Miklau, David D. Jensen, Philipp Weis, and Siddharth Srivastava. Anonymizing social networks. 2007.

[22] Benjamin Heitmann, Felix Hermsen, and Stefan Decker. k - RDF-neighbourhood anonymity : Combining structural and attribute-based anonymisation for linked data. In Proceedings of the 5th Workshop on Society, Privacy and the Semantic Web - Policy and Technology (PrivOn2017) co-located with ISWC, 2017.

[23] Mike Hintze and Khaled El Emam. Comparing the benefits of pseudonymisation and anonymisation under the gdpr. volume 2, pages 145–158, 2018.

[24] Noah M. Johnson, Joseph P. Near, and Dawn Song. Towards practical differential privacy for SQL queries. Proc. VLDB Endow., 11(5) :526–539, 2018.

[25] Stephan Kessler, Jens Hoff, and Johann-Christoph Freytag. SAP HANA goes private - from privacy research to privacy aware enterprise analytics. Proc. VLDB Endow., 12(12) :1998–2009, 2019.

[26] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Incognito : Efficient full-domain k-anonymity. In Fatma Özcan, editor, Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005, pages 49–60. ACM, 2005.

[27] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Mondrian multidimensional k-anonymity. In Ling Liu, Andreas Reuter, Kyu-Young Whang, and Jianjun Zhang, editors, Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, 3-8 April 2006, Atlanta, GA, USA, page 25. IEEE Computer Society, 2006.

[28] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness : Privacy beyond k-anonymity and l-diversity. In Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007, pages 106–115, 2007.

[29] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. L-diversity : Privacy beyond k-anonymity. ACM Trans. Knowl. Discov. Data, 1(1), March 2007.

[30] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. L-diversity : Privacy beyond k-anonymity. ACM Trans. Knowl. Discov. Data, 1(1) :3–es, March 2007.

[31] Alexander Maedche and Valentin Zacharias. Clustering ontology-based metadata in the semantic web. In Principles of Data Mining and Knowledge Discovery, PKDD 2002, pages 348–360, 2002.

[32] Christopher D. Manning and Hinrich Schütze. Foundations of Statistical Natural Language Processing. MIT Press, Cambridge, MA, USA, 1999.

[33] Frank McSherry. Privacy integrated queries. Communications of the ACM, 53 :89–97, Sept 2010.

[34] Adam Meyerson and Ryan Williams. On the complexity of optimal k-anonymity. In Catriel Beeri and Alin Deutsch, editors, Proceedings of the Twenty-third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS '04, pages 223–228. ACM, 2004.

[35] Li Ninghui, Li Tiancheng, and Suresh Venkatasubramanian. t-closeness : Privacy beyond k-anonymity and l-diversity. In Proceedings - International Conference on Data Engineering, pages 106–115, 2007.

[36] Oracle. Using oracle data redaction with oracle database features. https ://-bit.ly/3AN3YDh, 2021.

[37] Jian Pei, Jian Xu, Zhibin Wang, Wei Wang, and Ke Wang. Maintaining k-anonymity against incremental updates. In 19th International Conference

on Scientific and Statistical Database Management (SSDBM 2007), pages 5–5, 2007.

[38] Fabian Prasser, F. Kohlmayer, R. Lautenschläger, and K. Kuhn. ARX - a comprehensive tool for anonymizing biomedical data. AMIA ... Annual Symposium proceedings. AMIA Symposium, 2014 :984–93, 2014.

[39] Filip Radulovic, Raúl García-Castro, and Asunción Gómez-Pérez. Towards the anonymisation of RDF data. In SEKE, pages 646–651, 2015.

[40] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. Int. J. Comput. Vision, 40(2) :99–121, nov 2000.

[41] Julian Salas and Vicenç Torra. A general algorithm for k-anonymity on dynamic databases. In Data Privacy Management, Cryptocurrencies and Blockchain Technology : ESORICS 2018 International Workshops, DPM 2018 and CBT 2018, Barcelona, Spain, September 6-7, 2018, Proceedings, volume 11025 of Lecture Notes in Computer Science, pages 407–414, 2018.

[42] Latanya Sweeney. k-anonymity : A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 10 :557–570, 2002.

[43] Latanya Sweeney. K-anonymity : A model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl.-Based Syst., 10(5) :557–570, October 2002.

[44] M. Thouvenot, O. Curé, and P. Calvez. Knowledge graph anonymization using semantic anatomization. In 2020 IEEE Int. Conf. on Big Data (Big Data), pages 4065–4074, 2020.

[45] Maxime Thouvenot, Olivier Curé, and Philippe Calvez. Preventing attribute and entity disclosures : Combining k-anonymity and anatomy over RDF graphs. In Yixin Chen, Heiko Ludwig, Yicheng Tu, Usama M. Fayyad, Xingquan Zhu, Xiaohua Hu, Suren Byna, Xiong Liu, Jianping Zhang, Shirui Pan, Vagelis Papalexakis, Jianwu Wang, Alfredo Cuzzocrea, and Carlos Ordonez, editors, 2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, December 15-18, 2021, pages 5460–5469. IEEE, 2021.

[46] Maxime Thouvenot, Olivier Curé, and Philippe Calvez. Preventing attribute and entity disclosures : Combining k-anonymity and anatomy over RDF graphs. In Yixin Chen, Heiko Ludwig, Yicheng Tu, Usama M. Fayyad, Xingquan Zhu, Xiaohua Hu, Suren Byna, Xiong Liu, Jianping Zhang, Shirui Pan, Vagelis Papalexakis, Jianwu Wang, Alfredo Cuzzocrea, and Carlos Ordonez, editors, 2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, December 15-18, 2021, pages 5460–5469. IEEE, 2021.

[47] Zhibiao Wu and Martha Stone Palmer. Verb semantics and lexical selection. In Association for Computational Linguistics, pages 133–138, 1994.

[48] Xiaokui Xiao and Yufei Tao. Anatomy : Simple and effective privacy preservation. In Proceedings of VLDB, Seoul, Korea, September 12-15, 2006, pages 139–150, 2006.

[49] Xiaokui Xiao and Yufei Tao. M-invariance : towards privacy preserving re-publication of dynamic datasets. In Chee Yong Chan, Beng Chin Ooi, and Aoying Zhou, editors, Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007, pages 689–700. ACM, 2007.

[50] Weiqin Xu, Olivier Curé, and Philippe Calvez. Knowledge graph management on the edge. In Yannis Velegrakis, Demetris Zeinalipour-Yazti, Panos K. Chrysanthis, and Francesco Guerra, editors, Proceedings of the 24th International Conference on Extending Database Technology, EDBT 2021, Nicosia, Cyprus, March 23 - 26, 2021, pages 229–240. OpenProceedings.org, 2021.

[51] Jiale Zhang, Bing Chen, Yanchao Zhao, Xiang Cheng, and Feng Hu. Data security and privacy-preserving in edge computing paradigm : Survey and open issues. IEEE Access, PP :1–1, 03 2018.

[52] Bin Zhou and Jian Pei. Preserving privacy in social networks against neighborhood attacks. In Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, Mexico, pages 506–515, 2008.

[53] Bin Zhou and Jian Pei. The k-anonymity and l-diversity approaches for privacy preservation in social networks against neighborhood attacks. Knowl. Inf. Syst., 28(1) :47–77, 2011.