THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Université Toulouse III – Paul Sabatier
Discipline : Informatique

Présentée et soutenue par M. Raddad AL KING

Le 11 Mai 2010
Titre :

Localisation de sources de données et optimisation de requêtes réparties en environnement pair-à-pair

---

JURY

Zohra BELLAHSENE	**Professeur à l'Université Montpellier II**
Bruno DEFUDE	**Professeur à l'École Télécom SudParis (Rapporteur)**
Abdelkader HAMEURLAIN	**Professeur à l'Université Paul Sabatier (Directeur de Thèse)**
Franck MORVAN	**MCF-HDR à l'Université Paul Sabatier (Encadrant)**
Florence SEDES	**Professeur à l'Université Paul Sabatier**
Farouk TOUMANI	**Professeur à l'Université Blaise Pascal (Rapporteur)**

Ecole doctorale : Mathématiques, Informatique, Télécommunications de Toulouse
Unité de recherche : Institut de Recherche en Informatique de Toulouse (IRIT)
Equipe d'accueil : Pyramide

Localisation de sources de données et optimisation de requêtes réparties en environnement pair-à-pair

Raddad AL KING

Mai 2010

Résumé

Malgré leur succès dans le domaine du partage de fichiers, les systèmes P2P sont capables d'évaluer uniquement des requêtes simples basées sur la recherche d'un fichier en utilisant son nom. Récemment, plusieurs travaux de recherche sont effectués afin d'étendre ces systèmes pour qu'ils permettent le partage de données avec une granularité fine (i.e. un attribut atomique) et l'évaluation de requêtes complexes (i.e. requêtes SQL).

A cause des caractéristiques des systèmes P2P (e.g. grande-échelle, instabilité et autonomie de nœuds), il n'est pas pratique d'avoir un catalogue global qui contient souvent des informations sur: les schémas, les données et les hôtes des sources de données. L'absence d'un catalogue global rend plus difficiles: (i) la localisation de sources de données en prenant en compte l'hétérogénéité de schémas et (ii) l'optimisation de requêtes.

Dans notre thèse, nous proposons une approche pour l'évaluation des requêtes SQL en environnement P2P. Notre approche est fondée sur une ontologie de domaine et sur des formules de similarité pour résoudre l'hétérogénéité sémantique des schémas locaux. Quant à l'hétérogénéité structurelle de ces schémas, elle est résolue grâce à l'extension d'un algorithme de routage de requêtes (i.e. le protocole Chord) par des Indexes de structure. Concernant l'optimisation de requêtes, nous proposons de profiter de la phase de localisation de sources de données pour obtenir toutes les méta-données nécessaires pour générer un plan d'exécution proche de l'optimal. Afin de montrer la faisabilité et la validité de nos propositions, nous effectuons une évaluation des performances et nous discutons les résultats obtenus.

Mots clés : *P2P, Bases de données, Hétérogénéité de schémas, DHT, Ontologie, Évaluation des performances.*

Remerciements

Je tiens tout d'abord à remercier Monsieur Luis Fariñas del Cerro, Directeur de l'Institut de Recherche en Informatique de Toulouse (IRIT), pour m'avoir accueilli au sein de son laboratoire.

J'adresse mes sincères remerciements aux Professeurs Bruno Defude et Farouk Toumani pour l'intérêt qu'ils ont porté à mon travail en acceptant d'être les rapporteurs de ma thèse. Je tiens aussi à remercier les Professeurs Zohra Bellahsene, Claude Chrisment, Claudia Roncancio et Florence Sèdes, membres du jury, pour le travail et le temps consacrés à l'examen de ma thèse.

Toute ma reconnaissance s'adresse à Monsieur le Professeur Abdelkader Hameurlain, le directeur de ma thèse qui m'a formé à la recherche. Je le remercie pour sa disponibilité, ses précieux conseils et ses encouragements qui m'ont toujours poussé à m'améliorer.

Je tiens également à adresser mes vifs remerciements à Monsieur Franck Morvan, Maître de Conférences habilité à diriger des recherches, qui a suivi de près mes travaux. Ses qualités humaines, son efficacité, ses connaissances, ses remarques, son soutien et ses encouragements m'ont beaucoup aidé dans la réalisation de cette thèse.

Je remercie aussi Monsieur Riad Mokadem, Maître de Conférences, pour son soutien et ses remarques constructives.

Je remercie les autres membres de l'équipe Pyramide pour la bonne ambiance que nous avons partagée durant ma thèse. Merci à Belgin Ergenç, Mahmoud El Samad, Mohammed Hussein, Christelle Pierkot, Sharifullah Khan, Imen Ketata, Deniz Cokuslu, Igor Epimakhov et sans oublier bien sûr Nadhem Marsit, à qui j'adresse un grand Merci.

Je remercie Madame Agathe Baritaud pour son accueil chaleureux et pour son aide lors de chacune de mes missions. Je remercie aussi Monsieur Jeaun-Pierre Baritaud pour sa bonne humeur communicative à chaque fois je suis allé demander de l'aide. Merci beaucoup à Monsieur Jeun-Philippe Cornille pour sa grande disponibilité et pour avoir toujours imprimé mes documents envoyés à la dernière minute. Je remercie également Monsieur Jeun-Claude Debelle et Monsieur Christophe Rosa pour leur aide rapide lors des problèmes de connexion. Merci à Madame Martine Labruyère qui m'a toujours accueilli avec le sourire pour répondre patiemment à mes questions.

Je remercie beaucoup toutes les personnes qui m'ont aidé durant mon séjour dans la ville rose Toulouse. Notamment, Madame Marie-Christine Comparain et Madame Françoise Lavaux. Je remercie également mes amis et leurs familles, surtout Ahed Alboody, Rami Mohammad, Ali Salem, Youssef Alahmad, Samer Rabih, Sam Aalahmad et Wassim Mahfouz pour leur aide, leur humanité et leur disponibilité.

Je pense beaucoup à la mémoire de mon père Ghassan qui nous a quitté depuis quelques mois et à ma mère Zohra à qui j'adresse toute ma gratitude. Je remercie vivement les membres de ma famille et mes proches pour leur patience et leur soutien tout au long de cette thèse.

J'adresse un grand et spécial merci à ma femme Manazel, sans elle je n'aurais certainement pas pu réaliser cette thèse. Sa patience, son aide, son encouragement, son amour et son humanité m'ont beaucoup aidé à dépasser des moments très difficiles durant ma thèse.

À mes parents à qui je dois tout

À Manazel, Youssef, Louay et Yara à qui je dois beaucoup

À tous les miens

Raddad

Table des matières

Table des figures

Chapitre I

Sommaire

Introduction générale

Ces dernières années, les systèmes pair-à-pair (désormais P2P) sont devenus très populaires. Cette popularité vient des bonnes caractéristiques offertes par ces systèmes comme : la grande échelle, l'autonomie des nœuds et le contrôle décentralisé. Ce type de systèmes offre une bonne opportunité pour répondre aux limites des systèmes basés sur le paradigme Client/Serveur. En évitant d'éventuels goulets d'étranglement et en offrant une bonne tolérance aux pannes, les systèmes P2P sont bien adaptés aux environnements distribués à grande échelle dans lesquels les nœuds (appelés indifféremment nœuds ou pairs) peuvent partager leurs ressources (e.g. puissance de calcul, capacité de stockage, bande passante) d'une façon autonome et décentralisée. Chaque nœud peut jouer le rôle d'un serveur en offrant ses ressources aux autres nœuds, d'un client en consommant les ressources des autres nœuds existants dans le système, d'un routeur en propageant les requêtes et les messages venant des autres nœuds et d'un hôte[1] de source de données en partageant ses propres données avec d'autres nœuds. Dans ce chapitre, nous présentons le contexte général, la problématique abordée et les objectifs de notre thèse tout en survolant nos principales contributions.

1.1. Contexte général et motivations

Les systèmes P2P sont utilisés avec succès dans plusieurs domaines comme : le partage de fichiers, le partage de capacité de calcul et l'échange de messages instantanés. Grâce à leurs caractéristiques avantageuses, de nouveaux domaines se dirigent vers l'utilisation de ces systèmes dans de nouvelles applications. Dans le domaine de la santé publique par exemple, nous pouvons citer quelques exemples de ces applications:

(i) Médecins d'hôpital : dans un hôpital, certaines données concernant les patients (e.g. nom, adresse) sont stockées dans une base de données. D'autres données (e.g. rayons-X, prescription, histoire, réaction aux médicaments) sont stockées sur l'ordinateur du spécialiste qui soigne ces patients [NOT+03]. Normalement, le spécialiste partage la plupart de ses données avec d'autres collègues et il en cache une partie (e.g. il cache les données d'une

[1] Tout au long de ce mémoire, nous utilisons "Hôte de sources de données" pour désigner un site sur lequel une source de données est stockée.

nouvelle expérience sur un nouveau médicament concernant la maladie d'Alzheimer). Les autres spécialistes peuvent profiter des données partagées pour soigner les patients qui ont les mêmes symptômes. Ceci leur permet de prendre une meilleure décision. Dans cet exemple, on a besoin d'un système informatique reliant les ordinateurs de tous les spécialistes en permettant à chaque spécialiste de se connecter/se déconnecter au/du système à n'importe quel moment afin de partager ses données qui peuvent être représentées par un schéma différent des autres schémas. Le système doit permettre à un nouveau spécialiste d'y participer facilement sans avoir besoin d'une administration centralisée. Nous pensons qu'un système P2P de partage de données est capable de mieux répondre à ce besoin.

(ii) Recherche médicale : plusieurs chercheurs travaillant, autour du monde, sur un médicament concernant la maladie d'Alzheimer veulent partager leurs données pendant la durée d'une expérience. Cet exemple reflète le besoin de faire combiner les techniques du partage de données en environnements distribués et celles développées dans les systèmes P2P afin de créer un système P2P de partage de données.

(iii) Données génétiques : la gestion de données génétiques nécessite une grande capacité de stockage et une forte puissance de calcul [NOT+03]. En effet, la découverte d'une nouvelle protéine nécessite une analyse complexe pour déterminer ses fonctions et ses classifications. La technique utilisée par les scientifiques est constituée de deux phases : (a) chercher dans une base de données des protéines connues et y extraire l'ensemble des protéines ressemblant à la nouvelle protéine et (b) analyser les fonctions et les classifications des protéines extraites pour essayer de trouver des caractéristiques communes avec la nouvelle protéine. Il y a beaucoup de serveurs stockant des données génétiques (e.g. GenBank[2], EMBL[3]), et il y a beaucoup de données génétiques produites chaque jour dans différents laboratoires dans le monde entier. Les scientifiques créent leurs bases de données concernant les nouvelles protéines découvertes et ils partagent les résultats avec les autres scientifiques dans le monde. Afin de mieux gérer les données génétiques stockées dans plusieurs bases de données dans le monde, on a besoin d'un système informatique capable de gérer ce grand volume de données génétiques, de relier ces bases de données qui peuvent être hétérogènes et de garder, en même temps, l'autonomie de chaque base de données.

[2] http://www.ncbi.nlm.nih.gov/Genbank/
[3] http://www.embl.fr/

(iv) Histoire médicale des patients : nous prenons le cas d'une personne ayant un accident pendant son voyage à l'étranger, le médecin traitant cette personne a besoin d'accéder aux données stockées dans les bases de données situées dans le pays d'origine de cette personne afin de connaître son histoire médicale. Le médecin traitant, autant qu'un utilisateur, doit avoir la capacité de relier sa base de données avec les bases de données des médecins ayant déjà traité son patient. Nous croyons qu'un système P2P de partage de données permet au médecin traitant d'accéder aux données souhaitées. Cet accès peut avoir lieu qu'une seule fois et à n'importe quel moment, c'est pourquoi, il ne doit pas être coûteux et il ne doit pas nécessiter l'intervention d'un administrateur qui ne peut pas être disponible à tout moment.

Nous croyons que les systèmes P2P de partage de données peuvent répondre aux besoins présentés dans les exemples précédents. Cependant, jusqu'au jour de la rédaction de ce manuscrit, il n'y a pas de consensus entre les chercheurs sur comment concevoir ce type de système. Cela est dû aux problèmes ouverts concernant l'architecture, la représentation de données, l'évaluation de requêtes, la sécurité, etc. A travers de la section suivante, nous précisons la problématique abordée dans le cadre de notre thèse et nous précisions nos objectifs.

1.2. Problématique et objectifs

A notre connaissance, trois approches sont principalement utilisées pour permettre le partage de données en environnements distribués. La première approche consiste à utiliser un SGBD réparti [OzV99]. Cette approche est convenable au cas où les données sont fragmentées sur plusieurs sites et si le contrôle est centralisé sur un seul site. Il existe un schéma global représentant toutes les données gérées par le système et un catalogue global stockant les méta-données de ces données. Le catalogue global contient aussi des informations sur la localisation des fragments et sur leurs caractères [OzV99]. Pratiquement, cette approche ne permet de gérer que quelques dizaines de bases de données [VaP04].

Avec le développement de l'Internet ces dernières années, des données stockées dans des milliers de sources de données peuvent être accessibles. Une source de données a un sens plus général qu'une base de données, elle peut être une base de données ou un fichier. Cependant, tout au long de cette thèse nous utilisons ''source de données'' pour désigner une base de données. Afin de partager les données stockées dans ce type de sources de données, les systèmes d'intégration de données sont utilisés. Ces systèmes utilisent l'une de

deux approches : l'entrepôt de donnée (*Data Warehousing*) et la médiation de données (*Data mediation*). L'approche de l'entrepôt de données consiste à remplacer les sources de données par une source unique. Toutes les requêtes doivent être exécutées sur le site de cette source. Deux grandes limites empêchent cette approche de passer à grande échelle : (a) la mise à jour de données stockées sur le site unique est coûteuse surtout lorsque les hôtes des sources de données sont instables et (b) l'exécution de requêtes sur un seul site peut créer un goulet d'étranglement. Quant à l'approche de médiation de données, cette approche permet d'accéder aux données sur leurs sources en offrant une intégration virtuelle de données. Un schéma global représentant toutes les données partagées dans le système et des règles de correspondances entre le schéma global et les schémas de sources permettent de résoudre le problème de l'hétérogénéité des schémas de sources de données. Malgré ses utilisations majeures de nos jours, les systèmes d'intégration de données ne permettent de gérer que quelques centaines de sources de données [VaP04].

Les trois approches précédentes ne sont pas convenables en environnement P2P dans lequel des centaines de milliers de nœuds peuvent partager leurs données via un réseau largement distribué (i.e. l'Internet) d'une façon autonome et décentralisée. Une nouvelle approche apparaît en recherche. Elle consiste à utiliser un système P2P de partage de données qui forme, d'un côté, une nouvelle génération des systèmes P2P et, d'un autre côté, un nouveau pas dans le long chemin de la recherche en bases de données. Même si les auteurs de [OST03] considèrent un système P2P de partage de données comme un système de bases de données hétérogènes et réparties, les auteurs de [NOT+03] ont cité quatre différences entre les systèmes P2P et les systèmes de bases de données réparties (SBDR) existants :

(i) Dans les systèmes P2P, les nœuds peuvent se connecter/se déconnecter à n'importe quel moment. Dans les SBDR, les nœuds sont ajoutés et supprimés d'une façon contrôlée (i.e. lors de l'évolution du système ou du retrait d'un nœud).

(ii) Dans les systèmes P2P, souvent il n'y pas de schéma global représentant toutes les données partagées dans le système. Les requêtes sont souvent basées sur la recherche par mots clés. En effet : (a) la plupart des applications dans les systèmes P2P ne nécessitent pas ce type de schéma. (b) l'instabilité de nœuds empêchent d'avoir un schéma global reflétant les données partagées existantes dans le système à un moment donné. Dans les SBDR, les nœuds sont typiquement stables et partagent un schéma global.

(iii) Dans les systèmes P2P, les réponses aux requêtes ne sont pas complètes. Parfois, les nœuds contenant les réponses valides ne sont pas connectés au système. De plus, dans certains cas, il est difficile d'obtenir une réponse complète à cause du grand nombre de nœuds dans le système. Dans les SBDR, on peut prévoir et obtenir toutes les réponses valides existantes dans le système considéré.

(iv) Dans les systèmes P2P, le placement exact d'une donnée est connu par la propagation de requête vers les voisins. Dans les SBDR, la requête est envoyée directement vers le placement exact de la donnée demandée. Ce placement est typiquement connu dès le début avant de soumettre la requête.

Malgré les nombreux travaux de recherche [e.g. AMP+07, HCH+05, BoT03, TIM+03, BGK+02, KMH03, NOT+03] menés dans le cadre de la conception d'un système P2P de partage de données, nous remarquons que ces travaux n'arrivent toujours pas à proposer un système P2P de gestion de données capable d'évaluer efficacement des requêtes complexes exprimées dans un langage de haut niveau (i.e. SQL). Une évaluation efficace signifie un temps de réponse réduit et une consommation de ressources minimisée. Nous croyons qu'une des raisons de cette remarque est que dans un système P2P aucun nœud n'a une vision globale sur le système. C'est pourquoi dans ces systèmes il n'est pas pratique d'avoir un catalogue global qui est un composant essentiel dans les SBDR. Avoir un catalogue global centralisé dans un système P2P peut créer un goulet d'étranglement. Un catalogue dupliqué sur tous les nœuds n'est pas une solution non plus à cause d'un nombre important de messages nécessaires pour mettre les différents exemplaires de ce catalogue à jour. Un catalogue global totalement distribué est très compliqué à gérer. Le catalogue global contient souvent des informations sur les règles de correspondances entre les schémas et sur le placement de données. Il contient des statistiques permettant de générer un plan d'exécution proche de l'optimal.

En environnement P2P, deux problèmes majeurs sont des conséquences à l'absence d'un catalogue global : (i) la localisation de source de donnée et (ii) l'optimisation de requêtes. Le problème de la localisation de sources de données peut être expliqué comme suivant. Supposant qu'un nœud A a besoin d'une entité de données D qui est stockée sur plusieurs nœuds dans le système. Le nœud A ne connaît pas la localisation de D sachant que D peut être représentée de manières différentes sur les nœuds qui la stockent. Une question capitale ici est comment peut-on trouver les nœuds stockant D d'une façon efficace en l'absence d'un catalogue global, et en tenant compte de l'instabilité et de la grande échelle de

l'environnement P2P ? Quant au problème de l'optimisation globale, étant donné une requête en environnement P2P, la question qui se pose est la suivante : avec l'absence de catalogue global, comment générer un plan d'exécution proche de l'optimal permettant une exécution efficace de requêtes?

Nos objectifs au sein de cette thèse sont : (i) trouver des méthodes efficaces pour résoudre les problèmes précédents et (ii) évaluer les performances des méthodes proposées afin de montrer leurs validité et efficacité. Dans la section suivante, nous allons expliquer ce que nous avons fait pour atteindre ces deux objectifs.

1.3. Contributions

Cette section est consacrée à présenter nos contributions dans le contexte de l'évaluation de requêtes SQL en environnement P2P dans lequel les nœuds partagent des données représentées par des schémas structurellement et sémantiquement hétérogènes. Dans la suite, nous expliquons les contributions effectuées dans le cadre de cette thèse:

(i) Notre première contribution consiste à présenter une architecture logicielle [KHM09a] d'un système P2P de partage de données et une approche pour l'évaluation de requêtes dans cette architecture. Cette approche est une adaptation de celle utilisée dans les systèmes d'intégration de données avec l'environnement P2P en tenant en compte les contraintes imposées par un tel environnement.

(ii) Afin de résoudre le problème de l'hétérogénéité sémantique entre les schémas locaux, nous proposons d'utiliser une ontologie de domaine dupliquée sur tous les nœuds. Nous supposons que cette ontologie est rarement modifiée et qu'elle forme une interface unique pour l'interaction entre les nœuds dans le système. Pour qu'un nouveau nœud puisse entrer dans le système sans besoin d'une administration ou d'une autorisation centralisée, l'utilisateur doit avoir la capacité de relier sa base de données avec le système. Pour cette raison, nous avons proposé des formules de similarité [KHM09b] permettant à l'utilisateur de créer des règles de correspondance entre l'ontologie de domaine et son schéma local. Ces règles doivent être stockées localement afin d'être utilisées lors du traitement de requêtes.

(iii) Après avoir réalisé un état de l'art, nous avons constaté que les systèmes P2P structurés sont plus performants que ceux non-structurés en termes de nombre de messages échangés entre les nœuds et en termes de qualité de réponses obtenus. Ils sont aussi plus tolérants aux pannes que les systèmes P2P super-pairs qui centralisent leurs services sur les super-pairs. Lorsqu'un super-pair tombe en panne, tous ses clients seront privés de ses

services et deviennent isolés du reste du système. Pour ces raisons, le système considéré par notre étude est un système P2P structuré. Nous avons choisi le protocole Chord [KHM07b] pour effectuer le routage de requêtes dans le système car il est simple et "efficace" en même temps. Afin de résoudre le problème de l'hétérogénéité structurelle entre les schémas locaux, nous avons étendre le protocole Chord par des indexes de structure [KHM08b] qui décrivent la structure de données sur leurs sources. Nous avons proposé une méthode montrant comment les indexes de structure peuvent être utilisés par le protocole Chord afin de localiser les sources de données pertinentes. Par "pertinentes", nous désignons les sources qui stockent les données désirées par la requête évaluée.

(iv) Nous avons constaté que l'absence d'un catalogue global en environnement P2P a un impact important sur l'optimisation globale de requêtes. Nous avons remarqué que pour générer un plan d'exécution proche de l'optimal, il n'est pas nécessaire d'avoir un catalogue global dans lequel se trouvent des informations concernant tous les nœuds et tous leurs contenus. Cependant, il suffit d'avoir des informations sur les données et sur les hôtes de leurs sources afin de générer un plan d'exécution exécutable sur les hôtes des sources de données et sur le nœud initialisant la requête. Pour cette raison, nous avons proposé d'étendre notre méthode de localisation de sources de données pour qu'elle permette d'obtenir toutes les méta-données nécessaires pour effectuer l'optimisation de requêtes [KHM07b]. Le fait d'obtenir des méta-données existantes sur les hôtes des sources de données permet d'obtenir des informations fiables. De plus, le fait d'obtenir ces méta-données durant la localisation de sources de données permet d'éviter un accès supplémentaire au réseau caractérisé par une grande latence et par un faible débit.

(v) Notre dernière contribution consiste à montrer la faisabilité et la validité de nos propositions. Les performances des formules de similarité, de la méthode de localisation de sources de données et de la méthode d'obtention de méta-données sont évaluées et les résultats obtenus sont discutés.

Malgré les aspects abordés dans ce manuscrit, plusieurs perspectives sont présentées afin de montrer comment on peut améliorer les solutions proposées dans un prochain avenir. Dans la section suivante, nous présentons l'organisation de ce manuscrit.

1.4. Organisation du manuscrit

Ce manuscrit est constitué de cinq chapitres. Après l'introduction générale, nous présentons dans le chapitre 2 un état de l'art décrivant les systèmes P2P, leurs

caractéristiques, leurs classes et les approches utilisées pour résoudre le problème de la localisation de sources de données en environnement P2P. Deux aspects liés à ce problème seront étudiés : le routage de requêtes et le *matching*[4] de schémas. Nous citons également des travaux de recherche abordant l'optimisation de requêtes et nous présentons une comparaison qualitative entre nos propositions et des principaux travaux voisins.

Le chapitre 3 a pour objectif de décrire les contributions que nous avons citées dans la section précédente. Les hypothèses prises en compte sont : (i) Les bases de données considérées par notre étude sont des bases de données relationnelles. (ii) nous ne considérons que des requêtes SQL en mode lecture seule. (iii) nous utilisons un système P2P structuré basé sur le protocole Chord. (iv) nous utilisons une ontologie de domaine pour jouer un rôle similaire à celui du schéma global dans les systèmes de médiation de données.

Quant au chapitre 4, nous décrivons l'évaluation des performances des formules et des méthodes proposées au sein de notre thèse. Nous expliquons les hypothèses prises en compte, les jeux de données utilisées et les méthodologies d'évaluations. Nous montrons également les résultats obtenus et nous commentons ces résultats.

Dans le chapitre 5, nous concluons ce manuscrit en établissant une synthèse du travail proposé au sein de notre thèse. Nous récapitulons les problèmes ouverts et nous décrivons notre plan concernant nos futurs travaux liés à l'amélioration de nos propositions.

1.5. Publications

[KHM07a] R. A. King, A. Hameurlain and F. Morvan. "*Metadata Lookup for Distributed Query Optimization in P2P Environment"*. International Conference on Parallel and Distributed Computing Systems (PDCS'07), International Society for Computers and their Applications (ISCA), Las Vegas, Nevada, USA, pp. 36-43, 2007.

[KHM07b] R. A. King, A. Hameurlain and F. Morvan. *"O-Chord: A Method for Locating Relational Data Sources in a P2P Environment"*. International Conference on Systems, Computing Sciences and Software Engineering (SCSS'07), University of Bridgeport, Connecticut, USA, pp. 416-421, 2008.

[KHM08a] R. A. King, A. Hameurlain and F. Morvan. *"Ontology-Based Query Processing in a Large-Scale P2P Environment"*. IEEE International Conference on

[4] La traduction du mot anglais "*Matching*" est "Appariement" en français. Nous remarquons que cette traduction n'est pas populaire. Pour cette raison nous utilisons, au sein de cette thèse, le mot anglais "*Maching*".

Information and Communication Technologies: from Theory to Applications (ICTTA'08), Damascus, Syria, 2008.

[KHM08b] R. A. King, A. Hameurlain and F. Morvan. "*Ontology-Based Data Source Localization in a Structured Peer-to-Peer Environment"*. International Database Engineering & Applications Symposium (IDEAS'08), Coimbra, Portugal, pp. 9-18, 2008.

[KHM09a] R. A. King, A. Hameurlain and F. Morvan. "Ontology-Based Method for Schema *Matching in Peer-to-Peer Database System"*. British National Conference on Databases (poster session) (BNCOD'09), Birmingham, UK, pp. 208-212, 2009.

[KHM09b] R. A. King, A. Hameurlain and F. Morvan. *"Matching Heterogeneous Schemas in a Large-Scale Peer-to-Peer Database Environment"*. Int. Conf. on Parallel and Distributed Computing and Communication Systems (PDCCS'09), Louiseville, Kentucky, USA, International Society for Computers and their Applications (ISCA), pp. 135-142, 2009.

[KHM10] R. A. King, A. Hameurlain et F. Morvan, "*Query Routing and Processing in Peer-To-Peer Data Sharing Systems*". International Journal of Database Management Systems, Academy & Industry Research Collaboration Center (AIRCC), Vol. 2 N. 2, pp. 116-139, Mai 2010.

Chapitre II.

Sommaire

Partage de données en environnement pair-à-pair : état de l'art

Le terme ''pair-à-pair'' (désormais P2P) devient très populaire ces dernières années. Ce paradigme représente une nouvelle façon de partage de ressources via l'Internet. Grâce à ses avantages, plusieurs domaines ont déjà profité de ce paradigme (e.g. partage de fichiers[5], partage de capacité de calcul[6] et échange de messages instantanés[7]). Un des domaines tendant à en profiter est le partage de données à une granularité fine (e.g. attribut atomique) en utilisant un langage de requêtes de haut niveau (i.e. SQL). A ce stade, trouver efficacement les sources de données pertinentes et générer un plan d'exécution proche de l'optimal sont deux problèmes ouverts en environnement P2P. Quels sont les principaux travaux de recherche menés pour résoudre ces problèmes ? Quels sont leurs avantages et leurs limites ? Dans ce chapitre, nous répondons à ces questions, juste, après une courte synthèse des systèmes P2P, de leurs caractéristiques et de leurs classes.

2.1. Introduction

Depuis quelques années, le paradigme P2P attire, de plus en plus, l'attention de plusieurs chercheurs en Informatique. Ce paradigme est apparu comme une solution aux quelques limites de la méthode classique de partage de ressources basée sur le paradigme Client/Serveur. En évitant des goulets d'étranglement susceptibles et en offrant une bonne tolérance aux pannes, le paradigme P2P est approprié pour des environnements distribués à grande échelle dans lesquels les nœuds (appelés indifféremment nœuds ou pairs) peuvent partager leurs ressources (e.g. puissance de calcul, capacité de stockage, bande passante) d'une façon autonome et décentralisée.

Le partage de fichiers musicaux via l'Internet était la motivation originale des premiers systèmes P2P [SGG03]. Même si le paradigme P2P a connu un grand succès dans ce

[5] e.g. Gnutella homepage, http://www.gnutella.fr
[6] e.g. SETI@home homepage, http://setiathome.berkeley.edu/
[7] e.g. ICQ homepage, http://www.icq.com/

domaine, il ne support qu'un langage d'interrogation "pauvre" habituellement basé sur la recherche d'un fichier en utilisant son nom. Récemment, de nouveaux domaines semblent avoir besoin de profiter du paradigme P2P. Par exemple, dans le domaine médical, plusieurs chercheurs, autour du monde, travaillant sur la maladie d'Alzheimer veulent partager leurs données pendant la durée d'une expérience. Dans cet exemple, il est nécessaire de faire combiner les techniques du partage de données en environnements distribués et celles développées dans les systèmes P2P. Le but d'une telle combinaison est de permettre aux systèmes P2P de supporter le partage de données à une granularité plus fine qu'un fichier et d'évaluer des requêtes complexes exprimées dans des langages de haut niveau comme SQL. L'évaluation de requêtes complexes est souvent supportée par l'existence d'un catalogue global stockant des informations concernant les données et les caractéristiques des hôtes des sources de données. Ces informations sont utilisées pour générer un plan d'exécution proche de l'optimal. En environnement P2P, à cause de ses caractéristiques, aucun nœud n'a une vision globale. Par conséquent, ce n'est pas pratique d'avoir un catalogue global dans un tel environnent. Cela rend la localisation de sources de données et la génération d'un plan d'exécution proche de l'optimal des problèmes ouverts. Dans ce chapitre, nous présentons un état de l'art concernant ces deux problèmes.

Après cette introduction, nous présentons, dans la section 2, les approches existantes pour le partage de données en environnements distribués. Ensuite et à travers de la section 3, nous présentons les systèmes P2P, leurs caractéristiques, leurs classes et les types de requêtes traitées dans ces systèmes. Dans l'objectif de localiser les sources de données en environnement P2P, deux sujets sont abordés : (i) le routage de requêtes qui sera l'objet de la section 4 et (ii) le *matching* de schémas qui sera étudié dans la section 5. Quant à la section 6, nous abordons le sujet de l'optimisation de requêtes. Ensuite, la section 7 présente une partie des travaux de recherche menés dans le cadre de l'évaluation de requêtes en environnement P2P. Nous comparons ces travaux avec nos solutions proposées dans la section 8. Nous résumons le contenu de ce chapitre au sein de la section 9 et nous présentons notre conclusion.

2.2. Partage de données en environnements distribués

Au cours de notre étude bibliographique, nous avons pu constater que trois principales approches sont traditionnellement utilisées pour le partage de données en environnements distribués. Lorsque les données sont fragmentées sur plusieurs bases de données, l'approche de SGBD réparti permet de gérer ce type de données. Avec le

développement de l'Internet ces dernières années, des données stockées dans des milliers de sources de données peuvent être accessibles via l'Internet. Les sources de données distribuées sur l'Internet sont hétérogènes et gérées d'une façon autonome. Chaque source est développée et maintenue indépendamment des autres sources. Afin de partager les données stockées dans ce type de sources de données, on utilise des systèmes d'intégration de données qui sont basés sur l'une de deux approches : l'entrepôt de donnée (Data Warehousing[8]) et la médiation de données (Data Mediation) [OzV99]. Dans la suite, nous présentons ces approches et nous citons leurs limites.

2.2.1. L'approche de SGBD réparti

Afin de donner une idée sur comment une requête est évaluée selon cette approche, nous prenons le cas des bases de données relationnelles qui sont largement utilisées ces jours-ci. Un SGBD réparti évalue une requête, écrite selon le schéma global du système, en quatre phases [OzV99] : (a) décomposition de requêtes en requêtes algébriques; (b) localisation de fragments (chaque relation est fragmentée sur plusieurs sites); (c) optimisation globale et (d) optimisation locale et exécution. Les trois premières phases sont effectuées sur un site centralisé qui utilise des informations stockées dans le catalogue global du système. Cependant, la quatrième phase est effectuée sur des nœuds désignés durant la phase de l'optimisation globale. Dans les SGBD répartis, de plus du schéma global, le catalogue global stocke des informations concernant les données partagées. Il contient aussi des informations sur les placements des fragments et sur leurs caractères physiques. Par conséquent, ce catalogue joue un rôle essentiel dans le traitement de requêtes surtout lors de la localisation de fragments et de l'optimisation globale.

2.2.2. L'approche de l'entrepôt de données

L'approche de l'entrepôt de données consiste à remplacer les sources de données par une source unique (appelée dépôt de données). Toutes les requêtes doivent être exécutées sur le site du dépôt de données. Cette approche offre une bonne performance en termes de temps de réponse (supposant l'utilisation d'un "bon" SGBD) car l'exécution de requêtes ne prend pas en compte les communications inter-sites. Cependant, la mise à jour de données stockées dans le dépôt de données est typiquement complexe et coûteuse surtout lorsque les sources de données sont instables et nombreuses. De plus, l'exécution de requêtes sur un

[8] The Data Warehousing Information Center, http://www.dwinfocenter.org/

seul site peut créer un goulet d'étranglement particulièrement lorsque le nombre de sources de données est élevé. On peut ajouter aussi que dans certains cas, on ne peut pas dupliquer des données stockées sur certaines sources à cause des raisons de sécurité, ou des spécifications particulières de ces sources.

2.2.3. L'approche de médiation de données

Au lieu de déplacer les données, cette approche permet d'accéder directement aux données sur leurs sources en offrant une intégration virtuelle de données. Un schéma global représentant toutes les données partagées dans le système. L'architecture classique de l'intégration virtuelle de données est l'architecture médiateur/adaptateur (Mediator/Wrapper). Des règles de correspondances entre le schéma global et les schémas locaux des adaptateurs sont créées et stockées d'une façon centralisée. Le médiateur offre un point d'accès unique et uniforme aux sources de données. Il offre aussi une interface permettant d'interroger les sources de données et il contient des informations concernant l'intégration de données. L'adaptateur est un composant reliant une source de données au médiateur. Il rend l'implémentation interne de cette source transparente au médiateur. Il offre des informations concernant la source de données au médiateur en termes de schéma du médiateur. Il peut accepter d'exécuter de sous- requêtes venant du médiateur. L'évaluation d'une requête dans les systèmes de médiation de données est effectué en quatre phases [OzV99] : (i) reformulation de requête en termes des schémas locaux des adaptateurs. (ii) décomposition de la requête reformulée en sous-requêtes afin de les envoyer aux adaptateurs (étant donné qu'il y a plusieurs méthodes pour décomposer une requête, cette phase inclut une étape d'optimisation distribuée). (iii) exécution des sous-requêtes sur les adaptateurs et (iv) envoi des résultats des sous-requêtes au médiateur qui continue à exécuter les étapes restées et qui délivre enfin un résultat complet.

En résumé, les approches existantes pour le partage de données en environnement distribués sont limitées du point de vue de passage à la grande-échelle (gestion des centaines de milliers de sources de données) et du point de vue de la centralisation de contrôle et des informations. Pratiquement, les SGBD répartis ne permettent de gérer que quelques dizaines de bases de données [VaP04] et les systèmes d'intégration de données ne peuvent gérer que quelques centaines de bases de données [VaP04]. Comme, nous allons voir dans la section prochaine, la grande échelle et la décentralisation du contrôle sont deux caractéristiques des systèmes P2P. Pour cette raison, est apparue une nouvelle approche utilisant les systèmes P2P pour partager les données dans un environnement distribué à grande-échelle.

2.3. Systèmes P2P

Les systèmes P2P offrent une opportunité importante pour qu'un grand nombre (centaines de milliers) de nœuds se coopèrent l'un avec les autres afin de partager leurs ressources via un réseau largement distribué (i.e. l'Internet). Plus le nombre de ressources disponibles dans un système P2P est élevé, plus la puissance de calcul et la capacité de stockage ont des valeurs importantes. Cet avantage permet aux systèmes P2P d'exécuter de tâches complexes avec un coût relativement faible sans avoir besoin d'installer des gigantesques serveurs. Au sein d'un système P2P, les nœuds jouent des rôles équivalents. Chaque nœud joue le rôle d'un : client lorsqu'il consomme de ressources disponibles, serveur lorsqu'il offert des ressources, routeur lorsqu'il propage de requêtes reçues aux autres nœuds dans le système et hôte de source de données lorsqu'il partage ses propres données avec d'autres nœuds.

Au cours de notre étude bibliographique, nous avons trouvé plusieurs définitions de "Systèmes P2P". Par exemple, dans [Shi01] le "pair-à-pair" est défini comme « une classe d'applications profitant des ressources (stockage, calcul, contenu, présence humaine) disponibles sur des nœuds connectés à l'internet ». Les auteurs de [ThS04] définissent aussi les systèmes P2P comme "des systèmes répartis consistant des nœuds interconnectés l'un aux autres et capables de s'organiser selon des topologies de réseau afin de partager leurs ressources (contenu, cycles de CPU, stockage et bande passante). Ils sont tolérants aux pannes de leurs nœuds et capables d'accueillir des informations concernant d'autre nœuds, tout en maintenant une connectivité et une performance acceptables, sans exiger l'intermédiation d'un serveur global ou l'utilisation d'une autorité centralisée". Nous remarquons que la première définition représente aussi des systèmes dans lesquels se trouvent plusieurs serveurs comme SETI@home and Kazza ou un seul serveur comme Napster[9]. Cependant, la deuxième définition ignore ces types de systèmes.

Une autre définition, avec qui nous sommes d'accord, est présentée dans [MKL+02] : « Le terme "pair-à-pair" représente une classe de systèmes et d'applications qui utilisent des ressources distribuées afin d'atteindre un objectif précis d'une façon décentralisée. Les ressources incluent puissance de calcul, données (stockage et contenu), bande passante et disponibilité (ordinateurs, être humaine ou autre ressources). L'objectif peut être la répartition de calcul, le partage de données/contenu, la communication et la collaboration

[9] Napster (http://free.napster.com/) est le premier système P2P connu. Même s'il contient un index central il est considéré comme un système P2P particulier.

ou le partage des services d'une plateforme. La décentralisation peut être appliquée sur les algorithmes, les données et les méta-données». Puisque les chercheurs ne sont pas d'accord sur une définition unique (ou standard) des systèmes P2P, nous résumons les principales caractéristiques de ces systèmes.

2.3.1. Caractéristiques des systèmes P2P

Les systèmes P2P possèdent des caractéristiques avantageuses par rapport aux autres systèmes basés sur le paradigme Client/Serveur. Pour cette raison, ils sont très populaires. Même s'il n'y a pas, à notre connaissance, un système P2P existant possédant toutes les caractéristiques suivantes, chaque système P2P essaie de posséder un grand nombre de ces caractéristiques.

Grande- échelle[10] : il s'agit de faire coopérer un grand nombre de nœuds (jusqu'à des milliers ou des millions) pour partager leurs ressources tout en maintenant une bonne performance des système. Cela signifie qu'un système P2P doit offrir des méthodes bien adaptées avec un environnement dans lequel il y a un grand volume de données à partager, un nombre important de messages à échanger entre un grand nombre de nœuds partageant leurs ressources via un réseau largement distribué.

Autonomie de nœuds : chaque nœud gère ses ressources d'une façon autonome. Il décide quelle partie de ses données à partager. Il peut se connecter ou/et se déconnecter à n'importe quel moment. Il possède également l'autonomie de gérer sa puissance de calcul et sa capacité de stockage. Il est capable de décider la méthode de conception de bases de données et le SGBD le plus adéquat à ses besoins.

Environnement dynamique : à cause de l'autonomie de nœuds, chaque nœud peut quitter le système à n'importe quel moment ce qui fait disparaître ses ressources du système. De nouvelles ressources peuvent être ajoutées au système lors de la connexion de nouveaux nœuds. Alors, à cause de l'instabilité de nœuds, les systèmes P2P doivent être capables de gérer un grand nombre de ressources fortement variables. La sortie d'un nœud du système (ou la panne d'un nœud) ne doit pas mettre le système en échec. Elle doit être tolérée et avoir un "petit" impact sur la performance de tout le système.

[10] Nous utilisons l'expression ''passage à grande-échelle'' comme traduction du mot ''Scalability'' venant de la langue anglaise.

Hétérogénéité : à cause de l'autonomie de nœuds possédant des architectures matérielles et/ou logicielles hétérogènes, les systèmes P2P doivent posséder des techniques convenables pour résoudre les problèmes liés à l'hétérogénéité de ressources.

Décentralisation : le fait que chaque nœud gère ses propres ressources permet d'éviter la centralisation de contrôle. Un système P2P peut fonctionner sans avoir aucun besoin d'une administration centralisée ce qui permet d'éviter les goulets d'étranglements et d'augmenter la résistance du système face aux pannes et aux défaillances.

Auto configuration : puisque les systèmes P2P sont souvent déployés sur l'Internet, la participation d'un nouveau nœud à un système P2P ne nécessite pas une infrastructure coûteuse. Il suffit d'avoir un point d'accès à l'Internet et de connaître un autre nœud déjà connecté pour se connecter au système. Un système P2P doit être un environnement ouvert ; c'est-à-dire, un utilisateur sur un nœud doit être capable de connecter son nœud au système sans avoir besoin de contacter une personne et sans avoir besoin de passer par une administration centralisée.

2.3.2. Taxonomie de systèmes P2P

Selon l'étude bibliographique menée dans le cadre de notre thèse, nous avons évoqué l'existence d'une grande diversité d'algorithmes utilisés par les systèmes P2P dans le cadre du partage de données. Ces algorithmes dépendent fortement de la topologie du système P2P considéré. Les nœuds en environnement P2P forment un réseau virtuel situé au-dessus du réseau physique (i.e., l'Internet). La topologie d'un système P2P est la façon dont les nœuds sont situés sur le réseau virtuel. Ella a un fort impact sur la performance du système en termes de routage de messages échangés entre les nœuds, de qualité de résultats obtenus et de nombre de ces résultats. Les systèmes P2P sont classés selon leurs topologies en trois classes essentielles : (i) systèmes P2P non-structurés, (ii) systèmes P2P structurés et (iii) systèmes super-pairs.

2.3.2.1. Systèmes P2P non-structurés

Ce sont les systèmes P2P purs dans lesquels tous les nœuds jouent des rôles équivalents. Il n'y a pas de serveurs ou de nœuds privilégiés. Le degré d'autonomie de chaque nœud est élevé. Les nœuds ne sont pas obligés d'être structurés selon une forme géométrique prédéfinie. Chaque nœud établit une connexion directe avec un ou plusieurs nœuds appelés voisins (ou parfois des connaissances comme dans [BGK+02]). Généralement,

ce n'est pas nécessaire qu'un nœud ait connaissance des ressources disponibles sur ses voisins.

2.3.2.2. Systèmes P2P structurés

L'apparition des systèmes P2P structurés est due au problème de grand nombre de messages échangés dont les systèmes P2P non-structurés souffrent. Ces systèmes permettent de diminuer le nombre de messages échangés au prix de diminuer le degré de l'autonomie de nœuds. Les nœuds doivent, obligatoirement, être structurés sous une forme géométrique précise (e.g. anneau, arbre). Ils doivent aussi connaître quelques informations concernant leurs voisins et leurs contenus sans avoir la capacité de choisir ces voisins. Souvent, ces informations sont utilisées pour guider le routage des requêtes/messages dans le système.

2.3.2.3. Systèmes P2P super-pairs

Certains chercheurs trouvent que l'hypothèse des rôles équivalents des nœuds, en environnement P2P, représente une forte contrainte. Pratiquement, les nœuds dans un système ne possèdent ni la même puissance de calcul ni la même capacité de stockage. Alors, il est mieux d'attribuer aux nœuds des rôles convenables à leurs puissances et à leurs capacités. Pour cette raison, une topologie hybride entre les deux paradigmes P2P et Client/Serveur a été proposée afin de profiter des avantages de chaque paradigme. Dans une telle topologie, des nœuds puissants, appelés super-pairs, jouent le rôle d'un serveur et les autres nœuds jouent le rôle d'un client. Les super-pairs s'organisent selon la topologie structurée ou non-structurée. Le fait d'avoir plusieurs super-pairs facilite la gestion des ressources partagées dans le système. Les super-pairs peuvent effectuer des tâches complexes telles que l'indexation, l'évaluation de requêtes, le contrôle d'accès et la gestion de méta-données. Les systèmes P2P super-paris sont moins tolérants aux pannes par rapport aux autres types de systèmes P2P. Lorsqu'un super-pair tombe en panne, ses clients deviennent privés de ses services et isolés du reste du ststème. Pour résoudre ce problème, les super-pairs peuvent être dynamiquement remplacés en présence d'une panne. Rien n'empêche qu'une élection de nouveau super-pair soit fondée, par exemple, sur la disponibilité, la capacité de calcul ou la bande passante réseau.

2.3.3. Types des requêtes évaluées en environnement P2P

La majorité des systèmes P2P existants dans la vie quotidienne sont des systèmes de partage de fichiers. Dans un tel environnement on trouvent un nombre important (des

milliers) de nœuds et un très grand nombre de fichiers. Etant donné le nom d'un fichier recherché par l'utilisateur, le système cherche les nœuds stockant ce fichier et envoie les adresses IPs de ces nœuds à l'utilisateur afin de choisir un nœud et de télécharger ensuite le fichier. Habituellement, le nom du fichier est connu par la majorité des utilisateurs dans le système. Cette connaissance est obtenue via le média (e.g. TV) ou via l'univers social des utilisateurs (e.g. l'université, l'école). Ainsi, cette connaissance permet d'avoir une sémantique commune concernant les noms de fichiers partagés ce qui mène à un échange compréhensif de fichiers. C'est-à-dire, que lorsqu'un nœud répond à une requête, il y a une grande probabilité que ce nœud envoie le fichier souhaité par le nœud initialisant la requête. Par conséquent, les requêtes dans ce type de partage de données sont des requêtes "simples". Elles consistent à chercher les adresses IPs des nœuds stockant le fichier demandé en se basant sur le nome du fichier. Dans ce type de requêtes, on ne traite pas l'hétérogénéité sémantique entre les noms des fichiers partagés. Si le fichier obtenu par ce type de traitement n'est pas celui demandé, dans ce cas-là le nœud initialisant la requête choisit un autre nœud afin de télécharger le fichier demandé. Par conséquent, il faut exécuter une autre fois la requête ce qui nécessite de consommer plus des ressources disponibles dans le système. La phase la plus délicate dans le partage de fichiers est le routage de requêtes vers les nœuds stockant les fichiers demandés. Les systèmes P2P offrent de "bonnes" techniques permettant d'effectuer un routage de requêtes relativement efficace en termes de consommation de la bande passante réseau et de temps nécessaire pour le routage.

Les approches utilisées pour le partage des données réparties sur l'Internet sont capables d'évaluer des requêtes plus complexes que celles soumises dans les systèmes P2P de partage de fichiers. Cependant, elles sont limitées du point de vue du nombre de sources de données partagées tout en garantissant une évaluation efficace de requêtes. Récemment, une nouvelle tendance est apparue pour dépasser ces limites. Les chercheurs dans le domaine de gestion de données essaient de combiner les techniques P2P et celles utilisées dans les bases de données. Cela permet aux systèmes P2P de partager de données ayant une granularité plus fine (i.e. un attribut atomique) et d'offrir la capacité de traiter des requêtes complexes (i.e. SQL). La difficulté de trouver une solution pour évaluer des requêtes complexes mène aux travaux de recherche abordant des types spécifiques de requêtes. Avant d'expliquer, plus tard dans ce chapitre, une partie ''importante'' de ces travaux, il est mieux de citer les différents types de requêtes.

(i) **Requêtes d'intervalle**[11]**:** ce sont les requêtes par lesquelles l'utilisateur demande des données ayant des valeurs appartenant à un intervalle précis. Les auteurs de [GAA03, GS04, HJS+03, SGA+02] ont étudié ce type de requêtes en environnement P2P.

(ii) **Requêtes meilleur- k**[12] **:** dans ce type de requête, l'utilisateur peut être satisfait des meilleur-k réponses trouvées dans le système. Ce type de requêtes attire beaucoup d'attention dans plusieurs domaines de l'informatique comme la recherche d'informations [MTW05, BNS+05], Monitoring de systèmes et de réseaux [BO03, KOT+04, CW04], bases de données multimédias [CGM04, DMN+02], analyse de données spatiales [BBK01, HS03]. Récemment, ce type de requêtes est utilisé dans le partage de données dans les systèmes P2P [APV06, BAH+06].

(iii) **Requêtes d'agrégation**[13] **:** ce sont des requêtes contenant un opérateur d'agrégation comme la requête suivante : Select Op_Agr(Col) From R Where conditon-selection ; Op-Agr est un opérateur d'agrégation comme : Sum, Count, AVG etc. Col est un attribut numérique de la relation R ou une expression contenant plusieurs attributs. La condition-selection décide quels sont les tuples à être présentés dans l'agrégation.

(iv) **Requêtes de jointure**[14] : ce sont de requêtes multi-relations multi-attributs dans lesquelles il y a un ou plusieurs opérateurs de jointure.

(v) **Requêtes complexes :** ce sont des requêtes constituées des combinaisons des requêtes précédemment citées ci-dessus. e.g. requêtes SQL générales (sans aucune spécification).

Les données interrogées par ces types de requêtes sont souvent des données représentées par des schémas hétérogènes. Cette hétérogénéité est due à l'autonomie de nœuds qui caractérise les environnements P2P. Donc, au moment de l'évaluation de ces requêtes, nous avons besoin de faire un *matching* de schémas permettant de créer des règles de correspondance et servant à traduire une requête donnée entre les différents schémas.

Par conséquent, le routage de requêtes et le *matching* de schémas sont des problèmes abordés par les chercheurs en bases de données pair-à-pair (P2P Databases) afin de localiser les sources des données désirées par une requête. Dans les deux sections suivantes, nous

[11] Range Queries
[12] Top-k Queries
[13] Aggregation Queries
[14] Join Queries

présentons une partie des principaux travaux de recherche effectués pour résoudre ces deux problèmes.

2.4. Routage de requêtes en environnement P2P

Nous remarquons que les méthodes de routage de requêtes dépendent beaucoup des topologies des systèmes P2P; autrement dit des types de systèmes P2P. Dans la suite, nous discutons une partie importante de ces méthodes au sein de chaque type des systèmes P2P.

2.4.1. Routage de requêtes dans les systèmes P2P non-structurés

Afin d'envoyer une requête vers les sources pertinentes de ses données, le nœud initialisant la requête (NIR) envoie des messages identiques à ses voisins. Chaque message contient la requête, l'identifiant du NIR et une valeur du paramètre TTL (Time To Live). Ce paramètre représente le nombre maximum de sauts. C'est-à-dire, la longueur du chemin de chaque message en termes de nombre de nœuds parcourus dans le système. Après avoir reçu le message par un nœud, ce nœud exécute la requête et envoie la réponse au NIR. De plus, il diminue la valeur de TTL par 1. Si la valeur de TTL devient 0, ce nœud détruit le message, sinon le nœud envoie, à son tour, le message contenant la nouvelle valeur de TTL à ses voisins. En continuant ce processus, tous les nœuds situés à une distance ayant une valeur inférieure à la valeur initiale de TTL vont recevoir le message. Les autres nœuds dans le système ne reçoivent pas le message même s'ils possèdent de réponses valides. Le système de partage de fichiers Gnutella[15] était à l'origine de cette approche qui garde un degré élevé d'autonomie de nœuds. Cependant, cette approche consomme beaucoup de ressources, surtout de la bande passante réseau. Un nombre important de messages échangés entre les nœuds est nécessaire pour le routage de chaque requête. Cela pourrait inonder le réseau et empêcher le système utilisant cette approche de passer à grande-échelle.

Nous remarquons qu'avant qu'un nœud envoie un message à l'un de ses voisins, aucune information concernant le contenu de ce voisin n'est disponible. Pour cette raison, cette approche de routage est appelée : le routage aveugle. Plusieurs solutions ont été proposées pour améliorer la performance de cette approche. Dans [KGZ02], les auteurs proposent une solution pour diminuer le nombre de messages échangés. Le principe de cette solution est d'envoyer le message reçu par un nœud à une partie de ses voisins au lieu de l'envoyer à tous les voisins. Le choix de cette partie est effectué au hasard. Alors, cette

[15] Gnutella homepage, http://www.gnutella.fr

solution diminue le nombre de messages échangés. Cependant, le fait de ne pas envoyer le message à tous les voisins pourrait ignorer beaucoup de réponses valides.

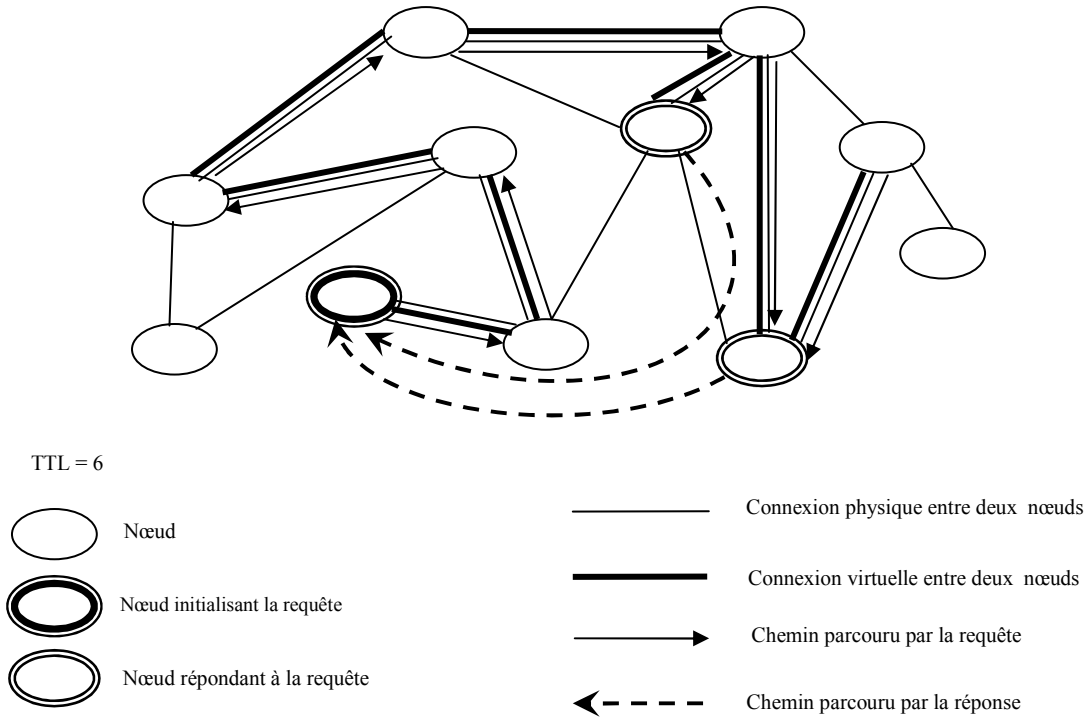


Fig. 1. Routage de requêtes dans un système P2P non-structuré

Une autre solution plus intelligente a été proposée dans le même papier [KGZ02], l'idée de base est d'utiliser des statistiques concernant des requêtes auxquelles on a déjà répondu, appelées anciennes requêtes. Chaque nœud stocke pour une ancienne requête, les réponses venant de chaque voisin. Pour une requête donnée, le NIR compare cette requête avec les anciennes requêtes qu'il stocke. S'il trouve que cette requête ressemble (selon des critères bien définis) à une des anciennes requêtes, il choisit une partie de voisins répondant à cette requête en se basant sur le nombre de réponses venant d'eux. Chaque voisin ayant une réponse envoie un message suivant le chemin inverse de celui parcouru par le message initial. Ceci permet de mettre à jour les statistiques sur les nœuds par lesquels le message initial a été passé. Selon ce mécanisme, c'est vrai qu'on ne choisit pas une partie de voisins au hasard. Mais, c'est vrai aussi qu'on produit beaucoup de messages de mise à jour afin de modifier les statistiques concernant non seulement les anciennes requêtes mais aussi les nœuds. Les auteurs de [LCC+02] proposent l'idée de choisir k voisins au hasard mais chaque

voisin doit choisir un seul voisin au hasard aussi au lieu de choir un sous-ensemble. Dans ce cas, le nombre de messages produits est k*TTL au maximum. Les inconvénients de cette idée est qu'on ignore des réponses valides et qu'on ne peut pas obtenir de statistiques concernant des anciennes requêtes.

Une méthode itérative a été proposée dans [YG02] pour améliorer la performance de l'approche du routage aveugle. Supposant que TTL= V_{max}, cette méthode consiste à choisir plusieurs valeurs à TTL entre 1 et V_{max}. Pour chacune, on répète l'approche du routage aveugle. Si, pour une valeur de TTL inférieure à V_{max}, on arrive à trouver un nombre satisfaisant de réponses, on s'arrête de répéter cette approche. La méthode proposée améliore la performance de l'approche du routage aveugle en termes de nombre de messages échangés mais le temps de réponse de cette méthode est plus élevé.

Les auteurs de [TR03] ont proposé un mécanisme basé sur l'idée de donner une valeur de probabilité à chaque voisin afin de choisir les voisins auxquels il faut envoyer le message. Toujours, il y a l'inconvénient de l'ignorance de réponses valides. De plus, si le système est très dynamique en termes de l'entrée et de la sortie de nœuds, les probabilités des nœuds changent beaucoup ce qui a un impact direct sur la validité des réponses. Les auteurs de [YG02] et [CG02] ont proposé le mécanisme d'Indices locaux selon lequel chaque nœud indexe les données stockées sur les nœuds se trouvant dans une surface ayant r comme diamètre. Ce mécanisme améliore relativement le temps de réponse de l'approche de routage aveugle. Cependant, lorsqu'un nœud se connecte/se déconnecte au/du système, une inondation de messages d'un diamètre de *r* nœuds peut être apparue lors de la mise à jour des indices locaux des autres nœuds.

Les auteurs de [MK02] ont proposé un protocole pour la localisation de ressources réparties (*Distributed Resource Location Protocol*). Ils utilisent ce protocole pour localiser les sources de données. Le principe de ce protocole est que chaque nœud doit indexer toutes les données stockées sur un nœud ayant déjà répondu à une ancienne requête. Cette solution est bonne pour les requêtes répétitives. Par contre pour une requête quelconque ce n'est pas le cas. Un des inconvénients de ce protocole est le nombre important de messages de mise à jour lors de la sortie d'un nœud du système.

L'inefficacité du routage des requêtes en termes de nombre important de messages échangés et la mauvaise qualité des réponses dans les systèmes P2P non –structurés, malgré les grands efforts menés dans cette direction, reflètent la difficulté de réaliser un routage efficace de requêtes tout en conservant un niveau élevé d'autonomie pour chaque nœud. Les

réponses à une requête dans les systèmes P2P non-structurés sont souvent incomplètes. C'est-à-dire, certaines réponses valides ne sont pas retrouvées même si elles existent dans le système. Une des solutions à cette limite est d'envoyer la requête à tous les nœuds. Cependant, dans un environnent P2P, cette solution n'est pas pratique ; elle mène à l'inondation du réseau sur lequel se base le système P2P.

2.4.2. Routage de requêtes dans les systèmes P2P structurés

Dans les systèmes P2P structurés, les nœuds s'organisent sous une forme géométrique bien précise. La façon dans laquelle un système P2P structuré localise les sources de données dépend de sa forme géométrique. Dans la suite, nous discutons quelques exemples de formes géométriques selon lesquelles les nœuds peuvent s'organiser et nous discutons également les impacts de ces formes sur le routage de requêtes dans ce type de systèmes. Ces formes se différencient l'une des autres par la manière par laquelle un nœud choisit ses voisins et de la façon dont il en partage des informations. Dans la suite, nous présentons ces aspects pour des systèmes ayant la forme d'un anneau, d'un espace cartésien de d dimensions, d'un arbre et d'un hypercube.

(i) Anneau

Selon cette géométrie, les nœuds doivent être ordonnés sur un anneau virtuel. Chaque nœud a un identifiant sur cet anneau. Plusieurs systèmes utilisent cette géométrie comme [RD01, SMK+01, CLM+04, ZHS+04]. Une comparaison entre ces systèmes est déjà publiée dans [MKL+02]. Dans la littérature on constate l'utilisation de la technologie de DHT (*Distributed Hash Table*) qui consiste à stocker les informations partagées dans une table de hache distribuée sur tous les nœuds. La DHT est utilisée pour éviter l'inondation de réseau en guidant le routage de requêtes et pour garantir de trouver la majorité des réponses valides existantes dans le système. Les informations de la DHT sont réparties sur les nœuds selon une fonction de hachage connue par tous les nœuds. On donne des identifiants aux nœuds et on représente chaque entité de données (objet, fichier, relation, etc.) partagée par une clé créée en utilisant la même fonction de hachage. Chaque nœud stocke les couples (clé, identifiant) concernant les clés dont il est responsable dans sa partie de DHT. Le protocole Chord [SMK+01] (désormais Chord) est un bon exemple de ces systèmes. Dans Chord, chaque nœud a un identifiant sur m bits, et le responsable d'une clé c est le nœud ayant le premier identifiant qui est égal à ou succède c. La partie de DHT d'un nœud p contient m

entrées. Le $i^{ème}$ entrée contient l'identifiant du nœud succédant p par au moins 2^{i-1}. i.e. successeur $(p+ 2^{i-1})$.

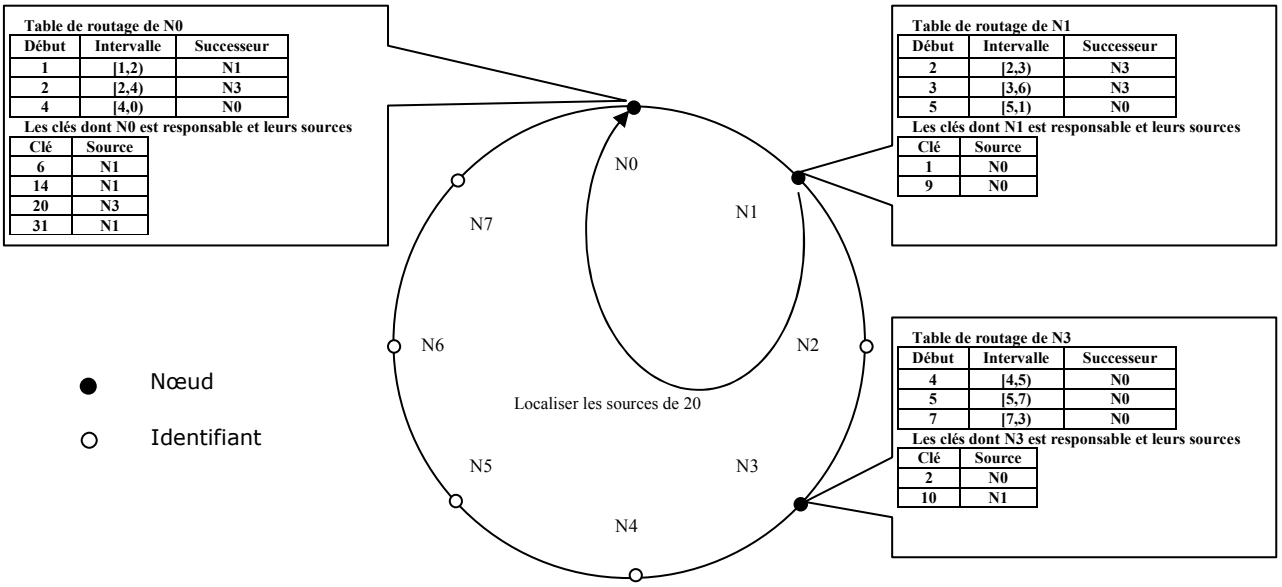


Fig. 2. Exemple du protocole Chord

Chord permet de localiser le responsable d'une clé en effectuant un routage ayant une complexité de l'ordre de *O(LogN)*, où N est le nombre de tous les nœuds dans le système. Selon la géométrie d'anneau, les nœuds ne choisissent pas leurs voisins, c'est le rôle du système d'attribuer à chaque nœud ses voisins.

Un algorithme de routage permet de localiser le nœud responsable de la clé représentant l'entité de données recherchée. Ce nœud connaît les identifiants des hôtes des sources de cette entité de données. Afin d'illustrer le routage de requêtes dans Chord, nous considérons le cas d'un anneau avec des identifiants ayant m = 3 bits. Dans la figure 2, nous supposons que le nœud N1 a besoin de localiser les sources de la clé 20. Il doit consulter sa table de routage. Au lieu de chercher la clé 20 il cherche la clé 4 (car *20 mod (8) = 4*). Après cette consultation, il sait qu'il faut envoyer la demande de localisation au nœud N0. Le nœud N0 sait que l'entité de données représentée par la clé 20 est stockée sur N3. Il peut informer N1 de cette information. Si N1 demande d'obtenir l'entité de données représentée par la clé 20, dans ce cas là, le nœud N0 envoie la requête au nœud N3 qui, à son tour, répond à la demande de localisation en envoyant l'entité demandée à N1.

(ii) Espace cartésien de *d* dimensions

Cette géométrie est basée sur une décomposition d'un espace de d dimensions en un ensemble de zones séparées. On attribue chaque zone à un nœud. Les systèmes P2P utilisant le réseau virtuel CAN (Content Addressable Network) [RFH01] sont basés sur cette géométrie. Dans CAN, chaque nœud est responsable d'un espace de stockage (sa zone). C'est-à-dire que ce nœud stocke toutes les paires (clé, identifiant) dont les clés tombent dans sa zone selon la fonction de hachage utilisée par le système. L'espace de stockage est un espace cartésien de d dimensions. Cet espace est indépendant du réseau physique sous-jacent. Chaque clé, dans CAN, est représentée sous forme de point ayant d coordonnés c = (c1, c2, .. cd).

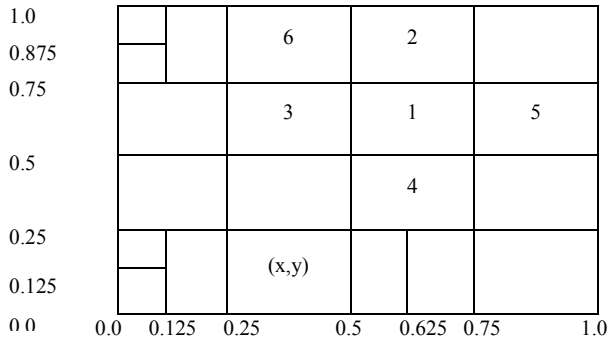
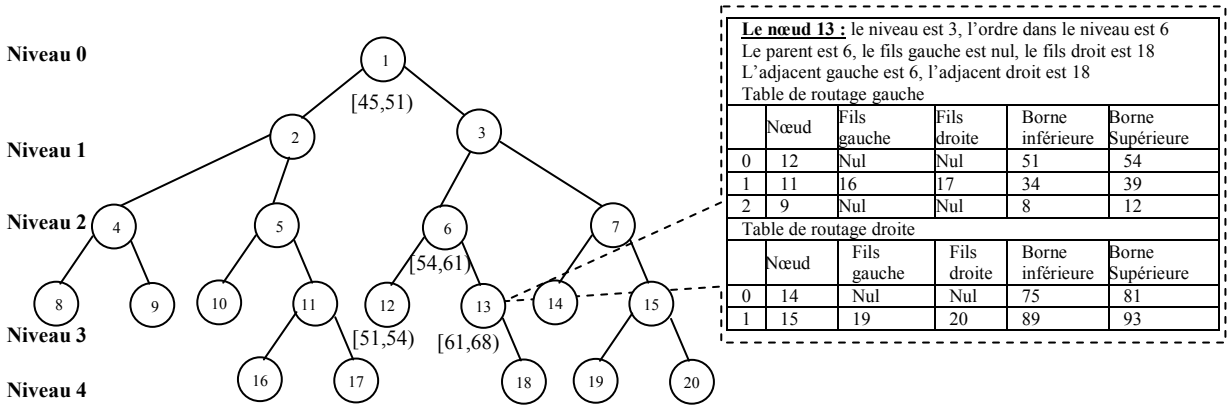


Fig. 3. Exemple de CAN

Un algorithme glouton (*Greedy Algorithm*) est utilisé pour le routage d'une requête donnée vers les voisins ayant les coordonnées les plus proches de la zone dans laquelle la clé recherchée est tombée. Par exemple, dans la figure 3, nous considérons un espace cartésien de d=2 dimensions et nous supposons qu'un nœud possédant une zone i est appelé Ni. Supposons que N1 a besoin de localiser les sources de l'entité de données représentée par la clé (x,y). Ce nœud envoie sa requête à N4 car ce nœud possède la zone la plus proche, en termes de distance euclidienne, de (x,y). Ensuite, le nœud N4 envoie la requête vers le voisin possédant la zone la plus proche à la zone du nœud responsable de (x,y). Ce processus est continué jusqu'à ce que la requête arrive à sa destination qui est un nœud possédant la zone dans laquelle tombe (x,y). Selon ce type de système, chaque nœud doit stocker des informations concernant 2d voisins et le nombre moyen de sauts pour localiser le responsable d'une clé est de $O(dN^{1/d})$ où N est le nombre total de nœuds dans le système.

(iii) Arbre

Comme exemple de cette géométrie, nous prenons le cas de Baton [JOV05] qui est utilisé pour concevoir une structure d'index distribué en répartissant les données sur les nœuds. Chaque nœud dans l'arbre représente un nœud du système. Il a un identifiant logique correspondant à son niveau et à son ordre dans ce niveau et un identifiant physique



Le nœud 13 : le niveau est 3, l'ordre dans le niveau est 6
Le parent est 6, le fils gauche est nul, le fils droit est 18
L'adjacent gauche est 6, l'adjacent droit est 18
Table de routage gauche

	Nœud	Fils gauche	Fils droite	Borne inférieure	Borne Supérieure
0	12	Nul	Nul	51	54
1	11	16	17	34	39
2	9	Nul	Nul	8	12

Table de routage droite

	Nœud	Fils gauche	Fils droite	Borne inférieure	Borne Supérieure
0	14	Nul	Nul	75	81
1	15	19	20	89	93

Fig. 4. Exemple de l'arbre de Baton

correspondant à son adresse IP. De plus, chaque nœud garde un lien (en utilisant les identifiants physiques) avec son parent, son voisin gauche, son voisin droit, ses fils et autres voisins sélectionnés dans le même niveau que le sien. Les voisins sélectionnés sont maintenus dans deux tables de routage, une pour les voisins droits et l'autre pour les voisins gauches. Le nombre d'entrées de chacune est O(log N) ou N est le nombre de nœuds dans le système. La $i^{ième}$ entrée dans la table à droite (à gauche) d'un nœud n contient un lien avec le nœud ayant le numéro $n+2^{i-1}$ (respectivement $n-2^{i-1}$) dans le même niveau. Les données sont redistribuées sur les nœuds d'une façon que chaque nœud devient responsable d'un intervalle de valeurs. Dans la figure 4, le nœud 6 est le parent des nœuds 12 et 13 alors son intervalle ([54, 61)) doit être au milieu de leurs intervalles ([51,54) et [61,68) respectivement).

Le routage d'une requête d'égalité se fait de la façon suivante. Si les voisins ayant les bornes supérieures (inférieures) de leurs intervalles plus petits (plus grands) que celles de l'intervalle de la requête, la requête est envoyée horizontalement vers le voisin le plus loin qui a une borne supérieure (inférieure) plus petite (plus grande) que les valeurs de l'intervalle de la requête. Ensuite, la requête est propagée verticalement en suivant le voisin

gauche (droit) ou le fils gauche (droit). Ce processus est répété jusqu'à ce que la requête arrive sur les sources de données souhaitées. Le nombre de sauts à effectuer pendant ce routage est O(LogN). En ce qui concerne les requêtes d'intervalle, le processus précédent est répété jusqu'à ce qu'on trouve le nœud qui stocke la borne inférieure (ou supérieurs) de l'intervalle de la requête, après il faut chercher le reste de l'intervalle. Pour cela, la requête doit visiter encore x nœuds. Par conséquent, la complexité du routage devient $O(\log N + x)$.

(iv) Hypercube

HyperCup [SSD+02] est un bon exemple de cette forme géométrique. Selon HyperCup, les nœuds sont organisés dans un hypercube ayant un diamètre[16] $\Delta = O(\log_b N)$, où b est la base du hypercube (c'est-à-dire que dans chaque dimension il y b nœuds) et N est le nombre de nœuds dans la structure. Le nombre de dimensions est (D+1) où D est donnée comme suit : $N= b^{D+1}$. Chaque nœud a (b-1) (D+1) voisins. Les liens entre les nœuds sont libellés par des numéros entre 0 et b. Ces libelles permettent d'ordonner les voisins d'une façon symétrique. C'est-à-dire, lorsque le lien entre deux nœuds X et Y est égal à i, cela signifie que le nœud X est le $i^{ème}$ voisin du nœud Y et le nœud Y est le $i^{ème}$ voisin du nœud X. Dans la figure 5, nous présentons un hypercube à la base de b = 2. Nous remarquons que le diamètre est $D = \log_2 8 = 3$ et que chaque nœud a 3 voisins.

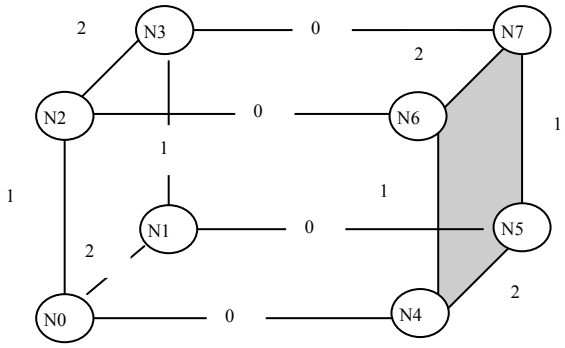


Fig. 5. Exemple de HyperCup

Le routage d'une requête dans le hypercube se fait en envoyant la requête vers les voisins avec un TTL. Lorsqu'un nœud reçoit une requête venant du voisin j, il l'envoie vers les voisins ayant des libelles plus grands que j. Par exemple, lorsque le nœud N0 envoie une requête aux voisins N4, N2 et N1, le nœud N4 envoie la requête aux nœuds N5 et N6. Le nœud N1 n'envoie pas la requête parce que leurs voisins N3 et N5 ayant un numéro inférieur à 2 qui est le numéro du voisin N0. Quant au nœud N2, il envoie la requête au nœud N3. La

[16] La plus longue distance entre n'importe quels deux nœuds

troisième étape est que N6 envoie la requête à N7. Cette méthode de routage permet de localiser un objet recherché en $O(\log_b N)$ sauts.

Comparaison

Topologie	Algorithme	Paramètres	Complexité de routage	Complexité de l'entrée et de la sortie	DHT
Anneau	Chord	*N est* le nombre de nœuds	*LogN*	Log^2N	*Oui*
Espace Cartésien de d dimensions	CAN	*N est* le nombre de nœuds d le nombre de dimensions de CAN	$dN^{1/d}$	*Entrée* $d/2N^{1/d}$ *Sortie* $2d$	*Oui*
Arbre	Baton	*N est* le nombre de nœuds x est le nombre de nœuds à contacter après avoir trouvé la première partie de l'intervalle demandé par la requête	*LogN et LogN+x pour une requête d'intervalle*	*Log N*	*Non*
Hypercube	HyperCup	*N est* le nombre de nœuds b est la base du Hypercube	Log_bN	Log_bN	*Non*

Fig. 6. Comparaison entre les systèmes P2P structurés étudiés

Dans la figure 6, nous comparons les quatre systèmes structurés étudiés ci-dessus. Du point de vue de Complexité de routage**,** le meilleur est HyperCup et le pire est CAN. Cependant, HyperCup utilise le principe de TTL (Time To Live) qui ne permet pas d'obtenir toutes les réponses valides existantes dans le système. Dans Baton, les adresses physiques jouent un rôle essentiel dans le partage de requêtes. Ce système ne sépare pas le réseau virtuel du réseau physique sous-jacent. Une autre limite est que seulement les noeuds formant les feuilles de l'arbre peuvent quitter le système à n'importe quel moment. Cependant, si un autre nœud veut quitter le système il doit trouver un nœud feuille pour qu'il le remplace. C'est-à-dire, les deux nœuds doivent échanger leurs identifiants y inclus leurs adresses IPs. Ceci est pratiquement difficile à atteindre. On peut aussi ajouter que dans le cas de Baton, il faut redistribuer les données sur tous les nœuds selon leurs valeurs et cela n'est pas pratique lorsque le volume de données partagées est très grand. Il est plus pratique de redistribuer des informations concernant les placements de données et d'accéder aux données sur leurs sources comme dans le cas de Chord. Par conséquent, Baton est convenable pour construire un index de données totalement distribué dans des applications où on peut déplacer les données. Nous remarquons que du point de vue de Complexité de l'entrée et de la sortie, les deux systèmes Baton et HyperCup sont plus efficaces que les systèmes CAN et Chord. Cela est dû au fait que Baton et HyperCup ne possèdent pas de DHT.

2.4.3. Routage de requêtes dans les systèmes P2P super-pairs

Nous rappelons que dans les systèmes P2P super-pairs, il y a deux types de nœuds : (i) les super-pairs qui sont souvent des nœuds puissants et (ii) les nœuds normaux, ou tout simplement les nœuds, qui forment des groupes dont chacun est relié à un super-pair. Chaque super-pair est responsable d'indexer les données stockées sur les nœuds y reliés. Les super-pairs entre eux forment un système P2P. Ce système peut être structuré ou non. Nous avons déjà constaté que ce type de systèmes est hybride entre les deux paradigmes Client/Serveur et P2P. Dans les systèmes P2P super-pairs, le routage de requête est effectué de la manière suivante. Le NIR doit envoyer un message contenant la requête à son super-pair. Ce dernier cherche dans son index local s'il trouve les sources de données souhaitées, il envoie la requête aux hôtes des sources de données pour être évaluée par ces hôtes sinon il envoie le message à ses voisins dans le système P2P constitué des super-pairs. Chaque super-pair cherche dans son index local, s'il trouve les hôtes des sources de données, il envoie le message aux hôtes et obtient, ensuite, les réponses afin de les retourner au super pair auquel le NIR est relié.

Discussion

Les approches de routage de requêtes citées ci-dessus sont convenables pour le partage de fichiers et pour le partage de données dont la sémantique de données partagées est connue par tous les nœuds. Elle ne peut pas être appliquée directement (sans aucune extension) dans le cas des bases de données représentées par des schémas locaux hétérogènes. En environnement P2P, l'efficacité du routage de requêtes dépend de la topologie du système P2P considéré et de l'algorithme utilisé pour le routage de requêtes. Nous trouvons que l'autonomie de nœuds est mieux respectée dans les systèmes P2P non-structurés. Par contre, malgré les différents travaux de recherche menés pour améliorer l'évaluation de requêtes dans ce type de systèmes, les solutions proposées restent limitées. Le grand nombre de messages échangés lors de la localisation de sources de données et l'ignorance de grand nombre de réponses valides existantes dans le système sont les principaux inconvénients de ces systèmes. Les systèmes P2P structurés sont capables de retourner un nombre important des réponses valides existantes tout en maintenant un nombre relativement petit de messages échangés lors de la localisation de sources de données. Les systèmes super-pairs sont des systèmes hybrides entre les deux paradigmes Client/Serveur et P2P. L'évaluation de requêtes dans ces systèmes est simple et efficace

lorsque les hôtes des sources de données et le NIR ayant le même super-pair. Cependant, dans le cas inverse, l'efficacité de ce processus dépend du type du système P2P reliant les super-pairs et de la méthode de localisation utilisée par ce système. Les systèmes P2P super-pairs sont moins tolérants aux pannes par rapport aux autres types de systèmes P2P. Lorsqu'un super-pair tombe en panne, tous ses clients seront privés des services fournis par le système.

2.5. *Matching* de schémas en environnement P2P

En environnement P2P, lors de l'évaluation de requêtes complexes, un routage efficace de requêtes ne suffit pas pour localiser les sources de données pertinentes. Dans un tel environnement, les bases de données sont conçues et développées d'une façon autonome. Chaque nœud décrit ses données en utilisant son propre schéma local. A cause de cette autonomie dans la conception de bases de données, les schémas locaux pourraient être sémantiquement et structurellement hétérogènes. Le problème de l'hétérogénéité de schémas a un impact direct sur la localisation de sources de données. À cause de ce problème, une source de données peut être ignorée lors du routage de requête même si cette source est une source pertinente. Elle pourrait être, dans certains cas, la seule source pertinente disponible dans le système. Afin de localiser les sources de données pertinentes, il faut résoudre ce problème. Lorsqu'un nœud soumet une requête écrite selon son schéma local, les autres nœuds doivent comprendre la requête afin d'être capables d'y répondre.

Le *matching* de schémas est le processus permettant de trouver pour chaque élément d'un schéma, son correspondant dans un autre schéma et d'écrire des règles de correspondance (RC) permettant de reformuler une requête écrite selon l'un des deux schémas en termes de l'autre schéma.

2.5.1. Types de *Matching*

Le *matching* de schémas est un processus ayant un impact sur l'évaluation de requêtes. Si les règles de correspondance sont mal définis, la qualité de réponses retenues sera relativement mauvaise. C'est-à-dire, que certaines réponses peuvent être invalides ou ne peuvent pas être celles souhaitées par le nœud initialisant la requête. A ce stade, des questions importantes peuvent se poser comme : (i) comment créer le *matching* de schémas ? (ii) à quel moment ? et (iii) par qui, par des experts ou par des utilisateurs finaux (en anglais, End Users) ?

Etant donné que ce n'est pas possible d'effectuer le *matching* de schémas d'une façon totalement dynamique (sans intervention humaine) [Hal05, Aum05], le *matching* de schémas peut être effectué manuellement (sans aucune intervention de machine) ou d'une façon semi-automatique. Le *mathcing* manuel de schémas est coûteux en termes de temps nécessaire pour le faire. Des approches semi-automatiques sont apparues comme solution à cet inconvénient du *matching* manuel. Les chercheurs essaient toujours d'automatiser le plus possible le *matching* de schémas. Une synthèse des approches proposées dans ce domaine est présentée dans [ShE05].

Nous pouvons constater l'existante de deux types de *matching* de schémas. Le premier consiste à créer des règles de correspondance durant la phase de conception du système considéré. Afin d'être utilisées lors de l'exécution de requêtes, les règles de correspondance peuvent être stockées d'une façon centralisée sur un ou plusieurs sites connus par tous les nœuds ou d'une façon distribuée sur tout les nœuds. Ces règles sont souvent créées par des spécialistes de la conception de schémas qui connaissent la sémantique commune de schémas reliés par les règles de correspondance. Lorsque les règles de correspondance sont stables et ne sont pas modifiées (sauf, bien sûr en raison de maintenance), nous appelons ce type de *matching* par *matching* statique. Dans le cas contraire, lorsque les règles de correspondances sont souvent modifiées, nous appelons ce type de *matching* par *matching* dynamique. Dans ce type de *matching*, on peut modifier les règles de correspondance et/ou créer à la volé de nouvelles règles à n'importe quel moment et même au moment de la soumission de requêtes.

2.5.2. Approches de *Matching* de schémas en environnement P2P

En environnement P2P, les approches de *maching* de schémas peuvent être classifiées comme suit : approche basée sur un schéma global, approche basée sur *matching* deux-à-deux et approche basée sur le principe de recherche d'information (RI).

(i) Approche basée sur un schéma global

Certains chercheurs trouvent que l'hypothèse de ne pas avoir un schéma global dans les systèmes P2P est une contrainte très forte. Même si l'instabilité de nœuds et la grande échelle empêchent d'avoir des informations concernant le placement de données, les nœuds peuvent être d'accord sur un schéma global. Deux approches LAV (local-as-view) [LRO96] et GAV (global-as-view) [MPQ+97] existent pour créer le *matching* entre le schéma global et les schémas locaux. Selon l'approche (LAV), les schémas locaux sont définis comme vues

(requêtes) formulées en termes du schéma global. Une requête écrite selon le schéma global doit être reformulée en termes des schémas locaux. Ceci est un problème relativement difficile. Par contre, dans l'approche (GAV), le schéma global est défini comme une vue sur les schémas locaux. Selon cette approche, on traduit la requête (écrite en termes de schéma global) en sous-requêtes écrites selon les schémas locaux. Cette approche est relativement simple. Dans GAV si on effectue des modifications sur le schéma local, il faut modifier le schéma global. Cette approche est convenable pour des applications dans lesquelles tous les schémas locaux sont connus dès le début et ils ne sont pas modifiés. Pour les raisons précédentes, l'approche GAV n'est pas convenable aux environnements P2P. Dans LAV, les correspondances entre un schéma local et le schéma global peuvent être effectuées localement. Pour cette raison, l'approche LAV est mieux adaptée aux environnements distribués à grande-échelle.

(ii) Approche basée sur le *matching* deux-à-deux

L'utilisation d'un schéma global pourrait être inacceptable pour certaines applications dans lesquelles, il est très difficile de trouver un accord commun entre les nœuds sur un tel schéma. Pour cette raison, certains chercheurs préfèrent utiliser un *matching* direct entre les schémas locaux deux-à-deux (*Pairwise Matching*). De cette façon, on crée un réseau, appelé réseau sémantique, dont les nœuds sont les schémas locaux et les liens sont les règles de correspondance.

(iii) Approche basée sur le principe de recherche d'informations

Selon cette approche, l'utilisateur doit jouer un rôle essentiel dans le processus de *matching* de schémas en fournissant des mots-clés. Cette approche ressemble beaucoup à celle utilisée dans la recherche d'information (RI) sur l'Internet avec n'importe quel moteur de recherche. Le principe de recherche d'informations (RI) consiste à (i) fournir des mots clés par l'utilisateur ; (ii) trouver les sites stockant les données ayant la sémantique la plus proche de celle des mots clés et les délivrer à l'utilisateur; (iii) sélectionner les sites pertinents par l'utilisateur afin de télécharger les données souhaitées de ces sites.

Après aborder les sujets du routage de requêtes et de *matching* de schémas, un autre sujet important dans l'évaluation de requêtes en environnement P2P, c'est celui de l'optimisation de requêtes. Dans la section suivante, nous abordons ce sujet.

2.6. Optimisation de requêtes

L'optimisation de requêtes et une phase responsable de générer, pour une requête soumise, un plan d'exécution proche de l'optimal. Cette phase nécessite deux composants : (i) un optimiseur qui représente le raisonnement et (ii) un modèle de coûts qui représente la connaissance. Dans un plan d'exécution, l'optimiseur définit, en s'appuyant sur le modèle de coûts, en détailles : (i) le choix des sources de données pertinentes, (ii) l'ordre des opérations de la requête et le meilleur algorithme pour chaque opération et (iii) pour chaque opération, le choix des nœuds adéquats pour l'exécuter. Le modèle de coût est constitué (i) des statistiques sur les données et sur les sites stockant ces données et (ii) des formules de coûts. L'optimiseur définit des plans d'exécution dont il choisit l'un le plus proche de l'optimal qui minimise une fonction de coûts (e.g. temps de réponse). Plus les statistiques sont à jour, plus le modèle de coût est précis et le plan d'exécution, par conséquent, est proche de l'optimal. Plus un plan d'exécution proche de l'optimal, plus l'exécution de requêtes est efficace. Par "efficace", nous désignons une exécution correspondante à une valeur minimisée de la fonction de coûts.

Dans les systèmes de bases de données traditionnels, les statistiques du modèle de coûts sont stockées dans un catalogue global qui peut être centralisé ou dupliqué sur plusieurs serveurs. Dans les systèmes P2P de partage de données, à cause de la grande échelle et l'instabilité des nœuds, ce n'est pas pratique d'avoir un catalogue global. Un catalogue centralisé peut créer un goulet d'étranglement. Un catalogue dupliqué peut créer un problème de mise à jour car dans un environnement instable où un nombre important de requêtes peut être existant à chaque moment, ce n'est pas pratique de passer tout le temps pour effectuer la mise à jour des statistiques. Pour les mêmes raisons, dans les catalogue locaux il peut se trouver des statistiques obsolètes et un manque de certaines statistiques concernant certaines données dans le système. Par conséquent, le modèle de coût peut être imprécis et cela peut mener à une génération d'un plan d'exécution sous-optimal.

Une des solutions au problème précédent est de localiser les sources de données pertinentes et d'envoyer les données au nœud initialisant de requêtes. Sur ce nœud tout le calcul doit être effectué. Cependant, cette méthode est coûteuse en termes de la consommation de la bande passante réseau. Papadimos et Maier [PaM01] ont proposé l'approche Mutant Query Plan (MQP) selon laquelle, le nœud initialisant la requête génère un plan d'exécution en utilisant les statistiques stockées dans son catalogue local. Ensuite, ce plan est envoyé aux nœuds distants qui le ré-optimisent en utilisant les statistiques stockées dans leurs propres catalogues locaux. Chaque nœud reçoit la requête avec le plan et des

résultats intermédiaires venant d'un autre nœud. Il optimise à nouveau la requête et exécute une partie du plan d'exécution avant d'envoyer ses résultats avec le nouveau plan d'exécution et la requête à un nouveau nœud. L'approche MQP a deux limites : (i) la transmission des résultats intermédiaires avec le plan d'exécution plusieurs fois pour une seule requête est coûteuse en termes de consommation de la bande passante de réseau et (ii) la requête peut être optimisée et exécutée sur un nœud sans aucune connaissance préalable de la capacité de ce nœud d'optimiser la requête au moment de la réception de requête. Les auteurs de [MaH08] ont proposé une approche semblable à MQP. Ils diffèrent leur approche de celle de MQP par la considération de la décomposition de question et l'utilisation de plan d'exécution dynamique basé sur le coût de la transmission de résultats intermédiaires et sur des informations à jour des voisins. Cependant, les auteurs de cette approche écrivent dans le même papier [MaH08] que pour éviter la transmission d'une grande quantité de résultats intermédiaires, leur approche nécessite un échange de messages ayant une complexité de l'ordre de O (N) où N est le nombre de nœuds dans le système.

Dans notre papier [KHM07a], afin de résoudre le problème de l'imprécision du modèle de coût qui peut mener à générer un plan d'exécution sous-optimal, nous avons proposé de profiter de la phase de la localisation de sources de données pour obtenir des méta-données à jour. Notre solution évite un accès supplémentaire au réseau pour rechercher ces méta-données et permet d'avoir des méta-données fraîches car elles sont obtenues des sites stockant les données désirées par la requête.

A cause des rôles essentiels du routage de requêtes, du *matching* de schémas et de l'optimisation de requêtes dans le traitement de requêtes en environnement P2P et afin de montrer ces rôles, nous présentons une partie des travaux de recherche concernant ces sujets dans le reste de ce chapitre.

2.7. Systèmes P2P de partage de données

Récemment, plusieurs systèmes P2P de partage de données ont vu le jour. Ces systèmes se différencient par leurs méthodes de routage de requêtes, par leurs approches de *matching* de schémas et par leurs méthodes de l'valuation de requêtes. Dans cette section, nous présentons quelques systèmes P2P de partage de données.

(i) APPA

APPA [AMP+06, AMP+07] est un système P2P fondé sur une plateforme totalement distribuée et adaptable avec tout type de topologies: structuré, non-structuré ou super-pairs.

Les nœuds dans APPA doivent avoir des vues sur un schéma global, appelé CSD (*Common Description Schema*), connu par tous les nœuds. Selon les auteurs d'APPA, leur approche ressemble à l'approche LAV (Local As View) sauf que les requêtes dans ce système sont écrites en termes des vues locales et pas en termes de CSD. Chaque nœud stocke localement les règles de correspondance entre son propre schéma et le CSD. Actuellement, les auteurs de APPA proposent une solution pour évaluer de requêtes meilleur-k (top-k queries) en environnement P2P non-structuré. Étant donné une requête Q et une valeur de TTL, l'évaluation de Q suit les phases suivantes. (i) Propagation de requête : Q est envoyée aux voisins du nœud initialisant la requête. (ii) Exécution locale et attente : chaque voisin p exécute Q localement et si la valeur de TTL est supérieur à zéro, il l'envoie, à son tour, vers ses voisins. p accède aux données locales qui correspondent les plus à la requête Q et donne des scores à chaque élément de données selon une fonction de scores. Ensuite, p choisit les meilleurs k réponses, les stockent localement avec leurs scores et attend les réponses venant de ses voisins. Le temps d'attente de chaque nœud est calculé en fonction de la valeur reçue de TTL, des paramètres réseau et des paramètres du traitement local. (iii) Assemblage de réponses : après la fin du temps d'attente, chaque nœud reçoit les scores de ses voisins. Il crée une liste ordonnée des couples (a, s) où a représente une adresse d'un voisin et s représente son score. Ensuite, il envoie cette liste à son parent (le nœud qui lui a envoyé la requête). Ce processus est répété sur chaque parent jusqu'à ce que le NIR crée la liste finale. (iv) Obtention de données : les nœuds possédant les meilleurs k scores sont contactés par le NIR et les résultats sont assemblés et délivrés à l'utilisateur.

Des spécialistes en conception de schémas créent des règles de correspondance entre les schémas locaux et le schéma global. Selon les auteurs d'APPA, le fait d'avoir un schéma global n'empêche pas le passage à grande échelle puisque les nœuds stockent leurs règles correspondance localement. En effet, cette solution est bonne lorsque le schéma partagé ne change pas beaucoup. Dans le cas inverse, pour chaque changement de schéma global il faut modifier la version ancienne du schéma global et il faut modifier aussi les règles de correspondance de chaque nœud. Ce qui nécessite une intervention des spécialistes qui pourraient ne pas être disponibles lors de la modification.

(ii) PIER

PIER [HHL+03, HCH+05] est un système P2P constitué de trois couches : (i) la couche de routage qui est responsable du routage de message et de gérer l'entrée et la sortie d'un nœud, (ii) le gestionnaire de stockage qui est responsable du stockage temporaire de données

dans une DHT et qui offre des fonctions pour stocker, supprimer et retrouver un objet ; et (iii) fournisseur qui relie les deux couches précédentes et possède une interface d'utilisateur. Une instance de ces couches se trouve sur chaque nœud. PIER ne suppose pas l'existence d'un catalogue global par contre il suppose l'existence de schéma standard dupliqué sur tous les nœuds. Il utilise le protocole CAN [RFH01] pour permettre aux centaines de milliers de nœuds de partager leurs données via l'Internet. Une clé est construite par l'espace de nom (namespace) et l'identifiant de ressource (resource ID). Il y a un espace de nom pour chaque relation et l'identifiant d'une relation est sa clé primaire. Les requêtes de jointure entre deux relations doivent être diffusées à tous les nœuds concernant les deux espaces de noms de ces deux relations. Deux algorithmes sont utilisés pour exécuter les requêtes d'équi-jointure : la jointure par hachage symétrique et Fetch Matches. Les auteurs testent aussi deux autres techniques, particulièrement, la semi-jointure symétrique et la Bloome Filter Rewrite, afin de réduire la grande consommation de la bande passante de la jointure par hachage symétrique. Même si les auteurs de PIER évaluent la performance de ces algorithmes pour un système de 10 000 nœuds, ces algorithmes ne sont pas évaluées lorsque les deux relations de jointure ont une grande taille. Autrement dit, lorsque les relations de jointure ont un grand nombre de tuples, cette solution n'est pas efficace, surtout en termes de coût de communications.

(iii) AmbientDB

AmbientDB [BoT03] est un prototype d'un système P2P de gestion de bases de données. Il suppose l'existence d'un schéma global connu par tous les nœuds. Il est constitué des composants suivants : (i) le processeur de traitement de requêtes qui est capable de traiter une requête donnée sur tous les nœuds ; (ii) le protocole P2P qui gère les interactions entre les nœuds au sein d'une DHT ; (iii) l'intégrateur de schéma qui est responsable de résoudre les hétérogénéités entre les schémas locaux en utilisant des fichiers XML et (iv) le composant d'une base de données local qui est gérée par un SGBD relationnel. Pour évaluer une requête, les auteurs supposent l'existence d'un plan d'exécution sans rien dire sur d'où viennent les informations sur lesquelles ce plan est basé et sans rien dire sur comment créer ce plan.

(iv) Piazza

Le système Piazza [HIM+03, TIM+03] est un des systèmes P2P super-pairs [BCO+08]. Il permet de partager des données représentées par un schéma XML qui définit la structure et le contenu d'un nœud. Les règles de correspondance dans ce système sont stockées,

actuellement, dans un index central ce qui rend ce système ressemblant beaucoup à un moteur de recherche. Les utilisateurs de Piazza peuvent écrire leurs requêtes selon leurs schémas locaux. La reformulation de requêtes est le problème le plus abordé par les auteurs de Piazza qui proposent un algorithme de reformulation permettant d'obtenir toutes les réponses existantes dans le système. Cependant, pour traiter une requête, une chaîne de reformulations pourrait être nécessaire. Le système Piazza actuel ne permet que d'exécuter des requêtes simples.

(v) coDB

coDB [FKL+04] est un système P2P de bases de données dans lequel les bases de données sont interconnectées par des règles de coordination basées sur l'approche GLAV [Len02]. Chaque nœud utilise son schéma local pour obtenir de données stockées sur ses voisins si une règle de coordination existe entre les schémas de ces deux nœuds. L'architecture de coDB s'appuie sur la plateforme JXTA qui est responsable des activités des nœuds au niveau du réseau. Les super-pairs dans coDB peuvent changer dynamiquement la topologie du réseau durant le temps d'exécution (*at run-time*). De plus, un super-pair fournit des services pour start-up tous les nœuds, pour créer des règles de coordination et pour obtenir des statistiques. Chaque super-pair stocke un rapport et le délivre à l'utilisateur qui peut voir les interactions avec les voisins et la découverte des nouveaux nœuds (non voisins).

(vi) PGrid

PGrid[KMH03] est un autre système structuré. Il utilise le *Matching* deux-à-deux. Il suppose l'existence des règles de correspondance prédéfinies entre les différents schémas dans le système. Ces règles sont créées par des experts. Le système PGrid est capable d'extraire de nouvelles règles de correspondance entre deux schémas locaux qui ne sont pas directement liés à l'avance. PGrid propose une solution pour des requêtes d'intervalle.

(vii) PeerDB

PeerDB [OST03, NOT+03] est un système P2P super-pair permettant à ses nœuds de partager des données relationnelles. Il utilise l'approche IR (Information Retrieval) et les agents mobiles pour trouver les sources des données souhaitées. Les agents mobiles participent également à l'évaluation de requêtes. Trois couches essentielles existent dans ce système et dupliquées sur tous les nœuds: (i) une couche P2P qui est responsable d'échange de données et d'informations avec les autres nœuds; (ii) une couche de gestion de données

qui offre le stockage de données et le traitement de requêtes et (iii) une interface SQL qui permet à l'utilisateur de soumettre des requêtes SQL. Pour chaque relation et pour chacun de ses attributs, l'utilisateur crée des mots clés (des synonymes). Ces mots clés sont stockés localement sur les nœuds. Lorsqu'un nœud soumet une requête, des agents mobiles sont créés et envoyés vers les autres nœuds dans le système sans avoir une connaissance préalable du contenu de ces nœuds. Les agents mobiles apportent la requête et les mots clés correspondants à chaque relation et à chaque attribut de cette requête. Ils essaient de trouver des correspondants entre le schéma dont la requête écrite et les schémas des nœuds visités. Selon une stratégie d'appariement (*Matching Strategy*) proposée par les auteurs, les agents retournent au NIR des informations (les adresses IP, les correspondances sémantiques) concernant les nœuds qui sont susceptibles d'être sources pertinentes de données (les nœuds candidats). L'utilisateur à son tour choisit les nœuds pertinents parmi la liste des candidats relevés par les agents mobiles. Afin d'améliorer le routage de requêtes, PeerDB utilise quelques nœuds spéciaux, appelés LIGLO (Location Independent Global Names Lookup). Ces nœuds jouent le rôle des serveurs ayant pour rôle de donner des identifiants uniques aux nœuds et de stocker des traces concernant leurs statuts actuels (connecté/déconnecté). Le but de ces statistiques est d'améliorer la localisation de sources de données. Dans le système PeerDB, un nœud quelconque n'est pas obligé de stocker des informations concernant les autres nœuds. De plus ce système permet à l'utilisateur de modifier la partie partagée de ses données selon ses besoins. Cependant, la capacité de trouver toutes les réponses existantes dans le système reste loin d'être obtenue.

(viii) Edutella

Edutella[17] [NWQ+02] est un système P2P super-pair basé sur JXTA [G01]. Il utilise RDF[18] (Ressource Description Framework) pour noter les ressources (y compris les données). Toutes les ressources sont identifiées par des identifiants uniques (URI : Uniform Ressource Identifier). Toutes les notations sont représentées par un triplet < Sujet, Propriété, Valeur >, où Sujet identifie la ressource qu'on veut décrire (utiliser URI), Propriété indique quel attribut on spécifie et Valeur indique la valeur de l'attribut spécifié. La valeur doit être s'exprimer comme type de données ou un autre URI indiquant une autre ressource. Les descriptions concernant un group de nœuds sont stockées sur son super-pair. Un schéma

[17] Edutella, ''The Edutella Project'', http://edutella.jxta.org/, 2005.
[18] http://www.w3.org/TR/rdf-concepts

RDF (RDFS19) est utilisé. Ce schéma contient des classes de ressources, des propriétés et des conditions sur les propriétés (domaine, intervalle, etc.). Les propriétés peuvent être utilisées pour intégrer deux schémas, relier plusieurs ressources ou créer des relations hiérarchiques entre les ressources.

(ix) PinS

PinS [VRL04] est un système d'indexation et d'interrogation de données dans des systèmes P2P basés sur DHT. Le système PinS[20] stockent n'importe quel type de données (e.g. images, vidéos, documents, etc.). Il propose de stocker également les méta-données (i.e. nom_attribute=valeur) qui décrivent les données dans le système afin de faciliter la recherche de leurs sources. L'originalité de PinS réside dans le fait qu'on peut l'utiliser avec n'importe quel système P2P basé sur DHT et avec n'importe quel type de données. Ce système fournit plusieurs services comme l'évaluation de requêtes déclarative [VRL04] et l'indexation personnalisée [VRL04]. Cependant, PinS propose de dupliquer les objets (entités de données) partagés et leurs méta-données sur plusieurs nœuds. Lorsque le nombre de ces objets est très élevé, la gestion de leurs duplicatas devient très difficile. Les auteurs de PinS [VRL04] n'abordent actuellement pas le problème de l'hétérogénéité entre les objets stockés dans PinS. Leurs futurs travaux seront consacrés à ajouter de la sémantique aux requêtes traitées par le système et à fournir différents niveaux d'expression pour ces requêtes.

(x) autres travaux de recherche

Dans [BNS+05], les auteurs proposent un algorithme pour traiter des requêtes meilleur-k au sein d'Edutella. Les super-pairs sont responsables de traiter les requêtes meilleur-k et les autres nœuds exécutent localement d'autres types de requêtes et attribuent des scores à leurs ressources. Etant donné une requête Q, le NIR envoie Q à tous les super-pairs. Les super-pairs envoient Q aux nœuds pertinents qui sont connectés à eux. Chaque nœud possédant des unités de données demandées par Q donne des scores à ces unités et envoie celles qui ont le maximum score aux super-pairs. Chaque super-pair choisit les unités de données qui ont le plus grand score parmi tous les résultats de ses nœuds. Après cela, il envoie les scores de ses données au super pair de NIR pour extraire les meilleur-k unités et pour envoyer la requête aux nœuds stockant ces données pour retourner les réponses au NIR.

[19] http://www.w3.org/TR:rdf-schema
[20] http://www-lsr.imag.fr/users/Maria-Del-Pilar.Villamil/PinS/index.htm

Pour traiter de requêtes plus complexes, chaque super-pair crée un plan d'exécution local qui est proche de l'optimal et enfin les résultats sont assemblés et retournés au NIR.

Dans [HHH+02], les auteurs proposent une méthode fondée sur DHT pour traiter des requêtes complexes. Des fichiers sont manipulés comme des relations. Les caractéristiques (e.g. nom, identifiant) de chaque fichier sont manipulées comme des attributs d'une relation. Le système considéré est constitué de trois couches : (i) une couche de stockage local de données (fichiers) pour accéder aux données partagées et fournir les attributs de chacune; (ii) une couche de DHT responsable d'indexer les données et de guider le routage de messages entre les nœuds et (iii) une couche de traitement de requêtes ayant deux interfaces l'une est graphique et l'autre est une interface SQL. Le protocole de routage a été modifié afin de choisir des sources pertinentes. Malgré la capacité de cette méthode de traiter de requêtes complexes, elle souffre de deux limites. La première est que les utilisateurs doivent connaître la sémantique de données partagées ce qui n'est pas toujours valide. La deuxième est qu'elle permet d'échange de données ayant une grosse granularité (fichiers). Cette méthode a été étendue dans [HHL+03] pour permettre un partage de données ayant une granularité fine. La qualité de réponses retournées par cette approche dépend fortement des techniques automatiques utilisées pour découvrir les correspondances entre des schémas (ou ressources) hétérogènes. En effet, pratiquement aucune technique totalement automatique n'existe pour trouver des correspondances entre deux schémas [Hal05].

Dans [RSW05], les auteurs proposent une approche, dite meilleurs efforts, pour traiter de requêtes complexes en environnement P2P structuré. A cause de la nature dynamique et la grande échelle de cet environnement, les auteurs constatent que ce n'est pas possible d'avoir un plan d'exécution basé sur une connaissance générale concernant tous les nœuds dans le système. Pour cela, les auteurs ne visent pas la réponse exacte d'une requête donnée. Cependant, ils essaient de trouver la meilleure réponse en utilisant un plan d'exécution basé sur la connaissance locale disponible sur le nœud soumettant la requête. En utilisant deux fonctions de hachage, l'une pour les relations et l'autre pour les tuples, les auteurs étendent le protocole CAN [RFH01] afin de faciliter l'accès aux données. Les données sont redistribuées sur les nœuds en trouvant un compromis entre les deux cas extrêmes : (i) chaque relation doit complètement stockée sur un seul nœud et (ii) les tuples d'une relation sont distribués sur tous les nœuds. En réalité, le fait de redistribuer toutes les données dans un système réparti à grande échelle est très coûteux en termes de consommation de

ressources. Surtout, lorsque cette solution ne permet pas d'obtenir la bonne réponse à une requête donnée.

Les auteurs de [GAA03] proposent une architecture basée sur le protocole Chord [SMK+01] pour partager des données relationnelles en environnement P2P. Ils proposent également une méthode pour répondre aux requêtes d'intervalle. De plus de schéma conceptuel global connu par tous les nœuds, les auteurs supposent que les sources de données sont, aussi, connues par tous les nœuds. L'accès aux données dans leurs ressources n'est pas toujours disponible à cause des problèmes de surcharge ou de connectivité. Les nœuds dans le système stockent des fragments horizontaux des relations en se basant sur des requêtes déjà exécutées. Les auteurs proposent une méthode pour localiser les nœuds stockant les fragments pertinents. Sahin et al. [SGA+02] utilisent un système P2P basé sur le protocole CAN pour traiter des requêtes d'intervalle. Ils supposent que les résultats des requêtes déjà exécutées sont stockés et exploités dans l'exécution de nouvelles requêtes. Une requête donnée doit être envoyée vers les nœuds qui sont responsable d'une partie de valeurs appartenant à l'intervalle. Cependant, les auteurs de [GAA03] et de [SGA+02] ne traitent que des requêtes mono- attribut, ils ont laissé les requêtes multi-attributs comme leur futur travail.

Les auteurs de [AX02] proposent une solution basée sur le protocole CAN pour traiter de requêtes d'intervalle. Ils proposent d'attribuer les sous-intervalles qui sont les plus proches aux nœuds proches l'un des autres. Un sous groupe de nœuds, appelés les gardians d'intervalle, est responsable d'un sous-domaine du domaine de l'attribut considéré. Lorsqu'une requête d'intervalle est soumise, elle est envoyée aux gardians et via ces nœuds aux nœuds stockant les données souhaitées. Cependant, l'approche proposée est dépendante du protocole CAN ayant un espace virtuel de 2-dimensions. C'est très difficile d'étendre cette approche pour fonctionner avec un autre type de DHT comme Chord [SMK+01].

GrouPeer [KTS09] est un système P2P non-structuré. Il permet de partager des données relationnelles en regroupant les nœuds en groupes sémantiques. Les auteurs de GrouPeer se concentrent sur le problème de l'incapacité d'un pair d'obtenir des informations sur des données désirables ou sur des pairs ayant des intérêts semblables. Lorsque le *Matching* deux-à-deux entre les schémas locaux est déjà établi, un pair recevant une requête reformulée peut être incapable de répondre suffisamment à la requête. Cette requête pourrait être corrompue pendant la reformulation sur des pairs intermédiaires. GrouPeer offre une

approche dynamique pour créer et maintenir des groupes sémantiques de pairs. Par cette approche, un pair peut re-reformuler automatiquement une requête reçue.

OntoZilla [JoC09] est un système P2P évitant l'inondation de réseau en utilisant des ontologies pour regrouper les pairs en groupes d'intérêt commun. Ceci permet de guider le routage de requêtes. Les ontologies permettent la description des ressources des pairs afin d'automatiser la gestion d'informations. OntoZilla est plus flexible que des systèmes P2P basés sur DHT puisque les groupes des pairs sont basés sur les intérêts spéciaux des pairs qui pourraient être variés au cours du temps. Actuellement, les auteurs d'OntoZilla développent un prototype. A notre connaissance, il n'y a aucune information sur la validité du système OntoZilla disponible encore.

HePTex [BCL+10] est un système P2P pour partager des données stockées dans des bases de données XML hétérogènes. L'intérêt central de HePTex est la déduction automatique des règles de correspondance entre les schémas locaux et la traduction des requêtes XQUERY entre les différents schémas. Le routage de requêtes dans HePTex est totalement décentralisé et a une complexité logarithmique (comme dans des systèmes P2P basés sur DHT).

eSciGrid [SiG10] est un système P2P pour partager une énorme quantité de données en environnement de Grille. Il permet aux communautés d'e-science de partager des données d'une façon décentralisée et coopérative. Ce système fournit un protocole décentralisé pour stocker les résultats des requêtes d'intervalle. Ce protocole prend en compte le trafic du réseau et la distance physique entre des pairs. eSciGrid doit être étendu pour être utilisé dans d'autres domaines comme l'E-santé et l'Astrophysique.

	Topologie	*Matching* de schémas		Optimisation de requêtes
		Approche	Type	
APPA	Adapté avec n'importe quelle topologie P2P	Basée sur schéma global	Statique	-
PIER	structuré	Basée sur schéma global	Statique	Basée sur méta-données obtenues par un service de monitoring
AmbientDB	structuré	Basée sur schéma global	Statique	-
Piazza	Super-pair [BCO+08]	*Matching* deux-à-deux	Statique	-
coDB	Super-pair	*Matching* deux-à-deux	Statique	-
PGrid	Structuré	*Matching* deux-à-deux	Statique	-
PeerDB	Super-pair	Basée sur Recherche d'Informations	Dynamique	-
Edutella	Super-pair	Basée sur Recherche d'Informations	Statique	Basée sur méta-données stockées sur super-pairs
PinS	Structuré	-	-	-
Notre Approche	Structuré	Hybride : Basée sur Recherche d'Informations et sur schéma global (schéma global remplacé par une ontologie)	Dynamique	Basée sur méta-données des hôtes des sources de données pertinentes

Le signe "-" signifie que ce problème n'est pas abordé selon notre connaissance

Fig. 7. Comparaison entre les systèmes P2P de partage de données étudiés

2.8. Comparaison qualitative

Dans cette section, nous présentons une comparaison qualitative entre, d'un côté, les systèmes P2P [BoT03, FKL+04, NWQ+02, TIM+03, NOT+03, MKH03, HCH+05, VRL04, AMP+07] étudiés dans la section précédente et l'approche proposée au sein de notre thèse de l'autre côté.

En se basant sur l'état de l'art effectué précédemment, nous avons pu constater que les systèmes P2P non-structurés sont tolérants aux pannes plus que les autres systèmes (i.e.

structurés et super-pairs). Ils offrent aussi plus d'autonomie à leurs nœuds tout en respectant l'égalité totale entres les nœuds en termes de leurs rôles. Cependant, leurs méthodes de localisation de sources de données sont moins performantes des points de vue du nombre de messages échangés et de la capacité de trouver toutes les réponses valides disponibles dans le système.

Les systèmes coDB[FKL+04], Edutella [NWQ+02], Piazza [TIM+03], PeerDB [NOT+03] sont des systèmes P2P super-pairs. Dans ce type de systèmes, les nœuds ne jouent pas le même rôle. Des nœuds distingués, appelés super-pairs, sont responsables de fournir de services aux autres nœuds, appelés des clients, existants dans le système. le système Piazza [TIM+03] utilise un nœud (ou plusieurs nœuds) pour stocker un index central utilisé dans la reformulation de requêtes. Le système PeerDB [NOT+03] utilise des serveurs LIGLO [NOT+03] pour améliorer le routage de requêtes. Quant au système coDB [FKL+04], il stocke des règles de correspondance entre les schémas hétérogènes sur les super-pairs. Le système Edutella [NWQ+02] utilise plusieurs super-pairs pour améliorer la localisation de sources de données d'un côté et pour exécuter de requêtes de l'autre côté. Malgré le nombre réduit de messages échangés lors de la localisation de sources de données, ces systèmes sont moins tolérants aux pannes par rapport aux autres systèmes P2P (structurés et non-structurés). Ainsi, la capacité de ces systèmes de trouver un nombre important des réponses valides existantes dans le système dépend du type du système P2P formé par les super-pairs.

Les systèmes AmbientDB [BoT03], PGrid [MKH03], PIER [HCH+05], PinS [VRL04] et notre système sont des systèmes P2P structurés capables d'apporter des solutions aux limites des systèmes P2P non-structurés et des systèmes P2P super-pairs. Un de leurs avantages est la capacité de trouver la majorité des réponses valides disponibles dans le système tout en utilisant un nombre "acceptable" de messages échangés lors de la localisation de sources de données. Les systèmes structurés souffrent du fait que le degré de l'autonomie de leurs nœuds est relativement réduit. C'est vrai que les nœuds ne peuvent pas choisir leurs voisins, mais ceci peut être toléré par rapport aux avantages de ces systèmes. Les systèmes PIER [HCH+05], PinS [VRL04] et notre système sont basés sur une DHT contrairement au système PGrid [MKH03] qui se distingue par la séparation entre l'identifiant d'un nœud et sa position sur le réseau physique (i.e. son IP). Un des avantages de PGrid [MKH03] est que plusieurs nœuds peuvent être responsables des méta-données concernant une entité de données partagée. Ceci améliore la tolérance aux pannes. Par contre, la gestion de l'entrée et de la sortie d'un nœud serait plus compliquée. Les auteurs de PGrid [MKH03] ne fournissent

pas d'informations concernant la complexité de ces deux processus. Une DHT peut faciliter le routage de requêtes tout en respectant une complexité "acceptable" de l'entrée et de la sortie des nœuds. Le système PIER [HCH+05] est basé sur le protocole CAN [RFH01] qui est moins efficace, en termes du nombre de messages échangés lors du routage de requêtes, que les deux systèmes Pastry [RoD01a] et Chord [SMK+01] qui sont utilisés par PinS [VRL04] et par notre système respectivement. Le système PinS [VRL04] est caractérisé par le fait qu'il peut être utilisé avec n'importe quel protocole de routage en utilisant n'importe quelle fonction de hachage. Même s'il utilise actuellement le protocole Chord [SMK+01], notre système est également indépendant du protocole du routage et de la fonction de hachage utilisée.

Quant au problème de l'hétérogénéité de schémas, les systèmes AmbientDB [BoT03], Piazza [TIM+03], coDB [FKL+04] et PGrid [MKH03] utilisent l'approche de *matching* deux-à-deux qui peut nécessiter une chaîne de *matching* lors de la localisation de sources de données mentionnées par une requête. Les systèmes APPA [AMP+07] et PIER [HCH+05] utilisent l'approche de *matching* basée sur un schéma global connu par tous les nœuds. Cette solution ressemble beaucoup à celle utilisée dans les systèmes d'intégration de données sauf que dans le système APPA [AMP+07] les requêtes sont écrites selon les schémas locaux. Afin d'utiliser un schéma global, il faut prévoir tous les nœuds capables de se connecter au système et il faut connaître dès le début leurs schémas locaux pour créer un schéma global. Le fait d'avoir un schéma global rend difficile pour un nouveau nœud de se connecter au système car l'utilisateur de ce nœud n'a pas forcement la capacité de comprendre la sémantique des termes utilisés dans le schéma global. Dans notre système et afin de créer le *matching* entre les schémas locaux, nous utilisons une approche hybride entre l'approche basée sur schéma global et celle basée sur la recherche d'information (RI) qui est utilisée par les deux systèmes PeerDB [NOT+03] et Edutella [NWQ+02]. Ce qui distingue notre système est que nous remplaçons le schéma global par une ontologie de domaine dupliquée sur tous les nœuds dans laquelle la sémantique peut être explicitement expliquée. Ceci rend plus facile à un nouveau nœud de se connecter au système car l'utilisateur peut comprendre la sémantique des termes utilisés par l'ontologie et il peut ensuite créer le *matching* entre l'ontologie et son schéma local. Un autre avantage d'utiliser une ontologie de domaine est qu'aujourd'hui des milliers d'ontologies de domaine sont disponibles sur l'Internet[21] ce qui rend notre choix pratique. Le *matching* dans le système Edutella [NWQ+02] est statique. Cependant, dans

[21] Swoogle homepage, http://swoogle.umbc.edu/

PeerDB [NOT+03] et dans notre système le *matching* est dynamique et bien adapté avec les environnements P2P de nature dynamique. A ce point là, la différence entre notre système et le système PeerDB [NOT+03] est que le système PeerDB [NOT+03] utilise des agents mobiles pour localiser les sources de données tout en effectuant la *matching* entre les schémas locaux en même temps. Les agents mobiles peuvent visiter des centaines de nœuds et effectuer le *matching* entre les schémas de ces nœuds et le schéma local du nœud initialisant la requête sans avoir une connaissance préalable du contenu des nœuds visités. Par conséquent, les agents mobiles peuvent visiter plusieurs nœuds en consommant plusieurs ressources (e.g. bande passante réseau) sans aucun bénéfice. Cependant, nous utilisons dans notre système une ontologie de domaine et nous effectuons le *matching* localement entre le schéma local du nœud initialisant la requête et l'ontologie de domaine. Même si le système PeerDB utilise des serveurs LIGLO [NOT+03] pour guider (en se basant sur des statistiques) le routage de ses agents, notre système basé sur une DHT reste plus efficace. Les auteurs de PinS [VRL04] n'abordent actuellement pas le problème de l'hétérogénéité entre les objets stockés dans PinS. Ils vont ajouter de la sémantique aux requêtes traitées par PinS durant leurs futurs travaux de recherche.

L'optimisation de requêtes devient un problème majeur lorsqu'on n'a pas un catalogue global contenant les méta-données nécessaires pour générer un plan d'exécution proche de l'optimal. A notre connaissance, aucun des systèmes étudiés dans la section précédente n'aborde, en détailles, ce problème. Les systèmes Edutella [NWQ+02], coDB [FKL+04], AmbientDB [BoT03] et APPA [AMP+07] (dans son architecture super-pairs) utilisent des informations disponibles sur les super-pairs. Le système Piazza [TIM+03] utilise des informations stockées dans son index global qui peut créer un goulet d'étranglement. Le système PIER [HCH+05] utilise un service de monitoring pour obtenir des méta-données à jour. En ce qui concerne notre système, nous savons que ce n'est pas possible d'avoir des informations globales (i.e. concernant tous les nœuds dans les systèmes) à jour. Nous nous basons sur l'idée que ce n'est pas nécessaire d'avoir des informations globales pour traiter une requête. Des informations concernant les hôtes des sources de données suffisent pour générer un plan d'exécution global proche de l'optimal. Ce plan sera exécuté sur les hôtes des sources de données. Par conséquent, nous avons proposé d'obtenir toutes les méta-données nécessaires durant la phase de localisation de sources de données. Les détailles de notre système font l'objet du chapitre suivant.

2.9. Conclusion

Les systèmes P2P représentent un domaine de recherche très actif grâce à leurs actuelles utilisations et à leurs futures applications. Les systèmes P2P existants dans la vie quotidienne sont des systèmes de partage de fichiers dans lesquels l'intérêt essentiel est le routage de requêtes vers les nœuds stockant le fichier recherché afin de télécharger ce fichier. Ces systèmes ne sont pas adaptés au partage des données ayant une granularité fine (i.e. attribut atomique) et ils ne supportent pas le traitement de requêtes complexes écrites au moyen d'un langage d'interrogation de haut niveau (i.e. SQL). A cause des caractéristiques des systèmes P2P, aucun nœud ne peut avoir une vue globale sur le système. De plus, les données sont souvent représentées par des schémas hétérogènes. Par conséquent, ce n'est pratique d'avoir un catalogue global. Ceci a un impact direct sur la localisation de sources de données et sur la génération d'un plan d'exécution proche de l'optimal. Dans ce contexte, nous avons cité des principaux travaux de recherche qui sont les plus proches de notre travail tout en montrant leurs avantages et leurs limites. Nous avons également mené une comparaison qualitative entre ces travaux incluant nos propositions.

L'état de l'art effectué au sein de ce chapitre nous a pu constatés qu'il n'y a pas un consensus sur une architecture de système P2P de partage de données et que les approches proposées pour l'évaluation de requêtes sont dépendantes de l'architecture utilisée. Pour cette raison, nous allons présenter dans le chapitre suivant une architecture et une approche d'évaluation de requêtes. Nous allons également présenter notre solution au problème de localisation de sources de données qui prend en compte l'hétérogénéité de schémas. Enfin, nous allons montrer notre solution concernant la génération d'un plan d'exécution proche de l'optimal.

Chapitre 3.

Sommaire

Localisation de sources de données et obtention de méta-données dans un système P2P de partage de données

L'objectif essentiel de ce chapitre est de présenter notre contribution menée dans le cadre de la thèse. Dans le contexte d'évaluation de requêtes SQL en environnement P2P et à cause de la grande échelle, de l'autonomie et l'instabilité de nœuds, ce n'est pas pratique d'avoir un catalogue global. La localisation de sources de données représentées par des schémas locaux hétérogènes est un problème ouvert lié directement à l'absence d'un catalogue global. Pour résoudre ce problème, nous proposons une solution permettant à un nouveau nœud de se connecter "facilement" au système considéré. L'utilisateur de ce nœud peut profiter, en utilisant des formules de similarité, de la sémantique représentée explicitement dans une ontologie de domaine pour publier/modifier la partie partagée de ses données sans avoir besoin d'une autorité centralisée. Notre solution introduit le principe des indexes de structure qui permet de résoudre l'hétérogénéité structurelle entre les schémas locaux. Le problème de l'optimisation globale de requêtes est une autre conséquence de l'absence d'un catalogue global qui contient souvent des méta-données nécessaires pour créer un plan d'exécution proche de l'optimal. Afin d'éviter un problème de sous-optimalité du plan d'exécution, la solution proposée permet également de profiter de la phase de la localisation pour obtenir toutes les informations nécessaires pour la phase de l'optimisation. Ceci permet d'obtenir des méta-données à jour et d'éviter un accès supplémentaire au réseau.

3.1. Introduction

A cause des caractéristiques des systèmes P2P, l'évaluation de requêtes SQL dans ces systèmes est plus difficile par rapport aux systèmes de bases de données traditionnels. Dans un tel environnement distribué à grande échelle dont les nœuds sont instables et autonomes, ce n'est pas pratique d'avoir un catalogue global. Dans les systèmes de bases de données traditionnels, le catalogue global contient souvent des informations concernant les

placements des données dans le système et des méta-données nécessaires pour générer un plan d'exécution proche de l'optimal. Ceci rend difficile la localisation de sources de données représentées par des schémas locaux sémantiquement et structurellement hétérogènes. La génération d'un plan d'exécution proche de l'optimal devient difficile aussi.

Comme nous avons pu le constater dans le chapitre précédent, l'utilisation des techniques de routage de requêtes utilisées dans les systèmes P2P de partage de fichiers, dans le contexte de bases de données a fait l'objet de nombreux travaux de recherche. Plusieurs travaux de recherche supposent l'existence d'un schéma global et l'utilisation des méthodes de *matching* statique pour relier les schémas locaux avec le schéma global. Cette solution est inspirée par les systèmes d'intégration de données dans lesquels les sources de données peuvent être vues comme une source unique de données. Cette solution est convenable pour des applications dans lesquelles tous les nœuds voulant participer au système sont connus dès le début. Dans ce type d'applications, les règles de correspondance entre le schéma global et l'un des schémas locaux sont créées par des spécialistes qui connaissant parfaitement les sémantiques des deux schémas. Généralement, les règles de correspondance sont transparentes à l'utilisateur qui n'est pas capable de modifier la partie partagée de ses données parce qu'il ne connaît pas la sémantique du schéma global. Pour cette raison, un nœud imprévu ne peut pas entrer dans le système sans contacter les spécialistes connaissant la sémantique du schéma global. Donc, l'ajout d'un nœud au système dépend de la disponibilité de ces spécialistes. Autrement dit, d'une administration centralisée. Ceci rend le système un monde ''fermé''. Généralement, un environnement P2P doit être un monde ''ouvert'' dans lequel un nouveau nœud doit entrer dans le système sans aucune administration centralisée. Dans un tel environnement, on ne peut pas voir les sources de données comme une source unique de données.

En environnement P2P, la phase d'optimisation globale devient, encore, plus difficile à cause de l'absence d'un catalogue global. Une requête SQL est souvent traduite, durant cette phase, en plan d'exécution optimisé dont la génération se base sur des informations concernant (i) les caractéristiques physiques de données (e.g. taille des relations) et (ii) les paramètres physiques des hôtes des sources de données (e.g. valeur CPU). D'ici jusqu'à la fin de ce mémoire, nous appelons ces informations par méta-données[22]. Le manque de certaines informations et l'obsolescence des statistiques sont deux facteurs influençant l'optimalité du plan d'exécution. Les méta-données sont, souvent, stockées dans un catalogue centralisé ou

[22] Le terme "Méta-données" a un sens plus général représentant toutes les informations concernant les données.

dupliqué sur plusieurs serveurs. La mise à jour des méta-données se base sur des statistiques issues des exécutions précédentes. A cause de la décentralisation et de la grande échelle des systèmes P2P, ce n'est pas pratique d'avoir un catalogue centralisé qui pourrait engendrer un goulet d'étranglement. Ce n'est même pas pratique d'avoir un catalogue global dupliqué sur tous les nœuds. Ce type de catalogue nécessite un nombre important de messages de mise à jour lorsqu'un nœud se connecte/se déconnecte au/du système.

Dans ce chapitre, nous proposons une approche permettant, d'un côté, de localiser les sources de données pertinentes et de résoudre le problème de sous-optimalité d'un plan d'exécution lié au manque et à l'obsolescence de méta-données de l'autre côté. La section 2 décrit le contexte de notre travail en représentant en même temps les hypothèses prises en compte et un exemple d'étude. Dans la section 3, nous présentons le principe de notre approche qui consiste à utiliser une ontologie de domaine pour représenter explicitement la sémantique de données partagées et à choisir une technique P2P convenable pour trouver le placement des données désirées par une requête. Des formules de similarité sont proposées dans la section 4 pour aider l'utilisateur à créer des règles de correspondance entre son schéma local et l'ontologie de domaine. Dans la section 5, nous proposons le principe des indexes de structure qui permet à la technique P2P choisie de résoudre le problème de l'hétérogénéité structurelle entre les schémas locaux. Avant de conclure ce chapitre dans la section 7, nous expliquons au sein de la section 6 comment obtenir des méta-données nécessaires pour la phase d'optimisation lors de la localisation de sources de données.

3.2. Contexte général et exemple d'étude

Cette section a pour objectif de présenter les hypothèses prises en compte dans notre étude et de donner un exemple permettant de mieux comprendre le travail effectué au sein de cette étude.

3.2.1. Contexte général

Récemment, les systèmes P2P de partage de données sont apparus, d'un côté, comme une nouvelle génération des systèmes P2P et, de l'autre côté, comme un nouveau pas dans le long chemin de recherche en bases de données. Etant donné que le modèle relationnel de bases de données est largement utilisé, nous limitons notre étude à ce modèle. Dans notre travail nous considérons un système P2P de partage de données qui est, informellement, définit comme un système réparti à grande-échelle dans lequel les nœuds sont autonomes et

peuvent se connecter/se déconnecter au/du système à n'importe quel moment d'une façon complètement décentralisée. Chaque nœud a son propre système de bases de données (SBD) qui est constitué d'un SGBD et d'une (ou plusieurs) base(s) de données qu'il gère. Dans un tel environnement, les nœuds jouent des rôles symétriques. Chaque nœud joue le rôle d' : (i) un client lorsqu'il consomme de ressources disponibles sur d'autres nœuds dans le système; (ii) un serveur lorsqu'il fournit des services aux autres nœuds comme par exemple le stockage des méta-données concernant ses voisins ou le stockage des relations temporaires lors des exécutions de requêtes ; (iii) un routeur lorsqu'il propage des requêtes et des messages vers d'autres nœuds dans le système et (iv) un hôte des source de données lorsqu'il partage ses données avec d'autres nœuds dans le système. Chaque nœud gère d'une manière autonome sa propre base de données. Les bases de données ont un intérêt commun (e.g. des bases de données médicales de chercheurs travaillant sur la maladie d'Alzheimer). Elles stockent des données similaires avec une interdépendance sémantique. Néanmoins, à cause de l'autonomie de nœuds, les données pourraient être représentées par des schémas locaux hétérogènes. Nous supposons qu'aucun nœud n'a une vision globale sur le système. Il n'y a pas de catalogue global ni centralisé ni dupliqué.

Dans l'environnement considéré, les nœuds désirent partager leurs données d'une façon compréhensible et efficace. Le mot ''compréhensible'' signifie que lorsqu'un nœud soumet une requête dans le système, les autres nœuds doivent comprendre la sémantique de cette requête afin d'envoyer les données souhaitées et d'éviter l'envoi des réponses invalides. Ces réponses invalides nécessitent une consommation de ressources (e.g. bande passante, capacité de stockage et puissance de calcul) pour aucun bénéfice. Par le mot ''efficace'', nous désignions qu'il faut réduire le temps de réponse d'une requête donnée tout en minimisant, le plus possible, la consommation des ressources disponibles lors de l'évaluation de la requête.

L'environnement considéré forme un monde ouvert. C'est-à-dire, un nœud imprévu doit être capable d'entrer dans le système sans aucune administration centralisée. L'utilisateur de chaque nœud doit être capable de modifier ''facilement'' la partie partagée de ses données. Il doit être capable, aussi, de choisir pour une requête soumise si cette requête doit être exécutée localement en utilisant le SBD local ou globalement (en permettant aux autres nœuds de participer à l'évaluation de sa requête).

3.2.2. Exemple d'étude

Afin de bien expliquer les différents aspects de notre travail, nous donnons un exemple illustrant les problèmes abordés et les solutions proposées. Dans la figure 8, nous considérons 7 nœuds. Chacun a développé sa base de données indépendamment des autres nœuds et il utilise son propre schéma local. Par conséquent, les données partagées sont représentées par des schémas locaux hétérogènes. Nous pouvons distinguer deux niveaux d'hétérogénéité entre les schémas :

N1: Doctor (Name, Paycheck)
N8: Doctor (Name, Salary)
Ill (Name, Address, Doctor_Name)
N14: Physician (Name, Address)
Patient (Name, Age)
N21: Doctor (Name, Earnings, Address)
N32: Consultant (Name, Salary, Telephone)
N42: Doctor (Name, Address)
N48: Patient (Name, Age, Address, Doctor_Name,
Disease, Treatment, Social_Security_Number)

Fig. 8. Schémas locaux de certains nœuds dans le système

(i) Niveau sémantique : dans ce niveau de l'hétérogénéité, on peut distinguer deux situations :

- Synonymie (noms différents) : cette situation apparaît lorsque deux (ou plusieurs) noms de relations (ou d'attributs) appartenant à deux schémas différents ont le même sens; e.g. ''*Doctor*'' selon N8 et ''*Consultant*'' selon N32, ces deux mots représentent le concept sémantique ''*Médecin*''.
- Polysémie (sens différents) : cette situation apparaît lorsqu'un nom de relation (ou d'attribut) se trouve dans plusieurs schémas, mais, il porte plusieurs sens; e.g. le mot "*Doctor*" selon N8 est un médecin par contre selon N21, il est une personne ayant soutenu une thèse de doctorat (e.g. doctor en informatique).

(ii) Niveau structurel : une relation (e.g. *Doctor*) peut être représentée de manières différentes selon les schémas locaux en termes d'attributs. Chaque schéma représente cette relation selon l'intérêt de son organisation (ou utilisateur); e.g. *Doctor (Name, Paycheck)* selon le schéma de N1 et *Doctor (Name, Address)* selon le schéma de N42.

Nous allons nous baser sur cet exemple tout au long de ce chapitre.

3.3. Approche d'évaluation de requêtes SQL en environnement P2P

Dans les approches traditionnelles de partage de données en environnements distribués, le problème de la localisation des sources de données hétérogènes est, souvent, résolu grâce à la définition d'un schéma global et des règles de correspondance entre ce schéma et les schémas locaux. La définition d'un schéma global nécessite un accord sur les concepts modélisés par les données. Ce schéma doit être défini avant que les bases de données puissent stocker ou échanger les données. Le schéma global et les règles de correspondance sont souvent stockés dans un catalogue global. Cependant, en environnement P2P, la grande échelle et la nature dynamique empêchent d'avoir un catalogue global. Afin de trouver une méthode permettant de jouer le rôle d'un catalogue global dans la localisation de sources de données, nous commençons par diviser les informations nécessaires pour effectuer cette phase en deux parties [KHM08b]:

(i) Schéma conceptuel qui correspond à la description des relations entre toutes les données partagées dans le système sans aucune description de leur implantation ou de leur stockage physique. Ce schéma garantit un échange de données d'une façon compréhensible. La sémantique des termes utilisés dans ce schéma n'est pas explicitement représentée.

(ii) Schéma de placement qui décrit les placements de données sur les nœuds et les règles de correspondance entre les schémas locaux et le schéma conceptuel. Grâce au schéma de placement, on peut trouver n'importe quelle relation (ou attribut) dans le système.

L'utilisation d'un schéma conceptuel dupliqué sur tous les nœuds ne permet pas aux utilisateurs de modifier leurs parties partagées de données à cause du manque d'une représentation explicite de la sémantique des termes de ce schéma. Cette sémantique se trouve souvent dans la tête du concepteur de schéma qui ne peut pas être disponible lors de la publication ou de la modification des données partagées. Récemment, les ontologies de domaine sont apparues comme un moyen pour décrire les données (et parfois les méta-données) partagées via le Web en les enrichissant avec une description de leurs sens. "Une ontologie définit les termes utilisés pour décrire et pour représenter un secteur de connaissance. Les ontologies sont utilisées par les êtres humains, les bases de données et les applications ayant besoin de partager des informations d'un domaine (comme la médecine, la fabrication d'outil, l'immobilier, la réparation automobile, la gestion financière, etc.)"[23]. Grâce à leur représentation explicite de la sémantique, les ontologies de domaine peuvent

[23] http://www.w3.org/TR/webont-req/#onto-def

être considérées comme un modèle de représentation de données ''convivial'' aux utilisateurs. Quant au schéma de placement, il faut le remplacer par une technique dynamique capable de s'adapter avec la grande-échelle et la nature dynamique des systèmes P2P. Comme nous avons déjà constaté dans le chapitre précédent, la technologie de DHT sur laquelle sont basés les systèmes P2P structurés est très convenable pour jouer le rôle de schéma de placement dynamique et capable de s'adapter avec la grande-échelle des systèmes P2P. Dans les deux sous-sections suivantes, nous présentons nos choix concernant le type de l'ontologie de domaine utilisée et la technique P2P que nous adoptons et nous justifions nos choix.

3.3.1. Ontologie de domaine au lieu d'un schéma global

D'un côté, le schéma conceptuel est une description unifiée et globale de toutes les données et de toutes relations entre les données. Il donne une vision précise sur les données partagées au sein d'un système. Dans ce schéma, l'importance est de représenter les relations entre les données. Le schéma conceptuel ne contient aucune représentation explicite de la sémantique portée par ses termes. Pour cela, le sens de certains termes dans le schéma conceptuel peut être ambigu.

De l'autre côté, une ontologie de domaine est une « description intentionnelle de ce qui nous connaissons autour de l'essence des entités d'un domaine particulier en utilisant des concepts et des relations entre ces concepts. » [SuL06]. Elle organise, sous forme d'un graphe, les connaissances sur le domaine en regroupant les objets du domaine en sous-catégories suivant leurs caractéristiques essentielles. Dans les systèmes d'informations, l'ontologie de domaine est principalement utilisée comme une terminologie partagée par les utilisateurs pour la description explicite et cohérente de leurs connaissances. Elle restreint l'interprétation des concepts qu'elle définit au contexte spécifié par le domaine. Ceci a l'avantage de limiter l'ambiguïté des termes définis dans l'ontologie.

Malgré cette séparation fonctionnelle, il existe des liens d'interaction entre ces deux notions. Comme par exemple : (i) le modèle Entité-Association utilisé dans la description conceptuelle des données manipulées dans un SBD forme un schéma sémantique ressemblant à l'ontologie de domaine, (ii) dans le contexte du Web Sémantique, l'ontologie de domaine est utilisée comme un schéma pour d'intégration de données. Le principe d'un tel schéma est de fournir une interface unique pour l'interrogation de sources de données hétérogènes.

Pratiquement, une ontologie de domaine est plus générale et sémantiquement plus riche qu'un schéma conceptuel. Dans le contexte de notre travail, nous utilisons une ontologie de domaine comme un schéma conceptuel. Cette ontologie forme principalement une terminologie partagée par les utilisateurs pour la description explicite et cohérente de leurs connaissances. Une telle ontologie permet de créer facilement un point de départ pour un échange compréhensif de données. De nos jours, pour un domaine donné, on peut trouver une ontologie de domaine disponible sur le Web. Par exemple, au moment de la rédaction de cette thèse, le moteur de recherche basé sur le Web sémantique Swoogle[24] effectue ses opérations de recherche dans plus de 10 000 ontologies. Avec le temps et dans le prochain futur, on peut trouver pour chaque domaine une ontologie. Ceci rend l'utilisation d'une ontologie de domaine plus pratique qu'un schéma conceptuel.

Le choix de l'ontologie

Actuellement, "Le Web Sémantique est une vision pour l'avenir du Web dans lequel on donne une signification explicite aux informations"[25]. Le langage d'ontologie pour le Web, désormais OWL (Web Ontology Language) est l'une des cinq recommandations de W3C (World Wide Web Consortium)[26] pour le Web Sémantique. Les ontologies écrites par le OWL sont largement utilisées dans la modélisation des ressources distribuées sur le Web. Etant donné que les nœuds d'un système P2P partagent leurs ressources via le Web et que nous avons besoin d'un modèle de représentation de données riche en sémantique, nous avons choisi d'utiliser une ontologie écrite par OWL dans notre travail. Avant d'expliquer comment utiliser cette ontologie, nous présentons un bref aperçu sur ce type d'ontologie.

Une ontologie est "une représentation des termes et de leurs interrelations"[21]. Le langage OWL est capable de représenter la sémantique plus que XML, RDF, et RDF-S. Il représente une ontologie en utilisant des classes, des propriétés et des types de données (*datatypes*). Les classes sont utilisées pour représenter les concepts de l'ontologie de domaine. Les propriétés sont divisées en deux catégories : (i) propriétés des objets (*objet properties*) qui représentent les relations entre les classes et (ii) propriétés des types de données qui relient une classe avec des types de données (e.g. chaîne de caractère, nombre entier). Les relations entre les classes forment une structure hiérarchique en utilisant des relations inter- classes de type (e.g. isA). Deux classes peuvent être ''équivalentes''. "L'égalité peut être utilisée pour

[24] Swoogle homepage, http://swoogle.umbc.edu/
[25] http://www.w3.org/TR/2004/REC-owl-features-20040210/#s1.2
[26] W3C homepage, http://www.w3.org/

créer des classes synonymes"[27]. En outre, chaque classe pourrait avoir une définition spécifique. Par exemple, "*Doctor*" peut être défini comme "une personne formé pour traiter les personnes malades. Deux propriétés peuvent aussi être équivalentes. "L'égalité peut être utilisée pour créer des propriétés synonymes"[23].

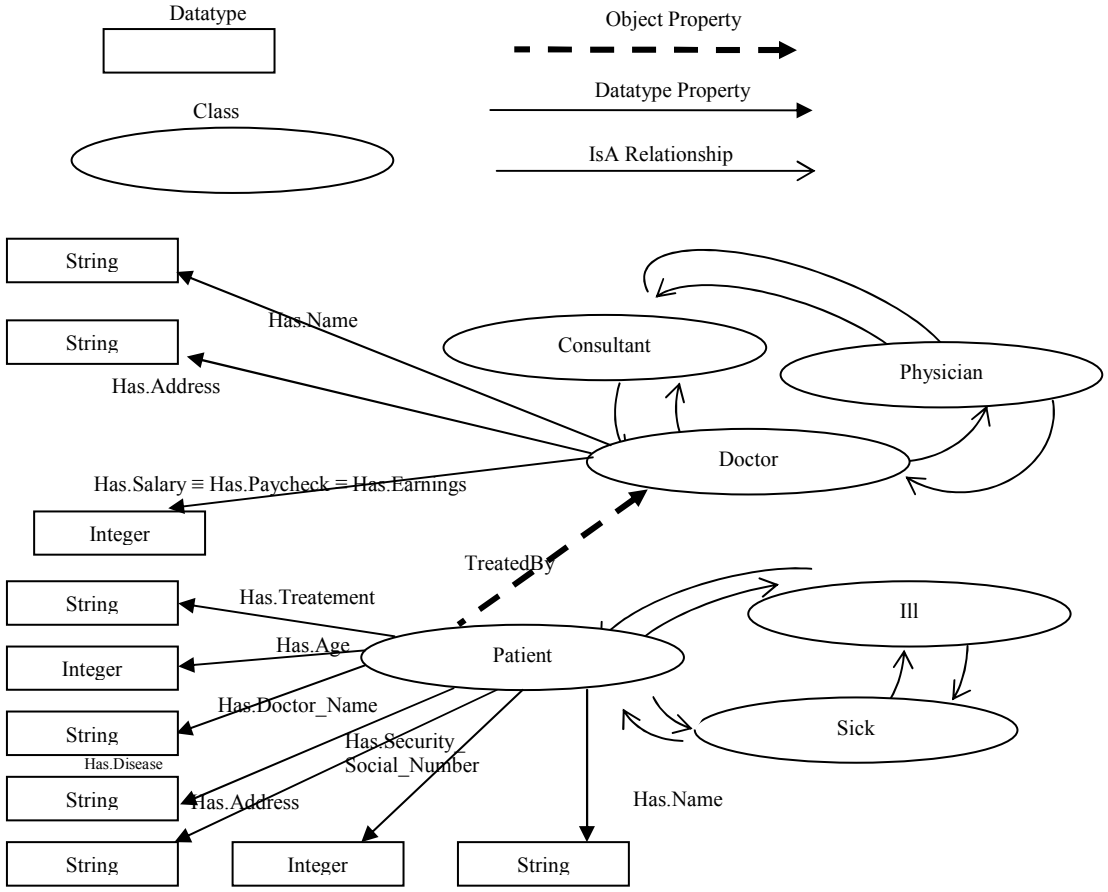


Fig. 9. Une partie de l'ontologie de domaine utilisée dans notre travail

3.3.2. Choix de la topologie du système et de l'algorithme de routage

Etant donné qu'avec l'absence de catalogue global indiquant le placement de chaque donnée dans le système, la localisation de sources de données est une phase délicate lors d'une évaluation de requête. Afin de localiser ces sources, on peut utiliser les techniques utilisées dans les systèmes P2P de partage de fichiers. Comme nous avons vu dans le chapitre précédent, l'efficacité de ces techniques dépendent de la topologie du système P2P considéré et de l'algorithme utilisé pour le routage de requêtes entre les nœuds. Le mot ''efficacité'' ici signifie un nombre minimal de messages échangés entre les nœuds et un temps de localisation réduit. Nous préférons toujours les techniques permettant de trouver

[27] http://www.w3.org/TR/2004/REC-owl-features-20040210/#s3.2

toutes les réponses valides existantes dans le système qui conviennent les plus aux requêtes évaluées.

Nous rappelons que les systèmes P2P sont classés, selon la topologie de leurs nœuds, en trois types: (i) non structurés, (ii) structurés et (iii) super-pairs. Nous constatons que l'autonomie de nœuds est mieux respectée dans les systèmes P2P non-structurés. Par contre, malgré les différentes recherches menées pour améliorer l'efficacité des techniques de routage de requêtes utilisées dans ces systèmes, les solutions proposées restent limitées. Le grand nombre de messages échangés lors de la localisation de sources de données et l'ignorance du grand nombre de réponses existantes dans le système sont les principaux inconvénients de ces systèmes. Les systèmes P2P structurés sont capables de retourner toutes les réponses existantes tout en maintenant un nombre relativement petit de messages échangés lors de la localisation de sources de données. Le prix à payer pour cette amélioration est que le niveau de l'autonomie des nœuds est relativement diminué. Les nœuds ne sont plus libres dans le choix de leurs voisins et le choix du nombre de ces voisins. Les systèmes super-pairs sont des systèmes hybrides entre les deux paradigmes Client/Serveur et P2P. Les techniques utilisées pour localiser les sources de données sont efficaces lorsque les hôtes des sources de données recherchées et le nœud initialisant la requête ont le même super-pair. Cependant, dans le cas inverse, l'efficacité des ces techniques dépend du type du système P2P reliant les super-pairs. Les systèmes P2P super-pairs sont moins tolérants aux pannes par rapport aux systèmes P2P structurés et non structurés. Lorsqu'un super-pair tombe en panne, tous ses clients (les nœuds dont il est responsable) seront privés des services fournis par le système. Etant donné que nous nous intéressons aux techniques les plus efficaces et les plus tolérantes aux pannes plus que celles respectant les plus l'autonomie de nœuds, nous choisissons un système P2P structuré. Une question se pose ici, c'est qu'elle est l'algorithme de routage que nous allons utiliser pour localiser les sources de données ?

Dans notre travail, nous utilisons un système P2P structuré basé sur le protocole Chord [SMK+01]. Ce type de systèmes s'adapte de manière performante avec la nature dynamique et la grande échelle de l'environnement P2P. En effet, Pastry [RoD01a] est un autre système P2P structuré qui ressemble beaucoup à Chord. Il utilise la géométrie de l'anneau et une DHT. Il permet également un routage de requêtes ayant une complexité de l'ordre de O(Log(N)). Sa complexité de l'entrée est $O(Log_{2^b}(N))$ et de la sortie $O(Log_b(N))$ où b est choisi d'une façon que chaque nœud a un identifiant représenté sur 2b bits (dans le

protocole Chord la complexité de l'entrée/sortie est $(Log_2 N)^2$. La DHT dans Pastry contient de plus de la table de routage et des paires (clés, identifiants), une troisième partie concernant les adresses IP des voisins. Pastry tient en compte la localisation physique des nœuds afin de choisir le successeur (on a plusieurs successeurs et pas un seul comme Chord) le plus proche en utilisant des métriques réseau. Donc, Pastry ne sépare pas le réseau logique du réseau physique. Cela rend la maintenance de DHT est plus compliqué. En environnent distribué à grande échelle, c'est très difficile de garder les métriques du réseau à jour. La maintenance de la DHT est de $O(\frac{1}{b}(2^b-1)Log_2(N))$ par contre dans le cas de Chord est de l'ordre de $O(2Log_2(N))$ sauts. Par conséquent, Chord est plus simple et plus pratique en termes de maintenance de DHT que Pastry même si ce dernier est plus efficace en termes de l'entrée et de la sortie des nœuds.

Afin d'utiliser Chord pour la localisation de sources de données, les nœuds doivent s'organiser selon un anneau. Chaque nœud a un identifiant sur cet anneau. Chaque entité de données partagée (e.g. relation) doit être représentée par une clé. Le nombre de bits nécessaires pour créer une clé est égal au nombre de bits nécessaires pour représenter les identifiants. Pour cette raison, il faut choisir un nombre suffisant *m* de bits pour représenter un nombre important de nœuds. Par exemple, pour $m = 64$ on peut représenter 2^{64} nœuds. Les clés ne dépendent ni du nombre d'entités de données existantes dans le système ni de nombre de nœuds. Ceci permet au protocole Chord de gérer un grand nombre d'entités de données réparties sur un nombre important de nœuds. Les identifiants de nœuds et les clés sont créés en utilisant la même fonction de hachage qui est connue par tous les nœuds. Lors de son entrée dans le système, chaque nœud doit utiliser la fonction de hachage pour créer une clé pour chaque entité de données partagée. Les clés sont injectées dans le système en utilisant la fonction de hachage. Chaque clé doit être stockée avec l'identifiant du nœud qui la crée (le hôte de la source stockant la donnée représentée par la clé) sur le nœud appelé le responsable de la clé. Le responsable d'une clé est le nœud qui connaît l'identifiant du hôte de la source de cette clé. Les paires (clé, identifiant) forment une table de hachage distribuée sur tous les nœuds. Afin de localiser la source d'une entité de donnée sur un nœud p, il faut tout d'abord créer une clé pour elle et ensuite chercher le nœud responsable de cette clé. Le responsable de la clé répond à la demande de localisation en envoyant au nœud demandant l'identifiant correspondant à la clé recherchée. Pour trouver le responsable de la clé, le protocole Chord possède un algorithme efficace de routage. Chaque nœud doit connaître les intervalles des clés dont ses voisins sont responsables. Les voisins de chaque nœud sont

choisis par le système. Les voisins avec les intervalles dont ils sont responsables sont stockés dans des tables de routage. Alors, chaque nœud possède une partie de la table de hachage pour connaître les sources des clés dont le nœud est responsable et une table de routage pour guider le routage de messages. Afin de localiser la source d'une clé, un nombre de messages de l'ordre de O(LogN) est nécessaire, où N est le nombre total de nœuds. Etant donné la requête SQL suivante sur le nœud N8 : ***Select Name From Doctor Where Salary = 2000 ;*** la figure 10 illustre la façon dont Chord localise la source de la clé C54 qui représente la relation ''*Doctor*''. En utilisant les tables de routage, la demande de localisation arrive au nœud N56 après avoir visité les nœuds N42 et N51. Le N56 est le nœud responsable de la clé C54. Ce nœud consulte sa partie de DHT et envoie la réponse {N1, N8, N21, N42} à N8.

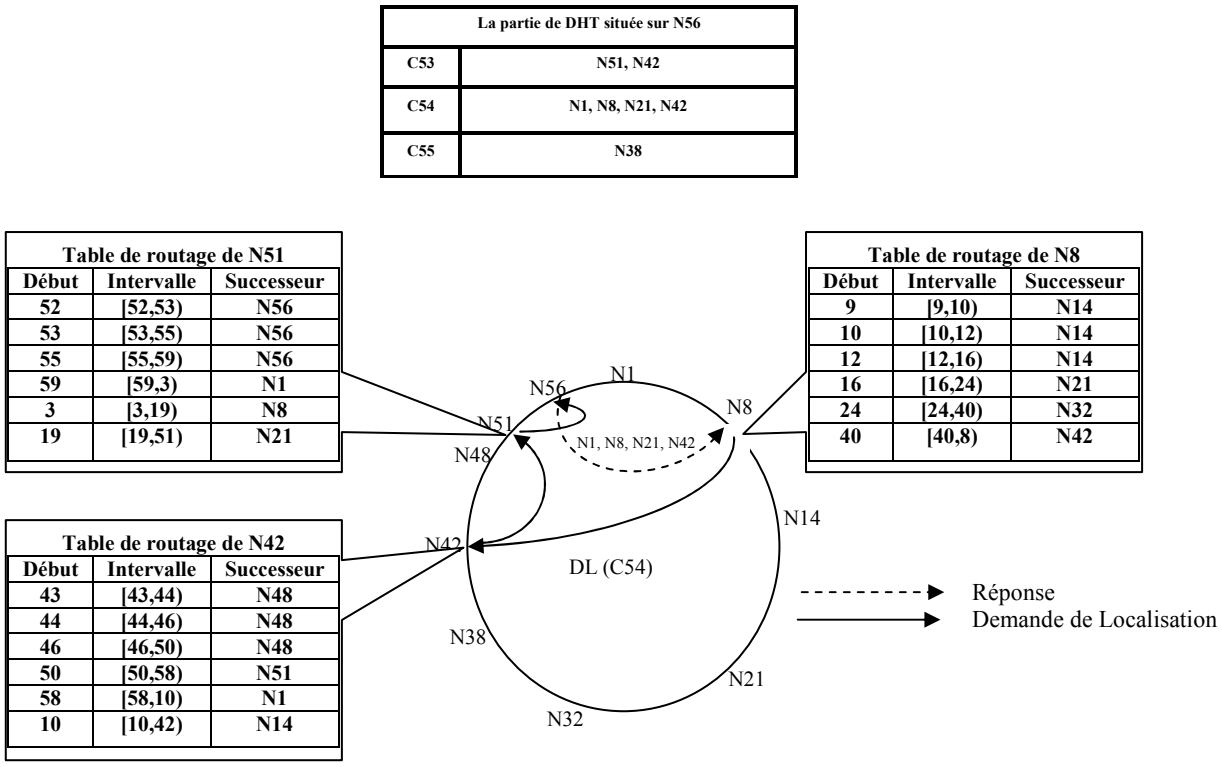


Fig. 10. Localisation des sources de la clé C54 selon le protocole Chord

Traditionnellement, aucune sémantique ne relie les clés et les relations représentées par ces clés. Dans la requête le mot ''*Doctor*'' signifie un médecin. Par contre, dans la réponse du protocole Chord, nous trouvons N21 qui contient une relation nommée ''*Doctor*'' mais elle porte le sens d'une personne ayant soutenu une thèse. Une autre limite de ce protocole est que les clés ne prennent pas en compte les structures des relations sur leurs sources. Par

exemple, le nœud N42 contient une relation nommée ''*Doctor*'' et portant le même sens que celle stockée sur N8 mais cette relation ne contient pas l'attribut ''*Salary*''. Alors, même si le nœud N42 stocke la relation ayant la sémantique demandée par N8, cette relation n'a pas la structure demandée par N8. Ces deux limites fonctionnelles empêchent le protocole Chord de sélectionner les hôtes des sources de données pertinentes pour répondre à une requête donnée. Ainsi, pour que le protocole Chord puisse dépasser ses limites, il faut étendre ce protocole.

Avant d'expliquer l'extension du protocole Chord, nous présentons dans la section suivante, l'architecture logicielle du système considéré par notre étude. Nous expliquons également comment une requête SQL peut être évaluée au sein de cette architecture et le rôle de l'ontologie de domaine et celui du protocole Chord dans cette évaluation.

3.3.3. Architecture logicielle pour l'évaluation de requêtes SQL en environnement P2P

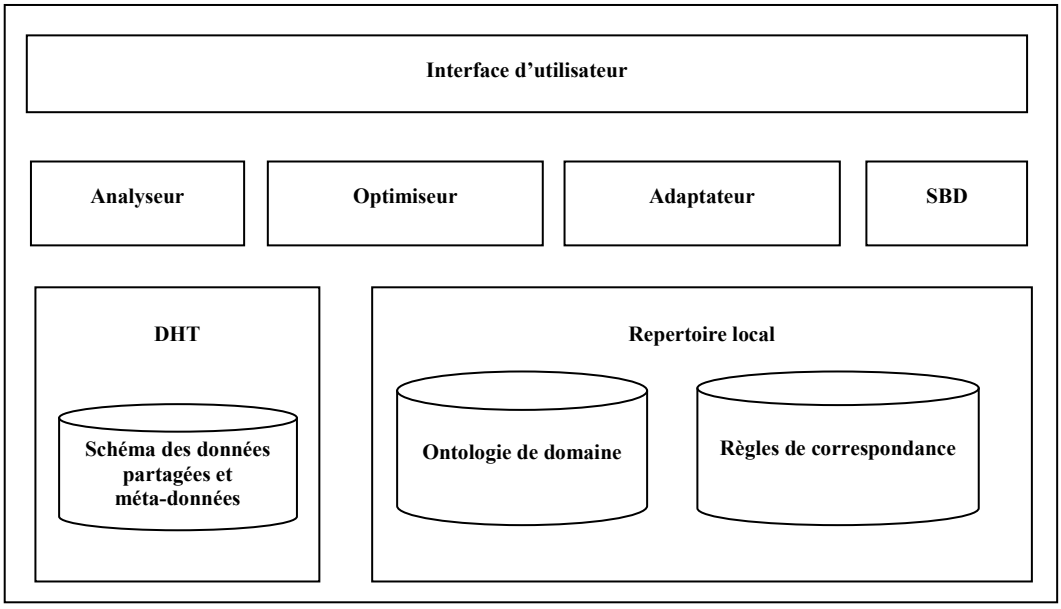


Fig. 11. Architecture logicielle dupliquée sur chaque nœud

Cette section est consacrée à présenter une approche pour évaluer des requêtes SQL en environnement P2P. Dans un tel environnement, une ontologie de domaine est responsable d'assurer un échange de données d'une façon compréhensible en représentant explicitement la sémantique des données partagées. Elle forme l'interface unique pour l'interaction entre les nœuds. L'utilisation de l'ontologie de domaine n'impose aucune contrainte sur les schémas locaux. Les termes de l'ontologie de domaine sont utilisés pour

décrire la sémantique des données partagées. Lorsqu'un nouveau nœud "imprévu" veut participer au système, l'utilisateur de ce nœud crée des règles de correspondance entre le schéma local de ce nœud et l'ontologie de domaine. Ces règles sont stockées localement sur le nœud de cet utilisateur. Avant d'être capable de faire participer son nœud au système, l'utilisateur doit publier une partie de son schéma représentant les données partagées avec les autres nœuds. Pour cette raison, le protocole Chord est utilisé pour générer pour chaque relation (écrite en termes de l'ontologie de domaine) une clé. Pour chaque clé, il y a un nœud responsable qui connaît l'hôte de la source de cette clé. Chaque nœud est responsable d'injecter ses clés au système au moment de sa participation au système et il est responsable de retirer ses clés au moment de son départ. En regardant la figure 11, nous résumons notre approche comme suit.

L'utilisateur soumet une requête SQL en utilisant l'interface d'utilisateur (IU) qui offre la capacité d'écrire la requête et de choisir où est ce que la requête va être exécutée localement ou globalement. Dans la suite, nous décrivons seulement, l'évaluation de requête global parce que la requête locale peut être exécutée comme dans les systèmes de bases de données traditionnels. Etant donné une requête écrite selon le schéma local du nœud initialisant la requête (NIR). Après l'analyse de la requête, une phase de reformulation est nécessaire afin de réécrire la requête en termes de l'ontologie de domaine qui est connue par tous les nœuds. Un adaptateur est responsable d'effectuer cette phase en utilisant les règles de correspondance créées au moment de la participation au système. Ces règles sont stockées dans un répertoire local. L'objectif de cette phase est de rendre la requête compréhensive par les autres nœuds dans le système. Après cette phase, une phase de localisation de sources de données et d'obtention de méta-données est effectuée en utilisant une version étendue du protocole Chord. L'hétérogénéité structurelle est résolue durant cette phase grâce aux indexes des structures. Après la réception des méta-données des hôtes des sources pertinentes, une phase d'optimisation globale est effectuée sur le NIR. Le plan d'exécution est décomposé en sous-plans qui sont envoyés aux hôtes des sources de données. Sur chaque source, une autre phase d'optimisation concernant le sous-plan reçu est effectuée. Ensuite, les sous-plans d'exécution sont exécutés et finalement les résultats sont envoyés aux nœuds désignés par le plan d'exécution global. Après l'exécution de la requête, les résultats sont assemblés sur le NIR , réécrits en termes du schéma local de NIR et délivrés finalement à l'utilisateur.

3.4. Règles de correspondance entre les schémas locaux et l'ontologie de domaine

Dans le but de créer des règles de correspondance entre un schéma local et l'ontologie et de modifier ces règles à n'importe quel moment, l'utilisateur doit créer pour chaque relation et pour chacun de ses attributs des synonymes (cf. figure 12). Par exemple, pour le mot ''*Doctor*'', l'utilisateur peut créer les synonymes ''*Physician*'' et ''*Consultant*''. En se basant sur ces synonymes, nous proposons des formules de similarité pour aider l'utilisateur à trouver pour chaque relation (respectivement pour chaque attribut) les classes (respectivement les propriétés de types de données) correspondantes dans l'ontologie de domaine. Dans la suite, nous expliquons les formules de similarité que nous avons proposées.

Nœuds	Schémas locaux	Synonymes
N1	Doctor	Physician
N8	Doc Traet	Physician, Doctor, Consultant Treatement
N14	Physician Patient	Doctor, Consultant Ill, Sick

Fig. 12. Quelques synonymes ajoutés par des utilisateurs

Formules de similarité

Nous expliquons les formules de similarité comme suit. Etant donné $R(a_1, a_2, \ldots a_I)$ une relation appartenant à un schéma local, où $a_{1\leqslant i\leqslant I}$ sont ses attributs et $C(d_1, d_2, \cdots d_J)$ est classe appartenant à l'ontologie de domaine, où $d_{1\leqslant j\leqslant J}$ sont ses propriétés de types de données. La similarité totale $0\leqslant SimTotal(R,C)\leqslant 1$ entre R et C peut être donnée par l'équation suivante :

$$Sim_{Total}(R,C) = \alpha * SimRC + (1-\alpha) * SimAD \quad \text{with} \quad 0 \leq \alpha \leq 1 \qquad (1)$$

où le terme **SimRC** représente la similarité au niveau des relations et des classes, le terme **SimAD** représente la similarité au niveau des attributs et des propriétés de types de données. α est une valeur indiquant l'importance de chaque niveau de similarité par rapport à l'autre. Par exemple, dans notre étude α = 0.5. Ceci signifie que les deux niveaux de similarité ont la même importance. Pour une relation donnée R, nous calculons la similarité totale avec chaque classe C de l'ontologie de domaine. Les classes ayant une valeur de similarité totale

plus grande qu'un seuil donné, seront sélectionnées et délivrées avec leurs propriétés de types de données, leurs types de données et leurs définitions à l'utilisateur. L'utilisateur choisit, ensuite, les classes les plus appropriées pour représenter la relation R. Dans les lignes suivantes, nous allons voir comment calculer les deux termes.

Le terme SimRC : ce terme est calculé en utilisant les mots-clés créés par l'utilisateur et la mesure de Tversky présentée dans [Tve77]. Cette mesure est basée sur l'effet que deux objets sont plus similaires lorsqu'ils partagent plus de caractéristiques. Les deux objets sont moins similaires lorsque les caractéristiques similaires sont minimales. La mesure de Tversky est donnée par l'équation suivante :

$$Sim(a,b) = \frac{|A \cap B|}{|A \cap B| + \beta * |A \setminus B| + (1-\beta) * |B \setminus A|} \quad \text{with} \quad 0 \le \beta \le 1 \qquad (2)$$

B est une valeur indiquant l'importance des caractères communs par rapport aux caractères non communs. $|X|$ représente la cardinalité de l'ensemble X. En adaptant la mesure de Tversky [Tve77] avec notre contexte, nous obtenons l'équation 3 dans laquelle, Syn(R) est l'ensemble des synonymes de R crées par l'utilisateur, et Syn (C) est l'ensemble de sous-classes de C et des classes équivalentes à C. En effet, la similarité sémantique entre le nom d'une relation et le nome d'une classe dépend des synonymes communs entre l'ensemble de synonyme de R et l'ensemble de synonymes de C. Plus, les deux ensembles ont des membres communs, plus les deux noms sont sémantiquement similaires. En se basant sur cette idée, nous pouvons trouver dans l'ontologie de domaine la classe qui correspond le plus à la relation considérée dans un schéma local.

$$SimRC = \frac{|Syn(R) \cap Syn(C)|}{|Syn(R) \cap Syn(C)| + \beta * |Syn(R) \setminus Syn(C)| + (1-\beta) * |Syn(C) \setminus Syn(R)|} \qquad (3)$$

Les sous-classes de C peuvent être considérées comme synonymes de C. Par exemple, dans une ontologie médicale, "*Dermatologist*'', qui est une sous-classe de ''*Doctor*'' peut être considérée comme synonyme de ''*Doctor*''. Mais, on ne peut pas considérer une classe comme synonymes de sa sous-classe. Un ''*Doctor*'' n'est pas forcément un ''*Dermatologist*''.

Le terme SimAD : ce terme calcule la similarité au niveau des attributs et des propriétés de types de données. Il est donné par l'équation 4 :

$$SimAD = \frac{1}{I} * \sum_{i=1}^{I} \underset{1 \le j \le J}{Max} \left(Sim(a_i, d_j) \right) \qquad (4)$$

Dans cette équation, $Sim(a_i, d_j)$ est la similarité entre un attribut a_i et une propriété de type de données d_j. Cette similarité est calculée en utilisant la mesure de Tverskey en remplaçant A par Syn (a_i) et B par Syn(d_j). Syn(a_i) est l'ensemble des synonymes créés par l'utilisateur et Syn(d_j) est l'ensemble des propriétés de types de données équivalentes à d_j.

L'ontologie de domaine et les formules de similarité permettent de résoudre l'hétérogénéité sémantique entre les schémas locaux. Etant donné une requête reformulée en termes de l'ontologie de domaine, afin de localiser les sources de données mentionnées par cette requête, il faut extraire les noms des relations, trouver la clé de chaque relation en utilisant la fonction de hachage et utiliser Chord pour trouver les nœuds stockant les relations. En effet, les relations peuvent être répliquées et structurellement hétérogènes. Chord ne peut pas résoudre ce type de l'hétérogénéité. Pour cette raison, nous avons proposé d'enrichir Chord par des Indexes de structure permettant de décrire pour chaque clé la structure des relations représentées par cette clé. Dans la section prochaine, nous allons expliquer le principe des Indexes de structure.

3.5. Indexes de structure

Afin d'effectuer la localisation de sources de données, nous proposons d'étendre le protocole Chord. Nous proposons que chaque clé représente un concept unique dans l'OD. Ce concept peut représenter, à son tour, plusieurs relations stockées sur plusieurs nœuds. Ces relations sont sémantiquement similaires. Etant donnée une clé, le protocole Chord est capable de localiser les sources des relations ayant la sémantique demandée par le NIR. Cependant, Chord reste incapable de choisir parmi ces relations celles qui ont les structures des relations demandées par le NIR. Ainsi, il y a une nécessité d'enrichir la DHT par des informations permettant de décrire les structures des relations sur leurs hôtes. Pour cela nous proposons d'associer à chaque clé c un index de structure IS(c).

Table des IS du N1	
C57	(N38, {Nom, Age}), (N42, {Nom, Age})
C58	(N8, {Nom, Adresse, Nom_Médecin}), (N14, {Nom, Age}), (N48, {Nom, Adresse, Nom_Médecin})
C59	(N38, {Nombre, Service}), (N42, {Nombre, Service})

Table des IS du N56	
C53	(N38, {Nom, Age}), (N42, {Nom, Age})
C54	(N1, {Nom, Salaire}), (N8, {Nom, Salaire}), (N14, {Nom, Adresse}), (N32, {Nom, Salaire, Téléphone}), (N42, {Nom, Adresse})
C55	(N38, {Nombre, Date}), (N42, {Nombre, Montant})

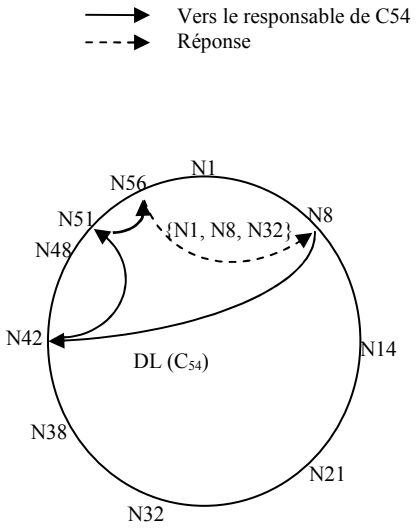


Fig. 13. Localisation de la clé C54 représentant le concept ‘‘médecin’’ dans l’OD

Cet index contient les différentes structures des relations représentées par le concept associé à la clé c. IS(c) permet au responsable de la clé c de sélectionner les hôtes des sources pertinentes pour répondre à la requête. Ces hôtes stockent les sources des relations ayant les sémantiques et les structures demandées par le NIR. Dans la figure 13, le concept *‘‘Doctor’’* est associé à la clé C54. Le nœud N56 qui est le responsable de cette clé contient l’index IS(C54). Cet index décrit les structures de toutes les relations représentées par le concept *‘‘Doctor’’*.

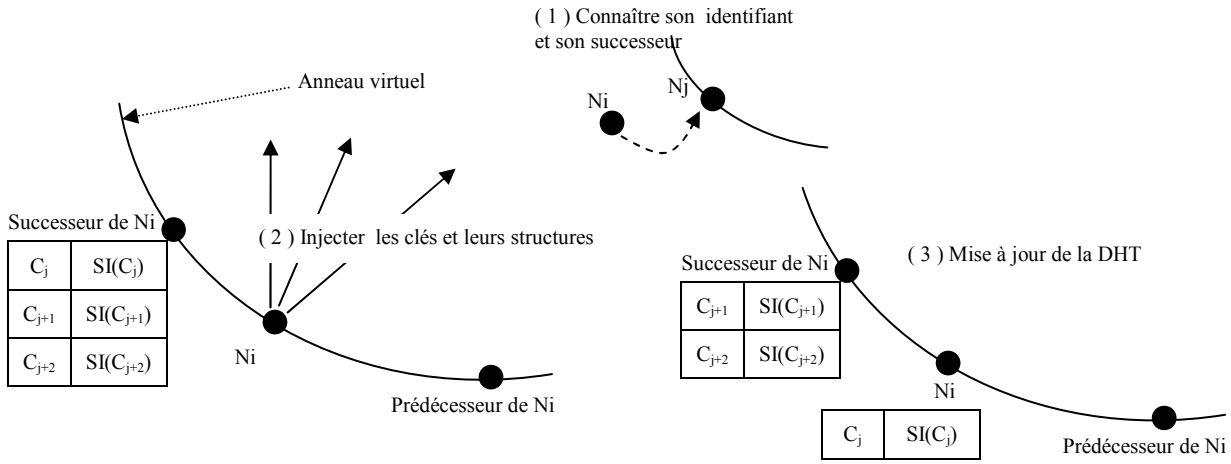


Fig. 14. La connexion du nœud Ni au système

Chaque IS(c) est créé au même moment de la création de la clé c. Il est stocké sur le nœud responsable de la clé c. Lors de sa connexion au système, chaque nœud est responsable d'injecter les clés représentant ses relations et les structures décrivant les relations représentées par ces clés. Il est aussi responsable de leur suppression du système lors de sa déconnexion. Dans la figure 14, le nœud Ni veut se connecter au système, il contacte un autre nœud Nj du système. Chord associe Ni avec son identifiant sur l'anneau virtuel et l'informe de son successeur. Dès que le nœud Ni devient un membre du système, il commence à injecter ses clés et les indexes de structure correspondants dans le système. Après l'injection, la DHT doit être mise à jour de la même manière que celle expliquée dans [SMK+01]. Au moment de sa déconnexion, le nœud Ni utilise Chord pour localiser les responsables actuels des clés qu'il a déjà injectés lors de sa connexion au système. Le nœud sortant Ni informe les responsables de ses clés de son départ afin qu'ils puissent supprimer les clés et les indexes de structures associés à ces clés.

L'algorithme de locallisation

Dans le protocole Chord, la demande de localisation (notée DL) est constituée de la clé recherchée et de l'identifiant de NIR. Par contre, nous étendons la demande de localisation en lui ajoutant les noms des attributs (écrits selon l'OD) appartenant à la relation représentée par la clé. Ces informations ajoutées à la DL permettent au nœud responsable de la clé de sélectionner les hôtes des sources qui stockent les relations ayant ces attributs. Le routage de demande de localisation est constitué des étapes suivantes :

- **(i)** Localisation du nœud responsable de la clé : cette étape est similaire à celle effectuée par le protocole Chord avant l'extension.
- **(ii)** Sélection les hôtes des sources pertinentes (SP) pour répondre à la DL: le nœud responsable de la clé consulte l'index de structure de la clé recherchée afin de choisir les nœuds stockant des relations ayant les attributs référencés par la DL.
- **(iii)** Envoi de la réponse au NIR pour continuer l'évaluation de la requête SQL soumise sur ce nœud.

OD : Ontologie de Domaine
NIR : Nœud Initialisant la Requête
DL(c) : Demande de Localisation concernant la clé c
SP(Ri) : Sources Pertinentes qui stockent la relation Ri
HSP(Ri) : Hôtes des Sources Pertinentes qui stockent la relation Ri
PPP(c) : le nœud ayant l'identifiant Précédent le Plus Proche de la clé *c* sur l'anneau virtuel du protocole Chord.
Res(c) : le nœud ayant l'identifiant qui succède directement la clé *c* sur l'anneau virtuel du protocole Chord, il est appelé aussi successeur de la clé *c*

Fig. 15. Rappel des symboles utilisés dans l'algorithme de localisation

Dans la figure 16, nous présentons l'algorithme de localisation. Les symboles utilisés dans cet algorithme sont présentés dans la figure 15. Pour mieux comprendre l'exécution de cet algorithme, l'exemple suivant explique les différentes étapes de la phase de la localisation.

L'algorithme de localisation

Entrée : $\{(Ri/\ att_1,\ att_2,\ \ldots att_j)_{j=1,2,..m}\}_{i=1,2,..,n}$ relations existantes dans une requête SQL reformulée et écrite selon l'OD
Sortie : $\{Identifiants\ des\ HSP(Ri)_{\ i=1,2,..,n}\}$

Début
Pour chaque *Ri* **faire** *Ci* ← *Engendrer_clé(Ri) // Génération des clés représentants les relations*
Pour chaque Ci **faire**
Début
Si NIR n'est pas le *Res(Ci)* **alors**
Début
N ← PPP(Ci)
Envoyer DL(Ci, NIR, $\{ att_j\}_{j=1,2,..m}$) à N

/* Si N n'est pas le *Res (Ci)*, envoyer DL à un nœud ayant un identifiant le plus proche à Ci [STO01]. Cette étape est répétée jusqu'à ce que la DL arrive sur le Res (Ci) qui sélectionne les HSP(Ri) et répond à la DL en envoyant les identifiants des HSP(Ri) au NIR */

Recevoir les identifiants des HSP(Ri)
Fin_Si
Sinon sélectionner SP(Ri) // le NIR est le Res(Ci)
Résultat [i] ← identifiants des HSP(Ri)
Fin_Pour
Fin

Fig. 16. L'algorithme de localisation

3.5.1. Exemple

Pour illustrer l'algorithme présenté dans la figure 16, nous supposons qu'on ait les schémas relationnels représentés par la figure 8, la relation "*Doctor*" sur le nœud N21

représente une personne ayant soutenu une thèse, cette relation est représentée par le concept ''*PHD person*'' dans l'OD. Cependant sur les autres nœuds, les relations "*Consultant*", "*Physician*" et "*Doctor*" sont représentées par le seul concept ''*Doctor*'' dans l'OD. Les attributs "*Salaire*", "*Mensualité*", "*Récompense*" sont représentés par le concept ''*Salaire*'' dans l'OD. Les relations ''Patient'' sur N8, ''*Malade*'' sur N14 et ''*Malade*'' sur N48 sont représentées par le concept ''*Malade*'' selon l'OD. Les attributs ''*Doctor*'' et ''*Nom-Doctor*'' sont représentés par le concept ''*Nom-Médecin*'' selon l'OD. Etant donné la requête SQL suivante sur le nœud N8:

Q: Select Patient.nom, Patient.adresse, Doctor.nom
From Patient, Doctor
Where Patient.Nom_Doctor = Doctor.nom
And Doctor.Récompense = ''2000€''

La requête Q est écrite selon le schéma local de N8. Afin que la requête Q soit compréhensive par les autres nœuds, elle doit être réécrite par le médiateur de N8 en termes de l'OD. La reformulation de Q est effectuée à deux niveaux, au niveau des relations et au niveau des attributs. Le symbole '' ≡ '' signifie ''ayant pour concept dans l'OD''

Au niveau des relations :
Dans R1 : Patient ≡ Malade,
Dans R2 : Doctor ≡ Médecin

Au niveau des attributs :
Dans R1 : Nom ≡ Nom, Adresse ≡ Adresse, Nom_Doctor ≡ Nom_Médecin
Dans R2 : Nom ≡ Nom, Récompense ≡ Salaire

Supposons que la clé C54 représente le concept ''Médecin'', selon le protocole Chord nous remarquons dans la figure 13 que le nœud N56 est le responsable de cette clé. Parmi les hôtes des sources stockant la relation ''Médecin'', le nœud N56 sélectionne celles ayant les attributs référencés dans Q. Le nœud N56 envoie la réponse {N1, N8, N32} au NIR. Dans la figure 13, Nous remarquons que le nœud N21 n'existe pas dans l'index IS(C54) car la relation ''*Doctor*'' du nœud N21 n'est pas attaché au concept ''Médecin'' dans l'OD. D'une manière similaire, Chord localise les sources stockant le concept ''Malade'' qui est représenté par la clé C58. La réponse de la demande de localisation concernant la clé C54 est {N8, N48}.

Nous avons déjà constaté que l'extension proposée au protocole Chord impose deux modifications sur l'algorithme de routage et sur la DHT. En ce qui concerne l'algorithme de routage, nous n'imposons aucune augmentation sur le chemin parcouru par la demande de localisation. Ce chemin reste de l'ordre de O(log (N)) (N est le nombre total de nœuds dans le système). Cependant, des nouvelles informations sont ajoutées sur la demande de localisation. Ceci augmente la taille de la demande de localisation d'une quantité relativement petite. Supposons que chaque relation référencée par la requête SQL ait a attributs et la taille du nom de chaque attribut est o octets, la taille des informations ajoutées à la demande de localisation devient i= a*o octets. Si on choisit des valeurs raisonnables pour les paramètres a et o comme : 7 et 4 respectivement, on trouve que i = 28 octets.

L'enrichissement de la DHT par des indexes de structure IS augmente la taille moyenne de la DHT sur chaque nœud. Cependant, l'espace de stockage nécessaire pour faire cet enrichissement est relativement petit. Si on suppose que chaque nœud est responsable de e clés, donc, il doit stocker e index de structure. Supposons que chaque clé est représentée par r structures différentes et la taille moyenne de chaque structure est t octets. Alors, l'espace nécessaire pour faire l'enrichissement est s = e*r*t. Si on choisit des valeurs raisonnables pour les paramètres e, r et t comme : 100, 20 et 8, respectivement, on trouve que s = 15.625 Ko.

3.6. Obtention de méta-données

Généralement, selon le protocole Chord, chaque nœud, participant au système, stockes dans sa table de routage les informations suivantes : les identifiants des nœuds voisins et un intervalle des clés associés à chacun de ses voisins. Ces informations sont suffisantes pour effectuer la phase de localisation d'une manière efficace [SMK+01]. Dans la figure 18 qui illustre la façon dont Chord trouve le nœud responsable de la clé C54, dès que la demande de localisation (notée DL(C54)) arrive au nœud N56 qui est le responsable de la clé C54, ce nœud cherche la clé dans sa partie de la DHT la réponse au nœud N8. La réponse est constituée des identifiants des hôtes des sources de données pertinentes. Les informations retenues par la phase de localisation ne suffisent pas pour engendrer un plan d'exécution optimal (ou proche de l'optimal). De plus, les statistiques trouvées sur NIR (N8 dans notre exemple) peuvent être obsolètes ou elles ne contiennent pas toutes les informations concernant une certaine requête. Dans un environnement instable et réparti à grand échelle comme P2P, il n'est pas possible de mettre à jour les statistiques trouvées sur tous les nœuds après chaque modification.

Afin d'effectuer l'optimisation, on a besoin d'informations sur les données et sur l'environnement d'exécution pour engendrer un plan d'exécution proche de l'optimal. Ces informations forment les méta-données (MD) nécessaires pour la phase de l'optimisation, nous pouvons les classer en deux groupes :

(i) les paramètres concernant les caractéristiques physiques de données (PPD) : comme, pour chaque attribut : le nom de l'attribut, la longueur de l'attribut (en octets), la valeur minimale, la valeur maximale, le nombre de valeurs distinctes, et le nom de la relation à laquelle l'attribut appartient.

(ii) les paramètres physiques des hôtes des sources de données (PPHS) : comme, l'adresse IP, l'identifiant sur le réseau virtuel, la taille de mémoire, la bande passante E/S, la valeur CPU et la charge moyenne de la CPU.

L'algorithme de l'obtention de méta-données

Entrée : *{(Ri/ att_1, att_2, ...att_j)$_{j=1,2,..m}$ }$_{i=1,2,..,n}$* relations existantes dans une requête SQL reformulée et écrite selon l'OD
Sortie : *{ MD(Ri, HSP(Ri)) }$_{i=1,2,..,n}$ // MD(Ri, HSP(Ri)) :Méta-données concernant la relation Ri // et les hôtes des sources pertinentes concernant Ri (HSP(Ri))*

```
Début
Pour chaque Ri faire   Ci ← Engendrer_clé(Ri) //      Génération des clés représentant les relations
    Pour chaque Ci faire
        Début
          Si NIR n'est pas le Res(Ci) alors
                Début
                N ← PPP(Ci)
                Envoyer DL( Ci, NIR, { attj}j=1,2, ..m ) à N

                /* Si N n'est pas le Res (Ci), envoyer DL à un nœud ayant un identifiant
                le plus proche à Ci [STO01]. Cette étape est répétée jusqu'à ce que la DL
                arrive sur le Res (Ci) qui sélectionne les SP(Ri) et qui envoie, ensuite,
                une DMD aux HSP(Ri)*/

                Fin_Si
           Sinon                                        // le NIR est le Res(Ci)
                Début
                Sélectionner SP(Ri)
                Envoyer une DMD aux HSP(Ri)
                Fin_Sinon

        Recevoir les MD(Ri, HSP(Ri))
        Résultat [i] ← MD(Ri, HSP(Ri))
        Fin_Pour
Fin
```

Fig. 17. L'algorithme de l'obtention de MD

Pour obtenir les MD nécessaires pour générer un plan d'exécution proche de l'optimal, nous proposons d'étendre le protocole Chord pour supporter l'obtention de ces MD. Dans cet objectif, nous proposons de modifier l'algorithme de routage utilisé par Chord original pour qu'il puisse obtenir les paramètres physiques concernant les données et les hôtes des sources de données. L'algorithme d'obtention des MD est une extension de la méthode de localisation de sources de données illustrée dans la figure 16. Après la sélection des sources pertinentes par le nœud responsable d'une clé recherchée, ce nœud envoie la demande de localisation aux hôtes des sources pertinentes afin d'obtenir des méta-données à jour. La figure 17 illustre l'algorithme de recherche de MD. Par DMD, nous désignons Demande de Méta-Données.

Table des IS du N1	
C57	(N38, {Nom, Age}), (N42, {Nom, Age})
C58	(N8, {Nom, Adresse, Nom_Médecin}), (N14, {Nom, Age}), (N48, {Nom, Adresse, Nom_Médecin})
C59	(N38, {Nombre, Service}), (N42, {Nombre, Service})

Table des IS du N56	
C53	(N38, {Nom, Age}), (N42, {Nom, Age})
C54	(N1, {Nom, Salaire}), (N8, {Nom, Salaire}), (N14, {Nom, Adresse}), (N32, {Nom, Salaire, Téléphone}), (N42, {Nom, Adresse})
C55	(N38, {Nombre, Date}), (N42, {Nombre, Montant})

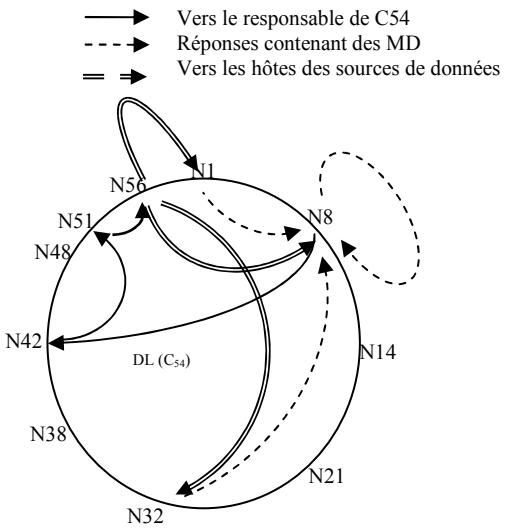


Fig. 18. Obtenir les MD concernant la clé C54 selon la méthode MOM

L'exemple suivant illustre comment la recherche des MD associées à une requête Q s'effectue selon l'algorithme présenté dans la figure 10.

3.6.1. Exemple

Supposons que les trois noeuds (N38, N48, N56) aient les schémas relationnels suivants :

N38 : Patient (SSN, Name, Pa_Doctor, Age)
N48 : Service (SrvNumber, Type)
N56 : Physician (Dr_SSN, Dr_Name, Speciality, Dr_Address),
Treatment (PtSSN , DrName, SrvNum, Cost)

Etant donné la requête SQL suivante émise depuis le nœud N8

Q : Select *Name*
from *Patient, Physician*
where *Patient.Pa_Doctor = Physician.Dr_Name*
And *Physician.Dr_Name="value"*

N_{38}	**$PPHS_{38}$**	**C_{33}/ SSN / Patient /PPD_{33}** **C_{35}/ Name / Patient / PPD_{35}** **C_{37}/ Pa_Doctor / Patient / PPD_{37}** **C_{100}/Age/ Patient / PPD_{100}**
N_{48}	**$PPHS_{48}$**	**C_{39}/SrvNumber/ Service / PPD_{39}** **C_{40}/Type/Service / PPD_{40}**
N_{56}	**$PPHS_{56}$**	**C_{52}/Dr_SSN/Physician/ PPD_{52}** **C_{54}/Dr_Name/Physician/PPD_{54}** **C_{55}/Speciality/Physician/ PPD_{55}** **C_{56}/Dr_Address/Physician/ PPD_{56}** **C_{116}/PtSSN/Treatment/ PPD_{116}** **C_{117}/DrName/Treatment/ PPD_{117}** **C_{118}/SvrNum/Treatment/ PPD_{118}** **C_{119}/Cost/Treatment/ PPD_{119}**

Fig. 19. Parties des catalogues locaux des hôtes des sources de données référencées par Q

Après l'application de la fonction Extraire (Q), on obtient les attributs suivants :

$$ATT = \{Name,\ Patient.Pa\_Doctor,\ Physician.Dr\_Name\}$$

L'application de la fonction Engendrer-clés (ATT), donne les clés CLE illustrées par l'ensemble suivant :

$$CLE = \{C_{35}, C_{37}, C_{54}\}$$

Après l'extraction des attributs et la génération des clés correspondantes, on recherche les MD concernant chaque clé appartenant à CLE. Les résultats obtenus sont présentés par la table suivante :

Clé	Source	PPD	PPHS
C_{35}	N_{38}	PPD_{35}	$PPHS_{38}$
C_{37}	N_{38}	PPD_{37}	$PPHS_{38}$
C_{54}	N_{56}	PPD_{54}	$PPHS_{56}$

Le nœud N38 envoie les PPD et les PPHS concernant les clés (C35 et C37) au nœud N8 et le nœud N56 envoie les PPD et les PPHS concernant la clé (C54) au nœud N8. Après avoir obtenu ces MD, l'optimiseur peut générer un plan d'exécution proche de l'optimal.

3.7. Conclusion

Dans ce chapitre, nous avons décrit une approche pour l'évaluation de requêtes en environnement P2P. Le principe de cette approche est d'utiliser une ontologie de domaine, dupliquée sur tous les nœuds, pour représenter la sémantique de données partagées par tous les nœuds. Les termes de l'ontologie de domaine forment une interface unique pour les interactions entre les nœuds. L'utilisateur sur chaque nœud, afin de représenter la partie partagée de ses données, doit créer des synonymes pour chaque relation et pour chacun de ses attributs. Des formules de similarité sont proposées pour aider l'utilisateur à trouver pour chaque relation ses correspondants dans l'ontologie de domaine. Afin de localiser d'une façon efficace les sources de données, nous avons proposé d'étendre le protocole Chord. La DHT, sur laquelle Chord est basée, est enrichie par des indexes de structure ayant pour rôle de représenter la structure en termes d'attributs de chaque relation partagée dans le système. Ceci permet de choisir lors de la localisation des sources de données les relations ayant la structure désirée par la requête. A cause de l'absence d'un catalogue global dans un système P2P et pour avoir des méta-données à jour pour créer un plan d'exécution proche de l'optimal, nous avons proposé de profiter de la phase de localisation pour obtenir toutes les informations nécessaires pour créer le plan d'exécution.

Dans l'objectif de montrer la faisabilité et l'efficacité de nos propositions, le chapitre suivant représente une évaluation des performances des méthodes et des formules expliquées au sein de ce chapitre.

Chapitre IV

Sommaire

Evaluation des performances

Ce chapitre est consacré à nos études concernant la faisabilité et la validité de nos propositions présentées dans le chapitre précédent. Les performances des formules de similarité, de la méthode de localisation de sources de données et de la méthode d'obtention de méta-données sont évaluées. Les résultats obtenus sont, également, analysés et commentés.

4.1. Introduction

Au sein de ce chapitre, nous présentons et discutons les résultats obtenus dans le cadre de la thèse. Nous avons effectué trois évaluations de performance décrites dans les sections suivantes. Dans la section 2, nous étudions l'efficacité des formules de similarité proposées dans le chapitre précédent. Quant à la section 3, nous présentons l'évaluation de performance de la méthode proposée pour la localisation de sources de données en mettant en évidence le rôle des indexes de structure dans le choix des sources de données pertinentes. Dans la section 4, nous évaluons la performance de la méthode d'obtention des méta-données nécessaires pour créer un plan d'exécution proche de l'optimal. Enfin, nous concluons ce chapitre dans la section 5 en résumant les résultats obtenus.

4.2. Evaluation des formules de similarité

Cette section est consacrée à une étude expérimentale montrant les avantages de notre méthode proposée pour effectuer la reformulation de requêtes et que nous l'appelions MSO (Méthode basée sur Synonymes et sur Ontologie). Nous comparons notre méthode avec la méthode MNE (Méthode de Noms Exacts) et la méthode MSs (Méthode basée sur Synonymes seulement). Nous avons implémenté ces méthodes au dessus du simulateur décrit dans la section suivante et utilisé pour la localisation de sources de données.

Hypothèses et jeux de données

Afin d'effectuer une évaluation de performance des formules de similarité, nous avons utilisé l'ontologie University-Benchmarks [GPH05] et nous avons utilisé notre simulateur pour créer 500 nœuds dans le système. Sur chaque nœud, nous avons créé un schéma relationnel. Pour chaque nom de relation et pour chaque nom d'attribut, nous avons

produit des synonymes. Le nombre des synonymes pour un nom donné peut avoir une valeur entre 1 et 5.

Méthodologie d'évaluation

Le principe de notre méthodologie d'évaluation est qu'un mauvais processus de *matching* entre les schémas locaux conduit à une mauvaise localisation de sources de données. Par le mot ''mauvaise'', nous désignons une localisation non capable de trouver les sources de données pertinentes. Nous étudions deux situations, dans la première (situation 1), nous considérons une demande de localisation des sources d'une relation ayant deux attributs. Dans la deuxième situation (situation 2), nous considérons une demande de localisation concernant une relation ayant quatre attributs. Pour chaque relation, nous avons injecté dans le système dix relations pertinentes. En se basant sur l'une des méthodes de localisation suivantes, chaque demande de localisation doit trouver des sources de données pertinentes.

Méthode de Noms Exacts (MNE)

Afin de créer le *matching* entre les schémas locaux, cette méthode ne prend en compte que les noms des relations et les noms des attributs. Il n'y a ni d'ontologie de domaine ni d'autre modèle représentant la sémantique commune des données partagées. De plus, Il n'y a pas de synonymes ajoutés par l'utilisateur. Cette méthode est basée sur le *matching* direct entre les schémas locaux en se basant sur les noms des relations et les noms des attributs partagés. Afin de trouver les relations correspondantes, nous utilisons les équations suivantes:

$$Sim(Q,R) = (SimQR + \sum_{i=1}^{I} Simatt_i)/(1 + \sum_{i=1}^{I} att_i) \quad (1)$$

$$Simatt_i = (\sum_{j=1}^{J} Sim_{ij}) /(\sum_{j=1}^{J} att_j) \quad (2)$$

Où Q est la relation dont on cherche les sources et $att_{1 \le i \le I}$ sont ses attributs. R est une relation candidate à être une correspondante à Q, $att_{1 \le j \le J}$ sont ses attributs. Les autres termes sont calculés comme suit :

$$\text{SimQR} = \begin{cases} 1 & \text{si Q et R ont exactement le même nom} \\ 0 & \text{sinon} \end{cases} \quad (3)$$

$$Sim_{ij} = \begin{cases} 1 \text{ si } att_i \text{ et } att_j \text{ ont exactement le même nom} \\ 0 \text{ sinon} \end{cases} \quad (4)$$

Selon cette méthode le processus de *matching* se fait à distance par les nœuds susceptibles d'être des hôtes des sources pertinentes de données.

Méthode basée sur Synonymes seulement (MSs)

Cette méthode suppose que l'utilisateur crée des synonymes pour chaque nom de relation et pour chaque nom de ses attributs. Nous distinguons deux sous-méthodes variées selon les formules de similarité utilisées pour créer le *matching* entre deux relations appartenant à deux schémas différents :

MSs1 : nous utilisons la formule de similarité proposée et évaluée dans [NOT+03].

MSs2 : nous utilisons les formules de similarité présentées dans le chapitre précédent en remplacent l'ontologie de domaine par un schéma relationnel. La motivation de cela est de comparer nos formules de similarité avec d'autres formules présentées dans un travail voisin et proposées dans le même contexte que le notre.

Méthode basée sur Synonymes et sur Ontologie (MSO)

C'est notre méthode proposée dans le chapitre précédent. Nous rappelons au lecteur que, dans cette méthode, l'utilisateur ajoute des synonymes pour chaque nom de la relation et pour chaque nom de ses attributs. En outre, une ontologie de domaine est utilisée comme un intermédiaire pour créer le *matching* entre les schémas locaux. Dans cette méthode, nos formules proposées sont utilisées et le processus de *matching* entre un schéma local et l'ontologie de domaine est effectué sur le nœud initialisant la requête.

Paramètres d'évaluation

Nous utilisons les deux paramètres Precision et Recall. Le paramètre Precision est utilisé pour mesurer le niveau de réponses correctes obtenues. Quant à Recall, ce paramètre est utilisé pour mesurer la complétude de réponses correctes obtenues. C'est-à-dire, il représente le pourcentage de réponses correctes obtenues par rapport aux réponses correctes existantes dans le système. Supposons que Relevant est l'ensemble de toutes les réponses correctes existantes dans le système et Retrieved est l'ensemble des résultats obtenus en utilisant une des méthodes présentées ci-dessus. Les deux paramètres Precision et Recall sont respectivement calculés par les équations suivantes :

$$\text{Precision} = \frac{|\text{Relevant} \cap \text{Retrieved}|}{|\text{Retreieved}|} \quad (5)$$

$$\text{Recall} = \frac{|\text{Relevant} \cap \text{Retrieved}|}{|\text{Relevant}|} \quad (6)$$

Ni le paramètre Precision tout seul, ni le paramètre Recall tout seul peut représenter la qualité du processus de *matching* entre les schémas locaux. Par conséquent, il y a une autre mesure, appelée F-mesure (β), incluant les deux paramètres et capable de refléter la qualité du processus de *matching* en utilisant une seule formule. Le F-mesure est traditionnellement calculé comme suit:

$$F - \text{Measure}(0.5) = 2 * \text{Precision} * \text{Recall}/(\text{Precision} + \text{Recall}) \quad (7) \text{ , ici } \beta = 0.5$$

Analyse de résultats et discussion

Nous calculons les valeurs des deux paramètres Precision et Recall pour chaque méthode et pour chaque demande de localisation. Pour les méthodes MNE et MSs, nous considérons qu'une relation est correspondante à une autre relation si et seulement si la similarité entre les deux relations est plus grande qu'un seuil donné. Dans le cas de la méthode MSO, nous considérons qu'une classe est correspondante à une relation si et seulement si la similarité entre cette classe et la relation est plus grande que le même seuil. Le seuil est varié entre 0.1 et 0.9 d'un pas de 0.2.

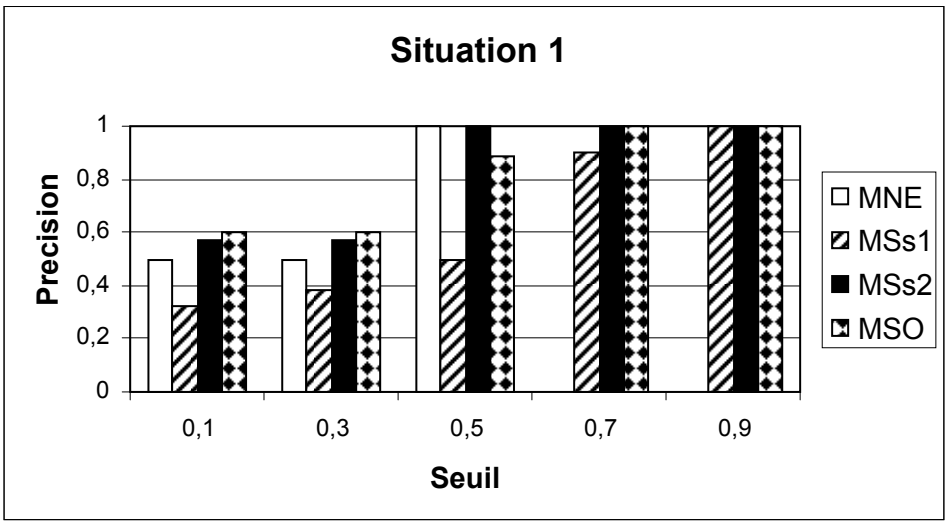


Fig. 20. Le paramètre Precision concernant la situation 1

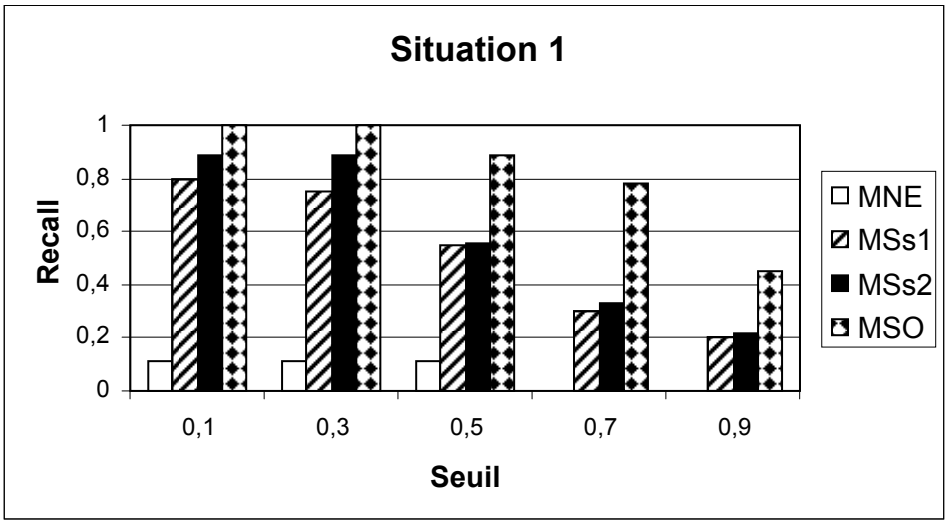


Fig. 21. Le paramètre Recall concernant la situation1

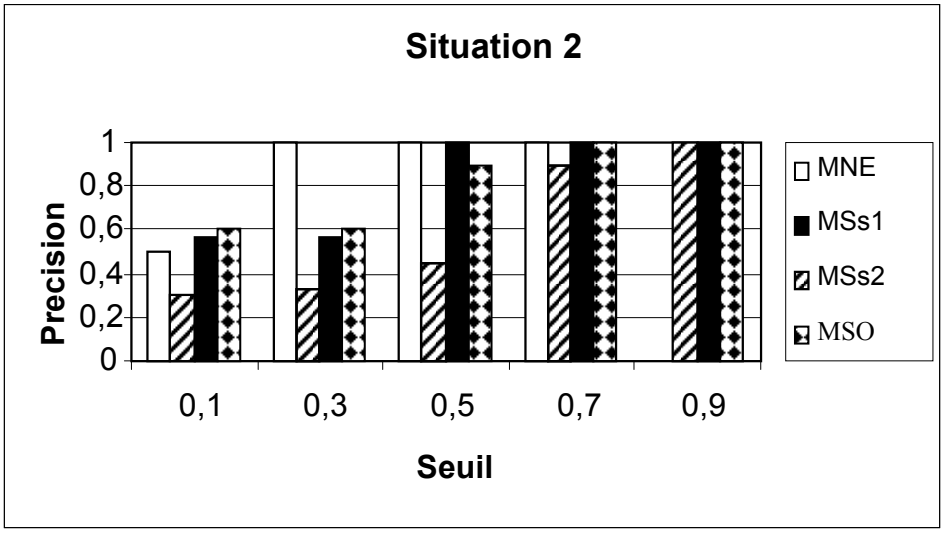


Fig. 22. Le paramètre Precision concernant la situation2

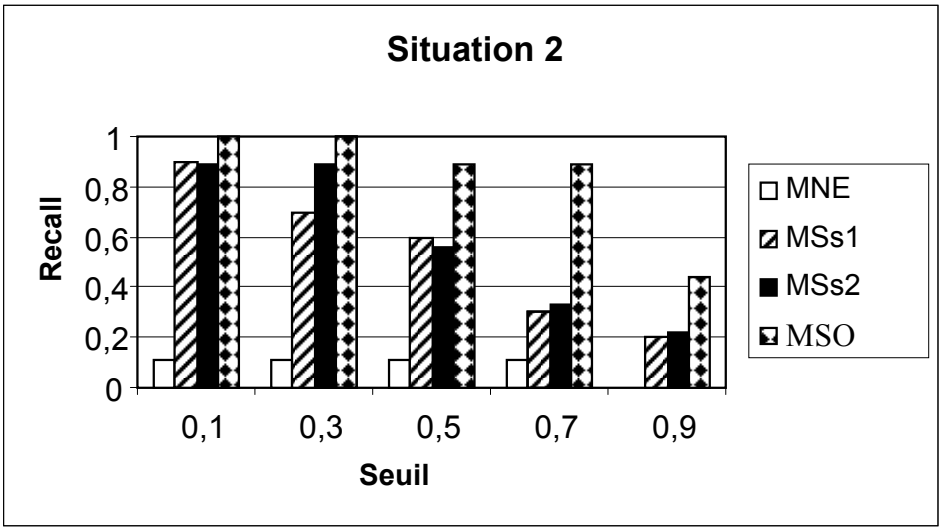


Fig. 23. Le paramètre Recall concernant la situation 2

En regardant les figures 20, 21, 22 et 23 on peut constater que la méthode MNE n'est pas capable de donner des résultats corrects ni pour la situation 1 (cf. figures 20 et 21) lorsque le seuil est plus grand que 0.5, ni pour la situation 2 (cf. figures 22 et 23) lorsque le seuil est plus grand que 0,7. Même si la méthode MNE retourne 100% des résultats corrects quand le seuil est égal à 0,5 dans la situation 1 et à 0,5 et 0,7 dans la situation 2, les résultats retournés ne forment plus de 10% des résultats corrects disponibles dans le système. En ce qui concerne la méthode MSO, toutes ses réponses sont correctes lorsque le seuil est supérieur à 0,7 pour les deux situations étudiées. Cependant, il est capable de rentrer seulement (78% quand le seuil est de 0,7 et 45% lorsque le seuil est de 0,9) des résultats corrects dans la situation 1 et (86% quand le seuil est de 0,7 et 44% lorsque le seuil est de 0,9) dans la situation 2. D'autre part, la méthode MSs1 retourne 100% des résultats corrects quand le seuil est de (0,9 dans la situation 1 et 0,9 pour la situation 2). Mais, elle ne retourne pas plus de 20% des résultats corrects disponibles dans le système dans les deux situations 1 et 2. Quand on regarde les résultats réalisés par la Méthode MSs2, nous pouvons noter que tous ses résultats sont corrects quand le seuil est plus grand de 0,5 dans les deux situations étudiées. Cependant, la méthode MSs2 n'est pas capable de retourner plus de 35% des résultats pertinents lorsque le seuil est plus grand de 0,7 dans les deux situations. Evidement, pour toutes les méthodes envisagées, lorsque Précision atteint une valeur élevée, Recall a une valeur relativement petite.

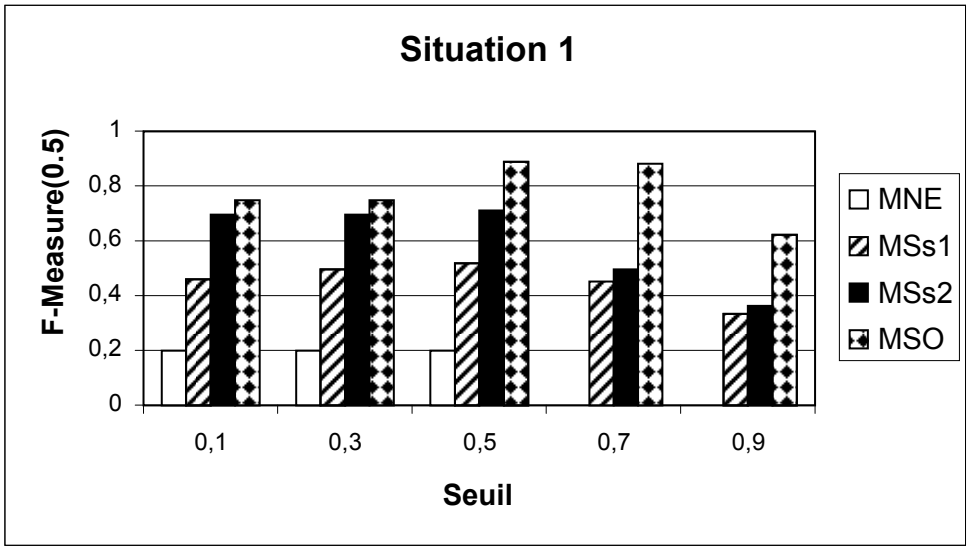


Fig. 24. F-Mesure (0.5) de la situation 1

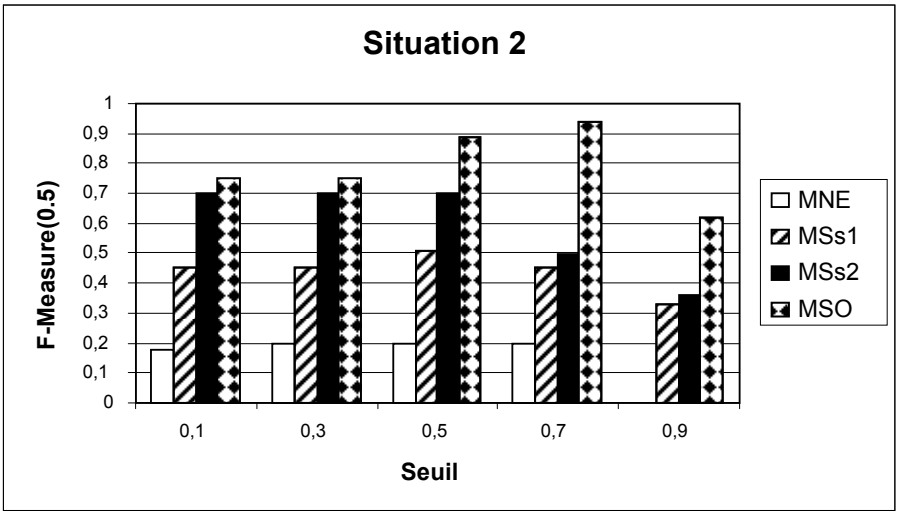


Fig. 25. F-Mesure (0.5) de la situation 2

En regardant les figures 24 et 25, on peut évaluer la qualité de *matching* entre les schémas locaux effectué par les méthodes étudiées. Comme résultat final, la meilleure méthode est la méthode MSO et la pire est la méthode MNE pour les deux situations 1 et 2. La méthode MSs2 est meilleure que la méthode MSs1 en termes de la qualité de *matching*.

4.3. Evaluation de performance de la méthode de localisation de sources de données

L'objectif de cette section est d'évaluer la méthode de la localisation des sources de données pertinentes en prenant en compte l'hétérogénéité structurelle des schémas locaux. Nous supposons que le problème de l'hétérogénéité sémantique est déjà résolu. Nous rappelons que notre méthode est basée sur une version étendue du protocole Chord [SMK+01]. Notre première intention était de travailler sur une implémentation existante du protocole Chord que nous aurions étendue afin d'y intégrer les solutions que nous proposons. Pour cela nous avons choisi OpenChord[28], une implémentation open-source du protocole Chord (désormais Chord tout simplement) écrite en Java, réalisée et testée à l'université de Bamberg en Allemagne. Mais ce projet est encore en développement, et nous avons été confrontés à des dysfonctionnements. De plus, l'implémentation comporte des différences par rapport à l'algorithme original du protocole Chord. Nous avons donc décidé d'écrire une implémentation complète de Chord, aussi proche que possible de l'algorithme original. Nous avons choisi d'utiliser le langage Java, afin d'obtenir un programme portable et exécutable sur n'importe quelle machine disposant d'une implémentation de la machine virtuelle Java, par exemple des machines sous Microsoft Windows, Linux, UNIX, ou Mac OS.

[28] Open Chord homepage, http://www.uni-bamberg.de/en/pi/bereich/research/software_projects/openchord/

4.3.1. Environnement de tests

Nous avons tout d'abord exécuté avec succès des tests réels sur quatre machines interconnectées grâce à cette implémentation. Mais nous n'avions pas la possibilité de disposer de plusieurs milliers de machines pour effectuer des tests à grande échelle. Cela nous a contraints à modifier notre programme afin de pouvoir simuler l'exécution d'un nombre variable (de quelques dizaines à plusieurs milliers) de nœuds. Le seul élément du programme qui a subi des modifications est celui qui concerne les communications réseau. Le reste du programme (i.e. la partie concernant la DHT), par contre, reste non modifié.

Le simulateur permet de créer un réseau P2P comportant un nombre quelconque de nœuds et de leur faire manipuler les informations stockées dans la DHT comme le feraient des nœuds en situation réelle. Ce simulateur est paramétré grâce à des fichiers contenant les informations sur les nœuds. De plus, il prend comme paramètre une description formelle des demandes de localisation à effectuer pour les tests. Les résultats des différents tests sont écrits dans un fichier à partir duquel nous avons extrait des données afin de tracer plusieurs courbes qui nous permettront de comparer l'efficacité de nos propositions.

Les jeux de données de tests comprennent des nœuds et des relations dont le nombre varie, de 50 à 2000 pour les nœuds, et de 100 à 3000 pour les relations. Afin de simuler le problème de l'hétérogénéité structurelle dans nos tests, chaque relation est représentée dans le système par plusieurs structures différentes, avec un nombre d'attributs variant de 1 à 8. Nous considérerons trois types de demandes de localisation (appelées DL1, DL2, et DL3). Le type DL1 porte sur la localisation des sources d'une relation, pour une requête SQL nécessitant un seul attribut (dans la clause de projection ou dans la clause de condition). Le type DL2 requiert trois attributs, et le type DL3 requiert sept attributs. Une réponse valide à une demande de localisation ne contient que des sources stockant les relations ayant les attributs requis par la requête SQL correspondante. La figure 26 montre des exemples de requêtes SQL et les demandes de localisation correspondantes.

SELECT P.Nom, P.Adresse
FROM Docteur D, Patient P
WHERE D.Nom = P. Docteur ;

La requête SQL ci-dessus nécessite les demandes de localisation suivantes :

- Relation : Docteur, Attribut : Nom (type DL1)
- Relation : Patient, Attributs : Nom, Adresse, Docteur (type DL2)

SELECT Nom, Prénom, Adresse, Téléphone, NumSS
FROM Patient
WHERE Age > 20 AND Docteur = ‘‘Durand’’ ;

La requête SQL ci-dessus nécessite la demande de localisation suivante :

- Relation : Patient, Attributs : Nom, prénom, Adresse, Téléphone, NumSS, Age, Docteur (type DL3)

Fig. 26. Requêtes SQL utilisées et demandes de localisation correspondantes

Avant d’étudier l’impact de l’extension de Chord, nous allons montrer l’importance de cette extension en illustrant comment Chord répond aux demandes de localisation ci-dessus sans subir aucune extension.

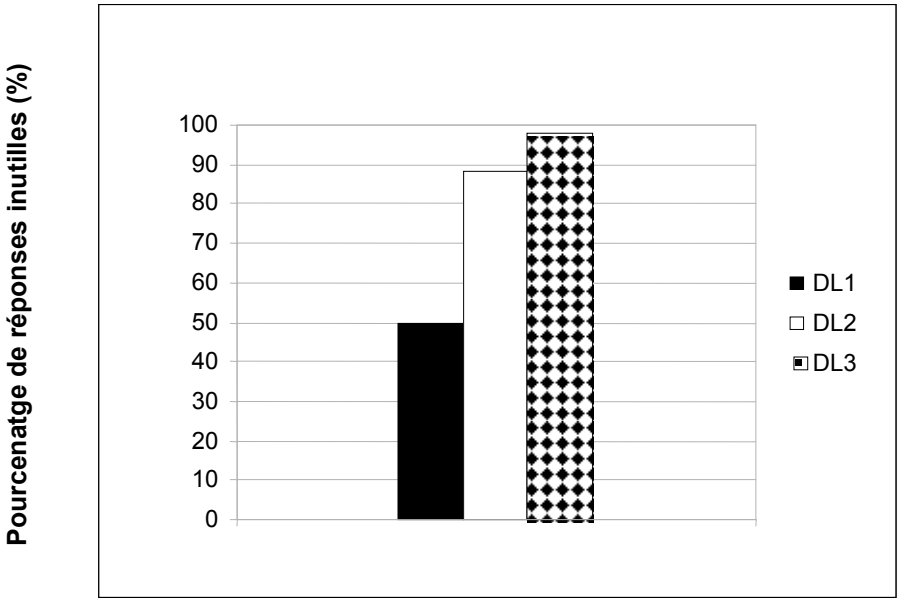


Fig. 27. Pourcentage des réponses inutiles retenues par le protocole Chord

4.3.2. Intérêt de l'extension de Chord

Nous avons calculé le taux de réponses inutiles retenues par Chord lors d'une localisation des sources de données. La figure 27 montre les résultats de ce test sur les trois types de demandes de localisation définis dans la figure 26. Nous remarquons que le nombre d'attributs a un impact important sur la qualité des réponses retenues par Chord. On constate que plus le nombre d'attributs nécessaires est élevé, plus le taux de réponses inutiles est grand (Chord renvoie 97% de réponses inutiles dans le cas de DL3, 88% dans le cas de DL2 et 50% dans le cas de DL1). En effet, pour une clé donnée, Chord renvoie tous les identifiants de toutes les sources stockant les relations représentées par cette clé sans prendre en compte la structure de ces relations (en termes d'attributs) sur leurs sources. Le résultat obtenu illustre bien l'intérêt de notre proposition concernant l'enrichissement de la DHT par des index de structure afin de permettre de sélectionner les sources de données pertinentes et par conséquent d'éviter de renvoyer des réponses inutiles. Dans tous les tests effectués dans cette section nous comparons les deux situations suivantes :

- **Situation A :** cette situation représente Chord original tel que celui qui est décrit dans [SMK+01]. Dans cette situation, chaque nœud stocke une partie de la DHT contenant les clés dont il est responsable et les identifiants des sources de données représentées par ces clés. En se basant sur ces informations, chaque nœud peut localiser les sources de la relation représentée par une clé donnée sans distinguer les relations ayant la structure requise pour la requête SQL correspondante. Cela est dû à l'absence d'informations concernant la structure des relations sur leurs sources.

- **Situation B :** cette situation représente Chord étendu tel que celui qui est décrit dans le chapitre précédent. La DHT est enrichie par des index de structure afin de sélectionner les sources stockant des relations possédant les attributs décrits dans la requête SQL correspondante. La sélection se fait par le nœud responsable de la clé représentant la relation dont on cherche les sources.

Nous verrons par la suite l'impact de l'enrichissement de la DHT par des index de structure sur la réduction du volume d'informations transmises sur le réseau en évitant des réponses inutiles aux demandes de localisation. Cette réduction peut être expliquée en mesurant le temps de localisation. Evidement, lorsque la taille des informations transmises sur le réseau est grande, le temps nécessaire pour transférer ces informations est grand aussi. Nous avons mesuré, également, le prix à payer pour éviter des réponses inutiles en calculant la taille moyenne de DHT sur chaque nœud avant et après cet enrichissement.

4.3.3. Temps de localisation

Comme les communications réseaux étaient simulées lors des tests, nous devions faire des hypothèses concernant la bande passante. Nous considérons qu'un tel environnement de tests a pour objectif d'être utilisé par n'importe quel utilisateur en permettant de connecter son ordinateur personnel au système pour partager ses données. Donc, nous partons du principe que les connexions réseau utilisées sont celles accessibles aux particuliers de nos jours pour se connecter à l'Internet. Bien qu'il existe des offres très diverses, et que les débits montants et descendants ne soient pas symétriques dans le cas de connexions ADSL, nous avons choisi une valeur moyenne pour les tests, fixée à 160 kb/s. Le fait d'utiliser une valeur fixe peut amener à des résultats erronés dans l'absolu, mais nous cherchons surtout à comparer les temps de localisation de différentes méthodes dans des conditions similaires, et non pas à obtenir des valeurs exactes du temps de localisation.

Puisque les communications réseau ont été simulées lors de nos tests, nous avons effectué des tests de temps de localisation en utilisant un modèle analytique (formule 1). Dans cette formule, seule la bande passante a été estimée. Les autres paramètres, tels que la taille d'une demande de localisation et le nombre de nœuds par lesquels passe une demande de localisation, ont été mesurés. Grâce à ce modèle analytique, nous avons pu calculer les temps de localisation des trois types de demandes de localisation DL1, DL2 et DL3 (présentés sur les figures 28, 29 et 30 respectivement) pour les deux situations A et B.

Formule 1 : Modèle analytique

Le temps de localisation est égal à la somme du temps de trouver le responsable de la clé recherchée et du temps d'envoi de la réponse :

$$TL = S * \frac{Taille(DL)}{BP} + \frac{Taille(Rep)}{BP}$$

Avec :

S : le nombre de nœuds par lesquels passe la demande de localisation

Taille (DL) : la taille de la demande de localisation

Taille (Rep) : La taille des réponses envoyées au nœud initialisant la requête

BP : la bande passante estimée.

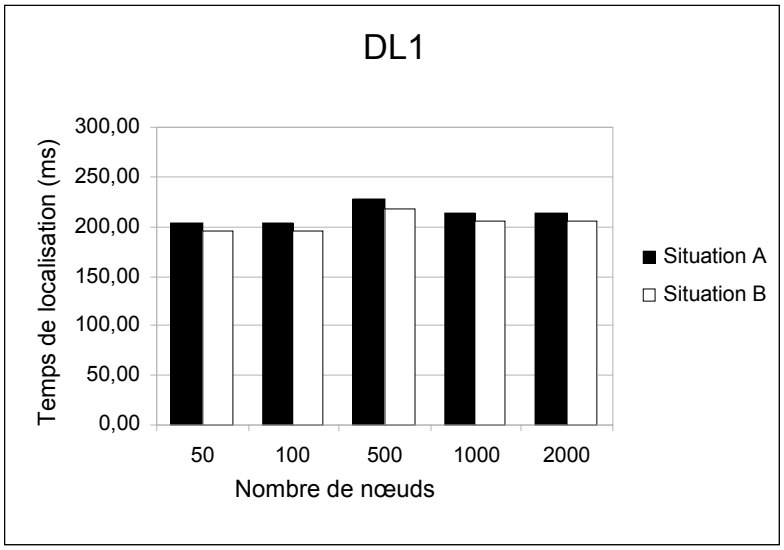


Fig. 28. Le temps de localisation concernant le type DL1

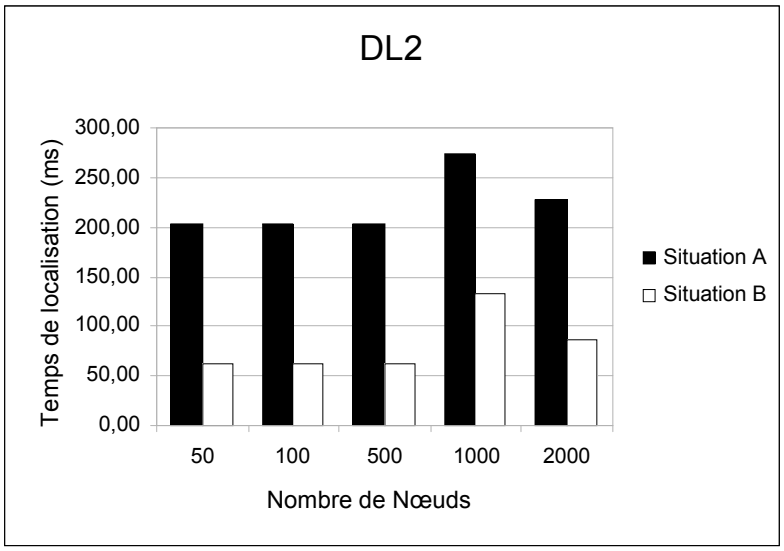


Fig. 29. Le temps de réponse concernant le type DL2

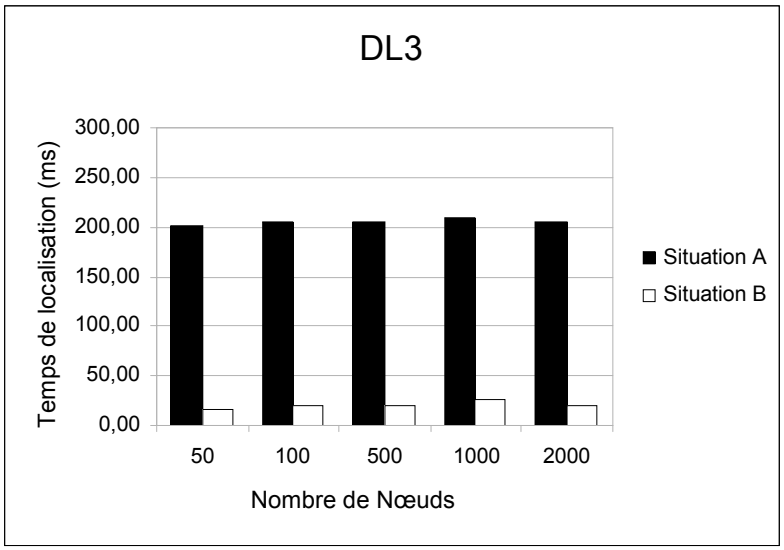


Fig. 30. Le temps de localisation concernant le type DL3

Le fait de sélectionner les sources pertinentes réduit le volume d'informations transférées sur le réseau et, ainsi, diminue le temps de localisation. Comme nous l'avons vue, cette sélection a un impact particulièrement important sur les demandes de localisation qui nécessitent beaucoup d'attributs car ce sont celles qui génèrent le plus de réponses inutiles avec Chord (jusqu'à 97% de réponses inutiles sur notre jeu de données). La conséquence de cette sélection est un gain sur les temps de localisation qui est faible sur le type DL1 (quelques millisecondes). Mais pour DL2, la réduction du temps de localisation varie entre 50% et 70%, et dans le cas de DL3 on atteint 90% de réduction du temps de localisation.

Afin d'affiner notre comparaison entre les deux situations A et B, nous avons introduit le paramètre de facteur d'accélération (noté FA) comme $FA = DL(A) / DL(B)$ suit :

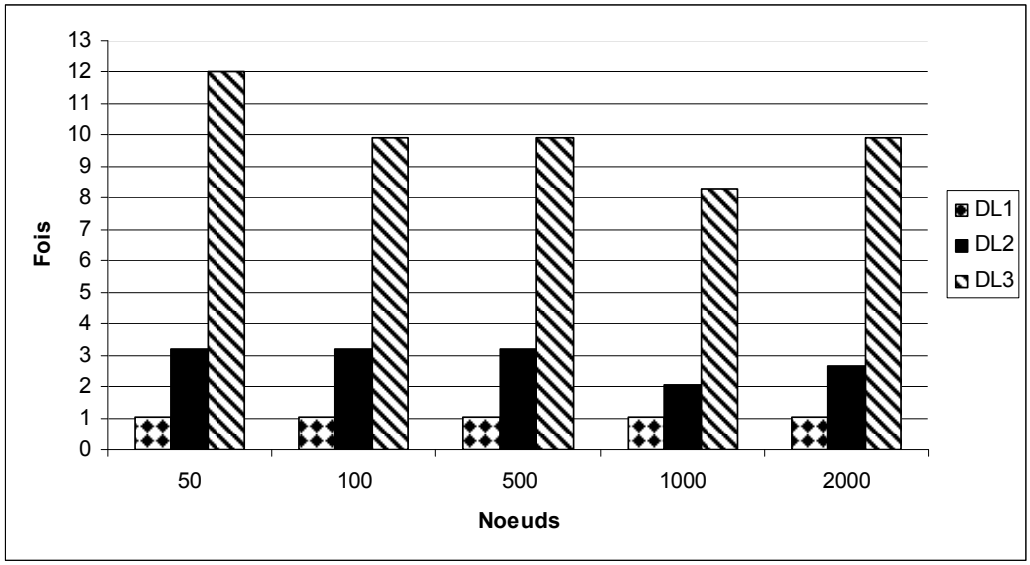


Fig. 31. Facteur d'accélération du temps de localisation

L'étude de la figure 31 nous permet de constater que, en termes de temps de localisation, la situation B est meilleure que la situation A. En effet, FA ayant une valeur (entre 1,04% et 1,05 %), (entre 2,07% et 3,22%) et (entre 8,29% et 12,02%) pour DL1, DL2 et DL3 respectivement. Par conséquent, plus le nombre d'attributs est élevé, plus le facteur d'accélération est grand.

Sur les figures 28, 29, 30 et 31, on peut voir que nous avons fait varier le nombre de nœuds dans le système pour chaque test, afin de vérifier que la caractéristique du passage à grande-échelle de Chord est bien conservée lors de l'enrichissement de la DHT par des index de structure. On constate que les performances restent stables quand on varie le nombre de nœuds.

Dans la sous-section suivante, nous allons étudier le prix à payer pour obtenir le résultat présenté ci-dessus.

4.3.4. Taille moyenne de la DHT

L'extension apportée à Chord nous a amenés à enrichir la DHT par plus d'informations. La taille d'informations stockées sur les nœuds est, donc, plus importante dans la situation B que celle dans la situation A. Les mesures présentées ici ont pour objectif de quantifier cette augmentation de taille. Nous avons comparé la taille moyenne des

informations qu'un nœud stocke dans sa partie de la DHT, en faisant varier le nombre de nœuds et en fixant le nombre de relations dans un premier temps. Puis en faisant varier le nombre de relations et en fixant le nombre de nœuds, dans un second temps. Les mesures ont été effectuées en calculant les tailles d'informations stockées sur chaque nœud, puis en calculant la moyenne de ces tailles.

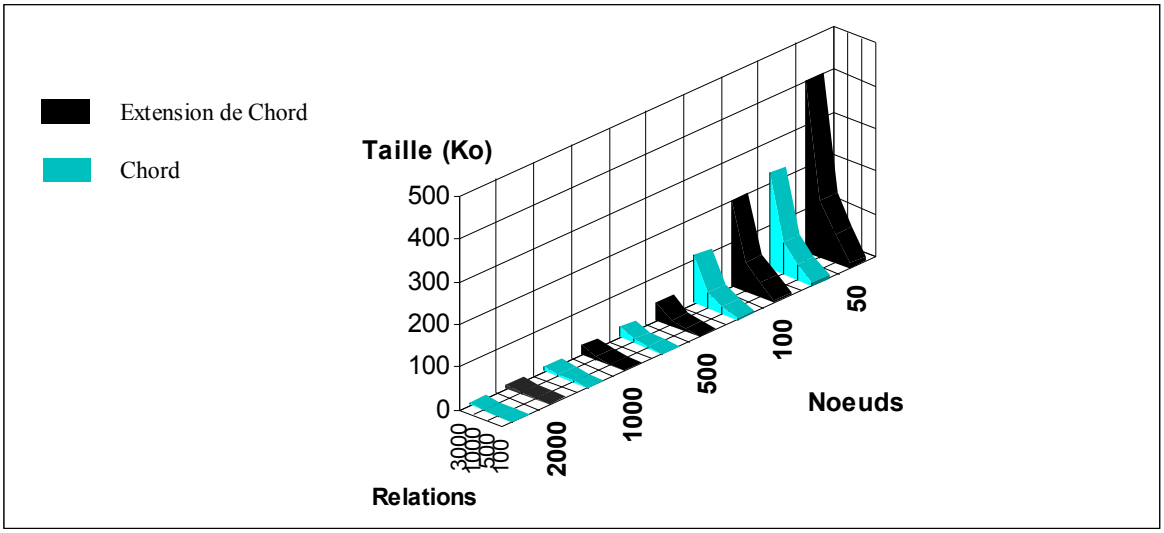


Fig. 32. Taille moyenne de la DHT avec et sans extension de Chord

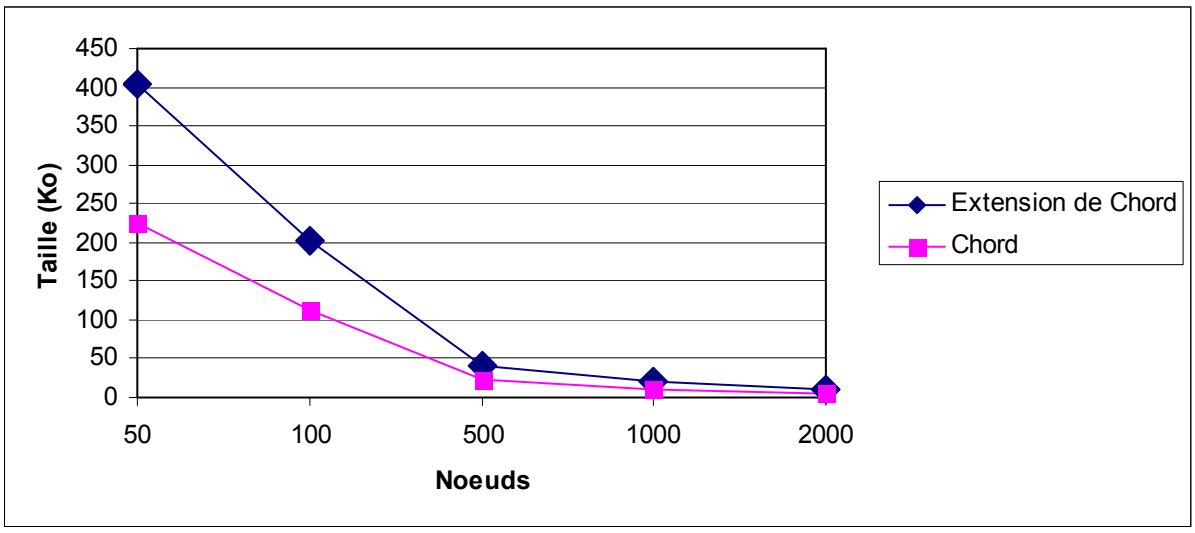


Fig.33. Taille moyenne de la DHT avec et sans extension de Chord lorsque le nombre de relations est 3000

Les figures 32, 33 et 34 montrent une augmentation notable de la taille des informations stockées sur chaque nœud. Quelle que soient le nombre de nœuds et le nombre de relations présents dans le système, on observe sur notre jeu de données de test une augmentation, environ, de 100% de la taille moyenne de la DHT. D'après les tests du temps de localisation exposés dans la sous-section précédente, cela ne semble pas affecter notablement les

performances des demandes de localisation. Ceci s'explique par la faible quantité d'informations (quelques centaines de kilo-octets seulement).

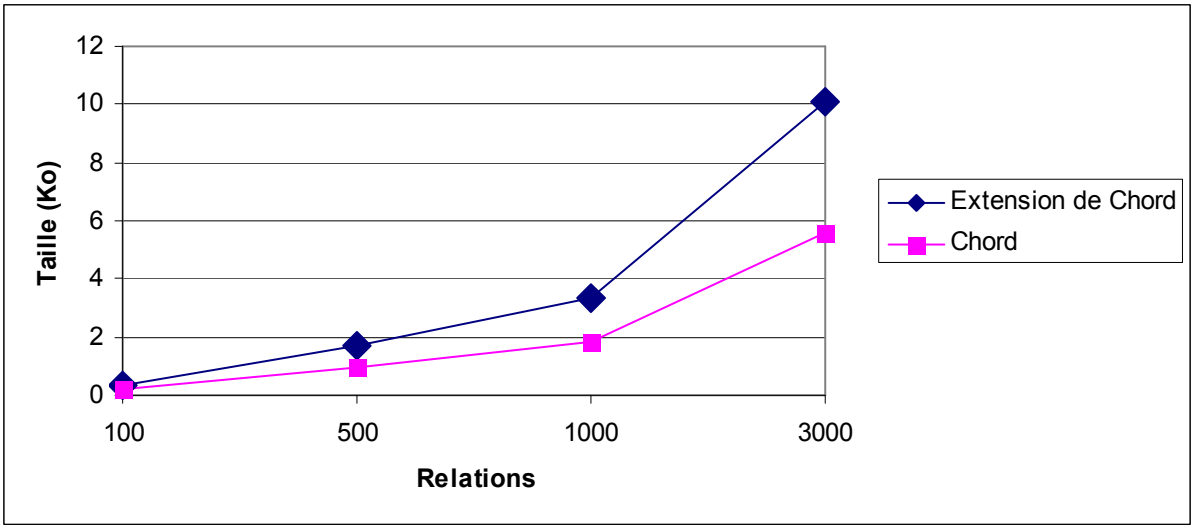


Fig. 34. Taille moyenne de la DHT avant et après l'extension de Chord lorsque le nombre de nœuds est 2000

4.4. Evaluation de la méthode d'obtention de méta-données

Dans cette section, nous évaluions la performance de la méthode d'obtention de méta-données en termes du temps de réponse et du volume de données transférées inter-site. Nous rappelons que cette méthode est une extension de la méthode de localisation de sources de données. L'objectif est de mettre en évidence l'importance de profiter de la phase de localisation pour obtenir les méta-données nécessaires pour effectuer l'optimisation globale. L'évaluation est effectuée par un modèle de simulation qui nous permet de tester des configurations matérielles plus importantes que celles que nous possédons. Dans la suite, nous décrivons le modèle de simulation utilisé, les requêtes étudiées, les méthodes simulées et les résultats obtenus.

4.4.1. Modèle de simulation

Le simulateur est consisté de deux parties essentielles. La première sert à estimer le temps de localisation de sources de données. Cette partie s'appuie sur le simulateur décrit dans la section précédente. Nous considérons le cas d'un système P2P composé de N= 2000 nœuds et stockant 3000 relations. Une autre extension a été imposée sur la localisation de sources de données. Dans cette extension, lorsque le nœud responsable de la clé recherchée reçoit la demande de localisation, il sélectionne les sources pertinentes et ensuite au lieu de répondre lui-même à cette demande, il l'envoie aux sources sélectionnées. Ces sources vont

répondre à la demande de localisation en envoyant au nœud initialisant la requête les méta-données nécessaires pour créer un plan d'exécution proche de l'optimal. Le temps de localisation est calculé en utilisant la formule 2 :

Formule 2 : Modèle analytique du temps de localisation

$$TL = (S+1) * \frac{Taille(DL)}{BP} + Max(\frac{Taille(Rep_i)}{BP})$$

Avec :

S : le nombre de nœuds par lesquels passe la demande de localisation pour arriver au nœud responsable de la clé recherchée.

Taille (DL) : la taille de la demande de localisation

Taille (Rep_i) : La taille de la réponse envoyée au nœud initialisant la requête par la source i.

BP : la bande passante estimée.

La demande de localisation est transmise S fois, comme dans la formule 1, mais elle est aussi envoyée aux sources de données. On ne compte ces transmissions qu'une seule fois car la demande de localisation est transmise à toutes les sources en parallèle.

Quant à la deuxième partie du simulateur, cette partie sert à estimer : le temps de réponse d'un plan d'exécution donné et le volume de données transférées inter-site lors de l'exécution. Cette partie s'appuie sur un simulateur développé et validé au sein de l'équipe PYRAMIDE à l'IRIT [Dup01]. Les valeurs CPU des nœuds participants au système sont : 550, 1100 et 1500 MHz. Les opérations de jointure s'effectuent par hachage simple. Le facteur de sélectivité d'une jointure entre deux relations Ri et Rj est choisi dans l'intervalle [0.5/max(||Ri||,||Rj||), 1.5/max(||R||,||S||)], où ||Ri|| et ||Rj|| sont respectivement les nombres de tuples de Ri et Rj.

Requêtes étudiées

Etant donné que l'opération de jointure est l'opération SQL la plus fréquente et la plus coûteuse du point de vue de consommation de ressources (e.g. bande passante réseau, CPU), nous allons traiter cinq requêtes différentes du point de vue du nombre de jointures incluant dans chacune. Ces requêtes sont présentées dans la figure 35.

```
Q1:  Select   R3.b, R4.d
     From     R3, R4
     Where    R3.c = R4.c

Q2: Select    R3.c, R2.l
    From      R1, R2, R3
    Where     R1.a = R2.a
    And       R2.b = R3.b

Q3: Select    R3.j, R2.b
    From      R1, R2, R3, R4
    Where     R1.a = R2.a
    And       R2.b = R3.b
    And       R3.d = R4.d

Q4: Select    R1.a,  R5.c, R8. l
    From      R1, R2,  R5, R6, R7, R8, R9
    Where     R1.a = R9.a
    And       R2.b = R6.b
    And       R5.d = R7.d
    And       R6.e = R7.e
    And       R8.f = R7.f
    And       R8.j  = R9.j

Q5: Select    R7.h,  R5.c, R8.d
    From      R1, R2, R3, R4, R5, R6, R7, R8, R9
    Where     R2.b = R5.b
    And       R5.c = R9.c
    And       R5.d = R7.d
    And       R6.e = R7e
    And       R8.f = R1.f
    And       R4.i  = R3.i
    And       R3.j  = R1.j
    And       R8.j  = R7.j
```

Fig. 35. Les requêtes étudiées

Les requêtes utilisées dans l'évaluation de performance peuvent être classifiées en trois classes:

C1 : requête simple incluant au plus deux jointures (i.e. Q1 et Q2).

C2 : requêtes moyennes, cette classe peut contenir des requêtes incluant entre trois et cinq jointures (i.e. Q3).

C3 : requêtes complexes, ces requêtes peuvent inclure au minimum six jointures (i.e. Q4 et Q5).

La figure 35 illustre les relations de base des requêtes évaluées, leurs tailles, la façon dont ces relations sont distribuées sur les nœuds, les valeurs CPU de chaque nœud et le pourcentage de charge de chaque CPU. Ces valeurs sont considérées au moment du traitement des requêtes. Elles se trouvent dans les catalogues locaux des sources de données.

Chaque relation est constituée de 16 attributs. La taille d'un tuple d'une relation est 128 octets.

Relations	Nœuds	Taille (tuples)	CPU (MHz)	Charge
R1	**N1**	**40 000**	**1100**	**70%**
R2	**N2**	**28 000**	**1100**	**20%**
R3	**N3**	**380 000**	**550**	**80%**
R4	**N4**	**26 000**	**1100**	**20%**
R5	**N5**	**30 000**	**1100**	**20%**
R6	**N6**	**340 000**	**1100**	**25%**
R7	**N7**	**4 000**	**1100**	**25%**
R8	**N8**	**36 000**	**1100**	**20%**
R9	**N9**	**360 000**	**550**	**40%**

Fig. 36. Méta-données trouvées dans les catalogues locaux des hôtes des sources de données

Méthodes simulées

Afin d'évaluer la performance de la méthode d'obtention de méta-données (désormais MOM), nous l'avons comparée quantitativement avec les deux méthodes suivantes : Méthode Classique (MC) et Méthode Gourmande (MG). Dans la méthode MC, l'étape de localisation sert, uniquement, à localiser la source d'une clé représentant une relation. L'optimisation s'appuie sur des MD trouvées dans le catalogue local du nœud initialisant la requête (désormais NIR). Dans ce catalogue, il peut manquer certaines informations concernant les caractéristiques physiques des sources de données, du réseau ou des données elles-mêmes. De plus, les MD trouvées dans ce catalogue peuvent être obsolètes. Alors, la méthode CM consiste à exécuter les opérations unaires (e.g. sélection, projection) sur les sources de données. Par contre, les exécutions des opérations binaires (e.g. jointure) s'effectuent d'une façon aléatoire sur l'une des deux sources des opérandes de relation. La figure 36 illustre les MD, concernant les requêtes évaluées, trouvées dans le catalogue local du NIR au moment de soumission de requête. Le point d'interrogation dans la figure 37 signifie qu'aucune information n'est disponible. Les valeurs mises en italique signifient que ces valeurs sont obsolètes. Dans la figure 36, les MD trouvées dans les hôtes des sources de données sont illustrées. La deuxième méthode MG consiste à envoyer toutes les relations de

base au NIR, toutes les exécutions des opérations binaires du plan d'exécution sont exécutées sur le NIR.

Relations	Nœuds	Taille (tuples)	CPU (MHz)	Charge
R1	**N1**	**40 000**	**1100**	**70%**
R2	**N2**	**28 000**	**1100**	**20%**
R3	**N3**	***20 000***	**550**	**80%**
R4	**N4**	**26 000**	**1100**	**20%**
R5	**N5**	**30 000**	**1100**	**20%**
R6	**N6**	***300 000***	**1100**	**25%**
R7	**N7**	**?**	**?**	**?**
R8	**N8**	**36 000**	**1100**	**20%**
R9	**N9**	**?**	**?**	**?**

Fig. 37. MD trouvées dans le catalogue local du NIR

4.4.2. Analyse de performance

Les figures 38 et 39 montrent les résultats obtenus lors de l'évaluation de 5 requêtes (cf. figure 35) avec respectivement les méthodes : MG, MC et MOM. Nous avons calculé le temps de réponse (cf. figure 38) et le volume de données transférées inter-site (cf. figure 39) pour chaque requête selon les trois méthodes simulées. Le temps de réponse considéré est le temps nécessaire pour retourner la totalité des résultats finaux au NIR. En ce qui concerne le volume de données transférées, nous calculons la totalité de données transférées inter-site pendant l'exécution.

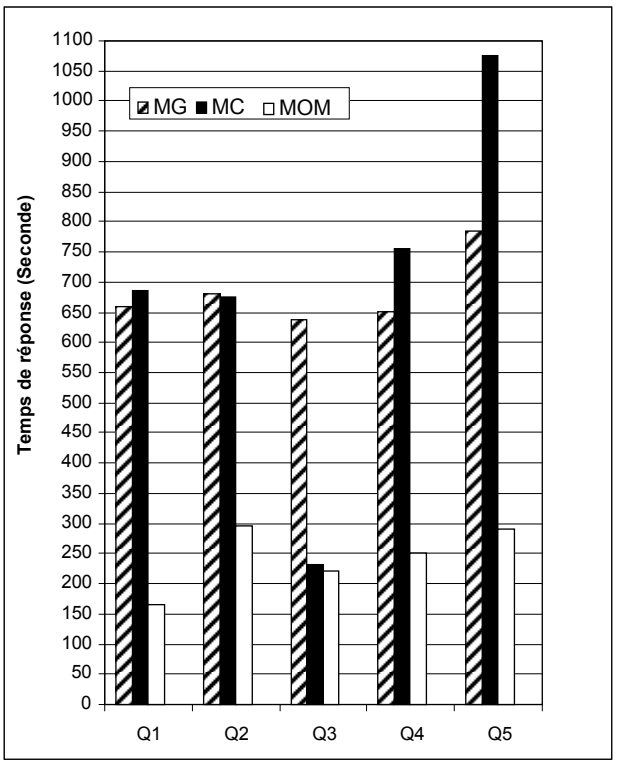


Fig. 38. Temps de réponse

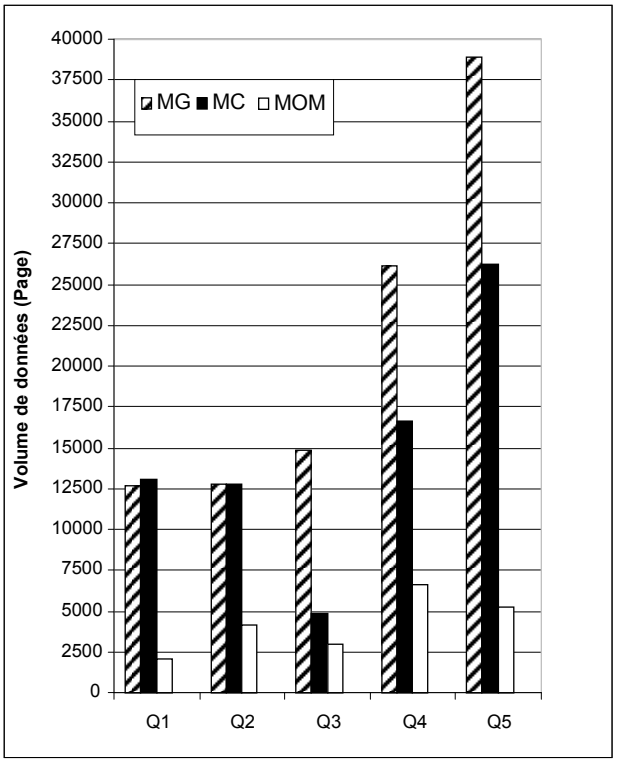


Fig. 39. Volume de données transférées

De manière générale, on peut constater sur les figures 38 et 39 que l'écart des différentes méthodes dépend de la classe de la requête. Dans la classe C1 (i.e. Q1 et Q2), il y a une grande différence entre la méthode proposée (MOM) et les méthodes (MG et MC). Le

temps de réponse dans le cas de la méthode MG est plus grand que celui calculé selon la méthode MOM parce que la méthode MG nécessite de transférer toutes les relations de base au NIR. De plus, les exécutions des opérations sont effectuées d'une façon séquentielle. Les résultats obtenus montrent que la MC peut être plus coûteuse que la méthode MG en termes du temps de réponse et du volume de données transférées. Des erreurs des estimations des tailles des relations temporaires peuvent être engendrées à cause du manque d'informations ou à cause de statistiques obsolètes. Ces erreurs imposent de transférer un volume de données inter-site plus grand que celui dans la méthode MG (cas de Q1). Malgré le parallélisme utilisé par la méthode MC, le temps nécessaire pour transférer la différence des volumes de données transférées inter-site dans les deux cas (MG et MC) reste supérieur au temps gagné par le parallélisme. La méthode proposée (MOM) est plus efficace par rapport aux méthodes (MG et MC) car elle utilise de méta-données à jour. Cela permet d'éviter les erreurs des estimations engendrées à cause du manque d'informations ou de statistiques obsolètes.

Quant à la classe C2 (i.e. Q3), nous remarquons que la méthode MG est la plus coûteuse en terme du temps de réponse et du volume de données transférées inter-sites. Malgré les statistiques obsolètes, la méthode MC a réussi d'exécuter la requête Q3 d'une façon relativement efficace par rapport à la méthode MG. Cependant, la méthode proposée MOM est la plus efficace.

Enfin, pour la classe C3 (i.e. Q4 et Q5), les écarts entre la méthode proposée (MOM) et les méthodes (MG et MC) sont plus importants. La méthode MOM offre toujours une amélioration en termes du temps de réponse et du volume de données transférées inter-sites. Nous pouvons remarquer que même si la méthode MC nécessite de transférer un volume de données inférieur à celui de la méthode MG, le temps de réponse de la méthode MC est supérieur à celui de la méthode MG. En effet, La méthode MC peut engendrer des certaines décisions qui peuvent retarder l'exécution du plan d'exécution.

Pour affiner notre comparaison des différentes méthodes en termes du temps de réponse, nous introduisons deux facteurs d'accélération :

F1 : Temps de réponse de MC / Temps de réponse de MOM

F2 : Temps de réponse de MG / Temps de réponse de MOM

L'étude des diagrammes de la figure 40 représentant les deux facteurs d'accélération F1, F2 pour les cinq requêtes étudiées, permet d'affiner les résultats précédents.

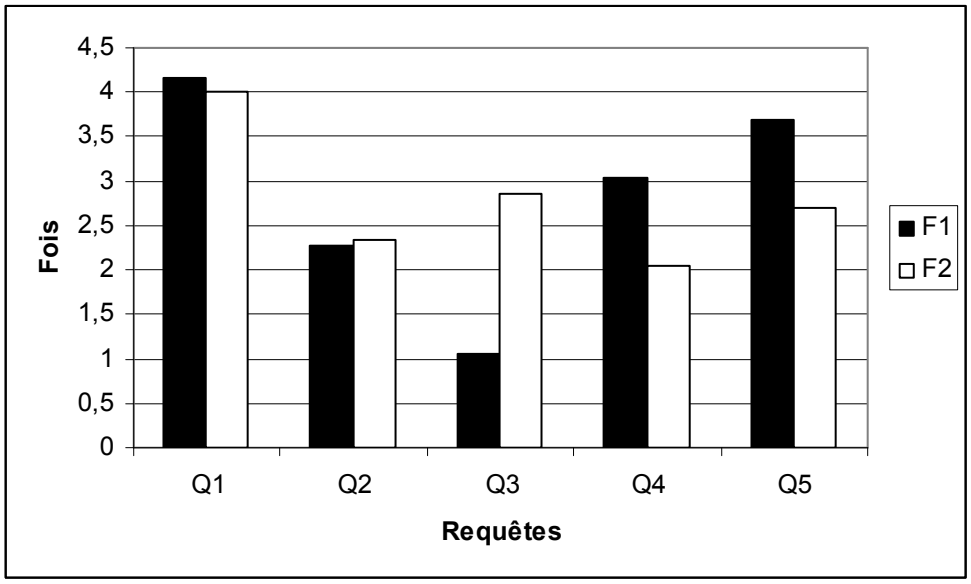


Fig. 40. Facteurs d'accélération du temps de réponse

Dans l'état de la classe C1, la méthode MOM améliore le temps de réponse d'un pourcentage de (4,17 fois pour Q1 et 2,27 fois pour Q2) par rapport à la méthode MC et de (4,00 fois pour Q1 et de 2,33 fois pour Q2) par rapport à la méthode MG. Pour la classe C2 (cas de Q3), la méthode MOM n'améliore que de 1,05 fois le temps de réponse de la méthode MC. Cependant, elles améliorent le temps de réponse de la méthode MG d'un pourcentage de 2,86 fois. En ce qui concerne la classe C3, la méthode MOM améliore d'une façon remarquable le temps de réponse. En effet, la méthode MOM réalise une amélioration de (3,03 fois pour Q4 et 3,70 fois pour Q5) par rapport à la méthode MC et de (2,05 fois pour Q4 et 2,70 fois pour Q5) par rapport à la méthode MG.

En résumé, à partir des figures 38 et 39, à cause de la façon aléatoire d'exécution des opérations binaires, la méthode MC peut être plus efficace que la méthode MG dans certains cas (cas de Q3) ou plus coûteuse dans autres cas (cas de Q1). De plus, nous remarquons de manière générale que la méthode proposée (MOM) est plus efficace que les méthodes (MG et MC) en termes du temps de réponse et du volume de données transférées inter-sites.

Nous avons déjà noté que la méthode MOM impose deux modifications sur l'algorithme de routage utilisé par Chord. Même si la modification de l'algorithme de cet algorithme nécessite d'ajouter un saut sur le chemin parcouru durant la phase de localisation, la localisation de sources de données reste de l'ordre de log(N), ou N est le nombre total des nœuds participant au système. Un autre impact de la méthode MOM peut être remarqué est l'augmentation de la taille de la réponse de demande de localisation. Cette augmentation peut engendrer une augmentation du temps de localisation. Mais, l'augmentation du temps

de localisation reste inférieure au temps gagné par la méthode MOM. Si on suppose que la taille de réponse d'une demande de localisation selon Chord original soit 2 Ko et la bande passante réseau est 160 Kb/s, le temps nécessaire pour envoyer cette réponse est de 50 ms. Par contre, si on suppose que la taille de réponse de la même demande de localisation est 10 Ko selon la méthode MOM, le temps nécessaire pour envoyer cette réponse est de 250 ms. Alors, l'augmentation du temps de localisation est de l'ordre de quelques dizaines de millisecondes. Cependant, l'évaluation de performance montre que le temps gagné par la méthode MOM est de l'ordre de quelques dizaines de secondes.

4.5. Conclusion

Dans ce chapitre, nous avons validé nos propositions par l'implantation et le test d'un simulateur permettant de créer un système P2P structuré basé sur le protocole Chord [SMK+01]. Grâce à ce simulateur, nous avons évalué les performances de la méthode de création de *matching* entre les schémas locaux, de la méthode de localisation de sources de données et de la méthode d'obtention de méta-données.

Quant à la méthode de création de *matching* entre les schémas locaux, nous avons étudié l'impact de l'hétérogénéité sémantique sur la qualité des réponses de demandes de localisation de sources de données. Nous avons testé l'efficacité de *matching* basé sur les noms des membres des schémas locaux considérés. Nous avons également montré que l'utilisation des synonymes ajoutés par l'utilisateur peut améliorer la qualité de *matching* entre les schémas locaux. L'évaluation de performance de notre méthode de *matching* qui est basée sur une ontologie de domaine d'un côté et sur des synonymes ajoutés par l'utilisateur d'un autre côté, montre que notre méthode est plus efficace. Elle permet d'obtenir un pourcentage de réponses correctes plus élevé que la méthode basée sur les noms exacts et que la méthode basée sur des synonymes.

Nous avons observé que lors de la localisation de sources de données, le pourcentage de réponses incorrectes retenues par le protocole Chord peut atteindre le seuil de 97% dans certains cas. Cela est dû au problème de l'hétérogénéité structurelle entre les schémas locaux. Les réponses incorrectes influencent non seulement les résultats finaux des requêtes SQL considérées, mais elles consomment beaucoup de ressources sans aucun bénéfice. Le fait d'enrichir la DHT du système par des indexes de structure permet d'améliorer non seulement la qualité de réponses en évitant les réponses incorrectes, mais il permet aussi de diminuer d'une façon très claire le temps de localisation de sources de données. Nous avons également étudié l'impact de cet enrichissement en calculant la taille moyenne de la DHT sur

chaque nœud avant et après l'enrichissement. Nous avons trouvé que la taille moyenne de la DHT est presque doublée sur chaque nœud. Cependant, la valeur de cette augmentation est relativement petite, elle est égale à quelques centaines de kilo octets dans le pire des cas. Avec l'amélioration technologique au niveau de conception d'ordinateurs dont les mémoires vives peuvent atteindre quelques giga octets, cette valeur n'a pas un impact important sur les performances des services disponibles sur chaque nœud.

Finalement, nous avons étudié le fait de profiter de la phase de la localisation de sources de données pour obtenir toutes les méta-données nécessaires pour créer un plan d'exécution proche de l'optimal. Cette proposition permet non seulement d'éviter un double accès supplémentaire au réseau ayant une faible latence et une bande passante limitée, mais il permet aussi d'obtenir des méta-données stockées sur les hôtes des sources de données elles-mêmes. Ceci permet d'obtenir des méta-données à jour et par conséquent d'éviter les sous-optimalités des plans d'exécutions en évitant les erreurs liées au manque d'informations et aux statistiques obsolètes. L'évaluation de performance de cette proposition montre une amélioration notable en termes du temps de réponse et du volume de données transférées entres les nœuds lors de l'exécution d'une requête SQL.

Chapitre V

Sommaire

Conclusion générale

Ce chapitre est consacré à synthétiser les travaux effectués au sein de notre thèse. Nous résumons nos propositions et nous récapitulons les résultats obtenus. Nous discutons également les limites de ces propositions et nous citons des perspectives permettant d'étendre et d'améliorer les travaux effectués.

5.1. Synthèse et bilan

Grâce à son succès dans le domaine du partage des fichiers musicaux dans nos jours, le paradigme P2P attire, de plus en plus, l'attention des chercheurs en Informatique dans le monde entier. Contrairement au paradigme Client/Serveur qui peut créer des goulets d'étranglement et qui est moins tolérant aux pannes, le paradigme P2P permet de concevoir des systèmes bien adaptés aux environnements distribués à grande échelle. Les systèmes P2P permettent à un très grand nombre de nœuds de partager leurs ressources d'une façon autonome et décentralisée.

5.1.1. Contexte et problématique

A cause des limites des approches de partage de données réparties du point de vue du passage à grande échelle, une nouvelle tendance est apparue chez les chercheurs en bases de données pour faire combiner les techniques des bases de données réparties et celles utilisées dans les systèmes P2P. L'objectif d'une telle combinaison est de créer des systèmes P2P supportant le partage de données à une granulaire plus fine qu'un fichier et l'évaluation de requêtes complexes (i.e. SQL). Après l'état de l'art que nous avons effectué, nous avons constaté qu'il y a plusieurs travaux menés dans cette direction de recherche. Nous remarquons que ces travaux n'arrivent pas à proposer un système P2P de partage de données capable d'évaluer des requêtes complexes avec un temps de réponse relativement petit et une consommation de ressources relativement diminuée. Une des raisons de cette remarque est qu'en environnement P2P on peut avoir un catalogue global à cause des

caractéristiques de cet environnement (e.g. grande-échelle, autonomie et instabilité de nœuds). La localisation de sources de données et la génération d'un plan d'exécution proche de l'optimal sont deux problèmes ouverts liés directement à l'absence d'un catalogue global.

5.1.2. Approche d'évaluation de requêtes SQL

Etant donnée qu'il n'y a pas un consensus sur une architecture de système P2P de partage de données et sur comment une requête SQL est évaluée dans ce type de systèmes, nous avons proposé notre propre architecture et notre approche de l'évaluation de requêtes SQL au sein de cette architecture. L'approche proposée est fondée sur une ontologie de domaine connue par tous les nœuds et forme une interface unique pour les interactions entre les nœuds. Nous avons choisi d'utiliser une ontologie de domaine pour jouer le rôle d'un schéma global car l'ontologie de domaine représente explicitement la sémantique de ses termes et elle peut être disponible sur un site web et prête à être utilisée. Un schéma standard est difficile à être créé et il n'est pas compréhensif par un utilisateur final car la sémantique de ses termes est stockée dans la tête du concepteur du schéma qui peut être absent au moment de participation au système. L'approche proposée est constituée des phases suivantes : (i) reformulation de requêtes, (ii) localisation de sources de données et obtention de méta-données, (iii) optimisation globale et (iv) optimisation locale et exécution. Le type du système considéré par notre étude est un système P2P structuré permettant un routage de requête plus efficace que les systèmes P2P non-structurés en termes du nombre de messages échangés et de la capacité de retourner la totalité de réponses valides disponibles dans le système. Une autre raison d'utiliser un système P2P structuré est que ces systèmes sont plus tolérants aux pannes et aux défaillances que les systèmes P2P super-pairs qui peuvent créer des goulets d'étranglement au niveau des super-pairs.

5.1.3. Reformulation de requêtes

La phase de reformulation permet de réécrire une requête donnée en termes de l'ontologie de domaine. L'utilisation d'une telle ontologie permet de résoudre l'hétérogénéité sémantique entre les schémas locaux. Un adaptateur est responsable d'effectuer la phase de reformulation. Il utilise des règles de correspondance écrites selon une méthode semi-automatique utilisant des formules de similarité que nous avons également proposées. Nous avons comparé cette méthode avec les deux méthodes MNE (Méthode de Noms Exacts) et

MSs (Méthode basée sur Synonymes seulement) et nous avons montré que notre méthode est la plus efficace en termes de qualité de *matching* entre les schémas locaux.

5.1.4. Localisation de sources de données

En ce qui concerne la localisation de sources de données, nous avons utilisé le protocole Chord [SMK+01] pour effectuer le routage des demandes de localisation des sources des relations mentionnées par la requête évaluée. Ce protocole est simple, il permet de localiser les sources d'une relation représentée par une clé en O(Log N) sauts où N est le nombre total de nœuds dans le système. Cependant, il ne permet pas de distinguer la structure de la relation recherchée en termes d'attributs. Dans certains cas, il retourne jusqu'à 97% de réponses inutiles à cause de cette limite. Nous avons introduit le principe des indexes de structure au protocole Chord et nous avons étendu ce protocole afin de lui permettre de résoudre le problème de l'hétérogénéité structurelle entre les schémas locaux. Nous avons comparé les performances du protocole Chord dans les deux situations : avant et après l'extension. Nous avons remarqué que le protocole Chord étendu devient capable d'éviter les réponses inutiles. Ce qui diminue le temps de localisation. Le prix à payer pour atteindre ce résultat est l'augmentation de la taille moyenne de la DHT utilisée par Chord de quelques kilooctets sur chaque nœud. Cette augmentation est acceptable dans nos jours car, avec le développement rapide de l'architecture des ordinateurs, les mémoires vives peut avoir des valeurs importantes.

5.1.5. Obtention de méta-données

Quant à la phase de l'optimisation globale, nous avons remarqué que pour générer un plan d'exécution proche de l'optimal ce n'est pas nécessaire d'avoir un catalogue global dans lequel se trouvent des informations concernant tous les nœuds et tous leurs contenus. Cependant, il suffit d'avoir des informations concernant les données et les hôtes de leurs sources afin de générer un plan d'exécution proche de l'optimal et exécutable concernant les sources de données et le nœud initialisant la requête. Pour cette raison, nous avons proposé d'étendre notre méthode de localisation de sources de données pour qu'elle permette d'obtenir toutes les méta-données nécessaires pour effectuer l'optimisation globale. Le fait de chercher les méta-données sur les sources de données permet d'obtenir des informations fiables et à jour. Nous avons comparé notre méthode d'optimisation globale avec une méthode basée sur des informations locales seulement et avec la méthode gourmande qui

transfère toutes les données souhaités sur le nœud initialisant la requête afin d'effectuer l'exécution sur ce nœud. Les évaluations des performances montrent que notre méthode est la plus efficace en termes du temps de réponse et du volume de données transférés entre les sites.

Après avoir synthétisé les travaux effectués pendant la durée de notre thèse, nous mettons les points, dans la section suivante, sur quelques limitations concernant nos propositions. Nous allons discuter ces limitations et proposer des idées pour les éviter tout en expliquant nos futurs travaux.

5.2. Perspectives

Nous croyons que cette thèse n'est qu'un pas sur le chemin de concevoir un système P2P de partage de donnée fiable et capable d'être utilisé dans notre vie quotidienne. C'est vrai que nous avons utilisé au sein de notre étude le protocole Chord, mais rien n'empêche l'utilisation d'une autre technique utilisant la technologie de DHT. Comme nouveaux pas sur notre chemin, nous citons une partie de nos perspectives.

5.2.1. Mise en œuvre d'un prototype de système P2P de partage de données

Au sein de cette thèse, nous avons proposé une architecture logicielle d'un système P2P de partage de données. Nous avons évalué quelques modules de cette architecture, nous pensons à continuer à construire et à tester les autres modules de notre architecture. C'est vrai que nous avons effectué des simulations, mais ce n'était qu'un premier pas car nous n'avions pas une idée claire sur la faisabilité et la validité de nos propositions. Maintenant, nous pensons à effectuer des évaluations des performances dans un environnement expérimental réel. Nous pensons aussi à mettre en œuvre un prototype et à effectuer des comparaisons quantitatives entre notre prochain prototype et d'autres travaux voisins.

5.2.2. Étudier l'impact des indexes de structure sur l'entrée/sortie de nœuds

Nous avons déjà montré dans le chapitre 4 que l'enrichissement de la DHT par des index de structure augmente la taille moyenne de DHT sur chaque nœud de quelques centaines de kilooctets au pire des cas. Avec le développement rapide de la technologie de l'architecture matérielle des ordinateurs, cette augmentation ne crée pratiquement pas un vrai problème au niveau de stockage. Cependant, lors de l'entrée d'un nœud au système, ce nœud doit injecter ses indexes de structure dans le système et il doit les retirer lors de son

départ du système. Il faut étudier l'impact de l'augmentation de la taille des informations injectées dans le système lors de l'entrée d'un nœud.

5.2.3. Utilisation de plusieurs ontologies de domaine

Etant donné que pour chaque domaine ils peuvent se trouver plusieurs ontologies, notre approche d'évaluation de requêtes doit être étendue pour s'adapter avec cette situation. Dans cette perspective, on peut supposer que notre système est capable d'utiliser un nombre, relativement, petit d'ontologies de domaine et que chaque utilisateur peut choisir l'ontologie qui lui convient. Donc, il n'y aura plus la même ontologie sur tous les nœuds. En supposant que le *matching* entre ces ontologies est déjà créé par des spécialistes du domaine, la question qui se pose est comment stocker les règles de correspondance (RC) liées à ce *matching* ? Une des solutions proposées est de stocker les RC sur des nœuds stables dans le système. Ces nœuds sont connus par tous les autres nœuds. Dans ce cas-là, pour chaque requête il faut visiter ces nœuds afin de réécrire les requêtes en termes des autres ontologies. A notre avis, cette solution n'est pas pratique. Une autre solution est de trouver un moyen pour distribuer les RC sur tous les nœuds. Ce sujet forme un de nos futurs travaux.

Références

[ABC+03] S. Abiteboul, A. Bonifati, G. Cobena, I. Manolescu and T. Milo. *"Dynamic XML documents with distribution and replication"*. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD), pp. 527-538, 2003.

[Abe01] K. Aberer. *"P-Grid: A Self-Organizing Access Structure for P2P Information Systems"*. In Proc. of the 9th Int. Conf. on Cooperative Information Systems (CoopIS'01), pp. 179-194, 2001.

[ACD+03] K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva and R. Schmidt. *"P-Grid: a self-organizing structured P2P system"*. ACM SIGMOD Record, 32(3), pp. 29-33, 2003.

[ACK+02] D.P. Anderson, J. Cobb, E. Korpela, M. Lebofsky and D. Werthimer. *"SETI@home: an experiment in public-resource computing"*. Communications of the ACM, 45(11), pp. 56-61, 2002.

[ACP+96] S. Adali, K. Candan, Y. Papakonstantinou and V. Subrahmanian, *"Query Caching and Optimization in Distributed Mediator Systems"*, Proc. SIGMOD, pp. 137-148, 1996.

[ADM+05] D. Aumueller, H. H. Do, S. Massmann and E. Rahm. *"Schema and Ontology Matching with COMA++"*. In Proc. of the 2005 ACM SIGMOD (SIGMOD'05), pp. 906-908, 2005.

[AGT10] S. Alaei, M. Ghodsi and M. Toossi, "*Skiptree: A new scalable distributed data structure on multidimensional data supporting range-queries*", Journal of Computer Communications, V(33), N(1), pp. 73-82, January 2010.

[AHA03] D. Anwitaman, M. Hauswirth and K. Aberer. *"Updates in highly unreliable, replicated peer-to-peer systems"*. IEEE Int. Conf. on Distributed Computing Systems (ICDCS), pp. 76-85, 2003.

[AKK+03] M. Arenas, V. Kantere, A. Kementsietsidis, I. Kiringa, R.J. Miller and J. Mylopoulos. *"The hyperion project: from data integration to data coordination"*. *SIGMOD Rec.* 32, 3 (Sep. 2003), pp. 53-58, 2003.

[AkM07] R. Akbarinia and V. Martins. *"Data management in the APPA P2P system"*. Journal of Grid Computing, Vol 5(3), pp. 303-317 , September 2007.

[AMD+03] K. Aberer, P. C. Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva and R. Schmidt. *"P-Grid: a self-organizing structured P2P system"*. SIGMOD Rec. 32, 3 (Sep. 2003), pp. 29-33, 2003.

[AMH02] K. Aberer, P. C. Mauroux and M. Hauswirth. *"A Framework for Semantic Gossiping"*. ACM SIGMOD (SIGMOD'02) Record 31(4), pp. 48-53, 2002.

[AMH03] K. Aberer, P. C. Mauroux and M. Hauswirth. *"The chatty Web: emergent semantics through gossiping"*. Int. Conf. on World Wide Web (WWW'03), pp : 197-206, 2003.

[AMP+04] R. Akbarinia, V. Martins, E. Pacitti and P. Valduriez. "*Replication and query processing in the APPA data management system"*. Int. Workshop on Distributed Data and Structures (WDAS04), pp. 19-33, 2004.

[AMP+06] R. Akbarinia, V. Martins, E. Pacitti, and P. Valduriez. *"Design and implementation of APPA"*. Global Data Management (Eds. R. Baldoni, G. Cortese and F. Davide), IOS Press, 2006.

[AMP+07] R. Akbarinia, V. Martins, E. Pacitti and P. Valduriez. *"Top-k query processing in the APPA P2P system"*. Int. Conf. on High Performance Computing for Computational Science (VecPar), LNCS 4395, Springer, pp : 158-171, 2007.

[AnX02] A. Andrzejak and Z. Xu. *"Scalable, efficient range queries for grid information services"*. In Proc. of the IEEE Int. Conf. on P2P computing, pp : 33-40, 2002.

[APH+02] K. Aberer, M. Punceva, M. Hauswirth and R.Schmidt. *"Improving Data Access in P2P Systems"*. IEEE Internet Computing 6, 1 (Jan. 2002), pp. 58-67, 2002.

[APM+03] F.M. C. Acuna, C. Peery, R.P. Martin and T.D. Nguyen. *"PlanetP: using gossiping to build content addressable peer-to-peer information sharing communities"*. IEEE Int. Symp. on High Performance Distributed Computing (HPDC'03), pp. 236-249, 2003.

[BAH+06] R. Blanco, N. Ahmed, D. Hadaller, L.G.A. Sung, H. Li and M.A. Soliman. *"A survey of data management in peer-to-peer systems"*. Technical Report CS-2006-18, University of Waterloo, 2006.

[BaO03] B. Babcock and C. Olston. *"Distributed top-k monitoring"*. In Proc. of the ACM Int. Conf. on Management of Data (SIGMOD'03), pp. 28-39, 2003.

[BBK01] C. Böhm, S. Berchtold and D.A. Keim. *"Searching in high-dimensional spaces: index structures for improving the performance of multimedia databases"*. ACM Computing Surveys, 33(3), pp. 322-373, 2001.

[BCL+10] A. Bonifati, E. Chang, T. Ho, L. V. Lakshmanan, R. Pottinger and Y. Chung, "*Schema mapping and query translation in heterogeneous P2P XML databases*". The VLDB Journal, V(19), N(2), pp. 231-256, April 2010.

[BCO+08] A. Bonifati, P. K. Chrysanthis, A. M. Ouksel and K. U. Sattler. *"Distributed databases and Peer-to-Peer Databases: Past and Present"*. SIGMOD Rec. 37, 1 (Mar. 2008), pp : 5-11.

[BDK+03] I. Brunkhorst1, H. Dhraief, A. Kemper, W. Nejdl and C. Wiesner. *"Distributed Queries and Query Optimization in Schema-Based P2P-Systems"*. In Proc. of the Int. Workshop on Databases, Information Systems, and Peer-to-Peer Computing (DISP2P'03), pp. 184-199, 2003.

[BGK+02] P. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini and I. Zaihrayeu. *"Data management for peer-to-peer computing: a vision"*. In Proc. of the Int. Workshop on the Web and Databases (WebDB), pp : 89-94, 2002.

[BKR+04] A. Bhargava, K. Kothapalli, C. Riley, C. Scheideler, and M. Thober. *"Pagoda: a dynamic overlay network for routing, data management, and multicasting"*. In

Proc. of the ACM Symp. on Parallelism in Algorithms and Architectures (SPAA'04), pp. 170-179, 2004.

[BNS+05] W. T. Balke, W. Nejdl, W. Siberski and U. Thaden. *"Progressive distributed top-k retrieval in peer-to-peer networks"*. In Proc. of the Int. Conf. on Data Engineering (ICDE), pp : 174-185, 2005.

[BoT03] P. Boncz and C. Treijtel. *"AmbientDB: Relational Query Processing in a P2P Network"*. Int. Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P), 2003.

[CaF04] M. Cai and M. Frank. *"RDFPeers: a scalable distributed RDF repository based on a structured peer-to-peer network"*. Int. Conf. on World Wide Web (WWW'04), pp. 650-657, 2004.

[CaS06] J. Cardoso and A. P. Sheth, "Semantic Web Services, Processes and Applications", Semantic Web And Beyond Computing for Human Experience, Vol. 3, Springer 2006, ISBN 978-0-387-30239-3.

[CaW04] P. Cao and Z. Wang. *"Efficient top-k query calculation in distributed networks"*. ACM Symp. on Principles of Distributed Computing (PODC), 206-215, 2004.

[CBB+07] M. A. Casanova, K. K. Breitman, D. F. Brauner and A. L.A. Marins. *"Database Conceptual Schema Matching"*, In the Computer magazine, IEEE Computer Society, Vol. 40, Iss. 10, pp. 102-104, 2007.

[CDK+02] M. Castro, P. Druschel, A. M. Kermarrec and A. Rowstron. *"SCRIBE: a largescale and decentralized application-level multicast infrastructure"*. IEEE Journal on Selected Areas in Communication (JSAC), 20(8), pp. 1489-1499, 2002.

[CGM04] S. Chaudhuri, L. Gravano and A. Marian. *"Optimizing top-k selection queries over multimedia repositories"*. IEEE Transactions on Knowledge and Data Engineering, 16(8), pp. 992- 1009, 2004.

[CFC+03] M. Cai, M. Frank, J. Chen and P. Szekely. *"MAAN: a multi-attribute addressable network for grid information services"*. Int. Workshop on Grid Computing, pp : 184-191, 2003.

[CiP02] P. Ciaccia and M. Patella. *"Searching in metric spaces with user-defined and approximate distances"*. ACM Transactions on Database Systems, 27(4), pp. 398-437, 2002.

[CJK+03] M. Castro, M.B. Jones, A-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang and A. Wolman. *"An evaluation of scalable application-level multicast built using peer-to-peer overlays"*. Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM'03), pp. 1510-1520, 2003.

[CLM+04] A. Crainiceanu, P. Linga, A. Machanavajjhala, J. Gehrke, and J. Shanmugasundaram. *"P-ring: An index structure for peer-to-peer systems"*. In Cornell Technical Report, 2004.

[CLM+07] A. Crainiceanu, P. Linga, A. Machanavajjhala, J. Gehrke and J. Shanmugasundaram. *"P-ring: an efficient and robust P2P range index structure"*.

In Proc. of the 2007 ACM SIGMOD int. Conf. on Management of Data (SIGMOD '07), pp : 223-234, 2007.

[CMH+02] I. Clarke, S. Miller, T.W. Hong, O. Sandberg and B. Wiley. *"Protecting free expression online with Freenet"*. IEEE Internet Computing, 6(1), pp. 40-49, 2002.

[CMH+97] S. Chawathe, H. G. Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman and J. Widom. *"The TSIMMIS Project: Integration of Heterogeneous Information Sources"*. Journal of Intelligent Information Systems, vol. 8, No. 2, pp. 117-132, March, 1997.

[CRB+03] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham and S. Shenker. *"Making Gnutella-like P2P systems scalable"*. ACM SIGCOMM Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications, pp. 407-418, 2003.

[CRB99] B. Chandrasekaran, J. R. Josephson and V. R. Benjamins. "*What Are Ontologies, and Why Do We Need Them?"*, IEEE Intelligent Systems, pp. 20-26, 1999.

[CrM02] A. Crespo, and H. G. Molina. *"Routing indices for peer-to-peer systems"*. IEEE Int. Conf. on Distributed Computing Systems" (ICDCS'02), pp : 23-33, 2002.

[CrM03] A. Crespo and H. G. Molina. *"Semantic Overlay Networks for P2P Systems"*. Technical report, Stanford University, January 2003.

[CTT+08] M. Cannataro, D. Talia, G. Tradigo, P. Trunfio and P. Veltri, "*SIGMCC: A system for sharing meta patient records in a Peer-to-Peer environment*". Journal of Future Generation Computer Systems, V(24), N(3), pp.222-234, March 2008.

[DFM00] R. Dingledine, M. Freedman and D. Molnar. *"The FreeHaven project: distributed anonymous storage service"*. Workshop on Design Issues in Anonymity and Unobservability, pp. 67-95, 2000.

[DMY03] N. Daswani, H. G. Molina and B. Yang. *"Open problems in data-sharing peer-to-peer systems"*. Int. Conf. on Database Theory (ICDT'03), pp. 1-15, 2003.

[DMD+03] A. Doan, J. Madhavan, R. Dhamankar, P. Domingos and A. Halevy. *"Learning to match ontologies on the semantic web"*. VLDB Journal, 12(4), pp. 303-319, 2003.

[DMN+02] A.P. DeVries, N. Mamoulis, N. Nes and M.L. Kersten. *"Efficient k-NN search on vertically decomposed data"*. ACM Int. Conf. on Management of Data (SIGMOD'02), pp. 322-333, 2002.

[DuG97] O.M. Duschka and M.R. Genesereth, "Answering Recursive Queries Using Views" Proc. 16th ACM SIGACT-SIGMODSIGART Symposium. Principles of Database Systems, pp. 109-116, 1997.

[Dup01] A. Duphil. *"Conception et implantation d'une plate forme de simulation pour évaluer les performances des méthodes de placement de tâches de programmes bases de données sur une architecture parallèle à mémoire distribuée''*, Mémoire CNAM, IRIT, Toulouse, Juin, 2001.

[ESP02] J. O. Everett, R. Stolle, V. d. Paiva, D. G. Bobrow, R. Crouch, C. Condoravdi. M. van den Berg and L. Polanyi. "*Making ontologies work for resolving redundancies across documents"*. Communication of the ACM, Vol. 45, No. 2, pp. 55-60, 2002.

[FKL+04] E. Franconi, G. Kuper, A. Lopatenko, I. Zaiharayeu, "*Queries and Updates in the coDB Peer to Peer Database System*", in the proceedings of the 30th VLDB, pp. 1277-1280, 2004.

[GAA03] A. Gupta, D. Agrawal and A. El Abbadi. *"Approximate range selection queries in peer-to-peer systems"*. First Biennial Conference on Innovative Data Systems Research (CIDR'03), pp. 141-151, 2003.

[GaS04] J. Gao and P. Steenkiste. *"An adaptive protocol for efficient support of range queries in DHT-based systems"*. IEEE Int. Conf. on Network Protocols (ICNP'04), pp. 239-250, 2004.

[GGG+03] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. *"The impact of DHT routing geometry on resilience and proximity"*. ACM Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM'03), pp.381-394, 2003.

[GGJ+05] P. R. Gianolli, M. Garzetti, L. Jiang, A. Kementsietsidis, I. Kiringa, M. Masud, R. J. Miller and J. Mylopoulos. *"Data Sharing in the Hyperion Peer Database System"*. In Proc. of the VLDB 2005, pp. 1291–1294, 2005.

[GHI+01] S. Gribble, A. Halevy, Z. Ives, M. Rodrig and D. Suciu. *"What Can Databases Do for Peer-to- Peer?"*. In Proc. of the WebDB Workshop on Databases and the Web, 2001.

[GiZ04] F. Giunchiglia and I. Zaihrayeu. *"Implementing Database Coordination in P2P Networks"*. The Second Workshop on Semantics in Peer-to-Peer and Grid Computing (SemPGrid'04), 2004.

[GKB00] U. Güntzer, W. Kießling and W.T Balke. *"Optimizing multi-feature queries for image databases"*. Int. Conf. on Very Large Databases (VLDB), 419-428, 2000. inria-00128221, version 2 - 6 Feb 2007

[Gon01] L. Gong. *"Project JXTA: A Technology Overview"*. Sun Microsystems, Palo Alto, CA, USA, 2001.

[GPH05] Y. Guo, Z. Pan and J. Hein. *"LUBM: A benchmark for OWL knowledge base systems"*. Journal of Web Semantics, 3(2-3): pp : 158-182, 2005.

[Gru93] T. R. Gruber. *"A Translation approach to portable ontology specifications"*. International Journal of Knowledge Acquisition for Knowledge-based Systems, Vol. 5, No.2 , 1993.

[GWJ+03a] L. Galanis, Y. Wang, S. R. Jeffery and D. J. DeWitt. *"Locating data sources in large distributed systems"*. In Proc. of the 29th VLDB Conference (VLDB'O3), pp. 874-885, 2003.

[GWJ+03b] L. Galanis, Y. Wang, S. R. Jeffery and D. J. DeWitt.*"Processing Queries in a Large Peer-to-Peer System"*. (CAiSE 2003), LNCS 2681, pp. 273–288, 2003.

[FLN03] R. Fagin, J. Lotem and M. Naor. *"Optimal aggregation algorithms for middleware"*. Journal of Computer and System Sciences, 66(4), pp. 614-656, 2003.

[FNV07] D. Faye, G. Nachouki and P. Valduriez, *"Semantic Query routing in SenPeer, a P2P Data Management System"*, Network-Based Information systems, first international conference, NBis2007, Regensburg, Germany, September, 2007.

[Hal05] Halevy, A. *"Why Your Data Won't Mix: Semantic Heterogeneity"*. ACM Queue 3(8), pp. 50–58, 2005.

[HaS04] P. Haase and R. Siebes. "*Peer selection in Peer-to-Peer networks with semantic topologies"*. WWW'04, pp. 75-107, 2004

[HCH+05] R. Huebsch, B. Chun, J. M. Hellerstein, B. T. Loo, P. Maniatis, T. Roscoe, S. Shenker, I. Stoica and A. R. Yumerefendi. *"The Architecture of PIER: an Internet-Scale Query Processor"*. In Proc. of the 2005 CIDR Conf. pp : 28-43, 2005.

[Hel 04] J. Hellerstein. *"Architectures and Algorithms for Internet-Scale (P2P) Data Management"*. VLDB'04, 1244, 2004.

[HHH+02] M. Harren, J.M. Hellerstein, R. Huebsch, B.T. Loo, S. Shenker and I. Stoica. *"Complex queries in DHT-based peer-to-peer networks"*. Int. Workshop on Peerto-Peer Systems (IPTPS'02), pp. 242-259, 2002.

[HHL+03] R. Huebsch, J. M. Hellerstein, N. Lanham, B. T. Loo, S. Shenker and I. Stoica. *"Querying the Internet with PIER"*. In Proc. of the 29th Int. Conf. on Very Large Data Bases (VLDB'03), pp : 321-332, 2003.

[HIM+03] A. Halevy, Z. Ives, P. Mork and I. Tatarinov. *"Piazza: data management infrastructure for semantic web applications"*. Int. Conf. on World Wide Web (WWW'03), pp : 556-567, 2003.

[HjS03] G.R. Hjaltason and H. Samet. *"Index-driven similarity search in metric spaces. ACM Transactions on Database Systems"*, 28(4), pp. 517-580, 2003.

[HJS+03] N. Harvey, M. Jones, S. Saroiu, M. Theimer and A. Wolman. *"Skipnet: a scalable overlay network with practical locality properties"*. USENIX Symp. on Internet Technologies and Systems, pp : 113-126, 2003.

[IKS+03] N. Ishikawa, T. Kato, H. Sumino, J. Hjelm, Y. Yu and Z. Zhu. "A platform for peer-to-peer communications and relation to semantic web application". The 12th Int. World Wide Web conf ,2003.

[IQN+09] A. Ismail, M. Quafafou, G. Nachouki and M. Hajjar, "*Data mining effect in peer-to-peer queries routing*". In Proceedings of the International Conference on Management of Emergent Digital Ecosystems (MEDES '09), ACM, NY, pp.65-72, 2009.

[JAB01] M. Jovanovic, F. Annexstein and K. Berman. "*Scalability issues in large peer-topeer networks: a case study of Gnutella"*. Technical report, ECECS Department, University of Cincinnati, 2001.

[JoC09] Y. Joung and F. Chuang, "*OntoZilla: An ontology-based, semi-structured, and evolutionary peer-to-peer network for information systems and services*". Journal of Future Generation Computer Systems, V(25), N(1), pp. 53-63, January 2009.

[JOV05] H.V. Jagadish, B. C. Ooi and Q. H. Vu. "BATON: A Balanced Tree Structure for Peer-to-Peer Networks". VLDB'05, pp. 661-672, 2005.

[JWZ03] R. Janakiraman, M. Waldvogel and Q. Zhang. *"Indra: a peer-to-peer approach to network intrusion detection and prevention"*. IEEE Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03), pp. 226-231, 2003.

[KAM03] A. Kementsietsidis M. Arenas and R.J. Miller. *"Managing Data Mapping in Hyperion Project"*. ICDE'03, Bangalore, India, 2003.

[KBC+00] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells and B. Zhao. *"Ocean- Store: an architecture for global-scale persistent storage"*. ACM Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp. 190-201, 2000.

[KeR02] A.V.M. Keromytis and D. Rubenstein. *"SOS: secure overlay services"*. ACM SIGCOMM Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications, pp. 61-72, 2002.

[KGY02] V. Kalogeraki, D. Gunopoulos and D. Z. Yazti. *"A local search mechanism for peer-to-peer networks"*. ACM Int. Conf. on Information and Knowledge Management (CIKM'02), pp. 300-307, 2002.

[KHM07a] R. A. King, A. Hameurlain and F. Morvan. "*Metadata Lookup for Distributed Query Optimization in P2P Environment"*. International Conference on Parallel and Distributed Computing Systems (PDCS'07), International Society for Computers and their Applications (ISCA), pp. 36-43, 2007.

[KHM07b] R. A. King, A. Hameurlain and F. Morvan. *"O-Chord: A Method for Locating Relational Data Sources in a P2P Environment"*. International Conference on Systems, Computing Sciences and Software Engineering (SCSS'07), pp. 416-421, 2008.

[KHM08a] R. A. King, A. Hameurlain and F. Morvan. *"Ontology-Based Query Processing in a Large-Scale P2P Environment"*. IEEE International Conference on Information and Communication Technologies: from Theory to Applications (ICTTA'08), 2008.

[KHM08b] R. A. King, A. Hameurlain and F. Morvan. "*Ontology-Based Data Source Localization in a Structured Peer-to-Peer Environment"*. International Database Engineering & Applications Symposium (IDEAS'08), pp. 9-18, 2008.

[KHM09a] R. A. King, A. Hameurlain and F. Morvan. *"Ontology-Based Method for Schema Matching in Peer-to-Peer Database System"*. British National Conference on Databases (poster session) (BNCOD'9), pp. 208-212, 2009.

[KHM09b] R. A. King, A. Hameurlain and F. Morvan. *"Matching Heterogeneous Schemas in a Large-Scale Peer-to-Peer Database Environment"*. Int. Conf. on Parallel and Distributed Computing and Communication Systems (PDCCS'09), International Society for Computers and their Applications (ISCA), pp. 135-142, 2009.

[KHM10] R. A. King, A. Hameurlain et F. Morvan, "*Query Routing and Processing in Peer-To-Peer Data Sharing Systems*". International Journal of Database Management Systems, Academy & Industry Research Collaboration Center (AIRCC), Vol. 2 N. 2, pp. 116-139, Mai 2010.

[KOT04] N. Koudas, B.C. Ooi, K.L. Tan and R. Zhang. "*Approximate NN queries on streams with guaranteed error/performance bounds"*. Int. Conf. on Very Large Databases (VLDB'04), pp. 804-815, 2004.

[KTS09] V. Kantere, D. Tsoumakos, T. Sellis and N. Roussopoulos "*GrouPeer: Dynamic clustering of P2P databases*". Journal of Information Systems. V(34), N(1), pp. 62-86, March 2009.

[LCC+02] Q. Lv, P. Cao, E. Cohen, K. Li and S. Shenker. *"Search and replication in unstructured peer-to-peer networks"*. ACM Int. Conf. on Supercomputing (ICS'02), pp : 84- 95, 2002.

[Len02] M. Lenzerini. *"Data integration: a theoretical perspective"*. ACM Symp. on Principles of Distributed Computing (PODC'02), pp. 233-246, 2002.

[LRO96] A.Y. Levy, A. Rajaraman and J. J. Ordille. *"Querying Heterogeneous Information Sources Using Source Descriptions"*, In Proceedings of 22nd VLDB, pp. 251-262, 1996.

[LRS+02] K. Lakshminarayanan, A. Rao, I. Stoica and S. Shenker. *"Flexible and robust large scale multicast using i3"*. Technical report CSD-02-1187, University of California, Berkeley, 2002.

[MAA07] P. C. e . Mauroux, S. Agarwal and K. Aberer. *"GridVine: An Infrastructure for Peer Information Management"*. IEEE Internet Computing, 11(5), pp. 36-44, 2007.

[MaH08] M. Masud and M. A. Hossain, "*Dynamic Query Plan for Efficient Query Processing in Peer-to-Peer Environments*", INFOCOMP Journal of Computer Science, Vol.7 (3). pp.01-06, 2008.

[MaM02] P. Maymounkov and D. Mazieres. *"Kademlia: a peer-to-peer information system based on the XOR metric"*. Int. Workshop on Peer-to-Peer Systems (IPTPS), pp. 53- 65, 2002.

[MAP+06] V. Martins, R. Akbarinia, E. Pacitti and P. Valduriez. *"Reconciliation in the APPA P2P system"*. IEEE Int. Conf. on Parallel and Distributed Systems (ICPADS'06), pp. 401-410, 2006.

[MBD+05] J. Madhavan, P.A. Bernstein, A. Doan and A. Halevy. *"Corpus-based schema matching"*. In Proc. of the Int. Conf. on Data Engineering (ICDE'05), pp. 57-68, 2005.

[McP04] P. McBrien and A. Poulovassilis. *"Defining peer-to-peer data integration using both as view rules"*. Int. Workshop on Databases, Information Systems and Peerto- Peer Computing, pp. 91-107, 2004.

[MeK02] D. Menascé and L. Kanchanapalli. *"Probabilistic scalable P2P resource location services"*. SIGMETRICS Performance Evaluation Review, 30(2), pp. 48-58, 2002.

[MKK05] M. M. Masud, I. Kiringa and A. Kementsietsidis, *"Don't Mind Your Vocabulary: Data Sharing Across Heterogeneous Peers"*. (CoopIS'05), pp. 292-309, 2005.

[MKL+02] D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins and Z. Xu. *"Peer-to-Peer Computing"*. Tech. Report: HPL-2002-57, available on line at: http://www.hpl.hp.com/techreports/2002/HPL-2002-57.pdf

[MNR02] D. Malkhi, M. Naor and D. Ratajczak. *"Viceroy: a scalable and dynamic emulation of the butterfly"*. ACM Symp. on Principles of Distributed Computing (PODC'02), pp. 183-192, 2002.

[MPQ+97] H. G. Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman and J. Widom. *"The TSIMMIS project: Integration of heterogeneous information sources"*. Journal of Intelligent Information Systems, 8(2), March 1997.

[MTW05] S. Michel, P. Triantafillou and G. Weikum. *"KLEE: a framework for distributed top-k query algorithms"*. Int. Conf. on Very Large Databases (VLDB'05), pp. 637-648, 2005.

[NeR99] S. Nepal and M.V. Ramakrishna. *"Query processing issues in image (multimedia) databases"*. Int. Conf. on Data Engineering (ICDE'99), pp. 22-29, 1999.

[NOT02] W. S. Ng, B. C. Ooi, K. L. Tan "*BestPeer: A self-configurable peer-to-peer system"*. In Proc. of the 18th Int. Conf. on Data Engineering (ICDE'02), pp. 272, 2002.

[NOT+03] W. S. Ng, B. C. Ooi, K. L. Tan and A. Zhou. *"PeerDB: a P2P-based system for distributed data sharing"*. Int. Conf. on Data Engineering (ICDE'03), pp : 633-644, 2003.

[NSS03] W. Nejdl, W. Siberski and M. Sintek. *"Design issues and challenges for RDFand schema-based peer-to-peer systems"*. ACM SIGMOD Record, 32(3), pp. 41-46, 2003.

[NWQ+02] W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmer, and T. Risch. *"Edutella: a P2P networking infrastructure based on RDF"*. Int. Conf. on World Wide Web (WWW'02), pp. 604-615, 2002.

[OST03] B. C. Ooi, Y. Shu and K. L. Tan. *"Relational Data Sharing in Peer-based Data Management Systems"*. ACM SIGMOD, 23 (3), pp : 59-64, 2003.

[OzV99] M.T. Ozsu and P. Valduriez. *"Principles of Distributed Database Systems"*. Prentice Hall ISBN 0-13-659707-6, 1999. Second Edition.

[PaM01] V. Papadimos and D. Maier "*Mutant Query Plans*". In OOPSLA, 2001.

[PaM03] V. Papadimos and D. Maier "*Distributed Queries without Distributed State*". In WebDB, 2002.

[PMT03] V. Papadimos, D. Maier and K. Tufte. *"Distributed Query Processing and Catalogs for Peer-to-Peer Systems"*. The 2003 CIDR Conference, 2003.

[PVM07] E. Pacitti, P. Valduriez and Marta Mattoso, *"Grid Data Management: Open Problems and New Issues"*. Journal of Grid Computing 5(3): pp. 273-281, 2007

[RFH+01] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker. *"A scalable content-addressable network"*. ACM SIGCOMM Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications, pp. 161-172, 2001.

[RhK02] S. Rhea and J. Kubiatowicz. *"Probabilistic location and routing"*. Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM'02), pp. 1248- 1257, 2002.

[RoD01a] A. Rowstron and P. Druschel. *"Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems"*. IFIP/ACM Int. Conf. on Distributed Systems Platforms (Middleware), pp. 329-350, 2001.

[RoD01b] A. Rowstron and P. Druschel. *"Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility"*. ACM Symp. on Operating Systems Principles (SOSP'01), pp. 188-201, 2001.

[RPL+09] C. Roncancio, M. Pilar Villamil, C. Labbé and P. Serrano-Alvarado, "*Data Sharing in DHT Based P2P Systems*". In Transactions on Large-Scale Data- and Knowledge-Centered Systems I, A. Hameurlain, J. Küng, and R. Wagner, Eds. Lecture Notes In Computer Science, vol. 5740. Springer-Verlag, Berlin, Heidelberg, pp. 327-352, 2009.

[RSW05] P. Röosch, K. Sattler, C. von der Weth and E. Buchmann. *"Best Effort Query Processing in DHT-based P2P Systems"*. In ICDE Workshop NetDB'05, 2005.

[SeR06] J. Seidenberg and A. L. Rector. *"Web ontology segmentation: analysis, classification and use"*. WWW'06, pp. 13-22. ACM, 2006.

[SGA02] O. D. Sahin, A. Gupta, D. Agrawal and A. El Abbadi. *"Query processing over peerto-peer data sharing systems"*. Technical Report 2002-28, University of California, Santa Barbara, 2002.

[SGA+ 04] O. D. Sahin, A. Gupta, D. Agrawal and A. El Abbadi. *"A Peer-to-peer Framework for Caching Range Queries"*. The 20th Int. Conf. on Data Engineering (ICDE'04), 2004.

[SGG03] S. Saroiu, K. P. Gummadi and S. D. Gribble, *"Measuring and analyzing the characteristics of Napster and Gnutella hosts"*. Multimedia Systems. Vol. 9, no. 2 pp : 170-184, 2003.

[SGM+03] L. Serafini, F. Giunchiglia, J. Mylopoulos and P. A. Bernstein. *"Local Relational Model: A Logical Formalization of Database Coordination"*. In Proc. of the 4th International and Interdisciplinary Conference on Modeling and Using Context (Context'03), pp : 286-299, 2003.

[ShE05] P. Shvaiko and J. Euzenat. *"A survey of schema-based matching approaches"*. Journal on Data Semantics. IV. LNCS, vol. 3730, pp. 146–171. Springer, Heidelberg, 2005.

[Shi01] C. Shirky. *"What is P2P and What Isn't"*. The O'Reilly Peer to Peer and Web Service Conf., Washington, D.C. November 5-8, 2001. Available on: http://conferences.oreillynet.com/p2p/.

[SiG10] M. Sínchez-Artigas and P. García-López, "*eSciGrid: A P2P-based e-science Grid for scalable and efficient data sharing*". Journal of Future Generation Computer Systems, V(26), N(5), pp.704-719, May 2010.

[SMK+01] I. Stoica, R. Morris, D.R. Karger, M.F. Kaashoek and H. Balakrishnan. *"Chord: a scalable peer-to-peer lookup service for internet applications"*. ACM Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'01), pp. 149-160, 2001.

[SNG10] I. Sarr, H. Naacke and S. Gançarski, "*TransPeer: adaptive distributed transaction monitoring for Web2.0 applications*". In Proceedings of the 2010 ACM Symposium on Applied Computing (SAC '10), ACM, NY, pp. 423-430, 2010.

[SSD+02a] M. Schlosser, M. Sintek, S. Decker and W. Nejdl. *"HyperCuP"*. Technical Report, Stanford University, 2002.

[SSD+02b] M. Schlosser, M. Sintek, S. Decker and W. Nejdl. *"HyperCuP – Hypercubes, Ontologies and Efficient Search on P2P Networks"*. International Workshop on Agents and Peer-to-Peer Computing, 2002.

[StR05] D. Stutzbach and R. Rejaie. "*Characterizing Today's Gnutella Topology"*. ACM SIGMETRICS, Alberta Canada, 2005.

[SuL06] W. Sun and D. X. Liu, *"Using Ontologies for Semantic Optimization of XML Databases"*, KDXD 2006, Singapore, April, 2006.

[TaH 04] I. Tatarinov and A. Halevy. *"Efficient Query Reformulation in Peer Data Management System"*. ACM SIGMOD'04, pp. 539-550, 2004.

[ThS04] S. A. Theotokis and D. Spinellis. *"A survey of peer-to-peer content distribution technologies"*. ACM Computing Surveys, 36(4), pp. 335-371, 2004.

[TIM+03] I. Tatarinov, Z.G. Ives, J. Madhavan, A. Halevy, D. Suciu, N. Dalvi, X. Dong, Y. Kadiyska, G. Miklau and P. Mork. *"The Piazza peer data management project"*. ACM SIGMOD Record, 32(3), pp : 47-52, 2003.

[TJW+10] Y. Tao, H. Jin, S. Wu and X. Shi, "*Scalable DHT- and ontology-based information service for large-scale grids*". Journal of Future Generation Computer Systems, V(26), N(5), pp. 729-739, May 2010.

[TrP03] P. Triantafillou and T. Pitoura. *"Towards a unifying framework for complex query processing over structured peer-to-peer data networks"*. Int.Workshop on Databases, Information Systems and Peer-to-Peer Computing, pp. 169-183, 2003.

[TRV98] A. Tomasic, L. Raschid and P. Valduriez. *"Scaling access to heterogeneous data sources with DISCO"*. IEEE Transactions on Knowledge and Data Engineering, 10(5), 808-823, 1998.

[TsR03a] D. Tsoumakos and N. Roussopoulos. *"A Comparison of peer-to-peer search methods"*. Int. Workshop on the Web and Databases (WebDB), pp. 61-66, 2003.

[TsR03b] D. Tsoumakos and N. Roussopoulos. *"Adaptive probabilistic search (APS) for peer-to-peer networks"*. In Proc. of the Int. IEEE Conference on P2P Computing, pp : 102-109, 2003.

[Tve77] A. Tversky. "Features of similarity". Psychological Review, 84:327－352, 1977

[Ull97] J.D. Ullman. *"Information integration using logical views"*. In Proc. of the Int. Conf. on Database Theory (ICDT'97), pp. 19-40, 1997.

[Val93] P. Valduriez. *"Parallel database systems: open problems and new issues"*. Distributed and Parallel Databases, 1(2), pp. 137-165, 1993.

[VaP04] P. Valduriez and E. Pacitti, *"Data Management in Large-Scale P2P Systems"*. VECPAR'04, pp. 104-118, 2004.

[VRL04] M.P. Villamil, C. Roncancio, C. Labbé, *"PinS: Peer-to-Peer Interrogation and Indexing System"*. IDEAS'04, pp. 236-245, 2004.

[VTS04] V. Vlachos, S. A. Theotokis and D. Spinellis. *"Security applications of peer-to-peer networks"*. Computer Networks Journal, 45(2), pp. 195-205, 2004.

[Wie92] G. Wiederhold. *"Mediators in the architecture of future information systems"*. IEEE Computer, 25(3), pp. 38-49, 1992.

[XCK06] Y. Xia, S. Chen and V. Korgaonkar. *"Load balancing with multiple hash functions in peer-to-peer networks*". IEEE Int. Conf. on Parallel and Distributed Systems (ICPADS'06), pp. 411-420, 2006.

[YaM02a] B. Yang, H. G. Molina. *"Improving search in peer-to-peer networks"*. In Proc. of the IEEE Int. Conf. on Distributed Computing Systems (ICDCS'02), pp. 5-14, 2002.

[YaM02b] B. Yang, H. G. Molina. *"Comparing Hybrid Peer-to-Peer Systems"*. VLDB'02, pp. 561-570 2002.

[YaM03] B. Yang and H. G. Molina. *"Designing a Super-Peer Network"*. ICDE'03, 2003.

[ZHS04] B.Y. Zhao, L. Huang, J. Stribling, S.C. Rhea, A.D. Joseph and J.D. Kubiatowicz. *"Tapestry: a resilient global-scale overlay for service deployment"*. IEEE Journal on Selected Areas in Communications (JSAC), 22(1), pp : 41-53, 2004.

Résumé

Malgré leur succès dans le domaine du partage de fichiers, les systèmes P2P sont capables d'évaluer uniquement des requêtes simples basées sur la recherche d'un fichier en utilisant son nom. Récemment, plusieurs travaux de recherche sont effectués afin d'étendre ces systèmes pour qu'ils permettent le partage de données avec une granularité fine (i.e. un attribut atomique) et l'évaluation de requêtes complexes (i.e. requêtes SQL).

A cause des caractéristiques des systèmes P2P (e.g. grande-échelle, instabilité et autonomie de nœuds), il n'est pas pratique d'avoir un catalogue global qui contient souvent des informations sur: les schémas, les données et les hôtes des sources de données. L'absence d'un catalogue global rend plus difficiles: (i) la localisation de sources de données en prenant en compte l'hétérogénéité de schémas et (ii) l'optimisation de requêtes.

Dans notre thèse, nous proposons une approche pour l'évaluation des requêtes SQL en environnement P2P. Notre approche est fondée sur une ontologie de domaine et sur des formules de similarité pour résoudre l'hétérogénéité sémantique des schémas locaux. Quant à l'hétérogénéité structurelle de ces schémas, elle est résolue grâce à l'extension d'un algorithme de routage de requêtes (i.e. le protocole Chord) par des Indexes de structure. Concernant l'optimisation de requêtes, nous proposons de profiter de la phase de localisation de sources de données pour obtenir toutes les méta-données nécessaires pour générer un plan d'exécution proche de l'optimal. Afin de montrer la faisabilité et la validité de nos propositions, nous effectuons une évaluation des performances et nous discutons les résultats obtenus.

Mots clés : *P2P, Bases de données, Hétérogénéité de schémas, DHT, Ontologie, Évaluation des performances.*

Abstract

Despite of their great success in the file sharing domain, P2P systems support only simple queries usually based on looking up a file by using its name. Recently, several research works have made to extend P2P systems to be able to share data having a fine granularity (i.e. atomic attribute) and to process queries written with a highly expressive language (i.e. SQL).

The characteristics of P2P systems (e.g. large-scale, node autonomy and instability) make impractical to have a global catalog that stores often information about data, schemas and data source hosts. Because of the absence of a global catalog, two problems become more difficult: (i) locating data sources with taking into account the schema heterogeneity and (ii) query optimization.

In our thesis, we propose an approach for processing SQL queries in a P2P environment. To solve the semantic heterogeneity between local schemas, our approach is based on domain ontology and on similarity formulas. As for the structural heterogeneity of local schemas, it is solved by the extension of a query routing method (i.e. Chord protocol) with Structure Indexes. Concerning the query optimization problem, we propose to take advantage of the data source localization phase to obtain all metadata required for generating a close to optimal execution plan. Finally, in order to show the feasibility and the validity of our propositions, we carry out performance evaluations and we discuss the obtained results.

Keywords: *P2P, Databases, Schema Heterogeneity, DHT, Ontology, Performance Evaluations.*

Résumé

Malgré leur succès dans le domaine du partage de fichiers, les systèmes P2P sont capables d'évaluer uniquement des requêtes simples basées sur la recherche d'un fichier en utilisant son nom. Récemment, plusieurs travaux de recherche sont effectués afin d'étendre ces systèmes pour qu'ils permettent le partage de données avec une granularité fine (i.e. un attribut atomique) et l'évaluation de requêtes complexes (i.e. requêtes SQL).

A cause des caractéristiques des systèmes P2P (e.g. grande-échelle, instabilité et autonomie de nœuds), il n'est pas pratique d'avoir un catalogue global qui contient souvent des informations sur: les schémas, les données et les hôtes des sources de données. L'absence d'un catalogue global rend plus difficiles: (i) la localisation de sources de données en prenant en compte l'hétérogénéité de schémas et (ii) l'optimisation de requêtes.

Dans notre thèse, nous proposons une approche pour l'évaluation des requêtes SQL en environnement P2P. Notre approche est fondée sur une ontologie de domaine et sur des formules de similarité pour résoudre l'hétérogénéité sémantique des schémas locaux. Quant à l'hétérogénéité structurelle de ces schémas, elle est résolue grâce à l'extension d'un algorithme de routage de requêtes (i.e. le protocole Chord) par des Indexes de structure. Concernant l'optimisation de requêtes, nous proposons de profiter de la phase de localisation de sources de données pour obtenir toutes les méta-données nécessaires pour générer un plan d'exécution proche de l'optimal. Afin de montrer la faisabilité et la validité de nos propositions, nous effectuons une évaluation des performances et nous discutons les résultats obtenus.

Mots clés : P2P, Bases de données, Hétérogénéité de schémas, DHT, Ontologie, Évaluation des performances.

Abstract

Despite of their great success in the file sharing domain, P2P systems support only simple queries usually based on looking up a file by using its name. Recently, several research works have made to extend P2P systems to be able to share data having a fine granularity (i.e. atomic attribute) and to process queries written with a highly expressive language (i.e. SQL).

The characteristics of P2P systems (e.g. large-scale, node autonomy and instability) make impractical to have a global catalog that stores often information about data, schemas and data source hosts. Because of the absence of a global catalog, two problems become more difficult: (i) locating data sources with taking into account the schema heterogeneity and (ii) query optimization.

In our thesis, we propose an approach for processing SQL queries in a P2P environment. To solve the semantic heterogeneity between local schemas, our approach is based on domain ontology and on similarity formulas. As for the structural heterogeneity of local schemas, it is solved by the extension of a query routing method (i.e. Chord protocol) with Structure Indexes. Concerning the query optimization problem, we propose to take advantage of the data source localization phase to obtain all metadata required for generating a close to optimal execution plan. Finally, in order to show the feasibility and the validity of our propositions, we carry out performance evaluations and we discuss the obtained results.

Keywords: P2P, Databases, Schema Heterogeneity, DHT, Ontology, Performance Evaluations.