Chapitre 2

Etat de l'art sur la composition des Services Web

Sommaire

2.1 Introduction

Ce chapitre est essentiellement consacré à l'état de l'art relatif à la composition des Services Web. Nous commençons tout d'abord par introduire le besoin de composition des Services Web dans la section 2.2 et nous présentons les approches existantes pour la composition des Services Web dans la section 2.3. Ensuite, nous introduisons le problème de réécriture de requêtes en termes de vues pour lequel, nous présentons trois algorithmes (section 2.4). Enfin, nous discutons notre travail par rapport aux autres travaux.

2.2 Composition des Services Web

Ces dernières années, la recherche dans le domaine des Services Web a été très développée. Une grande partie de cette recherche a été consacrée à la composition de Services Web. Réellement, il n'est pas toujours facile de trouver des Services Web qui s'apparient avec les requêtes des utilisateurs. Par conséquent, la composition des services satisfaisant la requête est un besoin grandissant de nos jours. La composition de services est une tâche complexe. Cette complexité est due principalement aux raisons suivantes [4] :

- Le nombre de Services Web disponibles sur le Web est potentiellement très important et en constante augmentation ;
- Les Services Web peuvent être créés et modifiés, le système de composition doit donc détecter et gérer ces changements ;
- Les Services Web sont créés par des organisations différentes qui n'ont pas forcément les mêmes modèles de concepts pour décrire ces services.

La composition des Services Web devient de plus en plus incontournable dans un environnement ouvert et dynamique comme Internet. Il est évident qu'une composition à la demande sera très préférable, en plus, il sera judicieux de prendre en charge la sémantique durant la composition des Services Web afin de minimiser les fausses réponses et d'améliorer la qualité globale des résultats.

La composition des Services Web est le processus de construction de nouveaux Services Web à valeur ajoutée à partir de deux ou plusieurs Services Web déjà présents et publiés sur le Web. Un Service Web est dit composé ou composite lorsque son exécution implique des interactions avec d'autres Services Web et des changements des messages entre eux afin de faire appel à leurs fonctionnalités. La composition de Services Web spécifie quels services ont besoin d'être invoqués, dans quel ordre et comment gérer les conditions d'interactions [5].

2.3 Approches existantes

Il existe principalement deux grandes approches de composition des Services Web : *l'approche statique* et *l'approche dynamique*. Nous présentons brièvement chacune de ces approches dans ce qui suit.

2.3.1 Composition statique des Services Web

Une composition est statique quand tous les Services Web qui font partie de la composition sont connus, ainsi que leur ordre d'exécution. Dans cette composition, les services sont connus avant leur exécution, c'est-à-dire, au moment où la composition est faite. De plus, nous connaissons également l'ordre d'exécution ainsi que la localisation et la description des Services Web. Dans la composition statique, tout est complètement spécifié et en particulier tous les Services Web sont connus. [7]

De plus, la création des processus métiers se fait durant le développement du système et reste statique pendant son utilisation, or l'environnement des Services Web est dynamique. Pour ces raisons, les chercheurs fournissent beaucoup d'efforts pour la création des processus métiers et le rendre plus dynamique en restreignant, autant que possible, l'intervention du développeur dans le choix et la composition des Services Web. Ce type de composition est la composition dynamique. **Microsoft Biztalk** et **Bea Weblogic** sont deux exemples de moteurs de composition statique de Services Web [18]. Si les fournisseurs de services proposent d'autres services ou changent les anciens services, des incohérences peuvent être causées, ce qui demanderait un changement de l'architecture du logiciel, voire de la définition du processus et créerait l'obligation de faire une nouvelle conception du système. Dans ce cas, la composition statique des Services Web est considérée trop restrictive : les composants doivent s'adapter automatiquement aux changements imprévisibles.

2.3.2 Composition dynamique des Services Web

Une composition est dynamique quand les Services Web qui font partie de la composition sont connus progressivement, ainsi que l'ordre d'exécution nécessaire pour la composition. Une approche dynamique pour la composition de services offre le potentiel de réaliser des applications flexibles et adaptables, en sélectionnant et en combinant les services de manière appropriée sur la base de la requête et du contexte de l'utilisateur [7].

Dans cette approche, les Services Web à composer sont choisis au moment de l'exécution durant laquelle une recherche des services est effectuée dans les registres. À titre d'exemple, **StarWSCOP** [8] [18]. Pour faire la composition dynamique des Services Web, StarWSCOP effectue les quatre étapes suivantes :

- Les fournisseurs des Services Web publient leurs services dans un registre.
- StarWSCOP décompose les requêtes des utilisateurs en services abstraits et envoie une requête SOAP au registre pour trouver les services appropriés.
- Le registre de Services Web disponibles renvoie une liste de Services Web concrets.
- StarWSCOP envoie une requête SOAP aux Services Web trouvés, ensuite se lie avec eux.

StarWSCOP contient plusieurs modules : un système intelligent pour décomposer les requêtes des utilisateurs en plusieurs services abstraits, un moteur de recherche de Services Web pour découvrir les Services Web qui respectent les conditions de l'utilisateur, un moteur de composition qui exécute les services en ordre. Une couche axée ontologie est aussi rajoutée à UDDI afin de faire de l'appariement sémantique pour les Services Web.

2.4 Réécriture de requêtes en termes de vues

Le problème de réécriture de requêtes a été principalement abordé dans les applications d'optimisation de requêtes et d'intégration de données. La principale question à laquelle la réécriture essaie de répondre est : Etant donné une requête Q est un ensemble de vues $V = \{V_1, ..., V_n\}$, est-il possible de répondre à Q en utilisant uniquement les données des

8. Star Web Services Composition Platform

vues V ?. Dans le contexte d'intégration de données, la réécriture d'une requête consiste à déterminer les vues pertinentes pour l'exécution de la requête d'utilisateur et à utiliser leurs définitions pour reformuler cette requête. Les techniques de réécriture de requêtes ont été largement explorées dans le domaine de base de données. Une bonne étude des algorithmes de réécriture de requête est présentée dans [2].

Dans ce qui suit, nous décrivons trois algorithmes de réécriture permettant de répondre aux requêtes en utilisant des vues. Ces algorithmes sont : ***"l'algorithme Bucket"***, ***"l'algorithme de règles inversées"*** et ***"l'algorithme MiniCon"***.

2.4.1 L'algorithme Bucket [2]

L'idée principale de l'algorithme Bucket est de réduire le nombre de réécritures qui doivent être étudiées en considérant chaque sous-but de la requête en isolation afin de déterminer les vues pertinentes pour chaque sous-but. L'algorithme procède en deux étapes : (i) construction d'un bucket pour chaque sous-but de la requête contenant les vues contributives pour la réécriture de ce sous-but , et (ii) construction de toutes les requêtes conjonctives possibles en prenant une source de chaque bucket. Le résultat final de l'algorithme est l'union de toutes les requêtes conjonctives (réécritures candidates).

Dans sa première phase, l'algorithme Bucket essaie de trouver l'ensemble des vues qui sont pertinentes pour chaque sous-but de la requête. Une vue V est pertinente pour un sous-but g d'une requête Q si les conditions suivantes sont vérifiées :

- V contient un sous-but g_1 tel que g peut être unifié avec g_1. L'unification est faite sur des sous-buts ayant le même nom en tenant compte de la position des variables. Elle consiste à construire les mappings entre les variables des deux sous-buts ;
- Toutes les variables distinguées de Q sont mappées à des variables distinguées de V lors de l'unification de g et g_1 ;
- Les prédicats de comparaison de V et de Q ne sont pas conflictuels.

Dans la deuxième phase, l'algorithme Bucket trouve un ensemble de réécritures de requêtes conjonctives. Il considère toutes les combinaisons possibles de sources de don-

nées en prenant une source de chaque bucket. Chacune de ces réécritures représente un moyen d'obtenir une partie de la réponse à Q à partir des vues. L'inconvénient majeur de l'algorithme Bucket est son incapacité de détecter le fait qu'il doit utiliser la même vue pour couvrir plusieurs sous-buts de la requête. Cet algorithme est également incapable de trouver les vues qui ne peuvent pas être pertinentes avant d'avoir testé toutes les solutions possibles.

2.4.2 L'algorithme de règles inversées [2]

Le fonctionnement de l'algorithme de règles inversées est basé sur la construction d'un ensemble de règles qui montrent comment construire les tuples du schéma global à partir des tuples des vues. Le principe de construction des règles inversées est le suivant :

- Pour chaque vue V_i définie par l'expression $V_i(\overline{X}) : - r_1(\overline{X}_1)$ ...., $r_n(\overline{X}_n)$., construire n règles telles que l'entête de la règle R_j j=1... n, est l'atome $r_j(\overline{X}_j)$ du corps de V_i et le corps de R_j est l'entête de la vue $V_i(V_i(\overline{X}))$.
- Remplacer toutes les variables existentielles de la définition de V_i qui se retrouvent dans les entêtes des règles inversées par des fonctions de Skolem. Utilisez la même fonction de Skolem pour toutes les occurrences de la même variable existentielle de V_i. Les fonctions de Skolem indiquent les variables dont les valeurs sont inconnues en dehors de la définition de la vue.
- L'ensemble des règles inversées est l'union des règles produites pour chaque vue.

L'avantage de l'approche de réécriture de requêtes à base de règles inversées est la construction des règles qui peut être fait en temps polynomial. Par contre, le calcul des résultats de la requête peut répéter des calculs qui ont été faits pour le calcul des vues. Un autre inconvénient de cette approche est que lors de calcul des tuples, l'algorithme va utiliser l'ensemble des règles inversées, même si certaines règles ne sont pas pertinentes à la réécriture de la requête i.e., il n'est pas possible d'obtenir des résultats pour la requête en utilisant ces règles.

2.4.3 L'algorithme MiniCon [2]

Comme l'algorithme Bucket, l'algorithme MiniCon fonctionne en deux étapes qui sont la recherche des vues pertinentes pour la réécriture de la requête et la combinaison de ces vues afin d'obtenir les réécritures de la requête. Dans la première phase, il cherche des correspondances entre les sous-buts de la requête et ceux des vues. A la différence de l'algorithme Bucket qui construit des tas indépendants pour chaque sous-but, lorsque l'algorithme MiniCon détecte une correspondance entre un sous-but g de la requête Q et un sous-but g_1 d'une vue V, il examine les variables de jointure de la requête afin de déterminer l'ensemble minimal de sous-buts de V qui doivent être unifiés avec des sous-buts de Q étant donné que g sera unifié avec g_1. Pour chaque vue, l'algorithme note les sous-buts de la requête qui peuvent être couverts par cette vue. L'ensemble des informations décrivant les mappings entre une vue V et une requête Q sont représentés par un quadruplet $(h, V(\overline{Y}), \Phi, G)$ appelé MiniCon Descriptor (MCD) où :

- h est un homomorphisme défini sur les variables de l'entête de V. Il exprime le fait que parfois il peut être nécessaire d'unifier des variables de l'entête de la vue.
- $V(\overline{Y})$ est le résultat de l'application de h sur l'entête de V.
- Φ est le mapping (unification) partiel des variables de Q vers V.
- G est l'ensemble de sous-buts de Q qui sont couverts par la vue V.

Une fois les MCDs construits, l'algorithme les combine afin d'obtenir des réécritures de la requête. La combinaison de MCDs est faite en suivant le principe que l'ensemble des MCDs formant une réécriture candidate de la requête doivent couvrir l'ensemble des sous-buts de la requête et que chaque paire de MCDs doivent couvrir des sous-buts disjoints i.e. : soient $C_1, C_2, \ldots, C_k$ un ensemble de MCDs formant une réécriture candidate de Q. Alors, les conditions suivantes doivent être vérifiées :

- $G_1 \cup \cdots \cup G_k = \text{sous-buts } (Q)$ et
- $\forall i, j, i \neq j, G_i \cap G_j = \emptyset$

Le principal avantage de l'algorithme MiniCon est sa capacité de filtrer les vues non pertinentes pendant la phase de construction des MCDs, les MCDs construits peuvent être combinés sans vérifier la satisfaisabilité des prédicats de jointure.

2.5 Discussion

Nous avons présenté les travaux concernant deux manières de réifier l'intégration de données :

- En se basant sur des techniques de réécriture des requêtes en termes de vues ;
- En se basant sur des plates-formes permettant une composition des données accessibles à distance de manière statique ou dynamique.

Pour la première, ces techniques de réécriture ont été appliquées sur les premiers systèmes de médiation réalisés le plus souvent pour interroger de manière uniforme des bases de données distribuées.

Pour la seconde manière d'intégrer l'information, beaucoup des plates-formes utilisées sont industrielles et se basent sur des Services Web DaaS. Nous donnons à titre d'exemple les produits mis au point pour faire la création de Services Web DaaS : **AquaLogic Data Services Platform (ALDSP)** par BEA Systems [8], **Astoria** par Microsoft [10], la plateforme **MetaMatrix** par RedHat [16], **Composite Data Virtualization Platform** par Composite Software [9] et la plateforme **XIC (Xcalia Intermediation Core)** par Xcalia [24].

Le système que nous allons proposé et présenté dans la suite, s'inscrit dans le cadre de travaux issus de ces deux manières d'intégrer l'information. Notamment, notre cadre permet de combiner des techniques de réécriture de requêtes en termes de vues représentées sous forme de Services Web DaaS. Ces deux classes d'intégration ont été considérées le plus souvent de façon isolée et très peu de travaux les ont appliquées dans un même cadre de médiation. Nous pouvons cités les travaux dans les deux thématiques, les travaux de Mahmoud Barhamgi [6]. Cependant, son algorithme de réécriture ne fait aucune référence aux algorithmes classiques de réécriture de requêtes contrairement à notre travail. En outre, notre approche traite sans ambiguïté tous les cas possibles contrairement à

l'approche de Barhamgi [6] qui laisse dans un certains nombre de cas certaines questions sans réponses.

Dans la suite, nous allons présenter notre approche de composition de Services Web DaaS en se basant sur des techniques de réécriture de requêtes.