관계형 데이타베이스와 온톨로지를 이용한 웹 서비스 컴포지션 및 검색

박규호° 권준호 이대욱 이석호
서울대학교 전기·컴퓨터공학부
{parkqho°, bluerain, dwlee}@db.snu.ac.kr, shlee@snu.ac.kr

Web Service Composition and Search Using Relational Database and Ontology

Kyuho Park°, Joonho Kwon, Daewook Lee, Sukho Lee
School of Electrical Engineering and Computer Science, Seoul National University

요 약

웹 기술이 발전하면서 웹 서비스에 대한 정보를 UDDI에 저장, 검색하는 기법이 등장하였다. 이런 웹 서비스 정보 저장, 검색 기법들은 주로 단일 웹 서비스만을 대상으로 하였으나, 최근에는 단일 웹 서비스 뿐만이 아니라 웹 서비스들의 컴포지션 역시 저장하고 검색하는 연구가 진행되기 시작하였다. 웹 서비스 컴포지션은 한 웹 서비스의 출력과 다른 웹 서비스의 입력이 같을 경우 연결시켜 하나의 웹 서비스처럼 보이게 하는 것이다. 웹 서비스들의 컴포지션만으로도 단일 웹 서비스만을 저장, 검색하는 것보다는 유용하다. 그러나 그 알고리즘이 복잡하고, 여전히 사용자의 만족도는 낮다. 본 논문에서는 관계형 데이타베이스와 온톨로지를 이용하여 쉽고 간단한 알고리즘으로 웹 서비스 컴포지션과 검색을 하여 사용자 만족도를 높이는 기법을 제안한다.

1. 서 론

근래에 네트워크와 인터넷에 대한 수요가 증대되고 관련 기술이 발전하면서 웹에 관한 관심 역시 높아졌다. 웹과 관련하여 다양한 기술들이 나오게 되었는데 그 중 e-비즈니스와 웹 서비스 같은 전자상거래 기술이 주목을 받고 있다. 이 기술들은 웹 상에 존재하여 사용자가 원하는 기능을 수행하도록 한다. 특히 웹 서비스는 웹 서비스 제공자에 의해 UDDI(Universal Description, Discovery and Integration) 레지스트리에 관련 정보가 등록, 저장되고, 웹 서비스 요청자(사용자)가 검색을 요청하면 해당 웹 서비스의 정보가 검색된다. 또한 제공자와 요청자 간에는 SOAP(Simple Object Access Protocol) 프로토콜로 연결된다. 그림 1은 웹 서비스 등록, 검색, 바인딩 구조를 나타내고 있다[1].

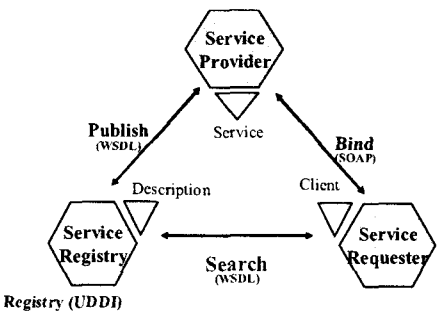

그림 1. 웹 서비스 등록, 검색, 바인딩 구조

여러 웹 서비스 관련 논문에서는 웹 서비스를 정확하고 효율적으로 저장하고 검색하는 방법을 연구하였다[2][3][4]. 웹 서비스 검색 방법에는 두 가지가 있다. 하나는 키워드 기반 검색(keywords search)이고 다른 하나는 온톨로지 기반 검색(ontology search)이다[3][4].

키워드 검색은 웹 서비스에 있는 키워드로 웹 서비스를 찾는 것이다. 따라서 키워드에 정확히 일치되는 웹 서비스가 있을 때만 해당 웹 서비스가 반환된다. 키워드 기반 검색 중 템플릿 검색(template search)이 있는데, 요청자가 웹 서비스를 검색할 때 찾고자 하는 웹 서비스의 입력과 출력을 명세하여 찾는 검색 방법이다. 템플릿 검색을 할 때 웹 서비스를 컴포지션하여 검색을 할 수 있다. 이 컴포지션 검색은 한 웹 서비스의 출력과 다른 웹 서비스의 입력이 같은 경우 전자의 출력과 후자의 입력 사이에 링크를 만듦으로써 그 두 웹 서비스를 연결하여 마치 하나의 웹 서비스처럼 보이게 하는 웹 서비스 컴포지션을 찾는 것이다. 연결될 수 있는 웹 서비스는 두 개 이상이다.

온톨로지 기반 검색은 웹 서비스 검색 시 시맨틱(semantic) 웹 기술인 온톨로지를 이용하여 비슷한 의미를 지닌 웹 서비스도 검색하는 기법이다. 온톨로지는 일반적으로 공유된 개념을 계층구조로 분류한 명세로서 추론 규칙도 포함되어 있다.

본 논문에서는 웹 서비스 컴포지션 및 검색에 관계형 데이타베이스(Relational Database)를 이용하여 기존 연구보다 더 간단해진 알고리즘을 기본적으로 제안한다. 또한 온톨로지(ontology)를 사용함으로써 웹 서비스 컴포지션에 의미를 부여하여 검색 범위를 넓히는 기법도 제안한다.

본 논문의 구성은 다음과 같다. 2장은 관련 연구 및 연구 동기로 웹 서비스에 대한 기존 연구들을 살펴본 후 관계형 데이타베이스와 온톨로지를 이용한 컴포지션 및 검색이 필요한 이유와 이점을 예로 설명한다. 구체적인 컴포지션 및 검색 알고리즘과 기법들은 3장에서 제시하

고 4장에서 결론을 맺는다.

2. 관련 연구 및 연구 동기

2.1 WSDL 화일의 구조

웹 서비스는 WSDL(Web Service Description Language)로 기술되며, 해당 기능을 수행하는 연산(operation)과 입력(input), 출력(output)으로 구성된다. WSDL 화일은 XML 문서 형태로 되어 있으며 XML 문법을 따른다. WSDL 화일의 예는 그림 2와 같다.

```
<?xml version="1.0" encoding="utf-8" ?>
 <wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org
  /wsdl/soap/" xmlns:tm=.../wsdl/">
...
 <wsdl:types>
...
 <wsdl:operation name="CityToZipCode">
  <wsdl:documentation  xmlns:wsdl="http://schemas.xmlsoap.org/
   wsdl/">Returns a list of zip codes for a supplied city
</wsdl:documentation>
  <wsdl:input message="City" />
  <wsdl:output message="ZipCode" />
  </wsdl:operation>
...
```

그림 2. WSDL 화일의 예

이 예에서 웹 서비스의 연산인 CityToZipCode, 입력인 City, 출력인 ZipCode는 UDDI 레지스트리 안에서 키워드 형태로 저장된다. 사용자는 검색 질의를 할 때 입력과 출력을 명세할 수 있다.

2.2 웹 서비스 컴포지션의 이점

웹 서비스 검색 기법 중 가장 기본적인 키워드 검색은 이미 오래 전부터 연구되고 많은 발전이 있었다. UDDI에서도 웹 서비스 정보들이 키워드 형태로 저장되기 때문에 사용자가 명세한 입력과 출력으로 검색하는 템플릿 검색도 키워드를 기반으로 수행된다. 하지만 정확히 일치하는 키워드가 아니면 검색이 되지 않는 한계점이 있다.

이를 보완하기 위해 여러 연구에서 키워드 사이에 같이 발생하는 빈도에 따라 클러스터링하여 검색하는 기법[2]과, 키워드에 의미를 부여하는 시맨틱 웹 관련 기술인 온톨로지를 이용한 검색[3][4] 기법이 등장하여 질의 키워드와 유사한 키워드 검색 및 매칭에 사용되었다.

그러나 이 논문들은 주로 사용자 질의를 만족하는 결과를 웹 서비스 하나 단위로 검색해주는 기법을 연구했다. 즉, 웹 서비스를 하나의 레코드처럼 보고 그 웹 서비스 레코드 하나를 찾는 기법에만 초점이 맞추어져 있었다. 또는 두 웹 서비스들 간의 매칭 문제만을 주로 다루었다. 웹 서비스의 컴포지션과 검색에 대해서는 간략히 설명만 하고 향후 연구에만 포함시킬 뿐 구체적인 설명이나 방향을 설정하지 않았다. 하지만 웹 서비스 컴포지션을 하지 않는다면 다음과 같은 문제가 있다.

웹 서비스 컴포지션을 지원하지 않는 검색 시스템에서 그림 2의 웹 서비스를 포함한 그림 3처럼 여러 웹 서비스들의 정보가 UDDI에 등록, 저장되어 있다고 하자. 웹 서비스 정보는 웹 서비스 아이디(ID), 연산, 입력, 출력으로 구성된다. 예를 들어 아이디가 W1이라는 웹 서비스는 WeatherForCity라는 연산을 가지고 있으며, 그 연산은 City를 입력으로 받고, Weather를 출력으로 내보낸다.

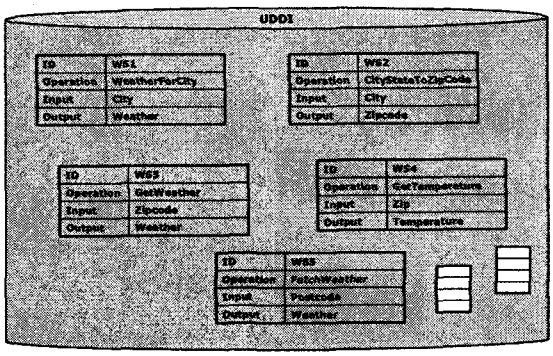

그림 3. UDDI에 등록된 웹 서비스 정보들의 예

여기서 사용자가 그림 4처럼 '입력이 도시(City)이고 출력이 날씨(Weather)인 웹 서비스를 찾아라' 라는 질의로 검색 요청을 했다고 하자.

User Query(Q) : Input = "City", Output = "Weather"

그림 4. 사용자 질의(Q)의 예

그림 3에서 웹 서비스의 컴포지션 없이 템플릿 검색만 한다면 그림 4의 질의에 맞는 웹 서비스는 W1이다. 하지만 만일 어떤 웹 서비스 등록자도 W1을 등록하지 않았다고 하면 어떻게 될 것인가. 웹 서비스 컴포지션이 지원되지 않는 시스템에서라면 UDDI내에 이 W1이 없을 경우 사용자는 원하는 웹 서비스를 찾을 수 없게 된다[2][3]. 이런 경우는 실생활에서 충분히 흔하게 일어날 수 있다.

웹 서비스를 컴포지션한다면 사용자가 원하는 웹 서비스 자체가 UDDI에 등록이 안 되어 있을 경우라도 웹 서비스 컴포지션을 통해 사용자 질의를 만족시키게 할 수 있다[5]. 예를 들어 사용자가 원하는 웹 서비스 W1가 없을 경우에도 W2의 출력인 Zipcode와 W3의 입력인 Zipcode가 같기 때문에 W2와 W3를 연결할 수 있다. 컴포지션을 리스트(list) 자료구조로 표현한다고 하면, 컴포지션 (W2, W3) 전체의 입력과 출력은 사용자 질의 Q의 입력, 출력과 일치하기 때문에 사용자는 찾아진 컴포지션 (W2, W3)를 이용하면 된다. 이렇게 되면 검색 대상에 개개의 웹 서비스뿐만이 아니라 웹 서비스 컴포지션도 포함되어 사용자에게 검색 결과 선택 폭이 넓어진다.

2.3 연구 동기

2.3.1 기존 웹 서비스 컴포지션 연구의 보완

[2], [3], [4] 이후에 웹 서비스 컴포지션에 관련된

연구들이 많이 진행되었는데, [6]은 웹 서비스 컴포지션 검색을 목적으로 한 것이 아니라 웹 서비스들이 실행되어 나온 출력 값(value)을 워크플로우를 따라 넘겨줄 때 실제 웹 서비스 응용 서버 간에 메시징 기법을 사용하는 실시간 전달 문제 해결법을 제시했다.

[5]는 웹 서비스들을 컴포지션하려는 이유와 목적은 컴포지션 검색을 위한 것으로 본 논문과 같다. [5]는 컴포지션 검색 시 결과들의 순위를 낼 때 컴포지션에 참여하는 웹 서비스들이 참조된 빈도수를 기준으로 점수를 차등 배분하여 순위를 정하는 PR(PageRank) 기법을 사용한 것이 특징이다. 또한 웹 서비스들을 미리 연결해 놓고 사용자의 질의 요청이 오면 웹 서비스 컴포지션을 검색하는데, 우선 질의의 입력에 맞는 웹 서비스를 찾아 그에 연결된 웹 서비스들을 깊이 우선 탐색으로 질의의 출력을 가진 웹 서비스를 만날 때까지 UDDI 내에서 찾는다.

하지만 웹 서비스 컴포지션 방법이 연결 가능한 웹 서비스들 사이에 인접 행렬을 사용하여 구현되었기 때문에 복잡하고 반드시 필요하지 않은 저장 공간이 많이 필요하다.

본 논문에서는 이 단점을 보완하기 위해 웹 서비스 컴포지션을 수행하고 검색하는 데에 관계형 데이타베이스를 사용하는 것을 제안한다. 관계형 데이타베이스를 사용하여 테이블 형태로 웹 서비스의 정보들을 저장하면 컴포지션 알고리즘과 검색 알고리즘이 간단해진다. 더욱이 웹 서비스들의 양이 방대해진다 해도 관계형 데이타베이스를 이용하기 때문에 확장가능하다는(scalable) 장점도 있다.

그리고 [5]에서처럼 컴포지션 검색 시 깊이 우선 탐색을 하면 사용자가 원하는 출력을 가진 목표 웹 서비스를 찾을 때까지 가장 최적인 경로를 이동한다는 보장이 없을 뿐만 아니라 가능한 웹 서비스 컴포지션들이 여러 개라도 그 중 일부 컴포지션들만을 찾을 수도 있다는 단점이 있다.

따라서 깊이 우선 탐색보다는 최단 경로 알고리즘을 사용하는 것이 보다 정확하고 효율적이다. 왜냐하면 사용자 질의의 입력과 출력을 만족시키는 두 웹 서비스들의 쌍을 지정하고 최단 경로를 찾으면 최적의 웹 서비스 컴포지션을 포함한 모든 컴포지션을 찾을 수 있기 때문이다. 본 논문에서는 다익스트라(Dijkstra)의 최단 경로 알고리즘을 사용하여 컴포지션을 검색한다. 또한 2.3.2절과 3.3.1절, 3.3.2절에서 설명할 온톨로지를 사용한 컴포지션 및 검색에서도 최단 경로 알고리즘은 반드시 필요하다.

2.3.2 온톨로지를 이용한 유사 검색 지원

[5]에서는 웹 서비스 컴포지션까지 지원하긴 하지만 키워드가 정확히 일치하지 않는다면 컴포지션 및 검색이 가능한 범위는 매우 좁아지게 되어 있다. 예를 들어 앞의 예에서 W1이 없는 상황에서 W2는 있지만 W3마저 없다면, 사용자 질의에 맞는 웹 서비스 또는 웹 서비스 컴포지션은 어떤 것도 없게 된다.

이런 문제를 해결하기 위해 본 논문에서는 이 컴포지션에 의미를 부여하여, 유사한 컴포지션까지 찾아낼 수 있도록 시맨틱 웹 기술인 온톨로지를 이용하려고 한다. 그림 5와 같이 개념적으로 계층구조를 이루고 분류가 되어 있는 온톨로지를 이용한다. 이 온톨로지에서는 최상위 클래스에 Thing이라는 일반적인 개념(concept)이 위치하며 그 하위에는 Zip과 Zipcode, Weather 등의 서브 클래스들이 존재한다.

단, 지금까지 연구된 바로는 온톨로지를 자동적으로 구축하는 기법이 없으므로 수동적(manual)이거나 반자동적(semi-automatic)으로 생성된 이미 구축되어 있는 온톨로지를 사용한다고 가정한다. 온톨로지를 자동적으로 생성하는 기법은 본 논문의 내용을 벗어나므로 다루지 않는다.

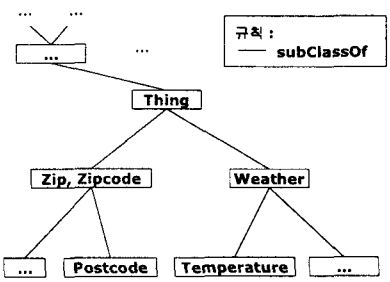

그림 5. 미리 구축된 온톨로지의 예

온톨로지를 이용하면 키워드는 다르지만 온톨로지 상에서 같은 클래스나 자식(child) 클래스에 있는 비슷한 개념들끼리 자동적으로 매칭을 시킬 수 있다.

이런 온톨로지를 이용한 매칭에는 매칭의 정확도에 따라 exact, plug in, subsumes, fail이 있다[4].

exact는 키워드와 키워드가 온톨로지에서 같은 클래스에 있는 경우를 뜻한다. plug in과 subsumes는 기준이 되는 키워드와 비교가 되는 키워드가 순서에 따라 각각 포함되는 경우와 포함시키는 경우를 뜻한다. 웹 서비스 컴포지션에서는 한 서비스의 출력이 다른 서비스의 입력에 포함되어야 정보 손실이 없으므로 기준 키워드는 출력 키워드가 된다. fail은 클래스의 계층 관계가 없거나 전혀 다르거나 2세대 이상 떨어진 클래스에 키워드들이 있을 때이다.

그림 3의 예에서는 W4와 W5의 입력인 Zip과 Postcode가 W2의 출력인 Zipcode랑 온톨로지 상에서 각각 같은 클래스와 자식 클래스에 있기 때문에 exact-매칭, subsumes-매칭 관계가 성립된다. 그러면 그림 6과 같이 W2, W3 사이뿐만 아니라 W2, W4와 W2, W5 사이에도 링크를 만들 수 있다.

이렇게 연결된 컴포지션 중 (W2, W5)의 입력과 출력은 사용자가 명세한 질의의 입력과 출력에 일치한다. 그리고 (W2, W4) 컴포지션의 입력과 출력도 사용자의 입력과 출력에 상당히 유사하다고 할 수 있다. 온톨로지 상에서 W4의 출력인 Temperature가 사용자 질의의 출력인 Weather의 자식 클래스에 있으므로 비슷한 개념을 나타내는 plug in-매칭 관계가 성립되기 때문이다.

따라서 그림 6과 같이 (W2, W3) 컴포지션뿐만 아니라

(W2, W4)와 (W2, W5)도 질의에 유사하므로 질의를 만족시키는 결과로 반환할 수 있다. 그렇게 되면 사용자가 원하는 웹 서비스가 전혀 없어서 웹 서비스를 이용하지 못할 확률은 현격히 줄어들 것이다. 그리고 사용자에게 더 많은 웹 서비스를 반환함으로써 더 넓은 선택폭을 제공할 수 있다. 특히, 유사도나 인기도, 또는 검색 빈도에 따라 랭킹을 매김으로써 사용자 지향적인(user-oriented) 웹 서비스 검색을 지원하게 된다.

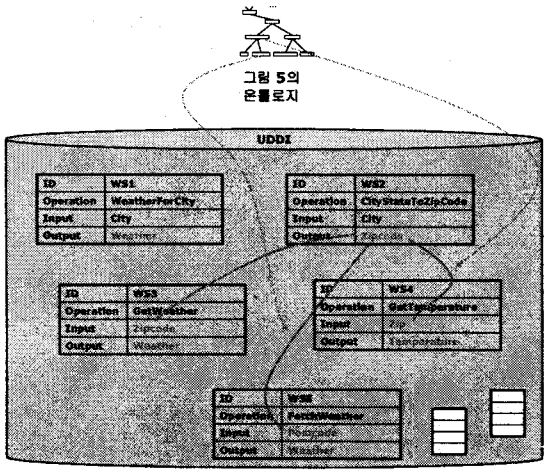

그림 6. 온톨로지를 이용한 웹 서비스 컴포지션의 예

즉, 온톨로지를 이용한 웹 서비스 컴포지션의 이점은 두 가지로 요약할 수 있다. 첫 번째, 사용자가 명세한 질의를 만족하는 웹 서비스가 UDDI에 등록되어 있지 않아도 등록된 다른 유사 웹 서비스들을 자동적으로 컴포지션함으로써 사용자 검색 질의를 만족시킬 확률을 높일 수 있다. 두 번째, 사용자에게 다양한 결과 선택을 제공하여 사용자 지향적인 검색을 할 수 있다.

3. 웹 서비스 컴포지션 및 검색

이 장에서는 웹 서비스들을 어떻게 정확하고 효율적으로 컴포지션하여 사용자에게 반환할 것인지에 대한 알고리즘과 관련 기법을 제안한다.

3.1 문제 정의

본 논문에서 해결하고자 하는 문제는 두 가지이다. 첫 번째는 UDDI 레지스트리에 있는 여러 연결 가능한 웹 서비스들 사이에 링크를 만들어 컴포지션을 수행하는 문제이다. 두 번째는 사용자가 입력과 출력을 명세한 질의를 던졌을 때 만족되는 질의 컴포지션 결과들을 검색하는 문제이다.

첫 번째 문제는 정의 1과 같이 웹 서비스 하나를 하나의 정점으로 보고, 연결될 수 있는 링크들은 간선들로 보아 방향 그래프를 만들어내는 문제로 치환할 수 있다. 컴포지션 수행에서 ≈ 기호는 온톨로지 상에서 유사한 정도인 plug in이나 subsumes를 의미한다.

또한 두 번째 문제 역시 정의 1에서와 같이 만들어진 방향 그래프 내에서 입력 웹 서비스 정점과 출력 웹 서비스 정점이 주어졌을 때 두 정점 간 모든 경로를 찾는 문제로 바꾸어 기술할 수 있다.

웹 서비스 집합을 나타내는 아래와 같은 그래프가 주어졌을 때
$WS = \{ (I_1, O_1), (I_2, O_2), \cdots, (I_n, O_n) \}$
컴포지션 수행
For all $j, k, (1 \le j, k \le n)$
If $(O_j = I_k \,||\, O_j \approx I_k)$
Then create an edge $O_j \rightarrow I_k$
컴포지션 검색
When I_u and O_u are user's input and output respectively $(1 \le u \le n)$, search the list of paths $(I_u \rightarrow O_u)$.
If the list of paths $(I_u \rightarrow O_u)$ exists, return it.
Else, return null(or message of no path)

정의 1. 웹 서비스 컴포지션 및 검색의 그래프 표현

컴포지션 수행과 검색에 대한 자세한 내용은 3.3절에서 설명한다.

3.2 시스템 구조

웹 서비스 컴포지션 및 검색 시스템의 구조는 그림 7과 같다.

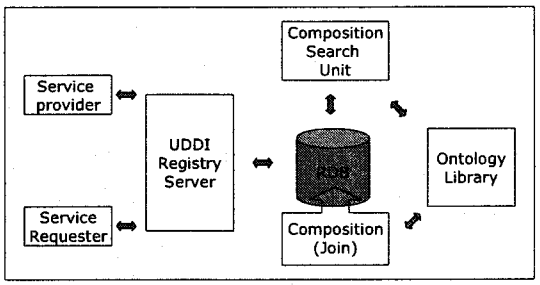

그림 7. 온톨로지를 이용한
웹 서비스 컴포지션 및 검색 시스템 구조

서비스 제공자는 UDDI 레지스트리 서버에 웹 서비스를 등록한다. 시스템은 UDDI에서 웹 서비스의 각 연산과 그 입·출력을 추출하여 관계형 데이타베이스에 테이블로 저장한다. 컴포지션은 관계형 데이타베이스 내에서 이 테이블을 이용해 조인 연산을 함으로써 자동적으로 수행되는데, 이때 온톨로지가 사용된다. 이 작업들은 모두 사용자 질의가 들어오기 전에 미리 수행된다.

서비스 요청자가 UDDI 레지스트리 서버에 웹 서비스를 질의하면, 컴포지션 검색 유닛은 조인 연산의 결과 테이블을 이용해 사용자 질의에 맞는 최적 컴포지션을 포함한 모든 컴포지션을 온톨로지를 사용하여 검색하고 반환한다.

3.3 온톨로지를 이용한 웹 서비스 컴포지션 및 검색

3.3.1 온톨로지를 이용한 웹 서비스 컴포지션

웹 서비스들의 컴포지션을 수행하는 데 있어 먼저 가정해야할 것이 있다. 연결된 방향 그래프를 간단히 표현

하기 위해서, 하나의 웹 서비스는 연산, 입력, 출력을 각각 하나씩 갖는 것으로 가정한다.

컴포지션의 과정은 다음과 같다. 서로의 출력과 입력이 연결될 수 있는 각 웹 서비스들 간에 가능한 모든 방향 간선들을 미리 연결시켜 정답 공간을 만든다. 이 때 관계형 데이타베이스를 사용하여 보다 쉽고 간단하게 컴포지션을 수행한다. 웹 서비스의 입력과 출력을 추출해 내어 그림 8과 같이 관계형 데이타베이스에 테이블 형태로 저장한다. 웹 서비스의 연산 이름은 레코드의 유일성을 보장하지 않기 때문에 아이디를 키로 지정한다.

Table : WS

ID	Operation	Input	Output
WS1	WeatherInfo	City	Weather
WS2	CityStateToZipCode	City	Zipcode
WS3	GetWeather	Zipcode	Weather
WS4	GetTemperature	Zip	Temperature
WS5	FetchWeather	Postcode	Weather
...			

그림 8. 관계형 데이타베이스에 저장한 웹 서비스 테이블 WS

이렇게 관계형 데이타베이스에 저장한 후 WS 테이블에 셀프 조인(self-join) 연산을 수행하면 연결 가능한 모든 간선들이 생성된다. SQL 표현은 다음과 같다.

```
Select t1.ID, t2.ID, t1.Input, t2.Output
From WS AS t1, WS AS t2
Where t1.Output = t2.Input and t1.ID ≠ t2.ID
```

하지만 이렇게만 할 경우 한 웹 서비스의 출력과 다른 웹 서비스의 입력의 이름이 정확히 일치하는 것만 연결되어 충분한 검색 결과가 나오지 않을 수도 있다는 단점이 있다. 온톨로지를 이용하여 클래스에 대한 매칭을 통해 유사한 입력과 출력 역시 연결할 수 있다. 예를 들어 둘 다 지역번호를 의미하지만 서로 정확히 일치되는 키워드가 아니기 때문에 서로 매칭이 되지 않았던 'zip'과 'zipcode'는 온톨로지의 같은 개념(concept)으로 묶여 연결될 수 있다. 이렇게 함으로써 사용자는 더 많은 질의 결과를 얻을 가능성을 갖게 된다.

온톨로지를 이용하여 셀프 조인 연산을 한 결과 테이블은 그림 9와 같이 나오는데 마지막 열에 온톨로지 매칭 정도를 추가한다. 예를 들어 그림 9의 두 번째 투플을 보면 W2의 출력인 Zipcode와 W4의 입력인 Zip이 온톨로지 상에서 같은 클래스에 속하므로 exact-매칭 관계가 성립되므로 마지막 열인 매칭 정도에 exact라는 값을 준다. 이 테이블은 실체화 뷰(materialized view)와 같이 하드디스크에 저장되어 검색 질의가 들어올 때마다 접근할 수 있도록 한다.

이 결과 테이블은 모든 웹 서비스 정점들 간에 가능한 모든 간선들이 연결되었다는 것을 의미한다. 컴포지션 가능한 모든 (t1.ID, t2.ID)이 생성되었기 때문이다. 그림 3의 예에서는 (W2, W3), (W2, W4), (W2, W5) 컴포지션들이 생성되었다.

Result Table : RT

t1.ID	t2.ID	t1.Input	t2.Output	Degree of Matching
WS2	WS3	City	Weather	exact
WS2	WS4	City	Temperature	exact
WS2	WS5	City	Weather	subsumes
...				

그림 9. WS 테이블의 셀프 조인 연산 결과 테이블 RT

앞서 2.3.2절에서 온톨로지를 이용한 매칭에는 매칭의 정확도에 따라 exact, plug in, subsumes, fail이 있다고 했다. 이에 따라 그림 9의 셀프 조인 연산 결과 테이블 RT를 이용하여 그래프의 간선들에 가중치 레이블링(labeling)을 할 수 있다. 예를 들어 각 간선마다 온톨로지 매칭이 exact인 경우에는 가중치 0, plug in인 경우에는 가중치 1, subsumes인 경우에는 가중치 2를 부여한다. 그리고 fail인 경우는 가중치 ∞을 부여하는 것이 아니라 아예 정점들 사이의 간선을 생성하지 않으면, 그림 10처럼 가중치 방향 그래프가 된다. 따라서 최단 경로 알고리즘을 이용할 수 있고, 유사도를 기준으로 최소 가중치를 갖는 경로를 찾으면 최대 유사도를 갖는 웹 서비스 컴포지션을 찾을 수 있다.

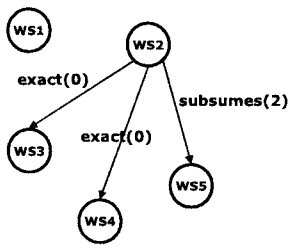

그림 10. 유사도로 레이블링한 가중치 방향 그래프

3.3.2 온톨로지를 이용한 웹 서비스 컴포지션 검색

이제 웹 서비스를 나타내는 각 정점 사이에 연결 가능한 가중치를 가진 간선들이 모두 생성되었다. 이 상태에서 사용자가 입력과 출력을 명세한 어떤 웹 서비스 컴포지션을 검색하라고 질의를 던지면, 시스템은 시작 웹 서비스 정점에서 끝 웹 서비스 정점에 이르는 모든 경로를 찾아 반환해준다. 유사도를 기준으로 검색 결과를 사용자에게 돌려줄 경우, 앞서 말한 다익스트라의 최단 경로 알고리즘을 사용하여 가중치가 가장 큰 경로부터 사용자 질의에 맞는 웹 서비스 컴포지션들을 모두 검색, 반환한다.

다익스트라의 최단 경로 알고리즘에서는 매번 새로운 최단 경로를 찾는다. 즉 최단 경로는 매번 알고리즘의 순환문 안에서 갱신되기 때문에 그 전까지 찾았던 경로들을 따로 저장하여야 한다. 이 경로들은 모두 사용자 질의에 맞는 웹 서비스 컴포지션들이 되므로 이 컴포지션을 가중치에 따라 정렬 후 순위를 매겨 사용자에게 반환한다. 가중치가 높을수록 정확도가 높은 웹 서비스 컴포지션이며 정확도에 따라 사용자가 선택할 수 있다. 이

렇게 검색 결과 역시 랭킹을 매김으로써 사용자 지향적인 시스템을 만들 수 있다.

혹은 정확도보다는 웹 서비스 기능의 예상 수행 시간과 같은 효율성을 더 중시할 경우 역시 최단 경로 알고리즘을 이용하여 간선마다 가중치를 예상 수행 시간으로 부여해 가중치 합이 작은 순서로 경로들을 랭킹을 매겨 반환할 수도 있다.

최단 경로 알고리즘을 사용하면 순환(cycle)도 방지할 수 있기 때문에 잘못된 결과가 반환되는 것을 막는다는 장점도 있다.

최단 경로 알고리즘에서 관계형 데이타베이스를 사용하면 다음과 같이 두 가지 이점이 있다.

원래 최단 경로 알고리즘에서는 순환문에서 새로운 최단 경로 계산를 계산할 때 가중치 인접 행렬을 사용한다. 가중치 인접 행렬을 사용하면 매 순환문 내에서 가중치 인접 행렬의 원소 값이 ∞인지를 확인하여야 한다. 간선이 별로 없는 그래프라면 사간적인 낭비가 크다. 하지만 본 논문에서는 가중치 인접 행렬이 아닌 셀프 조인 결과 테이블을 사용하므로 테이블 스캔을 통해 간선이 있는 경우만 순환을 하도록 개선된 알고리즘을 사용하기 때문에 불필요한 순환을 하지 않게 한다.

또한 가중치 인접 행렬은 연결되지 않은 정점들 간의 정보도 ∞와 같이 중복되는 값으로 채워야 한다. 하지만 셀프 조인 결과 테이블은 서로 인접하는 정점들에 대한 정보들만 모두 갖고 있기 때문에 일반적인 경우 가중치 인접 행렬보다 불필요한 저장 공간이 훨씬 줄어든다.

3.3.3 웹 서비스 변경(update)의 자동적인 처리

UDDI 레지스트리에 변경이 있을 경우, 즉 웹 서비스가 삽입(insert)됐거나 삭제(delete)됐거나 내용이 바뀌었을 때(입력, 출력이 바뀌었을 때)는 앞에서 이미 만들어 놓은 셀프 조인 결과 테이블을 사용할 경우 데이타의 일관성이 없어진다. 그러므로 UDDI 레지스트리에 변경이 있으면 실시간으로 새로운 셀프 조인 결과 테이블을 만들어야 한다. 이럴 경우를 대비해 이전의 셀프 조인 결과 테이블의 복사본을 유지하고 셀프 조인이 끝난 후에 그 복사본과 새 셀프 조인 결과 테이블을 교환하는 그림자 페이징과 같은 기법을 사용한다.

서비스 등록자 및 UDDI 관리자가 UDDI 레지스트리의 내용을 서버 사이트 접속을 통해 변경하면, 이를 이벤트로 처리하여 매 이벤트 발생 시 삽입, 삭제, 내용 변경이 자동적으로 이루어지게 한다. 웹 서비스 벌크 생성(bulk construction) 역시 계속해서 들어오는 웹 서비스들을 데이타 스트림 기법으로 처리하여 자동화할 수 있다.

4. 결론 및 향후 연구

본 논문에서는 관계형 데이타베이스와 온톨로지를 이용한 웹 서비스 컴포지션 및 컴포지션 검색 기법을 제안하였다. 관계형 데이타베이스를 이용하여 간단하면서도 효율적인 컴포지션 및 검색 방법을 제안하였다. 또한 온톨로지를 이용해 웹 서비스 컴포지션에 의미를 부여하여 유사한 웹 서비스 컴포지션도 검색 가능하게 하였다. 이로써 사용자가 질의에 대한 응답을 받지 못할 확률을 줄였으며, 결과의 선택 범위를 넓혀 사용자 지향적인 웹 서비스 컴포지션 검색을 가능하게 하였다.

추후 연구 과제로 다중 연산, 다중 입·출력을 가진 웹 서비스들에 대한 처리 방법과 저장된 테이블들에 대한 인덱싱 방법, 웹 서비스를 수행하는 서버들을 경로에 따라 거치는 데 걸리는 시간 제공 방법 등이 있다. 또 웹 서비스 컴포지션 수행 시 온톨로지 뿐만 아니라 입력과 출력 간의 도메인 혹은 타입 매칭을 이용하는 방법도 연구할 필요가 있다.

5. 참고 문헌

[1] 김미혜, 이규철, "웹 서비스 등록/검색 클라이언트의 개발", 충남대학교 대학원 컴퓨터공학 석사 학위논문, 2003

[2] Xin Dong, Alon Halevy, Jayant Madhavan, Ema Nemes and Jun Zhang, "Similarity Search for Web Services", VLDB Conference, 2004. http://www.cs.washington.edu/woogle

[3] Kaarthik Sivashanmugam, Kunal Verma, Amit Sheth and John Miller, "Adding Semantic to Web Services Standards", ICWS, 2003.

[4] Massimo Paolucci, Takahiro Kawamura, Terry R. Payne and Katia Sycara, "Semantic Matching of Web Services Capabilities", Semantic Web Conference on The Semantic Web, 2002.

[5] John Gekas and Maria Fasli, "Automatic Web Service Composition Using Web Connectivity Analysis Techniques", W3C Workshop, 2005.

[6] Daniel Berardi, Diego Calvanese, Giuseppe De Giacomo, Richard Hull and Massimo Mecella, "Automatic Composition of Transition-based Semantic Web Services with Messaging", VLDB Conference, 2005.

[7] W3C Note, "Web Service Description Language (WSDL) 1.1", http://www.w3.org/TR/wsdl#_rpcexample (2001)

[8] OASIS UDDI Specification Technical Commitee Draft, "UDDI Version 3.0.2 Specification", http://uddi.org/pubs/uddi_v3.htm (2004)

[9] 이석호, "데이타베이스 시스템", 정익사, 2006

[10] 이석호, "자료구조와 자바", 정익사, 2004

Acknowledgement

본 연구는 2006년도 두뇌한국21사업과, 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성, 지원 사업(IITA-2005 -C1090-0502-0016)의 연구결과로 수행되었음.