객체지향 데이터베이스를 이용한 지식베이스 모형(OOKS) 개발※

허순영* · 김형민* · 양근우* · 최지윤**

Development of OOKS: a Knowledge Base Model Using an Object-Oriented Database

Soon-Young Huh* · Hyung-Min Kim* · Kun-Woo Yang* · Ji-Yun Choi**

Abstract

Building a knowledge base effectively has been an important research area in the expert systems field. A variety of approaches have been studied including rules, semantic networks, and frames to represent the knowledge base for expert systems. As the size and complexity of the knowledge base get larger and more complicated, the integration of knowledge bases with database technolngy becomes more important to process the large amount of data. However, relational database management systems show many limitations in handling the complicated human knowledge due to its simple two dimensional table structure.

In this paper, we propose Object-Oriented Knowledge Store (OOKS), a knowledge base model on the basis of a frame sturcture using an object-oriented database. In the proposed model, managing rules for inferencing and facts about objects in one uniform structure, knowledge and data can be tightly coupled and the performance of reasoning can be improved. For building a knowledge base, a knowledge script file representing rules and facts is used and the script file is transferred into a frame structure in database systems. Specifically, designing a frame structure in the database model as it is, it can facilitate management and utilization of knowledge in expert systems. To test the appropriateness of the proposed knowledge base model, a prototype system has been developed using a commercial ODBMS called ObjectStore and C++ programming language.

Keyword: Knowledge Base, ODBMS (Object-Oriented Database Management Systems)

※ 이 논문은 1996년도 한국학술진흥재단의 공모과제 연구비에 의하여 연구되었음
* 한국과학기술원 테크노경영대학원
** 대림정보통신

1. 서 론

전문가의 지식을 효과적으로 표현하고 활용하기 위한 연구는 인공지능 연구의 핵심 분야 중의 하나인 전문가시스템 연구의 중심이 되어 왔다. 전문가시스템의 추론을 위하여 표현된 전문가의 지식들은 반복사용과 여러 사용자들 사이의 공유를 위해 시스템의 영속적인 저장장소(persistent storage)에 저장되어야 한다. 이를 위해서 전통적인 전문가시스템에서는 파일시스템을 이용하여 왔는데 [3, 12, 23], 파일시스템에 저장되어 있는 지식이 추론에 이용되기 위해서는 추론 전에 지식이 메모리에 적재(load)되어야 하므로 지식이 방대해짐에 따라 메모리 용량에 따른 한계를 가지게 되며, 시스템의 초기 적재 시간이 길어지는 단점을 가지고 있다. 이러한 단점을 극복하기 위한 효과적인 방법은 전문가시스템의 지식베이스를 구축함에 있어서 지식을 데이터베이스 내에서 관리하는 것이다[8, 9, 17]. 지식을 데이터베이스를 이용하여 저장하게 되면 다음과 같은 장점을 가질 수 있다. (1) 추론시 초기에 필요한 지식을 모두 메모리에 적재할 필요가 없으며, 추론기관(inference engine)이 추론을 진행하면서 필요한 지식을 데이터베이스에서 가져오게 되므로 메모리의 한계를 극복할 수 있다. (2) 데이터베이스가 제공하는 다중사용자 접근 기능을 이용할 수 있게 됨에 따라 저장된 지식에 여러 명의 사용자가 동시에 접근할 수 있다. (3) 데이터베이스 관리시스템(DBMS)에서 제공하는 최적화된 질의, 데이터 보안 등의 다양한 기능들을 지식베이스 구축에 활용할 수 있다.

전문가시스템의 지식을 데이터베이스를 이용하여 저장, 관리하기 위한 연구들은 주로 기존에 구축되어 있는 전문가시스템과 데이터베이스를 연계하기 위한 방법들로서, 대부분 관계형 데이터베이스를 근간으로 하고 있다[2, 6, 25, 29, 35]. 특히, Owrang[25]은 규칙기반 전문가시스템에서 사용되는 지식을 관계형 데이터베이스로부터 추출할 수 있는 방법을 제안하였으며, Yang[35]은 SQL코드를 기반으로 기존에 구축되어 있는 전문가시스템과 관계형 데이터베이스를 연계할 수 있는 모형을 제시하였다. 그러나 전문가시스템의 지식을 관계형 데이터베이스를 이용하여 관리하고자 하는 경우 다음과 같은 약점들을 가지게 된다. (1) 지식이 가지는 객체간의 상속(inheritance) 관계나 처리절차(procedure) 등과 같은 지식의 복잡한 구조를 테이블 형태의 자료로 표현하는데 어려움이 있다. (2) 전문가시스템의 지식베이스에서 다루는 지식의 구조와 관계형 데이터베이스의 테이블 구조가 상이하기 때문에 중간에 자료구조의 변환을 수행하는 인터페이스를 필요로 한다. (3) 추론을 위한 지식을 관계형 데이터베이스 내에 저장하고자 하는 경우, 정규화(normalization)의 필요성 때문에 그 내용이 여러 개의 테이블에 나누어 저장되어야 하며, 이는 추론의 수행시에 추론속도를 현저하게 저하시키는 요인이 되는 조인 (join)연산을 많이 필요로 하게 된다. 이러한 점들은 기본적으로 관계형 데이터베이스 시스템과 SQL과 같은 질의어가 테이블 형태의 2차원적인 데이터를 다루는데 적합하도록 설계되었을 뿐 전문가시스템에서 다루는 지식을 표현하고 이를 다루는 방법들을 잘 수용하지 못하기 때문으로 여겨진다.

따라서, 전문가시스템에서 다루는 지식을 데이터베이스 내에 저장, 관리하기 위해서는 새로운 형태의 지식베이스 모형의 개발이 필요하며, 이를 위하여 최근에는 복잡한 객체구조를 쉽게

표현할 수 있는 객체지향 기법을 활용하여 지식베이스를 구축하는 방법들이 연구되고 있다. 예를 들어, Higa[15]는 지식베이스와 데이터베이스의 통합을 위한 객체지향 모델링 방법론(structured object model)을 제시하였으며, Xu[34]는 지식베이스의 구축을 위한 객체지향 논리 체계(object-oriented logic framework)를 제시하였다. 또한, Babiker[7]는 기존의 비 객체지향 방법을 통해 개발된 전문가시스템을 객체지향 기법을 활용하여 재설계함으로써 시스템의 확장과 유지보수가 용이하고, 다양한 환경에서 운용가능함을 보였으며, Leung[22]은 객체지향 전문가시스템 구조를 설계하기 위한 객체지향 기술의 적용가능성을 제시하였다. 그러나 이러한 기존의 연구들은 문제해결을 위한 전문가들의 지식이 실제로 복잡한 구조를 가지고 있기 때문에 이를 지식베이스 내에 표현하기 위해서는 객체지향 기법을 지식베이스 구축에 이용하여야 한다는 개념적인 방향 제시에 그치고 있으며, 실제로 객체지향 데이터베이스를 이용한 구체적인 구현 모형을 제시하고 있지는 못하다. 따라서 이 논문에서는 객체지향 데이터베이스를 이용한 지식베이스 구축을 위한 첫 단계로서 지식을 객체지향 데이터베이스 내에 저장하고 저장된 지식을 이용하여 추론을 수행하기 위한 구체적인 지식베이스 모형인 Object-Oriented Knowledge Store(OOKS)를 제시한다. 제시된 지식베이스 모형을 통해 전문가시스템의 추론을 위한 지식을 데이터베이스 내에 객체로 관리하며, 이들을 추론기관이 직접 검색하게 하여 지식의 저장 관리와 추론기능의 효율성을 높일 수 있다. 특히, 이 논문에서는 지식표현을 위한 여러 가지 방법들 중에서 현재 전문가시스템에서 추론을 위해 많이 사용되고 있는 사실(fact)과 규칙(production rule)[30]에 중점을 두어 논의하고자 한다.

객체지향 데이터베이스는 데이터베이스의 스키마(schema)를 클래스 구조를 이용하여 정의하도록 되어 있다[18]. 하지만 지식을 객체지향 데이터베이스 내에 저장한다고 하여 지식을 클래스 구조로 표현하여 이를 바로 데이터베이스 스키마로 사용할 수는 없다. 왜냐하면, 지식의 형태가 다양하므로 객체지향 데이터베이스의 클래스 구조에 지식을 바로 저장하게 되면, 새로운 지식을 저장할 때마다 거기에 맞는 클래스를 추가해야 함으로써 데이터베이스 스키마 설계를 다시 해야 하기 때문에 이 방안은 현실적인 안이 되지 못하기 때문이다. 따라서 다양한 형태의 지식을 저장하기 위한 틀(template)로서 이 논문에서는 프레임(frame)구조[4, 30]를 이용한다. 프레임 구조는 매우 유연한 구조로 구성되어 있기 때문에 규칙이나 사실 등과 같은 추론을 위한 지식을 쉽게 표현할 수 있으며, 객체로 프레임을 구현하는 것도 어렵지 않은 장점이 있다. 이 논문의 구성은 다음과 같다. 2장에서는 지식표현을 위한 프레임의 구조에 대하여 알아보고, 3장에서는 제안된 지식베이스 모형인 OOKS의 구성과 지식저장을 위한 객체지향 데이터베이스 설계에 대하여 설명한다. 4장에서는 정의된 지식이 데이터베이스 내에 저장되는 방법에 대하여 알아보고, 5장에서는 제안된 지식베이스 모형을 이용한 추론 수행시의 내부처리절차와 이에 따른 장점에 대하여 설명한다. 마지막으로 6장에서 논문의 결과와 향후연구 과제를 정리한다.

2. 지식의 표현과 저장

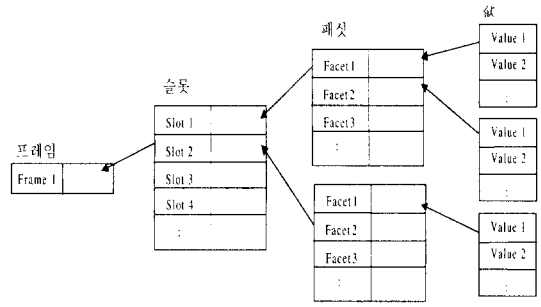
전문가가 소유하고 있는 지식을 표현하는 방법 (knowledge representation)에 관한 연구는 그동안 활발히 진행되어 왔는데, 가장 널리 알려진

지식표현의 방법으로는 의미망(semantic network) [30], 프레임, 규칙 등이 있다. 이중 프레임은 의미망의 단순한 노드구조를 보다 체계화하여 어떤 대상에 대한 여러 가지의 상황 정보들을 하나의 구조화된 틀로서 표현할 수 있는 자료구조로서 다양한 형태의 지식을 담을 수 있는 유연한 구조를 제공한다.

2.1 지식표현을 위한 프레임의 구조

지식표현을 위한 방법들 중 프레임에 의한 방법은 객체지향적 요소를 많이 가지고 있으며 인간이 객체를 이해하고 표현하는 방법과 유사하기 때문에 전문가의 지식을 나타내는데 적합한 구조를 제공한다. 특히, 지식이 가지는 계층적인 구조를 프레임간의 IS-A관계로 표현할 수 있으며, 프레임의 구체적인 인스턴스(instance)들은 해당 프레임과 INSTANCE-OF관계로 맺어 줄 수 있다. 프레임은 기본적으로 어떤 대상에 대한 여러 가지 정보들을 슬롯(slot)과 해당 슬롯에 대한 값(value)으로 표현하는데, 하나의 슬롯은 두 개 이상 복수의 값을 가질 수 있으며, 각각의 값을 다른 이름으로 구분해 줄 필요가 있을 경우에는 패싯(facet)이라는 단위로 슬롯을 나누어 값을 저장할 수 있는 유연한 구조를 가지고 있다[4, 26]. 특히, 프레임은 슬롯 값으로 다른 프레임을 가질 수 있는데 이는 객체지향 기법에서의 객체값을 갖는 속성(object-valued attribute)과 같은 것으로서, 이러한 성질을 이용하면 지식표현에 있어서 구조적으로 복잡한 구성관계를 개념적 변환 없이 직접 표현할 수 있다는 장점을 가지게 된다. 이러한 프레임의 기본적인 구조를 그림으로 나타내면 [그림 1]과 같다.

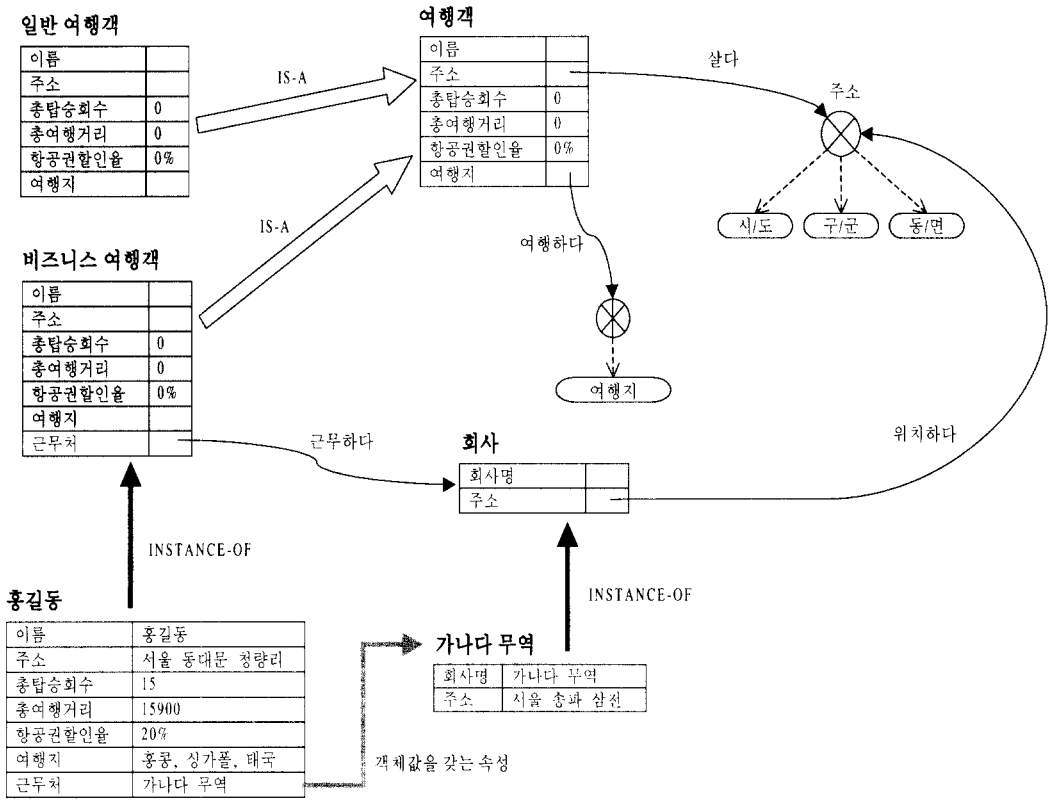
즉, 하나의 프레임은 여러 개의 슬롯을 가질



〔그림 1〕 지식표현을 위한 프레임 구조

수 있으며, 슬롯은 다시 여러 개의 패싯을 가질 수 있다. 그리고 각각의 패싯은 그 안에 여러 개의 값들을 가질 수 있다. 물론, 슬롯은 여러 개의 패싯을 가지지 않고 직접 여러 개의 값을 가질 수도 있으며, 이 경우는 이 슬롯이 하나의 패싯을 가지고 그 패싯이 여러 개의 값을 가지는 것으로 생각할 수 있다. [그림 2]는 이러한 프레임의 구조를 이용하여, 데이터 모델에서 지식의 표현에 많이 활용되는 Hull[16]의 세계 여행객(world traveler) 예에 나타나는 지식을 표현한 것이다.

[그림 2]의 예에서 일반 여행객과 비즈니스 여행객 프레임은 여행객이라는 상위 프레임과 IS-A 관계를 가지게 되며, 따라서 상위프레임이 가지는 슬롯들을 하위프레임에서 상속(inheritance) 받게 됨을 알 수 있다. 이러한 슬롯들 중 주소는 여러 개의 값들이 모여 하나의 값을 이루는 집합(aggregation) 슬롯이며 여행지는 복수개의 값을 가질 수 있는 슬롯임을 나타낸다. 또한 비즈니스 여행객이 가지는 슬롯 중 근무처 슬롯은 회사라는 다른 프레임을 그 값으로 가지는데 이는 앞서 설명한 바와 같이 객체지향 기법의 객체값을 갖는 속성을 이용하여 표현한다. 예를 들어 홍길동 프레임은 비즈니스 여행객 프레임의 구체적인 인스턴스로서 비즈니스 여행객 프



〔그림 2〕 여행객에 관한 지식을 표현하는 프레임의 예

레임과 INSTACE-OF 관계로 연결되어 있으며 근무처 슬롯에 회사 프레임의 구체적인 인스턴스인 가나다무역을 그 값으로 가지고 있다.

이와 같이 프레임 구조를 가지는 객체에 대한 사실 정보를 데이터베이스 내에 저장하기 위한 방법으로는 크게 그래픽 사용자 인터페이스(Graphical User Interface)를 이용하여 정해진 격자구조(widget)를 통해 사용자로부터 사실정보를 입력 받아 저장하는 방법과 시스템이 이해할 수 있는 명령어를 통해 텍스트 형태로 표현된 자료를 구문해석(parsing)하여 저장하는 방법이 있다. 전자의 경우 직관적인 표현구조로 사용자에게 편리성을 제공하나 지식입력을 위해 정해진 화면을 이용하면 다양한 지식구조를 표현하기 위한 유연성이 떨어지고 사용자가 자유롭게 격자구조를 정의하여 활용할 수 있도록 하려면 시스템 설계가 복잡해지는 단점을 가진다. 한편 후자의 경우 사실을 정의하기 위한 명령어를 시스템에서 제공할 필요가 있고 이를 사용자가 익혀야 하는 부담이 있는 반면에 주어진 명령어를 이용하여 다양한 지식구조를 표현할 수 있는 장점이 있다. 이 논문에서 제시하는 지식베이스에서는 복잡한 구조를 가진 사실 정보의 표현을 비교적 간단한 시스템 설계로 가능케 하는 명령어를 이용한 구문해석 방식을 채택한다. 시스템에서 제공하는 명령어의 구조와 실제로 이를 이

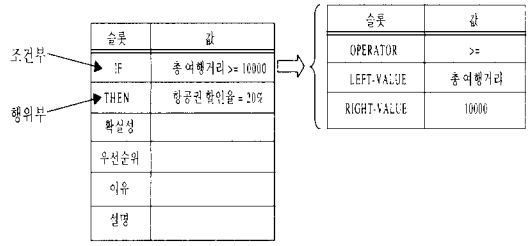
용하여 데이터베이스에 사실 정보를 저장하는 방법에 대해서는 4장에서 자세히 설명하기로 한다.

2.2 프레임 구조를 이용한 규칙의 저장

이 절에서는 추론을 위한 규칙을 프레임 구조를 이용하여 저장하는 방법과 이렇게 프레임 구조에 담긴 규칙을 표현식 프레임이라는 구조로 나누어 객체지향 데이터베이스에 저장하는 방법을 통해 추론을 수행할 때 얻을 수 있는 장점에 대해 설명한다.

지식의 표현방법 중 규칙에 의한 방법은 그 내용을 이해하기가 쉽고 규칙간의 독립성이 보장되며 추론과 설명이 쉽게 도출될 수 있기 때문에 전문가시스템에서 추론의 수행을 위한 지식 표현에 많이 사용되고 있다. 그러나 이러한 규칙에 의한 표현방법은 규칙의 수가 많아지면 지식의 분류 및 검색이 어려워진다는 단점을 가지고 있다[4, 32]. 이 논문에서 제안하고자 하는 지식베이스 모형은 규칙으로 표현된 지식들을 데이터베이스 내에 저장하고, 추론기관이 추론을 수행할 때에 저장된 규칙들을 직접 검색할 수 있게 지원한다. 따라서 규칙을 데이터베이스 내에 저장함에 있어서, 규칙의 검색과 찾아진 규칙의 평가를 쉽게 수행하기 위한 일관성 있고 구조화된 틀이 필요한데, 이를 위해서 앞에서 설명한 바와 같이 지식 표현에 있어서 유연한 구조를 제공하는 프레임 구조를 이용한다. 프레임 구조를 이용하면 하나의 프레임에서 다른 프레임으로의 연결을 앞서 설명한 바와 같이 객체 값을 갖는 속성을 통해 표현하는 것이 가능하여 추론에 필요한 규칙의 검색이 빠르게 수행될 수 있다.

규칙을 프레임 구조를 이용하여 저장하는 기본적인 방법은 프레임 안에 IF 슬롯과 THEN 슬롯을 만들고 IF 슬롯의 값으로 상황, 전제, 증거, 원인에 해당하는 내용인 조건부(condition part)를 THEN 슬롯의 값으로 그 조건부가 만족하였을 경우에 참이 되는 행동, 결론, 가설, 결과인 행위부(action part)를 저장하는 것이다. 그리고 추론 시 사용되는 해당 규칙의 확실성(certainty), 우선순위(priority), 이유, 설명과 같은 규칙이 가질 수 있는 성질들도 별도의 슬롯으로 만들어 표현한다. [그림 3]의 (a)의 총 여행거리가 10000 마일 이상이면 항공권 할인율은 20%이다 라는 간단한 규칙을 프레임 구조 내에 저장한 예이다.



(a) 규칙의 저장 예 (b) 조건부 표현식의 저장 예

[그림 3] 프레임 구조를 이용한 규칙의 저장 예

규칙 형태의 지식을 프레임 구조를 이용하여 데이터베이스 내에 저장하기 위해서는 사실을 데이터베이스 내에 저장하는 방법과 마찬가지로 정해진 격자구조(widget)를 이용하거나 시스템 제공 명령어를 이용하여 텍스트 형태로 작성하는 방법을 사용할 수 있는데 이 논문에서는 규칙 표현에 있어서 유연성을 제공하며 현재 많은 전문가시스템에서 채택[3, 12, 23]하고 있는 명령어 방식을 이용한다. 이러한 명령어 방식을 이용하기 위해서는 규칙들을 시스템이 이해할 수 있는 형태로 정의할 수 있어야 하며, 시스템은 정의된 규칙들을 구문해석하여 그 결과를 데이터베

이스 내에 저장하여야 한다. 이때, 사용자는 규칙이 내부적으로 프레임 구조를 가지고 저장되는 것을 고려할 필요가 없도록 시스템이 명령어들을 제공해 주어야 하며, 이를 위해서는 개별 규칙을 정의하기 위한 명령어와 추론시 필요한 규칙의 검색을 쉽게 하기 위하여 정의된 규칙들 중 같은 목적으로 사용되는 규칙들을 하나의 그룹으로 묶어주기 위한 명령어가 필요하다. 이러한 명령어의 구조와 데이터베이스 내에 저장되는 구체적인 방법들은 4장에서 보다 자세히 설명한다.

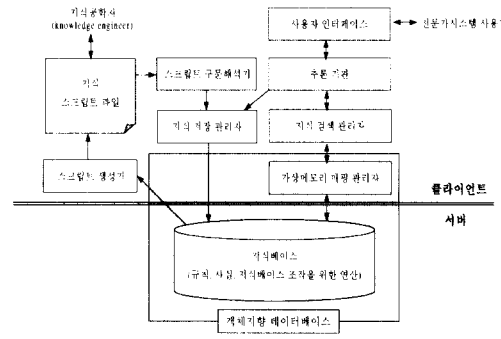
실제로 추론기관이 데이터베이스 내에 저장된 규칙을 이용하여 추론을 수행하게 되면, 조건부의 참 거짓여부를 평가하게 되며, 이를 위해서는 조건부에 나타나는 표현식(expression)을 [그림 3]의 (a)에 나타나 있는 바와 같이 하나의 텍스트 형태의 문장으로 관리하는 것보다 추론시 조건부를 평가할 때에 별도의 구문해석이 필요 없도록, 규칙을 데이터베이스 내에 저장하는 시점에 이를 완전히 구문해석하여 그 결과를 저장하는 것이 추론속도 면에서 유리하다. 따라서 이 논문에서는 [그림 3]의 (b)와 같이 조건부에 나타나게 되는 표현식을 연산자(operator)와 왼쪽 값(left-value), 오른쪽 값(right-value)으로 나누어 프레임 구조를 이용하여 저장, 관리한다. 이렇게 함으로써 추론의 수행시에 규칙에 대한 의미를 파악하기 위한 별도의 구문해석이 필요 없게 되며, 이는 추론속도 향상에 기여하게 된다.

3. 객체지향 데이터베이스를 이용한 지식베이스 모형 개발

이 장에서는 제안하고자 하는 지식베이스 모형의 구성과 지식을 객체지향 데이터베이스 내에 저장하기 위한 기본적인 구성체들에 대하여 논의한다. 특히, 제안된 지식베이스 모형을 이용하여 추론에 필요한 객체에 대한 사실과 규칙형태의 지식을 지식베이스에 저장하는 구조에 대해 중점적으로 설명한다.

3.1 지식베이스 아키텍춰

이 논문에서 제안하고자 하는 객체지향 데이터베이스를 이용한 지식베이스는 지식을 표현하는 규칙과 사실 등을 저장하며, 지식베이스 조작을 위한 연산들을 제공한다. 전문가나 자료로부터 지식을 추출하여 지식베이스를 구축하는 지식공학자 (knowledge engineer)는 지식을 이러한 지식베이스에 저장하고, 저장된 지식의 내용을 보기 위하여 지식 스크립트 파일 (knowledge script file)을 이용하는데, 지식 스크립트 파일은 텍스트 파일 형태로 시스템으로 적재되며, 적재된 스크립트 파일은 시스템에 의해서 구문해석되어 데이터베이스 내에 여러 개의 프레임을 생성하면서 저장되게 된다. 객체지향 데이터베이스 내에 저장된 지식은 추론기관에 의해서 접근이 되며, 전문가시스템 사용자들은 추론을 위하여 사용자 인터페이스를 통해 추론기관을 이용



〔그림 4〕 OOKS의 구성

하게 된다. 이를 토대로 본 연구에서는 [그림 4]와 같은 지식베이스 모형인 Object-Oriented Knowledge Store(OOKS)를 사용하며, 그림에 나타나는 각 구성요소들은 다음과 같이 정의될 수 있다.

- 스크립트 구문해석기(script parser)는 지식을 데이터베이스 내에 저장하기 위한 첫 단계로서 지식공학자에 의해 정의된 지식 스크립트 파일에 대한 구문해석을 수행한다. 이를 위해서는 지식공학자가 지식을 표현하기 위해 사용하는 명령어의 기본 구조가 사전에 정의되어 있어야 하며, 스크립트 구문해석기는 텍스트 파일 형태로 작성된 지식 스크립트 파일을 읽어 들여서, 예약어(reserved word)와 비예약어를 구분하고 지식을 데이터베이스 내에 프레임 구조로 저장하기 위한 사전작업을 수행한다.
- 지식저장 관리자(knowledge load manager)는 스크립트 구문해석기에 의하여 구문해석된 지식 또는 추론의 결과로 생성되는 새로운 사실들을 실제로 데이터베이스 내에 프레임 구조를 이용하여 저장하는 역할을 담당한다. 이때 지식의 저장을 위해 사용되는 연산들은 OOKS에서 제공해 주게 된다.
- 스크립트 생성기(script generator)는 데이터베이스 내에 프레임 구조로 저장되어 있는 지식의 내용을 지식 스크립트 파일 형태로 복원시키는 기능을 수행한다. 기존에 저장되어 있는 지식의 내용을 검사하거나 수정하기 위해서 사용자는 저장되어 있는 지식의 내용을 볼 수 있어야 하며, 이때, 사용자는 지식이 실제로 데이터베이스 내에 저장되어 있는 구조를 모르더라도 지식의 내용을 이해할 수 있도록 지식 스크립트 파일로 복원된 지식을 이용한다.
- 사용자 인터페이스(user interface)는 추론을 실행하는 전문가시스템 사용자와 추론기관과의 연결을 담당한다. 이를 통해 사용자는 추론의 목표와 추론의 수행 중에 필요한 정보를 추론기관으로 입력하게 되며, 추론기관은 추론의 결과를 출력하게 된다.
- 추론기관(inference engine)은 지식베이스 내에 저장되어 있는 규칙과 사실 그리고 사용자로부터 입력 받게 되는 정보를 이용하여 설정된 추론목표를 달성하기 위한 계획을 세우고 실제 추론을 실행한다. 이때, 추론기관은 데이터베이스에 저장되어 있는 지식 중에서 필요한 규칙이나 사실을 직접 검색하게 된다. 그리고, 추론의 결과로 생성되는 새로운 사실은 지식저장 관리자를 통해서 지식베이스 내에 저장된다. 추론의 수행에 따른 내부처리절차는 5장에서 설명하고자 한다.
- 지식검색 관리자(knowledge retrieval manager)는 추론기관이 필요로 하는 지식을 데이터베이스로부터 검색하여 그 결과를 추론기관으로 넘겨주는 역할을 담당한다. 데이터베이스로부터 필요한 지식을 검색하기 위한 연산들은 객체지향 데이터베이스를 이용한 지식베이스 내에 정의되며, 지식검색 관리자는 이러한 연산들을 이용하여 지식을 데이터베이스로부터 가져오게 된다.
- 가상메모리 매핑 관리자(virtual memory mapping manager)는 지식검색 관리자에 의한 데이터베이스로의 접근 속도를 향상시키기 위하여 추론에 필요한 지식을 서버의 데이터베이스로부터 클라이언트의 가상메모리 상으로 옮기는 기능을 수행한다. 이러한 기

능의 필요성과 수행과정을 보다 자세히 알아보면 다음과 같다.

객체지향 데이터베이스 내에 저장되어 있는 지식을 추론기관이 필요로 할 때 마다 지식검색 관리자가 실제 데이터베이스가 존재하는 서버의 영속적 메모리(persistent memory)에 접근하여 지식을 가져오는 것은 추론의 성능을 저하시킬 수 있다. 이러한 문제를 해결하기 위하여 이 논문에서 제시하는 지식베이스 모형은 객체지향 데이터베이스 관리시스템(OODBMS)에서 제공하는 가상메모리 매핑 구조(virtual memory mapping architecture)[24, 31]를 이용한다. 가상메모리 매핑 구조는 데이터베이스 내에 저장되어 있는 지식을 추론시에 클라이언트 내의 가상메모리 상으로 이동시켜서 사용할 수 있도록 해주는 기능으로서 이 논문에서 원형시스템(prototype system)을 구현하기 위하여 사용한 ObjectStore[24]와 같은 객체지향 데이터베이스에서 데이터베이스 접근 속도를 향상시키기 위하여 제공하고 있는 구조이다. 앞에서 언급한 바와 같이 가상메모리 매핑 관리자가 이러한 기능을 수행하게 되는데, 가상메모리 매핑 관리자는 추론기관이 필요로 하는 지식을 지식 검색 관리자로부터 요구받게 되면 해당 지식이 가상메모리 상에 존재하는지를 먼저 검사하고, 존재하면 그 내용을 지식 검색 관리자에게 바로 넘겨주게 된다. 만일 필요로 하는 지식이 아직 가상메모리 상으로 옮겨지지 않고 데이터베이스 상에 있으면 가상메모리 매핑 관리자는 데이터베이스에 접근하여 필요한 지식을 가상메모리 상으로 가져오게 된다. 이때, 데이터베이스와 가상메모리 사이의 지식의 이동은 개별 객체 단위로 이루어지는 것이 아니라 세그먼트(segment) 단위로 이루어지게 되므로 추론에 함께 사용되는 관련된 지식들을 하나의 세그먼트에 저장시킴으로써 하나의 지식이 가상메모리 상으로 옮겨지게 되면 그와 관련된 다른 지식들도 함께 가상메모리로 옮겨지게 된다. 따라서 추후에 필요로 하는 지식은 대부분 가상메모리 상에 존재하게 되고, 일단 지식이 가상메모리 상으로 옮겨지면 클라이언트의 운영체제(operating system)는 지식이 클라이언트의 메모리 상에 존재하는 것과 마찬가지로 취급하게 되므로 추론시 지식검색 관리자에 의한 필요한 지식의 검색속도가 현저하게 향상된다.

3.2 객체지향 데이터베이스를 이용한 프레임의 설계

이 절에서는 지식베이스 모형의 구성 요소 중에서 실제로 객체지향 데이터베이스 내에 지식을 저장, 관리하기 위한 틀로서 사용하는 프레임 구조에 대하여 설명한다. 지식을 저장하기 위해서 사용하고 있는 객체지향 데이터베이스의 중요한 성질로는 클래스의 구성(class construction), 객체간의 관계표현(object relationship), 클래스의 계승(class inheritance), 캡슐화(encapsulation) 등을 들 수 있으며[18, 24], 이러한 성질들은 지식을 저장 관리하고, 저장된 지식을 이용한 추론의 수행을 효과적으로 수행할 수 있도록 하는 기반이 된다.

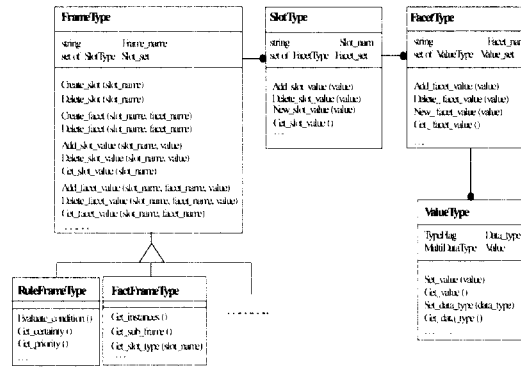
- 클래스의 구성 : 클래스란 데이터베이스에 저장하고자 하는 객체의 속성 (attribute)들과 그 속성들에 적용될 수 있는 연산 (operation)들을 정의해 놓은 것으로서, 이러한 클래스의 인스턴스들은 각각 객체식별자 (object identifier)라 불리우는 그들 고유의 식별자를 가지며, 이러한 객체식별자는 그 인스턴스들을 데이터베이스에서 불러내는데 사용된

다. 따라서 이 논문에서 지식을 저장하기 위한 틀로서 사용하고 있는 프레임 구조와 이에 적용되는 연산들을 클래스 구조로 정의할 수 있으며, 추론에 이용되는 구체적인 지식들은 이러한 클래스의 인스턴스들이 된다. 예를 들어, [그림 2]에서 홍길동, 가나다 무역 등의 프레임은 각각 비즈니스 여행객, 회사라는 프레임 클래스의 구체적인 인스턴스로서 객체식별자를 가지며 해당 클래스에 정의된 속성과 연산을 이용하게 된다.

- 객체간의 관계표현 : 객체지향 데이터베이스에서는 한 객체의 속성으로 다른 객체(엄밀하게는 객체식별자)를 가질 수 있도록 해줌 (object-valued attribute)으로써 한 객체를 통한 다른 객체의 접근을 가능하도록 한다. 이는 프레임 구조에서 슬롯이 그 값으로 다른 프레임을 가질 수 있는 것을 데이터베이스 내에 잘 표현해 줄 수 있으며, 데이터베이스 내에서 관련된 다른 프레임으로의 이동이 객체식별자를 이용하여 이루어지게 함으로써 추론시 관련된 프레임을 검색하기 위한 시간이 거의 소요되지 않는다. 한편, 객체지향 데이터베이스에서 클래스의 속성은 그 값을 여러 개 가질 수도 있는데 (multi-valued attribute), 객체지향 데이터베이스의 이러한 성질은 프레임 구조에서 슬롯이 여러 개의 패싯을 가질 수 있고, 또한 각각의 패싯은 여러 개의 값을 가질 수 있는 것을 데이터베이스 내에 잘 표현해 줄 수 있다. [그림 2]의 예에서 홍길동 프레임의 근무처 슬롯은 가나다 무역이라는 또 다른 프레임을 그 값으로 가지는데 이러한 객체 간의 관계표현은 객체지향 데이터베이스 내에서 홍길동 프레임의 속성값으로 가나다 무역 프레임의 객체식별자를 갖게 함으로써 가능하다.
- 클래스의 계승 : 객체지향 데이터베이스는 객체간의 계층적인 구조를 계승 메커니즘 (inheritance mechanism)을 이용하여 표현할 수 있다. 계승 메커니즘을 통해서 새롭게 정의되는 클래스는 이미 존재하는 클래스의 속성들과 연산들을 계승 받을 수 있으며, 여기에 새로운 속성들과 연산들을 추가로 정의할 수 있다. 전문가가 가지는 지식의 형태는 매우 다양하며, 지식베이스 내에 다양한 형태의 지식을 수용하기 위해서는 지식베이스를 구성하는 데이터 모델의 확장가능성이 중요하다. 이 논문에서는 지식의 표현방법으로 사실과 규칙에 중점을 두고 있으며, 그 밖의 새로운 형태의 지식표현 방법은 이 논문에서 지식저장을 위한 틀로 사용하고 있는 프레임을 표현하는 클래스를 계승받아 새로운 클래스를 정의함으로써 새로운 지식표현 방식으로 그 범위를 확장시켜 나갈 수 있다. [그림 2]를 보면 일반 여행객과 비즈니스 여행객 프레임은 그 상위 프레임인 여행객 프레임과 IS-A 관계로 연결되어 여행객 프레임의 속성들을 상속받고 있는데 이는 클래스간의 속성과 연산 계승 구조를 이용하면 쉽게 구현할 수 있다.
- 캡슐화 : 객체지향 데이터베이스는 한 클래스와 그 하위클래스 (subclass)에 적용되는 연산들을 그 클래스가 가지고 있도록 하며, 이러한 연산을 통해서만 그 클래스의 속성에 접근할 수 있도록 한다. 이러한 성질을 이용하여 프레임 구조를 이용한 지식베이스 조작을 위한 연산들을 데이터베이스 내에 정의해 놓을 수 있으며, 이는 한 프레임의

내용에 접근할 수 있는 인터페이스를 일정하게 고정시켜 놓음으로써 사용자가 프레임의 내용에 함부로 접근하는 것을 막고, 미리 정의되어 있는 인터페이스를 통하여 사용자는 원하는 지식에 쉽게 접근할 수 있다. 따라서 앞에서 설명한 지식저장 관리자나 지식검색 관리자는 지식을 저장하거나 지식을 검색할 때에 데이터베이스에 정의되어 있는 지식베이스 조작을 위한 연산들을 이용함으로써 지식이 저장되어 있는 데이터베이스에서의 내부 메커니즘은 신경 쓸 필요 없이 원하는 지식을 쉽게 저장 또는 검색 할 수 있다. 이 논문에서 제시하는 지식베이스 모형에서는 [그림 5]와 같이 프레임을 이용하여 저장하고자 하는 지식의 종류에 따라 상위 프레임 클래스로부터 계승된 규칙, 사실, 표현식 하위 프레임 클래스를 이용하게 된다. 이러한 하위 클래스들에는 해당 지식의 종류에 따라 지식을 저장하고 활용하기 위해 필요한 추가적인 속성과 연산들이 정의되어 있다.

지식을 데이터베이스 내에 저장하기 위한 틀로서 사용하고 있는 프레임의 구조는 [그림 1]에 나타나 있는 바와 같으며, 앞에서 설명한 객체지향 데이터베이스의 성질들을 이용하여 이들 간의 관계를 데이터베이스 내에 그대로 표현하기 위하여 프레임 타입(Frame Type), 슬롯 타입(Slot Type), 패싯 타입(Facet Type), 밸류 타입(Value Type) 이라는 클래스를 이용한다. 특히, 실제 슬롯 값이나 패싯 값이 가질 수 있는 데이터는 실수, 정수, 문자 등과 같이 다양하게 나타나게 되는데 이러한 다양한 데이터형을 모두 수용하기 위하여 다중 데이터형(MultiData Type)이라는 데이터형을 이용한다. 다중 데이터형은 정수형, 실수형, 문자형과 같은 기본적인 데이터형과 이들의 포인터 값 그리고 객체식별자 등을 모두 저장할 수 있는 데이터형으로 C나 C++ 프로그래밍 언어에서 하나의 데이터형으로 여러 가지의 데이터형들을 수용할 수 있도록 해주는 공용체 (union)를 이용하여 구현되었다. 프레임 구조를 데이터베이스 내에 표현하기 위한 기본적인 구성체들과 이들 간의 관계를 OMT(Object Modeling Technique)[28]를 이용하여 나타내 보면 [그림 5]와 같으며, 각 구성체는 C++ 프로그래밍 언어를 지원하는 객체지향 데이터베이스 시스템인 Ob- jectStore[24]를 이용하여 구현되었다.



〔그림 5〕 프레임 구조에 대한 객체 모델링

[그림 5]에서 각각의 클래스를 나타내는 사각형은 세 부분으로 이루어지는데, 위의 부분은 클래스의 이름을, 가운데 부분은 해당 클래스의 속성을, 그리고 아래 부분은 해당 클래스가 가지는 연산구조를 나타낸다. 클래스 간의 관계는 실선으로 표시되는데, 실선 끝의 검은 원은 두 클래스 간의 관계에 있어서 검은 원이 있는 쪽이 복수 개가 될 수 있음을 나타내며, 실선 중간의 삼각형은 삼각형 위로부터 아래로의 계승관계를 나타낸다. 객체지향 데이터베이스에서는

앞에서 설명한 바와 같이 하나의 속성이 여러 개의 값을 가지는 것을 허용하며, 이는 클래스의 속성을 선언할 때에 해당 속성이 집합(set)임을 명시해 줌으로써 가능하다. 따라서 프레임이 여러 개의 슬롯을 가질 수 있음을 나타내는 FrameType 클래스의 Slot_set속성과 슬롯이 여러 개의 패싯을 가질 수 있음을 나타내는 SlotType 클래스의 Facet_set속성, 그리고 패싯이 여러 가지의 값을 가질 수 있음을 나타내는 FacetType 클래스의 Value_set속성들은 모두 복수 개의 값을 가질 수 있음을 나타내는 집합으로 정의된다. 그리고 실제로 패싯이 가지는 값은 ValueType 클래스의 Value 속성에 저장되게 되는데 이는 다중 데이터형(MultiDataType)으로 정의된다. 따라서 각각의 패싯 또는 슬롯이 가질 수 있는 값으로는 정수형, 실수형, 문자형 등과 같은 기본적인 데이터형과 프레임을 표현하는 클래스의 객체식별자 등 지식을 저장하는데 있어서 필요한 대부분의 데이터형을 수용할 수 있다.

객체지향 데이터베이스에서는 앞서 설명한 바와 같이 클래스의 속성들을 조작하기 위한 함수들을 해당 클래스의 연산으로 정의하여 데이터베이스 내에서 관리할 수 있으며, 프레임을 다루기 위한 함수들은 프레임을 구성하는 클래스들의 연산들로서 정의된다. 따라서, 하나의 프레임을 데이터베이스 내에 생성하게 되면, 슬롯 및 패싯의 생성과 삭제([그림 5]의 FrameType 클래스의 연산들 중 Create_slot(), Delete_slot(), Create_facet(), Delete_facet()), 슬롯과 패싯 값에 대한 조작 ([그림 5]의 Frametype 클래스의 연산들 중 Add_slot_value(), Delete_slot_value(), Add_facet_value(), Delete_facet_value() 등) 등과 같은 프레임을 다루기 위한 연산들을 해당 프레임에 대하여 모두 사용할 수 있게 된다. 한편, 그림 5에 나타나 있는 FrameType, SlotType, FacetType, ValueType 클래스는 프레임의 기본적인 구조를 데이터베이스 내에 표현하기 위한 클래스들이며, 실제로 추론을 위한 규칙과 사실들을 프레임 구조를 이용하여 저장하기 위해서는 FrameType 클래스의 모든 속성과 연산들을 계승 받는 하위클래스인 RuleFrameType, FactFrameType 클래스 등을 이용한다. 이는 사실과 규칙이 모두 기본적으로는 프레임 구조 내에 저장되지만 이들을 조작하기 위한 연산들은 지식의 표현방식에 따라 차이가 있기 때문에 프레임 조작을 위한 기본적인 연산들만을 상위클래스인 FrameType 클래스가 가지게 하며, 지식표현 형태에 따른 특화된 연산들은 이를 계승 받은 하위클래스가 가지게 함을 의미한다. 예를 들어, [그림 5]에 나타나 있는 RuleFrameType 클래스는 규칙을 저장하기 위한 클래스로 사용되는데, 프레임을 표현하는 FrameType 클래스를 상위클래스로 갖는다. 따라서 규칙의 내용을 프레임 구조 내에 저장하기 위한 기본적인 프레임 조작을 위한 연산들을 모두 이용할 수 있으며, 특히 규칙을 다루는데 있어서 조건부의 평가 (Evaluation_condition()), 규칙에 대한 확실성 및 우선순위 검색 (Get_certainty(), Get_priority()) 등을 위한 연산들을 추가로 갖는다. 이러한 객체지향 데이터베이스의 계승 메커니즘을 이용한 지식의 저장구조는 다양한 지식표현 방식에 따른 각각의 특성들을 체계적으로 관리할 수 있도록 하며, 새로운 지식표현 방법을 데이터베이스 내에 수용함에 있어서도 기존에 존재하는 클래스를 계승 받는 새로운 하위클래스를 추가함으로써, 이를 쉽고 일관성 있는 방법으로 수용할 수 있는 확장성을 제공해 준다.

4. 지식의 정의 및 저장

이 장에서는 지식을 표현하는 사실과 규칙을

지식베이스에 저장하기 위한 명령어 구조를 소개하고, 실제로 데이터베이스 내에 프레임 구조를 이용하여 저장되는 방법에 대하여 설명한다. 지식을 저장하기 위한 명령어들은 크게 사실을 저장하기 위한 명령어와 규칙을 저장하기 위한 명령어로 나눌 수 있다. 먼저, 사실을 저장하기 위한 명령어로는 첫째, 사실 표현에 사용되는 변수의 타입을 정의하기 위한 명령어 (TYPE 명령어)와 둘째, 대상 객체에 대한 사실을 표현하기 위한 명령어 (FRAME 명령어)가 필요하며, 규칙을 정의하기 위한 명령어로는 첫째, 개별 규칙을 표현하기 위한 명령어 (RULE 명령어)와 둘째, 규칙들 중 추론시 같이 사용되는 규칙들을 묶어주기 위한 명령어 (GROUP 명령어)가 필요하다. 이중 특히, TYPE 명령어는 사실과 규칙을 연결시켜주는 연결고리의 역할을 한다. 다시 말해서, 사실의 표현에 사용된 변수의 타입을 규칙들의 목표 값의 타입을 표현하는데 다시 사용함으로써 특정 사실과 이와 관련된 규칙들을 연결해 줄 수 있다.

(TYPE 명령어의 문법)

```
TYPE <typename_A>
BEGIN
    <number><..><number> of <typename_B>
    | <string><..><string> of <typename_C>
    | <string>|<number> [{,<string>|<number>}]
END
(a)        (b)
```

(TYPE 명령어의 작성 예)

```
TYPE _DISTANCE
BEGIN
      0 .. 1000000 of INTEGER
END

TYPE _DISCOUNTRATE
BEGIN
      0, 0.05, 0.1, 0.15, 0.2
END
            (c)
```

(FRAME 명령어의 문법)

```
FRAME <framename>
BEGIN
    {<slotname>[<slotvalue>[{,<slotvalue>}]]
    |<slotname>[{.<facetname>[<facetvalue>[{,<facetvalue>}]]}]}
END
```

(FRAME 명령어의 작성 예)

```
FRAME  TRAVELER
BEGIN
    NAME
    ADDRESS
END

FRAME  BUSINESS_TRAVELER
BEGIN
    IS-A                        TRAVELER
    COMPANY
    TOTAL_TRAVEL_NO
    TOTAL_DISTANCE.TYPE   _DISTANCE
    DISCOUNT_RATE.TYPE    _DISCOUNTRATE
END

FRAME  HONG_GIL_DONG
BEGIN
    INSTANCE-OF BUSINESS_TRAVELER
    NAME                HONG_GIL_DONG
    ADDRESS             SEOUL_DONGDAEMUNGU
    COMPANY               GANADA_TRADING
    TOTAL_TRAVEL_NO     15
    TOTAL_DISTANCE      15900
    DISCOUNT_RATE
END
        (d)
```

〔그림 6〕 사실을 표현하기 위한 명령어의 문법과 작성 예

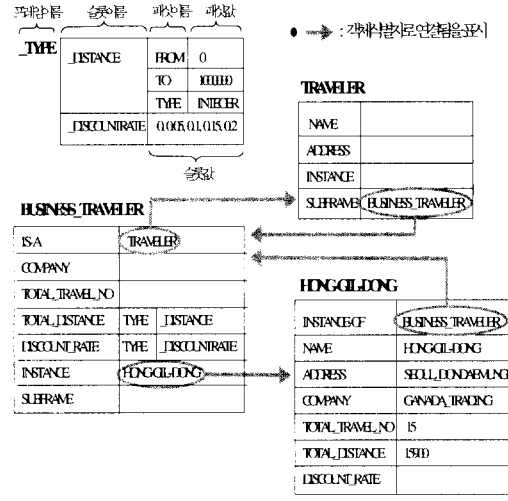
4.1 사실의 표현 및 저장

사실을 표현하기 위한 명령어는 앞에서 설명한 바와 같이 TYPE 명령어와 FRAME 명령어가 있으며, [그림 6]의 (a)와 (b)는 두 명령어의 문법을 BNF(Backus-Naur Form)로 나타낸 것이다. BNF의 표시법 (notation)에서 |는 OR관계를 나타내며, []는 선택적(optional)인 것을, 그리고 {}는 반복적으로 나타나는 것이 가능함을 표시한다.

먼저, TYPE명령어는 지식을 표현하기 위해 사용되는 변수들의 타입 즉, 변수들이 가질 수 있는 값의 범위를 정의하는데 사용된다. [그림 6]의 (a)에 나타나 있는 바와 같이 TYPE 명령어는 기본적으로 숫자형과 문자형으로 타입이 가질 수 있는 값의 범위를 정할 수 있으며, 이미 정의되어 있는 다른 타입을 이용하여 그 범위를 정의할 수도 있다. [그림 6]의 (c)는 이러한 TYPE 명령어를 이용하여 _DISTANCE라는 이름을 갖는 타입과 _DISCOUNTRATE이라는 이름을 갖는 타입을 정의하는 예를 보여준다. 이 작성 예에서 _DISTANCE는 정수이며 0에서 1000000까지의 범위를 가질 수 있는 타입임을 나타내며, _DISCOUNTRATE는 0, 0.05, 0.1, 0.15, 0.2 중에 하나의 값을 가질 수 있는 타입임을 나타낸다. FRAME 명령어를 이용한 작성 예는 다음으로 FRAME 명령어는 대상 객체의 사실정보를 프레임의 형태로 저장하기 위해서 사용한다. [그림 6]의 (b)는 이러한 FRAME 명령어의 문법을 나타내며, 프레임이 가지는 슬롯과 슬롯값 그리고 패싯과 패싯값에 대한 정보를 그대로 표현하도록 되어 있다.

FRAME 명령어를 이용한 작성 예는 [그림 6]의 (d)와 같으며, 이는 [그림 2]의 여행객에 대한 사실을 표현하는 프레임들의 일부를 나타낸다. 이 작성 예에서 BUSINESS_TRAVELER 프레임은 TRAVELER 프레임의 NAME슬롯과 ADDRESS 슬롯을 상속 받게 되며, HONG_GIL_DONG 프레임은 BUSINESS_TRAVELER 프레임의 인스턴스가 된다. 인스턴스가 아닌 프레임들은 TYPE 명령어에 의해 선언된 타입을 이용하여 슬롯이 가질 수 있는 값의 타입을 선언할 수 있는데, 위의 작성 예에서 BUSINESS_TRAVELER 프레임의 TOTAL_DISTANCE 슬롯과 DISCOUNT_RATE 슬롯은 [그림 6]의 (c)에서 정의한 _DISTANCE 타입과 _DISCOUNTRATE 타입의 값을 갖는다.

이와 같이 지식 스크립트 파일 내에 정의된 사실들은 스크립트 구문해석기에 의해서 구문해석이 되며, 그 결과는 지식저장 관리자를 통해 [그림 7]과 같은 형태로 데이터베이스 내에 저장되게 된다



〔그림 7〕 데이터베이스 내에 저장된 사실의 예

[그림 7]에서 보는 바와 같이 TYPE 명령어에 의해서 정의된 _DISTANCE 타입과 _DISCOUNTRATE 타입은 _TYPE이라는 프레임 내에 각각 하나씩의 슬

롯을 만들면서 저장되게 된다. 특히, _DISTANCE 타입의 경우는 0부터 1000000까지 정수 값을 가질 수 있다는 것을 표현하기 위하여 FROM, TO, TYPE이라는 이름의 패싯을 이용하게 된다. 한편, FRAME 명령어에 의해서 정의된 여행객에 대한 사실들은 각각 하나씩의 프레임을 데이터베이스에 생성하면서 저장된다. 특히, 인스턴스가 아닌 프레임들 ([그림 7]의 예에서 TRAVELER 프레임과 BUSINESS_TRAVELER 프레임)은 데이터베이스에 저장될 때에 자동적으로 SUBFRAME 슬롯과 INSTANCE 슬롯을 생성하게 되는데, SUBFRAME 슬롯은 IS-A 관계로 연결되는 하위프레임들을 관리하기 위한 슬롯이며, INSTANCE 슬롯은 INSTANCE-OF 관계로 연결되는 인스턴스 프레임들을 관리하기 위한 슬롯이다. 앞에서 설명한 바와 같이 본 지식베이스 모형에서는 슬롯 및 패싯의 값을 저장하기 위하여, 기본적인 데이터형뿐 아니라 데이터베이스에 저장되는 객체의 저장주소를 나타내는 객체식별자도 수용할 수 있는 다중 데이터형을 이용하고 있기 때문에 객체간의 관계표현시 이를 각각의 객체식별자를 이용하여 연결해 줄 수 있다. 따라서 프레임간의 계승관계를 나타내는 IS-A 슬롯과 SUBFRAME 슬롯 그리고 특정 프레임과 인스턴스들과의 관계를 나타내는 INSTANCE-OF 슬롯과 INSTANCE 슬롯의 값으로는 관련된 다른 프레임의 객체식별자를 저장하게 된다. 이렇게 함으로써 차후에 추론을 수행할 때에 상위 또는 하위 프레임을 찾고, 특정 프레임에 속하는 인스턴스들이 무엇인지를 찾아내는데 있어서 저장되어 있는 객체식별자를 이용하게 됨으로써 검색시간이 거의 필요하지 않게 된다.

4.2 규칙의 표현 및 저장

규칙 형태의 지식을 데이터베이스 내에 저장하고 이를 추론에 이용하기 위해서는 개별 규칙을 정의하기 위한 명령어와 정의된 규칙들 중 같은 목표를 가지는 규칙들을 그룹화하기 위한

(RULE 명령어의 문법)

```
RULE  <rulename>
BEGIN
    IF <expression> [{ AND | OR <expression> }]
    THEN <string>|<number>
END
```

(a)

(RULE 명령어의 작성 예)

```
RULE Rule-A
BEGIN
    IF   BUSINESS_TRAVELER.TOTAL_DISTANCE >= 10000
           OR
         BUSINESS_TRAVELER. TOTAL_TRAVEL_NO >= 20
    THEN  0.2          // DISCOUNT RATE을 나타냄
END
```

(c)

(GROUP 명령어의 문법)

```
GROUP  <groupname>
BEGIN
GOAL <typename>
RULE <rulename>[{,<rulename>}]
END
```

(b)

(GROUP 명령어의 작성 예)

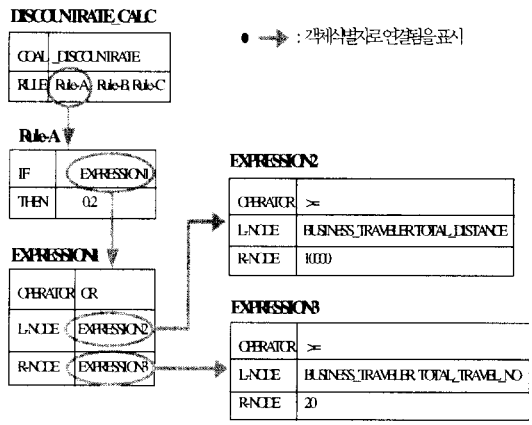
```
GROUP  DISCOUNTRATE_CALC
BEGIN
GOAL    _DISCOUNTRATE
RULE    Rule-A, Rule-B, Rule-C
END
```

(d)

〔그림 8〕 규칙을 표현하기 위한 명령어의 문법과 작성 예

명령어가 필요하다. 이를 위하여 RULE 명령어와 GROUP 명령어를 사용하며, [그림 8]의 (a)와 (b)는 이러한 두 명령어의 문법을 나타낸다. RULE 명령어는 개별 규칙에서 조건부와 행위부를 표현하는 IF-THEN 형태의 구조를 가지며, GROUP 명령어는 해당 그룹의 목표와 그 그룹에 속하는 규칙들을 표현한다.

[그림 8]의 (c)는 RULE 명령어를 이용하여 개별 규칙을 정의한 작성 예이며, 비즈니스 여행객의 총 여행거리가 10000마일 이상이거나 탑승 횟수가 20번 이상이면 할인율이 20%이다 라는 규칙을 표현한다. 이때, 작성 예에서와 같이 IF 부분의 표현식에는 FRAME 명령어를 이용하여 정의한 사실에 나타나는 프레임의 이름과 슬롯 및 패싯의 이름을 사용할 수 있으며, 참 거짓을 판별할 수 있는 형태가 되어야 한다. 또한 THEN 부분은 IF 부분의 표현식이 참이 되었을 때 반환할 값을 나타낸다. 한편, [그림 8]의 (d)는 GROUP 명령어를 이용하여, 정의된 규칙들을 그룹화하는 작성 예이다. GROUP 명령어 내에 나타나는 GOAL 부분은 해당 그룹의 목표를 나타내며, 규칙들의 행위부에 나타나는 값의 타입이 된다. 그리고 RULE 부분에는 그 그룹에 속하는 규칙들을 나열해 주게 된다. 위의 예는 항공권 할인율을 알고자 하는 목표를 가지는 규칙들의 그룹을 나타내며, GOAL은 앞의 사실의 표현 부분에서 TYPE 명령어를 이용하여 정의한 _DISCOUNTRATE이 되고, 이에 해당하는 규칙들은 Rule-A, Rule-B, Rule-C가 있음을 나타낸다. 이와 같이 RULE 명령어와 GROUP 명령어를 이용하여 규칙 형태의 지식을 지식 스크립트 파일 내에 표현할 수 있으며, 이러한 규칙들은 사실들과 마찬가지로 스크립트 구문해석기에 의해 구문해석이 되어 그림 9와 같은 형태로 데이터베이스 내에 저장된다.



〔그림 9〕 데이터베이스 내에 저장된 규칙의 예

[그림 8]의 (c)에서 RULE 명령어에 의해서 정의된 규칙은 하나의 규칙 프레임 ([그림 9]에서 Rule-A 프레임)을 생성하며, 규칙의 조건부에 나타내는 표현식은 표현식 프레임 ([그림 9]에서 EXPRESSION1, EXPRESSION2, EXPRESSION3 프레임)이라는 별도의 프레임에 저장된다. 모든 표현식 프레임들은 반드시 하나의 연산자만을 포함하게 되는데, 이렇게 함으로써 표현식 프레임은 하나의 연산자에 대한 연산만을 표현하는 연산의 기본 단위가 된다. 이를 위하여, 정의된 규칙을 데이터베이스에 저장하는 과정에서 조건부에 나타나는 표현식을 하나의 연산자만을 포함하는 여러 개의 하위 표현식으로 나누게 되며, 이렇게 나누어진 하위 표현식들은 각각 하나씩의 표현식 프레임에 저장되게 된다. 표현식 프레임에서 OPERATOR 슬롯은 연산자가 무엇인지는 나타내며, L-NODE슬롯과 R-NODE슬롯은 각각 연산자를 중심으로 왼쪽과 오른쪽의 피연산자를 나타낸다. 이와 같이 각각의 규칙은 하나의 규칙 프레임과 관련된 여러 개의 표현식 프레임에 나뉘어 저장되게 되는데, 이들간의 관계는 [그림 9]에서와 같이 객체식별자를 이용하

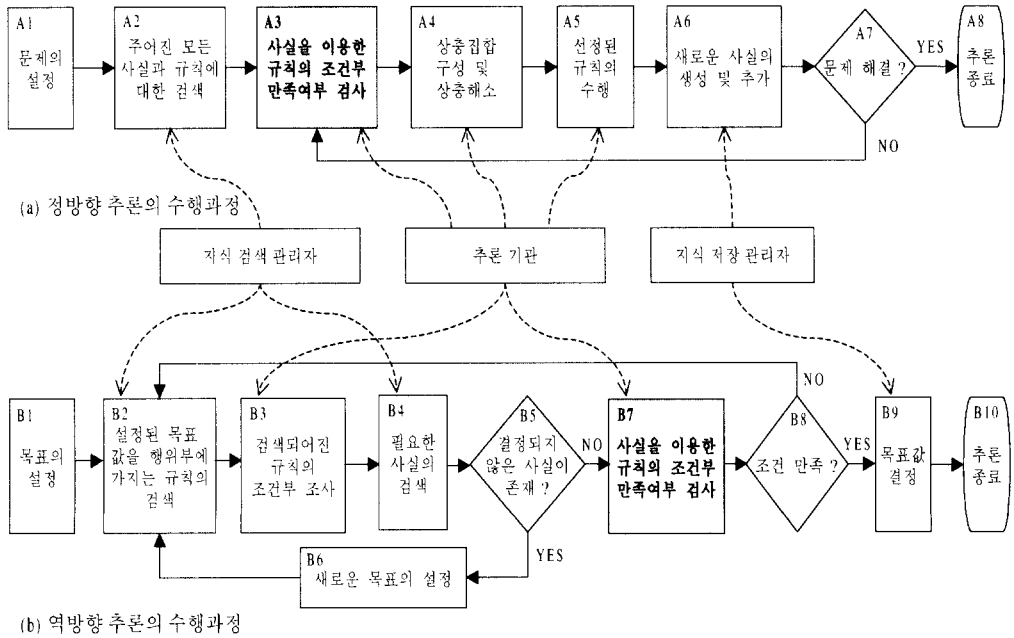
여 연결된다. 한편, GROUP 명령어에 의해서 정의된 그룹은 각각 하나씩의 프레임을 생성하게 되는데 ([그림 9]에서 DISCOUNTRATE_CALC 프레임), 그룹에 속해있는 각각의 규칙들 역시 객체식별자를 이용하여 연결되게 된다.

이와 같은 데이터베이스 내의 사실과 규칙의 저장구조는 추론시에 지식의 검색을 신속하게 실행할 수 있도록 지원하며, 추론의 수행을 위해서는 [그림 5]에 나타나 있는 바와 같은, 지식을 데이터베이스 내에 저장하기 위한 클래스 구조 내에 정의되어 있는 연산들을 이용하게 된다. 다음 장에서는 이러한 지식베이스의 저장구조와 연산구조들을 이용한 추론의 수행에 대하여 알아본다.

5. 지식베이스 모형을 이용한 규칙기반 추론의 수행

전문가시스템에서 규칙과 사실을 이용하는 대표적인 추론 방법에는 정방향 추론(forward chaining)방법과 역방향 추론(backward chaining) 방법이 있다. 이 중 정방향 추론은 주어진 사실로부터 출발하여 저장된 규칙들을 검색하고 이 중 조건부가 만족되는 규칙을 수행(firing)함으로써 결론에 도달하는 전략을 사용하며, 역방향 추론은 특정 목표에서 출발하여 해당 목표를 행위부에 가지는 규칙의 조건부가 만족하는지를 역으로 점검하는 전략을 사용한다[4, 30]. 이러한 두 가지 추론 방법의 수행과정과 이 논문에서 제시한 지식베이스를 이용하여 추론을 수행할 때 추론의 각 단계에 관여하는 OOKS [그림 4]의 구성요소를 나타내 보면 [그림 10]과 같다.

그림 10에서와 같이 두 가지 추론 방법은 서로 다른 수행과정을 가지고 있는데, 이 중 어떤 추론 방법을 사용할 것인가는 전문가시스템에서



[그림 10] 규칙기반 추론의 수행과정

해결하고자 하는 문제의 성격에 따라서 결정된다. 하지만 두 가지 추론방법 모두 알려진 사실들을 규칙에 적용하여 조건부에 대한 만족여부를 판별하기 위한 매칭 (matching)을 수행하게 되며 ([그림 10]의 A3, B7로 그림 4 OOKS의 추론기관에 의해 수행), 이를 위해서 추론에 이용되는 사실과 규칙에 대한 검색을 수행한다 ([그림 10]의 A2, B2, B4로 [그림 4] OOKS의 지식검색 관리자에 의해 수행). 그리고 이러한 사실과 규칙을 검색하고 규칙의 만족여부를 검사하는 과정이 추론 시간의 대부분을 차지하게 된다. 따라서 지식베이스가 이러한 부분을 효율적으로 지원할 수 있다면 추론의 성능을 크게 향상시킬 수 있다.

앞에서 설명한 바와 같이 RULE 명령어에 의해서 정의된 개별 규칙들은 [그림 5]의 RuleFrameType 클래스의 인스턴스로서 데이터베이스 내에 저장되게 되며, 저장된 규칙의 조건부의 만족여부를 판별하기 위하여 RuleFrameType 클래스의 연산인 Evaluate_condition()을 이용하게 된다. 이 연산이 수행되면 내부적으로 해당 규칙의 조건부를 나타내는 표현식 프레임을 검색하고 ([그림 9]참조), 해당 표현식 프레임의 좌우측 노드 값에 대한 비교연산을 수행하게 된다. 이 때, 지식베이스 내에 존재하는 규칙의 내용은 이미 모두 구문해석이 되어 저장되어 있으므로 추론의 수행시에 별도의 구문해석이 필요 없으며, 그 내용은 모두 객체식별자로 연결되어 있으므로 관련 표현식 프레임으로 빠르게 접근할 수 있다. 따라서, 추론의 수행시에 필요한 규칙에 대한 내용의 파악이 신속하게 이루어 질 수 있다. 반면에 만일 구문해석을 행한 규칙을 관계형 데이터베이스 내에 저장하려고 한다면, 정규화의 필요성 때문에 하나의 규칙을 여러 개의 테이블에 나누어 저장하여야 하며, 이는 추론의 수행시에 규칙의 내용을 파악하기 위해 많은 수의 조인 연산을 필요로 하기 때문에 추론속도가 현저하게 저하되게 된다.

한편, 규칙의 만족여부를 판단하기 위하여 필요한, 조건부에 나타나게 되는 변수들의 값은 저장되어 있는 사실들을 이용하여 결정하여야 하는데, 이를 위해서는 정의된 사실로의 접근이 쉽고 빠르게 이루어질 수 있어야 한다. 예를 들어, [그림 8]의 (c)에 있는 규칙의 조건부는 BUSINESS_TRAVELER 프레임의 인스턴스들의 TOTAL_DISTANCE 슬롯과 TOTAL_TRAVEL_NO 슬롯의 값을 필요로 한다. 앞에서 설명한 바와 같이 FRAME 명령어에 의해서 정의된 사실들은 그림 5의 FactFrameType 클래스를 이용하여 저장되게 되므로, BUSINESS_TRAVELER 프레임의 인스턴스로 접근하기 위하여 FactFrameType 클래스의 연산인 Get_instances()를 이용하게 된다. 이 때, 특정 프레임과 그 인스턴스들과의 관계는 객체식별자를 이용하여 연결되어 있으므로 ([그림 7] 참조), 해당 인스턴스로의 접근 또한 신속하게 이루어지게 된다. 그리고 FactFrameType은 FrameType 클래스로부터 프레임 조작을 위한 기본적인 연산들을 계승 받게 되므로 찾아진 인스턴스들의 TOTAL_DISTANCE 슬롯과 TOTAL_TRAVEL_NO 슬롯의 값을 구하기 위하여, FrameType 클래스의 연산인 Get_slot_value(TOTAL_DISTANCE)와 Get_slot_value(TOTAL_TRAVEL_NO)를 이용할 수 있다. 이와 같이 추론의 수행시 규칙의 만족여부를 판단하기 위해 필요한 사실들은 지식베이스가 제공하는 연산구조를 이용하여 쉽고 빠르게 접근될 수 있다.

지식을 표현하는 규칙과 사실들을 이 논문에서 제안하는 지식베이스 모형을 이용하여 저장관리함으로써 추론의 수행시에 얻을 수 있는 이점은 다음 두 가지로 요약할 수 있다. 첫째, 규

칙과 사실들을 저장하고 있는 관련된 프레임들을 객체식별자를 이용하여 연결시켜줌으로써 추론의 수행시 규칙의 내용 파악과 필요한 사실의 검색을 빠르게 수행할 수 있다. 둘째, 규칙과 사실을 다루기 위한 연산들을 지식베이스가 제공해 줌으로써 지식저장을 위한 객체지향 데이터베이스의 내부 메커니즘을 다 이해하지 않더라도 추론의 수행을 위한 지식베이스의 기능을 쉽고 효율적으로 활용할 수 있다.

6. 결론

전통적인 전문가시스템에서 지식을 영속적 저장장소에 저장 관리하기 위하여 사용되어온 파일 시스템은 추론의 수행 전에 지식을 시스템의 메모리에 적재하여야 하기 때문에 지식이 방대해 지는 경우 지식을 메모리에 모두 적재하지 못하며, 저장된 지식을 여러 명의 사용자가 동시에 접근할 수 없는 한계를 가지고 있다. 이를 극복하기 위한 방법으로는 전문가시스템의 지식을 데이터베이스를 이용하여 관리하는 방법이 효과적인데, 관계형 데이터베이스를 이용하여 지식을 저장하는 경우에는 지식을 모두 2차원적인 테이블 형태로 표현하여야 하기 때문에 지식이 가지는 의미 (예를 들어, 지식의 계층적 구조나 지식간의 연결관계)를 데이터베이스 내에 적절하게 표현하기가 어려우며, 여러 테이블에 나누어져 있는 규칙이나 사실들을 추론에 이용하기 위해서는 시스템 자원을 많이 필요로 하는 조인 연산을 다수 수행하여야 하기 때문에 추론시간이 오래 걸리게 된다. 이 논문에서는 파일 시스템과 관계형 데이터베이스의 이러한 한계점을 극복하기 위하여 객체지향 데이터베이스를 이용하여 전문가시스템에서 사용하는 지식을 저장 관리하기 위한 모형을 제시하고, 제시된 지식베이스 모형을 이용하여 추론을 수행하는 경우에 가질 수 있는 장점에 대하여 논의하였다.

이 논문에서 제안하고 있는 객체지향 데이터베이스를 이용한 전문가시스템을 위한 지식베이스 모형인 OOKS는 다음과 같은 특징을 가지고 있다.

첫째, 지식 표현을 위한 프레임이 가지고 있는 구조를 데이터베이스 모델에 그대로 표현함으로써 전문가의 지식이 가지는 의미를 보다 직관적인 구조로 표현할 수 있다. 특히, 프레임이 가지는 계승을 통한 지식의 계층적 구조와 프레임간의 연결관계를 데이터베이스 내에 표현하기 위하여 프레임의 저장주소인 객체식별자를 이용함으로써 특정 프레임으로부터 상위 프레임과 하위 프레임 또는 인스턴스 프레임으로의 이동이 연결된 객체식별자를 따라 이루어지게 된다. 따라서 추론을 위한 사실을 검색하는데 있어서 관련된 다른 프레임으로의 이동에 시간이 거의 필요하지 않게 된다.

둘째, 추론을 위한 규칙을 데이터베이스에 저장하는 시점에 규칙을 표현하는 지식 스크립트 파일을 구문해석하여 조건부에 나타나는 표현식을 각각 하나의 연산자만을 가지는 여러 개의 하위 표현식으로 나누어 프레임 구조 내에 일관성 있는 방법으로 저장함으로써 추론의 수행시에 데이터베이스에 저장되어 있는 규칙을 별도의 구문해석 없이 바로 사용할 수 있다. 특히, 하나의 규칙을 표현하는 여러 개의 표현식 프레임과 추론시 함께 사용되는 규칙들을, 사실을 저장할 때와 마찬가지로 객체식별자를 이용하여 연결함으로써 추론시 관련 규칙을 찾고, 찾아진 규칙의 내용을 파악하는 과정이 빠르게 수행될 수 있다.

셋째, 지식을 데이터베이스 내에 저장하고, 추론시 저장된 지식을 다루기 위한 연산들을 객체지향 데이터베이스 내에 정의하여 캡슐화시킴으로써 지식베이스 내에 저장되어 있는 지식에의 접근과 이용을 용이하게 한다. 이는 객체지향 데이터베이스를 이용한 지식베이스의 구현부분과 그 밖의 부분을 분리시킴으로써 지식베이스 사용자들에게 지식베이스 내의 지식의 복잡한 저장구조는 신경쓸 필요 없이 지식베이스의 기능을 활용할 수 있도록 한다.

이 논문에서 제시한 객체지향 데이터베이스를 활용한 전문가시스템의 지식베이스 모형과 쉘 구조는 상용 객체지향 데이터베이스인 ObjectStore [24]를 이용하여 원형시스템이 구현되었으며, 이 원형시스템은 ObjectStore가 지원하는 C++ 프로그래밍 언어를 이용하여 Windows NT를 기반으로 개발되었다. 향후 연구과제로는 현재 구축되어 있는 지식베이스 모형을 바탕으로 보다 정교한 규칙 표현 방법과 퍼지 (fuzzy) 논리 등을 표현할 수 있는 방안에 관한 연구가 있으며, 사례 기반 추론 (case-based reasoning)이나 신경망 (neural network) 등을 이용한 추론기능의 확장에 관한 연구가 있다.

참 고 문 헌

[1] 오선영, 백두권, "객체 중심 측면 모델에 의한 KB/DB 통합 방법론", *데이터베이스 저널*, 1, 1993, 3-24

[2] 유석인, *데이터베이스를 첨가한 전문가시스템의 구축에 관한 연구*, 과학기술처, 1991

[3] 이재규, 송용욱, 권순범, 김우주, 김민용, *UNIK를 이용한 전문가시스템의 개발*, 법영사, 1996

[4] 이재규, 최형림, 김현수, 서민수, 주석진, 지원철, *전문가시스템 원리와 개발*, 법영사, 1996

[5] 전재호, "다수의 전문가 체제와 관계형 데이터베이스의 결합을 위한 조정에 관한 연구", *한국과학기술원 석사학위논문*, 1992

[6] Al-Zobaidie, A. and Grimson, J. B., "Expert Systems and Database Systems: How can they serve each other?", *Expert Systems*, 4, 1987, 30-37

[7] Babiker, E., "A Model for Reengineering Legacy Expert Systems to Object-Oriented Architecture", *Expert Systems With Applications*, 12, 1997, 363-371

[8] Bic, L. and Gilbert, J. P., "Learning from AI: New Trends in Database Technology", *IEEE Computer*, March 1986, 44-54

[9] Brodie, M. and Mylopoulos, J., *On Knowledge Base Management System: Integrating Artificial Intelligence and Database Technologies*, Springer-Verlag, 1986

[10] Chew, G., Schneider, M., and Kandel, A., "Use of Object-Oriented Structures to Represent Knowledge in Expert Systems", *Proceedings of the 1993 ACM/SIGAPP*, 1993, 498-503

[11] Date, C. J. *An Introduction to Database Systems (6th ed.)*, Addison-Wesley, 1995

[12] Giarratano, J., *CLIPS Users Guide Version 6.0*, NASA Lyndon B. Johnson Space Center, August 1994

[13] Gordin, D. and Pasik, A., "Set-Oriented constructs: From Rete Rule Bases to

Database Systems", *ACM SIGMOD Record*, 20, June 1991, 60-67

[14] Hammer, M. and McLeod, D., "Database Description with SDM: A Semantic Database Model", *ACM Transactions on Database Systems*, 6, September 1981, 351-386

[15] Higa, K. et al., "Object-Oriented Methodology for Knowledge Base/Database Coupling", *Communications of the ACM*, 35, June 1992, 99-113

[16] Hull, R. and King, R., "Semantic Databse Modeling: Survey, Applications, and Research Issues", *ACM Computing Surveys*, 19, September 1987, 201-260

[17] Kerschberg, L., "Expert Database Systems: Knowledge/Data Management Environments for Intelligent Information Systems", *Information* Systems, 15, 1990, 151-160

[18] Kim, Won, *Introduction to Object-Oriented Databases*, The MIT Press, 1991

[19] Kitsuregawa, M. and Tanaka, H., *Database Machines and Knowledge Base Machines*, Kluwer Academic Publishers, 1988

[20] Lee, J. K. et al., "Automatic Rule Generation by the Transformation of Expert Diagram: LIFT", *International Journal of ManMachine Studies*, 32, 1990

[21] Lee, J. K. and H. R. Choi, "Scheduling Shipbuilding Using Constraint Directed Graph Search: DAS-ERECT", *Intelligent Systems in Accounting Finance and Management*, 3, 1994, 111-125

[22] Leung, K. S. and Wong, M. H., "An Expert System Shell Using Structured Knowledge: An Object-Oriented Approach", *IEEE Comput.*, 23, March 1990, 38-47

[23] Neuron Data Inc., *Nextpert Object Version 3.0 Users Guide*, December 1993

[24] Object Design Inc., *ObjectStore User Guide*, Release 4.0, June 1995

[25] Owrang, M. M. and Grupe, F. H., "Database Tools to Acquire Knowledge for Rule-Based Expert Systems", *Information and Software Technology*, 39, 1997, 607-616

[26] Pigford, D. V. and Baur, G., *Expert Systems for Business: Concepts and Applications*, MA:Boyd & Fraser, 1990

[27] Potter, W. D., "KDL-Advisor: A Knowledge /Data Based System Written in KDL", *Proceedings Twenty-First Annual Hawaii International Conference on System Sciences and Knowledge-Based Systems Track*, January 1988, 319-328

[28] Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, *Object-Oriented Modeling and Design*, NJ:Prentice-Hall, 1991.

[29] Rundensteiner, E. A., "The Role of AI in Databases versus the Role of Database Theory in AI", *Artificial Intelligence in Databases and Information Systems*, 1990, 233-252

[30] Turban, E., *Decision Support and Expert Systems: Management Support Systems* (3rd ed.), New York: Macmillan

[31] Vadaparty, K., "Memory-Mapped Architecture", *Journal of Object-Oriented Programming*, 8, Oct. 1995, 18-26

[32] Waterman, D., A *Guide to Expert Systems*, Addison-Wesley, 1986

[33] Wu, J. S. et al., "Enhancement of an Object-Oriented Expert System for Contingency Load Transfer of Distribution Systems", *Electric Power Systems Research*, 42, 1997, 87-94

[34] Xu, D., "Towards an Object-Oriented Logic Framework for Knowledge Based Systems", *Knowledge-Based Systems*, 10, 1998, 351-357

[35] Yang, H. L., "A Simple Coupler to Link Expert Systems with Database Systems", *Expert Systems With Applications*, 12, 1997, 179-188

[36] Znidarsic, A., J. Kocijan, and A. Skobe, "A Process Plant Simulator Developed within an Object-Oriented Expert System Shell", *Computers in Industry*, 35, 1998, 207-221