碩士學位論文

關係型 데이터베이스의 情報檢索을 위한 推論 인터페이스 構築

濟州大學校 大學院

情 報 工 學 科

文 哲 煥

1996年 12月

關係型 데이터베이스의 情報檢索을 위한 推論 인터페이스 構築

指導敎授 金壯亨

文 哲 煥

이 論文을 工學 碩士學位 論文으로 提出함

1996年 12月

文哲煥의 工學碩士學位 論文을 認准함

審査委員長 安 晃 午 印
委 員 郭 鎬 榮 印
委 員 金 壯 亨 印

濟州大學校 大學院

1996年 12月

The Construction of Inference Interface for Information Retrival with RDB

Seok-Hwan Moon
(Supervised by professor Jang-Hyung Kim)

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF ENGINEERING

DEPARTMENT OF INFORMATION ENGINEERING
GRADUATE SCHOOL
CHEJU NATIONAL UNIVERSITY

1996. 12.

목 차

SUMMARY

We studied on the construction of inference interface for information retrival with relational database system. The query language of relational database system is reinforced and usefully applied to the inference engine of first-order logic.

If we can add the inferencing ability of the logic base to the relational database system, the system becomes more powerful and useful. For this purpose, we studied an inference engine which applicable to the relational database system using first-order logic.

We constructed on intellectual interface for relational database using the proposed inference engine.

Lineage consulting prototype is developed using the proposed method with the prolog language.

As the results, the user's query about relational database could be processed by intellectual manner.

Ⅰ. 서론

정보통신의 발달로 인해 새로운 정보의 요구가 증가하면서 날로 정보의 양이 많아지고 그에 따라 데이터베이스의 구조도 복잡하고 방대하게 변화하고 있다. 이러한 대량의 정보들은 급속한 변화의 흐름 속에서 동적으로 변환되어가는 특징을 보이고 있다. 일반적으로 데이터베이스를 구축하는 목적은 많은 데이터 중에서 자신이 원하는 데이터를 쉽고 빠르게 찾는 데 있다. 이런 상황에서 데이터베이스에 보관되어있는 정보들 중에서 사용자가 필요로 하는 정보를 보다 효율적이고 빠르게 추출해내는 방법에 대한 연구가 활발해 지고 있다.

그런 노력들은 데이터베이스와 인공지능의 결합을 꾀하는 연구로 활발히 진행되어 왔으며, 이러한 연구분야 중의 하나가 데이터 베이스에서 발견된 지식을 사용하여 지식기반 시스템에 활용하고자 하는 시도이다(Shekher, 1994). 대부분의 이런 시도들은 추론엔진(inference engine)의 지속성을 활용하기 위하여 지식베이스를 사실베이스와 규칙베이스로 나누어 이중 사실베이스를 보조 기억장치에 저장하여 추론과정에서 이용하는 형태이다.

또한 데이터베이스 시스템에서의 데이터 모델들은 모델이론(model theory)을 기반으로 보다 효율적이고 구조적으로 표현할 수 있도록 연구되고 있으며, 지식 표현 시스템에서의 데이터 모델은 데이터를 추론(inference)하기에 적합한 형태의 변수구조에 중점을 두고 개발되어 왔다(Stonebraker, 1983)(Tsichritzis, 1982).

데이터 베이스 관리 시스템은 그 기능과 성과면에서 상당한 발전을 해 온 것이 사실이다. 그러나 정보사회의 복잡하고 다양한 변화는 단순히 데이터에 대한 접근 역할뿐만 아니라 저장된 정보로 부터 새로운 사실을 추론하여 의사결정을 내리는데 도움을 줄 수 있는 정보로서의 가치를 점점 더 필요로 하고 있다. 이와같은 필요에 의해 외부 데이터 베이스와 인공지능 언어를 결합하여 추론기능을 보유한 시스템으로 확장하려는 시도를 하였는데 데이터베이스에서의 로직을 응용하거나(Reiter,1978)(Waker, 1987)(Grant, 1988), 관계형 데이터베이스와 프롤로그의 접목(Ceri, 1986), 관

계형 데이터베이스에서의 개념기반 데이터의 정형화(Han, 1991), 지식발견 기술에 의한 지적인 질의 답변에 관한 연구(Han, 1996)등으로 최근까지 계속적으로 진행되고 있다.

관계형 데이터 베이스와 지식베이스라는 서로 다른 두 환경을 접목하려는 연구의 일환으로 현재의 지식베이스 기술들을 관계형 데이터베이스에 접목하여 인공지능 언어에 의한 데이터 관리의 가능성이 대두되었다(Bodie, 1984)(Frank, 1984)(Naqvi, 1984). 수많은 양의 데이터를 기반으로 새로운 사실을 추론해내려는 시도(Missikoff, 1984)(Sciore, 1984)가 있었고, 이런 대부분의 시도가 관계형 데이터베이스에서의 질의어로서 프롤로그의 장점과 단점을 보여 주었다. 또한 이러한 두 환경 상의 상호관계에 있어서의 중요한 효율성에 관한 연구도 있었다(Chakravarthy, 1984).

일반적으로 데이터베이스의 접근은 질의문의 형태를 통해 새로운 사실들이 만들어지는 것보다는 질의를 처리하는 동안 기존의 사실들로 부터 더 많은 추론을 해내는 프롤로그를 통하는 것이 효과적이다. 이는 데이터베이스에 대해서 통계적 추론(statistical inferencing)을 하기 위해서는 특별한 데이터 조직을 만들어야 하고 그만큼 방대해지는 데이터의 조직을 감수해야만 하기 때문이다. 이러한 이유에서라도 경험적 추론(heuristic inference)을 위한 인공지능의 기법과 데이터베이스의 결합이 요구되는데, 그러한 접근의 방식으로는 추론이 가능한 지식 베이스를 구축하는 지식베이스 관리 시스템(knowledge base management system: KBMS)과 기존의 관계 데이터베이스 시스템에 추론기능만을 보강하는 전위 시스템(front-end system)이 있을 수 있다.

본 논문에서의 접근은 전위시스템을 이용하는 것인데, 이는 기존의 구축된 관계형 데이터베이스에 대한 활용도를 높인다는 의미도 있다. 즉, 관계형 데이터베이스는 대량의 자료를 효율적으로 다룰수 있고 데이터의 표준화와 정확성등의 중요한 장점을 지녔지만 지식이라는 새로운 객체의 성질을 표현하기위한 방법이나 이를 활용하는 추론에 대한 능력이 부족한 것이 단점으로 지적되므로 이를 보완하는 기능을 추가한 것이다. 이는 데이터베이스와 지식베이스를 포함한 전위 시스템 사이에서의 효율적인 인터페이싱을 구축함으로써 해결할 수 있다. 이를 위한 전위시스템과 외부 데이터베이스 사이의 인터페이싱은 관계형 데이터베이스에서의 질의와 관계된 사실들과 프롤

로그 술어 사이의 패턴매칭(pattern matching)연산을 이용하여 사상되도록 하는 것으로 새로운 패러다임(paradigm)을 만드는 대신에 이전에 검색된 정보를 재사용하는 것이다. 즉, 가능한한 추론에 적합한 언어인 프롤로그의 의미론적 접근을 유지하면서 관계형 데이터베이스의 사실을 검색하기 위해 질의문인 SQL에 최종적으로 적용될 검색자료를 전위시스템에서의 추론을 위한 모듈로 포함된 프롤로그언어를 이용, 적절한 형태로 추론하게한다. 또한 프롤로그를 사용할 줄 모르는 사용자들도 쉽게 자료를 추론을 통해 볼 수 있도록 하기 위하여 전위(front-end)와 후위(back-end) 시스템은 호스트 언어인 비주얼베이직을 사용한 GUI형태의 사용자 인터페이스로 구축한다.

결국 윈도우즈 프로그래밍 언어에서의 객체화 과정을 통한 인공지능 언어와 대용량의 관계형 데이터베이스 시스템과의 동적결합에 의한 추론 인터페이스의 구현은 빠르게 한계를 극복해가는 시스템 간의 다양한 개발 가능성을 제시하는데 의의가 있다.

본 논문의 구성은 다음과 같다.

Ⅱ 장에서는 추론 인터페이스의 개요와 기존 방식, 추론엔진의 구축을 위한 지식베이스 구성요건을 기술하고 추론엔진에서의 지식베이스 구축에 적용되는 술어로직의 표현방식에 대해 설명하며 추론 인터페이싱 구조와 윈도우즈 환경에서의 사용자 인터페이스의 개발 필요성에 관하여 기술한다.

Ⅲ 장에서는 추론 인터페이스 시스템의 설계방식을 제시하면서 전위 시스템의 설계에 관한 사항과 프롤로그를 이용한 추론엔진의 기본 설계 알고리즘을 기술하고 구현과정의 사용자 그래픽 인터페이스를 제시한다.

Ⅳ 장은 추론 인터페이스 구현에 관한 부분으로 계보자문 프로토타입을 위한 개인정보 데이터베이스를 구축하고, 로직기반 추론엔진에 적용하여 계보자문 시스템을 구현함으로써 사용자 위주의 그래픽 인터페이스를 다양하게 개발할 수 있게 하여, 다양한 형태의 추론 시스템을 구축하기 위한 프로토타입을 제시한다.

Ⅴ 장은 이에 대한 평가를 통해 본 연구의 구현내용 및 결과와 이에대한 보완문제를 거론한다.

Ⅵ 장은 최종적으로 본 연구에 대한 결론을 내린다.

Ⅱ. 추론 인터페이스

본 장에서는 추론 인터페이스의 개요와 접근방식에 대해서 살펴보고, 추론 인터페이스의 핵심인 지식베이스 구성요건을 기술한 후, 로직 베이스로서의 프롤로그 술어 로직 표현형태와 추론 인터페이스의 내부구조에 대한 설명 및 현 시점에서 강력히 대두되는 손쉽고 빠르게 이해되는 사용자 위주의 그래픽 인터페이스의 개발 필요성에 대해 살펴 보기로 한다.

1. 추론 인터페이스의 개요

추론 인터페이스는 사전적 의미로 지적 대화를 위해 인공지능 컴퓨터가 자연언어로 대화하거나 음성 및 영상 정보를 직접적으로 이용할 수 있는 방법을 말하며, 인간의 사고, 문제해결, 그리고 학습등의 인지적 과정을 거쳐서 어떻게 인간처럼 지식을 표현 할 수 있고(knowledge representation), 어떻게 적절한 문제해결의 길을 찾게 하며(search), 어떻게 인간처럼 자연언어로 의사소통을 할 수 있게 할 것인가(natural language dialogue), 또는 어떻게 인간처럼 추론(inference)을 할 수 있게 할 것인가에 대한 문제 해결 부분을 말함이다. 이는 인지과학과 전산과학이 함께 만나 역할 분담을 하게되는 특정한 부분이다(김승광, 1991)(EDPS연구회, 1993).

본 논문에서는 이러한 광범위한 분야 중 어떻게 인간처럼 추론을 하게 할 것인가를 탐구하는 방법론적인 접근으로 컴퓨터와 사용자 간의 대화방식에 의거, 기존의 자료를 접근하기 위한 동적 추론엔진의 설계와 사용자가 쉽게 이해할 수 있는 인터페이스의 구축을 목적으로 한 시도이다. 구조적으로는 기존의 지식베이스 개발시 한계상황이됐던 메모리 부담을 줄이면서 추론기능을 확보하는 방안을 모색하고, 데이터베이스 관리 시스템과 지식 베이스 관리 시스템의 상호결합을 통해 서로의 장점을 이용할 수 있게 하여 데이터 자원을 자연스럽게 공유할 수 있도록 하기 위한 것이다. 이렇게 외부 데이터 자원을 공유한 지식 기반 시스템을 갖춤으로써 사용자의 입장에

서 보면 스스로 추론하는 인터페이스를 만나게 된다.

2. 추론 인터 페이스를 위한 접근방식

추론 인터페이스는 일반적인 전문가 시스템에서의 지식 베이스를 구축하는 방식을 활용하고 지식 베이스를 호스트 언어에서 호출하는 방식을 적용한다. 이는 기존의 지식 기반 시스템에서 지식베이스가 복잡해지고 대형화됨으로써 점점 메모리 상에서 관리하기 어려운 상황을 극복하기 위한 하나의 대안이 된다. 이에 대한 대안을 제시하기에 앞서 기존의 방식을 살펴보면 크게 지식 베이스 관리 시스템과 데이터 베이스 기반 전위 시스템으로 나눌 수 있다. 추론 엔진을 통한 지식 베이스 관리 시스템은 Fig. 1에서 나타낸 바와 같이 전통적으로 사용자의 질의를 받아서 사실 베이스와 규칙 베이스로 이뤄진 지식 베이스를 가지고 있는 추론엔진을 통한 작업 메모리에서의 추론과정을 거친 후 다시 사용자에게 질의에 대한 결과를 보여주는 방식으로 구성된다. 이와같은 지식 베이스 관리 시스템은 사용자의 질의를 처리하기 위한 추론엔진 부분을 자체적으로 작업 메모리에서 처리하기 때문에 작업 메모리의 크기에 따라 지식 베이스를 구성하는 한계성을 지니게 된다. 또한 이를 개선한 방식인 데이터베이스 기반 전위 시스템은 Fig. 2에서 나타낸 바 처럼 지식 베이스의 사실 베이스를 외부 데이터베이스화 시켜 메모리의 제약성을 감소시켰으나 사용자가 추론 질의어 방식을 습득한 상태에서는 질의어 작성에 어려움이 없으나 이에 익숙하지 못한 대부분의 사용자는 질의어 작성형식에 대한 제약을 받게된다. 이와 같은 사용자 질의형식의 제약성을 극복하고, 사실 베이스의 외부 데이터베이스화를 좀 더 효율적으로 개선하고자 하는 것이 본 논문의 주요 제안이다.

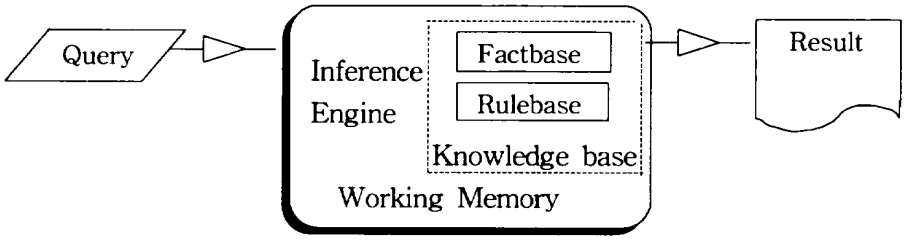


Fig. 1 Knowledge base management system

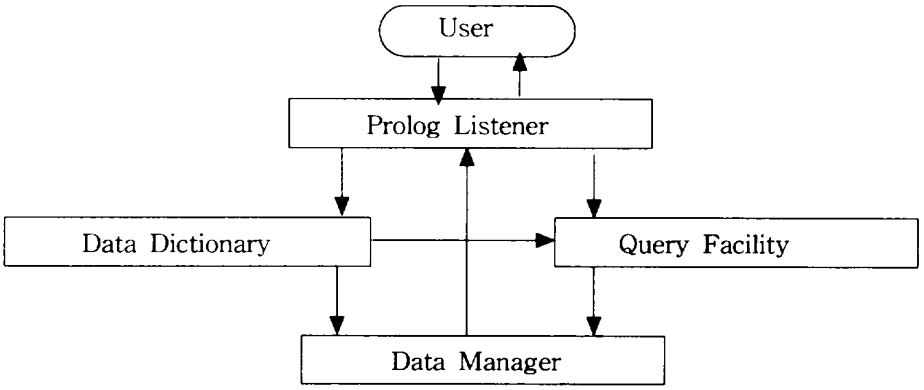


Fig. 2 Front-end system for Prolog database system

지식베이스를 이루기 위한 지식표현 기법으로는 술어로직(predicate logic)과 의미망(semantic network), 규칙(rule)선언, 프레임(frame)이 있는데 서로 장단점을 지니고 있다. 논리에 의한 지식의 표현은 대수학이나 논리학에서 지식을 정형화하기 위한 영역에 대해 적합한 장점이 있으나 지식베이스를 구성하는 사실에 대한 구성법칙이 부족해서 복잡한 구조를 표현하기 어렵다는 단점이 있다. 의미망은 지식을 망으로 연결하여 객체나 개념적 사건등을 노드(node)로 연결시켜 복잡한 인과관계를 추론하기에 자연스러운 표현의 장점이 있으나 지식베이스의 크기가 커지면 복잡해져 관리가 어려운 단점이 있다. 프레임 방식은 의미망의 한 부분으로 객체구조인 프레임과 객체의 속성인 슬롯(slot)에 대한 묘사에 중점을 두어 데이터와 프로시듀어를 하나의 구조로 묶을 수 있어 지식을 표현함에 있어 보다 자연스럽고 강력하다는 장점이 있지만 이 표현기법 역시 지식을 작성하기가 어렵다는 단점이 있다. 또한 가정(if)과 결론(then)에 의한 지식표현을 하는 규칙선언 기법은 자연어 방식의 애매성 처리에 효율적이라는 장점이 있으나 이 역시 너무 제한적 선언이라는 단점이 있다. 이런 장단점들로 인해 본 논문에서는 술어로직의 장점과 규칙선언의 장점을 혼합한 형태인 로직베이스를 추론 인터페이스의 중심모듈로 구성하고자 한다.

3. 추론 인터페이스와 지식 베이스

일반적인 실세계 지식의 표현에 있어 특정 분야를 지식베이스로 구축하기 위해서는 표현의 타당성과 추론의 적합성, 추론의 효율성, 지식획득의 효율성이 고려되야 한다. 표현의 적합성(representational adequacy)은 관계된 모든 지식을 적절히 표현할 수 있어야 하는 것이고, 추론의 적합성(inference adequacy)은 새로이 표현되는 지식에 대해서도 처리할 수 있는 능력을 가지는 것이다. 또한 추론의 효율성(inference efficiency)은 특정한 정보와 지식 구조를 결합시킬 수 있는 능력이고, 지식획득의 효율성(acquisitional efficiency)은 빠르게 새로운 정보를 획득할 수 있는 능력을 가지는 것이다. 이러한 지식표현의 특징들을 고려한 지식베이스의 구성을 위해서는 프로그램 자체에서 직접적인 제어를 할 수 있는 형태이어야 한다. 따라서 본 논문에서는 관계형 데이터베이스에 새로운 사실들이 사용자 인터페이스를 통해 직접적으로 조작되는 방식을 채택하고자 한다.

1) 지식베이스 구축 요소

지식베이스 생성시스템 모델(production system model)의 기본 구성 요소는 추론엔진이다. 추론엔진의 역할은 가정과 결론 형태의 규칙부와 현재 상태를 표현하는 사실들 간의 비교를 통해 매칭되는 규칙들을 결론부에서 결정을 내릴 수 있도록 한다. 본 논문에서 제안하는 추론 인터페이스의 지식베이스를 구성하기 위한 요건은 기존의 외부 데이터 저장 시스템과 추론엔진이다. 인공지능 언어인 프롤로그를 사용한 지식베이스의 추론엔진을 구축하여 외부 데이터베이스 저장 시스템과의 상호 데이터 교환은 물론 사용자 인터 페이스를 구축하는 호스트 언어와의 상호통신을 가능하도록 하는 것이 추론 인터페이스의 기본 구상이다.

2) 지식 베이스의 구성

추론 인터페이스의 구축을 위한 지식 베이스의 구성은 지식의 제한적 표현이 대부분으로 사실에 대한 정적인(static) 표현방식이고 술어로직를 적용해서 사실들을 처리하는 프로시저와 관계 질의어 및 의미론적 무결성을 검사하는 부분으로 구성된다.

또한 각각의 사실과 규칙은 문제영역에 대한 지식을 가정과 결론의 문장으로 구성하여 조건부에 만족하는 경우에 결론부를 수행한다. 규칙의 결론부는 새로운 사실을 데이터베이스에 추가하는 등의 조작에 관한 내용을 포함한다. 이 선언적 기법은 각 지식이 사용되어지는 범위에 관계없이 단 한번의 기록으로 유효하다. 이와 같은 점을 활용하여 본 논문에서의 지식 베이스의 설계는 객체지향적 프로그램 방식을 도입하여 집단적으로 같은 성질의 프로시저를 통합한 모듈화된 지식 베이스를 구축한다.

본 논문에서의 지식 베이스의 구성은 각 객체들 간의 집단적 계층구조에 대해 최상위 집단아래 하위 집단이 존재하고 그 하위집단아래 최하위 집단이 존재한다는 계층적 구조를 기반으로 한다. 또한 객체간의 유전성이 적용되어 어떤 객체의 속성을 찾을 경우 앞선 객체의 속성까지도 이용할 수 있게 한다. 데이터와 많은 명령문이 포함된 프로시저와 이를 제어하기 위한 규칙들에 대해 각각의 관계를 설정하여 표시하면 Fig. 3과 같다.

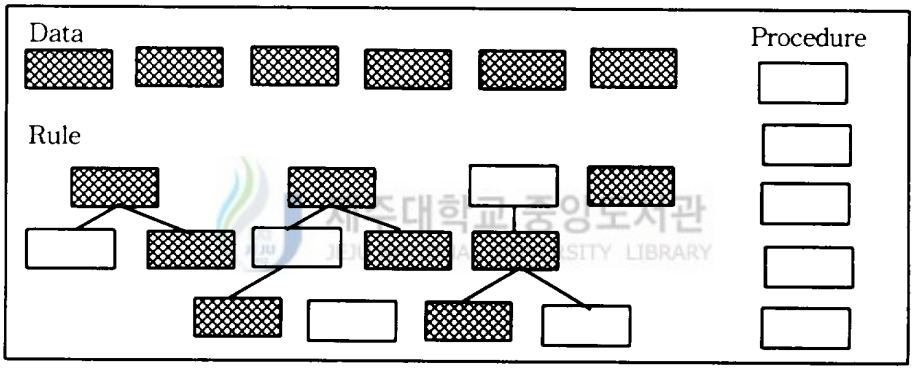


Fig. 3 The relation of data and procedure

4. 지식 베이스 구축과 술어로직

1) 술어로직의 표현

로직프로그래밍은 연역적(deductive)이고 비레코드(non-record) 중심의 정보를 가진 관계형 데이터베이스에 대한 확장으로 데이터베이스의 사실에 대한 스키마(schema),

메타데이터(meta-data), 변수(variables) 등을 다루는 프로그래밍 방식이다. 여기에서 기본을 이루는 것이 일차-순서 로직(first-order logic) 또는 술어로직(predicate logic)인데, 이는 외견상으로 관계형 데이터베이스를 위해 정형화된 언어로 적용될 수 있다. 이는 기존의 명제로직(propositional logic)이나 불린로직의 한계를 극복하기 위해 적용될 수 있으며(김형수, 1995), 정량자(quantifier)로서는 존재정량자(∃)와 전칭정량자(∀)를 사용한다. 이런 문장은 상당히 형식적이라서 맞는지 틀린 지를 확인할 방법이 없다. 따라서 실제 표현의 의미를 알려면 관계해석을 해야만 한다. 표현을 해석한다는 것은 항(term)으로 표기할 수 있는 객체의 집합인 전체집합을 정의한 후, 술어기호와 함수기호로 특정한 관계를 갖도록 결정하여 전체집합내에서 함수가 정의되도록 하는 것이다. 이러한 표현을 완전히 해석하게되면 객체 사이의 관계에 의하여 진, 위로 해석되는 것이다. 로직프로그래밍은 연역적이고 비레코드 중심의 정보를 가진 관계형 데이터베이스에 대한 확장으로 데이터베이스의 사실에 대한 스키마(schema), 메타데이터(meta-data), 변수(variables) 등을 다루는 프로그래밍 방식이다. 여기에서 기본을 이루는 것이 일차-순서 로직 또는 술어로직인데, 이는 외견상으로 관계형 데이터베이스를 위해 정형화된 언어로 적용될 수 있다. 술어의 일반적 형태는 아래와 같다.

$\forall x\ P(Z, x, y)$

$\forall (x,y,z)\ P(x,y,z) \Rightarrow P(R(x),y,R(z))$.

술어는 위의 예에서 처럼 술어기호(predicate symbol)인 P로 구성되며 콤마로 구분하고 괄호로 묶은 항(term)리스트를 인자(argument)로 한다. 여기서 항은 x,y,z같은 변수이거나, 이를 인자 리스트로 하는 P나 R과 같은 함수기호이다. 즉, 항은 전체집합을 표현하고, 술어는 이들 대상체 사이의 관계를 나타내는 것이다. 즉, 하나의 술어는 공식이다. 더 큰 공식은 공식을 논리 연결자(logic connective)로 연결해서 생성한다. 논리 연산자의 우선 순위는 부정(negation), 논리곱(conjunction), 논리합(disjunction), 함축(implication)의 순서로 처리되는데 괄호는 순위를 명확히 할 때나 우선 순위를 무시할 경우에 적용된다. 또한 공식 앞에는 정량자(quantifier)가 붙을 수 있는 데 우선 순위는 논리 연결자보다 낮다. 이는 기존의 명제로직(propositional logic)이나 불린로직의 한계를 극복하기 위해 적용될 수 있으며, 정량자(quantifier)로

서는 존재정량자(∃)와 전칭정량자(∀)를 사용한다. 앞의 예를 자연스럽게 표현하면 "어떤 객체 x에 대하여 객체Z는 x와 x로 D라는 관계가 있다"라고 읽고, 그 다음의 예는 "어떤 세 개의 객체 x, y, z 에 대하여 만일 관계 D에 x, y, z가 존재한다면 그 대상체는 S(x), y, S(z)이다."라고 읽는다. 이런 종류의 문장은 상당히 형식적이라서 맞는지 틀린 지를 확인할 방법이 없다. 따라서 실제 공식의 의미를 알려면 관계해석을 해야만 한다. 공식을 해석한다는 것은 항으로 표기할 수 있는 객체의 집합인 전체집합을 정의한 후, 술어기호와 함수기호로 특정한 관계를 갖도록 결정하여 전체집합 내에서 함수가 정의되도록 하는 것이다. 이러한 공식을 완전히 해석하게되면 객체 사이의 관계에 의하여 진,위로 해석되는 것이다. 이른바 삼단논법으로 알려진 관계, 즉 '어떤 공식 α와 $\alpha \Rightarrow \beta$의 어떤 공식으로 부터 공식 β를 도출한다.'라는 일반적인 표현을 불린로직(boolean logic)에 의거하여 표현할 수가 없다. 이는 아리스토 텔레스의 추론형태에서의 첫 번째 가정인 '모든 사람은 죽는다.' 와 '홍길동은 사람이다.'에서 두 관계의 확실한 관련을 알아낼 수 있지만 이 관계를 명제로 해석하는 불린로직에 의해서는 표현 할 수가 없다. 여기서 술어로직 또는 일차-순서로직이 필요하게 되고 적용된다. 이 두관계의 명확한 구분을 나타내면(예 1)과 같다.

(예 1)

(a) 불린 로직

가정: (1) 오늘이 일요일이라면 나는 일하지 않는다. $p \rightarrow q$

(2) 오늘은 일요일이다. p

결론: 나는 일하지 않는다. q

(b) 일차-순서 로직

가정: (1) 사자는 풀을 먹지 않는다. $\forall x(\ Lion(x) \rightarrow \text{-}Grasseater(x)\ )$

(2) 나약한 동물들은 풀을 먹는다. $\exists x(Animal(x)\ \&\ Grasseater(x)\ \&\ Tame(x))$

결론: 어떤 나약한 동물들은 사자가 아니다. $\exists x(Animal(x)\ \&\ Tame(x)\ \&\ \text{-}\ Lion(x)\ )$

이 술어로직은 일반적인 프로그램 언어나 질의언어와 비슷해서 자연어 형태의 일반문장으로 정형식 '제주도는 제주시를 포함한다(include)'(T1) 와'그랜드 호텔은 제주시에 있다(exist).'(T2) 라는 두 개의 사실을 술어로직을 적용하여 표현하면 다음과

같다.

(예 2)

T1 : include(chejudo, chejusi).
T2 : exist_in (grandhotel, chejusi).

상기 (예 2)의 T1과 T2에서 include 와 exist_in 처럼 두 개의 사실을 연결하는 부분을 술어(predicate) 라고 하며, 내부에 존재하는 각각의 술어심볼(predicate symbol)은 객체(object)의 집합인 인수(argument) 라고 한다. 이 인수는 콤마(,)로 구분하여 여러 인수들을 적용시킬 수 있으며 이런 인수를 항(term)이라 한다. 항은 대개 상수나 변수의 형태를 갖고 있으며 (예 1)에서의 'Lion'과 'Grasseater'는 일항술어(unary predicate), (예 2)의 'include' 와 'exist_in'은 이항술어(binary predicate)이다. 또한 이런 연속된 항을 필요할 경우 n개의 항으로 나열할 수 있다. 이는 다항술어(n-ary predicate)인 것이다.

(예 2)에서의 'grandhotel'과 'chejusi','chejudo'는 더 이상 분해가 불가능한 도메인 값(domain value)이다. 이러한 현실 세계의 사실들을 객관적으로 기술하여 프롤로그 내에 사실들의 집합을 만들게 되고 이를 프롤로그 데이터베이스라 하며, 관계형 데이터베이스와 아주 밀접한 관계를 갖고 있다. 관계형 데이터베이스에서의 구축된 하나의 튜플 예를 가지고 일차-순서 로직을 적용하면 (예 3)과 같다.

(예 3)

Class	name	room	address	price	tel
superdeluxe	grandhotel	super	chejusi-yundong	150	47-5000

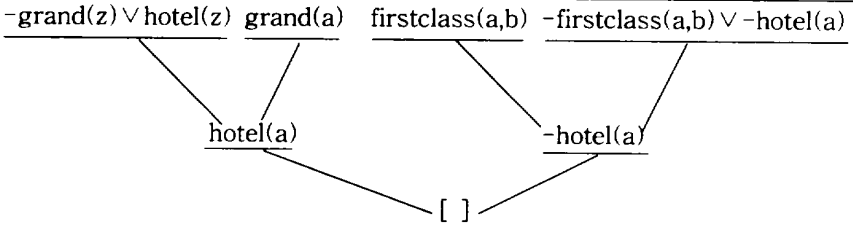
$\exists x$ ((Class(x) = superdelux) $\wedge$ (Name(x) = grandhotel) $\wedge$
(room(x) = super) $\wedge$ (address(x) = chejusi-yundong) $\wedge$
(price(x) = 150) $\wedge$ (tel(x) = 47-5000)).

2) 술어 논리의 표현에 대한 기계적 증명

관계대수와 관계해석을 통한 데이터 조작 언어로서 관계계산 형식의 일차-순위로직이 형식화 되어지는 과정과 처리 과정을 살펴보면 논리 연산자를 합성하여 불린로직을 추론하는 것은 진리표의 결과를 이용하여 항진명제나 모순명제의 구분을 통해 그 진위여부를 알 수 있다. 또한 특정한 사실들의 집합인 데이터베이스의 내용을 검색하기 위한 질의를 요구 받을 경우는 질의문을 통해 얻어진 요소들을 일단의 토큰(token), 즉 식별자들로 구별하는 검사과정(scanner)을 거쳐 구문 규칙(syntax rule)에 적절한 형태인지 여부를 검사하고 의미론적 적합성을 판단한다. 이 단계를 거치면 중간형태인 트리 형태나 그래프 형태로 질의문은 디코드 된다. 이 과정을 거친 후 데이터베이스의 내부 데이터와 직접적인 매칭(matching)을 시도하여 질의문에서의 지시된 방향으로 처리를 이동시킨다. 이것이 실행전략인 질의어의 최적화 과정이다. 특히 일차-순서 로직을 기반으로 개발된 프롤로그에서의 이 과정은 실제 데이터베이스와의 직접적인 교류를 통해 깊이 우선 알고리즘을 적용한 백트래킹(backtracking)에 의거하여 질의목표(goal)를 찾아낸다.

질의어의 대부분은 선두에 정량자를 두고 기본적인 논리 연산자를 결합시킨 형태인데, 그 논리적 함축을 규명해내기 위한 결정이론(resolution theory)을 적용하면 그 질의어에 대한 타당성 검색을 할 수 있다. 결정이론의 기본원리는 두 리터럴(literal)을 매치하는 데 필요한 가장 일반적인 대치 집합을 찾는 일치화 알고리즘이다. 여기서의 결정방법(resolution method)은 정량자와 연산자에 대한 스콜럼 정규형(skolem normal form)(전치 한정자를 소거하는 것)과 절형(clausal)으로 변환하여 절의 일체화를 통해서 최종적으로 분해하는 과정에서 공절(empty clause)인 [ ] 를 만나게 되면 질의어에 대한 타당성검색을 마치게 된다.

Table 1 The Resolution Method

술어 로직화	A = { ∀x(grand(x) → hotel(x)) } B = ∀x∀y(firstclass(x,y) & grand(x)→firstclass(x,y) & hotel(x))
논리적 동등	A & -B = ∀(grand(x)→hotel(x)) & -∀x∀y(firstclass(x,y) & hotel(x)→firstclass(x,y) & hotel(x))
전위결합 정규형	A & -B = ∃x∃y∀z((-grand(z)∨hotel(z)) & firstclass(x,y) & grand(x) & (-firstclass(x,y) ∨ -hotel(x))
절형	A & -B = -grand(z) ∨ hotel(z) firstclass(a,b) grand(a) -firstclass(a,b) ∨ -hotel(a)
분해과정	

3) 프롤로그를 통한 술어로직(prediate logic)의 적용

관계형 데이터베이스에 프롤로그 술어로직을 적용할 수 있다는 가능성은 관계형 데이터베이스의 개별적인 테이블들을 술어인 릴레이션으로서 적절히 대치하고, 그에 따른 튜플과 어트리뷰트는 술어심볼인 인자(argument)들로써 표현할 수 있다는 점이다. 일반적으로 관계형 데이터베이스와 여타의 데이터베이스에서도 SQL이나 QUEL, QBE 등의 언어를 데이터 언어로 사용하고 있지만 데이터베이스의 의미와는 별개로 너무 구문적인 것에만 주안점을 두고 있으므로 해서 구문적 변환규칙과 데이터베이스 스키마에 명세된 제약조건을 혼합해서 사용하는 의미론적 질의 최적화를 달성하기가 어렵다. 따라서 주어진 질의문을 연역적으로 처리하는 의미론적 질의 최적기로서 무결성 제약조건을 만족시키는 시스템의 구축을 위해서라도 관계형 데이터베이스

와 프롤로그와의 결합이 효율적인 데이터관리기법이라 할 수 있다. 이러한 방법으로는 일반적으로 다음의 (예 4) 에서 처럼 무결성 조건을 미리 가정하는 것이 바람직하다.

(예 4)

```
SELECT name                    <=   hotel 테이블에서 5등급 아래인 호텔의
FROM   hotel                        이름을 검색하시오.
WHERE  class ≤ 5 ;
만일 여기에서 지식기반 시스템을 사용하여 무결성 제약조건으로
class( X, hotel ) :- X <= 5 , !    을 정의한다면 이 질의문이
결과가 잘못이라는 것을 알기 때문에 검색을 하지 않음.
```

본 논문에서는 이와같이 관계 테이블을 사실들의 관계로 나타내기 위해 술어로직의 문법을 적용하였는데 질의어를 조작하기 위한 로직기반 시스템을 구성하였다. 여기서 관계형 데이터베이스의 튜플이나 어트리뷰트를 조작하기 위한 연산은 데이터베이스 시스템에서 사용자가 원하는 데이터까지의 도달절차가 어느정도 명시될 수 있는지에 따라 절차언어인 관계대수(relational algebra)와 비절차언어인 관계해석(relational calculus)으로 나뉘는데, 관계대수는 순수 관계 연산자로 관계형 데이터베이스에서 사용하기 위한 선택(select), 투사(project), 조인(join), 디비젼(division) 등을 가지고 있으며, 그에 반해 관계해석은 특별히 관계형 데이터베이스의 검색을 위해 일차-순서 로직에 기반을 두고 이뤄진다. 일반적으로 관계대수는 질의를 통해 원하는 정보를 얻기위한 일련의 연산을 순서에 입각해서 자세히 적용하는 한편, 관계해석은 원하는 정보를 얻는 방법을 정의하지 않아도 정량자 사용의 확대를 통해 질의에 대한 형식적 서술을 한다.

또한 관계해석은 변수가 튜플을 포함하고 있는 튜플 관계해석과 도메인의 값을 해석하는 도메인 관계해석으로 구분된다.

여기서의 튜플 관계해석은 {x | P(x) = ∃x (P(x))} 로 표시하는데 이는 튜플 x 에 대하여 술어 P(x)가 참인 모든 튜플 x의 집합을 검색하라는 뜻이다. 또한 도메인 관계에서의 해석은 어트리뷰트의 도메인 변수를 이용하여 그에 따르는 도메인 값을 취한다. 예를 들면 ∃x, ∃y(P(x,y)) ∧ ∀y(R(a,y)) 등의 형태로 {<x1,x2,...,xn>| P(x1, x2,....,xn)}의 형태를 가진다.

이때 P와 R은 정형식이고, xi는 도메인 변수 또는 상수의 집합이다.

5. 추론 인터페이스의 내부 구조

관계형 데이터베이스 시스템과 지식 베이스 시스템이라는 두 환경을 접목시키는데는 서로의 데이터 결합구조의 불일치를 줄일 수 있는 모델링 방법과 데이터 처리능력이 우선적으로 요구된다. 이를 위해서는 객체들 간의 상호 관련성 표현과 실세계 데이터 모델링을 위해 등장한 개념인 객체지향 데이터베이스 개념(전일수, 1994)을 적용해야한다. 이는 실세계 표현능력과 모델링 능력이 탁월함으로 해서 실세계 대상과 모델과의 결합을 용이하게 하고, 상호 시스템 간의 불일치 부분도 제거될 수 있다(Bancihon, 1988). 본 논문에서는 내부 모듈화된 지식 베이스를 만들기 위해 객체지향 패러다임을 선택하여 상호 시스템 간의 메시지를 교환할 수 있는 메카니즘을 사용하는 인터페이스를 구축한다. 전체적으로는 세 개의 서로 다른 어플리케이션이 구동되는 내부 인터페이스(interface)와 외부 인터페이스(interface)로 구성된다.

본 논문에서 시도될 비주얼베이직과 프롤로그의 내부 인터페이스 구조 는 다음의 Fig. 4 와 같다.

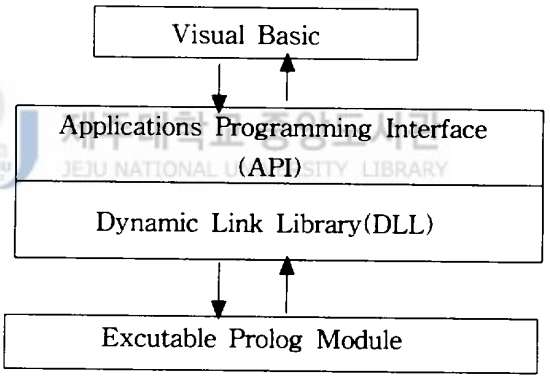


Fig. 4 The control stream of Visual Basic and Prolog

서로 다른 세 환경은 전체적인 윈도우즈 프로그래밍 기법인 윈도우즈 응용 프로그래밍 인터페이스(application programming interface : API)를 사용해야 한다. 이 API는 윈도우즈 프로그램 모듈인 다이나믹 링크 라이브러리(dynamic link library : DLL)에 포함되어 있고 이 DLL에는 윈도우즈에서 프롤로그 실행모듈을 불러 쓸 수

있게 하는 실행코드 및 데이터들이 포함된다. 일반적인 프로그램에서의 스태틱 링크(static link) 프로세스를 사용하지 않고 다이나믹 링킹(dynamic linking) 프로세스를 사용한 이유는 스태틱 링크에 비해 프로그램에서 필요로 하는 루틴이 실제로 프로그램에 포함되는 것이 아니라 별도의 DLL에 있는 코드를 필요로 하는 시점에서 프로시저에 대한 참조로서 해당 루틴을 실행할 수 있어 메모리를 상당부분 절약할 수 있다는 점이다. 이렇게 DLL을 이용한 지속적인 추론 인터페이스의 장점은 첫째, 비주얼 베이직을 사용하여 프롤로그 실행 모듈을 포함(embedding)하는 추론 인터페이스의 실행시에 추가적인 메모리의 부담을 줄일 수가 있고, 둘째로 지속적인 추론엔진의 구축을 가능하게 하는데 이것은 전체 프로그램의 수정없이도 프롤로그의 명령과 관련된 부분만을 DLL에서 수정해주면 된다. 셋째, 비주얼 베이직과 프롤로그 사이의 코드와 데이터를 공유하는 것이 가능해져 전체적으론 지식베이스를 공유하게 되고 넷째, 비주얼 베이직을 호스트언어로 사용하였으나 DLL자체는 C나 C++로 컴파일되어서 비주얼 C/C++이나 델파이등의 다른 언어로도 추론 인터페이스의 개발이 가능하다는 점이다.

6. 사용자 위주의 인터페이스 개발의 필요성

현재 대부분의 개발된 소프트 웨어 형태를 살펴보면 사용자를 위해 얼마나 노력을 기울이고 있는지를 여실히 실감할 수 있다. 이는 전체적인 소프트 웨어에 대한 기호도를 결정짓는 요소로 까지 발전하고 있는데, 이는 새로운 운영체제, 즉 윈도우즈(Windows)라는 32비트 운영체제의 빠른 성과에 의한 것으로 보여진다. 음성, 그래픽, 이미지, 문자 등을 포함하는 멀티미디어 산업이 각광받고 있는 것이 최근의 상황이고 보면 기존의 형태를 보완한 방식의 개발 필요성은 더욱 새삼스러운 것이 아니다. 이는 기존의 도스나 메인 프레임 어플리케이션과는 달리 그래픽 방식을 사용하여 정보를 표현하게 되므로 사용자가 직관적으로 프로그램을 이해하게 한다. 특히 인공지능 분야에서의 지식 기반 시스템을 운영하거나 전문가 시스템을 통하여 사용자에게 상호대화 방식의 시스템을 운영한다면 사용자 위주의 그래픽 인터 페이스가 중요한 개발 요소로 작용한다(김화수외 1, 1993)(김화수외 2, 1995). 사용자 인터 페이스의 설계

를 위한 고려사항으로는 사용자가 최초에 시스템을 접했을 때도 일괄적으로 무엇을 어떻게 해야 할 지를 알 수 있도록 통일성과 편리성을 우선 갖춰야 하므로 본 논문에서는 사용자 인터페이스 개발도구로 MS 사의 '비주얼 베이직 4.0 enterprize' 버전을 선정하였다. 여기에는 자체적인 데이터베이스 검색엔진을 갖추고 있으나 본 논문에서는 단지 사용자 인터페이스의 저작도구로서의 기능과 프롤로그 추론엔진에의 연결을 위해 사용한다. 일반적으로 전문가 시스템은 지식 베이스 모듈, 지식 습득 모듈, 추론엔진 모듈, 설명 모듈, 사용자 인터페이스 모듈의 5개 주요 부분으로 구성된다.

여기서 사용자 인터페이스 모듈은 사용자와의 직접적인 상호 대화를 하는 부분으로 전위(front-end)를 담당하는 부분이 된다. 따라서 일반 프로그래밍 언어들, 예를 들면 Visual Basic, Visual C/C++, Boland C/C++, DELPHI 등은 윈도우즈 프로그래밍을 위한 통합 개발 환경(IDE : integrated development environment)과 개발도구들을 상당부분 지원하고 있어 사용자 인터페이스의 개발에 편리하게 사용될 수 있다(Noton, 1996)(Conger, 1995). 사용자 인터페이스 모듈 개발도구로써 마이크로 소프트사의 비주얼 베이직을 선정한 요인 중의 가장 큰 특징은 타 프로그램과의 연계시에 특정함수를 호출해내어 강력한 API를 만들어 낼 수 있다는 점과 다른 도구와의 연계를 위한 DLL을 호출할 수 있다는 점이다. 또한 다양한 그래픽 자원의 지원과 데이터베이스와 연결이 자유롭고 프로토타입(prototype)과 그 이상의 기능을 갖는 어플리케이션 인터페이스를 결합할 수 있는 능력도 제공되기 때문이다.

본 논문의 개발환경에서는 아래와 같은 사항들을 고려해서 적용되었다.

1) 윈도우환경에 적합한 형태여야 한다.

운영체제의 환경적 지배를 고려해서 사용자가 매뉴얼에 의존하지 않고도 일반적인 형태의 아이콘을 사용하거나, 간단한 제목을 주어 무엇을 의미하는 것인지를 한 눈에 알아 보도록 해야한다.

2) 통일성이 있어야 한다.

사용자와의 대화 진행과정과 프로그램의 흐름 사이에 동일한 형태의 기준을 가지고 적용되야 한다. 사용자가 초기에 접근방법을 이해할 수 있으면 프로그램은 일관되게 부메뉴에서도 그러한 방식이 적용되도록 해야 한다.

3) 사용자 오류를 최소화 시켜야 한다.

이는 사용자가 프로그램을 접하는 자세를 좌우하므로 신중히 고려되야 할 사항이다. 사용자가 프로그램을 정확하게 이해하고 사용함에 있어서는 초기에 많은 시행착오를 겪을 수 있으므로 사용자가 오류를 범할 가능성을 최소화 할 수 있는 방법을 설계하고 오류를 범할지라도 쉽게 빠져나올 수 있는 수단을 제공해주어야 한다.

4) 미적이어야 한다.

현재까지는 화상을 통한 사용자 인터페이스를 기반으로 하기 때문에 일차적으로 보여지는 화면구성에 세심하고 미적인 요소를 가미하여 사용자가 아이콘만 보고도 무슨 기능인지를 아는 것 뿐만 아니라 시각적인 풍요로움도 느낄 수 있어야 한다.

5) 중요한 부분을 부각시킨다.

화면을 한 눈에 볼 수 있도록 하는 것은 물론이고 사용자와의 대화를 전개해 가는 부분을 중심에 배치하고, 텍스트부분은 제목과 내용의 색상구분을 주어 사용자의 시야에 한 번에 들어올 수 있어야 한다.

6) 사용자를 위한 선택의 여지를 남겨둔다.

꼭 필수적인 부분은 어쩔 수 없더라도 다양한 구성을 통해 사용자가 선택할 수 있는 부분의 가능성을 어지럽지 않을 정도에서 최대한 보장해 주도록 해야한다 (Sheuniderman, 1991).

Ⅲ. 추론 인터페이스의 설계

1. 전위 시스템의 설계

여기에서는 제안된 방법을 구현하기 위해 전형적인 형태의 한국적 가족관계 모델을 도식화 하였다. 제안된 전위 시스템(front-end system) 인 비주얼 베이직을 통해 로직 베이스를 구축하는 프롤로그와 후위 시스템(back-end system)인 관계형 데이터베이스와의 관계를 도식화하면 Fig. 5와 같다. 본 시스템의 구성은 나타낸 바와 같이 사용자 인터페이스(user interface)와 추론엔진(inference engine) 그리고 지식 습득 모듈(knowledge acquisition module), 로직 베이스 모듈(logic base module), 관계형 데이터베이스시스템으로 설계하였다.

사용자 인터페이스는 사용자와 시스템 간의 정보를 전달하는 역할하는 부분으로 윈도우 환경에 맞게 설계하였으며, 새로운 데이터의 생성과 추가, 수정,삭제를 할 수 있다.

추론엔진은 제어전략으로 목표지향 방식으로 개선된 역방향 추론인 규칙-값 추론을 채택하며, 단지 관련된 정보만을 시스템에 입력하는 것으로 제한한다. 모듈화된 로직 베이스를 기반으로 질의에 대한 응답을 위해 질의를 분해, 해석하고 저장된 데이터로 부터 새로운 지식을 추론하기 위해 어떤 규칙을 적용해야 하는지를 결정한다.

로직 베이스 모듈은 데이터들 간의 관계를 술어로 표현하여 규칙을 정하는 부분으로 전문가 시스템에서의 생성규칙을 적용한다.

규칙 릴레이션에 적용되는 규칙들은 결론 부분이 하나만 존재하는 혼절(horn clause)만을 취급하도록 한다.

관계형 데이터베이스는 관계모델을 이용해서 테이블을 생성하는데 시스템 구성에 있어 가장 선행되야 하는 부분이다.

오픈 데이터베이스 연결(ODBC)은 관계형 데이터베이스 관리 시스템과 전위 인터페이스를 구성하는 호스트 언어와의 상호통신을 지원하는 부분이다.

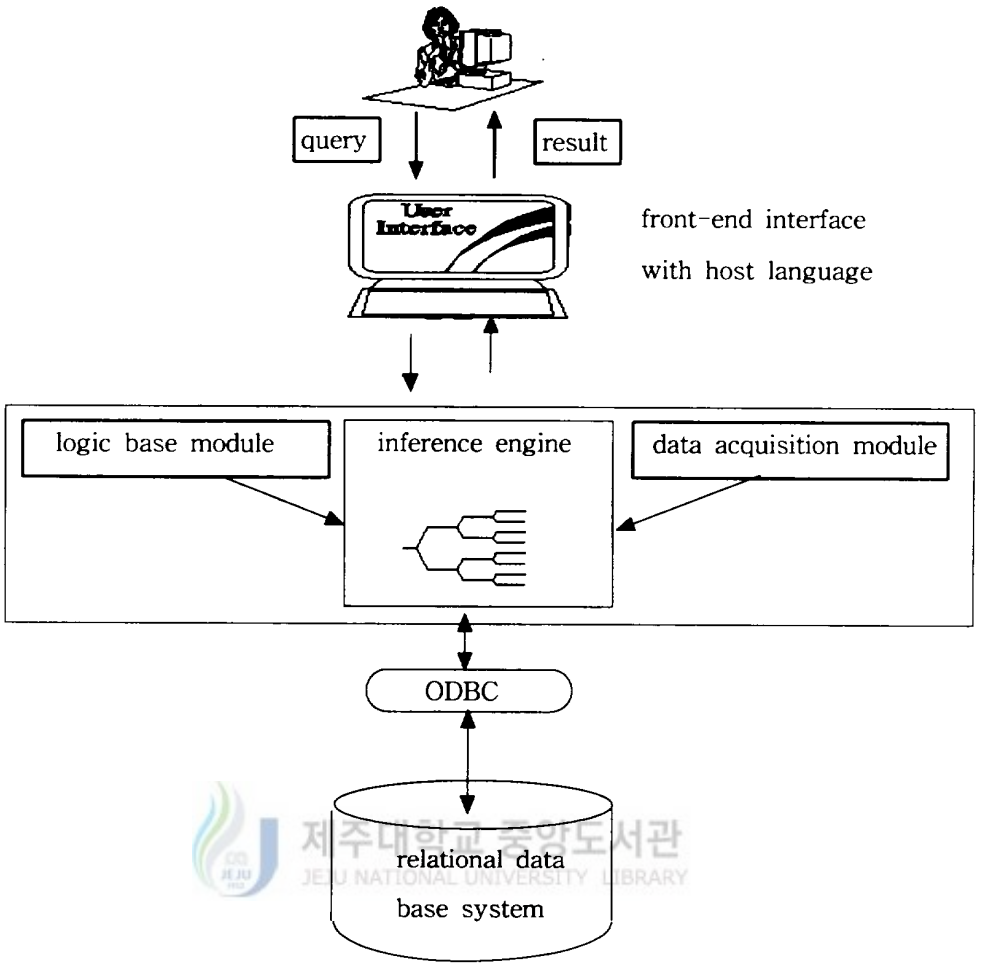


Fig. 5 Front-end system for Visual Basic and Prolog

2. 관계형 데이터베이스의 설계

1) 개념적 대상

여기서는 개념적 대상으로 자료의 일반화와 집단화를 이루기에 가능한 모든 것을 포함한다. 다시 말해서 데이터 중심의 개념 스키마와 처리 중심의 트랙잭션 모델링이 가능한 모든 객체를 대상으로 한다. 사용자의 요구조건 명세서를 기초로 사용자의 요

구에 맞도록 기존의 관계형 데이터베이스를 가지고 누구나 자연스럽게 이해할 수 있는 정보구조를 만들어서 짧은 응답시간 및 저장공간의 최소화 같은 성능 최적화가 이뤄지도록 한다. 이는 일반적인 데이터베이스 설계방식에 의거해서 진행하되 데이터 중심의 데이터베이스의 설계와 데이터 처리 및 그에 관계된 응용프로그램의 설계를 서로 밀접하게 관련지어서 진행함으로써 달성된다.

본 논문에서의 관계형 데이터베이스는 개념세계에서 생성된 논리적 구조를 이용해서 가족체계의 중심인 혈연관계를 기반으로 하는 관계 데이터 모델로 구성하였다. 개념적 설계를 위해 추상화 방식의 일반화(generalization)와 집단화(aggregation)를 적용하였다. 계보를 이루는 관계를 간단히 트리구조로 구축하면 다음의 Fig. 6, Fig. 7, Fig. 8 과 같다.

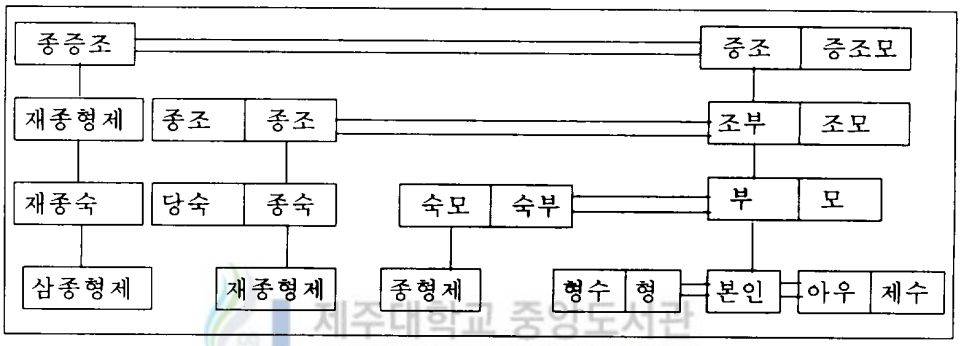


Fig. 6 A lineage of family members in a direct line (-: a degree, =: two degree)

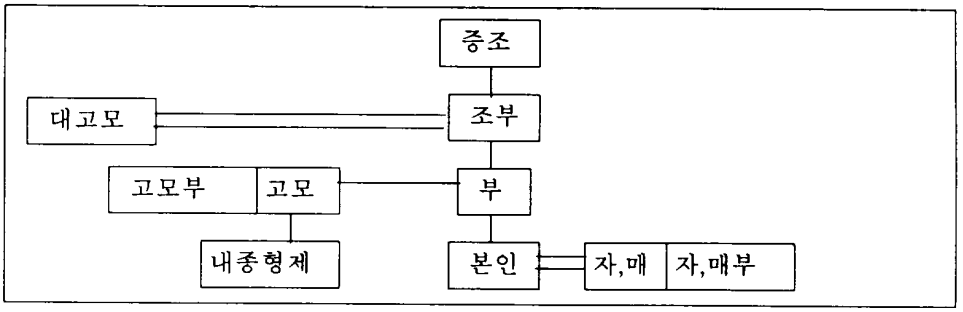


Fig. 7 A lineage of relative on the father's side

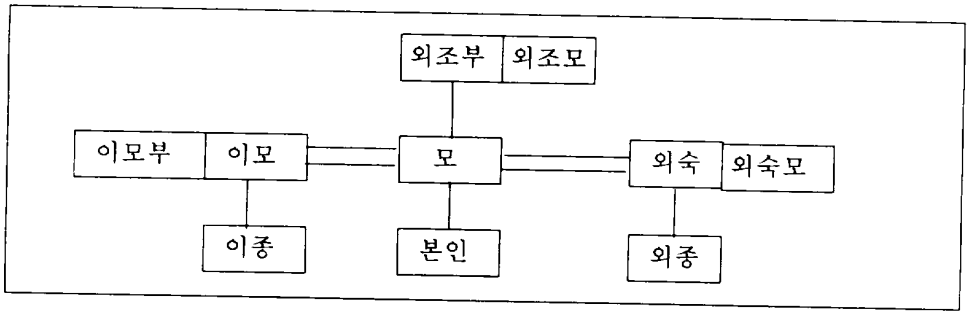


Fig. 8 A lineage of relative on the mother's side

2) 관계형 데이터베이스 시스템의 설계

계보자문시스템에 적용되어질 데이터들은 후위(back-end)부분을 담당하여 관계 데이터베이스로서 오픈 데이터베이스 연결(ODBC)을 통해 프롤로그가 담당할 로직기반 추론엔진으로 답변을 보내주는 역할을 수행한다. 이 시스템에 적용되어질 관계형 데이터베이스의 이름은 시스템과의 연결시에 이용된다.

이 관계형 데이터베이스는 마이크로 소프트사의 'MS ACCESS 7.0' Version을 사용하였으며 3개의 테이블을 생성하였다. 그 형태와 구성은 다음과 같이 하였다.

1) 개인 자료 Table (구성원에 대한 이름과 간단한 관계 기술)

Table 2 Person Table

필드 이름	데이터 형식	설 명
* personid	숫자(key)	(indexed)
* sectname	문자열[20]	
* name	문자열[40]	
* sex	숫자[1]	m/f
* father	숫자	(indexed) (a personid)
* mother	숫자	(indexed) (a personid)
birthdate	문자열[10]	(date 년-월-일)
deathdate	문자열[10]	(date 년-월-일)
birthplace	문자열[50]	
history	문자열[255]	

2) 혼인관계 테이블 (결혼으로 맺어진 부부관계를 기술)

Table 3 Marrige Table

필드이름	데이터 형식	설 명
* husband	숫자	(indexed) (a personid)
* wife	숫자	(indexed) (a personid)
marrydate	문자열[10]	(date 년-월-일)
divorce	문자열[10]	(date 년-월-일)
note	메모[255]	

(주) * 표시는 본 논문에서 연결되어진 부분임을 표시한 것이다.

3) 호칭설명 테이블(가족관계 구성원의 호칭에 관한 내용을 기술)

Table 4 Branch Table

필드이름	데이터 형식
personid	일련번호
호칭	문자열[10]
설명	메모[255]

개인정보의 관계스킴(relation scheme)과 릴레이션 사례의 설계는 다음과 같이 Table 5와 Table 6에 나타내었다.

Table 5 Personal information of relation scheme

person (personid, sectname, , name, sex, father, mother, birthdate,
deathdate birthplace, history)
marriage (husband, wife, weddingday, note)

Table 6 Personal information of a relation instance

(a) person relation instance

personid	sectname	name	sex	father	mother	birthdata
1	남평	문석환	m	0	0	1965-01-19
2	남평	문준식	m	2	2	1921-07-28
3	제주	양경전	f	1	1	1923-11-02

deathdate	birthplace	history
	제주시이도동	고향을 지키며 살고있으며....
	제주시이도동	성민약국을 경영하였으며.....
	제주시노형동	제주 여성유림회의 부지부장을 맡아...

(b) marriage relation instance

husband	wife	weddingday	note
2	3	1946-02-20	
4	5	1968-07-21	
5	6	1975-10-10	

3. 프롤로그 추론엔진 설계

이 부분은 프롤로그와 비주얼 베이직과의 엠베딩(embedding)을 통해 실질적인 데이터 제어를 하는 부분으로 여기서의 규칙 기반으로 파싱(pasing)처리를 하고, 의미론적(semantic) 차이의 감소와 탐색의 순서를 결정한 후, 관계형 데이터베이스에서 패턴매칭을 시도하여 데이터를 추론하기 위한 규칙기반 추론엔진이다. 여기서는 주로 데이터에 대한 추론을 하지만 비주얼 베이직과의 데이터 동적 교환 (DDE) 을 고려해야 한다.

로직 베이스를 이루는 프롤로그의 주요구성은 다음 Fig. 9 와 같은 모듈들로 설계한다. 코드를 이루는 주요부는 다음과 같다.

- 사용자가 선택한 관계를 써서 디스플레이하기 위한 관계리스트
- 프롤로그 질의로 부터 SQL 질의문을 구축을 위해 사용할 데이터베이스의 정의
- 데이터베이스 내의 직접적인 주요관계를 정의할 기본(low-level)규칙들
- 한 사람의 관계변화에도 영향을 받지 않는 복잡한 관계들을 규정하는 고 수준 (high-level) 규칙
- 호스트 언어에서 적용될 질의 인터페이스 디자인

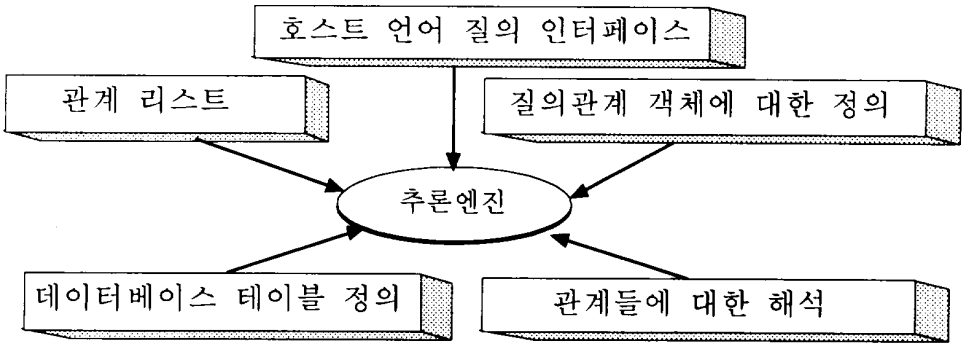


Fig. 9 Main structure of inference engine

추론엔진의 검색전략으로는 목표에 의한 추론방식인 개선된 역방향 추론인 규칙-값 추론(rule-value inference)을 선택하는데 이는 시스템의 불확실성을 제거 할 수 있는 정보를 요구하는 방식의 이론이다(전용진, 1989). 본 논문에서는 이를 응용하여 규칙-값 쌍에 기반하여 관련성의 정도에 따라 깊이와 넓이로 탐색하도록 규칙베이스를 구축한다.

1) 추론 알고리즘

프롤로그는 관계형 데이터베이스의 구축방식과 동일한 이론적 근거를 갖고 있다. 이런 사실은 본 논문에서 프로토타입으로 제시하는 계보 관계 트리구조에서 처럼 개인정보를 저장함에 있어 데이터베이스의 구축형태와 매우 흡사하다. 프롤로그의 사실들은 관계 테이블에서의 열인 튜플에 해당하고, 이는 한 개인에 대한 모든 사실을 묘사할 수 있다. 이 데이터들은 파일의 형태이지만 데이터베이스에서 처럼 직접적으로 질의를 할 수 있다. 질의를 처리하는 프롤로그의 특성은 먼저 질의패턴을 모든 사실에 매칭시키는 방법을 통해 질의들에 대한 해를 찾고, 그 패턴을 모든 규칙에 매칭시키는 것을 통해 검색을 마친다. 이런 사실들을 기반으로 본 논문에서 제안하는 추론 인터페이스의 로직베이스의 추론엔진에 적용하는 추론 알고리즘을 단계별로 구분하면 다음과 같이 나타낼 수 있다.

Step 1. 변환 질의어로부터 한 객체의 이름과 알고싶은 관계를 입력받고, 그 객체의

ID를 얻어서 데이터베이스 상에서의 그 객체의 위치와 객체와 관련된 모든 속성을 얻는다.

Step 2. 관계에 대한 근접성을 기준으로 질의받은 관계를 관계리스트의 매칭순서에 따라 규칙절에 대해 매칭을 시작한다. 이때 매칭기준은 객체ID간의 매칭이다.

Step 3. 관계에 따른 규칙들을 단일 항으로된 기초규칙과 기초규칙들을 포함하는 복합규칙의 적용을 통해 추출된 객체간의 관계를 설정한다. 첫번째 규칙부터 경로를 지정하여 데이터베이스를 검색하고 하나의 단일규칙이나 복합규칙의 구성에 대해 공절(empty)이면 되돌림(backtracking)진행을 위해 자신을 재귀호출(recusion)하여 다른 규칙에 적용되도록 한다. 하나의 객체와 일치되면 !(cut)명령을 사용하여 중단하고, 일치된 ID에 0 값을 받으면 최상위 관계로 더 이상 관련성이 없는 것으로 검색을 중단시키고 '실패'값을 반환한다.

Step 4. 일치하는 관련객체의 ID를 가지고 객체이름을 반환한다.

2) 의미론적 무결성 검출 알고리즘

이 부분은 데이터베이스의 의미를 계속적으로 확보, 유지하기 위해 새롭게 데이터 내용을 갱신하거나, 추가하는 것의 적합성을 검사한다. 본 논문에서의 무결성은 한 사람의 아버지와 어머니에 대한 성별을 맞게 작성하는 지, 혈연관계가 없는 배우자 관계인지, 조상에 대한 관계가 올바로 설정되었는지를 검사한다. 각 무결성에 대한 규칙은 제약조건을 깨뜨리는 구성이 아닌 지를 확인한다. 만약 그렇다면 이 질의는 '실패'라고 알리는 코드를 질의의 반환값으로 보내고 호스트 언어인 비주얼베이직에 의해서 연결될 수 있는 메시지를 삽입시킨다. 기본적인 단일 규칙과 규칙의 쌍을 이루어 어떤 객체는 그 자신의 조상이 될 수 없다는 사실을 확보할 수 있다. 그 규칙을 나타내면 다음과 같다.

```
ancestor_check(Name) :-
    ancestor(Name,Name),
    assert(message($스스로의 조상이나 후손이 될 수 없읍니다!$)),
    ! , fail.
    ancestor_check(_).
```

이 무결성 규칙에서 첫째 규칙은 입력받은 개인의 이름을 가져와서 그 이름의 조상이 등록되어 있는 지를 검사한다. 만약 이것이 참이면 메시지는 삽입되고, 심볼 !로써 더 이상은 검사하지 말라고 지시한다. 그리고 그 결과 값으로 'fail'을 반환한다. 반대로 첫째 규칙이 실패하여 그 자신이 조상이 아니라면 두 번째의 규칙이 실행되어 언제나 'true'를 반환한다. 또한 혈연관계가 없는 배우자 검사규칙도 이와같은 해석과정으로 첫째 규칙에서 이중혼인인지 아닌 지를 판별하고 두 번째 규칙에서는 혈연관계가 있는 지, 없는 지를 검사하여 근친혼 사실을 판별한다. 이 두 번째 규칙의 적용을 위해서는 기본규칙으로 혈연관계에 대한 규칙이 정의되어 있어야 한다.

```
spouse_check(Name, Spouse) :-
    blood_relative(Name, Spouse),
    assert(message($ 이 사람은 혈연관계가 있습니다! $),
    !, fail,
    sposu_check(_,_).
blood_relative(X,Y) :- (ancestor(X,Y) ; ancestor(Y,X)).
blood_relative(X,Y) :- sibling(X,Y).
blood_relative(X,Y) :- cousin(X,Y).
blood_relative(X,Y) :- (uncle(X,Y) ; uncle(Y,X)).
blood_relative(X,Y) :- (aunt(X,Y) ; aunt(Y,X)).
```


이와 같은 방식으로 하위규칙들을 정의하여 전체적으로 적용되는 무결성 알고리즘은 첫째, 이전의 모든 에러 메시지를 초기화하고, 둘째로 같은 객체가 선언되어 있는 지를 검사하며, 성공하면 새로운 객체의 모든 사실을 데이터베이스에 추가시킨다. 그

리고나서 셋째로는 그 사실이 데이터베이스에서 무결성 조건을 깨트리는 지의 여부를 위와 같은 제약조건들을 동원하여 다양하게 검사한다. 마지막으로 어떤 무결성 검사를 깨트리면 갱신을 위한 데이터를 철회시킨다.

3) 관계형 데이터베이스에 대한 프롤로그의 정의

이 부분은 위의 Table 2와 Table 3에서 디자인된 테이블의 행(Row)을 정의한 사실들이며, 질의문에서 적용되어진다. 구성은 다음과 같다.

Table 7 Definition of relational database within Prolog

person table	marriage table
db_table(person, personid, i).	db_table(marriage, husband, i).
db_table(person, sectname, s20)	db_table(marriage, wife, i).
db_table(person, name, s40).	db_table(marriage, married, s10).
db_table(person, sex, a1).	db_table(marriage, divorce, s10).
db_table(person, father, i).	db_table(marriage, note, s255).
db_table(person, mother, i).	
db_table(person, dirthdate, s10).	
db_table(person, deathdate, s10).	
db_table(person, birthplace, s50).	
db_table(person, history, s255).	

4) 관계 리스트 표현

상기의 코드를 포함하는 프롤로그 로직 베이스의 구성은 다음과 같이 하였다.

```
% FAMDB.PRO ---- Lineal Relationships
< 관계 리스트 >---- 정의된 모든 관계의 리스트를 하나로 포함시키는 기능
    relation ([parent, husband, wife, ancestor, descendent, sibling, elder_brother,
```

elder_brother_wife, younger_brother, younger_brother_wife, elder_sister, elder_sister_husband, younger_sister, younger_sister_husband, father, father_sister, father_sister_husband, father_brother, father_brother_wife, mother, mother_sister, mother_sister_huband,mother_brother, mother_brother_wife, child, son, daughter, step_sibling, step_parent, step_child, step_father, step_mother, step_son, step_daughter, nephew, niece, cousin, grandparent, gandfather, grandmother grandchild, grandson, granddaughter, grandfather_sister, grandfather_sister_husband, grandfather_brother, grandfather_brother_wife, grandmother_sister, grandmother_brother, great_grandfather, great_grandmother, great_grandson, great_granddaughter, great_grandsister, great_grandbrother, great_uncle, great_aunt]).

5) 사람에 대한 객체규정

여기서의 술어는 한 사람에 대해 적용된다. 모든 규칙들은 한 사람에 대한 기본적인 표현들을 기초로 술어를 작성하는데, 이것은 모두 데이터베이스에 선언된 형식을 따른다. 만일 객체에 대한 표현을 변경하고 싶다면 여기서 규칙만 바꿔주면 된다.

여기의 규칙들은 술어 'db_query/2'를 통해서 데이터베이스로 접근한다. 이 술어는 두 개의 인자를 가지는데, 첫 번째 인자는 테이블의 이름이고, 두 번째 인자는 '필드명= 값'의 인자이다. 이 필드명은 데이터베이스의 어트리뷰트에 일치되고, 값은 제한된 질의어이거나 검색을 위한 프롤로그 변수라 할 수 있다.

예를들면, '문선영'이라는 이름을 가진 사람의 아버지를 찾는다면 "db_query(father [name = X , name = '문선영'])"으로 명령라인을 적어 주면 된다. 이는 백트랙킹을 이용해 데이터베이스에서 사상되는 다음 단계들의 응답을 얻어낸다.

```
% person object에 대한 정의 %
  fullname(P, S, N) :-
      personid_ok(P),
```

```
    db_query(person, [personid=P, sectname=S, name=N] ).
male(P) :-
    personid_ok(P),
    db_query(person, [person, [personid=P, sex=m] ).
female(P) :-
    personid_ok(P),
    db_query(person, [person, [personid=P, sex=f] ).
mother(M, C) :-
    personid_ok(M),
    personid_ok(C),
    db_query(person, [personid=C, mother=M].
father(F, C) :-
    personid_ok(F),
    personid_ok(C),
    db_query(person, [personid=C, father=F] ).
husbandwife(H, W) :-
    personid_ok(H),
    personid_ok(w),
    db_query(marriage, [husband=H, wife=W] ).

% personid가 0인 경우는 전형적으로 조상연결의 끝부분을 의미하므로 검색을 마
% 치라는 부분.
pesonid_ok(P) :-
  var(P), !.
personid_ok(P) :-
  integer(P),
  P > 0.
```

6) 관련성에 대한 규정

여기서는 데이터베이스를 기초로 규칙을 선언하고 또한 몇 개의 우선적 규칙을 만들어서 디자인한다. 또한 선행규칙을 두어 mother/2나 husbandwife/2 같은 몇 개의 우선규칙을 적용해서 디자인 한다.

```
spouse(P1, P2) :- husbandwife(P1, P2), !.
spouse(P1, P2) :- husbandwife(P1, P2).
parent(P, C) :-
  (mother(P, C) ; father(P,C).
child(C, P) :- parent(P, C).
son(C, P) :- parent(P, C), male(C).
daughter(C, P) :- parent(P, C), female(C).
wife(W, P) :-
    spouse(W,P),
    female(W).
husband(H, P) :-
    spouse(H, P),
    male(H).
ancestor(A, P) :-
    parent(A, P).
ancestor(A, P) :-
    parent(X, P),
    ancestor(A, X).
                :
                :
grandchild(GC, X) :-
    parent(X, C),
    parent(C, GC).
```

7) 호스트 언어 질의 인터페이스

5개의 인자를 가진 'query/5'를 통해 로직베이스로 넘어간다. 호스트언어인 비주얼 베이직을 통해 로직베이스를 호출하는 것이다. 첫 번째 인자는 father(X, '문선영')과 같은 질의항이고, 나머지 4개의 인자는 personid, sectname, name, sex에 관한 인자이다.

```
query(Q, Personid1, Sectname1, Name1, Sex1) :-
    Q= . . [R, P1, P2, P3],
    get_pid(P1, Personid1],
    get_pid(P2, Personid2),
    Q2= . . [R, Personid1, Personid2],
    call(Q2),
    fullname(Personid1,Sectname1, Name1, Sex1)

% 아래서의 'get_pid/2'는 첫 번째 인자로 개인을 식별하고, 두 번째 인자에
% 그 개인의 ID가 할당된다. 즉, 개인의 이름을 기준으로 그 위치를 식별한다.
get_pid(Personid1, Personid2) :-
    var(Personid1),
    var(Personid2),
    !,
get_pid(0, _) :-
    !
  fail.
get_pid(Personid1, Personid2) :-
  integer(Personid1),
  !,
  Personid1=personid2.
get_pid(Personid1:_, Personid) :-
  integer(Personid1),
```

```
    !,
    Personid1=Personid2.
  get_pid([Sectname, Name, Sex], Personid) :-
    var(Person1),
    fullname(Personid, Sectname, Name, Sex).
  get_pid(Name, Personid) :-
    var(Personid),
    fullname(Personid, Sectname, Name, Sex).
  get_pid(Sectname, Personid) :-
    var(Personid),
    fullname(Personid, Sectname, Name, Sex).
```

8) 데이터 조작에 대한 의미론적 무결성 처리

이 부분은 데이터의 중복이나 관계의 무결성을 처리하기 위한 부분이다.

```
add_person( Sectname, Name, Sex, Father, Mother, Spouse ) :-
  retractall( massage(_) ),
  dup_check(Name),
  add( Sectname, Name, Sex, Father, Mother, Spouse ),
  ancestor_check(Name),
  father_check(Name, Sex, Father),
  mother_check(Name, Sex, Mother),
  spouse_check(Name, Spouse).

dup_check(Name) :-
  person(Name),
  assert(message($ 이미 존재하고 있는 사람입니다!$)),
   !, fail,
```

```
    dup_check(_).
        :
        :
spouse_check(Name, Spouse) :-
    blood_relative(Name, Spouse),
    assert(message($ 이 사람은 혈연관계가 있습니다! $) ),
     !, fail,
    spouse_check(_, _).

blood_relative(X, Y) :- (ancestor(X, Y) ; ancestor(Y, X)),.
blood_relative(X, Y) :- sibling(X, Y).
blood_relative(X, Y) :- cousin(X, Y).
blood_relative(X, Y) :- (uncle(X, Y) ; uncle(Y, X) ).
blood_relative(X, Y) :- (aunt(X, Y) ; aunt(Y, X) ).
% utility predicates
member(X, [X|_].
member(X, [_|Y] ) :- member(X, Y).
```


4. 시스템 내부 인터페이싱

추론을 위한 추론 인터페이스 구축의 중심이 되는 부분인 내부 인터페이싱은 윈도우 라이브러리 함수를 이용한다.

기본적으로 객체선언을 통한 추론방법을 적용하기 위해선 객체 자신의 값과 고유의 객체 식별자(object identity)를 갖고서 내부구조를 정의하고 규칙에 선언된 관계에 의해 저장된다. 객체 확장(object extension)을 위한 전체적인 시스템의 진행과정은 Fig. 10과 같이 구성한다.

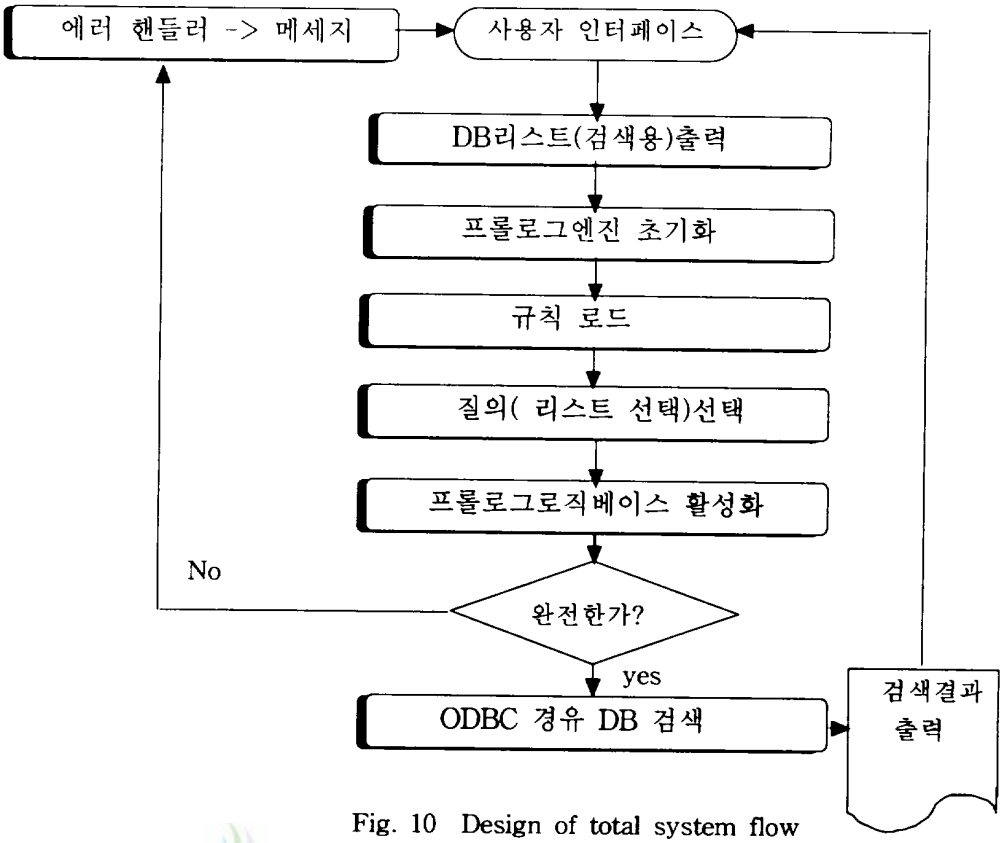


Fig. 10 Design of total system flow

시스템 내부 인터페이싱에 의한 진행흐름은 다음과 같다.

1) 추론 인터페이스 초기화---동적연결(VBRUN400.dll+ODBC.dll+LOGIC.dll)

2) 관계형 데이터베이스에 대해 데이터베이스 호출(외부 저장 데이터베이스 호출)
---동적연결 (ODBC.dll)

3) 사용자 인터페이스의 그래픽 입력창(user interface)을 통해 자료를 요청받고 취득된 데이터를 기준으로 규칙(rule)실행에 의한 검증.---동적연결(LOGIC.dll)

4) 데이터를 검색하고 매칭을 시도하기 전에 실행되야 할 무결성 검사.
---로직베이스(VBRUN400.dll+LOGIC.dll)

5) 패턴매칭을 통한 데이터 호출--- 동적연결 (ODBC.dll)

6) 외부데이터 처리.--- 동적연결 (VBRUN.dll+LOGIC.dll)

5. 추론 인터페이스 변환 메카니즘

본 논문에서 제안하는 추론 인터페이스를 구축하기 위한 윈도우즈 프로그래밍 언어와 프롤로그의 추론엔진을 인터페이싱하는 문제는 일반적으로 기존의 단일 프로그램을 컴파일해서 상호호환되도록 하는 문제와는 차원이 다르다. 이는 서로 동일한 변수형태를 사용하지 않고 명령의 형식과 결과를 받아들이는 방식이 차이를 극복야만이 추론결과를 상호교류할 수가 있다. 자연어 질의와 같은 프롤로그의 술어는 호스트 언어에서 호출되는 함수에 사상되도록 해야 하는 데 이에 대한 구상은 첫째, 호스트 언어에 프롤로그 매핑함수를 내부 모듈화하고, 둘째로 추론해야 할 질의어를 프롤로그의 질의형태인 '<술어>(검색항, <대상>)'에 맞게 문자열화시켜 API함수를 통해 표준 프롬프트 '?-' 로 활성화된 프롤로그 응답기(listener)로 전달하면 셋째, 추론된 해는 메모리의 힙(heap)상에 Term으로 저장되므로, 호스트 언어에서의 변수Term이 그 결과를 전달받아 데이터를 가져오게 하는 절차적 방식으로 질의를 처리한다. 즉, 호스트 언어에서의 변수항 Term과 힙영역에서 프롤로그 추론결과인 Term을 상호사상(mapping)되도록 하는 것이다. 이와 같은 방식으로 추론 인터페이스 설계를 위한 변환 메카니즘을 호출함수의 기능과 관련하여 나타내면 다음과 같다.

< 추론 인터페이스 변환 메카니즘 >

Step 1. person(X)에 대한 추론데이터의 입력

```
Person = PersonList.List(PersonList.ListIndex)
Relationship = RelationshipList.List(RelationshipList.ListIndex)
```

Step 2. 프롤로그 질의 형태인 '<relationship>(Variable, <person_name>)' 으로 만들기 위해 변환함수호출.

```
tf = CallStr(EngID, Term, Relationship+ "(X, '"+ Person +"')")
```

여기서 CallStr()함수는 사용자가 질의항에서 선택한 사람과 관계가 만약 '문선영'과 'ancestor'라면 프롤로그 질의항 "ancestor(X, 문선영)" 을 생성한다. 이것

은 프롤로그 응답기에 '?- ancestor(X, 문선영)'을 입력하는 것과 동일하다. 이때 항Term은 프롤로그의 규칙 베이스에서 일치되어지고, 변수X는 같은 값에 일치되어진다. 그러면 질의의 결과는 CalllStr()함수에 의해 다시 Term에 리턴되어 힙 상의 Term위치에 저장된 첫 번째로 매칭된 인자의 값을 가져오기 위하여 비주얼베이직 변수Term이 사용된다. 이때 그 값은 하나의 문자열로 저장되고 그 문자열은 리스트에 더해진다.

Step 3. 질의로 부터의 모든 결과를 알아내기 위하여 Backtracking을 실행.

```
While (tf)
  '질의항의 첫 번째인자에 매칭된 값을 가져온다.
  Getarg EngID, Term, 1, bSTR, StrVal
  '이 문자열 값을 비주얼베이직의 리스트에 추가한다.
  RelatedPersonList.AddItem StrVal
  '다음 해를 추론하기 위하여 다시 항을 셋팅하고 프롤로그 호출.
  '더 이상 해가 없을 때 tf는 0을 리턴한다.
  tf = Redo(EngID)
Wend
```

이 과정은 Redo()함수를 호출 함으로써 완료될 수 있는 데, 이때 변수Term은 그 다음을 매칭하기 위해서 준비한다. 이 과정은 Redo함수가 성공하는 동안은 계속된다. CallStr()와 Redo()함수는 TRUE와 FALSE값을 반환하는 함수이다.

6. 사용자 인터페이스 설계

사용자 인터페이스는 초기화면의 시작버튼을 클릭하여 메인 폼을 호출하는 것으로 시작되며 메인 폼에선 상단의 콤보박스에 관계형 데이터베이스의 객체 리스트와 규칙 부분에 설정된 관계형성 리스트를 보여주어 사용자는 간단히 마우스를 이용해서 객체를 선택하고, 그 객체와의 알고싶은 관계를 선택하면 추론의 결과인 관계된 모든 객체의 리스트를 하단의 리스트 박스에 표시되도록 한다. 다시 추론된 여러 객체 중에 단일 객체를 선택하면 그에 관련된 설명과 사진을 호출하여 보이도록 한다. 그리고 메뉴부분은 우선 새로운 객체의 삽입과 삭제, 수정을 가능하도록 하여 새로 생성되는 객체에 대해서도 데이터 베이스에 반영되도록 한다. 계보 자문 프로토타입의 초기화면 설계는 Fig. 11와 같다.

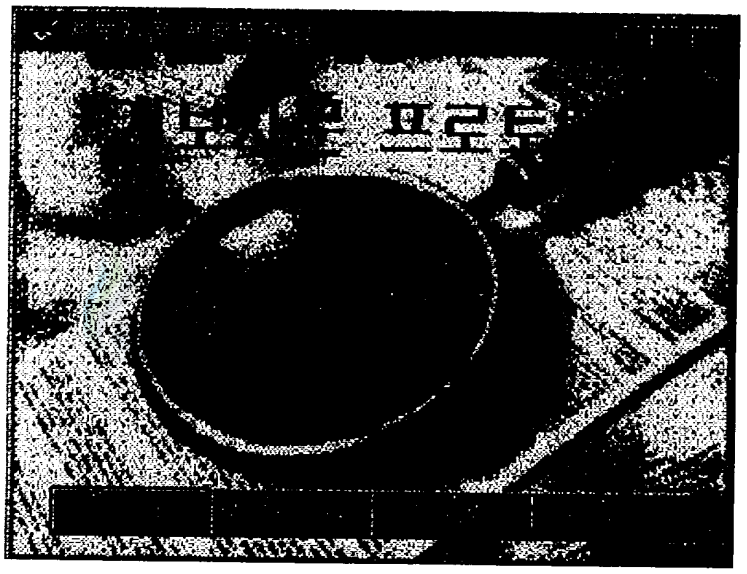
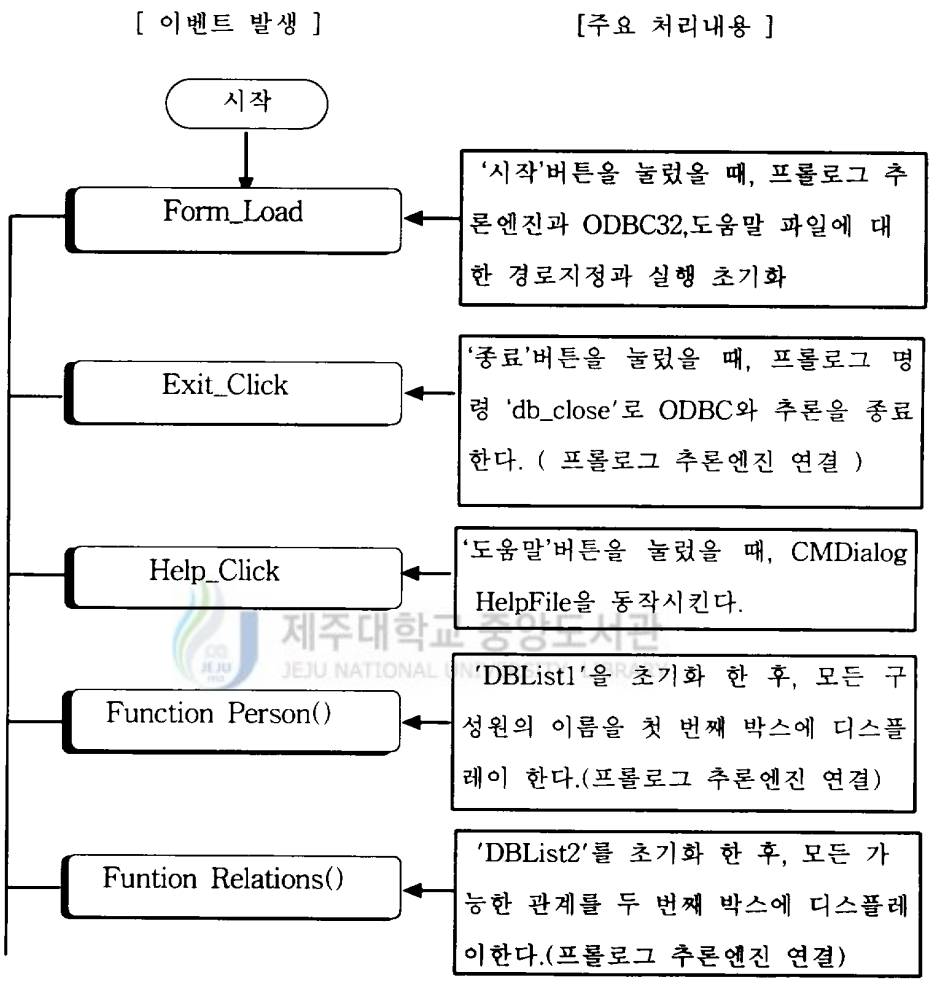
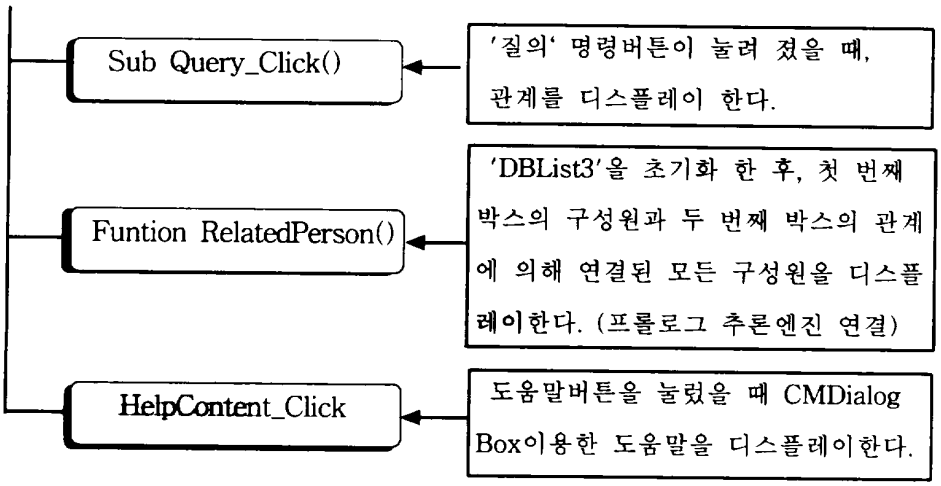


Fig. 11 The start form of lineage consulting prototype

1) 메인 폼 설계

계보자문 시스템의 '시작'이후의 메인 윈도우를 구성하는 주요 이벤트별 프로시저의 역할은 다음과 같다.





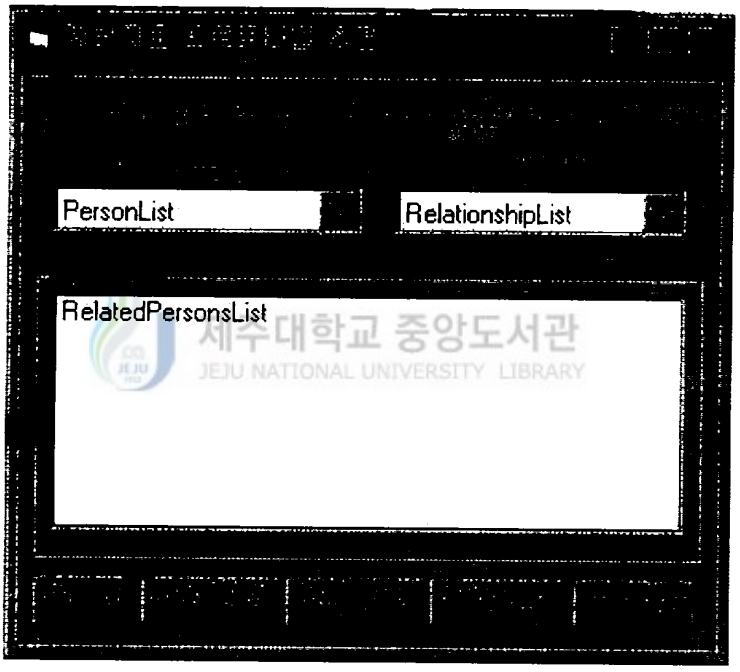
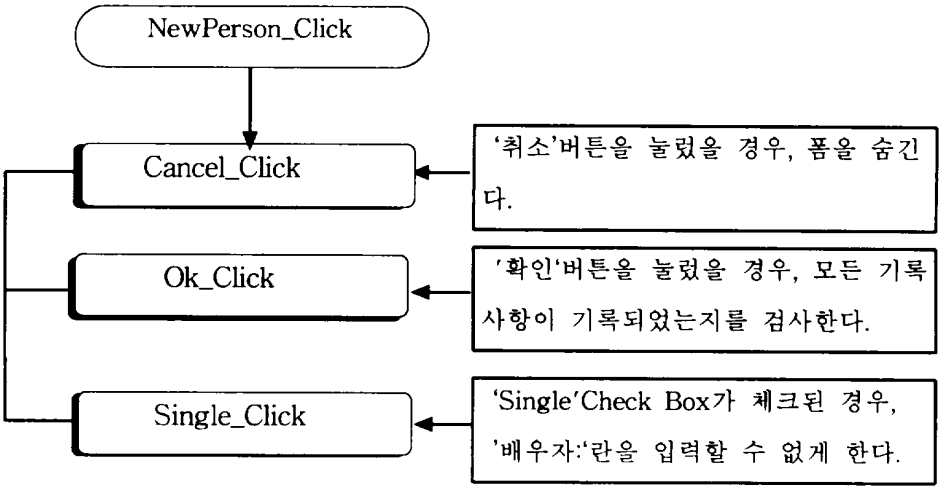


Fig. 12 Main form for inference engine

2) 구성원 추가 입력 폼 설계



여기서는 기존의 데이터베이스를 열고 새로운 구성원을 추가할 경우를 위하여 대화상자를 출력하고 사용자로부터 입력을 받아 이를 직접적으로 데이터베이스에 기록되도록 한다. 이벤트 발생은 사용자가 대화상자를 열었다가 취소를 하는 경우와 새로운 구성원을 입력 받았을 경우의 구성원 무결성 제약조건 검사에 관한 것이다.

새 구성원을 입력하는 입력창의 설계화면은 다음의 Fig. 13와 같다.

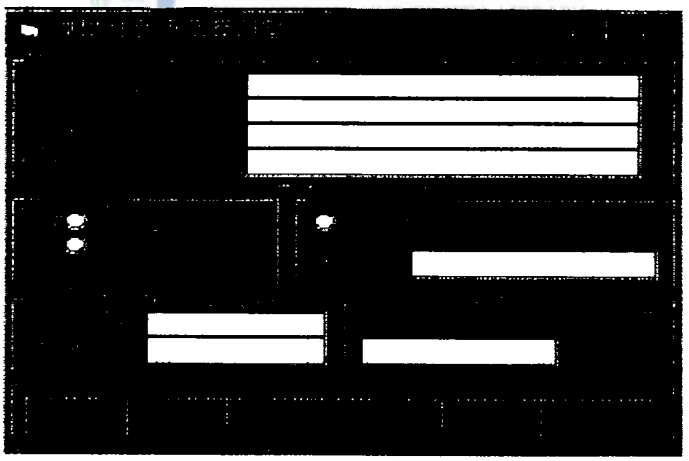
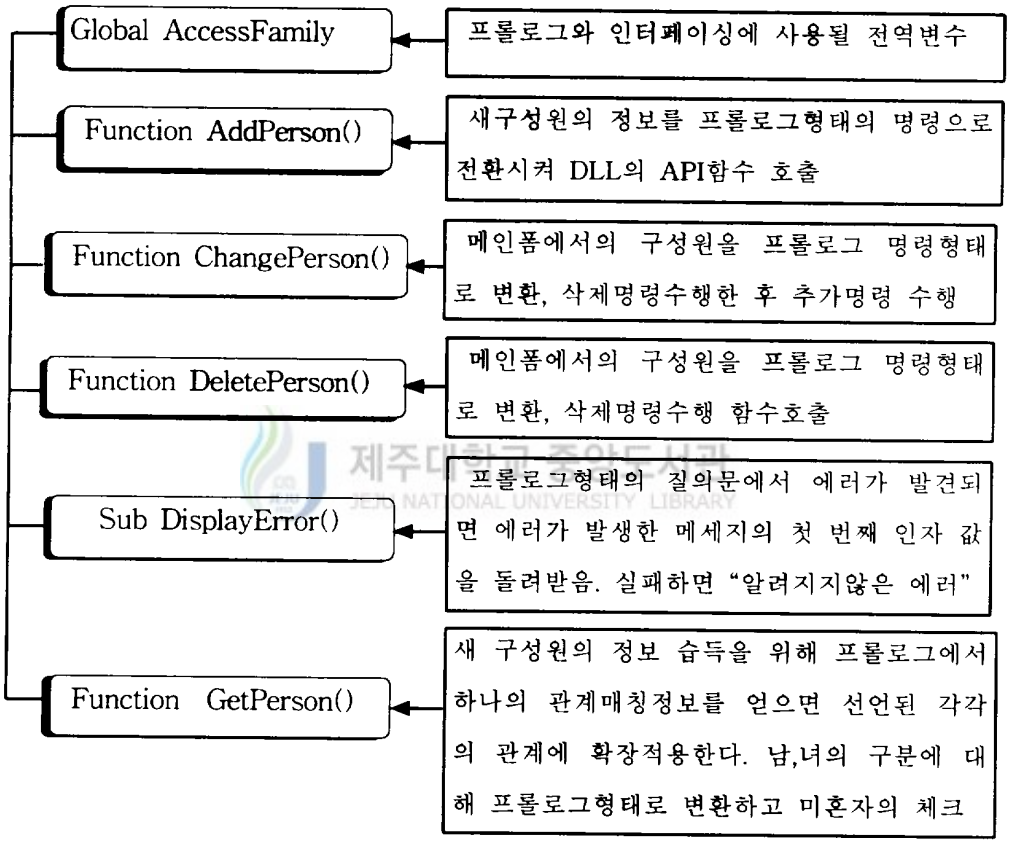


Fig. 13 Input form of new member

3) 프롤로그 호출 루틴 설계

이 부분은 전위 시스템 내부에서 비주얼 베이직과 프롤로그의 명령처리를 위한 API 함수들을 호출하는 부분으로 추론엔진동작을 위해 API선언 모듈과 함께 가장 중추적인 역할을 담당한다. 즉, 동적 연결 라이브러리에 선언되있는 함수를 이용하기 위해 프롤로그 호출 코드를 작성한다. 각각의 함수에서는 프롤로그 명령문의 질의형태로 변환한 후, DLL에 선언된 API함수로 변수와 인자로 값을 넘겨주고 다시 변수를 통해 추론결과를 넘겨받는 역할을 수행한다.



4) 프롤로그 변환 함수 선언 모듈설계

이 부분은 실질적인 프롤로그 프로그램 인터페이스로 비주얼 베이직에서 처리되는 데이터를 전달받아 MS C++로 컴파일되있는 Logic.dll 내부의 함수들을 Windows API를 통해 프롤로그 명령으로 변환 수행한 뒤, 값의 참조에 의해 다시 되돌려 받는 역할하는 비주얼 베이직 함수들을 정의한다.

다시 말해서 비부얼베이직으로 처리될 명령을 프롤로그가 수행할 수 있도록 중간 변환을 담당하는 부분이다. Logic.dll에 내부적으로 선언되어 있는 API함수들의 처리 형태는 데이터베이스에서의 ODBC와 비슷하게 처리된다. 이는 비주얼베이직의 변수를 프롤로그 명령에서의 술어와 항으로 매핑시켜서 전체적으로는 한 단위의 질의문으로 만드는 역할을 한다. Logic.dll을 통한 API의 역할을 간단히 설명하면 다음과 같다.

- 로드된 프롤로그 파일에 대한 consult명령 수행
- 비주얼 베이직에서 문자데이터를 넘겨받고, 프롤로그 질의문을 형성
- 검색 해를 얻기위한 redo 명령 수행
- 성공한 결과에 대해 비주얼베이직의 원래 변수로 redirection
- 실패한 결과에 대해 비주얼베이직이 에러를 처리할 수있게 에러 값 redirection

비주얼 베이직 내부모듈에서의 프롤로그 질의문으로의 변환형식은 예를 들면 아래와 같다.

```
'프롤로그 명령 <relationship> ( X, <person> ) 의 형식으로의 변환 과정
 tf = CallStr ( Term, Relationship + "( X, '"+ Person + "' ) " )
'관련된 모든 구성원을 얻고, 관계된 구성원의 리스트에 추가하기 위한 루틴
 While ( tf = True )
     GetArg ( Term, 1, bSTR, StrVal )
        RelatedPersonList.AddItem StrVal
        tf = Redo ( )
 Wend
```

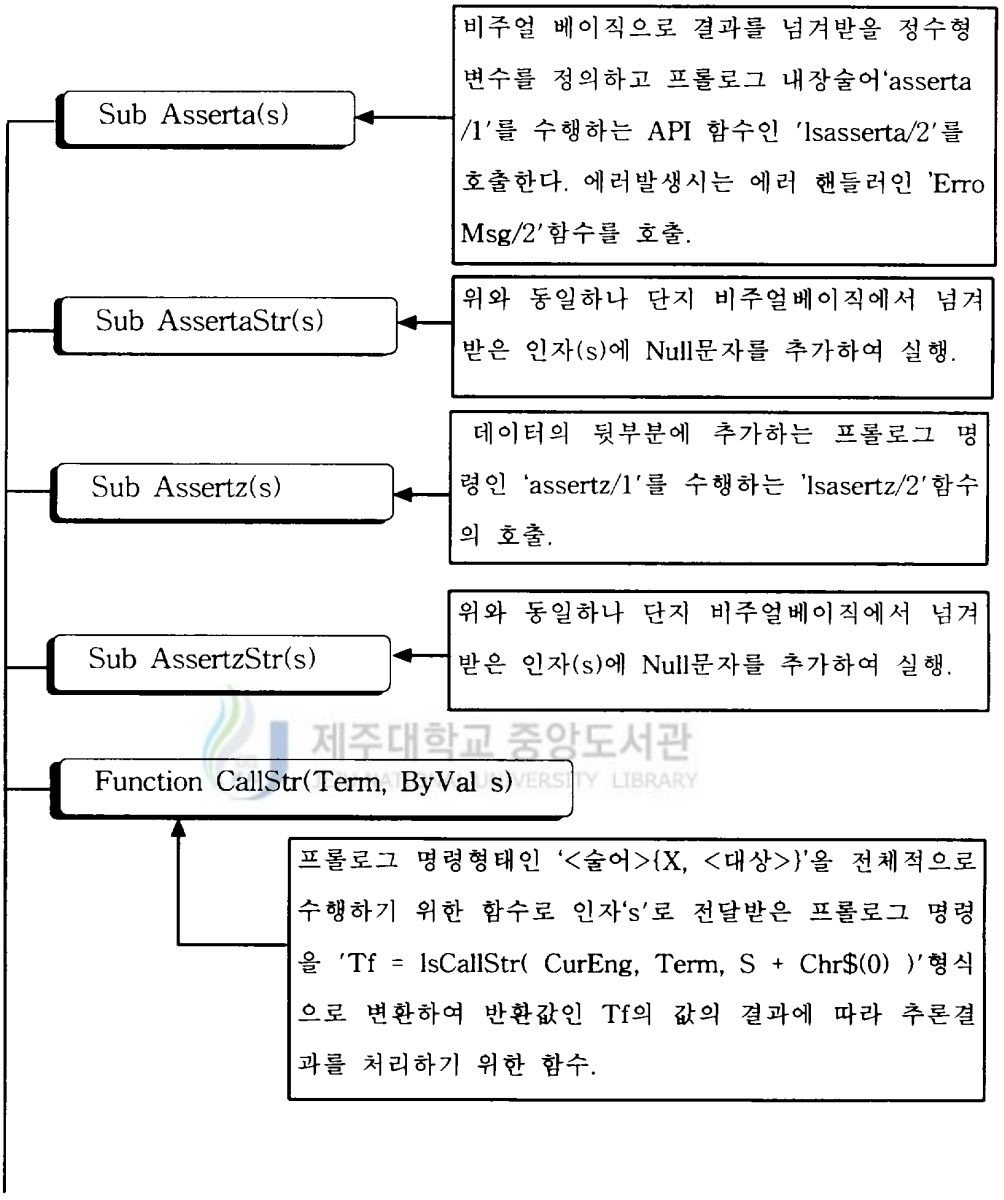
여기서 첫 번째 문자열은 질의문의 형식을 포함한다. 이 경우는 어떤 특정인을 사용자가 선택했을 때, 관계된 모든 사람을 찾기 위한 프롤로그 질의문으로의 변환인 것이다. 한번에 관계인을 모두 찾기 위하여 Logic.dll 내부함수인 CallStr()과 Redo() 함수를 호출한다. 질의문의 X에 대한 답을 찾은 결과를 GetArg()가 첫 번째 매칭된 값으로 넘겨주고, 다시 관계된 구성원들의 리스트에 추가되도록 하는 루틴이다.

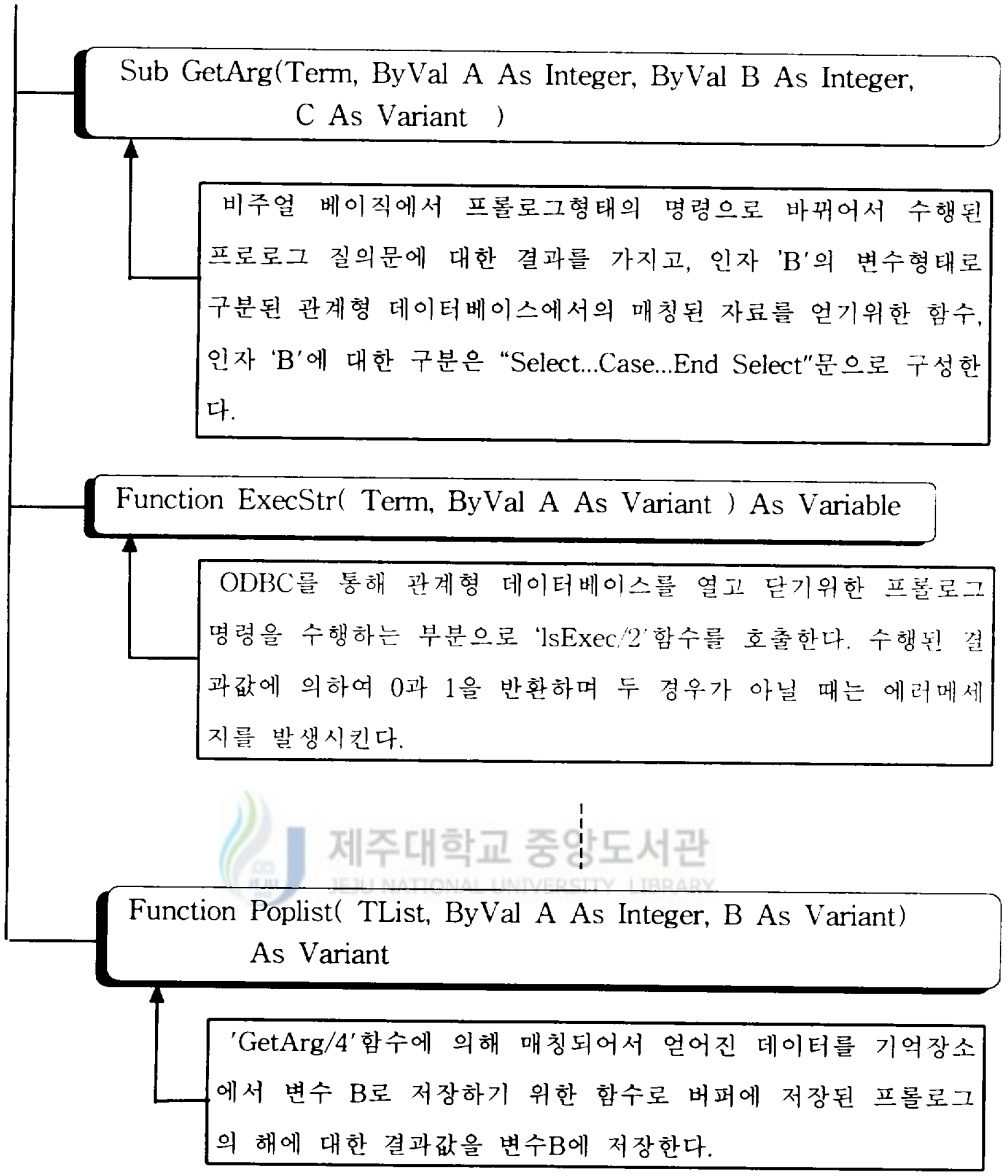
이 모듈에서는 이러한 역할을 하는 Logic.dll내의 함수와 연결을 담당한다.

< 함수 선언부 >

```
Option Explicit
Declare Funtion Asserta Lib "Logic.dll" Alias "lsAsserta" ( p1 As Any, ByVal p2
        As Any ) As integer
Declare Funtion AssertaStr Lib "Logic.dll" Alias "lsAssertaStr" ( p1 As Any,
        ByVal p2$ ) As integer
Declare Funtion Assertz Lib "Logic.dll" Alias "lsAssertz" ( p1 As Any, ByVal p2
        As Any ) As integer
Declare Funtion CallStr Lib "Logic.dll" Alias "lsCallStr" ( p1 As Any, ByVal p2
        As Any ByVal p3$ ) As integer
                :
Declare Funtion Redo Lib "Logic.dll" Alias "lsRedo" ( p1 As Any) As integer
Dim CurEng
```

< 함수정의부 >





추론기능을 보완하기 위한 위의 모듈부분을 포함해서 본 논문에서의 제안된 방식에 의한 프로토타입 설계화면의 연결된 구성부분을 나타내면 다음과 같다.

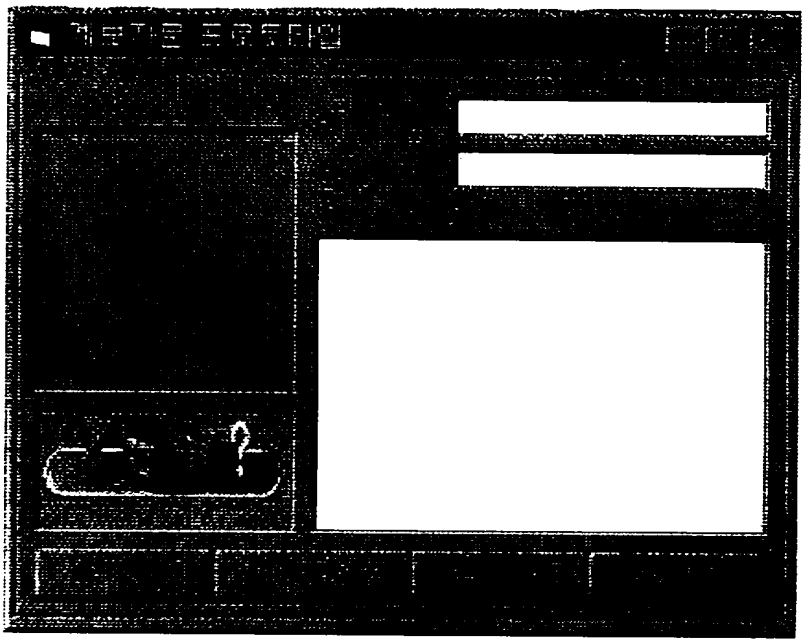


Fig. 14 Personal information form about inference result

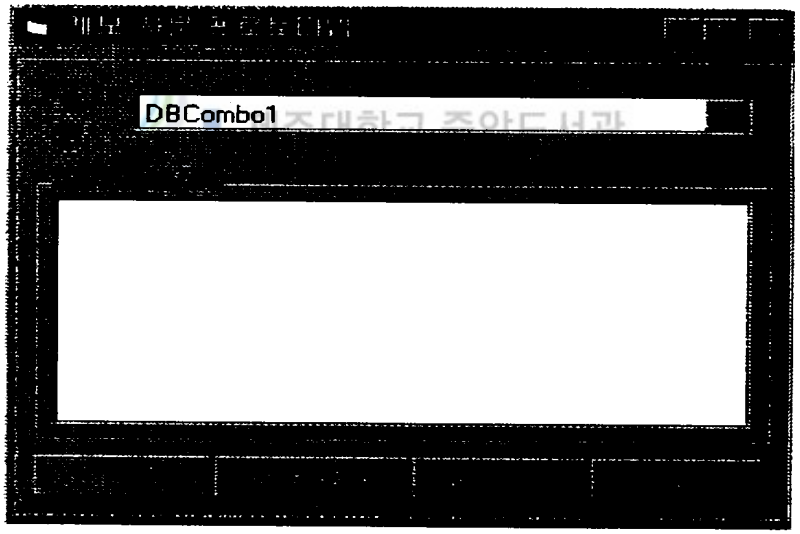


Fig. 15 Explanation form for relational name

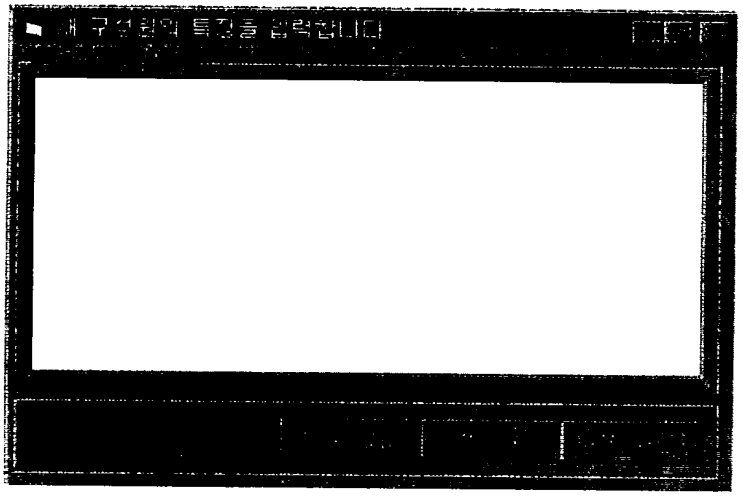


Fig. 16 The character input form of new member

Ⅳ. 시스템 구현 및 고찰

본 장에서는 제안된 방식에 의해 설계된 계보자문 프로토타입을 구현하고 이에대한 구현환경과 구현된 내용을 기술한 후, 평가와 고찰내용을 서술한다.

1. 계보 자문 프로토타입(Lineage Consulting prototype)의 구현

1) 구현환경

본 논문에서는 제안된 방식의 프로토타입 실행환경을 윈도우 기반 운영체제를 고려하여 32비트체제로 하였다. 적용된 프로토타입의 구현환경은 아래와 같다.

- 중앙처리장치 : Pentium 133MHZ
- 운영체제 : 윈도우즈 95
- 메모리 : 32MByte
- 윈도우즈 프로그래밍 언어 : 비주얼 베이직 4.0, SWI-Prolog 1.19,
- 관계형 데이터베이스 : MS ACCESS 7.0

비주얼 베이직을 호스트 언어로 적용한 사용자 질의 인터페이스를 중심으로 프롤로그와 질의 데이터를 동적 교환하는 부분(함수선언)인 윈도우즈 라이브러리를 이용하여 실행한다. 또한 관계형 데이터베이스를 ODBC로 연결하여 리스트 박스에 데이터와 관계를 디스플레이하는 기능까지를 포함시킨다. 이는 DLL에 선언된 API를 이용하여 추론이 필요한 부분에 프롤로그를 호출하면서 연결 작업을 수행한다.

2) 구현내용

본 논문에서 제안된 방식으로 설계되고 구현한 프로토 타입의 실행화면을 각각의 단계별로 나타내면 다음과 같다.

Fig. 17 The start form

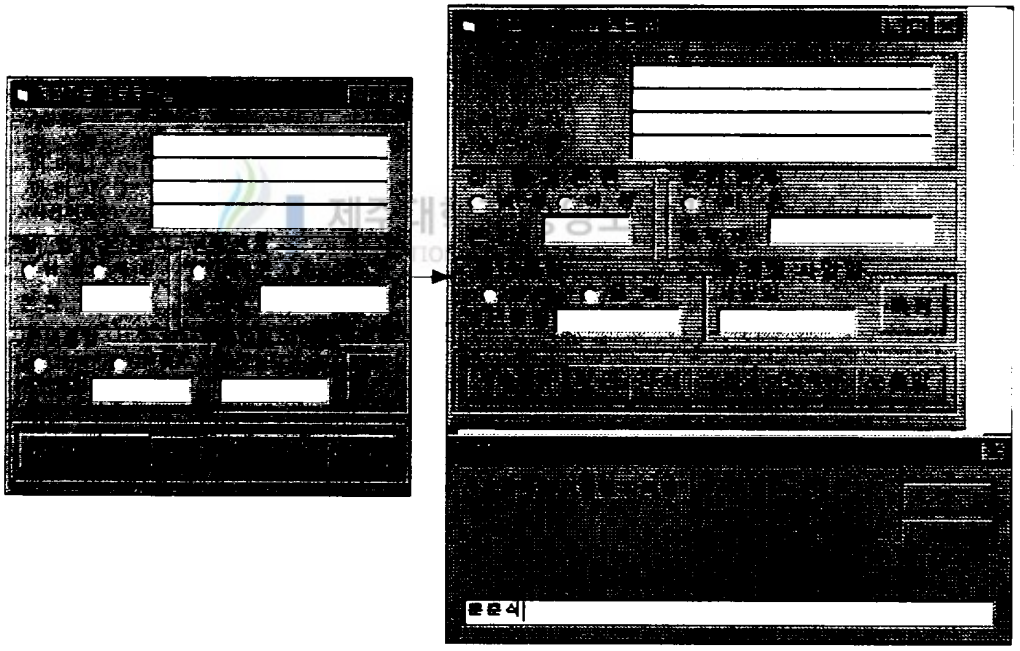


Fig. 18 Input of new member

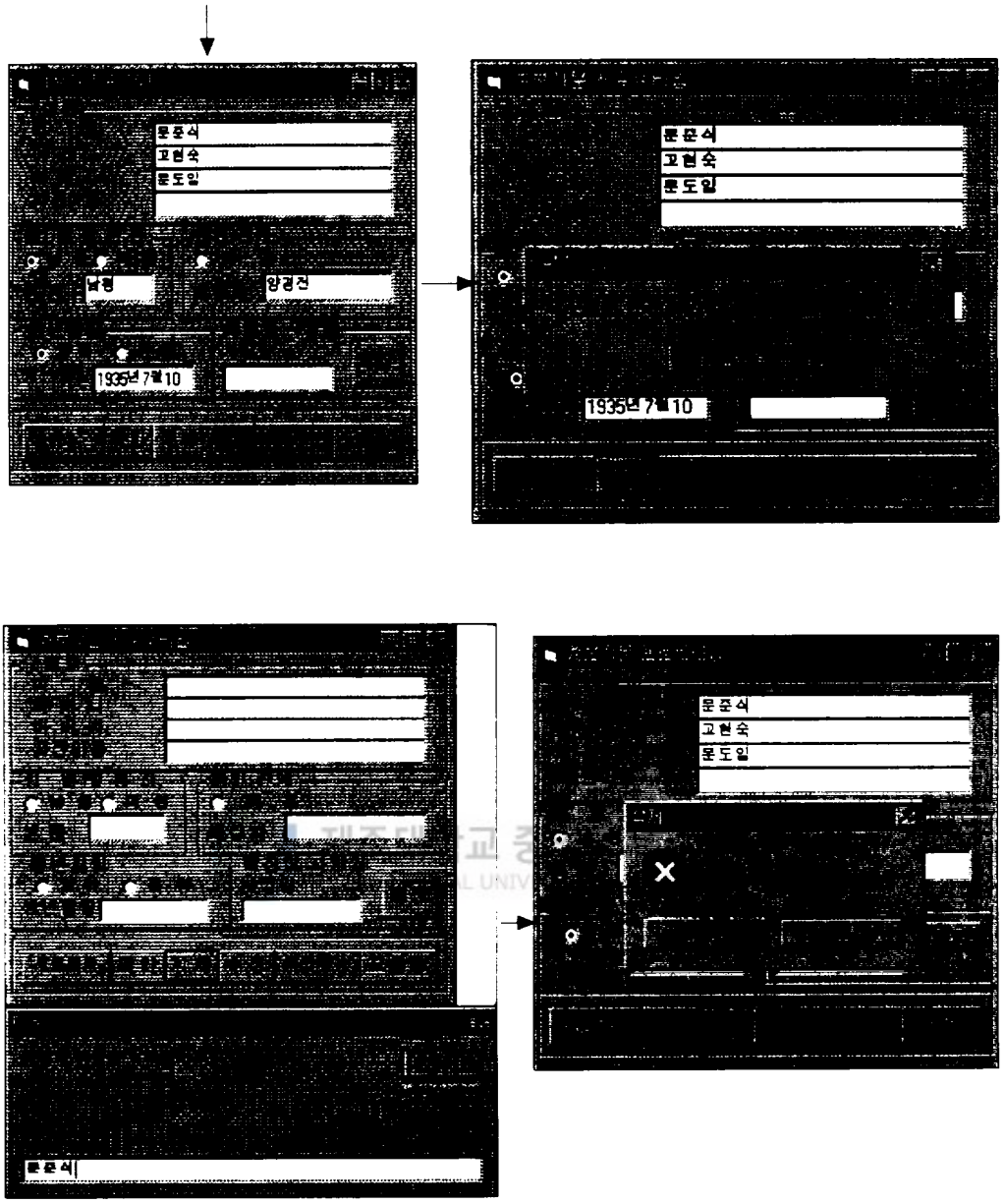


Fig. 19 Delete of new member

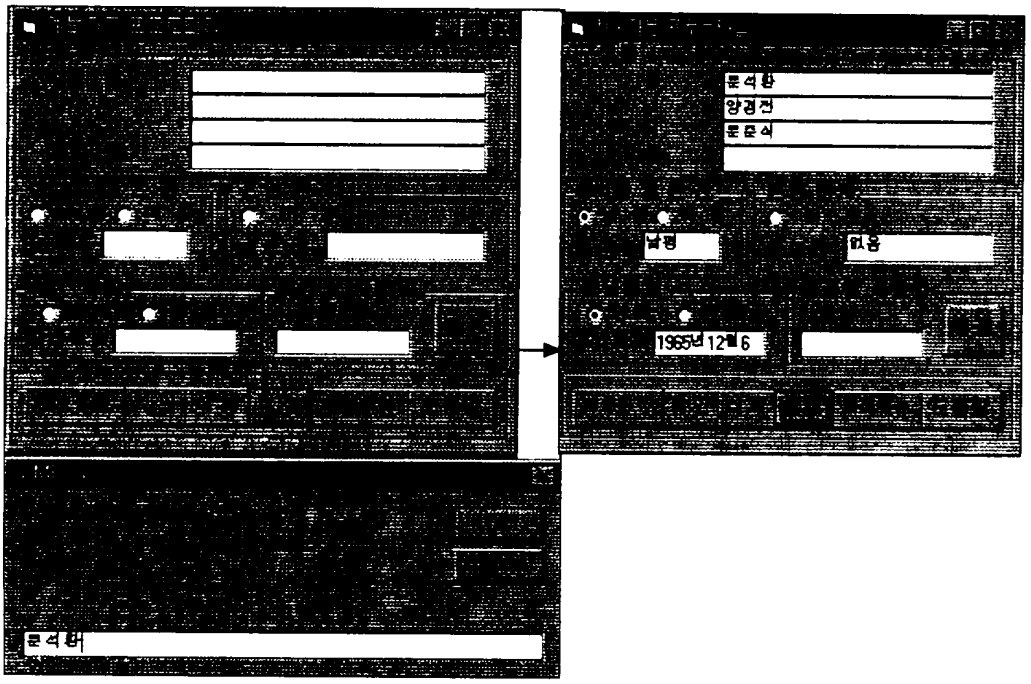
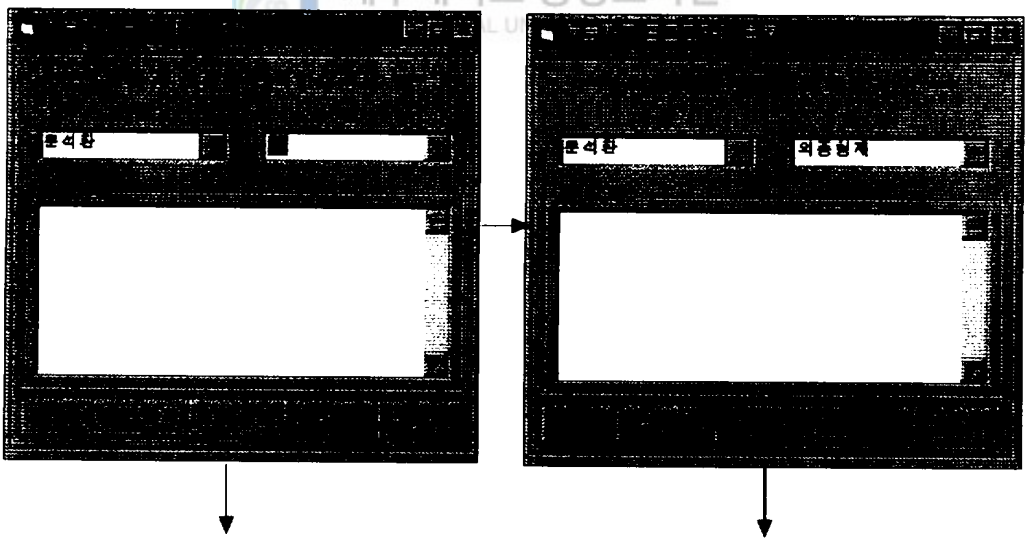


Fig. 20 Updata of member

· 질의 추론 화면 ·



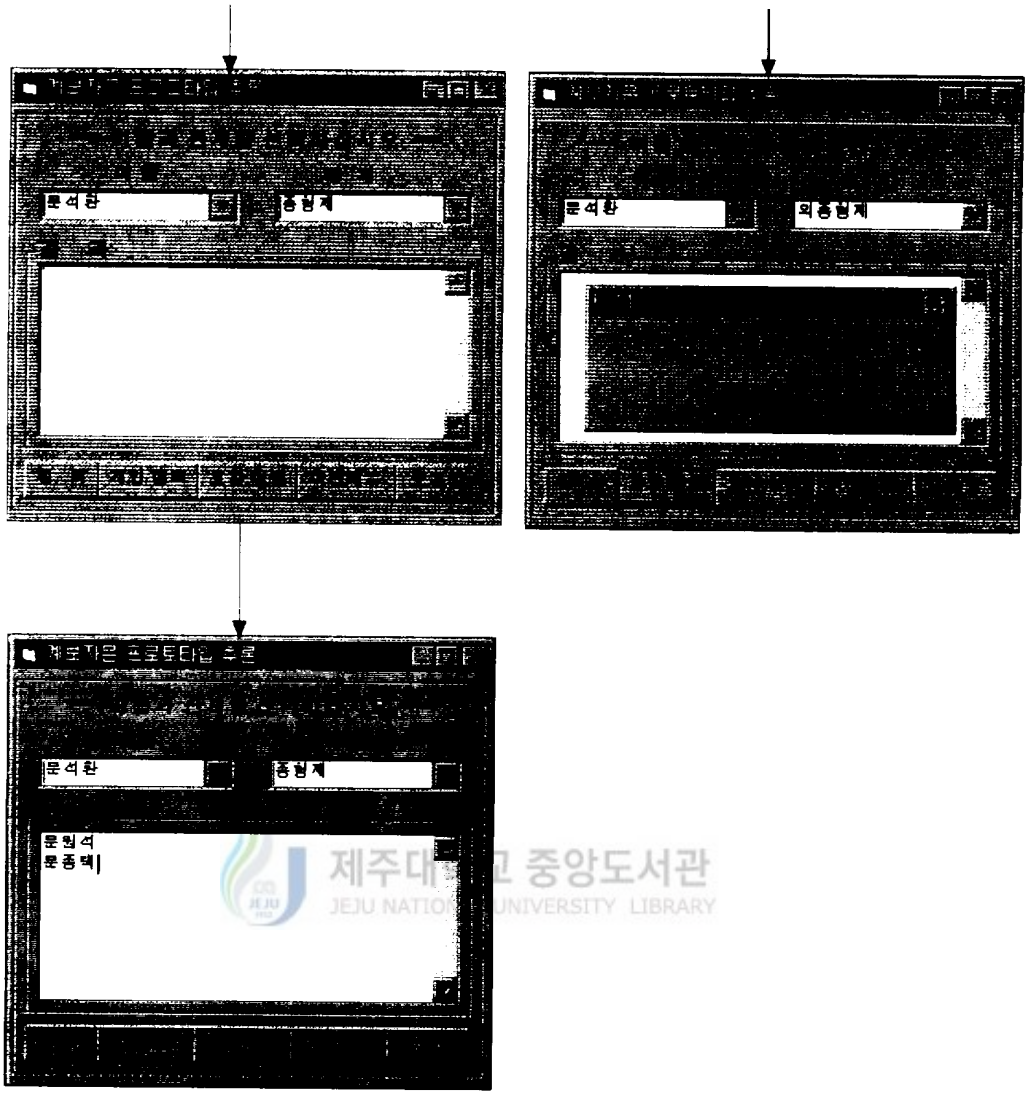


Fig. 21 Inference within relational database

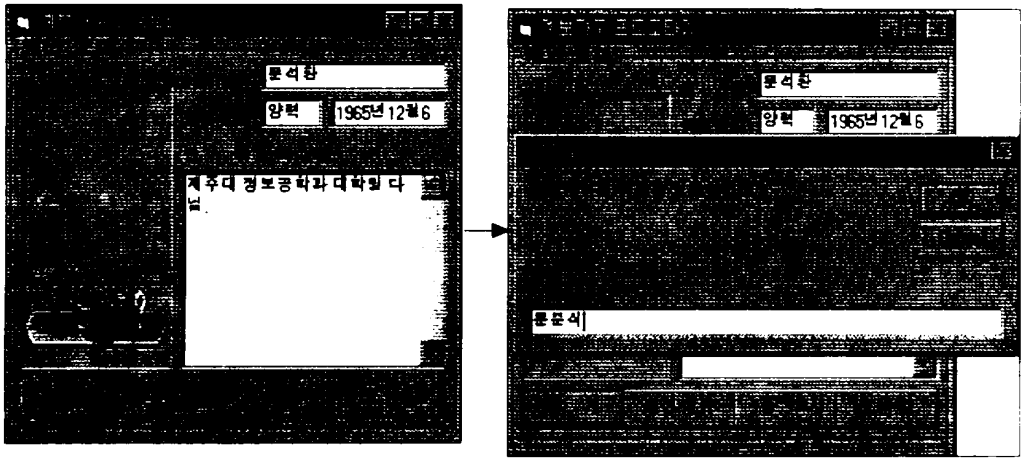


Fig. 23 Personal information about inference result

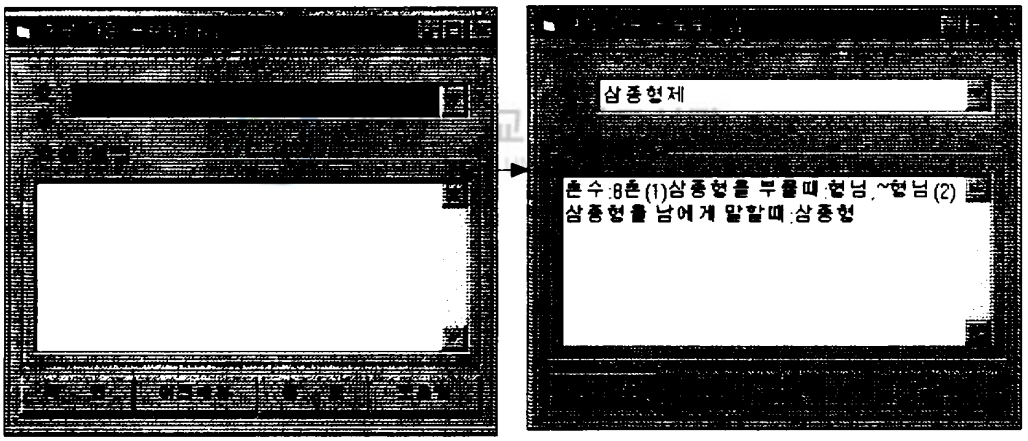


Fig. 24 Explanation for relational name

2. 결과 및 고찰

본 논문에서 제안한 방식으로 프로토타입을 설계하고 구현하는 적용실험에서는 로직베이스를 기반으로하는 추론엔진을 구축하고 이를 다시 비주얼베이직내로 내부 모듈화시켜 윈도우즈 프로그래밍을 이용한 지속적인 추론이 가능하도록 하였다. 추론의 핵인 빠른 패턴매칭을 위하여 관계형 데이터의 객체화를 시도하였고, 이는 프롤로그에 적용되어 추론이 가능한 추론 인터페이스를 구축하였다. 또한 여기에 기존의 관계형 데이터베이스와 추론엔진의 결합방안과 초보자라도 쉽게 사용할 수 있는 적응성이 뛰어난 사용자 인터페이스의 개발을 기본목표로 하였다. 본 실험에서는 완전한 자연어 인터페이스 구축에는 못미치지만 기존의 프롤로그를 사용하는 방식, 즉 도스나 유닉스와 같은 프롬프트 방식인"?-"에 대해 프롤로그로 작성된 술어를 개발자와 같은 방식으로 입력해서 추론해야하는 사용자의 어려움을 일차적으로 해결할 수 있었으며, 윈도우즈 환경에서의 윈도우즈 프로그래밍언어의 장점을 이용한 프롤로그 추론기능 확장의 접목 가능성을 제시하였다.

실험의 전반적인 효과로는 로직베이스를 활용함에 있어 초보자라도 쉽게 시스템에 접근할 수 있는 사용자 환경을 구축하였고, 로직베이스를 사실베이스와 분리하여 내부 모듈화 함으로써 동적 메모리 할당을 통한 빠른 추론을 실현시킬 수 있었다. 또한 추론기능 이용시의 각각 분리된 환경하에서의 개발방식을 다양한 방법으로 사용자 정보를 유지, 관리할 수 있게 하였다.

본 프로토타입을 위해 적용된 한국적 가족체계의 정형화를 통한 계보자문 프로토타입의 구현은 컴퓨터의 보급이 급속히 확산되고 적용범위도 확대됨에 따라 범용성을 갖추기 위해 일반 퍼스널 컴퓨터에서의 구현을 기준으로 하였다. 이에 대한 확장가능성은 일반적으로 '족보'라 불리우는 한국형 가족체계를 중심으로 데이터베이스의 간단한 추가작업만으로 확장시킬 수 있으며, 관공서등의 인구조사기관에 있어서 '지역주민 일람표'등의 부분에 적용되어 질 수 있다. 또한 지식베이스와 관계형 데이터베이스의 부분적 수정을 통해 '관광 자문 시스템'이라든가 '병원 진료 자문 시스템'등의 분야로 확장 적용할 수 있다.

본 연구에서의 가장 어려웠던 부분은 호스트 언어인 비주얼베이직에 프롤로그의 추론엔진을 모듈화 시키는 작업이었다. 이는 비주얼베이직과 프롤로그 명령체계로의 변환을 담당하는 변수의 전달과정에 의한 것으로 변수의 형태에 따른 실행함수의 구분을 해가는 과정이 어려움을 주었다. 또한 객체의 개념을 도입하였으나 객체의 특성을 표현하는 데 있어서는 제약사항으로 인한 지식표현의 한계성을 느껴야 했다. 또한 전반적인 추론 기반 시스템을 위한 보완사항으로는 여러 지식공학자를 위한 데이터 공유성을 포함할 수 있는 공동 작업 공간 관리 시스템의 개발이 필요하다.

V. 결론

본 연구에서의 시도는 제한적이긴 하지만 데이터에 대한 논리적 의미의 추론기능을 보유하고 이를 이용하여 사용자에게 좀 더 친숙한 환경인 윈도우즈 프로그래밍을 통한 그래픽 사용자 인터페이스를 제공하고자 하였다.

따라서 본 논문에서는 관계형 데이터베이스와 이에 대한 표현방식으로 술어로직을 적용하여 로직베이스를 구축하고, 이를 토대로 개인간 정보를 저장한 관계형 데이터베이스와 로직 기반 시스템을 정형화 하였다. 이에 대한 또 하나의 새로운 접근방식으로 윈도우즈 환경을 기반으로 하는 윈도우즈 프로그래밍 기법을 접목하여 사용자 위주의 자문 시스템 개발방식을 제안하였다.

이러한 접근은 기존 개발자 중심의 제 5 세대 언어인 프롤로그의 인터프리터 환경을 다양한 윈도우즈 프로그래밍과 결합하는 방법론을 부수적으로 가질 수 있었다.

이 방식은 최근 다양하게 지원되는 윈도우즈 프로그래밍 언어인 비주얼 베이직과 인공지능 언어인 프롤로그의 효율적 접목으로 ODBC를 통해 이와 연결되어 질 수 있는 다양한 형태의 관계형 데이터베이스 관리를 시도하였다.

본 시도에서의 강점을 간단히 나열하면 아래와 같이 요약된다.

첫째, 기존의 관계형 데이터베이스에서 관계해석을 통한 추론기능을 보강해서 외부 데이터베이스의 중복성을 제거하여 개별적인 데이터를 관리하기에 효율적이라는 점.

둘째, 사실베이스를 직접적으로 외부 데이터로 저장시키고 로직베이스만을 메모리상에 상주시킴으로써 메모리의 과다한 부담없이 추론과 의미론적인 데이터 무결성 처리를 할 수 있다는 점.

셋째, 프롤로그를 모르는 일반 사용자도 직접적으로 데이터의 추가, 삭제가 가능하고 사용상의 친숙함과 편리성을 제공한다는 점.

넷째, 텍스트 형태의 한정적인 인공지능 개발환경을 벗어나 다양한 사용자 위주의 인터페이스를 개발해 낼 수 있다는 점이다.

본 논문의 진행과제로서 추후 연구 방향은 기존의 데이터베이스에서 객체와 속성

에 대한 접근을 시도하여 지식을 발견해내는 지식 발견 시스템을 개발하는 것이고,

전반적인 향후 연구 과제로는 화학과 생물, 의학등의 자연과학 분야와의 연계를 통해 연구분야에 적용되는 불확실한 물성데이터를 처리하기 위한 통계적 퍼지이론을 기반으로한 지속성을 지닌 전문가 시스템의 개발이다.

참고문헌

Shekhar. S, Hamid zadeh. B, and Kohil. A. , 1994, "Learning Transformation Rules for Semantic Query Optimization: A Data Approach", IEEE Trans. on Knowedge And Engineering, Vol.5, No.3. pp 71~80

Stonebraker. M. ,1983, " Implimentation of Rules in Relational Database System", database Engineering, Vol. 6, No. 4, pp 33~41

Tsichritzis D.C. and Locchovsky, F.H. 1982, "Data Models", prentice-Hall .

Reiter. R, 1978, "On Closed World Database , In Logic & Databases", Plenum Press, NY,

Waker. A, Sowa. J. F ,1987, "Knowledge System & Prolog", IBM, T.J.Watson Research Center,

Grant. J, 1988, "Logical Introduction To Database", Brace. H Javanovich,Publis -hers,

Ceri, S. G. Gottlob, & G. wiederhold, 1986, "Interfacing Relational Databases And Prolog Effecciency", Proceding Of First International Conference On Exp -ert Database System, , pp.141~153.

Han. J, Cai. Y & Cercone. N, 1991, 7 "Concept-Based Data Classification in Relational Databases" , in Workshop notes of 1991 AAAI Workshop on Kno -wedge Discovery in databases(KDD'91),Anaheim,CA, pp 77~94

Han.J, Huang.Y, Cercone.N, and fu.F, 1996, "Intelligent Query Answering By Knowledge Discovery Techniques", IEEE *Transactions on Knowledge and Data Engineering*, 8(3):373-390

Maier.D, 1983, "The Theory of Relational Databases", Computer Science Press pp.224-243

조태남, 1995, "이론과 설계 데이터베이스 시스템", 대은 출판사.

김화수, 고순주, 1993, "인공지능의 이론과 실제" ,집문당

김화수 조용범, 최종욱, 1995, "전문가 시스템" ,집문당,

Sheuniderman, 1991, "Design of User Interface", Prentice Hall

김승광, 1991.12, "지능형 지리교육 시스템을 위한 자연어 인터페이스 활용에 관한 연구", 계명대학교 전산대학원 석사논문.

EDPS 연구회, 1993.6, 「컴퓨터 용어 대사전」, 대은출판사 pp 571.

Peter Noton, Harold, Phylips Davis, 1996.4, "Visual Basic For Windows95", Inf-obook

James L.Conger, 1995.3, "Windows API Bible", 정보문화사

전용진 編, 1989, "IBM PC XT/AT를 활용한 인공지능 TURBO PROLOG", 크라운 출판사,

전일수, 이상조, 1994.10, "객체지향 데이터 모델에서 일반화/세분화 및 집단화 관계를 이용한 집중화 기법", 정보공학회논문지, 제21권 제10호, pp1840-1847.

Bancihon F, 1988, "Object-Oriented Database Systems" Preceedings of the Ninth ACN SIGACT-SIGMOD-SIGART Symposium on Principles of Database System.

김형수 외3인, 1995.10, "대학입학 전형자료의 검색을 위한 로직 프로그래밍", 한국정보과학회지 '95 가을 학술발표논문집(A), 제22권 2호, pp 433~436.

노대식, 1995.12, "지속성을 부여한 멀티미디어 지식베이스의 구축", 경북대학교 대학원석사논문.

감사의 글

시작이 엇그제 였던 것 같은데 시간이 이렇게나 빠르게 흘러 대학원에서의 2년이란 세월이 지나가 버렸읍니다. 정신없이 지내오느라 최선의 노력을 경주하지 못한 저에게 그나마 기대를 저버리시지 않고 물심 양면으로 도움을 주신 분들게 감사드립니다. 논문이 나오게 되기까지 세심한 지적과 조언으로 많은 가르침을 주시고, 항상 관심을 갖고 격려의 말씀을 해주시며 지도편달을 아끼지 않으신 김장형 지도교수님과 이상준교수님, 그리고 논문심사 기간동안 적절한 지적과 격려를 해주셨던 안기중 교수님, 곽호영교수님, 변상용 교수님, 송왕철 교수님께 많은 감사 드립니다. 또한 지난 학기동안 본 연구가 이루어지도록 주위에서 지속적으로 지켜봐 주시고 물심양면으로 도와주신 한라전문대학교의 김형수 교수님께 감사 드립니다. 그리고 같이 어려운 시기를 이겨나가게 해준 제주전문대학교의 김대영 선배님과 연구실 동료인 성실한 이유경양에게도 감사의 말씀을 전합니다. 웃을 날이 멀지않음을 알려주고 마지막까지 열심히 노력한 행진 후배, 정희, 영민, 영호에게도 같이 고마움을 나눕니다. 열심히 연구하는 중에도 항상 걱정해주던 회국이와 옆에서 헌신적인 노력을 아껴주지 않은 송병운 군과 평소 관심어린 눈으로 기꺼이 조언을 해준 문치웅 선생과 전영회 선생에게도 깊은 감사를 드립니다.

끝으로 오늘이 있기까지 변함없는 기대와 사랑으로 힘을주고, 편찮으신 중에도 용기를 돋워주신 부모님과 가족들에게 이 논문을 바칩니다.