온톨로지 활용도를 높이기 위한 분산 온톨로지 저장소의 사용의 필요성

이기원[O] 양정진
가톨릭대학교
greyo0116@hanmail.net, jungjin@catholic.ac.kr

Need Of Employing Distributed Ontology Repositories for Enhancing Ontology Utilization

KiWon Lee[O] JungJin Yang
School of Computer Science and Information Engineering
Ther Catholic university of Korea

요 약

오늘날 온톨로지의 활용이 높아짐에 따라서 점점 더 방대한 양의 온톨로지가 구축이 되어가고 있다. 그러나 각 분야에 따라서 온톨로지의 분야와 분류가 다르고 내부에서 처리하는 데 걸리는 시간도 다르기 마련이다. 이러한 문제로 방대한 양의 온톨로지로 구성된 온톨로지 저장소는 검색 및 저장의 효율성이 떨어질 수 밖에 없다.

본 논문에서는 이러한 상황의 온톨로지 검색과 저장의 효율성을 높이고자 온톨로지 저장소는 분산되어져야 함을 실제 사례를 들어 보이고 분산 온톨로지 저장소 사용의 필요성을 보여준다.

1. 서 론

오늘날 사회에서 다루는 정보량은 날이 갈수록 급증하고 있다. 날이 갈수록 늘어가는 정보들을 다루기 위해 메타데이타를 이용하여 필요한 정보에 쉽게 접근하는 방법이 제안되고 있다.[1]

이러한 메타데이타를 바탕으로 이용한 온톨로지는 매우 많은 분야에서 활용이 되어지고 있다. 온톨로지의 활용이 높아짐에 따라서 온톨로지의 크기는 점점 더 방대하게 구축이 되어 지고 있다. 비록 각 분야에 따라서 온톨로지의 분류가 다르고 구성이 틀리고 내부에서 처리하는 데 걸리는 시간도 다르기 마련이지만 온톨로지의 사용이 커짐에 따라서 온톨로지를 저장하기 위한 저장소의 사용은 피할 수 없게 되었다. 하지만 단일 온톨로지를 사용할 경우에 온톨로지가 커짐에 따라 추론시간이 늘어날 것이라 예상된다.

따라서 본 논문에서는 온토롤지 활용도를 높이기 위한 분산 온톨로지 저장소 사용의 필요성을 보이기 위해 다음장에서는 관련연구로 Ontology에 대한 조사를 하였고 Ontology를 기술하기 위한 언어의 표준인 OWL (Web Ontology Lanugage) 에 대해서 조사했다.

그리고 온톨로지 저장소로 사용하기 위해 IBM에서 제공하는 Minerva와 Concordia University와 Hamburg University of Technology에서 개발한 LISP 기반의 추론 엔진인 RacerPro에 대해서 조사하였다.

이어지는 3장에서는 이번 연구에서 자동으로 OWL파일을 생성하기 위한 OWLGenerator에 대한 설명으로 예로 들은 book 온톨로지의 구조와 OWL파일 생성원리를 보여 준다.

4장에서는 작업환경 및 실험결과를 보여주었고 5장에서 윗장에서 실험한 결과를 갖고 결론을 맺고 향후 연구 과제에 대하여 논하였다.

2. 관련연구

2.1 Ontology

온톨로지는 어떤 관심 분야를 개념화하기 위해 명시적으로 정형화한 명세서라고 할 수 있다[2]. 즉, 온톨로지는 어떠한 도메인을 구성하는 개념들과 그 개념들이 서로간에 갖는 관계에 대하여 정의하여 사람과 컴퓨터 모두 이해하고 처리할 수 있는 정형화 시킨 방법으로 표현한 것이라 할 수 있다.

이때 구성되어지는 온톨로지의 구조는 트리플(Triple) 구문이라고 하여 주어, 술어, 목적어를 기본요소로 기술하는 RDF (Resource Description Framework)[1]를 바탕으로 한다. Ontology를 표현하는 방법으로 사용되는 언어로 XML (eXtensible Markup Language)[3]을 사용함으로써 위에서 언급한대로 사람과 컴퓨터 모두가 읽고 해석할 수 있다.

2.2 OWL (Web Ontology Language)

W3C에서는 온톨로지를 기술하기 위한 언어로 RDF, DAML+OIL 등을 개발해왔고 2004년 2월 온톨로지 개발을 위한 언어의 표준으로 OWL을 발표했다.[4]

그리고 OWL의 표준은 다음과 같은 6개의 W3C 표준으로 이루어진다.[5]

- OWL 개요서(OWL Overview) : OWL의 구성어휘를 나열하고 각각에 대한 간단한 설명을 제공한다.
- OWL 가이드(OWL Guide): 다양한 예제를 통해 OWL 언어의 활용 방법의 설명을 포함한다.

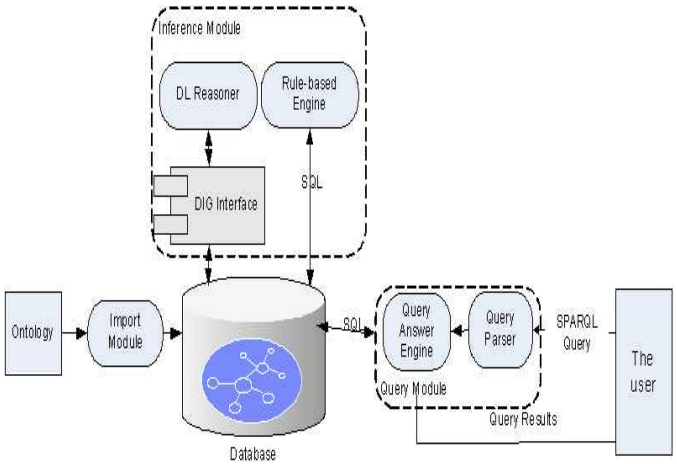
- OWL 참고서(OWL Reference): OWL의 모든 모델링 어휘의 설명을 포함한다.
- OWL 의미론 및 추상 문법 명세서(OWL Semantics and Abstract Syntax) : OWL 언어의 형식적인 표준 정의를 제공한다.
- OWL 테스트 케이스 명세서(OWL Web Ontology Test Cases) : OWL 언어에 대한 테스트 케이스를 포함한다.
- OWL 용례 및 요구사항 명세서(OWL Use Cases and Requirements) : OWL에 대한 요구사항들을 제공한다.

OWL은 표현력의 범위에 따라 OWL Lite, OWL DL, OWL Full 세가지로 분류하며 표현력은 Lite < DL < Full 순으로 커진다.

이러한 OWL의 폭넓은 표현력으로 인하여 온톨로지를 구성한다고 하여도 실제로 이를 OWL등을 이용한 온토로지 언어를 이용하여 표현하는 것도 쉬운 일만은 아니다.

2.3 Minerva

Minerva는 RDB인 DB2 또는 Derby를 이용시 우수한 성능을 내는 온톨로지 저장소이며 RDF[2]를 사용하는 W3C의 OWL(Web Ontology Language)와 SPARQL Query Language를 따른다. Minerva는 Description Logic 추론기를 TBox 추론에 사용하며 논리 규칙의 집합을 번역하여 Description Loginc Programming을 이용하여 ABox추론을 한다. 미네르바는 logic rules의 번역과 OWL 구성을 지원하는 효과적인 온톨로지 추론 둘다를 바탕으로 하는 back-end database 스키마를 구성한다.[6] 다음에 보이는 [그림 1]은 미네르바의 구조를 나타낸다.



[그림 1] Minerva의 구조

2.4 RacerPro

RacerPro는 Concordia University와 Hamburg University of Technology에서 개발한 LISP 기반의 추론 엔진이다.[7]

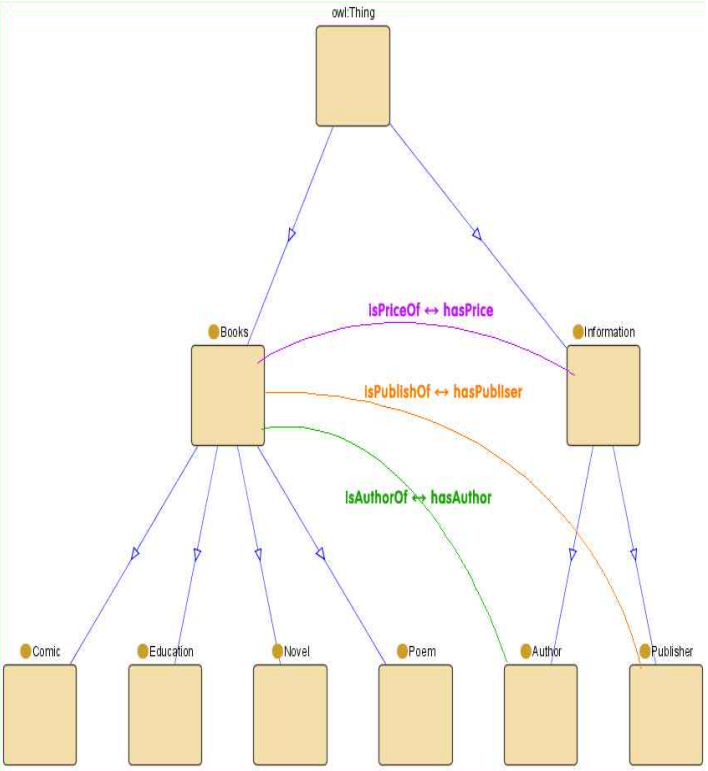
본 논문에서 사용한 RacerPro는 Version 1.9.0을 사용하였다. RacerPro를 사용하게 된 배경에는 추론기의 연계성[8]을 참고 하여 선정하게 되었다. 현재 RacerPro는 Tableaux 알고리즘을 사용한 상용 시스템으로 가장 많이 사용되는 추론 시스템이며, 서술논리의 SHIQ의 표현력을 가진다. 제공하는 질의 언어로는 nRQL을 지원한다.

3. OWL Generator

3.1 book 온톨로지 구조

실험 자료로 사용하기 위하여 간단한 클래스간의 관계로 이루어진 책과 관련된 온톨로지를 구축해 보았다.

[그림 2]에서 보면 알 수 있듯이 책에 대한 정보를 포함한 이 온톨로지는 크게 책과 정보로 나뉜다. 그리고 책에는 소설과 시, 만화 그리고 교육 부분으로만 나누었고 속성으로 책이름을 직접 가지게 된다. 정보에는 저자 정보, 출판사 정보, 가격정보로 나누었다.



[그림 2] book 온톨로지 구조

3.2 OWL 생성원리

위에서 제시한 book 온톨로지 구조를 바탕으로 인스턴스의 개수를 늘려가며 생성시킬수 있는 OWL Generator를 JAVA코드로 구현하였다.

OWL파일에서 온톨로지를 구성을 나타내는 XML을을 분석하여 헤드(Head)와 클래스(Class)의 구성을 이루는 부

분과 테일(Tail)부분을 뽑아내었다.

헤드(Head)와 클래스(Class), 테일(Tail)을 이루는 부분은 한번만 반복되고 인스턴스(Instance)부분들은 여러번 반복되는 점을 이용하여 메인 함수에서 각각의 부분을 생성하는 HeadGen()과 ClassGen(), BookGen(), TailGen()을 [그림 3]과 같이 호출하도록 하였다.

```
//// Head 생성
OG.HeadGen(out);     outbuf.flush();

//// Class 생성
OG.ClassGen(out);    outbuf.flush();

//// 각 인스턴스들 생성
OG.BookGen("Novel",out);          outbuf.flush();
OG.BookGen("Poem",out);           outbuf.flush();
OG.BookGen("Comic",out);          outbuf.flush();
OG.BookGen("Education",out);      outbuf.flush();

//// Tail 생성
OG.TailGen(out);     outbuf.flush();
```

[그림 3] Head, Class, Instance, Tail을 생성하는 JAVA 소스코드

각 인스턴스들은 BookGen() 메소드에서 넘어오는 인자값으로 해당 책들과 관련되어 생성되며 각 인스턴스들의 생성시에는 랜덤함수를 이용하여 각각의 정보 뒤에 숫자로 각각의 정보가 다름을 표현하였다. 예를 들어 저자정보라면 author0, author326식으로 표현하였고 출판사 정보라면 publisher164, publisher912으로 표현하였다. 또한 책 이름이라면 title235, title512처럼 표현하였다. 가격은 1000원단위로 떨어지며 20000원대에 가격이 많이 나오도록 구성하였다. 각각의 정보를 모아 완전한 하나의 인스턴스를 생성하기 위해 [그림 4]에서와 같이 작성하였다.

```
String str ="  <"+ bookName +" rdf:ID=\"book" + i + "\">\n" +
"    <hasPrice rdf:datatype=\"http://www.w3.org/2001/XMLSchema#int\">" + price +
"</hasPrice>\n" +
titleStr +
authStr+
pubStr+
"  </" + bookName + ">\n";
```

[그림 4] 인스턴스를 생성하는 JAVA 소스코드

실제 200개의 소설, 200개의 시, 200개의 만화, 200개의 교육에 대한 4종류의 책에 대하여 0~199까지의 200개의 숫자내에서 임의의 수를 중복 포함하여 200개씩 3종류를 생성하여 각각 제목, 저자, 출판사의 이름으로 사용하고 가격을 갖는 인스턴스들을 생성해보았다.

실행 방법은 java uic.OWLGenerator [OWL파일명]을 입력하면 최대로 생성할 책의 수를 입력하게 되는데 이때 200이라고 입력하게 되면 각 책별로 200의 범위내로 인스턴스 들을 생성하게 된다.

같은 200이라는 숫자를 넣더라고 실행시마다 전혀 다른 인스턴스를 갖는 OWL파일이 생성된다.

위에서 설명한 내용의 결과 화면은 [그림 5]에서 확인할 수 있다.

```
C:₩BookGen>java uic.OWLGenerator
Usage: java uic.OWLGenerator [Filename]

C:₩BookGen>java uic.OWLGenerator 200_1
최대로 생성할 책의 수를 양의 정수를 적으세요(기본:100):200
생성한 소설의 책번호의 범위 : 0~199
생성한 시의 책번호의 범위 : 200~399
생성한 만화의 책번호의 범위 : 400~599
생성한 교육의 책번호의 범위 : 600~799
생성한 총 서적의 갯수 : 800
생성한 각 저자,출판사,제목의 임의 생성범위 : 0~800
file output time : 31ms
filename : 200_1.owl

C:₩BookGen>java uic.OWLGenerator 200_2
최대로 생성할 책의 수를 양의 정수를 적으세요(기본:100):200
생성한 소설의 책번호의 범위 : 0~199
생성한 시의 책번호의 범위 : 200~399
생성한 만화의 책번호의 범위 : 400~599
생성한 교육의 책번호의 범위 : 600~799
생성한 총 서적의 갯수 : 800
생성한 각 저자,출판사,제목의 임의 생성범위 : 0~800
file output time : 47ms
filename : 200_2.owl

C:₩BookGen>dir
 C 드라이브의 볼륨에는 이름이 없습니다.
 볼륨 일련 번호: C495-E5A9

 C:₩BookGen 디렉터리

2008-04-17  오전 12:14    <DIR>          .
2008-04-17  오전 12:14    <DIR>          ..
2008-04-17  오전 12:14           340,015 200_1.owl
2008-04-17  오전 12:14           339,886 200_2.owl
2008-04-11  오전 10:33    <DIR>          old
2008-04-11  오후 09:36    <DIR>          uic
               2개 파일             679,901 바이트
               4개 디렉터리  96,003,551,232 바이트 남음

C:₩BookGen>java uic.OWLGenerator 500
최대로 생성할 책의 수를 양의 정수를 적으세요(기본:100):500
생성한 소설의 책번호의 범위 : 0~499
생성한 시의 책번호의 범위 : 500~999
생성한 만화의 책번호의 범위 : 1000~1499
생성한 교육의 책번호의 범위 : 1500~1999
생성한 총 서적의 갯수 : 2000
생성한 각 저자,출판사,제목의 임의 생성범위 : 0~2000
file output time : 63ms
filename : 500.owl

C:₩BookGen>dir
 C 드라이브의 볼륨에는 이름이 없습니다.
 볼륨 일련 번호: C495-E5A9

 C:₩BookGen 디렉터리

2008-04-17  오전 12:14    <DIR>          .
2008-04-17  오전 12:14    <DIR>          ..
2008-04-17  오전 12:14           340,015 200_1.owl
2008-04-17  오전 12:14           339,886 200_2.owl
2008-04-17  오전 12:14           848,751 500.owl
2008-04-11  오전 10:33    <DIR>          old
2008-04-11  오후 09:36    <DIR>          uic
               3개 파일           1,528,652 바이트
               4개 디렉터리  96,002,699,264 바이트 남음

C:₩BookGen>
```

[그림 5] OWLGenerator를 이용하여 OWL파일을 생성한 결과

4. 작업환경 및 실험결과

4.1 작업환경

CPU	Intel DualCore CPU 4300 1.8GHz
Memory	1024MB
HDD	130GB
OS	Windows XP Professional

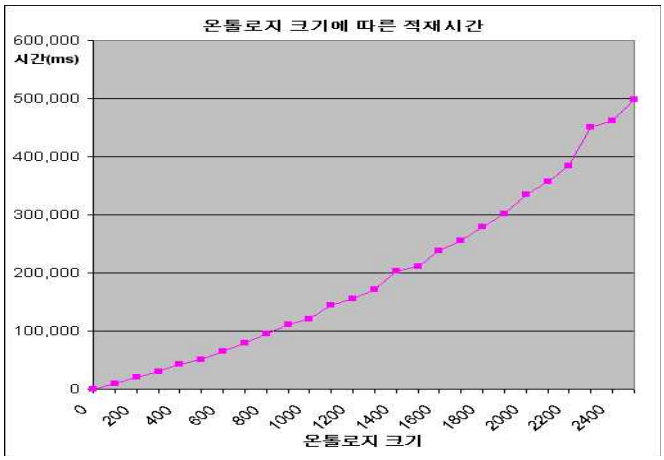
위와 같은 환경에서 Protege 3.3.1버전과 RacerPro 1.9 버전을 이용하여 실험을 진행하였으며 Protege에 OWLGenerator로 작성한 온톨로지 OWL파일을 갖고 실험하였다. BookGen(100)이라는 실험표현은 100개의 소설, 100개의 시, 100개의 만화, 100개의 교육에 대한 4종류의 책에 대하여 0~99까지의 100개의 숫자내에서 임의의 수를 중복포함하여 100개씩 3종류를 생성하여 각각 제목, 저자, 출판사의 이름으로 사용하고 가격은 위에서 언급한데로 하여 인스턴스들을 생성하는 것을 의미한다. BookGen(100)부터 BookGen(2500)까지의 값들을 100씩 증가시키며 입력하여 생성하였다. 그 후 Minerva와 Racer를 이용하여 온톨로지의 크기의 증가에 따른 온톨로지의 평균 적재시간 및 추론시간을 알아보았다.

4.2 실험결과

실험결과는 아래의 [표 1] [표 2] [표 3]과 [그림 5], [그림 6]과 같다.

BookGen(N) 값에 따른 평균 적재시간 (단위 : ms)				
100	200	300	400	500
10,195	20,171	30,218	42,366	51,515
600	700	800	900	1000
64,818	79,568	95,929	111,214	120,248
1100	1200	1300	1400	1500
143,917	155,782	171,445	202,914	211,758
1600	1700	1800	1900	2000
238,068	255,685	279,726	302,141	335,085
2100	2200	2300	2400	2500
357,714	384,727	450,679	462,000	498,452

[표 1] BookGen(N) 값에 따른 평균 적재시간



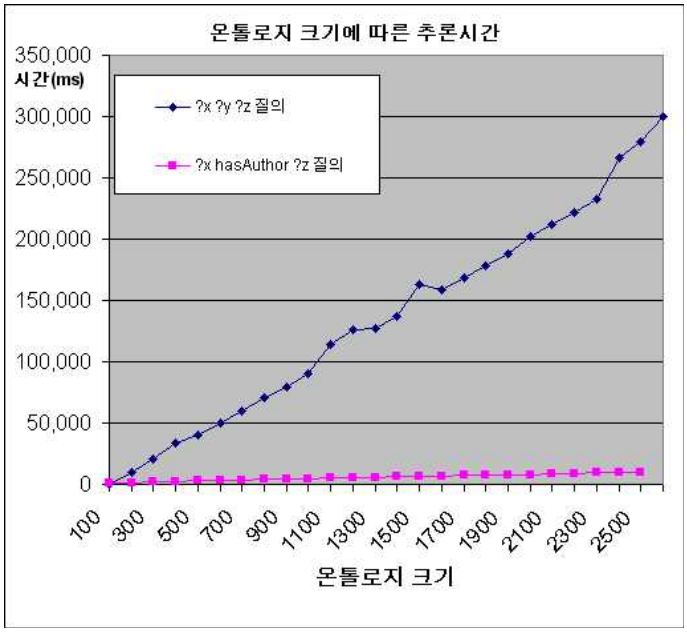
[그림 5] 온톨로지 크기에 따른 적재시간

BookGen(N) 값에 따른 평균 추론시간 (단위 : ms) ?x ?y ?z				
100	200	300	400	500
10,258	20,297	33,446	39,930	50,250
600	700	800	900	1000
59,844	70,282	78,837	89,862	113,851
1100	1200	1300	1400	1500
126,211	126,649	137,227	162,969	158,774
1600	1700	1800	1900	2000
167,996	178,555	187,688	202,584	211,578
2100	2200	2300	2400	2500
221,816	232,153	266,461	278,914	299,859

[표 2] BookGen(N) 값에 따른 평균 추론시간 ?x ?y ?z

BookGen(N) 값에 따른 평균 추론시간 (단위 : ms) ?x :hasAuthor ?z				
100	200	300	400	500
1,046	1,578	2,031	2,391	2,828
600	700	800	900	1000
3,031	3,219	4,859	4,828	4,891
1100	1200	1300	1400	1500
4,953	4,906	5,328	6,188	6,344
1600	1700	1800	1900	2000
6,672	7,328	7,719	7,703	7,984
2100	2200	2300	2400	2500
8,547	8,672	9,814	9,703	10,297

[표 3] BookGen(N) 값에 따른 평균 추론시간 ?x :hasAuthor ?z



[그림 6] 온톨로지 크기에 따른 추론시간

결과를 보면 알 수 있듯이 적재시에 온톨로지의 크기가 100일때는 10000ms 정도의 시간이 걸리고 200일때는 20000ms 정도의 시간 500일때는 50000ms정도의 시간이 걸리지만 1000일때는 120000ms 1500일때는 210000ms 2000일때는 335000ms 2500일때는 500000ms 의 시간으로 결국 온톨로지의 크기가 작을 경우에는 크기에 비례하여 추론적재시간이 증가하지만 온톨로지의 크기가 커질수록 선형적으로 추론적재시간이 늘어나는 것이 아니라 선형적인 그래프 보다 큰 곡선을 그리며 늘어남을 알 수 있다.

5. 결론 및 향후 연구

위의 실험결과에서 알 수 있듯이 단일 온톨로지로 구성되었을 경우 온톨로지의 크기가 커질수록 추론적재시간은 그에 비해 선형적인 그래프 이상으로 추론시간이 늘어나는 것을 확인할 수 있다.

아주 간단한 구조로 이루어진 온톨로지에서도 인스턴스의 값이 늘어남에 따라서 시간이 증가하는 데 복잡한 구성의 온톨로지도 시간이 증가하리라는 것은 자명하다.

비록 도메인마다 시간이 더 커지는 임계지점은 틀리겠지만 이는 어느 온톨로지든 온톨로지의 임계점 직전에서 여러 온톨로지로 분산시키는 방법론이 필요한 것으로 보인다. 그러므로 위의 실험은 분산 온톨로지의 필요성을 보였다.

향후 연구로 특정 도메인에서 실제 사례를 통한 분산 온톨로지 활용을 연구할 예정이다.

후기

본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 지식경제부의 유비쿼터스 컴퓨팅 및 네트워크 원천기반 기술개발사업의 08B3-S1-10M 과제로 지원된 것임.

참고문헌

[1] W3C, RDF Primer, W3C Working Draft 23 January 2003.

[2] Thomas R. Gruber, "Toward Principles for the Design of Ontologies Used for Knowledge Sharing", 1993

[3] W3C, Extensible Markup Language (XML)1.0 (Third Edition), 04 February, 2004

[4] W3C. Technical Reports and Publications http://www.w3.org/TR

[5] W3C. OWL (Web Ontology Language) http://www.w3.org/2004/OWL

[6] IBM, User guide of OWL Ontology Repository (Minerva) http://www.alphaworks.ibm.com/tech/semanticstk

[7] Racer Systems GmbH & Co.KG http://www.racer-systems.com/

[8] Eom Dongmyung, “A Comparison of Ontology Tools Based on OWL”, Korean Journal of Oriental Medicine Vol12. No.1, 2006.