연역 데이터베이스에서 SQL을 이용한 순환적 질의의 설계

*김 영 준 **김 정 태

* 한국외국어대학교 대학원 경영정보학과

** 한국외국어대학교 경영정보학과 교수

Abstract

The relational database management systems sometimes require extremely long and complicated queries for a certain retrieval. In this case, recursive retrievals are more efficient approach than the usual queries. Many researchers have tried to incorporate semantics in the traditional relational models using artificial intelligence techniques. This new concept becomes a deductive database and sometimes it is also called as a logic programming. However, the designer of a deductive database did not overcome the short of relational database.

In this paper, we propose a new way of designing queries for the deductive database. We also provide relations for recursive retrieval in the deductive database. These approaches are applied for the material requirement planning.

1. 서 론

관계형 DBMS(Database Management Systems)는 릴레이션(relation) 형태의 관계형 데이터모델을 사용하는 시스템으로, 대부분의 관계형 DBMS는 시스템내에서의 질의를 검색하기 위해 QBE(Query By Example), SQL(Structured Query Language), Quel 등의 질의어(query language)를 채용하고 있다. 데이터베이스 관리분야에서 관계형 DBMS는 현재 상당한 연구결과를 얻고 있으며, 인공지능(Artificial Intelligence)이론이 점차 정립됨에 따라 데이터베이스의 개념이 확대되어 오늘날에는 인공지능기법을 데이터베이스에 적용하는 것에 대한 필요성과 관심이 점차 증대하고 있으며 관련연구도 활발히 진행되고 있다.

그러나, 현재 관계형 DBMS의 대표적인 질의어로 널리 사용하고 있는 SQL은 인공지능기법, 즉 논리(logic) 프로그래밍 기법을 적용하는데 많은 어려움을 내포하고 있다. SQL에는 논리 프로그래밍 언어에서 아주 빈번히 사용하고, 쉽게 표현할 수 있는 순환적 질의(recursive query)를 처리할 수 있는 SQL 표현식을 허용하고 있지 않다[10]. 그러므로 SQL에 논리기법을 적용하려면 순환적 개념을 처리할 수 있는 알고리즘이 필요하다. 본 연구에서는 이러한 문제점을 해결하고 순환적 질의를 검색하기 위해 논리기법을 이용한 연역 데이터베이스의 릴레이션을 설계하고, 아울러 기존의 관계형 DBMS 릴레이션과의 비교분석을 통해 연역 데이터베이스 릴레이션의 효율적인 측면을 고찰한다. 그리고 SQL에서 순환적 질의를 처리할 수 있는 기능을 응용 프로그램내에 설계한다.

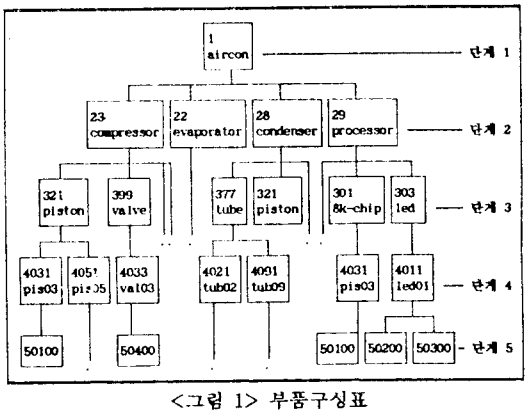
2. 연역 데이터베이스와 자재관리 시스템

2.1. 연역 데이터베이스 시스템

관계형 DBMS로부터 발전되어 나온 논리기반(logic-based) 데이터베이스 시스템인 연역 데이터베이스 시스템은 기존 관계형 DBMS의 데이터 조작기능과 논리 시스템의 추론기능의 결합으로, 추론을 통해 데이터베이스안에 명백하게 저장되어 있지 않은 데이터를 생성한다[7]. 즉, 연역 추론기법을 이용하여 이미 존재하는 데이터로부터 새로운 데이터 즉, 명시적(explicit)으로 저장되어 있지 않은 데이터를 검색한다. 연역 데이터베이스는 데이터를 릴레이션 뿐만 아니라 일반 규칙의 형태로도 저장 가능하게 하는 것으로, 특정 데이터들의 관계를 규칙으로 표현할 수 있는 경우에는 불필요하게 기억장소를 차지하면서 그 관계들을 일일이 저장할 필요없이, 필요에 따라서 추론에 의하여 데이터를 검색할 수 있어 기존의 관계형 DBMS보다 검색능력이 더 강력하다.

2.2. 자재관리 시스템

자재관리는 여러 겹으로 얽혀진 생산 조직에 대해 언제, 어떠한 행동을 취해야 하는가를 정확하게 제시하고, 그것에 따라 원자재에서부터 완제품에 이르기까지 자재의 흐름을 효율적이고 합리적으로 관리하는 방법이다. 조직내에서 자재의 전체적 흐름에 관한 시스템은 내부에서 이를 파악할 때에는 완제품에 대한 하위 시스템으로 형성되어 있으며, 그 하위 시스템은 조직의 기능이나 규모에 따라 그 구조를 각각 달리하고 있으나, 일반적으로 원자재, 재공품, 완제품 등 3개의 관리부분으로 구분하고 있다. 오늘날 자재관리는 회사 전체적인 차원에서 관리하기도 하고, 각 부분별로 구분하여 각각의 주관부서에서 관리하기도 한다.



<그림 1> 부품구성표

본 연구의 데이터베이스에서 사용한 자재들은 그 자재들의 소요 수준 즉, 완제품에 쓰이는가 또는 부품에 쓰이는가에 따라 다섯단계로 구분했다. <그림 1>은 본 연구에서 사용한 부품구성을 단순화시켜 나타낸 것으로, 단계 1은 최종 완제품이며, 단계 2는 단계 1에 필요한 부품이고, 단계 5는 단계 4에 필요한 부품들로서 단계 5의 부품은 본 연

구에서 사용한 데이터베이스의 최종 부품이다.

부품구성표에 표시된 숫자는 완제품 또는 각 부품의 번호이며, 완제품 또는 부품의 이름을 번호 바로 밑에 표기했다. 이 부품구성표를 관계형 DBMS의 릴레이션으로 표현한 것이 <표 1>로써, 관계형 DBMS 릴레이션은 product, part1_co, part2_co, part3_co 그리고 part4_co 등 다섯개이며, 각 릴레이션의 애트리뷰트(attribute)간의 관계는 다음과 같다. 릴레이션 product와 part1_co는 제품번호 1인 완제품 aircon을 생산하는데 compressor, evaporator, condenser, processor 등 네개의 부품이 필요하고 각각의 부품번호는 23, 22, 28, 29이다. 또한, 이 두 릴레이션은 부품 compressor에는 하위 부품으로 부품번호 321, 397, 399, 346의 부품들이 소요된다는 내용도 명시적으로 포함하고 있다. 릴레이션 part2_co, part3_co, part4_co도 이와 동일한 형식으로 데이터 상호간의 관계를 해석할 수 있다.

product

product#	pro_name	part1#
1	aircon	23
1	aircon	22
1	aircon	28
1	aircon	29

part1_co

part1#	pt1_name	part2#
23	compressor	321
23	compressor	397
23	compressor	399
23	compressor	346
22	evaporator	346
22	evaporator	377
22	evaporator	398
28	condenser	377
28	condenser	321
28	condenser	355
28	condenser	398
29	processor	376
29	processor	301
29	processor	303

part2_co

part2#	pt2_name	part3#
321	piston	4031
321	piston	4051
346	spring	-
399	valve	4033
398	sprocket	-
377	tube	4021
377	tube	4091
355	cam	4061
397	motor	4081
376	cpu1	4041
301	8k-chi	4031
303	led	4011

part3_co

part3#	pt3_name	part4#
4061	cam06	-
4031	pis03	50100
4051	pis05	-
4081	mot08	-
4021	tub02	-
4091	tub09	-
4011	led01	50200
4011	led01	50300
4033	val03	50400
4041	cpu04	50500

part4_co

part4#	pt4_name
50100	lev0300
50200	led0100
50300	led0101
50400	val0300
50500	cpu0400

<표 1> 자재관리의 관계형 DBMS 릴레이션

일반적으로, 기업의 부품구성표는 대부분 넷 또는 다섯단계 이상이고, 산업이 점차 발전함에 따라 생산공정의 성격도 크게 전문화되어 보다 작은 종류의 제품을 생산하게 됨으로써 다른 기업의 완제품을 자재로 사용하는 기업이 점차 늘어나고 있으며, 심지어 전혀 제품은 생산하지 않고 오직 타사의 완제품만을 조립하는 기업까지 생겨났다. 따라서, 기존의 관계형 DBMS 릴레이션으로 데이터베이스를 설계한 경우에는, 부품구성이 바뀌고 부품구성 단계가 추가될 때 새로운 릴레이션을 추가하고, 그에 따른 새로운 SQL 질의 표현을 사용해야만 사용자가 원하는 정보를 얻을 수 있다. 그리고 SQL 질의 표현 자체도 부품구성이 네단계 이상으로 구성되면 매우 복잡하게 되고, 처리는 아주 비효율적으로 된다. 이 비효율성을 극복하기 위하여, 본 연구의 수행 방향은 크게 두 가지로 나누어진다. 첫째는 <표 1>의 전통적 릴레이션 설계에서 탈피하여 자재관리에 적합한 새로운 릴레이션을 설계하는 것이고, 둘째는 새롭게 설계된 릴레이션을 사용하여 간결한 SQL 질의로 원하는 정보를 얻을 수 있도록 순환적 질의를 설계하는 것이다.

3. 자재관리에의 SQL질의

3.1. SQL이 단독으로 사용될 경우의 문제점

2.2의 <표 1>에 기술한 릴레이션을 기반으로 한 '릴레이션 part2_co에서 part3#가 4031인 part2#와 part2의 이름을 찾아라'와 '릴레이션 part1_co에서 part2#가 330보다 작은 part1#와 part1의 이름을 찾아라'는 SQL 표현식을 통해 데이터베이스에 명시적으로 저장되어 있는 데이터를 검색할 수 있다.

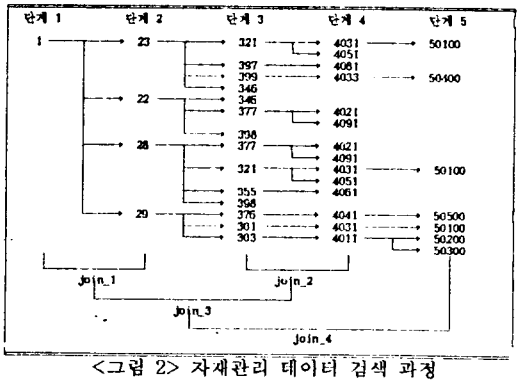
질의:
```
SELECT part2#, pt2_name
FROM   part2_co
WHERE  part3# = 4031
```

결과: part2#	pt2_name
321	piston
301	8k-chip

```
SELECT part1#, pt1_name
FROM   part1_co
WHERE  part2# < 330
```

part1#	pt1_name
23	compressor
28	condenser
29	processor

<표 1>의 관계형 DBMS 릴레이션에는 완제품 aircon의 최종 부품이 어떤 것인지에 관한 명시된 데이터가 포함되어 있지 않다. 또한, 부품 compressor에 어떤 최종 부품이 필요한지에 관한 명시된 데이터도 없다. 즉, 완제품 aircon의 최종 부품을 찾으려면 <그림 2>와 같이 부품번호 1 → 23 → 321 → 4031 → 50100 의 순으로 검색할 뿐만 아니라 1 → 23 → 321 → 4051 등의 순으로 검색하는 일련의 과정을 거쳐야만 원하는 결과를 얻을 수 있다. 현재, 관계형 DBMS에서 명시적으로 지장되어있지 않은 데이터를 처리하는 방법은 조인(join) 오퍼레이션을 이용하는 것으로, 본 절에서는 현재 관계형 DBMS에서 사용하고 있는 이 방법을 질의를 통해 어떤 문제점이 있는지 고찰한다.



<그림 2> 자재관리 데이터 검색 과정

질의 '완제품 aircon의 최종 부품들을 찾아라'를 처리하기 위해서는, 먼저 조인 오퍼레이션을 이용해 필요한 데이터들이 포함된 릴레이션들을 결합해 새로운 제3의 릴레이션을 만들고 그 새로운 릴레이션안에 데이터들을 명시적으로 저장하여 질의를 처리해야 한다. 그러나 조인은 한번에 단지 시로 다른 두개의 릴레이션만을 결합하기 때문에 관계형 DBMS 릴레이션에서 제품에 소요되는 최종 단계들 또는 하위 단계들의 부품을 알고자 할 때에는 릴레이션이 다섯개이므로 조인을 이용한 네번에 걸친 결합 과정이 필요하다. 즉, <그림 2>의 하단부분에 나타나 있듯이 릴레이션 product와 part1_co를 이용해 릴레이션 join_1을 만들고, part2_co와 part3_co를 이용해 릴레이션 join_2를 만드는 일련의 네번의 과정을 거쳐 릴레이션 다섯개를 모두 결합하여 새로이 결합된 릴레이션을 생성하여 join_4라고 하면, 질의 '새로이 생성된 릴레이션 join_4에

서 완제품 이름이 aircon인 최종 단계의 부품이름을 검색하라'의 결과는 다음과 같다.

```
질의 : SELECT  pro_name, pt4_name
       FROM    join_4
       WHERE   pro_name = aircon

결과 :
pro_name    pt4_name
aircon      lev0300
aircon      led0100
aircon      led0101
aircon      val0300
aircon      cpu0100
```

이 질의는 릴레이션 join_4를 사용자가 원하는 데로 성공적으로 형성된 경우에 한하여 옳은 정보를 주게된다. 그러나 결과에서 알 수 있듯이 결과가 단계 5의 부품만 검색되었다. 결국 이것은 완제품 aircon에 소요되는 최종 단계의 부품 전부가 아님을 알 수 있다. 왜냐하면 <그림 2>에서 완제품 aircon에 소요되는 최종 부품으로는 단계 5의 부품만 있는 것이 아니라, 부품번호 346, 398 등 단계 3과 부품번호 4021, 4081, 4051 등 단계 4에도 최종 부품이 있기 때문이다. 결국 관계형 DBMS를 기반으로 한 SQL은 사용자가 원하는 옳바른 정보를 제공해 주지 못하였다. 그 이유는 관계형 DBMS에서 지원해 주는 조인 오퍼레이션이 단지 릴레이션간의 결합을 행할 뿐이지 각 릴레이션 상호간의 논리적인 관계를 연결해 주지 못하고 있기 때문이다.

3.2. 내장된 SQL을 사용할 경우의 문제점

3.1.의 조인 오퍼레이션을 사용할 경우, 원하는 질의를 검색할 수 없는 문제점을 해결하는 방법으로 프로그램을 작성할 수 있다. 이 프로그램은 어떤 완제품 또는 부품의 최종 부품을 검색하는데 있어, 2.2절 <표 1>의 관계형 DBMS 릴레이션을 기반으로 작성한 내장된 SQL에서의 프로그램이다. 내장된 SQL은 SQL과 다른 프로그래밍 언어로 작성된 응용 프로그램과의 인터페이스를 위한 것으로, 본 연구에서는 프로그래밍 언어 파스칼(Pascal)을 사용해 개념적인 시스템을 설계했으며, 이 프로그램을 사용함으로서 조인 오퍼레이션 질의 검색의 문제점을 해결할 수 있다.

이 프로그램을 사용하면 3.1에서 기술한 SQL 상에서의 문제점을 해결할 수 있다. 그러나 이 프로그램은 프로그램의 라인(line)이 길어 프로그램을 해석하기가 쉽지 않을 뿐만 아니라, 만약 데이터베이스에 새로운 릴레이션이 추가된다면 프로그램을 다시 수정해야 하고, 추가된 만큼 프로그램이 길어지는 프로그램상에서 유연성(flexibility)이 없는 문제점이 있다. 그리고 이 프로그램이 적용된 질의 검색은 앞서 표기한 <그림 2>에서 데이터간의 관계가 동일한 성질을 가지고 있는 형태의 검색이다. 즉, 각 단계의 이는 완제품 또는 부품과 그에 속하는 부품간의 관계를 찾는 것이다. 따라서, 이러한 질의를 좀 더 효율적으로 처리하기 위해 순환적 질의를 적용하였고, 그 순환적 질의에 맞는 새로운 데이터베이스를 설계하였다.

3.2. 순환(recursion)

순환은 논리식의 머리 프레디키트(head predicate)와 몸체 프레디키트(body predicate)에 동일한 관계 기호가 나타나는 것을 의미하며, 자신이 자신을 되부름 하는 것이다[1][5]. 논리식의 순환규칙에 대한 SQL 질의는 관계형 DBMS에 반복 연산 기능이 없기 때문에 응용 프로그램을 이용해서 해결책을 얻을 수 있는 형태로의 전환이 불가피하다. 현재, 논리체계에서 실용적인 대부분의 순환적 질의는 선형적이며, 가장 기본적인 논리식의 형태가 a :- b , a. 이다. 인수(argument)가 두개인 순환규칙에 대한 형태로 need(Buyer,Product)의 기본 관계로부터 유도되는 buy(Buyer,Product) 관계의 선형적 순환을 표현할 수 있다[6]. 이를 'X가 Y를 필요로 하면 X는 Y를 구입한다'를 순환의 개념을 이용해 나타내면 다음의 논리식으로 표현된다.

```
buy(X,Y) :- need(X,Y).
buy(X,Y) :- need(X,Z), buy(Z,Y).
```

4. 자재관리 시스템에의 순환적 질의의 설계

4.1. 연역 데이터베이스의 릴레이션의 설계

순환이 가능하기 위해서는 부품과 그 부품에 쓰이는 다른 부품 또는 완제품간의 관계가 부품 단계와 무관하게 동일한 관계로 표현되어야 한다. 본 연구에서 설계해 자재관리에서의 순환적 표현에 사용한 연역 데이터베이스의 릴레이션은 <표 2>이며, 두 개의 릴레이션 p_name, p_use로 구성되어 있다. <표 2>에 표시된 릴레이션은 2.2절 <표 1>의 관계형 DBMS 릴레이션과 동일한 의미의 데이터이며, <표 1>의 관계형 DBMS 릴레이션 다섯 개를 연역 데이터베이스 릴레이션 두 개로 표현할 수 있다. 릴레이션 p_name은 완제품 또는 부품번호와 거기에 해당되는 완제품 또는 부품이름을 가지고 있는 릴레이션이며, 릴레이션 p_use는 완제품 또는 부품번호들로만 구성된 연역적 데이터를 가지고 있는 릴레이션이다.

p_name

pp_num	pp_name
1	aircon
22	evaporator
23	compressor
28	condenser
29	processor
321	piston
346	spring
399	valve
398	sprocket
377	tube
355	cam
397	motor
376	cpul
301	8k-chip
303	led
4061	cam06
4031	pis03
4051	pis05
4081	mot08
4021	tub02
4091	tub09
4011	led01
4033	val03
4041	cpu04
50100	lev03
50200	led01
50300	led01
50400	val1030
50500	cpu0400

p_use

pp_num	part_num
1	23
1	22
1	28
1	29
23	321
23	397
23	399
23	346
22	346
22	377
22	398
28	377
28	321
28	355
28	398
29	376
29	301
29	303
321	4031
321	4051
399	4033
377	4021
377	4091
355	4061
397	4081
376	4041
301	4031
303	4011
4031	50100
4011	50200
4011	50300
4033	50400
4041	50500

<표 2> 자재관리의 연역 데이터베이스 릴레이션

이 방법으로 새롭게 설계된 릴레이션에서는 완제품 또는 부품을 명시적으로 나타낼 필요가 없다. 왜냐하면, 어떤 질의 검색과정에서 pp_num에 있는 완제품 또는 부품이 part_num에 나타나지 않으면 그것은 완제품이며, 또 part_num에 있는 부품이 pp_num에 나타나지 않으면 그것은 최종 부품이라는 것을 알 수 있기 때문이다. 따라서, 본 연구에서는 앞으로 완제품 또는 부품을 서로 구별하지 않고 모두 부품으로 부르기로 한다.

4.2. 연역DB 릴레이션에서의 순환적 질의의 설계

2.2의 <표 1>에 표기된 관계형 DBMS 릴레이션만을 이용해서는 순환적 개념을 적용시킬 수가 없고, 그 릴레이션을 기반으로 프로그래밍 언어를 이용하여 응용 프로그램을 작성한다고 할지라도 프로그램안에 순환적 개념을 표현할 수는 없었다. 따

라서, 본 절에서는 4.1에서 설계한 연역 데이터베이스 릴레이션을 사용해서 순환적 개념을 적용해 결과를 도출하고자, 내장된 SQL 프로그램을 이용해 순환적 질의를 설계했다. 본 연구에서 설계한 프로그램은 연역 데이터베이스 릴레이션에서 순환적 개념을 이용해 부품에 소요되는 최종 부품의 리스트(list)를 출력하는 프로그램으로, 질의 '부품번호 1인 aircon에 소요되는 최종 부품을 모두 찾아라'는 3.1절의 <그림 2>의 원리를 이용해서 그 결과를 얻을 수 있다.

결과 : 346 398 4021 4051 4061 4081
4091 50100 50200 50300 50400 50500

이와 같은 결과는 프로그램내의 순환 프로시저가 다른 매개변수 값을 가지고 반복해서 수행되는 것이 가능하기 때문에 나타났다. 즉, 모든 변수값이 순환 프로시저에 의해 다시 생성되며, 그 순환이 가능한 이유는 부품간의 종속 관계가 하나의 릴레이션으로 나타나기 때문이다. <표 2>의 연역 데이터베이스 릴레이션을 기반으로 한 순환적 질의의 내장된 SQL 프로그램을 관계형 DBMS 릴레이션을 이용한 프로그램과 비교하면 프로그램의 라인(line)이 현저하게 줄어들었다. 그 이유는 순환적 질의 검색 프로그램에는 순서 제어에 관한 내용이 삽입되지 않아도 되기 때문이며, 따라서 프로그램의 의미를 보다 명확하게 반영한다고 할 수 있다. 또한, 이 프로그램은 관계형 DBMS 릴레이션을 사용한 프로그램과는 달리, 부품구성의 단계가 늘어나도 릴레이션이나 프로그램의 수정없이 기존의 p_use 릴레이션에 새로운 튜플만을 추가하여 그대로 사용할 수 있으며, 정형화된 형태(form)이기 때문에 예로 든 질의 이외에 다음과 같은 형태의 질의도 프로그램의 수정없이 가능하다. 질의 '부품 5051이 쓰이는 최종 부품(즉, 완제품)을 검색하라'는 연역 데이터베이스 릴레이션을 기반으로 한 순환적 질의의 내장된 SQL 프로그램의 순환 프로시저내에서 SQL 실행문 중 SELECT문을 약간 수정함으로써 결과를 검색할 수 있다. 즉, 이 질의를 검색하기 위해서는 다음과 같이 WHERE문에서 기존의 애트리뷰트 pp_num을 part_num으로, 애트리뷰트 이름을 변경해 주면 검색이 가능하게 된다.

```
EXEC SQL SELECT   pp_num, part_num
         FROM     p_use
         WHERE    part_num = num;
```

따라서, 본 연구에서는 어떤 부품에 소요되는 최종 부품만을 검색하는 질의 검색을 설계하였는데, 위에 기술한 바와 같이 이 질의 뿐만 아니라 프로그램에서 SELECT문의 수정을 통하여 특정 부품이 어떤 상위 부품에 필요한지 검색할 수 있는 등 다양한 종류의 순환적 개념이 포함된 질의들을 검색할 수 있음을 알 수 있다

5. 결 론

지금까지 자재관리 시스템의 데이터를 사용해서 관계형 DBMS 질의어인 SQL의 문제점을 예를 통해서 제시했고, 그 문제점들의 해결을 위한 순환적 질의의 표현을 중심으로 기술하였다. 그리고 관계형 DBMS와 연역 데이터베이스를 비교 기술하면서 연역 데이터베이스 시스템에서 순환적 질의를 가능하게 하는 자재관리 릴레이션을 설계하였으며, 파스칼(Pascal)을 사용해 내장된 SQL에서 시스템을 설계하였다. 결론적으로, 본 연구에서 설계한 연역 데이터베이스의 릴레이션 및 순환적 질의의 이점은 다음과 같다.

첫째, 관계형 DBMS 릴레이션을 이용하여 처리하는 프로그램에 비해 본 연구의 연역 데이터베이스 릴레이션을 사용하는 프로그램은 순서 제어에 관한 내용이 삽입되지 않아도 되기 때문에, 프로그램의 라인이 줄어들게 됨으로써 프로그램을 이해하기가 쉬워졌고, 따라서 프로그램의 의미를 보다 명확하게 반영한다. 둘째, 하위의 단계의 부품구성 단계가 추가되었을 때, 관계형 DBMS에서는 그 릴레이션을 새롭게 작성하여야 하지만, 본 연구에서 설계한 자재관리 연역 데이터베이스 릴레이션에서는 새로운 데이터 릴레이션을 만들 필요없이 기존의 데이터베이스 릴레이션에 새로운 튜플만을 추가하여 사용할 수 있다. 세째, 본 연구에서 작성한 프로그램은 매우 정형화된 형태이기 때문에 다른 자재관리 시스템에 적용한 경우에도 프로그램의 변경없이 적용하고자 하는 자재관리 시스템에 맞는 데이터와 그 애트리뷰트만을 수정하여 사용할 수 있는 프로그램의 유연성(flexibility)이 있다. 네째, SQL 표현식의 형태로 나타나는 다른 질의의 검색에도 프로그램 전부를 수정하는 것이 아니라, 프로그램에서 SQL 표현식만을 변경하여 사용할 수 있으므로, 본 연구의 연역 데이터베이스 릴레이션을 기반으로 하는 프로그램은 다른 질의의 검색에 쉽게 응용될 수 있는 이점이 있다.

그러나, 본 연구는 릴레이션의 애트리뷰트를 부품과 하위 부품으로 구성하여 그 관계만을 이용해 질의를 검색하였기 때문에, 실제 업무에서 사용하는 자재관리 데이터로서는 부족하다고 할 수 있다. 실제 자재관리 시스템에는 부품 소요량이 매우 중요한 정보이다. 따라서, 부품과 하위 부품의 관계뿐만 아니라 부품 소요량, 공급자 등 데이터의 범위를 확장하여 순환적 질의를 검색하면 더 효율적일 것이다.

참고 문헌

[1] Chakravarthy, U., J. Grant, and J. Minker, "Logic-Based Approach to Semantic Query Optimization," *ACM Transactions on Database Systems*, Vol.15, No.2, June, 1990, pp.162-207.

[2] Chase, R. B., and N. J. Aquilano, *Production and Operations Management : A Life Cycle Approach*, 5th ed., Richard D. Irwin Inc., 1989.

[3] Emerson, S. L., M. Darnovsky, and J. S. Bowman, *The Practical SQL Handbook Using Structured Query Language*, Addison-Wesley Publishing Company, 1989.

[4] Fratarcangeli, C., "Technique for Universal Quantification in SQL," *SIGMOD RECORD*, Vol. 20, No. 3, Sep., 1991, pp.16-24.

[5] Gallaire, H., J. Minker, and J. Nicolas, "Logic and databases : A Deductive Approach," *ACM Computing Surveys*, Vol.16, No.2, 1984, pp.153-185.

[6] Naughton, J. F., "Minimizing Function-Free Recursive Inference Rules," *Journal of the ACM*, Vol.36, No.1, Jan., 1989, pp.69-91.

[7] Parsaye, K., M. Chignell, S. Khoshafian, and H. Wong, *Intelligent Database*, John Wiley & Sons Inc., 1989.

[8] Saraiya, Y. P., "Hard Problems for Simple Logic Programs," *ACM SIGMOD*, 1990, pp.64-73.

[9] Sarda, N. L., "Extensions to SQL for Historical Database," *IEEE Transactions on Knowledge and Data Engineering*, Vol.2, No.2, June, 1990, pp.220-230.

[10] Uckan, Y., "Integrating Logic Programming into a Data Base Course : Views as Rules in Deductive Relational Data Bases," *ACM SIGMOD*, 1991, pp.184-191.

[11] Ullman, J. D., and C. Zaniolo, "Deductive Databases : Achievements and Future Directions," *SIGMOD RECORD*, Vol. 19, No. 4, Dec., 1990, pp.75-82.

[12] Youn, C., H. Kim, L. Henschen, "Classification and Compilation of Linear Recursive Queries in Deductive Databases," *IEEE Transactions on Knowledge and Data Engineering*, Vol.4, No.1, Feb., 1992, pp.52-67.