論文94-31B-9-2

부질의 기능을 추가한 확장된 Query-by-Example

(The Extended Query-by-Example Supporting Subqueries)

元 希 先*, 李 鍾 鄗**, 黃 奎 永**

(Hee Sun Won, Jong Hak Lee and Kyu Young Whang)

要 約

Query-by-Example(QBE)은 관계형 데이타베이스를 검색, 변경, 정의 및 제어하기 위해 편리하고 일관된 그래픽 형태의 사용자 인터페이스를 제공하는 전시 지향적 고급 데이타베이스 처리 언어로서, 관계적으로 완전한 언어이다. 그러나, QBE에서는 부질의(subquery)의 기능이 없어 QBE의 유용성을 크게 떨어 뜨린다. 특히, 어떤 질의는 하나의 윈도우로 표현되지 않는다. 본 논문에서는 QBE에 부질의 기능을 제공하기 위하여 부질의 상자(subquery box)를 정의하고, 부질의의 직접적 표현이 가능하도록 QBE를 확장한다. 따라서, 기존의 QBE에서는 하나의 윈도우로 표현할 수 없었던 질의를 표현가능하게 함으로써 질의의 작성에 드는 오버헤드 문제를 해결한다. 또한, 확장된 QBE를 위한 문법을 정의하여 이를 파싱하는 방법에 대해 기술하고, 확장된 QBE질의를 SQL문으로 변환하는 기법을 제안한다. 변환된 SQL문은 대부분의 SQL 시스템이 제공하는 동적 SQL 프로그램을 통하여 수행시킬 수 있다. 본 논문에서 제안한 확장된 QBE는 IBM의 OS/2상에서 OS/2 EE 데이타 베이스 관리 시스템을 이용하여 구현되었다.

Abstract

Query-by-Example(QBE) is a high-level display-oriented database manipulation language that provides a convenient and unified style for querying, updating, defining, and controling a relational database. QBE is relationally complete. However, lack of subquery constructs limits the usability of QBE significantly. In particular, certain queries cannot be represented in one window. In this paper, we define a subquery box and extend QBE for subquery construction. The Extended QBE makes it possible to represent the queries that the QBE cannot do in one window, reducing the overhead and complexity of composing those queries. We also define the grammar of the Extended QBE and present the parsing techniques. Finally, we present an algorithm to transform the queries in Extended QBE to those in SQL. The result of the transformation can be executed using dynamic SQL features of any SQL system. The proposed language has been implemented on OS/2 using the OS/2 EE Database Manager.

*正會員, 韓國放送公社 技術研究所
(Technical Research Institute, KBS)
**正會員, 韓國科學技術院 電算學科
(Dept. of Computer Science, KAIST)
接受日字 : 1994年 1月 4日

Ⅰ. 서 론

응용 소프트웨어의 종류가 다양해지고 컴퓨터 사용자의 범위가 확대됨에 따라 컴퓨터를 잘 알지 못하는 일반 사용자들에게 소프트웨어를 보다 쉽게 사용할 수 있도록 하는 사용자 친숙성(user friendliness)이 중요시 되고 있다. 최근 사용자 친숙성을 증가시키기 위한 다양한 윈도우 시스템과 그것을 이용한 사용자 인터페이스의 개발이 활발히 진행되고 있으며, 편리하고 사용하기 쉬운 인터페이스의 제공 여부는 소프트웨어의 유용성을 판가름하는 하나의 기준이 되고 있다. 이러한 추세에 따라 데이타베이스 관리 시스템에서도 일반 사용자들이 쉽게 데이타베이스를 다룰 수 있는 사용자 인터페이스를 제공하고자 많은 연구가 진행 중이다 .

현재 많은 데이타베이스 관리 시스템에서 제공하고 있는 데이타베이스 처리 언어(high -level database manipulation language)는 SQL [2] 과 QUEL [13] 등이 속하는 영어적 질의어(English-like query language)와 Query-by-Example(QBE) [18], EQBE [11], FORAL-LP [12], CUPID [10], GUIDE [16] [17] [4], PICASS [8] 등과 같이 그래픽 객체(graphical object) 들을 사용한 전시 지향적 데이타 처리 언어(display-oriented data manipulation language) 로 구별될 수 있다. 영어적 질의어는 언어의 구문 구조를 확실히 알아야 하므로 숙련된 사용자가 되기 위하여 비교적 많은 시간을 필요로 하는 반면에, 전시 지향적 데이타 처리언어는 사용자가 보다 쉽게 시스템을 다룰 수 있도록 해 준다.

Query-by-Example (QBE)은 관계형 데이타 모델을 근간으로 하는 고급 데이타베이스 처리 언어로서, 데이타베이스를 검색, 변경, 정의 및 제어하는데 편리하고 통일된 표현법을 제공하여 문법이 간단하므로 사용자가 쉽게 배워 사용할 수 있다. [18] QBE 사용자는 한 릴레이션과 대응되는 테이블 구조 (table skeleton)를 기반으로 질의를 작성하게 된다. 질의의 형식이 테이블 구조에 의해 이미 정의되어 있으므로 영어적 질의어와 같이 문(phrase)의 구조 자체가 잘못되는 경우는 없다. 또한 질의를 작성하는데 있어서도 특별한 순서가 없이 융통성을 제공하므로 사용자는 사고 과정을 따라서 테이블을 채워나갈 수 있으며, 점차 새로운 조건들을 추가해 나가는 방식으로 간단한 질의에서부터 복잡한 질의까지 자연스럽게 만들어 나갈 수 있다. 여러가지 심리학적 연구 결과에 따르면 프로그래머가 아닌 일반 사용자의 경우에도 세시간 이하의 강의만 들으면 QBE를 이용하여 꽤 복잡한 질의를 할 수 있다고 한다. [14]

그러나, QBE에는 한 질의문 내에서 질의의 대상을 또 다른 질의문으로 표현할 수 있는 부질의(subquery)의 기능이 없어, 부질의 구문을 지원하는 다른 질의어들과 비교하여 한 화면에서의 질의의 표현범위가 좁다. SQL에서는 완전한 SELECT문인 부질의를 정의할 수 있고 사용자는 그 결과를 이용하여 더욱 복잡한 조건들을 표현 할 수 있는 반면에, QBE에서는 부질의 기능의 결여로 인하여 부질의에 해당되는 질의를 수행시켜서 결과를 임시 테이블에 저장하고 그 테이블을 이용하여 다음 질의를 만들어야 하는 경우가 발생한다. 따라서, 질의의 절차가 길어지며 직관적인 사고가 감소되어 질의를 작성하기에 불편하고 복잡해진다. 특히, 두개 이상의 테이블이 참여하는 부정 부질의(subqueries with negated existential quantification)가 포함된 질의문을 작성하는 경우에는 이러한 현상이 반드시 발생한다. 이는 다른 대부분의 부질의(SQL에서 IN, EXISTS, ALL, ANY, SOME등의 키워드와 함께 사용된 부질의의 경우)가 포함된 질의문은 부질의가 없는 질의문으로 변환이 가능하지만, 부정 부질의가 포함된 경우에는 그렇지 못하기 때문이다. [7] 지금까지 QBE에서 부질의 기능을 제공하기 위한 연구의 예는 찾아볼 수가 없다.

본 논문에서는 위와 같은 QBE의 문제점을 해결하기 위하여 부질의 상자(subquery box)를 정의함으로써 부질의의 직접적 표현이 가능하도록 QBE를 확장하고, 이를 이용하여 우선적으로 특히 문제가 되는 부정 부질의가 가능한 확장된 QBE를 윈도우 환경하에서 구현한다. 윈도우 환경하에서는 메뉴와 다양한 윈도우 객체의 효율적인 사용으로 사용자가 보다 빠르고 쉽게 시스템에 적응할 수 있도록 해 준다. 또한 여러개의 윈도우를 동시에 사용할 수 있어서 사용자는 수행시킨 질의와 결과를 하나의 화면에서 볼 수 있다. 확장된 QBE의 질의처리를 위해서는 입력 윈도우상에서 작성된 질의의 구문분석을 통하여 요약 구문 트리(abstract syntax tree)를 만들고 이를 SQL문으로 변환한다. 이 결과는 대부분의 SQL 시스템이 제공하는 동적 SQL 프로그램 (dynamic SQL program) 을 통하여 수행시킬 수 있다.

본 논문의 구성은 다음과 같다. 제 Ⅱ 장에서는 QBE의 기본적인 개념 및 연산에 대해 소개한다. 제 Ⅲ 장에서 기존의 부질의 기능을 제공하지 않는 QBE의 문제점을 지적하고, 부질의 기능을 가지는 QBE의 확장에 대해 논한다. 제 Ⅳ 장에서는 확장된 QBE의 문법을 정의하고 구문분석을 위해 사용된 알고리

즘을 기술하며, 제 Ⅴ장에서 구문분석의 결과를 SQL 문으로 변환하는 알고리즘에 대해 논한다. 마지막으로, 제 Ⅵ장에서는 본 논문의 결론을 맺는다.

Ⅱ. Query-By-Example

본 장에서는 Query-by-Example(QBE)에 대하여 간단히 소개한다. 먼저 제 1절에서는 QBE의 구성요소인 테이블의 구조와 연산자들에 대하여 소개하고, 제 2절에서는 검색질의의 구문을 소개한다. 검색질의 외의 삽입, 삭제 및 수정 등을 위한 질의는 검색질의에 사용되는 출력 연산자 P. 대신 삽입 연산자인 I. 삭제 연산자인 D. 그리고 수정 연산자인 U. 등을 사용하는 것을 제외하고는 검색과 같은 방식으로 질의할 수 있다. [18]

1. QBE의 구성요소

SQL이나 QUEL 등과 뚜렷이 구별되는 QBE의 특징은 이차원 객체들을 이용하여 질의를 작성한다는 것이다. 즉 테이블 구조나 조건 상자(condition box)안에 원하는 결과의 예들과 조건식들을 채워 넣음으로써 질의가 작성된다. [6] [15] [18] 테이블 구조는 그림 1과 같이 테이블 이름 영역(table name field), 열 이름 영역(column name field), 행 연산자 영역(row operator field)과 엔트리 영역(entry field)등 네개의 영역으로 나뉘어 진다 .

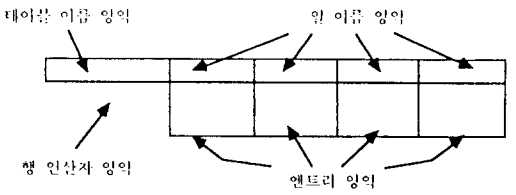


그림 1. 테이블 구조
Fig. 1. Table skeleton.

테이블 구조의 모든 영역에서 출력(print), 삽입(insert), 삭제(delete) 및 수정(update) 등을 위한 연산자들이 사용될 수 있으며, 사용된 위치에 따라 연산의 의미와 대상이 달라진다. 테이블 이름 영역에서 사용된 연산자는 테이블 이름이나 전체 열 이름에 대한 연산을 수행하고, 열 이름 영역에서 사용되면 그 열 이름에 대해서만 연산을 수행한다. 행 연산자 영역에서 쓰이면 테이블의 전체 행에 대하여 연산을 수행하고, 엔트리 영역에 쓰인 경우에는 해당되는 열에 대해 연산을 수행한다. [18] 이와같이 같은 연산자들이 테이블 구조의 모든 영역에서 사용될 수 있도록 하며, 사용영역에 따라 다른 의미를 부여함으로써 사용자에게 데이타베이스를 다루는 일관된 방식을 제공하게 된다. 테이블 구조와는 별도로 조건 상자(condition box)가 제공되는데, 이것은 논리식 등 테이블 구조내에서 표현하기 어려운 식들을 표현할 수 있는 기능을 제공한다.

QBE에서 사용되는 연산자는 데이타에 관한 조건들과 처리과정을 정의하기 위하여 사용되며, 유형에 따른 연산자들과 그 의미는 그림 2와 같다. [5]

유 형	연산자	의 미
시스템 연산자	P.	출 력
	U.	수 정
	I.	삽 가
	D.	삭 제
	G.	묶어서 처리 (group by)
	ALL.	중복 데이타도 처리
	UNQ.	중복 데이타는 처리 안함
	AO(n).	올림순으로 나열
	DO(n).	내림순으로 나열
집단 연산자	SUM.	합 계
	CNT.	총 개
	AVG.	평균 값
	MAX.	최대 값
	MIN.	최소 값
비교 연산자	=	같음
	!	부정
	!=	같지 않음
	>	크다
	>= 또는 =>	크거나 같다
	<	작다
	<= 또는 =<	작거나 같다
산술 연산자	+	더하기
	-	빼기
	*	곱하기
	/	나누기
	**	승
논리 연산자	& 또는 AND	논리곱
	| 또는 OR	논리합

그림 2. QBE에서 사용되는 연산자
Fig. 2. Operators used in QBE.

2. 검색 질의의 구문

검색 질의란 출력 연산자인 P.를 사용하여 데이타베이스 테이블들의 내용을 출력시키는 질의로서 하나 이상의 결과 테이블을 출력시킬 수 있다. 그림 3은 테이블 구조내에서 검색 질의를 위한 일반적인 구문을 보여준다. [18]

테이블 이름 영역의 출력 연산자는 입력 윈도우 상에서 테이블의 전체 열 이름들을 자동적으로 출력시키기 위해 사용된다. AO [(n)] .과 DO [(n)] .에서 (n)은 여러개의 열에 대해 올림순 또는 내림순으로 출력하려고 할 경우에 우선 순위를 주기 위해 사용된다. 집단 연산자(aggregation operators)에는 CNT.(총수), SUM.(총계), AVG.(평균), MAX.

(최대값), 그리고 MIN.(최소값) 등이 있으며, 중복되는 값도 모두 처리대상으로 하기 위해 ALL.이라는 연산자가 항상 뒤에 나와야 하며, 중복 데이타를 제외시키기 위해서는 ALL.앞에 UNQ.를 첨가시키면 된다.

조건 상자는 테이블 구조에서 표현하기 어려운 식을 나타내는데 이용하며, 각각 하나의 예 인자, 비교연산자, 그리고 상수 인자로 구성된 단순 조건식(simple conditions), 두개 이상의 단순 조건식이 하나로 결합된 복합 조건식(compound conditions), 그리고 논리식(logocal expressions)이 사용된 조건식과 같은 세가지 형태의 항목들이 사용될 수 있다. [18]

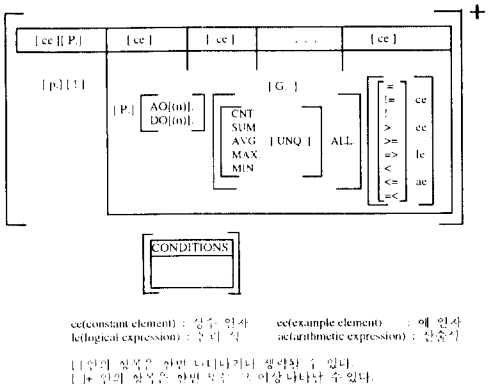


그림 3. 검색 질의의 구문
Fig. 3. Syntax for retrieval Queries.

이와같은 검색 질의의 구문을 이용하여 QBE에서 데이타베이스를 검색하는 질의의 예를 들어 본다. 앞으로 나오는 예들은 다음 테이블들을 기반으로 한다.

- EMP (NAME,SAL,MGR,DEPT, COMMISSION)
- SALES (DEPT, ITEM)
- SUPPLY (ITEM, SUPPLIER)
- TYPE (ITEM, COLOR, SIZE)

EMP 테이블은 고용인의 이름(NAME), 월급(SAL), 매니져(MGR), 일하는 부서(DEPT) 및 커미션(COMMISSION) 등으로 구성되고, SALES 테이블은 부서 이름(DEPT)과 그 부서가 판매하는 품목(ITEM)으로 구성된다. SUPPLY 테이블은 품목(ITEM)과 그 품목을 공급하는 공급자(SUPPLIER)로 구성되며, TYPE 테이블은 품목(ITEM)과 품목의 색상(COLOR) 및 크기(SIZE)로 구성된다.

그림 4는 QBE에서의 검색 질의의 여러 형태를 보이고 있다. 그림 4의 (a)는 색이 RED인 품목을 크기 순서대로 출력하라는 질의이며, (b)는 행 연산자 영역에 P. 연산자를 사용함으로써 (a)와 같은 결과를 출력하는 질의이다. (c)의 질의는 TOY 부서에서 판매하며 색이 GREEN인 품목을 모두 출력하라는 것으로 _NUT 이라는 예 인자는 두개의 테이블을 조인시키는 역할을 한다. (d)는 LEWIS보다 월급이 많은 고용인들의 이름과 월급을 출력하는 질의이며, (e)는 각 고용인에 대해 이름 및 월급과 커미션의 합을 출력하라는 질의로 EMP 테이블에는 월급과 커미션의 합을 위한 열이 없으므로 SUM이라는 열을 가지는 사용자가 정의한 출력 테이블인 OUT을 만들고 EMP 테이블로부터 데이타를 사상시키기 위해서 예 인자를 사용하였다. 마지막으로, (f)는 월급이 $10,000과 $15,000사이에 있고 $13,000는 아닌 고용인들의 이름을 출력하라는 질의로 조건 상자를 이용한 형태이다.

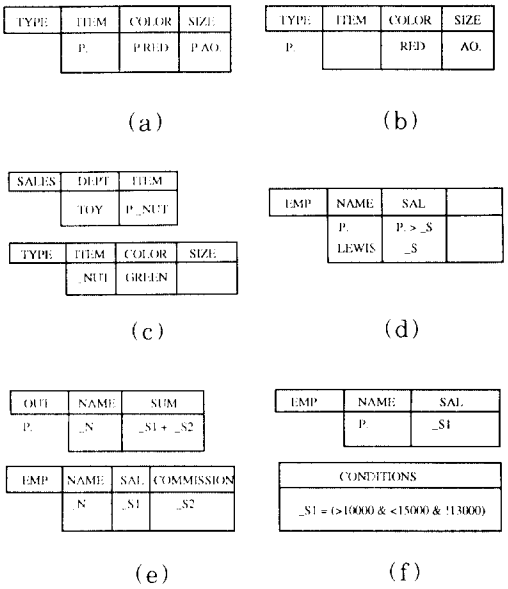


그림 4. QBE에서의 검색 질의의 여러형태
Fig. 4. Examples of retrieval queries in QBE.

Ⅲ. QBE의 확장

QBE에서는 현재 SQL과 같은 언어에서 제공하는 부질의의 기능을 제공하지 않고 있다. 본 장에서는 부질의 기능을 갖도록 QBE를 확장하는 방법에 대하여 논의한다. 먼저, 제 1절에서는 현재 QBE에서 부질의 기능을 제공하지 않음으로써 발생되는 문제점을

지적하고, 제 2절에서는 이 문제점을 해결하기 위한 QBE의 확장을 제안한다.

1. QBE의 문제점

SQL에서는 WHERE절이나 HAVING절의 검색 조건에 부질의를 사용하여 필요한 데이타를 따로 정의할 수 있으므로 질의의 표현 범위가 더욱 넓어진다. SQL의 부질의는 완전한 SELECT문으로서 그 안에 다시 부질의를 포함할 수 있으며 IN, EXISTS, NOT EXISTS, ALL, ANY 또는 SOME 등의 키워드나 비교 연산자와 함께 사용된다.[3] 예를 들어 키워드인 IN과 부질의를 사용하여 제 Ⅱ장 2절에서 정의한 테이블에서 "TOY 부서에서 판매하는 품목들과 색상을 나열하라"는 질의를 SQL로 표현하면 아래와 같다. EXISTS, NOT EXISTS, ALL, ANY 또는 SOME 등의 키워드도 유사한 형태로 부질의와 결합될 수 있다.

```
SELECT  ITEM, COLOR
FROM    TYPE
WHERE   ITEM IN (SELECT ITEM
                 FROM SALES
                 WHERE DEPT = 'TOY')
```

그러나, 부질의가 사용된 SQL문들 중에서 키워드 NOT EXISTS 와 함께 사용되거나 부질의에 집단 연산자(aggregation operators)가 사용된 경우가 아닌 경우에는 부질의가 없는 SQL문으로 변환될 수 있다.[7] 예를들어 위에 있는 키워드 IN과 함께 부질의를 사용한 SQL문은 다음과 같은 부질의가 없는 SQL문으로 표현할 수 있다. 특히, NOT EXISTS가 있으면 키워드 ALL은 키워드 NOT EXISTS와 함께 부질의의 내용을 부정함으로써 대체가 가능하다. 따라서, 키워드 NOT EXISTS와 함께 사용되는 부정 부질의의 기능이 SQL에서의 어떤 부질의의 기능보다 중요하다.

```
SELECT  TYPE.ITEM, TYPE.COLOR
FROM    TYPE, SALES
WHERE   TYPE.ITEM = SALES.ITEM AND
        SALES.DEPT = 'TOY'
```

QBE에는 이와 같은 기능을 가진 부질의를 정의하고 있지 않음으로 인하여 SQL의 키워드 NOT EXISTS와 함께 사용된 부정 부질의가 있는 하나의 SQL문을 QBE에서는 하나의 질의로 표현이 불가능하다. 예를 들어, "HENRY가 공급하는 품목을 한가지도 판매하지 않는 부서에서 일하는 고용인의 이름과 부서를 나열하라"는 다음과 같은 SQL질의를 QBE로 표현해 보자.

```
SELECT  NAME, DEPT
FROM    EMP
WHERE  NOT EXISTS (SELECT *
                   FROM SALES , SUPPLY
                   WHERE EMP.DEPT =
                     SALES.DEPT AND
                   SALES.ITEM =
                     SUPPLY.ITEM AND
                   SUPPLY.SUPPLIER =
                     'HENRY' )
```

QBE로 이와 같은 질의를 하기 위해서는 우선 부질의에 대응되는 질의를 먼저 수행시키고, 그 결과를 연산자 TEMP.을 이용하여 임시 테이블에 저장한 후 그 테이블을 이용하여 또 다른 질의를 만들어야 한다. 그림 5에서는 위의 SQL문과 같은 결과를 얻기 위하여 두개의 질의 (a)와 (b) 로 나타낸 QBE의 질의를 보여 준다. 이렇게 부질의 기능을 제공하지 않음으로 인하여 질의의 작성 절차가 길어지고, 하나의

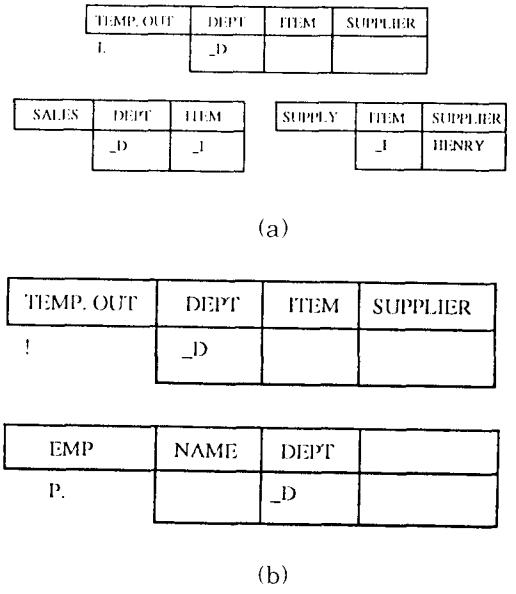


그림 5. TEMP. 연산자를 이용한 질의의 예
Fig. 5. An example of a query using the TEMP. operator.

질의로 표현이 불가능하며, 따라서 질의작성이 불편하고 복잡해짐을 볼 수 있다. 특히, QBE에서는 부정 연산자(negation operator)의 연산범위가 단일 테이블로 한정되므로, 두개 이상의 테이블을 조인한 결과를 부정하는 위와 같은 SQL문의 NOT EXISTS에 해당하는 부정 부질의가 필요한 경우에는 이러한 문제가 반드시 발생한다.

2. 부질의 기능을 갖는 QBE로의 확장

QBE에서 부질의가 가능하도록 QBE를 확장하기 위하여 부질의 작성에 사용되는 객체로서 부질의 상자(subquery box)를 그림 6과 같이 정의하여 사용한다. 부질의 상자는 부질의 연산자 영역과 부질의 영역으로 나뉘어 진다. 부질의 연산자 영역은 부질의에 대한 연산을 지정하는 장소이며, 부질의 영역에서는 테이블 구조, 조건 상자, 부질의 상자등 모든 QBE의 질의를 작성할 수 있다. 따라서, 중포된 부질의(nested subquery)가 가능하다.

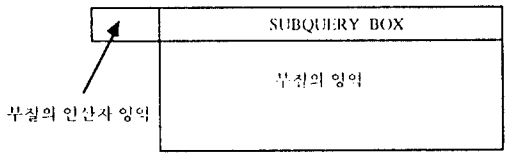


그림 6. 부질의 상자
Fig. 6. Subquery box.

예를 들어, 제 1절에서 나타낸 "HENRY가 공급하는 품목을 한가지도 판매하지 않는 부서에서 일하는 고용인의 이름과 부서를 나열하라"는 질의는 부질의 상자를 이용하면 그림 7과 같이 하나의 질의로 표현이 가능하다. 즉, 먼저 부질의 상자를 이용하여 부질의 영역에는 "HENRY가 공급하는 품목을 판매하는 부서"를 표현하는 부질의를 작성하고 부질의 연산자 영역에는 부정 연산자를 지정하여 부정 부질의어를 만들고, EMP테이블에서 부서 열에 부질의 영역의 부서 열에 사용한 예인자 _D로 바인딩하여 고용인의 이름과 부서를 출력하면 된다.

본 논문에서는 특히 두개 이상의 테이블을 조인한 결과를 부정하는 부정 부질의가 필요할 때 반드시 발생하는, 질의의 작성 절차가 길어지고, 한 화면에 하나의 질의로 표현이 불가능한 문제등 현재 QBE에서 가지고 있는 문제점을 해결하기 위해 위에서 정의한 부질의 상자를 이용하여 SQL의 NOT EXISTS에 해당하는 부정 부질의가 가능하게 QBE를 확장한다.

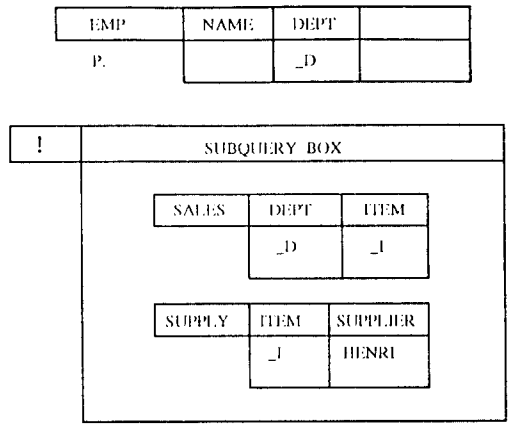


그림 7. 부질의 상자를 이용한 확장된 QBE질의의 예
Fig. 7. An example of a query in the Extended QBE using subquery box.

Ⅳ. 확장된 QBE의 구문분석

본 장에서는 확장된 QBE의 구문분석에 대한 내용을 설명한다. 그림 8 은 사용자로부터 작성된 QBE의 질의가 구문분석을 거쳐 SQL문으로 변환되어 수행되는 과정을 보이고 있다.

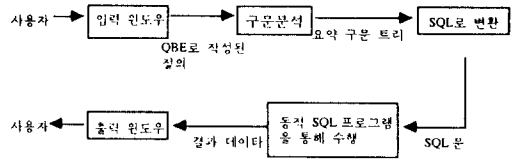


그림 8. 확장된 QBE의의 수행 과정
Fig. 8. Processing of Extended QBE queries.

구문분석의 방법은 크게 Top-down 방식과 Bottom-up 방식이 있다. Top-down 방식은 정의된 문법의 시작 심볼로부터 문법의 생성 규칙을 적용하여 주어진 질의의 구문구조를 표현하는 파스 트리(parse tree)를 생성하는 방법으로 하나의 생성 규칙에 대하여 하나의 함수로 구현되므로 이해하기가 쉽고 구현하기가 쉽다. 따라서, 우리는 Top-down방식의 한가지인 재귀적 하향 구문분석법(recursive descent parsing)으로 구문분석을 한다. 그러나, 재귀적 하향 구문분석에서 사용가능한 문법에는 약간의 제약 조건이 가해져야 한다. 1절에서는 QBE의 검색

질의를 위한 문법[1]을 이 제약 조건이 만족되게 정의하고, 제 2절에서는 문법의 제약 조건과 구문분석에 대하여 논의한다. 조건 상자에 나타나는 산술식이나 논리식의 경우에는 제약조건에 의한 문제로 연산자 순위 구문분석법(operator precedence parsing)을 사용한다.

1. 검색 질의의 문법 정의

본 절에서는 QBE 검색 질의의 구문을 그림 9와 같은 문법으로 정의한다. S는 전체 문법의 시작 심볼이고 S 에서 확장될 수 있는 TBFIELD, COLFIELD, ROWFIELD, ENTRYFIELD, 그리고 CONDBOX는 각각 테이블 구조 내의 테이블 이름 영역, 열 이름 영역, 행 연산자 영역, 엔트리 영역, 그리고 조건 상자를 나타내기 위한 시작 심볼들이다. COND는 엔트리 영역에서 사용될 수 있는 조건식을 나타내는 심볼이고, 이 조건식의 피비교연산자(comparand)를 나타내는 심볼은 AECOMPARAND이며 상수 인자, 예 인자, 스트링 식, 그리고 산술식으로 확장된다. FICOMPARAND는 조건 상자에서 처음 사용될 수 있는 피비교연산자를 나타내는 심볼로서 집단(aggregation) 연산자를 포함할 수 있으며, 일반적인 피비교연산자를 나타내는 심볼인 LECOMPARAND는 AECOMPARAND와는 달리 논리식으로 확장될 수 있다. CONDLIST는 복합 조건식을 위한 심볼로서 연속해서 비교연산자와 피비교연산자로 확장될 수 있다.

```
S             =>  TBFIELD | COLFIELD | ROWOPFIELD |
                  ENTRYFIELD | CONDBOX
TBFIELD       =>  NAME OPERATOR
COLFIELD      =>  NAME
ROWOPFIELD    =>  OPERATOR NEG
ENTRYFIELD    =>  OPERATOR ORDER GROUPOP COND
CONDBOX       =>  FICOMPARAND 'BCOMPOP LECOMPARAND
                  CONDLIST

NAME          =>  'STRING | EMPTY
OPERATOR      =>  'P. | EMPTY
NEG           =>  '¬ | EMPTY

ORDER         =>  ORDERKIND ORDERARG '. | EMPTY
ORDERKIND     =>  'AO | 'DO
ORDERARG      =>  '( 'NUM ') | EMPTY

GROUPOP       =>  'G. | AGGROPS UNQOP 'ALL. | 'ALL. | EMPTY
AGGROPS       =>  'CNT. | 'SUM. | 'AVG. | 'MAX. | 'MIN.
UNQOP         =>  'UNQ. | EMPTY

COND          =>  'UCOMPOP AECOMPARAND |
                  NEG AECOMPARAND | EMPTY
AECOMPARAND   =>  COMPARAND | '( AE ')

FICOMPARAND   =>  LECOMPARAND | AGGROPS UNQOP 'ALL. 'EE

CONDLIST      =>  'BCOMPOP LECOMPARAND CONDLIST | EMPTY
LECOMPARAND   =>  COMPARAND | '( LE ')

COMPARAND     =>  CE | 'EE | SE
SE            =>  CE 'EE NEXT1 | 'EE ' CE ' NEXT2
NEXT1         =>  ' CE ' NEXT2 | EMPTY
NEXT2         =>  'EE NEXT1 | EMPTY
CE            =>  'STRING | 'NUM
```

* 터미널(terminal)앞에 인용 부호(')를 붙여 넌터미널(nonterminal)과 구별한다
* EMPTY는 스트링의 길이가 영인 것을 의미한다
* LE(logical expression)와 AE(arithmetic expression)는 연산자 순위 구문 분석기(operator precedence parser)가 처리한다

그림 9. 테이블 구조와 조건 상자내의 구문을 위한 문법

Fig. 9. Grammar for syntax used in the table skeleton and the condition box.

1) QBE에서는 검색 질의 외의 모든 질의어에 대해서도 연산자를 제외하고는 같은 문법이 적용된다.

2. 구문분석

그림 10은 QBE 질의의 구문분석 과정을 그림으로 보인 것이다. 어휘 분석기(lexical analyzer)는 질의로부터의 문자열을 받아들여 구문 분석기(parser)가 사용하는 토큰들을 생성하고, 재귀적 하향 구문 분석기(recursive descent parser)와 연산자 순위 구문 분석기(operator precedence parser)는 생성된 토큰열로부터 요약 구문 트리를 만들어 낸다. 특히 연산자 구문 분석는 질의어에 포함된 산술식과 논리식에 대한 구문분석을 실행한다.

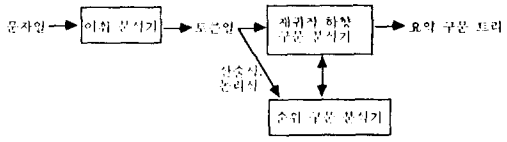


그림 10. 질의의 구문분석 과정

Fig. 10. Parsing Process of a query.

재귀적 하향 구문 분석기에서 사용가능한 문법에 대한 제약 조건으로 left recursive 문법을 다루지 못하고, 비터미날을 확장시키기 위해 사용할 수 있는 문법의 생성 규칙이 두개 이상 있을 경우에 백트래킹(backtraking)이 발생할 수 있다는 것이다.[1]

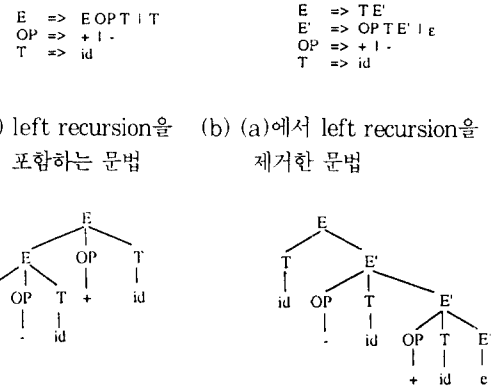
문법이 left recursive하다는 것은 $A \rightarrow A\alpha$ 와 같이 왼쪽에 있는 비터미날과 오른쪽의 첫번째 심볼이 같은 생성 규칙이 존재한다는 것이다. 무한 루프가 발생하지 않도록 하기 위해서는 다음과 같이 문법을 변환하여 left recursive 생성 규칙을 제거한다.[1] left recursive 생성 규칙의 왼쪽 비터미날 A를 위한 생성 규칙들을 $A \rightarrow A\alpha_1 | A\alpha_2 | \ldots | A\alpha_m | \beta_1 | \beta_2 | \ldots | \beta_n$ (단 β_i는 A로 시작하지 않는 심볼이다)과 같이 한 곳에 모아서, 이를 $A \rightarrow \beta_1 A' | \beta_2 A' | \ldots | \beta_n A'$ 과 $A' \rightarrow \alpha_1 A' | \alpha_2 A' | \ldots | \alpha_m A' | \epsilon$ 으로 변환시킨다. 이와 같이 문법을 변환하면 left recursive 생성 규칙들은 제거되는 대신 $A' \rightarrow amA'$ 등의 right recursive 생성 규칙들이 생성되게 된다.

재귀적 하향 구문 분석기에서는 백트랙킹을 없애기 위해 left factoring을 행한다. left factoring 은 비터미날 A를 확장시키기 위해 사용할 수 있는 생성 규칙이 두개 이상일 경우에 공통된 가장 긴 프리픽스(prefix) α 를 찾아서 $A \rightarrow \alpha\beta_1 | \alpha\beta_2 | \ldots | \alpha\beta_n | \gamma$ (γ 는 α로 시작하지 않는 비터미날 A의 모든 생성 규칙들이다) 의 형태로 만들어 이를 $A \rightarrow \alpha A' | \gamma$ 와 $A' \rightarrow \beta_1 | \beta_2 | \ldots | \beta_n$ 으로 변환하는 것이다. 여기서 A'은 새로운 비터미날이다. 각 비터미날에 대해 두 개 이상의 생성 규칙이 공통된 프리픽스를 갖지 않을 때까지 이러한 변환을 반복한다.

그림 9에서 보인 문법은 S와 COMPARAND를 제외한 생성 규칙들에 대해서 재귀적 하향 구문분석이 가능하도록 left recursion을 제거하고 left factoring을 시행한 결과이다. S와 COMPARAND의 생성 규칙들에 대해서는 left factoring을 하면 생성 규칙의 직관적인 의미가 없어지고 복잡해지므로 그대로 두었다. 대신 이들의 경우에는 생성 규칙간의 충돌을 막기 위해 테이블 구조내의 위치 정보나 다음 토큰(lookahead token)을 이용할 수 있으므로 백트래킹이 없는 구문 분석이 가능하다.

그러나, 재귀적 하향 구문 분석기를 사용하면 QBE 질의어에 포함된 산술식이나 논리식에 대한 구문분석에서 다음과 같은 문제점이 있다. 산술식과 논리식을 나타내는 문법에는 left recursive 또는 right recursive 생성 규칙들이 포함된다. left recursive 생성 규칙들은 무한 루프가 발생하지 않도록 앞에서 설명한 방법에 따라 재작성되어 제거되므로 재귀적 하향 구문 분석기가 사용하는 문법에는 right recursive생성 규칙들만 포함되게 된다. 그런데 right recursive 생성 규칙에 의해 파싱한 결과로 생성된 파스 트리(parse tree)는 오른쪽 아래로 향하게 된다. 이러한 파스 트리에는 이진 연산자들이 갖는 왼쪽 우선(left associative) 연산의 성질이 올바르게 반영되어 있지 않으므로 이 트리를 탐색하면서 연산자의 우선 순위대로 변환하기가 어렵다. 예를 들어 그림 11에서 (a)는 더하기와 빼기가 포함된 산술식을 나타내는 문법으로서 E ⇒ E OP T | T에 left recursion 이 포함되어 있다. (b)의 문법에서는 E ⇒ E OP T | T를 E ⇒ T E' 과 E' ⇒ OP T E' |ε 으로 변환시켜서 left recursion을 제거하였다. 여기서 E' ⇒ OP T E'은 right recursive 생성 규칙이다. (a)의 문법에 의해 생성된 파스 트리 (c)와 (b)의 문법에 의해 생성된 파스 트리 (d)를 비교하면, (c)는 왼쪽 아래로 향하며 (d)는 이와 반대로 오른쪽 아래로 향하고 있다.

따라서, 산술식이나 논리식에 대해서는 파스 트리에서 이진 연산자들의 왼쪽 우선 연산의 성질이 반영되게 하기 위하여 연산자 순위 구문분석을 통하여 구문분석을 한다. 연산자 순위 구문 분석기는 파싱 테이블(parsing table)을 작성하기가 용이하고 구현하기가 쉬우며, 특히 심볼들 사이의 순위 관계를 이용하여 구문 분석함으로써 연산자간의 순위를 부여할 수 있어 산술식과 논리식과 같은 연산식을 위한 구문분석에 적당하다. 반면에, 연산자 순위 구문분석이 가능한 연산자 문법은 ϵ-생성 규칙을 갖지 않아야 하고, 생성규칙의 오른쪽에 연속된 두개 이상의 비터미날을 가질 수 없다는 제약조건을 가지는 등 문법의 범위가 좁다. [1]



(a) left recursion을 포함하는 문법 (b) (a)에서 left recursion을 제거한 문법

(c) (a)를 이용한 id-id+id의 파스 트리 (d) (b)를 이용한 id-id+id의 파스 트리

그림 11. left recursion의 제거가 파스 트리에 미치는 영향

Fig. 11. Effect of elimination of left recursion on the parse tree.

구현된 구문 분석기의 각 함수는 구문 분석의 결과로써 요약 구문 트리(abstract syntax tree)를 생성한다. 그림 12는 테이블 구조의 각 영역과 조건 상자에 대한 요약 구문 트리를 예를 들어 표현한 것이다.

그림 12의 비터미날 COND와 CONDBOX에서 산술식을 나타내는 AE와 논리식을 나타내는 LE는 연산자 순위 구문 분석기로 분석된다. 따라서 비터미날 COND와 CONDBOX의 부트리(subtree)는 연산자 순위 구문 분석기가 반환한 파스 트리와 재귀적 하향 구문 분석기로부터 나온 파스 트리를 결합한 것이다. 이 부트리에서는 산술 연산자, 논리 연산자, 그리고 비

교 연산자 등이 비터미날이 되고, 피연산자인 예 인자, 숫자 상수, 그리고 문자 상수 등이 터미날이 된다.

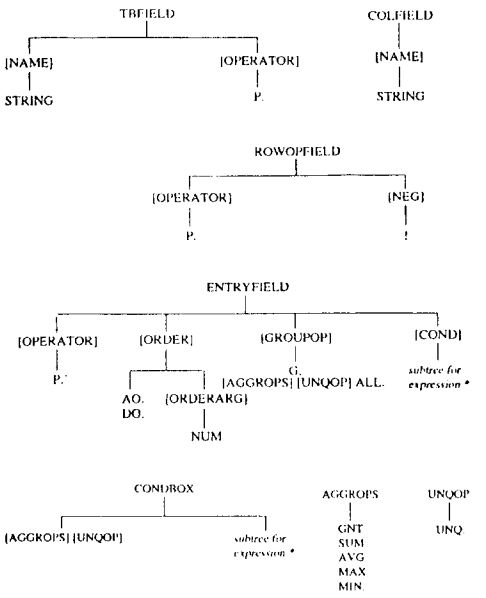


그림 12. 요약 구문 트리의 예
Fig. 12. Examples of abstract syntax tree.

V. 확장된 QBE에서 SQL로의 변환 알고리즘

본 장에서는 구문 분석의 결과로 만들어진 QBE 질의의 요약 구문 트리(abstract syntax tree)로부터 QBE 질의와 같은 의미를 지닌 SQL문으로 변환하는 알고리즘에 대하여 논의한다. 변환된 질의어는 SQL 시스템이 제공하는 동적 SQL 프로그램(dynamic SQL program)을 통하여 관계형 데이타베이스 관리 시스템에서 수행된다.

QBE 테이블의 각 영역마다 하나씩 생성된 요약 구문 트리를 직접 SQL로 변환하려면 알고리즘이 복잡하고 비효율적으로 된다. 예를 들어, 예 인자가 바인딩되거나 사용된 테이블의 이름과 열 이름등이 필요할 경우 매번 전체 트리를 탐색해야 한다는 오버헤드가 있다. 따라서, 트리를 직접 사용하지 않도록 요약 구문 트리에서 필요한 정보들을 정리하여 다른 데이타 구조들에 저장한 후 이 데이타 구조들을 검색하면서 SQL로 변환한다. 데이타 구조들에는 테이블을 관리하기 위한 테이블 데이타 구조, 행 데이타 구조, 열 데이타 구조, 예 인자들을 효율적으로 관리하기 위한 예 인자 데이타 구조, 예 인자 발생 데이타 구조, 그리고 테이블이나 조건 상자에 쓰인 식들을 저장하는 식 데이타 구조 등이 있다. 요약 구문 트리로부터 만들어진 데이타 구조로부터 SQL을 만들기 위해서는 테이블 데이타 구조, 행 데이타 구조, 그리고 열 데이타 구조등을 차례로 검색하면서 출력 연산자가 있는 행마다 SELECT절, FROM절, WHERE 절, 그리고 부질의절 등을 차례로 작성한다. 다음은 SQL의 각 절을 작성하는 과정에 대한 설명이다.

• SELECT 절

행 데이타 구조와 열 데이타 구조의 연산자 필드에서 출력 연산자가 사용되었는가를 검사한다. 출력 연산자만 사용된 경우에는 열 이름들만으로 SELECT 절을 만들고, 집단 연산자와 같이 사용된 경우에는 SQL의 해당 집단 연산자와 함께 SELECT절을 만든다. 그리고 테이블의 각 행마다 이름을 부여하여 이를 상관 이름(correlation name)이라 하고 열 이름 앞에는 항상 이 상관 이름을 쓰도록 한다.

• FROM 절

출력 연산자가 있는 행에 관계가 있는 예 인자를 연관된 예 인자라 하고, 연관된 예 인자들이 속한 모든 테이블들을 FROM절에 포함시킨다. 따라서, FROM절을 만들기 위해서 는 출력 연산자가 있는 행에 연관된 예 인자들을 모두 찾아야 하며 연관된 예 인자들을 다음과 같이 재귀적으로 정의한다.

1. 출력 연산자가 있는 행에서 사용된 예 인자들은 연관된 예 인자이다.

2. 연관된 예 인자가 사용된 또 다른 행이 존재하면 그 행에 나타난 모든 예 인자들도 연관된 예 인자이다.

연관된 예 인자들은 행 데이타 구조, 열 데이타 구조, 식 데이타 구조, 예 인자 데이타 구조, 그리고 예 인자 발생 데이타 구조 등을 검색하여 알 수 있다. 연관된 예 인자들을 찾은 후 연관된 예 인자들이 사용된 행마다 행이 속한 테이블의 이름과 행에 지정된 상관 이름을 연결하여 FROM절을 만든다.

• WHERE 절

연관된 예 인자들이 사용된 행들을 찾아서 각 열에 있는 식들을 SQL에 적합한 조건식들로 변환한 후 논리곱(AND)으로 연결시킨다. 각 예 인자마다 바인딩된 테이블, 행, 그리고 열 등은 하나로 지정되며, 예 인자들은 바인딩된 테이블의 행에 지정된 상관 이름과 열 이름을 연결하여 대치된다. 단항 비교 연산자가 사용된 경우에는 사용된 테이블의 행에 지정된 상관 이름과 열 이름을 또 다른 하나의 피연산자로 하여 완전한 비교식으로 만든다.

• 부질의 절

본 연구에서 구현한 부정 부질의는 SQL의 NOT EXISTS절로 변환된다. NOT EXISTS뒤의 부질의문에서 SELECT절은 항상 " SELECT * " 가 되며 FROM절과 WHERE절을 만드는 과정은 앞에서 설명한 것과 같다. 다만 FROM절에서 연관된 예 인자들은 부질의 상자 안에서만 사용된 예 인자를 대상으로 하며, 부질의 상자 안과 밖에서 모두 사용된 예 인자는 부질의문 밖의 FROM절에서 그 예 인자가 사용된 행에 대한 상관 이름이 주어진다.

그림 13은 "HENRY가 공급하는 품목을 한가지도 판매하지 않는 부서에서 일하는 고용인의 이름과 부서를 나열하라"는 제 Ⅲ장 2절에서 표현한 그림 7의 부정 부질의어가 포함된 QBE 질의를 SQL로 변환한 예를 보인 것이다. EMP 테이블에서 출력 연산자가 있는 행에 상관 이름으로 L1을 지정하면 대응되는 SELECT절은 "SELECT L1.NAME, L1.DEPT" 가 되고, 연관된 예 인자로 D가 있으므로 FROM절은 " FROM EMP L1 " 이 되며, 부질의 상자를 사용하여 표현한 부정 부질의는 SQL의 NOT EXISTS절로 변환된다. NOT EXISTS 뒤의 부질의문에서 SELECT절은 "SELECT * " 가 되고, 부질의 상자 안에서만 사용된 연관된 예 인자는 I로서 SALES 테이블의 첫번째 행과 SUPPLY 테이블의 첫번째 행에서 사용되었으므로 각 행에 상관 이름으로 L2, L3을 지정하면 FROM절은 "FROM SALES L2, SUPPLY L3"가 된다. WHERE절은 예 인자 D와 I에 대한 두개의 죠인 조건식과 상수 인자 HENRY에 대한 하나의 조건식으로서, 각 예인자들은 바인딩된 테이블의 행에 지정된 상관 이름과 함께 열 이름으로 대치되고 상수 인자 HENRY에 대한 조건식에는 이 행에 지정된 상관 이름과 열 이름이 새로운 피연산자로 사용되어, "WHERE L1.DEPT = L2.DEPT AND L2.ITEM = L3.ITEM AND L3.SUPPLIER = 'HENRY' "가 된다.

```
SELECT  L1.NAME, L1.DEPT
FROM    EMP L1
WHERE   NOT EXISTS (   SELECT *
                       FROM SALES L2, SUPPLY L3
                       WHERE L1.DEPT = L2.DEPT AND
                             L2.ITEM = L3.ITEM  AND
                             L3.SUPPLIER = 'HENRY'   )
```

그림 13. 부질의 상자가 있는 QBE 질의로부터 변환된 SQL 질의

Fig. 13. A SQL query transformed from a QBE query having subquery box.

Ⅵ. 결 론

본 논문에서는 사용자가 데이타베이스를 용이하게 관리할 수 있도록 설계된 전시 지향적 데이타 처리 언어 중의 하나인 QBE에 부질의 기능을 확장하고 이를 구현한 내용에 대하여 논하였다.

QBE에서는 질의의 형식이 테이블 구조와 조건 상자 등과 같은 그래픽 객체들에 의해 이미 정의되어 있고 데이타베이스에 대한 여러가지 연산을 통일된 방식으로 할 수 있으므로 문법이 간단하며 사용자는 쉽게 질의를 작성할 수 있다. 그러나, QBE에는 SQL에서와 같은 부질의의 기능을 정의하고 있지 않으므로 부질의가 포함된 하나의 SQL문을 QBE에서는 하나의 질의로 표현하기가 불가능하다. 본 논문에서는 이 문제를 해결하기 위해 QBE에 부질의 상자라는 새로운 객체를 정의하여 부질의의 직접적 표현이 가능하도록 확장하고, 이를 이용하여 SQL의 NOT EXISTS절에 해당하는 부정 부질의가 가능하도록 하였다. 따라서, 기존의 QBE에서 한 윈도우에 표현할 수 없었던 질의를 표현 가능하게 함으로써 질의의 작성에 드는 오버헤드를 해결하였다.

확장된 QBE의 질의의 구문을 분석하기 위해서는 재귀적 하향 구문분석기와 연산자 순위 구문분석기를 동시에 구현하였다. 재귀적 하향 구문 분석기는 문법의 각 생성 규칙에 대해 하나의 재귀적 함수로 구현이 되므로 이해하기 쉽고 구현이 쉽다. 그러나, 재귀적 함수는 left recursive 생성규칙에 대해서는 무한 루프를 발생시키므로 left recursive 생성규칙들은 규칙변환을 통해 제거되어야 한다. 이때 산술식이나 논리식의 경우에 left recursive 생성규칙들을 제거하면 구문분석의 결과로 만들어진 파스 트리에서 연산자의 우선순위를 반영하는 요약 구문 트리로의 변환이 어렵다. 따라서, 확장된 QBE에 나오는 산술식이나 논리식에 대해서는 연산자의 우선 순위의 반영을 쉽게할 수 있는 연산자 순위 구문 분석기를 구현하여 구문분석하게 하였고, 그외의 질의어에 대해서는 재귀적 하향 구문 분석기를 구현하여 구문분석하게 하였다.

또한, 본 논문에서는 QBE를 SQL로 변환하는 기법을 제안하였다. QBE 질의의 구문분석 결과로 만들어진 요약 구문 트리로부터 직접 SQL로 변환하지 않고 일단 변환 알고리즘에서 필요로 하는 정보를 쉽게 얻을 수 있는 데이타 구조를 구축하고, 이 데이타 구조를 차례로 검색하면서 출력 연산자가 있는 행마다 SELECT절, FROM절, WHERE절 그리고 부질의절 등을 차례로 작성하여 SQL질의를 만들었다.

확장된 QBE는 실현 가능성을 입증하기 위하여 IBM OS/2 상에서 구현하였으며, 보다 편리한 사용자 인터페이스를 제공하기 위하여 윈도우 시스템을 이용하였다. 앞으로 더 연구해야 할 방향은 SQL의 NOT EXISTS절에 해당하는 부질의 외에 IN, SOME, ANY, ALL, 집단 연산자(aggregation operators), 그리고 비교 연산자 등과 같이 쓰이는 부질의도 가능하도록 QBE를 확장하는 것이다.

參 考 文 獻

[1] Aho, A. V., Sethi, R., and Ullman, J. D., *Compilers: Principles, Techniques, and Tools*, Addison-Wesley Publishing Company, Massachusetts, 1988.

[2] Chamberlin, D. D. et al., "SEQUEL 2: A Unified Approach to Data Definition, Manipu-lation, and Control," *IBM J. Research and Development*, Vol. 20, No. 6, pp. 560-575, 1976.

[3] Date, C. J., A Guide to the SQL Standard, Second Edition, Addison-Wesley, Reading, 1989.

[4] Elmarsi, R. A. and Larson, J. A., "A Graphical Query Facility for ER Databases," In *Proc. 4th IEEE Int.l Conf. on EntityRelationship Approach*, pp. 236-245, 1985.

[5] IBM, Query-By-Example Terminal User's Guide, IBM Corp., New York, Fourth Edition, 1978.

[6] Jacobs, B. E. and Walczak, C. A., "A Generalized Query-by-Example Data Manipulation Language Based on Database Logic," *IEEE Trans. on Software Engineering*, Vol. SE-9, No. 1, pp. 40-56, Jan. 1983.

[7] Kim, W., "On Optimizing an SQL-like Nested Query," *ACM Trans. on Database Systems*, Vol. 7, No. 3, pp. 443-469, Sep. 1982.

[8] Kim, H. J. et al., "PICASSO: A Graphical Query Language," *Software-Practice and Experience*, Vol. 18, No. 3, pp. 169-203, Mar. 1988.

[9] Maier, D. and Ullman, J. D., "Maximal Objects and the Semantics of Universal Relation Databases," *ACM Trans. on Database Systems*, Vol. 8, No. 1, pp. 1-14, Mar. 1983.

[10] McDonald, N. and Stonebraker, M., "CUPID-the Friendly Query Language," In Proc. ACM Pacific Conf., San Francisco, pp. 127-131, 1975.

[11] Schauer, U., *Ein System zur interaktiven Bearbeitung umfangreiche Messdaten*, Hasselmeier and Spruth, Eds., SpringerVerlag, Berlin, p. 1213, 1976.

[12] Senko, M. E., "The DDL in the Context of a Multilevel Structures Description: DIAM II with FORAL," In *Data Base Description, Douque and Nijssen, Eds., NorthHolland, Amsterdam*, p.239, 1975.

[13] Stonebraker, M. et al., "The Design and Implementation of INGRES," *ACM Trans. on Database Systems*, Vol. 1, No. 3, pp. 189-222, 1976.

[14] Thomas, J. C. and Gould, J. D., "A Psychological Study of Query by Example," In *Proc. National Computer Conf.*, Vol. 44, pp. 439-445, 1975.

[15] Ullman, J. D., *Database and Knowledge-Base Systems*, Computer Science Press, Maryland, 1988.

[16] Wong, H. K. T. and Kuo, I., "GUIDE: Graphical User Interface for Database Exploration," In *Proc. 8th Intl. Conf. on Very Large Data Bases*, Mexico City, pp. 22-32, Sept. 1982.

[17] Zhang, Z. -Q. and Mendelzon, A. O., "A Graphical Query Language for Entity- Relationship Databases," In *Entity Relationship Approach to Software Engineering*, C.G. Davis *et al.*, Eds., Elsevier Science, New York, pp. 441-448, 1983.

[18] Zloof, M. M., "Query-by-Example: A Data Base Language," *IBM Systems Journal*, Vol. 16, No. 4, pp. 324-343, 1977.

著者紹介

元 希 先(正會員)

1990年 2月 연세대학교 전산과학과 졸업(B.S.). 1992年 2月 한국과학기술원 전산학과 졸업(M.S.). 1992年 4月 ~ 현재 한국방송공사 기술연구소 연구원. 주관심 분야는 그래픽 데이타베이스 질의어, 영상처리 등임.

李 鍾 鄗(正會員)

1982年 2月 경북대학교 전자공학과(전자계산전공) 졸업(B.S.). 1984年 2月 한국과학기술원 전산학과 졸업(M.S.). 1991年 정보처리 기술사. 1993年 3月 ~ 현재 한국과학기술원 전산학과 박사과정. 1984年 3月 ~ 1987年 5月 금성통신(주) 부설연구소 주임연구원. 1987年 5月 ~ 현재 한국통신 연구개발원 선임연구원. 주관심 분야는 컴퓨터 통신, 객체지향 데이타베이스, 데이타베이스 질의어처리 등임.

黃 奎 永(正會員)

1973年 서울대학교 전자과 졸업(B.S.). 1975年 한국과학기술원 전기 및 전자공학과 졸업(M.S.). 1982年 Stanford University(전산학, M.S.). 1983年 Stanford University(전산학, Ph.D.). 1975年 ~ 1978年 국방과학연구소 선임연구원. 1983年 ~ 1990年 IBM T.J. Watson Research Center, Research Staff Member. 1990年~현재 한국과학기술원 전산학과 교수, 인공지능 연구센터 데이타베이스 및 멀티미디어 연구실장. 1991年 Visiting Professor, Stanford University. 1992年 Visiting Associate Professor, Georgia Institute of Technology. 1993年 Hewlett-Packard Laboratories 기술자문(Palo Alto). 1992年 ~ 1994年 한국정보과학회 데이타베이스연구회(SIGDB) 운영위원장. 1993年 ~ 현재 체신부 통신진흥협의회 DB산업육성분과 위원장. Editor: The VLDB Journal. Editor: Distributed and Parallel Databases: An International Journal. Editor: International Journal of Geographical Information Systems. Associate Editor: The IEEE Data Engineering Bulletin. 1990-1993. 주관심 분야는 멀티미디어 데이타베이스, 객체지향 데이타베이스, 연역 데이타베이스, 공학 데이타베이스, 사무자동화, CASE, 전문가시스템 등임.