표현력 증대를 위한 복합 객체 대수의 설계

송 지 영[†] · 배 해 영[††]

요 약

복합 객체 모델은 복잡한 구조 데이타를 지원하기 위해 기존의 관계형 데이타 모델을 자연스럽게 확장한 값 중심 데이타 모델이다. 본 논문은 복합 객체 모델의 검색 연산을 대상으로 대수적 최적화 기법의 적용이 용이한 대수 설계를 제안한다. 이를 위해 우선, 복합 객체 처리를 목적으로 제안된 기존의 대수들을 대수적 최적화 기법의 적용성을 기준으로 비교 분석한다. 또한 단순 명료성, 데이타 구조에 대한 무제약성, 단일 표현 체계의 설계 원칙을 기반으로 복합 객체 대수를 설계한다.

제안된 대수의 특징은 다양한 구조의 복합 객체에 대한 대수의 안정성과 단순성을 유지하고 동시에 대수적 최적화 기법의 적용이 용이한 점이다. 끝으로 본 연구에서 설계된 대수가 기존 대수들과 동등하거나, 더 향상된 표현력을 갖음을 보인다.

Design of the Complex Object Algebra for Enhancing Expressive Power

Song Gy Young[†] · Bae Hae Young[††]

ABSTRACT

A complex object model is one of the value based data model which extends the existing relational data model for supporting complex structured data. This paper studies a method for designing algebra for the complex object model. For this some others' algebra supporting complex objects are compared and analysed in terms of the applicability of algebraic optimization strategies. The complex object algebra is designed, based on four principles, simple and clear definitions, no restriction on input data, single specification system.

The central nature of this paper is to keep the basis of algebraic optimization method through simplicity, safety and the applicability of algebraic optimization strategy. Finally, it is shown that the designed algebra has the equivallent or enhanced expressibility with others' algebra.

1. 서 론

논리적인 단계에서 값 중심 데이타 모델은 개체(entity) 조작시 값 개념을 사용한다. 예를 들어 개체 참조 또는 개체간의 관련성은 키-애트리뷰트 값을 사용하여 식별된다. 또한 개체 데이타의 접근, 변경 연산은 그들과 관련된 값을 참조함으로서 이루어진다.

관계형 데이타 모델[1]은 값 중심 데이타 모델의 대표적 예이다. 관계형 데이타 모델에서, 개체는 튜플 또는 키를 사용하여 표현되고, 개체 조작은 죠인 연산(join operation)과 같은 개체를 나타내는 값의 조작을 통해서 이루어진다. 이러한, 관계형 데이타 모델은 정형화된 데이타의 응용분야에서 가장 많이 사용되어져왔음에도 불구하고, 모든 개체를 튜플-애트리뷰

† 정 회 원:인하 대학교 전자 계산공학과
†† 종신회원:인하 대학교 전자 계산공학과
논문접수:1995년 10월 4일, 심사완료:1996년 8월 2일

트로 구성된 이차원의 단순 테이블 형태로 표현하므로 비정형 응용 분야에서 요구되는 복잡한 구조의 개체를 모델링하기에는 부적합하다. 실세계의 복잡하고 계층적인 구조를 갖는 대상들을 나타내는 복합 객체[2]를 관계형 데이타 모델을 이용해서 모델링하기 위해서는 제 1 정규화 규칙에 따라 평면적 구조로 바꾸어 모델링해야 한다. 이 과정에서 복합 객체를 구성하는 객체들간의 관련성이 여러 테이블로 나뉘게 되고, 개체 인식상의 문제(anomaly)가 발생하므로, 실세계에서 독립된 단일 의미를 나타내는 복합 객체 원래 의미를 상실한다[3]. 또한 기존의 관계 대수는 개체 구성 방법이 테이블, 뷰에 한정되어 있으므로 복잡한 구조의 개체 구성 방법이 제공되지 못하고, 이행성 폐포(transitive closure), aggregate와 같은 연산이 정의되어 있지 않으므로 표현력도 부족하다.

복합 객체 모델[2, 4]은 값 중심 데이타 모델로서 관계형 데이타 모델을 확장하였다. 즉 관계형 데이타 모델의 제 1 정규화 규칙의 제약 조건을 완화하여 릴레이션 애트리뷰트로 원자성을 갖지 않는 복합 객체를 허용함으로서 관계형 데이타 모델보다 더 용이하고 효과적으로 복합 객체를 모델링하는 수단을 제공한다. 이와 같이 효율적인 복합 객체 지원을 위해 새롭게 제안된 데이타 모델은 이전 관계형 데이타 모델과는 달리 강력한 모델링 도구를 가지고 복잡한 구조의 데이타를 손쉽게 모델링하는 기능을 제공하지만, 부가적으로 이들 데이타에 대한 효율적인 질의 처리 기법도 보장되어야 한다.

사용자가 작성한 비절차적인 질의로부터 결과 테이블을 추출하기 위해서는 일련의 데이타베이스 연산으로 구성된 절차적인 질의로의 변환이 요구된다. 관계형 데이타베이스 관리 시스템(Relational DataBase Management System : RDBMS)은 재배열 기법으로써 수학적 도구인 관계 대수(relational algebra)를 사용한다. 각각의 데이타베이스 연산들은 관계 연산자에 대응되고, 이들 연산자로 구성된 연산식을 연산자 사이의 항등 관계와 경험적 방법을 이용하여 최소의 실행 시간을 얻도록 재배열한다. 이와 같이 수학적으로 잘 정의된 관계 대수에 기반을 둔 질의 처리 기법은 RDBMS의 큰 장점이다.

복합 객체 지원을 위해 제안된 데이타 모델[4, 5, 6]은 대수적 질의 처리 기법의 재배열을 통한 최적화 및 기타의 효용성 때문에 관계형 데이타 모델의 관계 대수에 대응되는 대수를 지원한다. 이들 대수는 관계 대수의 제한된 표현력이 갖는 단점을 보완하고 복잡한 구조의 데이타에 대한 다양한 연산을 제공한다[3]. 그러나 이들 대수들의 표현력은 증대되었지만 관계 대수가 가지는 안정성(safety), 단순성(simplicity)을 유지하지 못한다.

관계 대수의 안정성은 집합 이론과 1차 프레디키트 논리(first-order predicate logic)에 이론적인 기반을 두고 모든 대수식은 반드시 종결되며 그 결과가 릴레이션이 되도록 보장한다. 그리고 관계 대수에 포함된 연산자들의 단순한 의미는 연산 구현 알고리즘의 전체적인 구조나 제어 흐름을 복잡하지 않게 하며 연산의 명확성을 보장한다. 이에 반해, 새로운 대수들은 이론적인 기반이 충분히 마련되지 않은 상태이며 다루는 데이타의 구조적 특성 때문에 연산자의 의미가 복잡하다[3, 5]. RDBMS에서 이용된 대수적 질의 처리 기법의 핵심은 관계 대수가 가지고 있는 안정성, 단순성에 의존하는 것이기 때문에 이런 성질이 유지되지 못하는 대수들을 이용한 시스템은 대수적 기법의 적용으로 얻어지는 최적화 이득을 얻을 수없다. 따라서 복합 객체를 포함한 질의의 효율적인 질의 처리 보장을 위해 대수적 최적화 기법의 적용이 용이한 대수에 대한 연구가 요구된다.

한편 집합, 리스트, 트리, 배열과 같은 개개 데이타로 부터 단일 의미를 나타내는 집단 데이타(bulk data)의 연산 처리에는 필수적으로 집단 구성 원소들에 대한 다양한 형태의 반복 처리가 요구된다. 이들 반복 처리의 추상화(abstraction)는 1차 논리 개념으로서는 형식화시킬 수없다. [6]의 함수 언어는 이러한 반복 처리에 대한 추상화를 함수적 서식(funtional form)인 고차 함수(higher order form)로 서식화하고 있지만, 언어 특성상 함수적 서식 사용에 제한이 없으므로 직접 데이타베이스 대수 체계에 적용이 불가능하다[7].

본 논문은 복합 객체 모델에서의 검색 연산을 대상으로 안정성과 단순성을 유지하는 동시에 대수적 최적화 기법이 용이한 대수를 제안한다. 복합 객체 모델은 관계형 데이타 모델을 확장한 값 중심 모델이고 모델링할 수있는 복합 객체 구조도 다양하기 때문이다. 복합 객체 모델에서의 질의 기본 단위는 복합 객체 모임 즉, 집단 데이타이고, 검색 연산은 데이타베

이스 내의 데이타에 부작용(side effect)이 없는 함수로 간주될 수 있다. 따라서 복합 객체 데이타에 대한 다양한 검색 연산 정의는 집단 데이타의 반복 처리에 대한 추상화를 형식화하는 문제와 유사하다. 이러한 접근 방식으로 복합 객체 데이타의 검색 연산을 정의하는 연구가 이루어지고 있다[7, 8, 9]. 이들 연구의 기본적인 접근 방식은 프로그래밍 언어 분야에서 처럼 기본 타입에 타입 구성자를 적용해서 구조화된 데이타 타입을 정의하고, 이들을 가능한 모든 형태로 결합해서 복합 객체의 계층 구조를 표현하는 것과 유사한 형태이다. 본 논문에서도 이와 유사한 접근 방법을 사용한다.

본 논문의 구성은 다음과 같다. 2장은 관련 연구로서 복합 객체 모델의 구조와 연산 특성을 기술하고, 중첩 관계 데이타 모델과 복합 객체 모델에서 제안된 대수를 복합 객체의 중첩 구조 관리 효율성, 대수적 최적화 기법의 적용성을 기준으로 비교 고찰한다. 3장에서는 복합 객체 대수를 구성하는 연산자들의 설계 원칙을 제안하고, 안정성과 단순성을 유지하는 동시에 효과적으로 연산식 재배열 기법을 적용할 수 있는 복합 객체 대수를 정의한다. 4장에서는 본 논문에서 제안한 복합 객체 대수와 기존 중첩 관계형 데이타 모델에서의 대수를 표현력, 대수적 최적화 기법 측면에서 비교, 평가한다. 끝으로 5장에서 결론을 맺는다.

2. 관련 연구

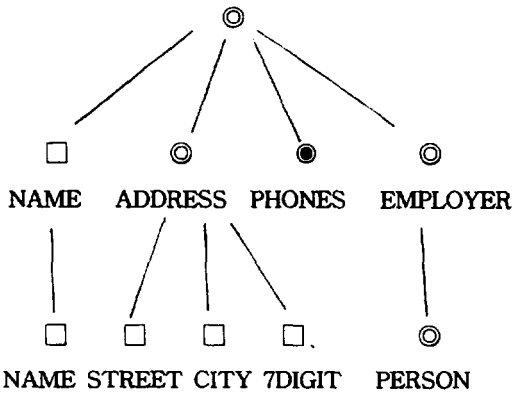
2.1 복합 객체 모델

값 중심 데이타 모델은 관계형 데이타 모델, 중첩 관계형 데이타 모델 그리고 복합 객체 데이타 모델을 포함한다. 관계형 데이타 모델[1]에서 릴레이션은 애트리뷰트 값으로 기본값을 갖는 튜플 집합이고, 관계 대수 연산자 집합은 합집합(∪), 차집합(−), 곱집합(×), 추출(Π), 선택(σ)의 연산자로 구성된다. 중첩 관계형 데이타 모델[6]의 릴레이션은 애트리뷰트 값으로 기본 값과 다른 릴레이션을 갖는 튜플 집합으로 구성된다. 즉 중첩 관계형 데이타 모델의 릴레이션 구성 원소는 스칼라 값에 국한되지 않고, 튜플집합(relation of tuple set)도 포함한다. 대수 연산자 집합은 기존 관계 대수의 연산자 집합과 중첩된 릴레이션의 조작 기능을 수행하는 NEST/UNNEST 연산자로 구성된다. 복합 객체 데이타 모델[5]의 객체는 집합 또는 튜플 타입 구성자로 부터 임의 형식으로 구성된 값 집합으로 표현되고, 두 개체 사이의 동일성은 전체 구조를 조사(traverse)함으로서 판별된다. 대수 연산자는 기본적으로 합집합, 차집합, 곱집합의 연산자를 포함하고, 멱집합(powerset)과 set-collapse를 포함한다. 복합 객체 대수에서는 기본 연산자를 반복 사용하여 중첩 관계 대수의 NEST/UNNEST 연산자를 충분히 표현 가능하다.

본 논문에서 설계하는 대수 도메인은 복합 객체 모델의 구조적 측면[2, 4]을 기반으로 한다. [4]의 복합 객체 모델은 튜플을 구성하는 애트리뷰트 값으로 기본값, 튜플값, 집합값을 허용하고 객체 참조시 값을 이용하는 값 중심 데이타 모델이다. 또한 이 데이타 모델은 관계형 데이타 모델과 중첩 관계형 데이타 모델의 릴레이션 구조를 모두 포함한다.

본 논문에서 복합 객체 모델의 구조를 나타내는 릴레이션은 애트리뷰트 값으로 복합 객체를 갖는 튜플 집합으로 정의된다. 객체는 (타입, 값)의 쌍으로 순환적으로 정의되고, 객체값은 타입에 의존한다. 임의 애트리뷰트가 $A_1, A_2, \cdots, A_n (n \geq 1)$이고, 그들의 각기 타입이 $T_1, T_2, \cdots, T_n$일 때, 튜플 타입은 $[A_1:T_1, A_2:T_2, \cdots, A_n:T_n]$, 타입 T의 집합 타입은 {T}로 각각 나타낸다. 한편, 값 집합이 $V=\{V_i\}$일 때, 위에 대응된 튜플값과 집합값은 각기 $[A_1:V_1, A_2:V_2, \cdots, A_n:V_n]$, $\{V_1, V_2, \cdots, V_n\}$로 나타낸다. 복합 객체는 서로 연관성있는 여러 객체로부터 계층적 구조의 단일 객체를 표현하기 위해 튜플 구성자와 집합 구성자를 사용한다. (그림 1)은 복합 객체 타입의 표현 예이다 . 단말 노드인 □은 기본 데이타 타입, ◎는 튜플 구성자, ⊙은 집합 구성자를 각기 나타낸다. 예를 들어 COMPANY 타입 인스턴스(instance)는 (NAME:w, ADDRESS:x)의 튜플 집합◎[C]로 나타내고, PERSON 타입 인스턴스는 (NAME:w, ADDRESS:x, PHONES:y, EMPLOYER:z)의 튜플 집합 ◎[T]로 나타낸다. COMPANIES는 COMPANY의 인스턴스 집합으로서 ⊙[C]로, PERSONS는 PERSON 인스턴스 집합인 ⊙[P]로 나타낸다.

연산적인 측면에서 복합 객체에 대한 검색 연산자는 크게 집합 연산자(set operator), 재구성 연산자(restructuring operator)로 구성된다. 집합 연산자는 질의의 기본 단위인 집합 즉 객체들의 모임에 대한 연산자이



(그림 1) 복합 객체 타입 'PERSON'
(Fig. 1) Complex object type "PERSON"

고, 재구성 연산자는 원하는 질의 결과를 얻기 위해 릴레이션 구조를 변경하는 연산자이다. 앞으로의 내용에서 특별히 구분할 경우를 제외하고 질의는 검색 질의만을, 연산은 검색 연산만을 의미한다.

2.2 대수의 비교

본 절은 관계형 데이타 모델을 확장해서 복합 객체를 지원하는 중첩 관계형 데이타 모델과 [4]의 복합 객체 모델에서 복합 객체를 지원하기 위해 제안된 기존 대수들을 비교, 고찰한다. 대상이 될 대수들은 중첩 관계형 데이타 모델을 기반으로 하는 [10]의 비순환 중첩 관계 대수와 [5, 8, 11]의 순환 중첩 관계 대수, 그리고 복합 객체 모델을 기반으로 하는 [4]의 복합 객체 대수이다. 이들 대수는 각기 릴레이션을 기반으로 하는 값 중심 데이타 모델의 대표적인 대수들이며, 대수 구성 연산자, 복합 객체의 중첩 구조를 다루는 방법, 대수적 최적화 기법의 적용성 관점에서 비교, 고찰한다.

2.2.1 중첩 관계형 데이타 모델의 대수

비순환 중첩 관계형 대수(non recursive nested relational algebra)의 연산자는 집합 연산자와 재구성 연산자로 구성된다. 집합 연산자는 UNION(∪), DIFFERENCE(−), CROS S-PRODUCT(×)으로 구성되고 이들의 의미는 관계 대수[1]에서와 동일하다. 재구성 연산자는 SELECT(σ), PROJECT(π), NEST(ν), UNNEST(μ)로 구성된다. 이중 SELECT, PROJECT 연산자는 관계 대수에서와 그 기능이 동일하다. NEST, UNNEST는 복합 객체를 조작하는 연산자로서 다음 의미를 갖는다[10].

$NEST_{A\rightarrow A*}(S)$: 릴레이션 S의 애트리뷰트 집합 A에 포함된 애트리뷰트를 제외한 모든 애트리뷰트 값이 동일한 튜플들로 단일 튜플을 구성하고, 애트리뷰트 집합 A에 포함된 동일한 애트리뷰트값들이 새로운 튜플의 애트리뷰트 A의 집합 원소이다.

$UNNEST_{A*}(S)$: NEST 연산의 역연산이며 릴레이션 S의 각 튜플들에 대하여 집합 값을 갖는 애트리뷰트A*의 원소마다 새로운 튜플을 구성한다.

이 대수의 특징은 모든 연산자들이 가장 외부 릴레이션 값에 대해서만 연산을 할 수 있다는 점이다. 이 특성으로 인해 깊이 내포된 릴레이션 값에 대한 연산은 UNNEST 연산자를 반복 적용하여 외부로 빼낸 뒤 연산을 수행하고, 연산 수행 후, 역으로 NEST 연산자를 취해 원래 위치로 환원시킨다. 이러한 방법으로 복합 객체의 중첩 구조를 관리하므로, 대부분의 질의 연산식은 길게 나열된 NEST, UNNEST 연산자를 포함한다. 한편 대수적 최적화 관점에서 보았을 때, UNNEST, NEST 연산자는 각기 그리고 다른 연산자와의 교환 법칙이 성립하지 않기 때문에 이들을 포함한 연산식에는 재배열 기법을 적용할 수 없다. 또한 NEST와 UNNEST 연산자 사이의 역연산 관계는 피연산자 릴레이션이 PNF(Partitioned Normal Form)의 제약 조건[10]을 만족해야 한다. PNF는 단일 릴레이션 내에서 기본값을 갖지 않는 모든 애트리뷰트들은 기본값을 갖는 애트리뷰트에 함수적으로 의존해야만 한다는 조건으로서 피연산자 데이타 구조에 대한 제약을 나타낸다.

순환 중첩 관계 대수(recursive nested relational algebra)[5, 11]는 비순환 중첩 관계 대수와 연산자 구성은 동일하지만 연산자 속성은 상이하다. 순환 중첩 관계 대수의 모든 연산자는 경로 수식(path expression) 역할을 하는 릴레이션 스킴(scheme)을 사용해서 깊이 내포된 릴레이션 애트리뷰트를 명시함으로서 재구성 연산의 문제점을 해결하고자 하였다. 연산자들은 스킴 형태에 따라 순환적으로 정의되지만 릴레이션 스킴을 갖는 연산자 경우 스킴 형태에 따라 그 의미가 다르므로 질의 최적화를 위한 항등 변환 규칙의 생성

이 용이하지 않다. [5]의 경우 PROJECT 연산자 의미가 10가지나 되고, 각 경우별로 충분한 항등 변환 규칙을 제공하지 못한다.

2.2.2 복합 객체 모델의 대수

[4]의 복합 객체 대수(complex object algebra) 연산자도 집합 연산자와 재구성 연산자로 구성된다. 집합 연산자는 UNION, INTERSECT, DIFFERENCE, CROSS-PRODUCT로 구성되고, 그 의미는 집합 이론에 정의된 연산자 의미와 동일하다. 재구성 연산자는 SET-COLLAPSE, POWERSET, REPLACE로 구성된다.

SET-COLLAPSE: 피연산자로서 멱집합(set of set)이 주어졌을 때, 원소 집합을 합하여 단일 집합을 구성한다.

POWERSET: 피연산자 집합의 모든 부분 집합을 생성한다.

$REPLACE_{\langle G \rangle}(S)$: 피연산자 집합 S의 각 원소에 대해 치환 명세(replace specification) G를 적용하여 그 결과를 구성 원소로 하는 새로운 집합을 생성한다.

복합 객체 모델은 중첩 관계형 데이타 모델보다 모델링할 수있는 데이타 구조가 더욱 다양하므로, 다양한 중첩 구조의 데이타를 조작하기 위한 여러 종류의 재구성 연산자가 요구된다. REPLACE 재구성 연산자는 SELECT, PROJECT, COLLAPSE 기능을 모두 포함하는 강력한 연산자이다. 특히 매개 변수로서 REPLACE 연산자의 반복 적용을 허용함으로서 깊이 내포된 릴레이션에 대해서도 재구성 연산을 가능케 한다. 대수적 최적화 관점에서 보았을 때, REPLACE 연산자는 다양한 연산 표현 기능을 제공하므로 연산자 수가 다른 연산자에 비해 상대적으로 적어지는 장점을 가지고 있지만, 생성 가능한 항등 변환의 수가 상대적으로 감소된다. 또한 REPLACE 연산자의 매개 변수를 기술하는 데 사용하는 치환 명세는 복합 객체 대수의 대수 연산식과 상이한 형태를 가지고 있기 때문에 이 치환 명세에는 재배열 기법을 적용할 수 없다.

3. 복합 객체 대수의 설계

대수적 최적화 기법 적용이 용이한 복합 객체 대수를 설계하기 위해서 효과적인 재구성 연산자의 정의가 요구된다. 재구성 연산자는 연산자가 단순 명료한 의미를 갖고, 연산식 내에서 자유로운 구성이 가능하도록 설계되어야 한다. 또한 피연산자 데이타에 대한 제약 조건이 없어야 한다. 2장의 고찰 결과를 통해 대수적 최적화 기법의 적용을 어렵게 하는 것은 재구성 연산자의 문제점임을 알 수 있다. 따라서 최적화 기법 적용이 용이한 복합 객체 대수를 만들기 위해서는 재구성 연산자들을 효과적으로 설계해야 한다. 본 연구에서 제안한 설계 원칙도 이점에 중점을 두고 있다. 설계 원칙은 다음과 같다.

(1) 순환 중첩 관계 대수의 경우와 같이 단일 연산자 의미가 여러 가지이거나 또는 복잡한 경우에 문제점이 발생하므로 연산자의 의미는 단순, 명료해야한다.

(2) 비순환 중첩 관계 대수의 재구성 연산자와 같이 입력 데이타 구조에 대해 제약을 가하는 경우에 발생할 수 있는 문제점을 방지하기 위하여 피연산자에 대한 제약이 없어야 한다.

(3) 기존 복합 객체 대수의 REPLACE 연산자와 같이 매개 변수에 재배열 기법을 적용할 수 없는 경우를 고려하여, 연산자들은 연산식 내에서 자유로운 구성이 가능해야 하며, 피연산자외의 매개 변수에도 연산식이 적용 가능해야 한다.

(4) 대수 연산자들은 반드시 소수 강한 연산자들의 모임 또는 다수 기본 연산자들의 모임일 필요는 없으며, 대수적 최적화 기법을 고려할 때 의미있는 연산자들의 모임으로 구성되어야 한다. 이 원칙은 기존 복합 객체 대수의 경우와 같이 소수의 강력한 연산자들로만 구성된 대수에서 발생할 문제점들을 고려한 것이다.

본 논문에서는 이러한 내용을 고려하여, 2장의 관련 연구에서 비교 기술한 대수 연산자의 제약 조건을 해결하고, 재배열 기법을 적용할 수 있는 복합 객체 대수를 설계한다.

3.2 복합 객체 대수

본 연구에서 설계한 대수는 크게 두가지 특성을 갖는다. 첫번째 특성은 다수 소트 대수(many-sorted algebra)에서와는 달리 타입에 해당되는 각 소트마다 연산자들이 정의된다. 이 특성은 실세계 데이타를 튜

플, 집합 등을 포함한 다양한 형태로 구조화하는 복합 객체 모델의 성질을 반영한다. 두번째 특성은 매개 변수로서 연산을 허용하는 고차 연산자와 관련된 것으로, 고차 연산자 표현시, 매개 변수 연산을 위한 언어를 따로 두지 않고 피연산자와 동일한 대수식을 사용한다. 이 특성은 모든 대수식이 단일 대수 체제를 기반으로 표현될 수있음을 의미한다.

본 연구에서 설계한 복합 객체 대수 연산자들은 2.1절에서 정의한 복합 객체를 대상으로 하며 연산 대상이 되는 데이타값의 타입에 따라 기본 연산자, 튜플 연산자, 집합 연산자로 나뉜다. 재구성 연산자도 집합값을 대상으로 하므로 집합 연산자 분류에 속한다. 이들 연산자 이외에 조건 연산자와 기존 관계 대수의 기능을 확장한 기타 연산자도 포함한다.

3.2.1 기본 연산자

숫자, 문자, 부울린의 기본값에 대한 기본 연산자는 사칙 연산자(+, −, *, /), 논리 연산자(AND, OR, NOT), 관계 연산자(=, <, <=, >, >=, <>)를 포함한다. 튜플값과 집합값에 대한 동등 비교 연산자는 (==)으로 나타내고, 집합값의 관련성은 원소 관계(∈)와 포함 관계(⊂)로 나타낸다.

3.2.2 튜플 연산자

튜플 연산자는 생성 연산자(TUP), 결합 연산자(CONCAT), 추출 연산자(EXTRACT)를 포함한다. 생성 연산자($TUP_a(v)$)는 애트리뷰트가 a이고 단일값이 v인 튜플을 생성한다. 결합 연산자(CONCAT(t_1, t_2))는 두 튜플 t_1과 t_2를 결합하여 새로운 단일 튜플 t_{12}를 생성한다. 추출 연산자($EXTRACT_a(t)$)는 튜플 t를 구성하는 애트리뷰트 a의 현재값을 반환한다.

3.2.3 집합 연산자

집합 연산자는 생성(SET), 집합(UNION, DIFFERENCE, PRODUCT), 재구성 연산자(COLLAPSE, APPLYTO, GROUP)로 구성된다. 집합 연산자 정의는 기존 관계 연산자에서와 동일하다. COLLAPSE(S)는 2단계 이상 중첩된 집합 S의 중첩 단계를 한단계 감소시킨 집합을 반환한다. $APPLYTO_E(S)$는 고차 연산으로서 집합 S의 각 원소에 연산식 E를 수행한 결과 집합을 반환한다. $GROUP_E(S)$는 집합 S의 각 원소에 연산식 E를 적용해서 결과가 동일한 원소로 구성된 집합을 반환하는 고차 연산이다. 예를 들어 2.2절에 기술된 예를 사용하여, 회사가 위치한 도시와 동일한 곳에 거주하는 사람들을 도시 이름으로 그룹화하기 위한 질의는 다음과 같다.

```
LiveWhereWorkPeople =
    select (x.ADDRESS = x.EMPLOYER.ADDRESS)
    (PERSON)
    group (x.ADDRESS)(LiveWhereWorkPeople)
```

결과 집합은 (CITY, PERSON)의 순서쌍으로서, 거주 도시와, 작업 도시가 동일한 사람들을 최소한 한명 이상 포함한다.

복합 객체 대수의 각 연산자의 연산 함수성은 (그림 2)에 나타나 있다. t, s, τ는 튜플 타입, 집합 타입, 복합 객체 타입을 각기 나타낸다. 이 중 집합 생성 연산자인 SET(v)은 값 v를 입력으로 받아 원소가 v인 집합을 생성한다.

사칙 연산자	:	num	×	num	→	num
논리 연산자	:	(bool)	×	bool	→	bool
관계 연산자	:	num	×	num	→	bool
		str	×	str	→	bool
동등 비교 연산자	:	t	×	t	→	bool
		s	×	s	→	bool
원소 관계	:	τ	×	s	→	bool
포함 관계	:	s	×	s	→	bool
튜플 생성 연산자	:	τ	→	t		
결합 연산자	:	t	×	t	→	t
추출 연산자	:	t	→	τ		
집합 생성 연산자	:	{ } : τ	→	SET(τ)		
COLLAPSE	:	{{τ}}	→	{τ}		
APPLYTO	:	{σ}	→	{τ}		
GROUP	:	{σ × τ}	→	{{σ × τ}}		
FILT	:	γ	→	γ | unit		

(그림 2) 복합 객체 대수 연산자의 함수성
(Fig. 2) The fuctionality of complex object algebra operators

3.2.4 조건 연산자

지금까지 정의된 대수 연산자들은 모두 함수 성질을 갖는다. 이러한 일관된 대수 체계를 유지하기 위해 본 연구에서는 조건식(predicate)도 단일 연산자로 취급한다. $FILT_p(v)$는 단일 값 v를 입력받아 매개 변수에 기술된 조건식을 평가한다. 평가 결과가 참이면 값 v를 반환하고 그렇지 않으면 널(NULL)값을 반환한다.

(정의 1) 조건식

(1) 임의의 대수 연산식 E_1, E_2에 대해 $E_1 \theta E_2$, $E_1 == E_2$, $E_1 \in E_2$, $E_1 \subset E_2$은 조건식이다. (θ는 비교 연산자)

(2) 임의의 두 조건식 P, Q에 대해 P AND Q, NOT P도 조건식이다. ▨

3.2.5 기타 연산자

기타 연산자들 중 일부 연산자는 위에서 정의된 연산자들로 표현 가능하고, 관계 대수에 포함된 연산자들 중 일부 연산자들의 기능을 한다. 그외 특별한 기능의 연산자들은 [5]를 값 주심의 복합 객체에 적합한 형태로 변형한 것으로서, 다음과 같이 정의된다.

(정의 2) 기타 연산자

(1) Intersection : $A \cap B = A - (A - B)$

(2) Selection : $\sigma_E(S) = APPLYTO_{FILTE(INPUT)}(S)$

(3) Product : REL_PROD(A, B) = $APPLYTO_{CONCAT(field1,\ field2)}\ (A \times B)$
이 때, $field_n \rightarrow EXTRACT_{field\ n}(INPUT)$

(4) Theta(Θ)join : $REL\_JOIN_{\Theta}(A, B) = APPLYTO_{FILT\Theta(INPUT)}(APPLYTO_{CONCAT(field1,\ field2)}(A \times B))$
이 때, Θ는 $f_1 \langle op \rangle f_2$ AND ⋯ AND $f_{n-1} \langle op \rangle f_n$ 이고, $\langle op \rangle$는 비교 연산자, f_i는 임의 대수 연산식이다.

(5) Tuple projection : $\Pi_{f1, \cdots, fn}(T) = CONCAT(TUP(f_1), CONCAT(\cdots, TUP(f_n)))$

(6) Projection : $REL\_PROJ_c(S) = APPLYTO_{\Pi c(input)}(S)$
이 때, c는 Tuple Projection의 $f_1, \cdots, f_n$이다.

(7) Constant : CONSTANT(v) = v

(8) Pairing : $PAIRWITH(Obj, S) = APPLYTO_{CONCAT(TUP1(obj),\ TUP2(INPUT))}(S)$ ▨

연산자 정의중 INPUT은 APPLYTO, GROUP, FILT의 고차 연산자의 매개 변수 연산식에 사용되는 바인딩(binding) 변수로서 피연산자 데이타 집합의 임의 원소를 나타낸다. INPUT이외에도 집합 내의 튜플값 원소에 사용된 애트리뷰트 이름이 매개 변수 연산식에 사용될 수있다.

4. 비교 및 평가

본 장은 3장에서 정의한 대수 연산자가 2장에서 고찰된 다른 대수들과 동등한 표현력을 갖으며, 더우기 본 논문에서 설계한 대수를 사용하면 대수적 최적화 기법의 적용이 용이함을 보인다.

4.1 대수의 표현력

일반적으로 대수 연산자들이 내포하는 연산은 데이타베이스에 저장된 데이타 집단을 임의의 다른 형태의 데이타 집단으로 사상시키는 함수이다. 대수 연산자를 함수로 간주하면 연산자들로 구성된 연산식은 함수들의 합성이다. 따라서 상이한 두 대수의 표현력을 보이기 위해서는 연산식을 구성하는 대수 연산자들과 동일한 함수성을 갖는 연산자(함수) 또는 연산식(합성 함수)이 비교 대수들에서 표현 가능함을 보이면 된다. 또한 임의의 두 함수 동일성은 동일 영역과 공변역을 갖고 공통의 영역 내 모든 원소에 대한 두 함수값이 같다는 조건을 만족해야한다.

본 연구에서 설계한 대수의 영역과 공변역은 2.1절의 복합 객체 모델의 복합 객체 타입으로부터 생성되는 데이타베이스이다. 따라서 중첩 관계 대수와 본 논문에서 정의한 복합 객체 대수는 각기 부분적으로 그리고 전체적으로 공통의 영역과 공변역을 갖는다.

[4, 5, 10, 11]의 복합 객체 대수와 중첩 관계 대수는 연산자들의 속성은 차이가 있지만 동등한 표현력을 갖는다. 이중 비교 대상으로서 [10]의 대수를 사용한다. 본 연구에서 설계한 대수가 [10]의 대수로 표현된 연산식과 동등한 의미로 표현할수 있음은 다음 연산자들의 항등 관계로 입증된다. 즉, [10]에서 기술된 대수 연산자의 함수성과 본 논문에서 제안된 대수 연산자들의 함수성이 동일함을 보임으로서 제안된 대수의 표현력이 최소한 [4, 5, 10, 11]과 동등함을 보인다.

(1)-(5) UNION, DIFFERNCE, CROSS-PRODUCT,

SELECT, PROJECT:
본 연구에서 설계한 동일 이름의 연산자들과 함수성이 같다.

(6) $NEST_{2\to 2*}(S) \equiv APPLYTO_E(PAIRWITH(S, REL\_PROJ_1(S)))$
함수성은 $\{\sigma\times\tau\}\to\{\sigma\times\{\tau\}\}$으로 동일하다.
$E = CONCAT(TUP1(EXT2(INPUT)), TUP_{2*}(REL\_PROJ_2(\sigma_P(EXT_1(INPUT)))))$
$P\to EXTRACT_1(INPUT(1)) = EXTRACT_2(INPUT)$

(7) $UNNEST_{2*}(S) \equiv COLLAPSE(APPLYTO_E(S))$
함수성은 $\{\sigma\times\{\tau\}\}\to\{\sigma\times\tau\}$으로 동일하다.
$E\to PAIRWITH(EXTRACT_1(INPUT), EXTRACT_2(INPUT))$

4.2 대수적 최적화 기법

이제부터 본 연구에서 설계한 대수가 다른 대수들에 비해 대수적 최적화 기법의 적용이 더 용이함을 보인다. 비순환 중첩 대수와 비교하기 위해 (그림 3)의 릴레이션을 가정한다. (그림 3)을 도메인으로 하면, 작성된 두 연산식 (1), (2)는 동일 결과 릴레이션 R4를 얻는다.

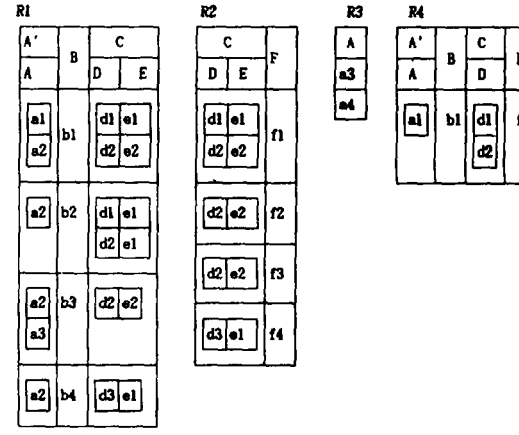
(1) $\nu_{A\to A'}((\mu_{A'}(\nu_{D\to C}(\Pi_{A'BDF}(\mu_C(R_1 \bowtie R_2))))) \bowtie R_3)$

(2) $\nu_{D\to C}(\Pi_{A',B,D,F}(\mu_C((\nu_{A\to A'}(\mu_{A'}(R_1 \bowtie R_3)) \bowtie R_2)))$

릴레이션 R1은 R2보다 R3과 일치되는 튜플수가 적으므로 R1과 R3을 먼저 죠인하는 연산식(2)가 더 효율적이다. 비순환 중첩 관계 대수에서 릴레이션의 가장 외부 값에 대한 죠인 연산은 결합 법칙이 성립한다. 그러나 내포된 릴레이션값에 대한 죠인 연산은 NEST 연산자(ν)를 포함해야하는 데, 죠인 연산자($\bowtie$)는 NEST연산자와 교환이 성립되지 않으므로 연산식(1)로부터 (2)가 유도될 수 없다. 반면, 본 논문에서 제안한 대수에서는 NEST연산자의 도움없이 내포된 릴레이션 값에 대한 죠인 연산을 수행할 수 있으므로 위 문제가 발생하지 않는다. 위 두 연산식과 동등한 연산식은 (3), (4)이다.

(3) $APPLYTO_{\Pi A',B,C,D,F(INPUT)}(REL\_JOIN_{\Theta 1}(R_1, R_2), R_3))$
이때, $\Theta 1\to R_1.C = R_2.C$ $\Theta 2\to T_1.A'.A = R_3.A$
$T_1\to REL\_JOIN_{\Theta 1}(R_1, R_2)$
$R_1.C\to EXTRACT_{R1.C}(INPUT)$
$R_2.C\to EXTRACT_{R2.C}(INPUT)$
$T_1.A'.A\to EXTRACT_A(T1.A')$
$R_3.A\to EXTRACT_{R3.A}(INPUT)$
또한, $\Pi_{A',B,C,D,F}$
$A'\to EXTRACT_{A'}(INPUT)$
$B\to EXTRACT_B(INPUT)$
$C.D\to EXTRACT_D(C)$
$F\to EXTRACT_F(INPUT)$

(4) $APPLYTO_{\Pi A',B,C,D,F(INPUT)}(REL\_JOIN_{\Theta 2}(REL\_JOIN_{\Theta 1}(R_1, R_3), R_2))$
이때, $\Theta_1\to R_1.A'.A = R_3.A$ $\Theta_2\to T_1.C = R_2.C$
$T_1\to REL\_JOIN_{\Theta 1}(R_1, R_3)$
$R_1.A'.A\to EXTRACT_A(R_1.A')$
$R_3.A\to EXTRACT_{R3.A}(INPUT)$
$T_1.C\to EXTRAC_{T1.C}(INPUT)$
$R_2.A\to EXTRACT_{R2.C}(INPUT)$



(그림 3) 예제 릴레이션
(Fig 3) Example relations

연산식(1), (2)와 동일한 의미를 나타내는 순환 중첩 관계 대수로 작성된 연산식은 (5), (6)이다.

(5) $\bowtie(\Pi((A', B, C(D), F)(\bowtie(R1, R2)))(A'), R3)$

(6) $\Pi((A', B, C(D), F)(\bowtie(\bowtie(R1(A'), R3), R2)))$

연산식 (6)은 (5)로 부터 용이하게 유도될 수 있는데, 이 때 적용될 규칙을 찾기 위해서는 연산식 내에 포함된 연산자 이외에도 릴레이션 스킴 구조를 파악하는 부가적인 작업이 요구된다. 순환 중첩 관계 대

수의 연산식에서 성립하는 재배열 규칙은 연산자 이외에도 릴레이션 스킴 구조에 따라 적용될 규칙이 상이하기 때문이다. 그러나 본 연구에서 설계한 대수로 구성된 연산식 (3)으로 부터 연산식 (4)를 유도하는 데는 이러한 부가적인 작업이 요구되지 않는다. 즉, 본 논문에서 제안된 대수로 구성된 연산식에 대한 재배열 규칙은 모든 릴레이션 스킴에 대해 일관된 방법으로 적용된다.

한편 (그림 4)의 데이타베이스를 가정한다.

Student
{[Name:str Dept:[Name:str Floor:num] Addr:Str]}
Department
{[Name:str Floor:num Employees:{[Name:str Salary:num]}]}
Employee
{[Name:str Dept:[Name:str Floor:num] Salary:num Pos:str]}

(그림 4) 예제 데이타베이스 : UNIVERSITY
(Fig. 4) Example database : UNIVERSITY

연산식(7)은 "5층에 위치한 학과의 학생 이름을 모두 구하라."의 질의를 [4]의 복합 객체 대수로 작성한 예이다. 연산식(8)은 본 논문에서 설계한 대수로 기술한 연산식이다.

(7) $\mathrm{REPLACE}_{(\mathrm{NAME})}(\mathrm{REPLACE}_{(\mathrm{IF\ FLOOR}=5\ \mathrm{THEN\ 5})}(\mathrm{Student}))$

연산식(7)는 더이상 변환이 가능하지 않지만, 연산식(6)은 매개 변수로서 대수 연산식을 사용하면 다음의 단순한 연산식(9)로 변환 가능하다. 이는 매개 변수 표현과 연산식 표현이 상이한 비순환 중첩 대수나 순환 중첩 대수에서는 얻을 수없는 효과이다.

(8) $\mathrm{APPLYTO}_{\Pi\ \mathrm{NAME(INPUT)}}(\mathrm{APPLYTO}_{\mathrm{FILT}\Pi\ \mathrm{FLOOR(DEPT)}=5\ (\mathrm{DEPT})}(\mathrm{Student}))$

(9) $\mathrm{APPLYTO}_{\Pi\ \mathrm{NAME}}(\mathrm{FILT}_{\Pi\ \mathrm{FLOOR(DEPT)}=5(\mathrm{DEPT})}(\mathrm{Student})$

위 연산식(3), (4), (9)로 부터 본 연구에서 설계한 대수가 기존의 대수들에 비해 대수적 최적화 기법의 적용이 더 용이함을 알수 있다.

5. 결 론

본 논문에서는 복합 객체 모델의 검색 연산을 대상으로 대수적 최적화 기법의 적용이 용이한 복합 객체 대수를 설계하였다. 단순 명료성, 데이타 구조에 대한 무제약성, 단일 표현 체계의 설계 원칙을 기반으로 설계된 대수는 정형화된 데이타에 대한 기본 연산자와, 튜플 집합 연산자, 조건식을 단일 연산자로 취급하는 조건 연산자, 복합 객체를 위한 확장된 관계 대수 연산자인 기타 연산자로 구성된다. 본 논문에서 제안된 대수를 비순환 중첩 대수와 비교하면, NEST 연산자를 사용하지 않고 내포된 릴레이션 값에 대한 조인 연산을 표현할 수 있으므로 단순한 방법으로 질의 연산식을 기술할 수 있다. 또한 순환 중첩 관계 대수와 비교하면, 질의 최적화를 위한 재배열 규칙을 연산자와 피연산자인 릴레이션 스킴 구조에 따라 적용 방법이 다르다. 그러나 본 논문에서 제안된 대수는 모든 릴레이션 스킴에 대해 일관된 방법으로 재배열 규칙이 적용된다. 또한 본 논문에서 제안된 대수가 기존 대수들과 비교할 때, 동등한 표현력을 갖지만 대수적 최적화 기법의 적용이 더 용이함을 보였다.

앞으로는 질의 최적화 알고리즘을 개발하기 위해, 본 논문에서 제안된 대수의 연산식 사이에 성립하는 상위 단계 최적화를 위한 항등 변환 규칙을 제안한다. 항등 변환 규칙은 관계 대수에서와 유사하지만 매개 변수로 연산식을 취하고자하는 고차 연산자들에 대한 변환 규칙을 포함하여야 한다. 연구의 연계성을 위해 관계 대수에서의 항등 변환 규칙들과 유사한 형태를 갖는 항등 변환 규칙을 제안한다. 한편, 비용 모델에 기반을 둔 효율적인 최적화 기법에 대한 연구가 요구된다. 데이타베이스 연산의 구현 기법과 실제 수행될 질의 실행 계획을 선택하는 하위 단계 최적화는 튜플수, 인덱스값 분포등의 통계 정보에 의존하므로 효율성이 보장된 비용 모델에 기반을 둔 최적화 기법의 연구가 요구된다.

참 고 문 헌

[1] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," Comm. ACM, Vol. 13, No.6, pp.377-387, 1970.

[2] Kim, W., et al., "Operations and Implementation of Complex Objects," IEEE Trans. on Software Engineering, Vol.14, No.7, pp.958-996, 1988.

[3] Vandenberg, S. L., "Algebras for Object-Oriented

Query Languages," Ph. d. Thesis, Dept. of Computer Science, Univ. of Wisconsin, 1993.

[4] Abiteboul, S., et al. (Eds.), Nested Relations and Complex Objects in Databases, LNCS 361, Springer-Verlag, 1990.

[5] Vandenberg, S.L., "Practical Complex Object Algebras," Proc. Int'l. Workshop on Database Query Optimization, pp.101-108, 1989.

[6] Schek, H. J., et al., "The Relational Model with Relation-Valued Attributes," Information Systems, Vol.11, No.2, pp.137-147, 1986.

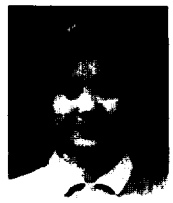
[7] Beeri, C., et al., "Algebraic Optimization of Object-Oriented Query Languages," Proc. Int'l Conf. on Database Theory, pp.72-88, 1990.

[8] Ohori, A., et al., "Database Programming in Machiavelli-a polymorphic Language with Static Type Inference," Proc. ACM SIGMOD, pp. 46-57,1989.

[9] Bancilhon, F., et al., "FAD, a powerful and Simple Database Language," Proc. 13th VLDB, pp. 97-105, 1987.

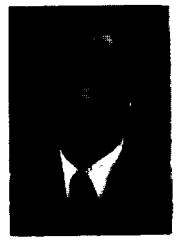
[10] Thomas, S. J., et al., "Nested Relational Structures," Advanced in Computing Research Ⅲ:The Theory of Databases, JAI press, pp.269-307, 1986.

[11] Colby, L. S., "A Recursive Algebra and Query Optimization for Nested Relations," Proc. ACM SIGMOD, pp.277-283, 1989.



송 지 영

1982년 인하대학교 전자 계산학과 졸업(학사)
1987년 인하대학교 대학원 전자계산학과(이학석사)
1993년 인하대학교 대학원 수학과 졸업(전자 계산학 이학박사)
1994년~1995년 인하대학교 대학원 연구원
1992년~1994년 인하대학교 전자 계산학과 대우 전임강사
1993년~현재 인하대학교 전자 계산공학과 강사
한국 정보 처리 학회 정회원
관심분야:복합 객체 데이타베이스 설계, 질의 최적화



배 해 영

인하대학교 공과대학을 졸업하고 연세대학교 공학석사, 숭실대학교에서 전자계산학으로 공학박사(1989) 학위를 취득.
미국 휴스턴(Houston) 대학의 객원교수, 인하대학교 전자계산소 부장, 소장을 역임하고 현재 인하대학교 전자계산학과에 교수로 재직중. 관심분야로는 멀티미디어 데이터베이스 시스템, 실시간 데이터베이스 시스템, 지리정보 시스템