효율적인 질의 처리를 위한 SQL3 질의의 정규화

권혁윤O 이기훈 황규영
한국과학기술원 전산학과/첨단정보기술연구센터
{hykwonO, khlee, kywhang}@mozart.kaist.ac.kr

Normalization of SQL3 Queries for Efficient Query Processing

Hyuk-Yoon KwonO Ki-Hoon Lee Kyu-Young Whang
Department of Computer Science &
Advanced Information Technology Research Center
Korea Advanced Institute of Science and Technology

요 약

SQL은 관계형 DBMS에서 사용되는 표준 질의 언어이다. SQL의 장점 중의 하나는 중첩 질의의 사용이나, 중첩 질의를 포함한 질의를 그대로 실행하는 것은 중첩 질의의 반복된 수행을 야기하여 비효율적이다. 본 논문에서는 SQL3 표준에 정의된 모든 유형의 중첩 질의에 대한 완전한 정규화 규칙을 제안한다. SQL3 표준에서 중첩 질의는 중첩 질의 반환 값의 유형에 따라 스칼라 중첩 질의와 테이블 중첩 질의로 분류된다. 스칼라 중첩 질의와 테이블 중첩 질의는 상관과 집계의 유무에 따라 다시 분류될 수 있다. 본 논문에서는 SELECT, FROM, WHERE 절에서 이러한 분류에 의해 가능한 모든 중첩 유형을 지원한다. 특히, SELECT, FROM 절의 일부 중첩 유형은 SQL3와 유사한 형태의 중첩 질의를 지원하는 질의 언어인 XQuery에서 제안된 정규화 규칙을 SQL3 문법에 맞게 응용하여 적용한다.

1. 서론

SQL은 관계형 DBMS(relational database management system)에서 널리 사용되는 표준 질의 언어이다. SQL의 장점 중의 하나는 질의의 중첩이 가능하다는 것이다. SQL 중첩 질의(nested query)는 다른 질의 안에 중첩되어 있는 SELECT-FROM-WHERE 표현식(expression)이다[9].

중첩 질의를 포함한 질의를 그대로 실행하면 중첩 질의가 반복해서 실행되기 때문에 비효율적이다. 따라서, 중첩된 SQL 질의를 중첩되지 않은 질의로 정규화(normalization)하는 연구들[1][2][5][8]이 진행되어 왔다.

SQL 질의의 정규화에 대한 많은 연구들이 이루어 졌지만, 기존에 제안된 정규화 규칙들은 완전하지 않다. 최근의 SQL 표준인 SQL3 표준[3]에 따르면 SELECT, FROM, WHERE 절 모두에 중첩 질의가 가능한데, 기존의 정규화 규칙들은 WHERE 절의 중첩 질의에 대해서만 제안되었다. 더욱이, WHERE 절의 중첩 질의에 대해 제안된 규칙들도 WHERE 절에 가능한 모든 유형의 중첩 질의를 처리할 수 없다.

한편, SQL3와 유사한 형태의 중첩 질의를 지원하는 질의 언어로 XML 데이터에 대한 표준 질의 언어인 XQuery[12]가 있다. SQL3에서와 같이 중첩 질의를 포함한 XQuery 질의를 그대로 실행하는 것은 비효율적이기 때문에 중첩된 XQuery 질의를 중첩되지 않은 질의로 정규화하는 많은 연구들[6][7]이 진행되어 왔다. 참고 문헌[6]에서는 XQuery의 기본 구조인 FOR, WHERE, RETURN절에 대한 완전한 정규화 규칙을 제안하였다. XQuery는 SQL3와 질의 언어의 의미가 다르기 때문에 XQuery의 정규화 규칙을 SQL3에 그대로 적용할 수는 없다. 그러나, 몇 가지 중첩 유형에 대해서는 적용이 가능한데, 자세한 설명은 제 2장과 제 3장에서 한다.

본 논문에서는 SQL3 표준[5]에 정의된 모든 유형의 중첩 질의에 대한 완전한 정규화 규칙들을 제안한다. SQL3 표준에서 중첩 질의는 중첩 질의 반환 값의 유형에 따라 스칼라 중첩 질의와 테이블 중첩 질의로 분류된다. 그리고, 스칼라 중첩 질의와 테이블 중첩 질의는 참고 문헌[5]의 분류에 따라 다시 Type-A, Type-N, Type-J, Type-JA, 그리고 Type-D 유형으로 분류될 수 있다. 본 논문에서는 이러한 분류에 따라 SELECT, FROM, WHERE 절에서 가능한 모든 중첩 유형에 대한 정규화 규칙들을 제안하며 제안한 모든 정규화 규칙들을 오디세우스 객체 관계형 DBMS[10]에 구현하였다.

* 본 연구는 첨단정보기술연구센터를 통하여 과학기술부/한국과학재단의 지원을 받았음.

본 논문의 구성은 다음과 같다. 제 2장에서는 관련 연구로서 SQL3 중첩 질의 표준 명세를 설명하고, 기존의 SQL 정규화 방법과 XQuery 정규화 방법에 대해 설명한다. 제 3장에서는 SQL3 질의의 정규화 규칙에 대해 설명한다. 마지막으로 제 4장에서는 결론을 내린다.

2. 관련 연구

본 장에서는 관련 연구로서 SQL3 중첩 질의 표준 명세, SQL 질의의 정규화, 그리고 XQuery 질의의 정규화에 대해서 설명한다. 제 2.1절에서는 SQL3 중첩 질의 표준 명세에 대해 설명하고, 제 2.2절에서는 SQL 질의의 정규화에 대해 설명한다. 제 2.3절에서는 XQuery 질의 의 정규화에 대해 설명한다.

2.1. SQL3 중첩 질의 표준 명세

본 절에서는 SQL3 표준[3]에 정의된 중첩 질의 명세를 요약한다. SQL3 중첩 질의는 다른 질의 안에 중첩되어 있는 SELECT-FROM-WHERE 표현식(expression)이다[9]. 이 때 중첩 질의를 포함하는 질의는 외부 질의(outer query)라고 한다. 중첩 질의는 스칼라 중첩 질의와 테이블 중첩 질의로 분류된다[3]. 스칼라 중첩 질의는 결과로 스칼라 값(scalar value)을 반환하며, 테이블 중첩 질의는 결과로 테이블을 반환한다.

SELECT, FROM, WHERE 절에 표현 가능한 중첩 질의를 표로 정리하면 그림 2.1과 같다. 먼저 SELECT 절에는 스칼라 중첩 질의만 표현 가능하다. SELECT 절에 테이블 중첩 질의를 표현할 수 없는 이유는, SELECT 절에 테이블 중첩 질의가 사용되면 질의 결과가 관계형 데이터 모델에서 릴레이션(relation)의 정의를 만족시키지 않기 때문이다. 즉, SELECT 절에 사용된 중첩 질의의 결과로 테이블이 반환되면 결과로 보여지는 애트리뷰트(attribute)의 값이 테이블이 되는데 이것은 릴레이션의 모든 애트리뷰트는 원자 값을 갖는다는 릴레이션의 정의에 위배된다. FROM 절과 WHERE 절에는 이와 같은 문제가 발생하지 않으므로 스칼라 중첩 질의와 테이블 중첩 질의가 모두 표현 가능하다.

절 종류 \ 중첩 질의 종류	스칼라 중첩 질의	테이블 중첩 질의
SELECT 절	O	X
FROM 절	O	O
WHERE 절	O	O

그림 2.1. SELECT, FROM, WHERE 절에 표현 가능한 중첩 질의.

SELECT 절의 스칼라 중첩 질의는 결과로 보여질 애트리뷰트로 사용된다. FROM 절의 스칼라 중첩 질의는 질의 대상이 되는 스칼라 값으로 사용되고, 테이블 중첩 질의는 질의 대상이 되는 테이블로 사용된다.

WHERE 절의 스칼라 중첩 질의는 스칼라 값을 갖는 피 연산자로 사용된다. SQL3 표준[3]에서 두 개의 피 연산자를 갖는 모든 연산자는 왼쪽 피 연산자로 항상 스칼라 값을 갖는다. 그 중 스칼라 비교 연산자(scalar comparison operator)는 오른쪽 피 연산자로도 반드시 스칼라 값을 갖는다. SQL3 표준에 정의된 스칼라 비교 연산자에는 =, <>, <, >, <=, >=가 있다. 또한, IS NULL/IS NOT NULL 연산자도 피 연산자로 스칼라 값을 갖는다.

WHERE 절의 테이블 중첩 질의는 집합 테스트 연산자(set test operator)의 피 연산자로 사용된다. SQL3 표준[3]에 정의된 집합 테스트 연산자는 EXISTS/NOT EXISTS, UNIQUE/NOT UNIQUE, IN/NOT IN, ALL, SOME, ANY이다. 여기서 SOME과 ANY는 동일한 의미를 갖는 같은 연산자이므로 본 논문에서는 SOME 연산자만을 다룬다.

2.2. SQL 질의의 정규화 규칙

참고 문헌[5]에서는 중첩 질의를 상관과 집계 유무에 따라 Type-A, Type-N, Type-J, Type-JA, 그리고 Type-D 유형으로 분류하고 각 유형에 대한 정규화 규칙을 제안하였다. 상관은 외부 질의의 릴레이션을 참조하는 조건식이 중첩 질의에 있는 것을 의미하고, 집계는 중첩 질의의 SELECT 절에 집계 함수가 사용된 것을 의미한다.

Type-A 중첩 질의는 상관은 없고, 집계만 있는 질의로서, 중첩 질의를 먼저 실행한 다음 그 결과를 중첩 질의와 치환하여 정규화한다. Type-N 중첩 질의는 상관과 집계가 모두 없는 질의이고, Type-J 중첩 질의는 상관은 있으나 집계는 없는 질의로서, 외부 질의의 릴레이션과 중첩 질의의 릴레이션을 조인하여 중첩을 없앤다. 한편, 참고 문헌[2]에서는 EXISTS/NOT EXISTS 연산자가 사용된 Type-N/Type-J 중첩 질의를 Type-A/Type-JA 유형의 WHERE 절 스칼라 중첩 질의로 변환하는 변환 규칙[1]을 제안하였다.

Type-JA 중첩 질의는 상관과 집계가 모두 있는 질의로서, 중첩 질의에서 조인에 참여하는 애트리뷰트들을 기준으로 그룹화하여 임시 릴레이션 T를 생성하고, T와 외부 질의의 릴레이션을 조인하여 결과를 얻는다. 그러나 [5]에서 제안된 정규화 규칙을 적용하면 COUNT bug[4]가 발생할 수 있다. COUNT bug를 해결하기 위해 많은 연구가 이루어졌으며, 그 중에서 대표적인 것이 Magic Decorrelation[8]이다. Magic Decorrelation은 외부 조인(outer join)을 이용하여 COUNT bug를 해결한다.

Type-D 중첩 질의는 2개의 중첩 질의를 피 연산자로 갖는 연산자가 WHERE 절에 있는 것으로, 적어도 하나의 중첩 질의는 외부 질의와 상관이 있어야 한다. 참고 문헌[5]에서는 Type-D 중첩 질의를 두 중첩 질의 결과 간의 division 연산으로 해석하여 정규화하고 있다. 그러나 SQL3 표준[3]에 정의된 Type-D 중첩 질의는 division 연산을 의미하는 것이 아니라 중첩 질의와 함께 사용된 연산자에 따라 각기 다른 의미를 갖는다.

2.3. XQuery 질의의 정규화

본 절에서는 SQL3와 유사한 형태의 중첩 질의를 지원하는 XQuery 질의 언어에서 사용되는 정규화 규칙에 대해 설명한다. XQuery는 XML 데이터에 대한 표준 질의 언어로서, SQL3에서의 SELECT-FROM-WHERE 표현식과 유사한 FLWR 표현식을 지원한다. FLWR 표현식은 FOR, LET, WHERE, RETURN절로 구성되며, 각 절은 중첩 질의를 허용한다. SQL3에서와 마찬가지로 중첩된 XQuery 질의를 그대로 실행하는 것은 비효율적이므로, 중첩된 XQuery 질의를 중첩되지 않은 XQuery 질의로 변환하는 정규화 규칙들[6][7]이 제안되었다.

[1] 변환 규칙은 중첩 질의를 정규화 규칙이 적용 가능한 형태의 중첩 질의로 변환한다.

참고 문헌[6]에서는 XQuery의 기본 구조인 FOR, WHERE, RETURN절에 대한 완전한 정규화 규칙을 제안하였다. 참고 문헌[6]에서는 각 절의 중첩 유형을 상관과 집계 유무에 따라 Type-A, Type-N, Type-J, Type-JA, 그리고 Type-D 유형으로 분류하고, 가능한 모든 중첩 유형에 대한 정규화 규칙을 제안하였다.

XQuery의 FOR, WHERE, RETURN절은 SQL3의 FROM, WHERE, SELECT 절에 대응되기 때문에 XQuery에 제안된 정규화 규칙을 SQL3에 그대로 적용하는 것이 가능해 보인다. 그러나, SQL3와 XQuery는 질의 언어가 가지는 의미가 다르기 때문에 XQuery 정규화 규칙을 SQL3에 그대로 적용할 수는 없다. 먼저, SQL3는 관계형 데이터 모델을 따르기 때문에 중첩 질의가 나타나는 위치에 따라 중첩 질의가 반환할 수 있는 값의 유형에 제약이 있다. 예를 들어, SELECT 절에는 제 2.1절에서 설명한 바와 같이 스칼라 중첩 질의만 가능하다. 반면, XQuery는 XML 데이터 모델을 따르기 때문에 위와 같은 제약이 없다. 즉, 중첩 질의의 표현이 가능한 모든 곳에 스칼라 중첩 질의와 테이블 중첩 질의 모두를 허용한다.

이와 같은 의미 차이로 인해, XQuery의 정규화 규칙을 SQL3 중첩 질의에 그대로 적용할 수는 없지만, 중첩 질의 반환 값이 스칼라로 한정되는 유형에 대한 정규화 규칙은 SQL3 중첩 질의에 적용이 가능하다. 이러한 정규화 규칙들에 대한 자세한 설명은 제 3장에서 한다.

3. SQL3 질의의 정규화 규칙

본 장에서는 SQL3 질의의 정규화 규칙에 대해서 설명한다. 제 3.1절에서는 SQL3 정규화 규칙의 완전성에 대해 설명하고, 제 3.2절에서는 WHERE 절 정규화 규칙에 대해 설명한다. 제 3.3절에서는 SELECT 절 정규화 규칙에 대해 설명하고, 제 3.4절에서는 FROM 절 정규화 규칙에 대해 설명한다.

3.1. SQL3 정규화 규칙의 완전성

SQL3 질의의 기본 구조는 SELECT, FROM, WHERE 절로 구성되며, 모든 절에 중첩 질의가 가능하다. 본 논문에서는 SELECT, FROM, WHERE 절 중첩 질의를 중첩 질의 반환 값의 유형에 따라 스칼라 중첩 질의와 테이블 중첩 질의로 분류한다[3]. 또한, 각 절의 스칼라 중첩 질의와 테이블 중첩 질의는 상관과 집계 유무에 따라 다시 Type-A, Type-N, Type-J, Type-JA, 그리고 Type-D 유형의 중첩 질의로 분류한다[5].

그림 3.1은 위의 분류에 의해 가능한 중첩 질의의 유형을 표로 나타낸 것이다. SELECT 절과 FROM 절의 Type-D 중첩 질의는 정의에 의해 가능하지 않다. 또한, SELECT 절의 테이블 중첩 질의, FROM 절과 WHERE 절의 테이블 중첩 질의에서 Type-A, Type-JA 유형은 SQL3 표준[3]에 정의되지 않은 중첩 질의 유형이다.

절 / 중첩 질의 반환 값 / 중첩 유형	SELECT 절		FROM 절		WHERE 절	
	스칼라	테이블	스칼라	테이블	스칼라	테이블
Type-A	● (NR4)	N/A	● (NR8)	N/A	○	N/A
Type-N	● (NR5)	N/A	● (NR9)	● (NR9)	● (NR1)	○ ● (NR3, TR1~3)
Type-J	● (NR6)	N/A	● (NR9)	● (NR9)	● (NR2)	○ ● (NR3, TR1~3)
Type-JA	● (NR7)	N/A	● (NR10)	N/A	○	N/A
Type-D	N/A	N/A	N/A	N/A	● (NR1~3, TR1~3)	● (NR1~3, TR1~3)

○: 기존 연구 [2][5][8] 에서 제안한 정규화 규칙
●: 기존 연구 [5][6][8] 에서 제안된 규칙을 응용한 정규화 규칙
●: 본 논문에서 새롭게 제안한 정규화 규칙

그림 3.1. SQL3 질의에서 가능한 중첩 유형의 분류.

그림 3.1은 본 논문에서 제안한 정규화 규칙의 완전성을 보여준다. 즉, 본 논문에서는 SQL3 SELECT-FROM-WHERE 표현식의 모든 절에서 가능한 모든 중첩 유형에 대한 정규화를 지원한다. 흰 원은 기존 연구[2][5][8]에서 제안된 정규화 규칙을 의미하고, 회색

원은 기존에 제안된 규칙[5][6][8]을 해당 유형에 맞게 응용한 정규화 규칙이다. 그리고, 검은 원은 본 논문에서 새롭게 제안한 정규화 규칙을 의미한다. 원 옆에는 본 논문에서 제안한 규칙의 정규화 규칙 번호와 변환 규칙 번호를 나열하였다.

3.2. WHERE 절 정규화 규칙

WHERE 절 중첩 질의에 대한 정규화 규칙은 중첩 질의 반환 값의 유형에 따라 스칼라 중첩 질의와 테이블 중첩 질의로 분류하여 제안한다. 먼저 스칼라 중첩 질의를 정규화하는 규칙을 제안하고, 다음으로 테이블 중첩 질의를 정규화하는 규칙을 제안한 후, Type-D 중첩 질의를 정규화하는 규칙은 별도로 제안한다.

3.2.1. 스칼라 중첩 질의를 정규화하는 규칙

WHERE 절의 스칼라 중첩 질의는 한 연산자의 피 연산자로 하나의 중첩 질의가 사용된 경우와 두 개의 중첩 질의가 사용된 경우로 분류된다. 중첩 질의가 하나만 사용된 경우는 상관과 집계 유무에 따라 Type-A, Type-N, Type-J, Type-JA 중첩 질의로 분류되고, 두 개가 사용된 경우는 Type-D 중첩 질의로 분류된다. 이 중 본 논문에서는 Type-N, Type-J, 그리고 Type-D 유형에 대한 정규화 규칙을 새롭게 제안한다. Type-N, Type-J에 대한 정규화 규칙은 본 절에서 설명하고, Type-D 유형에 대한 정규화 규칙은 제 3.2.3 절에서 설명한다.

Type-N 중첩 질의를 정규화하는 규칙

기존 연구[5]에서 제안한 NEST-N-J 정규화 규칙은 중첩 질의의 결과가 스칼라 값인지 확인하지 않기 때문에 스칼라 중첩 질의에 그대로 적용할 수 없다. 따라서, 본 논문에서는 WHERE 절 스칼라 중첩 질의의 Type-N 유형에 대한 새로운 정규화 규칙을 제안한다. WHERE 절 스칼라 중첩 질의의 Type-N 유형에 대한 정규화 규칙은 그림 3.2(a)와 같다. 정규화 규칙 NR1은 중첩 질의를 먼저 실행한 다음 그 결과가 스칼라 값이면 중첩 질의와 치환하여 정규화하고, 그렇지 않으면 에러를 반환한다.

Type-J 중첩 질의를 정규화하는 규칙

Type-J 중첩 질의도 Type-N 중첩 질의와 같은 이유로 기존 연구[5]에서 제안한 NEST-N-J 정규화 규칙을 적용할 수 없다. 따라서, 본 논문에서는 Type-J 중첩 질의에 대한 새로운 정규화 규칙을 제안한다. WHERE 절 스칼라 중첩 질의의 Type-J 유형에 대한 정규화 규칙은 그림 3.2(b)와 같다. 정규화 규칙 NR2는 중첩 질의에서 조인에 참여하는 애트리뷰트 별로 조인 조건을 만족하는 결과의 개수를 미리 구한다. 결과의 개수가 모두 1이하이면, NEST-N-J 정규화 규칙으로 정규화하고, 그렇지 않으면 에러를 반환한다.

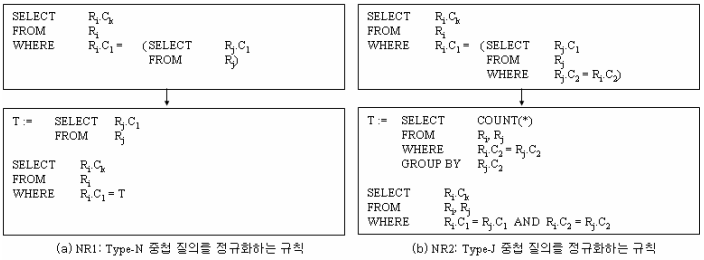

그림 3.2. WHERE 절 스칼라 중첩 질의의 Type-N/Type-J 정규화 규칙.

3.2.2. 테이블 중첩 질의를 정규화하는 규칙

테이블 중첩 질의가 가능한 연산자에는 EXISTS/NOT EXISTS, IN/NOT IN, ALL, SOME, UNIQUE/NOT UNIQUE가 있는데, 본 논문에서는 EXISTS/NOT EXISTS, IN/NOT IN, ALL, SOME에 대한 정규화 규칙을 제안한다. UNIQUE/NOT UNIQUE 연산자는 상용 DBMS에서 지원하지 않기 때문에 본 논문의 정규화 범위에서 제외한다. EXISTS/NOT EXISTS 중첩 질의는 기존 연구[2]에서 제안한 변환 규칙을 사용해 Type-A/Type-JA 유형으로 변환한 뒤 기존 연구[5]에서 제안한 Type-A/Type-JA 정규화 규칙으로 정규화하고, IN/NOT IN, ALL, SOME에 대한 정규화 규칙은 본 논문에서 새롭게 제안한다. 한편, 두 개의 피 연산자를 갖는 IN, ALL, SOME 연산자에 한해서는 Type-D 유형도 가능한데, 이는 제 3.2.3절에서 설명한다.

IN/NOT IN 중첩 질의를 정규화하는 규칙

기존 연구[5]에서 제안한 IN/NOT IN 중첩 질의 정규화 규칙은 조인 조건을 만족하는 튜플의 개수만큼 결과를 중복하여 보여주기 때문에 그대로 적용할 수 없다. 따라서, 본 논문에서는 IN/NOT IN 중첩 질의를 정규화 하는 규칙을 새롭게 제안한다. IN/NOT IN 중첩 질의의 변환 규칙은 그림 3.4와 같다. TR1은 IN/NOT IN 중첩 질의를 동일한 의미를 갖는 EXISTS/NOT EXISTS 중첩 질의로 변환한다. NOT IN 연산자를 사용한 중첩 질의는 같은 방식으로 NOT EXISTS 연산자를 사용한 중첩 질의로 변환할 수 있다. 이렇게 변환된 중첩 질의는 EXISTS/ NOT EXISTS 중첩 질의의 정규화 규칙[2][5]으로 정규화한다.

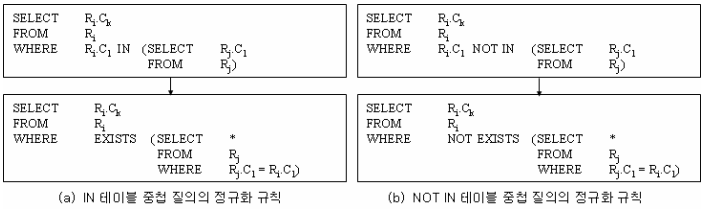

그림 3.4. TR1: WHERE 절 IN/NOT IN 테이블 중첩 질의의 정규화 규칙.

ALL 중첩 질의를 정규화하는 규칙

ALL 중첩 질의 정규화 규칙은 기존 연구[11]에서 제안한 2차원 데이터베이스 질의에 대한 규칙을 SQL3 질의에 맞게 확장하여 본 논문에서 새롭게 제안한다. ALL 연산자가 사용된 중첩 질의의 정규화 규칙은 그림 3.5(a)와 같다. 변환 규칙 TR2는 ALL 연산자가 사용된 중첩 질의를 NOT EXISTS 연산자가 사용된 중첩 질의로 변환한다. 이렇게 변환된 중첩 질의는 NOT EXISTS 중첩 질의의 정규화 규칙[2][5]으로 정규화한다.

SOME 중첩 질의를 정규화하는 규칙

SOME 중첩 질의 정규화 규칙은 기존에 제안된 적이 없기 때문에 본 논문에서 새롭게 제안한다. SOME 연산자가 사용된 중첩 질의에 대한 변환 규칙은 그림 3.5(b)와 같다. 변환 규칙 TR3은 SOME 연산자가 사용된 중첩 질의를 동일한 의미를 갖는 EXISTS 연산자가 사용된 중첩 질의로 변환한다. 변환 규칙 TR3에 따라 변환된 질의는 EXISTS 중첩 질의의 정규화 규칙[2][5]으로 정규화한다.

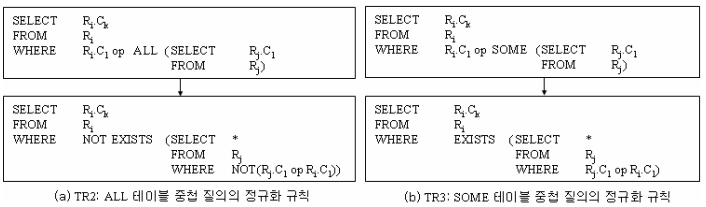

그림 3.5. WHERE 절 ALL/SOME 테이블 중첩 질의의 정규화 규칙.

3.2.3. Type-D 중첩 질의를 정규화하는 규칙

기존 연구[5]에서 해석한 Type-D 유형의 의미는 SQL3 표준[3]에서 정의한 Type-D 유형의 의미와 다르기 때문에 제안된 정규화 규칙을 그대로 적용할 수 없다. 따라서, 본 논문에서는 Type-D 유형에 대한 정규화 규칙을 새롭게 제안한다. Type-D 중첩 질의는 2개의 중첩 질의가 합쳐진 것으로 볼 수 있기 때문에 제 3.2.1절과 제 3.2.2 절에서 소개한 정규화 규칙을 두 번 연속으로 적용하여 정규화한다. 즉, 왼쪽 연산자에 해당하는 중첩 질의에 대한 정규화 규칙을 적용하고, 오른쪽 연산자에 해당하는 중첩 질의에 대한 정규화 규칙을 적용한다.

3.3. SELECT 절 정규화 규칙

SELECT 절 중첩 질의의 정규화 규칙은 기존에 제안된 적이 없기 때문에 본 논문에서 새롭게 제안한다. 먼저, Type-A나 Type-JA유형의 정규화 규칙은 그림 3.6(a)나 그림 3.6(b)와 같이 XQuery의 RETURN 절에 제안된 Type-A나 Type-JA 정규화 규칙[6]을 SQL3의 SELECT 절에 맞게 응용하여 적용한다. Type-JA 유형에서 COUNT bug가 발생하는 경우는 Magic Decorrelation[8]을 사용하여 정규화한다.

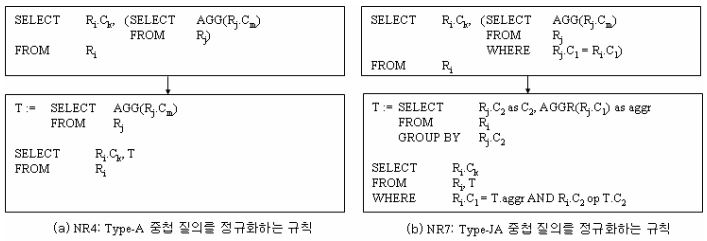

그림 3.6. SELECT 절의 Type-A/Type-JA 정규화 규칙.

Type-N이나 Type-J 유형의 정규화 규칙은 그림 3.7(a)나 그림 3.6(b)와 같이 본 논문에서 제안한 WHERE 절 스칼라 중첩 질의의 Type-N이나 Type-J 정규화 규칙을 SELECT 절에 맞게 응용하여 새롭게 제안한다.

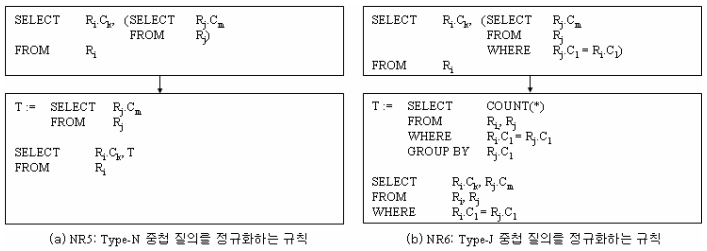

그림 3.7. SELECT 절의 Type-N/Type-J 정규화 규칙

3.4. FROM 절 정규화 규칙

FROM 절 중첩 질의의 정규화 규칙은 기존에 제안된 적이 없기 때문에 본 논문에서 새롭게 제안한다. 먼저, Type-A나 Type-JA 유형의 정규화 규칙은 그림 3.8(a)이나 그림 3.8(b)와 같이 XQuery의 FOR 절에 제안된 Type-A나 Type-JA 정규화 규칙[6]을 SQL3의 FROM 절에 맞게 응용하여 적용한다. Type-JA 유형에서 COUNT bug가 발생하는 경우는 Magic Decorrelation[8]을 사용하여 정규화한다.

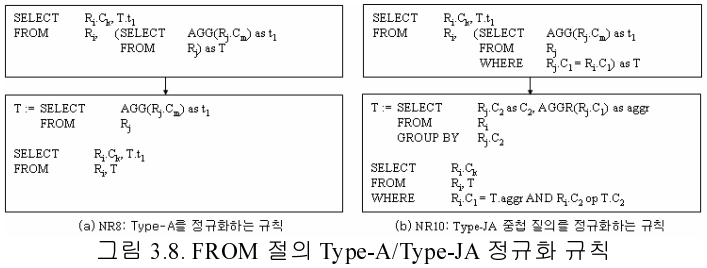

그림 3.8. FROM 절의 Type-A/Type-JA 정규화 규칙

Type-N이나 Type-J 유형의 정규화 규칙은 그림 3.9과 같이 WHERE 절의 Type-N이나 Type-J에 제안된 정규화 규칙[5]을 FROM 절에 맞게 응용하여 적용한다.

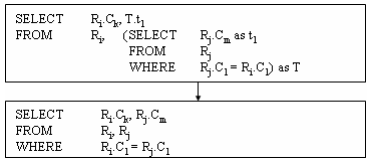

그림 3.9. NR9: FROM 절의 Type-N/Type-J 정규화 규칙

4. 결론

SQL3 중첩 질의는 고급 데이터 베이스 응용에서 빈번하게 사용된다. 중첩 질의를 포함한 SQL3 질의를 수행하는 것은 중첩 질의의 반복적인 실행을 야기하므로, 질의 처리 성능을 떨어뜨린다. 따라서, 중첩된 질의를 중첩되지 않은 질의로 정규화하여 처리하려는 많은 연구들[2][5][8]이 진행되었다. 그러나 기존에 제안된 정규화 규칙들은 완전하지 않다. 즉, 중첩 질의는 SELECT, FROM, WHERE 절 모두에 가능하지만, 기존의 규칙들은 WHERE 절의 중첩 질의에 대해서만 제안되었다. 더욱이, WHERE 절의 중첩 질의에 제안된 정규화 규칙은 SQL3 표준에 정의된 모든 중첩 유형을 처리하지 못한다.

본 논문에서는 효율적인 SQL3 질의 처리를 위해 SQL3 질의에 대한 완전한 정규화 규칙을 제안하였다. 즉, 중첩 질의를 중첩 질의 반환 값의 유형 및 상관과 집계의 유무에 따라 분류하고, SELECT, FROM, WHERE 절의 가능한 모든 중첩 유형에 대한 정규화 규칙을 제안하였으며 제안된 모든 규칙을 오디세우스 객체관계형 DBMS[10]에 구현하였다.

참고문헌

[1] Dayal, Y., “Of Nests and Trees: A Unified Approach to Processing Queries that contain Nested Subqueries, Aggregates and Quantifiers,” In *Proc. 13th Int'l Conf. on Very Large Data Bases,* pp. 197-208, Sept. 1987.

[2] Ganski, R. and Wong, K., “Optimization of Nested SQL Queries Revisited," In *Proc. Int'l Conf. on Management of Data,* ACM SIGMOD, pp. 23-33, May 1987.

[3] ISO/IEC 9075:1999, Information Technology-Database Languages-SQL-Part 2: Foundation, ISO/IEC, 1999.

[4] Kiessling, W., SQL-like and Quel-like Correlation Queries with Aggregates Revisited, UCB/ERL Memo 84/75, Electronics Research Laboratory, University of California, Berkeley, Sept. 1984.

[5] Kim, W., “On Optimizing an SQL-like Nested Query,” *ACM Trans. On Database Systems,* Vol. 7, No. 3, pp. 443-469, Sept. 1982.

[6] Lee, K., Kim, S., Whang, E., and Lee, G., “A Practitioner's Approach to Normalizing XQuery Expressions,” In *Proc. the 11th Int'l Conf. on Database Systems for Advanced Applications (DASFAA),* pp. 437-453, Apr. 2006.

[7] Manolescu, I., Florescu, D., and Kossmann, D., “Answering XML Queries over Heterogeneous Data Sources," In *Proc. the 27th Int'l Conf. on Very Large Data Bases,* pp. 241-250, Sept. 2001.

[8] Seshadri, P., Pirahesh, H., and Leung, T., “Complex Query Decorrelation,” In *Proc. the 17th Int'l Conf. on Data Engineering (ICDE)*, pp. 450-458, Feb. 1996.

[9] Silberschatz, A., Korth, H., and Sudarshan, S., *Database System Concepts*, McGraw-Hill, 2002.

[10] Whang, K., Lee, M., Lee, J., Kim, M., and Han, W., “Odysseus: a High-Performance ORDBMS Tightly-Coupled with IR Features,"' In *Proc. the 21st Int'l Conf. on Data Engineering (ICDE),* pp.1004-1005, Apr. 2005. This paper received the Best Demonstration Award.

[11] Whang, K., Malhotra, A., Sockut, G., and Burns, L., “Supporting Universal Quantification in a Two-Dimensional Database Query Language,” In *Proc. the 6th Int'l conf. on Data Engineering (ICDE),* pp. 68-75, Feb. 1990.

[12] World Wide Web Consortium, XQuery 1.0: An XML Query Language, W3C Working Draft, Apr. 2005 (available from http://www.w3.org/TR/xquery/).