오디세우스 객체관계형 DBMS를 위한 외부 조인의 설계 및 구현

김인중° 이기훈 황규영
한국과학기술원 전산학과/첨단정보기술연구센터
{ijkim°, khlee, kywhang}@mozart.kaist.ac.kr

Design and Implementation of Outer Join for the ODYSSEUS Object-Relational DBMS

In-Joong Kim° Ki-Hoon Lee Kyu-Young Whang
Department of Computer Science &
Advanced Information Technology Research Center
Korea Advanced Institute of Science and Technology

요 약

외부 조인은 조인에 참여하는 릴레이션들에서 조인 조건을 만족하지 않는 한쪽 또는 양쪽 튜플들도 결과로 반환하는 조인 연산으로 OLAP 질의 처리, 계층적 뷰 처리, 중첩 질의 처리 등의 다양한 고급 데이타베이스 응용에서 널리 사용된다. 많은 상용 DBMS에서 외부 조인을 지원하고 있으나, 상세한 구현 방법은 공개되어 있지 않다. 본 논문에서는 한국과학기술원 멀티미디어 및 데이타베이스 연구실에서 개발하고 있는 오디세우스 객체관계형 DBMS를 위한 외부 조인을 설계하고 구현한다. 본 논문에서는 거의 모든 DBMS에서 제공되는 가장 기본적인 조인 방법인 중첩 루프 조인 알고리즘을 확장하여 외부 조인 연산을 구현한다. 그리고, 외부 조인이 포함된 질의를 최적화하기 위해 조인 연산의 결과를 임시 릴레이션에 저장하는 대신에 다음 조인 연산의 입력으로 파이프라이닝시키는 것을 최대화 하는 방법을 제안한다.

1. 서론

외부 조인은 조인에 참여하는 릴레이션들에서 조인 조건을 만족하지 않는 한쪽 또는 양쪽 튜플들도 결과로 반환하는 조인 연산이다[1]. 외부 조인은 OLAP 질의 처리[2], 계층적 뷰(hierarchical view)의 지원[3], 스키마 사상(schema mapping)[4], 중첩 질의(nested query) 처리[5] 등의 다양한 고급 데이타베이스 응용에서 널리 사용된다.

외부 조인이 이와 같이 다양한 응용에서 사용되지만 상용 DBMS에서 외부 조인을 어떻게 구현하고 있는지가 자세히 공개되어 있지 않다. 외부 조인 연산을 구현하는 방법에는 외부 조인의 정의에 따라 내부 조인, 프로젝션(projection), 차집합, 외부 합집합(outer union) 등의 여러 개의 연산을 사용하여 구현하는 방법과 조인 알고리즘을 확장하여 구현하는 방법이 있다[6]. 여러 개의 연산을 사용하여 외부 조인 연산을 구현하면 성능이 떨어지게 되므로 일반적으로 조인 알고리즘을 확장하여 외부 조인 연산을 구현한다.

조인 알고리즘을 확장하여 외부 조인 연산을 구현하는 방법에는 정렬-합병(sort-merge) 조인 알고리즘을 확장하는 방법, 해시(hash) 조인 알고리즘을 확장하는 방법, 그리고 중첩 루프(nested loop) 조인 알고리즘을 확장하는 방법이 있다[6]. 세 가지 조인 알고리즘 중에서 중첩 루프 조인 알고리즘은 거의 모든 DBMS에서 제공되는 가장 기본적인 조인 방법이므로, 본 논문에서는 중첩 루프 조인 알고리즘을 확장하여 외부 조인 연산을 구현한다. 다른 조인 알고리즘을 확장하는 방법은 향후 연구로 한다.

외부 조인 질의의 처리 성능을 향상시키기 위한 연구로 질의에 외부 조인이 존재할 때, 질의 최적화기(query optimizer)가 최적의 조인 순서(join order)를 선택할 수 있도록 외부 조인을 포함한 질의의 유효한 모든 조인 순서를 구하는 방법에 대한 연구가 있다[7][8]. 내부 조인끼리는 항상 결합성(associativity)이 성립하기 때문에 모든 조인 순서가 유효하다. 하지만, 내부 조인과 외부 조인 사이의 결합성과 외부 조인끼리의 결합성은 경우에 따라 성립하지 않을 수도 있기 때문에, 외부 조인을 포함한 질의의 모든 유효한 조인 순서를 구하는 것은 간단한 문제가 아니다. 참고문헌 [7]에서는 외부 조인을 포함한 질의의 유효한 모든 조인 순서를 구하기 위하여 한쪽 외부 조인끼리의 유효한 결합성에 관한 기본적인 규칙을 제시하였으며, 참고문헌 [8]은 참고문헌 [7]의 규칙을 확장하여 내부 조인과 외부 조인 사이 및 외부 조인끼리의 유효한 결합성을 제시하고 이를 수학적으로 증명하였다.

외부 조인 질의 최적화를 위한 방법으로는 앞에서 설명한 방법 이외에 파이프라이닝(pipelining)을 사용하는 방법을 생각해 볼 수 있다. 파이프라이닝이란 한 연산의 결과를 다음 연산의 입력으로 직접 전달하여 여러 개의 연산이 동시에 수행되도록 하는 방법이다[6]. 파이프라이닝을 하면 각 연산의 전체 결과를 쓰고 다시 읽어 들이는 오버헤드를 줄일 수 있다. 일반적으로 질의에서 조인 연산을 파이프라이닝하여 처리함으로써 질의 처리 성능을 향상시킬 수 있으며, 조인 연산을 파이프라이닝하여 처리하는 방법들이 제안되었다[6].

그러나 대부분의 방법들이 내부 조인 질의에서 조인 연산을 파이프라이닝하여 처리하는 것만을 다루고 있으며, 외부 조인 질의에서 조인 연산을 파이프라이닝하여 처리하는 것에 대해서는 거의 연구된 바가 없다. 왜냐하면, 조인 연산을 파이프라이닝하여 처리하기 위해서는 파이프라이닝이 가능하도록 조인 순서를 바꿔야 하는데, 앞에서 설명한 바와 같이 외부 조인이 존재하는 경우에는 조인 순서를 바꾸는 것이 어렵기 때문이다. 본 논문에서는 내부 조인의 결합성과 참고문헌 [8]에서 제시한 결합성을 이용하여 외부 조인 질의에서 조인 연산을 파이프라이닝하여 처리하는 방법을 제안한다. 이를 위하여, 주어진 조인 트리를 최대한 파이프라이닝이 가능한 조인 트리로 변형하는 규칙을 제시한다.

본 논문에서는 오디세우스 객체관계형 DBMS를 기반으로 SQL 표준 명세[9]를 지원하는 외부 조인을 설계하고 구현한다. 오디세

* 본 연구는 첨단정보기술연구센터를 통하여 과학기술부/한국과학재단의 지원을 받았음.

우스는 한국과학기술원 전산학과 데이타베이스 및 멀티미디어 연구실에서 개발한 객체관계형 DBMS이다[10]. 본 논문의 공헌은 다음과 같다. 첫째, 오디세우스 객체관계형 DBMS를 확장하여 외부 조인을 설계하고 구현한다. 둘째, 외부 조인 질의를 최적화하기 위해 외부 조인 질의에서 조인 연산을 파이프라이닝하여 처리하는 방법을 제안한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 관련 연구로서 SQL 외부 조인 표준 명세, 외부 조인 연산의 구현, 외부 조인 질의 최적화에 대한 기존 연구를 설명한다. 제 3장에서는 오디세우스 객체관계형 DBMS 상에서 외부 조인의 설계와 구현에 대해 설명한다. 마지막으로 제 4장에서는 결론을 내린다.

2. 관련 연구

본 장에서는 관련 연구로서 SQL 외부 조인 표준 명세, 외부 조인 연산의 구현, 그리고 외부 조인 질의 최적화에 대해서 설명한다. 제 2.1절에서는 SQL 외부 조인 표준 명세에 대해 설명하고, 제 2.2절에서는 외부 조인 연산의 구현에 대해 설명하고, 제 2.3절에서는 외주 조인 질의 최적화에 대해서 설명한다.

2.1. SQL 외부 조인 표준 명세

외부 조인은 조인에 참여하는 릴레이션들에서 조인 조건을 만족하지 않는 한쪽 또는 양쪽 튜플들도 결과로 반환하는 조인 연산으로, SQL 표준 명세[9]에 포함되어 있다. 외부 조인에는 왼쪽 외부 조인(left outer join), 오른쪽 외부 조인(right outer join), 그리고 완전 외부 조인(full outer join)이 있으며, 각각 왼쪽, 오른쪽, 양쪽 릴레이션에서 조인 조건을 만족하지 않는 튜플들도 결과로 반환한다.

SQL 표준 명세에서 외부 조인은 FROM 절에 명시되며, 문법(syntax)은 다음과 같다.

<relation> {**LEFT**|**RIGHT**|**FULL**} [**OUTER**] **JOIN** <relation> **ON** <search condition>

여기서 LEFT, RIGHT, FULL은 각각 왼쪽 외부 조인, 오른쪽 외부 조인, 완전 외부 조인을 표시하기 위한 키워드이고 ON은 조인 조건을 나타내기 위한 키워드이다. 또한, <relation>은 기본 릴레이션 또는 조인 연산의 결과인 릴레이션이고, <search condition>은 불리언 값으로 평가되는 조인 조건이다.

2.2. 외부 조인 연산의 구현

외부 조인 연산의 구현에 앞서 먼저 본 논문에서 사용하는 표기법에 대해 설명한다. 본 논문에서는 참고문헌 [11]에서 사용하는 표기법을 따른다. 먼저 릴레이션 R의 스키마는 sch(R)로 표기한다. 왼쪽 외부 조인, 오른쪽 외부 조인, 완전 외부 조인은 각각 $\rightarrow$, $\leftarrow$, $\leftrightarrow$로 표기하며, 외부 조인 기호와 조인 조건을 같이 표기할 때는 조인 조건이 p라면 $\overset{p}{\rightarrow}$와 같이 화살표 위에 조인 조건을 표기한다.

외부 조인 연산을 내부 조인, 프로젝션, 차집합, 외부 합집합 등의 여러 개의 연산을 사용하여 구현하는 방법이 있으나, 여러 개의 복잡한 연산을 수행해야 하므로 성능이 떨어지는 문제가 있다. 이런 문제를 해결하기 위해 조인 알고리즘을 확장하여 외부 조인 연산을 구현하는 방법이 제시되었다[6]. 조인 알고리즘을 확장하여 외부 조인 연산을 구현하는 방법에는 세 가지가 있다[6]. 첫 번째는 정렬-합병 조인 알고리즘을 확장하는 방법이고, 두 번째는 해시 조인 알고리즘을 확장하는 방법이며, 세 번째는 중첩 루프 조인 알고리즘을 확장하는 방법이다.

정렬-합병 조인 알고리즘을 확장하는 방법에서는 한쪽 외부 조인 연산(왼쪽 외부 조인 연산과 오른쪽 외부 조인 연산)은 두 릴레이션을 정렬-합병한 후 한쪽 릴레이션에서만 조인 조건을 만족하지 않는 튜플에 null을 추가하여 반환하도록 구현한다. 완전 외부 조인 연산은 두 릴레이션을 정렬-합병한 후 양쪽 릴레이션에서 조인 조건을 만족하지 않는 튜플에 null을 추가하여 반환하도록 구현한다. 정렬-합병 조인 알고리즘을 확장하는 방법은 왼쪽 외부 조인, 오른쪽 외부 조인, 완전 외부 조인 연산 모두를 정렬-합병 조인과 유사한 방식으로 구현할 수 있다는 장점이 있다.

해시 조인 알고리즘을 확장하는 방법에서는 한쪽 외부 조인 연산은 두 릴레이션을 해시한 후 해시 값이 같은 버켓에 있는 튜플끼리 조인할 때 한쪽 릴레이션에서 조인 조건을 만족하지 않는 튜플에 null을 추가하여 반환하도록 구현한다. 완전 외부 조인 연산은 두 릴레이션을 해시한 후 해시 값이 같은 버켓에 있는 튜플끼리 조인할 때 양쪽 릴레이션에서 조인 조건을 만족하지 않는 튜플에 null을 추가하여 반환하도록 구현한다. 해시 조인 알고리즘을 확장하는 방법은 정렬-합병 조인 알고리즘을 확장하는 방법과 마찬가지로 왼쪽 외부 조인, 오른쪽 외부 조인, 완전 외부 조인 연산 모두를 해시 조인과 유사한 방식으로 구현할 수 있다는 장점이 있다.

중첩 루프 조인 알고리즘을 확장하는 방법에서는 왼쪽 외부 조인 연산은 조인 조건을 만족하지 않는 바깥쪽 릴레이션(outer relation)의 튜플에 null을 추가하여 반환하도록 구현한다. 오른쪽 외부 조인 연산은 $R_1 \overset{p}{\leftarrow} R_2 = R_2 \overset{p}{\rightarrow} R_1$와 같이 왼쪽 외부 조인 연산으로 바꾸어서 구현한다. 완전 외부 조인 연산은 $R_1 \overset{p}{\leftrightarrow} R_2 = (R_1 \overset{p}{\rightarrow} R_2) \cup (R_1 \overset{p}{\leftarrow} R_2)$와 같이 왼쪽 외부 조인 연산과 오른쪽 외부 조인 연산의 합집합으로 구현한다.

본 논문에서는 중첩 루프 조인 알고리즘을 확장하여 외부 조인 연산을 구현한다. 중첩 루프 조인 알고리즘을 확장하는 방법을 사용하는 이유는 이 조인 알고리즘이 거의 모든 DBMS에서 제공되는 가장 기본적인 조인 방법이기 때문이다. 다른 조인 알고리즘인 정렬-합병 조인 알고리즘과 해시 조인 알고리즘을 확장하는 방법은 향후 연구로 한다.

2.3. 외부 조인 질의 최적화

외부 조인 질의 최적화에 관한 연구로, 질의 최적화기가 최적의 조인 순서를 선택할 수 있도록 외부 조인 질의의 유효한 모든 조인 순서를 구하는 방법에 대한 연구가 있다[7][8]. 내부 조인끼리는 항상 결합성이 성립하기 때문에 모든 조인 순서가 유효하다. 하지만, 내부 조인과 외부 조인 사이의 결합성과 외부 조인끼리의 결합성은 경우에 따라 성립하지 않을 수도 있기 때문에, 외부 조인 질의의 유효한 모든 조인 순서를 구하는 것은 간단한 문제가 아니다. 따라서 외부 조인 질의의 유효한 모든 조인 순서를 구하기 위한 연구가 진행되었다.

참고문헌 [7]에서는 한쪽 외부 조인끼리의 유효한 결합성에 관한 기본적인 규칙을 제공하였다. 참고문헌 [8]에서는 참고문헌 [7]에서 제공한 기본적인 규칙을 확장하여 내부 조인과 한쪽 외부 조인 및 완전 외부 조인에 관한 유효한 결합성을 정의하고 이를 수학적으로 증명하였다. 참고문헌 [8]에서 정의하고 있는 결합성은 다음 식 (2.1) ~ (2.5)와 같다. 다음 식에서 p^{ij}는 R_i와 R_j사이의 조인 조건을 나타내는 술어이다. 또한, p^{ij} rejects nulls on $sch(R_k)$는 p^{ij}가 $sch(R_k)$에 대해 null을 허용하지 않는다는 것을 의미한다.

$$(R_1 \overset{p^{12}}{\leftarrow} R_2) \overset{p^{23}}{\rightarrow} R_3 = R_1 \overset{p^{12}}{\leftarrow} (R_2 \overset{p^{23}}{\rightarrow} R_3) \quad (2.1)$$

$$(R_1 \overset{p^{12}}{\leftarrow} R_2) \overset{p^{23}}{\rightarrow} R_3 = R_1 \overset{p^{12}}{\leftarrow} (R_2 \overset{p^{23}}{\rightarrow} R_3) \quad (2.2)$$
if p^{23} rejects nulls on sch(R_2)

$$(R_1 \overset{p^{12}}{\leftarrow} R_2) \overset{p^{23}}{\rightarrow} R_3 = R_1 \overset{p^{12}}{\leftarrow} (R_2 \overset{p^{23}}{\rightarrow} R_3) \quad (2.3)$$

$$(R_1 \overset{p^{12}}{\leftrightarrow} R_2) \overset{p^{23}}{\leftrightarrow} R_3 = R_1 \overset{p^{12}}{\leftrightarrow} (R_2 \overset{p^{23}}{\leftrightarrow} R_3) \quad (2.4)$$
if p^{12} and p^{23} rejects nulls on sch(R_2)

$$(R_1 \overset{p^{12}}{\leftrightarrow} R_2) \overset{p^{23}}{\rightarrow} R_3 = R_1 \overset{p^{12}}{\leftrightarrow} (R_2 \overset{p^{23}}{\rightarrow} R_3) \quad (2.5)$$
if p^{23} rejects nulls on sch(R_2)

식 (2.1) ~ (2.5)을 이용하면, 외부 조인 질의의 유효한 모든 조인 순서를 구할 수 있으며, 참고문헌 [8]에서 이를 증명하고 있다.

3. 외부 조인의 설계 및 구현

본 장에서는 외부 조인의 설계 및 구현으로 질의 처리기의 확장, 외부 조인 알고리즘, 그리고 파이프라이닝을 통한 외부 조인 질의 최적화에 대해 설명한다. 제 3.1절에서는 외부 조인을 구현하기 위한 질의 처리기의 확장에 대해 설명하고, 제 3.2절에서는 외부 조인 알고리즘에 대해 설명한다. 제 3.3절에서는 파이프라이닝을 통한 외부 조인 질의 최적화에 대해 설명한다.

3.1. 질의 처리기의 확장

본 논문에서는 오디세우스 객체관계형 DBMS[10]의 질의 처리기를 확장하여 외부 조인을 구현하였다. 오디세우스는 객체관계형 DBMS의 기능을 지원하기 위한 질의 언어 OOSQL(Odysseus Object-oriented SQL)을 제공하고 있다. OOSQL은 기존 SQL의 구문과 의미를 가능한 그대로 유지하면서 객체지향 개념을 지원할 수 있도록 확장한 객체지향 언어이다[13].

OOSQL 질의 처리기는 어휘 분석, 구문 분석, 의미 분석 및 내부 자료 구조(internal data structure)로의 변환, 실행 계획 생성, 실행의 과정을 거쳐 주어진 질의를 수행한다. 어휘 분석 및 구문 분석에서는 주어진 질의문이 구문 오류(syntax error)가 있는지 검사한 후, 오류가 없는 질의에 대해 질의문을 요약 구문 트리(abstract syntax tree)로 변환한다. 의미 분석 및 내부 자료 구조로의 변환에서는 주어진 요약 구문 트리에 대해 의미 오류(semantic error)가 없는지 검사한 후, 오류가 없으면 요약 구문 트리를 내부 자료 구조로 변환한다. 실행 계획 생성에서는 질의 최적화를 수행한 후 내부 자료 구조를 실행 계획으로 변환하고, 실행에서는 실행 계획에 따라 질의를 실행하고 질의 결과를 OOSQL 사용자에게 반환한다.

어휘 분석과 구문 분석 단계에서는 질의 처리기가 외부 조인을 문법적으로 처리할 수 있도록 확장한다. 어휘 분석기와 구문 분석기는 각각 lex(lexical analyzer generator)와 yacc(yet another compiler compiler)를 사용하여 구현되어 있으므로 lex와 yacc의 입력 파일을 수정한다. 외부 조인을 나타내기 위해 lex의 입력 파일에 6개의 키워드 LEFT, RIGHT, FULL, OUTER, JOIN, ON을 인식하도록 추가하고, yacc의 입력 파일에 외부 조인과 관련된 문법을 추가하였다. 질의에서 외부 조인이 표현되는 부분은 SELECT 문에서 FROM 절이므로 SELECT 문의 FROM 절 문법을 수정하였다.

의미 분석 단계에서는 요약 구문 트리로 표현된 외부 조인에 대해 다음 두 가지 오류를 검사하는 것을 추가하였다. 첫 번째로, 질의에 나타나는 모든 클래스와 속성들이 실제로 데이타베이스에 존재하는지 검사한다. 두 번째로, 비교 술어(comparison predicate)를 포함한 표현식(expression)에 대하여 속성과 값의 타입이 호환가능(type-compatible)한지 검사한다. 이 때 비교하는 값의 타입 변환(type conversion)이 필요하면 값의 타입을 변환하여 해당 심볼 테이블에 변환된 값을 대치한다.

내부 자료 구조로의 변환 단계에서는 외부 조인을 처리하기 위한 구조체를 추가하고 요약 구문 트리로 표현된 외부 조인을 이 구조체로 변환하는 과정을 추가하였다. 이 구조체는 외부 조인의 종류와 외부 조인의 왼쪽 및 오른쪽 피 연산자의 정보 그리고 조인 조건식에 대한 정보를 갖는다.

실행 계획 생성 단계에서는 질의 실행에 필요한 정보를 저장하는 자료 구조인 실행 계획에 외부 조인 정보를 추가하였다. 이 외부 조인 정보는 외부 조인의 종류와 외부 조인의 왼쪽 및 오른쪽 피 연산자의 정보 그리고 조인 조건식에 대한 정보이다.

실행 단계에서는 실행 계획에 있는 외부 조인 정보를 이용하여 외부 조인 연산을 수행하는 과정을 추가하였다. 외부 조인 연산을 수행하는 알고리즘은 제 3.2절과 같다.

3.2. 외부 조인 알고리즘

본 논문에서는 중첩 루프 조인 알고리즘을 확장하여 외부 조인 연산을 구현하는 방법을 채택하였다. 이 방법은 제 2.2절에서 언급한 바와 같이 기본적으로 왼쪽 외부 조인 연산을 처리하며, 오른쪽 외부 조인 연산은 왼쪽 외부 조인 연산으로 바꾸어서 처리한다. 완전 외부 조인 연산은 왼쪽 외부 조인 연산과 오른쪽 외부 조인 연산의 합집합으로 처리한다.

다음은 중첩 루프 조인 알고리즘을 확장하여 왼쪽 외부 조인 연산을 처리하는 알고리즘이다.

왼쪽 외부 조인 연산 처리 알고리즘

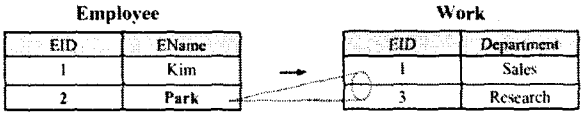
입력: 바깥쪽 릴레이션과 안쪽 릴레이션.

출력: 왼쪽 외부 조인 연산 결과.

절차:

1. 바깥쪽 릴레이션의 각 튜플 T_1에 대해 조인 조건을 만족하는 안쪽 릴레이션의 튜플 T_2가 존재하는지 찾는다.
2. 튜플 T_2가 존재하면 튜플 T_1과 T_2를 합친 결과를 생성한다.
3. 튜플 T_2가 존재하지 않으면 튜플 T_1과 안쪽 릴레이션의 모든 속성 값이 null인 튜플을 합친 결과를 생성한다.

예 1: 그림 3.1은 왼쪽 외부 조인 연산 처리 알고리즘을 실행하는 예이다. 그림에서 Employee 릴레이션은 바깥쪽 릴레이션에 해당되며, Work 릴레이션은 안쪽 릴레이션에 해당된다. 절차 1에서 바깥쪽 릴레이션의 튜플 (1, 'Kim')은 조인 조건을 만족하는 안쪽 릴레이션의 튜플 (1, 'Sales')이 존재하므로, 절차 2에서 이 두 튜플을 합친 (1, 'Kim', 1, 'Sales') 튜플을 생성하였다. 그리고, 튜플 (2, 'Park')은 조인 조건을 만족하는 튜플이 존재하지 않으므로, 절차 3에서 (2, 'Park') 튜플과 안쪽 릴레이션의 모든 속성 값이 null인 (null, null) 튜플을 합친 (2, 'Park', null, null) 튜플을 생성하였다. □



Employee E $\xrightarrow{E.EID = W.EID}$ Work W

E.EID	E.EName	W.EID	W.Department
1	Kim	1	Sales
2	Park	null	null

그림 3.1. 왼쪽 외부 조인 연산 처리 알고리즘의 실행 예.

3.3. 파이프라이닝을 통한 외부 조인 질의 최적화

일반적으로 질의에서 조인 연산을 파이프라이닝하여 처리함으로써 질의 처리 성능을 향상시킬 수 있으며, 조인 연산을 파이프라이닝하여 처리하는 방법들이 제안되었다[6]. 그러나 대부분의 방법들이 내부 조인 질의에서 조인 연산을 파이프라이닝하여 처리하는 것만을 다루고 있으며, 외부 조인 질의에서 조인 연산을 파이프라이닝하여 처리하는 것에 대해서는 거의 연구된 바가 없다.따라서 본 논문에서는 외부 조인 질의를 최적화하기 위해 외부 조인 질의에서 조인 연산을 파이프라이닝하여 처리하는 방법을 제안한다.

조인 연산을 파이프라이닝하여 처리하면 각 조인 연산의 전체 결과를 저장하고 다시 읽어 들이는 오버헤드를 줄일 수 있다[6]. 그림 3.2는 조인 연산을 파이프라이닝하여 처리하는 예이다. 그림 (a)는 조인 트리와 조인에 참여하는 릴레이션들을 나타낸다. 그림 (b)는 (A ⋈ B)의 중간 결과가 릴레이션 C와의 조인 연산의 입력으로 파이프라인(pipeline)된 것을 나타낸다. 그림 (c)는 그림 (b)에서 릴레이션 C와의 조인 연산이 이루어진 후를 나타낸다. 그림 (d)는 그림 (b)와 (c)의 과정을 계속 반복하여 생성한 (A ⋈ B) ⋈C의 전체 결과를 나타낸다. 이렇게 (A ⋈B)의 결과를 릴레이션 C와의 조인 연산의 입력으로 파이프라인하여 처리하면, (A ⋈ B)의 전체 결과를 저장하고 다시 읽어 들이는 오버헤드를 줄일 수 있다.

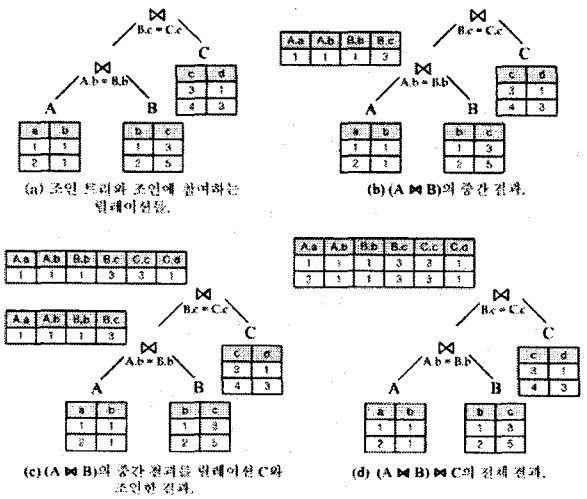


그림 3.2. 조인 연산을 파이프라이닝하여 처리하는 예.

본 논문에서는 중첩 루프 조인 알고리즘을 확장하여 외부 조인 연산을 구현하기로 하였으므로 중첩 루프 조인의 파이프라이닝 방법을 사용하여 조인 연산을 파이프라이닝한다. 중첩 루프 조인의 파이프라이닝 방법을 사용하려면, 조인 트리가 다음 두 가지 제약 조건을 만족해야 한다. 첫째, 좌향(left-deep) 조인 트리[6]이다. 둘째, 좌향 조인 트리 내에는 내부 조인과 왼쪽 외부 조인만 존재한다. 즉, 좌향 조인 트리 내에 존재하는 모든 외부 조인은 왼쪽 외부 조인이어야 한다. 본 논문에서는 이러한 제약 조건을 중첩 루프 조인을 파이프라이닝하기 위한 제약 조건(Constraints for Pipelining Nested Loop Join)이라 정의하고, 앞으로 이를 C-PNLJ라 표기한다.

조인 트리가 좌향 조인 트리이어야 하는 이유는 다음과 같다. 중첩 루프 조인 알고리즘은 바깥쪽 릴레이션을 순차적으로 스캔(scan)하면서 각 튜플에 대해 안쪽 릴레이션(inner relation)에서 조인 조건을 만족하는 튜플을 찾아 결과를 생성한다. 따라서 바깥쪽(즉, 왼쪽) 릴레이션에 해당하는 입력만 파이프라인 될 수 있으며, 이를 위해서는 조인 트리가 좌향 조인 트리이어야 한다.

좌향 조인 트리 내에 존재하는 모든 외부 조인이 왼쪽 외부 조인이어야 하는 이유는 다음과 같다. 제 2.2절에서 설명하였듯이 중첩 루프 조인 알고리즘을 사용하는 경우에는 왼쪽 외부 조인만 처리가 가능하다. 그러므로 좌향 조인 트리에 오른쪽 외부 조인이 있는 경우에는 오른쪽 외부 조인을 왼쪽 외부 조인으로 바꾼 다음 처리를 해야 한다. 그러나 그렇게 하면 조인 트리가 좌향 트리가 아니게 되어 파이프라이닝이 되지 않는다. 완전 외부 조인의 경우에는 왼쪽 외부 조인의 결과와 오른쪽 외부 조인의 결과를 생성해서 합집합해야 하므로 파이프라이닝 할 수 없다. 그러므로 좌향 조인 트리 내에 존재하는 모든 외부 조인은 왼쪽 외부 조인이어야 한다.

예 2: 그림 3.3은 C-PNLJ를 만족하는 조인 트리의 예이다. 이 조인 트리는 좌향 조인 트리이면서, 외부 조인은 왼쪽 외부 조인만 존재하므로 중첩 루프 조인의 파이프라이닝 방법을 사용하여 처리할 수 있다.□

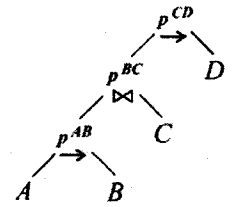


그림 3.3. C-PNLJ를 만족하는 조인 트리의 예.

본 논문에서 중첩 루프 조인의 파이프라이닝을 통해 조인 트리를 처리하는 방법은 다음과 같다. 첫째, 조인 트리에서 C-PNLJ를 만족하는 부 트리(subtree)는 파이프라이닝하여 처리한다. 둘째, C-PNLJ를 만족하지 않는 부 트리는 조인 결과를 임시 릴레이션에 저장하여 처리한다.

예 3: 그림 3.4는 중첩 루프 조인의 파이프라이닝을 통해 조인 트리를 처리하는 예이다. 그림에서 부 트리 ST_1은 C-PNLJ를 만족하므로 파이프라이닝하여 처리한다. 그리고 부 트리 ST_2는 C-PNLJ를 만족하지 않으므로 (C ⋈ D)의 전체 결과를 임시 릴레이션에 저장하여 처리한다.□

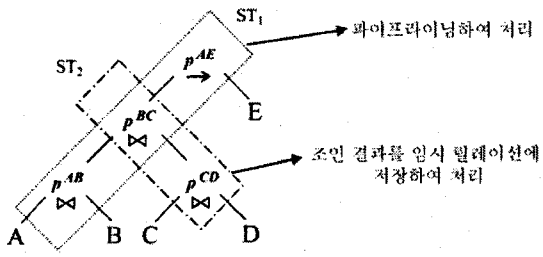


그림 3.4. 중첩 루프 조인의 파이프라이닝을 통해 조인 트리를 처리하는 예.

만약 그림 3.4의 조인 트리를 변형하여 ST_2를 파이프라이닝하여 처리할 수 있다면, 질의 처리 성능을 향상시킬 수 있다. 이렇게 파이프라이닝을 통해 처리할 수 있는 조인의 개수를 최대한 늘림으로써 질의 처리 성능을 향상시키는 방법을 최대로 파이프라인된 처리(maximally pipelined processing)라고 한다[12].

본 논문에서는 중첩 루프 조인의 파이프라이닝을 통해 처리할 수 있는 조인의 개수를 최대한 늘리기 위해 조인 트리를 변형한다. 조인 트리의 변형은 C-PNLJ를 만족하는 부 트리의 크기를 증가시키는 방향으로 한다. 또한, 조인 트리를 변형할 때에는 내부 조인끼리의 결합성과 제 2.3절에서 설명한 내부 조인과 외부 조인 사이의 결합성 및 외부 조인끼리의 결합성을 이용한다. 조인 트리 변형 규칙은 다음과 같다.

조인 트리 변형 규칙 1: $T_1 \overset{p^{12}}{\ltimes} (T_2 \overset{p^{23}}{\ltimes} T_3) \Rightarrow (T_1 \overset{p^{12}}{\ltimes} T_2) \overset{p^{23}}{\ltimes} T_3$

조인 트리 변형 규칙 2: $T_1 \overset{p^{12}}{\ltimes} (T_2 \overset{p^{23}}{\rightarrow} T_3) \Rightarrow (T_1 \overset{p^{12}}{\ltimes} T_2) \overset{p^{23}}{\rightarrow} T_3$

조인 트리 변형 규칙 3: $T_1 \overset{p^{12}}{\rightarrow} (T_2 \overset{p^{23}}{\rightarrow} T_3) \Rightarrow (T_1 \overset{p^{12}}{\rightarrow} T_2) \overset{p^{23}}{\rightarrow} T_3$

단, p^{23}이 $sch(T_2)$에 대하여 null을 허용하지 않을 경우

조인 트리 변형 규칙 1은 내부 조인끼리의 결합성을 이용한 규칙이며, 조인 트리 변형 규칙 2는 제 2.3절의 식 (2.1)을 이용한 규칙이다. 그리고 조인 트리 변형 규칙 3은 제 2.3절의 식 (2.2)를 이용한 규칙이다. 조인 트리 변형 규칙 1, 2, 3을 그림으로 나타내면 각각 그림 3.5, 3.6, 3.7과 같다. 각각의 그림에서 조인 트리 변형 규칙을 적용하기 전의 부 트리 ST_1은 C-PNLJ를 만족하지 않지만 규칙을 적용한 후의 부 트리 ST_2는 C-PNLJ를 만족함을 볼 수 있다.

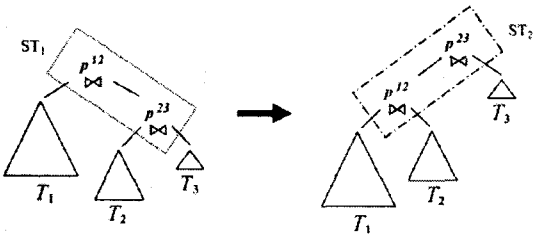


그림 3.5. 조인 트리 변형 규칙 1.

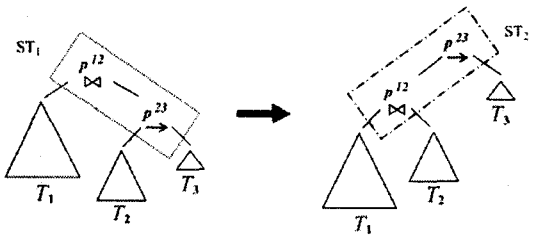


그림 3.6. 조인 트리 변형 규칙 2.

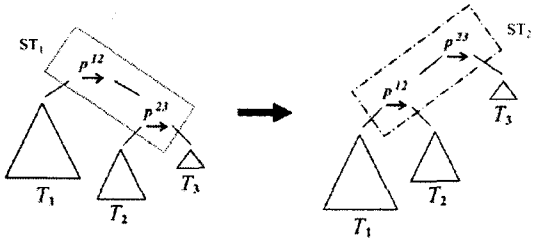


그림 3.7. 조인 트리 변형 규칙 3.

4. 결론

본 논문에서는 SQL 표준 명세를 따르는 외부 조인을 설계하고 오디세우스 객체관계형 DBMS에 구현하였다. 외부 조인은 조인에 참여하는 릴레이션들에서 조인 조건을 만족하지 않는 한쪽 또는 양쪽 튜플들도 결과로 반환하는 조인 연산으로 다양한 고급 데이타베이스 응용에서 널리 사용된다.

본 논문의 공헌은 다음 두 가지이다. 첫째, 오디세우스 객체관계형 DBMS를 확장하여 외부 조인을 설계하고 구현하였다. 본 논문에서는 오디세우스 객체관계형 DBMS의 질의 처리기를 확장하여 외부 조인 질의를 처리할 수 있도록 하였다.

둘째, 외부 조인 질의 최적화를 위해 외부 조인 질의에서 조인 연산을 파이프라이닝하여 처리하는 것을 최대화 하는 방법을 제안하였다. 이를 위해, 내부 조인끼리의 결합성과 참고문헌 [8]에서 제시한 결합성을 이용하여 주어진 조인 트리를 최대한 파이프라이닝이 가능한 조인 트리로 변형시키는 규칙을 제안하였다.

참고문헌

[1] Codd, E., "Extending the Relational Database Model to Capture More Meaning," *ACM Trans. on Database Systems*, Vol. 4, No. 4, pp.397-434, Dec. 1979.

[2] Carlos, A., Alberto, O., and Alejandro, A., "Updating OLAP Dimensions," In *Proc. 2nd ACM Int'l Workshop on Data Warehousing and OLAP*, pp. 60-66, Kansas City, Missouri, Nov. 1999.

[3] Lee, B. and Wiederhold, G., "Outer Joins and Filters for Instantiating Objects from Relational Databases Through Views," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 6, No. 1, pp. 108-119, Feb. 1994.

[4] Hernandez, M., Miller, R., and Haas, L., "Clio: A Semi-Automatic Tool for Schema Mapping," In *Proc. Int'l Conf. on Management of Data*, ACM SIGMOD, pp. 585-594, Santa Barbara, CA, May. 2001.

[5] Dayal, U., "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, Brighton, England, Sept. 1987.

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

[7] Dayal, U., "Processing Queries with Quantifiers," In *Proc. 2nd ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pp. 125-136, Atlanta, Geogia, Sept. 1983.

[8] Galindo-Legaria, C., Algebraic Optimization of Outerjoin Queries, Ph.D. Thesis, Department of Applied Science, Havard University, 1992.

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

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

[11] Galindo-Legaria, C. and Rosenthal, A., "Outerjoin Simplification and Reordering for Query Optimization," *ACM Trans. on Database Systems*, Vol. 22, No. 1, pp. 43-73, Mar. 1997.

[12] Liu, B. and Rundensteiner, E., "Revisiting Pipelined Parallelism in Multi-Join Query Processing," In *Proc. 31st Int'l Conf. on Very Large Data Bases*, pp. 829-840, Trondheim, Norway, Sept. 2005.

[13] 우 준호, ODYSSEUS 객체지향 데이타베이스 시스템을 위한 질의 처리기의 설계 및 구현, 석사 학위 논문, KAIST 전산학과, 1995.