한국산학기술학회논문지
Vol. 11, No. 12 pp. 5042-5048, 2010

시공간 데이터 스트림 처리를 위한 영역 기반의 연산자 공유 기법

정원일[1*], 김영기[2]
[1]호서대학교 정보보호학과, [2]인하대학교 컴퓨터정보공학과

Partition-based Operator Sharing Scheme for Spatio-temporal Data Stream Processing

Weonil Chung[1*] and Young-Ki Kim[2]
[1]**Dept. of Information Security Engineering, Hoseo University**
[2]**Dept. of Computer and Information Engineering, Inha University**

요 약 유비쿼터스 환경에서 다양한 센서로부터 생성되는 데이터 스트림에 대해 사용자가 요구하는 위치 기반 서비스를 효과적으로 지원하기 위해 연산자 네트워크와 공유 등을 통한 연속 질의 처리 기법이 활용되고 있다. 위치 정보 기반의 연속 질의 처리는 특정 영역을 중심으로 유사 질의가 집중적으로 발생할 수 있으므로, 본 논문에서는 유사한 조건을 갖는 공간 연산을 공유하기 위해 그리드 영역 분할을 이용한 공간 연산 공유 기법을 제안한다. 제안기법은 공간 연산을 직접 공유하지 않고 그리드 셀 별로 이동객체를 공유하기 때문에 유사한 조건을 갖는 공간 연산의 공유가 가능하여 공간 연산의 실행 빈도수를 크게 감소시켜 질의 처리 속도의 향상과 함께 메모리 이용률을 향상시킨다.

Abstract In ubiquitous environments, many continuous query processing techniques make use of operator network and sharing methods on continuous data stream generated from various sensors. Since similar continuous queries with the location information intensively occur in specific regions, we suggest a new operator sharing method based on grid partition for the spatial continuous query processing for location-based applications. Due to the proposed method shares moving objects by the given grid cell without sharing spatial operators individually, our approach can not only share spatial operators including similar conditions, but also increase the query processing performance and the utilization of memory by reducing the frequency of use of spatial operators.

Key Words : Spatio-temporal Data Stream, Operator Sharing, Continuous Query

1. 서론

유비쿼터스 환경에서 텔레매틱스 단말, 휴대폰과 같은 이동 객체는 시간에 따라 변하는 객체의 위치 정보를 포함하는 시공간 데이터를 생성한다[1,2]. 테마파크 서비스를 위한 "1시간동안 롤러코스터를 중심으로 반경 1km 이내에 있는 이용객 중 자유이용권 구매자가 아닌 이용객을 검색하라"와 같은 질의는 "반경 1km 이내"에 존재하는 고객을 검색하기 위한 공간 연산[3,4]과 "자유이용권 구매자가 아닌 이용객"을 검색하기 위한 비공간 연산이 결합되어 지정된 시간동안 지속적인 처리가 요구되는 연속 질의(CQ: Continuous Query)이다. 이러한 연속 질의 처리를 위해 입출력되는 시공간 데이터는 시간에 따라 변화 가능한 위치 정보가 포함된 시공간 데이터의 특성[1,2,5]과 지속적으로 발생하는 데이터 스트림의 특성[6,7]을 포함하고 있다. 이와 같은 서비스들을 제공하기 위해서는 연속 질의에 표현되는 공간 연산들이 유사한 조건에 대해 특정 공간 영역을 집중적으로 검색하는 경우 성능 저하를 방지하기 위해 질의 최적화 기술이 요구된다.

*교신저자 : 정원일(wnchung@hoseo.edu)
접수일 10년 10월 22일 수정일 (1차 10년 11월 22일, 2차 10년 12월 13일) 게재확정일 10년 12월 17일

데이터 스트림 관리 시스템은 데이터 스트림을 입력으로 등록된 연속 질의를 처리하여 그 결과를 외부 응용으로 전달하는 시스템이다[8-11]. 이러한 연구에서는 지속적으로 빠르게 생성되는 데이터 스트림에 대해 연속 질의 최적화를 위해 연속 질의에 공통적으로 표현된 연산자의 공유를 통해 시스템의 가용성을 높이기 위한 연산자 네트워크 기법을 활용하고 있다. 이러한 연산자 네트워크는 연속 질의의 추가에 따라 연산자 네트워크의 복잡도가 급격하게 증가할 수 있으므로 입력되는 데이터 스트림 개개의 데이터 크기가 작더라도 많은 연산이 수행되어야 하는 문제를 갖고 있다. 이에 데이터 스트림 관리 시스템은 질의 최적화를 위해 연산자 박스 결합이나 재배열을 통해 연산 횟수를 감소시킨다.

그러나 특정 영역을 기반으로 유사 조건을 포함하는 공간 연산이 집중적으로 포함되는 연속 질의를 처리하는 경우 정확하게 일치하는 조건을 기준으로 연산자 네트워크를 구성하여 최적화를 수행하는 기존의 기법에서는 이러한 유사 연산들은 모두 독립적으로 처리되므로 연산양의 증가로 처리율이 떨어지게 된다.

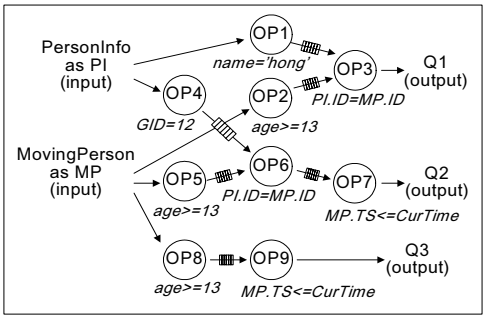
본 논문에서는 효과적인 유비쿼터스 응용 서비스 제공을 위해 시공간 연속 질의 처리를 위한 그리드 영역 기반의 연산 공유 기법을 제안한다. 제안 기법은 기존의 동일 조건 기반의 연산 공유 기법을 확장하여 질의 처리 비용이 높은 공간 연산의 연산 횟수 감소를 위해 시공간 객체의 위치와 시공간 연속 질의의 공간 연산 영역을 그리드 필터링을 통해 분할하고 해당 그리드 셀의 질의와 객체를 공유하여 중복 연산 수행을 감소시킨다. 또한, 연산 공유로 인한 데이터의 중복 적재를 방지하여 메모리 이용률을 높인다.

2. 관련 연구

2.1 연속 질의 처리

증권 분석, 네트워크 트래픽 분석 및 센서 네트워크 모니터링과 같이 시간에 따라 변화하고 연속적으로 생성되는 데이터 스트림 처리를 위해서는 기존의 DBMS는 한계를 가진다[4]. 이에 데이터 스트림에 대한 연속 질의 처리를 위한 연구로 Aurora[8], STREAM[9], NiagaraCQ[10], TelegraphCQ[11], StreamBase[12], Coral8[13] 등의 연구가 수행되었으며, 이러한 연구에서는 실시간 데이터 처리율을 높이기 위해 최적화 기법들을 활용하고 있다.

그림 1은 기존의 연구들에서 최적화를 위해 활용되는 연산자 네트워크에 대한 내용으로, 입력되는 시공간 데이터 스트림에 대해 연산자 박스를 통해 사용자가 원하는 데이터를 검색한 후 그 결과를 사용자 응용으로 전달하도록 하는 연산자 네트워크[8]를 표현하고 있다. 그러나 그림 1의 연산자 네트워크에서 질의(Q1)의 연산자 박스(OP2), 질의(Q2)의 연산자 박스(OP5) 그리고 질의(Q3)의 연산자 박스(OP8)가 동일한 연산을 수행하며, 질의(Q2)의 연산자 박스(OP7)와 질의(Q3)의 연산자 박스(OP9)도 동일한 연산을 수행한다. 결과적으로 Q1, Q2, Q3은 유사한 질의로 동일한 연산을 중복하여 수행하므로 연산 수행 횟수가 증가하여 질의 처리 비용의 증대를 야기한다. 따라서 중복된 연산을 독립적으로 수행하지 않고 연산 결과를 공유하는 연산 공유 기법이 필요하다.



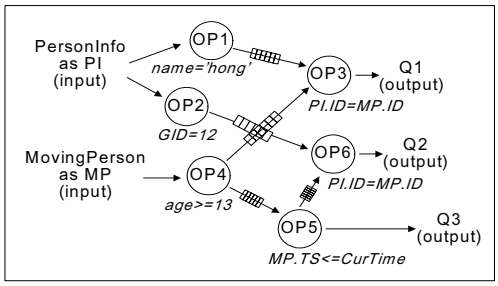
[그림 1] 연산자 네트워크

2.2 연산자 공유 기법

연속 질의 처리를 위한 연산자 네트워크에서 연속 질의의 등록에 따라 연산자 박스의 증가는 네트워크의 복잡도 증가를 야기하고, 유입되는 데이터 스트림 개개의 데이터가 크기에 관계없이 급격한 질의 처리 비용의 증가로 이어질 수 있는 문제가 있다. 이에 질의 최적화를 위해 연산 공유 기반의 연산자 박스 결합이나 재배열 등을 통해 연산의 중복 수행을 방지하여 연산 횟수를 감소시키기 위한 연산 공유 기법 연구가 수행되었다[8,9].

그림 2는 연산자 네트워크를 예시한 그림 1을 기반으로 공유되는 연산을 재배치하여 구성하고 있다. 그림 2에서는 연산 공유를 통해 그림 1의 Q1, Q2, Q3의 "MP.Age => 13" 연산을 하나의 연산자 박스(OP4)로 공유하며 유사 질의 경로를 갖는 Q2와 Q3의 "MP.TS <= CurTime" 연산을 공유하고 순서를 재구성하여 연산의 중복 수행을 방지한다. 그러나 기존의 연속 질의와 달리 공간 연산이 결합된 연속 질의는 연산 조건이 정확히 일치하는 연산보다 특정 영역에 대한 유사한 조건을 갖는 질의가 대량으로 발생하기 때문에 기존의 연산 공유 기법은 부적합

하다.



[그림 2] 연산 공유를 적용한 연산자 네트워크

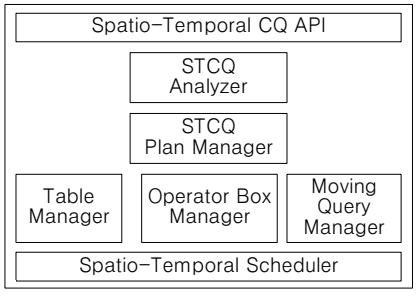
3. 그리드 영역 기반 연산 공유

시공간 연속 질의 처리 구조를 통한 중점 설계 사항은 연산자 박스 구조, 비공간 연산 공유를 이용한 2단계 필터링, 그리고 공간 속성에 대한 그리드 영역 기반의 필터링이다. 각 항목에 대한 내용은 아래 표 1과 같다.

[표 1] 설계 주요 항목

구분	내용
연산자 박스	동일 연산의 중복 방지를 위해 연산 특성을 반영한 상속 관계의 연산자 박스를 구성함
2단계 필터링	비공간 속성에 대한 유사 연산을 포함하는 질의 처리를 위해 느슨한 필터링과 엄격한 필터링으로 구분하여 수행함
영역 필터링	특정 영역을 중심으로 발생 빈도가 높은 공간 속성 기반의 연산을 처리하기 위해 그리드 기반의 필터링을 먼저 처리함

그리드 영역 기반의 연산 공유 기법을 이용하여 시공간 연속 질의 처리 구조는 아래 그림 3과 같다.

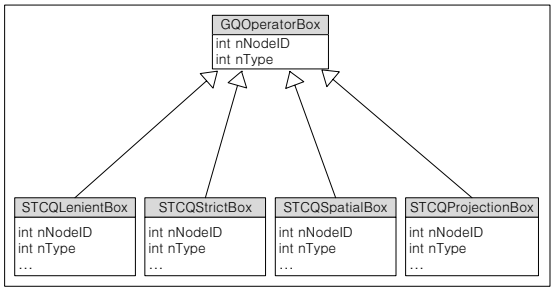


[그림 3] 시공간 연속 질의 처리 구조

그림 3에서 Spatio-Temporal CQ API(시공간 연속질의 API)는 외부 응용의 요청을 위한 인터페이스이며, STCQ Analyzer(시공간 연속질의 분석기)는 입력된 질의를 분석하고 내부 처리 형식으로 변환한다. STCQ Plan Manager (시공간 연속질의 질의 계획 관리자)는 질의 수행 계획의 설정을 담당하며, Operator Box Manager(연산자 박스 관리자)는 Plan Manager로부터 요구하는 연산자 박스를 생성하여 전달한다. Table Manager(테이블 관리자)는 시공간 데이터 스트림 관리를 위한 테이블, 연산자 박스의 연산자 처리 결과를 캐싱하는 가상 테이블, 가상 테이블을 참조하는 요약 테이블 등을 관리하며, Spatio-Temporal Scheduler(시공간 스케줄러)는 등록된 질의를 대상으로 연산자 수행을 담당한다.

3.1 연산자 박스 구조

그리드 영역 기반의 연산 공유 기법을 이용한 시공간 연속 질의 처리는 연산자 박스를 특성에 따라 구분하여 사용한다. 연산자 박스는 느슨한 필터링 연산자 박스 (Lenient Filtering Operator Box), 엄격한 필터링 연산자 박스(Strict Filtering Operator Box), 공간 연산자 박스 (Spatial Operator Box), 추출 연산자 박스(Projection Operator Box)로 구분된다. 필터링 연산자 박스는 입력 큐로부터 튜플을 입력 받아 비교 연산(>,<,=,>=,<=,<>)을 수행하여 출력 큐로 전달한다. 공간 연산자 박스는 입력 큐로부터 입력 받은 튜플들에 대해 공간 연산을 수행하여 출력 큐로 전달한다. 마지막으로 추출 연산자 박스는 연산 결과에 대한 추출 연산을 수행 후 출력 스트림으로 전달한다. 모든 연산자 박스는 추상화된 연산자 박스 구조로부터 상속 받아 구현되었다. 그림 4는 연산자 박스의 상속 관계를 나타낸다.



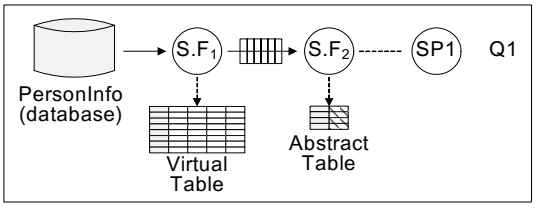
[그림 4] 연산자 박스 상속 관계

각각의 연산자 박스는 각 특성에 맞는 구조 외에 공통적으로 식별자(nNodeID)와 연산자 박스 타입(nType)을 갖으며 연산자 박스 수행을 위한 메소드를 포함한다.

3.2 비공간 연산 공유를 이용한 2단계 필터링

그리드 영역 기반의 연산 공유 기법을 이용한 시공간 연속 질의 처리는 공간 연산 부분과 비공간 연산 부분으로 분류하여 수행된다. 공간 연산은 그리드 영역 기반의 필터링 연산을 통해 시공간 데이터 스트림과 이동 질의 간의 융합처리를 수행하며 비공간 연산은 각 시공간 데이터 스트림의 비공간 속성에 대한 2단계 필터링 과정을 통해 공간 연산과 맵핑을 하여 질의를 수행한다.

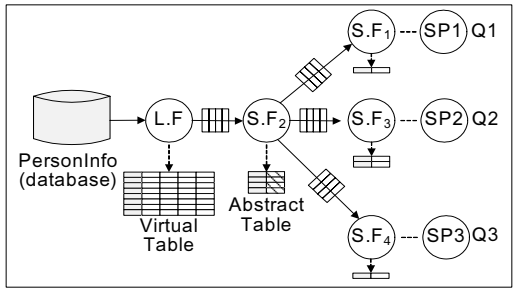
하나의 질의가 입력되면 먼저 비공간 연산 공유를 이용한 2단계 필터링을 수행한다. 2단계 필터링은 느슨한(lenient) 필터링과 엄격한(strict) 필터링으로 구분되며 동일한 비공간 속성에 대한 다른 연산을 포함한 유사 질의가 등록될 경우 동일한 비공간 속성에 대한 연산이 임계값 이상 존재한다면 해당 비공간 속성에 대한 연산을 모두 포괄하는 넓은 범위의 느슨한 필터링을 생성하고 공유하여 선 필터링 후 처리한다. 만약 유사 질의가 등록되지 않을 경우 엄격한 필터링만으로 수행된다. 비공간 속성에 대한 필터링 연산은 질의 등록 시 수행되며 단말 연산자 박스는 해당 질의의 공간 연산과 맵핑된다.



[그림 5] 엄격한 필터링 수행 과정

그림 5는 엄격한 필터링만으로 구성된 수행 과정이다. 질의 등록 발생 시 먼저 입력된 질의로부터 비공간 연산을 추출하여 각 연산에 대한 연산자 박스를 생성한다. 연산자 박스는 연산의 대상 비공간 정보 속성의 우선순위에 따라 배열되며 비공간 정보를 포함하고 있는 데이터베이스로부터 데이터를 가져와서 연산자 박스를 이동하며 연산을 수행한다. 첫 번째 연산자 박스는 수행한 결과 튜플을 가상 테이블로 저장하며 이후의 연산자 박스는 객체의 식별자와 앞의 수행 결과에 대해 빗금으로 표시된 참조자만으로 구성된 요약 테이블을 저장하여 데이터의 중복을 감소시킨다.

그림 6은 비공간 연산 공유를 이용한 2단계 필터링 수행 과정이다. 만약 임계값 이상의 유사 질의 발생 시 해당 연산자에 대한 느슨한 연산자 박스를 생성하고 해당 연산자 박스를 뒤로 이동한다.

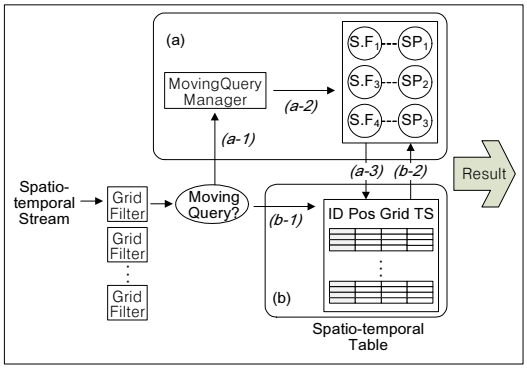


[그림 6] 비공간 연산 공유를 이용한 2단계 필터링 수행 과정

3.3 공간 속성에 대한 그리드 영역기반 필터링

공간 속성에 대한 연산은 그리드 기반 필터링을 선 수행 후 시공간 데이터 스트림 테이블에 갱신하여 해당 그리드 셀에 시공간 객체와 시공간 연속 질의를 공유하는 그리드 영역 기법을 사용한다. 갱신 이후 질의 수행은 입력된 데이터가 이동 질의를 구성하는 중심 객체일 경우와 그렇지 않은 경우로 나뉜다. 입력된 데이터가 이동 질의를 구성하는 중심 객체일 경우 이동 질의 관리자를 통해 이동 질의의 영역을 재구성하고 이동 질의의 영역에 속한 그리드에 대한 정보를 갱신한다. 그리고 시공간 데이터 스트림 테이블의 질의가 포함하는 그리드 셀에 공유된 시공간 객체에 대해서만 연산을 수행한다. 연산 수행은 먼저 타임스탬프를 비교하여 이동 질의 보다 과거에 갱신된 데이터는 제외하고 이동 질의에 맵핑된 비공간 단말 연산자의 테이블을 통해 식별자로 필터링한다. 마지막으로 필터링된 데이터에 한해서 공간 연산(CONTAINS)을 수행한다. 만약 입력된 데이터가 이동 질의를 구성하는 중심 객체가 아닐 경우 입력된 데이터가 해당하는 그리드 셀에 공유된 이동 질의와의 연산을 통해 질의 수행이 완료된다. 이동 질의와의 연산은 해당 이동 질의에 맵핑된 단말 비공간 연산자 박스의 테이블과 식별자에 대한 필터링 수행 후 공간 연산을 수행한다. 그리드 필터링과 맵핑 테이블의 식별자 필터링을 통해 상대적으로 고비용인 공간 연산의 횟수를 감소시켜 질의 지연 시간을 감소시킨다.

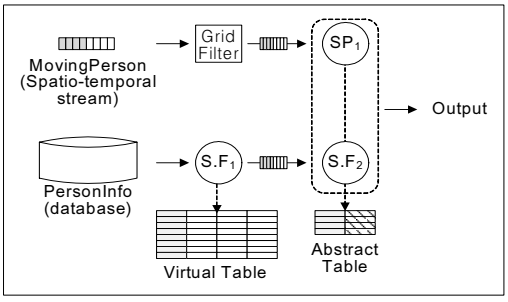
그림 7은 공간 속성에 대한 그리드 필터링 수행 과정을 나타내며, (a)는 입력 데이터가 이동 질의를 구성하는 중심 객체인 경우를, (b)는 입력 데이터가 이동 질의를 구성하는 중심 객체가 아닐 경우를 나타낸다.



[그림 7] 공간 속성에 대한 그리드 필터링 수행

3.4 질의 수행 계획

질의 수행은 방향성 그래프의 노드(node)와 에지(edge)로 구성되는 데이터 흐름도로 나타낸다. 노드는 연산자 박스를 나타내며, 에지는 연산자 박스를 연결하는 방향을 갖는 화살표 선을 말하며 연산자 박스를 통해 연산을 수행한 튜플은 에지를 따라 다음 연산자 박스로 이동하게 된다. 그림 8은 연결된 연산자 박스로 나타낸 그리드 영역 기반의 연산 공유 기법을 이용한 시공간 연속 질의 처리의 데이터 흐름도이다.



[그림 8] 데이터 흐름도

그림 8의 데이터 흐름도는 하나의 질의를 질의 수행 계획으로 나타냈다. 필터링 연산자 박스는 연산 수행 후 릴레이션 테이블을 생성하였으며 공간 연산자 박스와 단말 필터링 연산자 박스는 매핑 되어 시공간 데이터 스트림 객체에 대한 이동 질의를 수행한다.

4. 성능 평가

4.1 평가 환경

성능 평가에 사용된 환경은 인텔 펜티엄 3.00GHz CPU, 메모리는 4GB, 운영체제는 Fedora Linux 6 상에서 구동하였다. 개발 환경은 JAVA JDK 1.5 이며 테스트를 위한 시뮬레이션 툴은 C++/MFC로 구현하였다. 시뮬레이션 툴은 시공간 데이터 스트림을 생성하여 시스템으로 입력하는 시공간 데이터 스트림 발생 기능과 질의를 입력하는 질의 입력 기능을 제공한다. 시공간 데이터 스트림은 시뮬레이션 툴에 의해 초당 10,000건 발생하여 입력되고 평가에 사용되는 릴레이션 스키마 구조는 아래 표 2와 같다.

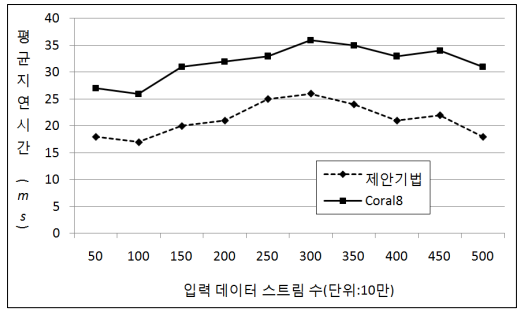
[표 2] 릴레이션 스키마 구조

MOVING_PERSON (ID INT, POS POINT, TIME TIMESTAMP)
PERSON_INFO (ID INT, NAME CHAR(20), AGE INT, FREEPASS BOOLEAN, FAVORITE CHAR(20), GID INT)

4.2 실험

4.2.1 데이터 스트림에 대한 질의 처리 시간 평가

성능 평가를 위해 시공간 데이터 스트림(MovingPerson)을 입력으로 상용 데이터베이스인 오라클에 기 구축된 이용객 정보(PersonInfo)와 융합 처리를 위한 예제 질의를 사용하였다. 예제 질의는 보호자와 피보호자의 거리가 일정 거리 이상 멀어지면 보호자에게 경고 메시지를 전송하는 질의로 그룹 식별자만 변경하여 12개를 등록하였다. 성능 비교는 제안 기법과 대표적인 상용 데이터 스트림 관리 시스템인 Coral8을 비교한다.



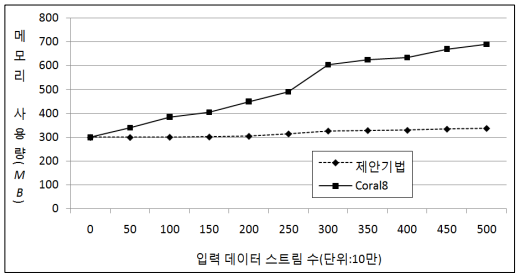
[그림 9] 질의 처리 시간 비용 평가

그림 9에서 Coral8은 데이터베이스의 릴레이션을 스트림화하기 위해 지속적으로 디스크에 접근하여 디스크 접근 비용이 발생한다. 또한, 릴레이션의 모든 데이터에 대해 시공간 데이터 스트림은 매 입력마다 조인 연산을

하는 문제점이 있다. 반면에 제안기법은 릴레이션과 시공간 데이터 스트림의 조인을 위해 먼저 비공간 속성에 대한 연산 공유 2단계 필터링을 수행한다. 따라서 릴레이션을 메모리에 로드 후 가상 릴레이션 형태로 캐싱하고 각 연산에 대해 필터링 되어 축소된 가상 릴레이션에 포함된 시공간 객체에 대해서만 조인 연산과 공간 연산을 수행하기 때문에 질의 지연 시간이 감소했다. 또한, 최종 종료 시간 역시 제안기법이 약 50초 정도 앞섰으며 약 66%의 성능 향상을 보였다.

4.2.2 유사 질의 처리 시 메모리 사용량 평가

제안기법은 유사 질의 등록 시 기 구축된 데이터베이스의 릴레이션으로부터 로드되는 비공간 속성에 대한 연산 공유를 수행한다. 이를 통해 데이터의 중복 적재를 방지하여 메모리의 불필요한 낭비를 해결할 수 있다. 평가를 위한 환경에서 메모리는 300MB를 먼저 할당한 상태로 수행하였다.

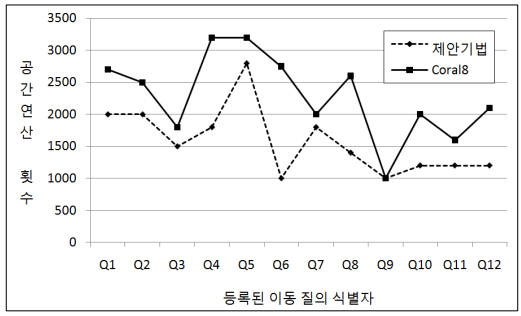


[그림 10] 유사 질의 처리에 따른 메모리 사용량 평가

그림 10에서 Coral8은 기 구축된 데이터베이스의 비공간 속성 정보에 대한 연산 시 데이터를 중복 적재하여 불필요한 메모리의 사용량이 증가하였으며 질의 지연에 따라 더욱 큰 폭으로 증가한 것을 볼 수 있다. 반면에 제안기법은 데이터의 중복 적재를 방지하여 메모리 사용량이 거의 증가하지 않았으며 질의 지연이 증가한 시점에서 약간의 메모리 사용량이 증가하였으나 질의의 지연이 완화되면서 할당된 메모리를 반환한 것을 볼 수 있다.

4.2.3 공간 연산 횟수에 대한 평가

제안 그리드 영역 기반 공간 연산과 비공간 속성에 대한 연산 공유 2단계 필터링 사용 시 공간 연산 횟수의 감소에 대한 성능을 평가한다. 제안기법은 비공간 속성에 대한 연산 공유 2단계 필터링을 사용하여 질의에 필요한 시공간 객체를 선별하여 조인 연산 및 공간 연산에 대한 횟수를 감소시킨다. 또한, 그리드 영역 기반 공간 연산을 통해 공간 연산이 포함하는 그리드 영역에 한하여 시공간 객체를 선별하기 때문에 공간 연산 횟수가 감소한다.



[그림 11] 유사 질의 처리에 따른 메모리 사용량 평가

그림 11에서 Coral8은 정확히 일치하는 조건을 갖는 연산에 대한 연산 공유 기법을 사용하기 때문에 유사한 조건을 갖는 공간 연산의 횟수가 감소하지 않았다. 반면에 제안기법은 그리드 분할 기법을 통해 공간 연산과 객체를 필터링하여 공유하였으며 비공간 속성에 대한 선 필터링을 통해 질의에 해당하는 그리드 내의 모든 시공간 객체가 공간 연산자 박스에 접근하기 전에 그리드 내에 불필요한 시공간 객체를 제거하여 실제 공간 연산 횟수를 감소시켰다. 따라서 동일한 시공간 연속 질의에서 Coral8 보다 연산 횟수가 감소한 것을 알 수 있다. 등록된 질의 9는 연산 횟수가 동일한 것을 볼 수 있는데 이는 그리드 내의 시공간 연속 질의의 영역에 해당하는 시공간 객체가 공간 연산에 필요한 것으로 구성되어 그리드 영역 밖으로 이동하지 않은 경우이다.

5. 결론

본 논문에서는 시공간 데이터 스트림에서 시공간 연속 질의 처리를 위한 그리드 영역 기반의 연산 공유 기법을 제안하였다. 제안기법은 고비용인 공간 연산의 연산 횟수 감소를 위해 시공간 객체의 위치와 시공간 연속 질의의 공간 연산 영역을 그리드 필터링을 통해 그리드 영역으로 분할하고 해당 그리드 셀의 질의와 객체를 공유하여 중복 연산 수행을 감소시켰으며 공간 연산뿐 아니라 비공간 연산의 중복 수행을 감소시키기 위해 조건이 일치하는 연산보다 동일한 속성에 대한 연산 조건을 갖는 연산을 공유하여 질의 처리 시 발생하는 지연 시간을 감소시켰다. 또한, 비공간 속성에 대한 연산 공유로 인해 데이터베이스로부터 데이터 중복 적재를 방지하여 메모리 이

용률을 높였다. 성능 평가에서는 질의 처리 지연 시간, 메모리 사용량, 공간 연산 횟수에 있어 제안 기법이 기존 연구보다 우수함을 보였다.

향후 연구는 시공간 데이터 스트림의 양이나 각 연산의 선택도 등의 실시간 정보를 통해 동적으로 연산자를 스케줄링 하여 질의 지연 시간을 감소시키는 기법에 대한 연구가 필요하다.

참고문헌

[1] B. Gedik and et al., "MobiEyes: Distributed processing of continuou sly moving queries on moving objects in a mobile system", Proc. of the International Conference on Extending Database Technology, 2004.

[2] H. Hu, J. Xu and D.L. Lee, "A generic framework for monitoring continuous spatial queries over moving objects" Proc. of the ACM SIGMOD International Conference on Management of Data, pp. 479-490, 2005.

[3] OGC, “OpenGIS Implemenation Specification for Geographic information - Simple feature access, Part1: Common Architecture”, www.opengeospatial.org, 2008.

[4] OGC, “OpenGIS Implemenation Specification for Geographic information -Simple feature access - Part1:SQL Option”, www.opengeospatial.org, 2008.

[5] Y. Tao, D. Papadias, J. Sun, "The TPR*-tree: an optimized spatio-temporal access method for predictive queries", Proc. of the International Conference on Very Large Data Bases, 2003.

[6] Lukasz Golab and M. Tamer Ozsu, “Issues in Data Stream Management”, In SIGMOD Record, Vol. 32, No. 2, pp. 5-14, June, 2003.

[7] 정원일, 김환구, “유비쿼터스 용응 서비스를 위한 공간 데이터 스트림 처리 플랫폼”, 한국산학기술학회논문지, 제11권, 제3호, pp. 906-913, 3월, 2010년.

[8] D.J. Abadi and et al., "Aurora: A new model and architecture for data stream management", VLDB J, Vol. 12, No. 2, pp. 120-139, 2003.

[9] A. Arasu and et. al., "STREAM: The Stanford Data Stream Management System" http://dbpubs.stanford.edu/pub/2004-20, 2004.

[10] J. Chen, D.J. DeWitt, F. Tian and Y. Wang, "NiagaraCQ: a scalable continuous query system for internet databases" Proc. of the ACM SIGMOD International Conference on Management of Data, pp. 379-390, 2000.

[11] F. Reiss and et al., "Data triage: an adaptive architecture for load shedding in TelegraphCQ" Proc. of the International Conference on Data Engineering, pp. 155-156, 2005.

[12] Coral8, "Coral8 Technology Overview", 2004-2008, http://www.coral8.com

[13] StreamBase, "The Eight Rules of Real-Time Stream Processing," 2008, http://www.streambase.com

정 원 일(Weonil Chung) **[정회원]**

- 1998년 2월 : 인하대학교 전자계산공학과(공학사)
- 2004년 8월 : 인하대학교 컴퓨터정보공학과(공학박사)
- 2004년 7월 ~ 2006년 7월 : 한국전자통신연구원 선임연구원
- 2007년 3월 ~ 현재 : 호서대학교 정보보호학과 교수

<관심분야>
데이터스트림, 이동객체, 시스템보안

김 영 기(Young-Ki Kim) **[준회원]**

- 2007년 2월 : 서원대학교 컴퓨터교육과(학사)
- 2008년 3월 ~ 현재 : 인하대학교 컴퓨터정보공학과 석사과정

<관심분야>
LBS, 데이터스트림, 공간데이터베이스