# 효율적인 센서 네트워크 관리를 위한 다중 연속 질의의 유사성 최소화*

조명현, 손진현
한양대학교 컴퓨터 공학과
{mhjo, jhson}@cse.hanyang.ac.kr

# Minimizing the Similarity of Multiple Continuous Queries for the Efficient Sensor Network Management

Myung Hyun Jo, Jin Hyun Son
Department of Computer Science and Engineering, Hanyang University

**요 약**

센서 네트워크의 급속한 성장에 따라 센서 네트워크의 효율적 관리를 위한 다양한 연구가 진행 중이다. 특히, 센서의 저 전력을 위한 다양한 기술들이 개발되고 있다. 본 논문은 센서에 전해지는 다중 연속 질의의 중복 성을 제거함으로써, 센서 네트워크의 효율적 관리를 제공할 수 있는 방법을 제안한다. 다중 연속 질의는 두 가지 단계로 최적화가 이루어진다. 먼저, 다중 연속 질의의 시간 속성 중복을 제거하기 위해 B+tree를 이용해 그룹 핑된다. 그룹 핑된 다중 연속 질의들은 연관 속성의 중복 여부 판단을 통해, 중복 성을 제거하여 재구성 된다. 그러므로 재구성된 다중 연속 질의가 센서 노드에 전해지게 되면, 센서는 중복된 결과를 전송하지 않기 때문에 센서 노드의 불필요한 전력을 낭비하기 않게 된다.

**키워드**: 연속 질의, 다중 질의 최적화, 센서 네트워크, 인덱스

## 1. 서 론

최근 유비쿼터스 컴퓨팅의 급속한 발전에 따라, 무선 센서 네트워크에 대한 다양한 연구와 개발이 진행되고 있다. 예를 들어, 사람이 살수 없는 오지의 동식물에 대한 데이터나 넓게 펼쳐진 해안의 조수 간만 차에 대한 데이터를 얻기를 위해 센서 네트워크 기술이 이용된다. 센서 네트워크에서 추출되는 무한한 데이터는 연속적이며 동적인 특성을 갖고 있는 스트림이다. 기존 데이터베이스 기술은 고정되고 변하지 않는 정적인 구조를 갖고 있기 때문에, 스트림을 처리하는데 한계를 갖고 있다. 그래서 데이터베이스는 연속적인 데이터를 추출하여 센서 네트워크의 모니터링을 할 수 있도록 연속 질의(Continuous Query)에 대한 메카니즘을 제공해야 한다.

연속 질의에 대한 연구는 센서 네트워크의 핵심적인 연구 분야로써, 다양한 연구가 진행되고 있다. 특히 센서 네트워크와 관련해 연속질의 최적화에 대한 연구가 활발히 진행 중이다. 본 논문은 센서의 저 전력을 위해, 연속 질의의 중복 성을 제거하는 최적화 기술을 제안한다.

연속 질의는 크게 데이터 변화에 따라 질의를 실행하는 연속 질의(Change-based Continuous Query)와 타이머에 따라 질의를 실행시키는 연속 질의(Timer-based Continuous Query)로 나뉘는 데[2], 본 논문은 타이머에 따라 질의를 실행시키는 연속 질의에 초점을 맞추었다. 타이머 기반의 연속 질의는 질의에 사용자가 추출하고자 하는 시간 범위를 지정할 수 있기 때문에, 센서들의 전력에 집적적인 영향을 준다. 예를 들어, 두 사용자가 동일한 질의를 동일한 시간 범위를 가지고 실행한다면, 센서들은 동일한 데이터를 두 번 전송해야 하기 때문에 불필요한 전력을 낭비하게 된다.

논문의 구조는 다음과 같다. 제 2장에서는 관련 연구로써 다중 질의 및 다중 연속 질의 최적화 방법에 대한 기존 연구와 본 논문을 비교, 분석 한다. 제 3장에서는 본 논문에서 사용할 연속 질의에 대해 정의한다. 제 4장에서는 다중 연속 질의를 그룹 핑하기 위한 기술을 제안하고, 그룹 핑된 다중 연속 질의를 재구성하기 위한 메카니즘을 기술한다. 제 5장에서는 결론을 맺는다.

* 본 연구는 대학 IT연구 센터 육성·지원 사업의 연구 결과로 수행되었음
* 본 연구는 한국 과학재단 목적 기초 연구 (R08-2003-000-10464-0) 지원으로 수행되었음

## 2. 관련 연구

연속질의는 가장 처음으로 SQL형태의 사용자 질의에 따라 스트림에서 문서를 필터링하기 위해 1992년 Terry에 의해 제안되었으며, 본 논문의 주제와 관련된 다중 질의 최적화 문제는 1988년 Sellis[1]에 의해 처음 시작되었다.

2001년 Chen이 제안한 NiagaraCQ[2]는 중복되는 서브 표현들을 그들만의 서명(signature)을 통해 그룹 핑을 시도 하였다. 또, 2002년에 Madden의 CACQ[4]는 NiagaraCQ[2]보다 좀 더 유연한 방법으로 다중 연속 질의를 그룹 핑하였다. 마지막으로, 2003년에 Yousuke[3]가 제안한 다중 연속 질의 최적화 방법이 본 논문의 다중 연속 질의 최적화 방법과 유사한데, Yousuke[3]는 다중 연속 질의의 시간 범위를 유사성 행렬을 이용해 클러스터 링 시킨 후, DAG에 의해 질의 계획(Query Plan)을 최적화 시켰다.

하지만, 아직까지 센서 네트워크의 저 전력을 위해 다중 연속 질의를 최적화 시킨 방법은 제안되지 않았다. 또, 기존 최적화 방법은 공통되는 서브 표현을 그룹 핑시키는 방법을 제안하였다. 이것은 서브 표현을 최적화한 것으로 중간 결과(Intermediate Result)를 공유하는 결과를 가져오지만, 질의 자체가 변경되지 않았기 때문에 최적화 능력이 없는 센서 노드는 중복된 결과를 보내게 된다. 그러므로 센서의 저 전력을 유지할 수 없다.

## 3. 연속 질의 정의

본 논문은 각 센서들이 추출 데이터에 대해 복잡한 SQL 연산은 못한다고 가정한다. 센서 노드가 직접 Join, Aggregate, Nested Query와 같은 복잡한 SQL 연산자를 처리한다면, 상당

한 에너지를 필요로 할 것이다. 그래서 본 논문은 복잡한 SQL 질의는 그림 1의 구조를 갖는 간단한 SQL질의로 변경된다고 가정한다. 다시 말해서, 그림 1은 연속 질의 최적화에 대한 간편성을 위해, SELECT-FROM-WHERE에서 WHERE절의 조건문은 오직 논리곱(conjunctive)들로만 구성되며, 논리합(disjunctive), 중첩된 질의(nested query), 집계(aggregation) 등의 SQL 연산자들은 포함되지 않는다고 가정한다. 그래서 그림 1에서 정의한 연속 질의는 오직 하나의 테이블(All_Sensors)만을 갖게 된다. 그리고 Time-Interval은 사용자가 질의에 대한 응답을 받고자 하는 시간 범위를 정의한다. 예를 들어, 사용자는 그림 2처럼 특정 날씨에 대한 위치 데이터를 특정 시간 범위에 모니터 링 할 수 있다.

```
SELECT *
FROM All_Sensors[Time-Interval start time, exit time]
WHERE sql
```

그림 1. 연속 질의 구문

```
SELECT location
FROM All_Sensors[Time Interval 2:00, 3:00]
WHERE temperature <20 AND temperature >= 10 AND
      humidity >40 AND atmosphere <30
```

그림 2. Query 1

#### 4. 다중 연속 질의 최적화 방법

본 장에서는 그림 1에서 정의한 연속 질의를 최적화하기 위한 방법을 제안한다. 연속 질의는 두 단계에 걸쳐 최적화 된다. 먼저, 시간 속성에 따라 그룹 핑 된다. 그 후, 그룹 핑된 연속 질의들은 중복 성을 제거하기 위해 재구성된다.

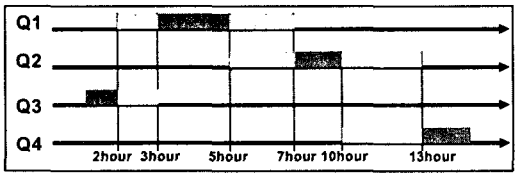


그림 3. Time-Interval 그래프

표 1. 3 | 대한 시간 영역의 Inverted File

| Time Interval | Query List |
|---|---|
| (1 , 2) | (Q3, 2) |
| (2 , 3) | (Q3, 2), (Q1, 3) |
| (3 , 5) | (Q1, 3) |
| (5 , 7) | (Q2, 1), (Q1, 3) |
| (7 , 10) | (Q2, 1) |
| (10 , 13) | (Q2, 1), (Q4, 3) |
| (13 , 15) | (Q4, 3) |

#### 4.1 연속 질의 그룹 핑

본 논문에서 제시한 연속 질의는 특정 시간 범위를 정의할 수 있도록 Time-Interval 속성을 제공 한다. 1장에서 설명한 것처럼 Time-Interval 속성은 센서 노드 전력에 가장 직접적인 영향을 끼치기 때문에, 중복된 영역은 반드시 제거 되 야 한다. 인덱스를 구성하는 기술 중 1차원에 대한 인덱스 방법은 B+tree, Hash등 다양한 방법이 있다. 하지만, Hash는 영역 질의(Range Query)를 할 수 없기 때문에, Time-Interval 인덱스에 따라 영역을 비교해서 질의를 삽입, 삭제해야 하는 구조에는 부적합하다. B+tree는 균형적인 구조와 순차 검색, 트리 검색 모두를 제공하기 때문에, Time-Interval 인덱스에 따라 다양하고 많은 양의 질의를 빠르게 삽입, 삭제해야 하는 구조에 적합하다. B+tree를 이용해 그룹 핑된 연속 질의들은 속성 개수에 따라 정렬되어 B+tree 리프 노드에 블록 형태로 저장된다. 특히 4.2절에서 설명할 연속 질의 재구성은 그룹 핑된 연속 질의들을 순차적으로 검색할 필요가 있는데, 여기서 B+tree 순차 검색의 특성이 유용하게 사용된다.

표 1은 그림 3처럼 도식화된 연속 질의의 시간 범위를 역 파일로 구성한 것이다. 그림 3에서 그림 5의 B+tree로 구성되는 것을 구체적으로 설명하기 위해 표1을 보인다. 예를 들어, 그림 2의 Query 1과 그림 4의 Query 3의 질의가 있다고 가정하자. Query 1과 Query2는 표1처럼 2시와 3시 사이에 온도와 습도에서 중복 성을 갖고 있기 때문에, 중첩된 결과를 얻게 된다. 즉, Query 3의 결과를 필터링하여 Query 1에 대한 결과를 생성할 수 있다면, 2시와 3시 사이에 센서는 두 번의 결과를 생성하지 않아서 불필요한 에너지를 소모하지 않게 된다.

```
SELECT location
FROM All_Sensors[Time-Interval 2:00, 7:00]
WHERE temperature <30 AND humidity >30
```

그림 4. Query 3

그림 3은 그림 5처럼 B+tree 구조로 인덱스 된다. B+tree의 리프 노드는 질의 아이디와 질의 속성의 개수를 블록 단위로 저장한다. 특히, 질의 리스트는 질의 속성 개수에 따라 정렬되어 저장된다. 질의 아이디는 시간 영역에 따라 질의를 분류하기 위한 것이고, 질의 속성 개수는 4.2절에서 설명할 질의 재구성을 위한 것이다. 자세한 내용은 4.2절에서 설명하겠다.

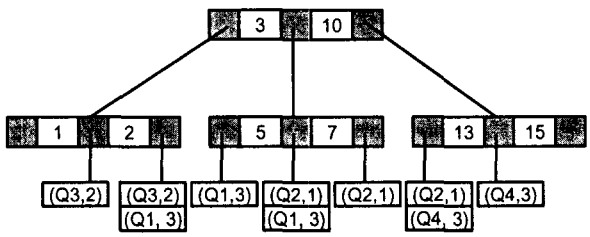


그림 5. 3 B+ tree 인덱스

표 2. 연속 질의 속성의 중복 여부 판단의 위한 그림 3 - 구성 매트릭스

| | 온도 | 습도 | 기압 | 속도 |
|---|---|---|---|---|
| Q1 | 1 | 1 | 1 | |
| Q2 | | | | 1 |
| Q3 | 1 | 1 | | |
| Q4 | | 1 | 1 | 1 |

#### 4.2 연속 질의 재구성

3장에서 정의한 것처럼 연속질의는 질의의 논리곱만으로 구성되기 때문에, 질의의 중복 성 체크를 위해서는 CNF를 구성해야하는 과정은 필요하지 않다. 그래서 질의의 속성 영역이 중복되는 것만을 체크하면 된다. 속성 영역의 중복 여부는 다음 질의에 따라 조사된다.

· 연속 질의는 중복되는 속성을 갖고 있는 가?

· 중복된 속성은 중복된 영역을 갖고 있는 가?

첫 번째, 연속 질의는 포함(Inclusion), 겹침(Interleaving), 배제(Exclusion)의 세 가지 경우를 가지고 속성의 중복 여부를 확인한다. 표 2는 중복 속성을 파악하기 위해, 표1의 질의가 포함하고 있는 속성을 보여주는 매트릭스이다. '1'로 표시된 부분은 질의가 그 속성을 조건 절에 포함하고 있다는 것을 의미한다. Q1과 Q3은 온도와 습도를 조건 절에 포함하고 있고, Q3의 모든 속성이 Q1의 모든 속성 내부에 포함되어 있기 때문에 Q3의 조건 절이 Q1의 조건 절보다 큰 것을 알 수 있다. 그래서 Q3은 Q1을 포함하는 관계를 갖고 있고, 포함 관계는 다음과 같이 정의된다. 이것은 교환법칙이 성립되지 않는다.

$$Q^i \supset Q^j$$

위 예제에서 그림 2의 Query 1와 그림 4의 Query 3이 "Query 3 ⊃ Query 1"의 포함관계를 갖는다. 겹침 관계는 표2의 Q3과 Q4의 관계를 나타낸다. Q3과 Q4는 모두 습도라는 속성을 조건 절에 포함하고 있고, 그 이외에 속성은 서로 다르게 구성된다. 이것은 다음과 같이 정의하며, 교환 법칙이 성립된다.

$$Q^i \cap Q^j$$

배제관계는 표2의 Q1과 Q2의 관계를 나타낸다. Q1과 Q2는 어떤 속성도 중복되는 것이 없기 때문에, 배제 관계를 보인다. 이것은 다음과 같이 정의 하며, 교환 법칙이 성립되지 않는다.

$$Q^i \varnothing Q^j$$

두 번째, 중복된 속성들은 속성 영역의 중복에 따라 포함(Inclusion), 역 포함(Inverse Inclusion), 겹침(Interleaving), 배제(Exclusion)의 네 가지로 구성된다. 포함관계는 두 질의의 조건 절에 속성 값의 범위가 포함관계를 보여줄 때를 말한다. 예를 들어, Query 3과 Query 1의 온도 속성은 Query 3이 Query 1을 포함하는 것을 보여준다. 지면 관계상 다른 속성들의 관계성에 대한 설명은 생략한다. 표 3은 속성 관계를 보여주며, 모든 관계는 교환 법칙이 성립되지 않는다.

표 3. 중복 속성들 간의 관계성

| Relationship | Definition | Example |
|---|---|---|
| 포함 | $Q_k^i \supset Q_k^j$ | "humidity>30" ⊃ "humidity>50" |
| 역 포함 | $Q_k^i \subset Q_k^j$ | "humidity>50" ⊂ "humidity>30" |
| 겹침 | $Q_k^i \cap Q_k^j$ | "humidity>30 & humidity <50" ∩ "humidity>40 & humidity <60" |
| 배제 | $Q_k^i \varnothing Q_k^j$ | "humidity>60" Ø "humidity<50" |

위에서 정의한 질의와 속성의 관계성은 서로 고려해야 하는 요소들이기 때문에, 총 12가지의 경우에 대해서 유사성을 제거해야 한다. 하지만, 모든 경우에 정확히 중복된 기술을 적용할 수 없다. 센서는 여러 가지 속성을 갖기 때문에, 속성 관계 또한 다양하게 정의될 수 있다. 만일 속성들이 다양하게 관계성을 갖게 된다면, 정확히 중복되지 않는다. 예를 들어, 표2의 Q3과 Q1의 온도는 포함관계를 갖고, 습도는 배제관계를 갖게 된다면, 온도는 중복을 갖더라도 습도 속성 때문에 질의를 두 번 할 수 밖에 없게 된다. 본 논문은 정확히 중복되지 않는 경우는 중복 효과를 보기 위해 어느 정도 손실을 필요로 하기 때문에 고려하지 않는다. 속성이 정확히 중복 성을 갖는 경우는 다음과 같다.

*{k| k>=0 and k<=duplicate($Q^i$,$Q^j$)}

*{t| t>=i+1 and t<=allNumberOfQueryInGrouping()}

· $Q^i \supset Q^j$ and $\forall k(Q_k^i \supset Q_k^j)$

· $Q^i \supset Q^j$ and $\forall k(Q_k^i \subset Q_k^j)$

· $Q^i \supset Q^j$ and $\forall k(Q_k^i \cap Q_k^j)$

· $\forall t(Q^i \varnothing Q^j)$

질의가 포함 관계를 갖고 질의의 중복된 모든 속성들이 포함관계를 갖는 다면, 질의 영역이 큰 질의가 영역이 작은 질의를 대신할 수 있다. 질의의 중복된 모든 속성들이 역 포함관계를 갖는 다면, 질의 개수가 많은 질의를 질의 개수가 적은 질의에 맞추어 쪼개어 중복된 영역을 제거한다. 겹침 관계 또한 역 포함 관계처럼 쪼개어 중복영역을 제거할 수 있다. 마지막으로, 특정 질의가 그룹 핑된 모든 질의에 대해 모두 배제 관계를 갖는다면, 모든 질의는 특정 질의의 중복 영역을 제거하기 위해 수정 되 어 중복 성을 제거한다.

위에서 정의한 정확히 중복되는 영역의 처리는 그림 6처럼 수도 코드(Pseudo Code)로 나타내어 질수 있다. 이것은 실행되는 질의리스트를 구하는 것을 목적으로 한다. 3장에서 B+tree의 리프 노드를 질의 속성 개수에 따라 순차적으로 저장한 것은 속성 개수가 적은 것이 큰 질의 영역을 포함하기 때문이다. 그래서 그림 6은 속성 개수가 적은 질의에서 속성 개수가 많은 질의로, 순차적으로 중복 여부를 판단한다. 중복된 질의는 대체나 분할을 통해 실행 질의 리스트에 저장되며, 질의에 의한 결과를 공유시키기 위해 사용자 질의의 아이디도 저장된다. 그림 6은 지면상 알고리즘의 일부만을 보여준다.

```
for(j = i+1; j < query_collection.length-1;j++){
    query1 = query_collection(j);
    attribute_relationship_matrix[];
    duplicate_attribute[] = findDuplicateAttribute(query1,query2);
    for(int i=0;i<duplicate_attribute.length;i++){
        attribute_relationship_matrix[i] =
            relationship(query1.rangeOfAttribute(duplicate_attribute[i]),
                         query2.rangeOfAttribute(duplicate_attribute[i]));
    }
    if(query2.attribute inclusion query1.attribute){
      if(all attributes relationship is inclusion){
            check=true;
            exec_collection.insert(query2,(query2,query1));
            query_collection.delect(i);
            query_collection.delect(j);
      }else if(all attributes relationship is inverse_inclusion){
            check=true;
            exec_collection.insert(query2,query1);
            split_query(query2,query1);
            query_collection.delete(i);
      }else if(all attributes relationship is interleaving){
            check=true;
            exec_collection.insert(query2,query1);
            split_query(query2,query1);
            query_collection.delete(i);
      }
    }else if(query2.attribute exclusion query1.attribute){
      exclusion_check++;
    }
}
```

그림 6. 다중 연속 질의의 최적화 방법

## 5. 결론

본 논문은 센서 네트워크에서 불필요하게 센서 노드의 전력을 낭비하는 것을 줄이기 위해, 다중 연속 질의의 중복 성을 제거하는 최적화 방법을 제안한다. 이것은 기존 방법과 달리 실행 질의가 분할되어 질의된다.

## 6. 참고 문헌


[1] Sellis, T. "Multiple Query Optimization." *ACM TODS* 13(1):23-52, March 1988.

[2] Chen, J., DeWitt, D., Tian, F., and Wang, Y., NiagaraCQ: A Scalable Continuous Query System for Internet Database. In SIGMOD(2000)

[3] Yousuke W., Hiroyuki., A Multiple Continuous Query Optimization Method Based on Query Execution Pattern Analysis. In DASFFA 2004

[4] Madden, S., Shah, M., Hellerstein, J., and Raman, V., Continuously Adaptive Continuous Quries over Streams. In SIGMOD(2002)