무선 센서 네트워크에서 다중 질의 최적화 기법을 이용한 에너지 효율적인 질의 처리 기법

(Energy Efficient Query Processing based on Multiple Query Optimization in Wireless Sensor Networks)

이 유 원 † 정 은 호 † 함 덕 민 † 이 충 호 ††
(Yu Won Lee) (Eun Ho Chung) (Deok Min Haam) (Chung Ho Lee)
이 용 준 ††† 이 기 용 †††† 김 명 호 †††††
(Yong Jun Lee) (Ki Yong Lee) (Myoung Ho Kim)

요 약 무선 센서 네트워크는 작은 크기의 센서 노드들과 베이스 스테이션으로 구성된다. 센서 노드들은 에너지를 지속적으로 공급받기 어려운 특성을 가지고 있기 때문에 제한된 에너지를 최대한 효율적으로 사용해야 한다. 센서 네트워크에서 정보를 얻을 때는 질의를 작성하는데, 여러 개의 질의가 센서 네트워크에 요청되었을 때 이들의 실행을 최적화시키면 센서 노드의 에너지 소비량을 줄일 수 있다. 본 논문에서는 센서 네트워크에 여러 개의 질의가 요청될 때, 질의 간의 포함관계를 활용하여 질의 실행을 최적화시키는 기법을 제안한다. 제안하는 기법은 질의가 새로 요청될 때, 센서 네트워크에서 실행 중인 질의들 중 새로 요청된 질의의 결과를 조합해낼 수 있는 질의들의 집합을 찾아낸다. 이 집합은 질의들의 샘플 주기, 프로젝션되는 어트리뷰트들, 그리고 조건절을 기준으로 구성된다. 이 집합에 속한 질의들의 결과로부터 새 질의를 센서 네트워크에서 실행시켰을 때와 동일한 결과를 유도해낼 수 있으면, 새 질의를 센서 네트워크에 전달하지 않고, 이 집합에 속한 질의들의 결과를 재활용하여 베이스 스테이션에서 결과를 조합해낸다.

키워드 : 다중 질의 최적화, 무선 센서 네트워크, SQL

Abstract A wireless sensor network is a computer network which consists of spatially distributed devices, called sensor nodes. In wireless sensor networks, energy efficiency is a key issue since sensor nodes must resides upon limited energy. To retrieve sensor information without dealing with the network issues, a sensor network is treated as conceptual database on which query can be requested. When multiple queries are requested for processing in a wireless sensor network, energy consumption

· 본 연구는 정보통신부 및 정보통신연구진흥원의 IT신성장동력핵심기술개발사업의 일환으로 수행하였음. [2007-S022-01, USN 미들웨어 플랫폼 기술 개발]
This work was supported by the IT R&D program of MIC/IITA. [2007-S022-01, Development of USN Middleware Platform Technology]

† 학생회원 : 한국과학기술원 전산학과 학생
ywlee@dbserver.kaist.ac.kr
ehjung@dbserver.kaist.ac.kr
dmhaam@dbserver.kaist.ac.kr
†† 정 회 원 : 한국전자통신연구원(ETRI) 텔레매틱스/USN 연구단 RFID/USN미들웨어연구팀 선임연구원
leech@etri.re.kr
††† 정 회 원 : 한국전자통신연구원(ETRI) 텔레매틱스/USN 연구단 RFID/USN미들웨어연구팀 책임연구원
yjlee@etri.re.kr
†††† 정 회 원 : 한국과학기술원 전산학과 연구조교수
kiyong.lee@gmail.com
††††† 종신회원 : 한국과학기술원 전산학과 교수
mhkim@dbserver.kaist.ac.kr

논문접수 : 2008년 2월 11일
심사완료 : 2008년 11월 11일

정보과학회논문지: 데이타베이스 제36권 제1호(2009.2)

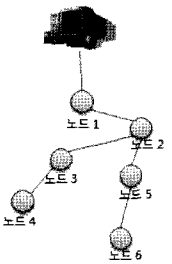
can be significantly reduced if common partial results among similar queries can be effectively shared. In this paper, we propose an energy efficient multi-query processing technique based on the coverage relationship between multiple queries. When a new query is requested, our proposed technique derives an equivalent query from queries running at the moment, if it is derivable. Our technique first computes the set of running queries that may derive a partial result of the new query and then test if this set covers all the result of the new query attribute-wise and tuple-wise. If the result of the new query can be derived from the results of executing queries, the new query derives its result at the base station instead of being executed in the sensor network.

Key words : multiple query optimization, wireless sensor networks, SQL

1. 서 론

무선 센서 네트워크란 여러 장소에 배치된 작은 크기의 센서 노드들과 베이스 스테이션으로 구성된 무선 네트워크를 말한다. 각 센서 노드는 다른 센서 노드들과 정보를 주고 받을 수 있도록 무선 통신 기능을 갖추고 있으며, 온도나 습도, 소리, 압력 등 여러 가지 물리적인 현상들을 관찰할 수 있도록 작은 크기의 센서를 장착하고 있다. 또한, 센서 노드는 주로 작은 크기의 배터리를 에너지 공급원으로 사용하기 때문에 에너지 공급이 제한적이라는 특성을 가진다. 베이스 스테이션은 센서 네트워크에 대한 사용자의 요청을 받아들이거나 네트워크에서 처리된 결과를 사용자에게 전달해주는 게이트웨이 역할을 담당하고, 센서 노드보다 강력한 계산 기능을 가지며 에너지도 지속적으로 공급받을 수 있다[1].

무선 센서 네트워크는 여러 개의 센서 노드들로 구성된 분산 컴퓨팅 환경이기 때문에 사용자가 직접 네트워크를 제어하기는 쉽지 않다. 따라서 최근에는 TinyDB [2,3]나 Cougar[4]처럼 사용자가 쉽게 무선 센서 네트워크를 제어하거나 네트워크로부터 원하는 정보를 얻어낼 수 있도록 도와주는 미들웨어들이 개발되고 있다. 이런 종류의 미들웨어는 주로 무선 센서 네트워크를 하나의 가상 테이블로 이루어진 데이타베이스로 생각하며, 이 테이블의 각 행은 센서 노드들이 센서를 사용하여 읽어들인 값으로 구성된다. 예를 들어, TinyDB는 센서 네트워크를 sensors라는 테이블로 표현하는데, 만약 센서 네트워크가 온도와 빛, 습도를 채집할 수 있는 센서 노드들로 구축되어 있으면, 이 센서 네트워크를 표현한 sensors 테이블의 각 행은 그림 1처럼 센서 노드의 번호와 온도, 빛, 습도 값으로 구성된다. 이와 같이, 센서 네트워크를 테이블로 바라보면 관계형 데이타베이스 시스템에서 SQL과 같은 선언적인 질의 언어를 사용하여 정보를 추출하는 것과 유사하게 센서 네트워크에서도 선언적인 질의 언어를 통해 원하는 정보를 손쉽게 얻을 수 있다는 이점이 있다. 예를 들어, 그림 1의 센서 네트워크에서 35 이상의 온도 값을 채집한 센서 노드로부터 해당 센서 노드의 번호와 온도 값을 4초마다 얻으려면, TinyDB에서는 그림 2의 질의를 작성하면 된다.



(a) 센서 네트워크

nodeid	temperature	light	Humidity
1	102	221	129
2	103	237	216
...	...	...	...

(b) sensors 테이블

그림 1 센서 네트워크와 가상 테이블

```
q1: SELECT nodeid, timestamp, temperature
    FROM sensors
    WHERE temperature >= 35
    SAMPLE PERIOD 4s
```

그림 2 센서 네트워크 질의

사용자가 센서 네트워크에 다수의 질의를 요청했을 때, 각 질의를 독립적으로 실행시키지 않고 질의 간의 상호 관계를 활용하면 에너지를 효율적으로 사용하여 질의들을 실행시킬 수 있다. 예를 들어, 센서 네트워크에서 그림 2의 q_1이 실행 중인 상태일 때, 사용자가 질의 "q_2: SELECT nodeid, timestamp, temperature FROM sensors WHERE temperature >= 40 SAMPLE PERIOD 4s"를 새로 요청하면 이 질의는 센서 네트워크에서 직접 실행되지 않아도 그림 3처럼 q_1의 결과를 활용하여 베이스 스테이션에서 결과를 유도해낼 수 있다. 베이스 스테이션은 센서 노드와는 달리 강력한 계산 능력을 가지고 있으며 에너지도 충분히 공급받을 수 있기 때문에 질의의 결과를 유도하는 비용은 크게 고려하지 않아도 된다. 그러나 센서 네트워크의 질의 실행에 관한 지금까지의 연구들은 대부분 단일 질의의 실행을 어떻게 최적화할 것인가에 초점이 맞추어져 있었다[5]. 이런 연구들은 센서 네트워크에서 다수의 질의를 실행시킬 경우 여러 질의의 상호 관계를 통해 얻을 수 있는 이점을 충분히 활용하지 못하는 한계를 가지고 있다.

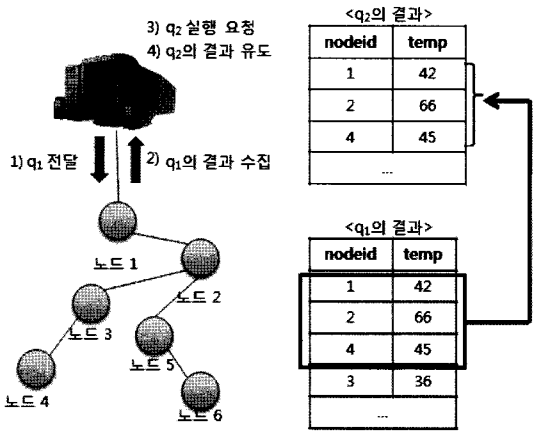


그림 3 질의 결과 유도

본 논문은 질의들 간의 포함 관계를 활용하여 센서 네트워크에서 질의 실행 시 소모되는 에너지의 양을 감소시키는 질의 실행 기법을 제안한다. 본 논문에서 제안하는 기법은 사용자가 다수의 질의를 요청했을 때, 센서 네트워크에서는 최소한의 사용자 질의들을 실행시키거나 필요한 경우에는 질의를 병합하여 실행시키고 나머지 질의들은 베이스 스테이션에서 결과를 유도해낸다. 질의의 결과를 유도할 때는 센서 네트워크에서 실행 중인 질의들이 보내온 스트림 형태의 결과 튜플들을 실시간으로 조합해낸다.

본 논문의 구성은 다음과 같다. 2장에서는 센서 네트워크에서의 질의 실행 최적화 기술과 관련된 연구들을 소개하고, 3장에서는 문제 정의 및 가정에 대해 설명한다. 그리고 4장에서는 본 논문에서 제안하는 다중 질의 최적화 기법에 대해 설명하고 5장에서는 실험을 통해 본 논문에서 제안한 기법의 효율성을 검증한다.

2. 관련 연구

지금까지 센서 네트워크에서 질의를 처리할 때 소모되는 에너지를 줄이기 위한 기법들이 여러 분야에 걸쳐 연구되었다. [6]에서는 센서 네트워크에서 에너지를 효율적으로 사용하여 질의를 처리할 수 있도록 도와주는 라우팅 프로토콜이 제안되었다. 또한 [7]에서는 네트워크 내 질의 처리 기술을 통해 에너지 소모를 줄여 보려는 시도가 있었으며 [8]에서는 질의 결과의 근사값을 만들어내는 기법이 제안되었다. 하지만 이런 연구들은 오랜 기간에 걸쳐 실행되는 단일 질의의 최적화에 초점이 맞추어져 있다.

[2-4]에서는 사용자가 센서 네트워크를 쉽게 사용할 수 있도록 도와주는 센서 네트워크용 미들웨어가 제안되었다. 이런 종류의 미들웨어들은 센서 네트워크를 추상화시켜 하나의 가상 테이블로 바라볼 수 있도록 해주는 데이타 모델을 정의하였으며 이 모델에 적합한 질의 언어도 제안하였다. 뿐만 아니라 질의가 센서 노드에서 실행될 때 질의의 실행을 최적화시키는 기법도 제안되었다. 그러나 이런 기법들도 단일 질의의 실행만 최적화시킬 뿐 다수의 질의가 처리될 때 질의 실행을 최적화시키려는 노력은 부족했다.

관계형 데이타베이스 분야에서는 여러 개의 질의가 데이타베이스 시스템에 요청되었을 때 질의들의 실행을 최적화시키기 위해 [9]와 같은 기법들을 제안했다. 그러나 관계형 데이타베이스를 위한 기법들은 다음과 같은 이유 때문에 센서 네트워크에 적용시키기가 어렵다. 첫 번째, 센서 네트워크는 정보를 스트림 형태로 생성시키며 그 값이 계속 변화하는데 반해 관계형 데이타베이스의 정보는 디스크에 저장되어 있다. 두 번째, 센서 네트워크에 요청되는 질의에는 센서 노드가 정보를 수집하는 주기인 "SAMPLE PERIOD"라는 구절이 추가된다. 세 번째, 관계형 데이타베이스를 위한 질의 최적화 기법의 목적은 질의의 응답 속도나 계산 비용을 줄이기 위한 것인 반면 센서 네트워크에서 필요한 질의 최적화 기법은 센서 노드의 에너지 소비량을 줄일 수 있는 것이다. 네 번째, 관계형 데이타베이스에서의 질의 재활용 기법들은 주로 다중 테이블 조인, 집단화 질의와 같은 다양한 형태의 질의를 처리하기 위한 단순한 수준의 재작성 기법을 다루고 있는 반면, 센서 네트워크에서는 좀 더 복잡한 수준의 재작성 기법을 사용하더라도 에너지 감소를 위해 가능한 질의의 수를 줄이는 것이 중요하다.

데이타 스트림 관리 시스템(DSMS) 분야에서도 센서 네트워크에 요청되는 여러 질의들의 실행을 최적화하려는 노력이 있었다. [10]에서 제안된 Fjords는 센서 프록시를 통해 센서 네트워크에 요청된 여러 개의 질의들을 실행시킨다. 하지만, Fjords는 데이타 스트림을 효율적으로 처리하는 데 초점이 맞추어져 있으며 센서 네트워크의 에너지를 효율적으로 이용하려는 노력은 크지 않다.

[11]에서는 센서 네트워크에 여러 개의 질의가 요청되었을 때 센서 네트워크의 특성을 고려하여 질의들의 실행을 최적화하는 기법이 제안되었다. 하지만 [11]은 공간과 관련된 질의들만 최적화시키는 기법이기 때문에 일반적인 질의에는 적용시키기 어렵다. [5,12]에서는 일반적인 질의를 최적화시키기 위해 질의 병합에 기반한 기법을 제안했다. 질의 병합에 기반한 기법들은 질의가 병합됐을 때 기존 질의에는 없던 질의 영역이 추가되거나, 질의가 사용자가 지시한 실행 주기보다 자주 실행되는 한계를 지니고 있다.

3. 문제 정의 및 가정

무선 센서 네트워크는 하나의 가상 테이블로 이루어진 데이타베이스로 생각할 수 있다. 본 논문에서는 이 테이블의 이름을 TinyDB처럼 sensors라고 하겠다. sensors 테이블에는 기존의 관계형 데이타베이스의 테이블과 구별되는 몇 가지 차이점이 있다. 센서 네트워크를 표현한 테이블은 기존의 데이타베이스와는 달리 테이블이 실제로 디스크에 저장되어 있지 않고 시간에 따라 변화하는 값을 노드에서 측정하여 실시간으로 읽어들이는 가상의 테이블이다. 또한 끊임없이 변화하는 값을 관찰하기 위해 일회성 질의가 아닌 주기적으로 같은 일을 반복하여 수행하는 질의가 요청되고 실행된다. 질의가 오랜 기간 동안 시스템에 머물기 때문에 여러 질의들 사이의 관계를 이용한 질의 최적화가 가능하다.

새로운 질의가 요청되었을 때 센서 네트워크에서 이미 실행 중인 질의들의 결과를 조합하여 새 질의의 결과를 유도해낼 수 있는 경우가 있다. 따라서 센서 네트워크에서 실행 중이던 기존 질의들의 결과를 조합하여 새 질의의 결과가 유도 가능한지를 판단하고 유도가 가능한 경우에는 어떻게 유도할 수 있는지를 알아낼 수 있다면, 새 질의를 센서 네트워크에서 직접 실행하지 않고 베이스 스테이션에서 결과를 유도해냄으로써 센서 노드의 에너지를 절약할 수 있다. 예를 들어 그림 4와 같은 질의들이 주어져 있고 nodeid와 timestamp가 키 어트리뷰트라고 가정하자. 질의 q_4는 "(SELECT * FROM $q_1 \bowtie q_2$) $\cup$ (SELECT * FROM q_3 WHERE temperature >=30 AND humidity<40)"과 동일한 질의이다. 따라서 이런 경우에는 센서 네트워크에서 모든 질의를 직접 실행시키기보다 q_1과 q_2, q_3만을 실행하고 q_4의 결과는 베이스 스테이션에서 q_1, q_2, q_3의 결과로부터 유도해 내면 센서 노드의 에너지를 절약할 수 있다.

```
q1: SELECT nodeid, timestamp, temperature
    FROM sensors
    WHERE temperature >= 35
    SAMPLE PERIOD 4s
q2: SELECT nodeid, timestamp, humidity
    FROM sensors
    WHERE humidity < 40
    SAMPLE PERIOD 5s
q3: SELECT nodeid, timestamp, temperature, humidity
    FROM sensors
    WHERE temperature <35 AND humidity<45
    SAMPLE PERIOD 20s
q4: SELECT nodeid, timestamp, temperature, humidity
    FROM sensors
    WHERE temperature >=30 AND humidity<40
    SAMPLE PERIOD 20s
```

그림 4 유도 가능 질의

이와 같은 질의 최적화를 위해 본 논문에서는 sensors 테이블에 대한 사용자 질의를 실행중인 질의에 대한 질의로 변형하는 방법을 제안한다. 우선, 질의에 대하여 다음과 같은 가정을 하겠다.

- 모든 사용자 질의는 sensors 테이블에 대한 셀렉션과 프로젝션 질의이다. 즉, 관계 대수식(relational algebra expression)이 프로젝션과 셀렉션으로 정의된 질의만 고려한다. 센서 네트워크의 질의는 주로 정보 수집용으로 사용되기 때문에 조인처럼 복잡한 형태가 아닌 SELECT절과 WHERE절로 구성된 단순한 형태의 질의가 사용된다.
- 프로젝션 리스트는 키 어트리뷰트인 nodeid와 timestamp를 포함한다. nodeid는 sensor 테이블의 튜플을 생성한 센서 노드의 ID를 나타내며, timestamp는 sensor 테이블의 튜플이 생성된 시각을 나타낸다. 키 어트리뷰트를 제외한 나머지 어트리뷰트 간의 종속관계는 없다고 가정한다.
- 셀렉션 절에 기술된 조건절에는 어트리뷰트들 간의 직접적인 비교가 없다고 가정한다. sensors 테이블의 각 어트리뷰트들은 물리적인 현상들을 표현한 값이기 때문에 단위가 다르다. 따라서, 어트리뷰트들 간의 직접적인 비교는 어렵다.
- 사용자 질의의 셀렉션 조건절은 리터럴의 논리곱이고, 리터럴은 임의의 어트리뷰트 x와 상수 c에 대한 관계식이다. 이것을 문맥 자유 문법(context free grammar)로 표현하면 다음과 같다.

condition → condition AND literal | literal
literal → (c<x) | (x<c) | (x=c) | (c<=x) | (x <=c) | ε
x → 어트리뷰트
c → 상수

질의 결과를 유도하기 위해 베이스 스테이션으로 전달된 튜플들을 조인했을 때, 그 결과가 올바르도록 만들기 위해서는, 즉 무손실 조인 분해(lossless-join decomposition)[13]를 위해서는 키 어트리뷰트가 반드시 전달된 질의 결과에 포함되어 있어야 한다.

본 논문에서의 질의의 결과는 고정된 데이타가 아닌, 연속적으로 데이타를 생성하는 데이타 스트림임에 유의하라. 일반적으로 데이타 스트림들을 조인하기 위해서는 조인의 대상이 되는 데이타를 제한하기 위해 각 스트림에 대한 윈도우를 지정해야 한다[10]. 그러나 본 논문에서 조인은 프로젝션 절에 키 어트리뷰트인 nodeid와 timestamp가 포함된 질의 간에만 이루어지며, 이 경우 조인 결과는 새로운 timestamp 시점에 발생된 각 질의

표 1 용어 정의

Q	사용자가 요청한 모든 질의들의 집합.
Q_E	실행중인 질의 집합.
Q_D	유도될 질의 집합.
A	sensors 테이블의 모든 어트리뷰트 집합.
$q = (A_q, C_q)$	사용자 질의이며 sensors에 대한 셀렉션 프로젝션 질의. 프로젝션 리스트와 셀렉션 조건절의 쌍으로 표현할 수 있다. 예) $q = (A_q, C_q)$ SELECT A_q FROM sensors WHERE C_q
f	센서 네트워크에서 실행 중인 사용자 질의에 대한 셀렉션 프로젝션 조인 질의.
d	사용자 질의를 분해한 질의 (sensors에 대한 질의). 또는 이와 동일한 결과를 가지는 실행중인 사용자 질의에 대한 질의 (실행 중인 사용자 질의에 대한 셀렉션 프로젝션 조인 질의인 f_i의 유니온).
α(C)	조건 C에서 쓰이는 모든 어트리뷰트의 집합.
R(C, x)	조건 C에서 어트리뷰트 x와 관련된 조건, (lx, ux), [lx, ux] 등으로 표현 가능.
$R_L(C, x)$	조건 C에서 어트리뷰트 x의 하한(lower bound) 조건.
$R_U(C, x)$	조건 C에서 어트리뷰트 x의 상한(upper bound) 조건.
x, y, z	어트리뷰트.
Q_x	실행중인 질의들을 서로 조인시켰을 때 어트리뷰트 x의 값을 알 수 있는 모든 조합. $P_x = \{q \in Q_E \mid x \in A_q$ 또는 $R(C_q, x) = (x=$상수$)\}$ 라고 했을 때, $Q_x = P_x \cup \{ q_i \bowtie q_j \mid q_i, q_j \in Q_E - P_x$ 이고 $R(C_{qi} \wedge C_{qj}, x) = (x=$상수$)\}$

들의 결과들만을 조인함으로써 스트림의 형태로 얻을 수 있다. 따라서 이 후 논문에서는 윈도우에 대한 설명은 생략한다.

질의의 유도 가능 조건 및 유도 알고리즘을 설명하기 전에 우선 표 1 같이 용어를 정의하겠다. 표 1에서 셀렉션 프로젝션 조인 질의란 관계 대수식이 셀렉션과 프로젝션, 조인으로만 정의된 질의를 의미한다.

4. 제안 방법

사용자가 센서 네트워크에 요청하는 질의는 sensors 테이블에 관한 프로젝션과 셀렉션 질의다. 이 질의를

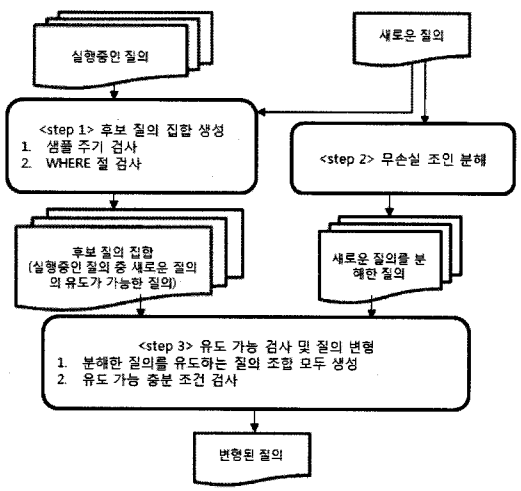


그림 5 제안 방법 개요

sensors 테이블이 아닌 실행중인 질의에 대한 질의로 변형시킬 수 있다면 새로운 질의를 직접 실행시키지 않고 실행중인 질의의 결과로부터 유도해낼 수 있다. 이를 위해 본 논문에서 제안하는 방법은 다음과 같다. 우선 센서 네트워크에서 실행되고 있는 질의들 중 새 질의의 결과를 유도해낼 수 있는 가능성을 가진 질의들을 선택하기 위해 후보 질의 집합을 생성한다. 이 집합에 속하지 않는 질의들은 새 질의의 결과를 유도하는 데 전혀 기여할 수 없는 질의들이다. 그 후, 새로운 질의를 무손실 조인 분해하고 분해된 각 질의는 후보 질의들로부터 유도될 수 있도록 변형한다. 변형된 질의는 후보 질의에 대하여 셀렉션, 프로젝션, 조인, 유니온을 하는 질의로 제한하겠다. 본 절에서 설명할 기법은 그림 5로 요약할 수 있다.

4.1 후보 질의 집합 생성

센서 네트워크에서 사용되는 질의의 형태는 관계형 데이타베이스에서 주로 사용되던 질의와는 다른 특성을 가진다. 우선, 센서 네트워크에 요청되는 질의는 센서 네트워크를 구성하는 각 센서들이 자신의 센서를 사용하여 주변의 상황을 인지할 주기를 정한다. 예를 들어, 자료 수집 주기인 SAMPLE PERIOD가 2s로 지정되어 있었다면, 각 센서는 매 2초마다 센서를 사용하여 주변의 상황을 수집한다. 또 다른 특성으로, 센서 네트워크에 요청된 질의는 매 SAMPLE PERIOD마다 자료를 수집하여 사용자에게 정보를 전달하기 때문에 질의 결과는 스트림 데이타로 생각할 수 있다.

이런 특성으로 인해, 센서 네트워크에 새로 요청된 질

의의 결과를 만들어 낼 수 있는 후보를 선택할 때는 우선적으로 SAMPLE PERIOD를 고려해야 한다. 본 논문에서는 새로 요청된 질의의 SAMPLE PERIOD가 t라면, Q_E에서 SAMPLE PERIOD가 t의 약수인 질의들을 선별한다. 선별된 질의들은 새로 요청된 질의의 SAMPLE PERIOD마다 자신이 채집한 센서 정보들을 새로운 질의에게 전달해 줄 수 있는 질의들이다. 예를 들어, 새 질의 q_{new}의 SAMPLE PERIOD가 8초라면, q_{new}는 매 8초마다 질의의 결과를 사용자에게 전달해야 한다. 이 경우, Q_E의 원소들 중에서 8초마다 q_{new}에게 질의 결과를 전달해 줄 수 있는 질의들은 SAMPLE PERIOD가 8의 약수인 1, 2, 4, 8초인 질의들이다. 샘플 주기가 새로운 질의의 주기의 약수인 질의만을 선별했다면 더 이상 샘플 주기가 필요하지 않으므로 앞으로 모든 예제 질의에서는 샘플 주기를 생략하도록 하겠다.

질의 q의 결과 튜플이 새 질의 q_{new}의 조건절을 절대 만족시킬 수 없으면, q는 q_{new}의 결과를 조합하는 데 가여할 수 없다. 따라서 현재 실행되고 있는 질의들 중 임의의 결과 튜플이 새 질의의 셀렉션 조건절을 만족시킬 수 있는 질의들을 선별해야 한다. 이를 위해, 두 질의의 조건절을 논리곱으로 합친 $C_q \wedge C_{qnew}$가 만족될 수 있는 조건인지 검사한다. 예를 들어, 두 질의의 합쳐진 조건이 "light > 15 AND light < 10"처럼 절대 만족될 수 없는 조건이면 이런 질의는 후보 질의 집합에 포함시키지 않고, 반대로 두 질의의 조건을 합친 형태가 "light > 10 AND light < 20"처럼 만족될 수 있는 형태이면 해당 질의를 후보 질의 집합에 포함한다.

알고리즘 1은 실행중인 질의들 중 새로운 질의의 결과에 기여할 가능성이 없는 질의를 걸러내는 알고리즘으로, 3번째 줄에서 샘플 주기를 검사하고 5번째 줄에서 WHERE 절을 검사한다. 이 알고리즘의 실행이 종료되면 QC_E에는 센서 네트워크에서 실행중인 질의들 중 새 질의의 결과를 유도해낼 수 있는 가능성을 지닌 질의들만 남고, QC_X에는 QC_E의 질의들 중 서로 조인하였을 때 어트리뷰트 x의 값을 알 수 있는 모든 질의 조합이 남는다. 알고리즘 1에서 사용하는 Q_x를 관리하는 방법은 4.3.2절에서 자세히 설명한다.

```
알고리즘 GenerateCandidateQuery(qnew)
/* QE와 Qx에 대하여 검사를 하여 QCE와 QCx 생성 */
1  QCE = QE, QCx = Qx //초기화
2  For each q ∈ QE ⋈ Qx
3    IF qnew's sample period is not multiple of q's sample
     period
4      Remove q From QCE and QCx
5    ELSE IF Cq ∧ Cqnew is not satisfiable
6      Remove q From QCE and QCx
```

알고리즘 1 후보 질의 집합 생성 알고리즘

4.2 질의의 무손실 조인 분해

테이블의 무손실 조인 분해란 주어진 테이블을 몇 개의 테이블로 분해했을 때 분해된 테이블을 조인하면 원래의 테이블을 얻을 수 있는 경우를 말한다. 이와 유사하게 질의 q_{new}를 식 (1)처럼 $d_1, d_2, \cdots d_m$로 분해한 후 다시 조인했을 때 q_{new}의 원래 결과와 동일한 결과를 얻을 수 있는 경우를 질의의 무손실 조인 분해라고 하겠다.

$$q_{new} = d_1 \bowtie d_2 \bowtie \cdots \bowtie d_m \quad (1)$$

무손실 조인이 가능하기 위해서는 조인되는 어트리뷰트에 키 어트리뷰트가 포함되어야 한다. 본 논문에서는 모든 질의가 키 어트리뷰트를 프로젝션 한다고 가정하였으므로 질의의 무손실 조인이 가능하다.

여러 질의의 결과를 조인한 결과는 각 질의의 조건절을 모두 만족하는 하나의 질의의 결과와 같다. 또한 이 결과에는 각 질의가 프로젝션하는 어튜리뷰트가 모두 포함되어 있다. 따라서 새로운 질의 $q_{new}= (A_{qnew}, C_{qnew})$를 무손실 조인 분해한 질의를 $d_i = (A_{di}, C_{di})$, i=1,…,m, 라고 한다면 $d_1, d_2, \cdots, d_m$는 $\cup A_{di} = A_{qnew}$와 $\wedge C_{di} = C_{qnew}$를 만족해야 한다.

한 질의를 무손실 조인 분해하는 방법은 여러 가지가 존재한다. 본 논문에서는 문제의 복잡도를 줄이기 위해 각각의 분해된 질의 di가 프로젝션하는 어트리뷰트의 수를 최소화시킨다. 이를 위해 각 질의 d_i가 키 어트리뷰트를 포함하여 3개 이하의 어트리뷰트 만을 프로젝션하도록 질의를 분해하겠다. 즉, 새 질의에 나타나는 어트리뷰트들이 nodeid, timestamp, $x_1, x_2, \cdots, x_m$이면 새로운 질의의 프로젝션 어트리뷰트와 셀렉션 조건절을 분해하여 $d_{x1}, d_{x2}, \cdots, d_{xm}$를 만든다. 다음은 질의 분해의 예제이다.

예제) 새로운 질의가 다음과 같이 주어져 있다고 하자.

q_{new} = SELECT nodeid, timestamp, x, y FROM sensors WHERE 10<x<20 AND 5<z

위의 질의에서 사용되는 어트리뷰트는 x, y, z이며 질의를 다음과 같이 분해할 수 있다.

d_x = SELECT nodeid, timestamp, x FROM sensors WHERE 10<x<20

d_y = SELECT nodeid, timestamp, y FROM sensors

d_z = SELECT nodeid, timestamp FROM sensors WHERE 5<z

위에서 d_x는 다음과 같은 두 질의 중 하나로 대체할 수도 있으며, 이와 비슷하게 d_y와 d_z도 다른 질의로 대체 할 수 있다.

d_x' = SELECT nodeid, timestamp, x FROM sensors WHERE 10<x<20 AND 5<z

d_x" = SELECT nodeid, timestamp, x FROM sensors WHERE 10<x<20 AND 0<z

예제에서 보듯이 q_{new}의 셀렉션 조건절을 어떻게 분해하는가에 따라 여러 가지 방법으로 qnew를 무손실 조인 분해시킬 수 있다.

새로운 질의 q_{new}를 어트리뷰트 x에 대해 분해한 질의를 d_x="SELECT A_{dx} FROM sensors WHERE C_{dx}" 라고 하자. 분해된 질의 d_x의 A_{dx}는 식 (2)와 같이 정의된다. 또한, 분해된 질의 dx의 셀렉션 조건절은 식 (3)을 만족하여야 한다. 즉, d_x는 그 결과가 "SELECT A_{dx} FROM sensors WHERE R(C_{qnew}, x)"의 결과의 부분집합이고 "SELECT A_{dx} FROM sensors WHERE C_{qnew}"의 결과의 초집합(superset)인 질의다.

$$A_{dx}\begin{cases}\{\text{nodeid, timestamp, x}\}, \\ \text{if } x \in A_{qnew} \\ \{x\}, \\ \text{if } x \in \alpha(C_{qnew}) - A_{qnew}\end{cases} \quad (2)$$

$$\forall x \in A, \quad C_{qnew} \Rightarrow C_{dx} \Rightarrow R(C_{qnew}, x) \quad (3)$$

4.3 분해된 질의의 변형

4.2 절에서는 새로운 질의를 무손실 조인 분해하는 방법에 대해 살펴보았다. 분해된 질의 d_x는 식 (2)와 식 (3)을 만족하는 질의로 sensors에 대한 프로젝션 셀렉션 질의이다. 본 절은 분해된 질의 d_x를 후보 질의에 대한 질의로 변형하는 방법을 제시한다.

후보 질의에 대한 셀렉션 프로젝션 조인 질의를 f_i라고 하면 $d_x = f_1 \cup f_2 \cdots \cup f_k$를 만족하는 f_1, …, f_k를 찾아야 한다. 이를 위해서는 dx의 결과의 부분집합을 유도할 수 있는 질의 f_i를 모두 찾은 후, $d_x = f_1 \cup f_2 \cdots \cup f_k$를 만족하는지 확인하면 된다. 다음은 d_x를 변형하는 예제이다.

예제) Q_x가 다음과 같이 주어져 있고, 새로운 질의를 어트리뷰트 x에 대해 분해한 질의인 d_x의 결과가 s_x 결과의 초집합이고 w_x결과의 부분집합이라고 하자.

w_x = SELECT nodeid, timestamp, x FROM sensors WHERE 0<x≤10

s_x = SELECT nodeid, timestamp, x FROM sensors WHERE 0<x≤10 AND 0<y<10

Q_x = { q_1: SELECT nodeid, timestamp, x FROM sensors WHERE 0<x<10 AND 5<y,

q_2: SELECT nodeid, timestamp, x, y FROM sensors WHERE 0<y<7,

$q_3 \bowtie q_4$: SELECT nodeid, timestamp FROM sensors WHERE x≤10

⋈ SELECT nodeid, timestamp FROM sensors WHERE 10≤x }

이런 상황에서 d_x의 부분집합인 질의 f_1, f_2, f_3를 아래와 같이 만들 수 있으며, 각 f_i는 Q_x에 대한 질의이다.

f_1 : SELECT nodeid, timestamp, x FROM q_1

f_2 : SELECT nodeid, timestamp, x FROM q_2 WHERE 0<x≤10

f_3 : SELECT nodeid, timestamp, x=10 FROM $q_3 \bowtie q4$

질의 f_1, f_2, f_3를 유니온 시키면 다음과 같은 d_x를 얻을 수 있다.

d_x = SELECT nodeid, timestamp, x FROM sensors WHERE x=10 OR (0<x<10 AND 0<y)

= $f_1 \cup f_2 \cup f_3$

4.3.1 변형된 질의의 조건

[14,15]에서는 테이블 R_1, R_2, … , R_t로 구성된 데이타베이스가 있을 때, 이 데이타베이스로부터 유도된 테이블 E_1, E_2, … , E_n 이 저장된 형태로 존재하는 환경에서 주어진 질의 E_0이 유도된 테이블로부터 계산 가능한가에 대한 충분조건을 다루고 있다. 각 E_i들을 센서 네트워크에서 후보 질의들이라고 생각하고, 주어진 질의 E_0를 분해된 질의 d_x라고 보면 [14,15]의 유도 가능 충분 조건을 본 논문의 문제에 적용할 수 있다.

테이블 R_1, R_2, … , R_t로 구성된 데이타베이스가 있으면, 유도된 테이블 E_1, E_2, … , E_n은 질의로 표현할 수 있다. 질의 q와 E_1, E_2, … , E_n가 모두 R_1, R_2, … , R_t에 대한 프로젝션 셀렉션 조인 질의라고 하자. 본 논문에서는 데이타베이스에 테이블이 하나라고 가정하였으므로 t=1이고 R_1=sensors라고 하면, $E_0=d_x=(A_{dx}, C_{dx})$와 $E_i=(A_{Ei}, C_{Ei})$는 sensors에 대한 프로젝션 셀렉션 질의로 볼 수 있다. 이 때, E_1, E_2, … , E_n에 대한 프로젝션 셀렉션 조인 질의 f_1, f_2, … , f_k가 $d_x = f_1 \cup f_2 \cup \cdots \cup f_k$를 만족하는 충분 조건은 다음과 같다. 임의의 f_i는 $E_{i1} \bowtie E_{i2} \bowtie \cdots \bowtie E_{is}$에 대한 프로젝션과 셀렉션 질의라고 하자.

조건 1. 튜플 포함 관계

d_x의 결과는 E_1, E_2, … , E_n의 결과의 합집합에 포함되어야 한다. 즉, 프로젝션 전의 q의 결과는 프로젝션 전의 E_1, E_2, … , E_n의 결과의 합집합의 부분집합이어야 한다. 이를 논리식으로 표현하면 다음과 같다.

$$C_{dx} \Rightarrow C_{E1} \vee C_{E2} \vee \cdots \vee C_{En}$$

조건 2. 어트리뷰트 포함 관계

임의의 f_i의 결과 튜플에서 A_{dx}의 모든 어트리뷰트의 값을 알 수 있어야 한다. 어떤 어트리뷰트의 값을 아는 방법은 f_i의 결과 튜플에서 직접 해당 어트리뷰트의 값을 얻는 방법과 결과 튜플에 해당 어트리뷰트가 없는 경우에는 셀렉션 조건절을 참고하여 어트리뷰트의 값을 임의로 계산하는 방법이 있다. x를 프로젝션하지 않는 임의의 f_i에서 x의 값을 알아 내려면 "x=상수" 또는 "$a \le x \le a$" 형태의 조건이 $C_{Ei1} \wedge \cdots \wedge C_{Eis}$ 중 직접 리터럴로 존재하여야 한다.

조건 3. 튜플 선발 가능성

임의의 $E_{i1} \bowtie E_{i2} \bowtie \cdots \bowtie E_{is}$의 결과에서 d_x에 속하는 튜플만을 선별할 수 있어야 한다. 즉, $E_{i1} \bowtie E_{i2} \bowtie \cdots \bowtie E_{is}$의 결과에는 d_x의 결과에 속하지 않는 튜플들이 포함되어 있을 수 있기 때문에 이 튜플들 중에서 C_{dx}를 만족하는 튜플들을 선택할 수 있어야 한다. 조인의 결과로 만들어진 어떤 튜플이 C_{dx}를 만족하는지 판단하기 위해서는 C_{dx}를 판단하는 데 필요한 모든 어트리뷰트 $y \in \alpha(C_{dx})$에 대하여 그 값을 알고 있거나 조인의 결과가 $R(C_{dx}, y)$를 만족하여야 한다.

$E_{i1} \bowtie \cdots \bowtie E_{is}$가 만족하는 조건은 $C_{Ei1} \wedge \cdots \wedge C_{Eis}$이다. 임의의 f_i의 셀렉션 조건절은 $E_{i1} \bowtie \cdots \bowtie E_{is}$의 결과에서 값을 알 수 있는 어트리뷰트에 대한 조건만 가능하므로 값을 알 수 없는 어트리뷰트 y에 대하여 f_i가 만족하는 조건은 $R(C_{Ei1} \wedge \cdots \wedge C_{Eis}, y)$이다. 따라서 임의의 f_i가 d_x의 결과에 속하기 위해서는 $\alpha(C_{dx})$ 중 그 값을 알 수 없는 모든 어트리뷰트 y에 대하여 $R(C_{Ei1} \wedge \cdots \wedge C_{Eis}, y) \bowtie R(C_{dx}, y)$이 언제나 참이어야 한다. 예를 들어, 질의 d_x 및 E_1, E_2, f_1이 다음과 같이 주어졌다고 가정하자.

예제) d_x : SELECT nodeid, timestamp FROM sensors WHERE 0<x<20 AND y<50

E_1 : SELECT nodeid, timestamp, y FROM sensors WHERE x<10

E_2: SELECT nodeid, timestamp FROM sensors WHERE 0<x

f_1: SELECT * FROM $E_1 \bowtie E_2$ WHERE y<50

$E_1 \bowtie E_2$로부터 d_x에 해당하는 튜플을 걸러내기 위해서는 어트리뷰트 x와 y의 값을 알아야 하지만 f_1의 결과 튜플에서는 y의 값은 알 수 있는 반면 x의 값은 알 수 없다. 그러나 $E_1 \bowtie E_2$은 "SELECT nodeid, timestamp, y FROM sensors WHERE 0<x<10" 와 동치이므로 f_1의 결과는 조건 0<x<10을 만족하고 $R(C_{dx}, x)=(0<x<20)$ 역시 만족한다. 따라서 $E_1 \bowtie E_2$의 결과에서 q의 결과에 속하는 튜플만을 뽑아낼 수 있다.

조건 4. 무손실 셀프 조인(Lossless self-join)

조인의 결과가 가짜 튜플을 만들지 않기 위해서는 조인되지 않는 모든 어트리뷰트의 값이 조인 되는 어트리뷰트에 종속되어야 한다. 여기서는 모든 질의가 키 어트리뷰트를 프로젝션 한다고 가정하였으므로 이 조건은 언제나 만족되며 이 조건에 대해서는 앞으로 언급하지 않겠다.

조건 5. 충분조건

조건 2~4를 만족하는 f_i의 결과는 d_x의 결과의 부분집합이며, 이런 f_1, f_2, … , f_k를 모두 유니온한 결과인 $f_1 \cup f_2 \cup \cdots \cup f_k$의 결과는 d_x의 결과를 포함하여야 한다. 조건 2~4를 만족하는 f_1, f_2, … , f_k에 대하여 f_i의 셀렉션 조건절이 C_i라고 했을 때 $C_i \wedge C_{Ei1} \wedge \cdots \wedge C_{Eis}$를 C_{fi}라고 하면 다음의 조건을 만족하여야 한다.

$$C_{dx} \Rightarrow C_{f1} \vee C_{f2} \vee \cdots \vee C_{fk}$$

조건 5를 만족하면 조건 1도 만족하므로 조건 1은 조건 5에 포함된다고 할 수 있다.

본 절을 정리하면 다음과 같다. 임의의 f_i가 식 (4)와 식 (5)와 같이 주어졌다고 하면 A_{fi}와 C_{fi}는 식 (6)과 같이 정의할 수 있다.

$$\begin{aligned} f_i &= \text{SELECT } A_{dx} \text{ FROM } E_{i1} \bowtie E_{i2} \bowtie \cdots \bowtie E_{is} \text{ WHERE } C_i \\ &= \text{SELECT } A_{dx} \text{ FROM (SELECT } \cup A_{Eij} \text{ FROM sensors WHERE } \wedge C_{Eij}) \text{ WHERE } C_i \\ &= \text{SELECT } A_{dx} \text{ FROM sensors WHERE } C_i \wedge C_{Ei1} \wedge \cdots \wedge C_{Eis} \end{aligned} \quad (4)$$

$$C_i = \wedge_{y \in \alpha(Cdx)} R(C_{dx}, y) \quad (5)$$

$$\begin{aligned} A_{fi} &= A_{Ei1} \cup A_{Ei2} \cup \cdots \cup A_{Eis} \\ C_{fi} &= C_i \wedge C_{Ei1} \wedge C_{Ei2} \wedge \cdots \wedge C_{Eis} \end{aligned} \quad (6)$$

식 (4)~(6)과 같이 정의된 f_1, …, f_k가 식 (7)을 만족하기 위해서는 조건 2, 조건 3, 조건 5를 만족하여야 한다.

$$d_x = f_1 \cup f_2 \cup \cdots \cup f_k \quad (7)$$

$\forall f_i, \forall y \in A_{dx}, y \in A_{fi}$ or $R(C_{fi}, y) = (y=\text{상수})$ 조건 (1)

$\forall f_i, \forall y \in \alpha(C_{dx}) - A_q, C_{fi} \Rightarrow R(C_{dx}, y)$ 조건 (2)

$C_{dx} \Rightarrow C_{f1} \vee C_{f2} \vee \cdots \vee C_{fk}$ 조건 (3)

4.3.2 변형된 질의의 생성 방법

이번 절에서는 분해된 질의 d_x가 주어졌을 때 조건 2

와 조건 3을 만족하는 모든 f_i를 찾는 방법을 제시한다. 후보 질의에 대한 프로젝션 셀렉션 조인 질의 f_i가 조건 2와 조건 3을 만족한다는 것은 f_i의 결과가 d_x의 결과의 부분집합이라는 뜻이다.

조건 2와 조건 3을 만족하는 모든 f_i를 찾으려면 후보 질의의 모든 조합을 검사해야 하기 때문에 검사해야 하는 f_i의 수를 줄이는 것이 효율적이다. 분해된 질의 d_x가 키 어트리뷰트를 제외하면 프로젝션되는 어트리뷰트가 한 개 이하이고 하나의 어트리뷰트에 대한 조건절을 만족하는 질의임을 고려하면 임의의 f_i가 조인하는 사용자 질의의 수를 제한할 수 있다.

정리 1

식 (7)을 만족하는 $f_1, \cdots, f_k$를 찾기 위해서는 조건 2와 조건 3을 만족하는 모든 f_i중 두 개 이하의 후보 질의를 조인하는 fi만을 고려하면 된다.

증명

<경우 1> 어트리뷰트 x가 새로운 질의의 프로젝션되는 어트리뷰트인 경우

f_i가 조건 2와 조건 3을 만족시키기 위해서는 x의 값을 알아야 한다. 하나의 질의에서 x의 값을 알 수 있는 경우는 이 질의가 x를 프로젝션하거나 셀렉션 조건절에 "x=상수" 형태의 조건이 존재할 때뿐이다. 혼자서는 x의 값을 알 수 없는 질의를 두 개 이상 조인하여 x의 값을 알 수 있는 경우는 임의의 상수 c에 대하여 조인되는 두 개의 질의 q_i와 q_j가 각각 "x>=c"라는 조건과 "x<=c" 라는 조건을 가지고 있는 경우뿐이다. 상한과 하한, 두 개의 조건만 필요하므로 세 개 이상의 질의의 조인은 고려하지 않아도 된다.

<경우 2> 어트리브튜 x가 새로운 질의의 조건절에만 있는 어트리뷰트인 경우

f_i는 언제나 조건 2를 만족한다. f_i가 조건 3을 만족시키기 위해서는 x의 값을 알거나 f_i가 $R(C_{qnew}, x)$ 조건을 만족시켜야 한다. 어트리뷰트 x의 값을 알 수 있는 모든 f_i는 위의 경우와 동일하므로 x의 값을 모르는 경우에 대해서만 고려해 볼 수 있다. $R(C_{qnew}, x)$ 조건은 x의 상한값 u와 하한값 l로 표현할 수 있다. x의 값을 알 수 없는 질의가 단독으로 조건 3을 만족하는 경우는 조건절에서 x에 관한 상한값과 하한값이 [l, u]에 포함되는 경우이다. 단독으로 $R(C_{qnew}, x)$을 만족할 수 없는 질의들을 조인하여 $R(C_{qnew}, x)$을 만족하는 질의를 얻기 위해서는 두 질의 qi와 qj가 각각 "x>=a" 형태의 조건과 "x<=b" 형태의 조건을 가지고 있고 a와 b가 [l, u]에 포함되는 경우이다. x의 값을 제한하려면 x의 상한값과 하한값을 알아야 하는데 이를 위해서는 최대 두 개의 질의만 필요하다. (증명 끝)

질의 d_x가 유도 가능한지 알기 위해서는 조건 2와 조건 3을 만족하는 모든 질의의 조합을 찾은 후 이를 합쳤을 때 조건 5를 만족하는지 테스트 하면 되는데 조건 2와 조건 3을 만족하는 모든 질의의 조합을 찾기 위해서는 Q_x를 계산하여야 한다. 왜냐하면 Q_x는 정의 상 Q_E 중 단일 질의에서 어트리브튜 x의 값을 알 수 있는 경우와 두 개의 질의를 조인하여 x의 값을 알 수 있는 경우의 집합으로 x의 값을 알 수 있는 모든 f_i의 집합과 같기 때문이다.

새로운 질의가 요청될 때 마다 필요한 어트리뷰트에 대하여 Q_x를 계산하기 보다는 새로운 질의를 센서 네트워크에서 직접 실행시킬 때 마다 Q_x를 업데이트 하여, 나중에 요청되는 질의들이 유도 가능한지 판단할 때 활용할 수 있도록 한다. 알고리즘 2는 이에 대한 알고리즘이다. 1~3줄은 새로운 질의로부터 알 수 있는 모든 어트리뷰트 x에 대해 새 질의를 Q_x에 추가시키는 과정이고 4~17줄은 새로운 질의와 실행 중인 다른 질의를 조인하여 어트리뷰트 x의 값을 알 수 있는 경우에 조인된 질의를 Q_x에 추가시키는 과정이다.

```
알고리즘 Query Execution Preprocessing (qnew)
/* 새로운 질의가 실행될 때마다 호출됨 */
/* QE와 Qx를 업데이트 */
1  Add qnew to QE
2  For each attribute x ∈ Aqnew
3    Add qnew to Qx
4  For each attribute x ∈ α(Cqnew)-Aqnew
5    If R(Cqnew, x) is (x=constant)
6      Add qnew to Qx
7    ELSE
8      IF RL(Cqnew, x) is (x<=c)
9        Find all q ∈QE s.t. RU(Cq, x) is (x>=c)
10         Add q⋈qnew to Qx
11     END IF
12     IF RU(Cqnew, x) is (x>= c)
13       Find all q ∈ QE s.t. RL(Cq, x) is (x<=c)
14         Add q⋈qnew to Qx
15       END IF
16   END IF
17 END FOR
```

알고리즘 2 Q_x 업데이트 알고리즘

어트리뷰트 x가 새로운 질의의 프로젝션 리스트에 있는 어트리뷰트인 경우 조건 2~3을 만족하는 모든 f_i는 Q_x와 동일하지만 어트리뷰트 x가 새로운 질의의 조건절에만 있는 어트리뷰트인 경우에는 Q_x뿐만 아니라 x의 값을 알 수는 없지만 조건 2~3을 만족하는 f_i도 모두 찾아야 한다. 즉, x의 값을 알 수는 없지만 식 (6)에서

정의된 C_{fi}에 대하여 $R(C_{fi}, x) \Rightarrow R(C_{dx}, x)$를 만족하는 모든 f_i를 찾아야 한다.

$R(C_{fi}, x)$에서 x에 관한 상한값과 하한값을 u_{fi}, l_{fi}라고 하고 $R(C_{dx}, x)$에서 x에 관한 상한값과 하한값을 u_{dx}와 l_{dx}이라고 하면 l_{fi}, u_{fi}, l_{dx}, u_{dx}는 $l_{dx} \le l_{fi} \le u_{fi} \le u_{dx}$를 만족해야 한다. 후보 질의 중 x의 값을 알 수는 없지만 x에 관한 하한값 조건 l_{fi}가 $l_{dx} \le l_{fi} \le u_{dx}$인 모든 질의의 집합을 L_X라고 하고, 후보 질의 중 x의 값을 알 수는 없지만 x에 관한 상한값 조건 u_{fi}가 $l_{dx} \le u_{fi} \le u_{dx}$인 모든 질의의 집합을 U_X라고 하자. 두 집합의 질의를 조인하는 모든 경우를 찾으면 어트리뷰트 x의 값을 알 수는 없지만 조건 3을 만족하는 모든 f_i를 찾을 수 있다. 단, 두 집합 L_X와 U_X에 공통으로 속하는 질의가 있다면 이 질의는 단독으로 조건 3을 만족하므로 다른 질의와 조인시키지 않는다.

4.3.3 충분 조건 검사

앞 절에서는 그 결과가 d_x의 결과의 부분집합인 f_1, …, f_k를 모두 찾았다. 식 (7) $d_x = f_1 \cup \cdots \cup f_k$를 만족하는지 알기 위해서는 조건 5를 검사하여야 한다. 조건 5는 $C_{dx} \Rightarrow C_{f1} \vee \cdots \vee C_{fk}$이 언제나 참이어야 한다는 것이다. C_{dx}는 $C_{qnew} \Rightarrow C_{dx} \Rightarrow R(C_{qnew}, x)$를 만족하는 임의의 조건이고 C_{fi}는 $C_{fi} \Rightarrow R(C_{qnew}, x)$를 만족하므로 이를 정리하면 $C_{qnew} \Rightarrow C_{dx} \Rightarrow C_{f1} \vee \cdots \vee C_{fk} \Rightarrow R(C_{qnew}, x)$이 된다. 이 조건에서 검증되지 않은 조건은 $C_{qnew} \Rightarrow C_{f1} \vee \cdots \vee C_{fk}$이다. 본 절은 $C_{qnew} \Rightarrow C_{f1} \vee \cdots \vee C_{fk}$이 언제나 참인지 검사하는 방법에 대해 설명한다. 가정에 의하여 사용자 질의의 셀렉션 조건절은 리터럴의 논리곱이며 리터럴은 임의의 어트리뷰트 x와 상수 c에 대한 관계식이다. C_{fi}는 사용자 질의의 셀렉션 조건절 논리곱이므로 $C_{qnew} \Rightarrow Cf1 \vee \cdots \vee C_{fk}$는 DNF 형태로 표현된다. DNF 논리식의 유효성을 검사하는 문제는 NP-complete이기 때문에 이를 해결하기 위해서는 휴리스틱 알고리즘이 필요하다. 본 논문에서는 이 문제에 대한 휴리스틱 알고리즘인 다면체 커버링(Hypercube covering) 알고리즘[16]을 사용한다.

어떤 질의의 셀렉션 조건절은 각각의 어트리뷰트를 축으로 하는 |A|차원의 공간에서 직각다면체로 나타낼 수 있다. 직각다면체는 모든 어트리뷰트에 대한 상한과 하한으로 정의되는 영역이다. "$C_{qnew} \Rightarrow C_{f1} \vee \cdots \vee C_{fk}$"이 언제나 참이라는 것은 |A| 차원의 공간에서 C_{qnew}의 영역을 C_{f1}, …, C_{fk}가 커버할 수 있느냐와 같다. 본 논문에서는 이를 판단하기 위해서 [16]에서 제안한 휴리스틱 알고리즘을 수정하여 사용한다. 이 방법은 C_{qnew}의 영역에서 남은 영역이 없을 때까지 C_{fi}가 차지하는 영역을 차례로 삭제해 나간다. 마지막에 남은 영역이 없으면 주어진 조건식은 언제나 참이다.

우선 어떤 영역에서 주어진 조건식이 차지하는 영역을 제외하고 그 나머지 부분을 분할하는 알고리즘을 생각해 보자. 예를 들어 영역 (0<x<10)(0<y<10)(0<z<10)에서 영역 (3<=x)(x<=5)(5<=z)을 삭제한다고 하자. 삭제하고 남은 부분은 (3<=x) 을 만족하지 않는 영역과 (x<=5)를 만족하지 않는 영역, (5<=z)를 만족하지 않는 영역으로 분할이 된다. 이 남은 세 영역을 서로 겹치는 부분이 없게 분할하기 위해서는 (3<=x) 을 만족하지 않는 영역과 (3<=x) 을 만족하지만 (x<=5)를 만족하지 않는 영역, (3<=x)(x<=5)를 만족하지만 (5<=z)를 만족하지 않는 영역으로 나누면 된다. 즉, 어떤 영역 C에서 영역 $B_1 \wedge B_2 \wedge \cdots \wedge B_t$를 삭제 한다고 하면 1부터 t까지의 정수 i에 대하여 $B_1 \wedge \cdots \wedge B_{i-1}$를 만족하지만 B_i는 만족하지 않는 영역이 분할되면서 생성되고 분할된 영역의 개수는 t개이다. 알고리즘 3은 이에 대한 알고리즘이다.

```
알고리즘 Delete(Left, C)
/* 영역 Left에서 영역 C가 차지하는 부분을 제외한 남은 영
   역의 집합을 반환한다. */
/* S와 C는 "(어트리뷰트) 부등호 (상수)" 형태인 리터럴들의
   논리곱이다. */
1  IF (C == NULL)
2    RETURN Φ
3  IF (Left ∧ C is FALSE)
4    RETURN { Left }
5  Let C be (B ∧ C') where B is a literal
6  RETURN { (Left ∧ not B) } ∪ Delete(Left ∧ B, C')
```

알고리즘 3 조건 영역 삭제 알고리즘

마지막 줄은 $B_1..B_{i-1}$을 만족하지만 B_i를 만족하지 않는 부분을 알고리즘 실행 결과에 추가하고 $B_1 \cdots B_i$를 만족하는 영역에 대하여 알고리즘을 재귀적으로 호출한다.

알고리즘 4는 주어진 영역에서 주어진 조건을 삭제하는 알고리즘을 이용하여 "$C_{qnew} \Rightarrow C_{f1} \vee \cdots \vee C_{fk}$"가 언제나 참인지 검사하는 알고리즘이다.

```
알고리즘 Hyper Cube Covering (Cqnew, ConditionSet)
/* Cqnew가 set of condition ConditionSet에 의하여 커버 되
   는지 검사*/
/* Cqnew ⇒ ∨C∈ConditionSet C 의 validity 검사 */
/* 영역이 커버가 되는 경우 영역을 커버하는 조건의 집합
   반환. */
/* 커버가 안 되는 경우 널 반환 */
1  UncoveredSpace = { Cqnew }
2  ReturnValue = Φ
3  FOR each C ∈ ConditionSet {
4    US = Φ
5    FOR each SpaceLeft ∈ UncoveredSpace
6      US = US ∪ Delete(SpaceLeft, C)
7    END FOR
8    ReturnValue = ReturnValue ∪ { C }
```

```
9    IF US is Φ
10     RETURN ReturnValue
11   UncoveredSpace = US
12 END FOR
13 RETURN NULL
```

알고리즘 4 HyperCubeCovering 알고리즘

알고리즘 4는 어떤 조건절을 먼저 선택하느냐에 따라서 남아있는 영역의 개수가 달라지게 된다. 남아있는 영역의 개수를 가능한 한 줄이기 위해서는 남은 영역의 분할이 가장 작은 조건절을 먼저 선택하여야 한다. 조건절은 리터럴의 수만큼 영역을 분할하므로 리터럴의 수가 작은 조건절부터 삭제해나가는 것이 유리하다. 단, 남은 영역이 참일 때 언제나 참인 리터럴의 경우에는 이 리터럴을 제거 할 수 있다. 예를 들어 $(0<x<10) \Rightarrow (0<x<20)(y<10) \vee (x<5)(0<y)$ 와 같은 식이 있다고 할 때 $(0<x<20)(y<10)$은 $(y<10)$과 마찬가지이므로 $(0<x<20)$ 조건을 삭제 한 후 $(x<5)(0<y)$ 보다 $(y<10)$을 먼저 선택한다.

4.3.4 질의 변형 알고리즘

4.2와 4.3에서는 요청된 새로운 질의를 분해하고 분해된 질의를 sensors에 대한 질의가 아닌 후보 질의에 대한 질의로 변형하는 방법에 대하여 논의하였다. 이 절에서는 이에 대한 알고리즘을 제시한다. 알고리즘 5는 새로운 질의를 후보 질의에 대한 질의로 식을 변형하는 알고리즘이다.

```
알고리즘 QueryTransform (qnew)
/* qnew가 후보 질의로부터 유도 가능하다면 변형된 질의 반
   환 유도 불가능하면 null 반환*/
/*QCE는 QE중 샘플 주기와 WHERE절 검사를 통과한 후보
   질의 집합 */
/* 임의의 어트리뷰트 x에 대하여 QCx는 Qx에서 샘플 주기
   와 WHERE절 검사를 통과한 후보 질의 집합*/
1  FOR each attribute X ∈ Aqnew
2    Test validity of Cqnew ⇒ Vq∈QCx Cq
3    IF test fails, RETURN
4    ELSE dx = ∪q∈QCx SELECT nodeid, timestamp,
     x FROM q WHERE R(Cqnew, x)
5  END FOR
6  FOR each attribute x ∈ α(Cqnew)-Aqnew
7    Compute Lx = {q ∈ QCE-QCx | RL(Cq, x) ⇒
     RL(Cqnew, x) }
8    Compute Ux = {q ∈QCE-QCx | RU(Cq, x) ⇒ RU
     (Cqnew, x) }
9    Compute Bx = Lx∩Ux
10   Compute Jx = { qa ⋈ qb | qa ∈ Lx - Bx and qb ∈
     Ux - Bx }
11   Test validity of Cqnew ⇒ Vq∈QCx Cq Vq∈Bx Cq Vq∈JX
     Cq
12   IF test fails RETURN
13   ELSE dx = ∪q∈QCx SELECT nodeid, timestamp
                FROM q WHERE R(Cqnew, x))
                ∪q∈(Bx∪Jx) (SELECT nodeid, timestamp
                FROM q)
14 END FOR
15 RETURN qnew = ⋈ dx
```

알고리즘 5 질의 변형 알고리즘

새로운 질의의 무손실 조인 분해는 새로운 질의에서 프로젝션 되는 어트리뷰트에 의해 분해된 질의와 셀렉션 조건절에 쓰이는 어트리뷰트에 의해 분해된 질의를 만든다. 알고리즘 5에서 1~4번째 줄은 전자에 대해서 분해된 질의를 변형하고 5~12번째 줄은 후자에 대해서 분해된 질의를 변형한다. 각각의 경우 조건 2와 조건 3을 만족하는 모든 f_i를 찾은 후 조건 5가 만족되는지 검사를 한다.

알고리즘 2에 따르면 새로운 질의가 유도 불가능으로 판명되어 실행될 때 마다, 새로운 질의를 이용하여 알 수 있는 모든 어트리뷰트 x에 대해서 Q_x에 새 질의를 추가시킨다. 따라서 QC_x는 1~4번째 줄의 경우에 조건 2와 조건 3을 만족하는 모든 질의의 집합이다.

5~12번째 줄의 경우에는 조건 2는 언제나 만족되며, QC_x와 더불어 x값을 모르지만 조건 3을 만족하는 f_i를 모두 찾아야 한다. 어트리뷰트 x에 대하여 상한값 u와 하한값 l을 가지는 새로운 질의에 대하여 그 값을 모르지만 조건 3을 만족하는 fi를 모두 찾기 위해서는 x에 대하여 [l,u] 사이의 값을 상한과 하한으로 가지는 모든 질의의 집합을 각각 구한 후 각각의 집합에서 임의의 질의를 조인하는 조합을 모두 찾으면 된다. 알고리즘 5에서 L_x와 U_x가 각각 x의 하한과 상한을 제한하는 질의의 집합이며 J_x와 B_x는 L_x와 U_x의 질의를 조인하는 모든 질의 집합이다.

조건 5는 각각의 분해된 질의에서 2번째 줄과 11번째 주에서 테스트되는 조건과 동일하다. 이 조건이 만족되지 않는 경우 새로운 질의는 후보 질의로부터 유도 불가능하다.

4.4 전체 알고리즘

지금까지 실행중인 질의 집합에서 새로운 질의의 결과에 기여할 수 있는 후보 질의를 골라내는 방법과 새로운 질의를 후보 질의에 대한 질의로 변형하는 방법에 대하여 살펴보았다. 알고리즘 6은 새로운 질의를 요청받았을 때, 질의의 실행을 최적화시키는 전체 알고리즘이다. 알고리즘 6에는 질의를 병합하는 알고리즘도 포함되어 있다. [5]에는 두 개의 질의를 각각 실행시키는 경우와 두 질의를 합친 질의를 실행시키는 경우의 비용을 비교하여 질의를 병합하는 기법이 제안되어 있다. 본 논문에서는 새로 요청된 질의를 후보 질의로부터 유도하

는 것이 불가능하다고 판단된 경우, 새로 요청된 질의를 실행시키기 전에 [5]의 알고리즘을 이용하여 새로운 질의와 실행중인 질의 중 하나를 합칠 것인지 결정한다. 합치는 것이 더 경제적이라면 합친 질의를 실행시키고, 합쳐진 질의들은 센서 네트워크에서 실행하지 않고 합친 질의로부터 유도한다.

```
알고리즘 Multiple Query Optimization (qnew)
/* 새로운 질의가 요청될 때마다 실행됨 */
/* QE와 QX가 존재한다고 가정 */
1  GenerateCandidateQuery (qnew)
2  q = QueryTransform (qnew)
3  IF q is NOT NULL
4     Add q to QD
5  ELSE IF QueryMergeBenefit(qnew) > 0
6     q = MergeQuery(qnew)
7     QueryExecutuionPreprocessing(q)
8     Execute q
9  ELSE
10    QueryExecutuionPreprocessing(q)
11    Execute q
12 END IF
```

알고리즘 6 다중 질의 최적화 알고리즘

1번째 줄에서 새로운 질의에 기여할 수 있는 기존 질의만을 선택한 후 2번째 줄에서 새로운 질의가 유도 가능하다면 변형된 질의를 생성한다. 3~4번째 줄은 새로운 질의가 유도 가능한 경우이고 5~11번째 줄은 새로운 질의가 유도 가능하지 않은 경우이다. 질의가 유도 가능하지 않다고 판단된 경우에는 5번째 줄에서 [5]의 알고리즘을 사용하여 질의를 병합하는 것이 더 효율적인지를 판단한다. 질의를 병합하는 것이 더 효율적인 경우에는 MergeQuery()를 통해 실행 중인 질의 중 q_{new}와 병합하기로 결정된 질의 q_i를 Q_E와 Q_X에서 제외시키고, q_{new}와 q_i를 Q_D에 추가시킨 후 병합된 질의 q를 생성한다. 7~11번째 줄은 전처리 작업을 거친 후 새로운 질의를 센서 네트워크에서 실행시키는 부분이다.

5. 실험

본 절에서는 제안된 기법의 에너지 효율성을 분석한다. 이를 위해 본 논문에서 제안된 기법을 Java를 사용하여 구현하였으며, [17]의 실제 센서 정보들을 사용하였다. [17]에는 버클리 대학에서 2004년 2월 28일부터 4월 5일까지 TinyDB와 Mica2Dot 54개를 사용하여 수집한 데이타가 저장되어 있다. 이 데이타에는 센서 정보가 수집된 날짜, 시간, 온도, 습도, 빛, 전압 등의 정보가 포함되어 있으며 튜플 수는 약 230만개이다.

위와 같은 환경 상에서 센서 네트워크에 여러 개의 질의들이 요청되었을 때, [17]를 사용하여 질의를 실행하였다. 그리고 질의 최적화 과정을 거치지 않은 경우(naïve), [5]의 질의 병합 기법만 적용된 기법(merge), 제안된 기법을 통해 최적화 과정을 거친 경우(our), 각각에 대해 센서 네트워크에서 베이스 스테이션으로 전달되는 튜플들의 수를 측정하여 본 논문에서 제안한 기법의 효율성을 검증하였다.

실험에 사용된 [질의 집합 1]은 유도 가능한 질의들이 많은 경우에 본 논문이 제안된 기법을 통해 얻을 수 있는 이득을 확인하기 위해 작성된 질의들이다. 제안된 기법에 따르면, Q1과 Q5는 병합되어 실행되며, Q6과 Q7, Q8은 Q1~Q5로부터 결과를 유도해낸다. 그리고 Q2, Q3, Q4는 단독으로 실행된다. Naïve의 경우에는 8개의

```
[질의 집합 1]
Q1: SELECT nodeid, timestamp, temp PERIOD 32
Q2: SELECT nodeid, timestamp, light, temp WHERE 5<=nodeid<=20 AND 200<=light<=800 PERIOD 8
Q3: SELECT nodeid, timestamp, light WHERE 5<=nodeid<=15 AND 250<=light<=700 AND 20<temp<30 PERIOD 8
Q4: SELECT nodeid, timestamp, light, temp WHERE 0<=nodeid<15 AND 150<=light<=950 PERIOD 16
Q5: SELECT nodeid, timestamp, light, temp PERIOD 64
Q6: SELECT nodeid, timestamp, temp WHERE 20<temp<35 PERIOD 32
Q7: SELECT nodeid, timestamp, temp WHERE 0<=nodeid<10 AND temp<27 PERIOD 64
Q8: SELECT nodeid, timestamp, light, temp WHERE 7<nodeid<14 AND 400<light<850 AND 20<temp<30 PERIOD 16

[질의 집합 2]
Q1: SELECT nodeid, timestamp, light, temp WHERE 0<=nodeid<=30 AND 50<=light<=800 AND 22<=temp<=29 PERIOD 4
Q2: SELECT nodeid, timestamp, light, temp WHERE 150<=light<=500 AND 23<=temp<=32 PERIOD 32
Q3: SELECT nodeid, timestamp, light, temp WHERE 10<=nodeid<=20 AND 100<=light<=750 AND 20<=temp<=28 PERIOD 4
Q4: SELECT nodeid, timestamp, light, temp WHERE 90<=light<=600 AND 22<=temp<=30 PERIOD 32
Q5: SELECT nodeid, timestamp WHERE 0<nodeid<10, 150<=light<=700 AND 23<=temp<=28 PERIOD 8
Q6: SELECT nodeid, timestamp, light WHERE 0<=nodeid<7 PERIOD 16
Q7: SELECT nodeid, timestamp, temp WHERE 2<=nodeid<10 AND temp<29 PERIOD 32
Q8: SELECT nodeid, timestamp, light WHERE 0<=light<=900 AND 27<=temp<32 PERIOD 16
```

그림 6 실험에 사용된 질의 집합

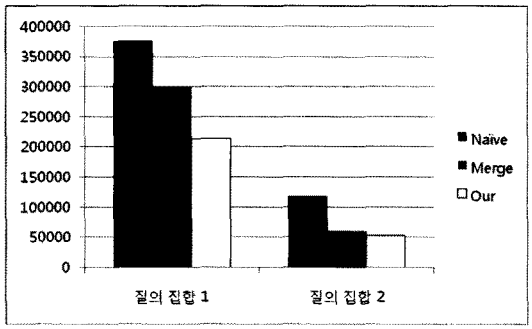


그림 7 질의 실행 결과

질의가 모두 센서 네트워크에서 실행된다.

[질의 집합 2]는 질의가 병합될 기회가 많은 경우, 제안된 기법의 효율성을 검증하기 위해 작성된 질의들이다. 제안된 기법에 따르면, Q1과 Q3, Q2와 Q4, Q6과 Q7은 병합되어 실행되며 Q5는 Q1과 Q3으로부터 유도된다. 그리고 Q8은 단독으로 실행된다. Naïve의 경우에는 8개의 질의가 모두 센서 네트워크에서 실행된다.

그림 7은 [질의 집합 1]과 [질의 집합 2]의 질의들을 실행시켰을 때 베이스 스테이션으로 전달되는 튜플 수를 측정해본 결과이다. 우선 [질의 집합 1]을 실행시킨 결과는 그림 7에서 확인할 수 있는 바와 같이, 결과를 유도해낼 수 있는 질의가 많은 경우 본 논문에서 제안한 기법이 질의 최적화를 수행하지 않은 경우보다 적은 수의 튜플을 만들어낸다. 제안된 기법을 사용할 경우 튜플 수를 질의 최적화를 거치지 않은 경우보다 43.3% 줄일 수 있었고, 질의 병합 기법만 적용한 경우보다 28.9% 줄일 수 있었다.

[질의 집합 2]를 실험 해본 결과, 질의 병합의 기회가 많은 경우에도 최적화를 거치지 않은 경우나 질의 병합 기법만 적용된 경우보다 적은 수의 튜플을 만들어냈다. 제안된 기법은 최적화 과정을 거치지 않은 경우보다 55.04%, 질의 병합 기법만 적용된 경우보다 13.4% 적은 수의 튜플을 베이스 스테이션으로 전달했다.

위의 두 가지 실험 결과, 본 논문이 제안한 기법은 질의의 결과를 유도할 수 있는 경우와 질의를 병합하여 실행할 수 있는 경우에 있어, 최적화 과정을 거치지 않거나 질의 병합 기법만 적용되었을 때보다 적은 수의 튜플을 베이스 스테이션으로 전달하는 것을 확인할 수 있었다. 만약 유도될 수 있는 질의나 병합될 수 있는 질의의 수가 더 많아지면, 더 좋은 성능을 보일 수 있을 것이다.

6. 결 론

무선 센서 네트워크는 에너지를 지속적으로 공급받기 어렵기 때문에 네트워크를 오랫동안 활용하려면 센서 노드에 주어진 에너지를 최대한 효율적으로 사용해야 한다[18]. 따라서 본 논문은 사용자가 다수의 질의를 센서 네트워크에 요청했을 때, 질의들의 상호 관계를 활용하여 에너지를 효율적으로 사용하는 질의 처리 기법을 제안하였다. 본 논문에서 제안한 기법은 사용자가 새로운 질의를 요청했을 때 센서 네트워크에서 실행 중이던 기존 질의들의 결과로부터 새 질의의 결과를 유도해낼 수 있으면, 해당 질의를 센서 네트워크에서 실행시키지 않는다. 따라서 최소한의 사용자 질의만 센서 네트워크에서 실행되기 때문에 모든 질의를 센서 네트워크에서 실행시키는 경우보다 에너지를 훨씬 효율적으로 사용할 수 있다. 특히 본 논문에서 제안하는 질의 변형 기법은 기존 질의 간의 조인 및 유니온까지 고려함으로써 기존 연구에 비해 더 다양한 형태의 질의 재작성 방법을 제공한다. 또한, 질의를 병합할 수 있는 경우에는 병합된 질의를 실행하기 때문에 더욱 높은 효율성을 기대할 수 있다. 실험 결과에 따르면 유도 가능한 질의가 존재하는 경우와 질의를 병합할 수 있는 경우, 최적화 과정을 거치지 않았을 때보다 적은 수의 튜플을 만들어냈다.

향후 연구 방향은 다음과 같다. 센서 네트워크에서 실행되고 있던 기존 질의들 중에서 새로운 질의가 센서 네트워크에서 실행됨으로 인해 결과 유도가 가능해진 질의가 존재할 수 있다. 따라서 이런 종류의 질의들을 효율적으로 검색할 수 있는 기법이 연구되어야 한다. 또한 추후 연구로서 베이스 스테이션이 아닌 네트워크 내(in-network)에서의 다중 질의 최적화 기법을 연구 중이다.

참 고 문 헌

[1] C. Intanagonwiwat, R. Govindan, and D. Estrin. "Directed diffusion: A scalable and robust communication paradigm for sensor networks", In Proc. of the 6th Annual International Conference on Mobile Computing and Networking (MobiCOM), 2000.

[2] S. R. Madden, M. J. Franklin, and J. M. Hellerstein. "TinyDB: an acquisitional query processing system for sensor networks", ACM Trans. on Database Systems, Vol. 30, No. 1, 122-173, 2005.

[3] S. R. Madden, J. Hellerstein, and W. Hong. "TinyDB: In-Network Query Processing in TinyOS", http://telegraph.cs.berkeley.edu/tinydb/.

[4] P. Bonnet, J. E. Gehrke, and P. Seshadri. "Towards Sensor Database Systems", In Proc. of the Second International Conference on Mobile Data Management, Hong Kong, January 2001.

[5] S. Xiang et al., "Two-Tier Multiple Query Optimization for Sensor Networks", In Proc. of the 27 th ICDCS, 2007.

[6] S. R. Madden, M. Frankln, J. Hellerstein, and W.

Hong. "TAG: a tiny aggregation service for adhoc sensor networks", In Proc. of OSDI, 2002.
[7] D. J. Abadi, S. Madden, and W. Lindner. "REED: Robust, efficient filtering and event detection in sensor networks" In VLDB, 2005.
[8] A. Deshpande et al., "Model-driven data acquisition in sensor networks", In VLDB, 2004.
[9] P. Roy et al., "Efficient and extensible algorithms for multi query optimization", In SIGMOD, 2000.
[10] S. R. Madden, M. J. Franklin. "Fjording the Stream : An Architecture for Queries over Streaming Sensor Data", In ICDE Conference, 2002.
[11] N. Trigoni et al., "Multi-query optimization for sensor networks", In DCOSS, 2005.
[12] R. Muller, G. Alonso, "Efficient Sharing of Sensor Networks", In Proc. of MASS, 2006
[13] A. Silberschatz et al., "Database System Concepts", Mc Graw Hill, 4th edition.
[14] P. A. Larson, H. Z. Yang, "Computing Queries from Derived Relations", In VLDB, 1985.
[15] P. A. Larson, H. Z. Yang, "Computing Queries from Derived Relations: Theoretical Foundation", Research report CS-87-35, Computer Science Department, University of Waterloo, 1987.
[16] J. Hoffmann, S. Kupferschmid. "A Covering Problem for Hypercubes".
[17] Intel Lab Data, http://www.select.cs.cmu.edu/data/labapp3/index.html
[18] Y. Yao and J. Gehrke. "Query processing for sensor networks", In Proc. of CIDR Conf., 2003.

이 유 원

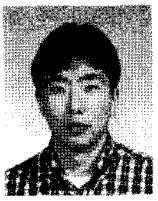
2005년 부산대학교 전자전기정보컴퓨터공학부 정보컴퓨터공학전공(학사). 2007년 한국과학기술원 전산학과(석사). 2007년~현재 한국과학기술원 전산학과 박사과정. 관심분야는 데이타베이스시스템, 센서 네트워크, 스트림 데이타 처리 등

정 은 호

2006년 8월 KAIST 전산학과 학사. 2006년 9월~현재 KAIST 전산학과 석사과정. 관심분야는 Wireless Sensor Networks, Data mining.

함 덕 민

2007년 2월 KAIST 전산학과 학사. 2008년 8월 KAIST 전산학과 석사. 2008년 9월~현재 KAIST 전자전산학과 전산학전공 박사과정. 관심분야는 Keyword Search, Data Mining.



이 충 호

1997년 2월 인하대학교 컴퓨터공학과 학사. 2003년 2월 인하대학교 컴퓨터공학과 박사. 2004년 5월~현재 한국전자통신연구원(ETRI) 선임연구원. 관심분야는 시공간 데이타베이스 시스템, GIS, 센서 네트워크 등.

이 용 준

1987년 2월 연세대학교 전산학 석사. 2001년 2월 충북대학교 전산학 박사. 1984년 2월~현재 한국전자통신연구원(ETRI) 책임연구원. 관심분야는 RFID/USN 미들웨어, e-Logistics, 데이타 마이닝 등

이 기 용

1998년 2월 KAIST 전산학과 학사. 2000년 2월 KAIST 전산학과 석사. 2006년 2월 KAIST 전자전산학과 전산학전공 박사. 2006년 3월~2008년 2월 삼성전자 기술총괄 소프트웨어연구소 책임연구원 2008년 3월~현재 KAIST 전산학전공 연구조교수. 관심분야는 Data warehouse, OLAP, Embedded DB.

김 명 호

1982년 서울대학교 컴퓨터 공학과(학사) 1984년 서울대학교 컴퓨터 공학과(석사) 1989년 Michigan State Univ. 전산학(박사). 1989년~현재 한국과학기술원 전산학전공 교수. 1995년 Univ. of Virginia 방문 교수. 관심분야는 Database, Distributed Systems, Workflow, Multimedia, OLAP, Data Warehouse