제 26 회 한국정보처리학회 추계학술발표대회 논문집 제 13 권 제 2 호 (2006. 11)

RFID 스트리밍 데이터의 효율적인 연속 질의처리를 위한 영역 연속 질의*

이기한, 박재관, 홍봉희

부산대학교 컴퓨터공학과
e-mail : {sephir0th, jkpack, bhhong}@pusan.ac.kr

Range Continuous Queries for Efficient Processing of Continuous Queries on RFID streaming data

Kihan Yi, Jaekwan Park, Bonghee Hong
Dept. of Computer Engineering, Pusan National University

초 록

RFID 미들웨어에서 처리하는 데이터는 스트림 데이터로써 질의색인 기법을 사용하면 효과적이다. 질의색인에서는 RFID 미들웨어의 표준 질의 인터페이스인 ECSpec 이 데이터가 되고, 리더가 태그를 인식하면서 발생하는 태그 이벤트는 질의 색인의 점 질의가 된다. 질의색인의 데이터인 ECSpec 은 태그 및 리더에 대한 수집 조건과 결과집합의 보고 주기를 포함한다. 이때, 태그 이벤트가 발생할 때마다 점 질의를 즉시 수행하는 것보다 보고 주기까지 지연하고 수집된 질의 집합에서 연속되는 태그 이벤트를 영역 질의로 수행하면 질의 수행 횟수를 줄일 수 있다. 본 논문에서는 일정기간 동안의 연속된 태그 이벤트를 영역 연속 질의(a range continuous query)로 처리하기 위한 큐의 구성 방안과 태그 이벤트 집합으로부터 영역질의를 구성하기 위한 자료구조 및 알고리즘을 제안한다.

1. 서론

객체 자동 식별 및 제품의 이력 추적을 위한 기술로써 RFID 가 각광 받고 있다. RFID 시스템은 개별 제품의 식별을 위해 부착되는 태그, 태그의 정보를 인식하는 리더 그리고 태그 이벤트 데이터 및 응용 시스템의 질의를 처리하는 미들웨어로 구성된다. RFID 응용은 객체 식별, 자산관리 및 공급망 관리 (SCM) 등 다양하다.

RFID 미들웨어는 리더의 인식에 의해 연속적이고 끊임없이 발생하는 태그 이벤트를 효율적으로 처리하기 위해 질의 색인을 사용한다[1]. 즉, RFID 표준 ALE 를 따르는 미들웨어에서는 연속질의인 ECSpec 을 질의 색인의 데이터로써, 태그 이벤트를 점 질의로써 처리한다[2][3].

RFID 물류환경에서 태그를 부착한 제품은 개별적으로 이동하는 것이 아니라 유사 제품그룹으로 이동하는 특징이 있다. 따라서 리더에 의해 인식되는 태그 이벤트는 대량으로 발생하며 그 이벤트들은 유사한 식별자를 가진다. RFID 미들웨어에서 사용되는 기존의 질의색인은 태그 이벤트의 이러한 특징을 고려하지 않는다. 즉, 질의의 결과가 동일하거나 유사한 다수의 점 질의를 개별적으로 처리함으로써 동일 또는 유사 검색과정을 반복적으로 수행하는 문제점이 있다.

이러한 문제를 해결하기 위해 ECSpec 에서 지정하는 보고주기 기간 동안 태그 이벤트를 수집하여 점

* 이 논문은 교육인적자원부 지방연구중심대학육성사업
(차세대물류 IT 기술연구사업단)의 지원에 의하여 연구되었음.

질의의 그룹을 구성하고 질의 색인에 영역 질의로써 수행한다. 이러한 그룹 질의 처리 기법은 유사 검색 과정을 제거하므로 질의색인의 성능을 향상 시킨다. 따라서 본 논문에서는 태그 이벤트를 효과적으로 수집하기 위한 큐의 구성 방안을 제안하고 수집된 태그 이벤트로부터 질의 그룹인 *sequence* 를 구성하기 위한 자료구조인 *Sequence Set* 을 제안한다.

본 논문의 구성은 다음과 같다. 2 장에서는 문제정의를 기술한다. 3 장에서는 영역 연속 질의 및 영역 연속 질의 구성을 위한 기본 아이디어를 기술한다. 4 장에서 태그 이벤트 수집을 위한 큐의 Sequence Set 의 구성 방안을 제시하고, 5 장에서는 Sequence 구성을 위한 Sequence Set 에 대해서 자세히 기술한다. 6 장에서는 제안한 기법에 대한 실험 및 분석 결과를 기술한다. 마지막으로, 7 장에서는 결론 및 향후 연구를 기술한다.

2. 문제정의

RFID 미들웨어는 리더에 의해 인식되는 대량의 태그 이벤트를 여과 및 수집하여 응용 시스템으로 전달한다. 리더의 인식에 의해 연속적이고 끊임없이 발생하는 태그 이벤트를 처리하기 위해서는 질의 색인을 사용하는 것이 효율적이다. 따라서 RFID 미들웨어는 ECSpec 을 질의 색인에 데이터로써, 태그 이벤트를 점 질의로써 처리한다.

RFID 응용 물류 환경에서 태그를 부착한 제품들은 개별적이 아니라 유사 제품 그룹을 이루어 이동하는 특징이 있다. 따라서 태그가 리더의 인식에 의해 발생하는 태그 이벤트는 대량이며 유사 식별자(EPC 코드)를 가진다. 질의 색인을 사용하는 기존의 RFID 미들웨어에서는 이러한 특징을 고려하지 않고 모든 태그 이벤트를 개별적으로 처리한다. 그로 인해 동일하거나 유사한 결과를 초래하는 다수의 점 질의를 개별적으로 처리함으로써 동일 또는 유사 검색과정을 반복적으로 수행해 미들웨어의 성능 저하를 가져오는 문제점이 있다.

이러한 문제점을 해결하기 위해서, 본 논문에서는 태그 이벤트의 이러한 특징을 고려하여 연속된 태그 이벤트를 개별적인 점 질의로 처리하지 않고 질의그룹을 구성하여 영역 질의로 수행함으로써 질의 수행 횟수를 줄일 수 있는 기법을 제안한다.

3. 영역 연속 질의

3.1. Sequence 및 영역 연속 질의 정의

질의 색인의 데이터(ECSpec)는 보고시간 주기를 가지는데 이것을 T 라고 하자. 이러한 주기 내에서는 태그 이벤트를 즉시 처리하지 않고 지연하는 것이 허용된다. 이렇게 수집된 태그 이벤트 집합으로부터 sequence 를 정의할 수 있다. Sequence 정의는 다음과 같다.

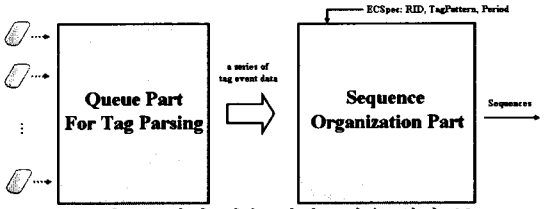
- **[정의 1] Sequence(S):** 매 보고 주기 T 시간 동안 수집된 동일한 RID(리더 식별자)를 가지는 태그 이벤트들의 집합. 따라서 수집된 태그 이벤트들은 다수의 sequence 로 표현된다.

Sequence 를 기반으로 *영역 연속 질의(a Range Continuous Query, RCQ)*를 아래와 같이 정의한다.

- **[정의 2] *영역 연속 질의(RCQ):*** RID 와 TID(태그 식별자)로 구성된 2 차원 평면상에서 동일한 rid 를 가지며 주어진 *sequence S* 의 최소 TID 와 최대 TID 사이에서 겹치는 모든 데이터를 찾는 질의. 따라서 (rid, S)로 표현된다.

3.2. 영역 연속 질의 구성을 위한 기본 아이디어

보고주기 기간 동안 수집되는 태그 이벤트로부터 sequence 를 구성하여 질의 색인에 영역 연속 질의로써 수행하면 질의수행 횟수를 효과적으로 줄일 수 있다. 따라서 본 논문에서는 그림 1 과 같이 태그 이벤트를 효과적으로 수집하기 위한 큐의 구성 방안을 제안하고 수집된 태그 이벤트로부터 영역연속 질의를 구성하기 위한 Sequence Set 을 제안한다.



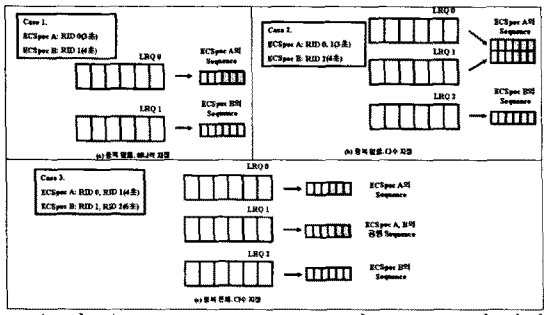
(그림 1) 영역 연속 질의구성을 위한 구조

4. 태그 이벤트 수집을 위한 큐와 Sequence Set 의 구성

ECSpec 은 대상 리더(논리적 리더)와 보고 주기를 명시한다. 등록된 ECSpec 의 조건에 따라서 sqeuence 의 구성이 달라진다. 즉, 효율적인 sequence 구성을 위해서는 태그 이벤트 수집을 위한 대상 리더 별 큐(LRQ)의 구성이 중요하다. ECSpec 과 큐의 관계는 3 가지 경우가 있다.

첫 번째, 다수의 ECSpec 이 중복 없이 리더를 하나씩 설정하는 경우이다. 이 경우는 그림 2-(a)와 같이 대상 리더 간에 중복이 없어 ECSpec 별로 sequence 가 생성된다. 두 번째, 하나의 ECSpec 이 중복 없이 다수 리더의 데이터를 요구하는 경우이다. 각 리더는 서로 다른 RID 가 부여되기 때문에 리더 별로 생성되는 데이터는 서로 다른 RID 를 가지게 된다. 따라서 RID 가 다른 데이터를 동일 sequence 로 묶을 수가 없기 때문에 그림 2-(b)와 같이 ECSpec A 에 의해 다수 개의 Sequence 가 생성된다. 또한, 첫 번째와 두 번째는 대상 리더의 중복이 존재하지 않아 해당 ECSpec 의 보고주기 동안 태그 이벤트의 수집이 가능하다. 마지막 세 번째 경우는 상이한 보고주기를 가진 다수의 ECSpec 이 같은 리더를 중복 지정하는 경우이다. RID 가 상이해 리더 별로 sequence 가 생성되는 두 번째 경우처럼 세 번째 경우 역시 그림 2-(c)와 같이 RID

가 다른 다수의 sequence 가 생성된다. 그러나 두 번째 경우와 달리 다수의 ECSpec 이 동일 리더를 중복 지정하기 때문에 해당 리더의 이벤트 수집 주기는 특정 ECSpec 의 것을 따르지 못하고 관련된 ECSpec 들의 보고 주기들에 대한 최대 공약수를 취해야만 한다.

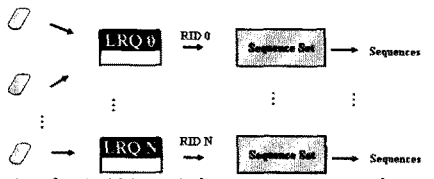


(그림 2) ECSpec, sequence 그리고 Queue 의 관계

등록된 ECSpec 들의 조합에 따른 분석 결과에 의해 ECSpec 과 sequence 그리고 LRQ(논리적 리더 단위의 큐)의 관계는 다음과 같다.

$$\textbf{ECSpec} : \textbf{Sequence} = 1 : N$$
$$\textbf{Sequence} : \textbf{LRQ} = 1 : 1$$

관계 분석에 따른 리더장치와 LRQ 그리고 sequence 추출 구조인 Sequence Set 에 대한 구성 방안은 그림 3 과 같다. 다수의 리더 장치에서 발생하는 태그 이벤트는 논리적 리더 (대상 리더) 별로 수집된다. 즉, 논리적 리더 별로 존재하는 큐(LRQ)로 수집된다. 그리고 각 LRQ 의 태그 이벤트는 1 대 1 로 연결되어 있는 Seqeunce Set 으로 전달된다.



(그림 3) 큐(LRQ)와 Sequence Set 의 구성

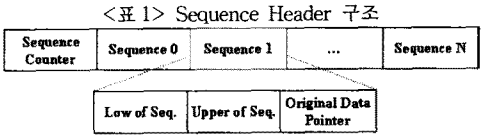
5. Sequence 구성을 위한 Sequence Set

Sequence Set 은 가변길이 배열과 연결 리스트의 하이브리드 구조이다. Sequence Set 은 삽입되는 데이터에 대해서 효율적으로 Sequence 를 구성하기 위한 자료구이다.

5.1. 자료 구조

Sequence Set 은 표 1 과 같이 태그 이벤트 집합에 대해서 Sequence 구성 정보를 표현하는 *Sequence Header* 와 구성된 sequence 의 원본 태그 이벤트의 리스트인 *Original Data List* 으로 구성된다.

<표 1> Sequence Header 구조



Sequence Header 는 다음과 같은 두 가지 성질을 만족한다.

$$\textbf{Low}(S_{i+1}) - \textbf{Upper}(S_i) > MaxGap$$
$$\textbf{Low}(S_{i+1}) > \textbf{Low}(S_i)$$

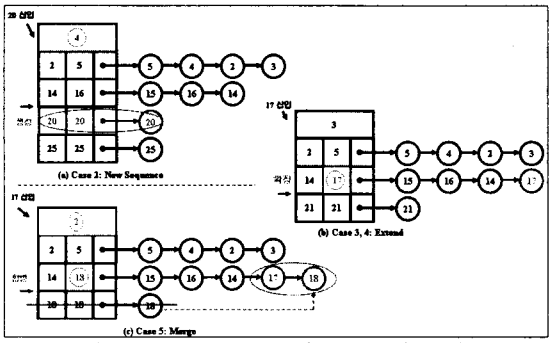
여기서 **Low()**와 **Upper()**는 sequence 의 하한 값과 상한 값을 각각 나타낸다. 그리고 sequence 생성 시 sequence 생성 수를 줄이기 위해 특정 값 이하의 차이를 가지는 태그 이벤트는 같은 sequence 로 묶는다. 이것을 최대 허용 간격(*MaxGap*)이라 한다.

5.2. 삽입 알고리즘 및 추출 알고리즘

삽입되는 태그 이벤트의 값을 *epc* 라고 하자. 먼저 *epc* 보다 작으면서 가장 큰 하한 값을 가지는 i 번째 sequence(S_i)를 찾는다. *epc* 삽입에 따른 S_i 의 처리는 5 가지 경우로 나뉜다(단, case 1 은 *MaxGap* 이 2 이상인 경우만 발생). 각 경우에 대한 처리 과정은 아래와 같다.

- **Case 1: 변화 없음**
- **Case 2: 새로운 sequence 의 생성**
- **Case 3: i 번째 sequence S_i 의 확장**
- **Case 4: i+1 번째 sequence S_{i+1} 의 확장**
- **Case 5: Sequence S_i 와 S_{i+1} 의 연결(Merge)**

그림 4 는 *MaxGap* 이 1 일 때 각 Case 에 대한 삽입 과정을 보여준다(Case 1 제외). 그림 4-(a)는 Case 2 의 경우를 보여준다. 먼저, 삽입되는 20 보다 작으면서 가장 큰 하한 값을 가진 sequence S(14,16) 찾는다. 이 때, 20 은 상한 값인 16 과 4 차이가 나며 또한 다음 sequence 인 S(25,25)의 하한 값과도 5 차이가 난다. 따라서 20 은 어느 sequence 에도 연결되지 못하므로 새로운 sequence $S_{new}(20,20)$을 생성하게 된다. Case 3, 4 를 보여주는 그림 4-(b)는 삽입되는 17 에 대해서 Case 2 와 같이 S(14,16)을 찾은 다음, 그것의 상한 값 16 과 1 차이를 보여 sequence 가 확장되는 경우를 보여준다. Case 4 의 처리도 이와 유사하다. 마지막으로 그림 4-(c)는 삽입된 17 이 S(14,16)과 S(18,18)을 이어 두 sequence 를 하나의 합쳐진 sequence $S_{merged}(14,18)$를 만드는 Case 5 의 경우를 보여준다.



(그림 4) Sequence Set 의 Case 별 삽입 과정

[알고리즘 1]은 Sequence Set 의 삽입 알고리즘을 보여준다. 단, 삽입되는 *epc* 가 가장 작거나 큰 경우에 대해선 고려하지 않았다. 하지만, 알고리즘의 간단한 확장을 통해 해결 가능하다.

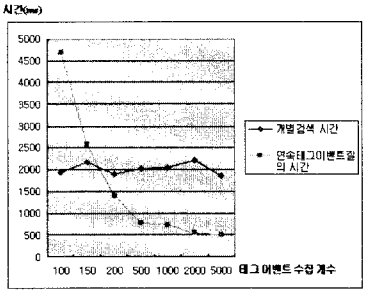
```
SequenceSet_Insert(epc)
SSI1: entry_i = find largest i if entry_i.low < epc
SSI2: if entry_i.upper > epc
        then entry_i.add(epc)
    else
        if epc-entry_i.upper > MaxGap and entry_i+1.lower – epc > MaxGap
            then newEntry = SequenceHeader.Insert(epc)
                 newEntry.add(epc)
        elseif epc – entry_i.upper ≤ MaxGap and entry_i+1.lower – epc > MaxGap
            then SequenceHeader.Extend(entry_i+1, epc)
                 entry_i.add(epc)
        elseif epc – entry_i.upper > MaxGap and entry_i+1.lower – epc ≤ MaxGap
            then SequenceHeader.Extend(entry_i+1, epc)
                 entry_i+1.add(epc)
        else
            then mergedEntry = SequenceHeader.Merge(entry_i, entry_i+1)
                 mergedEntry.add(epc)
        endif
    endif
```

[알고리즘 1] Sequence Set 의 삽입 알고리즘

Sequence 의 추출 알고리즘은 직관적이며 간단하다. *Sequence Header* 의 각 entry 는 sequence 를 나타내므로 구성된 sequence 의 추출은 *Sequence Header* 전체를 순차적으로 스캔하여 이루어진다.

6. 실험 및 분석

실험은 윈도우 XP OS 에 펜티엄 IV 3.0Ghz 프로세스, 1.5GB 메모리를 갖춘 시스템에서 이루어졌으며 구현은 C++로 이루어졌다. 질의색인으로는 A.Guttman 의 R-tree[4]를 사용하였으며, 삽입 데이터 1 만개, 검색 질의 10 만개를 사용하였다. 삽입 및 검색 데이터 모두 균일 분포를 따랐으며 허용 최대 간격은 1 로 설정하였다. 간단한 구현을 위해 보고 주기 대신 태그 이벤트의 수집개수를 사용하였다. 그림 5 는 태그 이벤트 수집 개수에 따른 질의 색인의 처리 시간을 측정한 것이다. 실험 결과는 수집 개수가 적을 때는 sequence 생성 비용이 상대적으로 커 개별 처리보다 검색 성능이 좋지 못한 결과를 보였지만 수집 개수가 늘어날수록 더 좋은 성능을 보여주었다.



(그림 5) 질의 수집 개수에 따른 검색 성능 비교

7. 결론 및 향후 연구

본 논문에서는 RFID 미들웨어에서 연속되는 태그 이벤트에 대해서 효율적인 처리를 위한 기법을 제시하였다. 표준 명세인 ECSpec 요청에 대한 결과집합을 일정 주기마다 보고하는 특성을 이용하여 리더에 의해 발생하는 태그 이벤트를 매번 개별적으로 처리하지 않고 보고 주기 동안 지연하여 단일화된 sequence 구성을 통한 영역 연속 질의로 처리하였다. 제안된 기법은 질의 수행 횟수를 줄여 RFID 미들웨어의 전반적인 성능 개선 효과를 가져올 것으로 예상된다.

향후 연구과제로는 본 연구에서 질의색인으로 사용한 R-tree 이외에 질의 데이터인 ECSpec 의 특성을 고려한 효과적인 질의색인 구조에 대하여 연구할 계획이다.

참고문헌

[1] S. R. Madden, M.A.Shah, J. M. Hellerstein and V.Raman, "Continuously adaptive continuous queries over streams", ACM SIGMODE, pp.49-60, 2002

[2] K.Traub, S.Bent, T.Osinski, S.N.Peretz, S.Rehling, S.Rosenthal And B.Tracey, "The Application Level Event(ALE) Specification, Version 1.0", EPCglobal, 2005

[3] 석수옥, 박재관, 홍봉희, "RFID 미들웨어에서 이벤트 필터링을 위한 질의 색인 기법", 한국정보과학회 제 32 권 제 2 호, pp.19-21, 2005

[4] A. Guttman. " R-Tree: A dynamic index structure for spatial searching". In Proc. of the 1984 ACM SIGMOD Intl. Conf. on Management of Data, pp.47-57, June 1984