RFID 미들웨어에서 이벤트 필터링을 위한 질의 색인 기법*

석수욱[O] 박재관 홍봉희
부산대학교 컴퓨터공학과
{seok1968[O], jkpack, bhhong}@pusan.ac.kr

A Query Indexing Method for Filtering Event Data in RFID Middleware Systems

Su wook Seok[O], Jae kwan Park, Bong hee Hong
Department of Computer Science and Engineering, Pusan National University

요 약

EPCglobal은 RFID와 관련된 다양한 분야의 표준화를 주도하고 있으며, 응용 표준으로써 Tag 정보의 운용을 위한 미들웨어 표준인 ALE Specification을 제시하였다. ALE의 ECSpec은 애플리케이션이 미들웨어에 등록하는 이벤트 필터링을 위한 스펙으로써 일정 시간동안 반복적으로 수행되는 연속 질의와 유사한 특성을 가진다. ECSpec을 연속질의로 변환할 때 해당 질의의 WHERE절이 가지는 Predicate는 매우 긴 길이를 가지는 Long Interval이 된다. 이러한 특성은 기존의 질의 색인들의 삽입과 검색 성능을 저하시키는 문제점을 가진다.

이 논문에서는 ECSpec을 연속 질의의 형태로 변환하고 해당 질의가 가지는 Predicate인 2D Interval의 특성을 반영한 새로운 질의 색인 구조로써 TLC-Index를 제안한다. 색인 구조는 그리드 방식의 큰 크기를 가지는 셀 분할 구조와 선분 모양의 가상 분할 구조를 병행하는 하이브리드 구조이다. 색인에서 Long Interval의 정의는 셀 분할 구조의 길이보다 크거나 같은 길이를 가지는 Interval이다. 제안하는 색인은 Long Interval을 큰 크기를 가지는 셀 분할 구조로 분할 삽입함으로써 저장 공간의 소모를 줄이고 삽입 성능을 향상시킨다. 또한 Short Interval들을 짧은 길이를 가지는 가상 분할 구조들로 분할 삽입함으로써 그리드 방식이 가질 수 있는 부분적 겹침을 제거하여 검색 성능을 향상시킨다.

1. 서 론

EPCglobal은 RFID를 사용하여 전 세계의 물류환경을 통합하기 위한 표준화를 진행하고 있다. RFID 미들웨어 시스템의 표준으로써 초기에 Savant라 불리는 내부 구조에 중점을 둔 표준을 제시하였으나, 현재 미들웨어의 인터페이스 표준화에 중점을 둔 ALE(Application Level Event) Specification[1]을 제시하였다.

RFID 리더로부터 미들웨어로 수집되는 EPC 이벤트 데이터는 센서 네트워크의 데이터 스트림과 매우 유사한 특성을 가진다. EPC 이벤트 데이터는 리더로부터 연속적으로 시간의 순서를 가지고 끊임없이 RFID 미들웨어 시스템으로 전달된다.

그림 1 RFID 미들웨어 시스템(ALE Engine)

애플리케이션은 미들웨어로 전달되는 대량의 EPC 이벤트 데이터들 중에서 필요한 데이터들만을 전달받기 위해 그림 1과 같이 미들웨어에 ECSpec(Event Cycle Specification)을 등록한다. ECSpec은 어떤 리더로부터 어떤 EPC 이벤트 데이터들을 필터링하여 보고받을 것인지에 대한 내용을 포함하고 있다. 이러한 ECSpec은 일정 시간의 사이클(Cycle)동안 삽입되는 이벤트 데이터들에 대해 필터링과 컬렉션을 반복적으로 처리하여 결과인 ECReports(Event Cycle Reports)를 생성하게 된다. 즉, ECSpec은 일정 시간 범위 동안 계속적으로 수행되어야 하는 연속질의(Continuous Query)[2]와 유사한 특성을 가진다.

최근 연속 질의를 처리하는 방법으로 질의 색인(Query Index) 기법에 대한 연구가 활발히 진행되고 있다. 질의 색인은 연속질의를 색인하여 실시간으로 삽입되는 데이터 스트림을 필요로 하는 질의가 무엇인지를 빠르게 검색할 수 있도록 한다.

본 논문에서는 ECSpec의 이벤트 필터링을 실시간으로 처리하기 위한 질의 색인 기법인 TLC(Two-Level Construct) Index를 제안한다. ECSpec이 가지는 질의 Predicate는 논리적인 리더명과 태그의 필터링 패턴으로 구성되는 2차원의 Interval이 된다. 이러한 Interval은 필터링 패턴에 의해 매우 긴 길이를 가지는 Long Interval이 된다. 기존의 질의 색인은 이러한 Long Interval에 의해 삽입과 검색성능이 급격히 저하되며 저장 공간의 소모가 증가하게 되는 문제점을 가진다. 이 문제를 해결하기 위해 TLC Index는 그리드 방식의 큰 크기를 가지는 셀 분할 구조와 선분 모양의 가상 분할 구조를 병행하여 사용한다. 색인에서 Long Interval의 정의는 셀 분할 구조의 길이보다 크거나 같은 길이를 가지는 Interval이다. 제안하는 색인은 Long Interval을 다양한 크기의 레벨을 가지는 셀 분할 구조로 분할 삽입함으로써 저장 공간의 소모를 줄이고 삽입 성능을 향상시킨다. 또한 Short Interval들을 짧은 길이를 가지는 가상 분할 구조들로 분할 삽입함으로써 그리드 방식이 가질 수 있는 부분적 겹침을 제거하여 검색 성능을 향상시킨다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 질의 색인에 관한 관련 연구를 기술하고 3장에서는 ECSpec이 가지는 Predicate 특성의 분석과 그에 따른 문제점을 정의한다. 4장에서 ECSpec의 이벤트 필터링에 적합한 질의 색인 기법을 제시하고 5장에서 기존의 질의 색인과의 검색 성능을 평가한다. 마지막으로 6장에서 결론을 맺는다.

2. 관련 연구

등록된 다수의 연속 질의들을 실시간으로 수행하기 위해서는 삽입되는 데이터 스트림과 관계되는 질의가 무엇인지를 빠르게 검색하는 것이 필요하다. 이를 위해 최근 연속 질의를 색인하는 기술이 연구 개발 되고 있다. 질의 색인은 실시간 처리를 위해 기본적으로 메인 메모리 기반 구조를 가진다. 질의를 색인하는 방법은 질의의 조건 절 즉, predicate를 삽입에 이용하는 것이다. 검색은 색인 영역 상에서 데이터 스트림이 점으로 표현되기 때문에 오직

* 이 논문은 교육인적자원부 지방연구중심대학육성사업 "차세대물류IT기술연구사업단"의 지원에 의하여 연구되었음.

점질의(Point Query)만을 포함한다. 대표적인 다차원 질의 색인 구조로는 VCR(Virtual Construct Rectangle) Index[3,4]와 CQI(Cell-based Query Indexing) Index[5]가 있다.

VCR-Index는 이동체에 대한 영역 질의를 색인한다. 영역 질의의 Predicate는 2차원의 사각형으로 표현이 되는데 이것을 미리 정의된 VCR(Virtual Construct Rectangle)이라는 가상의 사각형 구조를 이용하여 분할하고 해당 구조의 ID List에 질의의 ID를 삽입하는 방식을 사용한다. 검색 방법은 질의 점을 포함하는 모든 분할 구조의 ID List를 탐색하는 것이다.

CQI-Index는 셀 기반의 그리드 방식을 적용한 방법으로 이동체에 대한 영역 질의를 색인한다. 전체 영역을 일정 크기의 셀로 분할하고 각 셀을 위한 2개의 ID List인 Full List와 Part List를 가진다. Part List는 분할된 영역이 셀에 부분적으로 겹치는 경우를 위한 리스트이다. 메인 메모리 기반의 색인으로 R-tree보다 높은 질의 검색 성능을 보장하지만 검색 시 Part List로부터 얻어진 결과 질의의 셋들이 실제로 이동체를 포함하는지 비교하기 위한 정제 단계가 필요하기 때문에 검색 성능이 저하되는 단점이 있다.

3. ECSpec을 위한 Predicate의 특성 및 문제 정의

3.1 ECSpec을 위한 Predicate의 특성

ECSpec은 어떤 리더로부터 읽혀진 EPC 이벤트 데이터를 원하는지에 대한 논리적 리더명(Logical Reader Name, LRN), 어느 시간 간격동안 데이터를 수집할 것인지에 대한 ECBoundary Sepc, 어떤 EPC 패턴의 데이터들만을 필터링할 것인지에 대한 필터링 패턴(Filtering Pattern), 그리고 군집화(aggregation)를 위한 정보 등을 포함한다. 그림 2는 EPCglobal의 Specification에서 제시한 ECSpec의 예이다.

9.5.1 ECSpec: Door42PassThrough

Description:	TODO: Description of why included		
Parameters:			
readers	DockDoor42		
includeSpecInReports			
ECBoundarySpec:			
startTrigger		stopTrigger	
repeatPeriod	50 MS	duration	10 MS
stableSetInterval			
ECReportSpec			
reportName	DockDoorPassThrough	reportSet	ADDITIONS
reportIfEmpty	false	reportOnlyOnChange	true
output	MEMBERS		
filter			
group			

그림 2 ECSpec의 예

그림 3은 예로 보인 ECSpec을 연속질의 즉, CQL(Continuous Query Language)로 표현한 것이다. LRN을 ReaderID로 표현하고 필터링 패턴을 TagEPC로 표현했다.

```
SELECT epc FROM DataStream
[Range now, now + 10]
WHERE ReaderID = DockDoor42 AND
TagEPC = 20.*.*
REPEAT INTERVAL 50
```

그림 3 CQL로 표현한 ECSpec

위의 그림 3에서와 같이 WHERE절의 Predicate를 구성하는 요소는 ReaderID와 TagEPC가 된다. 이때 ReaderID는 여러 개의 물리적 리더명(Physical Reader Name, PRN)의 집합으로 구성이 되며 축 상에서 PRN인 EPC(예: *urn:epc:id:gid-96:1.1.1*)로 표현된다. 이때, LRN을 구성하는 PRN들은 서로 discrete한 특성을 가지기 때문에 ReaderID는 축 상에서 점으로 표현 된다. TagEPC는 필터링 패턴이 상수, 영역 또는 *(전체)로 구성되기 때문에 점 또는 Interval로 표현 된다. 따라서 ECSpec을 위한 질의의 Predicate은 ReaderID와 TagEPC로 구성되는 2차원의 Interval로 표현된다. 이 후 논문에서 Predicate를 2D Interval이라고 기술 한다.

	Header	General Manager Number (Company)	Object Class (Product)	Serial Number (Serial)
GID-96	8	28	24	36
	0011 0101 (Binary Value)	268,435,455 (Max decimal)	16,777,215 (Max decimal)	68,719,476,735 (Max decimal)

그림 4 96-bit EPC (GID-96)의 인코딩

TagEPC는 그림 4와같이 EPC 코드가 매우 큰 도메인(헤더를 제외하고 2^{80})을 가지며 패턴의 형태가 20.*.*와 같이 회사 필드 또는 제품 필드 전체를 포함하는 경우가 많기 때문에 색인 공간상에서 매우 긴 Long Interval이 되는 특성을 가진다.

3.2 문제 정의

2D Interval이 가지는 Long Interval의 특성은 색인의 삽입 및 검색 성능을 저하시킨다. R-tree와 같은 트리 구조의 색인들은 Long Interval에 의해 겹침(overlap)이 증가하여 검색 성능이 급격히 떨어지게 된다. VCR-Index의 경우 Long Interval이 다수의 가상 분할 구조에 의해 분할되기 때문에 저장 공간의 소모가 커지고 삽입 시간이 크게 증가하게 된다. VCR의 side length를 증가시키게 되면 검색 시 탐색해야 할 ID List의 수가 많아져 검색 성능이 저하되는 문제점이 있다. CQI-Index의 경우 Long Interval을 위한 큰 크기의 셀 구조를 사용할 수 있지만 셀 크기가 커질수록 Short Interval에 의한 Part List로의 삽입이 증가하여 정제단계로 인한 검색 비용이 증가하는 문제점이 있다. 본 논문에서는 Long Interval에 의한 성능 저하를 해결하기 위한 새로운 색인 구조를 제시한다.

4. 이벤트 필터링을 위한 TLC-Index

본 논문에서 제안하는 TLC(Two-Level Construct) Index는 기존의 가상 분할 구조(Virtual Construct) 방식과 그리드의 셀 분할 방식을 함께 사용한다. 제안 방법을 설명하기 위해 먼저 아래의 용어를 정의한다.

정의 1. LPI(Long Pattern Interval) : TagEPC의 길이가 셀 분할 구조의 길이보다 크거나 같은 2D Interval 또는 그것의 부분 Interval

정의 2. SPI(Short Pattern Interval) : TagEPC의 길이가 셀의 크기보다 짧은 2D Interval 또는 그것의 부분 Interval

4.1 Basic Idea

TLC-Index는 Long Interval의 문제점을 해결하기 위해 LPI를 저장하기 위한 매우 큰 크기의 셀 분할 구조를 사용한다. 셀의 크기는 서로 다른 크기를 가지는 여러 레벨들로 구성이 된다. LPI는 상위 레벨의 큰 크기를 가지는 셀 분할 구조에 우선순위를 가지고 분할된다. 또한 LPI가 셀 분할 구조에 완전히 겹쳐질 때만 삽입을 하여 정제단계로 인한 검색 비용을 줄일 수 있다. 셀 분할 구조에 부분적으로 겹칠 수 있는 작은 크기의 SPI들은 가상 분할 구조를 이용하여 분할한다. 가상 분할 구조의 최대 크기는 SPI들만을 대상으로 하기 때문에 검색 성능에 최적화할 수 있다.

4.2 색인 구조

그림 5는 TLC-Index의 구조를 나타내고 있다. 제안하는 색인은 두 가지 분할 구조를 가진다. 그리고 해당 분할 구조를 위한 노드와 ID List를 가진다. TagEPC와 ReaderID는 EPC 코드 값을 도메인으로 가지며 색인의 전체 영역은 먼저 크기에 따라 여러 레벨을 가지는 그리드의 셀로 분할된다. 가상 분할 구조는 그리드 셀의 최소 가로 크기보다 작은 크기의 분할 구조로써 Predicate의 데이터 표현과 동일한 2차원의 Interval 형태이다. 질의의 Predicate인 2D Interval은 가상 분할 구조와 셀 분할 구조로 분할되고 분할에 참여한 가상 구조 또는 셀 구조의 노드에 연결된 ID List에 엔트리로 저장된다. 분할에 참여하는 구조들만을 해당 구조의 ID로 Hashing하여 노드를 실제로 할당한다.

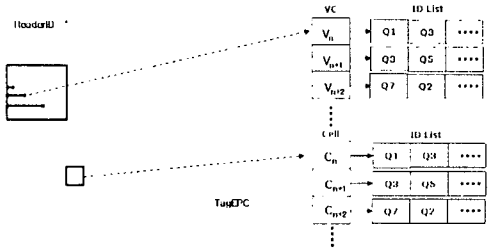

그림 5 TLC-Index의 구조

4.3 데이터 모델

색인에 삽입되는 Predicate인 2D Interval은 (*qid, tagEPC^{L}, tagEPC^{U}, readerID*)로 표현된다. *qid*는 질의의 아이디, *tagEPC^{L}*는 tagEPC의 lower bound, *tagEPC^{U}*는 tagEPC upper bound 그리고 *readerID*는 리더 명을 나타낸다. 색인으로의 질의인 EPC 이벤트 데이터는 (*readerID, tagEPC, time*)로 구성된다. *readerID*는 해당 이벤트 데이터를 읽은 리더의 EPC이며 *tagEPC*는 태그의 EPC이고 *time*은 데이터가 읽혀진 시간이다.

4.4 가상 분할 구조와 셀 분할 구조

SPI의 분할을 위한 가상 분할 구조는 그림 6과 같다. TagEPC를 x라 하고 ReaderID를 y라고 할 때 가상 분할 구조는 그림 6과 같이 (*x, y, length*)로 표현된다. 이때 x, y의 범위는 각 축의 최대값을 각각 R_x, R_y라 할 때 $0 \le x < R_x$, $0 \le y < R_y$ 가 된다. *length*는 TagEPC의 범위를 나타내며 2의 지수 승으로 표현한다. 따라서 위의 구조는 (*x, y, 2^i-1*)로 바꾸어 표현할 수 있다. 이때 i의 범위는 $0 \le i \le k$이며 2^k-1는 *length*의 최대값이 된다. 또한 i가 0일 때 *length*는 0이 되고 분할 구조는 점이 된다. 이러한 가상 선분 구조는 각 좌표마다 k+1개씩 존재한다. 가상 분할 구조는 고유한 ID를 가진다. (*x, y, 2^i-1*)의 ID는 각 좌표 당 가상 분할 구조의 개수를 N이라 하고 (*0, 0*), (*1, 0*),…, (R_x-*1, 0*), (*0, 1*),…, (R_x-*1, 1*),…, (*0, y*), (*1, y*),…, (*x-1, y*)과 같은 순서로 ID번호를 부여할 경우, $N(x+yR_x)+i$가 된다.

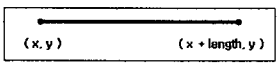

그림 6 가상 분할 구조

LPI의 분할을 위한 셀 분할 구조는 색인 공간을 일정한 크기의 그리드로 분할한 구조이다. 셀의 모양은 Interval의 모습과 동일하다. tagEPC를 위한 셀의 가로 크기는 2^n이며 이때 n의 범위는 가상 분할 구조의 최대 크기가 2^n-1이고 b = $\log_2 R_x$일 때 a < n < b가 된다. 서로 다른 n에 의해 하나 이상의 셀 분할 구조를 가질 수 있으며 필터링 패턴의 특징에 따라 구조의 개수를 가변적으로 최적화할 수 있다. 셀의 ID는 Peano Key를 이용하여 구한다.

4.5 데이터 삽입 및 삭제

2D Interval의 lower bound부터 두 가지 분할 구조로 분할을 시작한다. 각 셀 분할 구조에 완전히 겹쳐지는 LPI의 경우 셀 분할 구조로 분할되고 최소 길이의 셀 구조보다 작은 길이를 가지는 SPI의 경우 가상 분할 구조를 사용하여 분할한다. lower bound부터 upper bound까지의 모든 분할이 끝나면 사용된 모든 가상 분할 구조 또는 셀 분할 구조의 ID List에 <*QID, pointer*> 형태의 엔트리를 삽입한다. 삭제는 삽입 과정과 동일한 알고리즘으로 분할 구조의 ID List를 찾고 해당 리스트를 탐색하여 엔트리를 삭제하면 된다.

4.6 데이터 검색

검색은 리더가 RFID 미들웨어로 EPC 이벤트 데이터를 보낼 때마다 반복적으로 수행된다. 이러한 EPC 이벤트 데이터는 질의 색인의 공간상에서 점으로 표현이 되므로 검색은 곧 점질의가 된다.

검색 방법은 해당 점을 포함할 수 있는 모든 가상 분할 구조를 찾고 해당 구조의 노드에 연결된 ID List를 탐색하는 것이다. 또한 해당 점을 포함할 수 있는 모든 레벨의 셀 분할 구조를 찾고 해당 셀 구조의 노드에 연결된 ID List를 탐색한다. 이때 질의 점을 포함하는 셀 구조의 수는 레벨의 수와 일치한다.

5. 성능평가

TLC-Index와 VCR-Index의 검색 성능을 평가하였다. 실험 환경은 윈도우XP, 1GB 메인메모리, Pentium4 2.6Ghz이며 색인은 Java로 구현되었다. Long Interval을 가지는 Predicate 10000개를 삽입하고 리더 수 200개인 환경에서 이벤트 데이터 1000개를 점 질의하였다. 그림 7은 Long Interval의 길이의 증가에 따른 평균 검색시간을 측정한 것이다. Interval의 길이가 2^9보다 짧을 때 VCR-Index가 더 우수한 성능을 보였으나 Interval의 길이가 길어질수록 VCR-Index의 검색 성능이 급격히 저하되는 것을 볼 수 있다. 이것은 VCR의 side length 최대 크기의 증가로 인해 탐색해야하는 대상 노드의 수가 급격히 증가하기 때문이다. 제안하는 색인은 Long Interval을 완전히 겹쳐지는 셀 구조로 분할함으로써 탐색 노드수를 일정하게 유지하여 Interval의 길이가 길어져도 높은 검색 성능을 유지하는 것을 볼 수 있다.

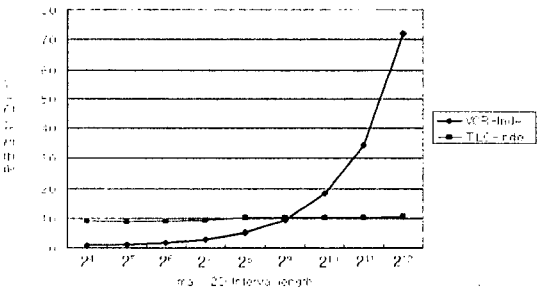

그림 7 Interval의 길이 증가에 따른 검색 성능 비교

6. 결론 및 향후 연구

본 논문에서는 RFID 미들웨어에 등록되는 ECSpec의 실시간 이벤트 필터링을 위한 질의 색인 기법을 제안하였다. 제시한 색인 기법은 ECSpec을 위한 Predicate가 가지는 Long Interval 문제를 해결하기 위해, 가상 분할 구조와 그리드 기반 셀 분할 구조를 병행하여 처리함으로써 삽입과 검색 성능을 향상시킨다. 향후 연구로써 Predicate의 길이에 따른 가상 분할 구조 및 셀 구조의 최적화된 길이를 실험적으로 측정하는 것이 필요하다.

7. 참고 문헌

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

[2] S. R. Madden, M. A. Shah, J. M. Hellerstein, and V. Raman, "Continuously adaptive continuous queries over streams." In Proc. of ACM SIGMOD, 2002.

[3] Kun-Lung Wu, Shyh-Kwei Chen, Philip S. Yu, "VCR indexing for fast event matching for highly-overlapping range predicates.", SAC 2004, pp740-747, 2004.

[4] Kun-Lung Wu, Shyh-Kwei Chen, Philip S. Yu: Processing Continual Range Queries over Moving Objects Using VCR-Based Query Indexes. MobiQuitous 2004: 226-235

[5] D.V.Kalashnikov, S.Prabhakar, W.G.Aref, and S.E.Hambrusch, Efficient evaluation of continuous range queries on moving objects. In Proc. of 13th DESA, 2002.