■ 2006년도 학생논문 경진대회 수상작

RFID 태그 데이타의 연속질의 처리를 위한 질의 색인

(A Query Index for Processing Continuous Queries over RFID Tag Data)

석 수 욱 † 박 재 관 †† 홍 봉 희 †††
(Su wook Seok) (Jae kwan Park) (Bong hee Hong)

요 약 RFID 기술 표준화를 추진하고 있는 EPCglobal의 ALE(Application Level Event)는 응용 애플리케이션과 RFID 미들웨어 사이의 인터페이스로서 ECSpec(Event Cycle Specification)과 ECReports (Event Cycle Reports)를 정의하고 있다. ECSpec은 애플리케이션이 원하는 태그 데이타에 대한 명세이며, ECReports는 ECSpec이 제시한 조건에 적합한 결과를 보고하기 위한 것이다. ECSpec은 애플리케이션이 미들웨어에 등록하는 이벤트 여과를 위한 명세로서 일정 시간 동안 반복적으로 수행되는 연속질의(continuous query)와 유사한 특성을 가진다. ECSpec을 연속질의로 변환할 때 해당 질의가 가지는 술어(Predicate)는 매우 긴 길이를 가지는 간격이 된다. 기존 질의색인들은 긴 간격 데이타에 의해 삽입과 검색 성능이 저하되는 문제점이 있다.

이 논문에서는 ECSpec을 연속질의의 형태로 변환하고 해당 질의가 가지는 술어인 2차원 간격의 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. TLC-index는 긴 간격을 큰 크기를 가지는 셀 분할 구조로 분할 삽입함으로써 저장 공간의 소모를 줄이고 삽입 성능을 향상시킨다. 또한 짧은 간격들을 짧은 길이를 가지는 가상 분할 구조들로 분할 삽입함으로써 그리드 방식이 가질 수 있는 부분적 겹침을 제거하여 검색 성능을 향상시킨다.

키워드 : 질의색인, 연속질의, 전자태그, 데이타스트림

Abstract The ALE specification of EPCglobal is leading the development of RFID standards, includes the Event Cycle Specification (ECSpec) describing how long a cycle is, how to filter RFID tag data and which reader is interested in. The ECSpec is a specification for filtering and collecting RFID tag data. It is registered to a middleware for long time and is evaluated to return results satisfying the requirements included in it. Thus, it is quite similar to the continuous query. It can be transformed into a continuous query as its predicate in WHERE clause is characterized by the long interval. Long intervals cause problems deteriorating insertion and search performance of existing query indices.

In this paper, we propose a TLC-index as a new query index structure for long interval data. The TLC-index has hybrid structure that uses the cell construct of CQI-index with the virtual construct of VCR-index for partitioning long intervals. The TLC-index can reduce the storage cost and improve the insertion performance through decomposing long intervals into one or more cell constructs that have long size. It can also improve the search performance through decomposing short intervals into one or more virtual constructs that have short size enough to fit into those intervals.

Key words : Query Index, Continuous Query, RFID, Data Stream

· 이 논문은 2006년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 발아 수행된 연구임(지방연구중심대학육성사업/차세대물류IT기술연구사업단)

† 비 회 원 : 삼성전자 정보통신총괄
suwook.seok@gmail.com
†† 학생회원 : 부산대학교 컴퓨터공학과
jkpack@pusan.ac.kr
††† 종신회원 : 부산대학교 전자전기정보컴퓨터공학부 교수
bhhong@pusan.ac.kr

논문접수 : 2006년 5월 18일
심사완료 : 2006년 12월 29일

1. 서 론

EPCglobal은 RFID기술을 사용하여 전 세계의 물류 환경을 통합하기 위한 표준화를 진행하고 있으며 특히, RFID 미들웨어 시스템의 표준으로써 초기에 Savant[1]라 불리는 내부 구조에 중점을 둔 표준을 제시하였으나, 현재 미들웨어의 인터페이스 표준화에 중점을 둔 ALE Specification[2]을 제시하였다. ALE는 응용 애플리케이션과 미들웨어 사이의 인터페이스로서 ECSpec과 ECReports를 정의하고 있다. ECSpec은 애플리케이션이 원하는 태그 데이타에 대한 명세이며, ECReports는 ECSpec이 제시한 조건에 적합한 결과를 보고하기 위한 것이다.

RFID 리더로부터 미들웨어로 전송되는 태그 데이타는 센서 네트워크의 데이타 스트림[3]과 매우 유사한 특성을 가진다. 태그 데이타는 RFID리더로부터 연속적으로 시간의 순서를 가지고 끊임없이 RFID 미들웨어 시스템으로 전송된다. 응용 애플리케이션은 미들웨어로 전달되는 대량의 태그 데이타들 중에서 필요한 데이타들만을 전달받기 위해 미들웨어에 ECSpec을 등록한다. ECSpec은 어떤 리더로부터 어떤 태그 데이타들을 필터링하여 보고받을 것인지에 대한 명세를 포함하고 있다. 이러한 ECSpec은 일정 시간 간격 동안 삽입되는 태그 데이타들에 대해 필터링 및 수집을 반복적으로 처리하여 결과인 ECReports를 생성하게 된다. 즉, ECSpec은 일정 시간 범위 동안 계속적으로 수행되어야 하는 연속질의[4]와 유사한 특성을 가진다. 최근 연속질의를 효율적으로 처리하는 방법으로 질의 색인 기법에 대한 연구가 활발히 진행되고 있다. 질의 색인은 연속질의를 색인하여 실시간으로 삽입되는 데이타 스트림을 필요로 하는 질의가 무엇인지를 빠르게 검색할 수 있도록 한다.

이 논문에서는 태그 데이타의 여과 및 수집을 실시간으로 처리하기 위해, 태그 데이타에 대한 연속질의인 ECSpec을 위한 질의 색인 기법인 TLC(Two-Level Construct) 색인을 제안한다. ECSpec이 가지는 질의의 술어(Predicate)는 논리적인 리더명과 태그의 여과 패턴으로 구성되는 2차원의 간격이 된다. 이러한 간격은 여과 패턴에 의해 매우 긴 길이를 가진다. 기존의 질의 색인은 이러한 긴 간격에 의해 삽입과 검색 성능이 급격히 저하되며 저장 공간의 소모가 증가하는 문제점이 있다. 이러한 문제점을 해결하기 위해 TLC 색인은 그리드 방식의 큰 크기를 가지는 셀 분할 구조와 선분 모양의 가상 분할 구조를 병행하여 사용한다. 색인에서 긴 간격의 정의는 셀 분할 구조의 길이보다 크거나 같은 길이를 가지는 간격이다. 제안하는 색인은 긴 간격을 다양한 크기의 레벨을 가지는 셀 분할 구조로 분할 삽입함으로써 저장 공간의 소모를 줄이고 삽입 성능을 향상시킨다. 또한 짧은 간격을 짧은 길이를 가지는 가상 분할 구조들로 분할 삽입함으로써 그리드 방식이 가질 수 있는 부분적 겹침을 제거하여 검색 성능을 향상시킨다.

이 논문의 구성은 다음과 같다. 먼저 2장에서는 질의 색인에 대한 개요 및 ECSpec 처리를 위한 연속질의 술어의 특성을 분석한다. 3장에서는 2장에서 분석한 특성에 따른 기존 색인들의 문제점을 기술한다. 4장에서는 제안하는 질의 색인 구조를 기술하고 5장에서 기존의 질의 색인과의 성능을 평가한다. 마지막으로 6장에서 결론 및 향후 연구를 기술한다.

2. RFID 시스템의 질의 색인과 연속질의 분석

2.1 질의 색인(query index) 개요

기존의 DBMS의 질의처리는 정적인 데이타를 저장하고 이러한 데이타들에 대해 일시적인 질의를 수행하는 방식이었다. 정적인 데이타 환경에서는 데이타의 양이 한정적이기 때문에 데이타베이스에 모든 데이타를 저장 가능하며 질의 수행 시 데이타의 빠른 검색을 위해서 데이타를 위한 색인 구조가 필요하다. 하지만 DSMS (DataStream Management System)와 같이 동적인 데이타 스트림을 처리하는 환경에서는 대용량의 데이타 스트림을 모두 저장한다는 것이 불가능하다. 따라서 응용 애플리케이션이 입력되는 데이타 스트림으로부터 원하는 결과를 얻기 위해 먼저 DSMS에 질의를 등록하고, 지속적으로 입력이 되는 데이타 스트림을 필요로 하는 질의들을 실시간으로 수행하는 방식으로 질의를 처리한다. 이때, 해당 데이타 스트림을 필요로 하는 질의를 실시간으로 검색하기 위해 질의 색인 기법이 등장하게 되었다.

질의를 색인하는 방법은 질의 조건절의 술어를 이용하는 것이다. 술어란 구조적 질의어인 SQL의 *WHERE* 절에 나타나는 조건으로서 참 또는 거짓을 판별할 수 있는 식이다. 아래의 CQL(Continuous Query Language)[5]의 예에서 술어는 "temperature BETWEEN 20 AND 30"부분이 되며 1차원 공간상에서 20에서 30까지의 범위를 가지는 간격으로 표현 가능하다. 따라서 아래의 예와 같은 질의를 색인하기 위한 질의 색인은 간격을 효율적으로 색인하기 위한 구조를 가져야 한다. 질의 색인은 실시간 질의 수행을 위한 질의 검색이 목적이므로 기본적으로 실시간 성능을 보장하기 위해 메인 메모리 기반 구조를 가진다. 질의 색인에서 검색은 데이타 스트림이 삽입될 때 수행되며, 색인 영역 상에서 데이타 스트림이 점으로 표현되기 때문에 오직 점 질의만을 포함한다.

```
SELECT Location FROM Environment [Range 1 Day]
WHERE temperature BETWEEN 20 AND 30
```

2.2 ECSpec 처리를 위한 연속질의 술어의 특성

ECSpec은 어떤 리더로부터 읽혀진 태그 데이타를 원하는지에 대한 논리적 리더명(Logical Reader Name, 이하 LRN), 어느 시간 간격 동안 데이타를 수집할 것인지에 대한 명세(ECBoundary Spec), 어떤 패턴을 가지는 태그 데이타들만을 여과할 것인지에 대한 여과 패턴(Filtering Pattern) 등을 포함한다. 그림 1은 EPCglobal의 ALE Specification에서 제시한 ECSpec의 예를 보이고 있다.

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	urn:epc:pat:gid-96:20.*.*		
group			

그림 1 ECSpec의 예

그림 2는 위의 ECSpec을 연속질의어 즉, CQL의 형태로 표현한 것이다. LRN을 readerID로 표현하고 여과 패턴을 tagEPC로 표현했다.

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

그림 2 Continuous Query로 표현한 ECSpec

그림 2에 나타나듯이 ECSpec을 연속질의의 형태로 변환 시 조건절의 술어를 이루는 속성은 readerID와 tagEPC가 된다. readerID와 tagEPC로 구성되는 2차원의 공간을 생각할 때 readerID는 축 상에서 서로 불연속적인 특성을 가지기 때문에 하나의 점으로 표현된다.

urn:epc:pat:gid-96:20.300.4000	Matches the EPC for UltraWidget serial number 4000.
urn:epc:pat:gid-96:20.300.*	Matches any UltraWidget's EPC, regardless of serial number.
urn:epc:pat:gid-96:20.*.[5000-9999]	Matches any XYZ Corporation product whose serial number is between 5000 and 9999, inclusive.
urn:epc:pat:gid-96:*.*.*	Matches any GID-96 tag

그림 3 여과 패턴의 예

그림 3은 EPCglobal의 ALE Specification에서 제시하는 여과 패턴의 예를 보이고 있다. 여과 패턴은 EPC의 패턴을 표현하는 것으로 EPC 포맷 중에서 GID-96 포맷을 보이고 있다. GID-96은 그림 4와 같이 전체가 96비트이며 헤더 부분이 8비트, 회사 필드가 28비트, 제품 필드가 24비트, 시리얼번호 필드가 36비트로 구성된다. 여과 패턴은 이러한 EPC를 구성하는 필드 중에서 헤더 부분을 제외한 부분의 패턴으로 구성된다. 따라서 여과 패턴을 이루는 비트 수는 GID-96 포맷의 패턴인 경우 96비트에서 8비트를 제외한 88비트가 된다.

	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

이 논문에서는 여과 패턴을 구성하는 전체 비트를 아래와 같이 하나의 십진 정수로써 축 상에 표현한다.

b87 b86... b60 b59 b58... b36 b35 b34... b0

urn:epc:pat:gid-96:10.20.1의 경우

· · · 1010. · · · 10100. · · ·1

$= 2^{63} + 2^{61} + 2^{40} + 2^{38} + 2^{0}$

위와 같이 변환할 때 tagEPC는 여과 패턴이 상수, 영역 또는 전체(*)로 구성되기 때문에 점 또는 간격으로 표현할 수 있다. urn:epc:pat:gid-96:20.300.4000의 경우 하나의 점으로 표현되며 urn:epc:pat:gid-96:*.*.*의 경우 tagEPC 전체 축의 길이와 일치하는 간격으로 표현 가능하다. 따라서 ECSpec을 처리하기 위한 연속질의의 술어는 그림 5와 같이 readerID와 tagEPC로 구성되는 간격으로 표현할 수 있다. 점의 경우는 간격의 길이가 0인 것으로 일반화할 수 있다. 이 후 논문에서 이러한 간격을 2차원 간격이라고 기술한다.

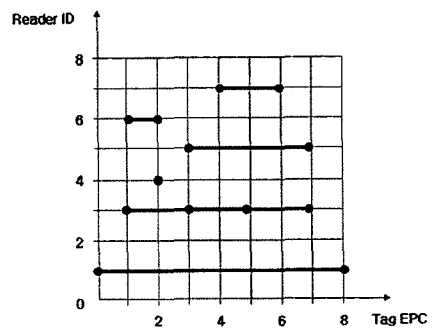


그림 5 readerID, tagEPC로 구성된 2차원 간격의 예

tagEPC는 EPC가 매우 큰 도메인(288)을 가지며 패턴의 형태가 20.*.*와 같이 회사 필드 또는 제품 필드 전체를 포함하는 경우가 많기 때문에 색인 공간상에서 매우 긴 간격이 되는 특징이 있다.

3. 관련 연구 및 문제 정의

이 장에서는 2차원 간격을 색인하기 위해 기존의 질의 색인 구조들을 포함하여 간격을 색인 가능한 기존의 다차원 색인 구조를 적용할 때 긴 간격 특성에 의해 발생하는 문제점을 기술한다.

먼저 2차원 간격을 색인 가능한 대표적인 다차원 색인 구조로써 R-tree 계열의 트리 구조의 색인들이 있다. 이러한 R-tree 계열의 색인들은 데이타 분할 방법의 대표적인 공간 색인으로서 공간 객체를 최소 경계 사각형(Minimum Bounding Rectangle)으로 표현하여 모든 데이타를 저장한다. 대표적인 색인으로 R-tree[10], R*-tree[11], R+-tree[12], Segment R-tree[13] 등이 있다. R-tree 와 R*-tree는 긴 간격에 의해 MBR간의 겹침(overlap)이 증가하여 검색 시 다수의 노드 접근이 필요하기 때문에 검색 성능이 급격히 떨어지게 되는 문제점이 있다. 특히 질의 색인이 대상으로 하는 점 질의의 경우 겹침에 의한 검색 성능의 저하 문제는 더욱 심각하다. R+-tree는 R-tree의 겹침 문제를 해결하기 위한 색인으로서 MBR을 분할 저장하는 방법을 사용하여 점 질의 성능을 높이고 있지만 긴 간격을 삽입할 때 다수의 데이타 분할에 의한 중복 저장이 필요하기 때문에 저장 공간 소모와 삽입 성능의 저하 문제가 발생한다. SR-tree는 긴 간격에 의한 겹침 문제를 해결하기 위한 방법으로 자식 노드에 완전히 걸쳐지는 긴 간격이 삽입될 경우, 데이타를 단말 노드에 저장하지 않고 걸쳐지는 노드의 부모 노드에 저장함으로써 긴 데이타에 의한 중첩을 최소화한다. 하지만 매우 긴 간격이 삽입될 경우, 간격을 잘라서 삽입하기 때문에 중복 저장에 의한 저장 공간의 소모가 크고, 삽입과 분할로 인하여 노드의 크기가 변할 경우 비 단말 노드에 저장되어 있는 간격 데이타를 상위 부모 노드 또는 하위 자식 노드로 이동해야 하는 부가적인 연산이 필요하기 때문에 실시간 질의 처리를 위한 질의 색인으로 적합하지 않다.

다음으로 2차원 간격을 위한 색인으로써 기존의 질의 색인을 적용할 때의 문제점을 기술한다. 질의 색인은 실시간 질의 수행을 목적으로 하기 때문에 빠른 검색 속도를 위해 기본적으로 메인 메모리 기반의 색인 구조를 가지며 데이타 기반이 아닌 공간 기반의 구조를 사용하여 점 질의 성능을 최우선으로 고려한다. 이러한 질의 색인 구조로써 VCR(Virtual Construct Rectangle) 색인[6,7]과 CQI(Cell-based query index) 색인[8]이 대표적이다. VCR 색인은 이동체에 대한 영역 질의를 색인하는 질의 색인 기법이다. 영역 질의의 술어는 2차원의 사각형으로 표현이 되는데, 이것을 미리 정의된 VCR (Virtual Construct Rectangle)이라는 사각형 구조의 가상 분할 구조를 이용하여 분할하고 해당 구조와 연관되는 노드의 ID 리스트에 질의 ID를 삽입하는 방식을 사용한다. VCR 색인의 가상 분할 구조는 (x, y, l_x, l_y)로 표현되며 x, y는 VCR의 좌측 하단 점을 나타내고 우측 상단 점은 (x + l_x, y + l_y)가 된다. 이때 l_x와 l_y는 2의 지수 승으로 표현이 되며 $l_x \in \{1, 2^1, \cdots, 2^k\}$, $l_y \in \{1, 2^1, \cdots, 2^k\}$의 범위를 가진다. 가상 분할 구조의 최대 크기는 2^k가 되며 이것을 VCR의 Max. Side Length라고 부른다. 그림 6은 Max. Side Length가 2^2 일 때 (0, 0)이 가지는 9개의 VCR구조를 보이고 있다.

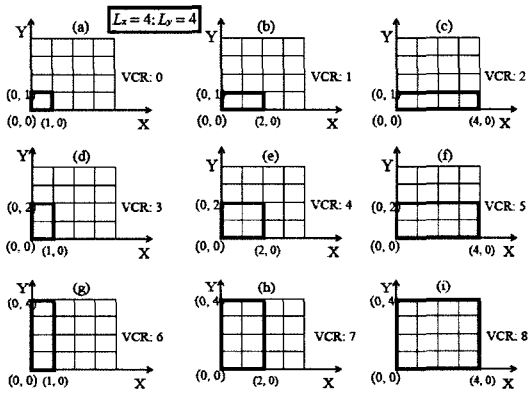


그림 6 Max. Side Length = 4일 때 (0,0)이 가지는 9개의 VCR 예

그림 7은 영역 질의 Q1의 술어인 (3, 3, 11, 6)이 삽입될 때 그림 에서 보인 9개의 VCR을 이용하여 분할 삽입한 모습을 보여주고 있다.

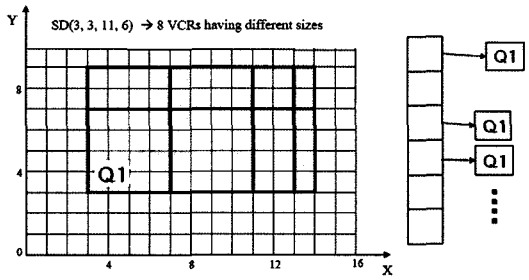


그림 7 VCR 색인에서의 삽입 예

VCR 색인에서 검색은 질의 점을 포함할 수 있는 모든 가상 분할 구조의 ID 리스트를 탐색하는 것이다. 이때, 질의 점을 (a, b)라고 하고 VCR의 Max. Side Length인 L_x와 L_y의 값이 2^2 일 때 아래 그림 8과 같이 점 (a, b)를 포함할 수 있는 가상 분할 구조는 좌측 하단의 점(a - L_x, b - L_y)에서부터 우측 상단의 점(a +

L_x, b + L_y)에 걸쳐 존재하는 모든 VCR들이 된다. 이러한 모든 VCR들의 집합을 Covering VCR set이라고 부른다.

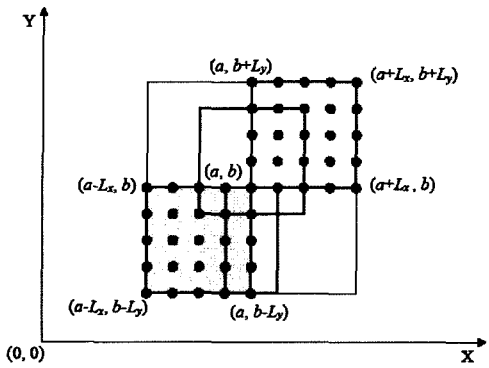


그림 8 질의 점(a, b)를 포함하는 Covering VCR set

VCR 색인과 같이 가상 분할 구조를 사용하는 질의 색인은 분할 구조의 최대 크기 즉, VCR의 경우 Max. Side Length의 설정에 따른 성능상의 상관관계가 존재한다. 위의 삽입, 검색 과정에서 볼 수 있듯이 VCR의 최대 크기가 커질 경우에 질의 영역을 분할 시 필요한 VCR의 개수가 작아져 삽입 시 필요한 저장 공간의 소모가 줄고 삽입 시간을 줄이는 이점이 있다. 하지만 이러한 최대 크기의 증가는 검색 성능을 저하시키는 요인이 된다. VCR 색인의 검색 방법은 질의 점을 포함할 수 있는 모든 Covering VCR set을 구하는 것이기 때문에 최대 크기인 Max. Side Length가 커질수록 점을 포함할 가능성이 있는 VCR들의 개수가 증가하기 때문이다.

긴 간격을 VCR 색인에 적용할 때 이러한 상관관계는 더욱 커지게 된다. 즉, VCR 구조의 Max. Side Length를 작게 할 경우 긴 간격을 삽입 시 매우 많은 수의 가상 분할 구조에 의해 분할되기 때문에 저장 공간의 소모가 커지고 삽입 시간이 크게 증가하게 된다. 반대로 Max. Side Length를 크게 할 경우에는 긴 간격을 위한 삽입 비용은 줄어들지만 검색 시 질의 점을 포함할 수 있는 Covering VCR Set이 매우 많아져 검색 성능이 급격히 저하되는 문제점이 있다. 예를 들어, ECSpec의 여과 패턴이 urn:epc:pat:gid-96:20.*.*일 때 tagEPC는 20.0.0에서 20.16777215.68719476735에 이르는 2^{60} 길이의 긴 간격이 된다. 이때, 2^{60}을 분할하기 위해 Max. Side Length를 2^{57}으로 한다면 2^3개 정도의 가상 분할 구조를 사용하는 분할이 필요하게 된다. 노드의 ID 리스트에 삽입되는 엔트리 하나를 위해 8 byte의 메인 메모리 공간이 필요하다고 할 때 $2^3 \times 8 = 64$ byte가 된다. 하지만 검색 수행 시 최대 크기 2^{57}인 Max. Side Length에 의해 아래의 그림 9와 같이 화살표 방향으로 질의 점을 포함할 수 있는 Covering VCR Set의 수가 급격하게 증가하게 되는 문제점이 있다. Covering VCR Set의 수를 줄이기 위해서 Max. Side Length를 작게 하면 삽입 시, 매우 큰 저장공간 소모와 삽입 비용이 발생하게 된다.

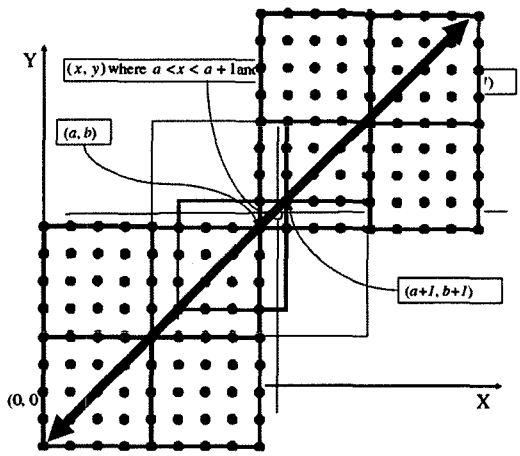


그림 9 큰 크기의 Max. Side Length로 인한 Covering VCR Set의 증가

CQI 색인은 그리드 기반의 셀 분할 구조를 사용하는 방식으로 VCR 색인과 마찬가지로 이동체에 대한 영역 질의를 색인한다. 그림 10과 같이 전체 영역을 일정 크기의 셀로 분할하고 각 셀에 연관되는 노드를 가진다. 셀 분할 구조는 일정한 크기를 가지고 있기 때문에 질의 영역을 분할 삽입 시 그림 와 같이 셀 구조에 완전히 겹쳐지는 부분과 부분적으로 겹치는 부분이 존재한다. 따라서 CQI 색인은 노드에 두 개의 리스트를 유지한다. 완전 겹침 리스트(Full List)는 분할된 영역이 셀에 완전히 겹쳐진 경우를 위한 것이며, 부분 겹침 리스트(Part List)는 분할된 영역이 셀에 부분적으로 겹쳐진 경우를 위한 리스트이다.

CQI 색인은 검색 시 부분 겹침 리스트를 탐색하여 얻은 결과인 질의 셋이 실제로 이동체를 포함하는지를 비교하기 위한 정제 단계가 필요한 특징을 가진다. CQI 색인은 셀 분할 구조의 크기 설정에 따라 성능상의 상관관계가 존재한다. 셀 분할 구조의 크기가 크면 클수록 질의 영역이 분할될 때 부분적 겹침이 발생할 가능성이 높기 때문에 부분 겹침 리스트로의 삽입이 많아지게 되며 이것은 곧 정제 단계 비용의 증가로 연결된다. 반대로 셀 분할 구조의 크기가 작을수록 다수의 분할이 필요하기 때문에 삽입 비용의 증가 및 저장 공간 소모가

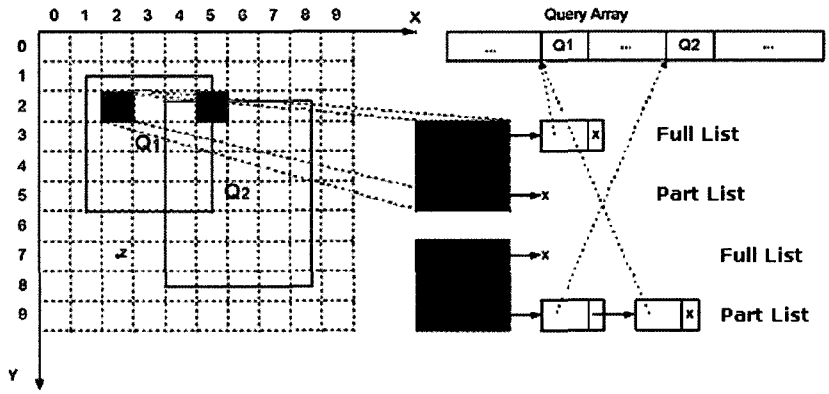


그림 10 셀 분할 구조를 사용하는 CQI 색인

증가하는 문제점이 있다.

이러한 CQI 색인의 상관관계는 긴 간격 데이타를 적용할 때 더욱 큰 문제점을 가진다. 긴 간격을 위한 큰 크기의 셀 구조를 사용할 수 있지만 셀 크기가 커질수록 짧은 간격에 의한 부분적 겹침이 많아지기 때문에 부분 겹침 리스트로의 삽입이 증가하여 정제단계로 인한 검색 비용이 증가하는 문제점이 있다.

4. RFID 미들웨어의 연속질의를 위한 질의 색인 구조

이 장에서는 RFID 미들웨어 시스템에 등록되는 ECSpec 즉, 연속질의를 위한 질의 색인 구조인 TLC(Two-Level Construct) 색인을 제안한다. TLC 색인은 기존의 가상 분할 구조 정책과 그리드 기반의 셀 분할 구조 정책을 병행하는 색인 구조를 가진다. 두 가지 서로 다른 분할 구조의 단점을 보완함으로써 긴 간격의 특징을 효율적으로 처리할 수 있는 삽입 및 검색 정책을 제안한다. 제안 방법을 설명하기 위해 먼저 아래의 용어를 정의한다.

4.1 기본 아이디어

가상 분할 구조와 셀 분할 구조 정책은 아래의 표 2와 같은 장단점을 가진다.

가상 분할 구조의 장점은 분할 구조의 크기가 2의 지수 승으로 1부터 Max. Side Length까지를 포함하기 때문에 항상 모든 질의의 술어를 완전히 겹치도록 분할 가능하다는 것이다. 따라서 가상 분할 구조 정책은 검색 과정에서 정제단계가 존재하지 않는 장점을 가진다. 셀 분할 구조의 장점은 셀 크기가 커지더라도 검색 시 질의 점을 포함하는 셀은 단 하나라는 점이다. 즉, 셀 크기의 증가가 검색 과정에서 여과 단계의 성능에 영향을

표 1 용어 정의

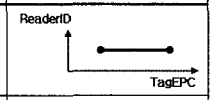
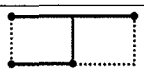
용어	설명	예
PI (Pattern Interval)	ReaderID와 TagEPC로 이루어지는 2차원의 Interval	
SPI (Short Pattern Interval)	TagEPC의 길이가 셀의 크기보다 짧은 Interval 또는 그것의 부분 Interval	
LPI (Long Pattern Interval)	하나 또는 그 이상의 셀 크기에 Fully Cover하는 긴 Interval 또는 그것의 부분 Interval	

표 2 가상 분할 구조와 셀 분할 구조의 장단점

	가상 분할 구조	셀 분할 구조
장점	- 분할 구조가 모든 간격을 완전 겹침 가능하여 정제 단계가 불필요.	- 셀 크기가 커져도 점 질의 시 검색은 하나의 셀에 매칭되어 여과 단계의 성능 우수
단점	- 분할 구조의 최대 크기가 커질 경우 검색 성능이 급격히 저하됨 - 분할 구조의 최대 크기가 작을 경우 저장공간소모가 커지고 삽입 비용이 증가됨	- 부분적 겹침이 많을 경우 정제 단계로 인해 검색 성능이 급격히 저하됨 - 셀 크기가 작을 경우 부분적 겹침은 줄어들지만 상대적으로 저장공간소모가 커지고 삽입 비용이 증가됨

미치지 않는다는 점이다.

이 논문에서 제안하는 TLC 색인은 긴 간격의 문제점을 해결하기 위해 LPI를 저장하기 위한 매우 큰 크기의 셀 분할 구조를 사용한다. 셀 분할 구조 정책은 셀의 크기가 커져도 검색 시 여과단계의 성능이 좋은 장점이 있기 때문에 긴 간격을 분할 가능하도록 크게 설정한다. 이때 문제가 되는 것이 큰 크기의 셀에 의한 부분적 겹침의 증가이다. 이러한 부분적 겹침을 제거하기 위해 TLC 색인은 셀 분할 구조에 부분적으로 겹쳐지는 SPI들을 가상 분할 구조를 이용하여 분할한다. 가상 분할 구조는 SPI들만을 분할하기 때문에 상대적으로 분할 구조의 Max. Side Length가 작아지게 되고 검색 성능에 최적화된다. TLC 색인은 위와 같이 두 가지 분할 구조의 장점을 이용하여 서로의 단점을 보완함으로써 두 분할 구조의 상관관계 문제를 해결하고 긴 간격 처리에 적합한 색인 성능을 보장한다.

4.2 색인 구조

그림 11은 TLC 색인의 구조를 나타내고 있다. TLC 색인은 두 가지 분할 구조를 가진다. 그리고 해당 분할 구조와 연관되는 노드와 ID 리스트를 가진다. tagEPC와 readerID로 이루어지는 색인의 전체 영역은 먼저 크기에 따라 여러 레벨을 가지는 그리드로 분할된다. 가상 분할 구조는 그리드 셀의 최소 가로 크기보다 작은 크기의 분할 구조로써 PI의 데이타 표현과 동일한 2차원의 간격 형태이다. 색인으로의 삽입 대상인 질의의 술어 PI는 (qid, tagEPC$^{\vdash}$, readerID, width)로 표현된다. qid는 질의의 식별자, tagEPC$^{\vdash}$는 tagEPC의 최소 경계 값, readerID는 LRN, 그리고 width는 tagEPC의 최대 경계 값에서 최소 경계 값을 뺀 값으로 간격의 길이를 나타낸다.

4.3 가상 분할 구조와 셀 분할 구조

SPI의 분할을 위한 가상 분할 구조는 2차원 간격 형태를 가진다. tagEPC를 x라 하고 readerID를 y라고 할 때 가상 분할 구조는 (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. length는 2의 지수 승 크기를 가진다. 즉, 위의 구조는 (x, y, 2^i-1)로 바꾸어 표현할 수 있다. 이때 i의 범위는 $0 \le i \le k$이며 2^k-1는 length의 최대값이 된다. 또한 i가 0일 때 length는 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, $N(x + y\ R_x) + i$가 된다. 그림 12는 가상 분할 구조의 최대 Length가 2^3-1일 때, (0, 0)이 가지는 4개의 가상 분할 구조를 보이고 있다.

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

4.4 데이타 삽입 및 삭제

PI의 최소 경계 값부터 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.

그림 13은 연속질의 술어 Q1의 삽입 예를 보이고 있다. Q1은 질의의 술어 PI로 (1, 1, 0, 7)을 가진다. TLC 색인이 가상 분할 구조 Max. Side Length로써 2^1-1을

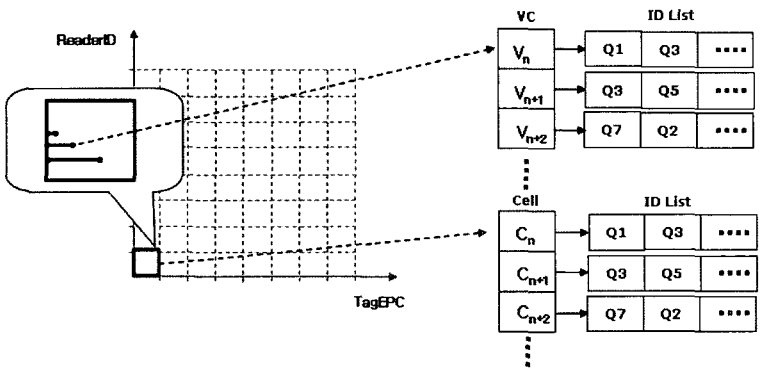


그림 11 TLC 색인의 구조

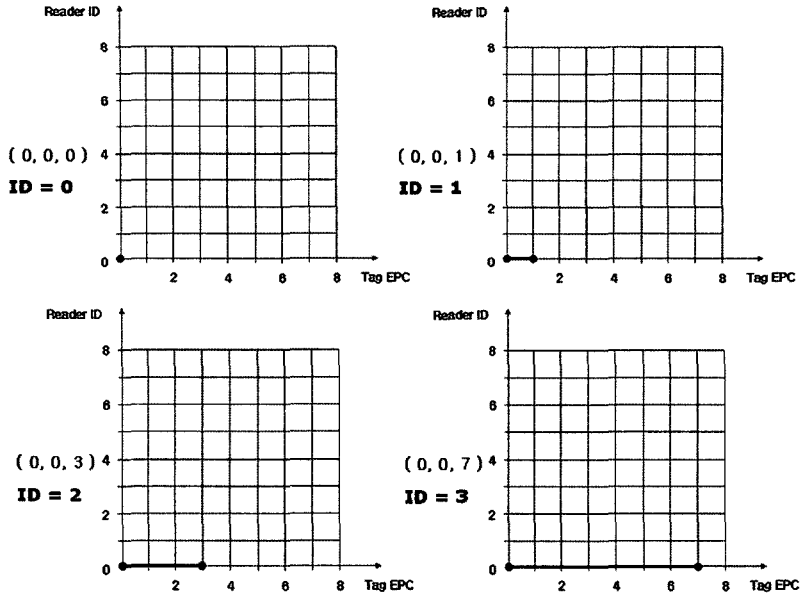


그림 12 Max Length 2^3-1 일 때 (0, 0)이 가지는 4개의 가상 분할 구조의 예

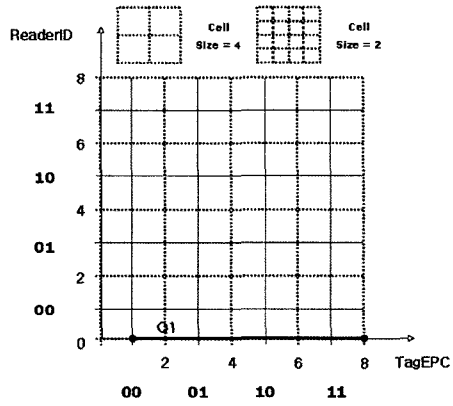


그림 13 연속질의 술어 Q1의 삽입 예

사용하고 2와 4의 크기를 가지는 두 개의 셀 분할 구조인 Cell(2)와 Cell(4)를 가진다고 할 때, Q1의 PI는 가상 분할 구조 (1, 0, 1)로 분할 가능한 SPI(1, 1, 0, 1), Cell(2)로 분할 가능한 LPI(1, 2, 0, 2), Cell(4)로 분할 가능한 LPI(1, 4, 0, 4) 이렇게 3개로 분할된다. 이와 같이 최소 경계 값부터 최대 경계 값까지의 모든 분할이 끝나면 사용된 모든 가상 분할 구조 또는 셀 분할 구조의 ID 리스트에 <QID, pointer> 형태의 엔트리를 삽입한다. 이때, 가상 분할 구조의 ID를 먼저 구하면 $N(x + y R_x) + i$ 공식에 의해 3이 되므로 ID=3에 해당하는 가상 분할 구조의 노드에 연결된 ID 리스트로 삽입한다. 다음으로 분할에 사용된 두 개의 셀 분할 구조의 ID를

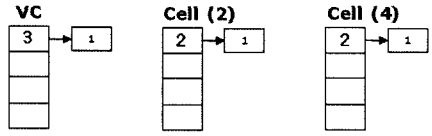


그림 14 각 분할 구조에서 ID 리스트에 엔트리가 삽입된 모습

Peano Key로 구하면 2가 되고 각각의 셀 노드에 연결된 ID 리스트로 삽입한다. 그림 14는 각 분할 구조의 노드에 연결된 ID 리스트로 엔트리가 삽입된 결과를 보여주고 있다.

알고리즘 1은 질의 술어 PI를 분할 삽입하는 알고리즘이다. 삭제는 삽입 과정과 동일한 방법으로 분할 구조의 ID 리스트를 찾고 해당 리스트를 탐색하여 엔트리를 삭제하면 된다.

4.5 데이타 검색

검색은 리더가 RFID 미들웨어로 태그 데이타를 보낼 때마다 반복적으로 수행된다. 이러한 태그 데이타는 질의 색인의 공간 상에서 점으로 표현이 되므로 검색은 오직 점 질의만을 포함한다. 검색 방법은 해당 점을 포함할 수 있는 모든 가상 분할 구조를 찾고 해당 구조의 노드에 연결된 ID 리스트를 탐색하는 것이다. 또한 해당 점을 포함할 수 있는 모든 레벨의 셀 분할 구조를 찾고 해당 셀 분할 구조와 연관되는 노드의 ID리스트를 탐색한다. 이때 질의 점을 포함하는 셀 분할 구조의 수는 레벨의 수와 일치한다. 알고리즘 5는 검색 알고리즘을 보이고 있다.

알고리즘 1 Insert

```
Algorithm Insert (QueryID qid, TagEPC ⊦ x, ReaderID y, IntervalLength width)
begin
  cellFitFlag = false
  /* 가장 작은 셀 분할 구조의 크기보다 작을 경우 가상 분할 구조로 분할 삽입*/
  if (width < minCellSize) then
    InsertByVirtualConstruct(qid, x, y, width)
  else
    /* 셀 분할 구조로 완전 겹침 가능할 경우 셀 분할 구조로 분할 삽입*/
    for each cellSize in cellSizeArray do
      if ( x % cellSize = zero ) and (width >= cellSize) then
        InsertByCell(qid, x, y, cellSize)
        x = x + cellSize
        width = width – cellSize
        Insert(qid, x, y, width)
        cellFitFlag = true
      end if
    end for
    /*셀 분할 구조에 완전 겹침, 부분적 겹침 부분을 나누어 재귀 호출*/
    if (cellFitFlag is false) then
      fitX = x + minCellSize – ( x % minCellSize )
      Insert(qid, x, y, fitX – x)
      width = width – minCellSize
      x = fitX
      Insert(qid, x, y, width)
    end if
  end if
end
```

알고리즘 2 InsertByVirtualConstruct

```
Algorithm InsertByVirtualConstruct
(QueryID qid, TagEPC ⊦ x, ReaderID y, IntervalLength width)
begin
  /* Interval을 분할하는데 필요한 모든 가상 분할 구조의 ID를 구한다*/
  decomposedSet = DecompositionWithVC(x, y, width)
  /*결과 ID에 해당하는 가상 분할 구조의 노드에 엔트리를 삽입한다*/
  for each vcID in decomposedSet do
    if ( there is a node associated with vcID ) then
      vcNode = GetNode(vcID)
      vcNode.add(qid)
    /*해당 ID의 노드가 존재하지 않을 경우 생성시킨다*/
    else
      CreateNewVcNode(vcID)
      vcNode = GetNode(vcID)
      vcNode.add(qid)
    end if
  end for
end
```

알고리즘 3 DecompositionWithVC

```
Algorithm DecompositionWithVC
(TagEPC ⊦ x, ReaderID y, IntervalLength width)
begin
  result = {}
  while ( width >= 0 ) do
    /*Interval(x, y, width)을 분할 가능한 가장 긴 가상분할 구조를 구한다*/
    virtualConstruct = FindMaximumVirtualConstruct(x, y, width)
    /*해당 ID를 결과 셋에 추가한다*/
    vcID = getVcID(virtualConstruct)
    result.add(vcID)
  end while
  return result
end
```

5. 성능 평가

이 장에서는 논문에서 제안하는 질의 색인 기법인 TLC 색인의 성능 평가를 위해 기존의 질의 색인 기법인 VCR 색인 및 CQI 색인과의 성능을 비교한다. 성능 비교의 기준은 메인 메모리 환경이므로 색인의 평균 삽입 시간과 저장 공간 비용이다. 또한 검색 성능을 평가하기 위해 점 질의에 대한 평균 검색 시간을 측정하였다. VCR 색인의 가상 분할 구조인 VCR과 CQI 색인의

알고리즘 4 InsertByCell

```
Algorithm InsertByCell
(QueryID qid, TagEPC x, ReaderID y, CellSize cellSize)
begin
  cellX = x / cellSize
  cellY = y
  cellID = GetPeanoKey(cellX, cellY)
  if ( there is a node associated with cellID ) then
    cellNode = GetNode(cellID)
    cellNode.add(qid)
  /*해당 ID의 노드가 존재하지 않을 경우 생성시킨다*/
  else
    CreateNewCellNode(cellID)
    cellNode = GetNode(cellID)
    cellNode.add(qid)
  end if
end
```

알고리즘 5 StabbingQuery

```
Algorithm StabbingQuery(TagEPC x, ReaderID y)
begin
  result = {}
  /*점(x, y)를 포함하는 모든 가상분할 구조의 노드를 탐색하여 qid를 구한다*/
  result += PointQueryToVirtualConstruct(x, y)
  /*점 (x, y)를 포함하는 모든 셀분할 구조의 노드를 탐색하여 qid를 구한다*/
  for each cellSize in cellSizeArray do
    result += PointQueryToCell(x, y, cellSize)
  end for
  return result
end
```

알고리즘 6 PointQueryToVirtualConstruct

```
Algorithm PointQueryToVirtualConstruct(TagEPC x, ReaderID y)
begin
  result = {}
  resultVcIDSet = GetAllCoveringVcSet(x, y)
  for each vcID in resultVcIDSet do
    if ( there is a node associated with vcID ) then
      vcNode = GetNode(vcID)
      result += vcNode.getAllEntry()
    end if
  end for
  return result
end
```

셀 분할 구조는 삽입 데이타인 2차원 간격에 적합하도록 TLC 색인의 분할 구조와 동일하게 수정하였다[14,15].

5.1 실험 환경

실험에 사용된 질의 술어 데이타 집합의 종류는 표 3과 같다. 패턴의 최대 길이는 회사, 제품, 시리얼번호의 값 범위 및 패턴 종류에 의해 결정된다. 예를 들어 urn:epc:pat:gid-96:20.30.*, urn:epc:pat:gid-96:20.30.[1000-5000]과 같이 회사와 제품 필드가 상수만을 포함하도록

알고리즘 7 PointQueryToCell

```
Algorithm PointQueryToCell(TagEPC x, ReaderID y, CellSize cellSize)
begin
  result = {}
  cellX = x / cellSize
  cellY = y
  cellID = GetPeanoKey(cellX, cellY)
  if ( there is a node associated with cellID ) then
    cellNode = GetNode(cellID)
    result += cellNode.getAllEntry()
  end if
  return result
end
```

표 3 실험에 사용된 질의 술어 데이타 집합의 종류

종류	질의 술어 개수	패턴 최대 길이	패턴 예
PD1	10000	2^{10}	*urn:epc:pat:gid-96:0.1.[503-979]*
PD2	10000	2^{16}	*urn:epc:pat:gid-96: 148421022.14376156.[39208-61184]*
PD3	15000	2^{16}	
PD4	20000	2^{16}	
PD5	25000	2^{16}	
PD6	30000	2^{16}	
PD7	35000	2^{16}	
PD8	40000	2^{16}	

하고 시리얼번호가 영역 또는 전체(*)를 포함하도록 데이타를 생성 시 패턴의 최대 길이는 전체(*)에 의해 2^{36}이 된다.

5.2 분할 구조 크기 설정에 따른 성능 비교

이 논문에서 제안하는 TLC 색인과 CQI 색인 그리고 VCR 색인의 성능을 비교하였다. 질의 술어 데이타는 PI의 최대 길이 2^{16} 크기를 가지는 PD2를 사용하였다. PI의 최대 크기는 3.1절에서 기술한 바와 같이 여과 패턴이 GID-96 포맷일 때 최대 2^{88}의 크기를 가질 수 있지만 2^{18} 이상의 길이에서 VCR 색인의 성능 저하로 인해 그래프 상에서 비교가 불가능하여 2^{16} 크기에서의 실험 결과를 통해 성능을 비교하였다. VCR 색인과 CQI 색인은 분할 구조 크기에 따른 성능상의 상관관계가 있기 때문에 아래의 표 4와 같이 분할 구조의 크기를 가변적으로 설정하여 실험하였다.

그림 15는 VCR 색인, CQI 색인 그리고 TLC 색인의 삽입 성능을 보여준다. 삽입 성능은 데이타 10000개의

표 4 분할 구조의 크기 설정 값

번호	가상 분할 구조 Max. Side Length	셀 분할 구조의 크기
1	2^6-1	2^9
2	2^7-1	2^{10}
3	2^8-1	2^{11}
4	2^9-1	2^{12}
5	2^{10}-1	2^{13}

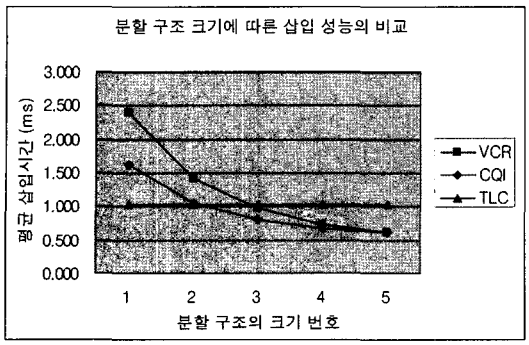


그림 15 질의 술어 데이타 PD2의 삽입 성능 비교

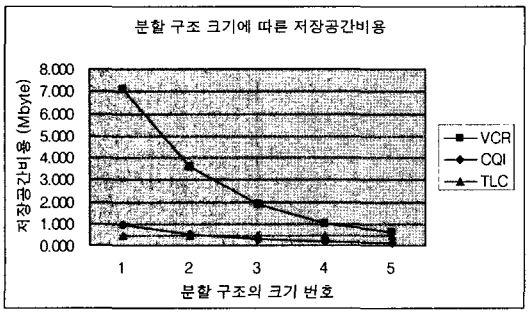


그림 16 질의 술어 데이타 PD2의 저장 공간 비용 비교

삽입 시 평균 삽입 시간을 측정한 것이다. TLC 색인의 삽입 성능은 VCR 색인보다 번호3 이하에서 더 좋은 성능을 보였다. 그리고 CQI 색인보다 번호 2이하에서 더 좋은 성능을 보였다.

그림 16은 질의 술어 데이타 PD2를 삽입 시 저장 공간 비용을 보이고 있다. TLC 색인은 VCR 색인보다 모든 분할 구조의 크기에서 더 작은 저장 공간의 소모를 보였다. 이것은 셀 분할 구조의 병행을 통해 상대적으로 질의 술어의 분할 개수가 작아지기 때문이다. TLC 색인과 CQI 색인의 저장 공간 비용은 번호 2와 3 사이에서 더 작은 셀 분할 구조 크기로 갈수록 TLC 색인이 우수했다.

그림 17은 질의 술어 데이타 PD2를 삽입한 상태에서 점 질의 10,000번을 수행하고 평균 검색 시간을 비교한 것이다. TLC 색인은 VCR 색인보다 모든 분할 구조의 크기에서 더 우수한 검색 성능을 보였다. 이것은 큰 크기의 셀 분할 구조의 병행으로 상대적으로 작은 크기의 Max. Side Length를 가지는 가상 분할 구조를 사용하기 때문에 Covering VCR Set의 수가 작아서 노드의 탐색 비용이 줄어들기 때문이다. 또한 점 질의에 해당하는 셀 분할 구조의 탐색 시에도 정제단계가 불필요하기 때문이다. TLC 색인은 CQI 색인보다 번호 1과 2사이에서 더 큰 셀 분할 구조 크기로 갈수록 검색 성능이

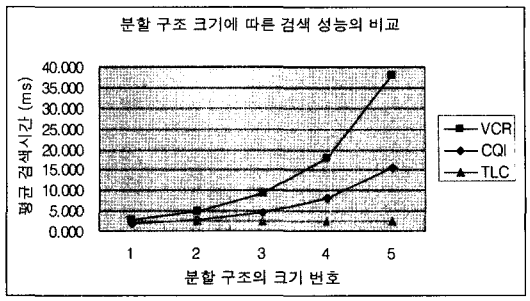


그림 17 질의 술어 데이타 PD2에서 점 질의 검색 성능 비교

더 우수했다.

위의 3가지 성능 평가를 종합하면 TLC 색인이 CQI 색인보다 번호 1과 2사이에서 번호 2와 3사이까지 저장 공간소모를 줄이면서 더 높은 검색 성능을 보이는 것을 확인할 수 있었다. 삽입 성능 역시 검색 성능이 유사할 때 TLC 색인이 더 좋은 성능을 보였다. VCR 색인은 삽입, 검색, 저장공간비용 모두 가장 낮은 성능을 보였다.

5.3 연속질의 개수에 따른 성능 비교

TLC 색인과 CQI 색인 그리고 VCR 색인의 성능을 연속질의의 개수를 증가시키면서 비교하였다. 질의 술어 데이타는 PI의 최대 길이 2^{16} 크기를 가지며 서로 다른 질의 술어 개수를 가지는 PD2부터 PD8까지를 사용하였다. VCR 색인은 가상 분할 구조의 Max. Side Length가 2^6-1인 VCR(6)과 2^9-1인 VCR(9) 두 가지를 실험하였고, CQI 색인은 셀 분할 구조의 크기가 2^9인 CQI(9)와 2^{12}인 CQI(12) 두 가지로 실험하였다.

그림 18은 연속질의 개수에 따른 삽입 성능을 측정한 결과이다. 이 그래프에서 볼 수 있듯이 TLC 색인이 VCR(6)과 CQI(9)보다 더 좋은 삽입 성능을 보였다.

그림 19는 질의 개수의 증가에 따른 저장 공간 비용을 측정한 결과이다. TLC 색인이 CQI(12)를 제외한 나머지보다 더 우수한 저장 공간 비용을 보였다.

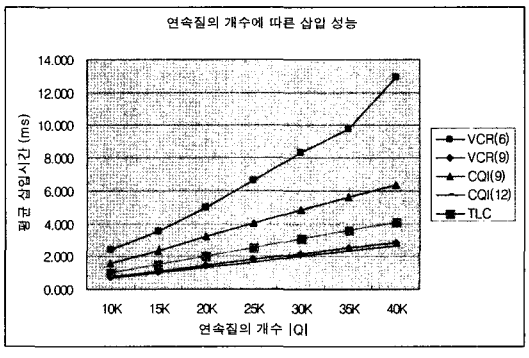


그림 18 연속질의 개수에 따른 삽입 성능 비교

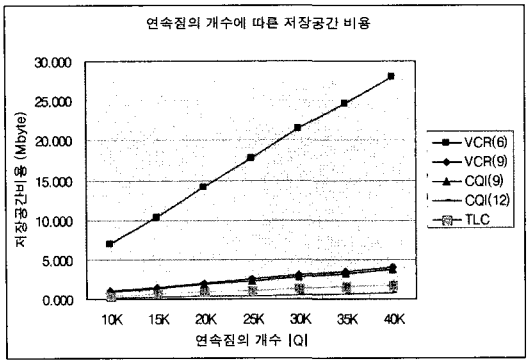


그림 19 연속질의 개수에 따른 저장 공간 비용 비교

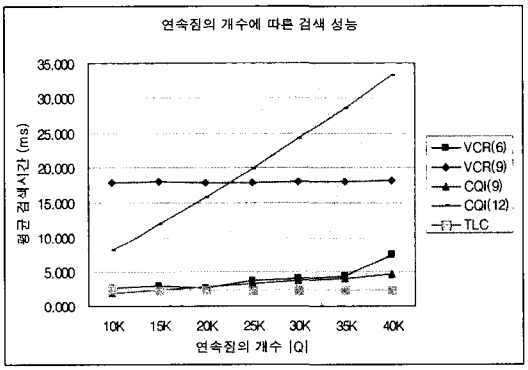


그림 20 연속질의 개수에 따른 점 질의 검색 성능 비교

그림 20은 연속질의 술어 개수 증가에 따른 검색 성능을 측정한 것이다. TLC 색인이 가장 우수한 검색 성능을 보였다. CQI(9)는 질의 술어 개수가 10K 개일 때 TLC 색인보다 더 좋은 검색 성능을 보이지만 TLC 색인에 비해 저장 공간 비용이 증가하고 삽입 성능이 저하되었다. 질의 술어 개수 15K 이상에서 TLC 색인은 CQI(9)보다 더 좋은 검색 성능을 보였다.

6. 결론 및 향후 연구

이 논문에서는 RFID 미들웨어에 등록되는 연속질의인 ECSpec을 실시간으로 검색하기 위한 질의 색인 기법으로 TLC 색인을 제안하였다. ECSpec을 연속질의로 변환했을 때 질의의 술어는 2차원의 간격 형태로 표현되며, 이러한 2차원 간격은 긴 간격의 특성을 가진다. 2차원 간격을 색인 가능한 기존의 다차원 공간 색인 및 질의 색인들은 이러한 긴 간격을 효율적으로 처리할 수 없는 문제점이 있었다. 특히 기존의 질의 색인들은 긴 간격을 처리 시 분할 구조의 크기 설정에 따른 상관관계에 의해 삽입, 검색, 저장공간비용을 함께 최적화시키지 못하는 문제가 있었다.

TLC 색인은 기존의 질의 색인 정책인 가상 분할 구조 정책과 셀 분할 구조 정책을 병행하여 서로의 단점을 보완하였다. 셀 분할 구조 정책은 셀의 크기가 커져도 점 질의 시 단 하나의 셀 노드만 탐색하면 되는 장점이 있고 가상 분할 구조는 모든 질의 술어를 부분적 겹침 없이 분할 가능하여 검색과정에서 정제단계가 불필요한 장점이 있다. TLC 색인은 셀 분할 구조에 완전히 겹쳐지는 긴 간격을 큰 크기의 셀 분할 구조로 분할하고 부분적 겹침을 발생하는 짧은 간격을 가상 분할 구조로 분할함으로써 셀 분할 구조 정책의 정제단계 비용을 없애고 가상 분할 구조의 Max. Side Length 증가에 의한 검색 성능 저하 문제를 해결한다. 이를 통해 긴 간격의 특성을 가지는 2차원 간격의 삽입 및 검색에 효과적인 성능을 보임을 실험을 통하여 확인하였다.

향후 연구로서 이 논문에서 언급하지 않은 2차원 간격의 또 다른 특성인 불연속적인 간격의 특성을 반영하기 위한 색인 정책의 연구가 필요하다. 또한 메인 메모리 기반 색인으로써 캐쉬 미스를 줄이기 위한 정책 연구가 필요하다.

참 고 문 헌

[1] Oat Systems & MIT Auto-ID Center, "Technical Manual: The Savant Version 0.1(Alpha)," Feb. 1, 2002.

[2] K. Traub and S. Bent, etc., The Application Level Events Specification, Version 1.0, EPCglobal Working Draft, Oct. 14, 2004.

[3] Donald Carney, Ugur Çetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Greg Seidman, Michael Stonebraker, Nesime Tatbul, Stanley B. Zdonik, "Monitoring Streams - A New Class of Data Management Applications," VLDB 2002, pp. 215-226, 2002.

[4] Samuel Madden, Mehul A. Shah, Joseph M. Hellerstein, Vijayshankar Raman, "Continuously adaptive continuous queries over streams," SIGMOD Conference 2002, pp. 49-60, 2002.

[5] Arvind Arasu, Brian Babcock, Shivnath Babu, John Cieslewicz, Mayur Datar, Keith Ito, Rajeev Motwani, Utkarsh Srivastava, Jennifer Widom, "STREAM: The Stanford Data Stream Management System," IEEE Data Engineering Bulletin 26(1), 2003

[6] Kun-Lung Wu, Shyh-Kwei Chen, Philip S. Yu, "VCR indexing for fast event matching for highly-overlapping range predicates," SAC 2004, pp. 740-747, 2004.

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

[8] Dmitri V. Kalashnikov, Sunil Prabhakar, Susanne E. Hambrusch, Walid G. Aref, "Efficient Evaluation of Continuous Range Queries on Moving Objects," DEXA 2002, pp. 731-740, 2002.
[9] Kun-Lung Wu, Shyh-Kwei Chen, Philip S. Yu, "Interval query indexing for efficient stream processing," CIKM 2004, pp. 88-97, 2004.
[10] 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, 1984.
[11] N. Beckmann and H. P. Kriegel, "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," Proc. ACM SIGMOD, pp. 332-331, 1990.
[12] Timos K. Sellis, Nick Roussopoulos, Christos Faloutsos, "The R+-Tree: A Dynamic Index for Multi-Dimensional Objects," VLDB 1987, pp. 507-518, 1987.
[13] C. Kolovson and M. Stonebraker, "Segment Indexes: Dynamic Indexing Techniques for Multi-Dimensional Interval Data," Proc. ACM SIGMOD, pp. 138-147, 1991.
[14] 석수욱, 박재관, 홍봉희, "RFID 미들웨어에서 이벤트 필터링을 위한 질의 색인 기법", 한국정보과학회 2005 가을 학술발표논문집(II) (제32권 제2호), pp. 19-21, 2005.
[15] 석수욱, 박재관, 홍봉희, "RFID 미들웨어에서 연속질의 처리를 위한 질의 색인 기법", 한국정보과학회 2005 하계 학술발표논문집(B) (제32권 제1호), pp. 28-30, 2005.
[16] Kun-Lung Wu, P. S. Yu, "Efficient Query Monitoring Using Adaptive Multiple Key Hashing," in Proc. of ACM Int. Conf. on Information and Knowledge Management, CIKM 2002, pp. 477-484, 2002.
[17] V. Gaede, O. G¨unther, "Multidimensional access methods," ACM Computing Surveys, pp.170-231, 1998.
[18] S. Prabhakar, Y. Xia, D. V. Kalashnikov, W. G. Aref, and S. E. Hambrusch, "Query indexing and velocity constrained indexing: Scalable techniques for continuous queries on moving objects," IEEE Trans. on Computers, pp. 1124-1140, 2002.
[19] J. Chen, D. J. DeWitt, F. Tian, and Y. Wang, "NiagraCQ: A scalable continuous query system for internet databases, In Proc. of the 2000 ACM SIGMOD Intl. Conf. on Management of Data, pp.379-390, 2000.

석 수 욱
2004년 부산대학교 컴퓨터공학과 졸업(공학사). 2006년 부산대학교 대학원 컴퓨터공학과 졸업(공학석사). 2006년~현재 삼성전자 정보통신총괄 무선사업부 사원. 관심분야는 RFID 시스템, RFID 미들웨어, 데이타스트림 처리

박 재 관
1999년 부산대학교 컴퓨터공학과 졸업(공학사). 2001년 부산대학교 대학원 컴퓨터공학과 졸업(공학석사). 2003년 부산대학교 대학원 컴퓨터공학과 박사과정 수료. 2001년~현재 ㈜사이버맵월드 부설 연구소 선임연구원. 관심분야는 지리정보 시스템, RFID 미들웨어, 모바일 GIS, 이동체 색인

홍 봉 희
정보과학회논문지 : 데이타베이스 제 34 권 제 1 호 참조