RFID 스트림 데이터를 위한 효율적인 비즈니스 이벤트 검출

노진석*, 복경수**, 유재수*
*충북대학교, **한국과학기술원

An Efficient Business Event Detection Scheme for RFID Data Streams

Jin-seok, Rho*, Kyung-su, Bok**, Jae-soo Yoo*
*Chungbuk National University, **Korea Institute of Science and Technology

요 약

RFID 미들웨어는 응용 서비스를 담당하는 어플리케이션이 등록한 비즈니스 이벤트를 빠르게 검출하여 이를 실시간적으로 전달할 수 있어야 한다. RFID 스트림 데이터는 대용량으로 발생되지만 어플리케이션에서 요구하는 비즈니스 이벤트를 항상 만족시키지는 못한다. 이에 따라, 미들웨어는 불필요한 이벤트를 처리하기 위해 많은 시간을 소요할 뿐만 아니라 대용량의 RFID 스트림 데이터를 정해진 시간 내에 처리하지 못하는 문제점이 발생한다. 이 논문에서는 대용량의 스트림 데이터에서 발생한 모든 이벤트를 처리하지 않고 어플리케이션이 등록한 비즈니스 이벤트를 구성하는 최소 조건을 만족하는 후보 집합을 찾는 연산을 제안한다. 이를 통해 후보 집합이 추출되면 실제 비교 연산을 통하여 조건을 만족하는 비즈니스 이벤트를 찾아내고 어플리케이션에 전달한다. 또한, 비트맵으로 이벤트의 발생을 표시를 하고 이 비트맵을 이용하여 실제 비즈니스 이벤트를 검출하는 기법을 제공한다.

1. 서 론

RFID(Radio Frequency IDentification)는 라디오 신호를 이용하여 객체에 부착된 태그(Tag) 정보를 인식하는 방법으로 최근 이슈가 되고 있는 USN(Ubiquitous Sensor Network)의 한 분야이다. 현재 RFID는 공장 자동화, 재고관리, 물류 유통 관리, 건강 의료분야 등에서 널리 활용이 되고 있다[1, 2]. RFID 시스템은 리더와 안테나, 태그, 미들웨어, 저장 시스템, 어플리케이션으로 구성되며 태그는 객체를 식별 할 수 있는 고유한 식별자 값을 가진다. RFID 리더는 대량의 태그를 동시에 인식하고 이를 미들웨어로 전달한다. 태그 정보는 미들웨어에서 필터링 과정을 거쳐 저장 시스템 또는 어플리케이션에 전달된다.

RFID 시스템은 태그의 정보를 얻기 위하여 다른RFID 시스템을 접근하여 처리하는 상황도 발생한다. 이때, 각각의 시스템들의 호환성이 문제가 대두 되게 되었다. 이에 따라, 적절한 표준이 필요로 하게 되었고 현재에는 EPCglobal에서 발표한 ALE(Application Level Event)기반의 EPCglobal Architecture Framework이 표준안으로 사용되고 있다[3]. EPCglobal Architecture Framework에 따르면 미들웨어는 EPCIS Capture와 EPCIS Repository, ONS등으로 구성된다. EPCIS Capture는 데이터의 필터링과 수집을 담당하며 리더가 발생시킨 데이터를 어플리케이션에게 가공 처리를 해서 전달하는 기능을 수행한다. EPCIS Repository는 발생된 데이터를 저장하며 ONS는 다를 시스템으로부터 EPC의 정보를 획득하는 기능을 수행한다.

대용량의 데이터를 처리하기 위해서 미들웨어에서는 특별한 처리 메커니즘이 필요하다. 어플리케이션에서 필요로 하는 데이터를 빠르게 처리하기 위하여 어플리케이션은 비즈니스 이벤트를 미들웨어에 등록하고 미들웨어는 어플리케이션이 등록한 비즈니스 이벤트의 패턴과 시간을 검사를 한다. 발생한 데이터가 어플리케이션에서 등록한 조건에 일치하면 어플리케이션에게 해당 이벤트를 전달한다[4, 5]. 이는 연속 질의(continuous Query) 혹은 XML 데이터 필터링 처리와 유사한 방식으로 이루어진다[6].

기존의 RFID 시스템에서 비즈니스 이벤트를 검출하는 연구는 발생되는 모든 기본 이벤트에 대하여 비즈니스 이벤트를 검출한다[6, 7]. 하지만 발생한 이벤트가 비즈니스 이벤트를 구성하는데 사용되지 않아도 이벤트에 대한 연산을 수행하기 때문에 연산의 양이 크게 증가하는 단점이 있다. 이러한 문제를 해결하기 위해 이 논문에서는 기본 이벤트가 발생할 때마다 비즈니스 이벤트를 검사하지 않고 실제 비즈니스 이벤트를 구성하는 기본 이벤트가 모두 발생할 때 비즈니스 이벤트를 검사하는 검출 방법을 제안한다. 제안하는 비즈니스 이벤트 검출 방법은 실제 비즈니스 이벤트를 구성하는 기본 이벤트를 색인으로 관리하고 각 비즈니스 이벤트는 연산의 특성에 따라 독립적인 트리를 구성한다. 실제 기본 이벤트가 발생하면 색인을 통해 비즈니스 이벤트를 검사하고 비즈니스 이벤트 추출 연산의 수행 여부를 판별한다. 따라서 이벤트의 발생량이 크게 증가해도 실제 연산양은 크게 증가하지 않는다.

이 논문은 2007년도 정부(과학기술부) 재원으로 한국과학재단(No.R01-2006-000-10809-0)과 2007년도 정부(교육인적자원부)의 재원으로 한국학술지능재단의 지원을 받아 수행된 연구임(지방연구중심대학육성사업/충북BIT연구중심대학육성사업단)

이 논문의 나머지 구성은 다음과 같다. 2장에서 기존에 제안된 RFID 스트림 데이터를 위한 이벤트 처리 연구들을 설명하고 3장에서는 기존 연구의 문제점과 제안하는 이벤트 검출 방식에 대하여 기술한다. 4장에서는 제안하는 방식에 대한 실험 평가 결과를 기술하고 마지막5장에서는 논문의 결론과 향후 연구에 대하여 기술한다.

2. 관련 연구

2.1 용어 정의

RFID 이벤트는 실제 리더에서 독립적으로 인식되어 미들웨어로 전달되는 기본 이벤트와 어플리케이션에서 필요로 하는 비즈니스 이벤트로 분류된다. 또한, 각 비즈니스 이벤트는 각 비즈니스 이벤트를 표현하기 위해 다수의 연산자들이 사용될 수 있다.

정의 1 기본 이벤트(Primitive Event)

기본 이벤트는 특정 시간에 특정 리더에서 인식된 태그 정보로 (r_{id}, t_{id}, t)로 구성된다. 이때, r_{id}는 태그를 인식한 리더의 식별자, t_{id}는 인식된 태그 식별자 그리고 t는 r_{id}에 의해 t_{id}가 인식된 시간을 의미한다.

예를 들어, 특정 태그가 부착된 상품o가 건물 입구에 설치된 리더 r에서 t 시간에 인식되었다면 이는 (r,o,t)라는 값으로 나타낸다. 이때, 태그에 부여된 EPC 코드의 특징에 의해 유사한 패턴을 갖는 태그들은 그룹으로 관리될 수 있다. 기본 이벤트는 리더가 태그의 EPC 코드를 인식함과 동시에 발생하며 발생 되거나 전혀 발생하지 않는다. 또한, 기본 이벤트는 어플리케이션에서 필요로 하는 비즈니스 이벤트를 구성하는 단위가 된다.

정의 2 비즈니스 이벤트(Business Event)

비즈니스 이벤트는 기본 이벤트들의 조합으로 이뤄지며 어플리케이션에서 의미를 가지는 이벤트이다. 이러한 비즈니스 이벤트는 기본 이벤트들을 다수의 연산자의 조합으로 구성된다.

예들 들어, 도서관에서 도서의 대여에 대한 비즈니스 이벤트를 처리한다고 하자. 도서 T_B은 처음에 도서관의 책꽂이 R_1에서 인식되고 기본 이벤트 $A(R_1,T_B,t_0)$로 표시된다. 이때, 도서관 사용을 인증 받은 사용자 T_C가 책을 들고 t_C 시간에 대여 출구 (R_2를 통과하게 되면 $B(R_2,T_C,t_C)$와 $C(R_2,T_B,t_C)$ 이벤트가 발생한다. 사용자가 도서관 출구 R_3을 t_E 시간에 통과하게 되면 $D(R_3,T_C,t_E)$와 $E(R_3,T_G,t_C)$ 이벤트가 발생하게 된다. 결과적으로 도서대여 비즈니스 이벤트는 A 이벤트가 발생한 후 B와 C 이벤트가 발생하고 D와 E 이벤트가 발생하는 조합으로 이루어진다.

RFID 스트림 데이터에서는 다수의 기본 이벤트를 조합하여 어플리케이션이 요구하는 비즈니스 이벤트를 생성할 수 있다. 이때 조합에서 사용될 수 있는 연산은 크게 5가지가 존재한다. AND 연산자는 두 개의 이벤트가 동시에 발생하는 것을 나타내기 위한 연산자로 이벤트 A와 이벤트 B에 대해 AND 연산자는 'A ∧ B'로 표현한다. OR 연산자는 두 개의 이벤트에서 하나 이상의 이벤트만이 발생해도 되는 것을 나타내기 위한 연산자로 이벤트 A와 이벤트 B에서 OR 연산자는 'A ∨ B'로 표현한다. NOT 연산자는 특정 이벤트가 발생하지 않는 것을 나타내기 위한 연산자로 이벤트 A에 대해 '!A' 로 표현한다. SEQUENCE 연산자는 특정 시간 내에 두 이상의 이벤트가 순차적으로 발생하는 것을 나타내기 위한 연산자로 이벤트 A와 이벤트 B에 대해 '(A, B: t)'로 표현한다. INTERVAL 연산자는 특정 시간 내에서 모든 이벤트가 발생하는 것을 나타내기 위한 연산자로 이벤트 A와 이벤트 B에 대해 '(A, B; t)'로 표현한다. 앞의 도서대여 이벤트는 $(A, (B \wedge C),(D \wedge E);10_{min})$과 같이 기본 이벤트와 연산자를 이용하여 비즈니스 이벤트로 표시된다.

2.2 기존의 이벤트 검출 기법

[6]에서는 연속적으로 발생하는 대용량의 RFID 데이터 스트리밍에서 이벤트를 처리하기 위한SASE를 제안하였다. SASE는 기본 이벤트의 패턴과 조건 그리고 시간 간격으로 구성을 통해 비즈니스 이벤트를 정의하기 위한 언어를 제안하였다.SASE는 5가지의 처리 과정을 수행하여 비즈니스 이벤트의 적합성을 검사한다. 먼저 등록된 비즈니스 이벤트는 오토마타 형태로 구성하고 이벤트가 발생함에 따라 이벤트의 상태 변화를 스택에 저장한다. 이때, 변화가 발생하기 전의 상태는 포인터로 연결된다. 이벤트 발생에 따라 상태의 변화가 오토마타의 최종 단계에 도착하게 되면 포인터를 따라 발생한 이벤트의 조합을 구성한다. 이 처리 과정을SSC(Sequence Scan and Construction)이라 한다. SSC를 통하여 구성된 이벤트 조합은 실제 비즈니스 이벤트의 후보 집합으로 SELECTION 연산을 수행하여 비즈니스 이벤트의 조건을 만족하는지 검사한다. 또한, WINDOW 연산을 통해 시간적인 조건을 만족하는지 검사한다. 마지막으로 NG(negation) 연산을 통하여 발생하지 않는 데이터를 처리하게 된다. 모든 조건을 만족하는 이벤트 조합은 비즈니스 이벤트를 등록한 어플리케이션에게 전달한다.

[7]에서는 그래프를 이용하여 RFID 스트림 데이터를 위한 비즈니스 이벤트를 처리기법을 제안하였다.[7]에서는 비즈니스 이벤트를 효율적으로 처리하기 위하여 비즈니스 이벤트를 정의하는 규칙을 제안하였고 비즈니스 이벤트를 구성하기 위해 기본 이벤트를 조합하기 위한 연산자를 정의하였다.이러한 연산자는 AND, OR, NOT과 같은 기본 연산자와 순차적인 이벤트를 묶는 SEQ, 시간 조건이 포함된 TSEQ 연산, 동일한 이벤트의 연속적인 처리를 위한SEQ+, TSEQ+ 연산 등이 있다. 실제 비즈니스 이벤트를 처리하기 위해[7]에서는 어플리케이션에서 등록한 비즈니스 이벤트는 계층적인 그래프로 저장한다. 그래프의 단말노드는 기본 이벤트로 구성되어 있으며 비 단말노드는 이벤트의 연산자로 구성되어 있다. 기본 이벤트가 발생하면 그래프의 단말 노드에서부터 이벤트의 조건을 검사하여 상위 노드로 전달된다. 이때, 발생하지 않는 이벤트의 연산자가 포함된 노드를 만난 경우는 상위노드에서 Pseudo 이벤트를 발생 시킨다. 발생 하지 않는 연산은 하위 노드에서 상위노드로 이벤트를 전달하는 시점을 선정 할 수 없기 때문에 발생된Pseudo 이벤트는 정해진 시간에 하위 노드로 질의를 전달하고 조건을 만족하는 경우 다시 상위 노드로 전달된다. 이벤트가 최종 루트노드에서 조건을 만족하게 되면 비즈니스 이벤트를 등록한 어플리케이션에게 최종 이벤트가 전달된다.

3. 제안하는 비즈니스 이벤트 검출

3.1 문제 정의

비즈니스 이벤트를 효과적으로 처리하기 위하여 기존에 많은 연구가 수행되어 왔다. 하지만 대용량의 데이터 스트림에서 이벤트를 처리하는데 연산이 크게 증가하고 이로 인해 주어진 시간 내에 이벤트 검출을 수행하지 못하여 시간 지연이 발생하는 문제점이 있다. [6]은 비즈니스 이벤트를 구성하는 연산자 중에서 SEQUENCE 연산에 집중되어 있다. 비즈니스 이벤트는 오토마타의 형태로 구성되고 각 단계와 단계별 상태를 저장한다. 그러나 비순차적인 연산은 오토마타의 단계와 단계별 상태의 구분이 어렵다. 또한 등록되는 비즈니스 이벤트를 구성하는 기본 이벤트들이 서로 공유될 수 없다. 각각의 비즈니스 이벤트마다 기본 이벤트가 연결되는 이전 상태의 포인터가 다르기 때문이다. 이는 비즈니스 이벤트의 저장 공간 측면에서도 효율적이지 못하다.

[7]은 이벤트가 발생함에 따라 노드의 조건을 만족할 경우 그래프의 하위 노드에서부터 상위 노드로 전달된다. 이는 비즈니스 이벤트의 조건을 만족하는 기본 이벤트가 모두 발생하지 않아 비즈니스 이벤트가 실제 발생하지 않는 조건에 대해 계속적인 연산을 수행하게 된다. 그림 1은 15초 시간 안에 최소한 e_1, e_2, e_3 혹은 e_1, e_2, e_4의 기본 이벤트가 발생해야만 비즈니스 이벤트의 조건을 만족하는 이벤트를 나타낸다. 그림 1에서 15초 이내에 e_1, e_2, e_4는 빈번히 발생하지만 e_2는 발생하지 않는다. 결과적으로 실제 비즈니스 이벤트의 조건을 만족하지 못한다. 이러한 상황에 시스템은 발생하지 않는 비즈니스 이벤트를 처리하기 위하여 불필요한 연산을 계속적으로 수행하게 된다또한, 단말 노드로부터 기본 이벤트가 발생하여 상위 노드로 전달되는 과정에서 각 상위 노드는 하위노드에서 전달한 이벤트를 저장하고 있어야 한다. 물론 하위노드 역시 다른 조합을 위하여 이벤트를 저장하고 있어야 한다. 이는 데이터를 중복하여 저장한다는 것을 의미한다.

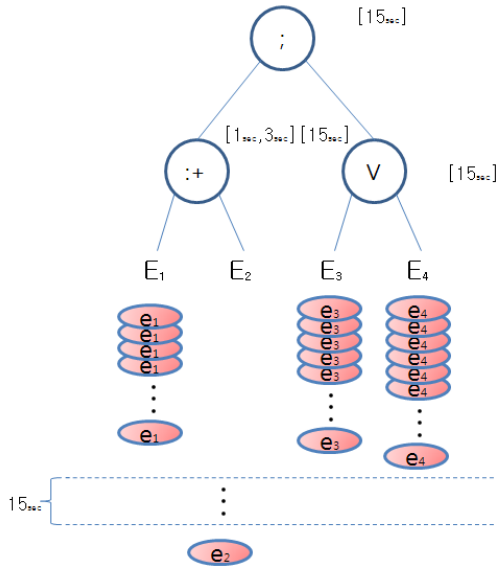
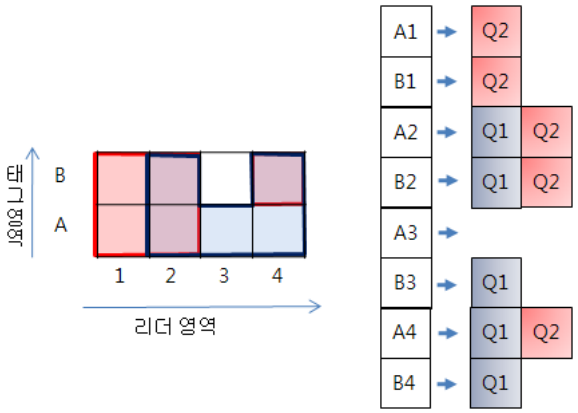


그림 1 그래프 방식의 이벤트 처리과정의 문제점

3.2 비즈니스 이벤트 관리

3.1에서 언급한 문제점을 해결하기 위하여 제안하는 비즈니스 이벤트 검출 방법은 비즈니스 이벤트를 구성하는 기본 이벤트들이 모두 발생할 때 비즈니스 이벤트의 후보 집합을 검출하도록 검출 시점을 조절한다. 비즈니스 이벤트에 대해 구성하는 모든 기본 이벤트들이 발생되었는지를 판별하기 위해 색인을 구성한다. 실제 비즈니스 이벤트는 연산자의 특성에 따라 트리 형태로 구성한다. 기본 이벤트가 발생하면 색인을 통해 비즈니스 이벤트를 구성하는 기본 이벤트의 발생 시간을 기록한다.만약 비즈니스 이벤트를 구성하는 모든 이벤트가 발생하면 실제 비즈니스 이벤트의 후보 집합을 추출하기 위해 실제 트리를 검사한다.

비즈니스 이벤트를 구성하는 기본 이벤트를 색인하기 위해 리더 축과 태그 축으로 가상의 그리드 영역을 구성한다.태그는 EPC 코드의 특징에 의해 유사한 패턴 단위로 그룹화 되며 리더 역시 그 위치와 역할에 의해 그룹화 한다. 각각의 비즈니스 이벤트들은 논리적 그리드의 영역에 매치되며 각 그리드 셀은 그 영역에 포함된 비즈니스 이벤트 식별자를 포함한다. 그림 2는 도서관에서 대여 반납에 대한 비즈니스 이벤트를 등록한 예제이다. 이때, 기본 이벤트 Ki는 태그 유형 K가 리더 i에서 인식된 것을 나타낸다. 그림 2(a)는 리더와 태그를 그룹 단위로 나누어 논리적인 셀을 구성한 상태를 나타내며 그리드에 비즈니스 이벤트가 등록된 상태를 나타낸다. 리더 영역의 1을 입구에 설치된 리더의 그룹, 리더 영역 2는 대출 및 반납실에 설치된 리더의 그룹, 리더 영역 3은 책 열람실에 설치된 리더 그룹, 리더영역 4는 출구에 설치된 리더 그룹을 나타낸다. 태그의 유형 A는 인증된 사용자의 태그 그룹이며 B는 도서들의 태그 그룹을 나타낸다. 그림 2(b)는 (a)에서 생성된 논리적 그리드를 배열로 저장하고 각각의 셀을 포함하는 비즈니스 이벤트 식별자를 링크로 유지한 것을 나타낸 것이다.



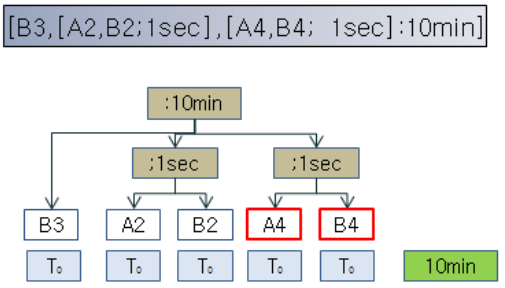
(a)가상 그리드 분할 (b)메모리상의 저장구조

그림 2 태그와 리더의 가상 그리드 영역

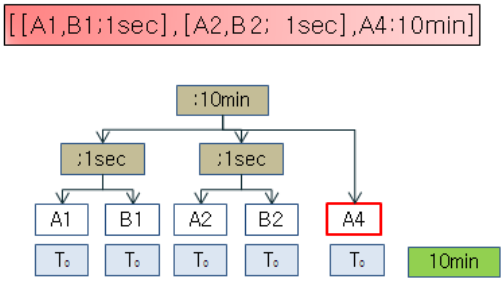
각각의 비즈니스 이벤트는 조건을 검사하는 연산자의 특성에 따라 트리 형태로 구성한다. 트리의 단말 노드는 비즈니스 이벤트를 구성하는 기본 이벤트를 저장하고 중간 노드에는 기본 이벤트를 조합하는 연산자와 시간 정보를 저장한다.비즈니즈 이벤트를 구성하는 기본 이벤트가 모두 발생하면 트리의 단말 노드에서부터 연산 조건을 검사한다. 조건을 만족하면 상위 노드로 전달하여 반복적으로 조건을 검사한다. 이때, 비즈니스 이벤트의 검사는 최대 시간 구간 내에서 발생된 기본 이벤트를 조합하여 후보 집합을 생성하는 것을 의미한다.

그림 3은 도서관에서 도서의 대여와 반납을 나타내는 비즈니스 이벤트를 나타낸 것이다. 이때, 대여와 반납 과정에서 도서와 사용자가 인식되는 순서는 비즈니스 이벤트에 연관이 없기 때문

에 INTERVAL 연산이 사용된다. 그림 3(a)은 대여에 대한 비즈니스 이벤트 Q1을 나타낸 것이다. 열람실 3에서 인식되었던 도서 B가 인증된 사용자 A와 함께 대여 및 반납실 2에서 인식되었다고 하자. 그 후 사용자가 도서를 가지고 출구4를 통과하면 사용자가 특정 도서를 대여한 이벤트가 발생한 것이다. 그림 3(b)은 반납의 이벤트 Q2를 나타낸다. 사용자 A가 반납할 도서 B를 가지고 출구 1에서 인식된 후 대여 및 반납실 2에서 역시 인식되었다고 하자. 그 후 출구에서 사용자 A만 인식되었다면 도서에 반납 이벤트가 발생한 것이다. 사용자가 도서를 반납하고 새로운 도서를 빌려 나가게 되면 두 개의 이벤트는 동시에 발생한다.



(a) Q1에 대한 비즈니스 이벤트 저장 구조



(b) Q2에 대한 비즈니스 이벤트 저장 구조

그림 3 비즈니스 이벤트 관리

비즈니스 이벤트를 최종적으로 구성할 수 있는 기본 이벤트를 트리거 셀이라고 한다. 위의 비즈니스 이벤트의 최상위 노드는 SEQUENCE 연산으로 연결되어있다. 따라서 위의 이벤트를 최종적으로 발생시키는 이벤트는 가장 마지막 이벤트들이 된다. 그러므로 이벤트 발생 조건을 검사하는 시점은 각 연산의 최상위 노드의 하위 노드 중에서 마지막 노드가 된다. Q1의 경우는 마지막 노드가 INTERVAL 연산이기 때문에 순서에 무관하다. 따라서 하위 노드인 A4, B4가 트리거 셀이 되며 Q2는 A4가 트리거 셀이 된다. 그림 3에서 각 이벤트의 최대 연산 시간은10분이다. 이는 사용자가 대여와 반납을 처리하는데 걸리는 최대 시간이 10분이라고 가정한 것이다. 비즈니스 이벤트의 트리거 셀이 이벤트에 의해 갱신되면 비즈니스 이벤트를 구성하는 모든 기본 이벤트의 발생 시간이 현재 시간으로부터 최대 연산 시간에 포함되는 여부를 검사한다. 이를 만족하면 적어도 하나의 비즈니스 이벤트가 발생할 가능성이 있으므로 실제 비즈니스 이벤트의 후보 집합을 검출한다.

비즈니스 이벤트의 후보 집합을 추출하기 위하여 기본 이벤트의 발생 시간별 발생한 이벤트 정보를 유지한다. 이를 비트맵이라 하며 시간에 따른 비트열의 형태로 저장한다. 전체 그리드 영역에 대하여 기본 이벤트가 발생한 영역은'1' 발생하지 않은 영역은 '0'으로 비트 값을 할당한다. 또한, 등록된 비즈니스 이벤트 중 최대 연산 시간과 동일한 시간 동안 저장 관리한다.실제 비즈니스 이벤트를 검사할 때는 이 비트맵에 슬라이딩 윈도우를 적용하며 각 연산 단위로 후보 집합을 찾는다.

3.3 알고리즘

비즈니스 이벤트 검출하기 위해서는 먼저 요청된 비즈니스 이벤트를 초기화 과정을 수행한다. 초기화 단계에서는 가상 그리드 영역을 생성하고 그리드 영역 내에 비즈니스 이벤트를 등록한다. 이러한 등록 과정은 비즈니스 이벤트를 관리하는 저장 구조의 단말 노드를 검사하여 해당 그리드 영역에 비즈니스 이벤트의 식별자를 할당한다. 비즈니스 이벤트의 등록을 마치면 비트맵을 저장할 영역을 할당한다. 그림 4는 초기화 알고리즘을 나타낸 것으로 모든 비즈니스 이벤트는 대하여 해당 셀에 비즈니스 이벤트의 식별자를 할당한다. 도서의 대여 반납 이벤트가 등록된 상태는 그림 2와 같고 각각 도서의 대여 이벤트와 반납 이벤트가 등록된 상태는 그림3과 같다. 기본 이벤트의 변경 시간은 T_0의 값으로 초기화한다.

```
1   algorithm init(query_list)  //초기화 알고리즘
2   GRID = 도메인 영역의 배열 생성
3   while(모든 비즈니스 이벤트의 셀을 순회할 때까지)
4     if(기본 이벤트 셀일 경우)
5       g[이벤트번호/도메인]에 Qid 생성하여 등록
6     end if
7     else(연산자일 경우)
8       MAX_TIME=연산자의 최대값 지정
9     end else
10  end while
11  BitMap = 그리드 영역/unit * MAX_TIME의 배열 생성
```

그림 4 초기화 알고리즘

그림 5는 실제 비즈니스 이벤트를 처리하는 과정을 알고리즘으로 나타낸 것이다. 현재 시간에 발생한 모든 이벤트를 그리드의 배열 순서에 따라 비트열로 할당된다(줄 4). 이렇게 구성된 비트열을 검사하면 1로 표시된 비트에 해당하는 그리드는 이벤트가 발생된 셀을 의미한다. 비트열을 따라 이벤트가 발생한 셀에 저장된 Qid를 찾아 이벤트의 발생 시간을 저장하게 된다(줄 7). 이때, 발생된 셀이 비즈니스 이벤트를 구성하는 트리거 셀인 경우에는 해당 비즈니스 이벤트를 구성하는 조합이 완벽히 이루어 졌는지를 검사하게 된다(줄 8,9). 이 연산은 비즈니스 이벤트를 구성하는 모든 기본 이벤트들의 발생 시간이 현재 시간으로부터 비즈니스 이벤트가 갖는 최대 시간 영역 안의 포함 여부를 검사하게 된다. 조건을 만족하는 것은 현재 시간에 비즈니스 이벤트를 구성하는 모든 기본 이벤트들이 발생하여 하나 혹은 그 이상의 비즈니스 이벤트가 발생할 수 있음을 의미한다. 이벤트의 발생 조건을 만족하는 경우에는 실제 비즈니스 이벤트의 발생을 검사해야 한다(줄 10,11). 실제 비즈니스 이벤트를 검출할 때는 각 시간 영역에서 저장된 비트열에 조합 연산자의 시간 영역에 해당하는 슬라이딩 윈도우를 적용하여and 연산을 수행한 결과가 모두 '1'의 값을 만족할 때 각 시간 영역별 후보 집합을 추출하여 구성한다. 슬라이딩 윈도우를 적용하여 조합할 때에는 비즈니스 이벤트를 구성하는 그래프의 하위 노드에서부터 상위 노드로 이동하며 만족하는 조합을 찾아낸다.

```
1  algorithm event_Dection(event_list) //이벤트 처리과정
2   BIT nowBit = 공백 비트열 생성
3   while(event_list 검사)
4      발생한 이벤트를 nowBit에 할당
5   end while
5   while(nowBit의 길이에 대해)
6     if(bit가 1일 때)
7        해당 bit에 대응하는 그리드를 찾고 그리드 영역에
         등록된 비즈니스 이벤트의 값을 갱신
8        if(비트가 비즈니스 이벤트의 트리거 셀일 경우)
9          check_query(Qid)
          //비즈니스 이벤트의 조건을 검사
10         if(검사 조건을 만족할 경우)
11           make_event_set(Qid)
             //비즈니스 이벤트를 생성
12         end if
13    end if
14  end while
15  발생한 모든 이벤트를 디스크에 저장
```

그림 5 이벤트 검출 알고리즘

그림 6은 T_5 시간에 기본 이벤트가 발생할 때 비즈니스 이벤트를 검출하는 알고리즘을 나타낸 것이다. T_5 시간에 발생한 기본 이벤트 B4는 그리드 영역을 검사하여 해당 영역에 등록된 비즈니스 이벤트 식별자를 검색한다. 비즈니스 식별자에 따라 해당 비즈니스 이벤트에 값을 변경한다. 이때 발생한 B4 이벤트는 비즈니스 이벤트 Q1의 트리거 셀이 된다. Q1은 트리거 셀이 갱신되었기 때문에 비즈니스 이벤트의 조건을 검사한다.Q1의 최대 시간 영역은10분이고 B4가 발생한 시간은 5초가 된다. 하지만 Q1을 구성하는 B2, A4는 초기 값이 할당된 후 변경되지 않았기 때문에 이벤트가 발생하지 않았음을 의미한다. 결국 비즈니스 이벤트의 조건을 만족하지 못하고 검사를 종료한다.

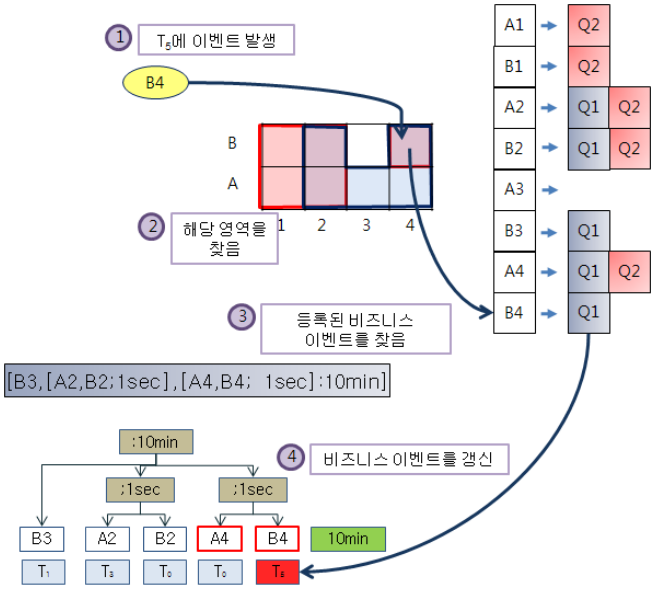


그림 6 T_5 시간에 발생한 B4 이벤트 처리 과정

만약 T_6 시간에 이벤트 B2, A4가 발생하면 Q1을 구성하는 기본 이벤트들이 모두 발생하게 된다. A4 역시 트리거 셀이기 때문에 Q1이 T_6 시간에 모든 기본 이벤트가 발생 하였는지를 검사한다. 이 조건이 만족할 경우 비트맵에 슬라이딩 윈도우를 적용하여 후보 집합을 찾는다. 그림 7은 트리의 노드를 검사하며 후보 집합을 찾는 과정을 나타낸다. ①②③의 과정에서는 부분적인 조건을 만족하는 후보 집합이 생성되지만 각 후보 집합에 대해 ④번 과정에서 순차적 발생 조건이 만족하지 못한다. 따라서, 비즈니스 이벤트는 발생하지 않는 것이다.

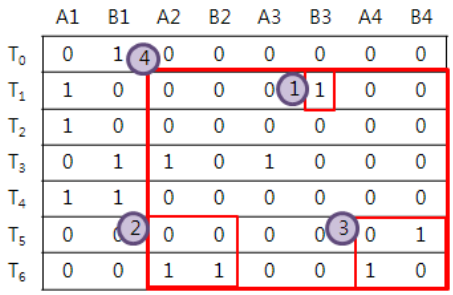


그림 7 비트열 검사

4. 성능 평가

이장에서는 제안하는 비즈니스 이벤트 검출 방법에 대한 성능 평가를 수행한다. 성능 평가는 .NET 환경에서 C++을 이용하여 구현되었으며 시스템 환경은 Pentium 2.8G에 1GB RAM이고 운영체제는 Windows XP를 사용하였다. 실험을 위해 이벤트의 발생 범위는 51200*51200으로 설정하고 태그 영역과 리더 영역을 각각 512*512개의 가상 셀로 분할한다. 등록된 비즈니스 이벤트는 2에서 4까지 임의의 레벨을 갖는 트리로 구성하였으며 100~10,000만개의 비즈니스 이벤트를 등록한다. 또한, 초당 발생하는 이벤트의 수를 10,000~1,000,000까지 변화시켜가면서 50초간의 평균 비즈니스 이벤트를 처리하는 시간을 측정한다.

그림 8은 등록된 비즈니스 이벤트 수를 변화에 따라 실제 검사해야 하는 비즈니스 이벤트와 필터링 조건을 만족하는 비즈니스 이벤트 수를 나타낸다. 실제 검사해야 하는 비즈니스 이벤트 수는 기본 이벤트가 발생함에 따라 접근하는 모든 비즈니스 이벤트의 수이다. 필터링 조건을 만족하는 비즈니스 이벤트 수는 기본 이벤트가 발생하여 비즈니스 이벤트에 접근 되고 비즈니스 이벤트의 발생 조건을 검사하여 후보 집합 연산을 수행하는 비즈니스 이벤트의 수를 의미한다. 비즈니스 이벤트 10,000개가 등록되어 있는 경우 실제 비즈니스 이벤트의 처리를 위해 매 초당 2,300개의 비즈니스 이벤트를 접근해야 한다. 기존의 방법은 2,300개의 비즈니스 이벤트에 대해 과거에 발생한 이벤트들과 조합하여 비즈니스 이벤트의 추출하는 연산을 수행하고 비즈니스 이벤트를 만족하면 이벤트를 발생 시키고 만족하지 않으면 중간단계까지 연산을 수행한다. 하지만 2,300개의 비즈니스 이벤트가 모두 비즈니스 이벤트를 발생시키지 못한다. 실제로 발생하는 비즈니스 이벤트의 수는 필터링을 통하여 나온 비즈니스 이벤트의 수인 150개 전 후가 되며 중간 단계까지의 연산은 후에 버려질 수도 있다. 제안하는 방법에서는 매 초당 150개의 비즈니스 이벤트를 검사하며 전처리를 통하여 약5~10%로 검사를 수행하는 비즈니스 이벤트 수가 줄었음을 의미한다.

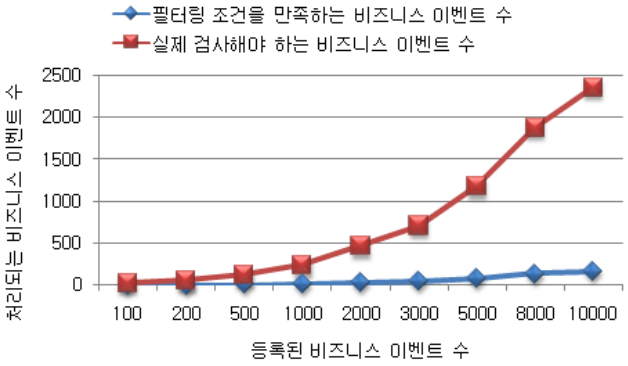


그림 8 비즈니스 이벤트 수에 따른 처리 이벤트 수

그림 9는 500개의 비즈니스 이벤트에 따라 처리되는 연산 시간을 나타낸다. 기존에 제안된 비즈니스 이벤트 검출 방법은 비즈니스 이벤트 수가 증가할 수 록 연산 시간이 매우 증가한다 [7]. 하지만 제안하는 기법은 전처리를 사용하여 비즈니스 이벤트가 생성될 가능성을 만족하는 시점에서 실제적인 비즈니스 이벤트를 검사하기 때문에 처리해야 할 이벤트의 수가 거의 선형적으로 증가한다.

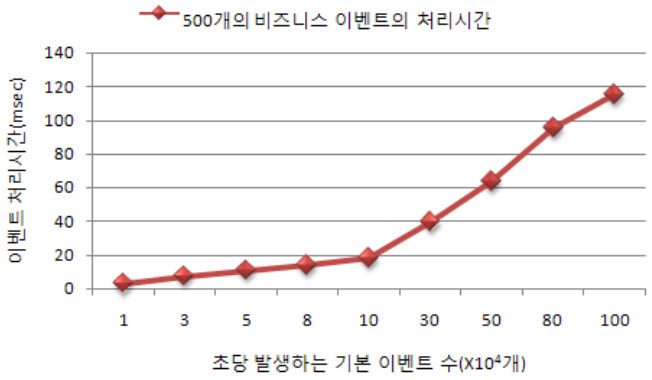


그림 9 기본 이벤트 수에 따른 비즈니스 이벤트 처리 시간

5. 결론 및 향후 연구

본 논문에서는 효과적으로 비즈니스 이벤트를 처리하기 위해 비즈니스 이벤트를 구성하는 기본 이벤트들이 모두 발생하는 시점을 검사하고 그 조건을 만족할 때 실제 비즈니스 이벤트를 검사하는 방법을 제안하였다. 실험을 통해 초당 발생하는 이벤트의 증가에 따라 검사해야 하는 비즈니스 이벤트의 수는 급격히 증가하는 반면 필터링을 통해 실제 검사를 수행해야 하는 비즈니스 이벤트 수는 크게 증가하지 않는 것을 보았다. 이 논문에서는 이벤트의 검사를 매번 수행하지 않고 비즈니스 이벤트의 구성 조건을 만족하는 순간에만 수행함으로 불필요한 연산을 방지하고 초당 처리하는 이벤트의 수를 감소시켜 연산의 효율성을 증가시켰다.

향후에는 그래프 방식으로 제안된 기존 연구와의 연산 시간을 비교하고 데이터를 디스크에 입출력하는 연산시간을 포함하여 실제 이벤트를 처리하는 모든 비용이 연산 시간 내에 수행하는지를 비교할 예정이다

참고문헌

[1] 유승화, “RFID 기술 현황 및 활용분야, 한국정보과학회지, 제23권, 제5호, pp.64-70, 2005.

[2] F. Wang and P. Liu, “Temporal Management of RFID Data”, Proc. VLDB, pp. 1128-1139, 2005.

[3] EPCglobal, "The EPCglobal Architecture Framework", 2005.

[4] C. Floerkemeier and M. Lampe, "RFID middleware design - addressing both application needs and RFID constraints", GI Jahrestagung, Vol.1, pp.277-281, 2005.

[5] T. Cheong and Y. Kim, "RFID Data Management and RFID Information Value Chain Support with RFID Middleware Platform Implementation", Proc. OTM Conferences, pp.557-575, 2005

[6] E. Wu, Y. Diao, and S. Rizvi, “High-Performance Complex Event Processing over Streams”, Proc. ACM SIGMOD, pp.407-418, 2006.

[7] F. Wang, S. Liu, P. Liu, and Y. Bai, “Bridging Physical and Virtual Worlds: Complex Event Processing for RFID Data Streams”, Proc. EDBT, pp.588~607, 2006.