韓國컴퓨터情報學會 論文誌
第17卷 第1號, 2012. 1.

2012-17-1-4-2

RFID와 센서 데이터 처리를 위한 미들웨어와 EPCIS 통합 설계

현 승 렬*, 이 상 정**

An Integrated Design of Middleware and EPCIS for RFID and Sensor Data

Seung-Ryul Hyun*, Sang-Jeong Lee**

요 약

RFID 태그 인식 정보와 센서 데이터는 지속적으로 변화하고, 위치에 따라 구분되며, 시간에 따라 변화하는 대량의 정보라는 면에서 유사한 자료라고 할 수 있으며, 두 자료가 함께 관리된다면 환경 변화에 따른 객체 인식 등의 융합 처리가 가능하다. RFID 미들웨어와 EPCIS 저장소가 통합된 시스템을 구현한다면 미들웨어의 기능과 저장소의 기능을 동시에 사용할 수 있으며, 미들웨어로부터 정보를 받아 가공 처리할 필요 없이 실시간 인식 정보 검색이 가능해진다. 본 논문에서는 일반적인 객체 인식과 환경 정보 변화에 의존하는 객체 검색을 실시간으로 가능하게 하기 위해, 장치로부터 정보를 지속적으로 읽어 데이터베이스에 저장하고, 수집된 자료를 기반으로 EPCglobal에서 표준으로 제안하는 ALE 기반 미들웨어와 EPCIS 저장소를 RFID와 센서 데이터를 함께 처리할 수 있도록 설계하고 구현한다.

▸ Keyword : EPCglobal, ALE, EPCIS, RFID, USN, 유비쿼터스, 센서네트워크

Abstract

RFID tag awareness information and sensor data continuously change, and are categorized with the position. They are able to similar data in the side, called massive data to change in time. If two data are managed together, a convergence process of object awareness along change of environment is possible. If RFID middleware and EPCIS repository realized the integrated system, it is usable with the functions of middleware and repository at the same time. The real-time awareness information retrieval is possible without process, getting information from another

• 제1저자 : 현승렬
• 투고일 : 2011 09. 23, 심사일 : 2011. 10. 18, 게재확정일 : 2011. 10. 24.
* 두원공과대학 인터넷정보과 (Dept. of Internet Information, Doowon University College)
** 순천향대학교 컴퓨터학부 (Dept. of Computer Science and Engineering, Soonchunhyang University)
※ 이 논문은 2011학년도 순천향대학교 교수 연구년제에 의하여 연구하였음

middleware. In this paper, it is able to continuously read information from RFID reader and sensor equipment and store to database in order to make general object awareness and an object retrieval dependent on an environmental information change possible by real time. ALE-compliant middleware and EPCIS repository proposing for standards at EPCglobal is designed and implemented to be able to deal with RFID and sensor data to bases on the collected data.

▸ Keyword : EPCglobal, ALE, EPCIS, RFID, USN, Ubiquitous, Sensor Network

Ⅰ. 서 론

RFID 태그 인식 정보와 USN(Ubiquitous Sensor Network)에서 환경의 변화를 감지하는 센서 데이터는 지속적으로 변화하고, 위치에 따라 구분되며, 시간에 따라 변화하는 대량의 정보라는 면에서 유사한 자료라고 할 수 있으며, 두 자료가 함께 관리된다면 환경 변화에 따른 객체 인식 등의 융합 처리가 가능하다.[1,2]

RFID 시스템의 미들웨어는 클라이언트의 요청에 따라 장치로부터 정보를 읽어오거나 태그에 정보를 저장해주는 역할을 하며, EPCIS(Electronic Product Code Information Services) 저장소는 캡처서비스(Capture Service)를 통해 미들웨어(들)로부터 이벤트 정보를 받아 데이터베이스에 저장하고, 클라이언트의 질의 요청에 대해 응답을 해주는 역할을 한다. 미들웨어와 EPCIS 저장소가 통합된 시스템을 구현한다면 클라이언트에서는 미들웨어의 기능과 저장소의 기능을 동시에 사용할 수 있고, 미들웨어의 읽기 기능을 미리 활성화시켜 놓으면 EPCIS는 미들웨어로부터 정보를 받아 가공처리할 필요 없이 실시간 인식 정보 검색이 가능해진다.

본 논문에서는 일반적인 객체 인식과 환경 정보 변화에 의존하는 객체 검색을 실시간으로 가능하게 하기 위해, 장치로부터 정보를 지속적으로 읽어 데이터베이스에 저장하고, 수집된 자료를 기반으로 EPCglobal에서 표준으로 제안하는 ALE 기반 미들웨어와 EPCIS 저장소를 RFID와 센서 데이터를 함께 처리할 수 있도록 설계하고 구현한다.

Ⅱ. 관련 연구

2.1 EPCglobal 표준

그림1은 EPCglobal에서 제안하는 표준들을 보여준다. RFID 시스템은 태그에 정보를 저장하거나 읽어오는 물리적인 장치, 장치로부터 필요 정보를 받아 정제해서 저장하는 ALE (Application Level Events) 기반 미들웨어, 미들웨어로부터 정보를 받아 질의 서비스를 제공하는 EPCIS (EPC Information System) 저장소, 그리고 웹상의 DNS (Domain Name System)와 유사하게 객체의 위치를 제공하는 ONS (Object Name Service) 등의 서브시스템으로 구분해서 구현할 수 있다. EPCglobal 은 RFID 시스템 구축에 필요한 서브시스템의 구현 방법을 제시하는 대신에 서브시스템간의 인터페이스에 대한 표준을 제안한다. [3,4,5,6]

ALE 버전 1.0에서는 장치로부터 자료를 읽어오기 위한 ALE API를 제안하였으며, 버전 1.1부터는 태그의 메모리의 일부분에 대한 이름을 지정해 주기 위한 태그 메모리 스펙(Tag Memory Specification) API, 장치로부터 자료를 읽어오기 위한 ALE 읽기 API, 태그에 정보를 저장하기 위한 ALE 쓰기(Writing) API, 리더의 논리적인 이름을 지정 및 관리하기 위한 ALE 논리 리더(Logical Reader) API, 그리고 장치에 대한 권한을 관리하는 접근 제어(Access Control) API 등으로 구분해서 표준을 제안한다. [7,8,9]

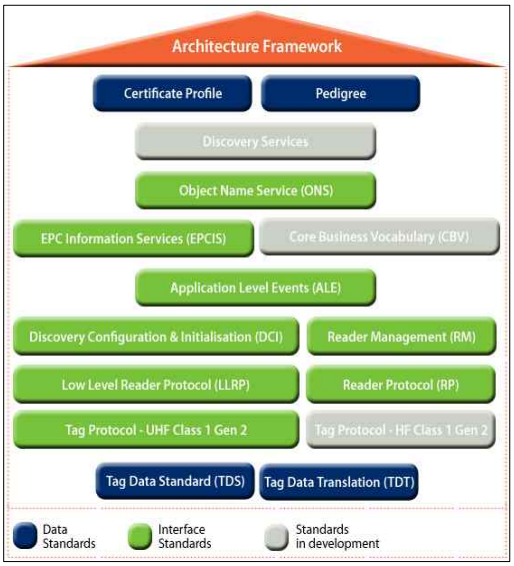

그림 1. EPCglobal 표준 개략
Fig. 1. EPCglobal Standard Overview

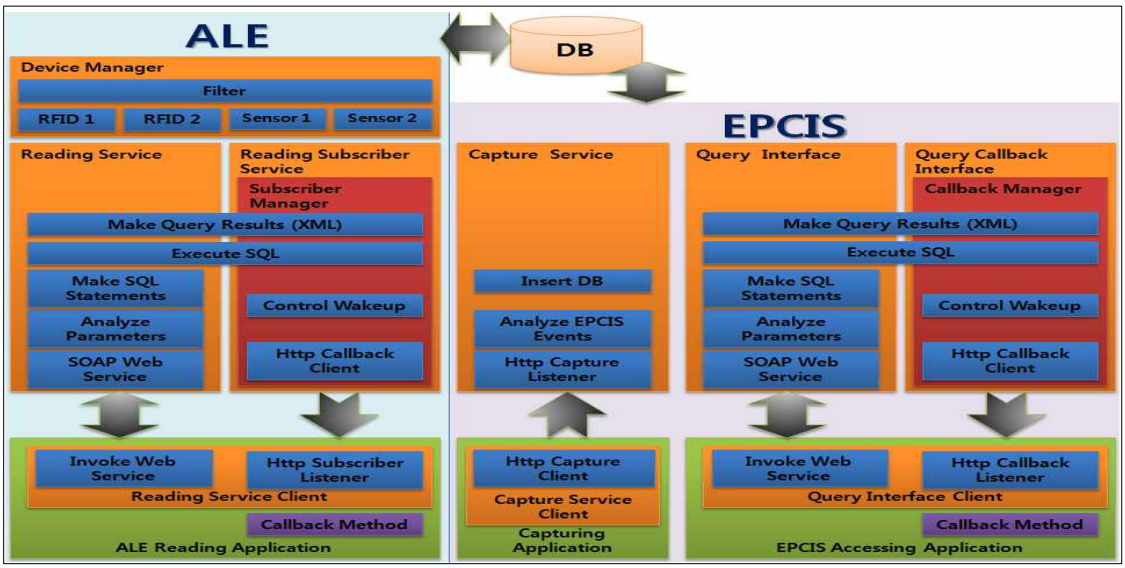

그림 2. 제안 시스템 구성
Fig. 2. Configuration of Proposed System

2.2 EPC 센서 네트워크

사물에 부착된 EPC를 통해 객체를 인식하는 RFID 시스템과 환경의 변화에 대한 감지 역할을 하는 USN 은 자료의 유사성에도 불구하고 별도의 서비스로 운영되어 왔다. EPC 센서 네트워크는 USN 과 RFID 기술을 효율적으로 연동할 수 있는 표준 인프라스트럭처를 제공함으로서 자료의 발견과 추적 등을 가능하게 하고 이기종 네트워크 사이에서의 데이터 호환을 보장하고 기술적 표준을 통한 연동을 제공한다. [10] 또한, EPC 센서 네트워크를 응용에 사용함으로서 환경 자료와 객체 자료의 관리가 용이해 질수 있다. [11]

III. 통합 시스템 설계

제안하는 통합 시스템은 참고문헌[2]의 EPCIS 저장소에 ALE 기반 미들웨어 부분을 추가해서 설계하였기 때문에 이 장에서는 ALE 미들웨어 관련 부분 위주로 기술한다.

3.1 시스템 구성

그림2는 제안하는 시스템의 전체 구성을 보여준다. 전체 시스템은 RFID 리더 및 센서 장치로부터 정보를 수집해서 데이터베이스에 저장하는 장치 관리자 (Device Manager)와 클라이언트의 요청에 따라 특정 장치의 감지 정보를 동기로 전달하는 읽기 서비스 (Reading Service)와 비동기적으로 전달하는 읽기 구독 서비스 (Reading Subscriber Service)로 구성된 ALE 기반 미들웨어 부분과 EPCIS 저장소 부분으로 나누어 구성한다.

3.2 데이터베이스 설계

그림3은 본 시스템을 위해 구성한 데이터베이스내의 테이블과 테이블간의 상관관계를 보여준다.

표 1. 테이블 목록
Table 1. Table List

테이블	설 명
ECSpecs	ALE 미들웨어에 정의된 ECSpec 정보
Subscribers	현재 Subscribe 서비스를 받고 있는 클라이언트의 URI 목록 관리
LocPath	경로명별로 물품의 장치간 이동 경로
LocPathName	물품의 이동 경로명 정의

표1은 제안한 시스템을 구현하기 위해 추가 설계한 테이블의 목록과 설명이다. ECSpecs 테이블은 ALE 클라이언트에서 정의한 ECSpec 들의 이름과 내용을 저장하고, Subscribers 테이블은 ALE 클라이언트에서 요청한 subscribe

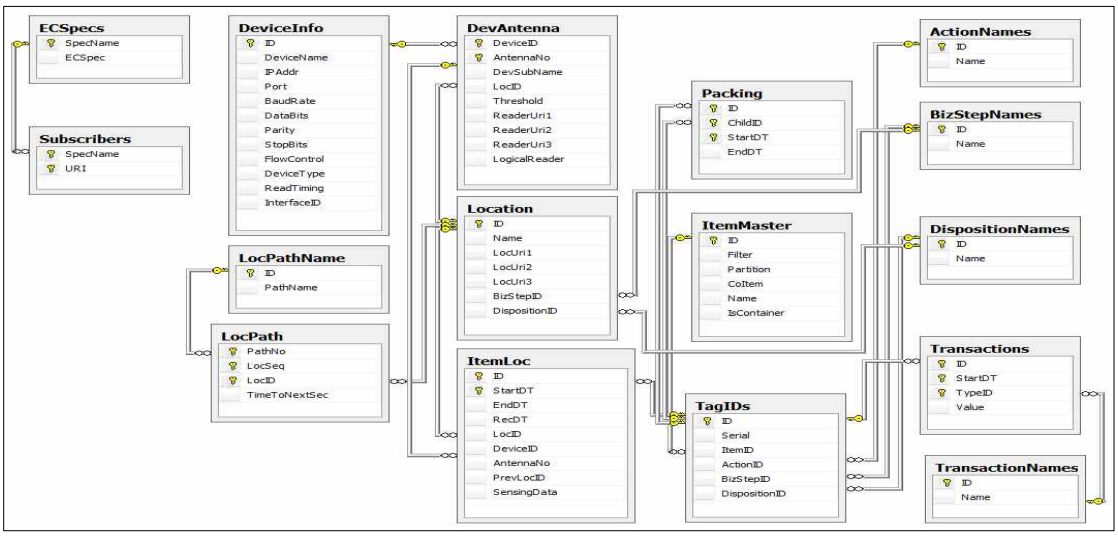

그림 3. 테이블간의 관계
Fig. 2. Relations among the Tables

요청에 대해 비동기적으로 전송하기 위한 클라이언트들의 URI (Uniform Resource Identifier)를 저장하며, 동일한 요청을 한 클라이언트가 여러 개일 경우 하나의 구독 서비스에서 처리할 수 있도록 수행 기간 동안 클라이언트의 정보를 저장한다. LocPathName 테이블은 여러 개의 물품의 이동 경로들을 표현하기 위해 경로명을 저장하며, LocPath 테이블은 각 이동 경로에서 물품이 이동할 수 있는 장치의 순서를 저장함으로서, 태그의 이동에 대한 정확한 검증을 수행할 수 있다.

표 2. 뷰 목록
Table 2. View List

뷰	설 명
V_AleTags	모든 태그와 센서 측정값에 대한 일시별 위치
V_AleTagsAddCur	V_AleTags 중에서 상태가 Additions 와 Current 인 자료만 선택
V_AleTagsDel	V_AleTags 중에서 현재 삭제된 자료만 선택
V_AleTagsOld	V_AleTags 중에서 과거의 자료만 선택

표2는 ALE 관련 질의 처리를 위해 SQL 구문 생성 시 구문을 단순하게 해 주는 역할을 하기 위해 작성한 뷰(View)의 목록이다.

3.3 ALE 미들웨어 설계

이 절에서는 전체 시스템 중에서 ALE 미들웨어와 관련된 부분에 대해 기술한다.

3.3.1 장치 관리자

장치 관리자는 미들웨어 시스템에 연결된 RFID나 센서 장치에 대한 관리를 수행한다. 그림4는 장치관리와 관련된 클래스 다이어그램이다. DeviceManagerForm 클래스는 미들웨어에 새로운 장치에 대한 세부 장치 정보와 통신 방식 등을 지정하고 각 장치에서 정보를 수집하는 프로세스를 시작 또는 종료시켜준다. 표준을 따르는 장치를 제외하고는 각 장치에 대한 통신 방식 또는 프로토콜이 서로 상이해서 각 장치와의 실질적인 자료 송수신은 개별적으로 구현해야할 필요가 있다. 이에, 각 장치에 대한 자료 송수신은 ReaderInterface를 구현한 다른 클래스에 위임한다. SetDeviceID() 메서드를 이용해서 프로세스에 대한 세부 수행 정보를 전달하고, Show() 메서드를 이용해서 현재의 자료 송수신 상태를 화면에 보여줄 수 있다. UserSelection 클래스는 미들웨어 사용자에게 소스 및 DLL(Dynamic Link Library) 형태로 제공되어 새로운 장치의 추가를 가능하게 한다.

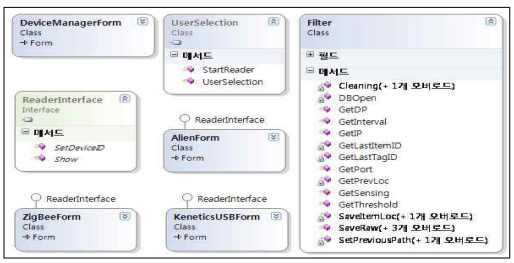

그림 4. 장치 관리자 클래스 다이어그램
Fig. 4. Device Manager Class Diagram

3.3.2 수집 자료의 정제

그림4의 Filter 클래스는 수집한 자료에 대한 정확성을 보장하고 저장 공간의 효율을 높이기 위해 아래와 같은 정제 과정을 수행한다.[12,13,14,15,16]

- 동일한 장소에서 같은 자료가 인식되었을 경우에는 종료일시만 갱신
- 지정된 이동 경로 중에서 거쳐 오지 않은 장소가 있을 경우에는 삽입
- 현재의 장소 이외의 모든 장소에서는 사라지게 함
- 상자처럼 포함 관계를 가진 물품에 대해서는 포장된 내부 물품의 위치는 저장하지 않음
- 센서 자료의 경우 임계값을 초과했을 경우에만 새로 저장
- 특정 위치 이동시 작업 단계와 조치 내역을 자동 생성

3.3.3 서버 측 읽기 서비스(Reading Service)

그림5는 서버 측 읽기 서비스 제공을 위한 클래스 다이어그램이다. 서버 측 읽기 서비스는 웹서비스(Web Service)를 통하여 제공하며, 클라이언트로부터 요청을 받으면 Reading 클래스를 통해 이미 저장된 데이터베이스로부터 결과를 보내준다. 이때 Reading 클래스는 SelectCommand 클래스를 이용해서 클라이언트의 요청에 필요한 SQL 구문을 만들어서 수행시키고 결과를 축출한다. ReadingSubscribe Service 클래스는 비동기 방식의 요청에 대한 쓰레드에 대한 관리를 하며, 실질적인 자료의 축출과 클라이언트로의 결과 전송은 Reading SubscribeThread 클래스가 수행한다.

그림 5. 서버 측 읽기 서비스 클래스 다이어그램
Fig. 5. Class Diagram of Reading Service on Server

3.3.4 클라이언트 측 읽기 서비스

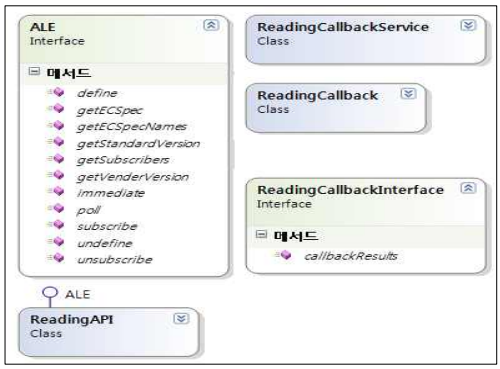

그림 6. 클라이언트 측 읽기 서비스 클래스 다이어그램
Fig. 6. Class Diagram of Reading Service on Client

그림6은 클라이언트 측 읽기 서비스(Reading Service)를 제공하기 위한 클래스 다이어그램이다. RFID 및 센서 장치로부터 자료를 읽어오기 위하여 ALE 인터페이스를 구현한 ReadingAPI 클래스는 웹 서비스를 통해서 서버로부터 자료를 가져와 클라이언트에 결과를 제공하며, DLL 형태로 제공함으로서 클라이언트 프로그램에서 연결해 사용할 수 있다. 그리고 ReadingCallback 과 ReadingCallbackService 클래스는 서버로부터 비동기 방식으로 HTTP POST 방식을 통해 결과를 수신하는 역할을 하며, ReadingCallbackInterface를 통해 클라이언트에서 작성한 프로그램에 수신한 결과를 전송해 준다.

3.3.5 센서 데이터 저장 방법

센서 측정값은 ItemLoc 테이블에 저장하며, 발생일시와 측정 장치 및 위치를 저장하고, 테이블간의 관계를 맞춰주기 위해 TagIDs 테이블의 ID가 0인 가상의 열(Row)과 연결해 준다. 그리고 시간에 따라 지속적으로 변화하는 측정값의 효율적인 저장을 위해 DevAntenna 테이블에 지정된 임계값 이상의 변화가 감지되었을 경우에만 변화된 값을 저장한다.

IV. 구현 및 적용

4.1 구현 환경

본 논문에서 제안한 시스템은 MS SQL Server 2008 데이터베이스를 사용하였으며, Visual Studio 2010을 이용해서 C# 언어로 개발하였다. ALE 읽기 서비스와 EPCIS 질의 인터페이스는 Windows 7 OS 에서 제공하는 IIS (Internet

Information Server)를 통해 웹 서비스 형태로 제공한다. 실험을 위해 클라이언트에서의 사용의 편의성을 위해 제공하는 DLL을 이용해서 실험 프로그램을 작성하였다. 그리고 ALE 미들웨어에서는 Alien사의 ALR-9800-KOR 리더기와 Kenetics 사의 USB 리더 그리고 온도, 습도, 조도 정보를 감지하기 위해 Crossbow사의 Zigbee 센서 장치를 사용하였다.

4.2 적용 시나리오

그림 7. 실험 시나리오
Fig. 7. Test Scenario

제안 시스템에서 제공하는 ALE 읽기 API와 EPCIS 저장소의 질의 인터페이스에 대한 동작을 확인하기 위해서, 그림7과 같이 2개의 창고를 관리하는 재고 관리에 제안 시스템을 적용한다. 각 창고에는 1개씩의 RFID 리더와 온도, 습도 및 조도를 측정할 수 있는 센서가 설치되어 있다. 각 창고에는 개별 물품과 파레트 위에 상자에 포장된 물품이 존재한다. 그리고 개별 물품 및 상자 그리고 파레트에는 그림7에 표시한대로 고유의 RFID 태그가 부착되어 있으며, 파레트와 상자와 물품 간에는 포함관계가 설정되어 있고, 그림7에 표시된 내용 이외에는 Action 에는 'Add', 작업 단계는 'Stocking', 조치 내역은 'Processing' 그리고 P/O(Purchase Order) 번호는 '9'로 설정되어 있다.

4.3 적용 및 결과

실험 프로그램을 이용해서 시나리오에 표현된 창고에 있는 물품에 대하여 RFID 및 센서 장치에 대한 ALE 읽기 API 실험과 EPCIS 저장소에 대한 센서 포함 질의 처리 내용을 확인한다.

4.3.1 ALE 읽기 API

RFID 장치 "RFID 1"과 센서 장치 "Humidity-1", "Temperature-1", "Illumination-1"로부터 15초 마다 자료를 전송해주는 subscribe 메서드를 실행한다.

그림8은 ALE 읽기 API 수행을 위해 서버 측에 보내질 ECSpec 의 내용을 XML 형식으로 보여주며 논리 리더, 수행 주기 그리고 보고서의 형태 등이 지정된 것을 확인할 수 있다. 그림9는 장치로부터 수집된 자료 중에서 ECSpec에 적합한 자료를 검색해 내기 위해 서버 측에서 자동으로 생성한 SQL 구문이다. 그림10은 서버 측에서 보내준 결과에 대한 XML 표현이며, 선택한 RFID 및 센서 장치로부터 EPC 자료와 온도, 습도, 조도 등의 센서 자료가 수집된 것을 확인할 수 있다.

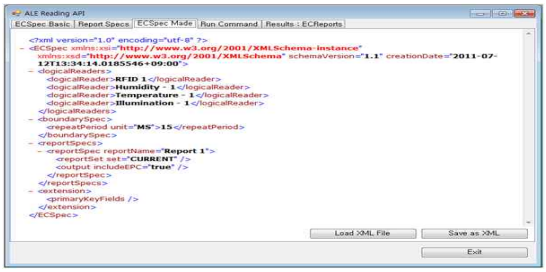

그림 8. 생성된 ECSpec
Fig. 8. Created ECSpec

```
SELECT * FROM V_AleTagsAddCur
WHERE ( LogicalReader = 'RFID 1' OR LogicalReader = 'Humidity - 1' OR LogicalReader = 'Temperature - 1' OR LogicalReader = 'Illumination - 1' )
```

그림 9. 생성된 SQL 구문
Fig. 9. Created SQL Statement

4.3.2 EPCIS 저장소 센서 포함 질의

EPCIS 저장소의 센서 관련 질의를 확인하기 위해 "창고의 조도가 50 이상이고 Action이 'Add'이거나 'Observe'이면서 회사코드가 1이고 물품코드가 4이면서 비즈니스 스텝이 'Storing'인 물품"을 검색해서 1시간에 한번 씩 결과를 반환하는 subscribe 질의를 수행한다. 그림11은 질의를 수행하기 위해 서버 측에 보낼 subscribe 질의 인수를 보여주며, 질의

```
<?xml version="1.0" encoding="utf-8"?>
<ECReports xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" schemaVersion="1.1"
creationDate="2011-07-12T15:48:24.2001953+09:00"
schemaURL="srhyun.doowon.ac.kr" date="2011-07-12T15:48:24.2001953+09:00"
specName="TagSensor " ALEID="HSR-ALE" totalMilliseconds="10">
  <reports>
    <report reportName="Report 1">
      <group>
        <groupList>
                                    <member>
<tag>urn:epc:id:sgtin:000001.0000003.1</tag>
                                    </member>
                                    <member>
<tag>urn:epc:id:sgtin:000001.0000003.2</tag>
                                    </member>
                                    <member>
<tag>urn:epc:id:sgtin:000001.0000004.1</tag>
                                    </member>
                                    <member>
<tag>urn:epc:id:sgtin:000001.0000004.2</tag>
                                    </member>
```

```
                                <member>
<tag>urn:epc:id:sgtin:000001.0000005.1</tag>
                                </member>
                                <member>
<tag>urn:epc:id:sgtin:000001.0000090.1</tag>
                                </member>
                                <member>
<tag>urn:epc:id:sgtin:000001.0000091.1</tag>
                                </member>
                                <member>
<tag>urn:epc:id:sgtin:000001.0000001.1</tag>
                                </member>
                                <member>
<tag>urn:epc:id:sgtin:000001.0000001.2</tag>
                                </member>
                                <member>
<tag>urn:epc:id:sgtin:000001.0000001.3</tag>
                                </member>
                                <member>
<tag>urn:epc:id:sgtin:000001.0000001.4</tag>
                                </member>
                                <member>
<tag>urn:epc:id:sgtin:000001.0000092.1</tag>
                                </member>
                                <member>
<tag>urn:epc:id:sgtin:000001.0000002.1</tag>
                                </member>
                                <member>
<tag>urn:epc:id:sgtin:000001.0000002.2</tag>
                                </member>
                                <member>
<tag>urn:epc:id:sgtin:000001.0000002.3</tag>
                                </member>
                                <member>
<tag>urn:epc:id:sgtin:000001.0000002.4</tag>
                                </member>
                                <member>
                                    <tag>urn:Sensor:Temperature:23.26</tag>
                                </member>
                                <member>
                                    <tag>urn:Sensor:Illumination:4.76</tag>
                                </member>
                                <member>
                                    <tag>urn:Sensor:Humidity:72.92</tag>
                                </member>
            </groupList>
        </group>
    </report>
  </reports>
</ECReports>
```

그림 10. 생성된 ECReports
Fig. 10. Created ECReports

조건이 XML로 표현된 것을 확인할 수 있다. 그림12는 서버 측에서 질의 인수를 분석해서 만든 SQL 구문으로서 FROM 절에 V_ObjItemSensor 뷰가 지정되었고, 모든 조건이 WHERE 절에 포함되어 있는 것을 확인할 수 있다. 그림13은 서버 측에서 보내준 질의 결과를 XML 형태로 보여주고 있으며, 조건에 맞는 '1.4.3' 물품 자료가 검색된 것을 확인할 수 있다.

```
<?xml version="1.0" encoding="utf-8"?>
<EPCISQueryDocumentType
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
schemaVersion="1.0"
creationDate="2011-07-12T19:00:36.7871093+09:00">
    <EPCISBody>
        <Subscribe xmlns="urn:epcglobal:epcis-query:xsd:1">
        <queryName xmlns="">Query Name 1</queryName>
        <params xmlns="">
            <param>
                <name>eventType</name>
                <value xsi:type="xsd:string">AggregationEvent
ObjectEvent QueryEvent TransactionEvent</value>
            </param>
            <param>
                <name>EQ_action</name>
                <value xsi:type="xsd:string">ADD
OBSERVE</value>
            </param>
            <param>
                <name>MATCH_epc</name>
                <value
xsi:type="xsd:string">urn:epc:id:sgtin:000001.0000004.*</value>
            </param>
            <param>
                <name>EQ_bizStep</name>
                <value xsi:type="xsd:string">Storing</value>
            </param>
            <param>
                <name>GE_Sensor.Illumination</name>
                <value xsi:type="xsd:float">50</value>
            </param>
        </params>
        <dest xmlns="">http://localhost:8888/Callback/</dest>
        <controls xmlns="">
            <schedule>
                <second />
                <minute />
                <hour>1</hour>
                <dayOfMonth />
                <month />
                <dayOfWeek />
            </schedule>
            <trigger />
            <reportIfEmpty>false</reportIfEmpty>
        </controls>
        <subscriptionID xmlns="">Subscription ID
1</subscriptionID>
        </Subscribe>
    </EPCISBody>
</EPCISQueryDocumentType>
```

그림 11. 질의 인수 XML
Fig. 11. XML Form of Query Parameter

```
SELECT * FROM V_ObjItemSensor
WHERE ( Action = 'ADD' OR Action = 'OBSERVE' ) AND ( Co =
'000001' AND Item = '0000004' ) AND ( BizStep = 'Storing' )
AND ( (DevSubName = 'Illumination' AND SensorData >= 50) )
```

그림 12. 생성된 SQL 구문
Fig. 12. Created SQL Statement

```
<?xml version="1.0" encoding="utf-8"?>
<QueryResults
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <queryName>Query Name 1</queryName>
    <subscriptionID>Subscription ID 1</subscriptionID>
    <resultsBody>
        <EventList>
            <ObjectEvent>
                <eventTime>2011-07-12T00:00:00</eventTime>
                <recordTime>2010-01-01T00:00:00</recordTime>

<eventTimeZoneOffset>+09:00</eventTimeZoneOffset>
                <epcList>

<epc>urn:epc:id:sgtin:000001.0000004.3</epc>
                </epcList>
                <action>ADD</action>

<bizStep>urn:epcglobal:epcis:bizstep:fmcg:Storing</bizStep>

<disposition>urn:epcglobal:epcis:disp:fmcg:Processing</disposition>
                <readPoint>
                    <id>urn:epc:id:sgln:1.8.2</id>
                </readPoint>
                <bizLocation>
                    <id>urn:epcglobal:fmcg:loc:1.7.2</id>
                </bizLocation>
            <Sensor
Name="Illumination">57.130000000000000000</Sensor>
            </ObjectEvent>
        </EventList>
        </resultsBody>
</QueryResults>
```

그림 13. 질의 결과 XML
Fig. 13. XML Form of Query Results

표 3. 성능 측정 값
Table 3. Performance Measurement Values

구분 \ 자료개수	10	50	100	500	1,000	1개 처리 평균
제안 시스템	0.047	0.269	0.806	2.744	6.567	0.0063
묶음 캡처서비스	0.080	0.420	0.931	4.393	8.703	0.0088
개별 캡처서비스	0.198	1.036	2.445	11.308	23.680	0.0233
캡처서비스 평균	0.139	0.728	1.688	7.8505	16.192	0.0160

V. 성능 평가

본 논문에서 제안한 통합 시스템은 본 논문의 특징이라고 할 수 있으며, 제안한 시스템과 유사한 방식으로 통합해서 제안한 사례는 찾아볼 수 없었다. 제안 시스템은 ALE 기반 미들웨어에서 태그 자료를 인식하자마자 객체의 위치에 대한 정보를 EPCIS 이벤트 정보로 변환해서 데이터베이스에 직접 저장하는 구조이고, 기존의 미들웨어와 EPCIS 저장소가 분리된 형태에서는 사용자가 업무 로직에 맞춰 추가로 작성한 캡처링 응용프로그램이 미들웨어를 통해 객체의 위치 정보를 확인 한 후, EPCIS 이벤트 정보를 만들어서, EPCIS에서 제공하는 캡처서비스를 통해 전달하는 구조이다.

이에 제안한 통합 시스템과 기존의 분리 형태에서의 사용 방법과 유사하게 비교 분석을 수행하기 위해, 아래와 같이 3가지 자료 전송 형태를 이용해서 자료의 발생부터 저장소에 자료가 저장될 때까지 걸리는 시간을 비교하도록 한다.

- 제안 시스템 : 제안 시스템의 자료 수집 형태를 이용해서 직접 데이터베이스에 저장한다.
- 개별 캡처 서비스 : 미들웨어와 EPCIS 저장소가 분리되어 있을 때 일반적으로 사용하는 방식으로서, 이벤트 자료를 캡처 서비스를 이용해서 저장소에 전달하며, 자료 1건에 대하여 하나의 ObjectEvent 를 발생시켜서 개별적으로 전송한다.
- 묶음 캡처 서비스 : 캡처 서비스를 이용하며, 다수의 자료를 하나의 ObjectEvent 안의 epcList 에 모두 담아서 한 번에 전송하지만 동일 장소, 동일 시간, 동일한 비즈니스 스텝일 경우에만 사용 가능하다.

성능 평가용 실험 프로그램은 태그 인식 자료를 요청하는 개수만큼 가상으로 발생 시켜서 제안 시스템에 위의 전송 형태에 따라 전송한 후 저장소에 저장하는 방식으로 개발하였으며, 자료의 발생에서부터 저장되기까지 걸리는 시간을 측정하였다. 실험에 사용된 PC는 Intel Pentium Dual Core 3.0 GHz의 CPU와 4G 바이트의 메인메모리를 사용하며, OS는 Windows 7을 이용한다. 성능 비교를 위해 각 전송 형태별로 자료 개수를 증가시키면서 실험하였고, 그림 14는 실험의 결과를 그래프로 보여준다.

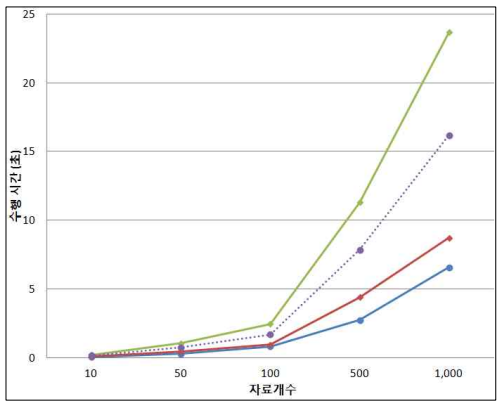

그림 14. 성능 비교 그래프
Fig. 14. Performance Comparison Graph

표 3의 측정값을 살펴보면 제안 시스템의 수행 속도가 캡처서비스를 이용하는 두 가지 방식보다 자료의 개수에 따라 적게는 2배에서 3배까지 속도가 빠른 것을 확인할 수 있었으며, 특히 자료가 많아질수록 시간의 격차가 많이 나는 것을 확인할 수 있다. 그리고 제안 시스템은 태그를 인식하자마자 바로 EPCIS 이벤트 정보를 저장할 수 있지만, 기존의 분리 시스템에서는 다른 미들웨어를 통해 태그를 인식하는데도 추가적인 시간이 필요할 수 있다. 또한, 실제 업무 상황에서는 개별 캡처서비스와 묶음 캡처서비스 형태가 별개로 발생하는 경우보다는 두 가지 종류가 중복해서 발생할 가능성이 훨씬 높다. 앞에서 실험한 성능 그래프에 점선으로 보이는 두 가지 캡처서비스의 평균과 제안 시스템의 성능을 비교해 보아도 2배 이상의 차이가 있음을 확인할 수 있으며, 이는 실제 업무에서도 차이가 날수 있음을 보여준다. 제안한 통합 시스템에서 하나의 태그를 인식하고 저장소에 저장하는데 걸리는 시간은 평균 0.0063 초이다. 이는 물품의 이동에 대한 검색이 실시간으로 가능하다는 것을 의미한다고 할 수 있다.

VI. 결론

본 논문에서는 일반적인 객체 인식과 환경 정보 변화에 의존하는 객체 검색을 실시간으로 가능하게 하기 위해, 장치로부터 정보를 지속적으로 읽어 데이터베이스에 저장하고, 수집된 자료를 기반으로 EPCglobal에서 표준으로 제안하는 ALE 기반 미들웨어와 EPCIS 저장소를 RFID와 센서 데이터를 함께 처리할 수 있도록 설계하고 구현하였다. 또한, 구현한 시스템을 두개의 창고를 관리하는 재고관리에 적용하여 ALE 읽기 API를 통해 RFID와 센서 장치로부터 자료를 읽어오는 것을 확인하고, EPCIS 저장소에서 센서 포함 질의를 수행하고 질의 결과를 확인하였다, 그리고 제안 시스템과 기존의 분할된 시스템과의 성능을 비교하였다.

본 논문에서 제안한 시스템은 ALE 버전 1.1.1 표준을 기반으로 읽기 API를 구현하였고, EPCIS 버전 1.0.1 표준을 기반으로 저장소를 구현하였으므로 표준에 맞게 개발한 미들웨어 및 저장소 관련 응용 프로그램에서 시스템의 변경 없이 사용할 수 있는 장점을 가진다. 또한, RFID 자료와 유사한 성격을 가지는 센서 데이터의 처리도 가능할 수 있도록 구현하였기 때문에 특정 장소의 센서 값의 변화에 따른 질의를 처리할 수 있는 장점이 있다.

제안 시스템은 ALE 미들웨어 기능만을 이용해서 RFID 리더나 센서 장치로부터 자료를 수집하거나 모니터링하는 시스템 개발에 응용 가능하고, EPCIS 저장소 기능만을 이용해서는 제조, 교통, 의료, 환경, 물류, 유통, 농축산업 등의 전 분야 시스템 구축에 이용 가능하다. 또한, 통합 시스템을 이용하면 냉장/냉동 창고나 축사, 비닐하우스 관리 시스템 등에서 실시간 환경 변화에 따른 객체 검색 및 제어 등에 응용할 수 있다.

참고문헌

[1] M. S. Yang, Y. C. Byun, "A method of sensor data stream processing based on RFID middleware," Journal of the Korean Institute of Maritime and Communication Sciences, Vol.12, No.2, pp.231-239, Feb. 2008.

[2] S. R. Hyun, S. J. Lee, "A Design and Implementation of EPCIS Repository for RFID and Sensor Data," Journal of The Korea Society of Computer and Information, Vol.15, No.12, pp.151-162, Dec. 2010.

[3] EPCglobal, http://www.epcglobalinc.org/.

[4] EPCglobal, "The Application Level Events (ALE) Specification, Version 1.0," September 2005.

[5] EPCglobal, "The Application Level Events (ALE) Specification, Version 1.1.1," March 2009.

[6] EPCglobal, "EPC Information Services (EPCIS) Version 1.0.1 Specification," September 2007.

[7] Y. M. Hong, Y. C. Byun, "A Design and Implementation of ALE-compliant RFID Middleware System," Journal of the Korean Institute of Maritime and Communication Sciences, Vol.11, No.4, pp.648-655, Apr. 2007.

[8] S. S. Kang, E. S. Chang, G. J. Park, "ALE v1.1 Middleware Implementation in RFID Systems," Journal of the Korean Institute of Information Technology, Vol.7, No.4, pp.124-133, Sep. 2009.

[9] Y. C. Kim, M. S. Park, "An Implementation of The RFID Middleware Based on Web-Service System Mutual Applications," Journal of the Korea Association of Information Systems, Vol.15, No.12, pp.151-162, Dec. 2010.

[10] J. W. Sung, D. Y. Kim, "EPC Sensor Network for RFID/USN Integrated Infrastructure," Information and Communications Magazine of the Korea Information and Communications Society, Vol.23, No.12, pp.37-46, Dec. 2006.

[11] J. H. Park, Y. M. Park, "The study of Ubiquitous healthcare service using RFID and Sensor network," Journal of the Korea Information and Communications Society, Vol.33, No.12, pp.467-472, Dec. 2008.

[12] K. S. Bok, Y. J. Cho, M. H. Yeo, J. S. Yoo, "Efficient Data Management Method for Massive RFID Data," Journal of the Korea Contents Association, Vol.9, No.6, pp.25-36, Jun. 2009.

[13] F. Wang and Peiya Liu, "Temporal Management of RFID Data," Proc. International Conference on VLDB, pp. 1128-1139, 2005.

[14] H. Gonzalez, J. Han, X. Li, and D. Klabjan, "Warehousing and Analyzing Massive RFID Data

Sets," Proc. International Conference on Data Engineering, pp. 83-83, 2006.

[15] D. Lin, H. G. Elmongui, E. Berino, and B. C. Ooi, "Data Management in RFID Applications," Proc. International Conference on Database and Expert Systems Applications, pp. 434-444, 2007.

[16] S. Lee, J. Kim, S. H. Shin, S. D. Nam, "Implemen ntation of Storage Manager to Maintain Efficiently Stream Data in Ubiquitous Sensor Networks," Journal of the Institute of Electronics Engineers of Korea, Vol.46, No.3, pp.24-33, May. 2009.

저 자 소 개

현 승 렬

1988 : 한양대학교 전자공학과 공학사.

1991 : 한양대학교 전자계산학과 공학석사.

2011 : 순천향대학교 컴퓨터학과 박사수료

현 재 : 두원공과대학 인터넷정보과 부교수

관심분야 : 데이터베이스, 객체지향프로그래밍, RFID, USN

Email : srhyun@doowon.ac.kr

이 상 정

1983 : 한양대학교 전자공학과 공학사.

1985 : 한양대학교 전자공학과 공학석사.

1988 : 한양대학교 전자공학과 공학박사

현 재 : 순천향대학교 컴퓨터학부 교수

관심분야 : 멀티코어 프로세서, 저전력/발열 시스템, RFID, 임베디드/네트워크 시스템 응용

Email : sjlee@sch.ac.kr