공간 DSMS 기반 RTLS의 설계 및 구현
(Design and Implementaion of RTLS based on a Spatial DSMS)

김 정 준* 김 판 규** 김 동 오*** 이 기 영**** 한 기 준*****
(Joung Joon Kim) (Pan Gyu Kim) (Dong Oh Kim) (Ki Young Lee) (Ki Joon Han)

요 약 최근 유비쿼터스 컴퓨팅 기술의 발전과 함께 센서 및 RFID에 관련한 정보 인식, 위치 측위와 같은 다양한 유비쿼터스 영역에 대한 관 심과 연구가 증대되고 있다. 특히, RFID 태그를 이용해 이동 객체의 위치 및 상태 정보를 제공하는 다양한 RTLS(Real-Time Locating Systems)가 개발됨에 따라 RTLS에서의 데이타 호환성과 상호 운용성을 위한 표준 명세가 필요하게 되었다. 이에, ISO/IEC에서는 RTLS의 데이타 호환성과 상호 운용성을 위해 RTLS 표준 명세를 제시하였다. 본 논문에서는 이동 객체의 데이타 스트림에 대한 효율적인 관리와 검색을 위해 공간 DSMS(Data Stream Management System)기반 RTLS를 설계 및 구현하였다. 공간 DSMS는 Stanford 대학교에서 개발한 STREAM(STanford stREam datA Manager)을 다양한 공간 연산이 가능하도록 확장한 시스템이다. 공간 DSMS 기반 RTLS는 서버와 클라이언트간의 상호 운용성을 위하여 SOAP(Simple Object Access Protocol) 메시지를 사용하며, 클라이언트의 SOAP 메시지를 공간 DSMS의 CQL(Continuous Query Language)로 변환한다. 마지막으로, 공간 DSMS 기반 RTLS를 사원 위치 관리 서비스에 적용해 봄으로써 시스템의 효용성을 입증하였다.

키워드 : 실시간 위치 추적 시스템, 데이타 스트림, 공간 데이타 스트림 관리 시스템

Abstract With the recent development of the ubiquitous computing technology, there are increasing interest and research in technologies such as sensors and RFID related to information recognition and location positioning in various ubiquitous fields. Especially, a standard specification was required for compatibility and interoperability in various RTLS(Real-Time Locating Systems) according to the development of RTLS to provide location and status information of moving objects using the RFID Tag. For these reasons, the ISO/IEC published the RTLS standard specification for compatibility and interoperability in RTLS. Therefore, in this paper, we designed and implemented RTLS based on the spatial DSMS(Data Stream Management Stream) for efficiently managing and searching the incoming data stream of moving objects. The spatial DSMS is an extended system of STREAM(STanford stREam datA Manager) developed by Standford University to make various spatial operations possible. RTLS based on the spatial DSMS uses the SOAP(Simple Object Access Protocol) message between client and server for interoperability and translates client's SOAP message into CQL(Continuous Query Language) of the spatial DSMS. Finally, we proved the efficiency of RTLS based on the spatial DSMS by applying it for the staff location management service.

Key words : RTLS, Data Stream, Spatial DSMS

1. 서론

최근 유비쿼터스 컴퓨팅 환경에서 공간 및 위치 정보를 제공하는 u-GIS 공간 정보 기술이 미래의 핵심 기술로 대두되고 있다[1,2]. 특히, RFID(Radio Frequency IDentification)나 GPS(Global Positioning System) 등과 같은 위치 측위 기술의 발달로 무선 통신 기술을 이용한 위치 추적 서비스가 증가함에 따라 이동 객체의 위치 정보를 실시간으로 서비스하기 위한 실시간 위치 추적 시스템인 RTLS(Real Time Locating System)의 연구 개발이 활발히 진행되고 있다[3]. RTLS는 RFID 태그를 부착한 이동 객체들의 위치 정보를 수집하여 어플

†본 연구는 국토해양부 첨단도시기술개발사업 - 지능형국토정보기술혁신 사업과제의 연구비지원(07국토정보C05)에 의해 수행되었습니다.

* 건국대학교 컴퓨터공학과 박사과정, jjkim9@db.konkuk.ac.kr
** 건국대학교 컴퓨터공학과 석사과정, pgkim@db.konkuk.ac.kr
*** 건국대학교 컴퓨터공학과 강의교수, dokim@db.konkuk.ac.kr
**** 을지대학교 의료산업부 교수, kylee@eulji.ac.kr(교신저자)
***** 건국대학교 컴퓨터정보공학과 교수, kjhan@db.konkuk.ac.kr

논문접수 : 2008.08.18 수정일 : 1차 2008.10.27 / 2차 2008.12.05 심사완료 : 2008.12.09

리케이션에 제공하는 실시간 위치 추적 시스템으로 최근 물류/유통 관리, 의료/보건, 자산 관리, 기업 보안, 공공 안전 등의 다양한 분야에서 널리 활용되고 있다.

특히, 실시간 위치 추적 시스템을 연구 개발할 때 중복 투자 방지 및 상호 운용성을 위해 ISO/IEC에서 RTLS (Real Time Locating System) 표준 명세를 제시하였다 [4,5,6]. RTLS 표준 명세는 응용 프로그램 인터페이스, 2.4 Ghz Air 인터페이스 프로토콜, 433 Mhz Air 인터페이스 프로토콜 세 부분으로 정의되어 있다. 이중 본 논문에서 다루고 있는 RTLS 응용 프로그램 인터페이스(API:Application Program Interface)는 RTLS와 클라이언트간의 통신 프로토콜과 각 서비스에서 필요한 스키마를 정의하고 있다.

RTLS에서는 효율적인 서비스 제공을 위해 RFID 태그로부터 끊임없이 입력되는 데이타 스트림에 대한 안정적인 처리가 필요하다. 그러나 기존의 RTLS는 주로 응용 서비스를 위한 기능에만 중점을 두고 있기 때문에 이러한 데이타 스트림으로 인해 시스템 부하가 크게 증가한다. 따라서, 데이타 스트림을 효율적이고 안정적으로 처리할 수 있는 데이타 스트림 처리 기능이 필요하다. 그리고 기존 RTLS에서는 다양한 위치 추적 서비스 제공을 위해 필요한 공간 연산 처리가 미흡하기 때문에 공간 데이타 처리 시 많은 연산 비용이 발생한다. 따라서, 실시간 위치 추적 서비스를 효율적으로 제공하기 위해서는 다양한 공간 연산자 지원이 필요하다.

최근 몇 년간 이러한 데이타 스트림 처리를 위하여 여러 대학과 연구 기관에서 DSMS(Data Stream Management System)에 대한 연구가 활발하게 진행되고 있다. 이러한 연구로는 인터넷에서 발생하는 이벤트에 대한 모니터링을 효율적으로 하기 위해 연산자 공유를 지원하는 NiagaraCQ[7], 연속 질의에 필요한 연산자들을 Box 개념으로 나눠서 질의를 구성하는 Aurora/ Borealis[8], 데이타 스트림 처리를 DBMS의 관점에서 접근하여 스트림과 릴레이션을 정의한 STREAM(STanford stREam datA Manager)[9]등이 있다. 그러나 이러한 연구들은 주로 비공간 속성에 대한 질의의 효율 향상이나 적응적 처리에 중점을 두며 공간 속성에 대한 처리를 고려하고 있지 않는 문제점이 있다.

본 논문에서는 실시간 위치 추적 서비스를 효율적으로 제공하기 위해서 Stanford 대학교에서 개발한 STREAM을 확장하고[10], 또한 이를 기반으로 하여 RFID 태그로부터 수집된 위치 정보를 효율적으로 처리 및 관리할 수 있는 '공간 DSMS 기반 RTLS'를 설계 및 구현하였다.

본 논문에서 확장 개발한 공간 DSMS는 여러 가지 형태의 공간 데이타를 처리하기 위해 OGC(Open Geospatial Consortium)에서 제시한 "Simple Feature Specification for SQL" 표준 명세[11]에 따라 공간 데이타 타입 및 공간 연산자를 제공하고, 다수의 질의가 동시에 처리될 때 여러 관계 연산자에 의해 공유되는 공간 객체를 효율적으로 관리하기 위한 공간 객체 관리 기능과 공간 DSMS에서 데이타 스트림으로 인한 입력 부하를 줄이기 위해 필터링 기능을 제공한다. RTLS를 위한 공간 DSMS는 기존의 DSMS와는 달리 공간 데이타 스트림이라는 새로운 형태의 데이타 모델을 처리하여 제공함으로써 RTLS의 효율성을 향상시킬 수 있다.

또한, 본 논문에서 연구 개발한 공간 DSMS 기반 RTLS는 ISO/IEC에서 제시한 RTLS 표준 명세에 따라 클라이언트에 대한 세션을 생성, 유지, 삭제한다. 또한 클라이언트로부터 전달된 SOAP(Simple Object Access Protocol)[12] 메시지를 분석하여 공간 DSMS에서 처리할 수 있는 CQL(Continuous Query Language)[13]로 변환하고, 질의 결과와 세션에 저장된 RFID 태그 정보를 SOAP 메시지로 작성하여 클라이언트에게 반환한다. 그리고 세션에서 RFID 태그 정보의 효율적인 관리를 위한 메모리 관리 기능을 제공한다.

본 논문의 구성은 다음과 같다. 제1장의 서론에 이어, 제2장에서는 관련 연구로 RTLS와 STREAM에 대하여 설명한다. 제3장에서는 공간 DSMS 기반 RTLS의 각 모듈에 대하여 기술하고, 제4장에서는 공간 DSMS 기반 RTLS의 실험 결과 및 구현 시나리오에 대하여 설명한다. 마지막으로, 제5장에서는 결론에 대하여 언급한다.

2. 관련 연구

본 장에서는 ISO/IEC에서 제시한 RTLS와 Stanford 대학에서 개발한 STREAM에 대하여 설명한다.

2.1 RTLS(Real-Time Locating Systems)

국제 표준화 단체인 ISO/IEC에서는 지정된 공간에서 자산 또는 사람 등에 RFID 태그를 부착하여 실시간으로 각 이동객체의 위치와 상태 데이타를 모니터링할 수 있는 시스템을 설계하고 상호 이기종 플랫폼 상의 데이타 운용성을 보장할 수 있도록 RTLS 표준 명세를 제시하였다. 현재 RTLS 표준 명세는 API, 2.4 Ghz Air 인터페이스 프로토콜, 433 Mhz Air 인터페이스 프로토콜 세 부분으로 정의되어 있다[4,5,6]. 이중 본 논문에서 다루고 있는 RTLS API에 정의된 통신 프로토콜은 그림 1과 같다.

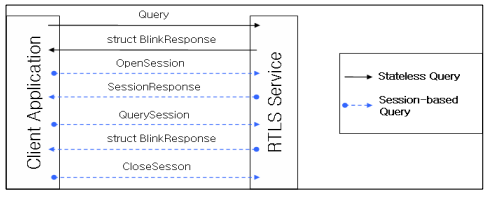

그림 1. RTLS API 통신 프로토콜그림

그림 1에서 보는 바와 같이 클라이언트는 원격지의 웹 서비스 시스템에서 제공하는 프로시저를 호출하여 그에 해당하는 응답을 받는 RPC(Remote Procedure Call) 형태의 동기적 SOAP 모델 방식을 사용한다. 실선은 RTLS가 클라이언트로부터 등록된 질의의 처리 결과를 유지하지 않는 일회성 질의 요청 응답 관계를 보여주고, 점선은 RTLS가 클라이언트로부터 요청된 질의를 세션이

유지되는 동안 계속해서 처리하여 세션에 저장 관리하도록 하는 연속적 질의 처리 응답 관계를 보여준다. Query는 클라이언트의 질의 조건에 맞는 이동 객체의 상태 및 위치 데이타를 RTLS로 부터 수집하는 비세션 기반 프로시저를 말한다. 그림 2는 RTLS API 표준 명세에 정의되어 있는 Query 스키마를 보여준다.

```
<?xml version="1.0" encoding="utf-8" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:rtls="http://www.incits.org/RTLS/" version="1.0">
  <xsd:element name="Query">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="QueryName" type="xsd:string"/>
        <xsd:element name="FilterBy" type="FilterType"/>
        <xsd:element name="Fields" type="FieldsType" />
        <xsd:element name="SortBy" type="SortType" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="FilterType">
    <xsd:sequence minOccurs="0">
      <xsd:any namespace="##any" maxOccurs="unbounded" processContents=" skip" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:simpleType name="relExp">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="(&lt;|&lt;=|&gt;|&gt;=|&lt;&gt;|=)(X,)+"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="FieldsType"><xsd:list itemType="xsd:string" /></xsd:simpleType>
  <xsd:complexType name="SortType">
    <xsd:all>
      <xsd:element name="Field" type="xsd:token" />
      <xsd:element name="Order">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="asc" /><xsd:enumeration value="desc" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
    </xsd:all>
  </xsd:complexType>
</xsd:schema>
```

그림 2. Query 스키마

그림 2에서 보여주는 Query 스키마에 정의된 Query-Name 엘리먼트는 RTLS에 질의 이름을 지정하고 FilerBy 엘리먼트는 검색 조건을 지정한다. 사용자는 FilterBy 엘리먼트에서 정의된 연산식을 사용하여 검색할 TagBlink의 조건을 지정할 수 있다. TagBlink란 하나의 실제 RFID 태그 데이타와 일대일 대응되는 논리적 태그 데이타 정보를 말한다. 표 1은 TagBlink의 데이타 구조에 대하여 보여준다.

표 1. TagBlink의 데이타 구조

필드	설 명	타입	선택사항
TagID	태그 식별자 정보 필드	String	필수
CoordRef	해당 태그의 좌표 참조를 위한 식별자 정보 필드	String	선택
Location	위치항목을 나타내는 엘리먼트를 포함하는 부모 XML 태그 정보 필드 NoLocate, X. Y. Z. ZoneID, Bearing, and Distance	Float 또는 String	필수
NoLocatee	TagBlink 데이터를 받았지만, 위치를 계산하지 못하는 경우 사용되는 필드	Boolean	선택
X	X축의 위치 정보 필드	Float	선택
Y	Y축의 위치 정보 필드	Float	선택
Z	Z축의 위치 정보 필드	Float	선택
ZoneID	Zone 식별자 정보 필드	String	선택
Bearing	리더의 각도 정보 필드	Float	선택
Distance	리더의 각도 정보 필드	Float	선택
RTLSBlink Time	RTLS blink의 생성 시간 필드	TimeStamp	필수
Locate Time	RTLS 송신기가 마지막으로 감지된 시간 필드	TimeStamp	선택
TgModel	RTLS 송신기의 모델 정보 필드	String	선택
Resource Type	송신기가 부착된 자산의 타입 정보를 위한 필드	String	선택
ReaderID	TagBlink의 위치를 결정한 ReaderID 정보를 위한 필드	String	선택
States	RTLS 송신기의 다양한 상태를 포함하는 부모 XML 태그 정보 필드	String	선택
General	RTLS 송신기로부터의 모든 telemetry 데이터를 포함하는 필드	String	선택
Buttons	1011과 같은 이진 문자열 정보 필드	Binary string	선택
ExciterID	RTLS 송신기를 동작하게 하는 장치의 식별자 정보 필드	String	선택
Motion	태그의 움직임을 나타내는 필드	Boolean	선택
BatteryLow	배터리 상태를 나타내는 필드	Boolean	선택
Blinking	송신기 동작 여부를 나타내는 필드	Boolean	선택
Registered	RTLS 가 송신기를 예비 등록하기 위한 필드	Boolean	선택
VendorSection	벤더들이 추가하는 엘리먼트 부분을 위한 필드	String	선택

Fields 엘리먼트는 TagBlink 필드의 이름을 지정하여 RTLS로 입력되는 데이타 스트림에서 필요한 필드만을 검색하여 읽어 올 수 있도록 하고, SortBy 엘리먼트는 검색된 TagBlink의 정렬 순서를 지정하며 Fields 엘리먼트에서 선택한 필드를 기준으로 정렬 순서를 결정한다. Query 프로시저의 응답은 RTLS API 표준 명세에 정의된 BlinkResponse 구조를 갖는 QueryResponse 스키마를 따라 작성된다.OpenSession은 RTLS에 새로운 세션을 생성하고, 검색한 이동 객체의 상태 및 위치 데이타를 세션에 저장하며, 생성한 세션을 클라이언트로 전달하는 세션 기반 프로시저를 말한다. 그림 3은 RTLS API 표준 명세에 정의되어 있는 OpenSession 스키마 구조를 보여준다.

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:rtls="http://www.autoid.org/iso24730-1/RTLS-schema" version="1.0">
  <xsd:include schemaLocation="Query.xsd"/>
  <xsd:element name="OpenSession">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="QueryName" type="xsd:string"/>
        <xsd:element name="FilterBy" type="FilterType"/>
        <xsd:element name="Fields" type="FieldsType"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

그림 3. OpenSession 스키마

그림 3에서 보여주는 OpenSession 스키마에 정의된 QueryName, FilterBy, Fields 엘리먼트는 Query 스키마에서 설명한 내용과 동일하다. OpenSession 프로시저의 응답은 RTLS API 표준 명세에 정의된 Session-Response 스키마를 따라 작성된다.

QuerySession은 클라이언트로부터 전송받은 세션 식별자와 일치하는 세션을 찾아 세션에 저장된 데이타를 검색하는 세션 기반 프로시저를 말한다. 그림 4는 RTLS API 표준 명세에 정의된 QuerySession 스키마 구조를 보여준다.

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" version="1.0">
  <xsd:element name="QuerySession">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="SessionID" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

그림 4. QuerySession 스키마

그림 4에서 보여주는 QuerySession 스키마에 정의된 SessionID 엘리먼트는 OpenSession을 통해 부여받은 세션 식별자 번호를 말한다. QuerySession 프로시저의 응답은 RTLS API 표준 명세에 정의된 BlinkResponse 구조를 갖는 QueryResponse 스키마를 따라 작성된다. 마

지막으로 CloseSession은 생성된 세션이 더 이상 필요하지 않을 경우에 해당 세션의 모든 리소스를 해제하는 세션 기반 프로시저를 말한다. 그림 5는 RTLS API 표준 명세에 정의된 CloseSession 스키마 구조를 보여준다.

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" version="1.0">
  <xsd:element name="CloseSession">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="SessionID" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

그림 5. CloseSession 스키마

그림 5에서 보여주는 CloseSession 스키마에 정의된 SessionID 엘리먼트는 종료하고자 하는 세션 식별자 번호를 말한다. CloseSession 프로시저의 응답은 RTLS API 표준 명세에 정의된 SessionResponse 스키마를 따라 작성된다.

2.2 STREAM(STanford stREam datA Manager)

STREAM은 데이타 스트림을 모니터링하고 필터링하기 위하여 Stanford 대학에서 개발한 DSMS이며 현재 서버 0.6 버전과 클라이언트 0.3 버전이 소스와 함께 BSD License 형태로 공개되어 있다[9]. 서버는 Linux 기반에서 C++로 구현되어 있으며 라이브러리로 사용이 가능하고, 데이타 스트림 처리를 위한 연속 질의 언어 CQL을 지원한다. 클라이언트는 운영체제에 독립적인 Java로 구현되어 있으며 데이타 입력 및 질의 입력, 실행 계획 뷰어 등을 제공한다. 그림 6은 STREAM의 동작 방식을 보여주고 있다.

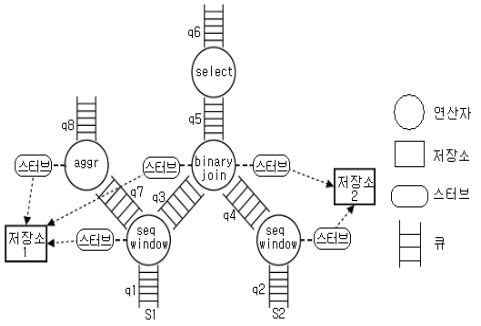

그림 6. STREAM의 동작 방식

사용자가 등록한 연속 질의는 Select, Project, Join 등의 연산자가 트리 형태로 모여 있는 실행 계획으로 변환되어 저장 및 처리된다. 각 연산자는 필요에 따라 저장소를 생성하게 되며 이 저장소에는 질의 처리의 중간 결과가 저장되어 여러 연산자에 의해 공유된다. 각 연산자는 스터브(Stub)라는 상태 저장 모듈을 갖고 있어서 저장소에 대한 참조 값을 보관하며, 연산자들은 큐로 연결되어 있어서 각 연산자는 입력 큐의 데이타들을 처리하여 출력 큐에 저장한다. 이 외에도, 하나의 질의 결과가 다른 질의들에 의해 공유되는 기능이 있어서 질의 처리 속도 및 공간 효율성을 향상시킨다.

STREAM은 CQL이라는 자체적으로 정의한 연속 질의 언어를 사용한다. CQL은 데이타 스트림으로부터 입력된 데이타를 릴레이션으로 바꾼 뒤 실행 계획에 따라 데이타를 처리하고, 결과를 스트림이나 릴레이션으로 출력한다. 여기서 스트림이란 갱신 및 삭제는 발생하지 않고 지속적인 입력만 발생하는 모델이며, 릴레이션이란 입력과 삭제가 발생하며 크기가 가변적이긴 하지만 논리적으로 유한한 모델이다.

데이타 스트림은 무선 센서 네트워크나 웹 서버 등으로부터 무한하게 입력되기 때문에 모든 입력 데이타에 대하여 질의를 처리한 후에 결과 데이타를 돌려주려고 할 경우 실행 시간이 무한대가 될 수밖에 없다. 이 때문에 CQL에서는 슬라이딩 윈도우(Sliding Window) 기법을 사용하여 무한한 스트림을 유한한 릴레이션으로 변환한다. 이러한 연산자로서 튜플 개수를 기준으로한 Row-Window 연산자와 시간 범위를 기준으로한 TimeWindow 등이 있다. 크가가 n인 RowWindow를 사용할 경우 릴레이션에 최고 n개까지 데이타만 저장하며 이후 데이타가 계속 입력되면 크기 n을 유지하기 위해 가장 오래된 데이타를 릴레이션에서 삭제한다. 시간 범위가 s초인 TimeWidnow를 사용할 경우 현재 타임스탬프를 t라고 할 때 t-s부터 t까지 입력된 데이타를 릴레이션에 저장한다. s초 동안 입력된 모든 데이타가 저장되어야 하므로 릴레이션에 저장될 튜플의 개수는 가변적이다. 이렇게 스트림을 릴레이션으로 변환함으로써 Join이나 집계 함수 등은 모든 스트림을 처리할 필요 없이 현재 릴레이션에 있는 데이타에 대해서만 처리한 뒤 결과를 반환한다.본 논문에서는 이러한 STREAM을 확장하여 공간 DSMS를 개발하기 위해 OGC에서 제시한 "Simple Feature Specification for SQL" 표준 명세[11]에 따라 공간 데이타 타입 및 연산자를 추가하였으며, 공간 연산자의 구현을 위해 GEOS(Geometry Engine Open Source)[14]를 사용하였다.

3. 시스템 설계

본 장에서는 공간 DSMS 기반 RTLS의 전체 구조에 대하여 기술하고, 공간 DSMS, RTLS, 그리고 클라이언트의 각 모듈에 대해 자세히 설명한다.

3.1 시스템 전체 구조

공간 DSMS 기반 RTLS는 클라이언트, RTLS, 공간 DSMS로 구성되어 있다. 그림 7은 본 논문에서 개발한 공간 DSMS 기반 RTLS의 시스템 전체 구조를 보여준다.

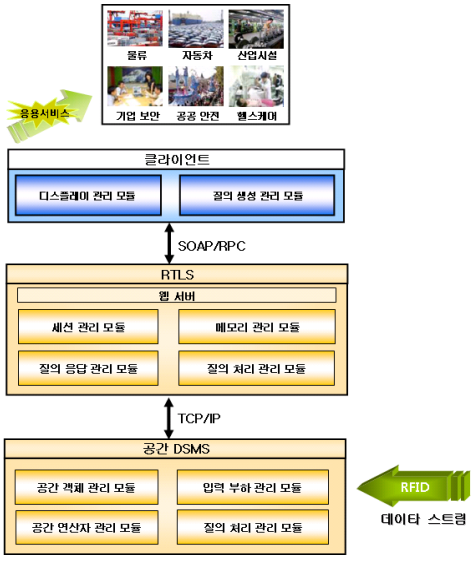

그림 7. 시스템 전체 구조

공간 DSMS는 연속질의를 분석 및 처리하는 질의 처리 관리 모듈, 공간 객체를 효율적으로 관리하기 위한 공간 객체 관리 모듈, 연속질의에 포함된 공간 연산자를 처리하는 공간 연산자 관리 모듈, 공간 DSMS에서 입력 데이타 스트림 양을 줄이기 위한 필터링 기능을 제공하는 입력 부하 관리 모듈로 구성된다. RTLS는 클라이언트의 세션을 관리하는 세션 관리 모듈, 세션에서 RFID 태그 정보 저장을 위한 메모리 관리 모듈, RFID 태그 정보를 SOAP 메시지로 작성하는 질의 응답 관리 모듈, SOAP 메시지를 분석하여 공간 DSMS에서 처리할 수 있는 CQL로 변환하는 질의 처리 관리 모듈로 구성된다. 그리고 클라이언트는 질의 처리 결과를 화면에 보여주는 디스플레이 관리 모듈, 사용자의 검색 조건을 분석하여 SOAP 메시지를 생성하는 질의 생성 관리 모듈로 구성된다.

3.2 공간 DSMS(Data Stream Management System)

공간 DSMS는 질의 처리 관리 모듈, 입력 부하 관리 모듈, 공간 연산자 관리 모듈, 공간 객체 관리 모듈로 구성된다.

3.2.1 질의 처리 관리 모듈

질의 처리 관리 모듈은 RTLS로부터 전달받은 연속질의를 공간 DSMS에 등록하고 분석 및 처리할 수 있는 기능을 제공한다. 또한, 대용량의 공간 데이타 스트림을 실시간으로 처리하기 위하여 SQL과 유사한 질의 언어인 CQL이라는 연속질의 언어를 사용한다. CQL은 공간 데이타 스트림과 윈도우를 타임스탬프로 정렬된 테이블로 간주하고 질의에 대한 결과를 전달하기 위한 변환 연산자를 제공한다. 그림 8은 공간 DSMS에서 제공하는 CQL 연산자를 보여준다.

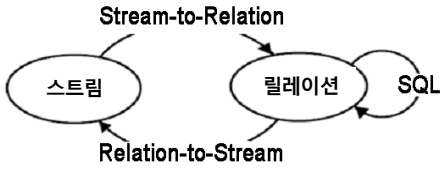

그림 8. CQL의 연산자

CQL은 스트림을 릴레이션으로 바꿔주는 Stream-to-Relation 연산자, 릴레이션을 다시 스트림으로 바꿔주는 Relation-to-Stream 연산자, 관계 대수의 연산자로 구성된다.

3.2.2 입력 부하 관리 모듈

입력 부하 관리 모듈은 공간 DSMS에서 처리하지 못할 과도한 입력이 들어왔을 때 정확도 손실을 최소화 하면서 입력 데이타 스트림 양을 줄여 과부하 문제를 해결하기 위한 필터링 기능을 제공한다. 그림 9는 공간 DSMS에서 공간 데이타가 필터링되는 과정을 보여준다.

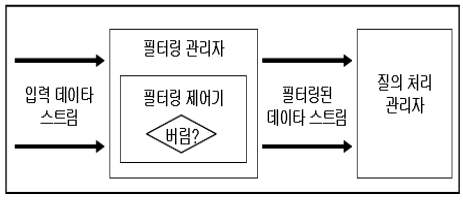

그림 9. 공간 데이타의 필터링 과정

공간 DSMS에서는 필터링을 위해 좌표 거리 차이, 특정 이동 객체의 아이디 등이 필터링 조건으로 지원된다.

3.2.3 공간 연산자 관리 모듈

공간 데이타를 효율적으로 처리하기 위해서는 복잡한 연산 과정이 필요하나 기존 STREAM은 다차원 형태인 공간 연산자를 지원하지 않는다. 따라서, 본 논문에서는

표 2. 공간 DSMS가 지원하는 공간 연산자 목록

연산자 종류	이 름	설 명
공간 관계 연산자	Equals	두 공간 객체가 동일하면 참을 반환함
	Disjoint	두 공간 객체가 교집합이 없으면 참을 반환함
	Touches	두 공간 객체의 경계선만 닿았으면 참을 반환함
	Within	첫 번째 두 공간 객체가 두 번째 공간 객체를 포함하면 참을 반환함
	Overlaps	두 공간 객체가 같은 차원이면서 겹쳐 있으면 참을 반환함
	Crosses	두 공간 객체가 교차하면서 교차된 영역의 차원이 같거나 줄어들면 참을 반환함
	Intersects	두 공간 객체가 교차하면 참을 반환함
	Contains	두 번째 공간 객체가 첫 번째 공간 객체를 포함하면 참을 반환함
	Relate	두 공간 객체의 관계를 DE-91M 문자열로 반환함
공간 분석 연산자	Distance	두 공간 객체간의 최단거리를 계산함
	Intersection	두 공간 객체가 겹치는 부분을 공간 객체로 반환함
	Difference	첫 번째 공간 객체에서 두 번째 공간 객체를 뺀 부분을 반환함
	Union	두 공간 객체을 합친 공간 객체를 반환함
	SymDifference	두 공간 객체를 합친 것에서 겹치는 부분을 뺀 공간 객체를 반환함
	Buffer	주어진 공간 객체를 주어진 크기만큼 증가시킨 공간 객체를 반환함
	ConvexHull	주어진 공간 객체를 포함하면서 가장 작은 볼록 폴리곤을 반환함

GEOS에서 제공하는 알고리즘을 사용하여 공간 연산자를 처리하는 공간 연산자 관리 모듈을 구현하였다. 표 2는 OGC에서 정의한 것 중 공간 DSMS가 지원하는 16개의 공간 연산자를 보여준다.

공간 관계 연산자는 두 공간 객체의 공간상의 관계를 파악하기 위한 연산자이며, 공간 분석 연산자는 하나 혹은 두 공간 객체로부터 새로운 공간 객체 혹은 의미있는 값을 얻기 위한 연산자이다.

3.2.4 공간 객체 관리 모듈

기존 STREAM은 메모리에 대한 포인터를 지원하지 않기 때문에 저장소가 더 이상 사용하지 않는 튜플을 메모리에서 해제할 때 참조 값이 적혀있는 메모리 영역만을 삭제해 주었다. 그러나 공간 객체는 Heap 영역에 생성되고 저장소에는 포인터 값만 저장되기 때문에 저장소에 저장되어 있는 포인터 정보가 메모리에서 삭제된다고 하더라도 Heap 영역에 있는 공간 객체가 메모리에서 삭제되지 않는다. 그리고 Heap 영역에 있는 공간 객체를 저장소가 직접 삭제하는 방식은 공간 객체가 여러 저장소에 동시에 존재할 경우 같은 작업을 중복 처리해야 하기 때문에 비효율적이다.

따라서 이러한 문제를 해결하기 위해 여러 저장소에서 참조하는 공간 객체를 효율적으로 관리하는 공간 객체 관리 모듈을 개발하였다. 공간 객체 관리 모듈은 여러 저장소에서 참조하는 공간 객체의 공간 객체 아이디와 참조 카운터를 관리함으로써 공간 객체가 사용하는 메모리 공간의 낭비를 최소화하고 중복되는 메모리 할당 및 해제 작업을 줄인다. 그림 10은 공간 객체가 여러 저장소에서 공유되는 예를 보여준다.

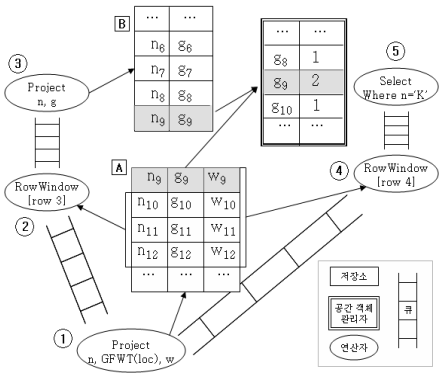

그림 10. 공간 객체 공유 예

그림 10을 통하여 공간 객체 관리 모듈의 기본적인 동작 방식을 살펴보면 다음과 같다. 연산자 1이 GFWT 함수를 통해 공간 객체를 생성한 뒤 이것을 저장소 A에 저장을 해두면 연산자 2와 4가 공유하게 된다. 이때, 연산자 2는 저장소 A에 있는 9번째 튜플을 처리하고 슬라이딩 윈도우를 10부터 12번 튜플로 옮긴 뒤 연산자 3이 이를 저장소 B에 저장하지만, 연산자 4가 9번째 튜플을 아직 처리하지 않아서 저장소 A에는 9번째 튜플이 남아있게 된다. 이러한 경우 저장소 A와 저장소 B 모두에 공간 객체 g9가 존재하게 되며 공간 객체 관리 모듈은 g9에 대한 참조 카운터를 2로 늘린다. 연산자 4가 9번째 튜플을 처리하면 저장소 A에서 9번째 튜플이 삭제되는데, 이때 공간 객체 관리 모듈은 Heap 영역에 있는 공간 객체를 바로 삭제하지 않고 참조 카운터만 1로 줄인 후, 저장소 B에서도 9번째 튜플이 삭제되면 참조 카운터가 0이 되면서 g9를 삭제한다.

3.3 RTLS(Real-Time Locating Systems)

RTLS는 질의 처리 관리 모듈, 질의 응답 관리 모듈, 세션 관리 모듈, 메모리 관리 모듈로 구성된다.

3.3.1 질의 처리 관리 모듈

질의 처리 관리 모듈은 SOAP 메시지의 검색 조건에 맞는 TagBlink를 검색하여 수집하고, 그 결과를 질의 응답 관리 모듈로 전달하는 작업을 담당한다. 또한, 질의 처리 관리 모듈은 SOAP 메시지를 CQL로 변환하는 매핑 규칙을 적용해서 생성한 CQL 문을 공간 DSMS로 전달한다. 그림 11은 SOAP 메시지의 CQL 매핑 규칙을 보여준다.

SELECT **Fields** FROM RTLS	① Fields 엘리먼트는 ","로 구분되어 CQL의 SELECT 절로 매핑된다.
[WHERE **FilterBy**]	② FilterBy 엘리먼트는 CQL의 WHERE 절로 매핑된다.
[ORDER BY **SortBy**]	③ SortBy 엘리먼트는 CQL의 Order By 절로 매핑된다.

그림 11. SOAP 메시지의 CQL 매핑 규칙

그림 11의 CQL 구문에서 Fields 절 부분은 생략될 수 없는 필수 부분이며, 대괄호로 쌓여있는 WHERE 절과 ORDER BY 절은 클라이언트가 SOAP 메시지를 RTLS로 전송할 때 FilterBy 또는 SortBy 엘리먼트를 생략하여 전송하였을 경우 생략될 수 있다. 그림 12는 연산자간의 매핑 규칙을 보여준다.

RTLS	CQL
<, >	<, >
<=, >=	<=, >=
<>, =	!=, =

그림 12. 연산자 매핑 규칙

그림 13은 질의 처리 관리 모듈로 전달된 Query 프로시저를 호출하는 SOAP 메시지의 예를 보여준다.

그림 13의 SOAP 메시지에서 보는 바와 같이 FilterBy 엘리먼트는 매핑 규칙에 따라 CQL의 WHERE 절로 매핑되고 Fields 엘리먼트는 CQL의 SELECT 절로 매핑된다. 그리고 SortBy 엘리먼트는 CQL의 Order By 절로 매핑된다. 그림 14는 그림 13의 SOAP 메시지를 매핑 규칙에 따라 변환한 CQL 문을 보여준다.

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Body>
  <Query xmlns="http://www.autoid.org/iso24730-1/RTLS-schema">
   <QueryName>RTLS_Blinks</QueryName>
    <FilterBy>
     <TagID><![CDATA[<>1234]]></TagID>
     <Location>
      <X><![CDATA[>567]]></X><X><![CDATA[<=1000]]></X>
      <Y><![CDATA[<1000]]></Y><Y><![CDATA[>300]]></Y>
     </Location>
     <BatteryLow>=true</BatteryLow>
    </FilterBy>
    <Fields>TagID Location BatteryLow</Fields>
    <SortBy><Order>asc</Order><Field>TagID</Field></SortBy>
   </Query>
  </env:Body>
</env:Envelope>
```

그림 13. SOAP 메시지 예

그림 14에서 보는 바와 같이 변환된 SOAP 메시지의 Fields 엘리먼트는 CQL 문의 SELECT TagID, Location, BatteryLow로 변환되었으며, FilterBy 엘리먼트는 CQL 문의 WHERE TagID != 1234 and X > 567 and X <= 1000 and Y < 1000 and Y > 300 and BatteryLow =true로 변환되었다. 그리고 SOAP 메시지의 SortBy 엘리먼트는 CQL 문의 ORDER BY TagID asc로 변환되었다.

```
SELECT TagID, Location, BatteryLow FROM RTLS
WHERE TagID != 1234 and X > 567 and X <= 1000 and Y < 1000
and Y > 300 and BatteryLow =true ORDER BY TagID asc
```

그림 14. 변환된 CQL 예

3.3.2 질의 응답 관리 모듈

질의 응답 관리 모듈은 RTLS의 질의 처리 관리 모듈과 세션 관리 모듈로부터 데이타 값을 전달받아 클라이언트로 SOAP 메시지를 작성하여 응답을 하는 작업을 담당한다. 질의 응답 관리 모듈은 RTLS가 지원하는 Query, OpenSession, QuerySession, CloseSession 프로시저를 호출하는 클라이언트의 SOAP 메시지 유형에 따라 작업 처리 담당 방식이 달라진다.

Query 프로시저를 호출하는 경우에는 질의 처리 관리 모듈에서 전달 받은 TagBlink 데이타를 SOAP 메시지로 작성하여 클라이언트로 응답하고, OpenSession 프로시저를 호출하는 경우에는 세션 관리 모듈로부터 전달받은 세션 식별자 번호를 SOAP 메시지로 작성하여 클라이언트로 응답한다. 또한, QuerySession 프로시저를 호출하는 경우에는 세션 관리 모듈에서 전달 받은 TagBlink 데이타를 SOAP 메시지로 작성하여 클라이언트로 응답하고, CloseSession 프로시저를 호출하는 경우에는 세션이 삭제되었다는 SOAP 메시지를 작성하여 클라이언트로 응답한다.

3.3.3 세션 관리 모듈

세션 관리 모듈은 클라이언트의 연결 정보 및 질의 관련 정보를 저장하기 위해 세션을 관리하는 작업을 담당한다. 세션에는 클라이언트의 검색 조건과 질의 응답 관리 모듈로부터 입력된 TagBlink 데이타가 저장되고, 세션 관리 모듈은 세션에 필요한 메모리 영역을 메모리 관리 모듈에게 요청한다.

또한, 세션 관리 모듈은 메모리 관리 모듈에서 세션의 메모리 영역을 효율적으로 관리할 수 있도록 메모리 관리 모듈의 벡터 사용 시간 변수에 세션 유효 시간을 저장한다. 세션 관리 모듈은 세션의 유효 시간이 초과될 때까지 생성된 세션을 클라이언트가 사용하지 않을 경우 해당 세션에 할당된 메모리 영역을 메모리 관리 모듈이 해제하도록 하고, 이와 반대로 세션의 유효 시간이 초과되기 전에 생성된 세션을 클라이언트가 사용하였을 경우 세션 관리 모듈은 메모리 관리 모듈의 벡터 사용 시간 변수에 새로운 세션 유효 시간을 다시 할당한다.

3.3.4 메모리 관리 모듈

메모리 관리 모듈은 세션 관리 모듈에서 세션에 클라이언트의 검색 조건과 TagBlink 데이타를 저장하기 위해 필요한 메모리를 관리하는 작업을 담당한다. 메모리 관리 모듈에서 메모리 관리를 위해 사용하는 버퍼 풀은 벡터 배열과 벡터 배열의 벡터 사용 유무 및 벡터 사용 시간을 저장하기 위한 구조체로 구성되어 있다. 메모리 관리 모듈은 세션 관리 모듈에서 버퍼 할당 요청을 할 경우에 사용 중이 아닌 벡터 객체를 검색하여 세션 관리 모듈로 전달한다. 이 때 세션 관리 모듈은 세션의 유효 시간을 벡터 사용 시간 변수에 할당한다.

메모리 관리 모듈은 세션 관리 모듈이 할당받았던 버퍼를 사용한 후 반환하면 반환된 벡터 객체의 해당 벡터 사용 유무 변수를 사용하지 않음으로 할당하고 버퍼를 버퍼 풀로 회수한다. 또한, 세션 관리 모듈에서 생성한 세션의 유효 시간이 초과되면 세션이 더 이상 사용되지 않는 것으로 간주하고 세션에 할당했던 버퍼를 강제로 회수하기 위해 벡터 사용 유무 변수를 사용하지 않음으로 할당하고 버퍼를 버퍼 풀로 회수한다.

3.4 클라이언트

클라이언트는 질의 생성 모듈과 디스플레이 모듈로 구성된다.

3.4.1 질의 생성 관리 모듈

질의 생성 관리 모듈은 클라이언트의 웹 브라우저에서 검색하고자 하는 객체의 검색 조건(RFID TagID, 배터리 상태, 이동상태 등)을 SOAP 메시지로 생성하는 모듈을 말한다. 클라이언트에서 지원하는 Query, OpenSession, QuerySession, CloseSession 서비스의 질의 생성 모듈은 각각의 스키마 작성 규칙에 맞게 SOAP 메시지를 생성한다.

3.4.2 디스플레이 관리 모듈

디스플레이 관리 모듈은 RTLS로부터 전달받은 SOAP 메시지를 웹 폼에서 사용자가 SOAP 코드를 볼 수 있게 한다. 또한, 디스플레이 관리 모듈은 SOAP 메시지를 파싱하여 검색된 이동 객체의 위치를 클라이언트 사용자

가 볼 수 있도록 Java Applet을 통해 위치를 점으로 표시한다. 이때 객체의 위치를 표시한 점은 사용자가 각 객체를 구별할 수 있도록 각각 다른 색으로 표시된다.

4. 실험 결과 및 시나리오

본 장에서는 본 시스템의 실험 결과를 살펴본 후 본 논문에서 개발한 공간 DSMS 기반 RTLS의 효용성을 검증하기 위하여 정의한 구현 시나리오에 대해 살펴본다. 마지막으로 공간 DSMS 기반 RTLS를 구성하는 각각의 서비스에 대해서 설명한다.

4.1 실험 결과

본 시스템을 테스트하기 위한 데이타 생성기는 공간 데이타를 생성할 수 있어야 하기 때문에 이에 적합한 이동 센서를 위한 데이타 생성기[15]를 사용하였다. 이동 센서를 위한 데이타 생성기는 센서의 특징을 고려하면서 공간 정보를 생성하기 위해 개발한 데이타 생성기로써 몇 가지 파라미터를 입력해서 일반적으로 사용되는 정규분포나 비정규분포 혹은 균등분포를 따르는 데이타를 생성한다.

그림 15는 본 데이타 생성기를 통하여 이동체에 대한 센서 데이타를 생성하기 위한 파라미터를 보여주는 화면이다.

```
DataSet.name              = stream_R.txt
DataSet.seed              = 14251312
DataSet.firstID           = 0
DataSet.nS                = 1
DataSet.loc.x.(min, max)                  = 0, 1000
DataSet.loc.y.(min, max)                  = 0, 1000
DataSet.nSS               = 50
DataSet.nG                = 1
DataSet.G[0].nNodes             = 50
DataSet.G[0].v_loc.v_x.init     = r, 0, 1000
DataSet.G[0].v_loc.v_x.d        = g_100_0.2, 0, 200
DataSet.G[0].v_loc.v_x.(min,max,Adj)= 0, 1000, m
DataSet.G[0].v_loc.v_x.e        = 0, 1000
DataSet.G[0].v_loc.v_y.init     = r, 0, 1000
DataSet.G[0].v_loc.v_y.d        = g_100_0.2, 0, 200
DataSet.G[0].v_loc.v_y.(min,max,Adj)= 0, 1000, m
DataSet.G[0].v_loc.v_y.e        = 0, 1000
```

그림 15. 이동체 센서 데이타 생성 파라미터

그림 15에서 보여지는 파라미터는 stream.txt라는 이름의 화일에 데이타를 저장하고, 랜덤함수의 시드(seed)값으로 14251312를 사용하며, 센서 id는 0번부터 시작된다. 스냅샷 수는 50회이며, 50개의 센서(nNodes)가 있는 하나의 그룹(nG) 데이타를 생성한다. x좌표는 0~1000 사이에서 랜덤하게 초기 위치가 정해져서 중심을 100, 편차를 0.2, 최소 및 최대를 0과 200으로 하는 정규 분포를 따라 다음 위치가 결정되며 y좌표도 기본적으로 x와 같다.

그림 16은 이동체의 센서 데이타와 사무실 영역의 공간 데이타를 보여준다.

그림 16에서 보듯이 이동체 100명에 대한 스냅샷 50회분량 데이타와 고정된 사무실 영역 11구역에 대한 공간 데이타를 생성하였고, 연속적인 처리를 위해서 이동체 데이타를 순환적으로 공간 DSMS에 입력하여 그 결과를 확인하였다.

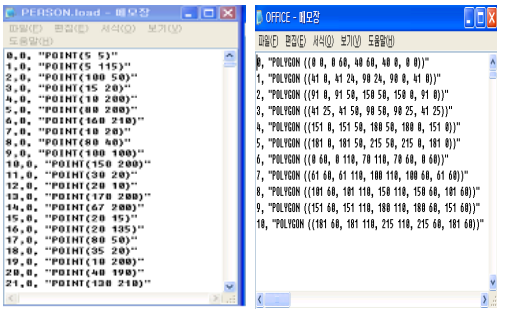

그림 16. 이동체 센서 데이타와 사무실 영역의 공간 데이터

그림 17은 이동체 데이타 입력 툴의 동작 화면을 보여준다.

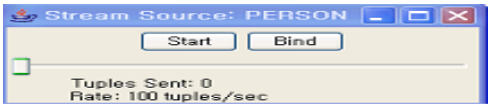

그림 17. 이동체 데이터 입력 툴

그림 17에 있는 데이타 입력 툴은 데이타 화일로부터 데이타를 읽어 들여서 서버로 전송하게 되며 초당 몇 개의 튜플을 보낼지를 조절할 수 있다. 그림 22에서 보이는 다이얼로그 박스는 이동체의 센서 데이타를 전송한다. 이동체 센서 데이타는 각각 초당 100개와 초당 10개를 생성했기 때문에 입력 툴에서 슬라이딩 바를 조절하여 각각 초당 100 튜플을 전송한다.

표 3은 테스트를 위해 사용한 이동체 데이타의 스트림 목록을 보여준다.

표 3. 스트림 목록

이름	타입	속성
PERSON	스트림	id Int, t Int, location Char(30)

표 3에서 PERSON은 이동체의 센서 데이타를 위한 스트림이다. 각 스트림에서 t는 샘플링 시간을 의미하며, location은 위치를 WKT 형태의 문자열로 표현한 것이다.

표 4. 질의문 목록

```
질의 1:
   Select id, t, GeomFromText(PERSON.location, 0)
   From PERSON [range 2 seconds];
질의 2:
   Select PERSON.id, PERSON.t, OFFICE.id
   From PERSON, OFFICE [range 2 seconds]
   Where OFFICE.id > 0 and OFFICE.id < 4
Contain(GeomFromText(OFFICE.boundary, 0),
 GeomFromText(PERSON.location, 0);
```

표 4에 있는 질의들은 다음과 같다. 질의 1은 “최근 2초 동안 PERSON에 입력된 이동체 센서 데이타에서 아이디, 샘플링 시간, 이동체의 위치를 출력하라.”는 질의문이며, 질의 2는 “최근 2초 동안 PERSON에 이동체 센서 데이타 중 사무실 영역 아이디가 0보다 크고 4보다 작은 사무실 안에 위치하고 있는 이동체의 아이디, 샘플링 시간, 사무실 영역 아이디를 출력하라.”라는 질의문이다.

센서 데이타가 포함하는 시간은 시스템 도착 시간(Arrival Time)과 센서가 값을 센싱한 시점의 시간을 의미하는 샘플링 시간(Sampling Time) 두 가지가 있다. 여러 지역에서 동시에 샘플링된 센서 데이타도 시스템에 도착하는 과정에서 시간 차이가 생길 수 있으며 이로 인해 서로 다른 시스템 도착 시간을 갖게 된다.

본 테스트에서는 이것을 고려하여 질의 1과 질의 2에서 시간 윈도우 크기를 2초로 설정하였다. 시간 윈도우는 시스템 도착 시간을 기준으로 계산이 되며 이렇게 시스템 도착 시간과 샘플링 시간을 따로 고려함으로써 샘플링 후 시스템에 도착하는 시간에 지연이 생기더라도 서버에서 2초 동안 윈도우에 저장되어 있는 데이타들과 조인이 가능하다.

그림 18은 질의 2를 만족하는 이동체의 아이디, 샘플링 시간, 사무실 영역의 아이디를 출력한 화면을 보여준다.

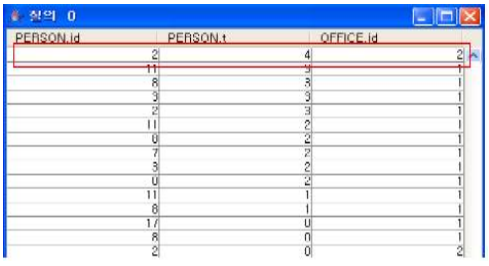

그림 18. 질의 2의 결과 화면

그림 18에 있는 질의 2번의 결과를 보면 현재 시스템 동작 후 사무실 영역 아이디가 2인 장소에 아이디가 2인 이동체가 있다는 것을 알 수 있다. 그림 16에 있는 데이타를 보면 해당 이동체의 위치는 “POINT(100 50)”이고 범위 조건에 해당되는 사무실 영역 아이디가 2인 사무실 영역의 위치는 “POLYGON((91 0, 91 50, 150 50, 150 0, 91 0))”이기 때문에 포함 관계에 있으므로 결과가 정확함을 알 수 있다.

본 논문에서는 RTLS 시스템에서 공간 데이타 스트림 처리 시 일반 DSMS 보다 본 논문에서 확장 개발한 공간 DSMS를 사용하였을 때 보다 효율적인 RTLS 서비스를 제공할 수 있다는 것을 입증하기 위해 다음 그림 19와 같이 RTLS 시스템에서 각각의 질의 수행 시간을 성능 비교하였다.

그림 19와 같이 공간 DSMS 기반 RTLS가 일반 DBMS 기반 RTLS 보다 최소 1.5배에서 최대 3배 빠른 질의 수행 결과를 보였다. 이는 공간 DBMS 기반 RTLS

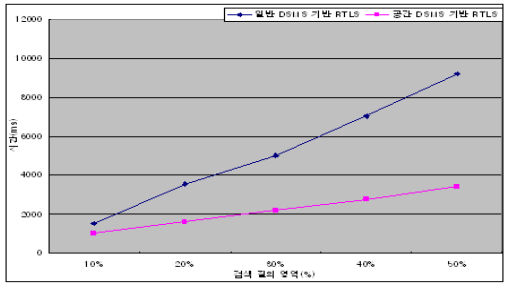

그림 19. 질의 수행 성능 비교

는 RTLS 서비스에서 효율적으로 공간 연산을 처리하기 위해 공간 데이타 타입과 최적화된 공간 연산자를 제공한다. 그리고 공간 DSMS 기반 RTLS는 센서로부터 끊임없이 입력되는 공간 데이타 스트림양을 줄이기 위한 필터링 기능을 제공하기 때문이다.

4.2 구현 환경 및 구현 시나리오

본 시스템을 개발하기 위해 사용한 구현 환경은 다음과 같다. 운영체제는 Fedora core 7을 사용하였고, 개발 언어는 JDK 1.5와 JSP를 사용하였으며, 클라이언트와 서버간의 메시지 교환을 위해 SOAP 1.1 프로토콜을 사용하였다. 웹 서비스 서버 환경을 구축을 위해 J2EE 1.4와 JWSDP 2.0을 사용하였고, 웹 서버로는 Sun Java System Application Server 8.2를 사용하였다.

본 논문에서는 공간 DSMS 기반 RTLS의 효용성을 판단하기 위해 그림 20에서 보는 바와 같이 실시간으로 입력되는 사무실 직원의 위치 데이타를 모니터링하여 회사의 기밀 정보 유출에 신속하게 대응하는 시나리오를 적용해 보았다.

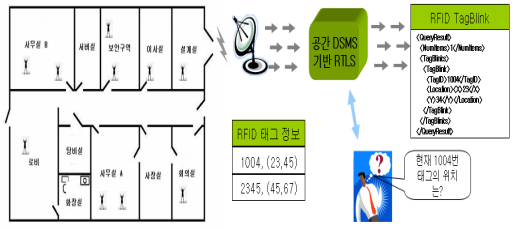

그림 20. 사무실 직원의 실시간 위치 검색 서비스

그림 20에서 보는바와 같이 사용자는 공간 DSMS 기반 RTLS를 통해 사무실 직원들의 현재 위치(태그의 위치)를 실시간으로 모니터링 할 수 있다.

4.3 Query 서비스

Query 서비스는 RTLS로부터 이동 객체의 상태 및 위치 데이타를 일회성으로 검색하는 서비스를 말한다. 그림

21은 Query 서비스의 화면 구성을 보여준다.

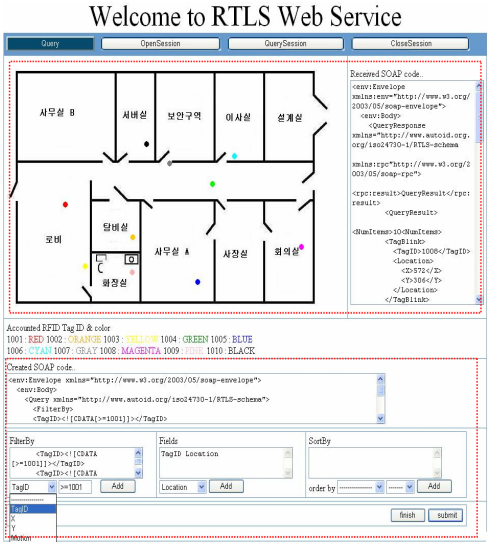

그림 21. Query 서비스 화면

그림 21에서 보는 바와 같이 Query 서비스의 웹 화면은 크게 2개의 영역으로 구성되어 있다. 화면 상단에 있는 2개의 영역은 클라이언트의 디스플레이 관리 모듈이 담당하는 부분이다. 왼쪽 영역은 RTLS로부터 전달받은 이동 객체의 위치를 사무실 이미지와 함께 표시하는 Java Applet 영역이고, 오른쪽 영역은 RTLS로부터 전달받은 SOAP 메시지를 보여주는 영역이다.

화면 하단에 있는 4개의 영역은 클라이언트의 질의 생성 관리 모듈이 담당한다. 아래 왼쪽부터 FilterBy 영역은 FilterBy 엘리먼트를 작성하기 위한 영역이고, Fields 영역은 Fields 엘리먼트를 작성하기 위한 영역이며, SortBy 영역은 SortBy 엘리먼트를 작성하기 위한 영역이다.

화면 하단의 위쪽에 있는 Created SOAP code 영역은 사용자가 FilterBy, Fields, SortBy 영역에서 검색 조건을 작성하고 “finish” 버튼을 클릭하면 사용자가 입력한 조건에 맞게 SOAP 메시지를 자동 생성하여 저장하는 영역이다. 최종적으로 사용자가 Created SOAP code 영역에 생성된 SOAP 메시지를 확인한 후 “submit” 버튼을 클릭하면 Query 서비스에서 작성한 SOAP 메시지가 RTLS로 전달된다.

4.4 OpenSession 서비스

OpenSession 서비스는 RTLS에 새로운 세션을 생성하고 이동 객체의 상태 및 위치 데이타를 QuerySession을 통해 반복해서 검색할 수 있도록 하기 위해 사용하는 서비스를 말한다. 그림 22는 OpenSession 서비스의 화면 구성을 보여준다.

그림 22에서 보는 바와 같이 OpenSession 서비스의 웹 화면은 크게 2개의 영역으로 구성되어 있다. 화면 상단의 오른쪽 영역은 클라이언트의 디스플레이 관리 모듈이 담당하는 부분으로서 RTLS에 생성된 세션 생성 정보에 대한 내용을 SessionResponse SOAP 메시지로 전달받아 화면에 보여주는 영역이다.

화면 하단의 3개 영역과 화면 상단의 왼쪽 영역은 클라이언트의 질의 생성 관리 모듈이 담당하는 부분으로 화면 상단 왼쪽의 Created SOAP Code 영역, 화면 하단의 FilterBy 영역, Fields 영역, SortBy 영역에 대한 내용은 Query 서비스에서 설명한 내용과 같다.

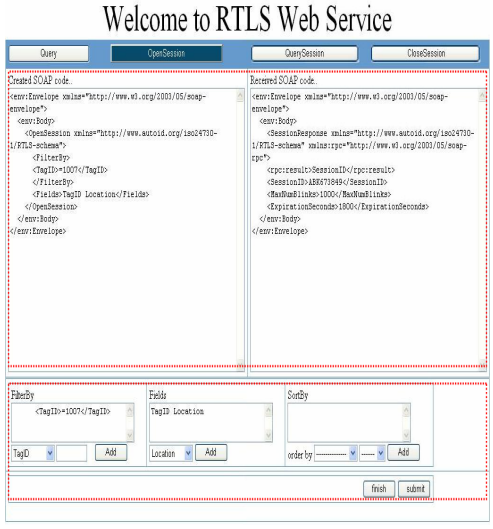

그림 22. OpenSession 서비스 화면

4.5 QuerySession 서비스

QuerySession 서비스는 OpenSession 서비스를 통해 생성된 세션 식별자를 RTLS로 전송하여 해당 세션이 가지고 있는 이동 객체의 상태 및 위치 데이타를 반복적으로 검색하여 클라이언트의 디스플레이 관리 모듈을 통해 보여주는 서비스를 말한다. 그림 23은 QuerySession 서비스의 화면 구성을 보여준다.

그림 23에서 보는 바와 같이 QuerySession 서비스의 웹 화면은 크게 2개의 영역으로 구성되어 있다. 화면 상단의 2개 영역은 클라이언트의 디스플레이 관리 모듈이 담당한다. 왼쪽 영역은 RTLS로부터 전달받은 이동 객체의 위치를 사무실 이미지에 표시하는 Java Applet 영역이고, 오른쪽 영역은 RTLS의 세션에 저장되어 있는 TagBlink들의 내용을 SOAP 메시지로 전달받아 화면에 보여주는 영역이다.

화면 하단은 클라이언트의 질의 생성 관리 모듈이 담당하는 영역으로 OpenSession 서비스에서 저장한 세션 식별자 번호를 읽어와 QuerySession 스키마 작성 규칙에 맞는 SOAP 메시지를 자동으로 생성하여 RTLS로 전송한다.

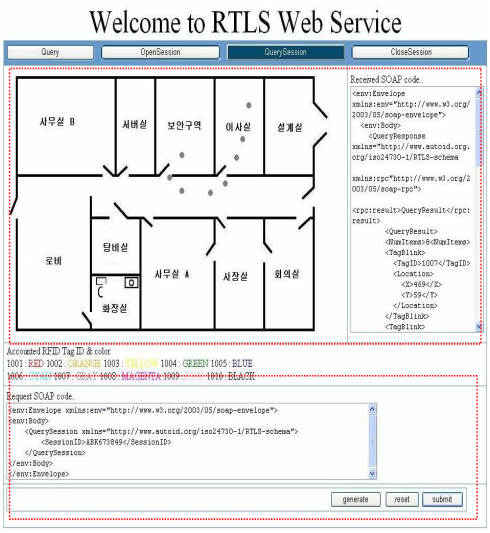

그림 23. QuerySession 서비스 화면

4.6 CloseSession 서비스

CloseSession 서비스는 OpenSession 서비스를 통해 전달받은 세션 식별자를 RTLS로 전송하여 해당 세션에 대한 삭제 처리를 담당하는 서비스를 말한다. 그림 24는 CloseSession 서비스의 화면 구성을 보여준다.

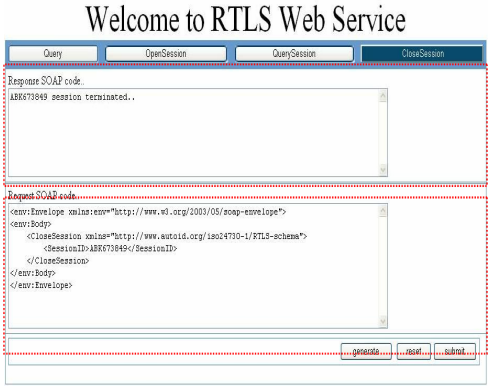

그림 24. CloseSession 서비스 화면

그림 24에서 보는 바와 같이 CloseSession 서비스의 웹 화면은 크게 2개의 영역으로 구성되어 있다. 화면 하단은 클라이언트의 질의 생성 관리 모듈이 담당하며, “generate” 버튼을 클릭하면 CloseSession SOAP 메시지를 자동 생성한다. 화면 상단은 RTLS로부터 CloseSession 서비스 요청에 대한 응답 결과를 보여준다.

5. 결론

최근 유비쿼터스 컴퓨팅 환경에서 RFID나 GPS 등과 같은 위치 측위 기술이 발달함에 따라 이동 객체의 위치 및 상태 정보를 서비스하는 다양한 RTLS가 개발되었고, ISO/IEC에서는 RTLS의 데이타의 호환성과 상호 운용성을 위해 RTLS 표준 명세를 제시하였다. 그러나 기존의 RTLS는 끊임없이 입력되는 데이타 스트림에 대한 처리와 RTLS 서비스에서 필요한 공간 연산에 대한 지원 등이 미흡하기 때문에 이에 대한 보완이 필요하다. 따라서, 본 논문에서는 기존의 STREAM에 공간 데이타 타입과 공간 연사자를 추가하여 공간 DSMS를 개발하고, 또한 이를 사용하여 RFID 태그로부터 수집된 위치 정보를 효율적으로 처리 및 관리할 수 있는 공간 DSMS 기반 RTLS를 설계 및 구현하였다.

본 논문에서 확장한 공간 DSMS는 연속질의 처리를 위해 CQL을 지원하고, 데이타 스트림으로 인해 발생하는 입력 부하를 줄이기 위한 필터링 기능을 지원하며, 기존의 DSMS와는 달리 공간 데이타 스트림이라는 새로운 형태의 데이타 모델을 처리하여 제공하기 때문에 RTLS의 효율성을 향상시킬 수 있다. 그리고, 본 논문에서 개발한 공간 DSMS 기반 RTLS는 ISO/IEC에서 제시한 RTLS 표준 명세를 따르고 있기 때문에 데이타의 호환성과 상호 운용성을 보장하고 OGC에서 제시한 "Simple Feature Specification for SQL" 표준 명세에 따라 공간 연산자를 제공하고 있기 때문에 다양한 형태의 공간 데이타 스트림을 처리할 수 있다.

마지막으로, 본 논문에서는 실험을 통해 RTLS 시스템에서 공간 데이타의 관리 및 처리를 위해 공간 DSMS를 사용하는 것이 효율적이라는 것을 성능 평가를 통해 입증하였다. 또한, 연구 개발한 공간 DSMS 기반 RTLS를 실시간 사원 위치 검색 서비스에 적용하여 시스템의 효용성을 검증하였고, 이를 통해 본 논문에서 개발한 시스템이 이동체의 위치 및 상태 데이타에 대한 실시간 추적 서비스가 필요한 환경에서 유용하게 사용될 수 있음을 확인하였다.

참 고 문 헌

[1] 이충호, 안경환, 이문수, 김주완, “u-GIS 공간정보 기술 동향,” 전자통신동향분석, 제22권, 제3호, 2007, pp.110-123.

[2] Kim, J. J., Hong, D. S., Kang, H. K., and Han, K. J., “TMOM : A Moving Object Main-memory Based DBMS for Telematics Services,” Proc. of the 6th International Workshop on Web and Wireless Geographical Information Systems (W2GIS 2006), Swiss, LNCS 4295, 2006, pp.259-268.

[3] 손해원, 모희숙, 성낙선, “UHF RFID 기술,” 전자통신동향분석, 제20권 제3호, 2005, pp.67-80.

[4] ISO/IEC 24730-1, Real Time Locating Systems (RTLS) - Part 1: Application Programming Interface(API), 2003.

[5] ISO/IEC 24730-2, Real Time Locating Systems (RTLS) - Part 2: 2.4 GHz Air Interface Protocol, 2003.
[6] ISO/IEC 24730-3, Real Time Locating Systems (RTLS) - Part 3: 433 MHz Air Interface Protocol, 2003.
[7] Chen, J., DeWitt, D. J., Tian, F., and Wang, Y., "NiagaraCQ: A Scalable Continuous Query System for Internet Databases," Proc. of the 2000 ACM SIGMOD Intl. Conference on Management of Data, Vol.29, No.2, 2000, pp.379-390.
[8] Abadi, D., Carney, D., Cetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N., and Zdonik, S., "Aurora: A New Model and Architecture for Data Stream Management," VLDB Journal, Vol.12, No.2, 2003, pp.120-139.
[9] Arasu, A., Babcock, B., Babu, S., Datar, M., Ito, K., Nishizawa, I., Rosenstein, J., and Widom, J., "STREAM: The Stanford Stream Data Manager," IEEE Data Engineering Bulletin, Vol.26, No.1, 2003, pp.19-26.
[10] 박치민, 홍동숙, 박춘걸, 한기준, "STREAM을 기반으로 하는 공간 DSMS의 설계 및 구현," 한국공간정보시스템학회 2006년 추계학술대회 논문집, 2006, pp.131-136.
[11] Open Geospatial Consortium Inc., OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 2: SQL option, 2005.
[12] W3Consortium, Simple Object Access Protocol (SOAP) 1.2, 2003.
[13] Arasu, A., Babu, S., and Widom, J., "The CQL Continuous Query Language: Semantic Foundations and Query Execution," The VLDB Journal, Vol.15, No.2, 2006, pp.121-142.
[14] Ramsey, P., The State of Open Source GIS, Refractions Research Inc., 2006.
[15] 박치민, 김동오, 홍동숙, 한기준, "이동 센서를 위한 데이타셋 생성기," 한국GIS학회 GIS/RS 공동 춘계학술대회논문집, 2006, pp.131-137.

김 정 준
2003년 건국대학교 퓨터공학과(공학사)
2005년 건국대학교 대학원 컴퓨터공학과 (공학석사)
2005년~현재 건국대학교 대학원 컴퓨터공학과 박사과정
관심분야는 간 메인 메모리 데이타베이스, GIS, LBS, 텔레매틱스

김 판 규
2007년 군산대학교 정보통신공학과 (공학사)
2007년~현재 건국대학교 대학원 컴퓨터공학과(공학석사)
관심분야는 데이타베이스, GIS, WCF, Silverlight, WPF

김 동 오
2000년 건국대학교 컴퓨터공학과(공학사)
2002년 건국대학교 대학원 컴퓨터공학과 (공학석사)
2006년 건국대학교 대학원 컴퓨터공학과 (공학박사)
2006년~현재 건국대학교 컴퓨터공학부 강의교수
관심분야는 데이타베이스, LBS, GML, 유비쿼터스 센서 네트워크, 정보시스템 감리

이 기 영
1984년 숭실대학교 전자계산학과(공학사)
1988년 건국대하교 대학원 컴퓨터공학과 (공학석사)
2005년 건국대하교 대학원 컴퓨터공학과 (공학박사)
1984년~1991년 한국해양연구원 연구원
1996년~1998년 한국컴퓨터정보학회 이사 및 서울동부지회장
1991년~현재 을지대학교 의료산업학부 교수
관심분야는 공간 데이타베이스, GIS, LBS, USN, 텔레매틱스

한 기 준
1979년 서울대학교 수학교육학과(이학사)
1981년 한국과학기술원(KAIST) 전산학과 (공학석사)
1985년 한국과학기술원(KAIST) 전산학과 (공학박사)
1985년~현재 건국대학교 컴퓨터공학부 교수
1990년 Stanford 대학 전산학과 Visiting Scholar
2000년~2002년 한국정보과학회 데이터베이스연구회 운영위원장
2004년~2006년 한국공간정보시스템학회 회장
2004년~2008년 한국정보시스템감리사협회 회장
관심분야는 공간 데이타베이스, GIS, LBS, 텔레매틱스, 정보시스템 감리