빈번한 변경을 요구하는 실시간 스트림 데이터의 효율적 관리 및 슬라이딩 윈도우 질의

김 진 덕*

An Efficient Management and Sliding Window Query for Real-Time Stream Data to Require Frequent Update

Jin-Deog Kim*

이 논문은 2006년도 동의대학교 교내연구비 지원에 의해 작성되었음

요 약

최근 다수의 외부 장치를 제어하는 시스템에서는 빈번하게 변하는 신호의 이력을 자동적으로 관리하는 기법이 요구된다. 그 신호들은 스트림데이터로서 다양한 종류, 짧은 보고 주기, 비동기적인 보고 시간을 가진다. 또한 처리 시스템은 스트림데이터에 대해 높은 신뢰성과 실시간 처리를 필요로 한다. 그리고 스트림 데이터에 대한 질의는 최신의 값을 검색하는 현재질의, 과거 특정시점의 값을 검색하는 스냅샷 질의, 과거부터 현재까지의 값들을 검색하는 슬라이딩 윈도우 질의 등이 있다.

이 논문에서는 소규모 운영체제에서 파일 구조화된 데이터베이스를 이용하여 스트림 데이터들을 효율적으로 저장하고 관리하는 기법을 제안하고자한다. 그리고 스트림 데이터에 대한 슬라이딩 윈도우 질의를 포함한 다양한 질의를 수용하는 질의 모델을 제안한다. 파일 기반 데이터베이스는 QNX의 적은 저장장치, 낮은 계산 능력을 감안하여 델타버전과 공유메모리 버퍼링 등의 방법을 도입한다.

ABSTRACT

Recently, the operator modules to control external devices are concerned about automatic management system to process continuously changed signals. These signals are the stream data of which characteristics are several numbers, a short report interval and asynchronous report time. It is necessary that the system brings about high accuracy and real time process for stream data. The typical queries of these systems consist of the current query to search the latest signal value, the snapshot query at a past time, the sliding window query from a past time to current.

In this paper, we propose the efficient method to manage the above signals by using a file structured database in small-size operating systems. We also propose a query model to accommodate various queries including the sliding window query. The file database in the QNX adopts a delta version and a shared memory buffering method for the resource limit of a small storage and a low computing power.

키워드

스트림데이터, Profibus-FMS 통신, 델타버전, 슬라이딩윈도우질의

* 동의대학교 컴퓨터공학과 부교수

접수일자 : 2007. 11. 22

Ⅰ. 서 론

지금까지의 데이터베이스 관리 시스템(DBMS)은 주로 문자데이터들을 저장하는 기능과 간혹 변경된 데이터에 대한 최신성을 유지하는 기능을 가진다. 그리고 결합연산, 집합연산 등 다양한 분석이 포함된 질의를 통해 응용 프로그램이 요구하는 정보를 제공하는 기능을 수행하여 왔다.

그렇지만 최근 산업 생산현장의 자동화가 활발히 진행되면서 기존 문자데이터와는 달리 센서네트워크로부터 끊임없이 전달되는 신호 데이터를 활용하는 응용 분야가 증가하고 있다.

이와 같은 데이터를 스트림 데이터(Stream Data)라고 하며, 매우 많은 종류, 짧은 보고 주기, 비동기적인 보고 시간, 끊임없이 변하는 특성을 가진다.

이러한 스트림 데이터는 데이터 전체를 저장하는 것은 공간 효율성이 매우 떨어지며, 질의 응답속도 또한 보장할 수 없다. 한편, 스트림 데이터는 소규모 특화된 운영체제에서 운용되는 경우가 많다. 따라서 스트림 데이터베이스 관리 시스템은 기존 DBMS와는 달리 높은 공간 효율성을 보장하는 스트림 데이터 관리 및 저장 기법과 스냅샷 질의 및 슬라이딩 윈도우 질의 등 새로운 질의 처리 기법이 요구된다.

그리고 센서 네트워크를 이용한 응용으로서 각종 외부 기기와의 통신을 통해 상태를 실시간으로 감시하고, 제어하는 시스템을 많이 이용하고 있다. 원자로 보호 계통 운전원 모듈 또한 그 대표적인 예이다. 기 개발된 원자로 보호계통은 핵증기 공급계통의 상태를 감시하여 정해진 안전계통 설정치에 도달하면 원자로를 정지시키는 기능을 제공하며, 계측채널의 신호를 수집하고 설정치를 비교하는 각종 주변 기기 및 제어스위치로 구성된 캐비넷운전원모듈(COM), 개시회로 등으로 구성되어 있다. 이러한 원자로 안전 계통 데이터는 실시간으로 끊임없이 새로운 값이 도달하는 스트림 데이터이다.

그러나 지금까지의 원자로 보호계통 운전원 모듈은 각 외부 장치로 부터의 현재 신호를 점검하여 동작상태를 검증하는 방식이었다. 그러나 최근에는 현재 신호 값뿐만 아니라 복잡하고 다양해지는 원자로 안전장치에 대한 과거 특정 시점의 신호 상태나 과거부터 현재까지의 이력 정보를 검색하는 기능과 기존 값으로부터 특정 함수로 연산 처리된 정보를 추출하는 논리 연산 기능 등이 필요하다.

따라서 본 논문에서는 원자로 보호계통의 구성 모듈로부터 입력되는 스트림 데이터들을 실시간으로 데이터베이스화하고, 현재와 과거 신호 및 이력 신호 정보를 검색할 수 있는 슬라이딩 윈도우 질의 모델을 설계하고자 한다.

제안한 시스템을 구현하기 위한 메시지 송수신 수단으로 Profibus-FMS 프로토콜을 사용하며, 원자로 보호계통 운전원 모듈이 운영되는 QNX 운영체제를 사용한다. QNX는 소규모 운영체제로서 적은 저장장치 용량과 낮은 계산능력을 지닌다. 따라서 이러한 QNX의 특징의 고려하고, 빈번한 변경이 요구되는 신호의 실시간 처리를 위해 파일 기반 스트림 데이터베이스를 구축한다. 특히 저장 공간의 효율화를 위해 델타버전(Delta Version)을 사용하며, 실시간 검색을 위한 IPC(Inter Process Communication) 기반의 버퍼링 기법을 도입한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 스트림 데이터 및 원자로 안전 모듈에 대한 관련 연구를 살펴보고, 제 3장에서는 이 논문에서 설계하고자 하는 DB 시스템 구성도 및 데이터 송수신 모듈, 델타버전을 이용한 스트림 데이터베이스 구축방법과 공유메모리를 이용한 프로세스간 통신에 관해 설명한다. 그리고 제 4장에서는 슬라이딩 윈도우를 포함한 질의 모델 등에 대해 자세히 설명하며, 제 5장에서 결론을 내린다.

Ⅱ. 관련 연구

2.1 스트림 데이터 관련 연구

데이터 스트림 처리에 관한 다양한 연구가 진행되어 왔다[4, 5, 7, 8]. Aurora[8]와 Telegraph[4]는 관계기반 질의어를 이용하여 슬라이딩 윈도우 질의를 제공한다.

그리고 Niagara[5]와 Telegraph{4]에서는 질의 색인을 활용하여 연속질의의 성능을 높이고자 하였다. 이 색인은 질의 대상 속성의 간격을 보유하고 있으며, 센서 노드로부터 이벤트가 발생할 때 이진 탐색을 기반으로 질의의 결과 값을 도출한다.

관련 연구 [7]에서는 최근 활성화되고 있는 RFID 미들웨어 시스템에서 연속질의를 원활하게 처리하기 위한 색인을 제시하였으며, 색인 구성에 질의를 기반으로 하였다.

위와 같이 데이터 스트림에 대한 다양한 질의어 모델이 제시되었지만, 지금까지 QNX와 같은 소규모 특수 목적 운영체제하의 데이터스트림에 관한 효율적인 관리 및 슬라이딩 윈도우 질의에 관한 연구는 연구된 바 없다.

2.2 원자로 보호 계통 관련 연구

기존의 원자로 보호계통 시스템은 핵증기공급계통의 상태를 감시하여 정해진 안전계통 설정치에 도달하면 원자로를 정지시키는 기능을 제공하며, 계측채널의 신호를 수집하고 설정치를 비교하는 비교논리프로세서(BP), 다중 채널의 트립신호를 Voting하는 동시논리프로세서(CP), 시험 및 유지보수를 관장하는 자동시험 및 연계프로세서(ATIP), 운전원화면 및 제어스위치로 구성된 캐비넷운전원모듈(COM), 개시회로 등으로 구성되어 있다. 또한 이들 각 구성장치들은 Profibus-FMS 프로토콜[2,3]을 이용하여 정보를 송수신하도록 연결되어 있다. Profibus는 실시간 처리 요구조건을 만족한다.

운전원 모듈의 운영체제로는 QNX[1]를 사용하고 있다. QNX는 실시간운영체제로서 임베디드시스템을 위한 운영체제로 각광받고 있으며, 특히 국내의 자동차 회사에서는 QNX Neutrino를 자사의 텔레매틱스를 위한 운영체제로 채택하고 있다[3].

본 논문에서는 PhAB(Photon Application Builder)를 이용하여 운전원 모듈의 GUI 응용프로그램을 지원하며, Profibus-FMS 카드를 이용하여 운전원 모듈에서 필요로 하는 정보를 실시간으로 수집하고 효율적으로 스트림 데이터를 관리하는 기법을 제시한다. 이에 따라 Profibus-FMS 카드를 이용한 메시지 처리 루틴을 개발하여야 하는데 현재 고려중인 Profibus-FMS 카드인 Hilscher사의 CIF-50PB, 80PB를 위한 API를 이용한다[5]. 또한 빈번하게 변경되는 실시간 스트림데이터에 대한 다양한 질의어 모델을 제시하고자 한다.

Ⅲ. 신호의 실시간 처리 기법

3.1 COM DB 시스템 구성도

COM 모듈은 BP, CP, ATIP, ETIP와 같은 주변 기기로부터 시험신호를 받아들여 각 장치의 상태를 조작자에게 보고하는 역할을 수행한다. 또한 조작자가 필요한 정보를 주변기기에 보내는 역할 또한 수행한다. 그러나 지금까지 시스템은 끊임없이 변경되며 누적되는 신호에 대한 체계적인 관리가 없어 현재 질의만 가능하고, 스냅샷 질의 및 슬라이딩 윈도우 질의는 불가능하였다.

이에 본 논문에서는 그림 1과 같이 주 운전원 모듈인 COM과 외부 장치 간에 스트림 데이터베이스 시스템을 두어 실시간으로 변경되는 신호들을 체계적으로 관리하며, 질의에 응답할 수 있는 환경을 제공하고자 한다.

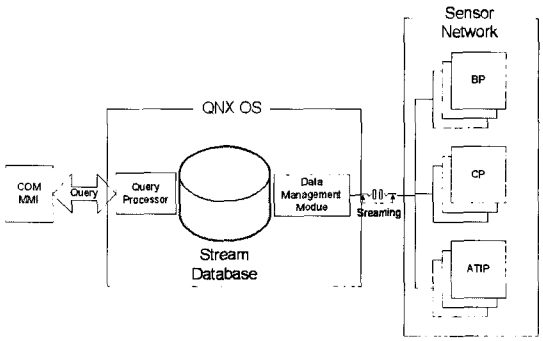

그림 1. COM DB 시스템 구조도
Fig. 1. COM DB System Architecture

스트림 데이터베이스 시스템은 소규모 모바일 QNX 운영체제 하에서 구축되며, 각 장치(BP, CP, ATIP)로부터 입력되는 신호는 약 10,465개이며, 출력되는 신호는 200개이다. 스트림 데이터베이스가 유지 및 관리하는 신호의 관리 요건 등은 다음과 같다.

- 신호의 대부분은 Historical Data로 유지되어야 하며, 일부 신호는 특정 값이 감지되는 즉시 조작자에게 통보되어야 한다.
- COM 장비에서 이들 데이터를 수집할 때 신호별로 집합체를 구성하여 전송되기도 하며, 일부 신호(Logical Signal)는 직접 입력되는 신호들(Physical Signal)의 조합에 의한 연산결과 값이 데이터베이스에 저장되기도 한다.

스트림 데이터베이스 시스템은 COM 장비에 탑재되어 동작되며, COM 장비의 리소스 한계(적은 메모리, 낮은 연산능력)와 사용되는 질의에 적합한 파일 구조 데이터베이스를 설계하고자 한다.

3.2 데이터 송수신 모듈

그림 1의 스트림 데이터는 주변 장치들로부터

Profibus 카드를 통해 수행된다[2,3].

Profibus 카드를 이용한 통신의 수행은 그림 2와 같이 COM 모듈에 장착된 Profibus 카드와 각 외부 장치 Profibus 카드의 통신 버퍼가 대응되는 구조이다[3].

본 논문에서는 COM에서 DevPutMessage()함수와 DevGetMessage()함수를 이용하여 입력되는 신호를 획득하여 DB에 저장하는 방식을 택한다. 신호의 획득을 위해 Profibus 카드내의 OD값을 읽거나 쓰기 작업을 수행하며, 구조체 자료형인 MSG_STRUC를 이용하여 보다 동기화된 신호의 송수신이 가능하다.

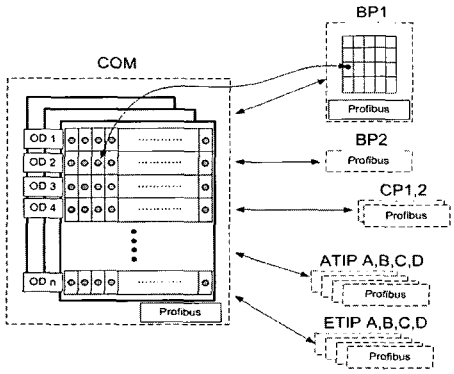

그림 2. Profibus를 이용한 통신
Fig. 2. Communication using Profibus

그림 2에서 각 Profibus에는 여러 개의 OD(Object Dictionary)가 있으며, 각 OD는 여러 개의 Address와 bit로 세분된다.

그림 3은 OD의 구조를 나타낸 것이다. 각 OD는 230바이트의 데이터영역이 있으며, Configuration 파일에 따라 각 Slot이 2, 4, 8 byte 등으로 구성된다.

그림 3의 case 1은 각 Slot이 2바이트이며 이 경우에는 각 Slot마다 정수값(int)또는 부울(Boolean)값이 비트 단위로 저장된다. 그림 3의 case 2는 각 Slot이 4바이트이며 이 경우에는 각 Slot마다 실수값(real)또는 부울(Boolean)값이 비트 단위로 저장된다. 이 외에도 1 또는 4 바이트 정수값을 저장하는 Slot을 지정할 수도 있다. 이 논문에서는 각 신호의 유형별로 OD를 지정하였으며, 해당 그룹 별로 색인의 주기를 달리하는 기법을 통해 효율적으로 데이터베이스를 구축한다.

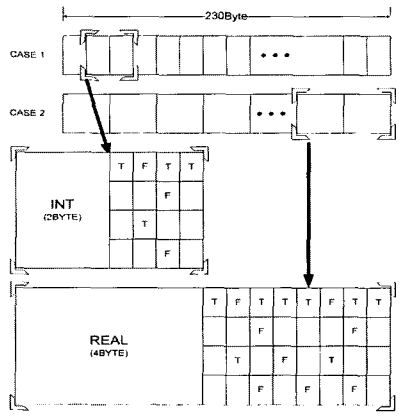

그림 3. OD 구조도
Fig. 3. Structure of OD

그림 4는 본 시스템에서 사용되는 신호의 예를 나타낸 것으로서 OD 칼럼은 OD의 번호를, Address는 OD 내에서의 Slot의 번호를, Bit는 각 Slot내에서의 비트 번호를 각각 의미한다. 전술한 바와 같이 10,000여개의 신호 전체가 수초 이내에 한 번씩 비주기적으로 매우 빈번히 입출력되며, 이 들 신호에 대한 이력관리와 실시간 현재값을 질의의 결과로 제공되어야 한다.

그림 4의 신호 중에서 OD 20은 각 Slot이 2바이트 정수형 자료로 구성되며, Bool 신호는 성격에 따라 하나의 Slot을 차지할 수도 있고, 27번 Slot처럼 각 비트를 최대한 활용하여 16개의 Bool 신호를 하나의 Slot에 저장할 수도 있다.

다음은 전술한 구조체 MSG_STRUC의 구성요소 및 특징을 나타낸 것이다.

```
typedef struct tagMSG_STRUC {
   unsigned char rx;    //Receiver
   unsigned char tx;    //Transmitter
   unsinged char ln;    //Length
   unsigned char nr;    //Number
   unsigned char a;     //Answer
   unsigned char f;                 //Fault
   unsigned char b      ;           //Command
   unsigned char e      ;           //Extension
   unsigned char data[255];         //Data
   unsigned char dummy[25]; // Compatible
} MSG_STRUC;
```

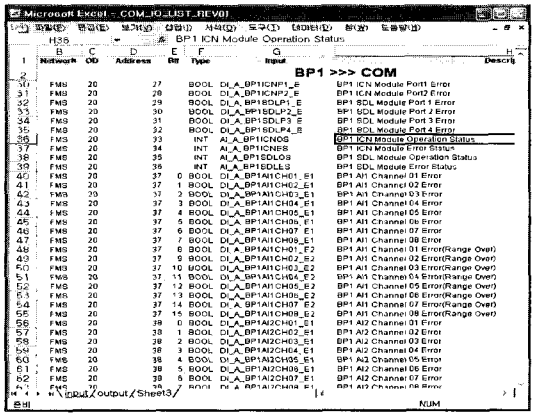

그림 4. 원자로 보호계통 신호의 예
Fig. 4. Signal Example of Reactive Protection System

여기서 구성요소 ln은 바이트단위로 tMessage.data[0] 이후로의 개수를 의미하며, nr은 메시지의 ID를 의미한다. data[2]는 OD 번호이며, data[5]는 data[8] 이후의 데이터 개수이며, 데이터 송신 시에는 실제 데이터를 tMessage.data[8]이후에 저장하면 된다. data[6]은 각 Slot의 데이터 타입으로서 1:integer16, 2:integer8, 4:integer32, 5:unsigned int8, 6:unsigned int16, 7:unsigned int32 등등을 지정할 수 있으며, data[8~255]는 전송할 데이터가 저장된다. 이것은 data[6]에서 지정한 타입에 따라 배열의 한 element의 타입이 지정된다.

이 논문에서는 원자로 보호 계통의 신호가 2바이트 정수, 2바이트 실수, 1비트 Bool값으로 구성되므로 각 OD는 2byte 단위의 Address로 구성하였으며, 저장 용량의 압축을 위해 각 비트를 최대한 활용하여 16개의 Bool 신호를 하나의 Slot에 저장할 수 있도록 하였다[6].

3.3 델타버전을 이용한 파일 DB 구축

전술한 Profibus 카드의 Configuration을 토대로 획득한 각 외부 장치의 신호를 파일 구조 기반 DB에 저장하고 관리한다. 원자로 보호계통 신호는 개수가 매우 많고, 빈번히 변하는 특성을 가지며, 특정 신호는 거의 변하지 않고, 주기에 따라 변하는 특성 또한 갖고 있다. 따라서 이 논문에서는 델타버전 기법을 이용한다. 주기적으로 전체 신호가 포함된 Master File를 구축하고, 일정 주기별로 이전 데이터와의 차이값 만을 수용하는 Delta 파일을 만들어 QNX의 작은 저장구조에 적합하도록 하였다.

각 외부 장치로부터 입력되는 신호 값이 이전 값과 같은 경우 무시되며, 값이 변화하는 값을 저장한다. 델타버전은 전체 신호 중 동일한 신호에 대한 값이 두 번째 변경될 때 새로운 버전을 생성한다. 그림 5는 이를 도식화한 것이다.

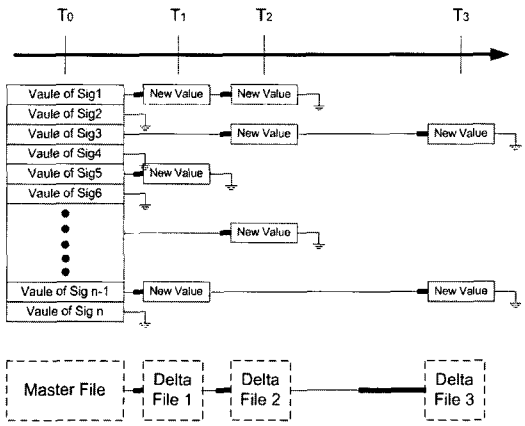

그림 5. Master 파일과 델타파일
Fig. 5. Master File and Delta File

그림 5에서 초기에 Master 파일이 생성되며, Delta 파일 1과 2는 3개의 변경된 값만을 갖고, Delta 파일 3은 3개의 변경된 값만을 갖는다. 각 파일은 생성 시간, 유효 신호 개수를 포함한다. 또한 신호 도착 시간이 매우 가변적이고, 값의 변화 또한 비주기적으로 변화하므로 델타파일의 생성 시간 또한 일정하지 않음을 그림 5에서 확인할 수 있다.

이 논문에서는 저장 공간의 효율성 및 질의 속도의 향상을 위해 각 파일은 논리적으로만 존재하며, Master 파일과 여러 개의 Delta 파일이 하나의 물리적인 파일 내에 포함되도록 하였다.

또한 관리의 용이함을 도모하기 위해 하나의 물리적인 파일의 크기가 일정 규모가 될 때까지 델타파일을 생성한다. 따라서 신호의 값이 거의 변하지 않거나, 신호의 전달 주기가 길 경우 하나의 물리적인 파일은 매우 긴 시간을 포함할 수 있다. 이 논문에서는 델타버전의 개수가 20개에서 80개가지 가변적으로 변함을 알 수 있었다.

그리고 각 물리적 파일의 이름과 생성시간을 연속적으로 저장하는 메타데이터와 가장 최근의 파일 명을 메타 데이터로 관리한다.

3.4 공유 메모리를 이용한 IPC

이 논문에서는 그림 6과 같이 실시간 현재 값을 검색하기 위해서는 공유 메모리를 이용하며, 이력 질의를 위해서는 델타 버전 데이터베이스를 이용한다. 이는 QNX의 낮은 계산 능력과 실시간 검색이라는 두 가지 요소를 감안한 것이다.

그림 6에서 프로세스간의 통신은 공유 메모리를 이용하는데, 이는 스트림 데이터의 특성상 실시간 처리를 하기 위함이다. 공유메모리를 통해 3개의 프로세서(INPUT_PROC, 현재 질의 처리 모듈, DB 생성 모듈)가 실시간 동시 통신을 한다.

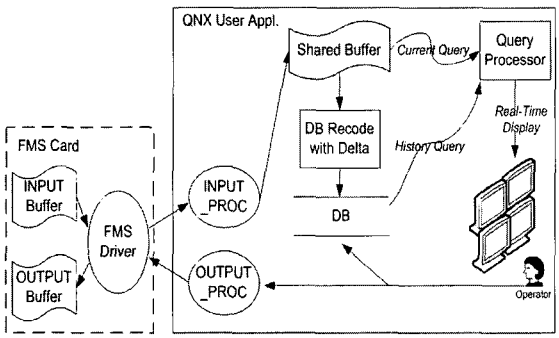

그림 6. 공유메모리를 이용한 프로세스간 통신
Fig. 6. IPC using Shared Memory

우선, 입력 신호 값들을 INPUT_PROC 프로세서가 받아 공유메모리에 실시간 쓰기 작업을 수행한다. 따라서 항상 최신의 신호 값을 보관하고 있다.

DB 생성 모듈은 전술한 델타버전을 이용하여 실시간으로 입력되는 공유메모리의 신호 값을 데이터베이스에 저장하는 역할을 한다.

그리고 질의 처리기는 '현재 질의'에 대한 요구가 있을 때 공유메모리를 참조하여 운영자에게 보여주는 역할을 하며, '이력(History) 질의'에 대한 요구가 있을 때는 저장되어 있는 데이터베이스로부터 자료를 검색하여 운영자에게 보여준다.

Ⅳ. 질의 모델

일반적으로 스트림데이터에 대한 질의는 일정 기간 동안 또는 일정 횟수에 걸쳐 연속적으로 수행되며, 새로운 데이터가 도착할 때 마다 즉시 반영하는 특징이 있다. 이러한 질의를 연속질의(continuous query)라 한다. 원자로 안전 보호 계통의 데이터들은 신호 값들을 현재부터 과거 특정 신호까지 연속적으로 모니터링하는 경우가 많다.

질의의 과거 신호에 대한 경계는 슬라이딩 윈도우를 사용하여 지정한다. 이 논문에서는 윈도우의 사이즈의 범위를 지정하는 방법, 검색하고자하는 신호의 개수, 질의생존 기간, 질의 연산 방법, 출력하는 방법 등에 따라 다양한 질의를 수행할 수 있다.

그림 7에서 질의 윈도우(*WINDOW*)의 크기를(Tstart, Tend)를 기준으로 질의는 크게 다음과 같은 5가지로 분류할 수 있음을 보여준다. 이 때 Tnow는 현재시점으로서 시간이 흐름에 따라 Tnow도 변한다.

1) 과거의 특정 시점의 신호 값을 검색하는 Snapshot 질의. 이 때 Tstart와 Tend는 같음.
2) 과거의 특정시점 Tstart부터 Tend까지의 신호 값을 검색하는 Past History 질의
3) 과거의 고정된 특정시점 Tstart부터 현재시점(Tnow)까지의 신호 값을 검색하는 Current History with Closed Past 질의
4) 과거의 변화하는 시점 Tstart로부터 Tnow까지의 신호 값을 검색하는 Current History with Open Past 질의
5) Tnow시점의 신호 값을 검색하는 Current 질의

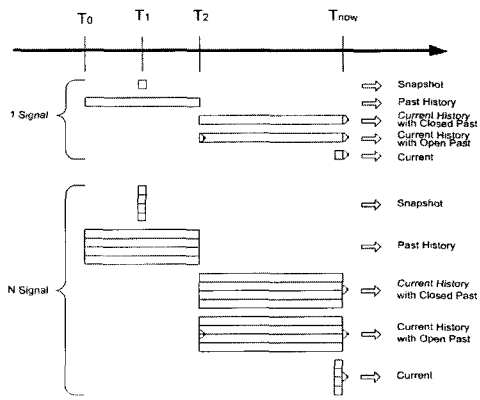

그림 7. 질의 모델
Fig. 7. Query Model

이상의 윈도우사이즈는 시간으로 정의한 것이지만, (Tend, #of data)와 같이 최종시간과 데이터의 개수로 정의할 수도 있다. 예를 들어 Snapshot질의는 (Tend, 1)로

표현된다. 이 논문에서는 시간으로 슬라이딩 윈도우의 크기를 지정한다.

그리고 출력되는 값(*SELECT*)은 기본 데이터값, 통계값, 기정의된 연산 결과값으로 구분할 수 있다. 통계값은 max, min, avg와 같은 함수를 적용할 수 있다. 그리고 기정의된 연산(Predefined OP)은 해당 신호가 물리적인 실제값인지, 논리적인 연산으로 유도 값인지를 표현한다. 그 값이 0일 경우 실제 신호 값임을 의미하며, 1 ~ 6까지는 정의된 연산에 의해 비트 단위의 and와 or 연산을 수행하여 기존 신호 값들로부터 유도된다.

또한 검색하고자 하는 신호의 이름을 정의하는 것(*FROM*)은 한 개의 신호 값을 검색하느냐, 여러 개의 신호 값을 검색하느냐로 구분할 수 있다.

또한 일정 조건(*WHERE*)을 만족하는 값만을 출력할 수 있으며, 질의 생존 기간(*TIME*)은 연속질의가 수행되는 시간을 의미하는 것으로서 알람이벤트는 1회와 사용자가 지정한 시간으로 표현할 수 있다.

그리고 출력되는 방법(*TO*)에 따라 PhAB에 의한 검색은 QNX의 PhAB와 Web기반의 TextOut으로 구분된다.

이상과 같은 슬라이딩 윈도우 질의를 조합하여 다음과 같은 질의 모델을 생성할 수 있다.

SELECT	Value | Aggregate_Function | Predefined_OP
FROM	{SignalID}+
WHERE	Predicate
WINDOW	Tstart, Tend
TIME	Interval
TO	PhAB, Text

- Aggregate_Function := max|min|avg
- Predefined_OP : 7 type OP
- Predicate := <,<=,=,>=,>, AND, OR
- Tstart := Ti(closed), Ti+(open), Tnow(open)
- Tend := Tj(closed), Tnow(open)
- Interval := 1, digit

아래의 SQL 구문은 Current 슬라이드 윈도우 질의로서 DI_A_BP1_E1 신호의 실시간 변화되는 값을 즉시에 PhAB 화면에 반영한다.

SELECT	Value
FROM	DI_A_BP1_E1
WINDOW	Tnow, Tnow
TO	PhAB

아래의 SQL 구문은 Past History 슬라이드 윈도우 질의로서 과거의 일정 기간(20070507113744 -20070507114047) 동안의 여러 개의 신호 값(AI_A_BP1HB, AI_A_BPIOSCRC)들을 SAMBA를 통해 웹페이지로 보여준다.

SELECT	Value
FROM	AI_A_BP1HB, AI_A_BPIOSCRC
WINDOW	20070507113744, 20070507114047
TO	Text

아래의 SQL 구문은 Current History with Open Past Set 슬라이드 윈도우 질의로서 출력된 결과가 시간이 흐름에 따라 최근 n개의 신호 값을 그래프로 출력한다.

SELECT	Value
FROM	AI_A_BP1APCRC
WINDOW	20070507113923, Tnow
TO	PhAB

이 논문에서 제안한 기법은 기존 시스템에서는 고려대상이 아니지만, 소규모 운영체제인 QNX와 Profibus 통신이라는 적은 용량의 저장장치와 동기화된 통신 특성을 감안하여 제안하였으므로, 향후 특수 목적 제어 시스템에 적절히 활용될 것으로 판단된다.

V. 결 론

이 논문에서는 소규모 운영체제에서 파일 구조화된 데이터베이스를 이용한 스트림 데이터를 효율적으로 관리하는 기법과 스트림 데이터에 적용가능한 슬라이딩 윈도우 질의를 설계하였다.

특히, 지금까지 시도된 바가 없는 안전 운전원 모듈의 QNX 운영체제에서 저장장치와 낮은 계산 능력을 고려하여 델타버전과 공유메모리 버퍼링 기법을 제안하였고, 이를 기반으로 스트림 데이터베이스 구축하였다. 그리고 다양한 질의가 가능하게 하였다.

이와 같은 연구 결과는 QNX를 널리 사용하는 국내의 텔레매틱스/스마트 홈 산업에서도 기술의 활용도가 높을 것으로 판단된다. 또한 다양한 센서데이터를 관리하고 질의를 제공하는 소규모 임베디드 시스템에 효과적

으로 적용할 수 있을 것으로 판단한다.

향후, 도로 네트워크의 센서들로부터 입력된 실시간 교통 정보를 활용하여 목적지까지 도달하는 최단 시간 경로 탐색 알고리즘을 개발하고, 이 논문에서 제안한 스트림데이터베이스 기법을 적용하고자 한다.

참고문헌

[1] QNX Software Systems, http://www.qnx.com/

[2] PROFIBUS International Open Solutions for the World of Automation, http://www.profibus.com/

[3] "PROFIBUS Specification," Normative Parts of PROFIBUS - FMS, -DP, -PA according to the European Standard EN 50 170 Volume 2, Edition 1.0, March, 1998

[4] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J.M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V Raman, F Reiss, M Shah, "TelegraphCQ:Continuous Dataflow Processing for an Uncertain World", Proc. of CIDR Conf. 2003

[5] J Chen, D. J. Dewitt, F Tian, Y Wang, "NiagaraCQ : A Sxalable Continuous Query system for Internet Databases", ACM SIGMOD. pp.379-390, 2000

[6] 김진덕, 진교홍, 이성진, 정해원, "빈번한 변경이 요구되는 데이터의 효율적인 실시간 처리 기법", 한국해양정보통신학회 춘계종합학술대회, 10권 1호, 2006

[7] 박재관, 홍봉희, 반재훈, "RFID 스트리밍 데이터 처리를 위한 연속 질의의 변환 기법", 정보처리학회논문지 D, 14-D권, pp. 273-284, 2007

[8] D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, S. B. Zdonik, "Monitoring Streams - A New Class of Data Management Applications", VLDB pp. 215-226, 2002

저자소개

김 진 덕(Jin-Deog Kim)

1993년 부산대 컴퓨터공학과 (공학사)
1995년 부산대 대학원 컴퓨터공학과 (공학석사)
2000년 부산대 대학원 컴퓨터공학과(공학박사)
1998.3~2001.2 부산정보대학 정보통신계열 전임강사
2001.3~ 현재 동의대학교 컴퓨터공학과 부교수
※관심분야 : 객체 지향 DB, 지리정보시스템, 공간 질의, 공간 색인, 모바일 데이터베이스, 텔레매틱스, GIS 스마트 동기화, 스트림 데이터베이스