시간지원 데이터베이스의 질의처리 시스템 구현

이 언 배† · 김 동 호†† · 류 근 호†††

요 약

시간지원 데이터베이스는 유효시간과 거래시간을 통해 객체에 대한 효율적인 이력관리를 제공한다. 유효시간은 현실세계에서 객체에 발생한 시간을 의미하며, 거래시간은 객체가 데이터베이스에 수록된 시간을 의미한다. 시간지원 질의처리 시스템은 기존의 관계형 데이터베이스에서 정의된 연산 뿐만 아니라 사용자 질의에 내포된 이력정보에 관련된 연산을 처리할 수 있도록 확장되어야 한다. 따라서 이 논문에서는 시간지원 질의언어인 TQuel(temporal query language)을 대상으로 하는 이전에 제안된 시간지원 질의 처리 시스템에 대하여 시간지원 구문분석과 시간지원 의미분석, 그리고 시간지원 코드 생성 및 시간지원 실행기 등의 기본적인 요소를 기반으로 거래시간 관리, 시간지원 집계, 시간지원 뷰, 그리고 시간지원 죠인 및 최적화 기능을 추가로 설계 구현하고, 그 처리과정을 예를 들어 설명한다.

Implementation of Query Processing System in Temporal Databases

Eun Bae Lee† · Dong Ho Kim†† · Keun Ho Ryu†††

ABSTRACT

Temporal databases support an efficient historical management by means of valid time and transaction time. Valid time stands for the time when a data happens in the real world. And transaction time stands for the time when a data is stored in the database. Temporal Query Processing System(TQPS) should be extended so as to process the temporal operations for the historical informations in the user query as well as the conventional relational operations. In this paper, the extended temporal query processing systems which is based on the previous temporal query processing system for TQuel(Temporal Query Language) consists of the temporal syntax analyzer, temporal semantic analyzer, temporal code generator, and temporal interpreter is to be described. The algorithm for additional functions such as transaction time management, temporal aggregates, temporal views, temporal joins and the heuristic optimization functions and their example how to be processed is shown.

1. 서 론

기존의 데이터베이스는 대부분 관계형 모델(relational model)을 기반으로 다양한 응용분야에서 매우 효과적으로 사용되어왔다. 최근들어 관계형 모델을 기반으로 하는 기존의 데이터베이스를 보다 다양한 응용분야에서 효과적으로 사용하기 위한 수많은 연구들이 진행되고 있는데, 대표적으로 공간 데이터베이스(spatial databases)와 지리정보시스템(geographic information system), 객체지향 데이터베이스(object-oriented databases) 및 과학 데이터베이스 등을 들 수 있다.

※ 이 연구는 통상사업부 공업기반기술사업의 지원과 정보통신 대학원 지원사업으로 수행되었음.
† 정 회 원 : 한국방송통신대학교 전자계산학과 교수
†† 준 회 원 : 충북대학교 대학원 전자계산학과 박사과정(수료)
††† 종신회원 : 충북대학교 컴퓨터과학과 교수
논문접수 : 1998년 3월 3일, 심사완료 : 1998년 4월 14일

이러한 연구들은 현실세계에서 존재하는 객체에 대해서 오직 현재 시점에서 유효한 버전만을 관리하기 때문에 과거에 유효했던 이력 데이터(historical data)가 요구되는 통계적 데이터 처리, 법령의 저장, 감사 추적, 범죄 추적, 그리고 의료 분야 및 데이터 웨어하우징(data warehousing)등과 같은 광범위한 응용분야에서는 로그 파일(log file) 등과 같은 별도의 처리 방법이 필요하기 때문에 처리 비용이 증가할 뿐만 아니라 사용자에게 복잡한 질의작성이라는 부담을 주는 문제점을 갖는다. 따라서 현재 시점에서 유효한 자료뿐만 아니라 과거에 유효했던 이력정보에 대한 효율적인 관리기능을 데이터베이스 관리 시스템에서 제공해야 한다.

지난 15년간 시간지원 데이터베이스(temporal databases)에 대한 연구가 계속 수행되었으며[3,7,12], 시간지원 데이터베이스 모델링과 질의언어, 시간지원 데이터베이스 저장구조 및 색인기법을 중심으로 많은 국내외 학술대회[18,21,24]와 여러 학위 논문 및 서적[2] 등을 중심으로 발전해왔다. 또한 최근에는 시간지원 데이터베이스 관리 시스템 프로토타입의 구현을 중심으로하는 연구[3,10]들과 시간지원 데이터베이스 모델 및 질의 표준화에 대한 연구[12,18]등이 수행되고 있다. 그럼에도 불구하고 지금까지 연구된 데이터베이스 관리 시스템은 대부분 범용 시스템을 위주로 연구개발되고 있고 많은 결과물들이 나왔으나, 특수한 목적의 데이터베이스 관리 시스템의 설계 구현은 국내에서는 거의 없는 실정이다.

따라서 이 논문에서는 기본적인 데이터베이스 관리 시스템에 이력관리 기능을 중점을 두어 제공하는 특수목적용용인 시간지원 데이터베이스 관리 시스템의 질의처리 시스템을 설계 구현한다. 이러한 시간지원 질의처리 시스템은 사용자로부터 입력된 시간지원 질의에 대하여 시간지원 구문분석과 시간에 관한 의미분석을 수행하며 질의 최적화를 위한 시간지원 조인 연산을 수행한다. 이때 이 논문은 시간지원 이퀴조인 연산을 질의상에서 제공하고 이를 처리하기 위해 시간지원 중첩반복(temporal nested-loop) 알고리즘을 적용하였다. 아울러 기존의 count, sum, avg, max, min등의 집계함수에 시간개념을 지원하며, 추가로 earliest, latest, timemax, timemin등과 같은 시간지원 집계함수[19]를 지원토록 하였다. 또한 거래시간 로그 기법[24]을 이용하여 질의상에서 거래시간 연산을 지원하고 이를 처리토록 하였으며, 시간개념을 부과한 뷰의 제공 및 효율적인 처리방안을 구현하였다.

이 논문에서 구현된 시간지원 질의처리 시스템을 효율적으로 기술하기 위하여 다음과 같이 구성하였다. 2장에서는 지금까지 수행된 시간지원 데이터베이스 질의언어 및 질의처리 시스템에 관련된 전반적인 연구들을 정리하며, 3장에서는 시간지원 질의처리 시스템을 구성하는 각 단계에 관한 내용과 시간지원 조인, 시간지원 집계처리, 시간지원 뷰처리, 그리고 시간지원 조인 및 최적화 기법을 설명한다. 4장에서는 시간지원 질의처리 시스템의 구현과 시스템에 의한 질의처리 과정 및 수행예를 보이며, 마지막으로 5장에서는 이 연구의 요약 및 앞으로의 연구방향을 제안한다.

2. 관련 연구

지금까지 제안된 시간지원 데이터베이스 질의언어에는 대표적으로 HSQL[11], TempSQL[15], TQuel[14], TSQL[16], 그리고 TSQL2[12]등이 있으며 대부분 시간지원 데이터베이스 모델 및 시간지원 연산 및 시간지원 질의 구문의 정의에 중심을 두고 있다. 특히 TQuel은 기존의 관계형 데이터베이스 질의언어인 Quel(query language)[4]을 기반으로 튜플 타임스탬핑 방식의 이원시간 릴레이션을 지원하며 관계해석(relational calculus)을 통하여 시간속성과 연산에 대한 명확한 구문 및 의미를 갖는 장점을 갖는다.

아울러 최근에는 Postgres, TimeMultical, TimeDB, T-REQUIEM, ChronoLog 및 TimeIT등과 같은 시간지원 데이터베이스 구현에 관한 몇가지 연구들이 수행되었다[10]. 하지만 이들중 TimeIT, TimeMultical 등과 같은 일부 연구는 유효시간 또는 거래시간 하나만을 지원하는 프로토타입이기 때문에 완전한 이력을 제공하지 못하는 문제점을 가진다. 또한 T-REQUIEM과 같은 일부 연구는 제한된 시간지원 연산만을 지원하는 프로토타입이며, ChronoLog, TemCASE와 같은 연구는 기존의 상용 시스템에 시간지원 쉘을 추가한 프로토타입이므로 시간값을 임의의 속성값으로 연산하기 때문에 사용자 질의상에 이력표현이 어색한 문제점을 가질 뿐만 아니라 관리 및 성능 측면에서 문제점을 갖는다.

따라서 기존의 데이터베이스 기능을 수용하고 완전한 이력관리 기능을 제공하는 독자적인 프로토타입이 요구되며, 이 논문에서는 이전의 Ingres를 기반으로 실

제된 TQuel 질의언어를 지원하는 TempIS 프로토타입에서 제공하는 기능을 수용하고 부가적인 기능을 확장하여 시간지원 데이터베이스 관리 시스템 프로토타입을 구현하였다. 즉, 기본적인 시간지원 질의 처리 시스템[22]을 개선하여 거래시간 지원[24], 시간지원 집계지원[19], 시간지원 뷰 지원[23], 그리고 시간지원 조인 및 최적화 기능[20]등을 추가적으로 지원하도록 완선한 시간지원 질의처리 시스템을 구현하였다.

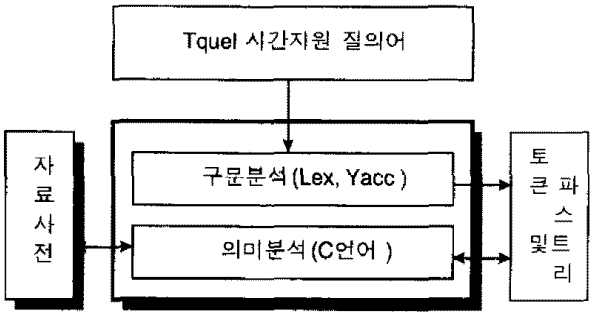
3. 시간지원 질의처리 시스템

이 논문에서 설명하는 시간지원 질의처리 시스템(temporal query processing system : TQPS)은 그림 1과 같이 시간지원 데이터베이스 관리 시스템을 구성하는 한 요소로서 상위층의 대화식 접속기(interactive interface) 혹은 그래픽 접속기(graphic interface)에 의해 생성된 TQuel 질의문을 받아 처리한 후 결과값을 반환하며, 하위층의 시간지원 트랜잭션 관리자와 시간지원 저장 관리기와 유지적인 상호작용을 수행한다. 이들 관리자들간의 연결은 UNIX 운영체제에서 제공하는 소켓(socket)연산을 사용하였다.

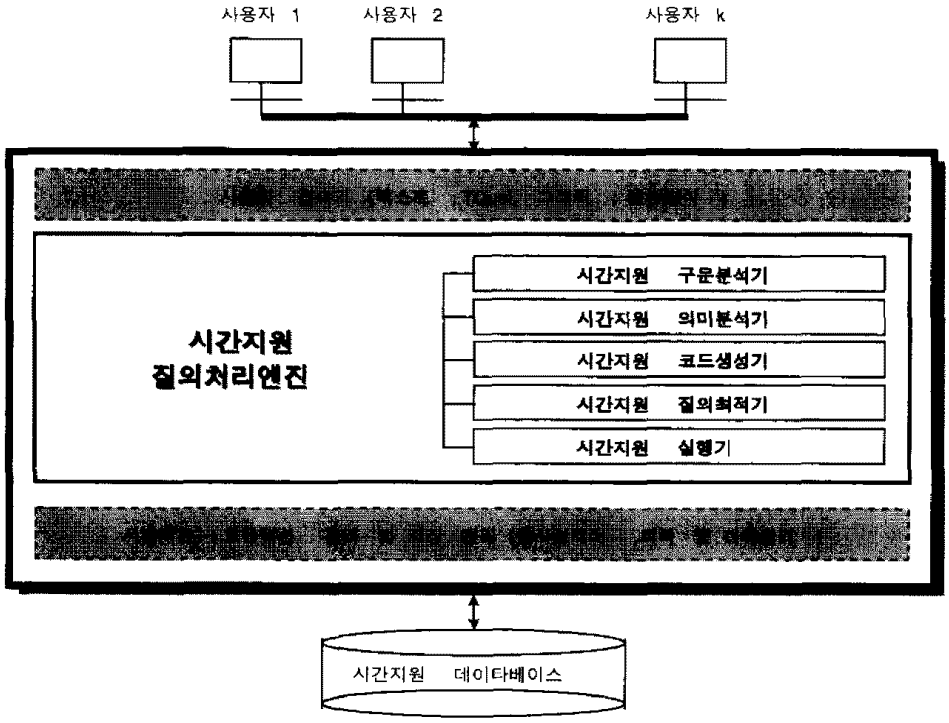
3.1 기본 구성 및 기능

시간지원 질의 처리 시스템은 그림 1과 같이 시간지원 구문 분석기(temporal syntax analyzer), 시간지원 의미 분석기(temporal semantic analyzer), 시간지원 코드 생성기(temporal code generator), 그리고 시간지원 실행기(temporal interpreter)와 같은 기본적인 요소[3]로 구성된다.

그림 2와 같이 시간지원 구문분석기는 Lex와 Yacc를 통해 작성되었으며, 사용자가 입력한 시간지원 질의어인 TQuel로부터 토큰 단위로 분해하고 파스 트리



(그림 2) 시간지원 구문분석 및 의미분석
(Fig. 2) Temporal Syntatic and Semantic Analyzer

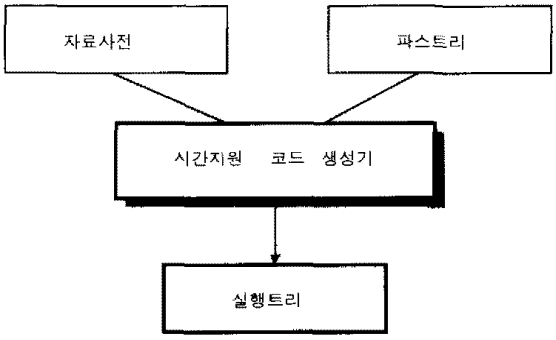


(그림 1) 시간지원 데이터베이스 관리 시스템 구조
(Fig. 1) Architecture of Temporal Database Management System

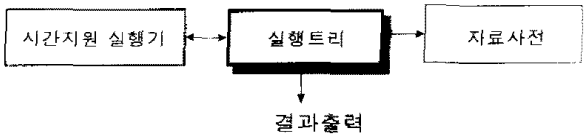
(parse tree)를 생성하여 구문의 정확성을 검사한다. 그리고 시간지원 의미분석기는 C언어로 작성되었으며, 시간지원 구문분석기에 의해 검토된 질의문에 대하여 스키마를 통해 지정된 릴레이션 이름, 릴레이션내 속성의 도메인 검사와 시간지원 조건자(temporal predicator) 및 생성자(constructor)로 구성된 시간지원 연산에 대한 타당성을 검사한다. 먼저 시간지원 조건자에는 overlap, precede, extend, equal 등의 연산자가 있으며 이들은 제시한 시간에 대한 참 또는 거짓을 출력으로 나타낸다. 아울러 시간지원 생성자는 하나 또는 두 개의 시간 간격을 입력받아 하나의 시간 간격을 반환하며, 단일 싯점을 반환하는 first, last연산자가 있고, 시간 간격을 반환하는 연산자로는 beginof, endof, overlap, extend, interval, event 등이 있다. 이때 overlap 연산자는 시간지원 조건자와 시간지원 생성자로서 모두 사용될 수 있는데 의미적으로 차이가 있다. 시간지원 생성자로서 overlap 연산은 두 시간 간격이 겹칠 경우 두 시간 간격이 겹치는 시간값을 반환하며, 시간지원 조건자로서 overlap 연산은 두 시간 간격이 겹치는 부분이 있을 경우 참값을 반환한다.

시간지원 코드 생성기(temporal code generator)는 그림 3과 같이 입력된 질의에 대한 실행 계획을 작성하는 모듈로 입력된 질의와 파스 트리로부터 시간지원 관계 대수(temporal realtional algebra)로 표현되는 실행 트리(execution tree)를 생성하는데, 실행 트리를 구성하는 노드에는 (1) range절에 기술된 릴레이션에 대한 검사를 수행하는 노드인 기저 노드(existing node), (2) 두 개 이상의 릴레이션에 대한 연산을 수행해야 할 경우 접근되는 카테시안 곱 노드(cartesian product node), (3) where절에 기술된 일반적인 조건을 검사하기 위해 접근되는 시간지원 선성 노드(temporal selection node), (4) valid절과 when절에서 기술된 시간조건과 시간 목적절 생성을 위해 접근되는 시간지원 유도 노드(temporal derivation node), (5) 출력될 속성을 추출하기 위해 접근되는 시간지원 추출 노드(temporal projection node), (6) 처리된 결과를 화면이나 릴레이션에 저장하기 위해 접근되는 시간지원 출력 노드(temporal display node) 또는 시간지원 저장 노드(temporal store node) 등이 있다.

시간지원 실행기는 C언어로 작성되었으며, 그림 4에서 처럼 시간지원 코드 생성기에 의해 만들어진 시간지원 실행 트리를 구성하는 각 노드에 대하여 초기화(initialize), 갱신(update), 그리고 전파(propagate) 단계를 루트 노드인 기저 노드로부터 터미널 노드인 시간지원 출력 노드 또는 시간지원 저장 노드에 이르기까지 재귀적으로(recursively) 수행한다.



(그림 3) 시간지원 코드 생성기
(Fig. 3) Temporal Code Generator



(그림 4) 시간지원 실행기
(Fig. 4) Temporal Interpreter

이상과 같은 기본적인 구조를 통해 이력관리를 위한 기본적인 기능을 제공하며, 이 논문에서는 부가적으로 거래시간 기능, 시간지원 집계 기능, 시간지원 조인 기능, 그리고 시간지원 뷰 기능을 지원하기 위하여 기본적인 구조를 확장하였으며, 코드 생성 단계에서 출력된 실행 트리를 대상으로 처리비용을 줄이기 위하여 시간지원 경험적 질의 최적화(temporal heuristic query optimizer)를 추가하였으며, 세부적인 추가 기능에 대하여 다음 절에서 차례대로 소개한다.

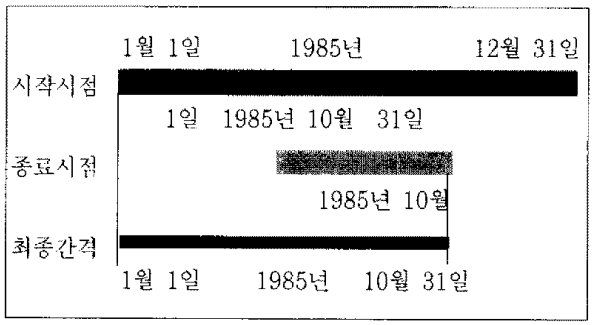
3.2 거래시간 지원

거래시간은 현실세계의 자료가 데이터베이스에서 처리된 시간을 의미하며 시스템에 의해 결정된다. 거래시간이 부여된 자료에 대한 삭제 연산은 논리적인 삭제만이 허용되며 물리적으로 결코 삭제되지 않으므로 완전한 이력을 제공한다. 또한 논리적으로 삭제된 이력자료에 대하여 오직 읽기 연산만이 허용되며, 기본적으로

현재 시점에서 이들을 참조할 수 없기 때문에 부가적인 처리과정을 필요로 한다.

거래시간에 대한 연산은 데이터베이스의 검색시점을 질의의 'as of' 절에 거래시간 조건으로 명시된 시간값에 해당하는 시간 만큼 과거로 되돌리는 연산으로서 거래시간 연산자(rollback operator)에 의해서 수행된다. 거래시간 연산자는 거래시간 로그 릴레이션을 대상으로 질의어의 'as of'절에서 묘사한 거래시간의 조건에 의해 정의된 시간 간격으로 시간 분할을 수행한다.

as of 절은 거래시간을 지원하는 절로서 거래시간 릴레이션이나 이원시간 릴레이션에 사용할 수 있으며, 시점과 간격을 묘사할 수 있다. 간격의 묘사는 'through'구를 이용한다. 예를들어 시간지원 질의내 as of절에서 "as of 1985 through Oct, 1985"와 같이 기술된 경우, 'as of'절이 실제 처리되어 갖는 시간값의 최종 간격은 시간 구성자를 이용하여 다음의 그림 5와 같이 결정된다.



(그림 5) As of절에 기술된 거래시간 간격의 범위 결정
(Fig. 5) Determine of Interval Range in As of Clause

(식 1)은 그림 5와 같이 시간지원 질의에서 사용된 'as of'절에 기술된 거래시간값으로부터 데이터베이스내 튜플에 대한 처리를 관계해석식으로 표현한 것으로서, l 은 대상이 되는 튜플 리스트를 의미하고 k는 검색문의 'range'구에서 묘사한 튜플의 갯수이다. Φ_α과 Φ_β는 시간값 $time_1$과 $time_2$에 대한 상수값을 의미한다.

$$(\forall l)(1 \le l \le k.(Before(\Phi_\alpha, tl[stop]) \wedge Before(tl[start], \Phi_\beta)) \quad \text{(식 1)}$$

이 논문에서는 거래시간 연산을 지원하기 위해 거래시간 로그 릴레이션(transaction time log relation) 기법[24]을 사용한다. 거래시간 로그 릴레이션은 기반 릴레이션의 새로운 상태를 발생하는 임의의 갱신 연산에 대한 세부적인 튜플의 정보를 저장하며, 거래시간 릴레이션이나 이원시간 릴레이션이 생성될 때 시스템에 의해 자동적으로 생성 및 관리되며 기반 릴레이션과 동일한 스키마 구조를 가진다. 거래시간 로그 릴레이션은 기반 릴레이션에 삭제 또는 갱신 연산이 적용되어 삭제 또는 갱신된 튜플의 정보를 보관하는 일종의 로그 릴레이션으로 로그 테이블로도 불리운다. 거래시간 로그 릴레이션에 대한 자료의 추가는 기반 릴레이션에서 삭제되는 자료가 발생할 때 시스템에 의해 자동적으로 이루어진다. 시스템은 삭제된 자료에 데이터를 부여하게 되는데, 이 데이터는 모든 자료들을 유일하게 구별할 수 있어 중복을 배제할 수 있는 숫자나 시간값을 이용할 수 있다. 시간값은 대표적으로 시스템 시간을 들 수 있고, 숫자로는 유일하게 증가되는 계수기를 이용해야 되고 이 숫자값을 필요에 따라서 그레고리안력으로 변경하는 처리루틴이 요구된다.

거래시간 로그 릴레이션을 이용한 거래시간 처리방안은 그림 6과 같은 거래시간 상태 검색 알고리즘(transaction time state finding algorithm)과 거래시간 튜플 검색 알고리즘(transaction time tuple finding algorithm)으로 구성된다.

그림 6(a)의 거래시간 상태 검색 알고리즘은 트랜잭션 식별자와 거래시간을 인자로 갖고, 트랜잭션 식별자 정보를 이용하여 거래시간이 지원되는 릴레이션인지의 여부와 그 트랜잭션이 현재 시간까지 완료된 트랜잭션 번호보다 작은 지의 여부를 점검하는 의미 점검 단계로서 결합상태가 옳으면 그 트랜잭션이 유효했던 상태를 반환하는데, 이때 롤백 릴레이션인 경우는 스냅 상태로 반환하고 이원시간 릴레이션일 경우는 이력 상태로 결정하여 해당 트랜잭션 식별자와 함께 반환한다. 그리고 그림 6(b)의 거래시간 튜플 검색 알고리즘은 의미상으로 맞는 경우에는 거래시간 조건 내에서 반환된 스냅 상태나 이력 상태에서 튜플을 찾는 과정이다. 그림 6(a)의 거래시간 상태 검색 알고리즘에서 반환된 트랜잭션 식별자와 이 트랜잭션 식별자와 결합된 해당 상태를 인자로 갖고 그 상태를 바탕으로 튜플의 생명주기를 거래시간 조건과 비교하여 검사함으로서 상태내의 튜플을 선정 또는 거부한다. 선정된 튜플들은 실행 트리의 후속 연산자 노드로 전파하기 위해 임시 릴레이션에 저장된다.

```
find_relation_state()
{
  if (릴레이션이 롤백 릴레이션이면) {
    foreachinSEQstate(릴레이션내의 상태 선정) {
      if (시간조건에 맞는 트랜잭션이 존재하면)
          해당 상태를 스냅 상태로 반환;
        else
        foreachinSEQstate(거래시간 로그 릴레이션내의 상태 선정) {
            if (시간조건에 맞는 트랜잭션이 존재하면)
              해당 상태를 스냅 상태로 반환;
        }
    }
  }
  else if (릴레이션이 이력시간 지원 릴레이션이면) {
    foreachinSEQstate(릴레이션내의 상태 선정) {
      if (시간조건에 맞는 트랜잭션이 존재하면)
          해당 상태를 이력 상태로 반환;
        else
          foreachinSEQstate(거래시간 로그 릴레이션내의 상태 선정) {
            if (시간조건에 맞는 트랜잭션이 존재하면)
              해당 상태를 스냅 상태로 반환;
          }
    }
  }
  else  오류 메시지 반환;
}
```

(a) 거래시간 상태 검색 알고리즘

(a) Transaction Time State Finding Algorithm

```
find_tuple()
{
   foreachinSEQtuple(선정된 상태내의 각 튜플 선정) {
            if (튜플의 거래시간이 조건시간내에 포함되면) {
                        임시 릴레이션으로 해당 튜플추가;
            }
   }
   if (lengthSEQrelation(임시 릴레이션) == 0) {
            해당 튜플 없음을 나타내는 메시지 반환;
   }
   else  임시 릴레이션을 후속 노드로 전파;
}
```

(b) 거래시간 튜플 검색 알고리즘

(b) Transaction Time Tuple Finding Algorithm

(그림 6) 거래시간 처리 알고리즘

(Fig. 6) Transaction Time Processing Algorithm

3.3 시간지원 조인 및 질의 최적화

시간지원 조인 연산(temporal join operation)은 시간값을 조인 조건으로 하여 두 개 이상의 릴레이션을 대상으로 수행하며, 기존의 데이터베이스에서 정의된 조인 연산이 동등 조건(equal predicate)에 기반을 둔 반면 시간지원 조인 연산은 시간값을 대상으로 주로 중첩(overlap), 이전(precede)등과 같이 비동등 조건(unequal predicate)에 기반을 두고 있어 처리비용이 크기 때문에 효율적인 조인 방안이 요구된다.

시간지원 조인 연산에는 유효시간 자연 조인(valid-time natual join)[8], 시간교차 조인(time intersection join)[1], 그리고 엔티티 조인(entity join 또는 event join)[1]등이 있으며, 시간지원 조인 처리기법에는 중첩-반복(nested loop) 알고리즘[1], 정렬후 병합(sort-merge) 알고리즘[17], 그리고 유효시간 분할 조인(validtime partition join) 기법[8]등이 있다. 이 논문에서는 유효시간 자연 조인과 유사한 시간지원 이퀴조인(temporal equijoin) 연산을 대상으로 시간지원 중첩-반복(temporal-nested loop) 알고리즘을 적용하였다.

먼저 시간지원 이퀴조인은 그림 7과 같이 대상 릴레이션에 존재하는 튜플들을 대상으로 조인 속성값이 일치하고 유효시간 간격이 서로 중첩(overlap)될 때 두 릴레이션에서 정의된 속성들과 중첩구간의 시간값을 갖는 새로운 튜플들을 구성요소로하는 새로운 릴레이션을 생성한다.

기존의 관계형 데이터베이스에서 적용된 중첩-반복 알고리즘을 확장한 시간지원 중첩-반복 알고리즘은 주어진 릴레이션이 주키 값과 유효시간 속성에 대하여 정렬되지 않은 경우에 적합하며, 이 논문에서는 이를 기반으로 구현하였다. 구현된 시간지원 중첩-반복 알고리즘은 그림 8과 같이 입력된 릴레이션들 중에서 내부 릴레이션과 외부 릴레이션으로 지정하며, 외부 릴레이션에 저장된 각각의 튜플들을 내부 릴레이션에 저장된 동일한 주키값과 유효 시간간격을 가진 전체 튜플들과 조인한다. 이러한 과정을 외부 릴레이션의 전체 튜플에 대하여 반복적으로 수행하기 때문에 처리비용이 큰 특징이 있다.

```
temporal equijoin(Atuple, Btuple, newtuple, inputnode)
{
   retrieve select node from inputnode;
   SelectNode = inputnode.VselectNode;
   evaluate_arithfunction(Atuple);
   evaluate_arithfunction(Btuple);
   if(strcmp(Atuple,Btuple) !=0)
      return (FALSE);
   DerivationNode=AdescendentInfo->descendent.VdevationNode;
   switch(typeof(temporalExpr) {
   case OVERLAP:
      evaluate_TemporalExpr(Atuple);
      evaluate_TemporalExpr(Btuple);
      if(overlap(Atuple,Btuple) == TRUE)) {
```

```
            appendrearSEQattributeValue(Atuple->attribute,
            newtuple);
            appendrearSEQattributeValue(Btuple->attribute,
            newtuple);
        }
        break;
    default: ERROR();
    }
}
```

(그림 7) 시간지원 이퀴조인 연산자 알고리즘
(Fig. 7) Temporal Equi-Join Operator Algorithm

```
릴레이션 r1과 r2의 튜플 x, y를 읽음;
do {
  if( x[조인속성]=y[조인속성] and
    x[유효시간속성]=y[유효시간속성])
    결과 튜플을 출력함;
  if(EOF( r2)) {
    if(EOF( r1)
      종료함;
    else
      다음 튜플 x를 읽음;
      r2의 첫번째 튜플을 읽음;
  }
  else
    다음 튜플 y를 읽음;
} while(1);
```

(그림 8) 시간지원 중첩-반복 알고리즘
(Fig. 8) Temporal Nested-Loop Algorithm

시간지원 질의 최적화(temporal query optimizer)는 시간지원 코드 생성기와 시간지원 실행기 사이에 위치하며, 시간지원 코드 생성기가 입력된 사용자 질의에 대하여 생성한 실행 트리에 대하여 시간지원 연산 조건과 카탈로그(catalog)로부터 추출된 릴레이션에 관한 정보를 기반으로 그림 9와 같이 시간지원 의미 최적화(temporal semantic optimizer)와 시간지원 휴리스틱 최적화(temporal heuristic optimizer) 과정을 통해 최적화를 수행한다.

시간지원 의미 최적화는 카탈로그로부터 수행될 시간지원 릴레이션들에 대하여 튜플수와 속성수등과 같은 정보를 추출하여 실행 순서에 따른 비용을 측정하며, 만일 실행 트리상의 노드에 지정된 시간지원 조건 연산이 대상 릴레이션이 갖는 시간 영역을 벗어나는 경우 실행치 않도록 해 준다. 그리고 시간지원 휴리스틱 최적화는 두 개 이상의 릴레이션을 대상으로 수행될 경우, 기본 적으로 실정된 시간지원 교차곱 연산을 대신하여 시간지원 조인 연산을 수행하며, 실행 트리상의 각 노드의 위치에 대하여 시간지원 조인 연산 보다 시간지원 선정 연산 또는 시간지원 추출 연산등을 먼저 수행하도록 재조정하는 기능을 갖는다.

```
TQuelOptimizer(inputroots, inputcatalog)
roots                inputroots;
catalog              inputcatalog;
{
  /* Algorithms */
  Acatalog = inputcatalog;
  currentRoots = inputroots;
  Theroots = currentRoots;
  if( TemporalSemanticRule(Acatalog, Theroots) != -1 ) {
    Aroots = currentRoots;
    AQEPcost = TemporalQEPCost(Acatalog, Aroots);
  }
  else {
    return -1;
  }
  TemporalHuristicRule(Acatalog, Theroots);
  TheQEPcost = TemporalQEPCost(Acatalog, Theroots);
  if (AQEPcost <= TheQEPcost)
    currentRoots = Aroots;
  else
    currentRoots = Theroots;
}
```

(그림 9) 시간지원 질의 최적화 알고리즘
(Fig. 9) Temporal Query Optimizer Algorithm

3.4 시간지원 집계 처리

시간지원 집계(temporal aggregates)는 시간의 변이에 따르는 자료에 대한 통계적 수치값을 계산하며, 일반적인 질의로 표현하기에는 복잡한 기능을 하나의 함수로서 계산될 수 있도록 해준다[19]. 이러한 시간지원 집계는 기본적인 집계함수인 count, any, sum, avg, min, max등에 시간개념을 추가하여 확장한 것과, 새로이 정의된 집계함수인 first, last, earliest, latest, stdev, avgti, varts, timemin, timemax등이 있다. avgti(average time increment)함수는 주가변화(포인트/시간), 교통량변화(자동차대수/시간)와 같이 시간당 속성값의 변화율을 반환하며, varts(variability time space)함수는 속성값이 변화하는 시간간격의 폭을 반환한다. 즉, varts함수가 반환하는 값에 대하여 0은 일정하게 변화함을 의미하고, 양의 값은 느

리게 변화함을 의미하고, 튜의 값은 빠르게 변화함을 의미한다.

시간지원 집계함수를 처리할 때, 릴레이션내에서 튜플들이 시간속성 값에 의해 정렬되었는가의 여부는 수행 효율에 큰 영향을 준다. 시간 속성에 의해 이미 정렬된 경우는 색인을 이용하여 디스크의 일부만을 대상으로 처리가 수행되기 때문에 비교적 간단하지만, 튜플이 시간에 의해 정렬화가 안된 경우는 릴레이션 전체를 여러 번 검색해야하는 문제점을 갖는다. 시간지원 집계 트리 전략(temporal aggregate tree strategy)[19]은 시간 속성에 의해 정렬화가 안된 릴레이션을 대상으로 시간지원 집계처리시 사용되는 알고리즘으로 그림 10과 같이 3단계로 구성된다. 첫 번째 단계에서는 주어진 릴레이션에 대하여 고정간격집합(constant interval set)을 구하여 이를 이용해서 이진 트리(binary tree)를 생성한다. 이때 사용자가 분할 속성을 지정한 경우, 복수개의 이진 트리가 생성되는데, CREATEaggregateTree()함수가 이진 트리를 생성한다. 두 번째 단계에서는 이전 단계에서 생성된 트리를 이용하여 CREATEtemporaryRelation()함수를 통하여 임시 릴레이션을 생성한다. 복수개의 집계함수가 사용된 경우 복수개의 임시 릴레이션이 생성된다. 마지막으로 세 번째 단계에서는 임시 릴레이션과 출력될 목표 릴레이션과 이퀴조인을 함으로써 최종적인 출력 릴레이션을 생성한다. COMPUTEaggregate()함수는 사용자가 지정한 집계함수가 두 번째 단계에서 생성된 임시 릴레이션을 대상으로 고유한 연산을 수행한다.

```
CREATEaggregateTree()
{
   TaggrOpType =
      typeof(AGGREGATEinfo[aggrTreeindex]->syn operator);
   TbyforwhereArgument =
      AGGREGATEinfo[aggrTreeindex]->byForRetrieveArg;
   TaggregateArgument =
      AGGREGATEinfo[aggrTreeindex]->syn argument;
   if(typeof(TbyforwhereArgument->syn ByClause) != KTvoid)
      CREATEaggrFuncTree();
   else
      CREATEaggrScalarTree();
}
CREATEtemporaryRelation()
{
   TbyforwhereArgument =
      AGGREGATEinfo[aggrTreeindex-1]->byForRetrieveArg;
   if(typeof(TbyforwhereArgument->syn ByClause) != KTvoid)
      CREATEaggrFuncRelation();
   else
      CREATEaggrScalarRelation();
}
```

```
COMPUTEaggregate(Atuple,  Avalue1,  AvalueF,
aggregateTime)
{
   TbyforwhereArgument =
      AGGREGATEinfo[aggrTreeindex-1]->byForRetrieveArg;
   if(typeof(TbyforwhereArgument->syn ByClause) != KTvoid)
      COMPUTEaggregateFunction(Atuple,  Avalue1,  AvalueF,
      aggregateTime);
   else
      COMPUTEscalarAggregate(Atuple,  Avalue1,  AvalueF,
      aggregateTime);
}
```

(그림 10) 시간지원 집계 트리 알고리즘
(Fig. 10) Temporal Aggregate Tree Algorithm

3.5 시간지원 뷰 처리

이 논문에서는 시간지원 데이터베이스에서 증진에 의한 뷰 형성(incremental view materialize)기법을 통해 뷰처리 알고리즘[24]을 그림 11에서와 같이 구현하였다. 형성 뷰 기법을 적용하는 경우는 데이터베이스의 규모가 급격히 증가하므로, 뷰에 대한 갱신은 뷰에 대한 데이터베이스 릴레이션의 변경이 있을 경우 이 변경된 내용만을 뷰에 전파하는 증진 뷰(incremental view) 기법으로 처리한다.

일반적으로 뷰에 관한 질의의 수가 베이스 릴레이션에 대한 질의의 수보다 많은 경우, 베이스 릴레이션의 크기가 클 경우, 뷰를 구성에 대한 조건이 적을 경우, 뷰에 대한 변경이 적을 경우, 그리고 질의의 대부분이 검색 연산일 경우에 대하여 형성 뷰 기법을 사용한다. 시간지원 데이터베이스는 삭제질의에 대하여 물리적인 튜플의 삭제를 수행하지 않고 논리적인 삭제만을 제공하는 비삭제 정책(non-deletion policy)에 의해 릴레이션의 크기가 시간이 경과함에 따라 계속 증가한다. 그러므로, 시간지원 질의 처리 시스템에서는 형성 뷰 기법으로 뷰를 지원하며, 뷰에 대한 베이스 릴레이션에 대한 내용에 변화를 뷰에 전파하는 기법은 변경된 정보만을 뷰에 전파하고, 기저 릴레이션에 데이터의 추가, 삭제, 변경이 있을 경우 변경된 정보만을 저장된 뷰의 실행 트리를 이용하여 즉시 뷰에 전파하는 증진 뷰 기법을 사용한다.

뷰 생성에 대한 스키마 구조는 시간지원 의미 분석기에서 결정되는데, 뷰에 대한 베이스 릴레이션의 형에 따라 뷰의 성격이 결정된다. 시간지원 코드 생성기는 뷰 정의문에 정의한 시간 조건과 일반 조건을 검토하여 뷰에 대한 실행 트리를 생성하여 뷰의 형이 이력 릴레이션 또는 스냅 릴레이션인 경우 뷰에 대한 실행 트리

을 저장한다. 아울러 시간지원 실행기에서는 실행 트리를 수행하여 뷰를 생성하게 된다.

```
int itq(currentTQuelStatement,currentdatabase, currentview_tree,
quitprogram)
{
    . . .
    case Kdefine_viewStmt:
        codegeneration(&currentStatement,v_execution_tree,
        &v_Nodeid,&v_thetupleVars);
            . . .
        interpreter(newcommand,v_execution_tree,  currentdatabase,
        currentStatement);
            . . .
            addSETview_info(((*currentdatabase)->view_infos),
                                temp_view);
        if(!flag)
        {
            a_view_tree = Nview_def_tree;
            a_view_tree->view_name =
            currentStatement.Vdefine_viewStmt->
            syn_relation->sem_entity->name;
            a_view_tree->update_tree = v_execution_tree;
            a_view_tree->view_query = currentStatement;
            addSETview_def_tree(((*currentview_tree)->
            view_tree), a_view_tree);
        }
    . . .
}
```

(그림 11) 시간지원 뷰 처리 알고리즘
(Fig. 11) Temporal View Processing Algorithm

4. 구현 및 수행 예

4.1 구현

이 논문에서 기술한 시간지원 질의처리 시스템은 SUN 워크스테이션에서 UNIX 운영체제를 기반으로 사용자 접속 언어(IDL)[13] 개발도구를 사용하여 자료구조를 정의하였으며, Lex와 Yacc 및 C언어를 이용하여 구현되었다. 시간지원 질의처리 시스템의 규모는 IDL 관련 생성 소스 프로그램을 제외하고 대략 30,000라인의 소스 프로그램과 1.5MB의 실행 파일 크기를 갖는 주기억장치 프로토타입이다. 즉, 데이터베이스와 연산자를 주기억장치에 적재한 후 연산이 이루어지고, 연산종료 후 사용자 접속 언어에서 제공하는 외부 파일 저장방법을 통해 일괄 저장하는 방식을 사용하였다. 또한 시간지원 질의 처리 시스템의 상위에 존재하는 시간지원 사용자 접속기 상의 메뉴 상에서 기술된 조건들을 대상으로 시간지원 질의가 생성되면, 이 논문에서 설명하는 시간지원 질의처리 시스템으로 입력되어 실제적인 처리가 이루어진다.

4.2 시간지원 질의 수행예

이 절에서는 구현된 시간지원 질의처리 시스템의 실행과정에 대하여 대표적인 경우로서 시간지원 이퀴죠인 연산을 포함하는 검색질의의 처리과정 및 수행결과를 설명한다. 거래시간 연산, 시간지원 집계, 시간지원 뷰와 같은 다른 확장 기능도 이와 유사한 과정을 통하여 실행후 출력된다. 그림 12(a)에 주어진 "학부 릴레이션(이름, 직급)"과 "봉급 릴레이션(이름, 봉급)"에 대하여 그림 12(b)와 같이 각 직원에 대한 이력정보를 출력하는 질의에 대한 처리과정 및 결과는 그림 12(c) 및 그림 12(d)와 같다. 그림 12(c)에서 시간지원 질의 최적화는 시간지원 코드 생성기가 생성한 실행 트리에 대하여 첫째, 시간지원 조인 조건과 해당 릴레이션들에 대한 스키마 정보를 검사하여 실행여부를 결정하고, 둘째, 시간지원 카테시안 곱 노드를 시간지원 이퀴조인 노드로 변경하며, 셋째, 시간지원 선정 노드와 추출 노드를 시간지원 조인 노드보다 우선적으로 실행되도록 실행 트리의 구조를 변경한다. 그림 12(c)에서 최적화가 이루어진 실행 트리는 시간지원 실행기로 전달되며, 실행 트리상의 각 노드에 해당하는 모듈이 동적으로 바인딩됨으로써 실제적인 연산이 이루어진다.

4.3 평가

이 논문에서 추가로 구현된 시간지원 질의처리 시스템은 거래시간 처리, 시간지원 조인 및 질의 최적화, 시간지원 집계와 시간지원 뷰 기능을 지원한다. 이들 기능을 통하여 기존의 시간지원 질의처리 시스템에 비하여 다음과 같은 성능 향상을 기대할 수 있다.

첫째, 유효시간 뿐만 아니라 거래시간에 대한 완전한 연산을 지원한다. 데이터베이스에서 처리된 시점을 기준으로 과거에 삭제된 자료를 검색할 수 있을 뿐만 아니라 완전한 이력을 보장할 수 있다.

둘째, 데이터베이스에서 빈번하게 발생되는 두 개 이상의 릴레이션에 대한 연산에 대하여 기존의 질의처리 시스템에서의 시간지원 카테시안 곱 연산을 시간지원 조인 연산으로 대치함으로써 질의처리 비용을 감소시킬 수 있다. 일례로 그림 12의 릴레이션과 입력 질의에서 기존의 카테시안 곱 연산을 사용하는 시간지원 질의처

봉급 릴레이션

이름	봉급	유효시간		거래시간	
		시작	종료	시작	종료
홍길동	100,000	1970.3	1971.9	1971.9	UC
홍길동	150,000	1971.9	1973.3	1973.3	UC
홍길동	200,000	1974.3	1975.3	1975.3	UC
홍길동	350,000	1975.3	forever	1975.3	UC

학부 릴레이션

이름	직급	유효시간		거래시간	
		시작	종료	시작	종료
홍길동	전임강사	1970.3	1973.3	1973.3	UC
홍길동	전임강사	1974.3	1975.3	1975.2	UC
홍길동	조교수	1975.3	forever	1975.2	UC

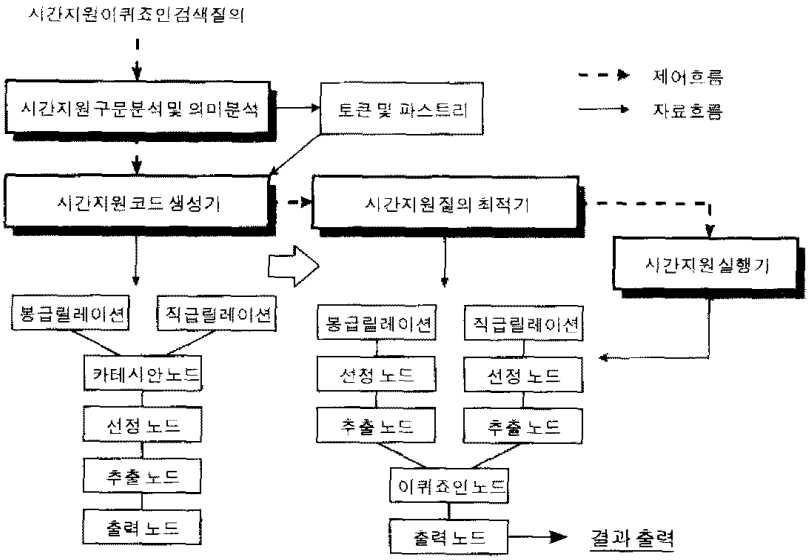
(a) 시간지원 릴레이션

(a) Temporal Relation

```
range of r is 학부
range of s is 봉급
retrieve(이름=r.이름, 직급=r.직급, 봉급=s.봉급)
        valid from begin of r to end of r
        where r.이름 = s.이름
        when r overlap s
```

(b) 시간지원 이퀴조인 질의

(b) Temporal Equi-Join Query



(c) 시간지원 질의처리 과정

(c) Temporal Query Processing

이름	직급	봉급	유효시간	
			시작시간	종료시간
홍길동	전임강사	100,000	1970.3	1971.9
홍길동	전임강사	150,000	1971.9	1973.3
홍길동	전임강사	200,000	1974.3	1975.3
홍길동	조교수	350,000	1975.3	forever

(d) 학부 TEq-join 봉급 수행결과

(d) Temporal Equi-Join Result between Factlty and Salary Realtion

(그림 12) 시간지원 이퀴조인 질의와 수행결과

(Fig. 12) Temporal Equi-Join Query and Its Result

의 방법은 4개의 속성과 12개의 튜플로 구성된 최종 결과를 출력하지만, 이 논문에서 제안된 시간지원 조인 방안을 사용하면 그림 12(d)와 같이 속성하나와 여덟 개의 불필요한 튜플을 제거할 수 있다. 극단적인 경우 시간 연산에 의해 질의 수행후 결과가 없는 경우도 존재하는데, 이때 이 논문에서 제안한 시간지원 조인 및 최적화 알고리즘은 질의와 스키마 정보를 통해서 실제적인 연산을 수행하지 않고도 질의 결과를 예측할 수 있는 장점을 갖는다.

셋째, 시간지원 뷰 기능과 집계기능의 지원으로 시간의 흐름에 따른 이력자료에 대한 보다 다양한 유형의 정보를 제공할 수 있다. 먼저 시간지원 뷰 기능을 통해 사용자에게 요구되는 개별적인 환경을 물리적인 릴레이션을 대신하여 쉽게 처리할 수 있도록 해주며, 데이터베이스에 변경이 발생하면 그 차분만을 자동적으로 이름 뷰에 전파하여 시간지원 데이터베이스의 특성에 적합토록 하였다. 또한 시간지원 집계 기능을 통해 시간의 흐름에 따른 자료에 대하여 이력 통계를 제공할 수 있다.

5. 결 론

시간지원 데이터베이스 시스템은 시간의 흐름에 따라 변경되는 정보를 효율적으로 취급할 수 있으며, 이력 정보를 취급해야하는 다양한 분야에서 사용될 수 있다. 이 논문에서 구현된 시간지원 질의 처리 시스템은 시간지원 구문분석기, 시간지원 의미분석기, 시간지원 코드생성기, 시간지원 질의최적화와 시간지원 실행기로 구성되도록 하였으며, 거래시간 처리, 시간지원 조인, 시간지원 집계와 시간지원 뷰 기능을 지원하도록 하였다.

거래시간지원은 거래시간 지원 로그 릴레이션 기법을 이용하는데 거래시간을 지원하기 위하여 이 기법은 거래시간 릴레이션 상태검색 알고리즘과 거래시간 튜플 검색 알고리즘으로 구성하였다. 시간지원 질의 최적화는 일반적으로 사용하는 경험적 질의 최적화와 시간지원 중첩-반복 알고리즘을 이용한 시간지원 이퀴조인을 사용하였다. 또한 기존의 집계함수 기능에 시간지원 집계 기능을 추가하였으며, 시간지원 뷰처리에서 시간지원 데이터베이스의 비-삭제방안과 같은 특성을 고려하여 증진 뷰 형성 기법을 사용하였다.

구현된 시간지원 질의처리 시스템은 이력을 관리하는 병원 이력관리 시스템, 데이터웨어 하우스에서 OLAP의 과거 정보 통계 제공등과 같은 이력관리를 필요로 하는 시스템에 활용될 수 있다. 앞으로 기존의 관계형 데이터베이스 기술을 기반으로 하는 시간지원 접근방안에 대한 문제점을 최소화하는 저장구조와 이를 고려한 시간지원 광역 질의 최적화에 대한 연구 및 기존 데이터베이스 관리 시스템의 일부 기능지원에 따른 성능평가 문제가 연구되어야 한다.

참 고 문 헌

[1] A. Segev, and H. Gunadhi, "Event-join Optimization in Temporal Relational Databases", Proceedings of 15th International Conference on Very Large Data Bases, 1989.

[2] A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass, "Temporal Databases : Theory, Design, and Implementation", The Benjamin/Cummings Publishing Company Inc., 1993.

[3] D. H. Kim, K. H. Jeon, K. J. Jeong, K. J. Kim, and K. H. Ryu, "A Temporal Database Management Main Memory Prototype", IEEE Region 10's Ninth Annual International Conference, Aug., 1994.

[4] G. Held, M. Stonebraker, and E. Wong, "INGRES-A Relational Database Management System", Proceedings of the AFIPS National Computer Conference, Vol.44, May, 1975.

[5] J. Allen, "Maintaining Knowledge about Temporal Intervals.", Communication of the Association of Computing Machinery, Vol. 26, No.11, Nov., 1983.

[6] M. Jarke and J. Koch, "Query Optimization in Database Systems, Computing Surveys", Vol.16, No.2, Jun., 1984.

[7] M. Soo, "Bibliography on Temporal Databases", ACM SIGMOD Record, Vol.20, No.1, Mar., 1991.

[8] M. Soo, R. Snodgrass, and C. Jensen, "Efficient Evaluation of the Valid-Time Natural Join", Technical Report TR93-17, University of Arizona, Jun., 1993.

[9] M. Stonebraker, "Implementation of Integrity Constraints and Views by Query Modification", Proceeding of ACM SIGMOD International Conference on Management of Data, Jun, 1975.

[10] M. Bohlen, "Temporal Database System Implementation", ACM SIGMOD Record, Vol. 24, No.4, Dec., 1994.

[11] N. Sarda, "Extensionsto SQL for Historical Databases", IEEE Transactions on Knowledge and Data Engineering, Vol.,2, No.2, Jul., 1990.

[12] R. Snodgrass ED., "The TSQL2 Temporal Query Language", Kluwer Academic Publishers, 1995.

[13] R. Snodgrass, "The Interface Description Language : Definition and Use", Computer Science Press, 1989.

[14] R. Snodgrass, "The Temporal Query Language TQuel", ACM TODS, Vol.12, No.2, Jun. 1987.

[15] S. Gadia, and J. Vaishnav, "A Query Language for a Homogeneous Temporal Database", In Proceedings of the ACM Symposium on Principles of Database Systems, Mar., 1985.

[16] S. Navathe, and R. Ahmed, "TSQL-A Language Interface for History Database", In Proceedings of the Conference on Temporal Aspects in Information Systems, May., 1987.

[17] T. Leung, and R. Muntz, "Query Processing for Temporal Databases", Proceedings of the Sixth International Conference on Data Engineering, Feb., 1990.

[18] 김동호, 류근호, "시간지원 데이터베이스 용어의 표준안", 한국정보과학회 논문집, 21권, 1호, 1994년 4월.

[19] 김동호, 이인홍, 류근호, "주기억장치에서 시간지원 데이터베이스의 집계 함수 설계 및 구현", 한국정보과학회 논문지, 21권, 8호, 1994년 8월.

[20] 김동호, 정경자, 류근호, "시간지원 Equi-join 연산의 구현", 한국정보과학회 논문집, 21권, 2호, 1994년 10월.

[21] 김동호, 정경자, 류근호, "시간지원 데이터베이스에서 시간 및 데이터 생신전략", 한국정보처리학회 논문집, 1권, 2호, 1994년 10월.

[22] 김동호, 정경자, 전근환, 김기중, 류근호, "시간지원 데이터 관리 시험대", 한국정보처리학회 논문지, 1권 1호, 1994년.

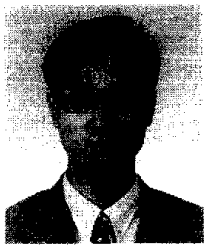
[23] 류근호, "시간지원 데이터베이스에서 뷰 형성 유지를 위한 실행트리", 한국정보과학회 논문지, 제20권, 제8호, 1993년 8월.

[24] 정경자, 전근환, 류근호, "시간지원 데이터베이스에서 거래시간 지원을 위한 거래시간 로그와 연산자의 설계 및 구현", 한국정보과학회 논문지, 제22권, 제6호, 1995년 6월.

[25] 정경자, 김동호, 이언배, 류근호, "시간지원 데이터베이스의 질의처리", 데이터베이스 연구회 지, 제12권, 제3호, 한국정보과학회 데이터베이스연구회, 1996년 8월.

이 언 배

1974년 서울 시립대학교 졸업
1982년 동국대학교 경영대학원 전산전공(석사)
1992년~1998년 8월 충북대학교 대학원 전자계산학과(박사) 예정
1974년~1983년 총무처 정부 전자계산소 근무
1983년~현재 한국방송통신대학교 전자계산학과 교수 재직
관심분야 : 데이터베이스, 소프트웨어공학 등



김 동 호

1993년 충북대학교 전자계산학과 졸업
1995년 충북대학교 대학원 전자계산학과 졸업(이학석사)
1995년~현재 충북대학교 대학원 전자계산학과 박사과정 (수료)
관심분야 : 시간지원 데이터베이스, 시공간 데이터베이스, 멀티미디어 시스템, 객체지향 데이터베이스 등

류 근 호

1976년 숭실대학교 전산학과 졸업

1980년 연세대학교 산업대학원 전산전공(공학석사)

1988년 연세대학교 대학원 전산전공(공학박사)

1976년~1986년 육군 군수 지원사 전산실(ROTC장교), 한국전자통신연구소(연구원), 한국방송통신대 전산학과(조교수)

1989년~1991년 Univ. of Arizona Research Staff

1986년~현재 충북대학교 컴퓨터과학과 교수겸 정보통신 연구소 소장

관심분야 : 시간지원 데이터베이스, 시공간 데이터베이스, DBMS와 정보검색, 객체 및 지식베이스 시스템 등