시간지원 집계 질의 처리의 성능 평가

이 종 연† · 김 동 호† · 이 인 홍†† · 류 근 호†††

요 약

시간지원 데이터베이스는 현실세계에 존재하는 객체에 대한 효율적인 이력 표현과 연산을 지원한다. 특히 시간지원 집계는 현재시점에서 유효한 자료뿐만 아니라 이력 자료로부터 일련의 계산을 통해 부가적인 정보를 생성해내는 기능을 갖는다. 따라서 다양한 유형의 집계함수의 지원은 전반적인 데이터베이스 시스템의 편리함과 성능을 결정하는 중요한 요소가 된다. 이 논문에서는 시간지원 집계 질의에 대한 효율적인 처리 방안으로서 시간지원 집계 트리 전략을 소개하고, 그 처리 비용을 분석하고 성능을 평가한다.

A Performance Evaluation of Temporal Aggregate Query Processing

Jong Yun Lee† · Dong Ho Kim† · In Hong Lee†† · Keun Ho Ryu†††

ABSTRACT

Temporal databases support an efficient historical representation and operation for an object in the real world. Especially, temporal aggregates generate an additional information by means of computations from objects that is valid at past as well as current time. It is one of important areas to serve to users as various type of aggregates as possible so that they enhance the overall system performance and efficiency. In this paper, we not only introduce temporal aggregate tree strategy as an efficient processing technique for given temporal aggregate query, but also analyze the overall processing cost and then evaluate its performance.

1. 서 론

기존의 데이터베이스는 현실세계에서 존재하는 객체에 발생된 사건으로부터 오직 현재시점에서 유효한 상태만을 관리한다. 이로 인해 시간의 흐름에 따라 객체에 대한 갱신 연산을 통한 자료값의 변경은 이전에 저장된 값에 새로운 값을 겹쳐 쓰게되므로 자료에 대한 이력을 제공하지 못하는 문제점을 갖는다.

이러한 문제점을 해결하기 위한 방안으로 제안된 시간지원 데이터베이스(temporal databases)[2,9,10, 12]는 객체에 대한 효율적인 이력 표현과 연산을 지원하며 지금까지 다양한 유형의 시간지원 데이터 모델, 시간지원 질의언어와 뷰 처리, 시간지원 저장구조 및 색인방안 등이 연구되었으며, 최근에는 시간지원 데이터베이스 관리 시스템의 구현에 관련된 왕성한 연구들이 발표되고 있다.

일반적으로 집계함수(aggregate function)[3]는 일반적인 데이터베이스 연산과 달리 저장된 자료로부터 일련의 계산을 통해 부가적인 정보를 생성해내는 기능

※ 이 연구는 한국과학재단 핵심전문연구(과제번호:951-0908-007-02) 및 정보통신부 대학기초연구지원사업에 의하여 수행되었음.
† 준 회 원 : 충북대학교 컴퓨터과학과
†† 종신회원 : 전주대학교 컴퓨터공학과 교수
††† 종신회원 : 충북대학교 컴퓨터과학과 교수
논문접수 : 1998년 2월 23일, 심사완료 : 1998년 6월 1일

을 갖는다. 따라서 다양한 유형의 집계함수의 지원은 전반적인 데이터베이스 시스템의 편리함과 성능을 결정하는 중요한 요소가 된다. 즉, 일반 질의로 복잡하게 표현되어야 하는 연산 과정을 시스템 내부에서 함수로서 정의하여 질의의 요소로서 지원함으로써 사용자에게 질의 기술의 간략화와 불필요한 연산을 제거함으로써 최적의 실행을 지원하는 등의 장점을 갖는다. 하지만 집계함수는 복잡한 연산과정과 방대한 자료량으로 인해 처리 비용이 매우 크며, 특히 시간지원 데이터베이스와 같이 현재시점의 자료 뿐만 아니라 이력 자료를 포함하는 집계연산은 더욱 비용이 많이 들기 때문에 효율적인 처리방안이 요구된다.

지금까지 수행된 시간지원 데이터베이스 연구에서 다른 세부분야에 비해 시간지원 집계연구는 상대적으로 미비한 상태로서 대표적인 연구로서 새로운 시간지원 집계함수의 제안[13]과 시간지원 집계질의 처리방안[5,7,13]등이 수행되었다. 일반적으로 시스템에 대한 신뢰도를 좌우하는 한 요소로서 성능 평가는 매우 중요한 역할을 가지는데, 지금까지 시간지원 조인 및 색인과 같은 주요 기능에 대한 이론적인 성능 분석이 수행되었지만 처리 비용이 다른 요소보다 상대적으로 많이 요구되는 시간지원 집계처리 방안에 대한 분석 연구는 아직까지 수행되지 못하고 있다. 따라서 이 논문에서는 시간지원 데이터베이스에서 집계함수가 포함된 질의에 대한 효율적인 처리방안으로서 시간지원 집계질의 처리 전략을 소개하고, 그 처리 방안의 성능을 분석 및 평가한다.

이 논문의 효율적인 전개를 위하여 다음과 같이 구성하였다. 먼저 2장에서는 관련연구로서 시간지원 데이터베이스에서 새로이 정의된 집계함수와 시간지원 집계함수를 포함하고 있는 질의처리 방안으로서 시간지원 집계 트리 전략에 대하여 설명한다. 3장에서는 시간지원 집계 트리 전략에 대한 성능을 메모리 요구량을 기준으로 분석하며, 4장에서는 질의상에 존재하는 분할속성 여부에 따른 시간지원 집계 트리 전략의 성능을 평가한다. 마지막으로 5장에서는 이 연구의 결론 및 앞으로의 연구방향을 제안한다.

2. 관련 연구

시간지원 집계함수는 기존의 관계형 데이터베이스에서 정의된 COUNT, SUM, AVG, MAX, MIN등의 기본적인 집계함수외에 부가적으로 TIMEFIRST, TIMELAST, TIMEMAX, TIMEMIN, AVGTI, VARTS등의 새로운 집계함수를 갖는다[7,13]. 이들 새로운 집계함수는 모두가 시간속성을 대상으로 연산이 이루어진다. TIMEFIRST함수는 주어진 테이블에서 가장 오래된 튜플의 유효시간을 선정하며, TIMELAST 함수는 주어진 테이블에서 가장 최근 튜플의 유효시간을 선정한다. TIMEMIN함수는 가장 짧은 유효시간 간격을 가지는 튜플의 시간간격을 반환하고, TIMEMAX 함수는 가장 긴 유효시간 간격을 가지는 튜플의 시간간격을 반환한다. 그리고 AVGTI함수는 주어진 속성값의 유효시간 간격의 평균변화율을 계산하며, VARTS함수는 주어진 시간단위에 대한 속성값의 변화에 대한 표준편차값을 반환한다.

시간지원 데이터베이스에서 집계함수를 처리할 때, 테이블내에서 튜플들이 시간속성 값에 의하여 정렬되었는가의 여부는 실행 효율에 큰 영향을 준다. 만일 튜플들이 시간속성에 대하여 이미 정렬 되어있다면, 색인을 사용하여 디스크의 일부만을 실행하므로 디스크 연산을 최소화 할 수 있다. 하지만 대부분의 경우는 튜플이 임의의 순서를 가지며, 이때는 지정된 테이블들에 대하여 이력 자료까지 포함하는 데이터베이스 전체를 대상으로 검색하므로 시스템의 처리 효율을 저하 시킬 수 있다.

이를 효율적으로 처리하기 위해 제안된 시간지원 집계 트리 전략(temporal aggregate tree strategy)[5,6]은 삼단계로 구성되는데, 먼저 주어진 테이블을 읽어들여 이진 트리를 생성하고, 이를 토대로 임시 테이블을 생성하여 지정된 집계 연산을 계산후 반환한다. 시간지원 집계 트리 전략에서의 세부적인 집계 질의 처리 절차는 아래와 같다. 첫째, 고정 간격 집합(constant interval set)을 계산한다. 고정 간격 집합은 각 테이블내에 저장된 튜플들이 갖는 중첩된 시간정보를 서로 구별되도록 제어하기 위해 사용된다. 둘째, 각각의 고정 간격 집합에 대하여 해당 집합에 소속되며 집계함수 내부의 일반조건과 시간조건을 만족하는 모든 적합한 튜플들을 선정한다. 셋째, 집계함수에 분할속성이 존재하는 경우 이를 기준으로 고정 간격 집합을 부집합(subset)으로 분할하여 집계 집합(aggregate sets)을 계산한다. 넷째, 각각의 집계 집합에 대해 지정된 집계함수 기능을 적용하여 집계값을 계산한다. 마지막으로 집계값의 유효시간과 질의상의 시간조건을 만족하는 나머지 속성과의 시간간격을 고려하여 결과 속성값으로

반환한다.

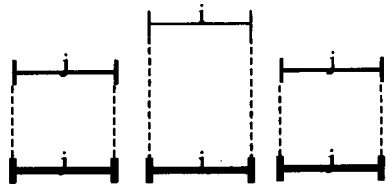
요약하면 시간지원 집계 트리 전략은 시간 속성을 기준으로 정렬되지 않은 테이블로부터 고정 간격 집합의 생성과 분할에 대한 처리를 용이하게 하는 장점을 갖는다. 또한 분할 속성을 기준으로 주어진 테이블을 분할한 후 고정 간격 집합을 계산하면 고정 간격 집합 내에서 튜플의 중복을 방지할 수 있을 뿐만 아니라 카테시안 곱 연산 혹은 죠인 연산을 실행하는 경우 요구되는 메모리량을 대폭 줄일 수 있다. 하지만 시간지원 집계 트리 전략이 가지는 문제점은 시간이 가지는 특성에 의하여 이진 트리가 사향 트리화(skewed tree)가 됨으로써 전반적으로 시간지원 집계처리 성능이 저하될 수 있으며, 특히 이력이 지속적으로 추가됨으로써 데이터베이스의 규모가 증가될수록 사용자 질의에 포함된 시간지원 집계를 위한 자료의 검색시간이 길어질 뿐만 아니라 중간결과를 저장하는 주기억장치내 공간의 비효율성을 야기 시킬 수 있다.

3. 집계 질의 처리 비용

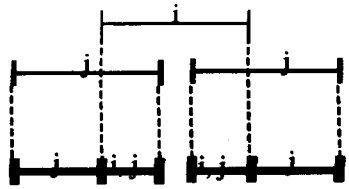
3.1 평가 요소 및 비용 산출 방안

일반적으로 데이터베이스 시스템의 성능 평가에 사용되는 요소로는 질의 응답시간, 디스크 입출력 시간, 디스크 입출력 횟수, 메모리 소요량 등이 있다. 이들 요소중에서 질의 응답시간과 디스크 입출력 시간은 구현후 다양한 형태의 실제적인 실험결과에 의해 계산되므로, 이 논문에서는 시간지원 집계질의가 입력되는 경우 제안된 처리방안에 대한 이론적인 관점에서 집계 계산에 소요되는 전체 메모리를 의미하는 메모리 소요량을 기준으로 한정하여 평가한다.

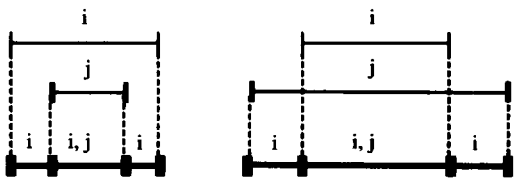
일반적으로 시간지원 집계질의의 처리 대상이 되는 데이터베이스내 임의의 한 테이블에 저장된 튜플이 갖는 시간 속성에는 유효시간(valid time)과 거래시간(transaction time)이 있는데, 유효시간은 현실 세계의 객체가 갖는 유효한 시간값을 의미하며 거래시간은 이들 객체가 데이터베이스내에서 처리된 시점을 가리킨다. 이중에서 사용자에 의해 지정되는 객체에 대한 유효시간 시간간격에 대한 형태는 다음의 세 가지 경우로 분류되며, 대부분의 경우에 유효시간 시간간격 수는 1과 T_{max}일 경우의 시간간격 수 사이에 존재한다. 첫째, T_{min}은 테이블내의 튜플들이 시간간격의 중복이 없는 경우로서, 이를 정형의미로 표현하면 $V_f^i < V_t^i < V_f^j < V_t^j$가 된다. 둘째, T_{mid}는 튜플들의 시간간격이 하나씩만 중복되는 경우로서, 이를 정형의미로 표현하면 $V_f^i < V_f^j < V_t^i < V_t^j$가 된다. 셋째, T_{max}는 임의의 튜플이 갖는 시간간격을 다른 튜플의 시간간격이 모두 포함하는 경우로서, 이를 정형의미로 표현하면 $V_f^i < V_f^j < V_t^j < V_t^i$가 된다. 여기서 V_f^i와 V_t^i는 i번째 튜플의 유효 시작시간과 유효 종료시간을 의미하며, V_f^j와 V_t^j는 j번째 튜플의 유효 시작시간과 유효 종료시간을 의미한다. 그림 1은 각 튜플이 갖는 유효시간을 기준으로 하는 T_{min}, T_{mid}, T_{max}의 경우에 대한 고정 간격 집합을 보여준다. 시간지원 데이터베이스에서 일반적으로 테이블내에 저장된 튜플이 갖는 시간 유형은 그림 1에서 표현된 T_{min}, T_{mid}, T_{max}의 세 가지 경우가 혼합된 구조를 갖기 때문에, T_{min}, T_{mid}, T_{max}는 테이블에 대하여 시간지원 집계 트리 전략에서 생성 가능한 최대 노드 개수를 의미한다.



(a) T_{min}



(b) T_{mid}



(c) T_{max}

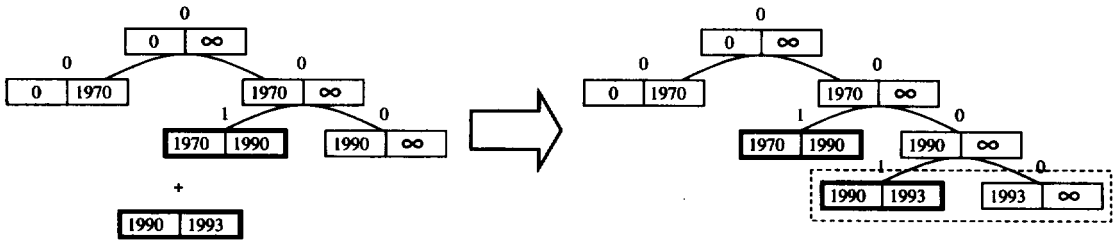
(그림 1) T_{min} , T_{mia} , T_{max} 의 고정 간격 집합

(Fig. 1) Constant Interval Sets for T_{min}, T_{mia}, T_{max}

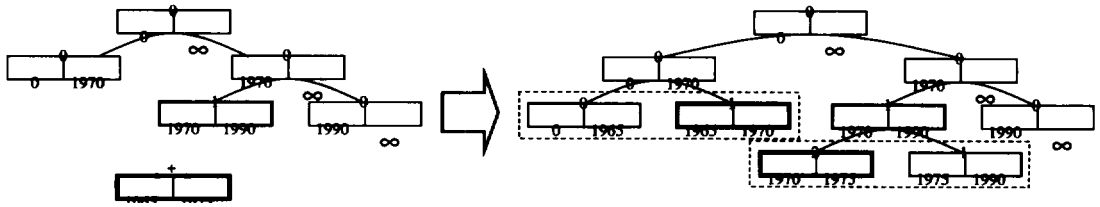
그림 1에서 표현된 T_{min}, T_{mid}, T_{max}의 세 가지 경우에 대하여 각각 생성되는 고정 간격 집합은 시간지원 집계 트리 전략에서 사용되는 이진 트리의 크기를 결정하여 전반적인 성능을 좌우하는 중요한 요소이다. 이 논문에서 시간지원 집계 트리 전략에 대한 평가는 이진 트리내 노드 개수와 분할 속성이 존재할 때 생성되는 트리 연결 리스트 개수로 구성되는 메모리 소요량을 처리비용의 척도로 사용한다.

3.2 노드 개수

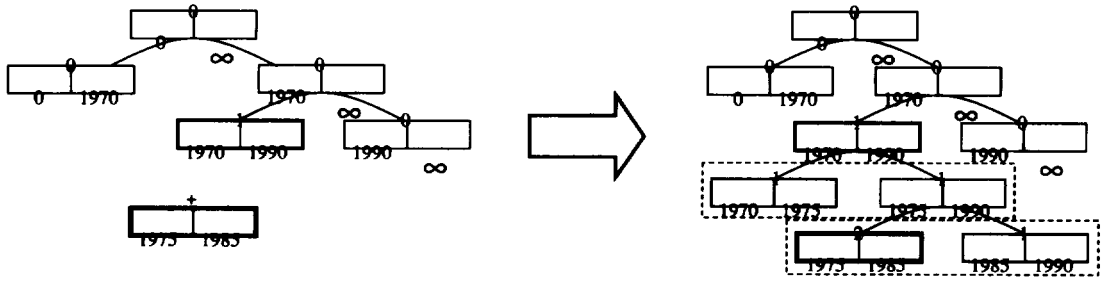
시간지원 집계 트리 전략에서 처음 하나의 튜플에 대하여 기본적으로 하나의 루트노드와 네 개의 자식노드로 구성되는 이진 트리가 생성된다. 이진 트리내 각 노드는 3.1절에서 정의된 고정 간격 집합을 의미하며, 이후 입력되는 튜플로부터 유효 시간값을 기준으로



(a) T_{min}



(b) T_{mia}



(c) T_{max}

(그림 2) T_{min} , T_{mia} , T_{max} 의 분할노드개수

(Fig. 2) Number of Split Node for T_{min} , T_{mia} , T_{max}

T_{min}, T_{mid}, T_{max}의 경우에 대하여 각각 0개 이상의 새로운 노드로서 구성되며, 루트노드를 제외한 나머지 노드에 대하여 동일한 속성값이 해당 노드에서 이미 존재하는 경우에 연결 리스트가 생성된다. 이를 기반으로 T_{min}, T_{mid}, T_{max}의 경우에 대하여 데이터베이스내 전체 튜플 개수를 N으로 하였을 때, 그림 2와 같이 이진 트리내 노드가 분할 되는 과정을 간략하게 표현하였다.

그림 2(a)에서 1970년부터 1990년까지의 유효시간을 갖는 첫 번째 튜플로부터 5개의 노드로 구성된 트리가 생성된다. 이후 1990년부터 1993년까지의 유효시간을 갖는 두 번째 튜플에 대하여 트리는 점선으로 표시된 영역에 2개의 새로운 노드를 추가한다. 이때 각 노드의 상위에 표시된 숫자값은 해당 노드내 속성값의 존재 유무를 표현하는데, 0인 경우는 다른 노드를 위해 임시적으로 생성되었음을 나타내고, 1인 경우는 해당 노드에 1개의 속성값이 존재함을 의미한다. 그림 2(b)는 두 번째 튜플의 유효시간 종료값이 첫 번째 튜플의 유효시간과 중첩되는 경우로서 점선으로 표시된 영역에 4개의 새로운 노드가 추가된다. 그리고 그림 2(c)는 두 번째 튜플의 유효 시간값이 첫 번째 튜플의 유효 시간값에 완전히 포함되는 경우로서 점선으로 표시된 영역에 4개의 새로운 노드가 추가된다. 그림 2에서 보여진 바와 같이 T_{min}, T_{mid}, T_{max}에 대하여 새로운 노드가 추가될 경우 생성되는 노드의 개수는 (식 1)과 같이 유도된다. 즉, 그림 2(a)에서 첫 번째 튜플만이 존재하는 경우의 노드 개수는 5개이며, 새로운 튜플이 삽입되면 7개가 됨을 알 수 있다. 또한 (식 1)의 노드 개수에서 T_{mid}와 T_{max}가 같은 이유는 두 경우 모두 이진 트리를 생성할 때 하나의 튜플은 시간간격의 중복으로 분할이 이루어지면서 4개의 노드가 필요하기 때문이다. 그리고 각 트리의 루트 노드내에 정의되는 연결 리스트는 주어진 시간지원 집계질의에서 지정된 분할 속성에 의해 두 개 이상의 구별되는 값이 발생되었을 때 개별적으로 생성된다.

$$T_{min}=2N+3,\ T_{mid}=T_{max}=4N+1 \quad \text{(식 1)}$$

3.3 고정 간격 집합과 집합내 튜플 수

고정 간격 집합은 2장에서 설명한 바와 같이 주어진 테이블으로부터 더 이상의 튜플이 추가되거나 삭제되지 않는 상태의 시간간격을 의미하며 주어진 튜플의 유효시간값에 따라서 최소 N개로부터 최대 2N-1개가 생성된다. (식 2)에서 보는 바와 같이 T_{mid}와 T_{max}에서는 하나의 튜플이 두 개의 새로운 고정 간격 집합을 생성하기 때문에 동일한 값을 갖는다. 그리고 임의의 한 고정 간격 집합내에 포함된 튜플의 개수는 (식 3)과 같이 계산되는데, 먼저 T_{min}에서는 시간간격의 중복이 없으므로 각 고정 간격 집합내에는 1개의 튜플만이 존재하며, T_{mid}와 T_{max}에서는 중복이 하나씩 발생하므로 한 고정 간격 집합에는 2개의 튜플이 존재한다.

$$T_{min}=N,\ T_{mid}=T_{max}=2N-1 \quad \text{(식 2)}$$

$$T_{min}=1,\ T_{mid}=T_{max}=2 \quad \text{(식 3)}$$

4. 성능 분석

이 논문에서 설명한 시간지원 집계 트리 전략은 입력된 시간지원 집계 질의내에 분할 속성의 존재 여부여부에 따라서 그 처리 과정이 세분화 되며, 각각에 대한 성능을 구분하기 위하여 메모리 요구량 관점에서 논리적인 분석을 통해 평가한다. 먼저 이 논문에서 설명하는 시간지원 집계 트리 전략의 구현에서 사용된 이진 트리와 임시 테이블에 대한 자료구조를 아래의 그림 3에서 표현하였다.

시간지원 집계 트리 전략의 성능을 평가하기 위해 사용된 수식에서 R_n은 루트 노드의 개수, C_n은 자식 노드의 개수, L_n은 연결 리스트의 개수, P는 분할 속성값의 개수, C_I는 임시 테이블내 고정 간격 집합의 개수, T_S는 임시 테이블에서 한 고정 간격 집합내의 중복된 튜플의 개수, C_{np}는 분할속성이 있을 때의 자식 노드의 개수, 그리고 n은 동일한 분할 속성을 갖는 튜플의 개수를 의미한다.

그리고 시간지원 집계 트리 전략에서 입력된 집계질의를 처리하는데 필요한 메모리의 크기는 (식 4)와 같이 각 처리 단계에서 요구되는 메모리의 총합으로 계산되는데, 그림 3에서 표현된 바와 같이 하나의 루트 노드는 28바이트, 자식 노드는 24바이트, 연결 리스트는 8바이트, 임시 테이블내 각 튜플과 연결 리스트는 각각 32바이트와 12바이트의 크기를 갖는다.

트리의 유효 시작시간	트리의 유효 종료시간
분할 속성값	다음분할속성으로의 포인터
다음 트리로의 포인터	
좌측 자식노드의 포인터	우측 자식노드의 포인터

루트노드 : 28바이트

노드의 유효 시작시간	노드의 유효 종료시간
집계 속성값	다음 집계 속성값
좌측 자식노드의 포인터	우측 자식노드의 포인터

자식노드 : 24바이트

(a) 이진 트리 구조

(a) Binary Tree Structure

집계 속성값	다음 집계 속성값

연결 리스트 : 8바이트

(b) 연결 리스트 구조

(b) Linked List Structure

계수기	분할 속성	다음 분할속성	집계 속성	다음 집계속성	유효시간		다음 튜플 포인터
					시작 시간	종료 시간	

메모리 주소	집계 속성값	다음 집계 속성값

연결 리스트 : 12바이트

(c) 임시 테이블 구조

(C) Temporary Table Structure

(그림 3) 시간지원 집계 트리 전략의 자료 구조

(Fig. 3) Data Structure for Temporal Aggregate Tree Strategy

(1 단계) 이진 트리=28* R_n+24* C_n+8* L_n,

(2 단계) 임시 테이블=32* C_I+12* L_n,

(3 단계) 집계 계산=32* C_I, (식 4)

전체 메모리=28* R_n+24* C_n+20* L_n+64* C_I

단, C_n=(T_{min}-1)또는(T_{mid}-1)또는(T_{max}-1)

4.1 분할속성이 없는 경우

분할속성이 없는 경우에 시간지원 집계트리 전략의 첫 번째 단계에서 요구되는 메모리량은 이진 트리내에 존재하는 모든 노드와 연결 리스트의 개수를 합한 값으로 정의된다. 즉, 분할 속성이 없는 경우에 오직 하나의 이진 트리가 생성되므로 1개의 루트 노드가 존재하며, 자식 노드와 견결 리스트 값은 (식 1)과 (식 3)에서 구한 T_{min}, T_{mid}, T_{max}의 경우에 대한 C_n과 L_n값을 (식 4)에 대입하여 필요한 메모리를 정리할 수 있으며, 전체 튜플의 수를 N으로 표현하였을 때 다음의 (식 5)와 같이 계산된다.

$$T_{min}=28*1+24*(2N+3)+8*0=48N+110 \quad (식\ 5)$$
$$T_{mid}=T_{max}=28*1+24*(4N+1)+8*(N-1)=104N+44$$

두 번째 단계인 임시 테이블 생성 단계에서 요구되는 메모리량은 이전 단계에서 추출되는 고정 간격 집합을 (식 4)에 적용하여 구할 수 있다. 이때 C_I값은 (식 2)에서 구한 값을 사용하며, (식 6)과 같이 계산한다. 그리고 세 번째 단계에서 요구되는 메모리량은 (식 7)과 같으며, 시간지원 집계 트리 전략에서 분할 속성이 없는 경우에 요구되는 전체 메모리량은 (식 8)과 같다.

$$T_{min}=32*N+12*0=32N \quad (식\ 6)$$
$$T_{mid}=T_{max}=32*(2N-1)+12*(N-1)=76N-44$$

$$T_{min}=32*N=32N$$
$$T_{mid}=T_{max}=32*(2N-1)=64N-32 \quad (식\ 7)$$

$$T_{min}=28*1+24*(2N+3)+20*1+64*N=92N+120$$
$$T_{mid}=T_{max}=28*1+24*(4N+1)+20*(N-1)+64*(2N-1)=244N-32 \quad (식\ 8)$$

4.2 분할 속성이 있는 경우

분할 속성이 있는 경우 시간지원 집계 트리 전략에서 전체 튜플 수(N)의 값이 분할 속성 값의 개수(p)와 분할 속성의 튜플 개수(n)를 곱한 값이 된다. 즉, 분할 속성이 있는 경우 시간지원 집계 트리 전략에서의 이진 트리내 전체 노드의 개수는 각 분할 속성에 의해 분할 되는 속성 값의 개수를 곱한 결과가 되며, 아래의 (식 9)와 같이 계산된다.

$$T_{min} = (2n+3)*p + 0*p = 2np + 3p \quad \text{(식 9)}$$
$$T_{mid} = T_{max} = (4n+1)*p + (n-1)*p = 5np$$

이상과 같이 시간지원 집계 트리 전략으로 집계 함수를 포함하는 질의를 처리하는데 필요한 메모리의 크기는 분할 속성이 없는 경우와 동일하게 각 처리 단계에서 요구되는 메모리를 합하면 된다. 즉, 각 단계에서 요구되는 메모리 량을 구하는 식과 전체 메모리 요구량은 (식 4)와 대부분 동일하지만 아래의 (식 10)과 같인 C_n 값이 변경된다. 그리고 (식 10)에 각 변수값을 T_{min}, T_{mid}, T_{max} 일 경우에 대입하여 필요한 메모리를 구하는 식을 정리하면 n과 p로 표현되며, 각각 다음의 (식 11)과 같다.

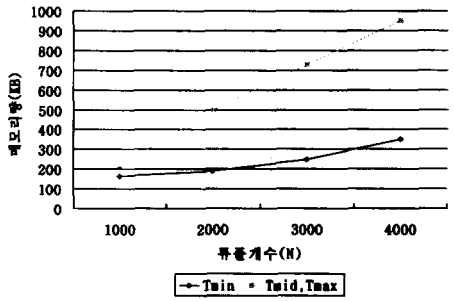
$$C_n = (T_{min} - p) \text{또는} (T_{mid} - p) \text{또는} (T_{max} - p) \quad \text{(식 10)}$$

$$T_{min} = 112np + 4p + 72,$$
$$T_{mid} = T_{max} = 244np + 4p - 60 \quad \text{(식 11)}$$

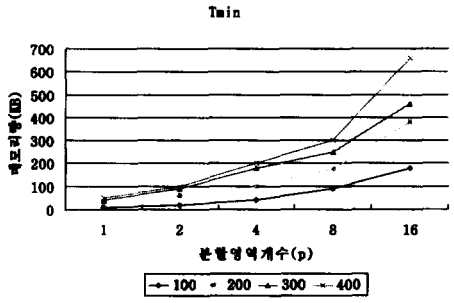
5. 성능 평가

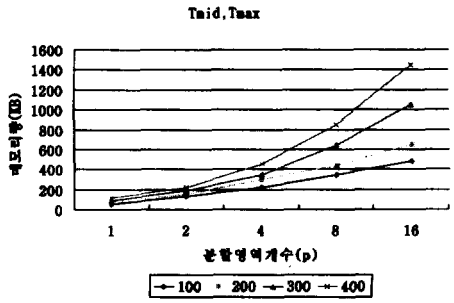
시간지원 집계 트리 전략에서 요구되는 메모리량을 분석하면 생성되는 노드의 개수는 분할 속성이 없는 경우 T_{min}에서는 2N+3이 되며 T_{mid}와 T_{max}에서는 4N+1이다. 이때 T_{mid}와 T_{max}가 같은 이유는 이진 트리를 만들 때, 한 튜플에 대하여 4개의 노드를 생성하기 때문이다. 아울러 분할 속성이 없는 경우의 메모리 소요량을 복잡성의 차수(degree of complexity) 측면에서 평가하면 T_{min}, T_{mid}, T_{max} 모두가 $O(n)$이지만 상수값의 폭이 급증한다. 또한 분할 속성이 있는 경우에 요구되는 메모리량은 복잡성의 차수 측면에서 평가하면 T_{min}, T_{mid}, T_{max} 모두가 $O(np)$이다.

그림 4는 분할 속성의 존재 여부와 튜플의 개수에 따른 시간지원 집계 처리 전략의 처리 비용을 논리적으로 모의 실험한 결과로서, 튜플의 개수와 분할 속성의 개수가 증가할수록 처리 비용이 큰 폭으로 증가함을 보여준다. 아울러 T_{mid}와 T_{max}의 경우가 대등 소이한 이유는 입력된 튜플로부터 생성되는 이진 트리내 노드 개수와 고정 간격 집합의 수가 유사하기 때문이다. 그림 4(b)에서는 분할 속성에 의해 분할 되는 영역을 x축으로 하여 2^x, x={0..4}인 경우에 고정 간격 집합내 튜플의 개수를 y축으로하여 각각 $n \times 10^y$, n=1..4, y=2..3으로 설정하여 얻은 수치값을 그래프로 표현한 것이다.



(a) 분할 속성이 없는 경우
(a) In case of No Partitioning Attribute





(b) 분할 속성이 있는 경우
(b) In case of Partitioning Attribute

(그림 4) 시간지원 집계 질의에 대한 메모리 소요량 비교
(Fig. 4) Comparison of Memory Requirement for Temporal Aggregate Query

6. 결 론

시간지원 데이터베이스는 현실세계에 존재하는 객체에 대한 효율적인 이력 표현과 연산을 지원한다. 특히 시간지원 집계는 현재시점에서 유효한 자료 뿐만 아니라 이력 자료로부터 일련의 계산을 통해 부가적인 정보를 생성해 내는 기능을 갖는다. 따라서 다양한 유형의 집계함수의 지원은 전반적인 데이터베이스 시스템의 편리함과 성능을 결정하는 중요한 요소가 된다.

이 논문에서는 시간지원 데이터베이스에서 집계 함수가 포함된 질의에 대한 효율적인 처리 방안으로서 시간지원 집계 질의 처리 전략을 소개하고, 그 처리 방안에 대한 성능을 처리 비용의 관점에서 분석 및 평가하였다. 세부적으로 T_{min}, T_{mid}, T_{max}등 세가지 경우로 분류하여 각 단계에서 요구되는 메모리량을 계산하기 위한 방안을 보였으며, 모의 실험을 통해 비록 계수가 비교적 크지만 O(N) 또는 O(np)의 결과를 도출할 수 있었다. 앞으로의 연구에서는 이 논문에서 제시한 시간지원 집계 트리 전략의 성능에 대한 실제적인 검증 및 개선 방안에 대한 연구가 이루어질 예정이다.

참 고 문 헌

[1] I. Ahn, *Performance Modeling and Access Methods for Temporal Database Management Systems*, Technical Report 86-018, Department of Computer Science, University of North Carolina at Chapel Hill, Aug., 1986.

[2] J. Allen, *Maintaining Knowledge about Temporal Intervals*, Communications of the ACM, Vol.26, No.11, Nov. 1983.

[3] R. Epstein, *Techniques for Processing of Aggregate in Relational Database Systems*, UCB/ERL M7918, Computer Science Department, Univ. of California at Berkeley, Feb. 1979.

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

[5] D.H. Kim and K.H. Ryu, *The Design of Aggregation Function for Temporal Databases*, Proceeding of The 16th KITE Summer Conference, Vol.16, No.1, Jul. 1993.

[6] D.H. Kim, K.H. Jeon, K.J. Jeong, K.J. Kim and K.H. Ryu, *A Temporal Database Management Main Memory Prototype*, IEEE Tencon '94 9th Annual Internaltional Conference, Aug. 1994.

[7] D.H. Kim, I.H. Lee and K.H. Ryu, *The Design and Implementation of Aggregate Function of Temporal Databases in Main Memory*, Journal of The KISS, Vol.21, No. 8, Aug. 1994.

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

[9] V. Lum, P. Dadam, R. Erbe, J. Guenauer, P. Pister, G. Walch, H. Werner, and J. Woodfill, *Designing DBMS Support for The Temporal Dimension*, In Proceedings of ACM SIGMOD International Conference on Management of Data, Jul. 1984.

[10] K.H. Ryu, *Execution Tree for Incremental View Materialization of Temporal Databases*, Journal of The KISS, Vol.21, No.8, Aug. 1993.

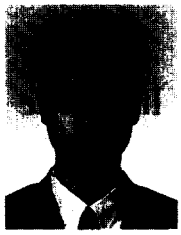
[11] N. Sarda, *Extensions to SQL for Historical Databases*, IEEE Transaction on Knowledge and Data Engineering, Vol.2, No.2, Jun. 1990.

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

[13] R. Snodgrass, S. Gomez, and E. Mckenzie, *Aggregate in the Temporal Query Language TQuel*, IEEE Transactions on Knowledge and Data Engineering, Vol.5, No.5, Oct., 1993.

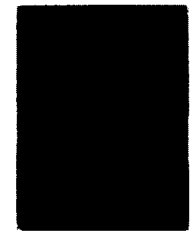
이 종 연

1985년 충북대학교 전자계산학과 (학사)
1987년 충북대학교 대학원 전자계산기공학과(석사)
1998년 충북대학교 대학원 전자계산학과 박사수료
1990년~1994년 현대전자(주) 소프트웨어연구소
1994년~1996년 현대정보기술(주) CIM사업부 근무 (책임)
관심분야 : 시공간 데이터베이스, 능동 데이터베이스, GIS, CIM



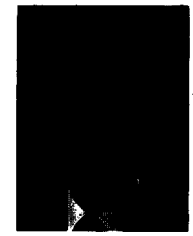
김 동 호

1993년 충북대학교 전자계산학과 (이학사)
1995년 충북대학교 대학원 전자계산학과(이학석사)
1997년 충북대학교 대학원 전자계산학과 박사수료
1997년 한국전자통신연구원 위촉연구원
관심분야 : 시간지원 데이터베이스, 시공간 데이터베이스, 집계처리시스템, 멀티미디어 시스템



이 인 흥

1975년 고려대학교 전자공학과 졸업(공학사)
1986년 광운대학교 대학원 전자계산학과 졸업(공학석사)
1996년 홍익대학교 대학원 전자계산학과 졸업(이학박사)
1984년~현재 전주대학교 컴퓨터공학과 교수
관심분야 : 시간 데이터베이스, 데이터웨어하우징



류 근 호

1976년 숭실대학교 전산학과 졸업(이학사)
1980년 연세대학교 산업대학원 전산전공(공학석사)
1988년 연세대학교 대학원 전산전공(공학박사)
1976년~86년 육군군수 지원사 전산실(ROTC장교), 한국전자통신연구소(연구원), 한국방송통신대 전산학과(조교수) 근무
1989년~91년 Univ. of Arizona Research Staff (TempIS 연구원 Temporal DB)
1986년~현재 충북대학교 컴퓨터과학과 교수 겸 컴퓨터정보통신연구소장
관심분야 : 시간지원 데이터베이스, 시공간 데이터베이스, 정보검색, 객체 및 지식베이스 시스템