시간지원 집단 함수 처리를 위한 시점 시퀀스

권준호*, 배진욱*, 송병호**, 이석호*
*서울대학교 전기·컴퓨터공학부
**상명대학교 소프트웨어학과
{bluerain, oblody}@db.snu.ac.kr, bhsong@sangmyung.ac.kr, shlee@cse.snu.ac.kr

Time Point Sequence for the Evaluation of Temporal Aggregate Function

Joonho Kwon*, Jinuk Bae*, Byoungho Song**, Sukho Lee*
*School of Electrical Engineering and Computer Science, Seoul National University
**Dept. of Software Science, Sangmyung University

요 약

시간에 따라 변화하는 자료들을 저장하는 시간지원 데이타베이스에서 집단 함수는 시간지원 그룹화를 통하여 집단 함수 값이 변하지 않는 시간 구간을 구하고 그 각각의 구간마다 집단 함수의 결과를 생성해야 하는 복잡한 연산이다. 기존의 시간지원 집단 함수 처리 기법들은 집단 함수를 포함하는 질의가 요구되었을 때, 불변 구간을 구하기 위해 트리와 같은 자료구조를 생성하고 이 트리의 노드들을 순회함으로서 집단 함수의 결과를 생성하였다. 이 논문에서는 미리 데이타베이스를 한 번 스캔하여 투플의 시작 시간과 종료 시간들의 정렬된 집합인 시점 시퀀스를 생성하고, 이를 이용하여 시간지원 집단 함수를 처리하는 방법을 제안한다. 또한 데이타베이스에서 저장된 데이타의 삭제나 새로운 데이타의 삽입에 따른 시점 시퀀스의 갱신 방법도 제시한다.

1. 서론

질의 연산자 중에서 모든 직원 월급의 평균 같은 집단 함수(aggregate function)는 릴레이션의 전체 혹은 그 일부를 구성하는 투플에 적용이 되어서 하나의 스칼라 값을 계산해 낸다. 이러한 집단 함수들은 질의 언어에서 매우 중요한 부분이며, 많은 응용 프로그램들에서 자주 사용된다. 질의 성능평가에서는 집단 함수를 포함하는 질의가 상당 부분 포함되어 있다[4,5]. 그러므로 데이타베이스 응용의 성능을 향상시키기 위해서 집단 함수를 효율적으로 처리하는 것이 중요하다.

기존의 데이타베이스 시스템은 현실 세계에서 발생한 사건에 대하여 가장 최근의 상태만을 반영한다. 반면에 시간지원 데이타베이스(temporal database) 시스템은 과거, 현재, 미래의 데이타를 유지한다. 시간지원 데이타베이스에서 투플들은 일반적으로 시작 시간과 종료 시간이라는 두개의 시간 속성을 가지고 있다.

따라서 기존의 데이타베이스 시스템에서의 집단 함수 처리와는 달리 시간지원 데이타베이스에서 집단 함수의 처리는 시간지원 그룹화(temporal grouping)를 수행해야 하기 때문에 더욱 복잡하다. 시간지원 그룹화란 시간축에 따라 불변 구간들을 분할하고, 투플들을 이 분할에 따라 그룹화를 하는 것이다. 시간지원 집단 함수의 결과는 시간 지원 그룹화에 의해 생성된 각각의 그룹들에 대하여 집단 함수를 적용한다.

본 논문에서는 시점 시퀀스(time point sequence)를 제안하고, 이를 기반으로 한 시간지원 집단 함수 처리 방법을 제시한다. 시점 시퀀스는 시간지원 그룹화를 위해 필요한 불변 구간들을 구하기 위해, 데이타베이스를 스캔하여 투플들의 시작 시간과 종료 시간을 미리 정렬하여 유지하는 것이다.

이 논문의 구성은 다음과 같다. 2장에서는 시간지원 집단 함수의 의미와 기존의 시간지원 집단 함수 처리 기법에 대해서 살펴본다. 3장에서는 시점 시퀀스를 소개하고, 시점 시퀀스의 생성과 시점 시퀀스를 이용한 시간지원 집단 함수의 처리에 대해 언급하고 시점 시퀀스의 갱신에 대해 설명한다. 마지막으로 4장에서는 연구 결과를 정리하고 결론을 맺는다.

2. 관련 연구

2.1 시간지원 집단 함수

시간지원 데이타베이스에서는 각 투플이 시간 속성을 포함하고 있으므로, 시간을 고려하여 질의를 처리해야 한다. TSQL2는 SQL-92 질의 언어에 시간지원에 관한 요소들을 추가하였다[6]. 특히 TSQL2는 SQL-92의 GROUP BY 절에 시간 지원 그룹화를 추가하였다. 따라서 집단 함수 처리시에 시간 지원 그룹화를 하여 시간 구간을 분할하며, 분할된 각각의 구간마다 집단 함수의 결과값을 구한다.

이 논문에서는 시간지원 데이타베이스에 기록되는 데이타들은 유효 시간(valid time)으로 시작 시간 T_s, 종료 시간 T_e를 시간 속성으로 가지고 있고, 이 구간 [T_s, T_e] 사이에서 이 데이타는 현실 세계에서 참이라고 말할 수 있다고 가정한다.

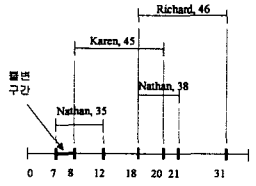
그림 1(a)와 같은 예제 테이블이 있다. 이 테이블은 은 어떤 회사에 채용된 직원의 월급, 부서, 기간을 기록하고 있다. 예제 테이블에 그림 1(b)와 같은 TSQL2 질의가 요청되었다고 하자. 시간지원 데이타베이스에서는 질의 처리시에 시간지원 그룹화를 하고, 각각의 구간 마다 집단 함수 값이 변화지 않는 불변 구간(constant interval)을 구하는 그림 1(c)와 같은 과정이 필요하다. 불변 구간을 구한 후 최종적으로 그림 1(d)와 같이 각 불변 구간마다 집단 함수 결과를 생성한다.

Name	Salary	Dept	Start	End
Richard	46K	Accounting	18	31
Karen	45K	Shipping	8	20
Nathan	35K	Marketing	7	12
Nathan	38K	Accounting	18	21

(a) 예제 테이블

```
SELECT COUNT(Name), MAX(Salary)
From Employed E
```

(b) TSQL2 질의



(c) 불변 구간

COUNT	MAX	Start	End
1	35K	7	8
2	45K	8	12
1	45K	12	18
3	46K	18	20
2	46K	20	21
1	46K	21	31

(d) 결과

그림 1 예제 테이블과 시간지원 집단 함수[1]

2.2 관련 연구

시간지원 집단 함수 계산을 위한 방법으로 집단화 트리(aggregation tree) 기법[1], AVL 트리의 균형화 기법을 사용한 PA-트리(Point-based Aggregation tree) 기법[2], 작은 규모의 시간 지원 집계에서 COUNT, SUM, AVG 같은 누적형 집단화(computational aggregation) 처리를 위한 레드-블랙 트리의 균형화 기법을 이용한 균형화된 트리(balanced tree) 기법과 MIN, MAX 같은 선택형 집단화(selective aggregation) 처리를 위해 합병 정렬를 사용한 기법[3]이 있다. 트리를 이용한 시간 지원 집단 함수의 처리 알고리즘은 불변 구간을 구하기 위한 트리의 생성 과정과 집계의 결과를 계산하기 위한 트리 노드의 순회 과정으로 이루어진다. 합병 정렬에 기반을 둔 기법은 두 개의 작은 중간 집계 결과들을 합침으로서 더 큰 중간 결과들을 계산하고 이 과정을 반복해서 최종 결과를 생성한다.

3. 시점 시퀀스(Time Point Sequence)

3.1. 시점 시퀀스의 생성

기존의 기법들은 불변 구간을 구하기 위하여 많은 메모리를 사용하며, 어느 정도의 수행 시간을 필요로 한다. 이 불변 구간을 집단 함수를 포함한 질의를 처리하는 도중에 구하지 않고, 불변 구간을 구할 수 있는 시간 값들을 시점 시퀀스로 유지하는 것이 이 논문의 기본 아이디어이다. 시점 시퀀스는 시간 축을 따라 정렬된 시간 값들의 순차이다. 시점 시퀀스에 저장될 수 있는 시간 값들은 어떤 투플의 시작 시간(T_s)이나 종료 시간(T_e)이다. 시점 시퀀스에 저장될 수 있는 시점들의 집합(Time Point Set)은 다음과 같이 형식화 할 수 있다.

$$\text{Time Point Set} = \{T_i \mid \exists t \in TDB((T_i = t.T_s) \vee (T_i = t.T_e))\}$$

시점 시퀀스의 생성은 다음과 같은 방법으로 한다. 미리 전체 데이타베이스를 한번 스캔한다. 이 때 투플의 시작 시간과 종료 시간만을 읽어들여서 시점 집합에 추가한다. 이 시점 집합을 정렬하면 시점 시퀀스를 얻을 수 있다.

그림 1(a)의 예제 테이블에 대한 시점 시퀀스를 구해보자. 시점 집합은 투플의 시작 시간과 종료 시간의 집합이므로 예제 테이블의 투플을 읽은 순서대로 시점들을 추가하면 시점 집합 = {18,31,8,20,7,12,18,21}이 되고, 이 시점 집합을 정렬하면 시점 시퀀스 = {7,8, 12,18,20,21,31}의 순서로 구성된다.

3.2. 시간지원 집단 함수의 처리

미리 만들어 놓은 시점 시퀀스는 시간 값들을 시간 축으로 정렬한 것이다. 따라서 이 시점 시퀀스에 있는 시점들을 차례대로 이웃한 2개씩 쌍을 만들면 그 사이의 시간이 불변 구간이 된다. 시간지원 집단 함수 질의 요청이 오면 미리 만들어서 유지하고 있던 시점 시퀀스에서 불변 구간을 구성한 다음 이를 메모리에 읽어 들여서 유지한다. 그런 다음 시간지원 집단 함수 질의가 수행되어야 하는 테이블의 투플들을 읽어 들인다. 이 때 투플의 시작 시간과 종료 시간이 포함되는 모든 불변 구간에 대해서 질의의 결과로 요구되는 연산을 수행하면 된다. 모든 투플들을 다 읽어 들인 후 최종적으로 생성된 값이 시간지원 집단 함수의 결과가 된다.

그림 2는 그림 1(a) 예제 테이블에 대하여 그림 1(b)와 같이 COUNT와 MAX 시간지원 집단 함수를 요구하는 TSQL2 질의를 처리하는 과정을 보여주고 있다.

먼저 시점 시퀀스에서 시간 값들을 읽어 들여서 불변 구간을 구하여 메모리에 로드한다. 그런 다음 해당되는 릴레이션의 투플들을 읽는다. 유효 시간이 [18,31]인 첫 번째 투플을 읽어 들이면, 메모리에서 [18,31]이 포함하는 구간인 [18,20]과 [20,21], [21,31]을 찾아서 그림 2(a)와 같이 COUNT 값에 1을 더하고, MAX는 46K로 한다. 두 번째 투플을 읽어 들이면 [8,12]과 [12,18], [18,20]의 구간에 값을 반영하고, 세번째 투플을 읽어

들여서 [7,8], [8,12]의 구간에 COUNT와 MAX값을 반영한다. [18,20], [8,12]의 구간은 두번째, 세번째 투플을 읽어 들이면 COUNT가 2가 되는 것이다. 세 번째 투플까지 읽었을 때 중간 결과는 그림 2(b)가 된다. 나머지 구간도 투플을 읽을 때마다 같은 방식으로 해당되는 집단 함수를 반영하여 중간 결과들을 갱신한다. 모든 투플을 다 읽으면 집계 함수에 대한 최종 결과가 된다. SUM, AVG, MIN의 경우에도 위와 같은 방법으로 처리할 수 있다.

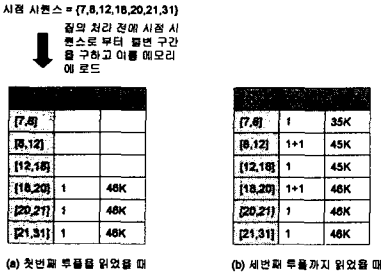


그림 2 시점 시퀀스를 이용한 시간지원 집단 함수의 처리

3.3. 시점 시퀀스의 갱신

시점 시퀀스는 미리 데이타베이스를 스캔하여 투플의 시작 시간이나 종료 시간들을 정렬하여 유지하고 있다. 따라서 새로운 데이타의 삽입이나 삭제가 될 때 시점 시퀀스도 갱신되어야 이를 바탕으로 정확한 불변 구간을 유지할 수 있다.

시점 시퀀스에 유지되는 시점은 하나의 투플에 의해서 발생한 경우도 있고, 두 개 이상의 투플에 의해 발생할 수도 있다. 이 두가지 경우의 구분을 위해서 시점 시퀀스에 시간 값 외에, 부가적으로 시점을 발생시킨 투플의 개수를 기록하는 발생 빈도 필드가 필요하다. 이 필드는 시간지원 집단 함수를 처리하는 과정에서는 필요하지 않고, 시점 시퀀스를 갱신하는 과정에만 필요하다.

새로운 데이타의 삽입이 이루어진 경우, 그 투플의 시작 시간과 종료 시간을 가지고 와서, 이미 정렬되어 있는 시점 시퀀스에서 해당되는 위치를 찾는다. 이미 존재하고 있는 시간 값인 경우 발생 빈도 필드의 값을 하나 증가시키고, 새로 생긴 시간 값이면 시점 시퀀스에 그 시점을 추가한다. 예를 들어 그림 3(a)와 같이 [13, 19]동안 참이고, 이름이 Mike이고 월급이 50K인 데이타가 삽입되었다면, 이 데이타로 인해 기존의 [12,18], [18,20]의 불변 구간이 [12,13], [13,18], [18,19], [19,20]의 불변 구간으로 더 나누어진다. 따라서 그림 3(b)와 같이 새로운 불변 구간을 정확히 유지하기 위해서 기존의 시점 시퀀스에 새 시점 13과 19를 추가하여 새로 정렬하는 시점 시퀀스의 갱신이 필요하다.

기존 데이타의 삭제가 발생한 경우, 우선 그 투플의 시작 시간과 종료 시간에 해당되는 값을 시점 시퀀스에서 찾는다. 그런 다음 그 시점들에 해당되는 발생 빈도 필드를 검사하여 발생 빈도가 1인 것은 삭제를 하고, 2 이상인 시점은 삭제하지 않고 발생 빈도를 하나 감소시키고 시점 시퀀스에는 그대로 유지한다.

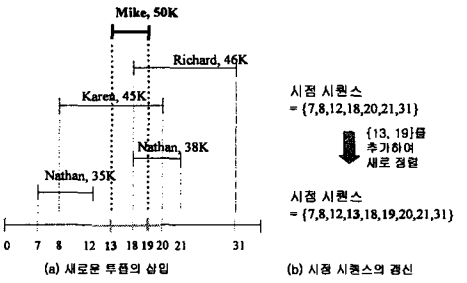


그림 3 데이타의 삽입에 따른 시점 시퀀스의 갱신

4. 결론

이 논문에서는 시점 시퀀스를 제안하고, 이를 이용하여 시간지원 집단 함수 질의를 처리하였다. 기존의 방법들은 시간지원 집단 함수 질의가 요구되었을 때 해당되는 트리를 생성하고 그 트리의 노드들을 순회하여 결과를 돌려준다. 그렇지만, 시점 시퀀스를 이용한 방법은 미리 시점 시퀀스를 생성해 놓고 시간지원 집단 함수 질의 요청이 있을 때 시점 시퀀스로부터 불변 구간을 구하여 메모리에 올린 다음 투플을 읽을 때 해당되는 불변 구간에 대한 값들을 갱신하는 방식으로 집단 함수를 처리한다. 또한 정확한 불변 구간을 구하기 위해 데이타의 삽입이나 삭제에 따른 시점 시퀀스의 갱신도 제안하였다.

참고문헌

[1] N. Kline and R.T. Snodgrass, "Computing Temporal Aggregates", In Proc. of the 11th Inter. Conference on Data Engineering, pp. 222-231, Taipei, Taiwan, March 1995.

[2] Jong Soo Kim, Sung Tak Kang, and Myoung Ho Kim, "*Effective Temporal Aggregation Using Point-Based* Trees", In Proc. of 10th Inter. Conference on Database and Expert Systems Applications, Florence, Italy, Aug/Sep 1999.

[3] Bongki Moon, Ines Fernando Vega Lopez and Vijaykumar Immanuel, "Scalable Algorithms for Large Temporal Aggregation", In Proc. of the 16th Inter. Conference on Data Engineering, pp. 145-154, San Diego, CA, March 2000.

[4] J. Gray, The Benchmark Handbook for database and transcation processing systems, Morgan Kaufmann, San Francisco, 1991.

[5] TPC, TPC benchmark™, Transaction processing performance council, 1994.

[6] R.T. Snodgrass, I. Ahn, G. Ariav, and et al. The TSQL2 Temporal Query Language, Kluwer Academic Publishers, Boston, 1995.