

시간 데이터베이스에서 시간 간격 분할 알고리즘의 구현 및 평가

(Implementation and Evaluation of Time Interval Partitioning
Algorithm in Temporal Databases)

이 광 규 [†] 신 예 호 ^{**} 류 근 호 ^{***} 김 홍 기 ^{****}
(Kwang Kyu Lee) (Ye Ho Shin) (Keun Ho Ryu) (Hong Gi Kim)

요약 조인 연산은 관계형 데이터베이스에서와 같이 시간 데이터베이스에서도 시스템 성능에 큰 영향을 미친다. 특히, 시간 조인은 조인 연산 단계 이전에 간격 분할의 최적화가 질의 처리 성능을 결정한다. 이 논문에서는 시간 데이터베이스의 병렬 조인 질의 처리 성능을 개선하기 위해 시간 조인 연산을 위한 시간 간격을 분할하는 최소 분할 기법을 제안하였고, 제안된 간격 분할의 최소 분할점을 결정하는 최소 간격 분할 알고리즘의 유효성을 예제 시나리오를 통해 검증하였으며, 기존 분할 알고리즘에 비해 성능 개선 효과가 있음을 확인하였다.

키워드 : 시간 데이터베이스, 간격 분할, 병렬 시간 조인, 시간지원 인덱스 구조

Abstract Join operation exert a great effect on the performance of system in temporal database as in the relational database. Especially, as for the temporal join, the optimization of interval partition decides the performance of query processing. In this paper, to improve the efficiency of parallel join query in temporal database, I proposed Minimum Interval Partition(MIP) scheme that time interval partitioning. The validity of this MIP algorithm that decides minimum breakpoint of the partition is proved by example scenario and I confirmed improved efficiency as compared with existing partition algorithm.

Key words : temporal database, interval partitioning, parallel temporal join, temporal index structure

1. 서 론

시간 데이터베이스에서 시간 흐름에 따라 변경되는 자료를 처리할 수 있도록 시간에 대한 명확한 의미(semantic)를 포함시킬 경우 데이터베이스 이용자는 질의어를 통하여 간편하게 시간 개념을 갖는 자료에 접근할 수 있다[1]. 특히, 시간 속성의 하나인 간격(interval)은 시간이라는 지속적인 타임스탬프 속성을 포함하므로 조인 연산 처리를 위한 중요한 데이터 유형이다[2]. 최근에 인터넷의 확산과 각종 멀티미디어 데이터의 증가와 함께 새롭고 복잡한 데이터 유형들 즉, 공

간, 시간, 오디오와 비디오 모델들이 DBMS[3,4]에서 소개되었다. 관계형 모델에 간격 데이터 기능을 확장시킨 많은 용용 분야중 IXSQL[5]과 TSQL2[6,7]는 비록 표준으로 채택되지는 않았지만 SQL3의 형식을 취하고 있다. 하지만 간격 데이터는 시간 모델에서 질의 처리시 여러 문제점을 제기하는데 그 중 하나는 간격 값에서 정의되는 데이터 분할(data partition)[2,8,9,10,11,12,13]이다. 데이터 분할은 상호 배타적으로 분리된 일부 데이터의 집합이나 해싱 조인, 병렬 조인등의 연산처럼 나누어 조인 연산을 처리하는데 사용된다. 간격 데이터 분할시 여러 단편(fragment)들이 생성되고 어떤 특정한 간격 도메인에서 이를 데이터 객체들은 서로 다른 단편을 교차(intersect)한다. 시간 조인을 수행하기 이전에 간격을 분할하는 가장 큰 문제점은 분리(disjoint)가 아닌 데이터 단편을 생성하는 것이다. 이는 질의 처리시 상당한 오버헤드를 준다. 중복을 감소시키기 위해 단편 내 간격수를 증가시키면 병렬 조인 차수가 감소되지만

* 경희원 : 신홍대학 컴퓨터정보개설 교수
kkloc@shc.ac.kr

** 비회원 : 충북대학교 전자계산학과
snowman@dblab.chungbuk.ac.kr

*** 종신회원 : 충북대학교 컴퓨터과학과 교수
khryu@dblab.chungbuk.ac.kr
hgkim@cbucc.chungbuk.ac.kr

논문접수 : 2001년 5월 29일
심사완료 : 2001년 10월 23일

많은 중복이 발생하며 단편내 간격수를 감소시키면 병렬 조인 차수가 증가하는 이론바 최소-최대 혼란(min-max dilemma)[14]을 야기한다. 그러므로, 간격 분할은 신중하고 타당성 있는 논리 전개가 필요하다. 지금까지 많은 연구가들에 의해 간격 분할 문제를 다루는 알고리즘 설계[2,8,10,15,16]와 인덱스 구조[17,18,19]가 소개되었지만 가장 최적의 분할점을 구할 수 있는 방법은 아직 제시된 적이 없다. 따라서, 이 논문에서는 시간 교차 조인 질의 성능을 개선하기 위해 시간 속성인 타임스탬프의 간격을 분할하는 기법인 최소 간격 분할(Minimum Interval Partition: MIP)알고리즘을 제안한다. 제안된 알고리즘은 언더플로우(underflow)분할[17]이 시작시점과 종료시점을 대상으로 간격 분할을 결정하므로 전체적인 조인 성능의 극대화를 시키지 못하는 단점을 개선하여, 간격들의 종료시점만을 분할 결정에 참여시킴으로서 중복되는 간격수를 최소화시켜 분할점 결정 연산 오버헤드를 감소시킬 수 있음을 보인다. 주기(span)를 동일한 크기인 m 개의 크로노(chronon)으로 분할하는 균등 분할(uniform partition)[15]과는 달리 제안된 알고리즘은 분할선을 좌에서 우로 라운드 로빈 방법으로 이동하여 중복을 최소로 하는 분할점을 찾아 분할하는 방법을 사용하였으며 적재되는 단편내 간격수 조절이 가능하였다. 또한, 모든 간격의 시작시점과 종료시점을 분할 대상으로 분할점을 결정하는 [17]과는 달리 제안된 알고리즘은 종료시점을 분할 대상으로 간격의 중복을 최소화하는 분할점을 찾을 수 있다.

2. 관련 연구

시간 데이터베이스에서 조인 방법[2,8,9,10,11,12,13,15,17,18,20]에 관한 연구는 대부분 순차처리 환경에서 시간 동치-조인 방법을 제안하였고 타임스탬프가 교차되는 시간 교차 조인에 관하여는 언급하지 못하였으며 병렬처리 환경에는 부적합하였다. 그러나 간격 데이터 분할에 관련된 질의 처리 기법중 시간 조인 연산은 간격 교차에 기본을 두고 있다. 이 조인 방법은 타임스탬프 값을 갖는 투플이 조인되면 대부분의 경우 타임스탬프 값들이 교차되어 나타나는데 이는 조인열의 속성 값이 같을 때 결합되는 관계형 데이터베이스의 동치 조인과는 대조적이다. 시간 조인 연산은 등등 조건식보다는 부등 조건식에 근거하여 수행된다는 점과 시간 데이터베이스양이 험저히 크다는 점 때문에 시간 조인 연산은 관계 조인 연산보다 수행이 복잡하다[6]. 시간 조인은 기존의 관계형 데이터베이스와는 달리 조인 연산 단계 이전에 데이터 분할의 최적화가 시스템의 처리 성능을

결정한다[8]. 시간 교차 조인인 경우 데이터 분할은 단편들 사이에 중복되는 투플을 생성하며, 분리가 아닌 단편들로 구성된다. 이는 분할 시간 조인 처리시 상당한 오버헤드를 부른다. 효과적으로 오버헤드를 줄이고 시간 조인 처리의 성능을 개선하기 위해서는 중복되는 간격 수를 최소화하는 분할점을 찾는 것이다. 이와 같은 간격 분할과 시간 조인에 관련된 연구를 살펴보면 다음과 같다. 초기 시간축 상의 최대값과 최소값의 차이를 최소 시간인 m 개의 크로노으로 분할하는 균등 분할은 분할 방법중 가장 간단하며 분할 수행 시간이 $O(n)$ 으로 빠르다는 장점이 있지만 분할된 단편내에 많은 중복이 발생하며 적재된 단편내 간격 조절이 불가능하다는 한계를 갖고 있다. 특히, 병렬 조인 연산시 한 릴레이션의 데이터가 비대칭[20]인 경우 심각한 성능 저하를 초래한다. 또한, 디스크의 용량이나 메모리가 허용되는 범위내에 단편내의 간격수를 제한하고 분할선을 좌에서 우로 순차적으로 간격을 채워나가는 언더플로우 분할은 균등 분할보다 중복되는 간격수를 감소시키고 적재되는 간격수 조절이 가능하지만 간격을 최소화 하는데는 효율성이 떨어졌다. 분할 기반 자연 조인[2]은 처리된 단편들을 캐시 메모리에 저장하고 다른 부분 조인을 수행한 후 캐시 메모리에 임시 저장된 부분 결과를 다시 적재하여 부분 조인을 반복적으로 수행하는 순차처리 방법이므로 디스크와 캐시 메모리간의 잦은 이동시간으로 인한 전반적인 입출력 성능의 저하가 문제점으로 지적되었고 병렬처리 환경에서는 적합하지 않았다. 분할에 기반한 최초의 병렬 시간 조인[8]은 투플의 중복을 부분적으로 감소시키고 병렬 처리 환경의 이론적인 방법만 제시하였을 뿐 데이터를 실험적으로 평가하지 못했으며 시간 교차 조인의 경우에는 적용이 불가능하였다. 릴레이션을 상호 독립적인 단편으로 분할하여 간격 데이터를 점(point)에서 2차원 격자(grid)로 사상시킨 공간 분할 시간 조인(spatially partitioned temporal join)[13]은 분할된 단편이 부분 조인되는 단편이외에도 다른 단편과도 조인이 되어 전체 조인 수행에 많은 비용이 소비되었으며, 투플의 부분적인 오버헤드를 감소 시킬 수는 있으나 중복 자체를 제거하는 추가적인 조인 비용이 요구되었다. 제3장에서는 이 논문의 효율적인 전개를 위하여 시간의미와 표현방법, 시간 연산 등 최소 분할 전략에 필요한 정의를 기술한다.

3. MIP 알고리즘

3.1 최소 분할 전략

시간 데이터베이스의 최소 분할 전략에서 연속적인

시간을 표현하기 위해서는 이산 시점, 시간 간격, 그리고 시간을 집합화하기 위한 요소가 있어야 한다[21]. 또한, 조합(collection)은 집합과는 달리 중복되는 원소를 포함할 수 있으므로 간격들의 모임 C를 조합이라하고, 이를 다음과 같이 정의한다.

【정의 1】 시간은 하나의 점으로 표현한다.

$$T = \{ t_{-\infty}, \dots, t_0, t_1, \dots, t_{now}, t_{now+1}, \dots, t_{\infty} \}$$

여기서 T는 전 순서 이산 시점들의 셀수 있는 무한 집합이며 t는 연속적으로 증가하는 시점을 표현한 것으로 t_{now} 는 현재시점을 나타내는 시간단위(chronon: 초, 분, 시, 일, 주, 월, 분기, 반기, 년 등)이며, t_{∞} 은 최소 시간점, $t_{-\infty}$ 은 최대 시간점을 나타낸다.

【정의 2】 시간 간격(time interval : TI)은 두 사건(event) 사이의 기간을 의미한다. 이 간격은 정의1로 규정된 연속적인 두 시점 t_{is} 와 t_{ie} 의 순서쌍의 집합이다.

$$\text{즉, } TI = \{ t_{i0}, t_{i1}, \dots, t_{is}, t_{ie} \}$$

여기서 t_{is} 는 (t_{is}, t_{ie}) , $[t_{is}, t_{ie}]$, $[t_{is}, t_{ie}]$ 또는 $(t_{is}, t_{ie}]$ 이며, $1 \leq i \leq n$ 이다. 세부적으로 "["와 "]"는 인접시점을 포함하고, "("와 ")"는 인접시점을 포함하지 않는다. 또한 t_{is} 는 시작시점을 나타내고, t_{ie} 는 종료시점을 나타내며, 그리고 이들은 시간의 전체 집합 T에 존재하는 시점들이다.

【정의 3】 시간 범위(range) T_C 는 간격들의 조합인 C에 존재하며 간격들에 의해 커버(cover)되는 시간 도메인의 일부이고 다음과 같이 표현된다.

$$\cdot T_C = \bigcup_{i=1}^n C_i, (i=1, \dots, n)$$

【정의 4】 간격의 시작시점 집합 S_C 와 종료시점 집합 E_C 는 다음과 같다.

- $S_C = \{ t_{is} : \exists t_{is} \in T_C \}$, 여기서 $[t_{is}, t_{ie}] \in C$
- $E_C = \{ t_{ie} : \exists t_{is} \in T_C \}$, 여기서 $[t_{is}, t_{ie}] \in C$

【정의 5】 분할 P = $\{p_1, \dots, p_{m-1}\}$ 는 순서화된 분할점들의 집합이며 범위 T_C 를 m개의 시간 간격 (p_{j-1}, p_j) ($j=1, \dots, m$)으로 분할한다.

최소 간격 분할점을 결정하기 위해 언더플로우 방법과 동일하게 임의의 단편내 허용되는 간격수를 'X' 개로 제한하여 문제의 해결 방법으로 접근한다. 언더플로우 방법은 분할선을 좌에서 우로 이동하면서 단편내에 간격수가 X개를 초과하지 않는다면, p_1 까지 적재된 누적 간격수를 계산하여 p_1 을 분할점으로 선택하고, 또 다른 분할점을 찾기 위해 다음 단편으로 이동하여 위 조건을 만족하면 p_2 를 분할점으로 선택한다. 위 과정을 반복하여 분할점을 p_3, \dots, p_m 을 결정한다. 임의의 간격 $g \in C$ 가 단편된 F_i 에 들어갈 필요충분 조건은 g 가 F_i ($i=1, \dots, m$)

에 대응되는 범위인 $(p_{i-1}, p_i]$ 를 교차하는 것이며, 분할선 p에서의 중복수 $D_C(p)$ 는 p를 기준으로 교차하는 간격들을 모은 것으로 식(1)과 같다. 또한, 분할점 p_i 를 기준으로 적재되는 단편수 F_k 는 이전까지 분할된 단편내의 간격수를 누적한 값이며 식(2)와 같다.

$$D_C(p) = |\{g \in C : t_{is} \leq p < t_{ie}\}| \quad (\text{분할점 } p \text{의 중복수}) \quad (1)$$

$$F_k = |\{g \in C : [t_{is}, t_{ie}] \cap (p_{k-1}, p_k] \neq \emptyset\}| \quad (k=1, \dots, m) \quad (2)$$

최소 분할점을 찾기 위한 방법은 그림1처럼 하나 이상의 간격이 종료하는 좌측의 분할선으로 이동시키면 $(p_{i-1}, p_i]$ 개의 단편으로 분할하면서 중복을 최소로하는 분할점을 종료시점에서 결정된다는 것을 보장할 수 있다. 그러므로, 정리1은 최소 분할 P가 임의의 단편내 하나 이상의 종료시점이 존재함을 증명하며 정리2에서는 분할선을 가장 가까운 종료시점으로 이동한다면 새로운 분할도 최소 분할이라는 것을 증명한다.

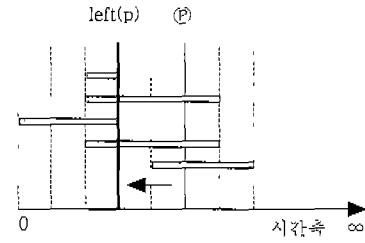


그림 1 분할선 P를 좌측으로 이동하여 가장 가까운 종료시점 left(p)를 탐색

【정리 1】

만약에 분할 $P = \{p_1, \dots, p_{m-1}\}$ 가 $p_{k-1} < p_k$ ($k=2, \dots, m-1$)에 서 간격의 최소 분할이면 임의의 $k=1, 2, \dots, m-1$ 에 대해 $p_{k-1} < e \leq p_k$ 을 만족하는 종료시점 $e \in E_C$ 가 반드시 존재한다. (증명)

어떤 k ($k=1, 2, \dots, m-1$)에 대해서 $p_{k-1} < e \leq p_k$ ($k=2, \dots, m-1$)인 종료시점 e 가 존재하지 않는다고 가정하면, F_k 에 속한 모든 간격은 F_{k+1} 에도 속한다. 즉, $F_k \subseteq F_{k+1}$ 이다. 하지만 $F_k \neq \emptyset$ 이므로 $D_C(p_k) > 0$ 이 된다. 이제 분할 P' = $\{p_1, \dots, p_{k-1}, p_{k+1}, \dots, p_{m-1}\}$ 을 생각해보자. 분할 P' 은 모든 $i=1, \dots, k-1, k+1, \dots, m-1$ 에 대해서 $|F_i| \leq X$ 인 새로운 단편 $F_1, \dots, F_{k-1}, F_{k+1}, \dots, F_{m-1}$ 을 얻는다. 또한,

$$\begin{aligned} \sum_{p \in P} D_C(p') &= \sum_{i=1}^{k-1} D_C(p_i) + \sum_{i=k+1}^{m-1} D_C(p_i) \\ &< \sum_{i=1}^{m-1} D_C(p_i) \\ &= \sum_{p \in P} D_C(p) \end{aligned}$$

이므로 P 는 최소 분할이 될 수 없다. 이것은 가정에 모순이다.

따라서, 임의의 k 에 대해서 $p_{k-1} < e \leq p_k$ 인 $e \in E_C$ 가 항상 존재한다. \square

【정리 2】

분할 $P = \{p_1, \dots, p_{m-1}\}$ 가 간격의 최소 분할이면 새로운 분할

$\mathcal{P} = \{\text{left}(p_1), \dots, \text{left}(p_{m-1})\}$ 도 최소 분할이다.

(증명)

분할점들은 범위 T_C 에 속하는 점들임을 가정한다. $P \subseteq E_C$ 이면 모든 $i=1, \dots, m-1$ 에 대해서 $p_i = \text{left}(p_i)$ 이므로 명백하다. $P \not\subseteq E_C$ 인 경우 $p_k \notin E_C$ 인 p_k 를 P 에서 임의로 선택하면 $\text{left}(p_k) < p_k$ 이고 또한 정리1에 의해 $p_{k-1} < \text{left}(p_k)$ 이다.

이제 새로운 분할 $\mathcal{P} = \{p_1, \dots, p_{k-1}, \text{left}(p_k), p_{k+1}, \dots, p_{m-1}\}$ 과 그에 따른 새로운 단편 $\bar{F}_1, \dots, \bar{F}_{m-1}$ 을 얻을 수 있다.

그런데

$$D_C(\text{left}(p_k)) = D_C(p_k) - \sum_{t \in \text{left}(p_k)} \sum_{t \leq p_i} S_C(t) \leq D_C(p_k)$$

이므로,

$$\begin{aligned} \sum_{p \in \mathcal{P}} D_C(p) &= D_C(p_1) + \dots + D_C(p_{k-1}) + D_C(\text{left}(p_k)) + D_C(p_{k+1}) \\ &\quad + \dots + D_C(p_{m-1}) \leq D_C(p_1) + \dots + D_C(p_{k-1}) + D_C(p_k) \\ &\quad + D_C(p_{k+1}) + \dots + D_C(p_{m-1}) \\ &= \sum_{p \in P} D_C(p) \end{aligned}$$

또한, $p_{k-1} < \text{left}(p_k) < p_k < p_{k+1}$ 이므로 모든 $j=1, \dots, k-1, k+2, \dots, m-1$ 에 대해서 $\bar{F}_j = F_j$ 이고, 분명하게 $|\bar{F}_k| \leq |F_k| \leq X$ 가 성립하고, 또한 $\text{left}(p_k)$ 의 정의에 의해 $|\bar{F}_{k+1}| = |\bar{F}_{k+1}| \leq X$ 가 성립한다. 따라서 \mathcal{P} 도 최소 분할이다. 만약 $\mathcal{P} \subseteq E_C$ 이면 증명이 끝나고, 그렇지 않은 경우 위 과정을 반복해 나가면 결국 E_C 에 속하는 점들로만 구성된 최소 분할 \mathcal{P} 를 구할 수 있다. \square

3.2 MIP 알고리즘의 구현

분할을 최소로 하는 분할점들은 앞 절에서 기술했듯이 시작시점과 종료시점의 집합에서 결정되지만 시작시점보다는 종료시점에서 중복을 최소로하는 분할점이 결정됨을 알 수 있었다. 분할점들을 q_1, \dots, q_n ($q_i < q_{i+1}$, $i=1, \dots, n-1$)으로 표현하고 q_1 에서 시작하여 q_n 에서 종료한다고 하면 임의의 q_i 는 q_1 전 까지 분할된 주기 중에서 최소 시간점인 m 개의 크로노스으로 분할하여 단편의 간격수가 제한된 X 개를 넘는 q_i 는 제외되고 X 개 보다 작은 분할점들만 선택된다. 알고리즘의 수행을 위해 End_Vector 를 이용해 각 시간 간격들의 종료시점을 수집해놓은 다음 이를 이용하여 알고리즘을 수행한다. 선택된 분할점들은 각각의 중복수 $D_C(p)$ 와 대응되는 누적 분할

점 $c(i)$ 를 합한 값 중에서 가장 작은 값을 취해 최소 분할점으로 결정하고 $fwd(k)$ 에 할당한다. 알고리즘은 임의의 분할점 q_1 전 까지 중복되는 수를 최소로 하는 분할점이 선택되었다면 순차적으로 최소 분할점인 $fwd(q_n)$, $fwd(fwd(q_n)), \dots$ 의 값이 결정된다. 간격 (q_i, q_{i+1}) ($i < n$)을 교차하면서 분할점 q_i 가 존재하지 않는다면 q_i 는 간격을 최소로 하는 분할점이 될 수 없다. End_Vector 는 연산 대상 데이터들의 종료시점 위치를 갖고 있는 1차원 배열로서 알고리즘의 순환부를 결정하는 q_i 의 실제값을 갖고 있다. $c(i)$ 의 초기값은 0으로 설정하며 1번째까지 분할된 주기중에서 $c(i)$ 의 값이 0인 경우는 이전까지의 누적 값이 없는 것이므로 최소 분할점에서 제외된다. tmp_bnd 는 조건문 내에서 중복수와 이전까지의 누적 중복수의 합을 계산하며, min_bnd 는 tmp_bnd 와 값을 비교하여 연산 과정에서 최소 중복수를 유지하는 변수이다. 상수 MAX_Cardinality 는 연산 대상이 되는 최대 간격수를 의미하며

각 종료시점에서 적재 단편수의 계산된 값을 갖는 행렬 $F_k(i, j)$ 는 식(2)에 의해 계산된다. break_ndx 는 $F_k(i, j)$ 의 적재된 간격수 중에서 X 개를 넘는 간격수를 배제하기 위한 변수로 사용된다. 그림2의 MIP 알고리즘에서 단계3~5는 최대 간격수를 min_bnd 변수에 할당하

```

/* c[i]의 초기화 */
단계1  c(0) = 0;
단계2  fwd_cnt = 1;

/* 간격 종료시점들의 수만큼 반복 */
단계3  for ( i = 1 ; i <= n ; i++ ) {
단계4  c(i) = NULL;
단계5  min_bnd = MAX_Cardinality;

/* 임의의 종료시점 i 번째에서 i 이전까지 적재된 단편내의
   간격수가 X 개를 넘지 않는 간격수와 이전까지의 누적
   중복수의 합이 최소인 간격의 종료시점을 결정 */
단계6  for ( j = break_ndx ; j < i ; j++ ) {
단계7  qj = End_Vector[j];
단계8  if( F_k(i, j) <= X ) {
단계9  tmp_bnd = D_C(qj) + c(j);
단계10  if( tmp_bnd < min_bnd ){
단계11  min_bnd = tmp_bnd;
단계12  if( min_bnd > 0 ) {
단계13  c(i) = qj;
단계14  }
단계15  }
단계16  }
단계17  }

/* 최소 중복이 존재하는 i번째 간격의 종료시점을 최소
   분할점으로 결정 */
단계18 if( c(i) != NULL ) {
단계19 fwd[fwd_cnt++] = c(i);
단계20 }
단계21 }
```

그림 2 MIP 알고리즘

여 초기화시키고 간격 종료시점들의 수만큼 반복한다.
단계6~17까지는 임의의 종료시점 i 번째에서 i 이전까지 적재된 단편내의 간격수가 X 개를 넘지 않는 간격수와 이전까지의 누적 종복수의 합이 최소인 간격의 종료시점을 결정하는 반복문이며 최소 분할점이 결정되면 단계18~20에서 다음 최소 분할점을 결정하기 위해 위 과정을 반복한다.

이상의 조건하에서 수행되는 알고리즘은 분할점을 결정하는 기준으로 시간 간격의 종료시점만을 이용하여 결정함으로서 중복되는 간격수를 최소화시켜 병렬 조인 연산시 불필요한 투플의 비교 횟수를 줄이고 시스템 성능을 향상시킬 수 있으며 알고리즘에 대한 자세한 내용은 그림 2에 정리하였다.

4. 성능 평가

4.1 시나리오 환경

이 논문에서 제안한 알고리즘은 C언어를 사용하여 구현하였으며 성능평가를 위해 사용된 그림 3의 시나리오는 임의의 간격을 20개로, 간격의 시작시점과 종료시점은 계산의 편의를 위하여 정수값으로 사용하였다. MIP 알고리즘과의 성능 비교를 위해 기준 분할 기법으로 사용된 균등 분할, 언더플로우 분할을 선정하였다. 균등 분할은 5개의 크로논으로 분할하였으며, 언더플로우와 MIP는 단편내 최대 간격수 X 를 10개로 제한하였다. 또한, 조인 연산시 투플의 비교 횟수는 별별 조인 연산의 대칭 분할[2]에 적용하였다. 전체적인 조인 성능 기준은 결정된 분할선이 간격의 중복을 최소로 해야하며 여러개의 분할선 중에서도 단편과의 조인 연산을 고려하여 가장 적절하게 결정되는 것에 두었다.

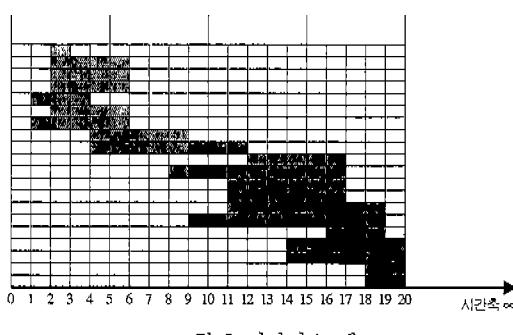


그림 3 시나리오 예

그림 4는 시간축을 $m=5$ 인 크로노으로 분할하여 단편이 4개 생성된 균등 분할의 예이며, 그림 5는 단편내 간

격수를 10으로 제한하여 생성된 언더풀로우 예이다. 그림 4에서 보듯이 단편내 충복되는 투풀과 각 단편내 생성된 투풀의 수가 그림5보다 많음을 알 수 있으며, 이는 조인 연산시 불필요한 투풀의 비교를 수행하므로 그림4에 비해 전반적으로 조인 질의 성능이 저하된다. 또한, 단편내의 투풀 비교 횟수를 구하면 그림 4는 $9^2 + 9^2 + 9^2 + 11^2 = 364$, 그림5는 $10^2 + 10^2 + 9^2 + 7^2 = 330$ 이 되어 언더풀로우 분할이 균등 분할보다 투풀 비교 횟수가 감소되어 개선된 분할 기법임을 알 수 있다.

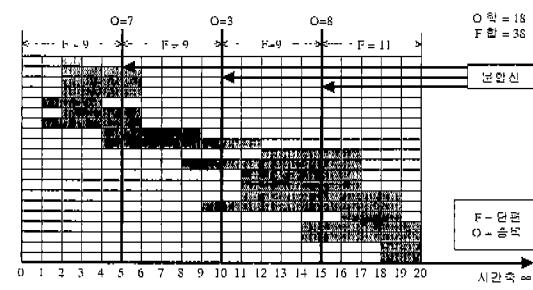


그림 4 균등 분할 시나리오

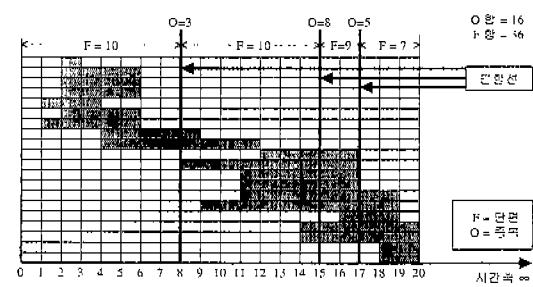


그림 5 업더플로우 불활 시나리오

4.2 MIP 알고리즘의 성능평가

MIP 알고리즘을 평가하기 위하여 앞 절의 시나리오를 이용한다. 시간 조인 처리의 성능을 결정하는 최소 간격 분할은 3.1절의 최소 분할 전략에서 기술했듯이 시작시점과 종료시점 중에서도 간격의 종복을 최소로하는 분할 점은 반드시 종료시점에서 찾을 수 있었다. 그럼 3에서 분할선을 좌에서 우로 이동시켜 하나 이상의 간격이 종료하는 점선의 위치에서 8개의 종료시점 즉, $n=8$ 인 {3, 4, 6, 9, 12, 17, 19, 20}을 구할 수 있다. 시간축에서 8개의 분할점을 3은 q_1 , 4는 q_2 계속해서 20은 q_8 로 대응을 시키며, $q_0=0$ 은 초기값으로 사용한다. 시나리오에서 단편 내에 적재된 간격수 $F_k(j, i)$ 는 다음과 같이 계산된다.

$q_1=3$ 일 때 이전까지의 적재된 누적 간격수는 7개, $q_2=4$ 는 9개이며, 이어서 $q_3=20$ 은 20개를 얻는다. 이 과정에서 q_1 부터 q_8 까지 적재된 간격수를 계산하였다. 계속해서 라운드 로빈 방식으로 간격수를 구하면 q_3 은 8개, q_5 은 8 개, ..., q_8 은 19개를 구할 수 있으며 최종적으로 q_8 은 4개를 구해 모든 경우의 적재된 간격수를 계산하여 이를 표1에 정리하였으며 음영색 부분은 적재된 간격수가 10을 초과하는 값으로 최소 분할점 결정시 제외된다.

표 1 시나리오 그림3의 적재된 단편 $F_k(i, j)$ 의 간격수

$F_k(i, j)$	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
3	7	9	9	11	15	17	19	20
4		8	8	10	14	15	19	19
6			7	9	13	15	18	18
9				4	8	11	13	13
12					7	10	12	12
17						9	11	11
19							7	7
20								4

간격 분할의 최소점들인 8개의 종료시점 $\{3, 4, 6, 9, 12, 17, 19, 20\}$ 은 현재 분할점 위치에서 중복되는 수와 분할점까지 적재된 간격수를 합하여 10을 초과하는 간격 수는 제외되므로 각각의 최소 분할점 결정은 $i=1$ 부터 8까지 부여하여 계산한다. 또한, 최소 중복수를 결정하기 위한 내부 순환 부분에서 중복수와 단편수의 합이 0인 경우는 간격 최소 분할점에서 제외된다. 순차적으로 이전의 누적 정보를 유지하여 계산되는 누적수 $c(i)$ 는 이전까지 간격의 중복수와 누적수를 합한 값 중에서 가장 작은 최소값을 선택하여 분할점을 결정한다. 결정된 최소 분할점의 정보는 최종적으로 함수 $fwd(q_i)$ 에 할당 한다. 이를 토대로 간격 분할의 구체적인 최소 분할점을 다음과 같이 계산된다.

$i=1 \Rightarrow q_1=3, D_c(q_1)=6, j=\{0\}, q_1$ 전에 중복되는 수가 없으므로 $c(q_1)=0$ 이고 누적 분할점의 합 $c(1)=\min\{D_c(q_0)+c(0)\}=0$ 이며, $fwd(q_1)=0$ 을 얻는다.

$i=2 \Rightarrow q_2=4, D_c(q_2)=7, j=\{0,1\}, c(2)=\min\{D_c(q_0)+c(0), D_c(q_1)+c(1)\} = \min\{0, 6\}$. 이 중에서 최소값을 선택하므로 0이 결정된다. 즉, $c(q_2)=0$, $fwd(q_2)=0$ 동일한 방법으로 $i=3$ 까지 모두 0을 출력한다.

$i=4 \Rightarrow q_4=9$ 일 때 표1에서 $F_k(1, 4)=11$ 은 10을 초과하므로 제외된다. 그러므로 $j=\{1,2,3\}$
 $D_c(q_4)=9, c(4)=\min\{D_c(q_1)+c(1), D_c(q_2)+c(2), D_c(q_3)+c(3)\}=\min\{6, 7, 2\}$ 값 중에서 최소 값은 $D_c(q_3)+c(3)=2, q_3=6$ 이 선택되고 $fwd(q_3)=6$ 을 얻는다.

계속해서

$i=8 \Rightarrow q_8=20$ 일 때 표1에서 $F_k(1, 8), \dots, F_k(8, 8)$ 값들 $\{20, 19, 18, 13, 12, 11, 7, 4\}$ 중에서 10을 초과하는 값은 제외되므로 $\{4, 7\}$ 에 해당되는 값인 $j=\{6, 7\}$ 이 결정된다.

$D_c(q_8)=0, c(8)=\min\{D_c(q_6)+c(6), D_c(q_7)+c(7)\}=\min\{10, 14\}$ 값 중에서 최소 값은 $D_c(q_6)+c(6)=10, q_6=17$ 이 선택되고 $fwd(q_6)=10$ 을 얻는다.

이를 정리하면 그림 6과 같으며 그림 7에서 간격 분할을 최소로 하는 분할점들이 결정되었다. 또한, 단편내의 투풀 비교 횟수는 $9^2 + 4^2 + 10^2 + 7^2 = 246$ 이 되어, 기존의 간격 분할 방법보다 비교 횟수가 현저히 감소되었음을 알 수 있다. 시나리오 예를 통한 논리적인 성능 평가 결과 표 2의 균등 분할, 언더풀로우 분할의 단편합에 비해서는 각각 21.1%, 16.7%, 중복합에 대해서는 44.4%, 37.5%, 그리고 조인 연산 비교 횟수에 대해서는 32.4%, 25.4%의 처리 비용을 감소시켜 제안된 알고리즘이 모든 경우에 성능이 우수함을 확인하였다. 그림8은 위 값을 근거로 처리 비용 감소율을 히스토그램으로 표현한 것으로 제안된 알고리즘은 시간 간격의 종료시점을 대상으로 간격 분할 연산을 수행함으로서 기존 분할 방법보다 연산량을 훨씬 줄이면서 효과적인 간격 분할을 수행하였다. 따라서 제안된 알고리즘을 시간 조인 처리에 적용한다면 효율적인 간격 분할을 할 수 있으며 전체적인 조인 성능을 향상 시킬 수 있다.

i	q_i	$D_c(q_i)$	j	$fwd(q_i)$	$c(q_i)$
1	3	6	{0}	$q_0=0$	0
2	4	7	{0,1}	$q_0=0$	0
3	6	2	{0,1,2}	$q_0=0$	0
4	9	3	{1,2,3}	$q_3=6$	2
5	12	6	{3,4}	$q_3=6$	2
6	17	5	{4,5}	$q_4=9$	5
7	19	4	{6}	$q_6=17$	10
8	20	0	{6,7}	$q_6=17$	10

그림 6 최소 분할 $P=\{q_3, q_4, q_6\}=\{6, 9, 17\}$ 를 갖는 MIP 시나리오

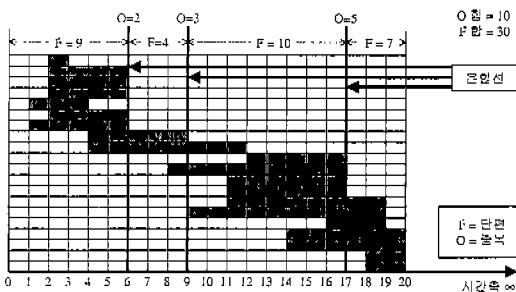
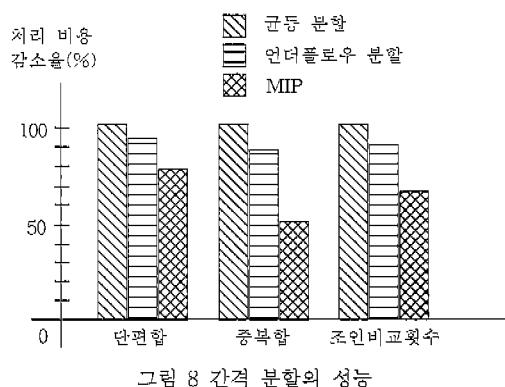


그림 7 MIP 시나리오

표 2 간격 분할의 논리적인 성능 비교

구분	단편합	충복합	조인 연산 비교 횟수
균등 분할	38	18	364
언더플로우 분할	36	16	330
MIP	30	10	246



5. 결 론

단순한 조인 접근 방법은 병렬 처리, 해싱과 균형 인덱스 트리 구조에 기반하는 다양한 기술적 처리과정의 오버헤드를 부르며, 관계형 데이터베이스와는 달리 시간 데이터베이스는 타임스탬프 속성 값으로 인하여 조인 연산시 간격 분할을 위한 최적의 전략을 필요로 한다. 이 논문에서는 시간 데이터베이스의 병렬 조인 질의 성능을 개선하기 위해 시간 조인 연산을 위한 시간 간격을 분할하는 최소 분할 기법을 제안하였고, 이 논문에서 제안된 MIP 알고리즘은 기존의 간격 분할 방법인 언더플로우 분할의 개선을 보이기 위해 간격을 분할하는 예로서 성능이 향상됨을 보였다. 제안된 MIP 알고리즘은

조인 연산 단계 이전에 간격 분할의 최소 분할점을 결정하는 이론적인 배경이 될 뿐만 아니라 기존에 사용된 분할 기법보다 성능이 우수하여 시간 조인 연산을 수행할 경우 효과적이고 빠른 입출력 처리가 가능하다.

참 고 문 헌

- [1] Snodgrass, R. The temporal query language TQuel. ACM Transactions on Database System, 12(2):247-298, 1987.
- [2] M. Soo, R. Snodgrass, and C. Jensen. Efficient Evaluation of the Valid-Time Natural Join. In Proc. of the 10th Int. Conf. on Data Engineering, Houston.
- [3] C. Breiteneder, S. Gibbs, and D. Tsichritzis. "Modelling of Audio/Video Data", In Proceedings of the 11th International Conference on the Entity-Relationship Approach. Karlsruhe, Germany, pp. 322-339, Oct. 1992.
- [4] E. Ardizzone, M. L. Cascia, "Automatic Video Database Indexing and Retrieval". Multimedia Tools and Applications, Vol.4, No.1, pp. 29-56, Jan. 1997.
- [5] N. Lorentzos and Y. Mitsopoulos. SQL Extension for Interval Data. Technical Rep. 105, Informatics Lab., Agricultural University of Athens, 1994.
- [6] R. Snodgrass, editor. The TSQL2 Temporal Query Language. Kluwer, Sept. 1995.
- [7] Clifford, J. Dyreson, C. Snodgrass, R., Isakowitz, T. and Jensen, Now in TSQL2. A TSQL2 Commentary. C.(1994).
- [8] T. Leung and R. Muntz. "Temporal Query Processing and Optimization in Multiprocessor Database Machines". In Proc. 18th Int. Conf. on VLDB, Vancouver, Canada, pp. 383-394, Aug. 1992.
- [9] DeWitt, D. and Gerber, Multiprocessor hash-based join algorithms. In Proc. of the 11th Int Conf. on Very Large Data Base(VLDB), Stockholm, Sweden, pages 151-164. R.1985.
- [10] Lo, M.-L. and Ravishankar, C. Spatial Hash-Joins. In Proceedings ACM SIGMOD Conference on Management of Data, Montreal, Canada, pages 247-258. 1996.
- [11] X. Jeffrey, Yu and Kian-Lee Tan, Scheduling Issues in Partitioned Temporal join, Joint Computer Science Technical Report Series, May 1995
- [12] Il. Gunadhi and A. Segev. Query Processing Algorithms for Temporal Intersection Joins. In Proc. of the IEEE int. Conf. on Data Engineering, pages 336-344, 1991.

- [13] H. Lu, B.-C. Ooi, and K.-L. Tan. On Spatially Partitioned Temporal Join. In Proc. of the 20th Internet Conf. on Very Large Data Bases(VLDB), Santiago de Chile, pages 63-113, Mar. Sept. 1994.
- [14] H.Zhou. Two-stage m-way graph partitioning. Parallel Computing, 19(12) pp. 1359-1373, Dec. 1993.
- [15] P. Mishra and M. Eich. Join Processing in Relational Databases. ACM Computing Surveys, pages 63-113, Mar.1992.
- [16] R. Elmasri, G.Wuu, and V. Kouramajian. The Time Index and the Monotonic B+-tree. In Tansel and et al. [10], chapter 18, pages 433-456.
- [17] H. Gunadhi and A. Segev. Efficient Indexing Methods for Temporal Relations. IEEE Transactions on Knowledge and Data Engineering, 5(3):496-509, June. 1993.
- [18] Rana, S. and Fotouhi, F.(1993). Efficient Processing of Time-Joins in Temporal Data Bases. In Proc. of the 3rd Internat. Symposium on Database System for Advanced Applications, pages 427-432.
- [19] 김동호, 이인홍, 류근호, "주기억장치에서 시간지원 데이터베이스의 집계함수 설계 및 구현" 한국정보과학회 논문지, 제21권, 제8호 pp.1405-1415, 1994.
- [20] J. David, F.Jeffrey, A. Donovan, S.Seshadri. Practical Skew Handling in Parallel Joins. In Proc. 18th VLDB, pp.27-40, Aug.1992.
- [21] R. Snodgrass, K. Nick, "Aggregate in TSQL2" The TSQL2 Language Design Committee, Mar. 1994.



류 근 호

1976년 충남대 전산학과 학사, 1980년 연 세대학교 산업대학원 전산전공(공학석사), 1988년 연세대학교 대학원 전산전공(공학 박사), 1976년 ~ 1986년 육군 군수지원 사 전산실(ROTC 장교), 한국전자통신연 구소(연구원), 한국방송통신대학교 전산 학과 조교수, 1989년 ~ 1991년 Univ. of Arizona 연구원 (TempIS Project), 1986년 ~ 현재 충북대학교 컴퓨터과학과 교수, 관심분야는 시간 데이터베이스, 시공간 데이터베이 스, 객체 및 지식데이터베이스 시스템, 지식기반 정보검색 시스템, 데이터베이스 보안, 데이터 마이닝, Bio-Informatics



김 홍 기

1961년 연세대학교 수학과(학사), 1975년 연세대학교 교육대학원 응용수학 교육학 과(교육학 석사), 1985년 중앙대학교 대 학원 응용수학(이학박사). 1980년 ~ 현 재 충북대학교 컴퓨터과학과 교수. 관심 분야는 퍼지 이론, 정보통신



아 광 규

1985년 동국대학교 수학과(학사), 1991년 동국대학교 대학원 응용수학(이학석사). 1995년 ~ 1998년 충북대학교 대학원 전 자계산학과 박사과정 수료, 1996년 ~ 현재 신홍대학 컴퓨터정보계열 조교수. 관심분야는 시간 데이터베이스, 퍼지 이

론, 정보검색



신 예 호

1996년 군산대학교 컴퓨터과학과 학사. 1998년 충북대학교 대학원 전자계산학과 (이학석사), 2000년 8월 ~ 충북대학교 대학원 전자계산학과 박사과정 수료, 관

심분야는 시공간 데이터베이스, 시간 퍼

지 데이터베이스, 능동 데이터베이스, 데이터

마이닝