DBpia

논문 01-26-6T-3 한국통신학회논문지 '01-6 Vol.26 No.6T

유효시간에 기반한 효율적인 시간 조인 처리 전략

정회원 이 광 규*, 김 홍 기**

Efficient Temporal Join Processing Strategy based on a Valid-Time

Kwang-Kyu Lee*, Hong-Gi Kim** *Regular Members*

요 약

시간 조인은 기존의 관계형 데이타베이스와는 달리 조인 연산 단계 이전에 데이터 분할의 최적화가 시스템의 처리 성능을 결정한다. 시간지원 교차 조건인 경우 데이터 분할은 단편들 사이에 중복되는 투플을 생성하며, 서로 소가 아닌 단편들로 구성된다. 이 결과는 분할 시간지원 조인 처리시 상당한 오버헤드를 준다. 효과적으로 오버헤드를 줄이고 시간 조인 처리를 위해서는 중복되는 간격 수를 최소화하는 분할 점을 찾는 것이다. 따라서, 본 논문에서는 시간 조인 처리 전략을 위한 최소 간격 분할(Minimum Interval Partition: MIP)알고리즘을 제안한다. MIP 알고리즘을 예제 시나리오에 적용한 결과 기존에 사용된 분할 기법보다 성능이 우수하며, 시간 조인 처리시 전체적인 비용을 최대한 감소시킬 수 있었다.

ABSTRACT

Optimization of data partition before join operation step decides the disposal performance of system unlike existing relational database of temporal join. In a temporal intersect condition, partitioning data generates overlapped tuples among fragments, and consists of fragments not disjoint. To reduce overhead effectively and to handle temporal join, we have to search for partition point to minimize the numbers of overlapping intervals. Therefore, in this paper I'd like to suggest minimum interval partitioning algorithm for temporal join processing strategy. After applying MIP algorithm to example scenarios, I was able to find more excellent partition method than before, and finally reduce the general cost also.

Ⅰ. 서 론

최근에 인터넷의 확산과 각종 멀티미디어 데이터의 증가와 함께 새롭고 복잡한 데이터 타입들 즉, 공간, 시간, 오디오와 비디오 모델들이 DBMS에서 소개되었다. 관계형 모델에 간격 데이터(interval data)기능을 확장시킨 많은 응용 분야중 IXSQL[1]과 TSQL2[2]는 가장 대표적인 예가 되며 시간 개념을 지원하는 표준 SQL3 형식을 갖게되었다[3]. 하지만 간격 데이터(interval data)는 시간지원 모델에서 질의 처리시 여러 문제점을 제기하는데 그 중 하나는 간격 값(interval value)에서 정의되는 데이터 분할(data partition)이다. 데이터 분할은 물리적 데이터-분할(physical data partitioning), 병렬 질의 처리(parallel query processing), 해싱(hashing)과 인덱스 트리 구조(index tree structure)등 과 같은 기법에서 아주 중요한 역할을 한다[4]. 이들 기법들은 소규모적으로 격리된 일부 데이터의 검색이나 해싱 조인(hashing join), 병렬 조인(parallel join), 병렬 입출력(parallel I/O) 처럼 독립적인 연산으로 나누어서 하나의 큰 연산을 처리하는데 사용된다. 시간 조인 연산은 가장 중요한 관계 연산자이며 데이터 정규화에 기인한 필수적인 연산자로서 질의어 최적화의 관점에서 상당히 중요한 위치를 차지한다. 시간지원 조인 연산은 동등 서술식(equality prediate)보다는 부등 서술식(inequality prediate)에 근거하여 서술된

* 신흥대학 컴퓨터 정보계열 조교수 ** 충북대학교 컴퓨터 과학과 교수
논문번호 : T01011-0511, 접수일자 : 2001년 5월 11일

www.dbpia.co.kr

다는 점과 시간지원 데이터베이스(temporal database)가 현저히 크다는 점 때문에 시간지원 조인 연산은 관계 조인 연산보다 중요하며 또한, 질의어 처리 비용에 훨씬 더 큰 영향을 미친다. 또한, 분할은 병렬 처리와 순차 조인 처리(sequential join processing)에 유용하게 사용되며 타스크를 완료하는데 처리되는 비용을 감소시킬 수 있다[11,15]. 간격 데이터 분할 시 여러 단편들이 생성되고 어떤 특정한 간격 도메인에서 이들 데이터 객체들은 서로 다른 단편을 교차(intersect)한다. 교차는 시간지원 데이터베이스(temporal database)에서 두 개 이상의 타임스탬프(timestamp)를 갖는 데이터 객체들이 중복되는 시간을 갖는 것을 의미하며 교차에 기반을 둔 간격 분할의 가장 큰 문제점은 서로소(disjoint)가 아닌 데이터 단편을 만들어 낸다. 이는 질의 처리시 상당한 오버헤드(overhead)를 주게된다. 그러므로, 간격 분할은 신중하고 타당성 있는 논리 전개가 필요하다. 지금까지 많은 연구가들에 의해 간격 분할 문제를 다루는 알고리즘 설계와 인덱스 구조가 소개되었지만[5,6,7,8,9] 최소 정지점을 구하는 방안이 제시된 것은 없었다. 본 논문에서는 유효시간(valid- time)에 기반한 간격 분할의 문제점을 분석하고 분할 방법 중 가장 일반적인 언더플로우(underflow)방법을[6] 개선하여 간격의 최소 정지점(breakpoint)을 찾는 최소 간격 분할 (Minimum Interval Partitioning:MIP) 알고리즘을 제안한다. 제안된 알고리즘은 시간 조인 처리시 기존의 분할 방법보다 처리 성능을 향상시킬 수 있었다. 논문은 다음과 같이 구성되어있다. 2절은 연구 배경및 목적을, 3절에서는 시간 조인 처리를 위한 간격 분할 알고리즘의 최소 제약조건을 제시하고, 4절은 제안된 알고리즘의 구현을, 5절에서는 제안된 알고리즘을 실험 및 평가하며, 마지막으로 6절에서 결론을 내린다.

Ⅱ. 연구 배경 및 목적

간격 데이터 분할에 관련된 질의 처리 기법중 시간을 갖는 조인 연산(temporal join)은 간격 교차에 기본을 두고 있으며[4,10,15] 타임스탬프 값을 갖는 투플(tuple)이 조인되면 대부분의 경우 타임스탬프 값들이 교차되어 나타나는데 이는 조인열의 속성 값이 같을 때 결합되는 관계형 데이터베이스(relational database)의 동등 조인(equi join)과는 대조적이다. 해시와 병렬 조인 알고리즘은 배타적이고 상호 독립적인 방법으로 조인 연산을 수행하는데 대수적 관점에서 두 릴레이션(relation) R과 Q의 조인 연산은 식(1)과 같이 표현된다[10,11].

$$R \bowtie Q = R_1 \bowtie Q_1 \cup \ldots \cup R_n \bowtie Q_n \quad (1)$$

R_i 와 Q_i는(i=1,...n) 각각 R과 Q의 단편(fragment)들 이고

$$R = \bigcup_{i=1}^{n} ri \ , \quad Q = \bigcup_{i-1}^{n} qi$$가 된다.

이 경우 R과 Q의 단편인 R_i 와 Q_i는 특정한 문자들로 시작하는 속성 값이나 범위(range)를 교차하는 투플을 갖는데 두 릴레이션이 조인 상대를 만나는 위치에서 R과 Q의 투플들이 집중되고 중복되어 나타난다. 그러므로 간격 데이터를 분할하여 얻는 장점은 두 가지로 요약할 수 있다.

- 해시 조인 알고리즘(hash join algorithm)을 이용하여 투플의 비교 횟수를 줄인다[4].
- (1) 식은 그림1 처럼 병렬 조인 알고리즘(parallel join algorithm)을 이용하여 여러 개의 상호 배타적인 부분 조인(partial join)연산으로 나누어 하나의 큰 연산을 실행할 수 있다[10,11,15].

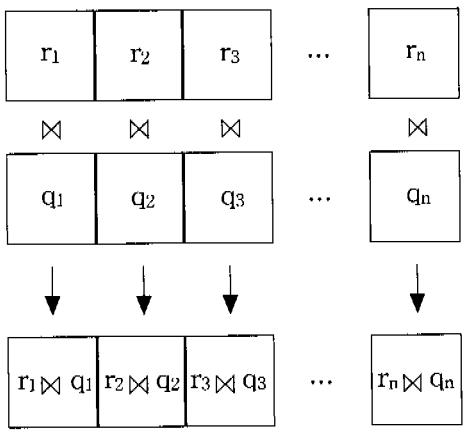


그림 1. $r_n \bowtie q_n$의 분할 조인

하지만 타임스탬프를 갖는 간격이 시간 축(time line)상의 특정 범위에서 교차되어 나타나기 때문에 단편들은 특별한 경우를 제외하고는 관계형 데이터베이스와는 다르게 서로소로 분할되지 않고, 간격이 중복되어 나타나는데 중복되면 시간 축 상에서 최소 정지점(breakpoint)을 결정하기가 어렵다. 즉, 목적은 중복되는 간격을 최소화하는 정지점을 찾는 것이다.

www.dbpia.co.kr

간격 분할의 정지점을 결정하는 방법중 일정 분할(uniform partition)은 주기(span) 시간 축 상의 최대값과 최소값의 차이를 최소 시간 값인 m 개의 크로논(chronon) 으로 분할하는 방법을 사용하였다[6]. 일정 분할은 m 개로 분할하기 때문에 알고리즘의 수행이 간단하고 수행 시간(run time)을 단축시켰지만 중복된 간격 수를 감소시키는데는 효율성이 떨어졌다. 일정 분할의 문제점을 보완한 방법으로 디스크의 용량이나 메모리가 허용되는 범위 내에서 단편의 간격 수를 제한하는 문제가 제기되었다. 즉, 추가적인 매개변수 X를 도입하여 단편내에 간격 수를 초과하지 않는 범위에서 간격들을 라운드 로빈(round robin)방식으로 적재하여 중복을 해결하는 방법인 언더플로우 분할(underflow partition)이 제안되었다[6,10]. 언더플로우 방법은 정지선을 좌에서 우로 이동시키면서 단편내에 간격 수가 X를 초과하지 않는다면, 임의의 p_1을 정지선으로 선택하고, 두 번째 단편으로 이동하여 X를 초과하지 않으면 p_2를 정지점으로 선택한다. 위 과정을 반복하여 정지점들 $p_3,...,p_m$을 구하는 방법이다. 언더플로우 방법이 일정 분할보다 중복된 수를 감소시키며 단편내에 적재된 간격 수를 조절하여 균형을 맞출수 있다는 장점이 있지만 간격의 중복수를 줄이는 정지점을 조절하는 방법이 없다는 것이다. 따라서, 본 논문에서는 언더플로우의 단점인 수행 시간을 O(n)으로 감소시키고 정지점의 조절이 가능한 최소 간격 분할 알고리즘(Minimum Interval Partitioning:MIP)을 제안한다. 또한, 제안된 알고리즘을 시간 조인 처리시 기존의 분할 전략보다 성능이 우수하다는 것을 예제 시나리오를 통하여 분석 평가한다.

Ⅲ. 시간 조인

3.1 시간지원 조인

시간지원 조인은 시간 프로젝트(time-project), 시간 유니온(time-union)등과 같은 시간지원 연산자들 중의 하나에 속한다. 시간지원 연산자는 관계대수의 형태로 표현할 수 있다는 점에서 기존 관계 연산자와 유사하며 주목할 점은 기존의 관계 연산자를 사용하여 시간 질의어를 처리 할 경우에 발생하는 비효과적인 면을 시간 질의어로 표현 할 수 있도록 조건 서술식에 타임스탬프를 관련시킨 시간 서술식(temporal predicate)을 이용한다는 것이다. 시간지원 조인은 일반적으로 동등 서술식만을 지원하는 기존 관계 조인 연산자를 시간 정보를 처리 할 수 있도록 타임스탬프와 부등 서술식을 관련시켜 확장한 연산자이다. 시간지원 조인 연산자에는 T-Join, TE-Join, 그리고 Event-Join의 세가지 유형으로 크게 분류된다[17,18]. TE-Join은 표준 동등 조인과 동일한 동등 조인이다. TE-Join은 조인에 참여하는 릴레이션의 두 개의 투플 x∈R 과 y∈Q에 대하여, 이들의 비시간 속성에 근거한 동등 조인 서술식 P를 만족하고 동시에 이들의 간격이 서로 교차할 때, 이 두 투플은 결합(concatenation)가능하다고 정의한다. 예를 들어 A, B 두 사원이 그림2와 같이 각 도시에서 지정된 기간 동안 업무를 보며 조인 조건은 A의 타임스탬프 값이 B 타임스탬프 값을 교차 즉, 타임스탬프(A) "교차(intersect)" 타임스탬프(B) 한다고 가정하자. 그림2의 릴레이션을 조인 연산을 위해 독립적인 타임스탬프 3 개의 값으로 분할한다면 그림3을 얻으며 이 경우에 분할된 단편들 r_1과 r_2에서 투플들이 중복되어 나타남을 알 수 있다. 투플들이 중복되면 조인 연산시 많은 비용과 중복 자체를 제거하는 추가적인 비용이 소비되므로 시스템 오버헤드의
주원인이 된다. 따라서, 효율적인 시간지원 조인을 수행하기 위해서는 단편내 중복을 감소시키고 수행 시간을 단축시킬 수 있는 새로운 알고리즘이 필요하다.

A 릴레이션

도시	시작	종료
서울	3	8
부산	2	10
청주	4	9
광주	1	8
대구	6	10
인천	1	10

B 릴레이션

도시	시작	종료
의정부	1	4
청주	1	4
대전	7	9
대구	2	5
전주	1	3
춘천	7	10

그림 2. 시간지원 릴레이션의 예

$A_1 \bowtie B_1$

도시	시작	종료
서울	3	8
부산	2	10
광주	1	8
인천	1	10

도시	시작	종료
의정부	1	4
청주	1	4
대구	2	5
전주	1	3

r_1: 간격이 [1,3]을 교차

www.dbpia.co.kr

$A_2 \bowtie B_2$

도시	시작	종료
서울	3	8
부산	2	10
청주	4	9
광주	1	8
인천	1	10

도시	시작	종료
의정부	1	4
청주	1	4
대구	2	5

r_2: 간격이 [4,5]을 교차

$A_3 \bowtie B_3$

도시	시작	종료
서울	3	8
부산	2	10
청주	4	9
광주	1	8
대구	6	10
인천	1	10

도시	시작	종료
대전	7	9
춘천	7	10

r_3: 간격이 [6,10]을 교차

그림 3. 간격을 서로소인 타임스탬프 값으로 분할한 시간지원 조인 예

3.2 기호 정의

간격 데이터를 릴레이션(relation)이나 시간지원 릴레이션의 값 보다는 일반적인 관점에서 접근하기로 한다. 조합(collection)은 집합과는 다르게 중복되는 원소를 포함할 수 있으므로 간격들의 모임을 조합으로 정의하고, 본 논문에서 사용되는 기호를 다음과 같이 정의한다.

【정의 3.1】

- $U = \langle u_1, ..., u_N \rangle$: 간격의 조합
- 시간 간격(time interval) : 주로 지속 시간을 처리해야 하는 스케쥴링(scheduling) 시스템에서 많이 사용하는 방법으로 다음과 같은 기호로 표현한다.
  - $[t_s, t_e] = \{x : t_s \le x \le t_e\}$
  - $(t_s, t_e] = \{x : t_s < x \le t_e\}$

  (t_s:시작시점, t_e:종료시점)
- $T_U = \bigcup_{i=1}^{N} u_i$: 범위(range) T_U는 간격에 의해 커버(cover)되는 시간 도메인(time domain)
- $L_U = [\min T_U, \max T_U]$: 주기 L_U는 간격에 의해 커버되지 않는 도메인의 일부를 포함하며 T_U에는 포함되지 않는다.
- 간격 시작시점의 집합 S_U와 종료시점의 집합 E_U는 다음과 같다.
  - $S_U = \{t_s : \exists t_e \in T_U\}$, 여기서 $[t_s, t_e] \in U$
  - $E_U = \{t_e : \exists t_s \in T_U\}$, 여기서 $[t_s, t_e] \in U$
- 분할 $P = \{p_1, ..., p_{m-1}\}$는 순서화된(ordered) 정지점들의 집합이며 L_U와 T_U를 m개의 시간 $(p_{j-1}, p_j](j=1,...,m)$으로 분할한다.

3.3 최소 분할 전략

언더플로우 방법에 기반하여 간격을 분할하기 위해서는 두 가지 제약조건이 수반되는데[6] 첫 째는 간격 분할에서 허용되는 간격들의 수는 임의의 양수인 X보다 적거나 같고 분할 처리 과정에서 모든 투플은 조인 연산에 참여하여 계산되지만 실제로 타임스탬프를 갖지 않는 모든 속성 값은 무시된다. 하나 이상의 시간지원 릴레이션에서 생성된 간격은 여러개 나타날 수 있고, 생성되는 간격을 최소화하는 것이 문제의 초점이 된다. 두 번째는 주기 간격 L_U의 분할 $P=\{p_1,...,p_{m-1}\}$를 최소화하는 정지점을 찾는 것이다. 임의의 간격 $u \in U$ 가 단편된 U_i에 들어갈 필요충분 조건은 u가 $U_i(i=1,...,m)$에 대응되는 범위인 $(p_{i-1}, p_i]$을 교차하는 것이다. 이를 요약하면 그림4와 같이 표현되며 최소 정지점을 찾기 위한 방법을 그림으로 고찰해보자.

- $U : \langle u_1, ..., u_N \rangle$
- X : 양의 정수
- U의 분할 $P = \{p_1, ..., p_{m-1}\}$는 $\sum_{p \in P} O_R(p)$를 최소화한다.
  - $O_R(p) = |\,[u \in U : u.t_s \le p < u.t_e]\,|$ (정지점 p의 중복 수)
  - $F_R(p_{j-1}, p_j) \le X$ (j 번째 단편 수는 X보다 적거나 같다)
  - $F_R(p_{j-1}, p_j) = |\,\langle u \in U : u \cap (p_{i-1}, p_i] \neq \varnothing \rangle\,|$ $(i=1,...,m: j < i)$

그림 4. 최소 분할 제약조건

그림5에서 정지선(굵게 표시된 수직선)을 하나 이상의 간격이 시작하고 종료하는 점선의 위치로 이동한다면 직관적으로 중복을 최소로하는 정지점을 $S_U \cup E_U$ 내에서 찾을 수 있으나 모든 정지점들이 비교 대상에 참여하므로 수행 시간이 $O(n^2)$가 된다는 단점이 있다. 특히, 투플의 수가 많고 새로운 투플이 데이터베이스에 추가될 때 더욱 큰 문제점으로 평가되었다[8]. 수행 시간을 줄이는 유일한 방법은 정지점들이 시작시점이나 종료시점에 존재하지만 시작시점보다 종료시점에 최소의 정지점이 존재한다는

www.dbpia.co.kr

것을 증명하므로서 해결할 수 있다.

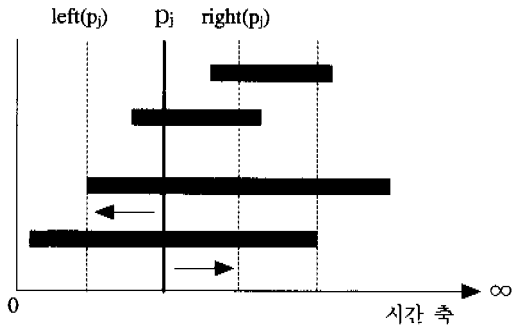


그림 5. 정지선(pj)을 좌,우측으로 이동하여 가장 가까운 시작시점 left(pj)과 종료시점 right(pj)을 찾음

【정리3.1】
만약에 분할 $P = \{p_1,...,p_{m-1}\}$가 간격의 최소 분할이면 $t_{min} \le e \le p_1$이 되는 종료시점 $e \in E(R)$가 반드시 존재한다.

(증명)
증명은 대우(contradiction)방법을 이용하도록 한다. 최소 분할 P가 $t_{min} \le e \le p_1$이 되며 종료시점 집합인 E(R)에 포함되지 않는다고 가정하자. 이 논리는 첫 번째 분할 범위내에 간격의 종료시점이 존재하지 않으므로 두 번째 범위를 교차하는 것을 의미한다. 다시말하면, 첫 번째 단편 R_1의 모든 간격이 R_2 에 포함되고 R_1의 단편수는 최소한 0이 아니며 p_1에서 중복수는 0 보다 크다는 것이다.
즉, $R_1 \subseteq R_2$ $(R1 \neq 0)$, $O_R(p_1) > 0$
이제 분할 $P' = \{p_2,...,p_{m-1}\}$을 생각해보자. 분할 P′은 $|F_R(p_{j-1}, p_j)| \le X$ 에 대해

$R_1 \cup R_2 = R_2$
$R_2 \cup R_3 = R_3$
...
$R_{m-1} \cup R_m = R_m$ (k = 2,...,m) 과 같은 여러 단편들을 생성한다.
그러므로

$$\sum_{p \in P'} O_R(p') = \sum_{k=2}^{m-1} O_R(p_k) < \sum_{k=1}^{m-1} O_R(p_k) = \sum_{p \in P} O_R(p)$$

이되고, P는 최소 분할이 아니며 가정에 모순이므로 결과를 만족한다.

□

Ⅳ. 알고리즘의 구현

분할을 최소로 하는 정지점들은 3.3절 분할 전략에서 살펴본 것 처럼 시작시점과 종료시점의 집합에서 발견된다. 특히, 수행 시간을 O(n)으로 줄이기 위해 종료시점으로 구성된 정지점들을 고려하여 알고리즘을 유도한다. 정지점들을 $q_1,...,q_n$ $(q_i<q_{i+1}, i=1,...,n-1)$으로 표현하고 q_1에서 시작하여 q_n에서 종료한다고 가정하면 임의의 q_i는 q_i전 까지 분할된 주기 중에서 최소 시간점인 m 개의 크로논 $[minT_U, q_i]$으로 나누어 단편의 간격수가 제한된 X개를 넘는 q_i는 무시되고 X개 보다 적은 정지점들만 갖게된다. 선택된 정지점들은 각각의 중복수와 대응되는 누적 정지점을 합한 값 중에서 가장 적은 값을 취해서 $fwd(q_i)$에 할당한다. 정지점을 찾는 q_i를 정리하면 다음과 같다.

- $c(q_i)$: q_i전 까지 생성된 최소 중복의 누적수
- $fwd(q_j)=q_j$, 만약에 q_j가 정지점이면 그 중에서 적은 값을 q_j로 선택

더미(dummy)점 q_0는 알고리즘의 이해를 돕기 위해 초기 값으로 두고 임의의 q_i 전 까지 중복되는 수를 최소로 하는 정지점이 발견되었다면 순차적으로 $fwd(q_n)$, $fwd(fwd(q_n))$,...의 값이 생성된다. 간격 $(q_j, q_i](j<i)$을 교차하면서 정지점 q_i 가 존재하지 않는다면 q_i는 간격을 최소로 하는 정지점이 될 수 없다. 논리적인 실험 결과 알고리즘의 수행시 간으로 단편들의 간격 수를 비교하는데 걸리는 시간 J 와 최소 함수값 $c(q_i)$를 찾는데 걸리는 시간은 O(n)으로 중복을 줄이고 최소 정지점을 찾을 수 있었으며, 알고리즘은 그림6과 같다.

```
/* q0를 더미점(dummy point)으로 사용 */
 O_{R∪Q}(q0) = 0
 c(q0) = 0
 • for i=1 to n do
    J = {j: 0≤j<i ∧ F_{R∪Q}(p_{j-1}, p_j)≤X}
    if J = 0 then output "최소 분할 아님" stop.
    c(qi) = min_{j∈J}{O_{R∪Q}(qj) + c(qj)} /* 최소값 qj
            를 선택 */
    fwd(qi) = qj
 end
```

그림 6. 일반적인 두 릴레이션 R과 Q의 최소 간격 분할 (MIP) 알고리즘

www.dbpia.co.kr

V. 실험 및 평가

본 절에서는 제안된 알고리즘의 효율성을 평가하기 위해 일련의 예제 시나리오를 통해 성능을 평가하였으며, 제안된 MIP 알고리즘과의 성능 비교를 위해 기존 분할 기법으로 사용된 일정 분할, 언더플로우 분할을 선정하였다. 예제 시나리오 릴레이션의 간격 수는 20개로, 일정 분할은 5개의 크로논으로 분할하였으며, 언더플로우와 MIP는 단편내 최대 간격 수 X를 10개로 제한하였다. 또한, 조인 연산시 투플의 비교 횟수는 병렬 조인 연산의 대칭 분할식(1)에 적용하였다. 그림9는 두 회사 R과 Q에서 각 사원들이 지정된 날짜에 프로젝트를 완성하며 시간 조인 조건은 R이 Q를 교차한다고 가정하였다. 일정 분할을 사용하여 시간 축을 m=5인 크로논(chronon)으로 나누면 그림10과 같이 단편이 각 4개 생성되며, 시간 조인 처리한 결과는 그림11과 같다. 일정 분할은 조인에 참여하는 투플의 타임스탬프를 전혀 고려하지 않고 단지 m 크로논으로 나누므로 비용이나 중복과는 무관하다. 그림에서 알 수 있듯이 단편들을 부분 조인한 결과 많은 중복이 발생하며, 투플을 비교한 횟수보다 조인 성공한 비율이 상대적으로 낮은 것을 볼수 있으며, 이 것은 전체적인 시스템의 처리 성능을 감소시키는 원인이 된다. 그림12는 일정 분할보다 성능이 진보된 언더플로우 방법을 사용하여 한 단편내 최대 간격수를 10으로 제한하여 단편들이 생성된 언더플로우 예제이다. 언더플로우 방법은 단편내의 간격수를 X=10으로 제한하고 정지점들을 좌에서 우로 채워나가는 방법이므로 정지점들의 조절이 불가능하다는 단점이 있으며, 일정 분할보다는 중복 및 처리 성능이 향상된 것을 볼 수 있다. 다음은 일정 분할된 시나리오를 제안된 최소 분할 알고리즘에 적용하여 간격의 최소 정지점들을 구해보자. 단편내 최대 간격 수를 10으로 제한하고 릴레이션에서 종료시점을 갖는 정지점들을 모두 구하면 n=8인 {3, 4, 6, 9, 12, 17, 19, 20}을 얻을 수 있고 시간 축 상에서 8개의 정지점을 $q_1,...q_8$로 대응을 시키며, q_0 = 0은 더미점(dummy point)으로 사용하고 단편 내에 적재되는 간격수 $F_R(q_i, q_j)$를 구하면 그림7의 결과를 얻을 수 있다.

$$\begin{pmatrix} 7 & 9 & 9 & 11 & 15 & 18 & 20 & 20 \\ & 8 & 8 & 10 & 14 & 16 & 19 & 19 \\ & & 7 & 9 & 13 & 15 & 18 & 18 \\ & & & 4 & 8 & 11 & 13 & 13 \\ & & & & 7 & 10 & 12 & 12 \\ & & & & & 9 & 11 & 11 \\ & & & & & & 7 & 7 \\ & & & & & & & 4 \end{pmatrix}$$

행: i=0,...,7

열: j=1,...,8

그림 7. 적재된 단편 $F_R(q_i, q_j)$의 값

i	q_i	$O_R(q_i)$	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

최소 분할 P ={q_3,q_4,q_6} = {6,9,17}

그림 8. X = 10을 갖는 MIP 시나리오 값

성명	시작 시점	종료 시점	성명	시작 시점	종료 시점
이상식	2	3	강경재	2	3
김일우	2	6	강병민	1	6
최천수	2	6	고일욱	1	6
고구희	2	6	곽정은	1	6
이희숙	1	4	김미애	3	6
정태규	2	6	김보경	2	6
이재만	1	6	감상정	4	9
이재경	4	9	김세환	8	9
이기동	4	12	김영필	4	11
강상식	12	17	김은미	11	16
박미경	8	17	박선희	11	16
이숙자	11	17	신은선	11	16
박한상	11	17	양소라	9	16
이갑재	11	19	왕현민	12	17
여순주	9	19	용미희	12	18
김다희	16	19	유지환	9	19
박상수	14	20	윤동렬	14	20
강미희	14	20	이병훈	14	20
황선애	18	20	이원민	18	20
김선후	18	20	허광재	18	20

그림 9. 시간지원 릴레이션 R과 Q

www.dbpia.co.kr

$r_1 \bowtie q_1$ 간격이 [1-5]를 교차

성명	시작시점	종료시점
이상식	2	3
김일우	2	6
최천수	2	6
고구희	2	6
이희숙	1	4
정태규	2	6
이재만	1	6
이재경	4	9
이기동	4	12

성명	시작시점	종료시점
강경재	2	3
강병민	1	6
고일욱	1	6
곽정은	1	6
김미애	3	6
김보경	2	6
감상정	4	9
김세환	8	9

$r_2 \bowtie q_2$ 간격이 [6-10]를 교차

성명	시작시점	종료시점
김일우	2	6
최천수	2	6
고구희	2	6
정태규	2	6
이재만	1	6
이재경	4	9
이기동	4	12
박미경	8	17
여순주	9	17

성명	시작시점	종료시점
강병민	1	6
고일욱	1	6
곽정은	1	6
김미애	3	6
김보경	2	6
김상정	4	9
김세환	8	9
김영필	4	11
양소라	9	16
유지환	9	19

$r_3 \bowtie q_3$ 간격이 [11-15]를 교차

성명	시작시점	종료시점
이기동	4	12
강상식	12	17
박미경	8	17
이숙자	11	17
박한상	11	17
이갑제	11	19
여순주	9	19
박상수	14	20
강미희	14	20

성명	시작시점	종료시점
김영필	4	11
김은미	11	16
박설희	11	16
신은선	11	16
양소라	9	16
왕현민	12	17
용미희	12	18
유지환	9	19
윤동렬	12	20
이병훈	14	20

$r_4 \bowtie q_4$ 간격이 [16-20]를 교차

성명	시작시점	종료시점
강상식	12	17
박미경	8	17
이숙자	11	17
박한상	11	17
이갑재	1	19
여순주	9	19
김다희	16	19
박상수	14	20
강미희	14	20
황선애	18	20
김선후	18	20

성명	시작시점	종료시점
김은미	11	16
박설희	11	16
신은선	11	16
양소라	9	16
왕현민	12	17
용미희	12	18
유지환	9	19
윤동렬	14	20
이병훈	14	20
이원민	18	20
허광재	18	20

그림 10. 일정분할되어 생성된 R과 Q의 각 4개의 단편들

www.dbpia.co.kr

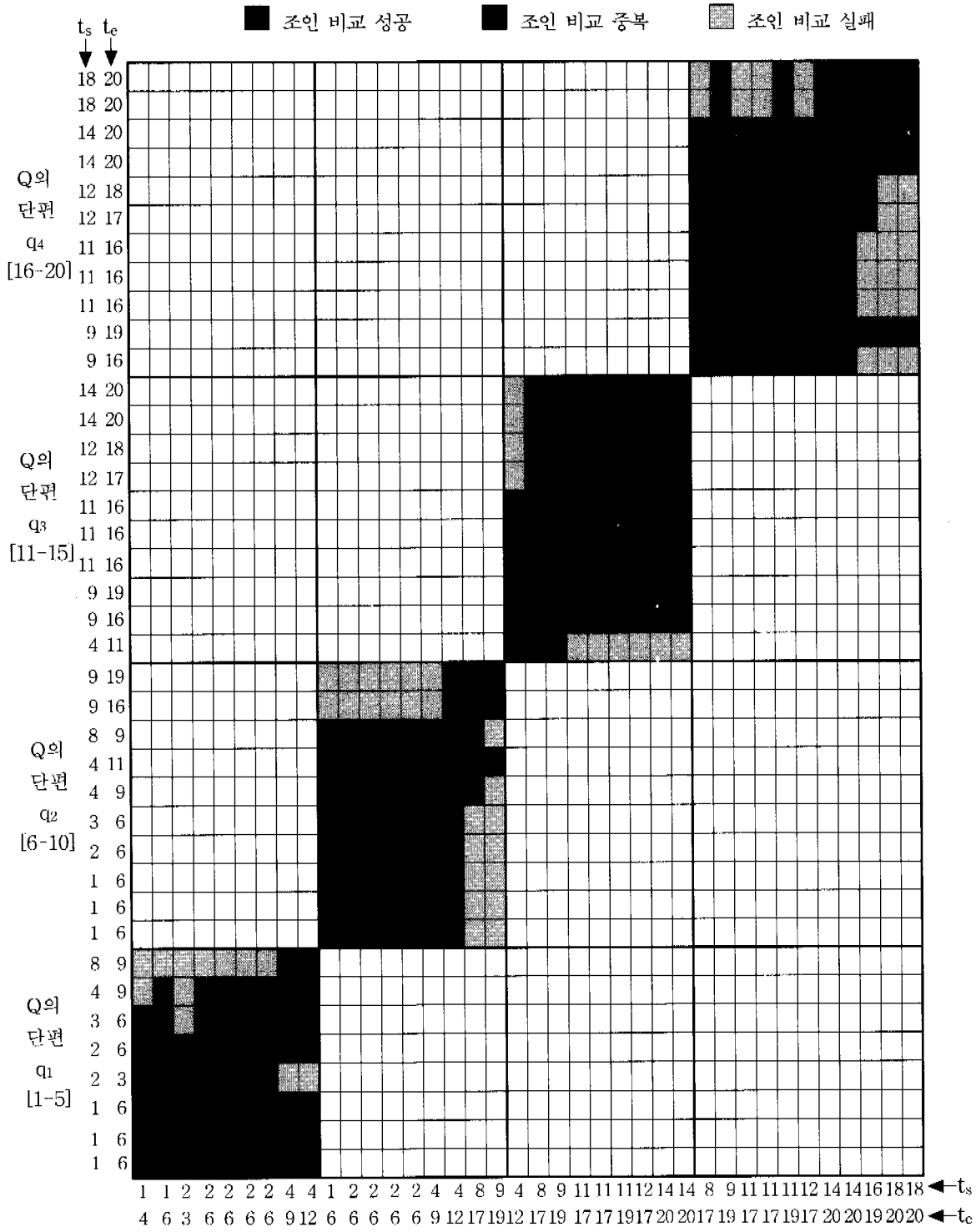


그림 11. 일정 분할된 시간지원 릴레이션 R⋈Q의 조인 처리 결과

www.dbpia.co.kr

$r_1 \bowtie q_1$ 간격이 [1-8]를 교차

성명	시작시점	종료시점
이상식	2	3
김일우	2	6
최천수	2	6
고구희	2	6
이희숙	1	4
정대규	2	6
이재만	1	6
이재경	4	9
이기동	4	12
강상식	12	17

성명	시작시점	종료시점
강경재	2	3
강병민	1	6
고일욱	1	6
곽정은	1	6
김미애	3	6
김보경	2	6
김상정	4	9
김세환	8	9
김영필	4	11

$r_2 \bowtie q_2$ 간격이 [9-15]를 교차

성명	시작시점	종료시점
이재경	4	9
이기동	4	12
강상식	12	17
박미경	8	17
이숙자	11	17
박한상	11	17
이갑재	11	19
여순주	9	19
박상수	14	20
강미희	14	20

성명	시작시점	종료시점
김상정	4	9
김세한	8	9
김영필	4	11
김은미	11	16
박설희	11	16
신은선	11	16
양소라	9	16
왕현민	12	17
용미희	12	18
유지환	9	19

$r_3 \bowtie q_3$ 간격이 [16-17]를 교차

성명	시작시점	종료시점
강상식	12	17
박미경	8	17
이숙자	11	17
박한상	11	17
이갑재	11	19
여순주	9	19
김다희	16	19
박상수	14	20
강미희	14	20

성명	시작시점	종료시점
김은미	11	16
박설희	11	16
신은선	11	16
양소라	9	16
왕현민	12	17
용미희	12	18
유지환	9	19
윤동렬	14	20
이병훈	14	20

$r_4 \bowtie q_4$ 간격이 [19-20]를 교차

성명	시작시점	종료시점
이갑재	11	19
여순주	9	19
김다희	16	19
박상수	14	20
강미희	14	20
황선애	18	20
김선후	18	20

성명	시작시점	종료시점
용미희	12	18
유지환	9	19
윤동렬	14	20
이병훈	14	20
이원민	18	20
허광재	18	20

그림 12. 언더플로우로 분할 생성된 R과 Q의 각 4개의 단편들

www.dbpia.co.kr

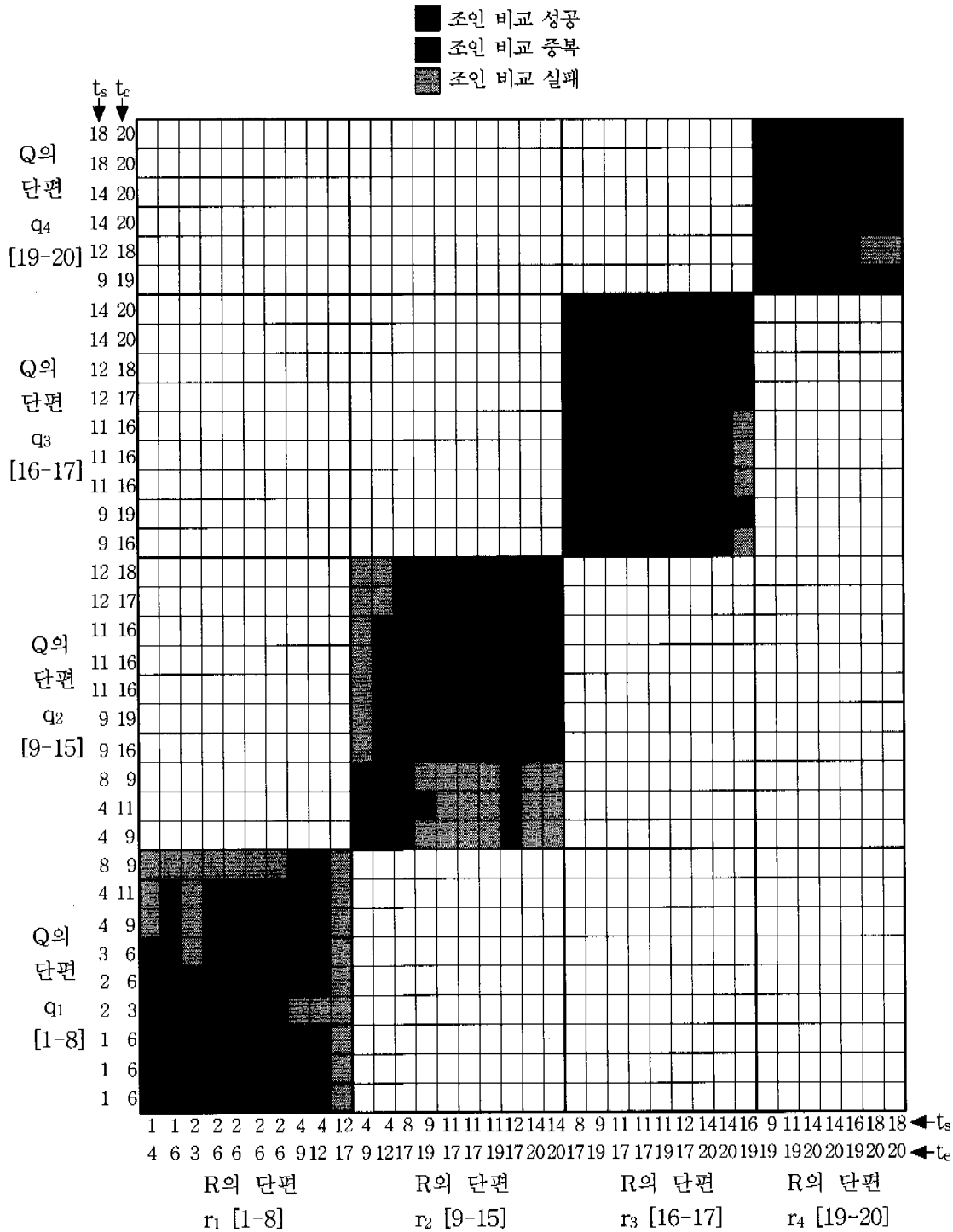


그림 13. 언더플로우 방법으로 분할된 시간지원 릴레이션 R ⋈ Q의 조인 처리 결과

www.dbpia.co.kr

$r_1 \bowtie q_1$ 간격이 [1-16]를 교차

성명	시작시점	종료시점
이상식	2	3
김일우	2	6
최천수	2	6
고구희	2	6
이희숙	1	4
정태규	2	6
이재만	1	6
이재경	4	9
이기동	4	12

성명	시작시점	종료시점
강경재	2	3
강병민	1	6
고일욱	1	6
곽정은	1	6
김미애	3	6
김보경	2	6
김상정	4	9
김영필	4	11

$r_2 \bowtie q_2$ 간격이 [7-9]를 교차

성명	시작시점	종료시점
이재경	4	9
이기동	4	12
박미경	8	17
여순주	9	19

성명	시작시점	종료시점
김상정	4	9
김세환	8	9
김영필	4	11
양소라	9	16
유지환	9	19

$r_3 \bowtie q_3$ 간격이 [10-17]를 교차

성명	시작시점	종료시점
이기동	4	12
강상식	12	17
박미경	8	17
이숙자	11	17
박한상	11	17
이갑재	11	19
여순주	9	19
김다희	16	19
박상수	14	20
김미희	14	20

성명	시작시점	종료시점
김영필	4	11
김은미	11	16
박설희	11	16
신은선	11	16
양소라	9	16
왕현민	12	17
용미희	12	18
유지환	9	19
윤동렬	14	20
이병훈	14	20

$r_4 \bowtie q_4$ 간격이 [18-20]를 교차

성명	시작시점	종료시점
이갑재	11	19
여순주	9	19
김다희	16	19
박상수	14	20
강미희	14	20
황선애	18	20
김선후	18	20

성명	시작시점	종료시점
용미희	12	18
유지환	9	19
윤동렬	14	20
이병훈	14	20
이원민	18	20
허광재	18	20

그림 14. MIP로 분할 생성된 R과 Q의 각 4개의 단편들

www.dbpia.co.kr

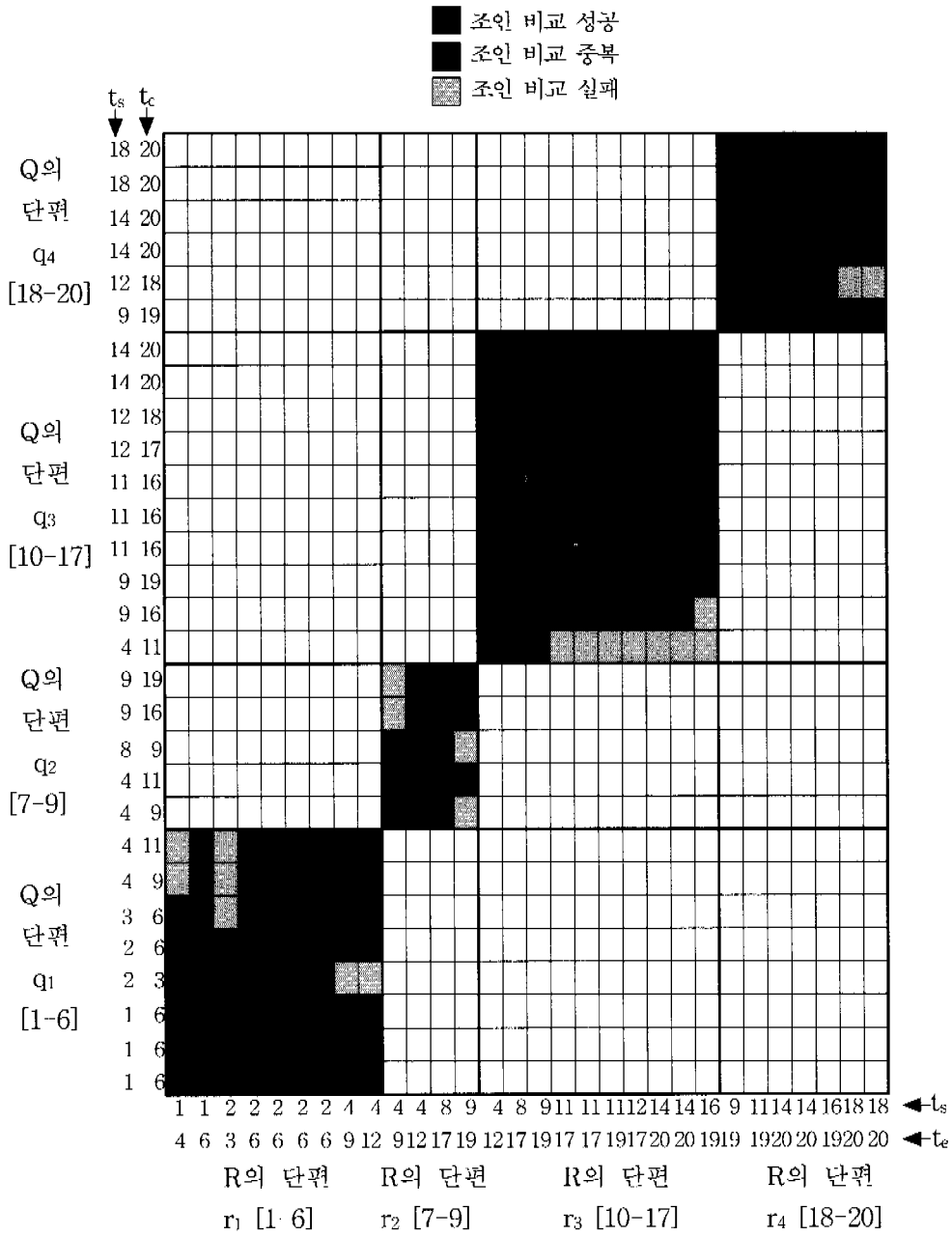


그림15. MIP로 분할된 시간지원 릴레이션 R ⋈ Q의 조인 처리 결과

이를 토대로 시간지원 조인 처리 성능중 간격 분할의 최소점은 다음과 같이 계산된다.

i=1 ⇒ q_1=3, J={0}, q_1 전에 중복되는 수가 없으므로 $c(q_1)=\min\{O_R(q_0)+c(q_0)\}$,
따라서 $fwd(q_1)=0$을 얻는다.

i=2 ⇒ q_2=4, J={0,1},
$c(q_2)=\min\{O_R(q_0)+c(q_0),O_R(q_1)+c(q_1)\}$

두 값 모두 0이므로 0을 출력한다. 동일한 방법으로 i=3 까지 모두 0을 출력한다.

i=4 ⇒ q_4=9, J 값 중에서 단편 내에 10을 초과하는 값은 제외되므로 J={1,2,3},
$c(q_4)=\min\{O_R(q_1)+c(q_1)$,

www.dbpia.co.kr

$O_R(q_2)+c(q_2)$, $O_R(q_3)+c(q_3)$} 즉, min{6, 7, 2} 값 중에서 최소 값은 $O_R(q_3)+c(q_3)=2$, $q_3=6$이 선택되고 $fwd(q_3)=6$을 얻는다. 계속해서 J=7 까지 정지점을 모두 구하면 $fwd(q_4)=9$, $fwd(q_6)=17$ 이며 간격 분할을 최소로 하는 정지점 $P=\{q_3,q_4,q_6\}=\{6,9,17\}$을 구하였다. 위 계산과정을 요약하여 그림8에 정리하였다. 결정된 정지점을 기준으로 릴레이션의 타임스탬프 값을 서로소인 단편으로 분할하면 그림14의 결과를 얻으며,부분 조인하여 처리된 결과는 그림15와 같다. 그림에서 보듯이 기존의 분할 기법보다 제안된 MIP 알고리즘이 수행 시간 및 중복을 현저히 감소시켰음을 알 수 있다. 예제 시나리오를 통한 논리적인 성능 평가 결과 표1의 일정 분할, 언더플로우 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 가장 성능이 우수함을 확인하였다. 위 값을 근거로 처리 비용 감소율을 그림16에 히스토그램으로 표현하였다. 이상과 같이 제안된 알고리즘이 시간 조인 처리에 적용된다면 효율적인 간격 분할을 통해 불필요한 투플의 비교 횟수를 줄이고 기존 언더플로우 분할보다 수행 시간이 O(n)으로 빠른 입출력 성능 개선 효과도 얻을 수 있었다.

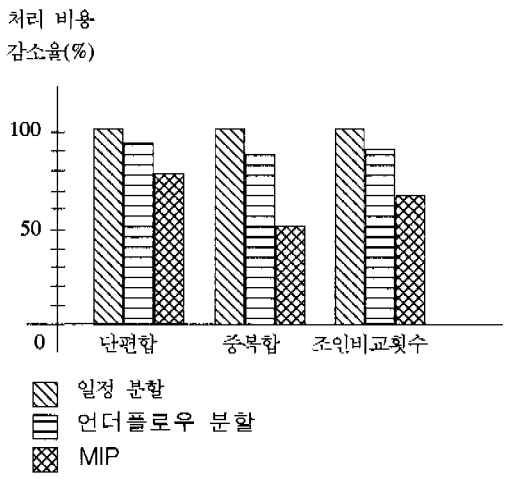


그림 16. 간격 분할의 성능

표 1. 간격 분할의 논리적인 성능 비교

구분 / 간격 분할	단편합	중복합	조인 연산 비교 횟수
일정 분할 (uniform partition)	76	36	728
언더플로우 분할 (underflow partition)	72	32	660
최소 간격 분할 (MIP)	60	20	492

Ⅵ. 결론

시간 조인 연산은 가장 중요한 관계 연산자이며 데이터 정규화에 기인한 필수적인 연산자로서 질의어 최적화의 관점에서 상당히 중요한 위치를 차지한다. 시간지원 조인 연산은 동등 서술식(equality prediate)보다는 부등 서술식(inequality prediate)에 근거하여 서술된다는 점과 시간지원 데이터베이스(temporal database)가 현저히 크다는 점 때문에 시간지원 조인 연산은 관계 조인 연산보다 중요하다. 본 논문에서는 시간 조인 단계 이전에 시간 데이터를 분할하는 간격 분할의 중요성을 살펴 보았으며, 시간지원 조인 연산시 기존의 분할 방법과 제안된 MIP 분할 방법을 비교, 분석하여 시간 조인 처리 성능을 향상 시킬 수 있는 전략을 고찰하였다. 간격 분할을 최소화하기 위한 제약조건은 3절에서 기술하였으며, 4절에서 기존의 간격 분할 방법인 언더플로우 방법을 개선하여 최소 분할을 결정하는 알고리즘을 구현하였다. 논리적인 성능 분석으로 제안된 MIP 알고리즘은 간격 분할의 최소 정지점을 결정하는 이론적인 배경이 될 뿐만 아니라 기존에 사용된 분할 기법보다 성능이 우수하며 시간 조인 연산을 수행할 경우 효과적이고 빠른 입출력의 접근 및 수행시간을 단축시킬 수 있었다.

참 고 문 헌

[1] N.Lorentzos and Y.Mitsopoulos. SQL Extension for Interval Data. Technical Rep. 105, Informatics Lab., Agricultural University of Athens, 1994.

[2] R. Snodgrass,editor. The TSQL2 Temporal Query Language. Kluwer, Sept. 1995.

[3] H. Drawen. Developments in SQL3. personal conversation, Feb. 1997.

www.dbpia.co.kr

[4] P. Mishra and M. Eich. Join Processing in Relational Databases. ACM Computing Surveys, pages 63-113, Mar.1992.

[5] 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.

[6] H. Gunadhi and A. Segev. Efficient Indexing Methods for Temporal Relations. IEEE Transactions on Knowledge and Data Engineering, 5(3):496-509, June. 1993.

[7] C. Kolovson. Indexing Techniques for Historical Databases. In Tansel and et al.[10], chapter 17, pages 418-432.

[8] T. Leung and R.Muntz. Temporal Query Processing and Optimization in Multiprocessor Database Machines. In Proc.18th Int.Conf.onV LDB, ,Vancounver, Canada, pp.383-394, Aug. 1992.

[9] 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, Texas,USA,pages 282-292, Feb. 1994.

[10] J. David, F.Jeffrey, A. Donovan, S.Seshadri. Practical Skew Handling in Parallel Joins. In Proc. 18th VLDB, pp.27-40, Aug.1992.

[11] K.Hua and C.Lee. Handling Data Skew in Multiprocessor Database Computers Using Partition Tuning. In Proc. 17th VLDB, Sept. 1991.

[12] J. Allen, Maintaining Knowledge about Temporal Intervals. CACM, 26(11):832-843, Nov, 1983.

[13] A.Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R.Snodgrass. Temporal Databases. Benjamin/Cummings, 1993.

[14] H. Zhou. Two-stage m-way graph partitioning. Parallel Computing, 19(12):1359-1373, Dec. 1993.

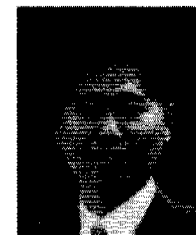
[15] X.Jeffrey, Yu and Kian-Lee Tan, Scheduling Issues in Partitioned Temporal join, Joint Computer Science Technical Report Series, May,1995

[16] 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 546-557, Sept.1994.

[17] A. Segev and H. Gunadhi, Event-Join Optimization in Temporal Relational Databases. In Proceedings of the onference on Very Large Databases,pages 205-215, August 1989.

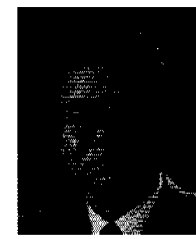
[18] H. Gunadhi and A. Segev, Query Processing Algorithm for Temporal Intersection Joins. In Proceeding of the 7th International Conference on Data Engineering, Kobe, Japan, 1991.pages 131-147, Charlotte, Nc, April 1990

이 광 규(Kwang-Kyu Lee)



1985년 : 동국대학교 수학과 (학사)
1991년 : 동국대학교 대학원 응용수학(이학석사)
1995년~1998년 : 충북대학교 대학원 전자계산학과 박사과정 수료
1996년~현재 : 신흥대학 컴퓨터정보계열 조교수
<주관심 분야> 시간 데이터베이스, 지식기반 정보검색, 퍼지 이론

김 홍 기(Hong-Gi Kim)



1961년 : 연세대학교 수학과 (학사)
1975년 : 연세대학교 교육대학원 응용수학 교육학과 (교육학 석사)
1985년 : 중앙대학교 대학원 응용수학(이학박사)
1980년~현재 : 충북대학교 컴퓨터과학과 교수
<주관심 분야> 퍼지 이론, 정보통신

www.dbpia.co.kr