시간간격을 고려한 시간관계 규칙 탐사 기법
(Discovering Temporal Relation Rules from Temporal Interval Data)

이 용 준 † 서 성 보 †† 류 근 호 ††† 김 혜 규 †
(Yong Joon Lee) (Sung Bo Seo) (Keun Ho Ryu) (Hye Kyu Kim)

요 약 데이타마이닝은 대용량 데이타베이스에 내재된 유용한 지식을 탐사하는 기술로 정의된다. 데이타마이닝에 대한 연구가 진행되면서 순차 패턴, 유사 시계열 탐사, 시간 연관규칙 탐사 등과 같이 시간값을 가진 데이타로부터 지식을 탐사하고자 하는 시간 데이타마이닝에 대한 연구가 수행되었다. 그러나 기존 연구는 트랜잭션의 발생 시점 만을 가진 데이타를 다루고 있으며 시간 간격을 가진 데이타는 거의 고려하고 있지 않다. 실세계에서는 환자의 병력, 상품 구매 이력, 웹 로그 등과 같은 시간간격을 가진 다양한 데이타가 존재하며 이로부터 여러 유용한 지식을 찾아낼 수 있다. Allen은 시간간격 데이타 사이에 발생할 수 있는 시간 관계와 시간 관계를 구할 수 있는 시간간격 연산자를 정의하였다. 본 논문에서는 Allen의 정의를 기반으로 시간간격 데이타로부터 시간관계 규칙을 효율적으로 탐사하기 위한 새로운 데이타마이닝 기법을 제안한다. 이 기법은 발생 시점을 가진 시간 데이타를 시간간격 데이타로 요약하여 일반화하는 전처리 알고리즘과 시간간격 데이타로부터 시간관계 규칙을 생성하는 규칙 탐사 알고리즘으로 구성된다. 이 기법은 기존 데이타마이닝 기법에서 찾지 못하는 유용한 시간 규칙을 탐사할 수 있다.

Abstract Data mining refers to a set of techniques for discovering implicit and useful knowledge from large database.

Many studies on data mining have been pursued and some of them have involved issues of temporal data mining for discovering knowledge from temporal database, such as sequential pattern, similar time sequence, cyclic and temporal association rules, etc. However, all of the works treat problems for discovering temporal pattern from data which are stamped with time points and do not consider problems for discovering knowledge from temporal interval data. For example, there are many examples of temporal interval data that it can discover useful knowledge from. These include patient histories, purchaser histories, web log, and so on.

Allen introduces relationships between intervals and operators for reasoning about relations between intervals.

We present a new data mining technique that can discover temporal relation rules in temporal interval data by using the Allen's theory. In this paper, we present two new algorithms for discovering temporal relationship. The first one, a preprocessing algorithm for the generalization of temporal interval data, summarizes timestamp data into temporal interval data. The second one, a rule discovery algorithm for generating temporal relation rules, discovers rules from temporal interval data. This technique can discover more useful knowledge in compared with conventional data mining techniques.

· 이 연구는 과학재단 특정기초연구(1999-2-30300-006-3)의 연구비 지원과 ETRI와 충북대의 공동 연구로 수행되었음.
† 정 회 원 : 한국전자통신연구원 우정기술연구부 연구원
yjl@etri.re.kr
hkkim@etri.re.kr
†† 비 회 원 : 충북대학교 전산학과
sbseo@comp.kbsi.re.kr
††† 종신회원 : 충북대학교 전기전자컴퓨터공학부 교수
khryu@dblab.chungbuk.ac.kr
논문접수 : 2000년 9월 27일
심사완료 : 2001년 8월 9일

1. 서 론

데이타마이닝은 "데이타로부터 이전에 잘 알려지지는 않았지만, 묵시적이고 잠재적으로 유용한 지식을 추출하는 기술"로 정의되며 전자상거래, 의사결정 지원, 의료 등의 다양한 분야에서 유용하게 활용될 수 있다. 데이타마이닝에 대한 연구가 진행되면서 순차 패턴(sequential pattern), 유사 시계열(Similar Time Sequence) 탐사,

시간 연관 규칙(temporal association rule) 탐사 등과 같이 타임스탬프를 가진 시간 데이타(temporal data)로부터 지식을 탐사하고자 하는 시간 데이터마이닝(temporal data mining)에 대한 연구가 수행되었다[1,2].

기존 연구는 트랜잭션의 발생 시점(time point) 만을 가진 시간 데이타를 다루고 있으며 시간 간격(temporal interval)을 가진 데이타는 거의 고려하고 있지 않다. 그러나 실세계에서는 환자의 병력, 상품 구매 이력, 웹 로그 등과 같은 다양한 시간간격 데이타가 존재하며 이로부터 여러 유용한 지식을 찾아낼 수 있다. 예를 들면 시간간격을 가진 구매 트랜잭션으로 구성된 데이타베이스로부터 "비디오 A를 대여한 고객의 50%가 다음에 비디오 B와 비디오 C를 대여한다"와 같은 순차 패턴 뿐만 아니라, "비디오 A를 대여한 고객의 50%가 A 대여 기간 중에 비디오 B를 대여하고 B의 대여가 끝나는 즉시 비디오 C를 대여한다" 같은 유용한 시간관계를 찾아낼 수 있다. Allen[3]은 시간간격 사이에 발생할 수 있는 시간관계(temporal relations)와 시간관계를 구할 수 있는 시간간격 연산자(interval operator)를 정의하였고, 이를 구현하기 위한 연구가 수행되었다[4,5].

본 논문에서는 시간간격 연산자를 기반으로 시간간격 데이타로부터 시간관계 규칙(temporal relation rule)을 효율적으로 탐사하기 위한 새로운 마이닝 기법을 제안한다. 이 기법에서 제안한 아이디어는 다음과 같다.

- 발생 시점 만을 가진 데이타를 시간간격 데이타로 요약하여 일반화(generalization)하는 전처리(preprocessing) 알고리즘을 제안한다. 예를 들어 3월부터 4월까지 1달 동안 환자 A가 매일 증상 B를 나타낼때 증상은 같고 발생 시점만 다른 여러 B 트랜잭션들이 발생한다. 따라서 B 트랜잭션들은 1달이란 기간(period) 동안 지속적으로 균등하게 발생하므로 이를 3월부터 4월까지의 시간간격을 가진 단일 B 트랜잭션으로 요약할 수 있다. 일반화를 통해 시간간격 데이타를 생성할 수 있을 뿐만 아니라 데이타 크기를 줄여 탐색 공간 및 시간을 절약할 수 있다.
- 시간간격 데이타로부터 트랜잭션 간에 발생하는 시간관계 규칙을 탐사하기 위한 알고리즘을 제안한다. 이 알고리즘은 대표적인 순차패턴 알고리즘인 AprioriAll [6]을 시간관계 규칙 탐사를 위하여 확장하였다. 시간간격 데이타는 상품 구매뿐만 아니라 환자 병력과 같은 여러 분야에서 발생하므로 이 알고리즘에서는 AprioriAll에서 사용하는 항목(item) 대신에 사건(event)이라는 용어를 사용한다. 또한 AprioriAll에서 가지치기(pruning) 작업을 위해 빈발 시퀀스(large sequences)을 구하는 것과 같이 빈발 시간관계 규칙을 생성한다.

본 논문의 2장에서는 관련 연구를 정리하고, 3장에서는 시간관계 규칙 탐사 문제를 수학적으로 정의하고, 4장에서는 시간관계 규칙 탐사를 위한 기법을 예를 들어 단계적으로 설명하였다.

5장과 6장에서는 이 기법의 알고리즘을 구현하여 시험한 결과를 보이고, 결론에서 향후 연구 방향을 제시하였다.

2. 관련 연구

시간 데이타로부터 의미있는 지식을 탐사하기 위한 시간 데이타마이닝 기법에 대한 여러 연구가 진행되었다[7, 8,9,10]. 이러한 연구들은 크게 순차패턴 탐사, 유사 시계열 탐사, 시간 규칙을 탐사하기 위한 연구로 분류된다.

순차패턴 탐사는 항목집합으로 구성된 트랜잭션들 간에 특정 항목집합이 순차적으로 발생하는 패턴을 탐사하는 기법으로 한 트랜잭션 내에서 발생하는 항목들 간에 연관성을 탐사하는 연관규칙에 시간적인 관계를 추가한 것이다[6,11]. 즉, 고객 트랜잭션 내에 항목 A가 존재한다면, 그 이후에 발생하는 같은 고객의 다른 트랜잭션 내에 항목 B, C, ...가 차례로 존재한다는 시퀀스(sequence)를 탐사하는 기법이다. 각 고객들의 트랜잭션을 시간순서로 볼 수 있는데 이를 고객 시퀀스라고 부른다. 순차패턴 문제는 사용자가 지정한 최소 지지도(minimum support threshold)를 만족하는 모든 시퀀스들 사이에서 최대 시퀀스를 탐사하는 것이다. 대표적인 순차패턴 알고리즘 AprioriAll, AprioriSome은 연관규칙 알고리즘인 Apriori를 기반으로 하고 있다.

이러한 순차패턴 기법을 개선, 확장한 기법으로는 GSP(Generalized Sequential Pattern)[12], SPIRIT[13]가 있다. GSP는 AprioriAll 같은 기존 알고리즘에 시간 제약 조건(time constraint), 슬라이딩 시간 윈도우, 분류(taxonomy)를 이용한 일반화 개념을 추가한 기법이다. SPIRIT는 사전에 정의된 제약조건(regular expression constraint)을 만족하는 모든 빈발 시퀀스를 찾는 기법이다.

빈발 에피소드(frequent episode) 탐사[14,15]는 일련의 사건 시퀀스(event sequence) 데이타로부터 빈번하게 발생하는 에피소드를 찾는 순차패턴 기법이다. 에피소드는 빈번하게 발생하는 특정 사건 시퀀스로 정의되며 시퀀스를 구성하는 사건은 서로 밀접하게 관련된 사건이다. 탐사 문제는 사용자가 지정한 윈도우 크기를 갖는 시간 윈도우들의 집합에서 에피소드가 발생한 윈도

우의 비율이 최소 발생도(frequency threshold) 이상을 만족하는 모든 빈발 에피소드를 찾는 것이다.

예를 들어 매초 마다 발생한 사건 시퀀스를 (A, B, D, A, C, B, D, A, D), 시간 윈도우 크기를 3초, 최소 발생도를 50%라 할때 (A, B)는 세 개의 윈도우 중에 두개의 윈도우 내에서 발생하였으므로 에피소드이다.

그 외에도 지지도나 발생도 만이 아니라 관심도(interestingness)를 기준으로 순차패턴을 탐사하는 기법[16,17]과 주기적으로 발생하는 순차패턴을 탐사하는 기법[18]이 연구되었다.

유사 시계열 탐사는 주식, 물가, 판매량 등과 같이 데이타가 시간축으로 나열된 시계열 데이타(time series)로부터 유사한 시계열 데이타를 찾는 기법이다[19,20]. 예를 들어 주식 동향을 나타내는 시계열 데이타로부터 특정 주식과 비슷한 형태의 데이타를 찾아낼 수 있다면 두 주식의 운영 전략에 유사성을 부여할 수 있게 된다. 탐사 문제는 주어진 시계열 시퀀스 집합으로부터 사용자가 질의한 질의 시퀀스(query sequence)와 유사한 모든 시퀀스 또는 유사 시퀀스(similar sequence)의 모든 쌍을 탐사하는 것이다. 유사 시퀀스의 쌍을 탐사하는 기법은 전반적으로 유사한 시퀀스 쌍을 탐사하는 기법과 부분적으로 유사한 시퀀스 쌍을 탐사하는 기법[21]으로 분류된다. 또한 공간 데이타, 이미지 데이타 등을 위한 유사 시계열 탐사 기법이 연구되었는데 공간 데이타에서 빠른 유사 조인(similarity join)을 위한 알고리즘 및 색인 구조가 제안되었다[22].

시간규칙 탐사 기법은 기존 데이타마이닝 기법에서 정의한 규칙을 확장하여 시간 관계 및 인과 관계(causal relationship)를 포함한 시간규칙을 찾는 기법이다. 여기에 속한 기법으로는 주기적으로 반복되는 연관규칙을 탐사하는 주기적 연관 규칙 탐사(Cyclic Association)[23], 달력(calendar)으로 표현된 시간 패턴을 가지는 연관규칙을 탐사하는 달력 연관규칙 탐사(Calendric Association)[24,25] 등이 있다.

이러한 연구들은 시간 데이타를 대상으로 수행되었으며 시간간격 데이타는 거의 고려하고 있지 않다. 따라서 최근에 시간간격 데이타로부터 시간관계를 탐사하기 위한 기초적인 연구가 일부 진행되었다[26,27]. 그러나 시간간격 데이타로부터 시간관계 규칙을 탐사하는 문제는 매우 복잡한 문제로서 아직 뚜렷한 연구는 이루어지지 않고 있다.

3. 문제 정의

이 장에서는 시간관계 규칙 탐사 문제를 형식화하여 정의(formal definition)한다. 이는 복잡한 문제로서 데이타마이닝 분야에서는 거의 연구가 이루어지지 않고 있으므로 체계적인 연구를 위해서는 문제를 명확히 정의하는 것이 필요하다. 먼저 시간관계, 시간간격 일반화, 시간관계 규칙에 대한 정의를 내리고 이로부터 시간관계 규칙을 탐사하는 문제를 단계별로 정의한다.

3.1 시간 관계

발생 시간을 가진 사건 e를 $e=(E, t)$라고 정의한다. E는 사건 종류, t는 e가 발생한 시점이다. 발생시간에 따라 정렬된 사건 시퀀스를 $S=\langle e_1, e_2, \ldots, e_n\rangle$이라고 정의한다. 단, $e_i=(E_i, t_i)$이고, $t_i \le t_{i+1}$, $i=1, \ldots, n-1$이다. S의 처음 사건 e_1와 마지막 사건 e_n 사이의 기간(period)을 $[t_1, t_n]$이라고 표현한다.

또한 시간간격을 가진 사건을 $e'=(E, vs, ve)$라고 정의한다. vs는 시간간격 시작점, ve는 시간간격 종료점이며 시작점을 $e'.vs$, 종료점을 $e'.ve$라고 표현한다. 시간간격에 따라 정렬된 사건 시퀀스를 $S'=\langle e'_1, e'_2, \ldots, e'_n\rangle$이라고 정의한다. 단, $e'_j=(E_j, vs_j, ve_j)$이고, $ve_j \le vs_{j+1}$, $j=1, \ldots, n-1$이다.

정리 3.1: 임의의 시간단위 U(time granularity)[28], 기준 시점 V가 주어졌을 때 시퀀스 S는 시간 간격을 가진 시퀀스 $S'=\langle (E_1, vs_1, ve_1), (E_2, vs_2, ve_2), \ldots, (E_n, vs_n, ve_n)\rangle$으로 변환된다. 단, $ve_i \le vs_{i+1}$, $i=1, \ldots, n-1$이고 vs_i, ve_i는 정수이다. 또한 S'의 기간을 $[vs_1, ve_n]$이라 정의하고 이는 $[1, m]$으로 변환할 수 있다. 단, m은 임의의 정수이고 $m \ge 1$이다.

증명: 임의의 시점 t는 시간 함수 $f(t, U, V)$를 이용하여 임의의 정수 m으로 변환될 수 있다[24]. 예를 들어 t를 2000년 10월 5일, U를 월, V를 2000년 1월 1일이라고 할 때 $f(2000/10/5, month, 2000/1/1)=10$이 된다. 즉 t는 시간간격 10월 1일과 10월 31일 사이에 속하며 U가 월이므로 10월로 변환된 후, V가 2000년 1월이므로 V을 기준으로 10번째 달로 인식되어 10이 계산된다.

따라서 사건 시퀀스 $S=\langle (E_1, m_1), (E_2, m_2), \ldots, (E_n, m_n)\rangle$으로 표현된다. 단, $m_i \le m_{i+1}$, $i=1, \ldots, n-1$이고 m_i는 정수이다. 이때 시간 순서로 정렬된 S는 시간간격을 가진 시퀀스 $S'=\langle (E_1, m_1, m_2-1), (E_2, m_2, m_3-1), \ldots, (E_{n-1}, m_{n-1}, m_n-1), (E_n, m_n, m_n)\rangle$으로 변환할 수 있다[29]. 단, m_j는 사건 e'_j의 시간간격 시작점, $m_{j+1}-1$은 시간간격 종료점이고 $j=1, \ldots, n-1$이다.

따라서 정수 m_j는 $vs_j, m_{j+1}-1$은 ve_j, m_{j+1}은 vs_{j+1}으로 대치되고 $ve_j \le vs_{j+1}$, $S'=\langle (E_1, vs_1, ve_1), (E_2, vs_2, ve_2),$

$\ldots,(E_{n-1}, vs_{n-1}, ve_{n-1}),(E_n, vs_n, ve_n)>$이 된다.

또한 S'의 기간 $[vs_1, ve_n]$은 V를 vs_1으로 지정할 때 $f(vs_1, U, vs_1)=1, f(ve_n, U, vs_1)=m$이 되어 정수 기간 $[1,m]$으로 변환할 수 있다. 예를 들어 사건 A의 발생시간이 2000년 1월 3일 9시 20분, B의 발생시간이 11시 30분, C의 발생시간이 20시 40분이라고 가정하면 기간은 [2000/01/03 09 : 20, 2000/01/03 20 : 40]이 된다. 시간 단위가 1시간이라고 가정하면 기간은 [1,12]로 변환되고 <(A,1,2), (B,3,11), (C,12,12)>의 사건시퀀스가 발생한다. ■

정의 3.1: 임의의 시간간격을 가진 사건 집합은 $IE=\{e_1', e_2', \ldots, e_n'\}$이고 $e'_j=(E_j, vs_j, ve_j)$이다. 여기서 $j=1, \ldots,n$이고, 사건 e_j'의 시간간격 시작점은 $e_j'.vs$, 종료점은 $e_j'.ve$이다.

위의 정의에서 사건관계 $\Omega=\{(x,y) \mid x,y \in IE, x \neq y\}$에 포함되는 임의의 사건 조합 (x,y)는 이항 시간 관계(binary temporal relationship) $R(x,y)$를 가진다. 단, x와 y는 서로 다른 사건이다. $R(x,y)$는 Allen[3]이 제안한 시간간격 연산자를 참조하여 다음과 같이 정의한다.

정의 3.2: 시간관계 $R(x,y)=\{P(x,y) \mid P \in IO\}, \forall(x,y) \in \Omega$이다.

시간간격 연산자의 집합은 $IO=\{before, equal, meets, overlaps, during\}$이고 $P(x,y)$는 x, y간의 시간관계를 표현하는 이진 술어(predicate)이다. $R(x,y)$는 다음과 같이 수학적으로 표현할 수 있다.

$before(x,y) \equiv x.ve < y.vs$
$equal(x,y) \equiv (x.vs = y.vs) \wedge (x.ve = y.ve)$
$meets(x,y) \equiv x.ve = y.vs$
$overlap(x,y) \equiv (x.vs < y.vs) \wedge (x.ve > y.vs)$
$during(x,y) \equiv (x.vs > y.vs) \wedge (x.ve < y.ve)$

예를 들어 $before(x,y)$는 사건 y의 발생기간 이전에 사건 x가 발생, $equal(x,y)$은 같은 기간 동안에 x와 y가 발생, $meets(x,y)$는 x의 발생기간이 끝난 직후에 y가 발생, $overlap(x,y)$는 x의 발생이 시작된 후 끝나기 전에 y가 발생, $during(x,y)$는 y의 발생기간 동안 x가 발생한 경우를 말한다.

IE에 속하는 임의의 세 사건을 e_1', e_2', e_3'라고 정의한다. e_1', e_2', e_3'에 대해 임의의 시간관계 $R_1(e_1', e_2'), R_2(e_2', e_3')$가 존재한다고 가정한다.

정리 3.2: 새로운 시간관계 $R_3(e_1', e_3')$는 $R_1(e_1', e_2')$, $R_2(e_2', e_3')$로부터 자동으로 구할 수 있다.

예를 들어 사건 e_1', e_2', e_3' 사이에 두 시간관계 $before(e_1', e_2')$, $meet(e_2', e_3')$가 존재한다면 이 두 관계로부터 $before(e_1', e_3')$관계가 유도된다. 이러한 관계는 표1과 같은 행렬로 요약할 수 있다.

증명: 이 정리는 Allen의 연산자[3]에 의하여 증명된다. ■

표 1 시간관계 행렬

$R_1(e_1', e_2')$ \ $R_2(e_2', e_3')$	<	m	o	d
before <	<	<	<	<, o, m, d
meets m	<	<	<	o, d
overlap o	<	<	<, o, m	o, d
during d	<	<	<, o, m, d	d

데이타베이스로부터 정의 3.2와 같은 시간관계를 탐사하는 비용은 매우 크다. 예를 들어 데이타베이스 D에 n개의 사건을 가진 사건집합 IE가 존재하며 시간간격 연산자의 수가 m이라고 할 때 발생될 수 있는 모든 시간관계 $R(x,y)$의 개수는 $(n)\times(n-1)\times m$이다. 따라서 정리3.2를 이용하여 새로운 시간관계를 자동으로 구할 수 있을지라도, 모든 사건 쌍을 순차적으로 비교하여 시간관계를 찾는 Allen의 알고리즘 1[3]을 이용한다면 최악의 경우에 시간 복잡도(time complexity)는 $O(N^2)$이 된다.

정리 3.3: 알고리즘 1의 시간 복잡도는 $O(N^2)$이다.

증명: 사건집합 IE로부터 구할 수 있는 모든 시간관계 집합을 RS라고 정의한다. 단, $RS \in R(x,y)$이다. RS를 구하기 위해 x,y를 비교한 총 횟수를 $f(n)$이라고 하면 $f(n)=(n)\times(n-1)=n^2-n$이 된다. 따라서 자연수에 대해 $f(n) \leq n^2$이므로 $f(n)=O(N^2)$이 된다. ■

알고리즘 1 Allen의 시간관계 탐사 알고리즘

```
Let IE be event set in definition 3.1
Let R(x,y) be temporal relation in definition 3.2
RS ← ϕ
for each event x in IE
    for each event y in IE
        RS ← RS ∪ R(x,y);
Return RS
```

이와 같이 Allen의 알고리즘은 시간 복잡도 $O(N^2)$을 가지므로 대용량 데이타베이스로부터 시간관계를 탐사하기 위해서는 적합하지 않다. 또한 모든 사건 쌍에 대

해, 모든 시간관계를 구함으로 인해 지지도가 낮은 시간관계를 탐사할 가능성이 높다. 따라서 본 논문에서는 이러한 문제점을 해결하기 위하여 Allen 알고리즘을 확장하여 대용량 데이타베이스로부터 유용한 시간관계 규칙을 효율적으로 탐사하기 위한 새로운 데이타마이닝 기법을 제안하였다. 이 기법은 시간간격 데이타를 요약하여 일반화하는 전처리 알고리즘과 시간간격 데이타로부터 시간관계 규칙을 생성하는 규칙 탐사 알고리즘으로 구성된다.

전처리 알고리즘은 정리 3.1과 정의 3.1과 같은 시간간격 데이타로 구성된 입력 데이타베이스의 크기를 줄이며, 규칙 탐사 알고리즘은 순차패턴 알고리즘인 AprioriAll[6]을 시간관계 규칙 탐사를 위하여 확장하여 데이타베이스로부터 최소 지지도 이상을 만족하는 시간관계 규칙 만을 탐사한다. 본 알고리즘의 이론적인 근거를 제시하기 위하여 시간간격 일반화와 시간관계 규칙 탐사 문제를 구체적으로 정의하면 다음과 같다.

3.2 시간 간격 일반화

트랜잭션들로 구성된 데이타베이스를 D라고 할 때 각 트랜잭션은 고객ID, 트랜잭션이 발생한 시점, 사건종류 집합으로 구성되며 같은 시점에 두개 이상의 트랜잭션이 발생하는 고객은 없다고 가정한다. 데이타베이스 스키마는 $\{C_{id}, T_{time}, \{E_1, E_2, ...E_n\}\}$으로 표현되며 C_{id}는 고객 ID, T_{time}은 트랜잭션 발생 시점, E는 사건종류 이다. 사건종류의 예로는 환자의 증상, 구매 상품, 방문한 웹페이지 등이 있다.

데이타베이스 D의 트랜잭션 집합은 C_{id}와 T_{time}을 기준으로 오름차 순으로 정렬되어 있다고 가정한다. 3.1절에 의해, 임의의 고객 C_{id}의 트랜잭션 집합을 사건 시퀀스 $S(C_{id})=\langle e_1, e_2, ..., e_n\rangle$이라고 한다. 단, $S(C_{id})=\langle(E_1, t_1),(E_2, t_2),...,(E_n, t_n)\rangle, t_i \le t_{i+1}, i=1,..., n-1$이다. 이때 $S(C_{id})$에 포함된 임의의 사건종류 E_i에 대하여 시점이 다른 동일한 사건 집합 $\{(E_i, t_1),(E_i, t_2),..., (E_i, t_m)\}$이 발생할 수 있다. 단, $t_j \le t_{j+1}, j=1,..., m-1$이다. 예를 들어 환자 C_1은 증상 E_1이라는 사건종류를 다른 시점에 여러 번 경험(사건)한다.

시간 간격 일반화는 E_1과 같이 다른 시점을 가진 연속된 여러 동일 사건들이 발생할 경우 이를 시간간격을 가진 한 사건으로 요약하는 작업을 말한다[27]. 또한 다른 시간간격을 가진 동일 사건들이 발생할 경우에도 이를 요약할 수 있다. 시간간격 일반화를 통해 데이타베이스 D로부터 시간 간격 데이타를 생성하고 효과적으로 D를 요약할 수 있다.

데이터베이스 D에 있는 모든 고객의 집합을 다음과 같이 정의한다.

정의 3.3: 데이타베이스 D에 있는 고객 $C=\{C_1, C_2, ..., C_n\}$이고, 이때의 사건 시퀀스 $S(C)=\{S(C_1), S(C_2), ..., S(C_n)\}$이다.

여기서 임의의 시퀀스 $S(C_{id})$에 포함된 모든 사건 종류를 후보 사건 종류(candidate event type)라고 부른다. 만일 $S(C_{id})$에서 임의의 후보 사건종류 E_i가 발생한다면 E_i를 발생시킨 고객 D는 사건종류 E_i를 지지한다고 한다. C_{id}에 속한 여러 트랜잭션들에서 E_i가 여러 번 발생하였더라도 한 번 발생한 것으로 간주한다. E_i의 지지도(support)를 $Supp(E_i)$라고 하고 이는 D에서 E_i를 지지하는 전체 고객의 수이다. 만일 사용자가 지정한 최소 지지도 $Supp_{min}$에 대해 $Supp(E_i) \ge Supp_{min}$이라면 E_i를 빈발 사건 종류(large event type), E_i를 가진 사건 e_i를 빈발 사건(large event)이라고 부른다. $Supp_{min}$은 전체 고객 수에 대한 지지 고객 수의 비율이다. 예를 들어 500명의 고객이 있고 $Supp_{min}$이 40%, $Supp(E_i)$가 200이면 E_i는 빈발사건 종류이다.

D로부터 빈발 사건종류 집합 $LE=\{E_1, E_2, ..., E_m\}$을 구할 수 있다.

앞의 3.1절에 의해, C_{id}와 상관 없이 데이타베이스 D의 모든 빈발사건을 발생시간에 따라 정렬한 사건 시퀀스를 $S=\langle e_1, e_2, ..., e_n\rangle$이라고 정의한다. 단, $e_i=(E_i, t_i)$, $t_i \le t_{i+1}$, $i=1,..., n-1$이고 $E_i \in LE$이다.

정리 3.1에 의해, S는 시간단위 U, 기준 시점 V가 주어졌을때 시간간격을 가진 사건 시퀀스 $S'=\langle(E_1, vs_1, ve_1), (E_2, vs_2, ve_2),..., (E_n, vs_n, ve_n)\rangle$으로 변환된다. 단, vs_i, ve_i는 정수이고 $ve_1 \le vs_{i+1}, i=1,..., n-1$이다. S'의 기간 $[1, m]$은 D에서 최초의 빈발사건이 발생한 시간부터 마지막 빈발사건이 발생한 시간까지의 전체 기간을 의미한다.

사용자가 지정한 시간 윈도우 크기를 w, w의 크기를 가진 임의의 윈도우를 W라고 정의한다. 단, w는 정수 값이고 $w > ve_i - vs_i, i=1,..., n$이다. 이때 S'의 윈도우 갯수 wn은 $\left\lceil \frac{[1, m]}{w} \right\rceil$이 되고 윈도우 시퀀스 $W_1, W_2, ..., W_{wn}$이 존재한다.

만일 S'에 속한 임의의 윈도우 W_i안에서 빈발 사건종류 E_i가 존재한다면 W_i에서 E_i이 발생하였다고 말한다. E_i의 발생도를 $Freq(E_i)$라고 정의하고 이는 S' 내에 있는 윈도우 시퀀스 $W_1, W_2, ..., W_{wn}$에 대해 E_i가 발생한 윈

도우의 수이다. 윈도우 내에서 E_i가 여러 번 발생하더라도 E_i는 한번만 발생한 것으로 간주한다. 만일 사용자가 지정한 최소 발생도 $Freq_{\min}$에 대해 $Freq(E_i) \geq Freq_{\min}$이라면 E_i는 S'의 기간 $[1, m]$에서 균등하게 발생한다고 정의하며 E_i를 균등 사건 종류(uniform event type)라고 부른다. 예를 들어 E_i의 수명 [1, 50] 사이에 윈도우 크기 5를 가진 10개의 윈도우가 존재하고, $Freq_{\min}$이 50%, $Freq(E_1)$이 6이면 E_1은 균등 사건종류이다. 사건 시퀀스 S'로부터 균등 사건종류 집합 $UE=\{E_1, E_2, ..., E_l\}$을 구할 수 있다.

모든 균등 사건종류는 빈발 사건종류이나, 모든 빈발 사건종류가 균등 사건종류는 아니다.

정의 3.4: 임의의 고객 C_{id}의 사건 시퀀스는 $S(C_{id}) = \langle e_1, e_2, ..., e_n \rangle$이다. 단, $S(C_{id}) = \langle (E_1, t_1), (E_2, t_2), ..., (E_n, t_n) \rangle$, $t_1 \leq t_{i+1}$, $i=1, ..., n-1$이고 $E_i \in UE$이다. 또한 $S(C_{id})$는 S의 부분 집합이다.

$S(C_{id})$는 정리 3.1에 의해, $S'(C_{id}) = \langle (E_1, vs_1, ve_1), (E_2, vs_2, ve_2), (E_n, vs_n, ve_n) \rangle$으로 변환할 수 있다. 단, $ve_1 \leq vs_{i+1}$, $i=1, ..., n-1$이다. $S'(C_{id})$는 기간 $[vs_1, ve_n]$을 가지며 S'의 기간 $[1, m]$에 포함된다. $[vs_1, ve_n]$은 고객 C_{id}의 트랜잭션 집합에서 최초의 사건 e_1'이 발생한 시간부터 마지막 사건 e_n'이 발생한 시간까지의 기간을 의미한다.

$S'(C_{id})$에 존재하는 임의의 균등 사건종류 E_j에 대하여 시간간격이 다른 동일한 사건시퀀스 $S''(E_j) = \langle (E_j, vs_k, ve_k), (E_j, vs_{k+1}, ve_{k+1}), ..., (E_j, vs_m, ve_m) \rangle$이 존재한다고 가정한다. 단, $ve_i \leq vs_{i+1}$, $i=k, ..., m-1$이다. 이때 $S''(E_j)$의 기간 $[vs_k, ve_m]$은 $S'(C_{id})$의 기간 $[vs_1, ve_n]$에 포함되며 vs_k, ve_m은 정수 값을 가진다.

정리 3.4: 임의의 고객 C_{id}에 의해 발생한 균등 사건종류 E_j의 시퀀스 $S''(E_j)$는 기간 $[vs_k, ve_m]$을 가지는 한 사건 (E_j, vs_k, ve_m)으로 요약하여 일반화할 수 있다.

증명: 균등 사건종류 E_j는 데이타베이스 D의 사건 시퀀스 S'의 기간 $[1, m]$에서 균등하게 발생한다고 정의하였다. 고객 C_{id}의 사건시퀀스 $S'(C_{id})$는 사건시퀀스 S'의 부분집합이고, $S''(E_j)$는 $S'(C_{id})$의 부분집합이다. 따라서 $S''(E_j)$의 기간 $[vs_k, ve_m]$에서 E_j는 균등하게 발생하므로 한 사건으로 일반화할 수 있다. ■

예를 들면 고객 C_1이 상품 A를 1월, 4월, 7월에 구매하고 시간단위가 월, 윈도우 크기가 분기이면 4분기 중 3분기를 구매하였으므로 $Freq(A)$는 3이다. 만일 $Freq_{\min}$이 50%이면 A는 균등 사건종류이므로 상품 A를 1월부터 7월까지 구매하였다고 간주하고 $[A, 1, 7]$로 요약한다.

3.3 시간 관계 규칙 탐사

데이타베이스 D의 균등사건종류 집합 UE를 이용하여 일반화된 데이타베이스를 ND라고 부른다. 정의 3.3에 따라, ND의 모든 고객 집합 C를 $\{C_1, C_2, ..., C_n\}$이라고 하면 ND는 사건 시퀀스의 집합 $\{S'(C_1), S'(C_2), ..., S'(C_n)\}$으로 구성된다.

그리고 정의 3.4에 의해, C에 속한 임의의 고객 C_{id}의 사건 시퀀스 $S'(C_{id})$를 구성하는 사건집합을 $IE(C_{id}) = \{e_1', e_2', ..., e_m',\}$이라고 할 수 있다. 단, $e_i' = (E_i, vs_i, ve_i)$, $i=1, ..., m$이고 $E_i \in UE$이다.

정의 3.5: 사건시퀀스 $S'(C_{id})$에 대해 $\Omega = \{(x, y) \mid x, y \in IE(C_{id}), x \neq y\}$이다.

정의 3.2에 의해, 임의의 사건 조합 (x, y)는 이항시간관계 $R(x, y) = \{P(x, y) \mid P \in IO\}$, $\forall (x, y) \in \Omega$를 가진다. 이때 $R(x, y)$를 후보 시간관계(candidate temporal relationship)라고 부른다.

예를 들어 고객 C_1의 사건집합 $IE(C_1) = \{e_1', e_2', e_3'\}$에 대해 후보 시간관계의 집합 $\{before(e_1', e_2'), overlap(e_2', e_3'), before(e_1', e_3')\}$가 발생한다. 데이타베이스 ND로부터 모든 후보 시간관계의 집합 $CR = \{R_1(x, y), R_2(x, y), ..., R_k(x, y)\}$를 구할 수 있다.

고객 C_{id}의 후보 시간관계 집합을 C_{id}의 고객 시간관계(customer temporal relationship), $CR(C_{id})$라고 부른다. 만일 고객 시간관계 $CR(C_{id})$가 임의의 후보 시간관계 $R_1(x, y)$를 포함하고 있다면 고객 C_{id}는 $R_1(x, y)$를 지지한다고 한다. C_{id}에 속한 여러 트랜잭션들에서 $R_1(x, y)$가 여러 번 발생하였더라도 한 번 발생한 것으로 간주한다. $R_1(x, y)$의 지지도를 $Supp(R_1)$이라고 하고 이는 데이타베이스 D에서 $R_1(x, y)$를 지지하는 고객들의 수이다. 만일 사용자가 지정한 최소 지지도 $Supp_{\min}$에 대해 $Supp(R_1) \geq Supp_{\min}$이라면 $R_1(x, y)$를 빈발 시간관계(frequent temporal relationship)라고 부른다. 예를 들어 500명의 고객이 있고 $Supp_{\min}$이 30%, $Supp(R_1)$이 100이면 $R_1(x, y)$는 빈발 시간관계이다. 후보 시간관계 집합 CR로부터 빈발 시간관계 집합 $FR=$

$\{R_1(x,y), R_2(x,y), ..., R_n(x,y)\}$를 구할 수 있다.

정리 3.5: 임의의 빈발 시간관계 $R_i(x,y)$를 구성하는 모든 사건 x, y는 빈발 사건이다.

증명: 만일 임의의 사건 e_k'가 존재하고 $R_i(x,y)$를 구성하는 사건 x, y에 대하여 $e_k' \subseteq \{x,y\}$이면, $R_i(x,y)$를 지지하는 모든 고객들이 반드시 e_k'도 지지하므로 $Supp(e_k') \geq Supp(R_i)$가 성립한다. 이때 $R_i(x,y)$가 빈발 시간관계라면 $Supp(R_i) \geq Supp_{\min}$이 성립되며, 이로부터 $Supp(e_k') \geq Supp(R_i) \geq Supp_{min}$이 유도되므로 시간관계 $R_i(x,y)$를 구성하는 x, y는 빈발사건이 된다. 이 정리를 통해 빈발 시간관계를 구하기 위해서는 반드시 빈발 사건종류를 먼저 구해야 함을 알 수 있다. ■

임의의 사건 집합 $\{e_1', e_2', e_3'\}$에서 발생한 빈발 시간관계 집합을 $\{R_1(e_1', e_2'), R_2(e_2', e_3'), R_3(e_1', e_3')\}$라고 정의한다.

정의 3.6: 시간관계 규칙 TR은 다음과 같이 정의된다.

$TR(e_1', e_2', e_3') = R_1(e_1', e_2') \mid Supp(R_1) \wedge R_2(e_2', e_3') \mid Supp(R_2) \wedge R_3(e_1', e_3') \mid Supp(R_3)$이다. 즉, 시간관계 규칙은 임의의 사건집합으로부터 발생한 빈발 시간관계의 연결로 표현할 수 있다.

예를 들어 빈발 시간관계 집합을 $\{before(e_1', e_2'), overlap(e_2', e_3'), before(e_1', e_3')\}$라고 가정하면, 시간관계 규칙 $TR(e_1', e_2', e_3') = before(e_1', e_2') \mid Supp(R_1) \wedge overlap(e_2', e_3') \mid Supp(R_2) \wedge beore(e_1', e_3') \mid Supp(R_3)$가 된다. 결론적으로, 데이타베이스 D로부터 모든 시간관계 규칙의 집합 $\{TR_1, TR_2, ..., TR_n\}$을 구할 수 있다.

알고리즘 2: 시간관계 규칙을 탐사하는 문제는 다음과 같은 절차로 수행된다.

입력 데이타: 발생시점을 가진 트랜잭션들로 구성된 데이타베이스 D, D의 스키마는 $\{C_{id}, T_{time}, \{E_1, E_2, ..., E_n\}\}$

출력 결과: 정의 3.2와 정의 3.6에 의해, 정의된 시간관계 규칙 집합 $\{TR_1, TR_2, ..., TR_n\}$

단계 1: 정리 3.5에 의해, 우선 데이타베이스 D로부터 사용자가 지정한 최소 지지도 $Supp_{\min}$ 이상을 가지는 빈발 사건종류 집합 $LE=\{E_1, E_2, ..., E_m\}$을 구함

단계 2: 정리 3.1에 의해, 시간단위 U와 기준시점 V가 주어졌을 때, 데이타베이스 D에서 LE를 가지는 트랜잭션만을 가지고 시간간격을 가진 사건시퀀스 $S' = \langle (E_1, vs_1, ve_1), (E_2, vs_2, ve_2), ..., (E_n, vs_n, ve_n) \rangle$를 생성

단계 3: 3.2절에 의해, S'로부터, 사용자가 지정한 윈도우 크기 w에 대해, 최소 발생도 $Freq_{\min}$ 이상을 가지는 균등 사건종류 집합 $UE=\{E_1, E_2, ..., E_l\}$을 구함.

단계 4: 정의 3.4, 정리 3.4에 의해, UE를 가지고 사건시퀀스 S'를 요약하여 일반화된 데이타베이스 ND를 생성

단계 5: 정의 3.2, 정의 3.5, 정리 3.2에 의해, ND로부터 모든 후보 시간관계 집합 $CR=\{R_1(x,y), R_2(x,y), ..., R_k(x,y)\}$를 구함.

단계 6: 3.3절에 의해, CR로부터 사용자가 지정한 최소 지지도 $Supp_{\min}$ 이상을 가지는 빈발 시간관계 집합 $FR=\{R_1(x,y), R_2(x,y), ..., R_z(x,y)\}$를 구함

단계 7: 정의 3.6에 의해, FR로부터 시간관계 규칙 $\{TR_1, TR_2, ..., TR_n\}$을 구함

4. 시간관계 규칙 탐사 기법

이 장에서는 3장의 문제 정의에 의거하여 시간관계 규칙을 탐사하기 위한 기법을 제시한다.

탐사 기법은 전처리 알고리즘과 규칙 탐사 알고리즘으로 구분하여 단계별로 설명한다. 전처리는 시간관계 규칙을 탐사하기 위해 필요한 시간간격 데이타를 생성하는 과정으로 발생 시점을 가진 시간 데이타를 시간간격 데이타로 요약하여 일반화한다. 전처리 알고리즘은 알고리즘 2의 단계1 부터 단계 4에 해당한다. 규칙 탐사는 시간간격 데이타로부터 빈발 시간관계를 구하여 시간관계 규칙을 생성하는 과정이다. 규칙 탐사 알고리즘은 알고리즘 2의 단계5 부터 단계 7에 해당한다.

그림 1은 발생시점을 가진 트랜잭션들로 구성된 데이타베이스로서 트랜잭션은 고객ID, 트랜잭션 시간, 사건종류 집합으로 구성된다. 그림 1은 전처리 과정을 거쳐 그림 5의 시간간격 데이타베이스로 변환된 후, 그림5로부터 그림 7과 같은 시간관계 규칙이 생성된다.

4.1 전처리 알고리즘

4.1.1 정렬 단계(Sort Phase)

이 단계는 입력 데이타베이스를 그림 1과 같이 고객ID와 트랜잭션 시간으로 정렬한다. 그림 1은 각 고객별, 트랜잭션 시간별로 정렬되어 있는 데이타베이스 D이다. 각 트랜잭션에는 월 단위로 발생한 사건종류가 저장된다.

4.1.2 빈발 사건종류 생성 단계(LE Phase)

이 단계는 알고리즘 2의 단계1에 해당하며 데이타베이스 D로부터 사용자가 지정한 최소 지지도 $Supp_{\min}$

Customer ID	Transaction Time	Event Type
101	99/1	D, B
101	99/2	D,A,E
101	99/3	A,E,B
101	99/5	C
101	99/7	E,C
102	99/1	B,E
102	99/4	B
102	99/6	C,E,G
102	99/9	C,E
103	99/1	B,D,F
103	99/3	B,E,F
103	99/4	B,F
103	99/7	E,C
103	99/10	E,C
104	99/1	A,D,F
104	99/4	C,F
104	99/5	C,G,E
104	99/12	C,E

그림 1 정렬된 데이타베이스

CE

Event Type	Support
A	2
B	3
C	4
D	3
E	4
F	2
G	2

그림 2-a 후보 사건 종류

최소지지도 ≥ 75%

LE

Event Type	Support
B	3
C	4
D	3
E	4

그림 2-b 빈발 사건종류

이상의 빈발 사건종류 집합, LE를 찾는다.

연관규칙 탐사[11]에서는 항목의 지지도를 항목이 존재하는 트랜잭션의 비율로서 결정하였지만 본 연구에서는 순차패턴[6]에서와 같이 사건종류가 발생한 고객 비율로서 지지도를 결정한다.

사건 사이의 시간관계를 찾는 문제는 전체 고객 관점에서 이루어져야 하기 때문이다. 그림 2-a는 D로부터 생성된 후보 사건종류 집합, $CE=\{A,B,C,D,E,F,G\}$이며 그림 2-b는 CE로부터 $Supp_{\min}$ 75% 이상을 만족하는 빈발 사건종류 집합, $LE=\{B,C,D,E\}$이다. 즉, 데이타베이스 D의 전체 고객 4명 중에서 3명 이상이 LE를 포함한다.

4.1.3 일반화 단계(Generalization Phase)

이 단계는 알고리즘 2의 단계2 부터 단계 4에 해당하며 그림 2-b의 빈발 사건종류 집합 LE를 이용하여 그림 1의 데이타베이스 D로부터 그림 5의 시간간격 데이타베이스를 생성한다.

시간간격 데이타로 변환하는데 있어 해결해야 할 문제는 다른 트랜잭션 시간에 발생한 동일한 여러 사건, 즉 같은 빈발 사건종류를 가진 사건들을 시간간격을 가진 한 사건으로 요약하여 일반화하는 문제이다. 이를 위해서는 요약을 위한 명확한 기준이 필요한데 [26]는 시간 지지도(time support) 개념을 제안하였다. 시간 지지도란 다음과 같이 정의된다.

임의의 사건종류 E_1에 대하여 시점이 다른 동일한 사건 집합 $\{(E_1,t_1),(E_1,t_2),\ldots,(E_1,t_m)\}$이 발생한다고 정의한다. 단, $t_j \le t_{j+1}$, $j=1,\ldots,m-1$이다.

이때 최초의 사건 (E_1,t_1)이 발생한 시간부터 마지막 사건 (E_1,t_m)이 발생한 시간까지의 기간 $[t_1,t_m]$을 E_1의 수명(lifespan)이라고 부른다. 시간지지도는 수명 $[t_1,t_m]$ 내에서 E_1이 발생한 횟수이다. 사용자가 지정한 최소 시간지지도를 만족하는 E_1은 기간 $[t_1,t_m]$ 내에서 지속적으로 발생한다고 간주하여 한 사건 (E_1,t_1,t_m)으로 요약할 수 있다. 예를 들어 E_1의 수명을 [1,6]이라 하고 최소 시간지지도를 50%, E_1의 시간지지도가 3이라 하면 E_1은 $(E_1,1,6)$으로 요약된다.

그러나 이 방법은 사건 E_1이 $[t_1,t_m]$ 사이의 특정 시간대에만 집중적으로 발생하고 꾸준히 지속적으로 발생하지 않더라도 요약되는 문제점을 가진다. 본 연구에서는 시간지지도의 문제점을 해결하기 위하여 다음과 같은 방법을 제안한다.

알고리즘 2의 단계2를 수행하여 시간단위 U와 기준 시점 V가 사용자에 의해 지정되었을 때, 그림 1의 데이타베이스 D를 시간간격을 가진 사건시퀀스 S'로 변환한다. 단, S'는 LE만을 포함한다. 정리 3.1에 의해 시간단위 U를 월, 기준 시점 V를 99년 1월로 주면 S'는 그림 3으로 표현된다. D의 트랜잭션 시간은 1부터 12까지의 정수값으로 변환되고 각 사건은 정수 시간간격을 가진다. 예를 들어 $\{B,D,E\}$의 시간간격은 [1,1]이다.

알고리즘 2의 단계3을 수행하여 그림 3의 사건시퀀스 S'로부터 사용자가 지정한 시간 윈도우 크기 w에 대해,

$Freq_{\min}$ 이상을 가지는 균등 사건종류 집합 $UE=\{B, C, E\}$를 구한다. 그림 4는 윈도우 크기 2에 대해 $Freq_{\min}$ 33% 이상을 가지는 UE이다. S'는 LE만을 포함하므로 LE의 일부만이 UE가 된다.

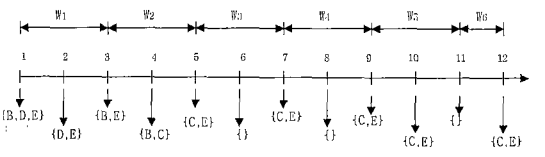

그림 3 시간간격을 가진 사건 시퀀스

정리 3.4에 의해, 균등 사건종류 UE를 가지고 사건 시퀀스 S'를 요약하여 일반화할 수 있다. 따라서 알고리즘 2의 단계4를 수행하여 그림 3의 사건시퀀스 S'를 그림 5와 같은 시간간격 데이타베이스 ND로 변환한다. ND는 사건종류 UE를 가진 시간간격 사건의 집합으로 구성된다.

이러한 방법을 통해 특정 시간대 에만 집중적으로 발생하는 사건을 가지치기하여 지속적으로 균등하게 발생하는 사건 만을 일반화한다. 단, 요약은 같은 고객의 트랜잭션들 내에서만 가능하다.

LE

Event Type	Frequency
B	2
C	5
D	1
E	6

최소발생도 ≥ 33%

UE

Event Type	Frequency
B	2
C	5
E	6

그림 4 균등 사건종류 집합

4.2 규칙 탐사 알고리즘

전처리 과정을 거쳐 생성된 시간간격 데이타베이스 ND로부터 빈발 시간관계 집합 FR을 구하여 시간관계 규칙을 생성하는 알고리즘이다.

4.2.1 시간관계 생성 단계(Temporal Relationship Phase)

이 단계는 알고리즘 2의 단계5에 해당하며 그림 5의 ND로부터 후보 시간관계 집합 $CR=\{R_1(B, C), R_2(C, E), \ldots, R_K(B, E)\}$를 구한다.

그림 6은 ND로부터 구해진 CR로서 트리 형태로 표현된다. 노드는 사건종류, 연결선은 두 사건 사이의 시간관계, 리프 노드(사각형)는 시간관계의 지지도를 나타낸다. 여기서 구해진 후보 시간관계는 $before(B, C)$, $during(C, E)$, $overlap(C, E)$, $overlap(B, E)$, $during(B, E)$이다. 시간관계 지지도는 데이타베이스 ND 내에서 시간관계가 발생한 고객의 수 이다. 트리 생성 방법은 다음과 같다. ND의 사건을 스캔하면서 기존 시간관계가 존재하면 지지도를 증가하고 새로운 시간관계가 존재하면 노드와 연결선을 추가하면서 동적으로 생성한다.

4.2.2 규칙 생성 단계(Rule Generation Phase)

이 단계는 알고리즘 2의 단계6부터 단계 7에 해당하며 그림 6의 CR트리로부터 사용자가 지정한 최소 지지도 $Supp_{\min}$ 이상을 가지는 빈발 시간관계 집합 $FR=\{R_1(x, y), R_2(x, y), \ldots, R_s(x, y)\}$를 구한 후, FR로부터 시간관계 규칙 $\{TR_1, TR_2, \ldots, TR_n\}$을 생성한다.

그림 7은 $Supp_{\min}$이 50%일때, 그림 6으로부터 구해진 FR과 $TR(B, C, E)$이다. FR은 $before(B, C)$, $during(C, E)$, $overlap(B, E)$이다. 정의 3.6에 의해 $TR(B, C, E)$는 FR의 연결 조합인 $before(B, C)$ | 0.75 ∧ $during(C, E)$ | 0.75 ∧ $overlap(B, E)$ | 0.5형태로 생성된다.

Customer ID	vs	ve	Event Type
101	1	3	B
101	5	7	C
101	2	7	E
102	1	4	B
102	6	9	C
102	1	9	E
103	1	4	B
103	7	10	C
103	3	10	E
104	4	12	C
104	5	12	E

그림 5 일반화된 시간간격 데이타베이스

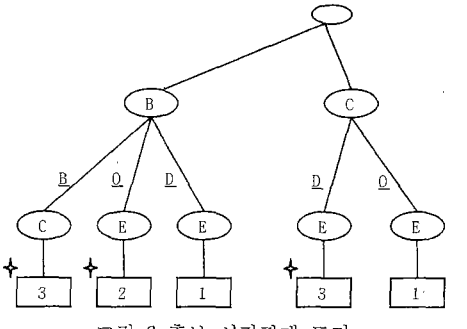

그림 6 후보 시간관계 트리

이는 그림 7에서 DAG(Directed Acyclic Graph) 형태로 표현할 수 있다. 노드는 사건종류, 연결선은 두 사건 사이에 발생한 빈발 시간관계를 나타낸다. DAG을 통해 시간관계 규칙을 보다 쉽게 이해할 수 있다.

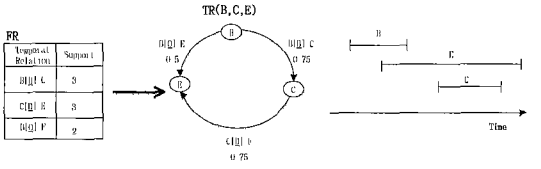

그림 7 빈발 시간관계 및 시간관계 규칙 그래프

지금까지 시간관계 규칙 탐사 기법을 전처리 알고리즘과 규칙 탐사 알고리즘으로 구분하여 단계별로 예를 들어 설명하였다. 알고리즘 3은 이 두 알고리즘을 가상 코드로 기술한 것이다.

알고리즘 3에서 그림 3의 사건시퀀스 S'는 트랜잭션 시간을 키로 하는 해쉬 테이블에 저장된다. 해쉬 테이블을 이용하여 동일한 시간을 가지는 사건들을 효율적으로 검색할 수 있다.

또한 그림 6의 후보 시간관계 트리도 시간관계를 키로 하는 해쉬 트리로 구현된다. 해쉬트리를 이용하여 시간관계를 효율적으로 검색, 생성할 수 있다.

알고리즘 4는 시간관계를 연산하기 위한 알고리즘이다. Derive_T_Relation은 정리 3.2에 의거해 기존 시간관계로부터 새로운 시간관계를 자동으로 구하는 유도함수이다. Interval_Compare는 정의 3.2에 의거하여 두 사건 간의 이항 시간관계를 구하는 시간간격 연산자 함수이다.

4.3 기존 기법과의 차이점

본 연구에서 제안한 시간관계 규칙 탐사 기법은 기존 데이타마이닝 기법과 다음과 같은 차이점을 가진다. 첫째, 시간간격 데이타로부터 기존 알고리즘에서 찾지 못하는 유용한 시간 규칙을 탐사할 수 있다. 예를 들어 그림 7의 시간관계 규칙 $before(B,C) \mid 0.75 \wedge during(C,E) \mid 0.75 \wedge overlap(B,E) \mid 0.5$는 다음과 같이 여러 분야에서 활용될 수 있다.

데이타베이스 D를 상품 판매 데이타라고 하면, 상품 B를 구매한 소비자의 75%는 이후 상품 C를 구매하고, 상품 E를 구매한 소비자의 75%는 상품 E를 구매하는 동안 상품 C를 구매하고, 상품 B를 구매한 소비자의 50%는 B의 구매기간이 끝나기 전에 상품 E를 중복하여 구매한다는 경향을 알 수 있다. 이러한 경향은 순차 패턴과 같은 기존 기법에서는 찾을 수 없다.

따라서 전자상거래에서의 고객 마케팅, 교차 상품 판매 등에 유용하게 활용될 수 있다. 데이터베이스 D를 환자 병력 데이타라고 하면 질병 B가 발생한 환자의 75%는 이후 질병 C가 발생하고, 질병 E가 발생하는 동안 질병 C가 발생할 확률은 75%이다. 또한 질병 B와 C는 50%의 확률로 중복되어 발생할 가능성이 있음을 알 수 있다. 따라서 환자의 질병 기간에 따라 다른 질병의 발생을 예측할 수 있어 사전 검사 권유 등에 활용될 수 있다. 이외에도 이력 데이타가 발생되는 여러 분야에서 지식 탐사를 위해 활용할 수 있다.

알고리즘 3 시간관계 규칙 탐사 알고리즘

```
1  LE ={빈발 시간종류} ,      //4.1.2 빈발 사건종류 생성 단계  빈발 시간종류 집합 LE를 생성
                              //4.1.3 일반화 단계  시간간격 데이타 생성
2  Convert_to_Sequence(D, U, V, S)  //데이타베이스 D로부터 시간간격 시퀀스 S 생성(정리 3.1),
                                    // S 은 트랜잭션시간을 키로 하는 해쉬 데이블에 저장
3  Count_per_time_window(S, w, LE); // 시퀀스 S 의 각 윈도우 w에서 LE의 발생 횟수를 계산
4  UE ={공통 시간종류}  // LE로부터 최소발생도 이상을 만족하는 공통 시간종류 집합 UE를 생성
5. for(k=1, end of D; k++) do  // 데이터베이스 D를 일반화하여 시간간격 데이터베이스 ND로 변환
   begin
     for each customer event e in the database D do
        Vs = min_time of event  // 사건의 시간간격 시작 시점 계산
        Ve = max_time of event  // 사건의 시간간격 종료 시점계산
        Write new database ND
   end
                                          // 1.2.1 시간관계 생성 단계
6  for(i=1, end of ND, i++) do // 시간간격 데이터베이스 ND로부터 모든 후보 시간관계를 구함
   begin
      for each customer in the database ND with ordered event x y in UE
         if (Derive_T_Relation(x, y))        // 시간관계 지정 유도(정리 3.2)
         else T_relation = Interval_compare(x, y),        // 시간관계 연산(정의 3.2)
         if(Search_hash_tree(T_relation, x y)) count++, // 기존 시간관계 지지도 증가
         else create_new_hash_tree(T_relation, x y), // 새 시간관계를 해쉬 트리에  추가
   end
7  FR = Check_Support(hash_tree, Sup_min ) //4.2.2 규칙 생성 단계: 시간관계 규칙 생성
```

전처리 알고리즘 / 규칙탐사 알고리즘

알고리즘 4 시간관계 연산 알고리즘

(a) 시간관계 유도 함수

```
Relation Derive_T_Relation(x, y)
begin
  if(Relation_Matrix(x, y)) //시간관계행렬 검색(표 1)
      return(T_relation); //행렬로부터 시간관계 유도
  else return(false);
end
```

(b) 시간간격 연산자 함수

```
Relation Interval_compare(x, y)
begin
  if (x.ve < y.vs)
            return(Before);
  if ((x.vs = y.vs) && (x.ve = y.ve))
            return(Equal);
  if (x.ve = y.vs)
            return(Meets);
  if ((x.vs < y.vs) && (x.ve > y.vs))
           return(Overlap);
  if ((x.vs > y.vs) && (x.ve < y.ve))
           return(During);
end
```

둘째, 대용량 데이타베이스로부터 시간관계 규칙을 효율적으로 탐사할 수 있다. 데이타베이스로부터 시간관계 규칙을 탐사하는 비용은 매우 크다. 예를 들어 그림 1의 데이타베이스 D가 n개의 사건을 가지고 있고 Allen 알고리즘(알고리즘 1)을 사용하여 시간관계를 탐사한다고 할때 정리 3.3에 의해 시간 복잡도는 $O(N^2)$이 된다. 본 연구에서는 $O(N^2)$의 시간 복잡도를 개선하기 위해 알고리즘 3과 같은 시간관계 탐사 알고리즘을 제안하였다. 이 알고리즘의 성능은 다음 5장에서의 실험 결과를 통해 크게 개선되었음을 알 수 있다.

5. 실험 및 결과

이 장에서는 알고리즘 3의 시간관계 규칙 탐사 알고리즘을 구현한 후, 전처리 과정을 통해 실험 데이타를 생성하고 실험 데이타로부터 시간관계 규칙을 탐사해 보았다. 또한 규칙 탐사에 소요된 시간을 분석하여 이 알고리즘의 성능을 분석하였다.

5.1 실험 환경 및 데이타 생성

실험에 사용한 시스템은 Pentium-III 550MHZ, Dual CPU를 장착한 윈도우 NT 서버이며 실험 데이타의 저장은 ORACLE DBMS를 이용하였다. 알고리즘의 구현 언어는 JAVA이며 개발 도구로서 JDK 1.2, ORACLE과의 연동을 위해 JDBC 드라이버를 사용하였다.

그림 8은 알고리즘을 구현한 화면이다. 알고리즘 실행을 위한 입력 값은 시간단위, 최소 지지도, 최소 발생도, 시간 윈도우 크기이다. 출력 결과는 두개의 윈도우로 구성되는데 좌측 윈도우는 시간관계 탐사 알고리즘의 각 단계에서 생성된 중간 결과를 보여주며, 우측 윈도우는 실험데이타로부터 생성된 시간관계 규칙으로 그림 7의 시간관계 규칙 그래프 형태로 표현된다.

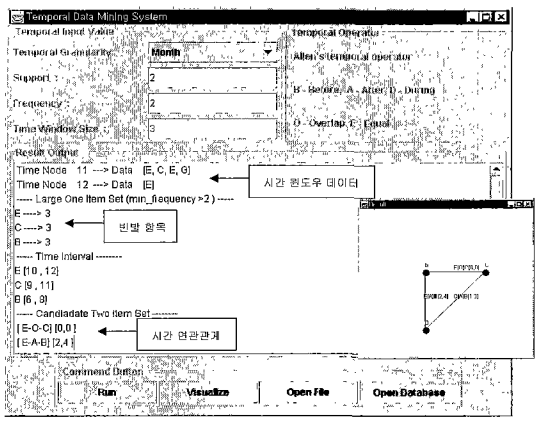

그림 8 알고리즘 구현 화면

실험 데이타는 그림 1과 같이 트랜잭션 시간을 가지고 있어야 하므로 전자상거래 쇼핑몰의 거래 데이타를 수집하여 사용하였다. 실험 데이타는 충분한 시간간격 데이타를 생성하기 위해 시간대 별로 트랜잭션이 균등하게 분포되어 있는 데이타를 대상으로 하였고, 많은 규칙을 생성하기 위해 고객 지지도가 높은 데이타를 대상으로 하였다. 실험에 충분한 실험 데이타를 생성하기 위해 그림 8의 화면에서 전처리 과정을 반복하였다. 생성된 실험 데이타는 약 10,000건의 트랜잭션을 가지며 각 트랜잭션은 시간간격을 가진다.

5.2 성능 분석

본 연구의 목적은 시간간격 데이타로부터 시간관계 규칙을 탐사하기 위한 기법을 연구하는 것이다. 그러므로 본 연구에서 제안한 알고리즘과 시간간격 데이타를 대상으로 하지 않는 기존 알고리즘 간에 성능을 비교, 분석하는 것은 적절하지 못하다. 따라서 다음과 같은 방법으로 알고리즘의 성능을 분석하였다.

첫째, 실험 데이타 건수를 증가해 가면서 Allen 알고리즘과 본 알고리즘을 반복하여 실행하고 건수별로 소요 시간을 측정하여 알고리즘의 시간복잡도를 비교, 분석하였다. 그 결과, 그림 9와 같이 Allen 알고리즘은 건수가 증가하면 소요시간이 급격히 증가하여 시간복잡도가 정리 3.3의 $O(N^2)$에 근접한 반면, 본 알고리즘의 소요시간은 건수에 비례하여 완만하게 증가하므로 Allen 알고리즘 보다는 크게 개선되었음을 알 수 있다. 그러나 건수에 비례하여 소요시간이 지속적으로 증가하므로 이를 줄이기 위한 연구가 필요하다.

둘째, 알고리즘의 각 단계별로 소요 시간을 계산하여 어느 단계가 가장 많은 비용이 소요되는지 분석하였다. 그림 10은 실험데이타로부터 규칙을 생성하였을때 알고

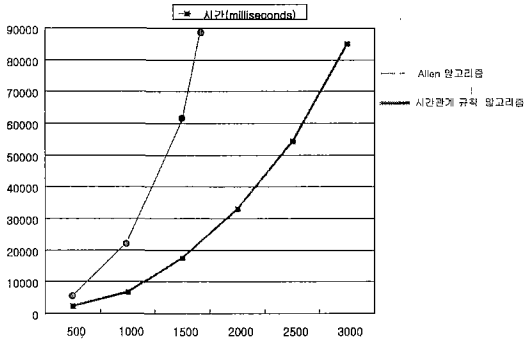

그림 9 데이타 건수별 규칙 탐사 시간

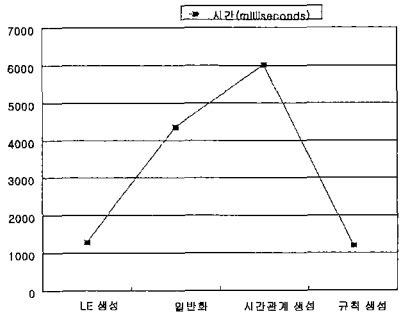

그림 10 알고리즘의 단계별 소요 시간

리즘의 각 단계별 평균 소요 시간을 비교한 것이다. 시간관계 생성 단계와 일반화 단계가 가장 많은 시간을 소요함을 알 수 있다. 따라서 이 단계에 대한 성능 개선 연구가 필요하다.

셋째, 기존의 연관규칙과 순차패턴 탐사 알고리즘은 후보 항목 집합을 생성하는데 많은 비용이 소요된다. 반면 이 알고리즘은 그림 10과 같이 시간관계 생성 단계에서 후보 시간관계를 생성하는데 가장 많은 비용이 소요되며, 규칙 생성 단계에서 후보 시간관계로부터 빈발 시간관계를 찾고 규칙을 생성하는 비용은 적다. 따라서 생성된 규칙의 갯수에 대한 소요 시간을 분석해 보았다. 그 결과, 그림 11과 같이 생성된 규칙 수와 소요 시간은 비례 하지만 특히 규칙 수가 약 1,000개를 넘어가면 소요시간이 급격히 증가함을 알 수 있었다. 따라서 규칙 수를 1,000개 이하로 줄일 수 있도록 적절한 입력값, 즉 시간단위, 최소 지지도, 최소 발생도, 윈도우 크기를 지정하는 것이 필요하다.

넷째, 규칙 수를 1,000개 이하로 줄이기 위해서 사용자가 지정한 최소 지지도에 따라 생성된 규칙 수를 조사하여 이 알고리즘에 가장 적합한 지지도를 찾아보았다. 그 결과, 그림 12와 같이 평균적으로 0.25와 0.35사이의 지지도에서 1,000개 이하의 규칙을 생성함을 알 수 있었다.

이러한 결과 외에도, 실험 과정에서 나타난 이 기법의 가장 큰 문제점은 일반화 방법에 있음을 알 수 있었다. 즉, 일반화를 위해 균등 사건종류를 구하는 것은 일부 시간대에서만 집중적으로 발생하는 사건이 아니라 전 기간에 걸쳐 지속적으로 균등하게 발생하는 사건종류를 구하고자 함인데, 실험 데이타를 일반화하는 과정에서 시간대 별로 트랜잭션이 균등하게 분포되어 있지 않은 데이타의 경우에는 일반화가 정확히 이루어지지 않았다. 그 이유를 분석해 보면 그림3의 시퀀스 S'으로부터 균등사건종류집합 $UE=\{B, C, E\}$를 구할 때 각 사건의 사용자별 발생 빈도를 무시하고, 전체 시간구간에서의 사건의 발생 빈도만을 고려함으로써 문제가 발생하였음을 알 수 있었다. 예를 들면 $\{B, C, E\}$가 그림 1의 데이타베이스에서 고객 101에 의해서만 집중적으로 발생하고 나머지 고객들에 의해서는 적게 발생될 경우에 이를 모두 일괄적으로 그림 5의 ND로 요약하기에는 문제가 있다.

그러므로 이러한 문제점을 해결하기 위해서는 윈도우를 그림 3과 같이 고정된 크기의 시간간격으로 정하는 것보다는 각 사건의 사용자별 발생 빈도를 고려하여 가변적으로 정한 후 이에 따라 UE를 구하는 것이 바람직하다. 또한 UE를 이용하여 ND로 요약할 때 각 사용자별로 다른 기준을 적용하여 요약할 필요가 있다. 따라서 본 논문에서 제안한 전처리 알고리즘을 윈도우 크기의 가변적인 결정 및 사용자별 요약이 가능하도록 개선하는 연구가 필요하다.

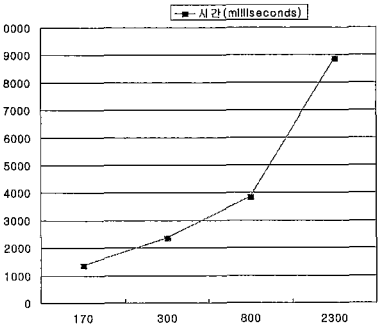

그림 11 규칙 갯수별 소요 시간

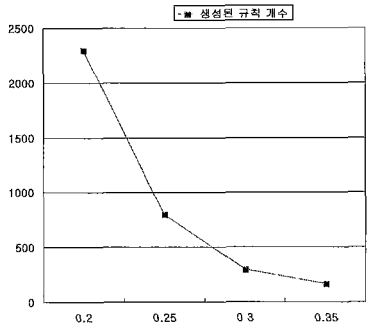

그림 12 최소지지도별 생성 규칙 수

6. 결론 및 향후 연구 방향

시간 데이타마이닝에 대한 기존 연구는 트랜잭션의

발생 시점만을 가진 시간 데이타를 다루고 있으며 시간 간격을 가진 데이타는 거의 고려하고 있지 않다. 그러나 실세계에서는 환자의 병력, 상품 구매 이력, 웹 로그 등과 같은 다양한 시간간격 데이타가 존재하며 이로부터 여러 유용한 지식을 찾아낼 수 있다. 따라서 본 연구에서는 Allen의 시간간격 연산자를 기반으로 시간간격 데이타로부터 유용한 시간관계 규칙을 효율적으로 탐사하기 위한 새로운 데이타마이닝 기법을 제안하였다.

이 기법은 발생 시점을 가진 시간 데이타를 시간간격 데이타로 요약하여 일반화하는 전처리 알고리즘과 시간간격 데이타로부터 시간관계 규칙을 탐사하는 알고리즘으로 구성되었다. 전처리는 시간관계 규칙을 탐사하기 위해 필요한 시간간격 데이타를 생성하는 과정으로 발생 시점을 가진 시간 데이타를 시간간격 데이타로 요약하여 일반화한다. 규칙 탐사는 전처리 알고리즘에서 생성된 시간간격 데이타로부터 빈발 시간관계를 구하여 시간관계 규칙을 생성하는 과정이다. 이 기법은 기존 연구와 비교하여 첫째, 시간간격 데이타로부터 순차패턴 등과 같은 기존 알고리즘에서 찾지 못하는 유용한 시간 규칙을 탐사할 수 있으며 둘째, 시간 데이타베이스로부터 시간관계 규칙을 효율적으로 탐사할 수 있다. 즉, 일반화를 통해 시간간격 데이타를 생성할 수 있을 뿐만 아니라 데이타 크기를 줄여 탐색 공간 및 시간을 줄일 수 있다.

본 연구에서 제시된 알고리즘의 이와 같은 특성을 검증하기 위하여 데이타 건수에 따른 탐사 시간과 알고리즘의 단계별 소요 시간, 규칙 갯수별 소요 시간 등을 실험하여 분석하였다. 그 결과는 다음과 같다. 그림 9와 같이 알고리즘의 소요시간은 건수에 비례하여 증가하나 완만하게 증가하는 것으로 분석되었으므로 Allen 알고리즘의 $O(N^2)$보다는 크게 개선되었다. 그러나 이 알고리즘을 구성하는 단계 중에서 시간관계 생성 단계와 일반화 단계가 가장 많은 시간을 소요하며, 생성된 규칙 수와 소요 시간은 비례 하지만 특히 규칙 수가 약 1,000개를 넘어가면 소요시간이 급격히 증가한다는 문제점이 발견되었다. 또한 평균적으로 1,000개 이하의 규칙이 생성되는 0.25와 0.35사이의 지지도가 이 알고리즘의 가장 적합한 최소 지지도임이 확인되었다.

본 연구에서 보완해야 할 점은 실험에서 확인된 문제점을 개선하고 시간대 별로 트랜잭션이 균등하게 분포되어 있지 않은 데이타의 경우에도 정확한 일반화가 이루어지도록 일반화 기법을 개선하는 일이다. 향후 연구 방향은 시간관계 탐사 성능을 향상하기 위한 알고리즘 및 저장 구조, 유용성 관점에서의 시간관계 탐사 기법 연구 등이다.

참 고 문 헌

[1] J. F. Roddick, K. Hornsby and M. Spiliopoulou, Temporal, Spatial and Spatio-Temporal Data Mining and Knowledge Discovery Research Bibliography, http://www.cs.flinders.edu.au, 2000.

[2] J. F. Roddick and M. Spiliopoulou, "Temporal data mining: survey and issues," Research Report ACRC-99-007, University of South Australia, 1999.

[3] J. Allen, "Maintaining Knowledge about Temporal Intervals," Comm. Of the ACM, Vol.26, No.11, Nov. 1983.

[4] J.Y. Lee, K.J. Oh, K.H. Ryu, "Integration with Spatiotemporal Relationship Operators in SQL," ACM-GIS, 1998.

[5] K.W. Nam, D.H. Kim, K.H. Ryu, "The Spatiotemporal Relationship Operator," ITC-CSCC, 1996.

[6] R. Agrawal and R. Srikant, "Mining sequential patterns," 11th International Conference on Data Engineering, Taipei, Taiwan, Mar. 1995.

[7] X. Chen and I. Petrounias, "A framework for temporal data mining," 9th International Conference on Database and Expert Systems Applications, 1998.

[8] C. Rainsford and J. F. Roddick, "Temporal data mining in information systems: a model," 7th Australasian Conference on Information Systems, 1996.

[9] M. H. Saraee and B. Theodoulidis, "Knowledge discovery in temporal databases," IEEE Colloquium on Knowledge Discovery in Databases, 1995.

[10] S. Ye and J.A Keane, "Mining association rules in temporal databases," International Conference on Systems, Man and Cybernetics, 1998.

[11] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," the VLDB Conference, Santiago, Chile, September 1994.

[12] R. Srikant and R. Agrawal, "Mining sequential patterns: generalisations and performance improvements," In Proc. International Conference on Extending Database Technology, Avignon, France, Springer-Verlag, 1996.

[13] Minos N. Garofalakis, Rajeev Rastogi and Kyuseok Shim, "SPIRIT: Sequential Pattern Mining with Regular Expression Constraints," the VLDB Conference, Edinburgh, Scotland, UK, 1999.

[14] H. Mannila and H. Toivonen, "Discovering generalized episodes using minimal occurrences,"

Int'l Conference on Knowledge Discovery in Databases and Data Mining (KDD-96), Portland, USA, Aug. 1996.
[15] H. Mannila, H. Toivonen, and A. I. Verkamo, "Discovery of frequent episodes in event sequences," Data Mining and Knowledge Discovery, Vol.1, No.3, Nov. 1997.
[16] G. Berger and A. Tuzhilin, "Discovering unexpected patterns in temporal data using temporal logic," Temporal Databases - Research and Practice, Springer-Verlag, 1998.
[17] S. Chakrabarti, S. Sarawagi, and B. Dom., "Mining surprising patterns using temporal description length," the VLDB Conference, New York City, USA, Aug. 1998.
[18] J. Han, G. Dong, and Y. Yin, "Efficient Mining of Partial Periodic Patterns in Time Series Database," 15th International Conference on Data Engineering, Sydney, Australia, 1999.
[19] R. Agrawal, G. Psaila, E. Wimmers, M. Zaot, "Querying shapes of histories," the VLDB Conference, Zurich, Switzerland, Sept. 1995.
[20] R. Agrawal, King-Ip Lin, Harpreet S. Sawhney, and Kyuseok Shim, "Fast similarity search in the presence of noise, scaling, and translation in time series databases," the VLDB Conference, Zurich, Switzerland, Sept. 1995.
[21] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," the ACM SIGMOD Conference on Management of Data, Minneapolis, USA, May. 1994.
[22] Kyuseok Shim, R. Srikant and R. Agrawal, "High-dimensional similarity joins," the 13th International Conference on Data Engineering, Brmingham, U.K., Apr. 1997.
[23] B. Ozden, S. Ramaswamy, and A. Silberschatz, "Cyclic association rules," the 14th International Conference on Data Engineering, Orlando, USA, 1998.
[24] X. Chen, I. Petrounias and H. Heathfield, "Discovering temporal association rules in temporal databases," International Workshop on Issues and Applications of Database Technology, 1998.
[25] S. Ramaswamy, S. Mahajan and A. Silberschatz, "On the discovery of interesting patterns in association rules," the VLDB Conference, New York City, USA, Sept. 1998.
[26] J. M. Ale, G. H. Rossi, "An Approach to Discovering Temporal Association Rules," SAC'00, Italy, 2000.
[27] C. Rainsford, Accommodating Temporal Semantics in Knowledge Discovery and Data Mining, PhD Thesis, University of South Australia, 1998.
[28] C. S. Jensen, et al, "A Glossary of Temporal Database Concepts," ACM SIGMOD Record, Vol.23, No.1, 1994.
[29] R. Snodgrass, "The Temporal Query Language TQuel," ACM TODS, Vol.12, No.2, 1987.

이 용 준
1984년 광운대학교 전산학과(학사). 1987년 연세대학교 전산학과(석사). 1993년 정보처리기술사(전자계산조직응용). 2001년 충북대학교 대학원 전산학과(박사). 1984년 ~ 현재 한국전자통신연구원 우정기술연구부 책임연구원. 관심분야는 시간 데이타베이스, 데이타 마이닝, 데이터베이스 보안

서 성 보
1999년 서원대학교 전산학과 졸업(공학사). 2001년 충북대학교 대학원 전산학과(이학석사). 관심분야는 데이타 마이닝, e-비즈니스 모델, XML, 시공간 데이타베이스 등

류 근 호
1976년 숭실대 전산과 졸업. 1980년 연세대학교 산업대학원 전산전공(공학석사). 1988년 연세대 대학원 전산전공(공학박사). 1976년 ~ 1986년 육군군수지원사전산실(ROTC장교), 한국전자통신연구소(연구원), 한국방송통신대 전산학과(조교수) 근무. 1989년 ~ 1991년 Univ. of Arizona 연구원(TempIS Project). 1986년 ~ 현재 충북대학교 전기전자컴퓨터공학부 교수. 관심분야는 시간 데이타베이스, 시공간 데이타베이스, 객체 및 지식베이스 시스템, 지식기반 정보검색시스템, 데이터 마이닝, Bio-Informatics

김 혜 규
1973년 서울대학교 공과대학 응용물리학과(학사). 1985년 서강대학교 경영학과(석사). 1995년 서강대학교 정보처리학(석사). 1979년 ~ 현재 한국전자통신연구원 우정기술연구부장 책임연구원. 관심분야는 물류 정보 시스템, 물류 최적화 시스템, e-비즈니스 모델, 정보 정책