버전 기반의 공간 레코드 관리를 위한 쓰레기 처리 기법

김희택[0], 김명근, 김호석, 배해영
인하대학교 컴퓨터·정보공학과
{htkim[0], kimmkeun, hskim, hybae}@dblab.inha.ac.kr

Garbage Collection Techniques for Version Based Spatial Record Management

Hee-Taek Kim[0], Myoung-Keun Kim, Ho-Seok Kim, Hae-Young Bae
Dept. of Computer Science & Information Engineering, Inha University

요 약

다중버전 알고리즘은 다수의 버전에 대한 접근을 통해 검색 연산이 갱신 연산 때문에 대기하거나, 갱신 연산이 검색 연산 때문에 대기하는 문제점을 제거하여 트랜잭션의 동시성을 향상시킨다. 이런 다중버전 알고리즘을 바탕으로 공간 데이터베이스 관리 시스템을 위한 버전 기반의 공간 레코드 관리 기법이 제안되었다. 버전 기반의 공간 레코드 관리 기법은 공간 레코드의 속성 데이터 버전과 공간 데이터 버전을 따로 생성 및 관리하는 기법이다. 하지만 하나의 공간 레코드를 위하여 여러 개의 속성 데이터 버전과 공간 데이터 버전을 계속 유지하기 때문에 저장 공간의 부하가 존재한다.

본 논문에서는 버전 기반의 공간 레코드 관리 기법에서 저장 공간의 부하를 최소화하기 위해서 검색 트랜잭션이 더 이상 사용하지 않는 공간 레코드 버전을 찾아 제거하는 기법을 제안한다. 본 기법은 트랜잭션 완료시 제거될 버전의 후보를 선정한 후, 진행중인 트랜잭션의 타임스탬프와 제거될 버전의 타임스탬프를 비교하여 향후 검색 트랜잭션이 사용하지 않는 버전을 제거하여 저장 공간의 부하를 최소화 하는 기법이다.

1. 서론

다중버전 알고리즘에서 버전은 갱신 트랜잭션 연산에 의해서 생성된 레코드의 복사본을 의미한다. 하나의 레코드에 대해 여러 개의 버전이 존재하기 때문에 검색 트랜잭션과 갱신 트랜잭션은 각각 다른 버전을 사용할 수 있다. 결국 다수의 버전에 대한 접근을 통해 검색 연산이 갱신 연산 때문에 대기하거나, 갱신 연산이 검색 연산 때문에 대기하는 문제점을 제거하여 트랜잭션의 동시성을 향상시킬 수 있다.

이런 다중버전 알고리즘을 바탕으로 [1]에서는 공간 데이터베이스 관리 시스템을 위한 버전기반의 공간 레코드 관리 기법을 제안하였다. 버전 기반의 공간 레코드 관리 기법은 공간 레코드의 속성 데이터 버전과 공간 데이터의 버전을 따로 생성 및 관리하는 기법이다.

그러나 버전 기반의 공간 레코드 관리 기법이 다중버전 알고리즘을 기반으로 제안되었기 때문에 하나의 공간 레코드를 위하여 여러 개의 속성 데이터 버전과 공간 데이터 버전을 계속 유지해야 함으로 저장 공간의 부하가 존재한다. 따라서 저장 공간의 부하를 최소화하기 위해서는 오래된 공간 버전을 찾아 이를 제거해야만 한다. 제거되는 공간 레코드 버전은 더 이상 검색 트랜잭션이 사용하지 않는 버전이다.

본 논문[1]에서는 버전 기반의 공간 레코드 관리 기법에서 저장 공간의 부하를 최소화하기 위해서 현재 진행중인 트랜잭션의 타임스탬프와 제거 가능한 버전의 타임스탬프를 비교하여 더 이상 검색 트랜잭션이 사용하지 않는 오래된 버전을 찾아 제거하는 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2장 관련 연구에서는 다중버전 알고리즘과 쓰레기 수집기 (Garbage Collector)에 대한 기존 연구에 대해 살펴본다. 3장에서는 본 논문에서 제안하는 버전 기반의 공간 레코드 관리 기법에서 더 이상 검색 트랜잭션이 사용하지 않는 오래된 버전을 찾아 제거하는 기법을 기술한다. 마지막으로 4장에서 결론 및 향후 연구를 기술한다.

2. 관련 연구

이 장에서는 다중버전 알고리즘과 저장 공간의 낭비를 최소화하기 위하여 사용하지 않는 버전을 제거하는 기존 연구에 대해 기술한다.

다중버전 동시성 제어에 관한 논의는 [2]에서 다루었으며 다중버전 트리-잠금 알고리즘은 [3]에서 다루었다. 다중버전 타임스탬프 순위는 [4]와 [5]에서 소개되었다. 이는 다중버전 알고리즘에 의해 사용되는 트랜잭션의 순위를 정하기 위해 타임스탬프를 사용하는 기법이다. 다중버전 2단계 잠금은 [6]에서 설명되고 있다. 이는 다중버전 동시성 제어의 장점과 2단계 잠금의 장점을 결합한 것으로, 갱신 트랜잭션들은 잠금을 사용하여 완료되는 시점에 따라 직렬화 된다.

더 이상 사용하지 않는 오래된 버전을 제거하기 위해서 [7]에서는 별도의 저장소에서 각각의 버전을 연결 리스트로 유지하며 더 이상 사용되지 않는 버전을 제거한다. 하지만 버전을 페이지 단위로 설계하였기 때문에 더 이상 사용되지 않는 버전을 제거하는 쓰레기 수집기는 단순히 디스크 I/O만 고려하여 설계되었다.

[8]는 [7]를 확장하여 같은 페이지에 버전을 군집화하는 기법을 사용하여 레코드 단위의 버전을 제시하였다.

또한 [9]에서는 레코드 단위의 논리적 버전 기법과 인덱스를 위한 페이지 단위의 물리적 버전 기법을 제시하며 더 이상 사용하지 않는 레코드의 버전, 페이지의 버전을 제거하기 위하여

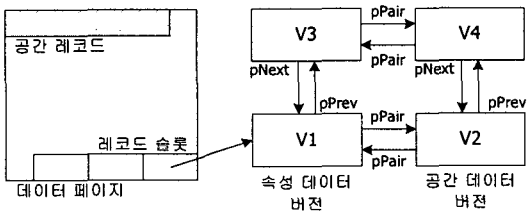
1) 본 연구는 대학 IT연구센터 육성 · 지원사업의 연구결과로 수행되었음

Ager라는 쓰레기 수집기를 사용하는 기법을 설명하고 있다.

3. 공간 레코드 버전의 쓰레기 처리 기법

이 장에서는 버전 기반의 공간 레코드 관리 기법에 대하여 살펴본 후, 본 논문에서 제안하는 버전 기반의 공간 레코드 관리를 위한 쓰레기 처리 기법을 기술한다.

3.1 버전 기반의 공간 레코드 관리 기법



[그림 3-1] 속성 데이터 버전과 공간 데이터 버전의 분리

버전 기반의 공간 레코드 관리 기법의 전체적인 모습은 [그림 3-1]과 같다. 공간 레코드 버전은 속성 데이터 버전과 공간 데이터 버전으로 분리하여 생성 및 관리되며, 공간 레코드의 버전이 존재할 경우 데이터 페이지의 각 레코드 슬롯은 최근 생성된 속성 데이터 버전을 가리키고 있다.

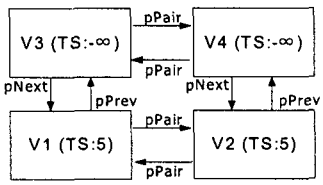
하나의 공간 레코드 버전은 속성 데이터 버전과 공간 데이터 버전의 쌍으로 구성되며, 반드시 속성 데이터 버전과 공간 데이터 버전이 1:1의 관계로 연결되는 것은 아니다. 여러 개의 속성 데이터 버전이 하나의 공간 데이터 버전과 연결되어 레코드를 구성할 수 있으며 자세한 내용은 [1]에서 기술하고 있다.

3.2 공간 레코드 버전의 쓰레기 처리 기법

이 장에서는 버전 기반의 공간 레코드 관리 기법에서 완료 연산시 제거될 버전의 후보를 선정하고, 제거될 후보 버전 중 제거가 가능한 쓰레기 버전을 조사하여 쓰레기 버전을 제거하는 기법을 기술한다.

3.2.1 트랜잭션 완료시 제거될 버전의 후보 선정

공간 레코드의 트랜잭션 완료시 [1]에서 기술된 내용처럼 먼저 시스템으로부터 증가된 타임스탬프 T_i을 부여 받은 후, 갱신 트랜잭션으로 생성된 모든 버전의 타임스탬프를 ∞에서 타임스탬프 T_i으로 변경한다. 또한 트랜잭션 완료로 새로운 완료 버전이 생성되었으므로 이전에 생성된 모든 완료 버전은 제거 되어야 한다. 이를 위하여 본 논문에서는 제거될 버전의 후보를 선정하기 위해 큐 구조의 제거 목록을 사용한다. 갱신 트랜잭션 이전에 생성된 모든 완료 버전을 타임스탬프 T_i과 함께 쓰레기 수집기의 제거 목록에 추가함으로 제거될 후보를 선정한다.



제거 목록

TS	Version
5	V3
5	V4

[그림 3-2] 트랜잭션의 완료시 이전 버전을 제거 목록에 추가

예를 들어 [그림 3-2]에서는 갱신 트랜잭션으로 생성된 V1, V2는 트랜잭션 완료시 타임스탬프가 ∞에서 시스템으로부터 부여 받은 타임스탬프 5로 설정되며, 이전에 생성된 완료 버전 V3, V4는 타임스탬프 5로 제거 목록에 추가된 모습이다. 즉, 제거 목록에 추가된 버전은 제거될 버전의 후보이며 자신을 제거 목록에 추가한 트랜잭션의 타임스탬프로 정렬되어 있다.

3.2.2 제거 목록에 추가된 버전의 제거 가능 조사

위에서 언급한 것처럼 제거 목록에 있는 버전들은 삭제될 후보 버전들이다. 하지만 제거 목록에 버전이 추가 되었더라도 현재 진행중인 검색 트랜잭션이 제거 목록에 추가된 버전을 사용할 수 있으므로 제거 목록에 추가된 버전을 바로 제거할 수 없다. 따라서 본 논문에서는 제거 목록에 추가된 버전의 제거 가능을 조사한다.

쓰레기 수집기는 제거 목록에 추가된 버전의 타임스탬프와 현재 진행중인 트랜잭션들의 타임스탬프 중 가장 작은 타임스탬프와 비교한다. 만약 제거 목록에 추가된 버전의 타임스탬프가 현재 진행중인 트랜잭션의 타임스탬프 중 가장 작은 타임스탬프보다 클 경우 쓰레기 수집기는 제거 목록에 추가된 버전을 제거할 수 없다. 현재 진행중인 트랜잭션이 제거 목록에 추가된 버전을 읽을 수 있는 가능성이 있기 때문이다.

트랜잭션 TS

TID	TS
34	23
41	26
47	37

제거 목록

TS	Version
32	V1
33	V2

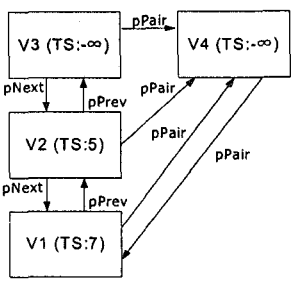
[그림 3-3] 제거 목록에 있는 버전을 제거할 수 없는 경우

[그림 3-3]에서처럼 현재 진행중인 트랜잭션들의 타임스탬프 중 가장 작은 타임스탬프가 23이고, 제거 목록에 추가된 타임스탬프가 32일 경우, 제거 목록에 추가된 타임스탬프가 가장 작은 타임스탬프보다 크므로 제거 목록에 추가된 버전을 제거할 수 없다. 타임스탬프 32보다 작은 트랜잭션 34, 41이 버전 V1을 읽을 수 있기 때문이다.

3.2.3 쓰레기 버전의 제거

본 논문에서는 제거가 가능한 버전을 쓰레기 버전이라 하며 쓰레기 버전의 제거는 다음 3가지 경우로 나뉜다.

1) 쓰레기 버전 이후에 생성된 버전이 없거나 두개 이상 존재할 경우 쓰레기 버전은 서로 연결된 다른 버전과 연결을 해제한 후 쓰레기 버전을 제거한다. 이후에 생성된 버전은 나중에 쓰레기 수집기에 의해 제거가 가능하므로 제거 목록에 있는 쓰레기 버전만 제거한다.



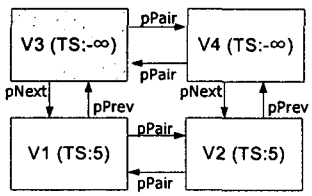
제거 목록

TS	Version
5	V3
7	V2

[그림 3-4] 여러 개의 버전이 존재할 경우의 버전 삭제

예를 들어 [그림 3-4]에서 제거 목록에 있는 버전 V3이 쓰레기 버전이고, V3 이후에 생성된 버전이 V2와 V1 두 개가 존재하므로 V3은 서로 연결된 V2, V4와의 연결을 해제한 후 제거된다.

2) 쓰레기 버전 이후에 생성된 버전이 하나만 존재하고, 이후에 생성된 버전의 pPair에 존재하는 버전이 두개 이상 존재할 경우 쓰레기 버전은 서로 연결된 다른 버전과 연결을 해제한 후 쓰레기 버전을 제거한다. pPair에 존재하는 버전은 나중에 쓰레기 수집기에 의해 제거가 가능하므로 제거 목록에 있는 쓰레기 버전만 제거한다.



제거 목록

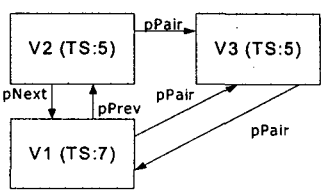
TS	Version
5	V3
5	V4

[그림 3-5] pPair의 버전이 두개 이상 존재할 경우

[그림 3-5]의 경우 버전 V3이 쓰레기 버전이고, V3 이후에 생성된 버전이 V1 하나만 생성되었으며, V1의 pPair에 존재하는 데이터 버전이 V2, V4 두개가 존재하기 때문에 V3은 V3과 서로 연결된 V1, V4와의 연결을 해제한 후 제거된다.

3) 쓰레기 버전 이후에 생성된 버전이 하나만 존재하고, 이 후에 생성된 버전의 pPair에 존재하는 버전이 한개만 존재할 경우 시스템으로부터 증가된 타임스탬프 T_i을 할당 받아 쓰레기 버전 이후에 생성된 버전과 pPair의 버전을 T_i과 함께 제거 목록에 추가한 후 쓰레기 버전을 제거한다. 쓰레기 버전 이후에 생성된 버전이 각각 하나만 존재할 경우, 레코드의 완료 버전이 하나만 존재하므로 더 이상 버전이 필요하지 않다. 하지만 해당 버전들은 다른 어떤 트랜잭션에 의해 제거 목록에 추가되지 않기 때문에 불필요하게 저장 공간에 남아 있다. 따라서 쓰레기 버전 이후에 생성된 버전이 각각 하나씩만 존재할 경우, 저장 공간의 부하를 최소화하기 위해 남아 있는 모든 버전을 제거 목록에 추가한다.

또한 쓰레기 버전 이후에 생성된 버전이 데이터가 없는 버전일 경우, 즉 레코드의 데이터가 없는 지워진 버전일 경우에는 데이터 페이지에서 해당 레코드의 레코드 슬롯을 삭제한다. 레코드의 삭제로 데이터 페이지에 더 이상 레코드가 존재하지 않는다면 해당 페이지 또한 삭제한다. 본 논문에서는 페이지 단위의 삭제는 다루지 않는다.



제거 목록

TS	Version
7	V2
8	V1
8	V3

[그림 3-6] pPair의 버전이 한개만 존재할 경우

[그림 3-6]은 버전 V2가 쓰레기 버전이고, V2의 이후에 V1 하나만 생성되었으며, V1의 pPair에 존재하는 데이터 버전이 V3만 존재한다. 레코드의 완료 버전이 V1, V3으로 하나만 존재하므로 더 이상 레코드의 버전이 필요하지 않다. 하지만 이 버전들은 다른 트랜잭션에 의해 제거 목록에 추가되지 않아 제거되지 않으므로 시스템으로부터 부여된 타임스탬프 8과 함께 V1, V3을 제거 목록에 추가한다. 또한 V2는 V2와 연결된 V1, V3과의 모든 연결을 해제한 후 제거된다.

4. 결론 및 향후 연구

본 논문에서는 버전 기반의 공간 레코드 관리 기법에서 더 이상 검색 트랜잭션이 사용하지 않는 오래된 버전을 찾아 제거하는 기법을 기술하였다. 트랜잭션의 완료 연산시 완료 연산 이전에 생성된 버전을 완료 트랜잭션의 타임스탬프와 함께 제거 목록에 추가하여, 제거 목록에 추가된 버전이 더 이상 사용되지 않을 때 버전을 삭제하였다.

따라서 검색 트랜잭션이 사용하지 않는 오래된 공간 레코드의 버전을 주기적으로 제거하기 때문에 저장 공간의 부하를 최소로 줄일 수 있었다.

향후 연구로는 사용되지 않는 공간 데이터 페이지의 효율적인 제거 알고리즘이 연구되어야 한다.

5. 관련 논문

[1] 김희택, 김명근, 김호석, 배해영, "공간 데이터베이스 관리 시스템을 위한 버전 기반의 공간 레코드 관리 기법", 정보과학회 2004년 춘계학술대회, VOL. 31 NO. 01, pp. 76-78, 2004.

[2] P.A. Bernstein, N. Goodman, and M. Y. Lai, "Analyzing Concurrency Control when User and System Operations Differ", IEEE Transactions on Software Engineering, Volumn SE-9, Number 3, pp. 223-239, 1983.

[3] A. Silberschatz, "A Multiversion Concurrency Control Scheme with No Rollbacks", Proc. of the ACM Symposium on Principles of Distributed Computing, pp. 216-223, 1982.

[4] D. Reed, Naming and Synchronization in a Decentralized Computer System. PhD thesis, Department of Electrical Engineering, MIT, Cambridge, 1978.

[5] D. Reed, "Implementing Atomic Actions on Decentralized Data", Transaction on Computer System, Volumn 1, Number 1, pp. 3-23, 1983.

[6] M. Y. Lai and W.K. Wilkinson, "Distributed Transaction Management in JASMIN", Proc. of the International Conf. on Very Large Databases, pp. 466-472, 1984.

[7] A. Chan, S.Fox, W-T.K. Lin, A. Nori, and D.R. Ries, "The Implementation of an Integrated Concurrency Control and Recovery Scheme", In ACM SIGMOD Conf. on the Management of Data 82, pp. 184-191, 1982.

[8] P. Bober and M. Carey, "On Mixing Queries and Transactions via Multiversion Locking", In Proc.IEEE CS Intl.Conf. on Data Engineering 8, 1992

[9] Rajeev Rastogi, S. Seshadri, Philip Bohannon, Dennis W. Leinbaugh, Abraham Silberschatz, S. Sudarshan, "Logical and Physical Versioning in Main Memory Databases", Proc. of 23rd International Conf. on Very Lage Databases, pp. 86-95, 1984.

[10] C. Mohan, Hamid Pirahesh, Raymond A. Lorie, "Efficient and Flexible Methods for Transient Versioning of Records to Avoid Locking by Read-Only Transactions", Pro. of the ACM SIGMOD International Conference on Management of Data, pp. 124-133, 1992.