한국산학기술학회논문지
Vol. 10, No. 11, pp. 3207-3212, 2009

반도체 공정의 생산성 향상을 위한 실시간 대용량 데이터의 효율적인 저장 기법

정원일[1*], 김환구[1]
[1]호서대학교 정보보호학과

An Efficient Storing Scheme of Real-time Large Data to improve Semiconductor Process Productivities

Weonil Chung[1*] and HwanKoo Kim[1]
[1]Dept. of Information Security, Hoseo University

요 약 반도체 산업이 발전함에 따라 생산 효율을 높이기 위해 무인 자동 생산 공정이 요구되고 있다. 이러한 무인 자동화 생산 관리 시스템은 생산성 향상을 위해 생산 공정에서 발생하는 대량의 실시간 데이터 분석 및 관리를 필요로 한다. 따라서 실시간으로 발생하는 대용량 데이터를 저장하기 위한 저장 관리 시스템이 요구된다. 기존의 저장 관리 시스템으로 오라클, MY-SQL, MS-SQL 등의 디스크 기반 DBMS가 있다. 하지만 기존의 디스크 기반 DBMS는 반도체 장비로부터 실시간으로 발생하는 대용량 데이터 처리에 한계가 있다. 본 논문에서는 대용량 데이터를 저비용으로 실시간 저장하기 위해 블록 단위 삽입 트랜잭션을 이용한 압축-합병 저장 기법을 제안한다. 제안 기법은 블록 단위 트랜잭션을 이용하여 실시간 데이터를 빠르게 저장하며 데이터를 압축하고 압축된 데이터를 합병하여 저장하기 때문에 보다 적은 디스크 공간을 사용하여 저장할 수 있다. 따라서 반도체 공정에서 빠르게 발생하는 대용량 데이터를 기존 DBMS보다 빠르게 저장이 가능하고 저장 공간 비용을 감소시킨다.

Abstract Automatic semiconductor manufacturing systems are demanded to improve the efficiency of the semiconductor production process. These systems include the functionalities such as the analysis and management schemes for very large real-time data in order to enhance the productivities. So, it requires the efficient storage management system to store very large real-time data. Traditional database management systems(e.g. Oracle, MY-SQL, MS-SQL) are based on disk. However, previous DBMS's have the limitation on the low storing performance. In this paper, we propose a compress-merge storing method of very large real-time data using insert transaction of a block unit. The proposed method shows better processing performances compare to conventional DBMS's. Also compress-merge method makes it possible that it can store large real-time data on low storage cost. Therefore, the proposed method can be applied to an efficient storage management system in the semiconductor production process.

Key Words : Semiconductor storage management system, Very large real-time data, Compress-Merge store, Block unit transaction

1. 서론

반도체 산업의 주요 관심사는 고밀도, 저비용, 고속 생산에 있다. 따라서 반도체 생산 공정의 저비용, 고속 생산을 통한 생산성 향상을 위해 무인 자동화 생산 시스템을 요하게 된다. 이러한 무인 자동화 생산 시스템은 생산 장비에서 발생하는 로그 데이터를 저장 관리 또는 분석하여 생산성 향상을 위한 각종 응용에 이용한다[1].이러한 응용은 생산 스케쥴 관리와 에러 가능 여부를 실시간으로 체크하여 자동화 공정의 활용률을 최대한 높여서 생

*교신저자 : 정원일(wnchung@hoseo.edu)
접수일 09년 10월 09일 수정일 (1차 09년 11월 04일, 2차 09년 11월 10일) 게재확정일 09년 11월 12일

산 기간을 단축시키고, 생산 과정에서 발생하는 불량을 최소화한다. 따라서 이와 같은 응용시스템은 장비에서 발생한 데이터를 실시간으로 분석 및 적용하기 위한 저장 관리 시스템을 사용한다.

그러나 초미세 나노 공정에 사용되는 차세대 공정제어 장비는 실시간으로 대용량 데이터를 발생시키기 때문에 기존의 오라클, 사이베이스, MY-SQL, MS-SQL과 같은 디스크 기반 DBMS로는 이러한 대용량 데이터를 실시간으로 처리하는데 한계가 있다[2,3,4]. 즉, 기존의 디스크 기반 DBMS는 튜플 단위의 트랜잭션 처리로 인한 동시성 제어 시 병목 현상과 각 튜플에 대한 로그를 남기고 검색을 위해 인덱스를 생성하는 비용으로 인해 자동화 생산 시스템의 대용량 데이터를 실시간으로 저장하기에는 한계가 있다. 또한, 압축 저장을 수행할 때 압축 후 남는 버퍼 공간을 활용하지 못하는 문제와 디스크 기록 시 내부 단편화 문제가 발생한다[5,6].

따라서 본 논문에서 제안하는 압축-합병 저장 기법은 실시간으로 발생하는 대용량 데이터를 저장하는 데 소요되는 비용을 최소화할 수 있는 기법을 제공한다. 또한, 제안 기법은 압축 저장 수행시 발생하는 버퍼 공간 활용 문제와 내부 단편화 문제를 블록 단위 삽입 트랜잭션을 통해 해결한다. 제안 기법은 기존의 압축 저장 기법과 비교해 최대 10%의 저장 비용이 감소됨을 보이며 기존의 압축 저장 기법과의 압축률에 있어서도 10% 정도 우수함을 보이고 있다. 다만, 제안 기법은 압축을 사용하지 않는 기법과의 비교시 압축 수행에 따른 연산 비용이 추가적으로 소요되지만 전체적인 수행 시간을 고려하면 저장 속도에서도 우수함을 보인다.

본 논문의 구성은 2장에서 본 논문의 관련 연구를 설명하고 3장에서는 대용량 데이터 저장 관리 시스템의 기본 구조와 블록 단위 삽입 트랜잭션 처리 기법의 수 행 과정을 설명하며 4장에서는 압축-합병 저장 기법의 구조를 기술 한다. 또한 5장에서는 본 논문이 제안하는 블록 단위 트랜잭션을 이용한 압축-합병 저장 기법의 성능 평가 및 결과를 보이고 이를 분석하며 마지막으로 6장에 서는 향후 연구 방향과 결론으로 끝을 맺는다.

2. 관련연구

2.1 데이터스트림관리시스템(DSMS)

기존의 DBMS는 영속적인 데이터와 일시적인 질의에 대한 응답을 위해 설계되었다. 하지만 증권 시세 표시기, 네트워크 트래픽 모니터링, 웹 로그 분석, 트랜잭션 로 그 분석 및 센서 네트워크 등과 같이 시간에 따라 변하 고 연속적으로 생성되는 대용량 데이터를 처리하기에는 한계가 있다. 이와 같이 연속적으로 생성되는 대용량 데이터를 데이터 스트림이라 한다. 기존의 영속적 데이터 와 다른 이러한 데이터 스트림을 처리하기 위해서 데이터 스트림 관리 시스템이 연구되었다.

데이터 스트림 관리 시스템은 일시적인 스트리밍 데이터에 대해 연속적인 형태의 질의를 지속적으로 처리해야 한다. 이러한 데이터 스트림 관리 시스템의 대표 적인 연구로는 NiagaraCQ, TelegraphCQ, Aurora 등이 있다[7-10]. 그러나 데이터 스트림 관리 시스템에서는 실시간 스트림 데이터를 모두 디스크에 저장하는 것이 목적은 아니다. 이에 본 연구에서는 반도체 공정이나 FPD 공정 등에서 발생하는 대용량 데이터를 모두 저장하기 위한 저장 기법을 제안한다.

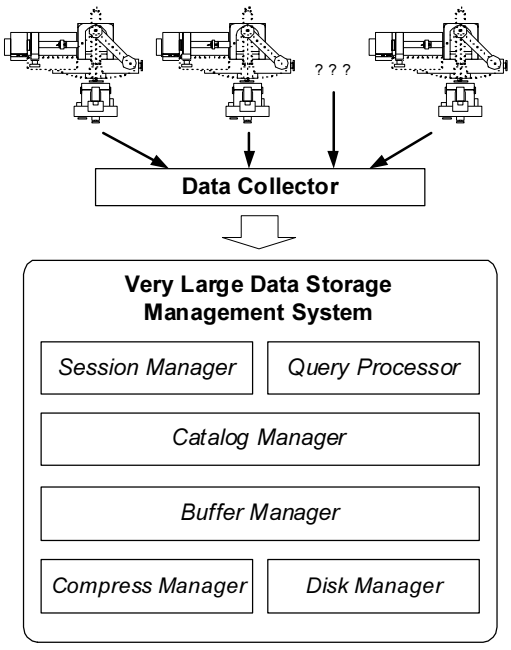
2.2 벌크 연산 (Bulk Operation)

최근 지리 정보 시스템이나 멀티미디어 데이터베이스 시스템 등의 발전으로 인해 대용량 데이터를 빠르게 삽입하는 방법에 관해 연구되고 있다[11,12]. 일반적인 데이터 삽입 기법은 데이터를 하나씩 삽입하면서 인덱스를 구성하기 때문에 대용량 데이터가 발생했을 경우 많은 시 간을 소요하게 된다. 또한 기존의 DBMS는 인덱스로 B+ 트리를 을 소기 때문에 데이터의 삽입마다 노드를 계속적으로 재구성하게 되는 경우가 발생하게 된다. 반복적인 노드 재구성 문제를 해결하기 위해 대량의 데이터를 한 번에 삽입하여 인덱스 구시 간효율적으로 하 는 벌크 연산이 등장했다. 이러한 벌크 연산은 대량의 데이터를 한 번에 삽입 하여 새로운 인덱스를 구성하는 벌크 로딩(Bulk Loading)과 이미 구시된 인덱스 구조에 대량의 데이터를 삽입하는 벌크 삽입(Bulk Insertion)이 있다[13-17]. 그러나 벌크 로딩과 벌크 삽입 역시 메모 리 상에서 인덱스를 재구성하는 인덱스 생성B+ 트리필요하고 생했갔별 트랜잭션을 위한 로그 데이터를 기록해야 하는 + 을 갖는다. 따라서 기존의 벌크 연산 되인덱스 재구성 + 과 로그 데이터 기록 + 으로 인해 실시간으로 발생하는 반도체 생산 장비의 데이터를 실시간으로 저장하기 어렵다. 이에 본 연구에서는 실시간 데이터 스트림에 대해 최적화된 대용량 데이터 삽입 기법을 제안하고 이를 통해 실시간 데이터 저장이 가능함을 보인다.

3. 블록단위 삽입 트랜잭션 처리기법

3.1 대용량 데이터 저장 관리 시스템 구조

데이터 수집기는 장비로부터 대용량 데이터를 수집하며 대용량 데이터 저장 관리자는 데이터 수집기로부터 발생하는 로그 데이터를 받아서 저장한다. 대용량 데이터 저장 관리자는 본 논문이 제안 하는 블록 단위 삽입 트랜잭션을 이용한 압축-합병 저장 기법을 사용하여 대량의 로그 데이터를 받아 디스크에 압축 저장 한다.

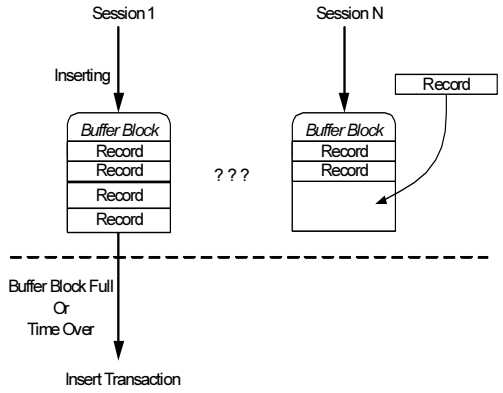


[그림 1] 대용량 저장 관리 시스템의 기본 구조

대용량 데이터 저장 관리자는 버퍼 관리기(Buffer Manager), 압축 관리기 (Compress Manager), 디스크 관리기(Disk Manager)로 구성된다. 기존의 DBMS는 튜플 단위의 삽입 처리를 하기 때문에 매 삽입 트랜잭션마다 로그 저장을 위한 비용이 발생한다. 또한 삽입 시 인덱스 노드를 재구성하는 경우가 발생할 수 있기 때문에 저장 성능이 크게 감소하게 된다. 따라서 본 논문에서는 삽입 트랜잭션의 단위를 튜플이 아닌 버퍼 블록 단위로 처리하여 디스크 I/O를 감소시킨다.

3.2 블록 단위 삽입 트랜잭션 처리 기법의 수행 과정

본 논문이 기반으로 하는 대용량 데이터 저장 관리자의 목적은 기존의 시스템과 달리 대량의 데이터를 실시간으로 저장하는 것이기 때문에 버퍼 블록 단위로 삽입 트랜잭션을 수행한다. 버퍼 블록 단위 삽입 트랜잭션은 대량의 레코드를 버퍼 블록 내에 저장하여 하나의 블록이 다수의 레코드 모임인 레코드 셋의 형태로 저장되어 처리된다.



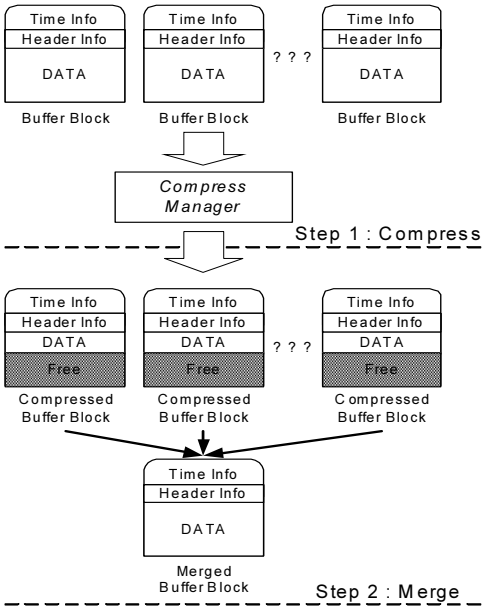
[그림 2] 블록 단위 삽입 트랜잭션의 수행 과정

클라이언트가 레코드 삽입 요청을 하면 대용량 데이터 저장 관리자는 기존의 DBMS와 달리 레코드 삽입을 위한 세션을 생성하며 각 세션은 데이터를 버퍼 블록에 지속적으로 저장하여 레코드를 모아 커다란 레코드 셋 으로 만든다. 그리고 이렇게 만들어진 커다란 레코드 셋 이 하나의 버퍼 블록을 모두 채우거나 삽입 대기 시간 이 일정 시간을 초과하면 현재까지 버퍼 블록에 저장 한 레코드 셋으로 삽입 트랜잭션을 생성하여 기록 한다.

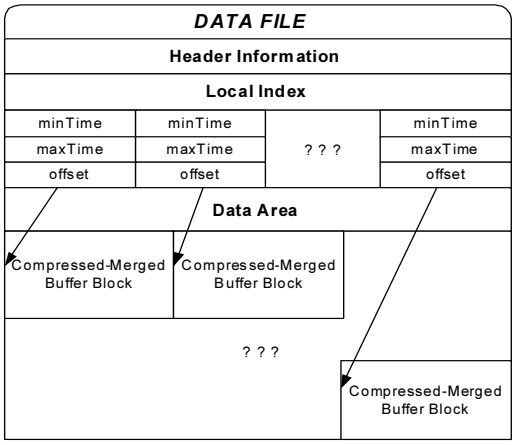
4. 압축-합병 저장 구조

대용량 데이터 저장 관리자는 실시간 대용량 데이터를 저비용으로 저장하기 위해 데이터를 블록 단위로 압축을 한다. 하지만 버퍼 블록을 압축하게 되면 압축 후 남은 버퍼 블록 공간을 활용하지 못하고 디스크 기록 시 내부 단편화 현상이 발생한다. 따라서 압축된 블록을 합병 하는 압축-합병 저장 구조를 사용한다.

그림 3은 압축-합병 과정을 나타낸 그림이다. 먼저 압축 관리자(Compress Manager)는 삽입 요청된 버퍼를 버퍼 관리자로부터 받아서 데이터를 압축한다. 그리고 압축 관리자가 생성한 압축된 버퍼블록들을 합병하여 합병된 버퍼 블록을 생성한다. 이때 합병 대상 버퍼간의 시간 정보를 비교 후 갱신하여 합병된 버퍼 블록에 저장한다. 그 다음 합병된 블록들을 디스크에 연속적으로 기록한다.



[그림 3] 압축-합병 저장 과정



[그림 4] 데이터 파일 저장 구조

그림 4는 데이터 파일 저장 구조이다. 하나의 데이터 파일(Data File)은 헤더 정보(Header Information)와 로컬 인덱스 (Local Index) 그리고 데이터 영역(Data Ar ea)으로 구성된다. 헤더 정보에는 데이터 파일의 운영 정보가 기록되며 로컬 인덱스 영역에는 각 압축-합병된 데이터 블록의 시간정보(minTime, maxTime)와 해당 데이터 블록의 디스크 내 위치(offset)을 저장한다. 그리고 데이터 영역에는 실제 압축-합병된 버퍼 블록이 저장된다. 이와 같은 압축-합병 저장 기법은 디스크 공간 활용을 최대화하여 대용량 데이터의 저장시 디스크 공간 비용을 다소 감소시킨다.

5. 성능분석

5.1 평가 환경

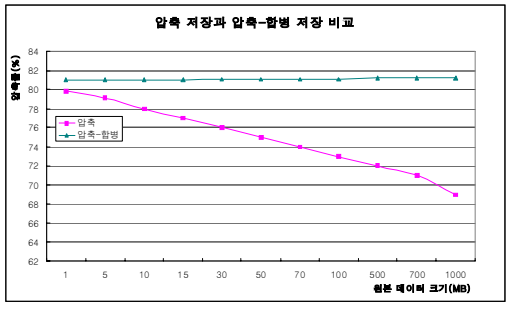
본 논문이 제안하는 압축-합병 저장 기법 평가에 사용된 시스템 환경은 CPU 칩셋은 Intel® Pentium® 4 2.6Ghz이며 메모리는 2GB이다. 평가에 사용된 데이터는 반도체 생산 공정 데이터처리 솔루션을 개발하는 ㈜비스텔에서의 샘플데이터로 초당 10만 건의 데이터를 생성 하는 생성기에 의해 생성하였다. 데이터는 필수적으로 데이터의 생성시간을 포함하며, 장비의 정보와 가 변 길이의 로그 데이터를 포함한다.

5.2 평가 방법

평가 방법은 본 논문의 제안 기법을 상용 디스크 기반 DBMS인 Oracle, MS-SQL과 비교하여 저장 속도와 저장 비용의 우수성을 증명한다. 저장 비용을 측정 할 시 본 논문이 제시하는 압축-합병 저장 기법은 저장되는 데이터를 압축하고 로컬 인덱스를 생성하기 때문에 비교 대상인상용 DBMS의 데이터 삽입 시에도 압축 저 장과 인덱스를 사용한다. 또한 본 제안 기법이 블록 단위 삽입 트랜잭션을 사용하기 때문에 상용 DBMS에서 도 유사한 저장 방식을 사용하기 위해 벌크 삽입 기법 을 이용한다.

5.3 압축 저장과 압축–합병 저장 시 비용 평가

이 절에서는 압축 저장 기법과 압축 후 합병을 사용하여 저장하는 본 제안기법을 측정하여 비교한다. 측정 방법은 동일한 압축 알고리즘을 사용하여 저장된 데이터 크기를 비교한다. 저장하는 데이터의 크기는 1MB ~ 1000MB로 변화시켜 저장 비용을 측정 하였다.



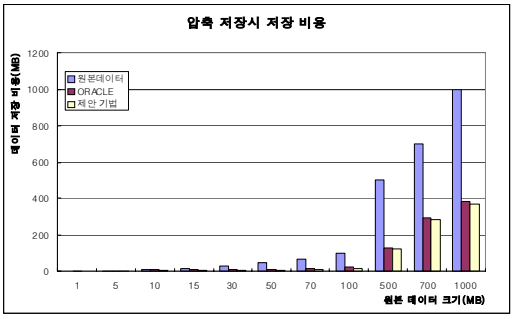
[그림 5] 압축 저장 비용과 압축-합병 저장시 저장 비용 비교

그림 5는 그래프를 통해 압축-합병 기법을 사용한 경우가 합병을 하지 않은 압축 저장 기법과 비교했을 때 최대 10% 정도 저장 비용이 감소 한 것을 보여준다. 실험 결과를 통해 압축-합병 저장 기법은 압축 후 남는 공간을

합병을 통해 사용하기 때문에 압축만 했을 때 보다 저장 비용이 감소된 것을 알 수 있다.

5.4 저장 비용 평가

압축-합병 저장 기법은 기존의 압축 기법으로 압축된 버퍼 간의 합병을 가한 후 연속적으로 기록하기 때문 에 저장 공간 비용을 감소시킬 수 있다. 타 상용 DBMS와의 성능 비교를 위해 동일한 데이터를 압축 저장 하여 저장 전과 후의 크기 변화를 비교 하였다. 저장하는 데이터의 크기는 1MB ~ 1000MB로 변화시켜서 비용 을 측정하였다.

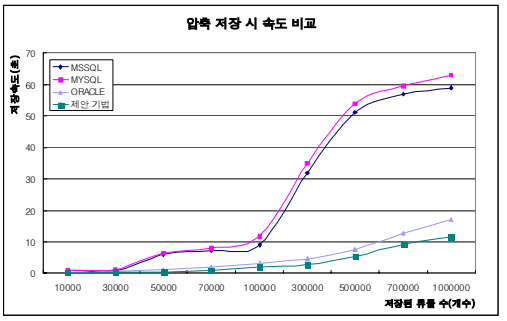


[그림 6] 압축-합병 저장 기법과 상용 DBMS의 압축 저장 기법의 저장 비용 비교

그림 6의 그래프는 위에서 제시한 조건에서의 압 축-합병 기법과 상용DBMS의 압축 저장 기법의 비용을 측정한 결과이다. 오라클의 경우 원본 데이터에 비해 약 75~80% 정도의 압축률을 보였고, 압축-합병 저장 기 법은 약 85~90% 정도의 압축률로 다소 우수하다는 것 을 알 수 있다.

5.5 압축 저장 속도 평가

압축 저장 시 비용의 감소는 앞의 5.3, 5.4 절에서 확인 하였다. 하지만 압축 시 발생하는 연산의 부하는 저장 시 속도에 큰 영향을 미친다. 따라서 압축-합병을 사용하였을 때 저장 속도에 미치는 영향을 측정하기 위해 압축을 사용하지 않은 타 DBMS(MS-SQL, MY-SQL)와 압축을 사용한 Oracle 그리고 본 제안 기법의 속도를 비교하였다. 측정 방식은 MS-SQL과 MY-SQL 그리고 Oracle에 레코드의 시간 속성을 인덱스로 생성하는 벌크 삽입 방식으로 튜플을 10,000개에서 1,000,000개까지 삽입하여 비교 하였으며 MS-SQL과 MY-SQL은 압축을 사용하지 않았고 Oracle은 압축 옵션을 설정하여 측정하였다.



[그림 7] 압축-합병 저장 기법과 상용 DBMS의 압축 저장 시 속도 비교

그림 7의 그래프는 압축을 사용하지 않은 타 DBMS(MY-SQL, MS-SQL)과 압축을 사용한 Oracle 그리고 본 제안 기법의 압축 저장이 저장 속도에 미치는 영향을 측정하기 위한 그래프이다. Oracle과 본 제안 기법은 압축 저장을 하지 않았을 때 보다 약 35~40% 정도의 저장 속도 감소가 있었지만 압축을 하지 않은 MY-SQL, MS-SQL보다 저장 속도에서 우수함을 보였다.

6. 결론 및 향후 연구

본 논문은 대용량 데이터를 실시간으로 저장하기 위하여 블록 단위 삽입 트랜잭션을 이용한 압축-합병 저장 기법을 제안하였다. 블록 단위 삽입 트랜잭션 처리 기법은 기존의 삽입 트랜잭션의 튜플 단위 수행으로 인한 일부 작업들을 생략 하여 삽입 트랜잭션의 속도를 향상 시켰다. 또한 이를 이용한 압축-합병 저장 기법은 데이터를 압축 하여 저장함으로써 저장 공간을 최소화 하였다. 향후 연구로는 블록 단위 트랜잭션을 효율적으로 사용하기 위한 효율적인 버퍼 교체 정책과 기존의 로깅 기법을 블록 단위로 확장한 블록 단위 로깅 기법, 그리고 본 논문에서 제시한 압축 기법을 보완하여 압축 속도를 향상 시키는 압축 기법의 연구가 필요하다.

참고문헌

[1] 김원태, “반도체장비기술교육센터(SETEC) 기술교육 시리즈(1-1) 반도체 장비 간 통신 표준”, IT Innovation.

[2] 알티베이스, “반도체 생산 관리 솔루션 EES, 하이브리드 DBMS 성공사례”, http://www.altibase.com

[3] 박상근, 박순영, 정원일, 김명근, 배해영, "GMS: 공간

데이터베이스 관리 시스템", 개방형지리정보시스템 학술대회, pp. 217-224, 2003.
[4] 김영기, 백성하, 이동욱, 정원일, 배해영, "대용량 데이터의 실시간 저장을 위한 블록 단위 삽입 트랜잭션 처리 기법과 압축-합병 저장 기법", 한국컴퓨터종합학술대회 2008, Vol 35, No, 01(A), pp. 35-38, 2008.
[5] K.Y. Whang, R. Krishnamurthy, "Query Optimization in a Memory-Resident Domain Relational Calculus Database System", ACM TODS, pp. 67-95, March 1990.
[6] P. Boncz, S. Manegold and M. Kersten, "Database architecture optimized for the new bottleneck: memory access", VLDB, pp. 54-65, 1999.
[7] R. Motwani, and et. al., "Query Processing, Resource Management, and Approximation in a Data Stream Management System", CIDR, 2003.
[8] Babcock, B., Babu, S., Datar, M., Motwani, R. and Widom, J., "Models and Issues in Data Stream Systems", PODS, 2002.
[9] Abadi, D. J and et. al, "Aurora: A New Model and Architecture for Data Stream Management", VLDB Journal, 2003.
[10] Chandrasekharan S., and J. Franklin, M., "Streaming queries over streaming data", VLDB, pp. 203-214, 2002
[11] Mohan, C., Haderle, DON., "ARIES: A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging", ACM TODS, Vol.17 No. 1, pp. 94-162, 1992.
[12] Lory. D., Krithi, R., "Recovery Protocals for Shared Memory Database Systems", SIGMOD, 1995.
[13] Jochen van den B., Bernhard, S., "An Evaluation of Generic Bulk Loading Techniques", VLDB, 2001.
[14] Arge, L., Hinrichs, K., et al. "Efficient Bulk Operations on Dynamic R-trees", ALENEX, pp. 328-348, 1999.
[15] 복경수, 송석일, 유재수, "고차원 색인 구조를 위한 벌크 삽입", 컴퓨터정보통신연구소, 제9권 제1호, 2001.
[16] 복경수, 이석희, 유재수, 조기형, "고차원 색인 구조를 위한 효율적인 벌크 로딩 알고리즘", 컴퓨터정보통신연구소, 제8권 제1호, 2000.
[17] 백성하, 이동욱, 어상훈, 정원일, 김경배, 오영환, 배해영, "블록 단위 트랜잭션을 이용한 대용량 데이터의 실시간 저장관리기", 한국공간정보시스템학회 논문지, 제 10권 제 2호, 2008.

정 원 일(Weonil Chung) **[정회원]**

- 1998년 2월 : 인하대학교 전자계산공학과(공학사)
- 2004년 8월 : 인하대학교 컴퓨터정보공학과(공학박사)
- 2004년 7월 ~ 2006년 7월 : 한국전자통신연구원 선임연구원
- 2007년 3월 ~ 현재 : 호서대학교 정보보호학과 교수

<관심분야>
데이터스트림, 이동객체, 시스템보안

김 환 구(HwanKoo Kim) **[정회원]**

- 1991년 2월 : 경북대학교 수학과(공학사)
- 1998년 5월 : U. of Tennessee -Knoxville 수학과(이학박사)
- 2002년 3월 ~ 현재 : 호서대학교 정보보호학과 교수
- 2007년 3월 ~ 현재 : 한국정보보호학회 이사

<관심분야>
평가 및 인증, 암호학