주기억 장치 데이타 베이스 시스템

문 송 천*·강 석 훈**
(*한국과학기술원 전산학과 조교수, **석사과정)

1. 서 론

최근 하드웨어의 발전과 더불어 정보의 다양화, 그리고 점차 사용자들의 질의에 대한 보다 빠른 응답 요구에 의해 데이타베이스 컴퓨터라는 새로운 분야가 연구되기 시작했다. 여기서 데이타베이스 컴퓨터란 데이타베이스의 성능을 향상시키기 위해 특수한 하드웨어 혹은 소프트웨어를 추가하거나, 데이타베이스만을 위해 사용되는 일반적인 후위 시스템(back-end system)등을 포함한 포괄적 의미이다[17]. 이러한 데이타베이스 컴퓨터에 관한 연구는 점차 하드웨어 자체를 새로이 설계하여 데이타베이스 전용 컴퓨터로 구성하고자 하는 연구와 대용량 주기억 장치를 갖는 시스템으로 구성하고자 하는 연구로 나뉘어지게 되었다.

하드웨어 시스템을 재 구성하는 연구로는 DIRECT[5], GAMMA[7], GRACE[11], ICOT, Mix[2], SABRE[31] 등에서 처럼 특수한 하드웨어를 추가하거나 시스템 전체를 재구성하여 입/출력 채널을 통과하는 데이타의 양을 줄이고자 노력했다.

한편, 지속적인 하드웨어 기술의 발전과 이로 인한 기억장치 칩의 가격 하락 현상은 멀지 않아 수 기가(giga) 바이트의 주기억 장치를 가지는 컴퓨터 시스템의 등장을 예고하게 되었다.[4][13][21] 이렇게 될 경우, 대부분의 자료가 주기억 장치내에 저장되어 디스크에 접근할 필요가 거의 없으므로 디스크 입/출력에 의한 병목 현상을 포함하여 기존의 데이타베이스 컴퓨터가 해결하지 못한 여러가지 문제들을 해결할 수 있게 되며, 이에 따른 상당한 연구가 대용량의 주기억 장치를 사용하는 데이타베이스 시스템에 집중되고 있다.[6][23][24][25][29]

대용량의 주기억 장치를 사용하는 방법에는 크게 두 가지로 구분될 수 있는데, 하나는 기존의 디스크 접근방법을 향상시키기 위하여 대용량의 주기억 장치를 버퍼로 사용하는 방법으로 Shapiro[29]는 대용량 버퍼 환경하에서 관계 연산자의 수행에 관한 연구를 발표 하였고 Gray[15]는 대용량 버퍼 환경에서 수행을 위한 최적의 버퍼 크기를 "Five Minute Rule"이라는 이론을 통해 제안 하였다. 그러나 이 방법에서 데이타베이스의 성능은 크게 향상시켰지만 아직도 디스크와 주기억 장치 사이의 입/출력 문제가 성능에 많은 영향을 주고 있다. 다른 하나는 전체 데이타베이스를 주기억 장치내에 상주 시킬 수 있다고 생각하는 주기억 장치 데이타베이스 시스템[12][23][24][25]으로 처음 데이타베이스가 주기억장치로 적재될 때와 회복을 위해 디스크 입/출력하는것 외에는 기존의 디스크 데이타베이스에서 가장 문제가 되었던 디스크 입/출력에서 발생되는 병목 현상을 줄일 수 있었다.

이와 같은 주기억 장치 데이타베이스 컴퓨터 시스템이 구성될 수 있는 근거는 다음과 같이 요약될 수 있다. 첫째, 주기억 장치 소자인 RAM의 가격이 매년 40% 정도씩 떨어지고 또한 RAM의 집적도가 매년 1.3배 이상 증가되고 있으므로 앞으로 주기억 장치의 크기는 매우 커질 것이다. 물론 어떤 특정 응용 분야에서는 주기억 장치의 크기가 커지는데 비례하여 정보도 많아 질 수 있다. 그러나 대부분 응용 분야에서의 데이타베이스의

크기는 사용자 수와 응용 분야에 의해 제한 되어 있기 때문에 기억 장치의 크기 만큼 빠른 속도로 증가하지 않는다[12][21]. 둘째, 디스크 데이타베이스에서 성능에 가장 큰 영향을 미치는 것은 디스크와 주기억 장치 사이의 입/출력 병목 현상이다. 그러므로 현재의 시스템에서 성능을 향상시키기 위해서는 보다 빠른 프로세서를 이용하는것 보다는 오히려 기억장치의 크기를 늘려서 입/출력 시간을 줄이는 것이 타당하다.[12]

2. 주기억 장치 데이타베이스 시스템에서의 주요 연구 분야

기존의 디스크 데이타베이스에서 설계된 많은 알고리즘들과 자료 구조들이 주기억 장치 데이타베이스 시스템에서는 적합하지 못하므로 시스템의 효율적인 운용을 위해서 다음 사항들이 중점적으로 연구되어 제안되고 있다.

2. 1 데이타 저장 구조 및 인덱스 구조에 관한 연구

전체 데이타가 대량으로 주기억 장치에 적재될 때, 가장 적합한 데이타 저장 구조에 관한 연구이다. 이 연구를 위해서는 시스템의 사용시 예상되는 응용 프로그램을 분석해서 응용 프로그램의 특성에 맞는 저장 구조로 개선하여야 한다. 특히 Lehman[23][24]이 설계한 T-트리 구조는 주기억 장치 데이타베이스 시스템의 환경에 적합한 인덱스 구조로 평가된다. 해쉬 인덱스 구조도 데이타가 주기억 장치 내에 상주하는 환경에서는 디스크에 상주하는 경우와 같은 빈번한 디스크 입/출력을 유발하지 않으므로 그 효용성이 높아지게 된다.

주기억 장치 데이타베이스 시스템에서는 데이타를 표현할 때, 포인터의 사용이 용이해짐에 따라, 기억장치를 보다 효율적으로 사용할 수 있게 된다. 그러나 포인터를 사용하는 방법은 실제 데이타를 얻기 위해서 한번의 부가적인 기억 장치 탐색을 요구하게 되므로 빈번히 사용되는 데이타의 표현에는 부적당한 방법이 될 수 있다. 그렇지만 주기억 장치의 특성상 효율적인 주기억 장치 데이타베이스 시스템의 자료 구조가 될 수도 있다.

해싱 방법이 인덱스로서 사용이 될 때는 정해진 수의 비교에 의해 목적 데이타를 찾게 되며, 관계 연산자, 특히 등가 결합 연산, 수행시 효율적으로 처리 할 수 있도록 허용되므로, 주기억 장치 데이타베이스 시스템 환경에서 질의어를 처리할 때 시스템 성능 향상의 큰 요인이 된다.

주기억 장치 데이타베이스 시스템에서 효율적으로 사용할 수 있는 해싱 방법은 chained 버캣 해싱[20], extendible 해싱[10], 선형 해싱[27], modified linear 해싱 등이 있다. Chained 버캣 해싱 방법은 주기억 장치와 디스크에서 사용되는 정적인 구조를 갖고 있으며, 데이타를 재구성하지 않는 잇점이 있다. 그러나 정적인 구조로 인해 동적인 환경에서는 성능이 좋지 않다. 화일 규모가 적을 경우에는 성능이 저하되며, 규모가 클 경우에는 기억장치의 낭비가 있다. Extendible 해싱 방법은 동적인 해싱 테이블을 사용하며, 해싱 테이블 크기를 미리 정할 필요가 없다. 디렉토리는 2의 배수로 증가하며, 해싱 함수가 random하지 않을 경우 디렉토리 크기가 급격히 증가하게 된다. 선형 해싱 방법도 동적인 해싱 테이블을 사용하나, 테이블이 선형으로 증가하며, 미리 정의된 순서에 의해 디렉토리가 증가하게 된다. 잇점으로는 버캣이 순차적으로 정렬될 수 있으며, 디렉토리가 필요없다. 또한 저장 공간 사용도에 의해 노드의 분할을 할 수 있다. Modified linear 해싱은 extendible 해싱 방법과 비슷한 디렉토리를 사용한다. 디렉토리는 단일 노드에 의해 연결되며, 선형으로 증가 한다. 노드의 연결된 길이에 의해 분할이 수행된다. 일반적인 해싱 방법이외에 성능 향상을 위해서 요구되는 해싱 방법의 개선 방향은 다음과 같다. 첫째로, 특정 버캣의 집중적인 오버플로우(overflow)를 예방하거나 이를 효과적으로 처리할 수 있는 해싱 함수에 관한 연구가 필요하다. 두번째로, 비등가 결합 연산을 효율적으로 처리하기 위해서는 순서 보존(order preserving) 해싱 함수에 대한 연구가 필요하다.

T-트리는 AVL-트리와 B-트리의 특성을 종합하여 주기억 장치 데이타베이스 시스템에 적합하도록 설계한 트리 구조이다. AVL-트리는 새로운 노드를 삽입할 때 앞 노드에만 영향을 미치며, 이진 탐색 트리의 특성 때문에 탐색 시간이 무척 빠르다. 그러나, 각 노드의 구조가 한 개의 데이타 항목, 두 개의 포인트, 그리고 제어 부분으로 구성되어 있으므로 기억 장치 이용율이 저조하다. 반면에 B-트리는 한 노드가 다수의 데이타 항목들을 가지므로 기억 장치 이용율이 좋지만, 탐색 시간면에서 AVL-트리보다 느리고 갱신 작업 속도는 빠르다. 이상과 같은 두 종류의 트리의 잇점을 사용한 T-트

리는 일반적으로 기억 장치 이용율이 높고 탐색 시간이 빠르다.[23) 24)]

2. 2 동시성 제어 기법에 관한 연구

디스크 데이타베이스 시스템에서 동시성 제어는 지금까지 많이 연구되어온 분야중의 하나이다. 이들 동시성 제어 방법은 크게 회의적 방법인 록킹(locking)[9) 14)]과 타임 스템프(time stamps)[3)], 그리고 낙관적 방법 (optimistic method)[22)30)]으로 분류 될수 있다. 이중에서 록킹 방법은 동시성 제어에서 구현이 용이하고 단순하기 때문에 가장 많이 사용되는 방법 중의 하나이다[30)].

주기억 장치 데아타베이스 시스템에서의 동시성 제어 방법들에 대한 상대적인 비용은 거의 비슷할 것으로 평가된 경우[13)]도 있는데, 이는 대부분의 데이타 베이스 시스템들이 최근에는 동시성 제어를 이미 주기억 장치내에 유지하고 있기 때문이다.

주기억 장치 데이타베이스 시스템에서는 입/출력 병목 현상을 제거했기 때문에 처리시간(processing time)과 throughput이 향상되므로 주기억 장치 데이타베이스 트랜잭션들을 직렬로(serial) 실행시켜도 향상된 성능이 동시성 제어의 필요성을 없앨 수 있다고 제안한 논문[12) 28)]도 있다.

Lehman[25)]은 록킹을 기초로 한 동시성 제어 방법을 새롭게 도입하였는데, 시스템의 동시성 요구에 따른 두개의 서로 다른 록킹 단위(locking granule size)를 사용하였다. 즉, 높은 레벨의 데이타를 공유하는 것이 필요로 할 때에만 비싸면서 세분화된 록킹 단위를 사용하고 그외의 경우에는 덜비싸면서 성긴 록킹 단위를 사용하여 전반적인 록킹 비용을 감소시키도록 하였다.

2. 3 회복 기법에 관한 연구

주기억 장치 데이타베이스 시스템에서는 데이타의 변경이 주기억 장치 내에서만 이루어지므로 고장(failure)이 발생할 경우 초기에 적재된 이후에 수행된 모든 트랜잭션들의 결과를 상실하게 된다. 따라서, 주기억 장치 데이타베이스 시스템의 회복은 종전의 디스크 데이타베이스 시스템에서 보다 더 큰 비중을 차지한다고 할 수 있으며, 최근에 가장 활발하게 진행된 연구 분야이다.

주기억 장치 데이타베이스 시스템 환경에서의 주된 문제점은 주기억 장치의 비영속성 때문이다. 이를 해결하기 위한 회복 기법에서 뱃터리 백업 전원이 지원 되는 적은 크기의 영속성을 갖는 안정된 주기억 장치가 제안되었다[6)12)]. 또 다른 문제점은 데이타베이스 처리를 위해 주기억 장치로 데이타베이스를 최초에 적재(load)하는데 필요한 데이타를 얻기위해 바로 전의 데이타베이스의 완전한 이미지인 archive DB에 있는 데이타와 로그(log)의 데이타를 병합하는 것이 필요하다. 이러한 시간을 줄이기 위해, archive 들은 매우 자주 생성되어야 하고, 여러개의 보조 기억 장치들에 분산될 수 있어야 한다.

주기억 장치 데이타베이스 회복 기법들을 분류하기 위한 회복의 선택항목들에는 영속성을 갖는 안정된 주기억 장치의 양, 특별한 로깅 하드웨어의 사용유무, 검사점 수행(checkpointing)이 트랙잭션 성능에 영향을 주는지의 여부, 완료처리(commit processing)의 형태들이 있다. 이러한 선택 항목들은 로깅(logging)을 하기 위한 입/출력 또는 CPU부담, 검사점 수행, 그리고 완료처리에 직접적으로 영향을 준다. 아마도 주기억장치 데이타 베이스 회복기법들을 분류하기 위한 회복의 선택항목들에는 영속성을 갖는 안정된 주기억 장치의 크기일 것이다. 만일 주기억 장치 전체가 영속성을 갖는다면 회복을 위한 시스템 고장(system failure) 처리는 고려할 필요가 없을 것이다. 그러나, 로그(log)와 archive DB 들은 매체 고장(media failure)이 발생할 경우에 백업(backup)을 제공하기 위하여 여전히 필요하다. 변경된 데이타의 사전 이미지 값(before image value)들은 트랜잭션 철회(transaction UNDO)를 용이하게 하기 위해 주기억 장치에 유지할 필요가 있다. 만일 영속성을 갖는 주기억 장치가 존재하지 않는다면 트랜잭션 완료처리는 모든 로그 입/출력이 성공적으로 수행 완료되기 전까지는 시작될 수 없다. 따라서 이러한 입/출력 부담은 응답 시간에 직접적으로 영향을 미친다. 만일 로크 버퍼를 위한 만큼의 영속성을 갖는 주기억 장치가 존재할 경우에는 일단 로그 레코드들이 버퍼로 쓰여지는 완료 처리가 진행될 수 있다. Eich의 연구[8)]와 IBM의 IMS/VS Fast Path에서의 회복 기법들은 영속적인 주기억 장치가 없다고 가정하였으며, Lehman[25)]과 Garcia- Molina[13)]들은 영속적인 주기억 장치에 있는 로그 버퍼를 가정하였다.

독립적인 검사점 프로세서(checkpoint processor)의

사용은 검사점 수행을 하는데 드는 CPU 부담을 제거하여 트랜잭션 성능에 중요한 영향을 주었다. Eich[8]의 기법이 이 범주에 속하며, 주기억 장치 전체가 영속적인 기법들도 이 범주에 속한다.

기존의 디스크 데이타베이스 시스템과 주기억 장치 데이타베이스 시스템의 회복에 필요로 하는 동작 내용들을 고장 형태에 따라 분류하여 비교하면 아래와 같다. 트랜잭션 고장(transaction failure)이 발생할 경우에는 두 시스템 모두 트랜잭션 철회(transaction UNDO)만을 수행하면 된다. 시스템 고장(system failure)이 발생할 경우에는 기존의 디스케이스 시스템에서는 global UNDO와 partial REDO를 수행하여야 하며, 주기억 장치 데이타베이스 시스템에서는 global REDO만을 수행하면 된다. 매체 고장(media failure)이 발생할 경우에는 기존의 데이타베이스 시스템에서는 global REDO만을 수행하면 되지만 주기억 장치 데이타베이스 시스템에서는 global REDO와 partial REDO를 수행하여야 한다.

3. 주기억 장치 데이타베이스 시스템의 사례연구

3. 1 IMS FASTPATH[18) 19)]

IBM의 IMS/VS Fast Path는 주기억 장치 데이타베이스를 사용하였고, 완료 처리를 한데 묶어 수행(commit groups)하는 시도를 최초로 수용한 시스템이었다. 갱신들은 특별한 데이타베이스 버퍼에서 수행되며, 주기억 장치 데이타베이스 페이지들은 완료 시점(commit time)까지는 변경되지 않는다. 시스템 전반에 걸친 Transaction- Consistent Checkpoint(TCC)들이 비동기적인 IMS task에 의해 주기억 장치 데이타베이스 처리 과정과 병렬로 수행된다. 이 시스템의 회복 기법에 있어서의 주요한 단점은 로그 입/출력이 완료 시점에만 수행되며, 검사점 수행을 위해 주기억 장치 데이타베이스 전체가 반드시 읽혀져야 하는 점이다.

3. 2 MMM(Massive Memory Machine) Project[12) 13) 28)]

Princeton 대학의 MMM 프로젝트는 주기억 장치를 매우 많은 양으로 지원하도록 특별하게 설계한 구조를 제시하였다. 이 프로젝트에서의 회복 기법은 HALO(-HArdware LOgging device)에 기초하고 있다. HALO는 모든 주기억 장치 데이타베이스 동작들을 가로채어 최초에 영속성을 갖는 주기억 장치로 기록된 사전 이미지(BFIM)와 사후 이미지(AFIM)로그 데이타를 생성시키며, 연속적인 Action-Consistent Checkpoint(ACC)가 수행된다. 영속성을 갖는 안정된 주기억 장치를 사용함으로써 완료 처리(commit processing)는 로그 버퍼들이 청소될 때까지를 기다릴 필요가 없어졌다. 이 시스템의 회복 기법에 있어서의 장점은 로그에 대한 비동기적인 갱신과 archive DB에 대한 갱신이 연속적이면서 병렬로 이루어질 수 있는 점이다.

3. 3 De Witt et al의 연구[6)]

UC/Berkeley에서 DeWitt를 포함한 연구원들은 주기억 장치 데이타베이스에 대한 접근방법(access method) 등 구현 기법에 관심을 두었다. 그들의 회복 기법은 사전 이미지와 사후 이미지를 갖는 로그와 빈번한 검사점을 사용한다. 로깅과 데이타베이스 처리를 비동기적으로 수행할 수 있도록 하는 pro-commint된 트랜잭션 개념이 사용된다. 즉, 어느 트랜잭션이 완료(commit)되었을 때 완료 레코드(commit record)는 로그 버퍼에 위치하게 되며, 다른 충돌중인 트랜잭션들은 로그 버퍼들이 디스크로 청소되지 않더라도 처리가 진행되도록 허용된다. ACC 검사점 방법은 주기억 장치 데이타베이스 전체를 읽어들이고, 변경된 페이지들을 인식함으로써 트랜잭션 처리와 병렬로 수행된다. 이들 연구의 회복 기법에 있어서의 주요한 단점은 일관성 상태(consistent state) 를 저장하기 위하여 데이타베이스가 중단되어야 하는 점이다.

3. 4 IBM의 OBE Project[1)]

자료 구조 표현 방법과 회복 기법을 포함한 주기억 장치 데이타베이스에 대한 설계가 IBM에서 OBE(Office By Example) 데이타베이스를 사용하기 위해 제안되었다. 회복 기법에서의 모든 회복 부담은 TOC(Transaction- Oriented Checkpoint)를 하기 위한 완료 동작(commit operation)으로 제한된다. 별도의 로그는 제안되지

않았으며, archive 데이타 베이스에 있는 쉐도우 페이지(shadow page)들이 사용된다. 완료 시점에서 변경된 모든 릴레이션(relation)들이 archive 데이타베이스에 있는 쉐도우 영역으로 기록된다.

3. 5 Hagmann의 연구[16)]

Hagmann은 "fuzzy dumps"기법과 로그 압축(log compression) 을 이용한 주기억 장치 데이타베이스 시스템의 회복 기법을 제안하였다. Fuzzy dumps 기법은 데이타베이스 시스템의 재시작(restart)을 신속하게 하기 위하여 주기억 장치의 이미지를 주기적으로 디스크에 기록하는 것으로 동기를 맞추지 않고 수행되기 때문에 트랜잭션의 부분적인 갱신을 포함하여 데이타베이스의 일관성 있는 상태를 유지하지 못하는 단점을 갖는다. Hagmann이 제안한 회복 기법의 설계 원칙은 디스크에서는 직렬 전송이 보다 좋은 동작성능을 갖는 다는 관측과 데이타베이스에 대한 로그의 크기를 작게 해야 한다는 것에 기초한다.

3. 6 Lehman의 연구[23) 24) 25)]

Wisconsin 대학의 Lehman이 제안한 MM-DBMS(-Main Memory Database Management System)에서는 정상적인 데이타베이스 프로세서(DP)와 병렬로 수행되는 별도의 회복 처리 프로세서(RP)가 사용되었다. 이때 DP는 주기억 장치 전부를 접근할 수 있으며, RP는 영속성 있는 주기억 장치에만 접근할 수가 있어 로깅과 검사점 수행에 서로 협동한다. Lehman의 시스템이 다른 시스템들과 구별되는 독특한 검사점 수행 방법은 RP가 완료된 리스트로부터 사후 이미지 로그 레코드들을 영속성 있는 주기억 장치에 있는 특별한 "로그 테일(log tail)"에 분할(partition)이라고 정의한 일종의 페이지 단위로 복사하는 점이다. 또한, Lehman은 주기억 장치 상주 데이타베이스에서의 화일, 인덱스 구조 및 관계 연산자의 수행 알고리즘에 대한 성능 평가도 수행하였다.

4. 결 론

주기억 장치 데이타베이스 시스템은 방대한 양의 주기억 장치에 데이타베이스 전체를 상주시켜 사용하는 방식이다. 디스크의 입/출력이 초기에 데이타베이스를 적재할 때와 회복(recovery)을 목적으로 디스크에 기록할 때를 제외하고는 발생하지 않는다.

질의어를 처리하는 과정에서 디스크에 접근하지 않으므로 데이타베이스 연산을 빠르게 수행할 수 있지만 주기억 장치 데이타베이스 시스템은 질의어 처리(query processing), 동시성 제어(concurrency control), 그리고 회복 기법에 관한 데이타베이스 관리 시스템의 자료 구조와 알고리즘의 변경을 요구한다.

주기억 장치 데이타베이스 시스템은 Amman[1)]과 Lelard[26)]에 의해 제시되기 시작하였고 Lehman[23)24)]이 주기억 장치 상주 데이타베이스에서의 화일, 인덱스 구조 및 관계연산자의 수행 알고리즘에 대한 성능 평가를 다루고 있다. 질의어 처리를 위한 관계 연산자의 수행 결과 릴레이션을 나타내기 위해서 포인터를 사용하고, 인덱스 구조로서는 AVL-tree와 B-tree의 혼합 방식인 T-tree 구조를 사용하는 것이 이러한 시스템에 더 적합하다는 연구 결과들은 주기억 장치 데이타베이스가 기존의 시스템에 비해 달라져야 하는 기능의 한 예라고 할 수 있다.

현재의 하드웨어 발전 추세로 이러한 주기억 장치 데이타베이스 시스템은 조만간 실현 가능하게 될 것이며 유용하게 쓰일 수 있을 것이다.

참 고 문 헌

1) A. Ammann, M. Hanrahan, and R. Krishnamurthy, "Design of a Memory Resident DBMS", Proc. IEEE COMPCON, San Francisco, Feb. 1985, pp. 54-57.

2) J.P. Armisen and J.Y. Caleca, "A Commecial Back-End Data Base System", Proceedings of the Seventh International Conference on VLDB, 1981, pp. 56-55.

3) P.A. Bernstein and N. Goodman, "Timestamp-based Algorithms for Concurrency Control in Distributed Database Systems", Proceedings of the Sixth International Conference on VLDB, Oct. 1980, pp. 285-300.

4) D. Bitton, "The Effect of Large Main Memory on Database Systems", Proceedings of the ACM SIGMOD International Conference on Management of Data, May 1986, pp. 337-339.

5) D.J. DeWitt, "A Multiprocessor Organization for Supporting Relational Database Management Systems", IEEE Trans. on Computers, Vol. C-28, No. 6, Jun. 1979.

6) D.J. DeWitt, R.H. Katz, F. Olken, L.D. Shapiro, M.R. Stonebrake and D. Wood, "Implementation Techniques for Main Memory Database Systems", Technical Report No. 529, Department of Computer Science, University of Wisconsin, Madison, Jan. 1984.

7) D.J. DeWitt, R.H. Gerber, G. Graefe, M.L. Heytens, K.B. Kumar and M. Muralikrishna, "GAMMA-A High Performance Detaflow Database Machine", Proceedings of the Twelfth International Conference on VLDB, 1986, pp. 228-237.

8) M. Eich, "MMDB Recovery", Technical Report No. 86-CSE-11, Department of Computer Science, Southern Methodist University, Mar. 1986.

9) K.P. Eswaren, et al., "The Notions of Consistency and Predicate Locks in Database Systems", CACM, Nov. 1976, pp. 624-633.

10) R. Fagin, J. Neivergeit, N. Pippenger and H.R. Strong, "Extendible Hashing: A Fast Access Method for Dynamic Files," ACM Trans. on Database Systems, Vol. 4. No. 3, Sep. 1979, pp. 315-344.

11) S. Fushimi, M. Kitsuregawa and H. Tanaka, "An Overview of System Software of a Parallel Relational Database Machine GRACE", Proceedings of the Twelfth International Conference on VLDB, 1986, pp. 209-219.

12) H. Garcia-Molina, R.J. Lipton and P. Honeyman, "A Massive Memory Database System", Technical Report No. 315, Department of Electrical Engineering and Computer Science, University of Princeton, Sep. 1983.

13) H. Garcia-Molina, R.J. Lipton and J. Valdes, "A Massive Memory Machine", IEEE Trans. on Computers, Vol. C-33, No. 5, May 1984.

14) J.N. Gray, "Notes on Data Base Operating Systems", Lecture Notes in Computer Science, Springer-Verlag, 1978, pp. 393-481.

15) J.N. Gray, "The 5 Minute Rule", Technical Note, Tandem Computer, May 1985.

16) R. Hagmann, "A Crash Recovery Scheme for a Memory-Resident Database System", IEEE Trans. on Computers, Vol. C-35, No. 9, Sep. 1986, pp. 839-843.

17) D.K. Hsiao, Advanced Database Machine Architecture, Reading, New Jersey: Prentice-Hall, 1983.

18) IBM, IMS Version 1 Release 1.5 Fast Path Feature Description and Design Guide, IBM World Trade Systems Centers, 1979.

19) IBM, Guide to IMS/VS V1 R3 Data Entry Database (DEDB) Facility, IBM International Systems Centers, 1984.

20) D. Knuth, The art of Computer Programming, Addison-Wesley, Reading, Mass., 1983.

21) I. Kojima and Y. Kambayashi, "Hash-Based File Organization Utilizing Large Capacity Main Memory", Proceedings of International Conference of Foundation of Data Organization, 1985.

22) H.T. Kung and J.T. Robinson, "On Optimisic Method for Concurrency Control", ACM Trans. on Database Systems, Vol. 6, No. 2, Jun. 1981, pp. 213-226.

23) T.J. Lehman and M.J. Carey, "Query Processing in Main Memory Database Management Systems", Proceedings of the ACM SIGMOD International Conference ofn Management of Data, May 1986, pp. 239-250.

24) T.J. Lehman and M.J. Carey, "A Study of Index Structures for Main Memory Detabase Management Systems", Proceedings of the Twelfth International Conferenee on VLDB, Aug. 1986, pp. 294-303.

25) T.J. Lehman, "Design and Performance Evaluation of a Main Memory Relational Database System", Technical Report No. 656, Department of Computer Science, University of Wisconsin, Madison, Aug. 1986.

26) M. Lelard and W. Roome, "The Silicon Database Machine", Proceedings of the Fourth International Workshop on Database Machines, Grand Bahama Island, Mar. 1985.

27) W. Litwin, "Linear Hashing: A New Tool for File and Table Addressing", Proceedings of Sixth International Conference on VLDB, Montreal, Canada, Oct. 1980.

28) K. Salem and H. Garcia-Molina, "Crash Recovery Mechanisms for Main Storage Database Systems", Technical Reprot No. CS-TR-034-86, Department of Computer Science, University of Princeton, Apr. 1986.

29) L.D. Shapiro, "Join Processing in Database Systems with Large Main Memories", ACM Trans. on Database Systems, Vol. 11, No. 3, Sep. 1986, pp. 239-264.

30) A.P. Sheth and M.T. Liu, "Intergrating Locking and Optimistic Concurrency Control in Distributed Database Systems", IEEE Proceedings of the Sixth International Conference on Distributed Computing Systems, May 1986, pp. 88-99.

31) P. Valduriez and G. Gardarin, "Join and Semijoin Algorithms for a Multiprocessor Database Machine", ACM Trans. on Database Systems, Vol. 9, No. 1, Mar. 1984, pp. 133-161.