**LOCK-FREE, SCALABLE READ ACCESS TO SHARED DATASTRUCTURES**

**잠금이 풀리고 확장 가능한 읽기 액세스**

1

공유 데이터 구조

기술 분야

본 명세서에 기술 된 주제는 기술에 관한 것이다

공유 객체에 대한 잠금 및 확장 가능한 읽기 액세스 제공

데이터베이스 커널에 포함됩니다.

배경

공유 데이터 구조에 대한 동기화 된 액세스가 필요합니다.

데이터 일관성을 보장하기위한 많은 컴퓨터 프로그램

공유 구조의. 많은 경우에, 그러한 공유 구조는 상대적으로 거의 수정되지 않지만 꽤 자주 읽습니다. 에서

데이터 일관성을 보장합니다. 이러한 구조는

수정 전용 읽기 / 쓰기 잠금을 사용하여 잠김

기본 데이터의 양이온이며 읽기 위해 공유됩니다.

작업. 그러나 읽기/쓰기 잠금은 특별히

저렴한 동기화 프리미티브 및 읽기 액세스

CPU에서 L2- 캐시 누락이 발생하여 seri가 발생할 수 있음

다중 코어 컴퓨팅 시스템의 성능을 크게 제한합니다.

이러한 문제는 경감 될 수 있지만

단일 읽기 / 쓰기에 대한 메모리 사용량이 많음

자물쇠. 특히 하나의 메모리 캐시 라인을 예약 할 수 있습니다.

각 CPU 코어는 해당 코어에서 공유 잠금

독점 잠금이 없을 때 캐시 라인을 계산할 수 있습니다

선물 요청.

그러나 그러한 배치에서는 적어도 두 가지 문제

여전히 남아있다. 첫째, 독점 액세스 권한은

해당 작업이 com이 될 때까지 공유 구조

주름을 잡았다. 이러한 제한으로 인해 성능 병목 현상이 발생할 수 있습니다.

특히 현대의 많은 핵심 아키텍처로

100 개 이상의 CPU 코어를 초과합니다. 인 메모리 데이터의 맥락에서

기지가 있기 때문에 문제가 더 두드러집니다.

쿼리 실행 시간을 지배하는 I / O 시간이 없습니다.

둘째, 하나를 사용하여 최적화 된 읽기 / 쓰기 잠금을 사용하더라도

CPU 코어 당 캐시 라인, 과도한 수정로드로 인해

독점 잠금 대기 중 높은 비율의 L2 캐시 누락.

이상적으로 공유 독자는 절대로 modi에 의해 차단되어서는 안됩니다

내부 구조의 허구.

개요

일 양상에서, 적어도 하나의 적어도 하나의 판독 동작

객체 그래프의 객체가 시작됩니다. 객체 그래프 char

루트 객체를 포함하여 객체의 계층 구조를 활성화합니다.

노드의 적어도 일부에 해당하는

를 가리키는 모서리 수를 지정하는 참조 횟수

관련 개체. 호환되는 두 가지 변경 사항을 모두 적용 할 수 있습니다

호환되지 않는 변경뿐만 아니라 그래프의 개체 그곳에

각 해당 읽기 작업에 대해 참조 후

루트 객체의 개수가 증가합니다 (참조와 함께)

처음에는 단일을 반영하는 루트 객체의 수 1

루트 객체를 가리키는 앵커 포인터). 결과적으로 하나

하나 이상의 개체가 변경되었습니다. 호환되지 않는 변경

루트 객체, 하나 이상의 변경된 객체 및

존재하는 객체의 계층 구조 내에서 객체를 개재

복제되었습니다. 그런 다음 앵커 포인터는 복제 된 루트에 연결됩니다

목적. 루트 객체는 나중에 참조 할 때 삭제됩니다

루트 객체의 개수는 0입니다. 또한 객체 그래프

루트 개체에서 시작하여 최소한 끝나는 경우

하나의 변경된 객체가 해당 객체의 모서리를 제거합니다.

그런 다음 참조 카운트가 0 인 각 객체가 삭제됩니다.

일부 구현에서, 참조 카운트는

둘 이상의 CPU에 걸쳐 스트라이프됩니다. 그러한 배치는

L-2 캐시 미스를 줄인다는 점에서 유리합니다.

2

일부 또는 전부 객체는 메모리 내 데이터베이스에 저장 될 수 있습니다. 그만큼

데이터베이스는 단조 증가로 행에 데이터를 저장할 수 있습니다

ing 행 식별자. 데이터베이스는 다중 버전을 사용할 수 있습니다

통화 관리.

다른 양상에서, 적어도 하나의 페이지의 판독 동작은

테이블이 시작됩니다. 읽기 작업은 앵커 객체를 사용합니다.

적어도 하나에 액세스하기 위해 제 1 테이블 헤더 객체를 가리키는 단계

페이지. 첫 번째 테이블 헤더 개체는 첫 번째 테이블에 대한 링크를 포함합니다.

복수의 페이지에 대한 핸들을 포함하는 링크 된 객체

하나 이상의 페이지를 포함합니다. 그 후, 연결된 객체

테이블 헤더 객체는

복제 된 링크 된 오브젝트가 포함하도록 읽기 조작

적어도 하나를 포함하는 복수의 페이지에 대한 핸들

페이지. 결과적으로 앵커 객체는

복제 된 테이블 헤더 객체. 하나의 추가 페이지를 연결할 수 있습니다

복제 된 연결된 개체에. 이후 첫 번째 테이블 헤더

읽은 후 오브젝트와 첫 번째 링크 된 오브젝트가 삭제됩니다.

작업이 완료되었습니다.

테이블은 메모리 내 데이터베이스에 저장 될 수 있습니다. 그만큼

제 1 링크 된 객체는

참조 횟수. 인 메모리를 강제하는 참조 카운트

첫 번째 테이블 헤더 개체와 첫 번째 테이블을 유지 관리하는 데이터베이스

읽기 작업 중 연결된 개체 복제 된 테이블

헤더 오브젝트는 처음에 첫 번째 링크 된 오브젝트에 링크 될 수 있습니다.

첫 번째 테이블 헤더 개체는 관련된 메타 데이터를 가질 수 있습니다.

해당 페이지를 특성화합니다. 복제 된 테이블

헤더 객체는 메타 데이터와도 연관 될 수 있습니다.

첫 번째 테이블 헤더 객체와 함께 인용됩니다.

비 일시적 컴퓨터 프로그램 제품도

컴퓨터 실행 명령을 저장하는

적어도 하나의 하나 이상의 데이터 프로세서에 의해 실행될 때

컴퓨터, 적어도 하나의 컴퓨터가 오페라를 수행하게한다

여기에 언급되어 있습니다. 마찬가지로 컴퓨터 시스템도 설명됩니다

프로세서와 메모리에 연결된 메모리를 포함 할 수 있습니다.

프로세서. 메모리가 일시적으로 또는 영구적으로

프로세서가 프로세서를 작동하게하는 하나 이상의 프로그램을 저장

본 명세서에 기술 된 동작들 중 하나 이상을 형성한다. addi에서

방법으로 지정된 작업은 다음과 같이 구현할 수 있습니다.

단일 컴퓨팅 내에서 하나 이상의 데이터 프로세서

시스템 또는 둘 이상의 컴퓨팅 시스템에 분산 될 수있다.

본 명세서에 기술 된 주제는 많은 진보를 제공한다

태그. 예를 들어, 현재 주제와 함께

공유 데이터 구조에 액세스하려는 작업은 결코

CPU 리소스를 훨씬 더 잘 사용할 수 있습니다.

테이블 수정로드가 심한 경우에도 마찬가지입니다.

주제의 하나 이상의 변형에 대한 세부 사항

본 명세서에 기술 된 것은 첨부 도면들에 기재되어있다

그리고 아래의 설명. 다른 특징 및 장점

본 명세서에 기술 된 주제는

설명 및 도면, 및 청구항으로부터.

도면의 설명

무화과. 도 1은 데이터를 포함하는 시스템을 도시 한 도면

저장 애플리케이션;

무화과. 도 2a는 로크를 도시 한 제 1 프로세스 흐름도이다.

데이터 구조에 대한 무료 확장 가능 액세스;

무화과. 도 2b는 잠금을 나타내는 제 2 프로세스 흐름도이다.

데이터 구조에 대한 무료 확장 가능 액세스;

도 3은도 1의 시스템의 세부 사항을 도시 한 도면이다.

1; 과

도. 도 4A-4K는 작성 및 사용을 나타내는 도면이다

복제 된 테이블 헤더 오브젝트 및 복제 된 링크 된 오브젝트

공유 데이터 구조에 대한 읽기 액세스 제공

호환되지 않는 수정.

상세 설명

무화과. 도 1은 시스템 (100)이

하나 이상의 프로를 포함 할 수있는 컴퓨팅 시스템 (102)

3

함께 배치 할 수있는 그래밍 가능한 프로세서

하나 이상의 네트워크 등이 하나 이상의 모듈을 실행합니다.

데이터 저장 애플리케이션의웨어 구성 요소 등

데이터 저장 애플리케이션 (104)은 하나 이상의 것을 포함 할 수있다.

데이터베이스, 기업 자원 프로그램, 분산

스토리지 시스템 (예 : NetApp에서 제공하는 NetApp Filer

Sunnyvale, Calif.) 등.

하나 이상의 모듈, 소프트웨어 구성 요소 등

컴퓨팅 시스템 (102)의 로컬 사용자가 액세스 할 수 있음

컴퓨팅 시스템에 액세스하는 원격 사용자뿐만 아니라

네트워크를 통해 하나 이상의 클라이언트 머신 (106)으로부터 102

연결 110. 생성 된 하나 이상의 사용자 인터페이스 화면

하나 이상의 제 1 모듈에 의해 사용자에게 디스플레이 될 수 있고,

로컬 디스플레이 또는 하나와 관련된 디스플레이를 통해

데이터 저장 장치의 데이터 단위

애플리케이션 (104)은 지속성 계층에 일시적으로 저장 될 수있다

112 (예 : 페이지 버퍼 또는 다른 유형의 임시 지속성

스토리지 페이지 형식으로 데이터를 쓸 수있는

하나 이상의 저장소 (114)에, 예를 들어 입력 / 출력을 통해

하나 이상의 스토리지 (114)는 다음을 포함 할 수있다.

하나 이상의 물리적 스토리지 미디어 또는 장치 (예 : 하드 디스크)

드라이브, 영구 플래시 메모리, 랜덤 액세스 메모리, opti

쓰기를 위해 구성된 칼 미디어, 자기 미디어 등)

장기 저장을위한 데이터. 유의해야합니다

스토리지 (114) 및 입 / 출력 컴포넌트 (116)는

그들의 존재에도 불구하고 컴퓨팅 시스템 (102)에 포함됨

도 1의 컴퓨팅 시스템 (102)의 외부에 도시되어있다. 1.

장기 저장소 (114)에 보유 된 데이터는 오가 일 수있다

각 페이지에 정의 된

저장 공간의 양. 일부 구현에서

각 페이지에 할당 된 저장 공간의 양은

스텐트와 고정. 그러나 다른 구현에서는

각 페이지에 할당 된 저장 공간의 양은 다를 수 있습니다

또한 현재 주제의 범위 내에있다.

무화과. 도 2a는 방법을 예시하는 프로세스 흐름도이다.

210A에서, 적어도 하나의 판독 동작이 적어도 200A 인 200A

객체 그래프의 한 객체가 시작됩니다. 객체 그래프

루트 객체를 포함하여 객체의 계층 구조를 나타냅니다.

노드의 적어도 일부에 해당하는

를 가리키는 모서리 수를 지정하는 참조 횟수

관련 개체. 호환되는 두 가지 변경 사항을 모두 적용 할 수 있습니다

호환되지 않는 변경뿐만 아니라 그래프의 개체 그곳에

220A에서, 대응하는 각각의 판독 동작에 대해 ref 이후

해당 객체의 erence count가 증가합니다.

처음에 루트 객체를 반영하는 루트 객체의 참조 횟수

루트 객체를 가리키는 앵커 포인터). 이후에

230A에서, 하나 이상의 객체가 변경된다. 호환되지 않습니다

240A에서 루트 오브젝트의 결과를 하나 이상 변경

변경된 오브젝트 및 계층 내의 모든 중간 오브젝트

복제되는 객체의 아치형. 그러면 앵커 포인터는

250A에서 복제 된 루트 개체에 연결됩니다. 루트 객체는

나중에 260A에서 루트에 대한 참조 카운트가 떨어졌을 때

개체가 0입니다. 또한 객체 그래프는

270A, 루트 객체에서 시작하여 적어도 하나에서 끝남

이러한 객체의 모서리를 제거하여 객체를 변경했습니다. 마다

그런 다음 참조 카운트가 0 인 객체가 삭제됩니다.

무화과. 도 2B는 210B에서,

테이블에 저장된 적어도 하나의 페이지의 읽기 동작

인 메모리 데이터베이스의 메모리가 시작됩니다. 읽은

오퍼레이션은 첫 번째 테이블 헤더를 가리키는 앵커 오브젝트를 사용합니다.

하나 이상의 페이지에 액세스 할 수 있습니다. 첫 번째 테이블 헤더

객체는 포함 된 첫 번째 연결된 객체에 대한 링크를 포함합니다.

적어도 하나의 페이지를 포함하는 복수의 페이지를 처리한다.

그 후, 220B에서, 링크 된 오브젝트 및 테이블 헤더

읽기 작업과 동시에 개체가 복제됩니다. 그만큼

복제 된 링크 된 객체는 복수의 페이지에 대한 핸들을 포함

하나 이상의 페이지를 포함합니다. 다음으로 230B에서 앵커

4

object가 복제 된 테이블 헤더 개체를 가리 키도록 설정되어 있습니다.

240B, 하나 이상의 추가 페이지가 복제 된

연결된 객체. 이후 250B에서 첫 번째 테이블 헤더

읽은 후 오브젝트와 첫 번째 링크 된 오브젝트가 삭제됩니다.

작업이 완료되었습니다.

달리 말하면, 읽기 조작은 루트 오브젝트로 시작합니다

객체 그래프의 (별칭 헤더 객체) (일 예에서

테이블 헤더) 및 이것에 하나의 (가상) 참조를 추가합니다

목적. 이 객체 그래프 (유향 그래프)에서

참조를 통해 연결된 추가 객체를 가질 수 있음

다른 물체에서. 일반적으로이 그래프는 나무입니다.

사이클). 객체는 그래프의 노드 / 정점, 참조

가장자리 (방향)입니다. 객체에 대한 각 참조

그래프의 위치가 계산되므로 객체의 참조 횟수

이 특정 객체를 가리키는 가장자리 수입니다.

루트의 참조 카운트는 1 (앵커의 경우)에 숫자를 더한 값입니다.

독자의 베르. 이 참조 카운트는 실제로 spe에 저장됩니다

L2- 캐시를 방지하기 위해 cially (CPU 코어에서 스트라이핑 됨)

갈등.

호환되는 변경은 그래프를 변경하거나 참조하지 않습니다

따라서 그래프의 개체 중 하나만 변경됩니다.

호환되지 않는 변경 사항은 영향을받는 개체와

그래프의 루트까지의 객체 (루트 / 헤더 포함)

영향을받는 객체 사이에 새로운 가장자리를 만듭니다.

새로 복제 된 객체와 오리지널 사이의 새로운 가장자리

이전 버전의 해당 객체에서 nally 연결된 객체

(진행중인 참조 횟수 증가). 다른 모든 개체

동일하게 유지하고 원래 객체에서 다시 연결

그래프 (즉, 두 개의 근을 가진 그래프가 나타납니다.

원래 루트 / 헤더 개체 및 복제 된 개체). 때

앵커 참조는 복제 된 원본, 원본으로 전송됩니다.

루트 객체는 참조를 잃습니다. 독자가없는 경우

카운트가 0으로 떨어지면이 개체는

제거되었습니다. 전 이적 으로이 객체에서 시작하는 가장자리는

또한 제거되어 참조 카운트가 감소합니다.

연결된 개체와 결국 제거됩니다. 이 계속

제거 할 것이 없을 때까지. 독자가 있으면

여전히 원래 루트 / 헤더 객체에 대한 참조를 보유하므로

전체 프로세스는 마지막 독자가 완료된 후에 만 ​​발생합니다.

무화과. 도 3은 다음과 일치하는 소프트웨어 아키텍처 (300)를 도시한다

현재 주제의 하나 이상의 특징. 데이터

하나에서 구현 될 수있는 스토리지 애플리케이션 (104) 또는

하나 이상의 하드웨어 및 소프트웨어에 하나 이상의

데이터베이스 응용 프로그램, 네트워크 연결 스토리지 시스템 또는

등. 적어도 일부 구현에 따르면

현재 주제, Sucha 데이터 저장 애플리케이션 (104)은

지속성 층 (112)을 포함하거나 그렇지 않으면 인터페이스

다른 유형의 메모리 버퍼 (예 : 지속성을 통한)

인터페이스 (302). 지속성 계층 내의 페이지 버퍼 (304)

112는 하나 이상의 논리 페이지 (306)를 저장할 수 있으며, 선택적으로

섀도 페이지, 활성 페이지 등을 포함 할 수있다. 그만큼

지속성 층 (112)에 보유 된 논리 페이지 (306)는

스토리지를 통해 스토리지 (예를 들어, 용어 스토리지 등) (114)에 기록됨

입 / 출력 컴포넌트 (116), 소프트웨어 모드 일 수 있음

ule, 하나 이상의 소프트웨어로 구현 된 하위 시스템

하드웨어 등. 저장 부 (114)는 하나 이상을 포함 할 수있다

저장된 페이지 (312)가 할당되는 데이터 볼륨 (310)

물리적 메모리 블록.

일부 구현들에서, 데이터 저장 애플리케이션 (104)

페이지를 포함하거나 이와 통신 할 수 있음

관리자 (314) 및 / 또는 세이브 포인트 관리자 (316).

연령대 (314)는 페이지 관리 모듈과 통신 할 수있다

자유 블록을 포함 할 수있는 지속 계층 (112)에서 320

페이지 상태 정보 (324)를 모니터링하는 관리자 (322)는

예를 들어 저장소 (114) 내의 물리적 페이지의 상태

지속성 계층 (112)의 논리 페이지 (및 선택적으로)

페이지 버퍼 (304)에서). 세이브 포인트 관리자 (316)는

5

Persis에서 세이브 포인트 코디네이터 (326)와 통신

저장 점을 처리하는 텐스 레이어 (204)

이후에 재시작을 위해 데이터베이스의 일관성있는 지속적 상태

가능한 충돌.

데이터 저장 애플리케이션 (104)의 일부 구현들에서.

지속성 계층 (112)의 페이지 관리 모듈은

섀도 페이징을 구현합니다. 자유 블록 관리자 322

페이지 관리 모듈 (320) 내에서

실제 페이지의 상태. 페이지 버퍼 (304)는

본 명세서에서 논의 된 바와 같이 동작하는 고정 페이지 상태 버퍼. 에이

commu의 일부이거나 commu 일 수있는 변환기 구성 요소 340

페이지 관리 모듈 (320)과의 니케이션은

논리적 페이지와 물리적 페이지 간의 매핑을 담당

변환기 (340)는 저장 부 (114)에 기록 될 수있다.

해당 물리에 대한 논리 페이지의 현재 매핑

변환기 테이블 (342)의 페이지를 계산한다. 변환기 (340)는

논리 페이지 306의 현재 매핑을

하나 이상의 변환기 테이블에서 실제 페이지 응답

342. 논리 페이지 (306)가 스토리지 (114)로부터 판독 될 때,

로드 할 스토리지 페이지는 하나 또는

변환기 (340)를 사용하는 더 많은 변환기 테이블 (342).

논리 페이지는 다음에 처음으로 스토리지 (114)에 기록된다

세이브 포인트, 새로운 무료 실제 페이지가 논리에 할당됩니다.

페이지. 자유 블록 관리자 (322)는 새로운 물리를 표시한다

페이지를 "사용 중이며 새 매핑이 하나 또는

더 많은 변환기 테이블 342.

지속성 계층 (112)은

데이터 저장 애플리케이션 (104)은 내구성이 있고 데이터가

저장 애플리케이션 (104)은 가장 최근에 복원 될 수있다

재시작 후 커밋 된 상태입니다. 저장소에 데이터 쓰기 114

쓰기 트랜스의 끝과 동기화 할 필요가 없습니다

동작. 따라서 커밋되지 않은 변경 사항을 디스크에 쓸 수 있습니다

커밋 된 변경 사항이 디스크에 아직 기록되지 않을 수 있습니다.

쓰기 거래가 완료되었습니다. 시스템 충돌 후

완료되지 않은 거래에 의한 변경은

복구하다. 이미 커밋 된 트랜스에서 발생하는 변경

이 프로세스에서 조치를 유실해서는 안됩니다. 로거 구성 요소

데이터에 대한 변경 사항을 저장하기 위해 344도 포함될 수 있습니다

선형 로그에서 데이터 스토리지 애플리케이션의 로거

구성 요소 344는 복구 중에 오페라를 재생하는 데 사용될 수 있습니다.

마지막 저장 점 이후 모든 작업이 수행되도록 보장

데이터 및 로그 된 "com"과의 트랜잭션에 적용

정지 상태로 롤백하기 전에 mit 레코드가 커밋 됨

복구 프로세스 종료시 트랜잭션.

일부 데이터 스토리지 응용 프로그램에서 디스크에 데이터 쓰기

반드시 글의 끝과 동기화되지는 않습니다

트랜잭션. 커밋되지 않은 상황이 발생할 수 있습니다.

변경 사항은 디스크에 기록되는 동시에

쓸 때 커밋 된 변경 사항은 아직 디스크에 기록되지 않습니다

ing 거래가 완료되었습니다. 시스템 충돌 후 변경

완료되지 않은 거래에 의해 만들어진 롤

커밋 된 트랜잭션에 의한 변경 및 변경 사항이 손실되지 않아야합니다.

커밋 된 변경 내용이 손실되지 않도록하려면 로그를 다시 실행하십시오.

정보는 로거 컴포넌트 (344)에 의해 기록 될 수있다

변경 될 때마다 이 정보는 쓸 수 있습니다

트랜잭션이 끝날 때 디스크에 최근에. 로그 항목은

일반 데이터가있는 동안 별도의 로그 볼륨에 유지

데이터 볼륨에 기록됩니다. 재실행 로그를 사용하여 커밋 된 변경 사항

해당 데이터 페이지가 아닌 경우에도 복원 가능

디스크에 기록됩니다. 커밋되지 않은 변경 사항을 취소하려면

지속성 계층 (112)은 실행 취소 로그 엔트리의 조합을 사용할 수있다

(하나 이상의 로그에서) 및 섀도 페이징.

지속성 인터페이스 (302)는 읽기 및 쓰기를 처리 할 수있다

상점 요청 (예 : 인 메모리 상점 등). 페르 시스

텐스 인터페이스 (302)는 또한 기록을위한 기록 방법을 제공 할 수있다

로깅이 있거나없는 데이터. 기록 된 경우

기록 동작이 사용되며, 지속성 인터페이스 (302)

6

로거 (344)는 로거 (344)를 호출한다.

매장 (예 : 인 메모리 매장 등)이

로그 항목을 로그 큐에 직접 추가하십시오. 로거 인터페이스

에서 로그 항목을 요청하는 방법도 제공합니다.

메모리 로그 큐가 디스크로 플러시됩니다.

로그 항목에는 로그 시퀀스 번호,

로그 항목 및 트랜잭션 식별자 에 따라

작업 유형 추가 정보는

로거 344. "update"유형의 항목 (예 :

영향을받는 레코드와 그 이후의 식별

수정 된 데이터의 이미지.

데이터 애플리케이션 (104)이 재시작 될 때, 로그 엔트리

처리해야합니다. 이 과정의 속도를 높이기 위해 리두 로그는

명시된대로 항상 처음부터 처리되는 것은 아닙니다.

위의 모든 것을 저장하는 저장 점을 주기적으로 수행 할 수 있습니다.

메모리 등에서 만들어진 디스크 변경

마지막 세이브 포인트 시스템을 시작할 때 로그 만

마지막 세이브 포인트를 처리 한 후 생성됩니다. 후

다음 백업 작업 저장 점 이전의 이전 로그 항목

위치를 제거 할 수 있습니다.

로거 (344)가 로그 엔트리를 기록하기 위해 호출 될 때, 로거 (344)

디스크에 즉시 쓰지 않습니다. 대신 로그를 넣을 수 있습니다

메모리의 로그 큐에 항목. 로그의 항목

대기열은 해당하는 경우 최신 디스크에 쓸 수 있습니다

ing 트랜잭션이 완료되었습니다 (커밋 또는 중단). guar로

커밋 된 변경 사항이 손실되지 않도록주의하십시오.

해당하기 전에 작업이 성공적으로 완료되지 않았습니다

ing 로그 항목이 디스크로 플러시됩니다. 에 로그 큐 항목 쓰기

디스크는 다른 이벤트에 의해 트리거 될 수도 있습니다 (예 :

로그 큐 페이지가 가득 찼거나 저장 점이 수행 될 때.

현재 주제로, 로거 (344)는

데이터베이스 로그 (또는 여기서는 간단히 "로그"라고 함)

자연스러운 순서로 메모리 버퍼에 tially (예 : 순차)

주문 등). 여러 개의 물리적 하드 디스크 / 스토리지 장치가있는 경우

로그 데이터를 저장하는 데 사용되는 여러 로그 파티션을 정의 할 수 있습니다.

그 후, 로거 (344) (상기 언급 된 바와 같이

로그 데이터 생성 및 구성)

로그는 사용 가능한 전체 로그 파티션을 버퍼링합니다. 경우에 따라

로드 균형 조정은 라운드 로빈 분포에 따릅니다.

다양한 작문 작업을 지향하는 계획

순차적이고 연속적인 방식으로 로그 버퍼. 이것으로

배열, 로그 버퍼는 단일 로그 세그먼트에 기록

다중 파티션 로그의 특정 파티션은 결과가 아닙니다

tive. 그러나 로그 버퍼는 로그에서 다시 정렬 될 수 있습니다.

올바른 순서로 복구하는 동안 모든 파티션의 세그먼트.

전술 한 바와 같이, 데이터 저장 애플리케이션 (104)은

세이브 포인트 관리자 (316)가

트랜잭션에 일관성이있는 저장 점. 그런 준비로

데이터 백업은 모든 데이터 페이지의 사본으로 구성됩니다.

첫 번째로 수행 된 특정 저장 점에 보관

데이터 백업 프로세스의 단계. 현재 주제

다른 유형의 데이터 페이지 스토리지에도 적용 할 수 있습니다.

데이터 저장 애플리케이션 (104)은 다중 버전을 이용할 수있다

트랜잭션 격리 및 연결을위한 동시 제어 (MVCC)

성실한 독서. 데이터베이스의 각 행을 연결할 수 있습니다

독특하고 단조 증가하는 식별자 (RowlD).

레코드의 새 버전이 작성되면이 새 버전

또한 새로운 RowlD가 될 수 있습니다 (예 : MVCC 의미론으로 인해

이전 버전은 병렬 판독기를 위해 보관해야하며

커밋 후 가비지 수집 중에 만 청소).

여기에서 페이지에 대한 참조는 테이블의 페이지를 참조 할 수 있습니다.

메모리 내부 데이터베이스의 메모리에 저장되어

MVCC 기반 데이터로

기본 테이블 구현, 모든 내부 임시 데이터 오브젝트

테이블의 버전을 지정할 수 있습니다. 이러한 데이터 객체에는 다음이 포함될 수 있습니다.

헤더 객체, 메타 데이터 객체, 기타 내부 상태 표

7

(s)로드 된 페이지의 벡터, 사전 해시 / 트리 등

압축 된 원주 테이블 등.

현재의 주제로 모든 테이블 제어 구조

독자가 사용하는 버전을 지정할 수 있습니다. 이러한 구조에는 다음이 포함됩니다.

예를 들어 페이지 목록, 값 간접 벡터, 내부

메타 데이터 등 독자는 데이터에 대한 잠금을 얻지 못합니다

구조가 아니라 현재 버전의 데이터로 작업

쿼리 또는 쿼리 계획 연산자가 끝날 때까지 구조. 이것으로

배열, 이전 버전은 짧은 기간 동안 만 남아 있습니다

시간 (예 : 서브 초). AS 버전 객체는 일반적으로

작은 메모리 오버 헤드도 작습니다. 또한

OLTP 시스템에서 호환되지 않는 변경은 거의 없습니다 (예 :

많은 동시 버전 등). 또한 일부

이전 버전의 우선 순위가 큰 / 큰 객체 인 경우 구현

(예를 들어, 원주 테이블의 주요 부분 등) 여전히 존재하며 새로운 것은 없습니다.

해당 객체의 버전을 만들 수 있습니다. 에 대한

예를 들어, 리더의 주요 부분에서 스캔을 수행하는 리더가있는 경우

원주 테이블 병합 중 시작된 원주 테이블

버전 n-1에서 버전 n까지,이 스캔은

버전 n-1. 버전 n으로의 병합이 완료된 후에도

버전에서 버전 n + 1로 병합하는 것이 오래 방지됩니다

버전 n-1의 주요 부분에서 실행되는 스캔이 있으므로

이것은 메모리 요구를 엄청나게 증가시킬 수 있습니다).

다음은 다이어그램 400A를 참조합니다.

도 1 내지도 3에 도시 된 400K 4A-4J. 를 나타내는 객체

메모리의 테이블은 트리에서 구성 할 수 있습니다.

테이블 헤더 객체 (402)는 앵커 포인터에 의해 지시된다

404. 앵커 포인터 (404)는 루트 객체를 가리키는 데 사용된다

테이블의. 모든 버전과 마찬가지로 테이블 헤더 객체

개체에는 참조 횟수가 있습니다. 도 3의 다이어그램 (400A)에서.

도 4a에서, 참조 카운트는 1이다. 메타 데이터 (406)는 특성화 할 수있다

헤더 객체로 표시되는 테이블에 대한 다양한 측면

402 (즉, 테이블 열과 데이터 형식을 정의하기 위해

스트레인 등). 테이블은 페이지 벡터 객체 (408)에 링크된다.

차례로 (페이지 핸들을 통해) 복수의 페이지로 링크 (410)

n (이 예에서는 첫 번째 페이지 410과 두 번째 페이지에 연결됨

메모리에로드 된 메모리 내 테이블의 410 페이지)를 참조하십시오. 페이지

핸들은로드 된 페이지를 고정하는 특별한 유형의 포인터입니다.

메모리에. 현재 배치로

내부 구조는 항상 동기화 될 수 있습니다

mutex lock 또는 in과 같은 일부 수단을 통한 동시 액세스

모든 수정을 실행하여 더 복잡한 구현

메시지 전달 서브의 단일 작업자에서 작업

체계.

현재 주제는 다음과 같을 수 있음을 이해할 것이다.

다양한 다른 물체와 관련하여 활용

그 페이지 벡터 객체는 많은 구현체 중 하나로 사용됩니다.

스테이션. 예를 들어, 현재 주제를 적용 할 수 있습니다

사전 값 벡터와 같은 객체, 사전 해시

및 / 또는 검색 트리, 다양한 메타 데이터 객체 등.

페이지 벡터의 경우 델타 파트에 대해 하나의 페이지 벡터가있을 수 있습니다.

병합 중 실행되는 델타 및 테이블

조작. main에 여러 페이지 벡터가있을 수도 있습니다.

부품 열 (또는 선형 공간의 여러 압축 벡터).

일반적으로 두 가지 유형의 수정이 있습니다.

호환 가능한 수정 및 호환되지 않는 수정.

호환 가능한 수정의 예는

모든 페이지를 보유하는 페이지 벡터에 대한 새 페이지 핸들

벡터가 수행하는 한 메모리 내 테이블이 메모리에로드 됨

크기를 조정할 필요가 없습니다.

현재의 주제는 호환되지 않는 모드를 해결합니다

허구. 다시도 1의 다이어그램 (400A)을 참조하면.

도 4a에 도시 된 바와 같이, 페이지 벡터 (408)는 최대 4 개의 페이지에 링크되도록 크기가 정해진 다.

제 3 페이지 (410) 추가 (도표 400B에 도시 된 바와 같이)

도 1의 4B)는 페이지 벡터 (408)의 크기 조정을 필요로하지 않는다.

데이터 구조 변경으로 3 번째 페이지 410

추가되었습니다. 이 제 3 페이지 (410)는이어서 페이지 벡터에 링크된다

8

408 및 테이블 헤더 객체의 참조가 증가합니다

헤더 테이블의 새 버전을 반영하기 위해 2로 멘 션됨

그 후, ref를 갖는 판독 동작의 일부로서

도 1의 다이어그램 (400C)으로 간다. 도 4c에서, 리더 (412)가 시작된다

로카를 식별하기 위해 테이블 ​​헤더 객체 (402)에 액세스하는

하나 이상의 링크 된 페이지들 (410)의 메모리에있는 것들.

도 3의 다이어그램 (400D)을 참조하면 4D로 변경

데이터 구조는 제 4 페이지 (410)가된다

추가되었습니다. 이 제 4 페이지 (410)는이어서 페이지 벡터에 링크된다

이어서,도 4에 도시 된 바와 같이. 4E, 데이터 변경

그 다음, 제 5 페이지 (410s)가 추가된다. 이것에서

경우에, 페이지 벡터 (408)는 핸들을위한 공간을 갖지 않는다

다섯 번째 페이지 410s. 다섯 번째 페이지를 수용하기 위해

410s에서 페이지 벡터 (408)는 복제 페이지로 복제 / 복사된다

벡터 (414) (도 4f의 다이어그램 400F에 도시 된 바와 같이)

처음 네 페이지 (410)에 대한 핸들 a. 또한

도 1의 다이어그램 (400G)을 참조한다. 4G, 테이블 헤더

객체 (402)는 클론 테이블 헤더 객체 (416)로 복제되고,

도 1의 다이어그램 (400H)을 참조하여 설명된다. 복제 된 4H

테이블 헤더 객체 (416)는 원본을 가리 키도록 업데이트된다

페이지 벡터 (408)를 복제 된 페이지 벡터 객체 (414)로 보낸다.

다음으로,도 3의 다이어그램 (400I)을 참조하면. 4I

앵커 포인터 객체 (404)는 복제 된 테이블 헤더로 설정된다

오브젝트 416 및 복제 된 테이블의 참조 번호

헤더 객체 (416)는 0에서 1로 증가하고

원본 테이블 헤더 객체 (404)의 참조 번호는

(2에서 1로 감소)

원래 테이블 헤더에 대한 앵커가 삭제되고 참조

새로운 테이블 헤더에 추가하여

앵커 포인트가 새 헤더 객체를 가리킴). 복제

페이지 벡터 (414)는 적어도 5 개의 페이지에 핸들을 저장할 수 있고,

도 4j의 다이어그램 (400J)을 참조하면, 제 5 페이지 (410s)

복제 된 페이지 벡터 객체 (414)에 링크된다.

도 1의 다이어그램 (400J)을 참조하여 설명된다. 4J는 여전히 읽을 수 있습니다

이전 4 개의 페이지 중 하나에 저장된 데이터

테이블 헤더 객체 (402).

도 4의 다이어그램 400K 4K, 제 6 페이지 (410)가 추가 될 수 있고

복제 된 페이지 벡터 객체 414에 연결됩니다.

도 412를 참조하여 412가 종료된다. 4K, 원본 테이블 헤더

객체 (402) 및 원본 페이지 벡터 객체 (408)는

더 이상 참조가 없으므로 삭제되었습니다.

위에서 알 수 있듯이 영향을받는 데이터

구조 (즉, 페이지 벡터 객체 (412))가 복제되고

에서 경로상의 객체 (예를 들어, 테이블 헤더 객체 (402))

수정 된 객체에 대한 앵커 객체 (404)가 복제된다. 클론

객체의 얕은 복사본으로 간주 될 수 있습니다. 새로운 것만

자식 객체의 버전 (이 경우 새로운 크기의 벡터)

부모의 복제본에 고정되고 다른 자식 개체는

각각의 참조 카운트를 늘리십시오 (표시되지 않음)

그림). 복제가 완료되면 앵커 포인터는

새로 복제 된 루트 개체를 가리 키도록 업데이트되었습니다 (이 경우

테이블 헤더 객체).

독자는 루트 개체의 참조 횟수를 증가시킬 수 있습니다

객체 계층 구조 (현재 예제 테이블 헤더에서)

목적). 이 버전 관리는 이전 버전의

메모리에있는 데이터

동시 수정에도 불구하고 그러한 버전을 작성)

호환되지 않는 구조 변경을 수행했습니다.

또한 데이터베이스가 MVCC를 사용하여 액세스하기 때문에

데이터베이스 테이블 데이터, 이전 데이터에서 동일한 데이터 세트를 읽을 수 있음

새로운 데이터 구조뿐만 아니라 위의 예에서 데이터

새 페이지에 작성된 트랜잭션이 실행중인 트랜잭션에 표시되지 않습니다

독자가 읽을 수 없으므로 독자가 읽을 수 없습니다.

그들. 마찬가지로 새 데이터를 추가하는 호환 가능한 변경 사항

기존 페이지도 독자에게는 변하지 않습니다.

이전 데이터 만 볼 수 있습니다. 반드시 서면으로 주문해야합니다

일관된 더티 읽기를 보장하는 방식으로 읽습니다.

9

내부 구조 (예를 들어 벡터에 먼저 쓰기)

요소를 읽은 다음 요소 OTOH를 먼저 증가시킵니다.

count를 읽은 다음 요소에 액세스합니다). 작가가 먼저

요소 수를 늘린 다음 요소를 씁니다.

독자는 경쟁 조건이있을 것입니다 : 이미 새로운 것을 읽을 수 있습니다

요소 수 및 아직 작성되지 않은 요소에 액세스

초기화 된 메모리), 일반적으로 충돌 및 / 또는

잘못된 결과.

L2 캐시 친 화성을 보장하기 위해 앵커 포인터

객체는 루트 객체와 루트의 참조 카운트를 포함 할 수 있습니다

각 CPU 코어에 대해 개별적으로 객체 포인터. 따라서 독자

원자 적 오페라를 사용하여 코어 전용 캐시 라인 만 수정

참조 횟수를 늘리고 동시에

현재 앵커 포인터. 호환되지 않는 구조

모든 핵심 개인 카운터의 참조 횟수 변경

기준 카운트에 하나씩 집계 될 수 있습니다.

루트 객체 및 앵커 포인터 객체가 원자 적으로 업데이트 됨

새로운 루트 포인터와 제로 참조 카운트를 포함

a와 충돌하는 경우 작업을 반복적으로 반복

리더). 오래된 구조를 벗어난 리더 (다른

앵커 포인터)는 공유 적으로 원자 적으로 감소해야합니다.

루트 객체의 이전 버전에있는 참조 횟수 (

참조 카운트 대신 테이블 헤더 개체)

앵커 포인터의 코어 전용 캐시 라인

본원에 기술 된 주제의 측면은

시스템, 장치, 방법 및 / 또는 기사에 구현

원하는 구성에 따라 특히, 다양한

본원에 기술 된 주제의 구현은

디지털 전자 회로, 집적 회로,

특별히 설계된 응용 분야별 집적 회로

(ASIC), 컴퓨터 하드웨어, 펌웨어, 소프트웨어 및 / 또는

이들의 조합. 이러한 다양한 구현은

하나 이상의 컴퓨터 프로그램에 구현 포함

프로그램에서 실행 및 / 또는 해석 가능한

하나 이상의 프로그래머블 프로세서를 포함하는 시스템

데이터를 수신하기 위해 결합 된 특수 또는 범용 일 수 있으며

a로부터의 지시 및 데이터 및 지시를

저장 시스템, 하나 이상의 입력 장치 및 하나 이상의

출력 장치.

이 컴퓨터 프로그램은

프로그램, 소프트웨어, 소프트웨어 응용 프로그램, 응용 프로그램,

구성 요소 또는 코드에는 전문가를위한 기계 명령어가 포함됩니다.

grammable 프로세서, 높은 수준으로 구현 가능

절차 적 및 / 또는 객체 지향 프로그래밍 언어

및 / 또는 조립 / 기계 언어. 본원에 사용 된

"기계 판독 가능 매체"라는 용어는 임의의 컴퓨터를 지칭한다

프로그램 제품, 장치 및 / 또는 장치. 와 같은

자기 디스크, 광 디스크, 메모리 및 Pro의 예

기계를 제공하는 데 사용되는 grammable Logic Devices (PLDS)

프로그램 가능 프로세서에 대한 명령 및 / 또는 데이터

기계를받는 기계 판독 가능 매체

기계 판독 가능 신호로서의 명령. 용어

"기계 판독 가능 신호"는 프로

기계 명령어 및 / 또는 데이터를 프로그램 가능

프로세서. 기계 판독 가능 매체는

예를 들어,

비 일시적인 솔리드 스테이트 메모리 또는 자기 하드

드라이브 또는 동등한 저장 매체. 기계 판독

유능한 매체는 대안 적으로 또는 추가적으로

과 같은 과도적인 방식으로 기계 명령

프로세서 캐시 또는 기타 임의 액세스와 같은 예

하나 이상의 물리적 프로세서와 관련된 메모리

COCS.

사용자와의 상호 작용을 제공하기 위해 주제

본 명세서에 기술 된 것은

음극선 관 (CRT)과 같은 디스플레이 장치

정보를 표시하기위한 액정 디스플레이 (LCD) 모니터

10

사용자와 키보드 및 포인팅 장치에 연결합니다. 이러한

예를 들어 마우스 또는 트랙볼과 같이 사용자가

컴퓨터에 입력을 제공하십시오. 다른 종류의 장치는

사용자와의 상호 작용을 제공하는 데 사용됩니다. 에 대한

예를 들어, 사용자에게 제공되는 피드백은

감각 피드백, 예를 들어 시각적 피드백, 아우디

토리 피드백 또는 촉각 피드백; 사용자로부터 입력

를 포함하여 (이에 국한되지는 않음) 모든 형태로 수신 될 수 있음

음향, 음성 또는 촉각 입력. 다른 가능한 입력 장치

터치 스크린 또는 기타 터치를 포함하지만 이에 국한되지는 않습니다

단일 또는 멀티 포인트 저항 또는

용량 성 트랙 패드, 음성 인식 하드웨어 및 소프트

도자기, 광학 스캐너, 광학 포인터, 디지털 이미지 캡처

장치 및 관련 통역 소프트웨어 등.

본 명세서에 기술 된 주제는

백엔드 구성 요소를 포함하는 컴퓨팅 시스템

예를 들어 하나 이상의 데이터 서버 또는

미들웨어 구성 요소 (예 : 하나 이상의)

응용 프로그램 서버 또는 프런트 엔드 구성 요소가 포함 된

예를 들어 하나 이상의 클라이언트 컴퓨터는

그래픽 사용자 인터페이스 또는 웹 브라우저를 통해

사용자는 주제의 구현과 상호 작용할 수 있습니다

본 명세서에 기술되거나, 이러한 백엔드의 임의의 조합,

미들웨어 또는 프론트 엔드 구성 요소. 클라이언트와 서버는

일반적으로, 배타적이지는 않지만 서로 멀리 떨어져 있고

일반적으로 통신 네트워크를 통해 상호 작용하고

시스템의 구성 요소를 서로 연결할 수 있지만

모든 형태 또는 매체의 디지털 데이터 통신.

통신 네트워크의 예로는 다음이 포함됩니다.

근거리 통신망 (“LAN”), 광역 통신망으로 제한

( "WAN") 및 인터넷. 고객과의 관계

서버는 컴퓨터에서 실행되는 컴퓨터 프로그램으로 인해 발생합니다.

각각의 컴퓨터와 클라이언트-서버 관계

서로에게.

전술 한 설명에서 설명 된 구현들

Sub와 일치하는 모든 구현을 나타내지는 않습니다.

본 명세서에 기술 된 주제. 대신, 그들은 단지 일부

설명과 관련된 측면과 일치하는 예

소재. 몇 가지 변형이 설명되었지만

본 명세서에서 상세하게, 다른 수정 또는 추가가 가능하다.

특히, 추가 특징 및 / 또는 변형이 프로

본 명세서에 기술 된 것들에 추가하여 제시된다. 예를 들어

위에서 설명한 구현은 다양한

개시된 특징의 조합 및 하위 조합

및 / 또는 하나 이상의 조합 및 하위 조합

본 명세서에 개시된 것들에 더 특징이있다. 또한

첨부 도면 및 / 또는에 도시 된 논리 흐름

본 명세서에 기술 된 것은 반드시 특정을 요구하지는 않는다

바람직한 결과를 달성하기 위해 도시 된 순서, 또는 순차적 순서.

다음의 청구 범위의 범주는 다른 단순한 것을 포함 할 수있다

멘션 또는 실시 예.

주장하는 것은 :

다음을 포함하는 방법 :

적어도 하나의 객체의 적어도 하나의 판독 동작을 개시하는 것

객체 그래프-계층을 특징 짓는 객체 그래프

인 메모리 데이터베이스 내에 저장된 객체의 아키

루트 객체, 노드의 적어도 일부를 포함하는

해당 참조 카운트를 지정하여

연결된 객체를 가리키는 가장자리 수

여기서 객체를 호환 가능하게 변경할 수 있습니다.

그래프와 호환되지 않는 변경 사항이있을 수 있습니다.

그래프의 객체;

대응하는 각각의 판독 동작에 대해, a

루트 객체의 참조 횟수,

초기에 앵커 포인터를 반영하는 루트 객체

루트 객체를 가리키는 것;

하나 이상의 객체를 변경하는 단계;

11

루트 객체, 하나 이상의 변경된 객체 및

객체의 계층 구조 내에서 객체를

하나 이상의 호환되지 않는 변경 사항이 있습니다.

호환되지 않는 객체를 수용하기 위해 변경된 객체

복제, 사이에 새로운 가장자리를 만드는 복제

영향을받는 개체와 새로 복제 된 개체 사이의 새로운 가장자리

이전 버전의 객체 및 원래 연결된 객체

적어도 하나의 변경된 객체 중;

앵커 포인터를 복제 된 루트 객체에 링크하는 단계;

에 대한 참조 카운트 때 루트 개체를 삭제

루트 객체는 0입니다.

루트 객체에서 시작하여 객체 그래프를 탐색하고

적어도 하나의 변경된 객체로 끝나서

그러한 물체의 가장자리; 과

해당 참조가있을 때 각 객체를 삭제

카운트가 0이됩니다.

여기서 :

데이터베이스는 단조롭게 행으로 데이터를 저장합니다.

행 식별자 증가;

데이터베이스는 다중 버전 동시성 제어를 위해

트랜잭션 격리 및 일관된 읽기

레코드의 새 버전이 작성되면 새 버전

레코드 버전에 새 행 식별자가 제공되고

이전 버전의 레코드는 병렬로 유지됩니다.

가비지 수집 프로 중에는 독자가 청소

커밋 후 cess.

제 1 항에있어서, 상기 기준 카운트는

둘 이상의 CPU에 걸쳐 스트라이프됩니다.

3. 비 일시적 컴퓨터 프로그램 제품 저장

적어도 하나의 프로그램에 의해 실행될 때

가능한 프로세서, 적어도 하나의 프로그램 가능한 프로

다음을 포함하는 작업 수행 실패 :

테이블의 하나 이상의 페이지에 대한 읽기 조작 시작

인 메모리 데이터베이스에 저장된 읽기 작업

첫 번째 테이블 헤더를 가리키는 앵커 객체 사용

적어도 하나의 페이지, 첫 번째 테이블에 액세스하는 객체

첫 번째 링크 된 오브젝트에 대한 링크를 포함하는 헤더 오브젝트

복수의 핸들을 포함하는 제 1 링크 된 객체

적어도 하나의 페이지를 포함하는 페이지들;

연결된 개체와 테이블 헤더 개체 복제

현재 읽기 작업에서 복제 된 링크

복수의 페이지에 대한 핸들을 포함하는 객체

적어도 하나의 페이지를 포함하는 단계;

복제 된 테이블 헤더를 가리 키도록 앵커 오브젝트 설정

목적;

하나 이상의 추가 페이지를 복제 된 링크에 링크

목적; 과

첫 번째 테이블 헤더 객체와 첫 번째 링크를 삭제

읽기 작업이 완료된 후의 객체

여기서 :

데이터베이스는 단조롭게 행으로 데이터를 저장합니다.

행 식별자 증가;

데이터베이스는 다중 버전 동시성 제어를 위해

트랜잭션 격리 및 일관된 읽기

레코드의 새 버전이 작성되면 새 버전

레코드 버전에 새 행 식별자가 제공되고

이전 버전의 레코드는 병렬로 유지됩니다.

가비지 수집 프로 중에는 독자가 청소

커밋 후 cess.

제 3 항에있어서, 상기 컴퓨터 프로그램 제품은

첫 번째 연결된 개체는 참조를 포함하는 버전이 지정된 개체입니다.

카운트, 인 메모리 데이터베이스를 강제하는 참조 카운트

첫 번째 테이블 헤더 개체와 첫 번째 연결된 개체 유지

읽기 작업 중

제 4 항에있어서, 상기 컴퓨터 프로그램 제품은

참조 카운트는 둘 이상의 CPU에서 스트라이핑됩니다.

12

제 3 항에있어서, 상기 컴퓨터 프로그램 제품은

복제 된 테이블 헤더 오브젝트는 처음에 첫 번째 링크에 링크됩니다

목적.

제 3 항에있어서, 상기 컴퓨터 프로그램 제품은

첫 번째 테이블 헤더 개체에는 메타 데이터 특성이 연결되어 있습니다.

해당 페이지를 호출하십시오.

제 3 항에있어서, 상기 컴퓨터 프로그램 제품은

복제 된 테이블 헤더 오브젝트는 메타와도 연관되어 있습니다.

첫 번째 테이블 헤더 객체와 관련된 데이터

9. 비 일시적 컴퓨터 프로그램 제품 저장

적어도 하나의 프로그램에 의해 실행될 때

가능한 프로세서, 적어도 하나의 프로그램 가능한 프로

다음을 포함하는 작업 수행 실패 :

적어도 하나의 객체의 적어도 하나의 판독 동작을 개시하는 것

객체 그래프-계층을 특징 짓는 객체 그래프

메모리 내 데이터베이스에 저장된 오브젝트

루트 객체, 노드의 적어도 일부를 포함

해당 참조 카운트를 지정하여

연결된 객체를 가리키는 가장자리 수

여기서 객체를 호환 가능하게 변경할 수 있습니다.

그래프와 호환되지 않는 변경 사항이있을 수 있습니다.

그래프의 객체;

대응하는 각각의 판독 동작에 대해, a

해당 객체의 참조 횟수, 참조

따라서 처음에 루트 오브젝트를 반영하는 루트 오브젝트의 수

루트 객체를 가리키는 앵커 포인터;

하나 이상의 객체를 변경하는 단계;

루트 객체, 하나 이상의 변경된 객체 및

객체의 계층 구조 내에서 객체를

하나 이상의 호환되지 않는 변경 사항이 있습니다.

호환되지 않는 객체를 수용하기 위해 변경된 객체

복제, 사이에 새로운 가장자리를 만드는 복제

영향을받는 개체와 새로 복제 된 개체 사이의 새로운 가장자리

이전 버전의 객체 및 원래 연결된 객체

적어도 하나의 변경된 객체 중;

앵커 포인터를 복제 된 루트 객체에 링크하는 단계;

에 대한 참조 카운트 때 루트 개체를 삭제

루트 객체는 0입니다.

루트 객체에서 시작하여 객체 그래프를 탐색하고

적어도 하나의 변경된 객체로 끝나서

그러한 물체의 가장자리; 과

해당 참조가있을 때 각 객체를 삭제

카운트가 0이됩니다.

여기서 :

데이터베이스는 단조롭게 행으로 데이터를 저장합니다.

행 식별자 증가;

데이터베이스는 다중 버전 동시성 제어를 위해

트랜잭션 격리 및 일관된 읽기

레코드의 새 버전이 작성되면 새 버전

레코드 버전에 새 행 식별자가 제공되고

이전 버전의 레코드는 병렬로 유지됩니다.

가비지 수집 프로 중에는 독자가 청소

커밋 후 cess.

제 9 항에있어서, 상기 컴퓨터 프로그램 제품은

참조 카운트는 둘 이상의 CPU에서 스트라이핑됩니다.

제 9 항에있어서, 상기 컴퓨터 프로그램 제품은

읽기 작업이 호환되지 않는 변경으로 구성되어 있습니다.

12. 다음을 포함하는 시스템 :

적어도 하나의 프로그램 가능 프로세서

메모리 저장 명령.

적어도 하나의 프로그램 가능 프로세서

다음을 포함하는

테이블의 하나 이상의 페이지에 대한 읽기 조작 시작

인 메모리 데이터베이스에 저장된 읽기 작업

첫 번째 테이블 헤더를 가리키는 앵커 객체 사용

적어도 하나의 페이지, 첫 번째 테이블에 액세스하는 객체

첫 번째 링크 된 오브젝트에 대한 링크를 포함하는 헤더 오브젝트

13

복수의 핸들을 포함하는 제 1 링크 된 객체

적어도 하나의 페이지를 포함하는 페이지의 시티;

연결된 개체와 테이블 헤더 개체 복제

읽기 작업과 동시에 복제

복수의 손잡이를 포함하는 연결된 물체 5

적어도 하나의 페이지를 포함하는 페이지들;

복제 된 테이블을 가리 키도록 앵커 오브젝트 설정

헤더 객체;

하나 이상의 추가 페이지를 복제 된 링크에 링크

목적; 과

첫 번째 테이블 헤더 객체와 첫 번째 링크를 삭제

읽기 작업이 완료된 후의 객체

여기서 :

데이터베이스는 단조롭게 행으로 데이터를 저장합니다.

행 식별자 증가;

데이터베이스는 다중 버전 동시성 제어를 위해

트랜잭션 격리 및 일관된 읽기

레코드의 새 버전이 작성되면 새 버전

레코드 버전에 새 행 식별자가 제공되고

이전 버전의 레코드는 병렬로 유지됩니다.

가비지 수집 프로 중에는 독자가 청소

커밋 후 cess.

제 12 항에있어서, 상기 제 1 링크 된 객체

참조 카운트를 포함하는 버전이 지정된 객체입니다.

따라서 메모리 내 데이터베이스를 강제로 유지하도록 카운트

첫 번째 테이블 헤더 개체와 첫 번째 연결된 개체

읽기 작업.

제 13 항에있어서, 상기 기준 카운트는

둘 이상의 CPU에 스트라이프됩니다.

제 12 항에있어서, 상기 복제 된 테이블

헤더 오브젝트는 처음에 첫 번째 링크 된 오브젝트에 링크됩니다.

제 12 항에있어서, 상기 제 1 테이블 헤더

객체는 상관 관계를 특징 짓는 관련된 메타 데이터를 가진다

스 펀딩 페이지.

제 12 항에있어서, 상기 복제 된 테이블

헤더 객체는 관련된 메타 데이터와도 연결됩니다

첫 번째 테이블 헤더 객체와 함께.

18. 다음을 포함하는 시스템 :

적어도 하나의 프로그램 가능 프로세서

메모리 저장 명령.

적어도 하나의 프로그램 가능 프로세서

다음을 포함하는

적어도 하나의 객체의 적어도 하나의 판독 동작을 개시하는 것

객체 그래프의 특징은

인 메모리 데이터베이스에 저장된 객체의 계층

14

루트 객체를 포함하며

대응하는 참조 카운트를 갖는 노드

관련을 가리키는 많은 모서리를 튀기다

호환 가능한 변경을 수행 할 수있는

그래프의 개체 및 호환되지 않는 변경은

그래프의 객체로 만들어졌습니다.

대응하는 각각의 판독 동작에 대해, a

해당 객체의 참조 카운트, 참조

처음에 루트 오브젝트를 반영하는 루트 오브젝트의 수

루트 객체를 가리키는 앵커 포인터;

하나 이상의 객체를 변경하는 단계;

루트 개체, 하나 이상의 변경된 개체 복제

계층 구조 내의 모든 중간 객체

호환되지 않는 변경이 이루어진 경우

그러한 것을 수용하기 위해 적어도 하나의 변경된 객체

호환되지 않는 변경 사항, 복제로 새로운 가장자리 생성

영향을받는 개체와 새 가장자리 사이

새로 복제 된 객체 및 원래 연결된 객체

적어도 하나의 변경된 객체의 구 버전으로부터;

앵커 포인터를 복제 된 루트 객체에 링크하는 단계;

에 대한 참조 카운트 때 루트 개체를 삭제

루트 객체는 0입니다.

루트 객체에서 시작하여 객체 그래프를 탐색하고

적어도 하나의 변경된 객체 제거에서 종료

그러한 물체의 모서리; 과

해당 참조가있을 때 각 객체를 삭제

카운트가 0이됩니다.

여기서 :

데이터베이스는 단조롭게 행으로 데이터를 저장합니다.

행 식별자 증가 :

데이터베이스는 다중 버전 동시성 제어를 위해

트랜잭션 격리 및 일관된 읽기

레코드의 새 버전이 작성되면 새 버전

레코드 버전에 새 행 식별자가 제공되고

이전 버전의 레코드는 병렬로 유지됩니다.

가비지 수집 프로 중에는 독자가 청소

커밋 후 cess.

제 18 항에있어서, 상기 기준 카운트는

둘 이상의 CPU에 스트라이프됩니다.

제 18 항에있어서, 상기 판독 동작

호환되지 않는 변경으로 구성됩니다.