(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2024년03월12일
(11) 등록번호 10-2645141
(24) 등록일자 2024년03월07일

(51) 국제특허분류(Int. Cl.)
G06F 16/13 (2019.01) *G06F 16/14* (2019.01)
G06F 16/22 (2019.01) *G06F 16/31* (2019.01)
G06F 16/901 (2019.01)
(52) CPC특허분류
G06F 16/137 (2019.01)
G06F 16/152 (2019.01)
(21) 출원번호 10-2020-0068315
(22) 출원일자 2020년06월05일
심사청구일자 2023년06월05일
(65) 공개번호 10-2021-0034471
(43) 공개일자 2021년03월30일
(30) 우선권주장
62/903,637 2019년09월20일 미국(US)
16/837,730 2020년04월01일 미국(US)
(56) 선행기술조사문헌
JP2014510318 A
US20190034427 A1
US20170250708 A1

(73) 특허권자
삼성전자주식회사
경기도 수원시 영통구 삼성로 129 (매탄동)
(72) 발명자
박희권
95014 미국 캘리포니아주 쿠퍼티노시 스티븐스 크릭 블러바드 20128, 유닛210
홍일구
95054 미국 캘리포니아주 산타 클라라 헤든 웨이 4448
(뒷면에 계속)
(74) 대리인
리앤목특허법인

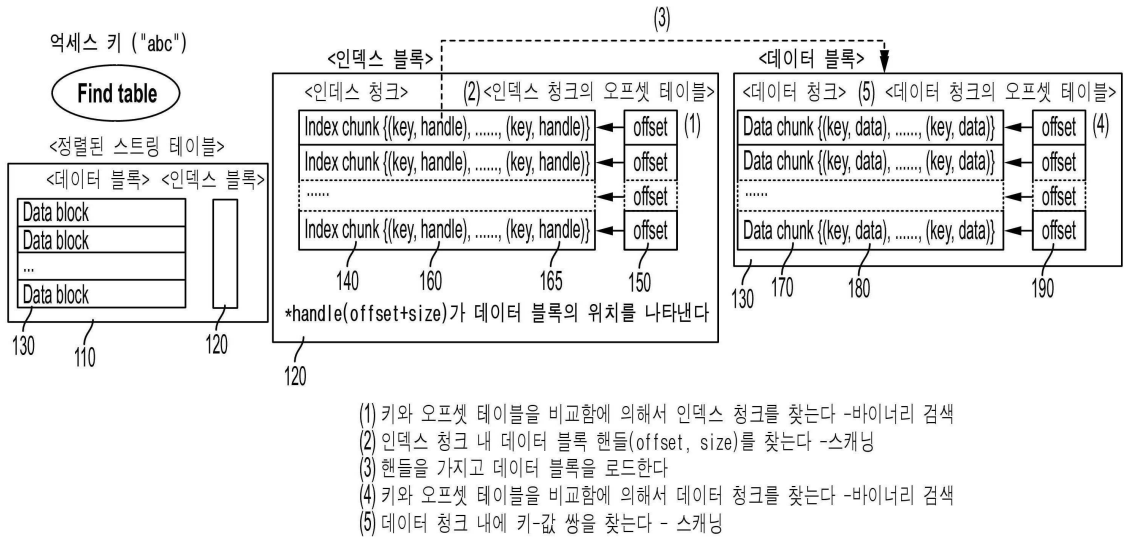
전체 청구항 수 : 총 21 항 심사관 : 박미정

(54) 발명의 명칭 **데이터베이스 내 타겟 키를 검색하는 방법, 시스템, 및 비-휘발성 컴퓨터 판독가능한 매체**

(57) 요 약

데이터베이스 내 타겟 키를 검색하는 시스템 및 방법이 개시된다. 방법은 정렬된 키 테이블의 해시-오프셋 테이블을 해시-오프셋 테이블 엔트리들로 채우고 - 상기 해시-오프셋 테이블 엔트리들은 각 키에 대응하는 해시-값, 및 해시 오프셋을 가짐 -; 상기 해시-값들에 기초하여 상기 해시-오프셋 테이블 엔트리들을 정렬하고; 상기 해시-오프셋 테이블 내 타겟 키에 대응하는 상기 해시-값들의 타겟 해시-값을 검색하고; 상기 타겟 해시-값에 기초하여 타겟 키에 대응하는 타겟 키-값 쌍을 찾고; 및 상기 타겟 키-값 쌍의 위치를 저장하는 것을 포함할 수 있다.

대 표 도



(1) 키와 오프셋 테이블을 비교함에 의해서 인덱스 청크를 찾는다 -바이너리 검색
(2) 인덱스 청크 내 데이터 블록 핸들(offset, size)를 찾는다 -스캐닝
(3) 핸들을 가지고 데이터 블록을 로드한다
(4) 키와 오프셋 테이블을 비교함에 의해서 데이터 청크를 찾는다 -바이너리 검색
(5) 데이터 청크 내에 키-값 쌍을 찾는다 - 스캐닝

(52) CPC특허분류
G06F 16/2255 (2019.01)
G06F 16/325 (2019.01)
G06F 16/9014 (2019.01)
(72) 발명자
이호빈
95117 미국 캘리포니아주 산호세 월턴 웨이 3345
기양석
94303 미국 캘리포니아주 팔로 알토, 알테어 워크 873

명 세 서

청구범위

청구항 1

시스템을 통해 데이터베이스 내 타겟 키를 검색하는 방법에 있어서,

상기 시스템이 스토리지 디바이스에서 정렬된 키 테이블의 해시-오프셋 테이블을 해시-오프셋 테이블 엔트리들로 채우는 단계 - 상기 해시-오프셋 테이블 엔트리들은 각 키에 대응하는 해시-값, 및 해시 오프셋을 가짐 -;

상기 시스템이 상기 해시-값들에 기초하여 상기 해시-오프셋 테이블 엔트리들을 정렬하는 단계;

상기 시스템이 상기 해시-오프셋 테이블 내 타겟 키에 대응하는 상기 해시-값들의 타겟 해시-값을 검색하는 단계;

상기 시스템이 상기 타겟 해시-값에 기초하여 상기 타겟 키에 대응하는 타겟 키-값 쌍을 찾는 단계; 및

상기 시스템이 스토리지 포맷 데이터 구조에 상기 타겟 키-값 쌍의 위치를 저장하는 단계를 포함하는,

데이터베이스 내 타겟 키를 검색하는 방법.

청구항 2

제1 항에 있어서,

상기 타겟 해시-값을 검색하는 단계는 바이너리 검색을 수행하는 단계를 포함하는 방법.

청구항 3

제1 항에 있어서,

상기 방법은 상기 타겟 키로부터 상기 타겟 해시-값을 계산하는 단계를 더 포함하는 방법.

청구항 4

제1 항에 있어서,

상기 타겟 키-값 쌍의 위치를 저장하는 단계는 상기 타겟 해시-값을 상기 타겟 키-값 쌍에 매핑하는 단계를 포함하는 방법.

청구항 5

제1 항에 있어서,

상기 타겟 키-값 쌍을 찾는 단계는 상기 해시 오프셋에 기초하여 상기 타겟 키-값 쌍을 찾는 단계를 포함하는 방법.

청구항 6

제1 항에 있어서,

상기 정렬된 키 테이블은 복수의 키-값 테이블 엔트리들을 포함하는 키-값 테이블을 추가적으로 포함하고, 상기 키-값 테이블 엔트리들은 상기 타겟 키-값 쌍을 포함하는 방법.

청구항 7

제6 항에 있어서,

상기 키-값 테이블 엔트리들의 수는 상기 해시-오프셋 테이블 엔트리들의 수와 동일한 방법.

청구항 8

제1항에 있어서,

상기 해시-값의 크기는 상기 해시-값에 대응되는 키의 크기보다 작은 방법.

청구항 9

온-메모리 데이터 구조, 스토리지 포맷 데이터 구조, 및 스토리지 디바이스를 포함하고,

상기 스토리지 디바이스의 상기 스토리지 포맷 데이터 구조 내에서 정렬된 키 테이블의 해시-오프셋 테이블을 해시-오프셋 테이블 엔트리들로 채우고 - 상기 해시-오프셋 테이블 엔트리들은 각 키에 대응하는 해시-값, 및 해시 오프셋을 가짐 -;

상기 온-메모리 데이터 구조의 온-메모리 정렬 구조에 따라 상기 해시-값들에 기초하여 상기 해시-오프셋 테이블 엔트리들을 정렬하고;

상기 해시-오프셋 테이블 내 타겟 키에 대응하는 상기 해시-값들의 타겟 해시-값에 대하여 상기 스토리지 포맷 데이터 구조를 검색하고;

상기 스토리지 포맷 데이터 구조 내에서 상기 타겟 해시-값에 기초하여 타겟 키에 대응하는 타겟 키-값 쌍을 찾고; 및

상기 스토리지 포맷 데이터 구조에 상기 타겟 키-값 쌍의 위치를 저장하는 것을 포함하는,

데이터베이스 내 타겟 키를 검색하도록 구성된 시스템.

청구항 10

제9 항에 있어서,

상기 시스템은 바이너리 검색을 수행함에 의해서 상기 타겟 해시-값을 검색하도록 구성된 시스템.

청구항 11

제9 항에 있어서,

상기 시스템은 상기 타겟 키로부터 상기 타겟 해시-값을 계산하도록 구성된 시스템.

청구항 12

제9 항에 있어서,

상기 시스템은 상기 타겟 해시-값을 상기 타겟 키-값 쌍에 매핑함에 의해서 상기 타겟 키-값 쌍의 위치를 저장하도록 구성된 시스템.

청구항 13

제9 항에 있어서,

상기 시스템은 상기 해시 오프셋에 기초하여 상기 타겟 키-값 쌍을 찾도록 구성된 시스템.

청구항 14

제9 항에 있어서,

상기 정렬된 키 테이블은 복수의 키-값 테이블 엔트리들을 포함하는 키-값 테이블을 추가적으로 포함하고, 상기 키-값 테이블 엔트리들은 상기 타겟 키-값 쌍을 포함하는 시스템.

청구항 15

컴퓨터 실행 가능 컴퓨터 코드를 포함하는 비일시적 컴퓨터 판독가능한 매체에 있어서, 상기 컴퓨터 코드가 시스템에 포함된 프로세서에 의해 실행될 때,

상기 시스템이 스토리지 디바이스에서 정렬된 키 테이블의 해시-오프셋 테이블을 해시-오프셋 테이블 엔트리들로 채우는 단계 - 상기 해시-오프셋 테이블 엔트리들은 각 키에 대응하는 해시-값, 및 해시 오프셋을 가짐 -;

상기 시스템이 상기 해시-값들에 기초하여 상기 해시-오프셋 테이블 엔트리들을 정렬하는 단계;

상기 시스템이 상기 해시-오프셋 테이블 내 타겟 키에 대응하는 상기 해시-값들의 타겟 해시-값을 검색하는 단계;

상기 시스템이 상기 타겟 해시-값에 기초하여 타겟 키에 대응하는 타겟 키-값 쌍을 찾는 단계; 및

상기 시스템이 스토리지 포맷 데이터 구조에 상기 타겟 키-값 쌍의 위치를 저장하는 단계를 포함하는 데이터베이스 내에 타겟 키를 검색하는 방법을 수행하는 것을 특징으로 하는 컴퓨터 실행 가능 컴퓨터 코드를 포함하는 비일시적 컴퓨터 판독가능한 매체.

청구항 16

제15 항에 있어서,

상기 컴퓨터 코드는 바이너리 검색을 수행함에 의해서 상기 타겟 해시-값을 검색하는 단계를 추가적으로 구현하는 비일시적 컴퓨터 판독가능한 매체.

청구항 17

제15 항에 있어서,

상기 컴퓨터 코드는 상기 타겟 키로부터 상기 타겟 해시-값을 계산하는 단계를 추가적으로 구현하는 비일시적 컴퓨터 판독가능한 매체.

청구항 18

제15 항에 있어서,

상기 타겟 키-값 쌍의 위치를 저장하는 단계는 상기 타겟 해시-값을 상기 타겟 키-값 쌍에 매핑하는 단계를 포함하는 비일시적 컴퓨터 판독가능한 매체.

청구항 19

제15 항에 있어서,

상기 타겟 키-값 쌍을 찾는 단계는 상기 해시 오프셋에 기초하여 상기 타겟 키-값 쌍을 찾는 단계를 포함하는 비일시적 컴퓨터 판독가능한 매체.

청구항 20

제15 항에 있어서,

상기 정렬된 키 테이블은 복수의 키-값 테이블 엔트리들을 포함하는 키-값 테이블을 추가적으로 포함하고, 상기 키-값 테이블 엔트리들은 상기 타겟 키-값 쌍을 포함하는 비일시적 컴퓨터 판독가능한 매체.

청구항 21

제20 항에 있어서,

상기 키-값 테이블 엔트리들의 수는 상기 해시-오프셋 테이블 엔트리들의 수와 동일한 비일시적 컴퓨터 판독가능한 매체.

발명의 설명

기 술 분 야

[0001] 본 개시의 하나 이상의 실시예들은 데이터베이스 내 타겟 키를 검색하는 방법, 시스템 및 비-휘발성 컴퓨터 판독가능한 매체에 관한 것이다.

배 경 기 술

[0002] 키-값(KV: key-value) 스토어에서 테이블 내 랜덤 키를 검색하고 복구하는 것은 심각한 오버헤드를 초래할 수

있다. 이러한 오버헤드는 일반적으로 복수의 키 비교들을 이용하는 검색에 기인한다. 따라서, 일부 키 검색 기술들은 일반적으로 다양한 키 비교와 관련된 많은 수의 CPU 사이클을 사용할 수 있다. 따라서, 테이블의 데이터 구조를 더 효율적이고 효과적으로 정의하는 것이 유용할 수 있다.

발명의 내용

해결하려는 과제

[0003] 본 개시의 실시예들에 따른 과제는 해시-오프셋 테이블을 키-값 테이블에 매핑함에 의해서 데이터 스토리지를 개선하기 위한 데이터베이스 내 타겟 키를 검색하는 방법, 시스템, 및 비-휘발성 컴퓨터 판독가능한 매체를 제공하는데 있다.

과제의 해결 수단

[0004] 본 개시의 일 실시예에 따르면, 데이터베이스 내 타겟 키를 검색하는 방법은 정렬된 키 테이블의 해시-오프셋 테이블을 해시-오프셋 테이블 엔트리들로 채우고 - 상기 해시-오프셋 테이블 엔트리들은 각 키에 대응하는 해시-값, 및 해시 오프셋을 가짐 -; 상기 해시-값들에 기초하여 상기 해시-오프셋 테이블 엔트리들을 정렬하고; 상기 해시-오프셋 테이블 내 타겟 키에 대응하는 상기 해시-값들의 타겟 해시-값을 검색하고; 상기 타겟 해시-값에 기초하여 타겟 키에 대응하는 타겟 키-값 쌍을 찾고; 및 상기 타겟 키-값 쌍의 위치를 저장하는 것을 포함할 수 있다.

[0005] 상기 타겟 해시-값을 검색하는 것은 바이너리 검색을 수행하는 것을 포함할 수 있다.

[0006] 상기 방법은 상기 타겟 키로부터 상기 타겟 해시-값을 계산하는 것을 추가적으로 포함할 수 있다.

[0007] 상기 타겟 키-값 쌍의 위치를 저장하는 것은 상기 타겟 해시-값을 상기 타겟 키-값 쌍에 매핑하는 것을 포함할 수 있다.

[0008] 상기 타겟 키-값 쌍을 찾는 것은 상기 해시 오프셋에 기초하여 상기 타겟 키-값 쌍을 찾는 것을 포함할 수 있다.

[0009] 상기 정렬된 키 테이블은 복수개의 키-값 테이블 엔트리들을 포함하는 키-값 테이블을 추가적으로 포함하고, 상기 키-값 테이블 엔트리들은 상기 타겟 키-값 쌍을 포함할 수 있다.

[0010] 상기 키-값 테이블 엔트리들의 수는 상기 해시-오프셋 테이블 엔트리들의 수와 동일할 수 있다.

[0011] 본 개시의 일 실시예에 따르면, 데이터베이스 내 타겟 키를 검색하는 시스템은 온-메모리 데이터 구조, 스토리지 포맷 데이터 구조, 및 스토리지 디바이스를 포함하고, 상기 스토리지 포맷 데이터 구조 내에서 정렬된 키 테이블의 해시-오프셋 테이블을 해시-오프셋 테이블 엔트리들로 채우고 - 상기 해시-오프셋 테이블 엔트리들은 각 키에 대응하는 해시-값, 및 해시 오프셋을 가짐 -; 상기 온-메모리 데이터 구조의 온-메모리 정렬 구조에 따라 상기 해시-값들에 기초하여 상기 해시-오프셋 테이블 엔트리들을 정렬하고; 상기 해시-오프셋 테이블 내 타겟 키에 대응하는 상기 해시-값들의 타겟 해시-값에 대하여 상기 온-메모리 데이터 구조 또는 상기 스토리지 맷 데이터 구조를 검색하고; 상기 온-메모리 데이터 구조 또는 상기 스토리지 포맷 데이터 구조 내에서 상기 타겟 해시-값에 기초하여 타겟 키에 대응하는 타겟 키-값 쌍을 찾고; 및 상기 온-메모리 데이터 구조 또는 상기 스토리지 포맷 데이터 구조에 상기 타겟 키-값 쌍의 위치를 저장하는 것을 포함할 수 있다.

[0012] 상기 시스템은 바이너리 검색을 수행함에 의해서 상기 타겟 해시-값을 검색하는 것을 추가적으로 포함할 수 있다.

[0013] 상기 시스템은 상기 타겟 키로부터 상기 타겟 해시-값을 계산하는 것을 추가적으로 포함할 수 있다.

[0014] 상기 시스템은 상기 타겟 해시-값을 상기 타겟 키-값 쌍에 매핑함에 의해서 상기 타겟 키-값 쌍의 위치를 저장하는 것을 추가적으로 포함할 수 있다.

[0015] 상기 시스템은 상기 해시 오프셋에 기초하여 상기 타겟 키-값 쌍을 찾는 것을 추가적으로 포함할 수 있다.

[0016] 상기 정렬된 키 테이블은 복수개의 키-값 테이블 엔트리들을 포함하는 키-값 테이블을 추가적으로 포함하고, 상기 키-값 테이블 엔트리들은 상기 타겟 키-값 쌍을 포함할 수 있다.

[0017] 본 개시의 일 실시예에 따르면, 데이터베이스 내에 타겟 키를 검색하는 시스템에 구현된 비-일시적 컴퓨터 판독

가능한 매체는 프로세서 상에서 실행될 때, 데이터 스토리지의 방법을 구현하는 컴퓨터 코드를 포함하고, 상기 방법은 정렬된 키 테이블의 해시-오프셋 테이블을 해시-오프셋 테이블 엔트리들로 채우고 - 상기 해시-오프셋 테이블 엔트리들은 각 키에 대응하는 해시-값, 및 해시 오프셋을 가짐 -; 상기 해시-값들에 기초하여 상기 해시-오프셋 테이블 엔트리들을 정렬하고; 상기 해시-오프셋 테이블 내 타겟 키에 대응하는 상기 해시-값들의 타겟 해시-값을 검색하고; 상기 타겟 해시-값에 기초하여 타겟 키에 대응하는 타겟 키-값 쌍을 찾고; 및 상기 타겟 키-값 쌍의 위치를 저장하는 것을 포함할 수 있다.

[0018] 상기 프로세서에 의해서 실행될 때, 상기 컴퓨터 코드는 바이너리 검색을 수행함에 의해서 상기 타겟 해시-값을 검색하는 것을 추가적으로 구현할 수 있다.

[0019] 상기 프로세서에 의해서 실행될 때, 상기 컴퓨터 코드는 상기 타겟 키로부터 상기 타겟 해시-값을 계산하는 것을 추가적으로 구현할 수 있다.

[0020] 상기 타겟 키-값 쌍의 위치를 저장하는 것은 상기 타겟 해시-값을 상기 타겟 키-값 쌍에 매핑하는 것을 포함할 수 있다.

[0021] 상기 타겟 키-값 쌍을 찾는 것은 상기 해시 오프셋에 기초하여 상기 타겟 키-값 쌍을 찾는 것을 포함할 수 있다.

[0022] 상기 정렬된 키 테이블은 복수개의 키-값 테이블 엔트리들을 포함하는 키-값 테이블을 추가적으로 포함하고, 상기 키-값 테이블 엔트리들은 상기 타겟 키-값 쌍을 포함할 수 있다.

[0023] 상기 키-값 테이블 엔트리들의 수는 상기 해시-오프셋 테이블 엔트리들의 수와 동일할 수 있다.

발명의 효과

[0024] 본 개시의 실시예들에 따르면, 해시-오프셋 테이블을 키-값 테이블의 엔트리들에 매핑하여 데이터의 개선된 검색 및 탐색을 위한 시스템 및 방법을 제공함에 의해서 데이터 스토리지를 개선할 수 있다.

[0025] 또한, 본 개시의 실시예들은 키-값(KV: key-value) 데이터베이스 시스템에 대한 빠른 단일 키 쿼리를 가능하게 하여, 이의 검색 계층을 단순화함에 의해서 정렬된 키 테이블 내 단일 키의 비교적 빠른 키 위치를 찾는 것을 가능하게 하고, 비교 오버헤드의 감소를 가능하게 하는 스토리지 데이터 포맷을 제공하고, 이에 따라 단일 키 리드 성능을 개선할 수 있다.

도면의 간단한 설명

[0026] 본 실시예들의 비-제한적인 실시예들은 첨부된 도면을 참조하여 설명되며, 달리 명시되지 않는 한, 유사한 도면 부호는 다양한 도면들 전체에서 유사한 부분을 지칭한다.

도 1은 정렬된 스트링 테이블의 검색 계층의 예를 나타내는 블록도이다.

도 2는 본 개시의 실시예들에 따른 키-값 쌍을 검색하는 방법을 나타내는 블록도이다.

도 3은 본 개시의 실시예들에 따른 키-값 쌍을 검색하는 방법을 나타내는 플로우차트이다.

도 4는 본 개시의 실시예에 따른 키-값 쌍을 검색하는 시스템을 나타내는 블록도이다.

대응하는 참조 부호는 여러 도면에 걸쳐 대응하는 구성요소를 나타낸다. 당업자는 도면의 요소들이 단순성 및 명료성을 위하여 도시되었으며, 크기에 맞게 그려지지는 않았음을 알 수 있을 것이다. 예를 들면, 도면들에서 일부 구성요소들, 층들, 및 영역들의 면적은 명료성 및 다양한 실시예들의 이해를 돕기 위하여 다른 구성요소들, 층들, 및 영역들에 비해 과장될 수 있다. 또한, 실시예들의 설명에 관련되지 않는 일반적이나 잘 이해된 구성 요소들 및 부분들은 다양한 실시예들에 대한 덜 방해받는 도면을 용이하게 하고 설명을 명확하게 하기 위하여 도시되지 않을 수 있다.

발명을 실시하기 위한 구체적인 내용

[0027] 본 개시의 특징들 및 이를 달성하기 위한 방법들이 실시예들의 상세한 설명 및 첨부된 도면들을 참조하여 더 쉽게 이해될 수 있다. 이하에서는 실시예들이 첨부된 도면들을 참조하면 더 상세하게 설명될 것이다. 그러나, 기술된 실시예들은 다양한 다른 형태들로 구현될 수 있고 본 명세서에서 개시된 실시예들에 한정되는 것으로 해석되어서는 안 된다. 오히려, 이러한 실시예들은 본 개시가 철저하고 완전해지도록 제공되고, 당업자에게 본 개시

의 양태들 및 특징들을 충분히 전달할 것이다. 따라서, 본 개시의 양태들 및 특징들의 완전한 이해를 위하여 당업자에게 필요하지 않은 프로세스들, 요소들 및 기술들은 설명되지 않을 수 있다.

[0028] 상세한 설명에서, 설명을 위하여, 다수의 특정 상세한 설명은 다양한 실시예들의 철저한 이해를 제공하기 위하여 제시된다. 그러나, 이러한 특정 상세한 설명 없이 또는 하나 이상의 동등한 배열을 가지고 실시될 수 있음은 명백하다. 다른 경우에, 잘 알려진 구조들 및 장치들은 다양한 실시예들을 불필요하게 모호하게 하는 것을 피하기 위하여 블록도 형태로 표시된다.

[0029] "제1", "제2", "제3" 등의 용어들이 다양한 구성요소들, 부분품, 영역들, 층들 및/또는 섹션들을 설명하기 위하여 본 명세서에서 사용될 수 있더라도 이러한 구성요소들, 부분품들, 영역들, 층들 및/또는 섹션들은 이러한 용어들에 의해서 한정되어서는 안된다는 것을 이해할 것이다. 이러한 용어들은 단지 하나의 구성요소, 부분품, 영역, 층 또는 섹션을 다른 하나의 구성요소, 부분품, 영역, 층 또는 섹션과 구별하기 위하여 사용된다. 따라서, 아래에 기술된 제1 구성요소, 부분품, 영역, 층 또는 섹션은 본 개시의 사상 및 범위로부터 벗어나지 않으면서 제2 구성요소, 부분품, 영역, 층 또는 섹션으로 지칭될 수 있다.

[0030] 여기에 사용된 용어들은 단지 특정한 실시예를 설명하기 위해 사용된 것으로, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 출원에서, "포함하다" 또는 "가지다" 등의 용어는 개시된 특징들, 숫자들, 단계들, 동작들, 구성요소들, 부분품들 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들, 숫자들, 단계들, 동작들, 구성요소들, 부분품들 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다. 여기에 사용된 용어 "및/또는"은 하나 이상의 관련된 열거된 항목들의 임의의 및 모든 조합을 포함한다.

[0031] 여기에 사용된 용어 "실질적으로", "약", "대략"및 유사한 용어들은 정도의 용어가 아니라 근사치의 용어로 사용되며, 당업자에 의해서 인식되는 측정되거나 계산된 값의 고유 편차를 설명하기 위한 의도이다. 여기에 사용된 "약" 또는 "대략"은 특정 양의 측정과 관련된 오차(즉, 측정 시스템의 한계)를 고려하여, 당업자에 의해서 결정된 값으로서 명시된 값을 포함하며, 특정 값의 허용가능한 편차 범위 내를 의미한다. 예를 들면, "약"은 하나 이상의 표준 편차 내 또는 명시된 값의 ± 30%, 20%, 10%, 5% 내를 의미한다. 추가적으로, 본 개시의 실시예를 설명할 때"할 수 있다(may)"의 사용은 본 개시의 하나 이상의 실시예를 지칭한다.

[0032] 어떠한 실시예가 다르게 구현될 때, 특정 프로세스 순서는 설명된 순서와 다르게 수행될 수 있다. 예를 들면, 2개의 연속적으로 기술된 프로세스는 실질적으로 동시에 또는 설명된 순서와 반대 순서로 수행될 수 있다.

[0033] 여기에 기술된 본 개시의 실시예들에 따른 전자 또는 전기 장치들 및/또는 다른 관련 장치들 또는 부분품들은 임의의 적절한 하드웨어, 펌웨어(예를 들면, 특정 용도 집적 회로(ASIC: application-specific integrated circuit), 소프트웨어, 또는 소프트웨어, 펌웨어 및 하드웨어의 조합을 이용하여 구현될 수 있다. 예를 들면, 이들 장치들의 다양한 부분품들은 집적회로(IC) 칩 또는 별도의 IC 칩 상에 형성될 수 있다. 또한, 이들 장치들의 다양한 부분품들은 플렉시블(flexible) 인쇄 회로 필름, 테이프 캐리어 패키지(TCP: tape carrier package), 인쇄 회로 기판(PCB: printed circuit board) 상에 구현되거나, 하나의 기판 상에 구현될 수 있다.

[0034] 추가적으로, 이들 장치들의 다양한 부분품들은 컴퓨터 프로그램 명령들을 수행하고 여기에 설명된 다양한 기능들을 수행하기 위하여 다른 시스템 부분품들과 상호 동작하는 하나 이상의 컴퓨팅 장치들 내 하나 이상의 프로세서들 상에서 실행하는 프로세스 또는 스레드(thread)일 수 있다. 컴퓨터 프로그램 명령들은 예를 들면, 랜덤 억세스 메모리(RAM: random access memory)와 같은 표준 메모리 장치를 사용하여 컴퓨팅 장치 내에 구현될 수 있는 메모리에 저장된다. 컴퓨터 프로그램 명령들은 또한 예를 들면, CD-ROM, 플래시 드라이버, 등과 같은 다른 비 일시적 컴퓨터 판독 가능 매체(non-transitory computer readable media)에 저장될 수 있다. 또한, 당업자는 본 개시의 실시예들의 사상 및 범위를 벗어나지 않으면서 다양한 컴퓨팅 장치들의 기능이 단일 컴퓨팅 장치에 결합되거나 통합될 수 있거나 특정 컴퓨팅 장치의 기능이 하나 이상의 다른 컴퓨팅 장치들에 분산될 수 있음을 인식해야한다.

[0035] 다르게 정의되지 않는 한, 여기에 사용된 (기술적 및 과학적인 용어들을 포함하는) 모든 용어들은 본 개시가 속하는 기술 분야에서 통상의 지식을 가진 자에 의해서 일반적으로 이해되는 것과 동일한 의미를 가진다. 또한, 일반적으로 사용되는 사전에 정의된 용어와 같은 용어는 관련 기술 및/또는 본 개시의 맥락에서 그 의미와 일치하는 의미를 가지는 것으로 해석되어야 하고, 여기에서 명확하게 정의되지 않는 한 이상적이거나 지나치지 공식적인 의미로 해석되어서는 안된다는 것이 이해될 것이다.

[0036] 본 개시의 실시예들은 (예를 들면, KVRocks과 같은 키-값 솔리드 스테이브 드라이버(KVSSD: key-value solid

state drive)에서) 키-값(KV: key-value) 데이터에 대한 검색을 수행하기 위하여 내부 키 해시 디렉토리 내 검색 계층(hierarchy) 및/또는 알고리즘에 대한 필요성을 감소하거나 제거할 수 있다. 따라서, 개시된 실시예들은 KVSSD 내에 저장된 주어진 테이블 내 랜덤 키를 찾기위하여 사용되는 KVSSD의 프로세서의 CPU 사이클의 수를 감소할 수 있다.

[0037] 개시된 실시예들의 내부 키 해시 디렉토리는 해시-오프셋 테이블 내 해당 키의 해시-값 및 오프셋을 가지는 다양한 엔트리들을 포함한다. 따라서, 내부 키 해시 디렉토리의 KV 테이블 엔트리들은 그들의 각 해시-값에 기초하여 정렬될 수 있다. 따라서, 랜덤 키를 검색할 때, 각 키가 해당 해시-값으로 변화되고 해시-값의 크기가 그것이 대응하는 키의 크기 보다 작을 것이기 때문에 본 개시의 실시예들은 내부 키 해시 디렉토리 내 탐색 타겟 키에 대응하는 해시-값을 검색함에 의해서 오버헤드를 줄일 수 있다.

[0038] 도 1은 정렬 스트링 테이블의 검색 계층의 예를 도시하는 블록도이다.

[0039] 도 1을 참조하면, 정렬된 스트링 테이블(SST: sorted string table)(110)은 키 정보를 저장하는 주요 온-스토리지 데이터 포맷일 수 있다. 또한, FACEBOOK®에 의해서 개발된 RocksDB, 또는 GOOGLE®에 의해서 개발된 LevelDB와 같은 로그-구조 KV 데이터베이스들은 KV 스토리지 내 SST(110) 내부에 비교적 복잡한 검색 계층 또는 검색 알고리즘을 포함한다. 검색 계층은 데이터가 검색을 요청하는 어플리케이션으로 리턴될 수 있도록 타겟 키에 대응하는 데이터를 검색할 때 따르는 일반적인 동작 순서로 간주될 수 있다. SST(110) 내 단일 키 검색은 랜덤 억세스 패턴의 경로일 수 있다는 것을 주목해야한다.

[0040] 예를 들면, KV 쌍(KV pair)(180)을 검색하고 찾기 위하여, 인덱스 블록(index block)들(120) 및 데이터 블록(data block)들(130)을 포함하는 LevelDB 테이블 구조에 사용되는 것과 같은 검색 알고리즘들은 먼저 인덱스 블록(120)으로부터 인덱스 청크(index chunk)(140) 핸들(nandle)을 찾을 수 있다. 인덱스 청크(140)는 인덱스 청크(140) 핸들을 사용하여 찾아 질 수 있다. 인덱스 청크(140)는 오프셋 테이블(offset table)(150)과 관련 키(key)를 비교함에 의해서 (예를 들면, 바이너리 검색을 수행함에 의해서) 찾아질 수 있다. 즉, 검색 알고리즘들은 오프셋 테이블(150)을 가지고 바이너리 검색을 수행함에 의해서 인덱스 청크(140)를 검색할 수 있다.

[0041] 인덱스 청크(140)가 찾아지면, 검색 알고리즘은 스캐닝 알고리즘을 수행함에 의해서 인덱스 청크(140) 내에 포함된 데이터 블록 핸들(160)을 유사하게 찾을 수 있다. 데이터 블록 핸들(160)은 오프셋(offset) 및 크기(size)를 포함하고, 관련 데이터 블록(130)의 위치를 지시할 수 있다.

[0042] 검색 알고리즘은 데이터 블록 핸들(160)을 사용함에 의해서 관련 데이터 블록(130)을 로드(load)할 수 있다.

[0043] 추가적으로, 검색 알고리즘은 데이터 블록(130) 내 오프셋 테이블(190)을 이용하여 바이너리 검색을 수행함에 의해서 데이터 블록(130) 내에 있고 (예를 들면, 타겟 키 "abc"에 대응하는) 탐색 타겟 KV 데이터/KV 쌍(180)을 가지는 데이터 청크(170)를 찾기를 시도할 수 있다. 이 후, 검색 알고리즘은 데이터 청크(170) 내 KV 쌍(180)을 찾기 위하여 KV 쌍(180)을 스캔하기 위한 스캐닝 동작을 수행한다.

[0044] 따라서, 상술한 구조는 KV 쌍(180)을 얻기 위하여 스트링 비교 동작을 사용하여 적어도 2개의 바이너리 검색들 및 2개의 스캔들을 수행한다. 따라서, 더 빠른 단일 쿼리를 지원하는 포맷을 제공하는 것이 유리할 수 있다.

[0045] 도 2는 본 개시의 실시예들에 따른 KV 쌍을 검색하는 방법을 설명하기 위한 블록도이다.

[0046] 도 2를 참조하면, 본 개시의 실시예들에 따른 예의 방법은 일반적으로 다음과 같이 진행될 수 있다.

[0047] 초기에, 본 개시의 실시예들의 시스템은 KV 쌍(280)의 탐색 키 또는 타겟 키를 포함하는 테이블(예를 들면, 정렬된 키 테이블)(210)을 억세스할 수 있다. KV 쌍(280)은 분류 데이터를 식별하기 위한 키 및 데이터를 포함하는 값 또는 객체에 대응한다. 탐색/타겟 키는 추가적으로 아래에 기술되는 것으로서 키 탐색 동안 시스템에 의해서 탐색되는 키일 수 있다. 테이블(210)은 KV 테이블(key-value table)(211) 및 해시-오프셋 테이블(hash-offset table)(212)(예를 들면, 내부 키 해시 디렉토리(intenal key hash directory))를 포함할 수 있다. 데이터베이스의 KV 쌍들(280)의 다양한 키들(214)은 하나 이상의 KV 테이블들(211)로 분리될 수 있고, KV 테이블들(211) 각각은 검색 동안 KV 테이블(들)(211)의 키들(214)이 찾아질 수 있도록 하기 위하여 각 해시-오프셋 테이블(212)에 대응할 수 있다.

[0048] 이 후, 시스템은 주어진 타겟 키(214)에 대한 검색을 수행할 수 있다. 본 예에서, 시스템은 키 "abc"를 검색한다. 따라서, 시스템은 초기에 (예를 들면, 식 "target hash-value = Hash("abc")"에 따라, 타겟 키(214)에 대응하는 타겟 해시-값(216)을 결정함에 의해서) 타겟 키(214)에 대응하는 타겟 해시-값(216)을 계산할 수 있다. 예를 들면, 해시-값(216)은 키들(214)의 스트링들로부터 계산될 수 있다. 따라서, 해시 정보의 개별 인스턴스들

(instances)의 수는 이에 대응하는 키들(214)의 수 각각과 동일할 수 있다. 예를 들면, 정렬된 키 테이블은 KV 테이블(211) 내 KV 테이블 엔트리들(270)과 동일한 수의 해시-오프셋 테이블 엔트리들(250)로 해시 오프셋 테이블(212)을 채울 수 있고, 해시-값들(216)을 계산한 후, 시스템은 해시-값들(216)에 의해서 해시-오프셋 테이블(212)을 정렬할 수 있다.

[0049] 추가적으로, 대응하는 KV 쌍(280)의 실제 위치의 오프셋은 각 해시-값(216)에 포함될 수 있다 (예를 들면, 각 키(214)를 사용함에 의해서, 시스템은 대응하는 해시-값들(216)을 생성할 수 있다). 따라서, 시스템이 메타데이터를 구축할 때, 해시-값(216)은 KVSSD 내 어떤 위치에 저장될 수 있고, 키 엔트리들(270)은 해시-오프셋 테이블 엔트리들(250)에 매핑될 수 있다.

[0050] 그리고, 시스템은 해시-오프셋 테이블(212) 내 타겟 해시-값(216)을 검색함에 의해서 바이너리 검색을 수행할 수 있다. 해시-오프셋 테이블(212)의 해시-오프셋 테이블 엔트리들(250)은 그들의 각 해시-값(216)에 기초하여 정렬될 수 있고, 대응하는 키 엔트리(270) 내에 대응하는 KV 쌍(280)을 찾기 위한 각 해시 오프셋(218)은 해시-오프셋 테이블(212)내에 위치한다.

[0051] 타겟 키(214)의 KV 쌍(28)에 대응하는 해시 오프셋(218)을 찾은 후, 시스템은 버퍼 내에 해시 오프셋(218)로부터 KV 쌍(280)을 검색한다. 따라서, 랜덤 타겟 키(214) (예를 들면, 키 "abc")를 찾는 경우, 시스템은 (예를 들면, 식 "hash = Hash("abc")에 따라) 해시 번호(hash number)/해시-값(216)을 결정하기 위하여 이에 대응하는 해시를 계산할 수 있고, 바이너리 검색을 수행함에 의해서 해시-오프셋 테이블(212) 내에 해시-값(216)을 검색할 수 있다. 이 경우에, 시스템은 해시 오프셋(218)을 획득하고, 해시 오프셋(218)이 키(214)를 찾기 위한 어떤 위치를 가리키는 것으로 결정한다.

[0052] 키들(214)의 임의의 스트링에 대하여, 일반적으로 많은 양의 관련 데이터가 있다. 상술한 바와 같이, 그리고 개시된 실시예들에 따라, 키들(214)은 정렬된 키 테이블(210) 내에 별개의 KV 테이블 엔트리들(270)로 분할될 수 있다. 이 후, 정렬된 키 테이블(210)은 KV 테이블(211) 및 해시-오프셋 테이블(212)을 포함할 수 있다. 해시-오프셋 테이블(212) 내에 해시-값(216)을 검색함에 의해서 시스템은 도 1에 도시된 예와 비교하여 비교적 적은 CPU 사이클을 이용하여 타겟 KV 쌍(280)을 찾을 수 있다.

[0053] 개시된 실시예들에서, 해시-값들(216)은 비교적 작은 양의 데이터를 전용하고, 모든 키들(214)은 대응하는 해시-값(216)으로 변환되거나, 매핑될 수 있다. 각 키(214)는 이에 대응하는 해시-값(216)을 비교하는 것보다 계산 비용이 더 비싸기 때문에, 개시된 실시예의 시스템은 도 1에 도시된 예보다 더 빠르고, 더 작고, 더 효과적일 수 있다. 즉, 상술된 것처럼, 본 개시의 실시예들에 따라 정렬된 키 테이블 내 내부 키 해시 디렉토리는 내부 키 해시 디렉토리 내에 검색 계층이 덜 요구되거나 요구되지 않을 수 있기 때문에 더 효과적일 수 있다. 따라서, KV 스토리지 내의 테이블 내에 키를 찾는 것과 관련되는 CPU 사이클의 수가 감소될 수 있다.

[0054] 도 3은 본 개시의 실시예들에 따른 KV 쌍을 검색하는 방법을 나타내는 블록도이다. 도 4는 본 개시의 실시예들에 따른 KV 쌍을 검색하는 시스템을 나타내는 블록도이다.

[0055] 도 3 및 도 4를 참조하면, KV 쌍을 검색하는 방법은 (예를 들면, 스토리지 포맷 데이터 구조(410a 또는 410b) 내에서) 정렬된 키 테이블의 해시-오프셋 테이블을 해시-오프셋 테이블 엔트리 및 해시 오프셋으로 채워넣는 것을 포함할 수 있다(310). 해시-오프셋 테이블 엔트리들은 각 키에 대응하는 해시-값을 가질 수 있다. 정렬된 키 테이블은 복수개의 KV 테이블 엔트리들을 포함하는 KV 테이블을 포함할 수 있고, KV 테이블 엔트리들은 KV 쌍을 포함할 수 있다. KV 테이블 엔트리들의 수는 해시-오프셋 테이블 엔트리들의 수와 동일할 수 있다.

[0056] 본 방법은 해시-값들에 기초하여 해시-오프셋 테이블 엔트리들을 정렬하는 것을 추가적으로 포함할 수 있다(320).

[0057] 본 방법은 해시-오프셋 테이블 내에 타겟 키에 대응하는 해시-값들의 타겟 해시-값을 검색하는 것을 추가적으로 포함할 수 있다(330). 타겟 해시 값을 검색하는 것을 바이너리 검색을 수행하는 것을 포함할 수 있다.

[0058] 본 방법은 타겟 키로부터 타겟 해시-값을 계산하는 것을 추가적으로 포함할 수 있다(340).

[0059] 본 방법은 타겟 해시-값에 기초하여 타겟 키에 대응하는 타겟 KV 쌍을 찾는 것을 추가적으로 포함할 수 있다(350). 타겟 KV 쌍을 찾는 것은 해시 오프셋에 기초하여 타겟 KV 쌍을 찾는 것을 포함할 수 있다.

[0060] 본 방법은 타겟 KV 쌍의 위치를 저장하는 것을 추가적으로 포함할 수 있다(360). 타겟 KV 쌍의 위치를 저장하는 것은 타겟 해시-값을 타겟 KV 쌍에 매핑하는 것을 포함할 수 있다.

[0061] 도 4는 본 개시의 실시예들에 따른 KV 쌍을 검색하는 시스템을 나타내는 블록도이다.

[0062] 도 4를 참조하면, 온-메모리 데이터 구조(420)는 일반적으로 내부 키 해시 디렉토리 생성 및 라이프 사이클(life cycle)에 관련된다. 온-메모리 데이터 구조(420)의 내부 동작들에 관한 세부 사항은 특히 본 개시의 실시예들에 관련되지 않음을 주목할 수 있다. 온-메모리 데이터 구조(420)는 해시 맵, 또는 온-메모리 해시 테이블(422), 및 온-메모리 정렬 구조(424)를 포함할 수 있다. 온-메모리 데이터 구조(420)는 초기에 온-메모리 KV 객체들(426)을 관리한다.

[0063] 온-메모리 데이터 구조(420)가 플러쉬(flush)를 야기할 때, 임의의 로깅된 KV들(426)이 메모리 공간을 비우기 위해 삭제로 표시될 수 있도록, 온-메모리 데이터 구조(420)는 내부 키 해시 디렉토리(412)를 포함하는 스토리지 포맷 데이터 구조(410a 또는 410b)를 병합하거나 생성할 수 있다. 예를 들면, 온-메모리 데이터 구조(420)는 기존 스토리지 포맷 데이터 구조(410a)와 병합될 수 있다. 스토리지 포맷 데이터 구조(410a 또는 401b)의 병합 및 생성은 도 3의 플로우차트에 대응한다 (예를 들면, 해시-오프셋 테이블을 채우고 정렬하는 것, 타겟 해시-테이블을 검색하고 계산하는 것, 및 타겟 KV 쌍의 위치를 찾고 저장하는 것).

[0064] 이 후, 새롭게 생성되거나 새롭게 병합된 스토리지 포맷 데이터 구조(410b)는 현존 스토리지 포맷 데이터 구조(410a)를 대체하고(Replace), 스토리지 디바이스(430)에 저장된다.

[0065] 추가적으로, 임의의 Read(get) 동작은 온-메모리 데이터 구조(420)가 최신 관련 KV 쌍(426)을 가질 수 있기 때문에, 온-메모리 데이터 구조(420)가 초기에 검색되도록 할 것이다. 온-메모리 데이터 구조(420)가 최신 관련 KV 쌍(426)을 가지지 않으면, 스토리지 포맷 데이터 구조(410b)가 KV 쌍(426)을 검색할 수 있다.

[0066] 따라서, 개시된 실시예들은 KV 데이터베이스 시스템에 대한 빠른 단일 키 쿼리를 가능하게 하여, 이의 검색 계층을 단순화함에 의해서 정렬된 키 테이블 내 단일 키의 비교적 빠른 키 위치를 찾는 것을 가능하게 하고, 비교 오버헤드의 감소를 가능하게 하는 스토리지 데이터 포맷을 제공하고, 이에 따라 단일 키 리드 성능을 개선할 수 있다.

부호의 설명

[0067] 110: 정렬된 스트링 테이블 120: 인덱스 블록

130: 데이터 블록 210: 정렬된 키 테이블

420: 온-메모리 데이터 구조 422: 온-메모리 해시 테이블

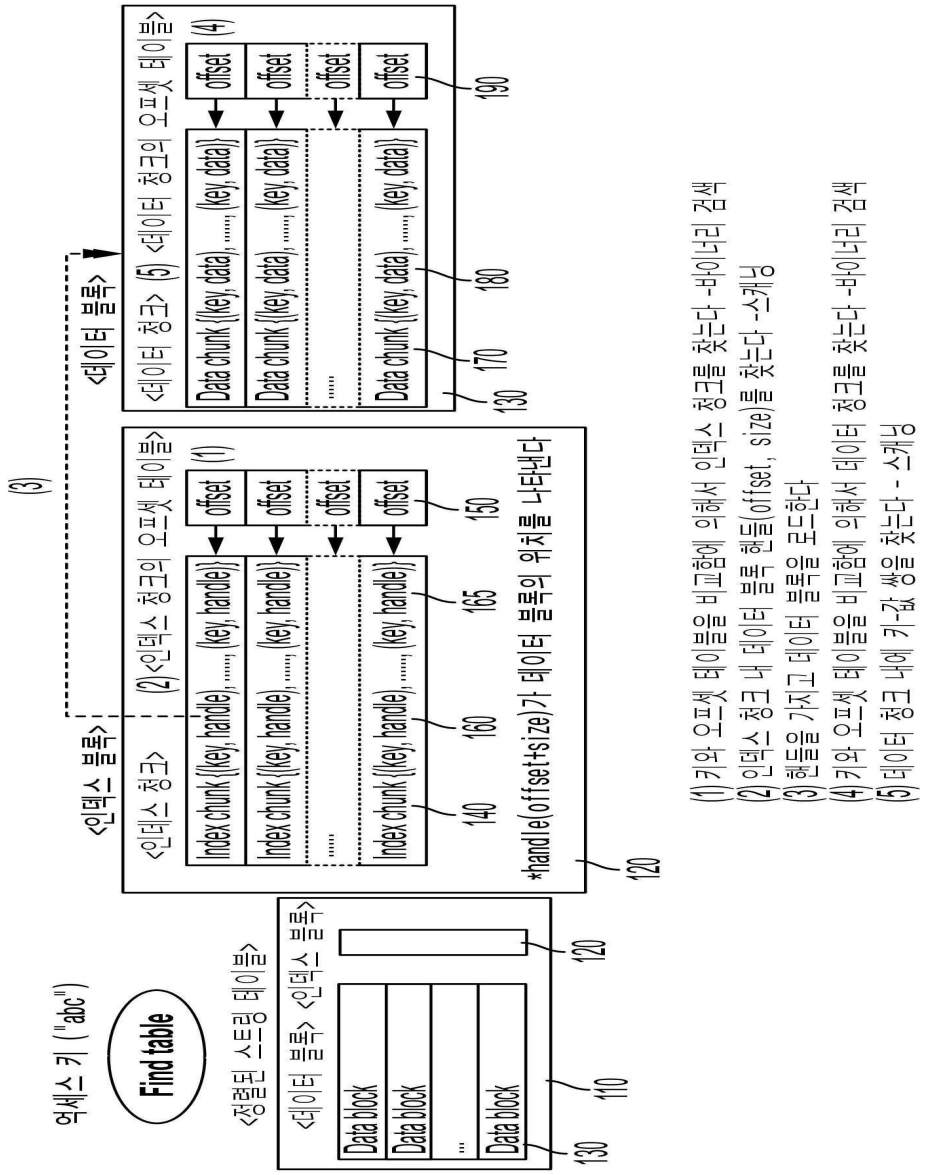
424: 온-메모리 정렬 구조 412: 내부 키 해시 디렉토리

410a, 410b: 스토리지 포맷 데이터 구조

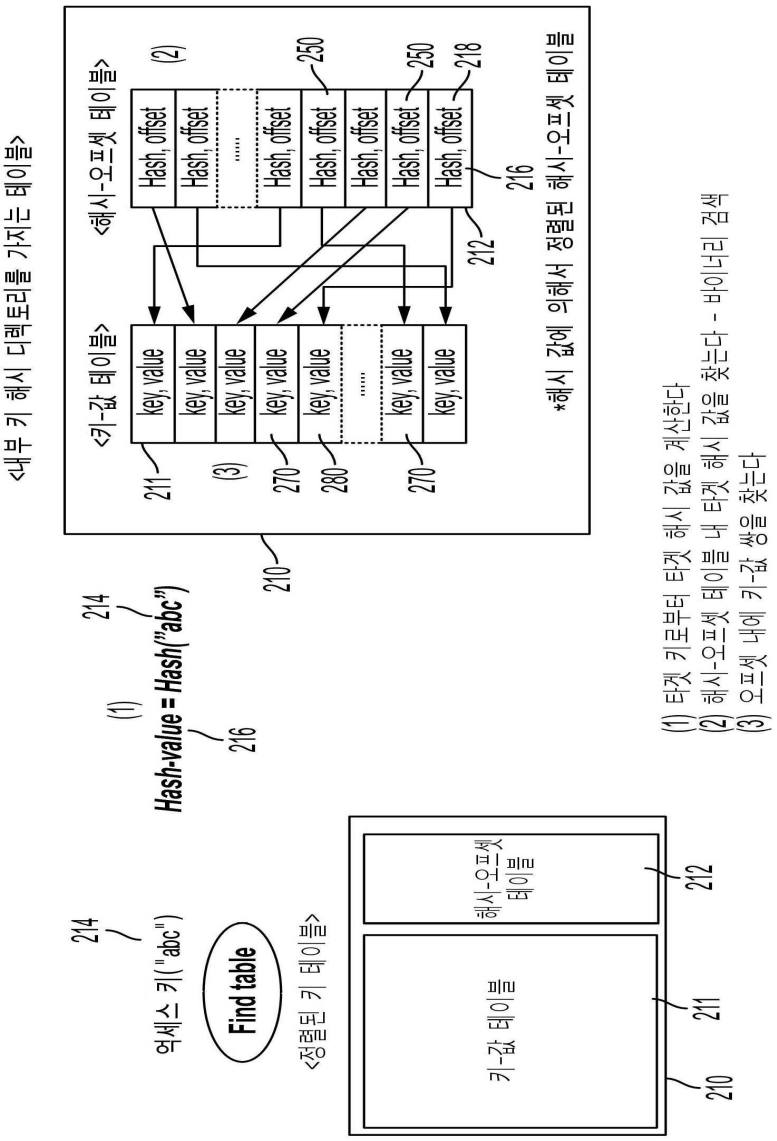
430: 스토리지 디바이스

도면

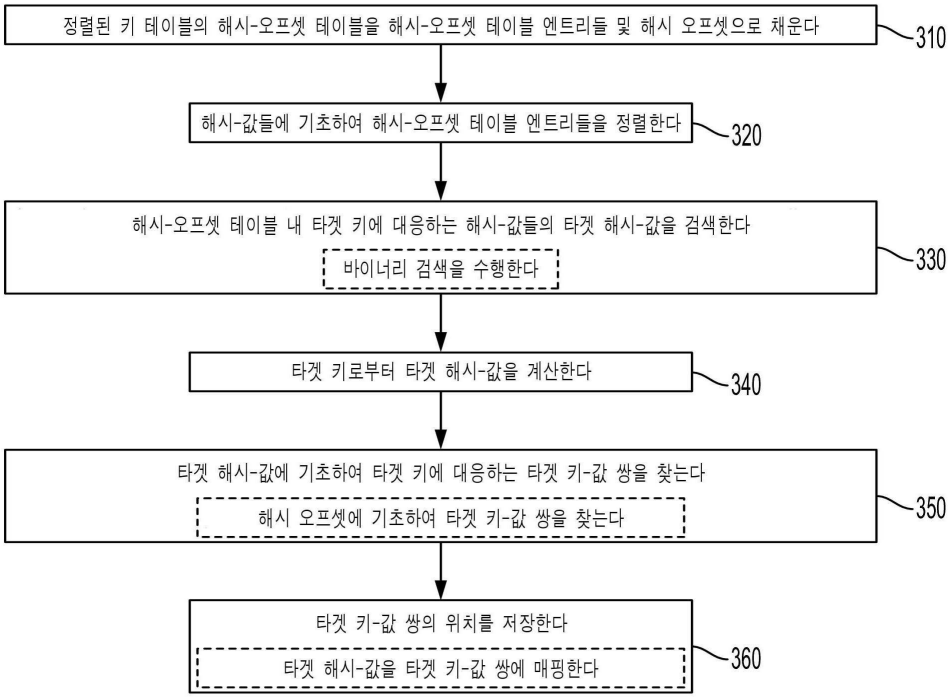
도면1


억세스 키 ("abc")
Find table
<정렬된 스트링 테이블>
<데이터 블록> <인덱스 블록>
Data block
Data block
...
Data block
130
110
120
<인덱스 블록>
(3)
<인덱스 청크>
(2)<인덱스 청크의 오프셋 테이블>
(1)
offset
Index chunk {{key, handle}, ..., {key, handle}}
140
160
165
150
*handle(offset+size)가 데이터 블록의 위치를 나타낸다
120
<데이터 블록>
<데이터 청크>
(5) <데이터 청크의 오프셋 테이블>
(4)
offset
Data chunk {{key, data}, ..., {key, data}}
130
170
180
190
(1)키와 오프셋 테이블을 비교함에 의해서 인덱스 청크를 찾는다 -바이너리 검색
(2)인덱스 청크 내 데이터 블록 핸들(offset, size)를 찾는다 -스캐닝
(3)핸들을 가지고 데이터 블록을 로드한다
(4)키와 오프셋 테이블을 비교함에 의해서 데이터 청크를 찾는다 -바이너리 검색
(5)데이터 청크 내에 키-값 쌍을 찾는다 - 스캐닝

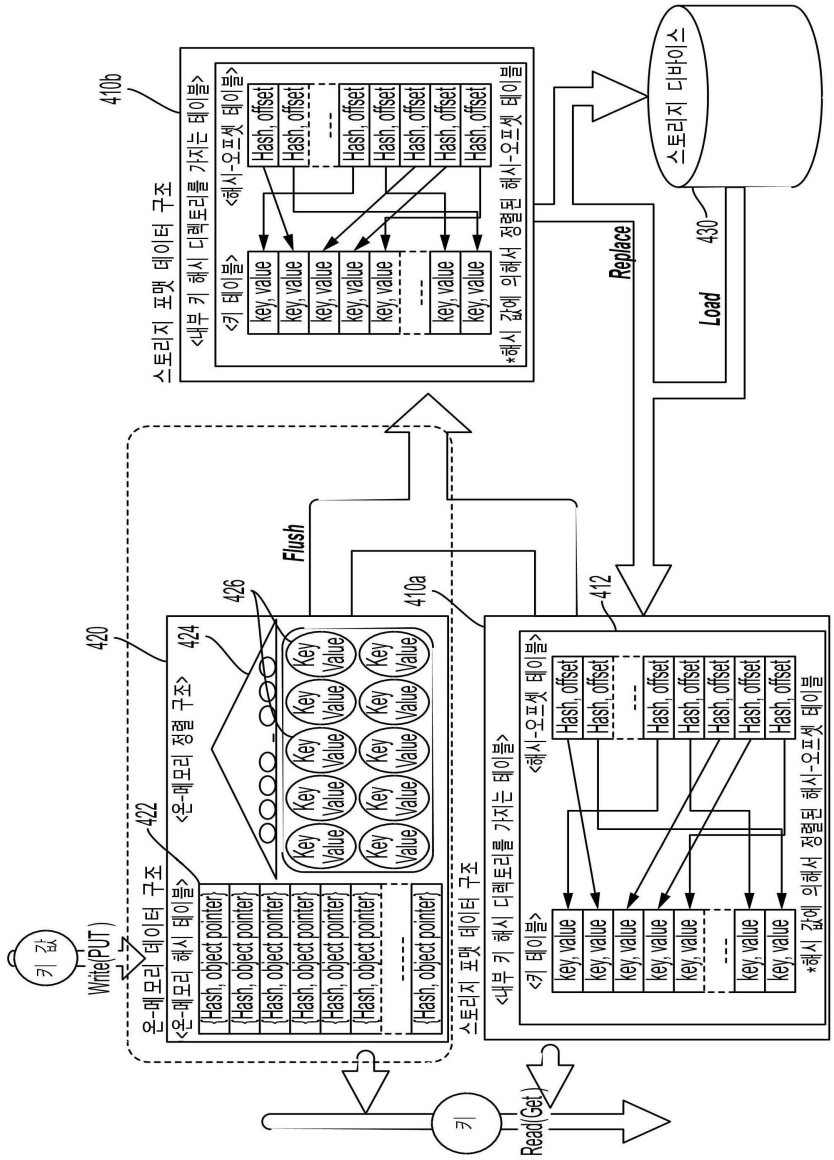
도면2


<내부 키 해시 디렉토리를 가지는 테이블>
<키-값 테이블>
<해시-오프셋 테이블>
(2)
Hash, offset
key, value
211
(3)
270
280
250
218
216
212
*해시 값에 의해서 정렬된 해시-오프셋 테이블
210
214
(1)
Hash-value = Hash("abc")
억세스 키("abc")
Find table
<정렬된 키 테이블>
해시-오프셋 테이블
키-값 테이블
(1) 타겟 키로부터 타겟 해시 값을 계산한다
(2) 해시-오프셋 테이블 내 타겟 해시 값을 찾는다 - 바이너리 검색
(3) 오프셋 내에 키-값 쌍을 찾는다

도면3


정렬된 키 테이블의 해시-오프셋 테이블을 해시-오프셋 테이블 엔트리들 및 해시 오프셋으로 채운다
310
해시-값들에 기초하여 해시-오프셋 테이블 엔트리들을 정렬한다
320
해시-오프셋 테이블 내 타겟 키에 대응하는 해시-값들의 타겟 해시-값을 검색한다
바이너리 검색을 수행한다
330
타겟 키로부터 타겟 해시-값을 계산한다
340
타겟 해시-값에 기초하여 타겟 키에 대응하는 타겟 키-값 쌍을 찾는다
해시 오프셋에 기초하여 타겟 키-값 쌍을 찾는다
350
타겟 키-값 쌍의 위치를 저장한다
타겟 해시-값을 타겟 키-값 쌍에 매핑한다
360

도면4


410b
스토리지 포맷 데이터 구조
<내부 키 해시 디렉토리를 가지는 테이블>
<해시-오프셋 테이블>
Hash, offset
<키 테이블>
key, value
*해시 값에 의해서 정렬된 해시-오프셋 테이블
스토리지 디바이스
430
Load
Replace
Flush
420
424
426
410a
412
422
온-메모리 데이터 구조
<온-메모리 정렬 구조>
<온-메모리 해시 테이블>
{Hash, object pointer}
Key
Value
키 값
Write(PUT)
키
Read(Get)