(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2016년12월28일
(11) 등록번호 10-1690288
(24) 등록일자 2016년12월21일

(51) 국제특허분류(Int. Cl.)
*G06F 17/30* (2006.01)
(52) CPC특허분류
*G06F 17/3048* (2013.01)
*G06F 17/30377* (2013.01)
(21) 출원번호 10-2015-7007498
(22) 출원일자(국제) 2013년09월04일
심사청구일자 2015년08월03일
(85) 번역문제출일자 2015년03월24일
(65) 공개번호 10-2015-0075407
(43) 공개일자 2015년07월03일
(86) 국제출원번호 PCT/EP2013/002655
(87) 국제공개번호 WO/2014/048540
국제공개일자 2014년04월03일
(30) 우선권주장
12368027.4 2012년09월27일
유럽특허청(EPO)(EP)
13/628,517 2012년09월27일 미국(US)
(56) 선행기술조사문헌
US08799409 B2

(73) 특허권자
**아마데우스 에스.에이.에스.**
프랑스 에프-06410 비요 소피아 앙띠폴리 루뜨 드 뻥 몽따르 485
(72) 발명자
**르두떼, 장-샤를르**
영국, 더블유2 3유더블유 런던, 웨스트본 테라스 10, 플랫 9
**생제, 조엘**
프랑스공화국, 에프-06600 앙띠브, 라스 빨마스 밧 으, 슈멩 데 4 슈멩 151
*(뒷면에 계속)*
(74) 대리인
**김태홍, 김진희**

전체 청구항 수 : 총 14 항 심사관 : 고재용

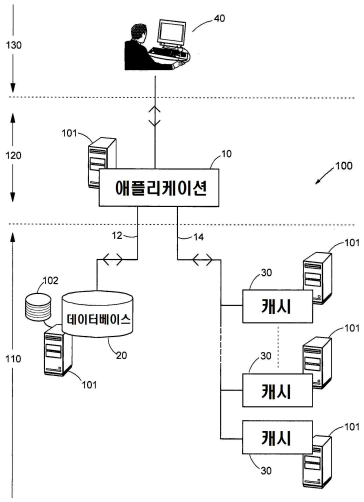
(54) 발명의 명칭 **데이터를 저장하고 검색하는 방법 및 시스템**

## (57) 요 약

본 발명은 소프트웨어 애플리케이션에 의해 데이터를 저장하는 방법 및 시스템에 관한 것이다. 하나 이상의 데이터베이스 시스템과 적어도 하나의 캐시 노드를 포함하는 데이터 저장 시스템에서 소프트웨어 애플리케이션은 제1 전용 인터페이스 상의 하나 이상의 데이터베이스 시스템과, 제2 전용 인터페이스 상의 적어도 하나의 캐시 노드

(뒷면에 계속)

*대 표 도* - 도1



와 독립적으로 인터페이싱한다. 본 발명의 방법 및 시스템은 소프트웨어 애플리케이션에 의해 데이터 저장 시스템의 각 판독 질문이 만약 이용가능한 경우 질문된 데이터를 리턴하는 복수의 캐시 노드로만 제일 먼저 발행되는 것을 특징으로 한다. 만약 이용가능하지 않은 경우, 소프트웨어 애플리케이션은 하나 이상의 데이터베이스 시스템으로부터 질문된 데이터의 페치를 트리거하는 미검출을 수신한다. 질문된 데이터를 검색하면, 소프트웨어 애플리케이션은 질문된 데이터를 적어도 하나의 캐시 노드에 추가한다. 본 발명의 방법 및 시스템은 또한 소프트웨어 애플리케이션에 의한 하나 이상의 데이터베이스 시스템의 각 기록이 적어도 하나의 캐시 노드에 동시에 수행되는 것을 더 특징으로 한다. 그리하여, 적어도 하나의 캐시 노드의 파퓰레이트는, 적어도 하나의 캐시 노드의 각 미검출된 판독 질문 및 데이터 저장 시스템의 각 기록 질문에 신속히 수행된다.

(52) CPC특허분류
*G06F 17/30581* (2013.01)

(72) 발명자
**바라르, 프로랭**
프랑스공화국, 에프-06700 생-로랭 뒤 바르, 아브뉘 드 베르덩 324
**프뤼돔므, 플로리앙**
프랑스공화국, 에프-13127 비트롤르, 뤼 떼오도르 오바넬, 밧 데7 레스 게 로지
**부떼룹, 로망**
프랑스공화국, 에프-06700 생 로랭 뒤 바르, 빌라 15, 아브뉘 데 쁠랑띠에 781
**삐트라, 꼴랑**
프랑스공화국, 에프-06600 앙띠브, 불르바르 뒤 쁘레지덩 윌슨 58

# 명 세 서

## 청구범위

### 청구항 1

데이터 저장 시스템에서 데이터를 저장하고 검색(retrieve)하는 방법에 있어서,

상기 데이터 저장 시스템은, 중간 계층(middle tier)을 구현하는 적어도 하나의 컴퓨터 - 상기 적어도 하나의 컴퓨터는, 적어도 하나의 데이터 프로세서와, 상기 적어도 하나의 데이터 프로세서에 의해 상기 적어도 하나의 컴퓨터 상에서 실행되고 있는 소프트웨어 애플리케이션을 포함함 - 를 포함하고,

상기 데이터 저장 시스템은, 저장 계층(storage tier)을 구현하는 복수의 캐시 노드 및 하나 이상의 데이터베이스 시스템을 더 포함하고,

상기 중간 계층은, 클라이언트 계층과 상기 데이터 저장 시스템의 저장 계층을 인터페이싱하도록 구성되고,

상기 방법은,

상기 중간 계층에서의 상기 적어도 하나의 컴퓨터에서,

상기 클라이언트 계층의 유저 장치로부터 데이터의 적어도 하나의 판독(reading)을 요청하는 제1 유저 요청을 수신하는 것에 응답하여, 상기 복수의 캐시 노드에만 판독 질의(read query)을 송신하는 단계와,

상기 판독 질의에 응답하여 적어도 하나의 캐시 노드로부터 제1 질의 응답 데이터(queried data)를 수신하는 것에 응답하여, 상기 적어도 하나의 데이터 프로세서로, 상기 제1 질의 응답 데이터를 사용하여 상기 제1 유저 요청을 처리하는 단계와,

상기 판독 질의에 응답하여 모든 캐시 노드로부터 미검출(miss)을 수신하는 것에 응답하여, 상기 적어도 하나의 데이터 프로세서로, 상기 제1 유저 요청에 기초하여 상기 하나 이상의 데이터베이스 시스템을 페치하는(fetching) 단계와,

상기 페치하는 것의 결과로서 상기 하나 이상의 데이터베이스 시스템으로부터 제2 질의 응답 데이터를 검색하는 것에 응답하여, 상기 하나 이상의 데이터베이스 시스템으로부터의 상기 제2 질의 응답 데이터를 사용하여 상기 제1 유저 요청을 처리하고, 상기 적어도 하나의 캐시 노드에 상기 제2 질의 응답 데이터를 추가하라는 명령과 상기 제2 질의 응답 데이터를 상기 적어도 하나의 캐시 노드에 송신함으로써, 상기 미검출된 판독 질의에 응답하여 상기 하나 이상의 데이터베이스 시스템으로부터 상기 적어도 하나의 캐시 노드로 상기 제2 질의 응답 데이터를 파퓰레이트하는(populate) 단계와,

상기 하나 이상의 데이터베이스 시스템으로부터 상기 제2 질의 응답 데이터를 페치하는 것과 동시에 업데이트된 데이터의 적어도 하나의 기록(writing)을 요청하는 제2 유저 요청을 수신하는 것에 응답하여, 상기 업데이트된 데이터로 상기 하나 이상의 데이터베이스 시스템을 기록하기 위한 명령을 송신하고, 상기 업데이트된 데이터로 상기 적어도 하나의 캐시 노드를 동시에 기록하기 위한 명령을 송신함으로써, 그에 의해 상기 데이터 저장 시스템의 각 기록 질의(write query)를 상기 복수의 캐시 노드에 파퓰레이트하고, 상기 복수의 캐시 노드에 상기 업데이트된 데이터가 저장되도록, 상기 적어도 하나의 캐시 노드에의 상기 제2 질의 응답 데이터의 후속 추가를 중지하는(aborting) 단계를 포함하는,

데이터 저장 시스템에서 데이터를 저장하고 검색하는 방법.

### 청구항 2

제1항에 있어서,

상기 제2 유저 요청은, 상기 데이터베이스 시스템에 데이터를 추가하는 것, 업데이트하는 것, 및 삭제하는 것 중 적어도 하나를 포함하는 것인,

데이터 저장 시스템에서 데이터를 저장하고 검색하는 방법.

**청구항 3**

제1항에 있어서,

상기 하나 이상의 데이터베이스 시스템으로부터 상기 적어도 하나의 캐시 노드로 상기 제2 질의 응답 데이터를 성공적으로 추가하는 것을 완료했을 때 긍정적인 수신확인(positive acknowledgement)을 수신하는 단계를 더 포함하는,

데이터 저장 시스템에서 데이터를 저장하고 검색하는 방법.

**청구항 4**

제1항에 있어서,

상기 중간 계층의 상기 적어도 하나의 컴퓨터는, 제1 전용 인터페이스 상에서 상기 하나 이상의 데이터베이스 시스템과 독립적으로 인터페이싱하고,

상기 중간 계층의 상기 적어도 하나의 컴퓨터는, 제2 전용 인터페이스 상에서 상기 복수의 캐시 노드와 인터페이싱하는 것인,

데이터 저장 시스템에서 데이터를 저장하고 검색하는 방법.

**청구항 5**

제1항에 있어서,

상기 복수의 캐시 노드의 데이터 모델과 상기 하나 이상의 데이터베이스의 데이터 모델은, 캐시 노드와 데이터베이스 데이터에 액세스하기 위해 동일한 어드레스 키들이 유도될 수 있도록, 일관된(consistent) 것인,

데이터 저장 시스템에서 데이터를 저장하고 검색하는 방법.

**청구항 6**

제1항에 있어서,

상기 중간 계층의 상기 적어도 하나의 컴퓨터는, 여행 제공자의 인벤토리 시스템인 것인,

데이터 저장 시스템에서 데이터를 저장하고 검색하는 방법.

**청구항 7**

제1항에 있어서,

상기 제1 유저 요청은, 여행사, 온라인 여행사, 온라인 고객 중 적어도 하나에 의해 송신되는 것인,

데이터 저장 시스템에서 데이터를 저장하고 검색하는 방법.

**청구항 8**

제1항에 있어서,

상기 하나 이상의 데이터베이스 시스템에서의 데이터의 기록을 위한 명령과 상기 적어도 하나의 캐시 노드를 동시에 기록하기 위한 명령을 송신하는 것에 응답하여,

상기 기록이 적용되는 현재 저장된 데이터를 상기 하나 이상의 데이터베이스 시스템으로부터 검색하고, 상기 현재 저장된 데이터를 록킹(locking)하는 단계와,

저장될 새로운 데이터를 상기 하나 이상의 데이터베이스 시스템에 기록하는 단계와,

상기 저장될 새로운 데이터를 일시적으로 유지하기 위해 상기 중간 계층의 상기 적어도 하나의 컴퓨터의 캐시 버퍼에 기록하는 단계와,

상기 저장될 새로운 데이터를 상기 적어도 하나의 캐시 노드로 포워딩하고 설정하는 단계와,

상기 하나 이상의 데이터베이스 시스템에 트랜잭션(transaction)을 커미트(commit)하는 단계를 더 포함

하는,

데이터 저장 시스템에서 데이터를 저장하고 검색하는 방법.

**청구항 9**

청구항 9은(는) 설정등록료 납부시 포기되었습니다.

제8항에 있어서,

상기 커미트가 실패하는 것에 응답하여, 상기 적어도 하나의 캐시 노드에서 상기 새로운 데이터를 삭제하는 단계를 더 포함하는,

데이터 저장 시스템에서 데이터를 저장하고 검색하는 방법.

**청구항 10**

제1항에 있어서,

상기 복수의 캐시 노드 중에서, 상기 제2 질의 응답 데이터를 추가하기 위한 명령 또는 상기 업데이트된 데이터로 상기 적어도 하나의 캐시 노드를 기록하기 위한 명령이 송신되는 상기 적어도 하나의 캐시 노드를 결정하는 단계를 더 포함하는,

데이터 저장 시스템에서 데이터를 저장하고 검색하는 방법.

**청구항 11**

청구항 11은(는) 설정등록료 납부시 포기되었습니다.

제10항에 있어서,

상기 복수의 캐시 노드 중에서의 상기 적어도 하나의 캐시 노드는, 부하 균형(load balancing)에 기초하여 결정되는 것인,

데이터 저장 시스템에서 데이터를 저장하고 검색하는 방법.

**청구항 12**

제1항에 있어서,

하나 이상의 데이터베이스 시스템이 상기 제1 유저 요청의 요청된 데이터에 연관된 미검출(miss)을 리턴(return)하는 것에 응답하여, 부재 데이터(data of absence)가 모든 후속 판독 질의에 대하여 즉시 이용가능하게 됨으로써, 상기 제1 유저 요청의 요청된 데이터를 검색하기 위한 상기 하나 이상의 데이터베이스 시스템의 후속 페치를 회피하도록, 상기 적어도 하나의 캐시 노드로의 추가를 위한 상기 제1 유저 요청에 대응하는 상기 부재 데이터를 상기 적어도 하나의 캐시 노드로 송신하는 단계를 더 포함하는,

데이터 저장 시스템에서 데이터를 저장하고 검색하는 방법.

**청구항 13**

제1항에 있어서,

상기 복수의 캐시 노드 각각은 헤더(header)를 포함하는 레코드(record)를 저장하고,

상기 적어도 하나의 캐시 노드의 헤더는, 상기 요청된 데이터가 상기 적어도 하나의 데이터베이스 시스템에서 미검출된 것 또는 상기 요청된 데이터의 값이 상기 요청된 부재 데이터를 나타내는 값으로 설정된다는 것을 나타내는 것인,

데이터 저장 시스템에서 데이터를 저장하고 검색하는 방법.

**청구항 14**

제1항에 있어서,

상기 복수의 캐시 노드의 레코드와 상기 하나 이상의 데이터베이스 시스템의 레코드는 상기 복수의 캐시 노드와

상기 하나 이상의 데이터베이스 시스템의 관련된 레코드가 일관되게 어드레스되도록 저장되는 것인,

데이터 저장 시스템에서 데이터를 저장하고 검색하는 방법.

**청구항 15**

청구항 15은(는) 설정등록료 납부시 포기되었습니다.

제14항에 있어서,

상기 레코드와 연관된 키(key)는 상기 하나 이상의 데이터베이스 시스템에 저장된 레코드가 록킹되게 하는 것인,

데이터 저장 시스템에서 데이터를 저장하고 검색하는 방법.

**청구항 16**

청구항 16은(는) 설정등록료 납부시 포기되었습니다.

제14항에 있어서,

상기 복수의 캐시 노드의 각각의 레코드와 상기 하나 이상의 데이터베이스 시스템의 각각의 레코드는 기능 엔티티(functional entity)에 의해 그룹화되고 키(key) - 상기 키는, 상기 하나 이상의 데이터베이스 시스템과 상기 복수의 캐시 노드에서 상기 레코드가 즉시 액세스 가능하게 함 - 에 의해 색인되는 것인,

데이터 저장 시스템에서 데이터를 저장하고 검색하는 방법.

**청구항 17**

청구항 17은(는) 설정등록료 납부시 포기되었습니다.

제16항에 있어서,

각 기능 엔티티는 항공편 날짜(flight-date)이고,

각 키는 항공편 날짜 키인 것인,

데이터 저장 시스템에서 데이터를 저장하고 검색하는 방법.

**청구항 18**

데이터 저장 시스템에서 데이터를 저장하고 검색하는 방법에 있어서,

상기 데이터 저장 시스템은, 중간 계층(middle tier)을 구현하는 적어도 하나의 컴퓨터 - 상기 적어도 하나의 컴퓨터는, 적어도 하나의 데이터 프로세서와, 상기 적어도 하나의 데이터 프로세서에 의해 상기 적어도 하나의 컴퓨터 상에서 실행되고 있는 소프트웨어 애플리케이션을 포함함 - 를 포함하고,

상기 데이터 저장 시스템은, 저장 계층(storage tier)을 구현하는 복수의 캐시 노드 및 하나 이상의 데이터베이스 시스템을 더 포함하고,

상기 중간 계층은, 클라이언트 계층과 상기 데이터 저장 시스템의 저장 계층을 인터페이싱하도록 구성되고,

상기 방법은,

상기 중간 계층에서의 상기 적어도 하나의 컴퓨터에서, 데이터의 판독을 요청하는 제1 유저 요청을 수신하는 것에 응답하여, 상기 복수의 캐시 노드에만 판독 질의(read query)을 송신하는 단계와,

상기 판독 질의에 응답하여 적어도 하나의 캐시 노드로부터 제1 질의 응답 데이터(queried data)를 수신하는 것에 응답하여, 상기 제1 질의 응답 데이터로 상기 제1 유저 요청을 처리하는 단계와,

상기 판독 질의에 응답하여 모든 캐시 노드로부터 미검출(miss)을 수신하는 것에 응답하여, 상기 제1 유저 요청에 기초하여 상기 하나 이상의 데이터베이스 시스템을 페치하는(fetching) 단계와,

상기 페치하는 것의 결과로서 상기 하나 이상의 데이터베이스 시스템으로부터 제2 질의 응답 데이터를 검색하는 것에 응답하여, 상기 제1 유저 요청을 처리하고, 상기 제2 질의 응답 데이터와 상기 적어도 하나의 캐시 노드에

상기 제2 질의 응답 데이터를 추가하라는 명령을 상기 적어도 하나의 캐시 노드에 송신함으로써, 상기 미검출된 판독 질의에 응답하여 상기 하나 이상의 데이터베이스 시스템으로부터 상기 적어도 하나의 캐시 노드로 상기 제2 질의 응답 데이터를 파퓰레이트하는(populate) 단계와,

상기 판독 질의에 응답하여 모든 캐시 노드로부터 미검출을 수신하고 상기 하나 이상의 데이터베이스 시스템이 상기 제1 유저 요청의 요청된 데이터와 연관된 미검출을 반환하는 것에 응답하여, 상기 제1 유저 요청의 요청된 데이터를 검색하기 위한 상기 하나 이상의 데이터베이스 시스템의 후속 페치를 회피하도록, 상기 제1 유저 요청의 상기 요청된 데이터가 상기 데이터 저장 시스템의 상기 하나 이상의 데이터베이스 시스템에 저장되어 있지 않음을 나타내는 부재 데이터(data of absence)를 상기 복수의 캐시 노드에 추가하는 단계를 포함하는,

데이터 저장 시스템에서 데이터를 저장하고 검색하는 방법.

**청구항 19**

청구항 19은(는) 설정등록료 납부시 포기되었습니다.

제18항에 있어서,

상기 데이터 저장 시스템은 항공사 인벤토리 시스템이고,

상기 하나 이상의 데이터베이스 시스템은 상기 항공사 인벤토리 시스템에 의해 관리되는 항공편에 대한 이용가능성을 저장하고,

상기 복수의 캐시 노드는 상기 항공사 인벤토리 시스템에 의해 관리되는 항공편에 대한 이용가능성을 저장하고,

상기 제1 유저 요청은 상기 항공사 인벤토리 시스템에 의해 관리되는 적어도 하나의 항공편에 관한 이용가능성에 대한 것인,

데이터 저장 시스템에서 데이터를 저장하고 검색하는 방법.

**청구항 20**

청구항 20은(는) 설정등록료 납부시 포기되었습니다.

제19항에 있어서,

적어도 하나의 항공편에 관한 이용가능성을 수정하기 위한 기록(writing)을 요구하는 제2 유저 요청을 수신하는 것에 응답하여, 상기 하나 이상의 데이터베이스 시스템을 기록하기 위한 명령을 송신하고, 또한 상기 복수의 캐시 노드를 동시에 기록하기 위한 명령을 송신함으로써, 상기 데이터 저장 시스템의 각 기록 질의에 응답하여 상기 복수의 캐시 노드를 파퓰레이트하는 단계를 더 포함하는,

데이터 저장 시스템에서 데이터를 저장하고 검색하는 방법.

**청구항 21**

청구항 21은(는) 설정등록료 납부시 포기되었습니다.

제20항에 있어서,

상기 제2 유저 요청은, 좌석을 구매하는 것, 좌석을 취소하는 것, 좌석을 변경하는 것 중 적어도 하나에 대응하는 것인,

데이터 저장 시스템에서 데이터를 저장하고 검색하는 방법.

**청구항 22**

데이터 저장 시스템에 있어서,

하나 이상의 데이터베이스 시스템;

복수의 캐시 노드;

적어도 하나의 데이터 프로세서; 및

소프트웨어 애플리케이션을 저장하는 메모리를 포함하고,

상기 적어도 하나의 데이터 프로세서에 의한 상기 소프트웨어 애플리케이션의 실행은, 상기 적어도 하나의 프로세서로 하여금,

유저 장치로부터 데이터의 제1 유저 요청을 수신하는 것에 응답하여, 상기 복수의 캐시 노드에 판독 질의(read query)을 송신하고,

상기 판독 질의에 응답하여 상기 복수의 캐시 노드로부터 제1 질의 응답 데이터(queried data)를 수신하는 것에 응답하여, 상기 제1 질의 응답 데이터를 사용하여 상기 제1 유저 요청을 처리하고,

상기 판독 질의에 응답하여 상기 복수의 캐시 노드로부터 미검출(miss)을 수신하는 것에 응답하여, 상기 제1 유저 요청에 기초하여 상기 하나 이상의 데이터베이스 시스템을 페치(fetching)하고,

상기 페치하는 것의 결과로서 상기 하나 이상의 데이터베이스 시스템으로부터 제2 질의 응답 데이터를 검색하는 것에 응답하여, 상기 제2 질의 응답 데이터를 사용하여 상기 제1 유저 요청을 처리하고, 상기 제2 질의 응답 데이터와 상기 복수의 캐시 노드에 상기 제2 질의 응답 데이터를 추가하라는 명령을 상기 복수의 캐시 노드에 송신함으로써, 상기 미검출된 판독 질의에 응답하여 상기 하나 이상의 데이터베이스 시스템으로부터 상기 복수의 캐시 노드로 상기 제2 질의 응답 데이터를 파퓰레이트하는(populate) 단계와,

상기 하나 이상의 데이터베이스 시스템으로부터 상기 제2 질의 응답 데이터를 페치하는 것과 동시에 업데이트된 데이터의 적어도 하나의 기록을 요청하는 제2 유저 요청을 수신하는 것에 응답하여, 상기 업데이트된 데이터로 상기 하나 이상의 데이터베이스 시스템을 기록하기 위한 명령을 송신하고, 상기 업데이트된 데이터로 상기 적어도 하나의 캐시 노드를 동시에 기록하기 위한 명령을 송신함으로써, 상기 데이터 저장 시스템의 각 기록 질의(write query)를 상기 복수의 캐시 노드에 파퓰레이트하고, 상기 복수의 캐시 노드에 상기 업데이트된 데이터가 저장되도록, 상기 복수의 캐시 노드에의 상기 제2 질의 응답 데이터의 후속 추가를 중지하는(aborting) 단계를 포함하는,

데이터 저장 시스템.

**청구항 23**

청구항 23은(는) 설정등록료 납부시 포기되었습니다.

제22항에 있어서,

상기 캐시 노드의 수는 상기 데이터베이스 시스템의 모든 컨텐츠를 보유하도록 구성되는 것인,

데이터 저장 시스템.

**청구항 24**

청구항 24은(는) 설정등록료 납부시 포기되었습니다.

제22항에 있어서,

상기 데이터 저장 시스템의 일부 데이터는 하나보다 많은 캐시 노드에 저장되는 것인,

데이터 저장 시스템.

**청구항 25**

청구항 25은(는) 설정등록료 납부시 포기되었습니다.

제22항에 있어서,

상기 데이터 저장 시스템은 여행 제공자의 인벤토리 시스템에 대응하는 것인,

데이터 저장 시스템.

## *발명의 설명*

### *기 술 분 야*

[0001] 본 발명은 일반적으로, 제공하는 전체 상품 및 이용가능성(availability) 수준을 추적하기 위해 제품 및 서비스의 대규모 제공자에 의해 사용되는 유형의 데이터 관리 시스템에 관한 것으로, 보다 상세하게는 적용결과 컨텐츠를 연속적(constantly)으로 데이터 저장매체에 업데이트하는 트랜잭션(transaction)의 완료에 영향을 미치지 않으면서, 지연이 없거나 매우 짧은 지연 내에 응답되어지는 데이터 저장매체의 원격-유저에 의해 발행(issued)되는 높은 레벨의 문의(inquiry)를 허용하는 시스템에 관한 것이다.

## *배 경 기 술*

[0002] 모두 상호 연결된 세계에서, 제품 및 서비스의 모든 대규모 제공자는 이제 제품 및 서비스 제공물의 특성, 명세 및 비용을 보유(hold)하는 대규모 데이터베이스 시스템을 설치하고 있다. 데이터베이스 관리 시스템(database management system: DBMS)의 제어 하에서 운영되는 컨텐츠는 전세계에 걸쳐 있는 많은 온라인 고객이 동시에 액세스가능하게 만들어진다. 이에 따라 온라인 고객이 여러 제품 및 서비스를 부킹(book)하고 구매할 수 있게 하는 특정 온라인 소프트웨어 애플리케이션을 사용하여 데이터베이스에 질문(query)하고 상거래를 완료하는 가능성이 온라인 고객에 제공된다.

[0003] 항공 산업에서, 이러한 매우 대규모 데이터베이스의 예로는 항공 회사의 상품 목록(inventory)을 보유하는 데이터베이스이다. 이러한 데이터베이스는 주어진 항공사에 의해 운영되는 항공편(flight)의 편대(fleet) 구성과 함께 실제 좌석 용량, 현재의 예약 상태를 실시간으로 추적하는데 사용된다.

[0004] 보다 정밀하게는, 항공사의 상품 목록은 통상적으로 이용가능한 좌석을 갖는 모든 항공편을 포함하고, 일반적으로 상이한 가격과 부킹 조건이 적용되는 서비스 등급(service class)(예를 들어 제1 등급, 비지니스 또는 이코노미 등급)과 많은 부킹 등급으로 분류된다. 상품 목록 관리의 핵심 기능 중 하나는 상품 목록의 조절이다. 상품 목록의 조절은 예를 들어 판매용 개별 부킹 등급을 개방하고 폐쇄하는 것에 의해 상이한 부킹 등급에서 얼마나 많은 좌석이 이용가능한지를 조절한다. 요금 견적 시스템(Fare Quote system)에 저장된 요금 및 부킹 조건과 함께 각 판매된 좌석에 대한 가격이 결정된다. 대부분의 경우에 상품 목록의 조절은 항공사의 수익 관리 시스템(Revenue Management System)과 인터페이스하며 요구사항(demand)의 변화에 응답하여 제공된 부킹 등급의 영구적인 최적화(permanent optimization)를 지원한다. 유저는 디스플레이 및 그래픽 유저 인터페이스를 구비하는 이용가능성 애플리케이션을 통해 항공사의 상품 목록에 액세스한다. 이 상품 목록은 상이한 부킹 등급에서 이용가능한 좌석이 있는 특정 도시-쌍(city-pair)에 제공된 모든 항공편을 포함한다.

[0005] 항공사 상품 목록 데이터베이스는 통상적으로 항공사에 의해 관리된다. 항공사 상품 목록 데이터베이스는 또한 항공사, 전통적인 여행사 및 모든 종류의 다른 온라인 여행 서비스 제공자를 포함하는 여행 산업의 많은 관련자(actor)에 여행 서비스를 제공하는 회사에 의해 더 설치될 수 있다. 이러한 회사로는 예를 들어 스페인의 마드리드에 본사를 둔 유럽 여행 서비스 제공자인 아마데우스(AMADEUS)가 있다. 일부 상품 목록은 직접 항공사에 의해 운영되고 글로벌 분배 시스템(global distribution system: GDS) 또는 중앙 예약 시스템(central reservation system: CRS)과 인터페이싱된다.

[0006] 이런 환경에서, 이 데이터베이스의 이용은 수 년간 크게 증가한 조회(interrogation) 또는 판독 질문 레벨로 특징지어진다. 사실, 데이터베이스가 처리하여야 하는 트랜잭션의 방문 대 부킹 비율(look-to-book ratio)이 매우 높아지고 있다. 그리하여, 여행 서비스 제공자는 이 상황에 대처하는데 필요한 컴퓨터 자원을 마련하여야 한다. 항공사의 경우에 항공 여행자에 좌석을 부킹하고 판매하는 것을 완료한 결과와 동시에 데이터베이스의 업데이트를 진행할 수 있으면서, 점점 더 증가하는 수의 온라인 고객이 데이터베이스에 효과적으로 질문하고도 여전히 신속한 응답을 얻을 수 있다.

[0007] 데이터베이스 관리 시스템 개발을 전문으로 하는 미국, 캘리포니아주, 레드우드 쇼어(Redwood Shores)에 본사를 둔 회사인 오라클(Oracle)과 같은 몇몇 전문화된 회사에서 제공하는 대규모 데이터베이스 시스템이 이 데이터베이스를 구현하는데 이용가능하고 주로 사용된다. 그러나 표준 DBMS는 단독으로는 제품 및 서비스의 대규모 서비스 제공자가 수 만 명의 잠재 고객에 동시에 서비스하여야 할 필요가 있는 것에 의해 제기되는 요구조건 레벨에 대처할 수 없다. 이 목적을 달성하기 위해, 데이터베이스는 어쨌든 직접 수신할 수 있었던 무수한 유저 질문으로부터 차단(shielded)되어야 한다.

[0008] 그리하여, 데이터베이스 컨텐츠를 캐싱(caching)하는 많은 솔루션이 개발되었다. 캐시는, 애플리케이션이 데이터베이스로부터 이전에 페치(fetch)한 데이터를 기본적으로 재사용하는, 애플리케이션 계층(application tier)에 위치된 애플리케이션 캐시일 수 있다. 이것은 데이터베이스 컨텐츠가 평균 시간(mean time)에 업데이트되었을 수 있는 것으로 인해 다른 유저 조회에 응답하여 전달되는 데이터 품질의 문제를 바로 야기한다. 이것은 데

이터베이스가 연속적으로 업데이트되어 고품질의 데이터를 요구하는 일부 응용에서 진정 문제될 수 있다. 이것은 예를 들어 데이터의 갱신(freshness)이 좌석을 판매할 가능성과 고객에 제공된 가격에 직접 영향을 미치는 항공사의 상품 목록과 관련된 응용 분야의 경우에 그러하다.

[0009] 따라서, 만약 이런 유형의 캐시에 의해 전달되는 데이터 품질이 중요하지 않고 그 밖의 것보다 더 정보 전달용인 것으로 볼 수 있는 것이 아니라면, 이런 유형의 애플리케이션 캐시는 데이터베이스에서 업데이트될 때 이전에 페치된 데이터를 무효화(invalidation)하거나 및/또는 대체(replacement)하여 애플리케이션 캐시와 데이터베이스 컨텐츠를 실제 일치(consistent)시키는 복잡한 메커니즘을 데이터베이스와 캐시 사이에 구현할 것을 요구한다. 종종, 캐시는 데이터베이스와 애플리케이션 사이 경로에 삽입되어 있어 항상 제일 먼저 애플리케이션에 의해 질문을 받는다. 질문된 데이터가 캐시에 존재하지 않는다면, 이 질문된 데이터는 데이터베이스로부터 페치되고 애플리케이션으로 전달되기 전에 캐시로 운반된다. 모든 이들 솔루션은 공통적으로 캐시와 데이터베이스가 타이트하게 결합될 것을 요구하고 서로 인식하고 있을 것을 요구한다. 그 결과, 이들 솔루션은 서비스 제공자가 트래픽 증가에 대처하기 위해 더 많은 컴퓨터 자원을 전개하고 시스템 성능을 유지하면서 더 많은 고객에 서비스하여야 할 때 용이하게 스케일러블(scalable)하지 않다.

[0010] 그러나, 다소 우수한 스케일가능성이 캐시와 데이터베이스 사이에 일부 독립성을 제공하는 특정 솔루션이 "System and method for routing database requests to a database and a cache"를 기술하는 미국 특허 6,609,126에 제시되어 있다. 개시된 솔루션에서 데이터베이스와 캐시는 애플리케이션의 제어 하에서만 별개로 구동되는 것에 의해 어쨌든 독립적으로 되어 있다. 그러나, 이 캐시는 판독 질문에 대답하는데에만 사용되는 반면 업데이트는 애플리케이션에 의해 데이터베이스에서만 수행된다. 그리하여, 데이터베이스에 일어난 변화를 캐시에 반영하기 위해 상기 특허 문헌은 데이터베이스에 포함된 복제 성분(replication component)이 캐시를 업데이트하는 것을 기술한다.

[0011] 상기의 모든 캐시 솔루션은 중요한 추가적인 작업 부하(workload)를 데이터베이스에 제공하지만 캐시와 데이터베이스가 항상 일치한다는 보장이 없고 데이터베이스는 여러 캐시를 인식하여야 한다. 이것은 새로운 캐시를 추가할 때 특정 동작이 데이터베이스에서 수행될 것을 요구하여 스케일가능성이 간단히 달성되지 못한다. 전술된 바와 같이, 미국 특허 6,609,126은 데이터베이스 관리 시스템이 외부 성분을 포함할 것을 요구한다. 이것은 표준 DBMS의 이용과 실제 호환가능하지 않게 한다.

## *발명의 내용*

### *해결하려는 과제*

[0012] 따라서 본 발명의 목적은 적절한 데이터 품질을 유저에게 제공하면서 높은 트래픽과 높은 스케일가능성을 허용하는 데이터베이스를 구비하는 컴퓨터 데이터 시스템을 기술하는 것이다.

### *과제의 해결 수단*

[0013] 본 발명의 추가적인 목적, 특징 및 잇점은 첨부된 도면을 참조하여 이하 상세한 설명을 참조하면 이 기술 분야에 통상의 지식을 가진 자에게는 명백할 것이다. 임의의 추가적인 잇점들이 본 명세서에 포함된 것으로 의도된다.

[0014] 상기 문제 및 다른 문제는 본 발명의 실시예에 따라 극복되고 다른 잇점이 실현된다.

[0015] 제1 측면에서 본 발명은, 소프트웨어 애플리케이션, 하나 이상의 데이터베이스 시스템 및 복수의 캐시 노드를 포함하는 데이터 저장 시스템에 데이터를 저장하고 상기 데이터 저장 시스템으로부터 데이터를 검색하는 방법으로서, 상기 소프트웨어 애플리케이션은 데이터의 적어도 하나의 판독(reading) 또는 데이터의 하나의 기록(writing)을 요구하는 유저 요청을 수신하도록 구성되고, 상기 소프트웨어 애플리케이션은 상기 유저 요청을 처리하기 위해 상기 데이터 저장 시스템에 판독 질문(reading query) 또는 기록 질문(writing query)을 송신하도록 더 구성되고, 상기 방법은 상기 소프트웨어 애플리케이션이 상기 하나 이상의 데이터베이스 시스템과 상기 복수의 캐시 노드와 독립적으로 인터페이싱하고, 상기 방법은 적어도 하나의 데이터 프로세서에서 상기 소프트웨어 애플리케이션에 의해 수행되는 다음의 단계, 즉 데이터의 적어도 하나의 판독을 요구하는 유저 요청을 수신할 때, 상기 소프트웨어 애플리케이션은 판독 질문만을 상기 복수의 캐시 노드에 송신하는 단계를 포함하는 것을 특징으로 하는 방법을 제공한다. 바람직하게는, 상기 소프트웨어 애플리케이션이 상기 판독 질문에 응답하여 적어도 하나의 캐시 노드로부터 질문된 데이터(즉, 검색된 데이터)를 수신하는 경우, 상기 소프트웨어 애플리케이션은 상기 질문된 데이터를 사용하여 상기 유저 요청을 처리한다. 바람직하게는, 상기 소프트웨어 애플리

케이션이 상기 판독 질문에 응답하여 모든 캐시 노드로부터 (데이터가 캐시 노드에서 발견되지 않았다는 것을 의미하는) 미검출(miss)을 수신하는 경우, 상기 소프트웨어 애플리케이션은 상기 하나 이상의 데이터베이스 시스템을 페치하고; 상기 하나 이상의 데이터베이스 시스템으로부터 상기 질문된 데이터를 검색할 때, 질문된 데이터가 데이터베이스 시스템에 존재하는 경우, 상기 소프트웨어 애플리케이션은 상기 질문된 데이터를 사용하여 상기 유저 요청을 처리하고, 상기 적어도 하나의 캐시 노드에 상기 질문된 데이터를 추가하는 명령과 상기 질문된 데이터를 적어도 하나의 캐시 노드에 송신한다.

[0016] 바람직한 실시예에 따라, 데이터의 적어도 하나의 기록을 요구하는 유저 요청을 수신할 때, 상기 소프트웨어 애플리케이션은 상기 하나 이상의 데이터베이스 시스템을 기록하는 명령을 송신하고 또한 상기 복수의 캐시 노드를 동시에 기록하는 명령을 송신하여; 상기 복수의 캐시 노드를 상기 데이터 저장 시스템의 각 미검출된 판독 질문에, 즉 질문된 데이터가 모든 캐시 노드에서 발견되지 않은 각 판독 질문에 및 각 기록 질문에 파퓰레이트(populating)한다. 그리하여 각 데이터는 상기 복수의 캐시 노드 중 적어도 하나의 캐시 노드에 및 상기 하나 이상의 데이터베이스 시스템에 동일하게 저장되어, 상기 데이터베이스 시스템과 상기 복수의 캐시 노드들이 항상 완전히 동기화되는 것을 보장한다.

[0017] 따라서, 본 발명은, 상기 데이터베이스에 통합된 복제 성분이 상기 캐시의 업데이트를 수행하여, 상기 데이터베이스와 캐시가 완전히 독립적이지 않아서 전체 저장 시스템의 스케일가능성을 제한하고 특정 데이터베이스를 요구하는 것을 수반하는 알려진 솔루션과는 달리, 상기 데이터베이스가 상기 복수의 캐시 노드를 포함하는 복수의 캐시와 완전히 독립적이게 한다.

[0018] 완전히 독립적이어서 서로를 인식하지 못하는 데이터베이스와 캐시를 구비하는 컴퓨터 데이터 시스템은 트래픽 증가에 대처하는 것이 필요할 때 더 많은 컴퓨터와 저장 용량을 간단히 가져 오는 것에 의해 데이터 시스템의 제한 없는 스케일가능성을 제공할 수 있다.

[0019] 나아가, 높은 스케일가능성은 장비의 비용을 제한하면서 달성될 수 있다. 특히, 본 발명은 표준 데이터베이스와 DBMS로 구현될 수 있다. 본 발명은 또한 유지보수 비용을 감소시킬 수 있다. 특히, 저장 자원을 증가시키는데 데이터베이스에 어떤 동작도 요구되지 않는다.

[0020] 소프트웨어 애플리케이션은 데이터베이스에 있는 데이터를 업데이트하는 일을 담당하고, 또 데이터베이스의 기록을 반영하거나 또는 데이터베이스에는 존재하지만 캐시에는 아직 존재하지 않는 질문된 데이터를 추가하는 것을 통해 캐시에 파퓰레이트하는 일을 담당하므로, 최종-유저는 고품질 데이터 즉, 가장 최근의 데이터를 제공받을 수 있다. 나아가, 캐시는 신속히 파퓰레이트될 수 있어서 새로운 캐시 노드를 시스템에 추가하자마자 처리량을 증가시킬 수 있다.

[0021] 나아가, 본 발명은 정확하고 고객에 맞춰진 답변(reply)을 유저에 제공할 수 있다.

[0022] 비 제한적인 실시예에 따라, 기록 질문은 상기 데이터베이스 시스템에 데이터를 추가하는 것, 업데이트하는 것 및 삭제하는 것 중 적어도 하나를 포함한다.

[0023] 선택적으로, 본 발명에 따른 방법은 다음의 선택적인 특징 및 단계들 중 어느 것을 포함할 수 있다:

[0024] 캐시와 데이터베이스의 데이터 모델은 동일할 수 있으나 엄격히 동일하여야 할 필요는 없다. 유일한 요구조건은 캐시와 데이터베이스 레코드에 액세스하기 위해 정확히 동일한 어드레스 키(addressing key)들이 유도될 수 있도록 이들이 일치하여야 한다는 것이다. 이 키는 또한 기록 동작의 일치(consistency)를 위해 데이터베이스 레코드를 록킹(locked)할 수 있어야 한다. 그리하여, 데이터 레코드는 데이터베이스와 (존재하는 경우) 캐시에 동일하게 저장되거나 또는 캐시와 데이터베이스에 동일한 데이터 레코드의 어드레스의 일치를 보장하는 방식으로 저장된다. 예를 들어, 캐시 데이터 모델은 데이터베이스 모델에 대해 적응될 수 있어야 데이터 검색이 촉진되어 2개의 개체들 사이에 어드레스가 완전히 일치되게 유지되면서 캐시의 액세스 시간이 개선될 수 있다.

[0025] 비 제한적인 실시예에 따라, 상기 캐시 노드의 데이터 모델은 상기 하나 이상의 데이터베이스의 데이터 모델과 동일하다. 각 캐시 노드의 각 데이터는 상기 데이터베이스 시스템에 동일하게 저장된다. 상기 데이터베이스 시스템의 각 데이터는 각 캐시 노드에 동일하게 저장된다.

[0026] 상기 하나 이상의 데이터베이스 시스템에 기록하는 명령은 상기 소프트웨어 애플리케이션에 의해 상기 하나 이상의 데이터베이스 시스템에 송신된다.

[0027] 상기 복수의 캐시 노드에 동시에 기록하는 명령은 상기 소프트웨어 애플리케이션에 의해 상기 복수의 캐시 노드

에 송신된다.

[0028] 하나의 단일 소프트웨어 애플리케이션이 상기 데이터베이스 시스템과 상기 캐시 노드에 액세스한다.

[0029] 상기 데이터 저장 시스템은 하나의 단일 데이터베이스 시스템을 포함한다.

[0030] 상기 캐시는 지속적(persistent)이지 않은 각 데이터 저장 수단을 포함하는 캐시 노드를 포함한다.

[0031] 상기 소프트웨어 애플리케이션은 상기 적어도 하나의 캐시 노드에 상기 질문된 데이터를 연속적으로 추가하는 것이 완료된 때 긍정적인 수신확인(positive acknowledgement)을 수신한다.

[0032] 상기 동일한 질문된 데이터가 상기 하나 이상의 데이터베이스 시스템으로부터 동시에 페치되는 동안 데이터의 기록이 일어나는 경우 상기 적어도 하나의 캐시 노드에 상기 질문된 데이터를 후속하여 추가하는 것이 중지(aborted)되고, 부정적인 수신확인(negative acknowledgement)이 상기 소프트웨어 애플리케이션으로 리턴되어; 상기 소프트웨어 애플리케이션이 대신 상기 기록된 데이터를 사용할 수 있게 된다.

[0033] 상기 하나 이상의 데이터베이스 시스템에 데이터를 기록하는 명령과, 상기 복수의 캐시 노드에 동시에 기록하는 명령을 송신할 때 상기 다음 단계들, 즉

[0034] 상기 하나 이상의 데이터베이스 시스템으로부터 상기 기록이 적용되는 현재 저장된 데이터를 검색하고 상기 현재 저장된 데이터를 상기 하나 이상의 데이터베이스 시스템에 록킹하는 단계;

[0035] 소프트웨어 애플리케이션에서 처리하고 저장될 새로운 데이터를 상기 하나 이상의 데이터베이스 시스템에 기록하는 단계

[0036] 저장될 상기 새로운 데이터를 일시적으로 보유하기 위해 캐시 버퍼를 소프트웨어 애플리케이션에 기록하는 단계;

[0037] 저장될 상기 새로운 데이터를 상기 적어도 하나의 캐시 노드에 전달하고 설정하는 단계;

[0038] 상기 하나 이상의 데이터베이스 시스템에 상기 트랜잭션을 커미트(commit)하는 단계가 수행된다.

[0039] 본 발명에서 캐시 노드 또는 캐시는 캐시 버퍼와는 상이하다. 캐시 버퍼는 기록하는 동안 데이터를 일시적으로 저장한다. 데이터는 유저 요청에 응답하여 캐시 버퍼로부터 검색되지 않는다. 캐시 버퍼는 기록 처리 전용이다.

[0040] 상기 커미트가 실패하는 경우, 상기 애플리케이션 소프트웨어는 이전에 설정된 상기 새로운 데이터를 삭제하는 명령을 상기 적어도 하나의 캐시 노드에 송신한다.

[0041] 상기 새로운 데이터를 포함하는 적어도 하나의 캐시 노드는 그 컨텐츠에서 이 새로운 데이터를 삭제한다. 복수의 캐시 노드가 상기 새로운 데이터를 포함하는 경우, 상기 복수의 캐시 노드의 모든 캐시 노드가 이 새로운 데이터를 삭제한다.

[0042] 소프트웨어 애플리케이션은 복수의 캐시 노드 중에서 데이터를 추가하는 명령 또는 데이터를 업데이트하거나 삭제하는 명령이 송신되는 캐시 노드 또는 캐시 노드들을 결정한다.

[0043] 이 결정은 부하 균형(load balancing)을 고려한다.

[0044] 상기 질문된 데이터가 상기 하나 이상의 데이터베이스 시스템에 존재하지 않거나 상기 복수의 캐시 노드에 존재하지 않는 경우,

[0045] 상기 하나 이상의 데이터베이스 시스템을 페치할 때 미검출이 상기 질문된 데이터 대신에 상기 소프트웨어 애플리케이션으로 리턴되고;

[0046] 상기 소프트웨어 애플리케이션은 상기 대응하는 질문된 데이터에 대해 상기 적어도 하나의 캐시 노드에 추가된 데이터의 부재(absence)를 적어도 하나의 캐시 노드에 송신하고, 상기 데이터의 부재는 모든 그 다음 질문에 바로 이용가능하게 되고;

[0047] 이에 의해, 상기 소프트웨어 애플리케이션이 상기 미검출된 질문된 데이터를 검색하려고 시도할 때 상기 하나 이상의 데이터베이스를 더 페치하여야 하는 것을 회피하게 한다.

[0048] 데이터베이스에서 종국적으로 발견되지 않는 최종-유저에 의해 요청된 데이터 유저는 "미검출 데이터"로 캐시에 저장되어, 캐시의 그 다음 조회시 유저 요청된 데이터가 캐시에도 존재하지 않고 데이터베이스에도 존재하지 않는다는 정보를 바로 리턴한다. 이것은 데이터베이스의 다른 조회가 데이터베이스 시스템을 느리게 하는 것을 방

지한다.

[0049] 하나의 비 제한적인 실시예에 따라, 각 데이터는 헤더와 연관되어 레코드를 형성하고, 여기서 상기 헤더는 적어도 하나의 데이터베이스 시스템에서 컨텐츠가 미검출되었는지 여부를 나타낸다. 따라서, 레코드의 헤더만을 판독하면 데이터베이스 시스템을 페치할 필요가 있는지 여부를 알 수 있다.

[0050] 다른 실시예에 따라, 캐시 노드는 데이터와 연관된 특정 값을 저장하는데, 상기 특정 값은 데이터가 데이터베이스에 존재하지 않는 것을 나타낸다.

[0051] 상기 소프트웨어 애플리케이션은 제1 전용 인터페이스 상의 상기 하나 이상의 데이터베이스 시스템과, 제2 전용 인터페이스 상의 상기 복수의 캐시 노드와 독립적으로 인터페이싱한다.

[0052] 데이터 모델은 데이터베이스와 캐시 사이에 직접 맵핑가능한 방식으로 선택된다.

[0053] 각 데이터 세트는 기능 개체로 그룹화되고 키에 의해 색인(indexed)되는데, 이 키는 이 키가 데이터베이스 시스템과 캐시 노드에 모두 있는 것으로 인해 데이터 세트를 전체적으로 바로 액세스가능하게 한다.

[0054] 데이터는 항공편-날짜로 그룹화되고 항공편-날짜 키에 의해 식별된다.

[0055] 소프트웨어 애플리케이션은 여행 제공자의 상품 목록의 소프트웨어 애플리케이션이다.

[0056] 소프트웨어 애플리케이션, 데이터베이스 시스템 및 캐시 노드는 여행 제공자의 상품 목록에 포함된다.

[0057] 일반적으로, 여행 제공자는 항공사이다.

[0058] 소프트웨어 애플리케이션에 수신된 유저 요청은 여행사, 온라인 여행사 및 온라인-고객 중 적어도 하나에 의해 송신된다.

[0059] 상기 캐시 노드와 상기 데이터베이스의 데이터 모델은 캐시 노드와 데이터베이스 데이터에 액세스하기 위해 정확히 동일한 어드레스 키들이 유도될 수 있도록 일치한다.

[0060] 데이터는 상기 데이터베이스와, 만약 존재하는 경우 적어도 하나의 캐시 노드에 동일하게 저장되거나 또는 캐시와 데이터베이스에 동일한 데이터의 어드레스의 일치를 보장하는 방식으로 저장된다.

[0061] 추가적인 측면에서 본 발명은 소프트웨어 프로그램 명령을 포함하는 비-일시적인 컴퓨터-프로그램 제품으로서, 적어도 하나의 데이터 프로세서에 의해 상기 소프트웨어 프로그램 명령을 실행하면 상기 방법을 실행하는 동작을 수행하는 것을 특징으로 하는 컴퓨터-프로그램 제품을 제공한다.

[0062] 또한 예시적인 실시예는, 소프트웨어 애플리케이션, 하나 이상의 데이터베이스 시스템 및 복수의 캐시 노드를 포함하는 데이터 저장 시스템에 데이터를 저장하고 상기 데이터 저장 시스템으로부터 데이터를 검색하는 방법으로서, 상기 소프트웨어 애플리케이션은 데이터의 적어도 하나의 판독 또는 데이터의 하나의 기록을 요구하는 유저 요청을 수신하도록 구성되고, 상기 소프트웨어 애플리케이션은 상기 유저 요청을 처리하기 위해 상기 데이터 저장 시스템에 판독 질문 또는 기록 질문을 송신하도록 더 구성되고, 상기 방법은 상기 소프트웨어 애플리케이션이 상기 하나 이상의 데이터베이스 시스템과 상기 복수의 캐시 노드와 독립적으로 인터페이싱하고, 상기 방법은 적어도 하나의 데이터 프로세서에서 상기 소프트웨어 애플리케이션에 의해 수행되는 다음의 단계, 즉

[0063] 데이터의 적어도 하나의 판독을 요구하는 유저 요청을 수신할 때, 상기 소프트웨어 애플리케이션은 판독 질문만을 상기 복수의 캐시 노드에 송신하는 단계;

[0064] 상기 소프트웨어 애플리케이션이 적어도 하나의 캐시 노드로부터 질문된 데이터(즉, 검색된 데이터)를 수신하는 경우, 상기 소프트웨어 애플리케이션은 상기 질문된 데이터를 사용하여 상기 유저 요청을 처리하는 단계;

[0065] 상기 소프트웨어 애플리케이션이 모든 캐시 노드로부터 미검출을 수신하는 경우, 상기 소프트웨어 애플리케이션은 상기 하나 이상의 데이터베이스 시스템을 페치하고; 상기 하나 이상의 데이터베이스 시스템으로부터 상기 질문된 데이터를 검색할 때, 상기 질문된 데이터가 상기 데이터베이스 시스템에 존재하는 경우, 상기 소프트웨어 애플리케이션은 상기 질문된 데이터를 사용하여 상기 유저 요청을 처리하고, 상기 적어도 하나의 캐시 노드에 상기 질문된 데이터를 추가하는 명령과 상기 질문된 데이터를 적어도 하나의 캐시 노드에 송신하고; 만약 데이터베이스에서 발견되지 않는 경우, 상기 데이터가 존재하지 않는다는 것을 나타내는 정보를 캐시에 추가하는 단계를 포함하고,

[0066] 각 데이터는 상기 복수의 캐시 노드 중 적어도 하나의 캐시 노드에 및 상기 하나 이상의 데이터베이스 시스템에

동일하게 저장되거나 또는 캐시와 데이터베이스에 동일한 데이터의 어드레스의 일치를 보장하는 방식으로 저장되는 것을 특징으로 하는 방법을 제공한다.

[0067] 선택적으로 그러나 유리하게는, 적어도 데이터의 기록을 요구하는 유저 요청을 수신할 때, 상기 소프트웨어 애플리케이션은 상기 하나 이상의 데이터베이스 시스템에 기록하는 명령을 송신하고 또한 상기 복수의 캐시 노드에 동시에 기록하는 명령을 송신하여; 상기 복수의 캐시 노드를 상기 데이터 저장 시스템의 각 미검출된 판독 질문에 및 각 기록 질문에 파퓰레이트한다.

[0068] 또 다른 측면에서 본 발명은, 소프트웨어 애플리케이션, 하나 이상의 데이터베이스 시스템 및 복수의 캐시 노드를 포함하는 항공사의 상품 목록의 데이터 저장 시스템에 데이터를 저장하고 상기 데이터 저장 시스템으로부터 데이터를 검색하는 방법으로서, 상기 소프트웨어 애플리케이션은 적어도 하나의 항공편에 관한 이용가능성을 획득하는 데이터의 판독과 적어도 하나의 항공편에 관한 이용가능성을 변경하는 데이터의 기록 중 적어도 하나를 요구하는 유저 요청을 수신하도록 구성되고, 상기 소프트웨어 애플리케이션은 상기 유저 요청을 처리하기 위해 상기 데이터 저장 시스템에 판독 질문 또는 기록 질문을 송신하도록 더 구성되고, 상기 방법은 상기 소프트웨어 애플리케이션이 상기 하나 이상의 데이터베이스 시스템과 상기 복수의 캐시 노드와 독립적으로 인터페이싱하고, 상기 방법은 적어도 하나의 데이터 프로세서에서 상기 소프트웨어 애플리케이션에 의해 수행되는 다음의 단계, 즉

[0069] 적어도 하나의 항공편에 관한 이용가능성을 획득하는 데이터의 적어도 하나의 판독을 요구하는 유저 요청을 수신할 때, 상기 소프트웨어 애플리케이션은 판독 질문만을 상기 복수의 캐시 노드에 송신하는 단계;

[0070] 상기 소프트웨어 애플리케이션이 적어도 하나의 캐시 노드로부터 질문된 데이터(즉, 검색된 데이터)를 수신하는 경우, 상기 소프트웨어 애플리케이션은 상기 질문된 데이터를 사용하여 상기 유저 요청을 처리하는 단계;

[0071] 상기 소프트웨어 애플리케이션이 모든 캐시 노드로부터 미검출을 수신하는 경우, 상기 소프트웨어 애플리케이션은 상기 하나 이상의 데이터베이스 시스템을 페치하고; 상기 하나 이상의 데이터베이스 시스템으로부터 상기 질문된 데이터를 검색할 때, 상기 질문된 데이터가 상기 데이터베이스 시스템에 존재하는 경우, 상기 소프트웨어 애플리케이션은 상기 질문된 데이터를 사용하여 상기 유저 요청을 처리하고, 상기 적어도 하나의 캐시 노드에 상기 질문된 데이터를 추가하는 명령과 상기 질문된 데이터를 적어도 하나의 캐시 노드에 송신하는 단계를 포함하고,

[0072] 각 데이터는 상기 복수의 캐시 노드 중 적어도 하나의 캐시 노드에 및 상기 하나 이상의 데이터베이스 시스템에 동일하게 저장되는 것을 특징으로 하는 방법을 제공한다.

[0073] 선택적으로 그러나 유리하게는, 적어도 하나의 항공편에 관한 이용가능성을 변경하는 기록을 적어도 요구하는 유저 요청은 좌석을 구매하는 것, 좌석을 취소하는 것, 좌석을 변경하는 것 중 적어도 하나에 대한 유저의 요청이다.

[0074] 또 다른 측면에서 본 발명은 하나 이상의 데이터베이스 시스템, 적어도 하나의 캐시 노드, 적어도 하나의 데이터 프로세서 및 소프트웨어 애플리케이션을 포함하는 데이터 저장 시스템으로서, 상기 적어도 하나의 데이터 프로세서에 의해 상기 소프트웨어 애플리케이션을 실행하면 상기 방법들 중 어느 방법을 실행하는 동작을 수행하고, 상기 하나 이상의 데이터베이스 시스템과 상기 적어도 하나의 캐시 노드는 상기 소프트웨어 애플리케이션에 의해 독립적으로 구동되도록 구성된 것을 특징으로 하는 데이터 저장 시스템을 제공한다.

[0075] 유리하게는 캐시 노드의 수 및 소프트웨어 애플리케이션을 실행하는 컴퓨터 수단의 처리 능력은 소프트웨어 애플리케이션의 모든 최종-유저에 의해 생성된 취합된 피크 트래픽을 충족하도록 구성된다.

[0076] 선택적으로, 본 발명에 따른 데이터 저장 시스템은 다음 선택적인 특징 및 단계들 중 어느 것을 포함할 수 있다:

[0077] 상기 캐시 노드의 수와 저장 자원은 상기 전체 데이터베이스 시스템 컨텐츠를 보유하도록 구성된다.

[0078] 상기 데이터베이스 시스템의 일부 데이터는 2개 이상의 캐시 노드에 저장된다.

[0079] 적어도 하나의 캐시 노드의 히트 비율 질문은 전체 데이터베이스 시스템 컨텐츠가 소프트웨어 애플리케이션에 의해 적어도 하나의 캐시 노드로 전달되었을 때 종국적으로 100%에 이른다.

[0080] 또 다른 측면에서 본 발명은 본 발명의 상기 데이터 저장 시스템을 포함하는 여행 제공자의 상품 목록을 제공한다.

## 도면의 간단한 설명

[0081] 도 1은 본 발명에 따른 데이터 저장 시스템을 도시하는 도면.

도 2는 최종 유저에 의해 요청되었으나 캐시에는 아직 존재하지 않는 데이터를 애플리케이션에서 종국적으로 획득할 수 있게 하는 공정을 도시하는 도면.

도 3은 애플리케이션으로부터 데이터베이스와 캐시에 동시에 기록하는 공정을 설명하는 도면.

도 4는 동시 기록 동작이 일어나는 특정 경우에 데이터베이스로부터 데이터를 캐시에 취득하는 공정을 도시하는 도면.

도 5는 애플리케이션에 의해 데이터베이스와 캐시에서 동시에 수행되는 데이터 기록 타이밍에 대한 추가적인 상세를 도시하는 도면.

도 6은 요청된 데이터가 캐시에도 존재하지 않고 데이터베이스에도 존재하지 않는 경우를 도시하는 도면.

도 7은 데이터베이스와 캐시의 기록이 삭제인 경우를 도시하는 도면.

## 발명을 실시하기 위한 구체적인 내용

[0082] 본 발명의 이하 상세한 설명은 첨부된 도면을 참조한다. 이 상세한 설명은 예시적인 실시예를 포함하지만, 다른 실시예들도 가능하고, 본 발명의 사상과 범위를 벗어남이 없이 설명된 실시예에 변형이 일어날 수 있다.

[0083] 도 1은 본 발명에 따른 데이터 저장 시스템(100)을 도시하는데, 소프트웨어 애플리케이션(10)이 한편으론 데이터베이스 시스템(20)과 독립적으로 인터페이싱하고, 다른 한편으론, 캐시라고도 언급되고 하나 이상의 캐시 노드(30)를 포함하는 캐시 시스템과 인터페이싱한다.

[0084] 이후 설명되는 본 발명의 데이터베이스 캐시 시스템은 데이터베이스 시스템(20)에 도달할 수 있었던 모든 판독 트래픽을 차단하여 데이터 저장 시스템(100)의 성능을 크게 개선시키는 프론트엔드 처리 층(front-end processing layer)으로 동작하는 캐시 노드 세트로 전체 데이터베이스 컨텐츠가 종국적으로 전달될 수 있는 것으로 인해 주로 특정된다는 것이 주목된다. 충분한 개수의 캐시 노드가 전체 트래픽을 지원하고 전체 데이터베이스 컨텐츠를 함께 처리하도록 전개된다. 그리하여, 시스템이 상당한 시간 기간 동안 업(up)되어 실행될 때 백엔드(back-end) 데이터베이스에 포함된 모든 데이터 개체가 종국적으로 캐시 노드 세트로 전달되거나 이에 존재하여 모든 판독 질문이 캐시 노드에 의해 처리되기 때문에 더 이상 캐시 미검출이 없게 된다. 데이터베이스의 기록은 캐시와 데이터베이스에서 체계적으로 수행되어 캐시와 데이터베이스 컨텐츠는 항상 일치한다. 이후 설명된 데이터 저장 시스템은 데이터 저장소로 사용되는 데이터베이스보다 더 고속의 프론트엔드 저장 및 처리 시스템이긴 하지만 본 발명의 이하 상세한 설명에서는 캐시라는 용어가 사용된다.

[0085] 데이터 저장 시스템(100)은 데이터 처리 시스템에 의해 종종 사용되는 전통적인 트리 계층 아키텍처를 따른다. 중간 계층(120)은 서비스 제공자의 전용 소프트웨어 애플리케이션(10)이 실행되는 소프트웨어 애플리케이션(10) 계층이다. 이전에 사용된 GDS 예에서 이것은 일반적으로 항공편의 항공사 편대에서 모든 좌석 예약 및 부킹을 추적하는 것을 목적으로 하는 임의의 항공사의 상품 목록 애플리케이션이다.

[0086] 클라이언트 계층(130)은 애플리케이션(10)의 모든 원격 위치된 유저(40)로 구성된다. 상기 항공사 상품 목록과 같은 서비스 제공자에 의해 설치된 여행 애플리케이션의 경우에 최종 유저는 일반적으로 전통적인 여행사의 직원이다. 이들 직원은 또한 여행 요청을 발행하고 가능하게는 온라인으로 항공 트립(trip)을 부킹할 수 있는 많은 이용가능한 여행 웹 사이트 또는 온라인 여행사 중 어느 하나를 사용하는 개인이다.

[0087] 하부 계층은 데이터베이스 시스템(20)을 포함하는 저장 계층(110)이다. 본 발명은 서비스 제공자에 의해 사용된 데이터베이스 시스템에 어떤 가정도 하지 않는다. 이것은 대부분 종종 상업적으로 이용가능한 표준 데이터베이스 관리 시스템(DBMS)에 기초하지만 이것은 전용 데이터베이스 시스템일 수도 있다. 서비스 제공자에 의해 사용되는 데이터베이스 시스템이 어느 것이든 상관없이 이 시스템은 서비스 제공자의 모든 데이터를 보유하고 처리하기에 충분한 양의 하드웨어와 소프트웨어 자원으로 구현된다. 도 1에서 데이터 저장 시스템(100)을 구현하는데 필요한 모든 하드웨어 자원은 참조 부호(101)로 전체적으로 표시된 개별 컴퓨터 같은 기계로 도시된다. 지속적인 비-휘발성 저장매체는 각 개별 컴퓨터로부터 이용가능한 것으로 가정되고 또한 예를 들어 데이터베이스 컨텐츠를 영구적으로 보유하기 위해 필요한 경우 별도의 데이터 디스크(102)로 이용가능한 것으로 가정된다.

[0088] 본 발명의 데이터 저장 시스템은 저장 계층(110)과 중간 계층(120)을 포함한다.

[0089] 본 발명에서, '유저 요청' 또는 '요청'이라는 용어는 유저(40)로부터 오고 애플리케이션(10)에 도달하는 디맨드를 나타낸다. 유저는 여행자 또는 여행사 직원과 같은 사람일 수 있고 또는 요청을 송신하는 컴퓨터 시스템일 수 있다.

[0090] 본 발명에서, '데이터 질문' 또는 '질문'이라는 용어는 애플리케이션(10)에 의해 캐시 노드(30) 및/또는 데이터베이스 시스템(20)으로 송신된 디맨드를 나타낸다. 질문은 판독 질문 또는 기록 질문일 수 있다.

[0091] 판독 질문(read query)은 적어도 캐시 노드로부터 취득하는 명령 또는 데이터베이스 시스템으로부터 데이터를 판독하는 명령을 포함한다. 일반적으로, 데이터베이스 시스템으로부터 데이터를 획득하는 동작은 '판독(read)'으로 지시된 반면, 캐시 노드로부터 데이터를 획득하는 동작은 "취득(get)"으로 지시된다. 질문된 데이터는 적어도 부분적으로 유저 요청을 충족하기 위해 적어도 취득되거나 판독되어야 하는 데이터이다.

[0092] 기록 질문(write query)은 데이터를 추가하거나 업데이트/설정하거나 삭제하는 명령을 포함한다. 일반적으로, 데이터베이스 시스템으로부터 데이터를 변경하는 동작은 '업데이트(update)'라고 지시된 반면, 캐시 노드로부터 데이터를 변경하는 동작은 "설정(set)"으로 지시된다.

[0093] 따라서, 이하 발명에서, 애플리케이션(10)은 유저 요청을 수신하고 데이터 질문을 송신하는데, 이들 질문은 판독 질문 또는 기록 질문일 수 있다.

[0094] 실제로 사용되는 시스템이 어느 것이든 상관없이, 본 발명은 데이터베이스(20)가 서비스 제공자의 최종 궁극적인 데이터 저장소인 것으로 가정한다. 데이터베이스(20)는 바람직하게는 원자성(Atomicity), 일치성(Consistency), 분리성(Isolation) 및 내구성(Durability) 면에서 신뢰성 있게 데이터베이스 트랜잭션을 처리하는 것을 보장하는 ACID (Atomicity, Consistency, Isolation and Durability) 특성 세트를 따른다.

[0095] 종래 기술에 알려지고 이전에 언급된 데이터베이스 시스템에 대해, 본 발명의 소프트웨어 애플리케이션(10)은 직접 연결된 채 유지되어서, 전용 인터페이스(12)를 통한 데이터베이스(20)와 독립적이다. 그리하여, 데이터베이스 시스템의 동작은 소프트웨어 애플리케이션(10)과 자체 전용 인터페이스(14)를 갖는 하나 이상의 캐시 노드(30)에 의해 전혀 영향을 받지 않는다. 본 발명의 이하 상세한 설명에 더 설명된 바와 같이, 반드시 처리해야만 하는 필수 트랜잭션, 즉, 새로운 부킹이 완료된 결과 및 일반적으로 예를 들어, 취소가 발생하여서, 예약 상태를 변경해야 할 때마다, 데이터베이스 컨텐츠를 영구적으로 업데이트하는 트랜잭션을 데이터베이스에 송신하는 것은 오로지 소프트웨어 애플리케이션(10)에만 달려 있다.

[0096] 따라서 캐시 노드(30)들 중 임의의 노드와 데이터베이스 시스템(20) 사이에는 연결이 없다. 데이터베이스 시스템과 캐시 노드(30) 사이에는 메시지, 명령 또는 데이터가 전혀 교환되지 않는다.

[0097] 데이터 저장 시스템(100)에서 소프트웨어 애플리케이션(10)에 의해 처리되는 모든 트래픽은 전용 캐시 소프트웨어 애플리케이션(10) 인터페이스(14)를 통해 지원된다. 도 1에 도시된 바와 같이 캐시는 기능적으로 데이터베이스와 같은 저장 계층에 위치된다. 인터페이스(14)와 하나 이상의 캐시 노드(30)는 타깃 처리량이 어느 것이든 상관없이 예상된 처리량을 충족시킬 만큼 충분한 하드웨어와 소프트웨어 자원을 소프트웨어 애플리케이션(10) 계층(120)과 캐시 노드의 저장 계층(110)에 바로 제공하고 전개하는 것에 의해 데이터 저장 시스템(100)의 모든 트래픽을 처리할 수 있는 것으로 가정된다. 그리하여, 더 많은 데이터를 처리하는 것은 기존의 것에 더 많은 연산 및 저장 자원을 추가하는 것에 의해 간단히 획득된다. 이렇게 하는 방식은 타깃 처리량을 달성하도록 전개되는데 필요한 컴퓨터 플랫폼의 수와는 다른 아키텍처 고려사항, 즉, 그 비용, 전력 소비 및 바닥 점유량에 의해 제한되지 않는 시스템 스케일가능성을 제공한다.

[0098] 상기 스케일가능성이 효과적으로 달성하기 위해, 데이터 저장 시스템(100)은 컨텐츠를 캐시와 데이터베이스에서 일치시켜 동일한 키를 사용하여 두 컨텐츠를 검색할 수 있게 하는 글로벌 키/값 데이터 모델에 기초한다. 이에 따라 데이터 모델은 데이터베이스와 캐시에 직접 맵핑가능한 방식으로 선택된다. 특히, 각 데이터 세트는 기능 개체에 의해 그룹화되고 공통 고유 키에 의해 색인된다. 이것은 이들 데이터가 데이터베이스와 캐시 모두에 있는 고유 키로부터 전체적으로 바로 액세스가능하게 하지만 컨텐츠들은 다소 상이할 수 있다. 전술된 바와 같이 동작시키는데 데이터 모델에 대한 유일한 요구조건은 다음과 같다:

[0099] - 업데이트 전에는, 캐시에서 업데이트될 데이터의 수퍼세트(superset)를 록킹하는 능력;

[0100] - 데이터베이스에서 주어진 업데이트에 의해 영향을 받는 모든 캐시 키를 추론하여 이들을 업데이트하는 능력.

[0101] 여행 산업 분야에서 취한 일반적인 예는 다음 테이블에 있는 바와 같다:

[0102]

| DB에 있는 키 | (DB에 있는) 록킹 레벨 | 캐시에 있는 키 | 캐시 키 생성 |
|---|---|---|---|
| 항공편-날짜 | 항공편-날짜 | 항공편-날짜 | DB에 있는 키와 같다 |
| O 및 D(*)-날짜범위 | O 및 D(*)-날짜범위 | O 및 D-날짜 | 날짜 범위에 있는 날마다 하나의 키 |
| 구간(**)-날짜 | 항공편-날짜 | 구간(**)-날짜 | DB에 있는 키와 같다 |
| 항공편-날짜 | 항공편-날짜 | 구간(**)-날짜 | 항공편에 있는 구간(**)마다 하나의 키 |

[0103] 여기서 (*) O와 D는 출발지와 목적지이다

[0104] (**) 구간(leg)은 항공편의 일부이다. 예를 들어, 항공편은 니스(NCE)로부터 파리(CDG)를 경유(stop)하고 뉴욕(NYC)으로 갈 수 있다. 이 항공편은 2개의 구간, 즉 NCE-CDG와 CDG-NYC를 가진다. (이 항공편은 3개의 O와 D, 즉 NCE-CDG, NCE-NYC 및 CDG-NYC를 포함하는 것이 주목된다.)

[0105] 상기 예에서 스케줄 정보는 관계 데이터베이스(relational database)에 저장된다. "모(mother)" 테이블은 항공편-날짜 기본 키를 구비한다. "자(child)" 테이블 중 하나는 구간-날짜 기본 키를 구비한다. 일부 기록(예를 들어, 업데이트)은 항공편 레벨에서 수행되고, 다른 기록은 구간 레벨에서 수행된다. 항공편 레벨에서 록킹하는 것은 두 경우에 사용된다. 이것은 항공편에 변경이 일어나는 것과 항공편의 모든 구간에 변경이 일어나는 것을 방지하는데 사용된다. 항공편의 업데이트는 모든 구간을 업데이트하고 동시 업데이트를 초래할 수 있어서 구간-날짜 레벨에서는 록킹이 설정될 수 없다.

[0106] 그리하여, 데이터베이스와 캐시의 데이터 모델은, 엄격히 동일하지 않은 경우, 데이터베이스 레코드들이 록킹되게 하면서 캐시와 데이터베이스 레코드에 액세스하기 위해 동일한 색인 키들이 유도될 수 있도록 일치하여야 한다.

[0107] 도 1에 도시된 아키텍처는 전체 처리량을 지원하고 또한 캐시 데이터의 일치의 관리를 상당히 간략화한 단일 계층 클라이언트측 분배된 캐시로 구성된 캐시와 동작한다. 클라이언트측 분배된 캐시란 캐시를 구성하는 여러 캐시 노드(30)들 중에 데이터 분배가 알려져 있고 소프트웨어 애플리케이션(10) 계층에 있는 클라이언트 측에서 연산이 일어나는 것을 의미한다. 그 결과, 모든 캐시 노드(30)는 완전히 독립적이고 시스템의 스케일가능성은 실제 잠재적으로 제한되지 않는다. 그러나, 저장 계층에 새로운 캐시 노드(30)를 추가하는 것에 의해 더 많은 처리 능력을 실제 취득하는 것은 노드들에 균형잡힌 데이터 분배가 또한 유지되는 경우에만 달성가능하다. 이 분배를 실제 균형잡기 위해, 데이터는 키 특성에 기초하여 분배된다. 예를 들어, 항공편 지향 데이터는 항공편 수에 기초하여 분배된다. 예를 들어, 이용가능한 캐시 노드의 수 또는 분배 파라미터의 수의 변화 때문에 분배 변화를 트리거할 수 있는 임의의 변경은 또한 재분배가 일어나는 동안 전체 캐시 시스템을 온라인으로 유지하고 정상 상태에서 동작하게 유지하는 적절한 재분배 절차를 통해 지원된다. 이를 위해 2개의 캐시 구성에 대해 일시적인 듀얼-피드(dual-feed)가 본 발명의 이하 설명에서 차후에 설명된다.

[0108] 본 발명의 데이터 저장 시스템(100)은 캐시와 데이터베이스 사이에 임의의 유형의 동기화 메커니즘을 요구하지 않는다. 캐시는 명시적인 방식으로 소프트웨어 애플리케이션(10)에 의해 사용되는데, 즉 동일한 유저 요청 동안, 예를 들어, 데이터베이스 또는 캐시에 기록하여야 할 때 2개의 데이터 소스, 즉 데이터베이스 또는 캐시 또는 이들 둘 모두를 사용하는 것은 소프트웨어 애플리케이션(10) 계층에 달려 있다. 이 접근법의 직접적인 효과는 데이터베이스가 캐시의 존재를 완전히 인식하지 못하고 본 발명의 데이터 구조에 캐시가 존재하는지 또는 부재(absence)하는지에 전혀 영향을 받지 않는다는 것이다. 그 반대도 또한 명백히 참이다: 캐시는 데이터베이스로부터 완전히 분리된다. 두 구조는 필요한 경우 완전히 독립적으로 전개될 수 있다.

[0109] 캐시 내 데이터 기록은 무효화 정책을 사용하지 않는 것이 필요하다. 모든 기록은 캐시 내 데이터를 바로 대체한다. 전체 데이터베이스 컨텐츠가 종국적으로 캐시에 맵핑되고 모든 이용가능한 캐시 노드(30)를 통해 분배될 때, 매우 높은 레벨의 동시 기록이 일어나는 경우에도 히트 비율(hit ratio)은 100%에 이른다.

[0110] 캐시 데이터는 항상 유효한 것으로 고려될 수 있고 이 유효성을 체크하기 위해 여분의 공정이 필요치 않다. 실

제, 모든 캐시 미검출은 미검출 값을 데이터베이스로부터 캐시로 추가하는 것을 트리거한다. 이것은 모두에 대해 한번 수행되어 데이터베이스에 최저 가능한 부하, 즉 검색할 데이터 개체마다 한번만 페치하는 것을 보장한다. 이것은 캐시가 동작하게 될 때, 예를 들어, 캐시 노드(30)의 추가 후 시스템의 전력 투입(power-on), 캐시 노드(30)의 실패, 유지보수 동작 후 등에 대부분 일어난다. 본 발명은 분배된 캐시 노드(30)에 전체 데이터베이스 컨텐츠를 수신할 공간이 충분한 것으로 가정한다.

[0111] 데이터베이스에 최종-유저에 의해 요청된 데이터의 부재가 또한 캐시에 레코드된다. 최종 유저에 의해 요청된 데이터가 캐시에서 발견되지도 않고 데이터베이스로부터 검색되지도 않는다면 이 데이터의 부재가 캐시에 레코드되어 그 다음 번에 캐시에 질문하여 대응하는 데이터의 페치가 데이터베이스로부터 시도되지 않게 하여 데이터베이스 부하를 더 제한한다.

[0112] 도 1에 설명된 아키텍처는 키-값 지향(key-value oriented)일 수 있는 임의의 유형의 데이터로 확장가능하다. 또한, 이 아키텍처는 키-값 지향일 수 있는 임의의 공정에 적용될 수 있다. 이 아키텍처는 특히 항공편 이용가능성을 체크하도록 고안된 임의의 공정에 적용될 수 있다.

[0113] 이하 도면은 데이터베이스와 캐시 사이에 소프트웨어 애플리케이션(10)에 의해 수행되어 캐시가 종국적으로 소프트웨어 애플리케이션(10)에 의해 생성된 전체 트래픽을 지원하여 모든 유저 요청을 서비스하는 동작을 설명한다.

[0114] 이전에 도시된 바와 같이 시스템의 캐시 부분은 상당히 간단하고, 기본 원격 키/값 프로토콜을 제공하는 하나 이상의 독립적인 컴퓨터로 구성된다. 소프트웨어 애플리케이션(10)이 캐시를 업데이트하고, 데이터베이스로부터 캐시를 파퓰레이트(populate)하고, 캐시로부터 데이터를 검색하는 3개의 기본 동작이 캐시에 한정된다. 이들 동작은 다음과 같다:

[0115] 설정(키, 값) : 키와 연관된 값을 캐시에 무제한 업데이트한다

[0116] 추가(키, 값) : 키가 캐시에 아직 존재하지 않을 때 키와 연관된 값을 추가한다

[0117] 취득(키) : 키와 연관된 값을 캐시로부터 리턴한다.

[0118] 본 발명은, 예상된 성능 레벨에 도달될 수 있는 한, 소프트웨어 애플리케이션(10)에 의해 실제 구현되는 방식에 어떤 가정도 하지 않는다. 유리하게는, 수 개의 기본 동작을 함께 송신하고 처리할 수 있게 하는 벌크 동작이 한정된다.

[0119] 시스템의 주요 부분은 모든 캐시 노드(30)에 대한 데이터 분배를 제어하는 소프트웨어 애플리케이션(10) 계층에 있다. 키/값 데이터는 캐시를 구성하는 노드들에 확산된다. 분배가 모든 노드에 걸쳐 가능한 한 균일하게 확산될 수 있도록 하기 위해 키의 특성이 추출되고 대응하는 캐시 노드(30)가 다음 수식으로 연산된다:

[0120] 노드_수 = 노드_수의 모듈로(MODULO)수로서의_키_특성

[0121] (node_number=key_property_as a_number MODULE the number_of_nodes)

[0122] 항공편 지향 데이터는 연속적인 항공편 번호가 동일한 특성을 가지는 항공편에 통상 사용되는 특성을 사용한다. 이 경우에 항공편 번호는 분배를 위한 기초로 직접 사용된다.

[0123] 항공편의 출발지와 목적지(O와 D)에 기초하여 항공편 지향 데이터에 대해 해쉬 값이 단독 O와 D 키에 대해 연산된다.

[0124] 이미 언급된 바와 같이, 모든 이용가능한 노드에 걸쳐 데이터 분배를 균형잡는 것은 비제한된 스케일가능성을 달성하는 실제 키이다.

[0125] 도 2 및 도 3은 소프트웨어 애플리케이션(10)의 단독 제어 하에 캐시가 파퓰레이트되고 데이터베이스 컨텐츠와 일치하게 유지되는 방식을 도시한다.

[0126] 도 2는 최종 유저에 의해 요청되고 캐시에는 아직 존재하지 않는 데이터를 소프트웨어 애플리케이션(10)에서 종국적으로 획득할 수 있게 하는 공정을 기술한다. 이런 상황은, 캐시가 파퓰레이트되고 있을 때, 예를 들어, 시스템의 전력 투입 후 또는 새로운 노드가 삽입되었거나 또는 제거되었을 때, 및 캐시 노드(30) 컨텐츠를 재균형잡는 것이 진행 중인 경우 대부분 발생한다.

[0127] 소프트웨어 애플리케이션(10)이 유저 요청에 대답할 필요가 있을 때, 제일 먼저 "취득" 동작을 통해 캐시가 판독된다(210). 항공사 상품 목록 데이터베이스의 예에서 이것은 좌석이 특정 날짜에 특정 항공편에서 특정 등급

등에서 이용가능한지를 찾기 위해 예를 들어 데이터베이스의 최종 유저에 의해 발행된 다수의 유저 요청 중 하나에 대답하는 것이다. 대응하는 데이터가 캐시에 존재하지 않는 경우, 즉, 일반적으로 대응하는 데이터가 이전의 판독에 의해 아직 캐시에 있지 않은 경우, 캐시는 "미검출"을 소프트웨어 애플리케이션(10)으로 리턴한다(220). 그렇지 않은 경우, 캐시로부터 소프트웨어 애플리케이션(10)으로 정보가 명백히 바로 리턴되고 "취득" 동작은 종료된다. 소프트웨어 애플리케이션(10)은 최종-유저의 유저 요청을 충족할 수 있다. 종국적으로, 이 애플리케이션은 질문된 데이터를 추가적인 데이터와 취합하여 이를 최종-유저로부터 온 요청에 응답하여 리턴한다. 추가적인 데이터는 일반적으로 유저 요청을 충족하도록 검색될 필요가 있을 수 있는 다른 데이터이다. 예를 들어, 일부 데이터는 캐시 노드로부터 취득될 수 있는 반면, 또한 동일한 유저 요청을 충족하는데 필요한 다른 데이터는 다른 캐시 노드로부터 취득되거나 및/또는 데이터베이스 시스템(20)으로부터 판독되어야 한다.

[0128] 질문된 데이터가 캐시에 존재하지 않는 정보를 수신할 때, 소프트웨어 애플리케이션(10)은 "판독" 동작에서 데이터베이스를 조회한다(230). 미검출 정보가 소프트웨어 애플리케이션(10)으로 리턴된다(240). 데이터베이스로부터 데이터를 판독하는 것은 이전에 설명된 데이터베이스 전용 인터페이스(12)에서 일어난다. 이것은 본 발명의 데이터 저장 시스템(100)에 의해 사용되는 데이터베이스 관리 시스템(DBMS)으로 대응하는 질문을 소프트웨어 애플리케이션(10)으로부터 발행하는 것에 의해 수행된다.

[0129] 캐시에 미검출된 데이터를 데이터베이스로부터 수신할 때 소프트웨어 애플리케이션(10)은 데이터를 캐시에 저장하는 "추가" 동작을 수행한다(250). 이 시간부터, 데이터는 캐시가 동작하는 한, 캐시에 존재하고(270) 재구성되지 않는다. 이 동작이 완료할 때 긍정적인 수신확인(positive acknowledgement: OK)(260)이 소프트웨어 애플리케이션(10)으로 리턴된다.

[0130] 이 공정은 캐시가 데이터베이스와 캐시 노드(30)에 동일하거나 일치하게 저장된 임의의 주어진 데이터에 대해 업되어 실행되는 동안 한번만 일어난다는 것이 주목된다. 이것은 데이터가 소프트웨어 애플리케이션(10)에 의해 요청되되 캐시에는 아직 존재하지 않는 제1 시간에 일어난다. 이후 대응하는 데이터는 예를 들어, 항공사 좌석이 판매된 것으로 인해 데이터베이스 컨텐츠가 변화될 필요가 있는 경우 업데이트될 수 있다. 이 경우에, 이후 설명되는 바와 같이, 소프트웨어 애플리케이션(10)은 캐시와 데이터베이스를 모두 업데이트하여 도 2의 공정을 재실행할 필요가 전혀 없다.

[0131] 도 3은 소프트웨어 애플리케이션(10)으로부터 데이터베이스와 캐시를 동시에 업데이트하는 공정을 기술한다.

[0132] 데이터베이스와 캐시 컨텐츠를 항상 일치시켜 유지하기 위해, 소프트웨어 애플리케이션(10)은 항상 캐시와 데이터베이스를 모두 업데이트한다. 캐시의 업데이트는 이전에 설명된 "설정" 동작에서 수행된다(310). 동시에, 데이터베이스의 "업데이트"(305)는 사용 중인 DBMS의 질문 언어를 사용하여 수행된다. 이 업데이트는 동작이 애플리케이션에 의해 데이터베이스에 커미트(320)된 후에 유효하다.

[0133] 보다 정밀하게는, 이 설정은 업데이트가 데이터베이스에 수행될 때에는 수행하지 않으나 커미트(commit)가 수행될 때에는 수행된다. 애플리케이션은 커미트가 수행될 때까지 데이터를 메모리에 설정된 채 유지한다. 업데이트(305)와 설정(310) 사이에 다수의 단계들이 있을 수 있다. 그러나, 설정(310)과 커미트(320)는 일렬로 수행되도록 의도된다.

[0134] 정상 상태에서, 즉, 시스템이 업되어 상당한 시간 기간 동안 실행한 후에는, 데이터베이스의 전체 컨텐츠가 종국적으로 모든 캐시 노드(30)로 와서 이들에 분배되고; 이후 업데이트 동작, 즉, 컨텐츠 업데이트, 삽입 및 삭제는 데이터베이스 인터페이스에 수행될 필요가 있는 유일한 동작이어서 데이터베이스 부하를 훨씬 더 낮춘다. 캐시에 있는 대응하는 데이터의 무효화(nullification)를 트리거하는 삭제 동작의 경우는 도 7에 설명된다.

[0135] 또한, 도 3의 공정은 임의의 특정 조건이 충족되어야 캐시에 기록될 필요가 있는 것을 가정하지 않으므로 본 발명의 캐시는 판독 및 기록 동작으로부터 모두 파퓰레이트된다는 것이 주목되어야 한다. 이것은 판독만을 사용하여 캐시를 파퓰레이트하는 시스템에 비해 전력 투입 후 캐시 노드(30)의 파퓰레이트를 촉진하는데 상당히 기여한다. 이것은 이미 언급된 바와 같이 데이터베이스와 캐시에 저장된 데이터 개체가 모두 업데이트된 채 유지되기 때문에 가능하고 이에 의해 간단하게 수행되는 것으로서, 이는 소프트웨어 애플리케이션(10)으로 전달된 캐시 데이터 개체가 데이터베이스의 여러 부분으로부터 추출된 데이터를 해체하여 형성될 때 또는 일반적으로 캐시 저장 요구조건을 최소로 유지하려는 시도에서 데이터베이스와 캐시 컨텐츠가 상당히 상이할 수 있는 다른 캐시 솔루션에서는 가능하지 않은 것이다.

[0136] 도 4는 데이터베이스의 동시 기록(예를 들어, 업데이트)이 소프트웨어 애플리케이션(10)에 의해 요청되어 실행

을 간섭하는 특정 경우에 도 2의 공정을 기술한다.

[0137] 도 2에 설명된 것과 동일한 방식으로 이 공정은 캐시로부터 데이터를 "취득"하는 것으로 시작하고(210) 이어서 데이터베이스로부터 미검출 데이터의 페치를 트리거하는 "미검출"(220)이 후속한다(230). 그러나, 미검출 데이터는 정상적으로 소프트웨어 애플리케이션(10)으로 리턴되지만(240), 동일한 데이터에 대한 기록 질문(410)이 또한 소프트웨어 애플리케이션(10)에 의해 수신된다. 이 기록은 도 3에 설명된 바와 같이 수행된다. 이 기록은 "설정" 동작(310)에서 캐시에 수행되고, "업데이트" 동작(305)에서 데이터베이스에 수행된다. 대응하는 데이터는 "설정"이 캐시로 발행될 때에는 바로 이용가능(420) 하게 되고, "커미트"(320)가 송신될 때에는 데이터베이스에서 이용가능하게 된다. 설정이 트리거되기 전에, 애플리케이션은 데이터를 메모리에 유지한다("메모리에 설정").

[0138] 이후, 이 특정 경우에, 캐시 컨텐츠는 데이터베이스로부터 미검출 데이터의 페치(230)로 인해 초래되는 후속 "추가"(250)에 의해 더 업데이트될 필요가 없는데 그 이유는 이 미검출 데이터의 페치가 평균 시간에 업데이트되었기 때문이다. 이후 "추가"(252)는 실제로 중지된다. 캐시의 업데이트가 "추가" 동작에 의해 실제 수행되지 않았다는 것을 소프트웨어 애플리케이션(10)에 알게 하는 부정적인 수신확인(negative acknowledgement: KO)(262)이 리턴된다.

[0139] 따라서, 데이터베이스에서 판독된 데이터를 캐시에 업데이트하기 위해, 본 발명은 추가 커맨드를 사용하여 데이터베이스에 있는 데이터를 록킹할 필요 없이 캐시에 데이터를 송신할 수 있다. 실제, 데이터를 추가하려고 시도할 때 이 데이터가 캐시에 여전히 없는 경우, 이 데이터는 효과적으로 추가될 수 있다. 캐시가 업데이트 공정에 의해 그 동안 업데이트되었다면, 추가가 실패할 수 있으나 이것은 예상된다: 업데이트 공정은 데이터베이스를 록킹하여 이 키에 대한 업데이트를 우수(primacy)하게 하여, 그리하여 이것이 캐시에 있는 것이 일반적이다.

[0140] 본 발명의 이들 특징은, 특히 데이터가 데이터베이스에서 한 번을 초과하여 판독되지 않아서 데이터베이스에 최저 가능한 부하를 야기하는 것을 여전히 보장하면서, 데이터베이스 시스템과 캐시가 서로 록킹하거나 서로의 성능에 영향을 미치지 않아서 업데이트 공정과 매우 원활한 통합을 가능하게 한다.

[0141] 도 5는 소프트웨어 애플리케이션(10)에 의해 데이터베이스와 캐시에 동시에 수행된 데이터 업데이트의 타이밍에 대한 추가적인 상세를 제공한다.

[0142] 소프트웨어 애플리케이션(10)은 현재 저장된 값을 검색하는 대응하는 질문(510)을 데이터베이스에 발행하는 것으로 업데이트 트랜잭션을 시작한다. 동시에, 다른 소프트웨어 애플리케이션(10)으로부터 동시 업데이트가 발생하는 것을 방지하기 위해, 데이터베이스 관리 시스템(DBMS)은 현재 저장된 값을 록킹한다. 소프트웨어 애플리케이션 계층에서, 데이터는 소프트웨어 애플리케이션(10)에 의해 처리된다. 데이터가 DBMS에 의해 이미 업데이트(530)될 준비가 될 때 소프트웨어 애플리케이션(10)의 버퍼 캐시(540)에 업데이트가 또한 수행되어 캐시에 저장되거나 전달될 새로운 데이터를 보유한다.

[0143] 이후, 소프트웨어 애플리케이션(10)은 변경(550)을 커미트할 수 있는데 이는 "설정" 동작에서 캐시(552)에 바로 수행되고 또 데이터베이스(554)로 커미트될 수 있다. 이에 따라 새로운 데이터는 데이터베이스에서 실제 커미트되어 이용가능하기(558) 얼마 전에(556) 캐시에서 먼저 이용가능하다는 것을 알 수 있다. 참조 번호(556)는 데이터베이스 시스템(20)에서는 아직 이용가능하지 않지만 업데이트가 캐시에서 최종 유저에 이용가능하게 되는 시간 프레임을 도시한다.

[0144] 어떤 이유로, 예를 들어, 하드웨어 및/또는 소프트웨어 실패 때문에, 캐시에서 이전의 기록 동작을 정상적으로 완료하기 위한 커미트가 실패하면, 즉, "설정" 동작(552)은 롤백(rolled back)되어 캐시 컨텐츠가 변경 없이 남아 있게 된다. 그리하여, 커미트가 실패하면, "커미트 KO"(560)가 애플리케이션에 제기되고 이 애플리케이션은 삭제(562)를 캐시로 발행하여 추가된 데이터를 제거한다. 그 결과, 잘못된 값이 평균 시간(564) 동안 캐시에 존재한다.

[0145] 따라서 최고 성능의 비 데이터베이스와 관련된 및 영향을 받은 데이터 품질이 캐시에 제공되고: 업데이트는 "커미트 요청" 데이터로 기록 선행 커미트(write ahead commit)를 사용하여 캐시로 전파된다. 캐시된 데이터에 대해 연기된 제약이 금지되면, 이것은 추가 비용 없이, 특히 통상적인 2 단계의 커미트 아키텍처의 매우 높은 비용 없이, 캐시에 있는 데이터를 데이터베이스보다 미리 "나쁘게(at worse)"만든다. 이러한 품질은 이용가능성 요청에 대한 데이터 품질 요구조건을 충족시키고 심지어 최종 클라이언트 관점으로부터 유리한 것으로 고려될 수 있다.

[0146] 도 6은 질문된 데이터가 캐시에도 존재하지 않고 또 데이터베이스에도 존재하지 않는 경우를 기술한다. 이것은

최종-유저가 데이터베이스에 보유하지 않는 정보를 요청하는 경우를 포함한다.

[0147] 이러한 상황이 발생할 때, 데이터베이스의 추가적인 조회를 방지하기 위해, 대응하는 데이터의 부재가 또한 캐시에 레코드된다. 이후, 그 다음 번에 캐시가 소프트웨어 애플리케이션(10)으로부터 조회될 때 질문된 데이터가 데이터베이스에 존재하지 않는다는 정보가 캐시 자체에 의해 직접 전달되어 데이터베이스 부하를 더 감소시킨다.

[0148] 공정은 도 2에 설명된 것과 유사하다. 캐시에 발행된 "취득" 동작(210)이 "미검출"(220)을 리턴한 후, 데이터베이스에서 대응하는 데이터의 판독(230)이 또한 소프트웨어 애플리케이션(10)으로 데이터베이스 "미검출"(640)을 리턴한다. 이후, 데이터의 부재가 캐시에 추가(650)된다. 데이터와 같이, 데이터의 부재는 캐시에 비러 이용가능하게(270) 되고 이 캐시는 또한 수신확인(260)을 소프트웨어 애플리케이션(10)으로 리턴한다.

[0149] 비 제한적인 실시예에 따라, 각 데이터는 헤더와 연관되어 레코드를 형성하는데, 여기서 이 헤더는 컨텐츠가 데이터베이스 시스템(20)에서 미검출되었는지 여부를 나타낸다. 따라서, 레코드의 헤더만을 판독하면 데이터베이스 시스템을 페치할 필요가 있는지 여부를 알 수 있다. 대안적인 실시예에 따라, 캐시 노드는 데이터와 연관된 특정 값을 저장하는데, 여기서 상기 특정 값은 데이터가 데이터베이스에 존재하지 않는 것을 나타낸다. 따라서, 레코드의 값만을 판독하면 데이터베이스 시스템을 페치할 필요가 있는지 여부를 알 수 있다.

[0150] 도 7은 애플리케이션으로부터 특정 업데이트 동작이 데이터베이스로부터 데이터를 삭제(705)하는 것인 도 3에 이미 언급된 경우를 도시한다. 이 동작은 삭제된 데이터가 캐시로부터 실제 제거되지 않고 "데이터의 부재"의 지시자로 대체된 것을 제외하고는 도 3에 설명된 바와 같이 전체적으로 수행된다. 삭제가 애플리케이션에 의해 데이터베이스에 커미트(320)될 때 대응하는 정보는 특정 "설정" 동작(310)에서 캐시에 저장된다. "데이터의 부재"는 즉시 이용가능(330) 하게 된다. 그리하여, 이전에 설명된 바와 같이, 캐시가 차후에 조회되면, 이 캐시는 요청된 데이터가 더 이상 캐시에도 존재하지 않고 데이터베이스에도 존재하지 않는다는 정보를 직접 제공할 수 있다.

[0151] 이하는, 예를 들어, 트래픽 증가에 대처하기 위해 본 발명의 데이터베이스 시스템의 구성을 변경하는데 필요한 경우를 설명한다. 여분의 캐시 노드를 추가하여 도 1에 도시된 시스템 구성을 확장하여 더 많은 캐시 저장 용량을 제공하고 더 많은 수의 처리 노드를 통해 증가하는 트래픽을 분배할 수 있다. 그러나, 더 많은 수의 노드를 가지고, 일반적으로 말하면, 능동 노드의 수가 변경되어야 할 때마다, 전체 트래픽이 새로운 전체 노드 세트에 걸쳐 실제 균일하게 확산될 수 있도록 노드에서 데이터를 고유하게 어드레스하는 키가 재연산되어야 한다.

[0152] 본 발명은 데이터베이스와 캐시로부터 동일하게 저장되고 검색되는 데이터 개체로부터 키를 연산하는 방식을 임의의 특정 방식으로 가정하지 않는다. 대부분, 특정 애플리케이션에 의해 처리되는 데이터의 유형에 따라, 일부 해쉬 함수가 사용되고 노드 어드레스가 해쉬 키 모듈로(modulo) 노드의 수를 추가적으로 연산하는 것에 의해 해쉬 키로부터 바로 유도된다. 그리하여, 노드의 수가 변경된 경우, 새로운 구성의 상이한 노드에서 특정 데이터 개체를 검색하여 찾을 필요가 있기 때문에 상이한 결과가 획득된다. 문제는 구성 업데이트가 원자적(atomic)이 아니고 데이터베이스 시스템이 완전히 동작하는 동안 투명하게 수행되어야 한다는 것으로부터 발생한다. 모든 캐시 클라이언트들이 동시에 새로운 구성을 인식하게 되는 것은 아니다. 이것은 일부 데이터 기록이 새로운 구성에 기초하여 수행될 수 있는 반면 다른 데이터 기록은 여전히 오래된(old) 구성을 사용할 수 있다는 것을 의미한다. 그 결과 캐시와 데이터베이스 사이에 데이터 세트의 불일치가 있을 수 있다.

[0153] 본 발명은 "듀얼-피드"라고 언급되는 절차를 인에이블하는 것에 의해 이것을 처리한다. 듀얼 피드는 통상적으로 캐시에 사용되는 것에 더하여 하나의 여분의 구성, 그리하여 "듀얼-피드"라는 이름을 유지하는 것에 있다. 여분의 구성은 디폴트로 사용되는 것이 아니고 구성 변경시에 활성화될 수 있다. 구성이 활성화될 때, 모든 기록 동작은 표준 구성과 듀얼-피드 구성으로 송신된다. 존재 시간(time-to-live: TTL)은 캐시 내 각 항목과 연관된 특성이다. 이름이 암시하는 바와 같이, 이 이름은 유효한 시간 기간 항목에 대응한다. 이 시간 기간이 만료하면, 이 항목은 캐시로부터 더 이상 검색될 수 없어서, 데이터가 미검출된 것처럼 캐시 미검출을 초래한다. 이것은 구성에 의해 설정될 수 있다: 표준 구성을 위한 것과 듀얼-피드 구성을 위한 것으로 설정될 수 있다. 존재 시간이 설정되지 않은 경우, 이 항목은 만료되지 않는다.

[0154] 듀얼-피드 구성의 활성화는 원자적이 아니므로, 이 듀얼 피드는 짧은 존재 시간에 제일 먼저 활성화되어야 한다. 듀얼 피드 구성이 완전히 활성화되면, 존재 시간은 제거될 수 있다. 일단 존재 시간이 만료된 경우에만 표준 구성과 듀얼-피드 구성이 교환(swapped)될 수 있다. 구성 변경이 종료되면, 듀얼 피드는 비활성화될 수 있

다. 구성이 전파되고 있는 단계(듀얼-피드의 활성화/비활성화) 동안, 일부 "무효" 데이터가 판독되지 않는 곳에서만 기록될 수 있다. 따라서, 절차는 다음과 같다:

[0155] - 짧은 TTL에서 듀얼-피드 구성을 생성한다

[0156] - 듀얼-피드 구성의 활성화하고, 그 전파를 기다린다

[0157] - 듀얼-피드 구성으로부터 짧은 TTL을 제거한다

[0158] - 표준 구성과 듀얼-피드 구성을 교환하고, 그 전파를 기다린다

[0159] - 듀얼-피드를 비활성화한다

[0160] 온라인 방식으로 시스템 상의 임의의 변경을 허용하는 절차 세트는 아래에 설명된다.

[0161] 제안된 아키텍처는 전체 시스템이 캐시가 없으면 적절히 동작하는 위치에 차후 있지 않을 수 있는 스케일가능성을 제안한다. 이러한 상황을 처리하기 위해, 본 발명의 일 실시예에 따라, 모든 유지보수 동작은 온라인으로 수행되는 것을 의미하고, 한번에 기껏 하나의 노드(또는 트래픽의 등가 비율)에 영향을 미처 (예를 들어, 하나씩 이루어진 캐시 노드 업그레이드 또는 대체하거나, 듀얼 피드 메커니즘을 사용하여 수행된 글로벌 캐시 변경을 수행하여) 데이터베이스에 대한 충격을 저하시키는 것이 제안된다.

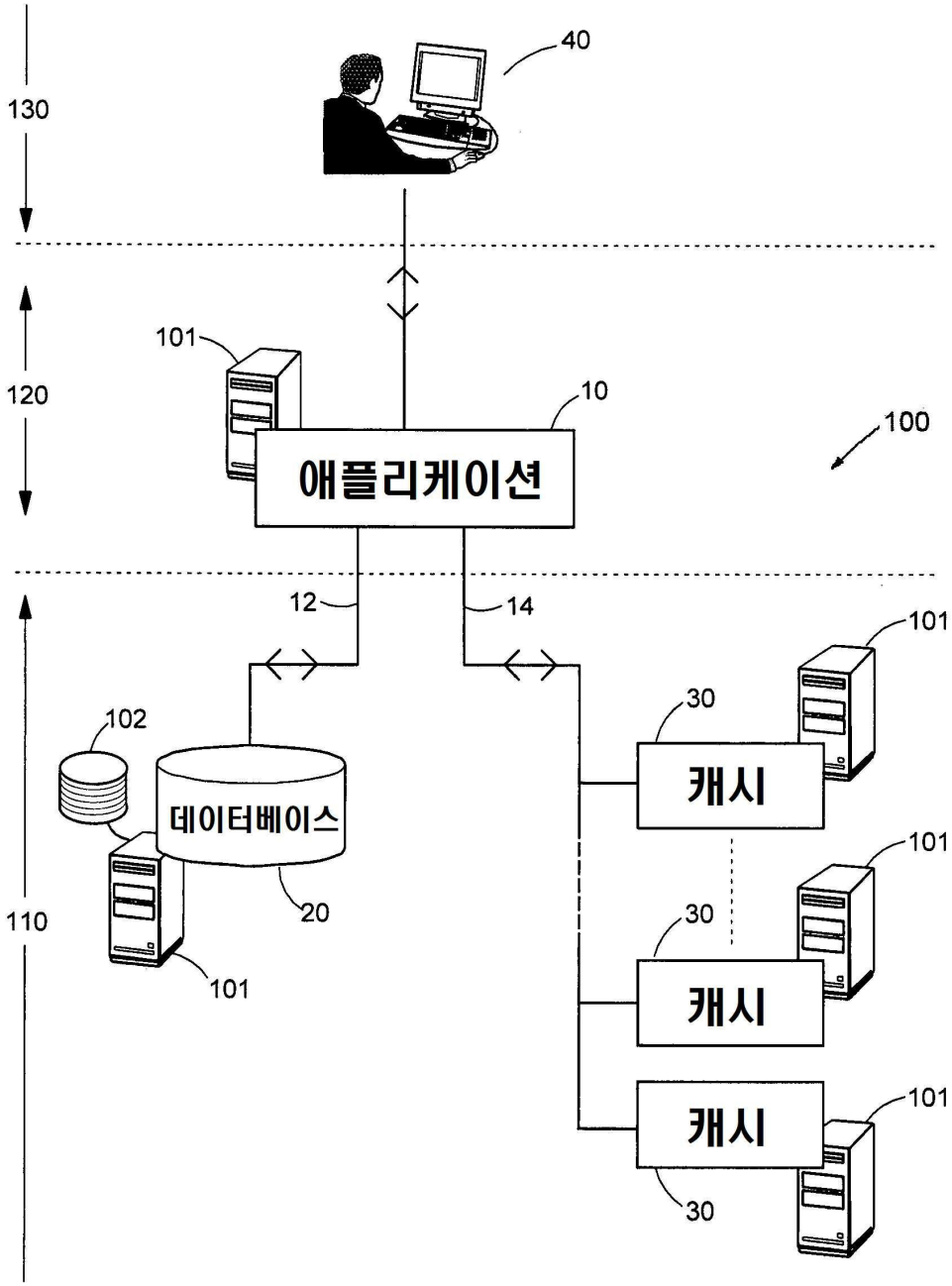
[0162] - 캐시 노드 업그레이드 또는 대체는 하나씩 이루어진다. 시스템은 바람직하게는 데이터베이스를 사용하여 이 노드에 의해 호스팅되었어야 하는 데이터를 검색한다.

[0163] - 글로벌 캐시 변경, 일반적으로, 이전의 문단에서 설명된 듀얼 피드 메커니즘을 사용하여 수행되는 글로벌 분배를 크게 변경시킬 수 있는 복수의 캐시 노드를 추가하거나 제거하거나 변경한다.

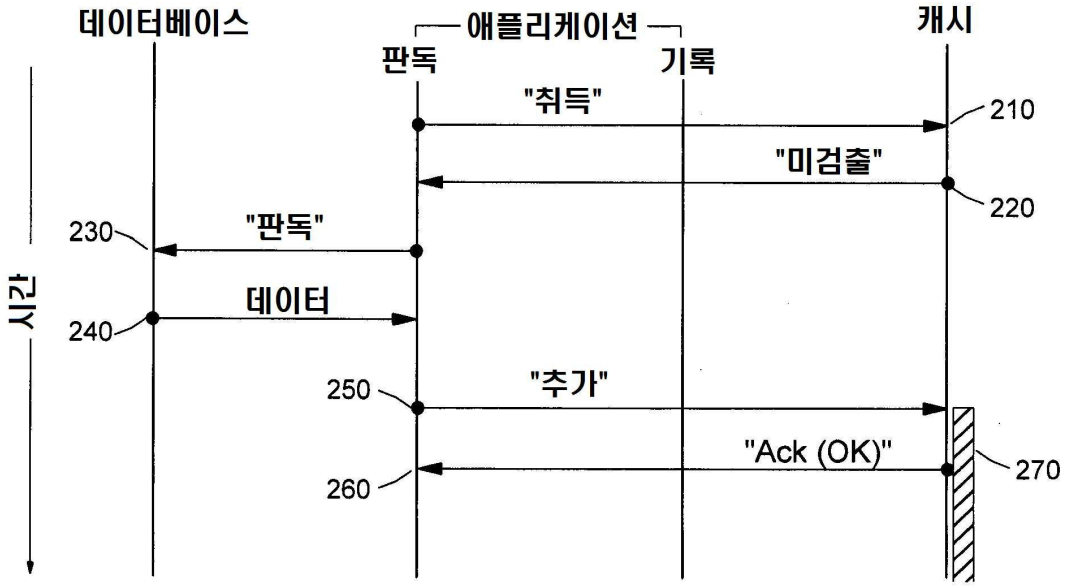
[0164] 상기 상세한 설명으로부터 본 발명은, 비-엄격히 말하면, ACID에 순응하면서도 매우 스케일가능하고, 데이터베이스에 영향을 주지 않아, 100% 히트 비율을 허용하고, 무엇보다도 특히 데이터 품질 요구를 완전히 충족시킬 수 있는 메커니즘으로 인해 캐시와 데이터베이스 사이에 데이터를 일치시킬 수 있다는 것이 명백히 이해된다. 나아가, 본 발명은 캐시의 오프로드 효과로부터 여전히 이익을 얻으면서도, 일반적으로 단일 데이터(unitary data)마다 초당 수 십 개의 기록까지 매우 동적으로 데이터를 캐싱할 수 있다.

도면

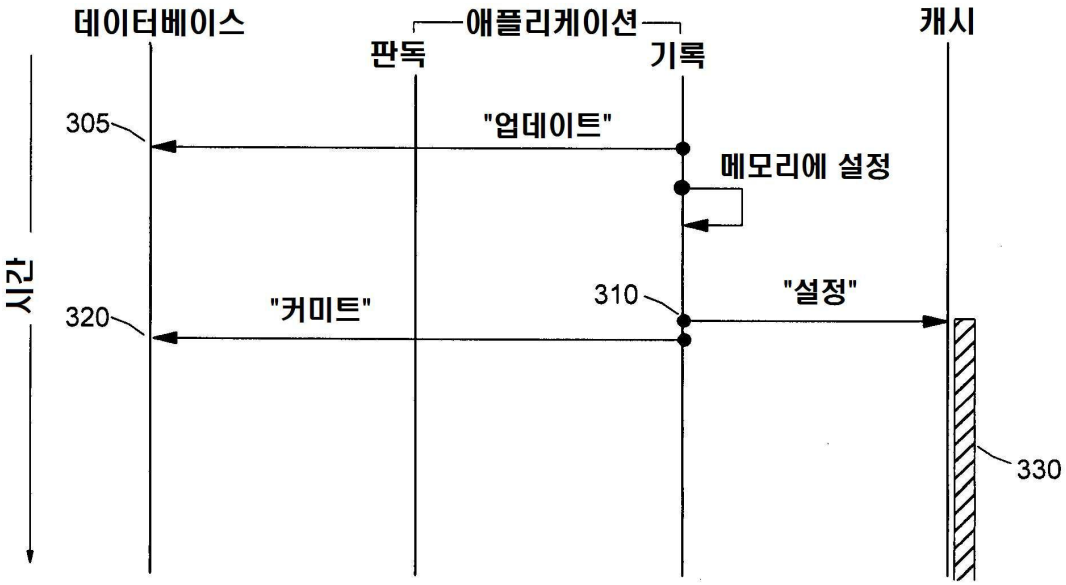
도면1


130
120
110
40
101
10
애플리케이션
100
12
14
102
데이터베이스
20
101
30
캐시
101
30
캐시
101
30
캐시
101

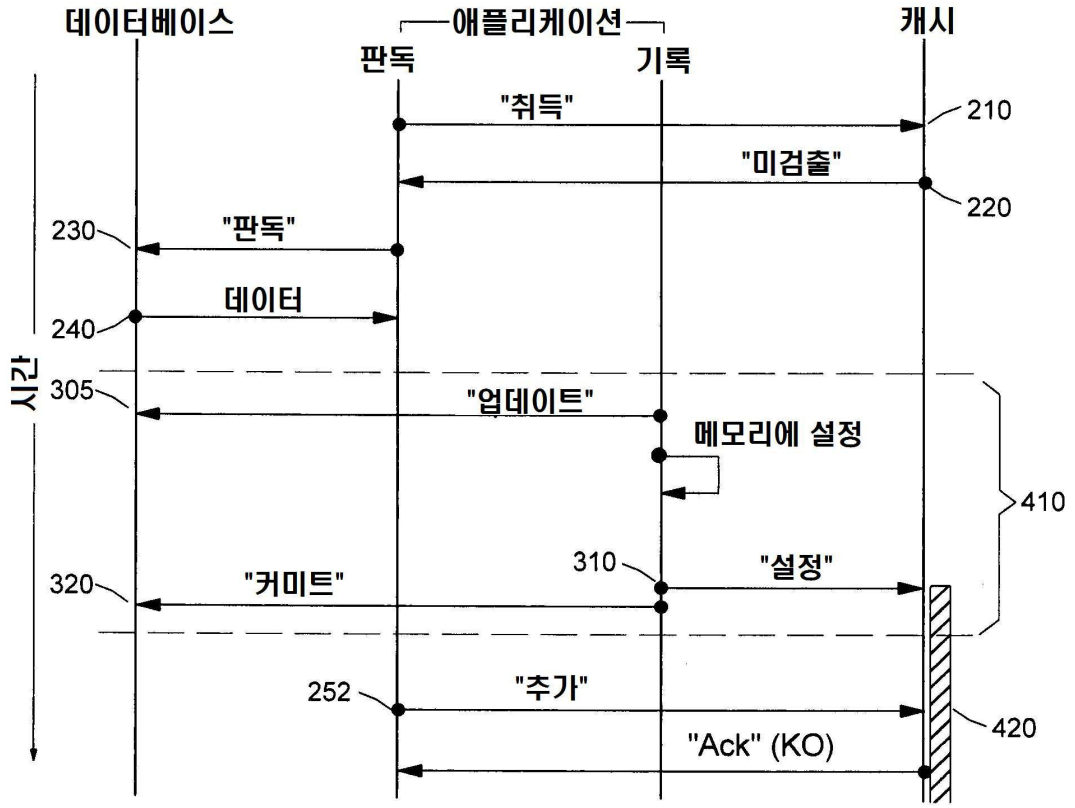
도면2


데이터베이스
애플리케이션
판독
기록
캐시
"취득"
210
"미검출"
220
230
"판독"
시간
데이터
240
250
"추가"
"Ack (OK)"
270
260

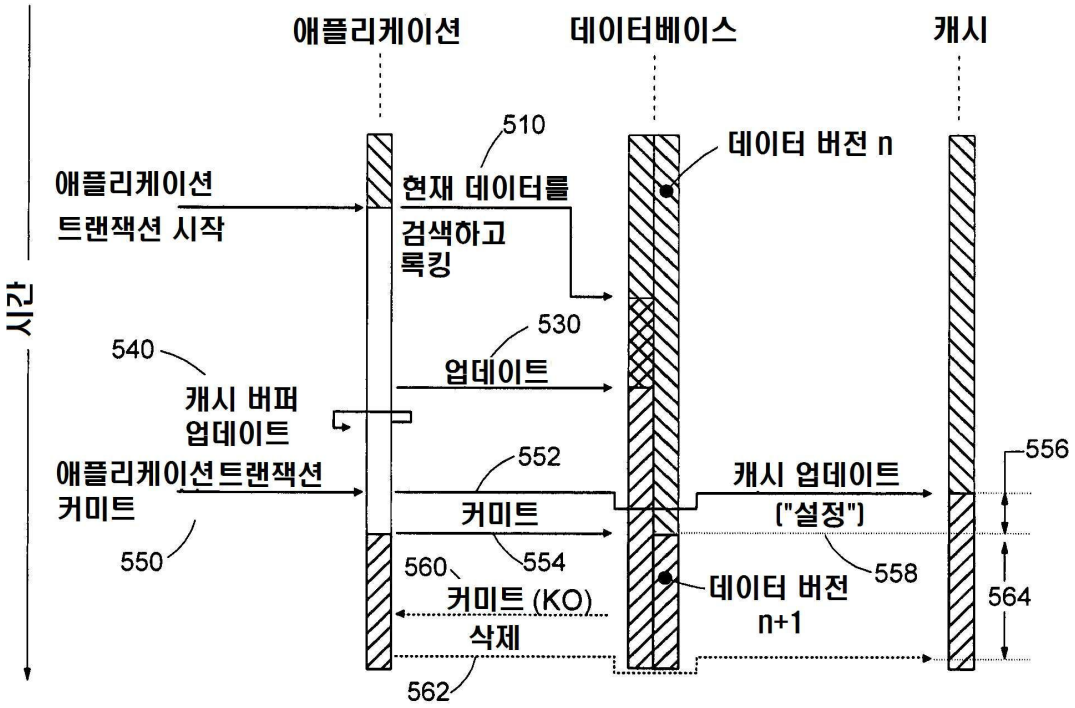
도면3


데이터베이스
애플리케이션
판독
기록
캐시
305
"업데이트"
메모리에 설정
시간
310
"설정"
320
"커미트"
330

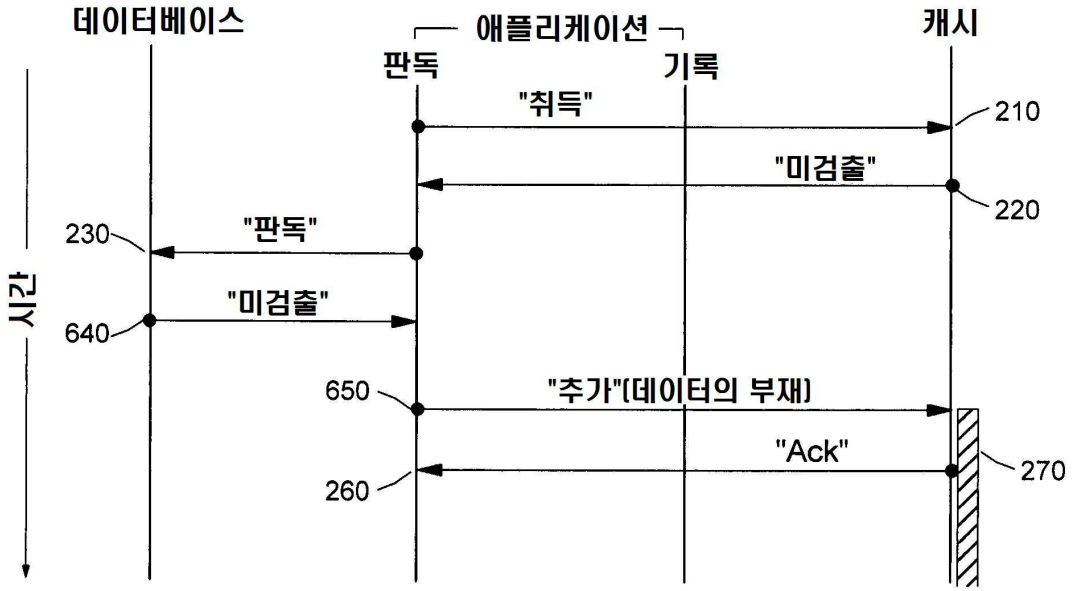
도면4


데이터베이스
애플리케이션
판독
기록
캐시
"취득"
210
"미검출"
220
230
"판독"
데이터
240
시간
305
"업데이트"
메모리에 설정
410
310
"설정"
320
"커미트"
252
"추가"
420
"Ack" (KO)

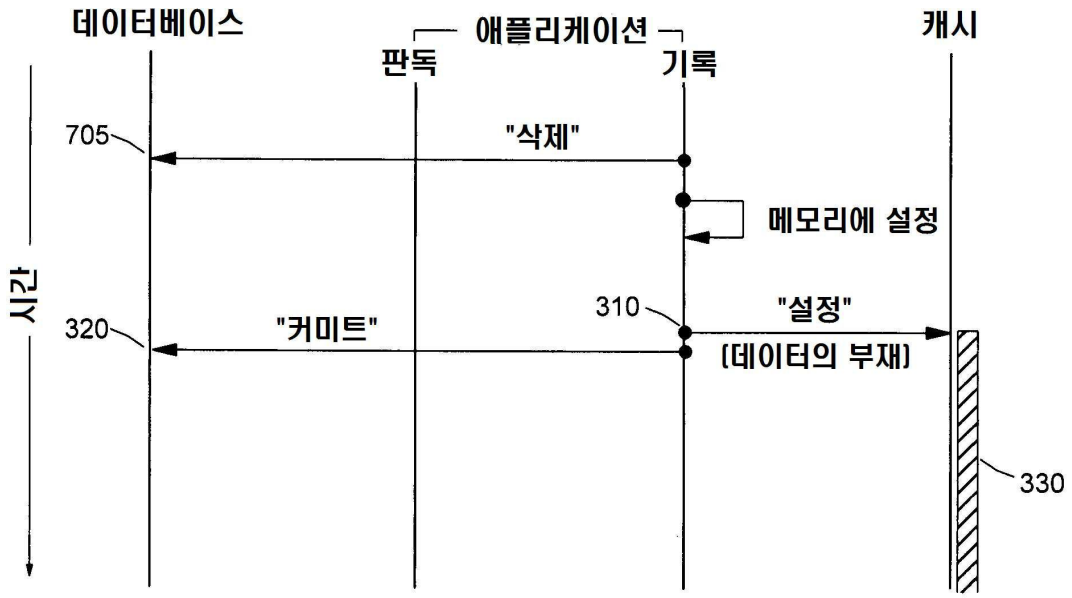
도면5


애플리케이션
데이터베이스
캐시
510
데이터 버전 n
애플리케이션
트랜잭션 시작
현재 데이터를
검색하고
록킹
530
540
업데이트
캐시 버퍼
업데이트
552
556
애플리케이션트랜잭션
커미트
캐시 업데이트
["설정"]
커미트
550
554
558
560
데이터 버전
n+1
564
커미트 (KO)
삭제
562
시간

도면6


데이터베이스
애플리케이션
판독
기록
캐시
"취득"
210
"미검출"
220
230
"판독"
시간
"미검출"
640
650
"추가"(데이터의 부재)
"Ack"
270
260

도면7


데이터베이스
애플리케이션
판독
기록
캐시
705
"삭제"
메모리에 설정
시간
310
"설정"
320
"커미트"
(데이터의 부재)
330