(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2022년09월08일
(11) 등록번호 10-2441299
(24) 등록일자 2022년09월02일

(51) 국제특허분류(Int. Cl.)
G06F 16/23 (2019.01) *G06F 16/22* (2019.01)
G06F 16/25 (2019.01)
(52) CPC특허분류
G06F 16/2386 (2019.01)
G06F 16/2282 (2019.01)
(21) 출원번호 10-2020-7017339
(22) 출원일자(국제) 2018년11월27일
심사청구일자 2020년07월01일
(85) 번역문제출일자 2020년06월16일
(65) 공개번호 10-2020-0103661
(43) 공개일자 2020년09월02일
(86) 국제출원번호 PCT/US2018/062652
(87) 국제공개번호 WO 2019/104338
국제공개일자 2019년05월31일
(30) 우선권주장
62/591,118 2017년11월27일 미국(US)
(56) 선행기술조사문헌
US20150026114 A1*
(뒷면에 계속)

(73) 특허권자
스노우플레이크 인코포레이티드
미국 몬태나주 59715 보즈먼 이스트 밥콕 스트리트 106, 스위트 3에이
(72) 발명자
후앙, 지앙쉥
미국 캘리포니아주 94401 샌 마테오 에스 엘즈워스 에이브이 101 #350
리앙, 지아싱
미국 캘리포니아주 94401 샌 마테오 에스 엘즈워스 에이브이 101 #350
(뒷면에 계속)
(74) 대리인
특허법인 광장리앤고

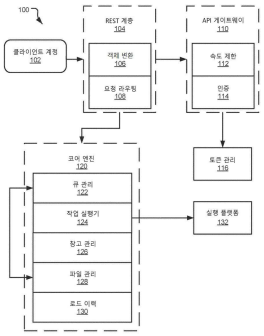
전체 청구항 수 : 총 30 항 심사관 : 이현중

(54) 발명의 명칭 **데이터베이스 시스템으로의 배치 데이터 수집**

(57) 요 약

데이터베이스의 테이블에 데이터의 배치 수집(batch ingestion)을 위한 시스템, 방법 및 디바이스. 방법은 데이터베이스로 수집할 클라이언트 계정으로부터 수신된 사용자 파일의 존재를 나타내는 통지를 결정하는 것을 포함한다. 방법은 사용자 파일에서 데이터를 식별하는 것, 및 사용자 파일에서 데이터를 수신하기 위해 데이터베이스의 목표 테이블을 식별하는 것을 포함한다. 방법은 데이터 및 목표 테이블을 나타내는 수집 작업을 생성하는 것을 포함한다. 방법은 수집 작업을 실행 플랫폼의 실행 노드에 할당하는 것을 포함하고, 실행 플랫폼은 데이터베이스 데이터를 집합적으로 저장하는 복수의 공유된 저장 디바이스에 독립적으로 동작하는 복수의 실행 노드를 포함한다. 방법은 데이터가 실행 노드에 의해 목표 테이블에 완전히 커밋된 후에, 메타데이터 저장소에서 목표 테이블에 관한 메타데이터를 등록하는 것을 포함한다.

대 표 도



(52) CPC특허분류
G06F 16/254 (2019.01)
(72) 발명자
지글러, 스콧
미국 캘리포니아주 94401 샌 마테오 에스 엘즈워스 에이브이 101 #350
유, 하오웨이
미국 캘리포니아주 94401 샌 마테오 에스 엘즈워스 에이브이 101 #350
데이지빌, 베노잇
미국 캘리포니아주 94401 샌 마테오 에스 엘즈워스 에이브이 101 #350
가네쉬, 배런
미국 캘리포니아주 94401 샌 마테오 에스 엘즈워스 에이브이 101 #350

(56) 선행기술조사문헌
US20120207075 A1*
US20100070753 A1*
US20130024424 A1*
US20170024248 A1*
*는 심사관에 의하여 인용된 문헌

명 세 서

청구범위

청구항 1

데이터베이스로의 배치 데이터 수집(batch data ingestion)을 위한 시스템으로서, 상기 시스템은:

데이터베이스로 수집될 파일의 존재를 나타내는 통지(notification)를 수신하는 수단 - 상기 데이터베이스는 데이터베이스 데이터를 포함하고, 상기 수신하는 것은 파일 큐(queue)가 폴링된(polled) 마지막 시간 이후 어느 새로운 파일이 상기 파일 큐에 커밋되었는지를(committed) 결정하기 위해 상기 파일 큐를 폴링하는 것을 포함함 -;

상기 파일에서 파일 데이터를 식별하는 수단;

상기 식별된 파일 데이터를 수신하기 위해 상기 데이터베이스의 목표 테이블을 식별하는 수단;

상기 식별된 파일 데이터 및 상기 목표 테이블을 나타내는 수집 작업을 생성하는 수단;

상기 수집 작업을 생성한 후 그리고 상기 식별된 파일 데이터 및 상기 목표 테이블을 수집하기 이전에, 상기 식별된 파일 데이터를 상기 목표 테이블에 대응하는 파일 큐에 유지하는 수단;

상기 수집 작업을 실행 플랫폼의 실행 노드에 할당하는 수단 - 상기 실행 플랫폼은 복수의 실행 노드를 포함하고, 복수의 공유된 저장 디바이스는 데이터베이스 데이터를 집합적으로 저장함 -;

상기 식별된 파일 데이터가 상기 실행 노드에 의해 상기 목표 테이블에 완전히 커밋된 후에, 메타데이터 저장소에 메타데이터를 등록하는 수단 - 상기 등록된 메타데이터는 상기 목표 테이블에 관련됨 -; 및

제2 목표 테이블에 대응하는 제2 파일 큐를 제공하는 수단 - 상기 제2 목표 테이블은 상기 목표 테이블과 상이하고, 상기 파일 큐는 제1 계정에 대응하고 상기 제2 파일 큐는 제2 계정에 대응함 - 을 포함하는, 시스템.

청구항 2

삭제

청구항 3

제1항에 있어서, 상기 통지를 수신하기 위한 상기 수단은 데이터 레이크(data lake)로부터 상기 통지를 수신하고, 상기 통지는 상기 파일이 상기 데이터 레이크에 추가되었음을 나타내고, 상기 데이터 레이크는 복수의 파일을 포함하는 데이터 저장소를 포함하는, 시스템.

청구항 4

제1항에 있어서:

활성화된 수집 작업의 현재 총 개수 및 활성화된 수집 작업의 원하는 개수를 식별하는 수단; 및

상기 활성화된 수집 작업의 현재 총 개수가 상기 활성화된 수집 작업의 상기 원하는 개수에 동일하도록, 상기 실행 플랫폼의 상기 복수의 실행 노드를 관리하는 수단을 더 포함하는, 시스템.

청구항 5

제1항에 있어서, 하나 이상의 파일로부터 어느 데이터가 상기 데이터베이스에 성공적으로 저장되었는지의 표시(indication)를 포함하는 수집 이력을 생성하는 수단을 더 포함하며, 상기 수집 이력은 메타데이터 저장소에 저장되고, 상기 수집 이력은 파일 이름, 테이블 식별, 파일 크기, 행 카운트 또는 수집 오류 코드 중 하나 이상을 포함하는, 시스템.

청구항 6

제1항에 있어서, 상기 목표 테이블의 테이블 식별을 해싱하는 것을 기초로 자원 관리자의 인스턴스에 상기 파일

을 할당하는 수단을 더 포함하는, 시스템.

청구항 7

제6항에 있어서, 자원 관리자의 새로운 인스턴스를 추가하는 수단을 더 포함하고, 상기 자원 관리자의 상기 새로운 인스턴스를 추가하는 것은 상기 해싱의 복수의 해시를 나누는 것 및 복수의 해시 각각을 자원 관리자의 복수의 인스턴스 중에서 할당하는 것을 포함하는, 시스템.

청구항 8

제1항에 있어서, 상기 수집 작업을 생성하는 수단은 상기 파일 큐에서의 작업량을 기초로 하나 이상의 수집 작업을 생성하도록 구성되며, 상기 파일 큐에서의 상기 작업량은:

상기 파일과 연관된 계정으로부터 최근에 수집된 파일의 평균 크기;

상기 파일 큐에서의 파일의 개수; 및

상기 파일의 크기

중 하나 이상을 기초로 결정되는, 시스템.

청구항 9

제1항에 있어서, 상기 데이터는 상기 목표 테이블에 대한 새로운 마이크로-파티션을 생성함으로써 상기 목표 테이블에 커밋되고, 상기 새로운 마이크로-파티션은 상기 데이터가 상기 목표 테이블에 커밋된 이후에 상기 복수의 공유된 저장 디바이스에 저장되는, 시스템.

청구항 10

제1항에 있어서, 상기 수집 작업을 상기 실행 플랫폼의 상기 실행 노드에 할당하는 수단은:

상기 실행 플랫폼에 의해 임계 개수의 작업이 이미 처리되고 있을 때, 상기 수집 작업을 상기 실행 노드에 할당하는 것을 지연시키는 것;

새로운 파일의 임계 개수가 계정 큐에 커밋될 때까지 상기 수집 작업을 상기 실행 노드에 할당하는 것을 지연시키는 것;

상기 파일이 계정 큐에 커밋된 이후에 상기 수집 작업을 상기 실행 노드에 할당하는 것; 및

새로운 파일의 임계 개수가 상기 계정 큐에 커밋될 때까지 상기 수집 작업을 상기 실행 노드에 할당하는 것을 지연시키는 것

중 하나 이상에 의해, 상기 실행 플랫폼에 의해 처리되는 작업의 총 개수를 관리하도록 구성되는, 시스템.

청구항 11

제10항에 있어서, 상기 실행 플랫폼에 의해 처리되는 상기 작업의 총 개수를 관리하는 것은 레이턴시를 임계 레벨 미만으로 유지하는 것을 포함하는, 시스템.

청구항 12

컴퓨팅 디바이스에 의해 수행되는, 데이터베이스로의 배치 데이터 수집을 위한 방법으로서, 상기 방법은:

데이터베이스로 수집될 파일의 존재를 나타내는 통지를 수신하는 것 - 상기 데이터베이스는 데이터베이스 데이터를 포함하고, 상기 수신하는 것은 파일 큐가 폴링된 마지막 시간 이후 어느 새로운 파일이 상기 파일 큐에 커밋되었는지를 결정하기 위해 상기 파일 큐를 폴링하는 것을 포함함 -;

상기 파일에서 파일 데이터를 식별하는 것;

상기 식별된 파일 데이터를 수신하기 위해 상기 데이터베이스의 목표 테이블을 식별하는 것;

상기 식별된 파일 데이터 및 상기 목표 테이블을 나타내는 수집 작업을 생성하는 것;

상기 수집 작업을 생성한 후 그리고 상기 식별된 파일 데이터 및 상기 목표 테이블을 수집하기 이전에, 상기 식

별된 파일 데이터를 상기 목표 테이블에 대응하는 파일 큐에 유지하는 것;

상기 수집 작업을 실행 플랫폼의 실행 노드에 할당하는 것 - 상기 실행 플랫폼은 복수의 실행 노드를 포함하고, 복수의 공유된 저장 디바이스는 데이터베이스 데이터를 집합적으로 저장함 -;

상기 식별된 파일 데이터가 상기 실행 노드에 의해 상기 목표 테이블에 완전히 커밋된 후에, 메타데이터 저장소에 메타데이터를 등록하는 것 - 상기 등록된 메타데이터는 상기 목표 테이블에 관련됨 -; 및

제2 목표 테이블에 대응하는 제2 파일 큐를 제공하는 것 - 상기 제2 목표 테이블은 상기 목표 테이블과 상이하고, 상기 파일 큐는 제1 계정에 대응하고 상기 제2 파일 큐는 제2 계정에 대응함 - 을 포함하는, 방법.

청구항 13

삭제

청구항 14

제12항에 있어서, 상기 통지를 수신하는 것은 데이터 레이크로부터 상기 통지를 수신하는 것을 포함하고, 상기 통지는 상기 파일이 상기 데이터 레이크에 추가되었음을 나타내고, 상기 데이터 레이크는 복수의 파일을 포함하는 데이터 저장소를 포함하는, 방법.

청구항 15

제12항에 있어서,

활성화된 수집 작업의 현재 총 개수 및 활성화된 수집 작업의 원하는 개수를 식별하는 것; 및

상기 활성화된 수집 작업의 현재 총 개수가 상기 활성화된 수집 작업의 상기 원하는 개수에 동일하도록, 상기 실행 플랫폼의 상기 복수의 실행 노드를 관리하는 것을 더 포함하는, 방법.

청구항 16

제12항에 있어서, 하나 이상의 파일로부터 어느 데이터가 상기 데이터베이스에 성공적으로 저장되었는지의 표시를 포함하는 수집 이력을 생성하는 것을 더 포함하며, 상기 수집 이력은 메타데이터 저장소에 저장되고, 상기 수집 이력은 파일 이름, 테이블 식별, 파일 크기, 행 카운트 또는 수집 오류 코드 중 하나 이상을 포함하는, 방법.

청구항 17

제12항에 있어서, 상기 목표 테이블의 테이블 식별을 해싱하는 것을 기초로 자원 관리자의 인스턴스에 상기 파일을 할당하는 것을 더 포함하는, 방법.

청구항 18

제17항에 있어서, 자원 관리자의 새로운 인스턴스를 추가하는 것을 더 포함하고, 상기 자원 관리자의 상기 새로운 인스턴스를 추가하는 것은 상기 해싱의 복수의 해시를 나누는 것 및 복수의 해시 각각을 자원 관리자의 복수의 인스턴스 중에서 할당하는 것을 포함하는, 방법.

청구항 19

제12항에 있어서, 상기 수집 작업을 생성하는 것은 상기 파일 큐에서의 작업량을 기초로 하나 이상의 수집 작업을 생성하는 것을 포함하며, 상기 파일 큐에서의 상기 작업량은:

상기 파일과 연관된 계정으로부터 최근에 수집된 파일의 평균 크기;

상기 파일 큐에서의 파일의 개수; 및

상기 파일의 크기

중 하나 이상을 기초로 결정되는, 방법.

청구항 20

제12항에 있어서, 상기 데이터는 상기 목표 테이블에 대한 새로운 마이크로-파티션을 생성함으로써 상기 목표 테이블에 커밋되고, 상기 새로운 마이크로-파티션은 상기 데이터가 상기 목표 테이블에 커밋된 이후에 상기 복수의 공유된 저장 디바이스에 저장되는, 방법.

청구항 21

제12항에 있어서, 상기 수집 작업을 상기 실행 플랫폼의 상기 실행 노드에 할당하는 것은:

상기 실행 플랫폼에 의해 임계 개수의 작업이 이미 처리되고 있을 때, 상기 수집 작업을 상기 실행 노드에 할당하는 것을 지연시키는 것;

새로운 파일의 임계 개수가 계정 큐에 커밋될 때까지 상기 수집 작업을 상기 실행 노드에 할당하는 것을 지연시키는 것;

상기 파일이 계정 큐에 커밋된 이후에 상기 수집 작업을 상기 실행 노드에 할당하는 것; 및

새로운 파일의 임계 개수가 상기 계정 큐에 커밋될 때까지 상기 수집 작업을 상기 실행 노드에 할당하는 것을 지연시키는 것

중 하나 이상에 의해, 상기 실행 플랫폼에 의해 처리되는 작업의 총 개수를 관리하는 것을 포함하는, 방법.

청구항 22

제21항에 있어서, 상기 실행 플랫폼에 의해 처리되는 상기 작업의 총 개수를 관리하는 것은 레이턴시를 임계 레벨 미만으로 유지하는 것을 포함하는, 방법.

청구항 23

비일시적 컴퓨터 판독가능 저장매체에 저장된 명령어를 실행하도록 프로그램 가능한 프로세서로서, 상기 명령어는:

데이터베이스로 수집될 파일의 존재를 나타내는 통지를 수신하는 것 - 상기 데이터베이스는 데이터베이스 데이터를 포함하고, 상기 수신하는 것은 파일 큐가 폴링된 마지막 시간 이후 어느 새로운 파일이 상기 파일 큐에 커밋되었는지를 결정하기 위해 상기 파일 큐를 폴링하는 것을 포함함 -;

상기 파일에서 파일 데이터를 식별하는 것;

상기 식별된 파일 데이터를 수신하기 위해 상기 데이터베이스의 목표 테이블을 식별하는 것;

상기 식별된 파일 데이터 및 상기 목표 테이블을 나타내는 수집 작업을 생성하는 것;

상기 수집 작업을 생성한 후 그리고 상기 식별된 파일 데이터 및 상기 목표 테이블을 수집하기 이전에, 상기 식별된 파일 데이터를 상기 목표 테이블에 대응하는 파일 큐에 유지하는 것;

상기 수집 작업을 실행 플랫폼의 실행 노드에 할당하는 것 - 상기 실행 플랫폼은 복수의 실행 노드를 포함하고, 복수의 공유된 저장 디바이스는 데이터베이스 데이터를 집합적으로 저장함 -;

상기 식별된 파일 데이터가 상기 실행 노드에 의해 상기 목표 테이블에 완전히 커밋된 후에, 메타데이터 저장소에 메타데이터를 등록하는 것 - 상기 등록된 메타데이터는 상기 목표 테이블에 관련됨 -; 및

제2 목표 테이블에 대응하는 제2 파일 큐를 제공하는 것 - 상기 제2 목표 테이블은 상기 목표 테이블과 상이하고, 상기 파일 큐는 제1 계정에 대응하고 상기 제2 파일 큐는 제2 계정에 대응함 - 을 포함하는, 프로세서.

청구항 24

삭제

청구항 25

제23항에 있어서, 상기 통지를 수신하는 것은 데이터 레이크로부터 상기 통지를 수신하는 것을 포함하고, 상기 통지는 상기 파일이 상기 데이터 레이크에 추가되었음을 나타내고, 상기 데이터 레이크는 복수의 파일을 포함하는 데이터 저장소를 포함하는, 프로세서.

청구항 26

제23항에 있어서, 상기 명령어는:

활성화된 수집 작업의 현재 총 개수 및 활성화된 수집 작업의 원하는 개수를 식별하는 것; 및

상기 활성화된 수집 작업의 현재 총 개수가 상기 활성화된 수집 작업의 상기 원하는 개수에 동일하도록, 상기 실행 플랫폼의 상기 복수의 실행 노드를 관리하는 것을 더 포함하는, 프로세서.

청구항 27

제23항에 있어서, 상기 명령어는 하나 이상의 파일로부터 어느 데이터가 상기 데이터베이스에 성공적으로 저장되었는지의 표시를 포함하는 수집 이력을 생성하는 것을 더 포함하며, 상기 수집 이력은 메타데이터 저장소에 저장되고, 상기 수집 이력은 파일 이름, 테이블 식별, 파일 크기, 행 카운트 또는 수집 오류 코드 중 하나 이상을 포함하는, 프로세서.

청구항 28

제23항에 있어서, 상기 명령어는 상기 목표 테이블의 테이블 식별을 해싱하는 것을 기초로 자원 관리자의 인스턴스에 상기 파일을 할당하는 것을 더 포함하는, 프로세서.

청구항 29

제28항에 있어서, 상기 명령어는 자원 관리자의 새로운 인스턴스를 추가하는 것을 더 포함하고, 상기 자원 관리자의 상기 새로운 인스턴스를 추가하는 것은 상기 해싱의 복수의 해시를 나누는 것 및 복수의 해시 각각을 자원 관리자의 복수의 인스턴스 중에서 할당하는 것을 포함하는, 프로세서.

청구항 30

제23항에 있어서, 상기 수집 작업을 생성하는 것은 상기 파일 큐에서의 작업량을 기초로 하나 이상의 수집 작업을 생성하는 것을 포함하며, 상기 파일 큐에서의 상기 작업량은:

상기 파일과 연관된 계정으로부터 최근에 수집된 파일의 평균 크기;

상기 파일 큐에서의 파일의 개수; 및

상기 파일의 크기 중

하나 이상을 기초로 결정되는, 프로세서.

청구항 31

제23항에 있어서, 상기 데이터는 상기 목표 테이블에 대한 새로운 마이크로-파티션을 생성함으로써 상기 목표 테이블에 커밋되고, 상기 새로운 마이크로-파티션은 상기 데이터가 상기 목표 테이블에 커밋된 이후에 상기 복수의 공유된 저장 디바이스에 저장되는, 프로세서.

청구항 32

제23항에 있어서, 상기 수집 작업을 상기 실행 플랫폼의 상기 실행 노드에 할당하는 것은:

상기 실행 플랫폼에 의해 임계 개수의 작업이 이미 처리되고 있을 때, 상기 수집 작업을 상기 실행 노드에 할당하는 것을 지연시키는 것;

새로운 파일의 임계 개수가 계정 큐에 커밋될 때까지 상기 수집 작업을 상기 실행 노드에 할당하는 것을 지연시키는 것;

상기 파일이 계정 큐에 커밋된 이후에 상기 수집 작업을 상기 실행 노드에 할당하는 것; 및

새로운 파일의 임계 개수가 상기 계정 큐에 커밋될 때까지 상기 수집 작업을 상기 실행 노드에 할당하는 것을 지연시키는 것

중 하나 이상에 의해, 상기 실행 플랫폼에 의해 처리되는 작업의 총 개수를 관리하는 것을 포함하는, 프로세서.

청구항 33

제32항에 있어서, 상기 실행 플랫폼에 의해 처리되는 상기 작업의 총 개수를 관리하는 것은 레이턴시를 임계 레벨 미만으로 유지하는 것을 포함하는, 프로세서.

발명의 설명

기 술 분 야

[0001] 연관된 출원에 대한 상호 참조

[0002] 본 출원은 2017년 11월 27일에 출원되고 명칭이 "배치 데이터 수집(BATCH DATA INGESTION)을 위한 시스템, 방법 및 디바이스"인 미국 가특허출원 제62/591,118호의 우선권을 주장하며, 이는 그 전체가 본원에 참조로 통합되고, 이는 이하에서 구체적으로 나타나는 이들 부분을 포함하고 이에 제한되지 않으며, 참조로의 통합은 다음의 예외: 위에 참조된 출원의 임의의 부분이 본 출원과 일치하지 않는 경우, 본 출원은 위에 참조된 출원을 대체함을 제외하고 이루어진다.

[0003] 기술분야

[0004] 본 개시는 데이터베이스에 관한 것으로, 보다 구체적으로 데이터베이스 또는 테이블에서 데이터의 증분 수집(incremental ingestion)에 관한 것이다.

배 경 기 술

[0005] 컴퓨팅 애플리케이션(computing applications)에서 데이터 저장 및 액세스를 위해 데이터베이스가 널리 사용된다. 데이터베이스는 질의를 사용하여 판독되거나, 수정되거나 또는 삭제될 수 있는 데이터를 포함하거나 참조하는 하나 이상의 테이블을 포함할 수 있다. 데이터베이스는 하나 이상의 테이블 내에 작은 데이터의 세트로부터 상당히 큰 데이터의 세트까지 어디에나 저장할 수 있다. 이 데이터는 조직에서 다양한 사용자에 의해 액세스되거나, 또는 심지어 웹 사이트 또는 애플리케이션 프로그램 인터페이스(application program interface, API)를 통해 공용 사용자에게 서비스하는 데 사용될 수 있다. 컴퓨팅 및 저장소 자원 양자뿐만 아니라, 그의 기본 아키텍처는 바람직한 데이터베이스 성능을 달성하는데 상당한 역할을 할 수 있다.

[0006] 데이터는 데이터 버킷(data bucket)으로부터 데이터베이스의 하나 이상의 테이블로 수집될 수 있다. 데이터베이스 또는 테이블에 데이터를 업로드하고 저장하기 위한 다양한 시스템이 개발되고 서술되며, 널리 알려진다. 예를 들어, 공유된 디스크 시스템에서 모든 데이터는 데이터 클러스터에서 모든 처리 노드로부터 액세스 가능한 공유된 저장 디바이스상에 저장된다. 이 타입의 시스템에서, 데이터 클러스터에서의 모든 처리 노드가 일치하는 버전의 데이터에 액세스함을 보장하기 위해, 모든 데이터 변경(change)은 공유된 저장 디바이스에 기록된다. 공유된 디스크 시스템에서 처리 노드의 개수가 증가할수록, 공유된 저장 디바이스(및 처리 노드와 공유된 저장 디바이스 사이의 통신 링크)에는 데이터 판독 및 데이터 기록 동작을 느리게 하는 병목 현상이 나타난다. 이 병목 현상은 더욱 많은 처리 노드의 추가를 통해 더 악화된다. 따라서, 기존의 공유된 디스크 시스템은 이 병목 현상 문제에 기인하여 확장성을 제한한다.

[0007] 기존의 다른 데이터 저장 및 검색 시스템은 "비공유 아키텍처(shared-nothing architecture)"로 지칭된다. 이 아키텍처에서, 데이터는 다수의 처리 노드에 분산되며, 각 노드는 전체 데이터베이스에서 데이터의 서브세트를 저장한다. 새로운 처리 노드가 추가되거나 제거될 때, 비공유 아키텍처는 다수의 처리 노드에 걸쳐 데이터를 재배열해야 한다. 이러한 데이터의 재배열은 데이터 재배열 동안 실행되는 데이터 판독 및 기록 동작에 대해 시간-소모적이고 지장을 줄(disruptive) 수 있다. 그리고, 특정한 노드에 대한 데이터의 관련성(affinity)은 인기 있는 데이터에 대한 데이터 클러스터 상에 "핫 스팟(hot spots)"을 생성할 수 있다. 더욱이, 각 처리 노드가 또한, 저장 기능을 수행하기 때문에, 이 아키텍처는 적어도 하나의 처리 노드가 데이터를 저장하는 것을 필요로 한다. 따라서, 모든 처리 노드가 제거되는 경우, 비공유 아키텍처는 데이터를 저장하지 못한다. 게다가, 다수의 상이한 처리 노드에 걸친 데이터의 분산에 기인하여, 비공유 아키텍처에서 데이터의 관리가 복잡하다.

[0008] 데이터 수집을 위한 기존의 시스템 및 방법은 파일이 손실될 수 있게 하는 하나의 스테이트먼트(statement)를 갖는/스테이트먼트가 없는 단일 명령을 사용한다. 예를 들어, 기존의 시스템에서, 파일 수집은 테이블의 수집 도중에 실패할 수 있고, 모든 이전에 수집한 데이터가 손실되게 할 수 있다. 또한, 종래의 데이터 수집에서, 사용자는 창고를 할당하고 명령을 발행하도록 요구될 수 있고, 데이터는 사용자가 특정 명령을 발행할 때까지 캡

쳐되지 않을 수 있다.

[0009] 본원에 서술된 시스템 및 방법은 기존의 시스템의 위에서 식별된 제약을 완화시키는 데이터 저장, 데이터 수집 및 데이터 검색에 대해 개선된 접근법을 제공한다.

도면의 간단한 설명

[0010] 본 개시의 제한적이지 않고, 철저하지는 않은 실시예가 다음의 도면을 참조로 서술되며, 달리 명시되지 않는 한 유사한 참조부호는 다양한 도면 전체에 걸쳐 비슷하거나 또는 유사한 부분을 지칭한다. 본 개시의 장점은 다음의 서술 및 첨부 도면에 관련하여 더 잘 이해될 것이다.

도 1은 본 개시의 교시 및 원리에 따른 자동화된 데이터 수집을 위한 시스템의 블록도 아키텍처이다.

도 2는 본 개시의 교시 및 원리에 따른 데이터를 수집하는 프로세스의 블록도이다.

도 3은 본 개시의 교시 및 원리에 따른 검색 및 데이터 저장 시스템의 구성요소의 블록도이다.

도 4는 본 개시의 교시 및 원리에 따른 자원 관리자의 일 실시예의 블록도이다.

도 5는 본 개시의 교시 및 원리에 따른 실행 플랫폼의 일 실시예의 블록도이다.

도 6은 본 개시의 교시 및 원리에 따른 동작 환경의 구성요소를 도시하는 블록도이다.

도 7은 본 개시의 교시 및 원리에 따른 배치 데이터 수집 시스템의 블록도이다.

도 8은 본 개시의 교시 및 원리에 따른 데이터베이스에 배치 데이터 수집을 위한 방법의 개략적인 흐름도이다.

도 9는 본원에 교시된 컴퓨터 프로세스를 가능하게 하는 개시와 일치하는 예시적인 컴퓨팅 디바이스의 블록도이다.

발명을 실시하기 위한 구체적인 내용

[0011] 데이터베이스 또는 테이블에 배치 데이터 수집을 위한 시스템, 방법 및 디바이스가 개시된다. 통상의 기술자에게 알려진 전통적인 데이터베이스 시스템에서, 사용자 파일은 복사 명령에 의해 데이터베이스 테이블에 삽입될 수 있다. 이는 데이터 수집에 할당된 구동되는 창고의 사용을 필요로 하며, 이는 동기화 동작으로서 실행된다. 또한, 이 전통적인 접근법에서, 데이터 수집 동작 동안 새로운 데이터가 테이블에 삽입될 때, 질의 또는 다른 동작을 위한 데이터베이스 테이블의 사용이 차단될 수 있다. 또한, 테이블에 대해 이루어진 임의의 업데이트 또는 클라이언트 계정으로부터 수신된 임의의 새로운 데이터는 사용자가 새로운 사용자 파일을 삽입하기 위해 복사 명령을 수동으로 발행할 때까지 데이터베이스에 의해 캡쳐되지 않을 것이다. 데이터가 사용자에 의해 수동으로 개시된 복사 명령과 같은 단일 명령에 의해 수집될 때, 데이터의 일부 또는 전체는 수집 작업이 실패하는 경우 손실될 수 있다. 통상의 기술자에게 알려진 이러한 전통적인 시스템에서, 데이터 수집 동작은 그 도중에 실패할 수 있고, 그러므로 전체 데이터 수집 동작이 반복되는 것을 요구할 수 있다.

[0012] 데이터베이스로의 데이터 수집을 위한 개선된 시스템, 방법 및 디바이스가 본원에 개시된다. 이러한 개선된 시스템, 방법 및 디바이스는 빈번한 데이터 로딩에 기인하여 발생하는 데이터베이스 시스템 상의 부담을 낮추고, 클라이언트 계정의 사용자에 대한 데이터 수집을 위한 프로세스를 단순화한다. 본원에 개시된 시스템, 방법 및 디바이스에서, 사용자 파일로부터의 데이터는 시스템 장애의 경우에도 이미 수집한 데이터가 손실되지 않도록, 데이터베이스 테이블의 마이크로-파티션에 점진적으로(incrementally) 커밋된다(committed). 시스템 장애의 경우, 장애가 발생한 시점으로부터 수집 동작이 계속될 수 있도록, 데이터가 분할되고, 파티션 단위로(partition-by-partition) 데이터베이스에 삽입될 수 있다.

[0013] 게다가, 본원에 개시된 시스템, 방법 및 디바이스에 대한 처리 자원은 필요에 따라 확장되거나 축소될 수 있다. 그와 같이, 통상의 기술자에게 알려진 전통적인 데이터베이스 수집 시스템과 달리, 본원에 개시된 시스템에는 창고가 요구되지 않는다. 본원에 개시된 시스템, 방법 및 디바이스는 사용자 파일의 비동기 및 서버리스(serverless) 데이터 수집을 제공한다. 하나 이상의 새로운 사용자 파일이 클라이언트 데이터 버킷에 추가되는 것과 같이, 클라이언트 데이터 버킷에 대해 이루어진 변경이 검출되고, 이러한 새로운 파일은 클라이언트 계정의 사용자로부터 특정한 명령어를 요구하지 않으면서, 적절한 데이터베이스 테이블로 자동으로 로딩된다. 또한, 하나 이상의 사용자 파일의 데이터에 대한 목표 테이블이 자동으로 결정되고, 이러한 사용자 파일은 데이터베이스의 특정 테이블로 새로운 데이터의 수집을 관리하도록 할당된 자원 관리자의 적절한 인스턴스(instance)에 할

당된다.

[0014] 본원에 개시된 시스템, 방법 및 디바이스는 데이터베이스에 어떤 데이터 및 어떤 사용자 파일이 성공적으로 커밋되었는지에 대한 메타데이터를 생성하고 및 유지하는 것을 더 포함한다. 메타데이터는 예를 들어, 데이터가 어느 마이크로-파티션에 및 어느 테이블에 성공적으로 삽입되었는지, 데이터가 언제 삽입되었는지, 삽입된 데이터의 크기 등을 더 나타낸다. 이러한 메타데이터는 실행 플랫폼에 걸쳐 및 데이터베이스를 집합적으로 저장하는 복수의 공유된 저장 디바이스에 걸쳐 공유될 수 있다. 메타데이터는 복수의 공유된 저장 디바이스와 별도로 저장될 수 있거나, 또는 복수의 공유된 저장 디바이스 내에 저장될 수 있다. 또한 메타데이터는 데이터베이스 테이블 내에 별도의 마이크로-파티션으로 저장되거나, 또는 데이터베이스 데이터를 포함하는 마이크로-파티션 내에 저장될 수 있다.

[0015] 본 개시의 일 실시예에서, 데이터베이스로의 데이터의 증분 수집을 위한 시스템이 개시된다. 시스템은 클라이언트 계정으로부터 수신되고 데이터베이스로 수집될 사용자 파일의 존재를 나타내는 통지를 결정하는 수단을 포함한다. 시스템은 사용자 파일에서 데이터를 식별하는 수단, 및 사용자 파일에서 데이터를 수신하기 위해 데이터베이스의 목표 테이블을 식별하는 수단을 포함한다. 시스템은 데이터 및 목표 테이블을 나타내는 수집 작업을 생성하는 수단을 포함한다. 시스템은 수집 작업을 실행 플랫폼의 실행 노드에 할당하는 수단을 포함하며, 실행 플랫폼은 데이터베이스 데이터를 집합적으로 저장하는 복수의 공유된 저장 디바이스와 독립적으로 동작하는 복수의 실행 노드를 포함한다. 시스템은 데이터가 실행 노드에 의해 목표 테이블에 완전히 커밋된 후에, 메타데이터 저장소에 목표 테이블에 관한 메타데이터를 등록하는 수단을 포함한다.

[0016] 본 개시의 일 실시예에서, 데이터베이스로의 배치 데이터 수집을 위한 방법이 개시된다. 방법은 클라이언트 계정으로부터 수신되고 데이터베이스로 수집될 사용자 파일의 존재를 나타내는 통지를 결정하는 것을 포함한다. 방법은 사용자 파일에서 데이터를 식별하는 것, 및 사용자 파일에서 데이터를 수신하기 위해 데이터베이스의 목표 테이블을 식별하는 것을 포함한다. 방법은 데이터 및 목표 테이블을 나타내는 수집 작업을 생성하는 것을 포함한다. 방법은 수집 작업을 실행 플랫폼의 실행 노드에 할당하는 것을 포함하고, 실행 플랫폼은 데이터베이스 데이터를 집합적으로 저장하는 복수의 공유된 저장 디바이스에 독립적으로 동작하는 복수의 실행 노드를 포함한다. 방법은 데이터가 실행 노드에 의해 목표 테이블에 완전히 커밋된 후에, 메타데이터 저장소에서 목표 테이블에 관한 메타데이터를 등록하는 것을 포함한다. 일 실시예에서, 방법은 데이터베이스 시스템의 자원 관리자(예를 들어, 302 참조)에 의해 수행된다.

[0017] 데이터베이스 테이블은 복수의 마이크로-파티션에 데이터를 저장할 수 있으며, 마이크로-파티션은 변경할 수 없는 저장 디바이스이다. 트랜잭션(transaction)이 이러한 테이블에서 실행될 때, 영향을 받는 모든 마이크로-파티션은 트랜잭션의 수정을 반영하는 새로운 마이크로-파티션을 생성하기 위해 재생성된다. 트랜잭션이 완전히 실행된 이후에, 재생성된 임의의 원래의 마이크로-파티션이 데이터베이스로부터 제거될 수 있다. 테이블에서 실행된 각 트랜잭션 이후에 새로운 버전의 테이블이 생성된다. 테이블에서의 데이터가 삽입, 삭제, 업데이트 및/또는 병합과 같은 다수의 변경을 겪는 경우, 테이블은 일 기간 동안 다수의 버전을 겪을 수 있다. 각 버전의 테이블은 어떤 트랜잭션이 테이블을 생성하였는지, 트랜잭션이 언제 순서화되었는지, 트랜잭션이 언제 완전히 실행되었는지, 트랜잭션이 테이블에서 하나 이상의 행을 어떻게 변경했는지를 나타내는 메타데이터를 포함할 수 있다. 저비용의 테이블 버저닝(versioning)을 위한 개시된 시스템, 방법 및 디바이스는 데이터에 대해 이루어진 수정에 응답하여 데이터베이스 데이터 상에서 실행될 작업을 트리거링하는데 효율적인 수단을 제공하기 위해 이용될 수 있다(leveraged).

[0018] 변경 추적 정보는 데이터베이스에서 메타데이터로서 저장될 수 있다. 이 메타데이터는 고객의 데이터베이스 테이블에 저장된 데이터를 서술하지만, 실제로 저장된 테이블 데이터인 것은 아니다. 특히 다수의 고객의 큰 데이터베이스 테이블이 있는 경우, 메타데이터는 매우 커질 수 있다. 현재 데이터베이스 시스템은 대량의 메타데이터를 처리하는데 심각한 제약을 갖는다. 현재 데이터베이스 시스템은 주메모리, 파일 시스템 및 키-값 저장소를 포함하는, 변경 가능한 저장 디바이스 및 서비스에 메타데이터를 저장한다. 이들 디바이스 및 서비스는 메타데이터가 그 자리에서(in-place) 업데이트된 데이터가 되는 것을 허용한다. 데이터 레코드가 변경되면, 이는 새로운 정보로 업데이트될 수 있으며, 오래된 정보는 덮어 씌워진다. 이는 메타데이터를 그 자리에서 업데이트함으로써, 데이터베이스가 변경 가능한 메타데이터를 쉽게 유지하는 것을 허용한다.

[0019] 하지만, 이들 변경 가능한 저장 디바이스 및 서비스는 제약을 갖는다. 제약은 적어도 두 개이다. 첫째, 주메모리 및 파일 시스템과 같은 변경 가능한 저장 디바이스는 저장소 용량에 대해 어려운 제한(hard limit)을 갖는다. 메타데이터의 크기가 이들 제한을 초과하는 경우, 그곳에 더욱 많은 메타데이터를 저장하는 것은 불가

능하다. 둘째, 키-값 저장소와 같은 변경 가능한 저장 서비스는 대량의 메타데이터를 판독할 때 저하되어 수행된다. 데이터를 판독하는 것은 범위 스캔을 사용하여 수행되며, 이는 완료에 긴 시간이 걸린다. 실제로, 범위 스캔은 대규모 전개에서 완료하는 데 몇 분 또는 심지어 1시간이 걸릴 수도 있다.

[0020] 이들 제약은 기존의 변경 가능한 저장 디바이스 및 서비스에 대량의 메타데이터를 저장하는 것을 불가능하게 한다. 본원에 개시된 시스템, 방법 및 디바이스는 마이크로-파티션(micro-partitions)과 같이 변경할 수 없는(변경 가능하지 않은) 저장소에 메타데이터를 저장하는 것을 포함하는 개선된 메타데이터 저장 및 관리를 제공한다. 본원에 사용된, 변경할 수 없거나 또는 변경 가능하지 않은 저장소는 데이터가 그 자리에서 덮어 씌워지거나 또는 업데이트되도록 허가될 수 없거나 또는 허가되지 않은 저장소를 포함한다. 예를 들어, 저장 매체의 셀 또는 영역에 위치된 데이터에 대한 변경은 저장 매체의 상이한, 타임-스탬프된(time-stamped) 셀 또는 영역에 새로운 파일로서 저장될 수 있다. 변경 가능한 저장소는 데이터가 그 자리에서 덮어 씌워지거나 또는 업데이트되도록 허가되는 저장소를 포함할 수 있다. 예를 들어, 저장 매체의 주어진 셀 또는 영역에서의 데이터는 저장 매체의 그 셀 또는 영역에 관련된 데이터에 대한 변경이 있을 때 덮어 씌워질 수 있다.

[0021] 일 실시예에서, 메타데이터는 클라우드의 변경 가능하지 않은 저장 서비스 상에 저장되고 유지된다. 이들 저장 서비스는 예를 들어, Amazon S3 ®, Microsoft Azure Blob Storage®, 및 Google Cloud Storage®를 포함할 수 있다. 이들 서비스 중 다수는 그 자리에서 데이터를 업데이트하는 것을 허용하지 않는다(즉, 변경 가능하지 않거나 또는 변경할 수 없다). 데이터 파일은 추가되거나 삭제될 수만 있고, 업데이트될 수는 없다. 일 실시예에서, 이들 서비스 상에 메타데이터를 저장하고 유지하는 것은 메타데이터에서의 모든 변경에 대해, 메타데이터 파일이 저장 서비스에 추가되는 것을 필요로 한다. 이들 메타데이터 파일은 백그라운드에서 더욱 큰 "압축된(compacted)" 또는 통합된 메타데이터 파일로 주기적으로 통합될 수 있다.

[0022] 일 실시예에서, 테이블에서의 모든 데이터는 마이크로-파티션으로 지칭되는 변경할 수 없는 저장 디바이스로 자동으로 나누어진다. 마이크로-파티션은 각 마이크로-파티션이 인접한 저장 유닛을 갖는 배치 유닛으로 고려될 수 있다. 예시로서, 각 마이크로-파티션은 50MB 내지 1000MB의 압축되지 않은 데이터를 포함할 수 있다(데이터가 압축되어 저장될 수 있기 때문에 저장소의 실제 크기는 더욱 작을 수 있음을 유의한다). 테이블에서 행의 그룹은 열 방식으로 구성된 개별적인 마이크로-파티션에 매핑될 수 있다. 이 크기 및 구조는 수백만 또는 심지어 수억 개의 마이크로-파티션으로 포함될 수 있는 매우 큰 테이블의 상당히 세밀한 제거(granular pruning)를 허용한다. 메타데이터는 마이크로-파티션에 저장된 모든 행에 대해 자동으로 수집될 수 있으며, 이는 마이크로-파티션에서 열 각각에 대한 값의 범위; 고유한 값(distinct values)의 개수; 및/또는 최적화 및 효율적인 질의 처리에 양자에 사용되는 추가적인 특성을 포함한다. 일 실시예에서, 마이크로-파티셔닝(micro-partitioning)은 모든 테이블 상에서 자동으로 수행될 수 있다. 예를 들어, 데이터가 삽입/로드될 때 발생하는 순서를 사용하여 테이블이 명료하게 분할될 수 있다.

[0023] 일 실시예에서, 파일 메타데이터는 메타데이터 저장소 내에 저장된다. 파일 메타데이터에는 테이블 버전 및 각 테이블 데이터 파일에 대한 정보를 포함한다. 메타데이터 저장소는 로컬 파일 시스템, 시스템, 메모리 등과 같은 변경 가능한 저장소(그 위에 기록되거나 또는 그 자리에 기록될 수 있는 저장소)를 포함할 수 있다. 일 실시예에서, 마이크로-파티션 메타데이터는 두 개의 데이터 세트: 테이블 버전 및 파일 정보로 구성된다. 테이블 버전 데이터 세트는 추가된 파일 및 제거된 파일의 목록에 대한 테이블 버전의 매핑을 포함한다. 파일 정보는 예를 들어, 마이크로-파티션 경로, 마이크로-파티션 크기, 마이크로-파티션 키 id 및 마이크로-파티션에 저장된 모든 행 및 열의 요약을 포함하는, 각 마이크로-파티션에 대한 정보로 구성된다. 테이블의 각 수정은 새로운 마이크로-파티션 및 새로운 마이크로-파티션 메타데이터를 생성한다. 테이블로의 삽입은 새로운 마이크로-파티션을 생성한다. 테이블로부터의 삭제는 마이크로-파티션을 제거하고, 마이크로-파티션에서 모든 행이 삭제되지 않은 경우 테이블에서의 나머지 행을 갖는 새로운 마이크로-파티션을 잠재적으로 추가한다. 업데이트는 마이크로-파티션을 제거하고, 이를 업데이트된 레코드를 포함하는 행을 갖는 새로운 마이크로-파티션으로 교체한다.

[0024] 일 실시예에서, 변경 추적 열을 포함하는 메타데이터는 변경할 수 없는 저장소에서 메타데이터 마이크로-파티션에 저장될 수 있다. 일 실시예에서, 시스템은 데이터베이스 테이블의 모든 수정을 위해 메타데이터 마이크로-파티션을 클라우드 저장소에 기록할 수 있다. 일 실시예에서, 시스템은 스캔 세트를 컴퓨팅하기 위해 메타데이터 마이크로-파티션을 다운로드하고 판독할 수 있다. 메타데이터 마이크로-파티션은 스캔 세트 컴퓨팅을 개선하기 위해 병렬로 다운로드되고, 수신될 때 판독될 수 있다. 일 실시예에서, 시스템은 백그라운드에서 메타데이터 마이크로-파티션을 주기적으로 통합할 수 있다. 일 실시예에서, 프리-페칭, 캐싱, 열 지향 레이아웃(columnar layout) 등을 포함하는 성능 개선이 포함될 수 있다. 또한, 열 지향 레이아웃을 갖는 메타데이터 파일로 암호화

및 무결성 검사를 포함하는 보안 개선이 또한 가능하다.

[0025] 데이터베이스는 각각 마이크로-파티션과 같은 변경할 수 없는 저장 디바이스를 더 포함할 수 있는 복수의 테이블을 포함할 수 있다. 사용자 파일에서 데이터가 손실되지 않도록 사용자 파일은 변경할 수 없는 마이크로-파티션 형태로 데이터베이스 테이블로 점진적으로 수집될 수 있다. 하나 이상의 사용자 파일로부터 데이터는 사용자 파일이 데이터베이스로 수집된 것으로 간주되기 전에, 데이터베이스 테이블에 완전하고 성공적으로 커밋되어야 하는 데이터의 증분 부분에서 수집될 수 있다.

[0026] 데이터베이스 시스템에 대한 클라이언트 계정은 데이터를 포함하는 하나 이상의 사용자 파일을 제공할 수 있다. 사용자 파일은 클라이언트 계정 큐에 커밋될 수 있으며 데이터는 사용자 파일에서 식별될 수 있다. 사용자 파일은 실행 플랫폼의 실행 노드에 할당될 수 있으며, 실행 플랫폼은 사용자 파일에서 데이터를 포함하는 데이터베이스의 목표 테이블에 삽입될 마이크로-파티션 또는 다른 변경할 수 없는 저장 디바이스를 생성할 수 있다.

[0027] 예를 들어, 이 새로운 시스템은 고객 또는 제3자 창고 또는 서버없이 동작될 수 있으며, 클라이언트 계정에 대한 데이터 변경을 단순화할 수 있다. 일 실시예에서, 데이터 수집은 시스템 장애로 인해 데이터가 손실되지 않도록, 파일이 데이터베이스에 배치로(in batches) 커밋되는 경우 점진적으로 완료된다. 일 실시예에서, 시스템은 파일에 대한 변경을 검출하고 사용자로부터의 특정 명령을 요구하지 않으면서 변경을 데이터베이스에 자동으로 로딩한다. 일 실시예에서, 시스템은 데이터의 배치를 수집하고, 데이터 수집 동안 장애가 발생할 경우 시스템이 동일한 장소에서 데이터 수집을 계속할 수 있도록, 데이터 풀링(data pulling) 및 데이터 커밋 동안 프로세스를 추적한다.

[0028] 본 개시의 일 실시예는 기존의 시스템에 비해 탄력성(elasticity)을 증가시키는 데이터 수집을 위한 시스템을 제공한다. 일 실시예에서, 시스템은 복수의 컴퓨팅 자원 및 가상 창고를 포함하고, 데이터는 이용 가능한 자원을 사용하여 자동으로 점진적으로 수집된다. 컴퓨팅 용량은 컴퓨팅 자원의 사용을 변경하고, 시스템 상의 작업 부하를 변화시킴으로써 동적으로 조정된다. 일 실시예에서, 복수의 클라이언트 계정에 대해 공통 자원의 풀이 제공되며, 컴퓨팅 코어의 사용은 모든 클라이언트 계정에 걸쳐 완전히 동적이고 유연하다.

[0029] 본 개시의 실시예는 기존 시스템에 비해 세분성(granularity)을 증가시킨 데이터 수집을 위한 시스템을 제공한다. 일 실시예에서, 시스템은 데이터를 점진적으로 입력 및 커밋하고, 잠재적인 네트워크 장애에 대해 데이터 수집을 보호한다. 일 실시예에서, 하나의 사용자 파일 또는 사용자 파일의 단편이 한번에 수집된다. 일 실시예에서, 임계 개수의 사용자 파일이 수신되면, 사용자 파일은 데이터베이스 테이블에 자동으로 커밋될 것이다. 사용자 파일은 변경할 수 없는 마이크로-파티션의 형태로 그 자리에서 변경될 수 없는 데이터베이스 테이블에 커밋될 수 있다. 일 실시예에서, 시스템이 어느 사용자 파일이 어느 위치에서 수집되었는지를 정확하게 복원할 수 있도록, 데이터베이스 테이블의 내용과 같은 데이터베이스 데이터의 상태는 메타데이터에 저장된다.

[0030] 본 개시의 다음의 서술에서, 본 개시의 일부를 형성하고 본 개시가 실시될 수 있는 특정 구현이 예시로서 도시된 첨부 도면에 대한 참조가 이루어진다. 다른 구현이 이용될 수 있고, 본 개시의 범주를 벗어나지 않으면서 구조적인 변경이 이루어질 수 있음이 이해된다.

[0031] 본 개시를 서술하고 주장하는 것에서, 다음의 용어는 아래에 제시된 정의에 따라 사용될 것이다.

[0032] 본 명세서 및 첨부된 청구 범위에 사용된 바와 같이, 단수형인 "하나의(a, an)" 및 "그(the)"는 문맥상 명백하게 달리 지시되지 않는 한 복수의 지시 대상을 포함한다는 것을 유의해야 한다.

[0033] 본 명세서 전체에 걸쳐 "하나의 실시예", "일 실시예", "하나의 구현", "일 구현", "하나의 예시" 또는 "일 예시"에 대한 참조는 실시예, 구현 또는 예시와 관련하여 서술된 특정한 피처, 구조 또는 특성이 본 개시의 적어도 하나의 실시예에 포함됨을 의미한다. 따라서, 본 명세서 전체에 걸쳐 다양한 위치에서 위에 식별된 어구의 출현은 모두가 동일한 실시예, 구현 또는 예시를 참조하는 것은 아니다. 또한, 본원에 제공된 도면은 통상의 기술자에게 설명하기 위한 목적으로 인식되어야 한다.

[0034] 본원에 사용된 "포함하는(comprising, including)", "함유하는(containing)" 및 그의 문법적인 등가물은 추가의 언급되지 않은 요소 또는 방법 단계를 배제하지 않는 포괄적이거나 오픈-엔드(open-ended) 용어이다.

[0035] 본원에 사용된, "테이블"은 레코드(행)의 집합으로 정의된다. 각 레코드는 테이블 속성(열)의 값의 집합을 포함한다. 테이블은 통상적으로, 다수의 더욱 작은 (가변 크기 또는 고정 크기) 저장 유닛 예를 들어, 파일 또는 블록에 물리적으로 저장된다.

[0036] 본 개시에 따른 실시예는 장치, 방법 또는 컴퓨터 프로그램 제품으로서 구현될 수 있다. 따라서, 본 개시는 전

체가 하드웨어-포함 실시예, (펌웨어, 상주 소프트웨어, 마이크로-코드 등을 포함하는) 전체가 소프트웨어-포함 실시예, 또는 모두 일반적으로 본원에서 "회로", "모듈" 또는 "시스템"으로 지칭될 수 있는 소프트웨어 및 하드웨어 양상을 결합한 실시예의 형태를 취할 수 있다. 또한, 본 개시의 실시예는 매체에 구현된 컴퓨터 사용 가능 프로그램 코드를 갖는 표현의 임의의 유형의 매체에 구현된 컴퓨터 프로그램 제품의 형태를 취할 수 있다.

[0037] 하나 이상의 컴퓨터 사용가능 또는 컴퓨터 판독가능 매체의 임의의 조합이 이용될 수 있다. 예를 들어, 컴퓨터 판독가능 매체는 휴대용 컴퓨터 디스켓, 하드 디스크, 랜덤 액세스 메모리(random-access memory, RAM) 디바이스, 판독 전용 메모리(read-only memory, ROM) 디바이스, 삭제 가능 프로그램 가능 판독 전용 메모리(EPROM 또는 플래시 메모리) 디바이스, 휴대형 컴팩트 디스크 판독 전용 메모리(Portable Compact Disc Read-only Memory, CDROM), 광학 저장 디바이스 및 자기 저장 디바이스 중 하나 이상을 포함할 수 있다. 본 개시의 동작을 수행하기 위한 컴퓨터 프로그램 코드는 하나 이상의 프로그래밍 언어의 임의의 조합으로 기록될 수 있다. 이러한 코드는 소스 코드로부터, 코드가 실행될 디바이스 또는 컴퓨터에 적합한 컴퓨터 판독가능 어셈블리 언어 또는 기계 코드로 컴파일될 수 있다.

[0038] 실시예는 또한, 클라우드 컴퓨팅 환경에서 구현될 수 있다. 이 서술 및 다음의 청구범위에서, "클라우드 컴퓨팅"은 가상화를 통해 빠르게 제공되고 최소한의 관리 노력 또는 서비스 제공자와의 상호 작용으로 릴리즈되며(released), 그 후, 그에 따라 확장될 수 있는 구성 가능한 컴퓨팅 자원(예를 들어, 네트워크, 서버, 저장소, 애플리케이션 및 서비스)의 공유된 풀에 대해 어디에서나(ubiquitous), 편리하게, 온-디맨드(on-demand) 네트워크 액세스를 가능하게 하는 모델로서 정의될 수 있다. 클라우드 모델은 다양한 특성(예를 들어, 온-디맨드 셀프-서비스, 광범위한 네트워크 액세스, 자원 풀링, 빠른 탄력성 및 측정된 서비스), 서비스 모델(예를 들어, Software as a Service("SaaS"), Platform as Service("PaaS") 및 Infrastructure as a Service("IaaS")) 및 전개 모델(예를 들어, 개인용 클라우드, 커뮤니티 클라우드, 퍼블릭 클라우드 및 하이브리드 클라우드)로 구성될 수 있다.

[0039] 첨부도면에서 흐름도 및 블록도는 본 개시의 다양한 실시예에 따른 시스템, 방법 및 컴퓨터 프로그램 제품의 가능한 구현의 아키텍처, 기능 및 동작을 도시한다. 이에 관련하여, 흐름도 또는 블록도에서 각 블록은 지정된 논리 기능(들)을 구현하기 위한 하나 이상의 실행 가능한 명령어를 포함하는 코드의 모듈, 세그먼트 또는 일부를 나타낼 수 있다. 또한, 블록도 및/또는 흐름도의 각 블록, 및 블록도 및/또는 흐름도의 블록의 조합은 지정된 기능 또는 동작을 수행하는 특수 목적의 하드웨어-기반 시스템, 또는 특수 목적 하드웨어 및 컴퓨터 명령어의 조합에 의해 구현될 수 있다. 이들 컴퓨터 프로그램 명령어는 또한, 컴퓨터 판독가능 매체에 저장된 명령어가 흐름도 및/또는 블록도의 블록(들)에 지정된 기능/동작을 구현하는 명령어 수단을 포함하는 제조 물품을 생산하도록, 컴퓨터 또는 다른 프로그램 가능 데이터 처리 장치가 특정 방식으로 기능하도록 지시할 수 있는 컴퓨터 판독가능 매체에 저장될 수 있다.

[0040] 본원에 서술된 시스템 및 방법은 새로운 데이터 처리 플랫폼을 사용하여 유연하고 확장 가능한 데이터 창고를 제공한다. 일부 실시예에서, 서술된 시스템 및 방법은 클라우드 기반 저장소 자원, 컴퓨팅 자원 등을 지원하는 클라우드 인프라 구조를 이용한다. 예시적인 클라우드 기반 저장소 자원은 저렴한 비용으로 온-디맨드로 이용 가능한 상당한 저장소 용량을 제공한다. 또한, 이들 클라우드-기반 저장소 자원은 내결함성(fault-tolerant)이 있고 높은 확장성이 있을 수 있으며, 이는 개인용 데이터 저장 시스템에서 달성하는데 비용이 많이 들 수 있다. 예시적인 클라우드 기반 컴퓨팅 자원은 온-디맨드로 이용 가능하며 자원의 실제 사용 레벨을 기초로 가격이 매겨질 수 있다. 통상적으로, 클라우드 인프라구조는 빠른 방식으로 동적으로 전개되고, 재구성되며 및 폐기된다(decommissioned).

[0041] 서술된 시스템 및 방법에서, 데이터 저장 시스템은 SQL(Structured Query Language) 기반 관계형 데이터베이스를 사용한다. 하지만, 이들 시스템 및 방법은 데이터 저장 및 검색 플랫폼 내에서 데이터를 저장하고 검색하기 위해 임의의 데이터 저장 아키텍처를 사용하고 임의의 언어를 사용하는 임의의 타입의 데이터베이스, 및 임의의 타입의 데이터 저장 및 검색 플랫폼에 적용 가능할 수 있다. 본원에 서술된 시스템 및 방법은 상이한 고객/클라이언트 사이의 및 동일한 고객/클라이언트 내의 상이한 사용자 사이의 컴퓨팅 자원 및 데이터의 분리를 지원하는 다중 테넌트 시스템(multi-tenant system)을 더 제공한다.

[0042] 본 개시의 일 실시예에서, 배치 데이터 수집 서비스를 위한 시스템, 방법 및 디바이스가 서술된다. 배치 데이터 수집 서비스는 빈번한 데이터 로딩에 의한 데이터 저장소 벤더의 부담을 줄일 수 있다. 서비스는 데이터 저장소 클라이언트가 데이터 저장소 벤더에 데이터를 로딩하게 하는 것을 더욱 편리하게 할 수 있다.

[0043] 본 개시의 일 실시예에서, 배치 데이터 수집 서비스는 언어-특정 래핑 애플리케이션 프로그램 인터페이스(APi

s)를 통한 표현 상태 변경(Representational State Transfer, REST) 서비스이다. 서비스의 일 실시예는 로딩을 위한 파일이 제출된 후에 파일로의 데이터의 로딩이 나중에 발생하도록, 비동기식이다. 서비스의 일 실시예는 REST 호출이 리턴되기 전에, 파일 이름이 지속적인 저장소에 커밋되도록 영속성(durable)이 있다. 서비스의 일 실시예는 클라이언트에게 SQL 명령을 기록하지 않으면서 데이터를 삽입하는 기능을 제공하고, 데이터베이스에 데이터를 게시하고 복사하는 다중 단계 프로세스를 회피한다.

[0044] 이제 도면을 참조하면, 도 1은 배치 데이터 수집을 위한, 그리고 데이터 파이프 라인 시스템을 통해 하나 이상의 테이블에 데이터베이스 데이터를 저장하기 위한 시스템(100)의 개략적인 블록도이다. 도 1은 데이터 또는 제어의 흐름의 표현은 아니다. 시스템(100)은 데이터베이스 시스템과 통신할 수 있는 클라이언트 계정(102)을 포함한다. 이러한 클라이언트 계정(102)은 데이터베이스에 커밋되도록 새로운 데이터를 제공하거나 또는 데이터를 업데이트할 수 있다. 시스템(100)은 객체 변환(object resolution, 106) 및 요청 라우팅(108) 시스템을 포함하는 REST(표현 상태 변경) 계층(104)을 포함한다. 시스템(100)은 속도 제한(112) 및 인증(114) 시스템을 포함하는 API(애플리케이션 프로그램 인터페이스) 게이트웨이(110)를 포함한다. 시스템(100)은 토큰 관리(116) 프로토콜을 포함한다. 시스템은 REST 계층(104)과 통신하는 코어 엔진(120)을 포함한다. 코어 엔진(120)은 큐 관리(122), 작업 실행(124), 창고 관리(126), 파일 관리(128) 및 로드 이력(130)을 담당하는 시스템 또는 프로토콜을 포함한다. 코어 엔진(120)은 데이터 수집 작업과 같은 하나 이상의 작업을 실행하도록 구성된 실행 플랫폼(132)과 통신한다.

[0045] 클라이언트 계정(102)은 데이터베이스 시스템의 자원 관리자(예를 들어, 204, 302 참조)와 직접적으로 또는 간접적으로 통신한다. REST 계층(104)은 자원 관리자(302)의 구성요소일 수 있다. 클라이언트 계정(102)은 데이터베이스로 수집될 사용자 파일을 제공한다. 사용자 파일은 Amazon Web Services™ 또는 다른 적합한 클라우드 컴퓨팅 서비스와 같은 벤더 서비스에 업로드될 수 있다. 자원 관리자(302)는 데이터베이스로 수집되어야 하는 사용자 파일이 클라이언트 계정에 추가되었거나 데이터베이스 내의 일부 데이터가 업데이트 되어야한다는 통지를 수신할 수 있다. 다양한 구현에서, 자원 관리자(302)는 임의의 사용자 파일이 추가되었는지를 결정하기 위해, 이러한 통지를 자동으로 수신하거나, 또는 클라이언트 계정(102)과 연관된 데이터 버킷을 주기적으로 폴링(poll)할 수 있다.

[0046] REST 계층(104)은 페이로드 변환(translation)을 내부 포맷으로 처리할 수 있는 얇은 외부 계층(thin outer layer)을 포함하고, 간단한 검증을 더 처리할 수 있다. 일 실시예에서, REST 계층(104)은 자원 관리자 내에 존재한다(예를 들어, 204, 302 참조). REST 계층(104)은 범위가 지정된(scoped) 테이블 이름으로부터 테이블 식별(본원에서 TableID로 지칭될 수 있음)로의 전환을 담당하는 객체 변환(106)을 포함한다. REST 계층(104)은 클라이언트 계정(102)으로부터 수신된 사용자 파일의 목적지 테이블에 대한 요청을 자원 관리자(302)의 적절한 인스턴스로 라우팅하는 것을 담당하는 요청 라우팅(108)을 더 포함한다. 일 실시예에서, 요청 라우팅(108)은 객체 변환(106) 이후에 발생한다. 요청 라우팅(108)은 어느 GS 인스턴스가 어느 테이블을 소유하는지를 관리하기 위해, 가상 노드와 일치하는 해싱을 사용할 수 있다.

[0047] 일 실시예에서, (사용자 파일을 수신하는 것을 담당하는 제3자 계정과 같은) 벤더 계정이 사용자 파일의 하나 이상의 이름을 수신할 때, 객체 변환(106) 프로토콜은 사용자 파일의 이름을 내부 이름으로 변환한다(resolves). 사용자 파일에 대한 내부 이름이 캐시된다.

[0048] 일 실시예에서, 자원 관리자(302)의 REST 계층(104)의 요청 라우팅(108) 프로토콜은 클라이언트 계정(102)으로부터 사용자 파일을 수신하고, 그 사용자 파일을 수집 및 처리를 위해 실행 플랫폼의 하나 이상의 실행 노드로 라우팅하도록 구성된다. 일 실시예에서, 벤더 계정(즉, 클라이언트 계정(102)으로부터 직접적으로 또는 간접적으로 사용자 파일을 수신하고 이들 사용자 파일을 예를 들어 자원 관리자(302)에 제공하는 것을 담당하는 제3자 계정)은 어느 자원 관리자(302)가 특정 데이터베이스 테이블을 소유하는지를 관리하기 위해 가상 노드와 일치하는 해싱을 사용할 수 있다. 벤더 계정은 사용자 파일과 자원 관리자(302) 사이의 매칭을 찾기 위해, 테이블 식별 및 특정 자원 관리자(302)에 대한 식별을 해시할 수 있다. 해시 공간은 동일하게 크기화된 파티션으로 나누어진다. 다수의 자원 관리자(302) 인스턴스가 주어지면, 각 자원 관리자(302)는 다수의 파티션을 취한다. 자원 관리자(302)가 (코어 엔진(120)과 같은) 실행 플랫폼에 실행 노드를 추가할 때, 자원 관리자(302)는 파티션 할당량(partition rations)의 비율을 유지하기 위해 각 실행 노드로부터 랜덤 파티션을 풀링할 것이다. 마찬가지로, 실행 노드에 장애가 발생할 때, 파티션의 할당량은 나머지 실행 노드로 분산된다. 이 매핑은 벤더 계정에 의해 유지될 수 있다.

[0049] 일 실시예에서, 모든 가상 노드 매핑을 포함하는 데이터베이스 레코드는 가상 노드 할당이 변경될 때마다 트랜

잭션적으로 수정된다. 가상 노드와 연관된 값은 데이터베이스의 주어진 테이블에 대한 요청을 처리하는 오브젝트와 함께 메모리에 저장될 것이다. 값은 자원 관리자(302)와 실행 플랫폼(132) 사이의 모든 상호작용의 앞뒤로 전달될 수 있다. 값은 폐기될 수 있는 오래된 상태 또는 요청을 검출하는 데 사용될 수 있다.

[0050] 일 실시예에서, tableList 슬라이스는 복구 상황 동안 벤더 계정과 통신하기 위해 사용된다. tableList 슬라이스는 어떤 데이터베이스 테이블이 주어진 가상 노드에 의해 관리되는지를 나타낼 수 있다. tableList 슬라이스는 필요에 따라 추가되거나 수정될 수 있으며, 테이블에 대한 벤더 계정 통지가 삭제되었거나 더 이상 활성화되지 않을 때, 최선의 노력으로 비워질 수 있다.

[0051] 일 실시예에서, 전체 가상 노드 테이블은 각 자원 관리자(302)상의 메모리에 캐시될 수 있다. 각 자원 관리자(302)는 데이터베이스 테이블에 대한 변경을 관찰하고(watch), 데이터베이스 테이블을 백업 조치(backup measure)로서 주기적으로 폴링할 수 있다. 일 실시예에서, 새로운 사용자 파일을 수집하기 위한 요청이 요청 라우팅(108) 프로토콜에 의해 라우팅되어야할 때, 벤더 계정은 tableId를 해시하고 해시를 포함하는 가상 노드를 결정할 수 있으며, 그 후 벤더 계정은 테이블에서 가상 노드를 룩업(lookup)하고, 특정 자원 관리자(302)로 라우팅할 수 있다.

[0052] API 게이트웨이(110)는 코어 엔진(120)으로의 액세스를 보호하기 위해 얇은 계층을 포함한다. API 게이트웨이(110)는 데이터 수집 요청의 대량 유입(influx)을 방지하기 위해 기본 제한을 담당하는 속도 제한(112)을 포함한다. API 게이트웨이(110)는 REST 요청에서 전달된 API 토큰을 검증하는 것을 담당하는 인증(114)을 포함한다.

[0053] 일 실시예에서, 인증(114) 프로토콜은 범위가 감소된 크리덴셜(down-scoped credentials)의 세트를 포함한다. 범위가 감소된 크리덴셜은 특정 테이블에 대해 범위가 지정된 API 토큰을 생성하는데 사용될 수 있다. 토큰은 이에 베이킹된(baked) TableID를 가질 수 있으며, 프로그래밍 방식으로 생성될 수 있다. 토큰은 짧은 수명을 가질 수 있다(일 실시예에서, 토큰은 30분, 1 시간, 2 시간, 3 시간 등 이후에 만료될 수 있다). 일 실시예에서, 클라이언트 계정(102)은 토큰의 만료를 지시하거나 및/또는 프로그래밍 방식으로 새로운 토큰을 수신할 수 있다. 일 실시예에서, 시스템(100)은 토큰을 수신하고, 토큰을 검증하고, 토큰에 지정된 TableID가 REST 요청에 지정된 테이블의 이름과 매칭하는지를 검증한다. REST 요청에 지정된 TableID 및 테이블이 매칭하지 않는 일 구현에서, 호출자는 새로운 토큰을 요청할 특정한 오류 응답을 수신할 것이다. 일 실시예에서, 시스템(100)은 테이블이 수정될 때마다 새로운 토큰을 요구한다.

[0054] 토큰 관리(116)는 요구 시(on demand) 새로운 토큰을 생성하고 이전 토큰을 취소하는 것을 담당한다. 코어 엔진(120)은 들어오는 데이터의 처리를 관리하는 코어 로직이다. 코어 엔진(120)은 큐(queue)로부터 파일을 추가하거나 또는 제거하는 것을 포함하여, 들어오는 파일의 큐를 관리하는 것을 담당하는 큐 관리(122)를 포함한다. 작업 실행기(124)는 컴파일러와의 상호 작용을 포함하여, 파일을 로딩하기 위한 실행 플랫폼 작업(jobs)을 개시하고 관리한다. 창고 관리(126)는 요구 시, 확장 및 축소를 포함하여, 로딩 창고를 관리한다. 파일 관리(128)는 네이티브 이진 파일(native binary files)을 등록하고 오류를 캡처하는 수집 버전을 처리하는 것을 담당한다. 로드 이력(130)은 주어진 테이블에 대한 로드 및 오류의 이력을 추적한다. 로드 이력(130)은 일 기간 이후 또는 최대 개수의 엔트리에 도달한 이후의 로드 이력을 더 제거할 수 있다(purge).

[0055] 일 실시예에서, 작업 실행기(124)는 활성화된 작업의 현재 총 개수 및 활성화된 작업의 원하는 개수를 안다. 작업 실행기(124)는 창고의 크기를 활성화된 작업의 원하는 개수로 유지하도록 노력할 자원 관리자(302)에게 활성화된 작업의 원하는 개수를 전달할 것이다. 자원 관리자(302)는 원하는 레이턴시의 단편인 일부 기간에 걸쳐 이동 평균에 의해 시간에 따른 요구를 평활화(smoothing)함으로써 이를 달성할 수 있다. 자원 관리자(302)는 창고의 크기를 일시적 스파이크를 수용할 실제 필요성보다 약간 크게 유지함으로써 더 달성할 수 있다. 자원 관리자(302)는 필요할 때 실행 노드의 합리적인 해제(freeing)를 허용하도록 사용량을 압축하는 방식으로 실행 노드를 신중하게 릴리즈하거나 및/또는 하나 이상의 실행 노드에 작업을 할당함으로써 이를 더 달성할 수 있다.

[0056] 일 실시예에서, 작업 실행기(124)는 수집 작업의 실행 계획을 생성한다. 실행 계획은 복사 명령의 계획에 유사할 수 있다. 작업 실행기(124)는 내부 불린 옵션(Boolean option) "ingest_mode"를 포함하는 복사 옵션의 코드 변경을 현재 복사 명령으로 생성할 수 있다. 실행 계획은 특정 기능을 비활성화하기 위해 SQL 텍스트 "copy into T ingest_mode = true"로부터 컴파일할 수 있다. 작업 실행기(124)는 사본이 수집 모드에 있는 경우, 참일 수 있는 불린 특성 "dynamic_scanset"을 포함하는 스캔 세트의 코드 변경을 더 포함할 수 있다.

[0057] 일 실시예에서, 창고 관리(126)는 데이터 수집을 위한 창고를 관리한다. 창고 관리(126)는 요구를 기초로 확장 및 축소를 제어하고, 작업을 실행 노드에 할당하고, 올바른 할당을 허용하기 위해 창고에서 작업의 상태를 추적

하며, 장애가 있는 서버를 추적하고 그에 따라 응답할 수 있다. 일 실시예에서, 창고 관리(126)는 자원 관리자(302)에 통합된다. 수집 작업이 단일 스레드로 이루어지기 때문에, 하나의 창고 노드에서 코어 당 하나의 작업이 할당될 것이다. 각 창고 노드에 대해, 구동되는 작업의 개수가 추적된다. 작업 실행기(124)는 새로운 작업을 스케줄링하고, 서버가 사용할 창고 관리(126)를 요청하며, 창고 관리(126)는 로드가 감소할 때 실행 노드를 더 쉽게 해제 할 수 있도록 이미 사용중인(busy) 서버를 선택할 것이다. 작업 실행기(124)는 창고 관리(126)에게 작업 완료를 알려야 한다.

[0058] 일 실시예에서, 로드 이력(130)은 로딩 결과를 모니터링하고, 파일 또는 데이터가 데이터베이스로 성공적으로 수집되었는지를 추적한다. 수집 이력은 데이터베이스 내의 또는 데이터베이스로부터 분리되고 자원 관리자(302)에 의해 액세스 가능한 메타데이터 저장소에 더 저장될 수 있다. 수집 기록은 예를 들어, 파일 이름, TableID, 파일 크기, 행 카운트, 상태 및 제1 오류를 포함한다. 일 실시예에서, 데이터 로딩의 오류 관리는 분리된 프로젝트일 것이다.

[0059] 도 2는 데이터베이스로 데이터를 수집하는 프로세스(200)의 개략적인 블록도이다. 프로세스(200)가 개시되고, 클라이언트 계정(102)은 202에서 수집 요청을 전송한다. 클라이언트 계정(102)은 수집 요청을 전송하기 위해 데이터베이스 시스템과 직접적으로 또는 간접적으로 통신할 수 있다. 일 실시예에서, 수집 요청은 제3자 벤더 저장소 계정에 의해 제공되는 통지이거나, 또는 수집 요청은 데이터베이스로 아직 수집되지 않은 임의의 사용자 파일이 클라이언트 계정(102)에 추가되었는지를 결정하기 위해 클라이언트 계정(102)과 연관된 데이터 레이크(data lake)를 폴링하는 자원 관리자(302)로부터 발생할 수 있다. 통지는 데이터베이스 테이블에 삽입할 사용자 파일의 목록을 포함한다. 사용자 파일은 데이터베이스의 수신 테이블에 특정한 큐에서 지속된다.

[0060] 수집 요청은 자원 관리자(204)에 의해 수신된다(또한, 302 참조). 자원 관리자(204)는 206에서 수집할 사용자 파일을 식별하고, 208에서 사용자 파일을 하나 이상의 실행 노드에 할당하고, 210에서 사용자 파일이 데이터베이스 테이블의 마이크로-파티션으로 수집된 이후에 데이터베이스 테이블과 연관된 마이크로-파티션 메타데이터를 등록한다. 자원 관리자(204)는 사용자 파일을 수집하는 것과 연관된 하나 이상의 작업을 수행하기 위해 실행 플랫폼(212)의 하나 이상의 실행 노드(214, 218)를 제공한다. 이러한 수집 작업(216a, 216b, 220a, 220b)은 예를 들어, 사용자 파일을 하나 이상의 파티션으로 분할하는 것(cutting), 사용자 파일을 기초로 새로운 마이크로-파티션을 생성하는 것 및/또는 새로운 마이크로-파티션을 데이터베이스의 테이블에 삽입하는 것을 포함한다.

[0061] 시스템(200)은 창고 상에서 구동될 IngestTask를 개시한다. IngestTask는 그렇게 하는 것을 멈추라는 알림을 받을 때까지, 데이터베이스 테이블에 대한 큐로부터 사용자 파일을 풀링할 것이다. IngestTask는 새로운 사용자 파일을 주기적으로 분할하며, 이를 데이터베이스 테이블에 추가할 것이다. 일 실시예에서, 수집 프로세스는 데이터베이스 또는 자원 관리자(204)에 의해 제공되는 통합된 서비스라는 점에서 "서버리스"이다. 즉, 클라이언트 계정(102)과 연관된 사용자는 수집 프로세스를 수행하기 위해 그 자신의 창고 또는 제3자 창고를 제공할 필요는 없다. 예를 들어, 데이터베이스 또는 (예를 들어, 자원 관리자(204)의 인스턴스를 통해) 제공된 데이터베이스는 그 후 데이터베이스 제공자의 하나 이상의 또는 모든 계정/고객을 서비스하는 수집 창고를 유지할 수 있다.

[0062] 주어진 테이블에 대해 큐로부터 하나보다 많은 IngestTask를 풀링할 있을 수 있음이 인식되어야 하며, 이는 들어오는 데이터의 속도를 유지하는 데 필요할 수 있다. 일 실시예에서, IngestTask는 이상적인 크기의 파일을 얻을 가능성을 높이고, 파일 크기가 하나 이상의 사용자 파일과 정렬될 경우 발생할 수 있는 "홀수 크기" 파일을 피하기 위해, 새로운 파일을 분할할 시간을 결정할 수 있다. 소비되는 파일의 추적 라인 번호가 추적되어야 하기 때문에, 이는 복잡성을 추가시킬 수 있다.

[0063] 일 실시예에서, 특정 테이블에 대한 모든 요청은 자원 관리자(204)의 단일 인스턴스로 라우팅될 것이다. 자원 관리자(204)의 각 인스턴스는 데이터베이스 테이블의 세트를 담당할 수 있다. 일 실시예에서, 이는 노드가 큐에 대한 동시 기록(write-through) 캐시로서 처리되는 것을 허가하는 가상 노드와 일치하는 해싱을 사용함으로써 달성되며, 이는 메타데이터 저장소로부터 큐에서의 항목을 판독할 필요성을 제거한다.

[0064] 이제 도 3을 참조하면, 본원에 개시된 방법을 구동시키기 위한 데이터 처리 플랫폼(300)이 도시된다. 도 3에 도시된 바와 같이, 자원 관리자(302)는 다수의 클라이언트 계정(314a, 314b 및 314n)에 결합될 수 있다. 특정 구현에서, 자원 관리자(302)는 실행 플랫폼(304) 및/또는 공유된 데이터베이스 저장소(308)에 대한 액세스를 원하는 임의의 개수의 클라이언트 계정을 지원할 수 있다. 클라이언트 계정(314a, 314b 및 314n)은 예를 들어, 데이터베이스에 수집될 사용자 파일, 데이터 저장 및 검색 요청을 제공하는 최종 사용자, 본원에 서술된 시스템 및 방법을 관리하는 시스템 관리자, 및 자원 관리자(302)와 상호작용하는 다른 구성요소/디바이스를 포함할 수 있다.

[0065] 자원 관리자(302)는 데이터 처리 플랫폼(300) 내의 모든 시스템 및 구성요소의 동작을 지원하는 다양한 서비스 및 기능을 제공할 수 있다. 자원 관리자(302)는 데이터 처리 플랫폼(300) 전체에 걸쳐 저장된 전체 데이터와 연관된 공유된 메타데이터(312)에 결합될 수 있다. 일부 실시예에서, 공유된 메타데이터(312)는 원격 데이터 저장 시스템에 저장된 데이터의 요약뿐만 아니라 로컬 캐시로부터 이용 가능한 데이터를 포함할 수 있다. 게다가, 공유된 메타데이터(312)는 원격 데이터 저장 시스템 및 로컬 캐시에서 데이터가 어떻게 구성되는지에 관한 정보를 포함할 수 있다. 공유된 메타데이터(312)는 시스템 및 서비스가 저장 디바이스로부터 실제 데이터를 로딩하거나 또는 액세스하지 않으면서 데이터(a piece of data)가 처리되어야 하는지를 결정하는 것을 허용할 수 있다.

[0066] 자원 관리자(302)는 아래에서 더 상세히 논의되는 바와 같이 다양한 데이터 저장 및 데이터 검색 작업을 실행하는 다수의 컴퓨팅 자원을 제공하는 실행 플랫폼(304)에 더 결합될 수 있다. 실행 플랫폼(304)은 새로운 사용자 파일을 수집하는 것 및 새로운 사용자 파일을 기초로 데이터베이스의 테이블에 대한 하나 이상의 마이크로-파티션을 생성하는 것을 포함하여, 데이터베이스와 관련된 다양한 작업을 처리하도록 구성된 복수의 실행 노드(306a, 306b, 306c 및 306n)를 포함한다. 실행 플랫폼(304)은 다수의 데이터 저장 디바이스(310a, 310b, 310c 및 310n)를 포함하여 공유된 데이터베이스 저장소(308)에 결합될 수 있다. 일부 실시예에서, 공유된 데이터베이스 저장소(308)는 하나 이상의 지리적인 위치에 위치된 클라우드 기반 저장 디바이스를 포함한다. 예를 들어, 공유된 데이터베이스 저장소(308)는 공용 클라우드 인프라구조 또는 개인용 클라우드 인프라구조의 일부일 수 있다. 공유된 데이터베이스 저장소(308)는 하드 디스크 드라이브(HDD), 고체 상태 드라이브(solid state drives, SSD), 저장소 클러스터 또는 임의의 다른 데이터 저장 기법을 포함할 수 있다. 게다가, 공유된 데이터베이스 저장소(308)는 (예를 들어, HDFS(Hadoop Distributed File Systems)와 같은) 분산 파일 시스템, 객체 저장 시스템 등을 포함할 수 있다. 공유된 데이터베이스 저장소(308)는 자원 관리자(302)의 하나 이상의 인스턴스에 의해 액세스 가능할 수 있지만 모든 클라이언트 계정(314a-314n)에 의해 액세스 가능하지 않을 수 있음이 인식되어야 한다. 일 실시예에서, 자원 관리자(302)의 단일 인스턴스는 복수의 클라이언트 계정(314a-314n)에 의해 공유된다. 일 실시예에서, 각 클라이언트 계정(314a-314n)은 그 자신의 자원 관리자 및/또는 실행 플랫폼(304)의 복수의 실행 노드(306a-306n)간에 공유되는 그 자신의 공유된 데이터베이스 저장소(308)를 갖는다. 일 실시예에서, 자원 관리자(302)는 공유된 데이터베이스 저장소(308) 내의 특정 데이터에 대해 특정 클라이언트 계정(314a-314n) 액세스를 제공하는 것을 담당한다.

[0067] 특정 실시예에서, 자원 관리자(302)와 클라이언트 계정(314a-314n), 공유된 메타데이터(312) 및 실행 플랫폼(304) 사이의 통신 링크는 하나 이상의 데이터 통신 네트워크를 통해 구현된다. 유사하게, 실행 플랫폼(304)과 공유된 데이터베이스 저장소(308) 사이의 통신 링크는 하나 이상의 데이터 통신 네트워크를 통해 구현된다. 이들 데이터 통신 네트워크는 임의의 통신 프로토콜 및 임의의 타입의 통신 매체를 이용할 수 있다. 일부 실시예에서, 데이터 통신 네트워크는 서로 결합된 두 개 이상의 데이터 통신 네트워크(또는 서브 네트워크)의 조합이다. 대안적인 실시예에서, 이들 통신 링크는 임의의 타입의 통신 매체 및 임의의 통신 프로토콜을 사용하여 구현된다.

[0068] 도 3에 도시된 바와 같이, 데이터 저장 디바이스(310a-310n)는 실행 플랫폼(304)과 연관된 컴퓨팅 자원으로부터 결합 해제된다. 이 아키텍처는 데이터 처리 플랫폼(300)에 액세스하는 사용자 및 시스템의 변화 요구뿐만 아니라 변하는 데이터 저장/검색 요구를 기초로 데이터 처리 플랫폼(300)에 대한 동적 변경을 지원한다. 동적 변경의 지원은 데이터 처리 플랫폼(300)이 데이터 처리 플랫폼(300) 내의 시스템 및 구성요소에 대한 변하는 요구에 응답하여 빠르게 확장되는 것을 허용한다. 데이터 저장 디바이스로부터 컴퓨팅 자원의 결합 해제는 대응하는 많은 양의 컴퓨팅 자원을 요구하지 않으면서 대량의 데이터의 저장을 지원한다. 유사하게, 자원의 이 결합 해제는 이용 가능한 데이터 저장 자원의 대응하는 증가를 요구하지 않으면서, 특정 시간에 이용되는 컴퓨팅 자원의 상당한 증가를 지원한다.

[0069] 자원 관리자(302), 공유된 메타데이터(312), 실행 플랫폼(304) 및 공유된 데이터베이스 저장소(308)가 개별적인 구성요소로 도 3에 도시된다. 하지만, 자원 관리자(302), 공유된 메타데이터(312), 실행 플랫폼(304) 및 공유된 데이터베이스 저장소(308)의 각각은 분산 시스템(예를 들어, 다수의 지리적 위치에서 다수의 시스템/플랫폼에 걸쳐 분산됨)으로서 구현될 수 있다. 게다가, 자원 관리자(302), 공유된 메타데이터(312), 실행 플랫폼(304) 및 공유된 데이터베이스 저장소(308)의 각각은 클라이언트 계정(314a-314n)으로부터 수신된 요청에 대한 변경 및 데이터 처리 플랫폼(300)의 변하는 요구에 의존하여 (서로 독립적으로) 확장되거나 또는 축소될 수 있다. 따라서, 데이터 처리 플랫폼(300)은 동적이며, 현재 데이터 처리 요구를 충족시키기 위해 규칙적인 변경을 지원한다.

[0070] 도 4는 자원 관리자(302)의 일 실시예를 도시하는 블록도이다. 도 4에 도시된 바와 같이, 자원 관리자(302)는

데이터 저장 디바이스(406)에 결합된 액세스 관리자(402) 및 키 관리자(404)를 포함한다. 액세스 관리자(402)는 본원에 서술된 시스템에 대한 인증 및 허가(authorization) 작업을 처리할 수 있다. 키 관리자(404)는 인증 및 허가 작업 동안 사용된 키의 저장 및 인증을 관리할 수 있다. 요청 처리 서비스(408)는 수신된 데이터 저장 요청 및 데이터 검색 요청을 관리한다. 관리 콘솔 서비스(410)는 관리자 및 다른 시스템 관리자에 의한 다양한 시스템 및 프로세스에 대한 액세스를 지원한다.

[0071] 자원 관리자(302)는 또한, 작업 컴파일러(412), 작업 최적화기(414) 및 작업 실행기(416)를 포함할 수 있다. 작업 컴파일러(412)는 수집 작업(ingest tasks)과 같은 작업을 파싱하고(parses), 사용자 파일의 수집을 위한 실행 코드를 생성한다. 작업 최적화기(414)는 처리되어야 하거나 및/또는 수집되어야 하는 데이터를 기초로 수집 작업을 실행하기 위한 최선의 방법을 결정한다. 작업 실행기(416)는 자원 관리자(302)에 의해 수신된 수집 작업에 대한 코드를 실행한다. 작업 스케줄러 및 코디네이터(coordinator, 418)는 수신된 사용자 파일을 컴파일, 최적화 및 실행 플랫폼(304)으로 보내기(dispatch) 위해 적절한 서비스 또는 시스템으로 전송할 수 있다. 가상 창고 관리자(420)는 실행 플랫폼에서 구현된 다수의 가상 창고의 동작을 관리한다.

[0072] 게다가, 자원 관리자(302)는 원격 데이터 저장 디바이스에 및 로컬 캐시에 저장된 데이터에 관련된 정보를 관리하는, 구성 및 메타데이터 관리자(422)를 포함한다. 모니터 및 작업 부하 분석기(424)는 자원 관리자(302)에 의해 수행된 프로세스를 감독하고, 실행 플랫폼에서 가상 창고 및 실행 노드에 걸친 작업(예를 들어, 작업 부하)의 분산을 관리한다. 구성 및 메타데이터 관리자(422) 및 모니터 및 작업 부하 분석기(424)는 데이터 저장 디바이스(426)에 결합된다.

[0073] 도 5는 실행 플랫폼(304)의 일 실시예를 도시하는 블록도이다. 도 5에 도시된 바와 같이, 실행 플랫폼(304)은 가상 창고 1, 가상 창고 2 및 가상 창고 n을 포함하는 다수의 가상 창고를 포함한다. 각 가상 창고는 각각 데이터 캐시 및 프로세서를 포함하는 다수의 실행 노드를 포함한다. 가상 창고는 다수의 실행 노드를 사용함으로써 다수의 작업을 병렬로 실행할 수 있다. 본원에서 논의된 바와 같이, 실행 플랫폼(304)은 시스템 및 사용자의 현재 처리 요구를 기초로, 새로운 가상 창고를 추가하고 기존 가상 창고를 실시간으로 드롭(drop)할 수 있다. 이 유연성은 이들이 더 이상 필요하지 않을 때 이들 컴퓨팅 자원에 대한 비용을 계속 지불하도록 강제되지 않으면서, 필요할 때 실행 플랫폼(304)이 대량의 컴퓨팅 자원을 빠르게 전개하는 것을 허용한다. 모든 가상 창고는 임의의 데이터 저장 디바이스(예를 들어, 공유된 데이터베이스 저장소(308)의 임의의 저장 디바이스)로부터 데이터에 액세스할 수 있다. 도 5에 도시된 각 가상 창고가 세 개의 실행 노드를 포함하지만, 특정한 가상 창고는 임의의 개수의 실행 노드를 포함할 수 있다. 또한, 추가적인 요구가 존재할 때 새로운 실행 노드가 생성되고 이들이 더 이상 필요하지 않을 때 기존의 실행 노드가 삭제되도록, 가상 창고에서 실행 노드의 개수는 동적이다.

[0074] 각 가상 창고는 도 3에 도시된 데이터 저장 디바이스(310a-310n) 중 어느 것에 액세스할 수 있다. 따라서, 가상 창고는 특정 데이터 저장 디바이스에 할당될 필요는 없으며, 그 대신에 공유된 데이터베이스 저장소(308) 내의 데이터 저장 디바이스(310a-310n) 중 어느 것으로부터 데이터에 액세스할 수 있다. 유사하게, 도 5에 도시된 실행 노드의 각각은 데이터 저장 디바이스(310a-310n) 중 어느 것으로부터 데이터에 액세스할 수 있다. 일부 실시예에서, 특정 가상 창고 또는 특정 실행 노드가 특정 데이터 저장 디바이스에 일시적으로 할당될 수 있지만, 가상 창고 또는 실행 노드는 나중에 임의의 다른 데이터 저장 디바이스로부터 데이터에 액세스할 수 있다.

[0075] 도 5의 예시에서, 가상 창고 1은 세 개의 실행 노드(502a, 502b 및 502n)를 포함한다. 실행 노드(502a)는 캐시(504a) 및 프로세서(506a)를 포함한다. 실행 노드(502b)는 캐시(504b) 및 프로세서(506b)를 포함한다. 실행 노드(502n)는 캐시(504n) 및 프로세서(506n)를 포함한다. 각 실행 노드(502a, 502b 및 502n)는 하나 이상의 데이터 저장 및/또는 데이터 검색 작업을 처리하는 것과 연관된다. 예를 들어, 가상 창고는 클러스터링 서비스, 구체화된 뷰 새로고침 서비스, 파일 압축 서비스, 저장 절차 서비스(storage procedure service) 또는 파일 업그레이드 서비스와 같이, 내부 서비스와 연관된 데이터 저장 및 데이터 검색 작업을 처리할 수 있다. 다른 구현에서, 특정 가상 창고는 특정 데이터 저장 시스템 또는 데이터의 특정 카테고리와 연관된 데이터 저장 및 데이터 검색 작업을 처리할 수 있다.

[0076] 위에서 논의한 가상 창고 1과 유사하게, 가상 창고 2는 세 개의 실행 노드(512a, 512b 및 512n)를 포함한다. 실행 노드(512a)는 캐시(514a) 및 프로세서(516a)를 포함한다. 실행 노드(512b)는 캐시(514b) 및 프로세서(516b)를 포함한다. 실행 노드(512n)는 캐시(514n) 및 프로세서(516n)를 포함한다. 게다가, 가상 창고 3은 세 개의 실행 노드(522a, 522b 및 522n)를 포함한다. 실행 노드(522a)는 캐시(524a) 및 프로세서(526a)를 포함한다. 실행 노드(522b)는 캐시(524b) 및 프로세서(526b)를 포함한다. 실행 노드(522n)는 캐시(524n) 및 프로세서(526n)를 포함한다.

[0077] 일부 실시예에서, 도 5에 도시된 실행 노드는 실행 노드가 캐싱하는 데이터와 관련하여 상태가 없다(stateless). 예를 들어, 이들 실행 노드는 실행 노드에 대한 상태 정보 또는 특정 실행 노드에 의해 캐시되는 데이터를 저장하지 않거나 다른 방식으로 유지하지 않는다. 따라서 실행 노드 장애의 경우, 장애가 발생한 노드는 다른 노드로 명료하게 교체될 수 있다. 장애가 있는 실행 노드와 연관된 상태 정보가 없기 때문에, 새로운(교체) 실행 노드는 특정 상태를 다시 생성할 염려 없이, 장애가 있는 노드를 쉽게 교체할 수 있다.

[0078] 도 5에 도시된 실행 노드 각각이 하나의 데이터 캐시 및 하나의 프로세서를 포함하지만, 대안적인 실시예는 임의의 개수의 프로세서 및 임의의 개수의 캐시를 포함하는 실행 노드를 포함할 수 있다. 게다가, 캐시는 상이한 실행 노드마다 크기가 다를 수 있다. 도 5에 도시된 캐시는 공유된 데이터베이스 저장소(308)에서 하나 이상의 데이터 저장 디바이스로부터 검색된 데이터를 로컬 실행 노드에 저장한다. 따라서, 캐시는 원격 저장 시스템으로부터 데이터를 지속적으로 검색하는 플랫폼에서 발생하는 병목 현상 문제를 감소시키거나 또는 제거한다. 원격 저장 디바이스로부터 데이터에 반복적으로 액세스하는 것 대신에, 본원에 서술된 시스템 및 방법은 실행 노드에서 캐시로부터 데이터에 액세스하며, 이는 상당히 더 빠르며 앞서 논의된 병목 문제를 회피한다. 일부 실시예에서, 캐시는 캐시된 데이터에 대한 빠른 액세스를 제공하는 고속 메모리 디바이스를 사용하여 구현된다. 각 캐시는 저장 디바이스 중 어느 것으로부터 데이터를 공유된 데이터베이스 저장소(308)에 저장할 수 있다.

[0079] 또한, 캐시 자원 및 컴퓨팅 자원은 상이한 실행 노드 사이에서 변할 수 있다. 예를 들어, 하나의 실행 노드는 상당한 컴퓨팅 자원 및 최소 캐시 자원을 포함할 수 있으며, 이는 실행 노드를 상당한 컴퓨팅 자원을 요구하는 작업에 대해 유용하게 한다. 다른 실행 노드는 상당한 캐시 자원 및 최소한의 컴퓨팅 자원을 포함할 수 있으며, 이는 실행 노드를 많은 양의 데이터의 캐싱을 필요로 하는 작업에 대해 유용하게 한다. 또 다른 실행 노드는 더욱 빠른 입출력 동작을 제공하는 캐시 자원을 포함할 수 있으며, 이는 대량의 데이터의 빠른 스캐닝을 필요로 하는 작업에 대해 유용하다. 일부 실시예에서, 특정 실행 노드와 연관된 캐시 자원 및 컴퓨팅 자원은 실행 노드에 의해 수행될 것으로 예상되는 작업을 기초로 실행 노드가 생성될 때 결정된다.

[0080] 게다가, 특정한 실행 노드와 연관된 캐시 자원 및 컴퓨팅 자원은 실행 노드에 의해 수행되는 작업을 변경하는 것을 기초로 시간에 따라 변할 수 있다. 예를 들어, 실행 노드에 의해 수행된 작업이 더욱 프로세서-집약적(processor-intensive)이게 되면 실행 노드는 더욱 많은 처리 자원을 할당 받을 수 있다. 유사하게, 실행 노드에 의해 수행된 작업이 더욱 큰 캐시 용량을 필요로 하는 경우, 실행 노드는 더욱 많은 캐시 자원을 할당 받을 수 있다.

[0081] 가상 창고 1, 2 및 n이 동일한 실행 플랫폼(304)과 연관되더라도, 가상 창고는 다수의 지리적 위치에서 다수의 컴퓨팅 시스템을 사용하여 구현될 수 있다. 예를 들어, 가상 창고 1은 제1 지리적 위치에서 컴퓨팅 시스템에 의해 구현될 수 있는 한편, 가상 창고 2 및 n은 제2 지리적 위치에서 다른 컴퓨팅 시스템에 의해 구현된다. 일부 실시예에서, 이들 상이한 컴퓨팅 시스템은 하나 이상의 상이한 엔티티에 의해 유지되는 클라우드 기반 컴퓨팅 시스템이다.

[0082] 게다가, 각 가상 창고는 도 5에서 다수의 실행 노드를 갖는 것으로 도시된다. 각 가상 창고와 연관된 다수의 실행 노드는 다수의 지리적인 위치에서 다수의 컴퓨팅 시스템을 사용하여 구현될 수 있다. 예를 들어, 가상 창고 1의 인스턴스는 지리적 위치에서 하나의 컴퓨팅 플랫폼 상에서 실행 노드(502a 및 502b)를 구현하고 다른 지리적 위치에서 상이한 컴퓨팅 플랫폼에서 실행 노드(502n)를 구현한다. 실행 노드를 구현하기 위해 특정 컴퓨팅 시스템을 선택하는 것은 특정 실행 노드에 필요한 자원의 레벨(예를 들어, 처리 자원 요건 및 캐시 요건), 특정 컴퓨팅 시스템에서 이용 가능한 자원, 지리적 위치 내에서 또는 지리적 위치 사이의 네트워크의 통신 능력, 및 어느 컴퓨팅 시스템이 가상 창고에서 다른 실행 노드를 이미 구현하였는지와 같은 다양한 인자에 의존할 수 있다.

[0083] 실행 플랫폼(304)은 또한 내결함성이다. 예를 들어, 하나의 가상 창고에 장애가 발생하면, 해당 가상 창고는 상이한 지리적 위치에서 상이한 가상 창고로 신속하게 교체된다.

[0084] 특정 실행 플랫폼(304)은 임의의 개수의 가상 창고를 포함할 수 있다. 게다가, 추가적인 처리 및/또는 캐싱 자원이 필요할 때 새로운 가상 창고가 생성되도록, 특정 실행 플랫폼에서 가상 창고의 개수는 동적이다. 유사하게, 가상 창고와 연관된 자원이 더 이상 필요하지 않을 때, 기존의 가상 창고는 삭제될 수 있다.

[0085] 일부 실시예에서, 가상 창고는 공유된 데이터베이스 저장소(308)에서 동일한 데이터에 대해 동작할 수 있지만, 각 가상 창고는 독립적인 처리 및 캐싱 자원을 갖는 그 소유의 실행 노드를 갖는다. 이 구성은 상이한 가상 창고 상의 요청이 요청 사이에 간섭 없이, 독립적으로 처리되게 하는 것을 허용한다. 가상 창고를 동적으로 추가

하거나 제거하기 위한 기능과 결합된 이 독립적인 처리는 기존의 사용자에 의해 관찰된 성능에 영향을 미치지 않으면서, 새로운 사용자를 위한 새로운 처리 용량의 추가를 지원한다.

[0086] 도 6은 가상 창고 관리자(502) 하에 다수의 가상 창고와 통신하는 큐(602)를 갖는 예시적인 동작 환경(600)을 도시하는 블록도이다. 환경(600)에서, 큐(602)는 다수의 가상 창고(606a, 606b 및 606n)를 통해 다수의 데이터베이스 공유된 저장 디바이스(608a, 608b, 608c, 608d, 608e 및 608n)에 액세스할 수 있다. 도 6에 도시되진 않았지만, 큐(602)는 자원 관리자(302)를 통해 가상 창고에 액세스할 수 있다. 특정 실시예에서, 데이터베이스(608a-608n)는 공유된 데이터베이스 저장소(308)에 포함되며 실행 플랫폼(212)에서 구현된 임의의 가상 창고에 의해 액세스 가능하다. 일부 실시예에서, 큐(602)는 인터넷과 같은 데이터 통신 네트워크를 사용하여 가상 창고(606a-606n) 중 하나에 액세스할 수 있다. 일부 구현에서, 클라이언트 계정은 (완료될 내부 작업을 저장하도록 구성된) 큐(602)가 특정 시간에 특정 가상 창고(606a-606n)와 상호작용해야 함을 지정할 수 있다.

[0087] (도시된 바와 같이) 일 실시예에서, 각 가상 창고(606a-606n)는 모든 데이터베이스(608a-608n)와 통신할 수 있다. 일부 실시예에서, 각 가상 창고(606a-606n)는 모든 데이터베이스(608a-608n)의 서브세트와 통신하도록 구성된다. 이러한 배열에서, 데이터의 세트와 연관된 개별적인 클라이언트 계정은 단일 가상 창고를 통해 및/또는 데이터베이스(608a-608n)의 특정 서브세트에 모든 데이터 검색 및 데이터 저장 요청을 전송할 수 있다. 또한, 특정 가상 창고(606a-606n)가 데이터베이스(608a-608n)의 특정 서브세트와 통신하도록 구성되는 경우, 구성은 동적이다. 예를 들어, 가상 창고(606a)는 데이터베이스(608a-608n)의 제1 서브세트와 통신하도록 구성될 수 있으며, 나중에 데이터베이스(608a-608n)의 제2 서브세트와 통신하도록 재구성될 수 있다.

[0088] 일 실시예에서, 큐(602)는 데이터 검색, 데이터 저장 및 데이터 처리 요청을 가상 창고 관리자(604)로 전송하며, 이는 요청을 적절한 가상 창고(606a-606n)로 라우팅한다. 일부 구현에서, 가상 창고 관리자(604)는 가상 창고(606a-606n)에 작업의 동적 할당을 제공한다.

[0089] 일부 실시예에서, 내결함성 시스템은 가상 창고의 장애에 응답하여 새로운 가상 창고를 생성한다. 새로운 가상 창고는 동일한 가상 창고 그룹에 있거나, 또는 상이한 지리적 위치에서 상이한 가상 창고 그룹에 생성될 수 있다.

[0090] 본원에 서술된 시스템 및 방법은 데이터가 컴퓨팅(또는 처리) 자원과 별개인 서비스로서 저장되고 액세스되는 것을 허용한다. 실행 플랫폼(212)으로부터 컴퓨팅 자원이 할당되지 않은 경우에도, 데이터는 원격 데이터 소스로부터 데이터의 재로딩을 요구하지 않으면서 가상 창고에 대해 이용 가능하다. 따라서, 데이터는 데이터와 연관된 컴퓨팅 자원의 할당과 독립적으로 이용 가능하다. 서술된 시스템 및 방법은 임의의 타입의 데이터에 유용하다. 특정 실시예에서, 데이터는 구조화되고 최적화된 포맷으로 저장된다. 컴퓨팅 서비스로부터 데이터 저장소/액세스 서비스를 결합 해제하는 것은 상이한 사용자 및 그룹간의 데이터의 공유를 단순화한다. 본원에서 논의된 바와 같이, 각 가상 창고는 심지어 다른 가상 창고가 동일한 데이터에 액세스하는 것과 동시에, 액세스 권한을 갖는 임의의 데이터에 액세스할 수 있다. 이 아키텍처는 로컬 캐시에 저장된 임의의 실제 데이터 없이 질의를 구동시키는 것을 지원한다. 본원에 서술된 시스템 및 방법은 명료한 동적 데이터 이동을 할 수 있으며, 이는 필요한 경우 시스템의 사용자에 대해 명료한 방식으로 원격 저장 디바이스로부터 로컬 캐시로 데이터를 이동시킨다. 또한, 이 아키텍처는 임의의 가상 창고가 컴퓨팅 서비스로부터 데이터 저장 서비스의 결합 해제에 기인하여 임의의 데이터에 액세스할 수 있기 때문에, 사전 데이터 이동 없이 데이터 공유를 지원한다.

[0091] 도 7은 자동화된 데이터 수집(700)을 위한 시스템의 블록도 아키텍처 모델이다. 시스템(700)은 예를 들어 클라이언트 계정 A 및 클라이언트 계정 B를 포함하는 복수의 클라이언트 계정을 포함한다. 클라이언트 계정은 사용자 파일을 포함하는 하나 이상의 데이터 버킷 또는 데이터 레이크를 포함할 수 있다. 클라이언트 계정의 각각은 데이터베이스에 수집될 모든 사용자 파일의 목록을 포함하는 클라이언트 계정 큐에 결합된다. 도 7에 도시된 바와 같이, 클라이언트 계정 A는 계정 A 큐(704)에 결합되고, 클라이언트 계정 B는 계정 B 큐(708)에 결합된다. 대안적인 실시예에서, 복수의 클라이언트 계정은 하나 이상의 클라이언트 계정 큐에 공급될 수 있다. 수집 폴러(ingest poller, 706, 710)는 클라이언트 계정의 각각과 연관된다. 일 실시예에서, 수집 폴러(706, 710)는 하나 이상의 클라이언트 계정에 대한 큐를 폴링할 수 있다. 도 7에 도시된 바와 같이, 수집 폴러(706)는 계정 A 큐(704)를 폴링하고 수집 폴러(710)는 계정 B 큐(708)를 폴링하는 것을 담당한다. 각 수집 폴러는 다수의 클라이언트 계정 큐에서 폴링할 수 있다. 수집 폴러(706, 710)는 클라이언트 계정 큐로부터 수신된 통지를 검사하고, 모든 이용 가능한 파이프(712, 714, 716, 718, 720, 722, 724 및 726)에 대해 통지를 매칭시킬 수 있다. 수집 폴러(706, 710)는 연관된 사용자 파일이 클라이언트 계정으로부터 검색되고 적절한 파이프로 전달될 수 있도록 매칭 파이프(712-726)의 각각에 통지를 전달하는 것을 담당한다. 클라이언트 계정 큐 및 연관된 수집

폴러는 통지 채널(702) 내에 구현된다.

[0092] 일 실시예에서, 수집 폴러(706, 710)는 폴링 또는 폴링된 동작을 수행한다. 일 실시예에서, 수집 폴러(706, 710)는 자원 관리자(302) 내에 존재하는 기능의 일부이다. 수집 폴러(706, 710)는 클라이언트 계정 큐(704, 708)로부터의 각 통지를 검사할 수 있고, 각 메시지에 대해 수집 폴러(706, 710)는 각 파이프(712-726)에 대한 통지에 매칭할 것이다. 수집 폴러(706, 710)는 매칭 파이프(712-726) 각각에 통지를 전달할 것이다.

[0093] 일 실시예에서, 시스템(700)은 Amazon Web Services™ S3 버킷의 일부로서 Simple Queue Service™(SQS) 큐의 풀을 포함하는 복수의 클라이언트 계정 큐를 포함한다. SQS 큐의 풀은 버킷에 사용자 파일을 추가하기 위해 클라이언트 계정에 제공될 수 있다. 하나 이상의 사용자 파일이 클라이언트 계정 데이터 버킷에 추가될 때 통지가 자동으로 생성될 수 있다. 복수의 고객 데이터 버킷이 각 클라이언트 계정에 제공될 수 있다. 클라이언트 계정 큐(704, 708)는 클라이언트 계정에 대한 다수의 데이터 버킷으로부터 데이터 이벤트(즉, 하나 이상의 사용자 파일의 수신)를 처리할 수 있다. 클라이언트 계정은 해당 클라이언트 계정에 대한 하나 이상의 전용 클라이언트 계정 큐(704, 708)를 갖는 복수의 데이터 버킷을 포함할 수 있다. 일 실시예에서, 각 클라이언트 계정 큐(704, 708)는 복수의 파이프(712-726)에 대한 사용자 파일을 수신할 단일 클라이언트 계정에 대한 복수의 데이터 버킷으로부터의 이벤트를 처리한다. 일 실시예에서, 각 수집 폴러(706, 710)는 다수의 클라이언트 계정 큐(704, 708)를 통해 폴링하고, 도 7에 도시된 바와 같이 단일 클라이언트 계정 큐(704, 708)에 전용되지 않는다.

[0094] 일 실시예에서, 파이프(712-726)의 생성은 외부 스테이지 또는 내부 스테이지 중 하나에서 발생한다. 두 개의 파이프가 동일한 매칭 조건으로 생성되면, 적용 가능한 사용자 파일은 파이프 양자로 로딩될 것이다. 일 실시예에서, 클라이언트가 파이프(712-726)를 드롭할 때, 동일한 클라이언트 데이터 버킷으로부터 자동-수집되는 다른 파이프가 있는 경우, 출력은 없을 것이다. 드롭되는 파이프(712-726)가 클라이언트 데이터 버킷에 대해 자동-수집되는 마지막 파이프인 경우, 큐 구성은 버킷 통지 구성으로부터 제거될 것이다.

[0095] 일 실시예에서, 클라이언트 계정 큐가 클라이언트 데이터 버킷으로부터의 메시지를 수락하기 위한 정책으로 구성될 필요가 있도록, 클라이언트 계정 큐에 대한 정책 제약이 존재한다. 일 실시예에서, 클라이언트 데이터 버킷이 클라이언트 계정 큐에 생성 이벤트를 전송할 수 있도록, 다음: 클라이언트 데이터 버킷이 원칙(principal)으로부터 허용되어야 함, 또는 클라이언트 데이터 버킷 소스가 그 조건에서 허용되어야 함 중 하나는 참이어야 한다. 클라이언트 데이터 버킷이 원칙으로부터 허용되면, 클라이언트 계정으로부터의 모든 엔티티는 클라이언트 계정 큐 상에 메세지 전송(send message)을 할 수 있을 것이다. 따라서, 클라이언트 데이터 버킷만이 알림을 전송하도록 제한하기 위해, 조건은 클라이언트 데이터 버킷 이름을 가리키도록 설정될 수 있다.

[0096] 일 실시예에서, 단일 클라이언트 계정에 대한 다수의 클라이언트 계정 큐가 존재한다. 이는 모든 가능한 클라이언트 계정에 대한 단일 클라이언트 계정 큐를 갖는 것이 단일 클라이언트 계정 큐가 다수의 클라이언트 데이터 버킷에 걸쳐 공유될 것을 요구하고 단일 클라이언트 계정 큐가 상당히 많은 복수의 클라이언트 데이터 버킷을 처리할 수 없는 실시예에서 이로울 수 있다.

[0097] 일 실시예에서, 시스템(700)은 각 클라이언트 계정에 대한 클라이언트 계정 큐의 풀을 유지한다. 각 클라이언트 계정 큐는 다수의 스테이지 및 파이프(712-726)를 서비스할 수 있다. 자원 관리자(302)의 각 인스턴스는 하나 이상의 클라이언트 계정 큐에 대한 토큰 링(token ring)을 구축하고, 적용 가능한 클라이언트 계정 큐에 할당된 해당 범위의 토큰을 관찰하며, 그 클라이언트 계정 큐를 더 폴링할 것이다.

[0098] 일 실시예에서, 파이프(712-726)가 자동으로 수집하도록 구성될 때마다, 수집 요청은 사용자 파일의 제공을 현재 처리하는 자원 관리자(302)의 인스턴스로 재지향된다. 자원 관리자(302)의 올바른 인스턴스가 수집 요청을 수신하면, 이는 그것이 캐시로부터 갖는 모든 클라이언트 계정 큐를 판독할 수 있다. 자원 관리자(302)의 인스턴스는 클라이언트 계정 큐 중 하나가 적용 가능한 클라이언트 데이터 버킷에 대한 기존의 수집 정책을 이미 가지고 있는지를 더 확인할 수 있다. 그러한 경우, 메타데이터 저장소에서 해당 클라이언트 계정 큐에 대한 참조를 증가시키고 해당 클라이언트 계정 큐를 반환할 수 있다. 자원 관리자(302)의 인스턴스는 클라이언트 계정 큐가 기존의 정책을 갖지 않는지를 더 결정하고, 기존의 클라이언트 계정 큐를 선택하거나 또는 새로운 클라이언트 계정 큐를 생성하기 위해 클라이언트 계정 큐 선택 정책을 구현할 수 있다. 이는 특정 상황 하에 해당 클라이언트 계정을 위한 새로운 클라이언트 계정 큐를 제공하기 위해 정책을 클라이언트 계정 큐에 추가하는 것을 포함할 수 있다. 이는 메타데이터 저장소에서 클라이언트 계정 큐에 대한 참조를 더 증가시키고 해당 클라이언트 계정 큐를 반환할 수 있다.

[0099] 일 실시예에서, 클라이언트 계정 큐 선택 정책은 클라이언트 계정별로 클라이언트 계정 큐를 그룹화하는 것을

포함한다. 일 실시예에서, 이는 단 하나의 클라이언트 계정 큐를 사용하는 것을 포함하지만, 클라이언트 계정에 대해 다수의 클라이언트 계정 큐가 있는 경우, 클라이언트 계정 큐 선택 정책은 최소 개수의 클라이언트 데이터 버킷을 갖는 하나의 클라이언트 계정을 선택하는 것을 포함한다. 클라이언트 계정 큐 선택 정책은 토큰 범위 기반 그룹화에 의해 클라이언트 계정 큐를 선택하는 것을 더 포함할 수 있다. 이는 다수의 트래픽을 관찰하는(seeing) 클라이언트 계정 큐를 식별하는 토큰 범위에 대한 메트릭(metrics)을 사용하는 것을 포함할 수 있으며, 최소량의 트래픽을 관찰하는 하나의 클라이언트 계정 큐를 선택할 수 있다. 이는 상이한 클라이언트 계정 큐에 있는 클라이언트 데이터 버킷의 개수를 결정하는 것 및 최소 개수의 클라이언트 데이터 버킷을 갖는 하나의 클라이언트 계정 큐를 선택하는 것을 더 포함할 수 있다.

[0100] 일 실시예에서, 각 오브젝션 생성 통지 메시지(objection creation notification message)는 클라이언트 데이터 버킷에서 객체(즉, 사용자 파일)의 완전한 경로를 제공하는 클라이언트 데이터 버킷 이름 및 전체 객체 이름을 포함한다. 단일 클라이언트 계정 큐는 매칭하는 규칙의 세트를 갖는 각 파이프로 여러 파이프에 공급될 수 있다.

[0101] 클라이언트 계정 큐는 마이크로 서비스, 분산 시스템 및 서버리스 애플리케이션을 결합 해제하고 확장하는 메시지 큐잉 서비스이다. 일 실시예에서, 클라이언트 계정 큐(704, 708)는 Amazon Web Services®에 의해 제공되는 Amazon® Simple Queue Service(SQS)이다. SQS는 각각 확장성 및 안정성을 개선하기 위해 개별적인 기능을 수행하는 각각의 구성요소로부터 애플리케이션을 구축하는 것을 가능하게 한다. SQS는 클라우드 애플리케이션의 구성요소를 결합 해제하고 조정함으로써 비용 효율성을 개선할 수 있다. SQS를 통해, 클라이언트는 메시지를 손실하거나 또는 다른 서비스가 항상 이용 가능하도록 요구하지 않으면서, 모든 볼륨(volume)에서 소프트웨어 구성요소 사이에서 메세지를 전송하고 저장하며 수신할 수 있다.

[0102] 일 실시예에서, SQS는 표준 큐 및 FIFO 큐를 포함하는 두 개의 타입의 메시지 큐를 제공할 수 있다. 표준 큐는 메세지의 최대 처리량, 최선의 순서화(best-effort ordering) 및 최소 한 번의 전달(at-least-once-delivery)을 제공한다. FIFO(선입선출, First In First Out) 큐는 메세지가 제한된 처리량으로 이들이 전송된 정확한 순서로 정확하게 한번 처리됨을 보장하도록 설계된다.

[0103] 도 8은 데이터베이스로의 배치 데이터 수집을 위한 방법(800)의 개략적인 흐름도이다. 방법(800)은 본원에 개시된 바와 같이 자원 관리자(302)와 같은 임의의 적합한 컴퓨팅 디바이스에 의해 수행될 수 있다. 방법(800)이 개시되고, 802에서 자원 관리자는 클라이언트 계정으로부터 수신되고 데이터베이스로 수집될 사용자 파일의 존재를 나타내는 통지를 결정한다. 방법(800)이 계속되고, 804에서 자원 관리자는 사용자 파일에서 데이터를 식별하고, 806에서 사용자 파일에서 데이터를 수신하기 위해 데이터베이스의 목표 테이블을 식별한다. 방법(800)이 계속되고, 808에서 자원 관리자는 데이터 및 목표 테이블을 나타내는 수집 작업을 생성한다. 방법(800)이 계속되고, 810에서 자원 관리자는 수집 작업을 실행 플랫폼의 실행 노드에 할당하며, 실행 플랫폼은 데이터베이스 데이터를 집합적으로 저장하는 복수의 공유된 저장 디바이스와 독립적으로 동작하는 복수의 실행 노드를 포함한다. 방법(800)이 계속되고, 812에서 자원 관리자는 데이터가 실행 노드에 의해 목표 테이블에 완전히 커밋된 이후에, 목표 테이블에 대한 메타데이터를 메타데이터 저장소에 등록한다.

[0104] 방법(800)은 사용자 개입 없이 자율적으로 실행될 수 있다. 방법(800)은 다수의 수집 작업이 실질적으로 병렬로 실행되도록, 다수의 수집 작업을 실행 플랫폼에 걸쳐 비동기식으로 실행하는 것을 포함할 수 있다.

[0105] 도 9는 예시적인 컴퓨팅 디바이스(900)를 도시하는 블록도이다. 일부 실시예에서, 컴퓨팅 디바이스(900)는 본원에서 논의된 시스템 및 구성요소 중 하나 이상을 구현하는데 사용된다. 예를 들어, 컴퓨팅 디바이스(900)는 사용자 또는 관리자가 자원 관리자(902)에 액세스하는 것을 허용할 수 있다. 또한, 컴퓨팅 디바이스(900)는 본원에서 서술된 시스템 및 구성요소 중 임의의 것과 상호작용할 수 있다. 따라서, 컴퓨팅 디바이스(900)는 본원에서 논의된 것과 같은 다양한 절차 및 작업을 수행하는데 사용될 수 있다. 컴퓨팅 디바이스(900)는 서버, 클라이언트 또는 임의의 다른 컴퓨팅 엔티티로서 기능을 할 수 있다. 컴퓨팅 디바이스(900)는 데스크탑 컴퓨터, 노트북 컴퓨터, 서버 컴퓨터, 휴대형 컴퓨터(handheld computer), 태블릿 등과 같은 매우 다양한 컴퓨팅 디바이스 중 임의의 것일 수 있다.

[0106] 컴퓨팅 디바이스(900)는 하나 이상의 프로세서(들)(902), 하나 이상의 메모리 디바이스(들)(904), 하나 이상의 인터페이스(들)(906), 하나 이상의 대용량 저장 디바이스(들)(908) 및 하나 이상 입/출력(I/O) 디바이스(들)(910)를 포함하며, 이들 모두는 버스(912)에 결합된다. 프로세서(들)(902)는 메모리 디바이스(들)(904) 및/또는 대용량 저장 디바이스(들)(908)에 저장된 명령어를 실행하는 하나 이상의 프로세서 또는 제어기를 포함한다. 프로세서(들)(902)는 또한 캐시 메모리와 같은 다양한 타입의 컴퓨터 판독가능 매체를 포함할 수 있다.

[0107] 메모리 디바이스(들)(904)는 휘발성 메모리(예를 들어, 랜덤 액세스 메모리(RAM)) 및/또는 비휘발성 메모리(예를 들어, 판독-전용 메모리(ROM))와 같은 다양한 컴퓨터 판독가능 매체를 포함한다. 메모리 디바이스(들)(904)는 또한 플래시 메모리와 같은 재기록 가능한 ROM(rewritable ROM)을 포함할 수 있다.

[0108] 대용량 저장 디바이스(들)(908)는 자기 테이프, 자기 디스크, 광학 디스크, 고체 상태 메모리(예를 들어, 플래시 메모리) 등과 같은 다양한 컴퓨터 판독가능 매체를 포함한다. 다양한 컴퓨터 판독가능 매체로부터의 판독 및/또는 그로의 기록을 가능하게 하기 위해, 다양한 드라이브가 대용량 저장 디바이스(들)(908)에 또한 포함될 수 있다. 대용량 저장 디바이스(들)(908)는 이동식 매체(removable media) 및/또는 비-이동식 매체를 포함한다.

[0109] I/O 디바이스(들)(910)는 데이터 및/또는 다른 정보가 컴퓨팅 디바이스(900)에 입력되거나 그로부터 검색되는 것을 허용하는 다양한 디바이스를 포함한다. 예시적인 I/O 디바이스(들)(910)는 커서 제어 디바이스, 키보드, 키패드, 마이크, 모니터 또는 다른 디스플레이 디바이스, 스피커, 프린터, 네트워크 인터페이스 카드, 모뎀, 렌즈, CCD 또는 다른 이미지 캡처 디바이스 등을 포함한다.

[0110] 인터페이스(들)(906)는 컴퓨팅 디바이스(900)가 다른 시스템, 디바이스 또는 컴퓨팅 환경과 상호 작용하는 것을 허용하는 다양한 인터페이스를 포함한다. 예시적인 인터페이스(들)(906)는 근거리 통신망(LAN), 광역 통신망(WAN), 무선 네트워크 및 인터넷에 대한 인터페이스과 같은 임의의 수의 상이한 네트워크 인터페이스를 포함한다.

[0111] 버스(912)는 프로세서(들)(902), 메모리 디바이스(들)(904), 인터페이스(들)(906), 대용량 저장 디바이스(들)(908) 및 I/O 디바이스(들)(910)뿐만 아니라 버스(912)에 결합된 다른 디바이스 또는 구성요소와 통신하는 것을 허용한다. 버스(912)는 시스템 버스, PCI 버스, IEEE 1394 버스, USB 버스 등과 같은 여러 타입의 버스 구조 중 하나 이상을 나타낸다.

[0112] 예시의 목적을 위해, 프로그램 및 다른 실행가능 프로그램 구성요소는 본원에서 개별적인 블록으로 도시되어 있지만, 이러한 프로그램 및 구성요소는 컴퓨팅 디바이스(900)의 상이한 저장 구성요소에서 다양한 시간에 상주할 수 있고 프로세서(들)(902)에 의해 실행된다는 것이 이해된다. 대안적으로, 본원에서 서술된 시스템 및 절차는 하드웨어로, 또는 하드웨어, 소프트웨어 및/또는 펌웨어의 조합으로 구현될 수 있다. 예를 들어, 하나 이상의 주문형 집적 회로(ASIC)는 본원에 서술된 시스템 및 절차 중 하나 이상을 수행하도록 프로그램될 수 있다. 본원에서 사용된 "모듈"이란 용어는 질의 동작의 전체 또는 일부를 수행하기 위한 목적으로, 하드웨어에 의해, 또는 하드웨어, 소프트웨어 및/또는 펌웨어의 조합에 의해 프로세스를 달성하기 위한 구현 장치를 의미하는 것으로 의도된다.

[0113] 본원에 서술된 시스템 및 방법은 데이터가 컴퓨팅(또는 처리) 자원과 별개인 서비스로서 저장되고 액세스되는 것을 허용한다. 실행 플랫폼으로부터 컴퓨팅 자원이 할당되지 않은 경우에도, 원격 데이터 소스로부터 데이터를 리로딩 할 필요없이 가상 창고에 데이터가 이용 가능하다. 따라서, 데이터는 데이터와 연관된 컴퓨팅 자원의 할당과 독립적으로 이용 가능하다. 서술된 시스템 및 방법은 임의의 타입의 데이터에 유용하다. 특정 실시예에서, 데이터는 구조화되고 최적화된 포맷으로 저장된다. 컴퓨팅 서비스로부터 데이터 저장소/액세스 서비스를 결합 해제하는 것은 상이한 사용자 및 그룹간의 데이터의 공유를 단순화한다. 본원에서 논의된 바와 같이, 각 가상 창고는 심지어 다른 가상 창고가 동일한 데이터에 액세스하는 것과 동시에, 액세스 권한을 갖는 임의의 데이터에 액세스할 수 있다. 이 아키텍처는 로컬 캐시에 저장된 임의의 실제 데이터 없이 질의를 구동시키는 것을 지원한다. 본원에 서술된 시스템 및 방법은 명료한 동적 데이터 이동을 할 수 있으며, 이는 필요한 경우 시스템의 사용자에 대해 명료한 방식으로 원격 저장 디바이스로부터 로컬 캐시로 데이터를 이동시킨다. 또한, 이 아키텍처는 임의의 가상 창고가 컴퓨팅 서비스로부터 데이터 저장 서비스의 결합 해제에 기인하여 임의의 데이터에 액세스할 수 있기 때문에, 이전의 데이터 이동 없이 데이터 공유를 지원한다.

[0114] 본 개시가 특정 바람직한 실시예와 관련하여 서술되었지만, 통상의 기술자에게는 본 개시의 이점을 제공하고 본원에 제시된 이점 및 특징 모두를 제공하지는 않는 실시예를 포함하는 다른 실시예가 명백할 것이며, 이는 또한 본 개시의 범위 내에 있다. 본 개시의 범주를 벗어나지 않으면서 다른 실시예가 이용될 수 있음이 이해될 것이다.

[0115] 예시

[0116] 다음의 예시는 추가적인 실시예에 관련된다.

[0117] 예시 1은 데이터베이스로의 배치 데이터 수집을 위한 시스템이며, 시스템은 클라이언트 계정으로부터 수신되고

데이터베이스로 수집될 사용자 파일의 존재를 나타내는 통지를 결정하는 수단을 포함한다. 시스템은 사용자 파일에서 데이터를 식별하는 수단, 및 사용자 파일에서 데이터를 수신하기 위해 데이터베이스의 목표 테이블을 식별하는 수단을 더 포함한다. 시스템은 데이터 및 목표 테이블을 나타내는 수집 작업을 생성하는 수단을 포함한다. 시스템은 수집 작업을 실행 플랫폼의 실행 노드에 할당하는 수단을 포함하고, 실행 플랫폼은 데이터베이스 데이터를 집합적으로 저장하는 복수의 공유된 저장 디바이스와 독립적으로 동작하는 복수의 실행 노드를 포함한다. 시스템은 데이터가 실행 노드에 의해 목표 테이블에 완전히 커밋된 후에, 메타데이터 저장소에 목표 테이블에 관한 메타데이터를 등록하는 수단을 포함한다.

[0118] 예시 2는 예시 1에서와 같은 시스템이며, 사용자 파일을 클라이언트 계정 큐에 커밋하는 수단을 더 포함하고, 사용자 파일의 존재를 나타내는 통지를 결정하는 수단은 클라이언트 계정 큐가 폴링된 마지막 시간 이후 어느 새로운 사용자 파일이 클라이언트 계정 큐에 커밋되었는지를 결정하기 위해 클라이언트 계정 큐를 폴링함으로써 결정하는 것을 포함한다.

[0119] 예시 3은 예시 1-2 중 어느 것에서와 같은 시스템이며, 사용자 파일의 존재를 나타내는 통지를 결정하는 수단은 사용자 파일이 추가되었음을 나타내는 데이터 레이크로부터 통지를 수신함으로써 결정하도록 구성된다.

[0120] 예시 4는 예시 1-3 중 어느 것에서와 같은 시스템이며, 활성화된 수집 작업의 현재 총 개수 및 활성화된 수집 작업의 원하는 개수를 식별하는 수단; 및 활성화된 수집 작업의 현재 총 개수가: 이동 평균에 의해 시간에 따른 복수의 실행 노드에 대한 요구를 평활화하는 것; 또는 활성화된 수집 작업의 현재 총 개수를 처리하기 위해 활성화된 실행 노드의 필요한 것보다 더욱 많은 개수를 갖는 실행 플랫폼을 유지하는 것 중 하나 이상에 의해 활성화된 수집 작업의 원하는 개수에 동일하거나 또는 이에 접근하도록 실행 플랫폼의 복수의 실행 노드를 관리하는 수단을 더 포함한다.

[0121] 예시 5는 예시 1-4 중 어느 것에서와 같은 시스템이며, 하나 이상의 사용자 파일로부터 어느 데이터가 데이터베이스에 성공적으로 저장되었는지의 표시를 포함하는 수집 이력을 생성하는 수단을 더 포함하며, 수집 이력은 메타데이터 저장소에 저장되고, 수집 이력은 파일 이름, 테이블 식별, 파일 크기, 행 카운트 또는 수집 오류 코드 중 하나 이상을 포함한다.

[0122] 예시 6은 예시 1-5 중 어느 것에서와 같은 시스템이며, 일치하는 해싱을 기초로 자원 관리자의 인스턴스에 사용자 파일을 할당하는 수단을 더 포함하며, 일치하는 해싱의 해시는 목표 테이블의 테이블 식별과 연관되고, 자원 관리자의 인스턴스는 목표 테이블과 연관된 해시에 대한 프로세스를 관리하기 위해 할당된다.

[0123] 예시 7은 예시 1-6 중 어느 것에서와 같은 시스템이며, 자원 관리자의 새로운 인스턴스를 추가하는 수단을 더 포함하고, 자원 관리자의 새로운 인스턴스를 추가하는 것은 복수의 인스턴스의 각 인스턴스가 동일한 또는 거의 동일한 개수의 테이블을 할당 받도록, 일치하는 해싱의 복수의 해시를 나누는 것 및 복수의 해시 각각을 자원 관리자의 복수의 인스턴스에 할당하는 것을 포함한다.

[0124] 예시 8은 예시 1-7 중 어느 것에서와 같은 시스템이며, 사용자 파일을 클라이언트 계정 큐에 커밋하는 수단을 더 포함하고, 수집 작업을 생성하는 수단은 클라이언트 계정 큐에서의 작업량을 기초로 하나 이상의 수집 작업을 생성하도록 구성되며, 클라이언트 계정 큐에서의 작업량은 클라이언트 계정으로부터 최근에 수집된 사용자 파일의 평균 크기에 기초한 사용자 파일의 대략적인 크기; 클라이언트 계정 큐에서의 사용자 파일의 개수; 또는 클라이언트 계정에 의해 나타난 사용자 파일의 크기 중 하나 이상을 기초로 결정된다.

[0125] 예시 9는 예시 1-8 중 어느 것에서와 같은 시스템이며, 데이터는 목표 테이블에 대한 새로운 마이크로-파티션을 생성함으로써 목표 테이블에 커밋되고, 새로운 마이크로-파티션은 데이터가 완전히 성공적으로 삽입된 이후에만 복수의 공유된 저장 디바이스에 저장된다.

[0126] 예시 10은 예시 1-9 중 어느 것에서와 같은 시스템이며, 실행 플랫폼의 실행 노드에 작업을 할당하는 수단은: 실행 플랫폼에 의해 임계 개수의 작업이 이미 처리되고 있을 때, 수집 작업을 실행 노드에 할당하는 것을 지연시키는 것; 새로운 사용자 파일의 임계 개수가 클라이언트 계정 큐에 커밋될 때까지 수집 작업을 실행 노드에 할당하는 것을 지연시킴으로써 최적으로 크기화된 마이크로-파티션을 구축하는 것; 사용자 파일이 클라이언트 계정 큐에 커밋되자마자 수집 작업을 실행 노드에 할당함으로써 레이턴시를 최소화하는 것; 또는 새로운 사용자 파일의 임계 개수가 클라이언트 계정 큐에 커밋될 때까지 수집 작업을 실행 노드에 할당하는 것을 지연시킴으로써, 수집 작업을 개시하고 중지시키는 처리 오버헤드를 최소화하는 것 중 하나 이상에 의해, 실행 플랫폼에 의해 처리되는 작업의 총 개수를 관리하도록 구성된다.

[0127] 예시 11은 예시 1-10 중 어느 것에서와 같은 시스템이며, 실행 플랫폼에 의해 처리되는 작업의 총 개수를 관리

하는 것은 작업의 총 개수가 최소화되면서, 고객 레이턴시가 임계 레벨 미만으로 유지됨을 보장하도록 복수의 레이턴시 인자의 균형을 맞추는 것을 포함한다.

[0128] 예시 12는 데이터베이스로의 배치 데이터 수집을 위한 방법이다. 방법은 클라이언트 계정으로부터 수신되고 데이터베이스로 수집될 사용자 파일의 존재를 나타내는 통지를 결정하는 것을 포함한다. 방법은 사용자 파일에서 데이터를 식별하는 것, 및 사용자 파일에서 데이터를 수신하기 위해 데이터베이스의 목표 테이블을 식별하는 것을 포함한다. 방법은 데이터 및 목표 테이블을 나타내는 수집 작업을 생성하는 것을 포함한다. 방법은 수집 작업을 실행 플랫폼의 실행 노드에 할당하는 것을 포함하고, 실행 플랫폼은 데이터베이스 데이터를 집합적으로 저장하는 복수의 공유된 저장 디바이스에 독립적으로 동작하는 복수의 실행 노드를 포함한다. 방법은 데이터가 실행 노드에 의해 목표 테이블에 완전히 커밋된 후에, 메타데이터 저장소에서 목표 테이블에 관한 메타데이터를 등록하는 것을 포함한다.

[0129] 예시 13은 예시 12에서와 같은 방법이며, 사용자 파일을 클라이언트 계정 큐에 커밋하는 것을 더 포함하고, 사용자 파일의 존재를 나타내는 통지를 결정하는 것은 클라이언트 계정 큐가 폴링된 마지막 시간 이후 어느 새로운 사용자 파일이 클라이언트 계정 큐에 커밋되었는지를 결정하기 위해 클라이언트 계정 큐를 폴링함으로써 결정하는 것을 포함한다.

[0130] 예시 14는 예시 12-13 중 어느 것에서와 같은 방법이며, 사용자 파일의 존재를 나타내는 통지를 결정하는 것은 사용자 파일이 추가되었는지를 나타내는 데이터 레이크로부터 통지를 수신함으로써 결정하도록 구성된다.

[0131] 예시 15는 예시 12-14 중 어느 것에서와 같은 방법이며, 활성화된 수집 작업의 현재 총 개수 및 활성화된 수집 작업의 원하는 개수를 식별하는 것; 및 활성화된 수집 작업의 현재 총 개수가: 이동 평균에 의해 시간에 따른 복수의 실행 노드에 대한 요구를 평활화하는 것; 또는 활성화된 수집 작업의 현재 총 개수를 처리하기 위해 활성화된 실행 노드의 필요한 것보다 더욱 많은 개수를 갖는 실행 플랫폼을 유지하는 것 중 하나 이상에 의해 활성화된 수집 작업의 원하는 개수에 동일하거나 또는 이에 접근하도록 실행 플랫폼의 복수의 실행 노드를 관리하는 것을 더 포함한다.

[0132] 예시 16은 예시 12-15 중 어느 것에서와 같은 방법이며, 하나 이상의 사용자 파일로부터 어느 데이터가 데이터베이스에 성공적으로 저장되었는지의 표시를 포함하는 수집 이력을 생성하는 것을 더 포함하며, 수집 이력은 데이터베이스의 테이블의 마이크로-파티션에 저장되고, 수집 이력은 파일 이름, 테이블 식별, 파일 크기, 행 카운트 또는 수집 오류 코드 중 하나 이상을 포함한다.

[0133] 예시 17은 예시 12-16 중 어느 것에서와 같은 방법이며, 일치하는 해싱을 기초로 자원 관리자의 인스턴스에 사용자 파일을 할당하는 것을 더 포함하며, 일치하는 해싱의 해시는 사용자 파일 및 목표 테이블과 연관되며, 자원 관리자의 인스턴스는 목표 테이블과 연관된 해시에 대한 프로세스를 관리하기 위해 할당된다.

[0134] 예시 18은 예시 12-17 중 어느 것에서와 같은 방법이며, 실행 플랫폼에 새로운 실행 노드를 추가하는 것을 더 포함하고, 새로운 실행 노드를 추가하는 것은 복수의 실행 노드의 각 실행 노드가 일치하는 해싱을 기초로 동일하거나 또는 거의 동일한 개수의 테이블을 할당 받도록, 일치하는 해싱의 복수의 해시를 나누는 것 및 복수의 해시 각각을 복수의 실행 노드에 할당하는 것을 포함한다.

[0135] 예시 19는 예시 12-18 중 어느 것에서와 같은 방법이며, 사용자 파일을 클라이언트 계정 큐에 커밋하는 것을 더 포함하고, 수집 작업을 생성하는 것은 클라이언트 계정 큐에서의 작업량을 기초로 하나 이상의 수집 작업을 생성하는 것을 포함하며, 클라이언트 계정 큐에서의 작업량은 클라이언트 계정으로부터 최근에 수집된 사용자 파일의 평균 크기에 기초한 사용자 파일의 대략적인 크기; 또는 클라이언트 계정에 의해 나타난 사용자 파일의 크기 중 하나 이상을 기초로 결정된다.

[0136] 예시 20은 예시 12-19 중 어느 것에서와 같은 방법이며, 데이터는 목표 테이블에 대한 새로운 마이크로-파티션을 생성함으로써 목표 테이블에 커밋되고, 새로운 마이크로-파티션은 데이터가 완전히 성공적으로 삽입된 이후에만 복수의 공유된 저장 디바이스에 저장된다.

[0137] 예시 21은 예시 12-20 중 어느 것에서와 같은 방법이며, 실행 플랫폼의 실행 노드에 작업을 할당하는 것은: 실행 플랫폼에 의해 임계 개수의 작업이 이미 처리되고 있을 때, 수집 작업을 실행 노드에 할당하는 것을 지연시키는 것; 새로운 사용자 파일의 임계 개수가 클라이언트 계정 큐에 커밋될 때까지 수집 작업을 실행 노드에 할당하는 것을 지연시킴으로써 최적으로 크기화된 마이크로-파티션을 구축하는 것; 사용자 파일이 클라이언트 계정 큐에 커밋되자마자 수집 작업을 실행 노드에 할당함으로써 레이턴시를 최소화하는 것; 또는 새로운 사용자 파일의 임계 개수가 클라이언트 계정 큐에 커밋될 때까지 수집 작업을 실행 노드에 할당하는 것을 지연시킴으로

써, 수집 작업을 개시하고 중지시키는 처리 오버헤드를 최소화하는 것 중 하나 이상에 의해, 실행 플랫폼에 의해 처리되는 작업의 총 개수를 관리하는 것을 더 포함한다.

[0138] 예시 22는 예시 12-21 중 어느 것에서와 같은 방법이며, 실행 플랫폼에 의해 처리되는 작업의 총 개수를 관리하는 것은 작업의 총 개수가 최소화되면서, 고객 레이턴시가 임계 레벨 이상으로 유지됨을 보장하도록 복수의 레이턴시 인자의 균형을 맞추는 것을 포함한다.

[0139] 예시 23은 비일시적 컴퓨터 판독가능 저장매체에 저장된 명령어를 실행하도록 프로그램 가능한 프로세서이다. 명령어는 클라이언트 계정으로부터 수신되고 데이터베이스로 수집될 사용자 파일의 존재를 나타내는 통지를 결정하는 것을 포함한다. 이 명령어는 사용자 파일에서 데이터를 식별하는 것, 및 사용자 파일에서 데이터를 수신하기 위해 데이터베이스의 목표 테이블을 식별하는 것을 포함한다. 명령어는 데이터 및 목표 테이블을 나타내는 수집 작업을 생성하는 것을 포함한다. 명령어는 수집 작업을 실행 플랫폼의 실행 노드에 할당하는 것을 포함하고, 실행 플랫폼은 데이터베이스 데이터를 집합적으로 저장하는 복수의 공유된 저장 디바이스에 독립적으로 동작하는 복수의 실행 노드를 포함한다. 명령어는 데이터가 실행 노드에 의해 목표 테이블에 완전히 커밋된 후에, 메타데이터 저장소에서 목표 테이블에 관한 메타데이터를 등록하는 것을 포함한다.

[0140] 예시 24는 예시 23에서와 같은 프로세서이며, 명령어는 사용자 파일을 클라이언트 계정 큐에 커밋하는 것을 더 포함하고, 사용자 파일의 존재를 나타내는 통지를 결정하는 것은 클라이언트 계정 큐가 폴링된 마지막 시간 이후 어느 새로운 사용자 파일이 클라이언트 계정 큐에 커밋되었는지를 결정하기 위해 클라이언트 계정 큐를 폴링함으로써 결정하는 것을 포함한다.

[0141] 예시 25는 예시 23-24 중 어느 것에서와 같은 프로세서이며, 사용자 파일의 존재를 나타내는 통지를 결정하는 것은 사용자 파일이 추가되었는지를 나타내는 데이터 레이크로부터 통지를 수신함으로써 결정하도록 구성된다.

[0142] 예시 26은 예시 23-25 중 어느 것에서와 같은 프로세서이며, 명령어는: 활성화된 수집 작업의 현재 총 개수 및 활성화된 수집 작업의 원하는 개수를 식별하는 것; 및 활성화된 수집 작업의 현재 총 개수가: 이동 평균에 의해 시간에 따른 복수의 실행 노드에 대한 요구를 평활화하는 것; 또는 활성화된 수집 작업의 현재 총 개수를 처리하기 위해 활성화된 실행 노드의 필요한 것보다 더욱 많은 개수를 갖는 실행 플랫폼을 유지하는 것 중 하나 이상에 의해 활성화된 수집 작업의 원하는 개수에 동일하거나 또는 이에 접근하도록 실행 플랫폼의 복수의 실행 노드를 관리하는 것을 더 포함한다.

[0143] 예시 27은 예시 23-26 중 어느 것에서와 같은 프로세서이며, 명령어는 하나 이상의 사용자 파일로부터 어느 데이터가 데이터베이스에 성공적으로 저장되었는지의 표시를 포함하는 수집 이력을 생성하는 것을 더 포함하며, 수집 이력은 데이터베이스의 테이블의 마이크로-파티션에 저장되고, 수집 이력은 파일 이름, 테이블 식별, 파일 크기, 행 카운트 또는 수집 오류 코드 중 하나 이상을 포함한다.

[0144] 예시 28은 예시 23-27 중 어느 것에서와 같은 프로세서이며, 명령어는 일치하는 해싱을 기초로 자원 관리자의 인스턴스에 사용자 파일을 할당하는 것을 더 포함하며, 일치하는 해싱의 해시는 사용자 파일 및 목표 테이블과 연관되며, 자원 관리자의 인스턴스는 목표 테이블과 연관된 해시에 대한 프로세스를 관리하기 위해 할당된다.

[0145] 예시 29는 예시 23-28 중 어느 것에서와 같은 프로세서이며, 명령어는 실행 플랫폼에 새로운 실행 노드를 추가하는 것을 더 포함하고, 새로운 실행 노드를 추가하는 것은 복수의 실행 노드의 각 실행 노드가 일치하는 해싱을 기초로 동일하거나 또는 거의 동일한 개수의 테이블을 할당 받도록, 일치하는 해싱의 복수의 해시를 나누는 것 및 복수의 해시 각각을 복수의 실행 노드에 할당하는 것을 포함한다.

[0146] 예시 30은 예시 23-29 중 어느 것에서와 같은 프로세서이며, 명령어는 사용자 파일을 클라이언트 계정 큐에 커밋하는 것을 더 포함하고, 수집 작업을 생성하는 것은 클라이언트 계정 큐에서의 작업량을 기초로 하나 이상의 수집 작업을 생성하는 것을 포함하며, 클라이언트 계정 큐에서의 작업량은 클라이언트 계정으로부터 최근에 수집된 사용자 파일의 평균 크기에 기초한 사용자 파일의 대략적인 크기; 또는 클라이언트 계정에 의해 나타난 사용자 파일의 크기 중 하나 이상을 기초로 결정된다.

[0147] 예시 31은 예시 23-30 중 어느 것에서와 같은 프로세서이며, 데이터는 목표 테이블에 대한 새로운 마이크로-파티션을 생성함으로써 목표 테이블에 커밋되고, 새로운 마이크로-파티션은 데이터가 완전히 성공적으로 삽입된 이후에만 복수의 공유된 저장 디바이스에 저장된다.

[0148] 예시 32는 예시 23-31 중 어느 것에서와 같은 프로세서이며, 실행 플랫폼의 실행 노드에 작업을 할당하는 것은: 실행 플랫폼에 의해 임계 개수의 작업이 이미 처리되고 있을 때, 수집 작업을 실행 노드에 할당하는 것을 지연

시키는 것; 새로운 사용자 파일의 임계 개수가 클라이언트 계정 큐에 커밋될 때까지 수집 작업을 실행 노드에 할당하는 것을 지연시킴으로써 최적으로 크기화된 마이크로-파티션을 구축하는 것; 사용자 파일이 클라이언트 계정 큐에 커밋되자마자 수집 작업을 실행 노드에 할당함으로써 레이턴시를 최소화하는 것; 또는 새로운 사용자 파일의 임계 개수가 클라이언트 계정 큐에 커밋될 때까지 수집 작업을 실행 노드에 할당하는 것을 지연시킴으로써, 수집 작업을 개시하고 중지시키는 처리 오버헤드를 최소화하는 것 중 하나 이상에 의해, 실행 플랫폼에 의해 처리되는 작업의 총 개수를 관리하는 것을 더 포함한다.

[0149] 예시 33은 예시 23-32 중 어느 것에서와 같은 프로세서이며, 실행 플랫폼에 의해 처리되는 작업의 총 개수를 관리하는 것은 작업의 총 개수가 최소화되면서, 고객 레이턴시가 임계 레벨 이상으로 유지됨을 보장하도록 복수의 레이턴시 인자의 균형을 맞추는 것을 포함한다.

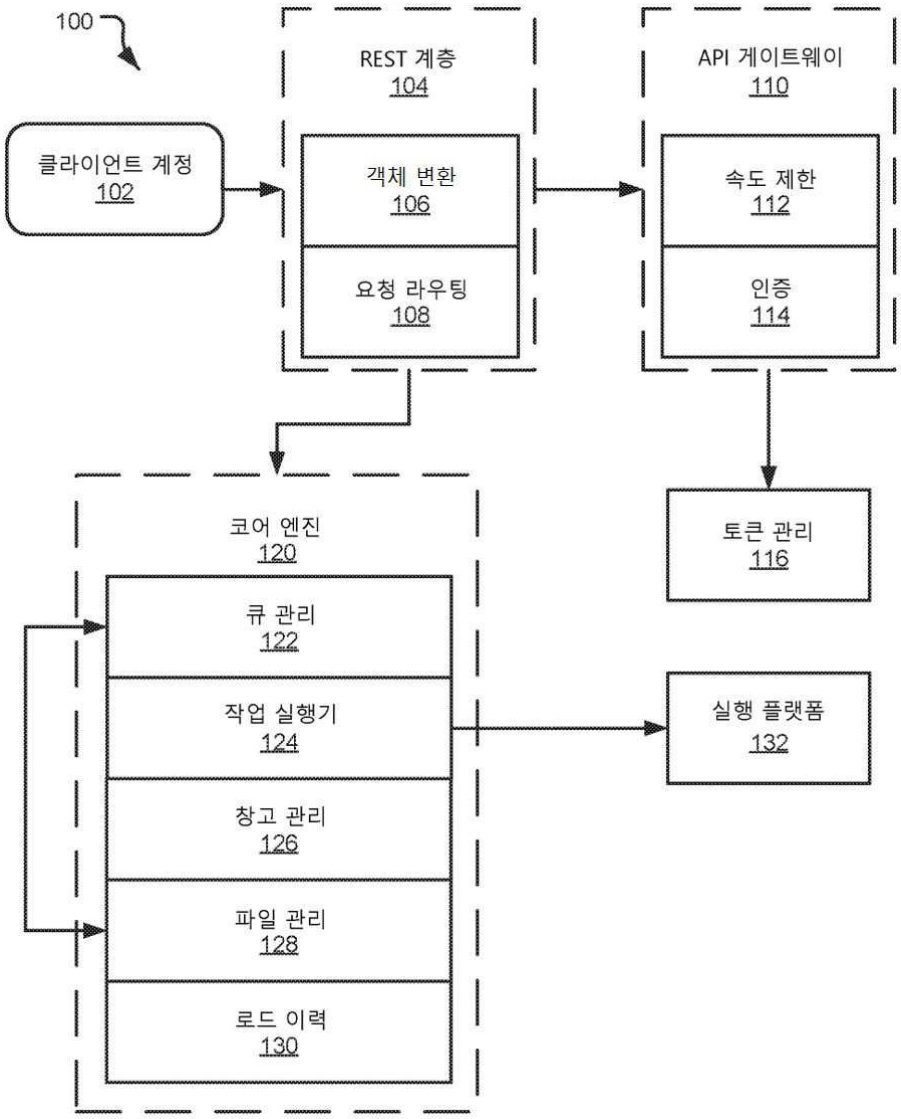
[0150] 본 개시의 일 실시예에서, 시스템은 데이터베이스 또는 테이블에 연속적인 데이터 수집을 제공한다. 이러한 실시예에서, 시스템은 데이터 버킷을 포함하는 클라이언트 계정을 포함한다. 클라이언트 계정은 SQS 큐와 통신할 수 있는 S3 버킷에 연결될 수 있다. SQS 큐는 복수의 파이프와 통신할 수 있다. 시스템은 클라이언트의 데이터 버킷 내에 저장된(housed) 클라이언트 데이터에 변경(alteration)이 이루어졌다는 통지를 수신할 수 있다. 시스템은 클라이언트 데이터 버킷에 대해 이루어진 변경을 수용하기 위해 필요에 따라 컴퓨팅 자원을 확장하고 축소할 수 있다. 시스템은 데이터 변경(change)을 자동으로 풀링하며, 데이터를 클라이언트의 데이터베이스 또는 테이블에 수집할 수 있다. 일 실시예에서, 시스템은 클라이언트 계정으로부터 특정 명령을 수신하지 않고 데이터 변경을 자동으로 수집한다.

[0151] 일 실시예에서, 시스템은 데이터베이스 테이블로 이전에 수집된 데이터를 자동으로 추적한다. 시스템은 데이터 파일이 변경될 때 S3 데이터 버킷으로부터 통지를 수신하고 해당 통지를 이전에 수집된 데이터에 대해 비교할 수 있다.

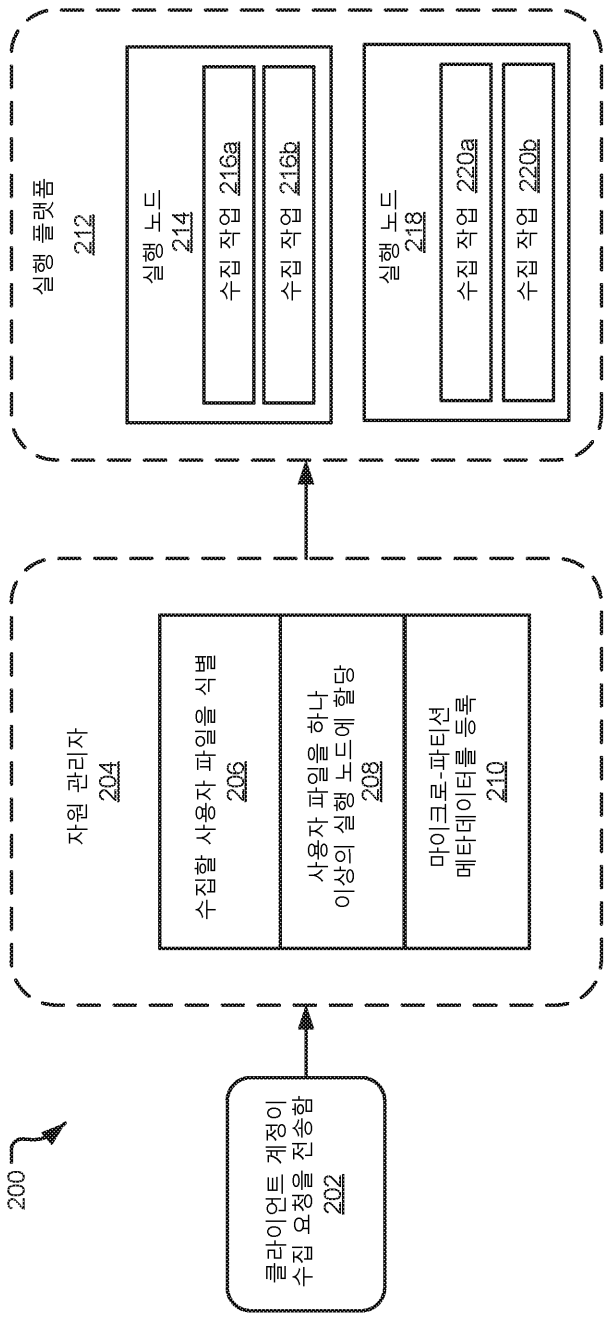
[0152] 일 실시예에서, 시스템은 실행이 실패할 경우 데이터 손실로부터 보호하는 증분 데이터 수집을 제공한다. 일 실시예에서, 데이터는 특정 개수의 네이티브 이진 파일에 도달한 후 데이터가 데이터베이스에 자동으로 커밋되도록 점진적으로 로딩된다. 시스템은 시스템이 정확히 어떤 파일이 어느 위치에서 수집되었는지에 대한 데이터를 저장하도록 메타데이터에서 데이터의 상태를 저장할 수 있다. 실행 실패의 경우, 시스템은 이전의 수집으로부터 중단된 파일 및 위치로부터 재시작할 수 있고, 클라이언트 계정이 중간에 실패한 SQL 명령을 사용한 경우 시스템이 수행 할 데이터를 재수집하는 것을 회피할 수 있으며, 그를 통해 시스템은 데이터를 재수집하는 것을 회피할 수 있다.

도면

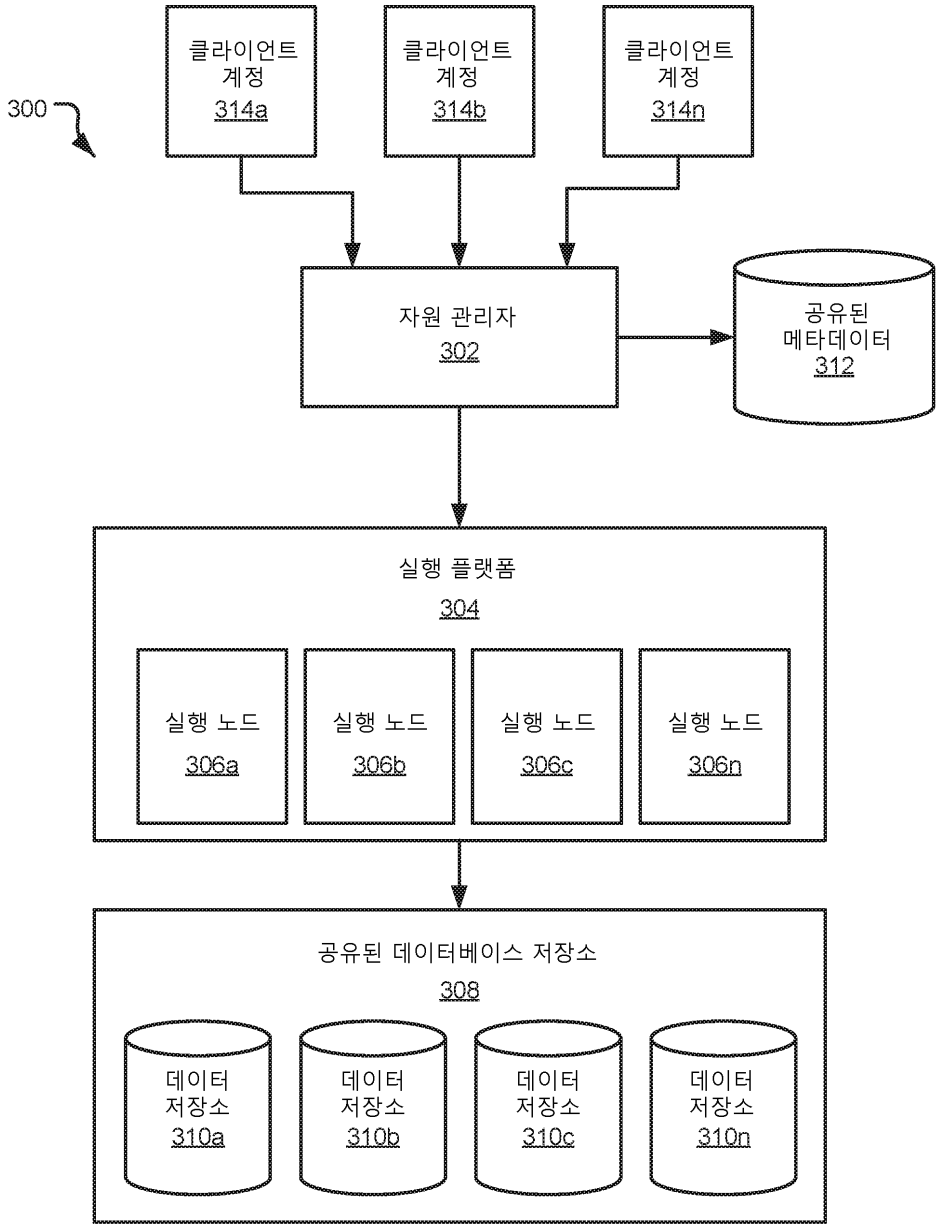
도면1


100
REST 계층
104
API 게이트웨이
110
클라이언트 계정
102
객체 변환
106
요청 라우팅
108
속도 제한
112
인증
114
토큰 관리
116
코어 엔진
120
큐 관리
122
작업 실행기
124
창고 관리
126
파일 관리
128
로드 이력
130
실행 플랫폼
132

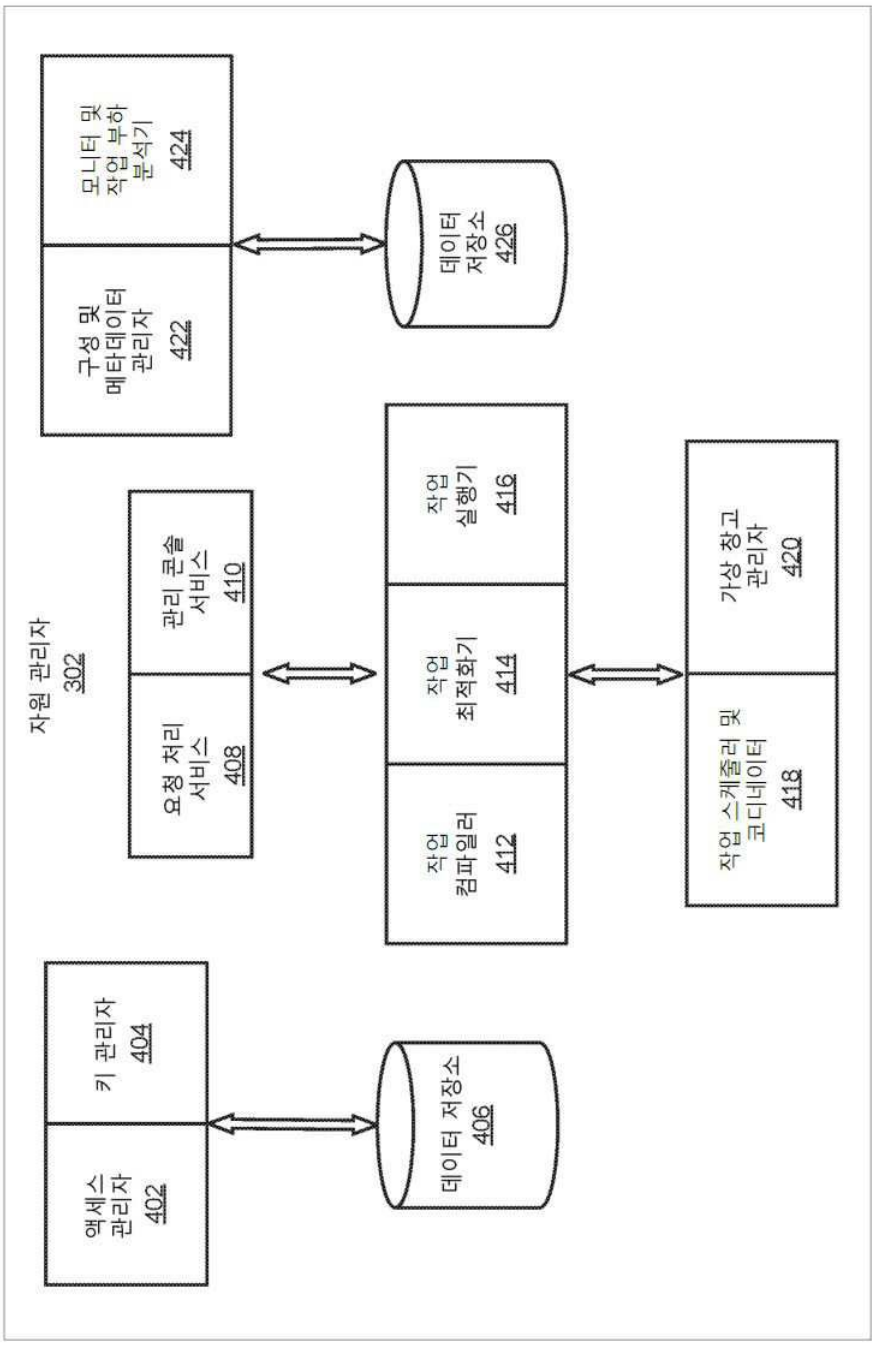
도면2


200
클라이언트 계정이
수집 요청을 전송함
202
자원 관리자
204
수집할 사용자 파일을 식별
206
사용자 파일을 하나
이상의 실행 노드에 할당
208
마이크로-파티션
메타데이터를 등록
210
실행 플랫폼
212
실행 노드
214
수집 작업 216a
수집 작업 216b
실행 노드
218
수집 작업 220a
수집 작업 220b

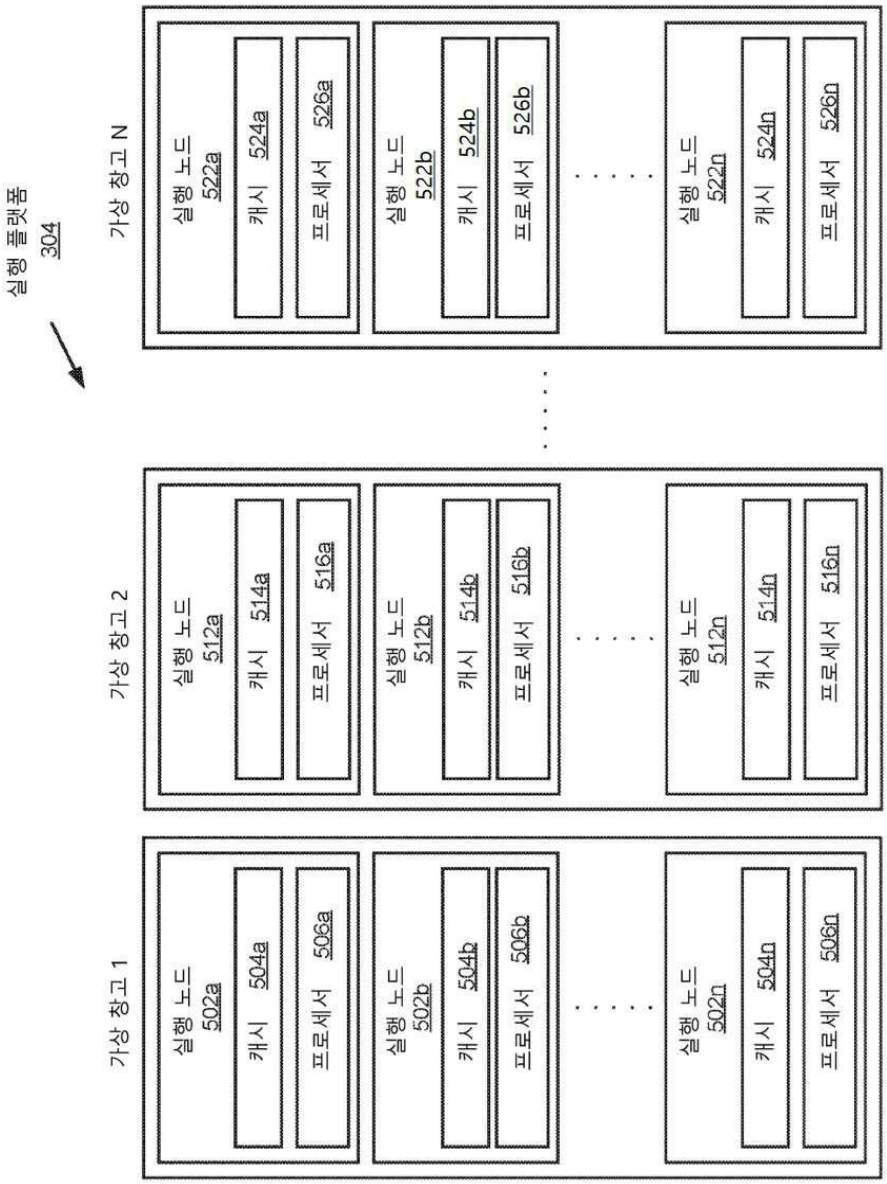
도면3


300
클라이언트 계정 314a
클라이언트 계정 314b
클라이언트 계정 314n
자원 관리자 302
공유된 메타데이터 312
실행 플랫폼 304
실행 노드 306a
실행 노드 306b
실행 노드 306c
실행 노드 306n
공유된 데이터베이스 저장소 308
데이터 저장소 310a
데이터 저장소 310b
데이터 저장소 310c
데이터 저장소 310n

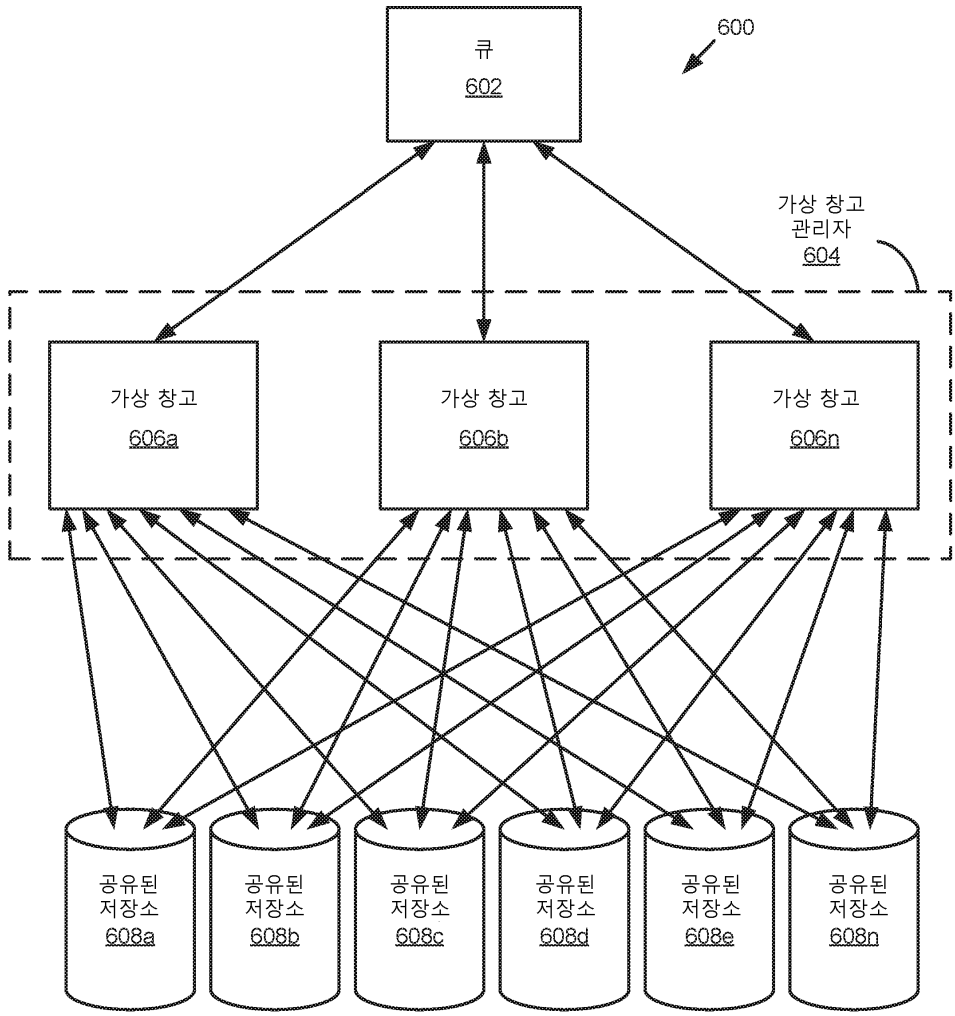
도면4


자원 관리자
302
액세스 관리자
402
키 관리자
404
데이터 저장소
406
요청 처리 서비스
408
관리 콘솔 서비스
410
작업 컴파일러
412
작업 최적화기
414
작업 실행기
416
작업 스케줄러 및 코디네이터
418
가상 창고 관리자
420
구성 및 메타데이터 관리자
422
모니터 및 작업 부하 분석기
424
데이터 저장소
426

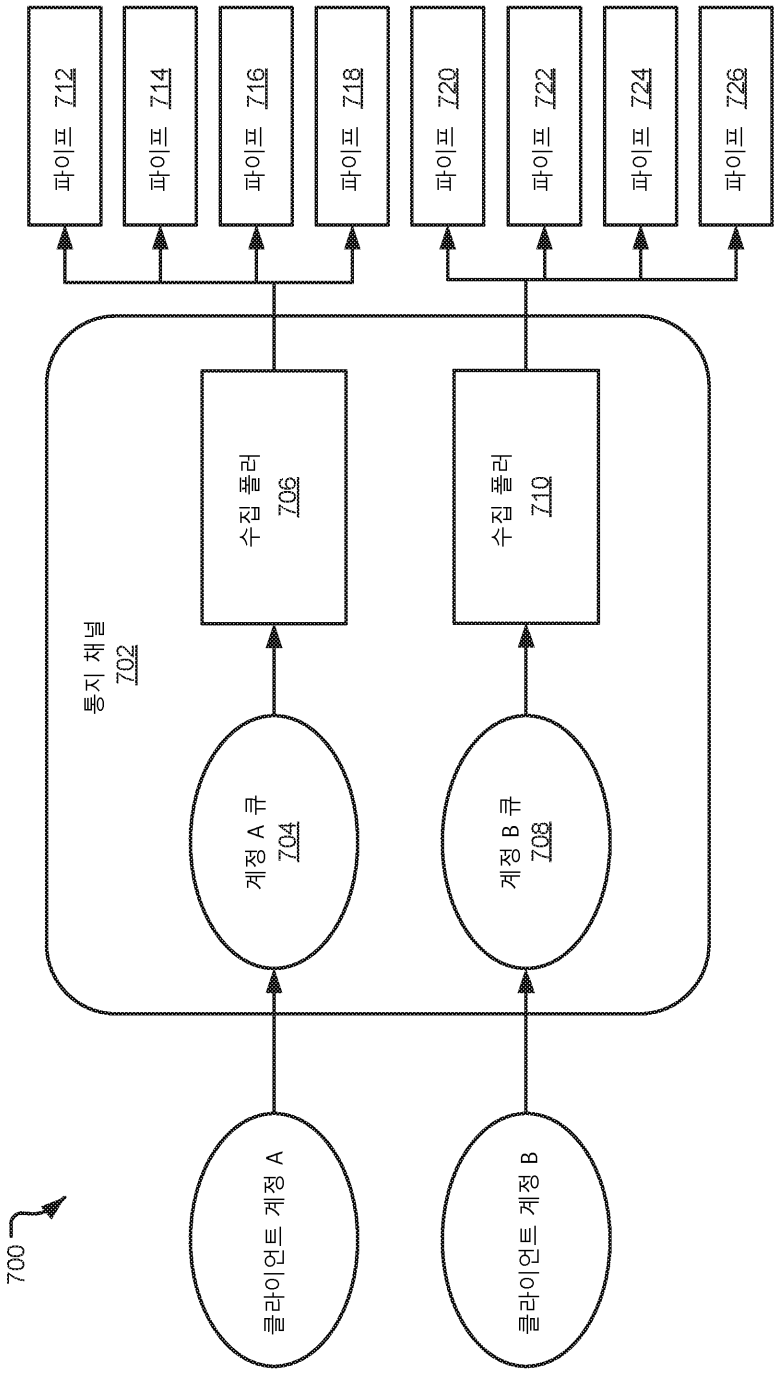
도면5


실행 플랫폼
304
가상 창고 1
가상 창고 2
가상 창고 N
실행 노드
502a
캐시 504a
프로세서 506a
실행 노드
502b
캐시 504b
프로세서 506b
실행 노드
502n
캐시 504n
프로세서 506n
실행 노드
512a
캐시 514a
프로세서 516a
실행 노드
512b
캐시 514b
프로세서 516b
실행 노드
512n
캐시 514n
프로세서 516n
실행 노드
522a
캐시 524a
프로세서 526a
실행 노드
522b
캐시 524b
프로세서 526b
실행 노드
522n
캐시 524n
프로세서 526n

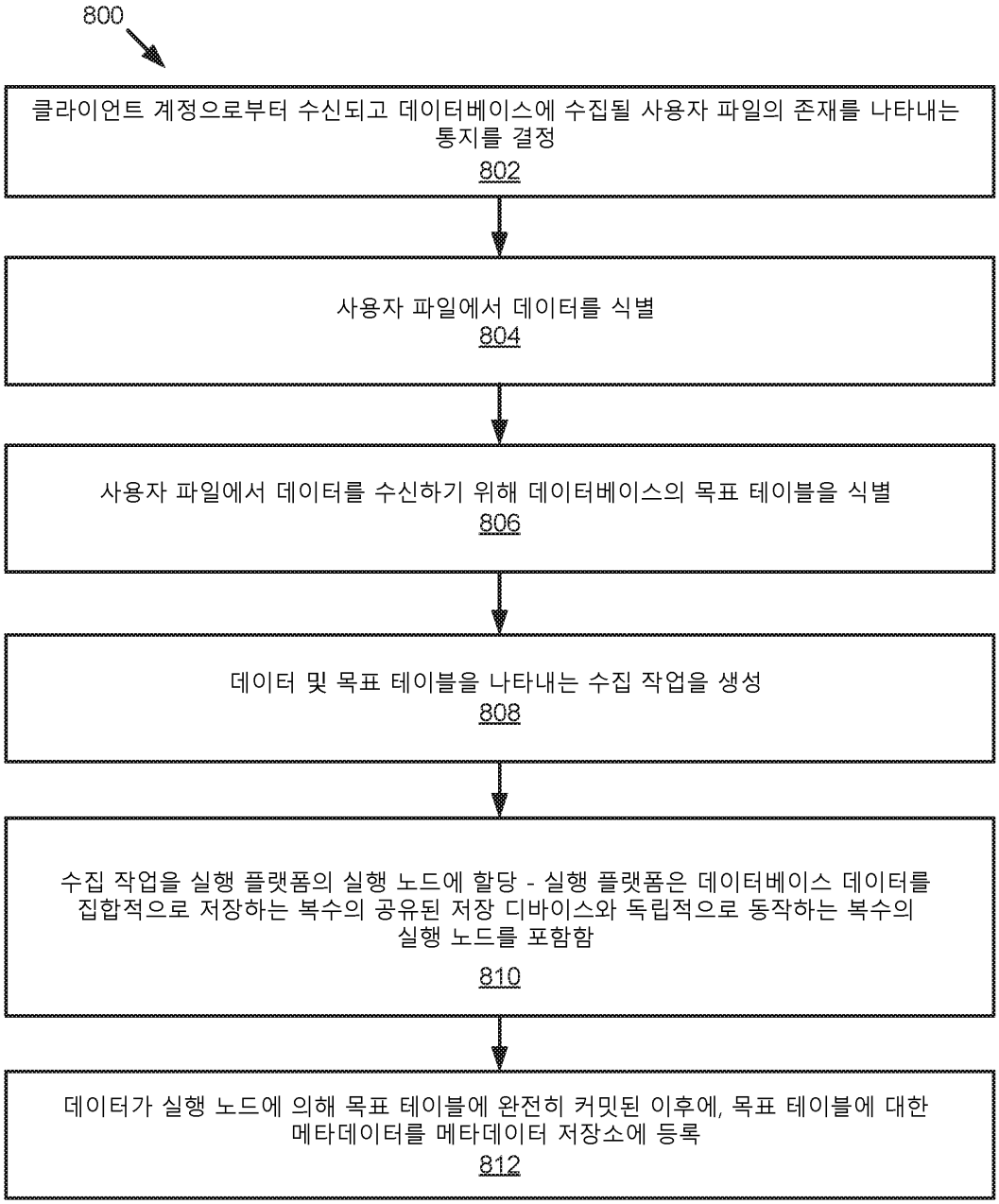
도면6


600
큐
602
가상 창고
관리자
604
가상 창고
606a
가상 창고
606b
가상 창고
606n
공유된
저장소
608a
공유된
저장소
608b
공유된
저장소
608c
공유된
저장소
608d
공유된
저장소
608e
공유된
저장소
608n

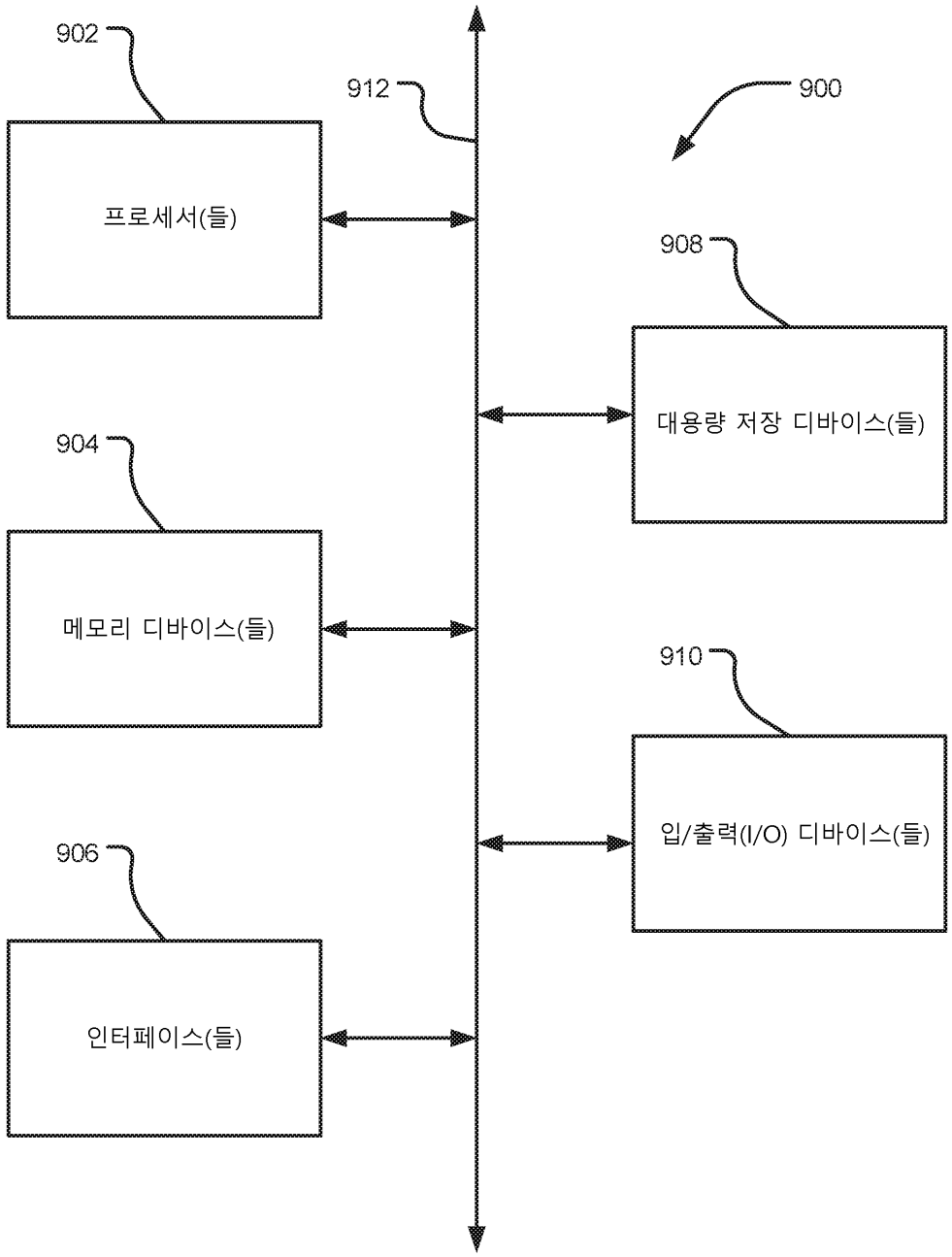
도면7


700
클라이언트 계정 A
클라이언트 계정 B
통지 채널
702
계정 A 큐
704
계정 B 큐
708
수집 폴러
706
수집 폴러
710
파이프 712
파이프 714
파이프 716
파이프 718
파이프 720
파이프 722
파이프 724
파이프 726

도면8


800
클라이언트 계정으로부터 수신되고 데이터베이스에 수집될 사용자 파일의 존재를 나타내는 통지를 결정
802
사용자 파일에서 데이터를 식별
804
사용자 파일에서 데이터를 수신하기 위해 데이터베이스의 목표 테이블을 식별
806
데이터 및 목표 테이블을 나타내는 수집 작업을 생성
808
수집 작업을 실행 플랫폼의 실행 노드에 할당 - 실행 플랫폼은 데이터베이스 데이터를 집합적으로 저장하는 복수의 공유된 저장 디바이스와 독립적으로 동작하는 복수의 실행 노드를 포함함
810
데이터가 실행 노드에 의해 목표 테이블에 완전히 커밋된 이후에, 목표 테이블에 대한 메타데이터를 메타데이터 저장소에 등록
812

도면9


902
912
900
프로세서(들)
908
대용량 저장 디바이스(들)
904
메모리 디바이스(들)
910
입/출력(I/O) 디바이스(들)
906
인터페이스(들)