(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2011년10월04일
(11) 등록번호 10-1069350
(24) 등록일자 2011년09월26일

(51) Int. Cl.
G06F 17/30 (2006.01)
(21) 출원번호 10-2004-0024478
(22) 출원일자 2004년04월09일
심사청구일자 2009년03월05일
(65) 공개번호 10-2004-0095639
(43) 공개일자 2004년11월15일
(30) 우선권주장
10/434,647 2003년05월09일 미국(US)
(56) 선행기술조사문헌
US6047289 A
KR1020010066852 A
KR1020010007111 A
JP08185346 A
전체 청구항 수 : 총 45 항

(73) 특허권자
마이크로소프트 코포레이션
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이
(72) 발명자
신그,람피.
미국98074워싱턴주삼마미시237번애비뉴에스이513
나라야난,차루마시
미국98074워싱턴주삼마미시191번플레이스엔이4609
(74) 대리인
제일특허법인

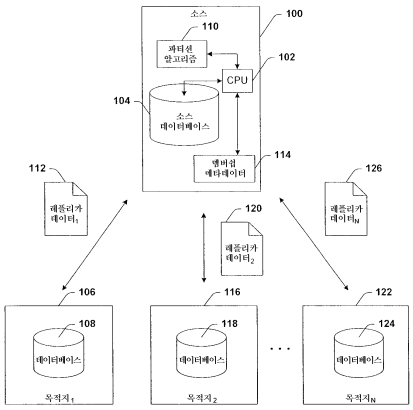
심사관 : 천대식

(54) 클라이언트/서버 환경에서 동기화를 용이하게 하는 방법 및 컴퓨터 판독 가능 기록 매체

(57) 요 약

본 발명은 데이터 소스/데이터 목적지(data destination) 환경에서 데이터 레플리카(data replica)를 사용하여 동기화(synchronization)를 도모하는 방법을 제공한다. 복수의 목적지가 소스에 동기화를 요청하는 경우, 동기화를 위해 선택된 첫 번째 목적지는 첫 번째 목적지 데이터 및 소스 데이터 간의 차이가 결정되고, 파티션 업데이트(partition update)에 지속되는 방식으로 처리된다. 첫 번째 목적지를 업데이트 한 후에, 동기화 프로세스는 지속된 데이터(persisted data)에 의해 결정되고, 그 후에 지속된 데이터에 의해 업데이트된 것처럼 파티션 변화에 의해 영향을 받게 될 잔존하는 목적지에 대해서만 계속된다.

대 표 도 - 도1



특허청구의 범위

청구항 1

복수의 클라이언트들에서 복수의 목적지 레플리카들(replicas)을 사용하여 클라이언트/서버 환경에서 동기화를 용이하게 하는 시스템을 저장하는 컴퓨터 판독 가능 기록 매체에 있어서,

상기 시스템은,

수신자 컴포넌트(receiver component),

결정 컴포넌트(determination component), 및

업데이트 컴포넌트(update component)

를 포함하고,

상기 수신자 컴포넌트는 첫 번째 클라이언트에서의 첫 번째 목적지 레플리카로부터 소스 레플리카로의 데이터의 파티션 업데이트(partition update)에 관련된 정보를 수신하고 - 상기 파티션 업데이트는 상기 첫 번째 목적지 레플리카의 마지막 동기화 이후 변형되었던 상기 첫 번째 레플리카의 파티션의 모든 행들(rows)을 나타내고, 상기 수신자 컴포넌트는 상기 파티션 업데이트의 모든 행들에 세대 식별자(generation identifier)를 할당하고, 상기 세대 식별자의 값은 상기 파티션 업데이트의 모든 행들에 대하여 동일하며, 파티션은 레플리카 데이터의 행들의 서브세트임 -,

상기 결정 컴포넌트는 상기 파티션 업데이트들을 전파하고, 다른 클라이언트들에 의한 동기화 이전에 상기 다른 클라이언트들의 목적지 레플리카들 중 어느 것이 상기 파티션 업데이트에 의해 영향을 받게 될지를 결정하고 - 상기 결정 컴포넌트는 결정을 위해 서버 테이블들 상의 세트 기반 질의(a set based query)를 이용하고, 상기 서버 테이블들은 파티션 메타데이터 테이블, 현재 변화 메타데이터 테이블, 및 과거 변화 메타데이터 테이블을 포함하고, 상기 파티션 메타데이터 테이블은 각각의 목적지 레플리카에 관련된 각각의 파티션을 식별하고, 상기 현재 변화 메타데이터 테이블은 현재 각각의 파티션의 부분인 각각의 행을 식별하며, 상기 과거 변화 메타데이터 테이블은 과거에 각각의 파티션의 부분이었던 각각의 행을 식별함 -,

상기 업데이트 컴포넌트는 상기 클라이언트들에 의한 동기화 시에, 상기 복수의 클라이언트들에 걸쳐 파티션 일관성을 도모하기 위하여, 상기 파티션 변화에 의해 영향을 받는 것으로 결정된 각각의 클라이언트가 그 클라이언트에게 전파되지 않은 모든 변화들의 세대들로 업데이트되는 - 상기 업데이트 컴포넌트는 각각의 파티션에 관련된 각각의 세대 식별자를 식별하는 세대 파티션 매핑 테이블 및 각각의 클라이언트에게 전파되지 않은 변화들을 식별하기 위하여 각각의 클라이언트에게 전파된 각각의 세대를 식별하는 세대 메타데이터 테이블을 이용함 -,

컴퓨터 판독 가능 기록 매체.

청구항 2

제1항에 있어서, 상기 파티션 업데이트에 관련된 정보는 복제 메타데이터(replication metadata)의 형태로 수신된 컴퓨터 판독 가능 기록 매체.

청구항 3

제1항에 있어서, 상기 서버 테이블들은 각각의 행에 대한 각각의 변화의 시간을 식별하는 행 메타데이터 테이블을 더 포함하는 컴퓨터 판독 가능 기록 매체.

청구항 4

제1항에 있어서, 상기 첫 번째 목적지 레플리카 및 상기 클라이언트의 상기 파티션들은, 업데이트를 받는 행(a row undergoing updating)이 업데이트되도록 정렬되는 컴퓨터 판독 가능 기록 매체.

청구항 5

제1항에 있어서, 클라이언트의 파티션은, 업데이트를 받는 행이 업데이트되고, 필터를 이용하여 확장되어 조인된 테이블들(joined tables)로부터 관련된 행들을 얻도록 상기 첫 번째 레플리카와 정렬되는 컴퓨터 판독 가능

기록 매체.

청구항 6

제5항에 있어서, 상기 행은 조인 필터(join filter)인 상기 필터를 이용하여 확장되고, 확장 계산(expansion computation)은 세트 기반 질의(set-based query)에 따르는 컴퓨터 판독 가능 기록 매체.

청구항 7

제5항에 있어서, 상기 필터는 동적 행 필터(dynamic row filter)인 컴퓨터 판독 가능 기록 매체.

청구항 8

제1항에 있어서, 상기 파티션 업데이트들은 상기 파티션 변화에 의해 영향을 받는 상기 다른 클라이언트들과의 동기화를 위하여 지속되는 멤버쉽 정보(membership information)인 메타데이터의 목적지 레플리카(destination replica)를 포함하는 컴퓨터 판독 가능 기록 매체.

청구항 9

제8항에 있어서, 상기 지속되는 멤버쉽 정보는 변화된 행의 행 식별자와 상기 다른 클라이언트들의 각각의 레플리카의 파티션 식별(partition identity) 사이의 매핑을 포함하는 컴퓨터 판독 가능 기록 매체.

청구항 10

제8항에 있어서, 행 삭제(row deletion) 및 행 업데이트 중의 적어도 하나를 포함하는 상기 지속되는 멤버쉽 정보는 과거 멤버쉽들에 대한 행 정보를 요청하는 컴퓨터 판독 가능 기록 매체.

청구항 11

제1항에 있어서, 상기 파티션 업데이트들은 마지막 동기화 동안 업데이트되지 않았던 상기 다른 클라이언트들만이 업데이트되도록 동기화 앵커 값(synchronization anchor value)에 따라 전파되는 컴퓨터 판독 가능 기록 매체.

청구항 12

제1항에 있어서, 상기 파티션 업데이트들은 상기 첫 번째 목적지 레플리카의 행이 업데이트, 삽입 및 삭제되는 것 중의 적어도 하나인 경우, 파티션 내의 변화를 처리하는 변화 추적 로직(change tracking logic)으로 추적되는 컴퓨터 판독 가능 기록 매체.

청구항 13

제1항에 있어서, 상기 시스템은, 동기화 동안 상기 다른 클라이언트들 내의 상기 파티션 업데이트의 변화를 열거하고, 마지막 동기화 이후에 발생한 상기 다른 클라이언트들 내의 변화들을 처리하는 변화 열거 메커니즘(change enumeration mechanism)을 더 포함하는 컴퓨터 판독 가능 기록 매체.

청구항 14

제13항에 있어서, 삭제 및 파티션 업데이트는 세션의 협의된 동기화 앵커(negotiated synchronization anchor)보다 새로운 동기화 앵커를 갖는 행을 선택함으로써 열거되는 컴퓨터 판독 가능 기록 매체.

청구항 15

제14항에 있어서, 상기 행은 상기 행의 과거 변화들을 반영하는 메타데이터로부터 선택되는 컴퓨터 판독 가능 기록 매체.

청구항 16

제13항에 있어서, 삽입 및 파티션 업데이트는 세션의 협의된 동기화 앵커보다 새로운 동기화 앵커를 갖는 행을 선택함으로써 열거되는 컴퓨터 판독 가능 기록 매체.

청구항 17

제16항에 있어서, 상기 행은 상기 행의 현재 변화들을 반영하는 메타데이터로부터 선택되는 컴퓨터 판독 가능 기록 매체.

청구항 18

제1항에 있어서, 상기 시스템은 상기 첫 번째 목적지 레플리카로부터 전파된 변화들에 할당된 동기화 앵커를 더 포함하고, 상기 동기화 앵커는 세대 식별 정보(generation identification information)의 형태인 컴퓨터 판독 가능 기록 매체.

청구항 19

제18항에 있어서, 첫 번째 파티션의 상기 세대 식별 정보는 상기 각각의 클라이언트가 동기화할 때, 두 번째 파티션의 세대 식별 정보로 덮어쓰기되는(overwritten) 컴퓨터 판독 가능 기록 매체.

청구항 20

제1항의 상기 시스템을 포함하는 서버 기반 시스템.

청구항 21

제1항의 상기 시스템을 포함하는 네트워크.

청구항 22

복수의 클라이언트들에서 복수의 목적지 레플리카들을 사용하여 클라이언트/서버 환경에서 동기화를 용이하게 하는 방법에 있어서,

첫 번째 목적지 레플리카로부터 데이터에서의 파티션 업데이트에 관련된 정보를 수신하는 단계 - 상기 파티션 업데이트는 상기 첫 번째 목적지 레플리카의 마지막 동기화 이후 변형되었던 상기 첫 번째 레플리카의 파티션의 모든 행들(rows)을 나타내며, 파티션은 레플리카 데이터의 행들의 서브세트임 - ;

상기 파티션 업데이트의 모든 행들에 세대 식별자(generation identifier)를 할당하는 단계 - 상기 세대 식별자의 값은 상기 파티션 업데이트의 모든 행들에 대하여 동일함 - ;

다른 클라이언트들의 목적지 레플리카들에 의한 동기화 이전에 상기 다른 클라이언트들 중 어느 것이 상기 파티션 업데이트에 의해 영향을 받게 될지를 결정하는 단계 - 상기 결정 단계는 결정을 위해 서버 테이블들 상의 세트 기반 질의(a set based query)를 이용하고, 상기 서버 테이블들은 파티션 메타데이터 테이블, 현재 변화 메타데이터 테이블, 및 과거 변화 메타데이터 테이블을 포함하고, 상기 파티션 메타데이터 테이블은 각각의 목적지 레플리카에 관련된 각각의 파티션을 식별하고, 상기 현재 변화 메타데이터 테이블은 현재 각각의 파티션의 부분인 각각의 행을 식별하며, 상기 과거 변화 메타데이터 테이블은 과거에 각각의 파티션의 부분이었던 각각의 행을 식별함 - ;

상기 클라이언트들에 의한 동기화 시에, 각각의 파티션에 관련된 각각의 세대 식별자를 식별하는 세대 파티션 매핑 테이블 및 각각의 클라이언트에게 전파되지 않은 변화들의 세대들을 식별하기 위하여 각각의 클라이언트에게 전파된 각각의 세대를 식별하는 세대 메타데이터 테이블을 이용하는 단계; 및

상기 복수의 클라이언트들에 걸쳐 파티션 일관성을 도모하기 위하여, 파티션 변화에 의해 영향을 받는 것으로 결정된 클라이언트들을 각각의 클라이언트에게 전파되지 않은 모든 변화들의 세대들로 업데이트하는 단계

를 포함하는, 동기화를 용이하게 하는 방법.

청구항 23

제22항에 있어서, 상기 파티션 업데이트에 관련된 정보를 복제 메타데이터의 형태로 수신하는, 동기화를 용이하게 하는 방법.

청구항 24

제22항에 있어서, 상기 서버 테이블들은 각각의 행에 대한 각각의 변화의 시간을 식별하는 행 메타데이터 테이

블을 더 포함하는, 동기화를 용이하게 하는 방법.

청구항 25

제22항에 있어서, 업데이트를 받는 행이 업데이트되도록, 상기 첫 번째 목적지 레플리카 및 상기 클라이언트의 파티션들을 정렬하는 단계를 더 포함하는, 동기화를 용이하게 하는 방법.

청구항 26

제22항에 있어서, 업데이트를 받는 행이 업데이트되고, 필터를 이용하여 확장되어 조인된 테이블들로부터 관련된 행들을 얻도록 상기 첫 번째 목적지 레플리카 및 상기 클라이언트의 파티션들을 정렬하는 단계를 더 포함하는, 동기화를 용이하게 하는 방법.

청구항 27

제26항에 있어서, 상기 행은 조인 필터인 상기 필터를 이용하여 확장되고, 확장 계산(expansion computation)은 세트 기반 질의(set-based query)에 따르는, 동기화를 용이하게 하는 방법.

청구항 28

제26항에 있어서, 상기 필터는 동적 행 필터인, 동기화를 용이하게 하는 방법.

청구항 29

제22항에 있어서, 상기 파티션 업데이트들은 상기 파티션 변화에 의해 영향을 받는 상기 다른 클라이언트들과의 동기화를 위하여 지속되는 멤버쉽 정보인 메타데이터의 부분적인 레플리카를 포함하는, 동기화를 용이하게 하는 방법.

청구항 30

제29항에 있어서, 상기 지속되는 멤버쉽 정보는 변화된 행의 행 식별자와 상기 다른 클라이언트들의 각각의 레플리카의 파티션 식별 사이의 매핑을 포함하는, 동기화를 용이하게 하는 방법.

청구항 31

제29항에 있어서, 행 삭제 및 행 업데이트 중의 적어도 하나를 포함하는 상기 지속되는 멤버쉽 정보는 과거 멤버쉽들에 대한 행 정보를 요청하는, 동기화를 용이하게 하는 방법.

청구항 32

제22항에 있어서, 상기 파티션 업데이트들은 상기 마지막 동기화 동안 업데이트되지 않았던 상기 다른 클라이언트들만이 업데이트되도록 동기화 앵커 값에 따라 전파되는, 동기화를 용이하게 하는 방법.

청구항 33

제22항에 있어서, 상기 파티션 업데이트들은 상기 첫 번째 목적지 레플리카의 행이 업데이트, 삽입 및 삭제되는 것 중의 적어도 하나인 경우, 상기 파티션 내의 변화를 처리하는 변화 추적 로직으로 추적되는, 동기화를 용이하게 하는 방법.

청구항 34

제22항에 있어서, 마지막 동기화 이후에 발생한 상기 다른 클라이언트들 내의 변화들을 처리하는 변화 열거 메커니즘을 이용하여 동기화 동안 상기 다른 클라이언트들 내의 변화들을 열거하는 단계를 더 포함하는, 동기화를 용이하게 하는 방법.

청구항 35

제34항에 있어서, 삭제 및 파티션 업데이트는 세션의 합의된 동기화 앵커보다 새로운 동기화 앵커를 갖는 행을 선택함으로써 열거되는, 동기화를 용이하게 하는 방법.

청구항 36

제35항에 있어서, 상기 행은 상기 행의 과거 변화들을 반영하는 메타데이터로부터 선택되는, 동기화를 용이하게 하는 방법.

청구항 37

제34항에 있어서, 삽입 및 파티션 업데이트는 세션의 합의된 동기화 앵커보다 새로운 동기화 앵커를 갖는 행을 선택함으로써 열거되는, 동기화를 용이하게 하는 방법.

청구항 38

제37항에 있어서, 상기 행은 상기 행의 현재 변화들을 반영하는 메타데이터로부터 선택되는, 동기화를 용이하게 하는 방법.

청구항 39

제22항에 있어서, 상기 첫 번째 목적지 레플리카로부터 전파된 변화들에 동기화 앵커를 할당하는 단계를 더 포함하고, 상기 동기화 앵커는 세대 식별 정보의 형태인, 동기화를 용이하게 하는 방법.

청구항 40

제39항에 있어서, 첫 번째 파티션의 상기 세대 식별 정보는 상기 각각의 클라이언트가 동기화할 때, 두 번째 파티션의 세대 식별 정보로 덮어쓰기되는, 동기화를 용이하게 하는 방법.

청구항 41

제22항의 상기 방법을 수행하기 위한 컴퓨터 실행 가능 명령어들이 저장되어 있는 컴퓨터 판독 가능 기록 매체.

청구항 42

레플리카들을 사용하여 클라이언트/서버 환경에서 동기화를 용이하게 하는 시스템을 저장하는 컴퓨터 판독 가능 기록 매체에 있어서,

상기 시스템은,

첫 번째 클라이언트에서의 첫 번째 레플리카로부터의 데이터의 파티션 업데이트에 관련된 정보를 수신하는 수단 - 상기 파티션 업데이트는 상기 첫 번째 레플리카의 마지막 동기화 이후 변형되었던 상기 첫 번째 레플리카의 모든 행들을 나타냄 - ;

상기 파티션 업데이트의 모든 행들에 세대 식별자를 할당하는 수단 - 상기 세대 식별자의 값은 상기 파티션 업데이트의 모든 행들에 대하여 동일하고, 파티션은 레플리카 데이터의 행들의 서브세트임- ;

다른 클라이언트들에 의한 동기화 이전에 상기 다른 클라이언트들의 레플리카들 중 어느 것이 상기 파티션 업데이트에 의해 영향을 받게 될지를 결정하는 수단 - 상기 결정은 결정을 위해 서버 테이블들 상의 세트 기반 질의를 이용하고, 상기 서버 테이블들은 파티션 메타데이터 테이블, 현재 변화 메타데이터 테이블, 및 과거 변화 메타데이터 테이블을 포함하고, 상기 파티션 메타데이터 테이블은 각각의 목적지 레플리카에 관련된 각각의 파티션을 식별하고, 상기 현재 변화 메타데이터 테이블은 현재 각각의 파티션의 부분인 각각의 행을 식별하며, 상기 과거 변화 메타데이터 테이블은 과거에 각각의 파티션의 부분이었던 각각의 행을 식별함 - ;

상기 클라이언트들에 의한 동기화 시에, 각각의 파티션에 관련된 각각의 세대 식별자를 식별하는 세대 파티션 매핑 테이블 및 각각의 클라이언트에게 전파되지 않은 변화들의 세대들을 식별하기 위하여 각각의 클라이언트에게 전파된 각각의 세대를 식별하는 세대 메타데이터 테이블을 이용하는 수단; 및

상기 복수의 클라이언트들에 걸쳐 파티션 일관성을 도모하기 위하여, 파티션 변화에 의해 영향을 받는 것으로 결정된 클라이언트들을 각각의 클라이언트에게 전파되지 않은 모든 변화들의 세대들로 업데이트하는 수단

을 포함하는,

레플리카들을 사용하여 클라이언트/서버 환경에서 동기화를 용이하게 하는 시스템을 저장하는 컴퓨터 판독 가능 기록 매체.

청구항 43

레플리카들을 사용하여 클라이언트/서버 환경에서 동기화를 용이하게 하는 방법에 있어서,

첫 번째 레플리카로부터의 데이터의 파티션 업데이트(partition update)에 관련된 정보를 수신하는 단계 - 상기 수신된 정보는 다른 클라이언트들과의 동기화를 위하여 지속되는 첫 번째 레플리카의 마지막 동기화 이후의 상기 첫 번째 레플리카의 모든 변화된 행들의 멤버쉽 정보인 복제 메타데이터의 형태이고, 상기 복제 메타데이터는 상기 파티션 업데이트의 행들의 모든 변화들에 대한 세대 식별자를 포함하고, 상기 세대 식별자의 값은 상기 파티션 업데이트의 모든 행들에 대하여 동일하며, 파티션은 레플리카 데이터의 서브세트를 식별함 - ;

다른 클라이언트들에 의한 동기화 이전에, 상기 다른 클라이언트들 중 어느 것이 상기 파티션 업데이트에 의해 영향을 받게 될지를 결정하는 단계 - 상기 결정 은 결정을 위해 서버 테이블들 상의 세트 기반 질의(a set based query)를 이용하고, 상기 서버 테이블들은 파티션 메타데이터 테이블, 현재 변화 메타데이터 테이블, 및 과거 변화 메타데이터 테이블을 포함하고, 상기 파티션 메타데이터 테이블은 각각의 목적지 레플리카에 관련된 각각의 파티션을 식별하고, 상기 현재 변화 메타데이터 테이블은 현재 각각의 파티션의 부분인 각각의 행을 식별하며, 상기 과거 변화 메타데이터 테이블은 과거에 각각의 파티션의 부분이었던 각각의 행을 식별함 - ;

상기 클라이언트들에 의한 동기화 시에, 각각의 파티션에 관련된 각각의 세대 식별자를 식별하는 세대 파티션 매핑 테이블 및 각각의 클라이언트에게 전파되지 않은 변화들의 세대들을 식별하기 위하여 각각의 클라이언트에게 전파된 각각의 세대를 식별하는 세대 메타데이터 테이블을 이용하는 단계; 및

상기 복수의 클라이언트들에 걸쳐 파티션 일관성을 도모하기 위하여, 상기 파티션 변화에 의해 영향을 받는 것으로 결정된 클라이언트들을 각각의 클라이언트에게 전파되지 않은 모든 변화들의 세대들로 업데이트하는 단계

를 포함하는, 동기화를 용이하게 하는 방법.

청구항 44

레플리카들을 사용하여 데이터 콜렉션들의 동기화를 용이하게 하는 시스템을 저장하는 컴퓨터 판독 가능 기록매체에 있어서,

상기 시스템은,

수신자 컴포넌트,

결정 컴포넌트, 및

업데이트 컴포넌트

를 포함하고,

상기 수신자 컴포넌트는 첫 번째 레플리카로부터의 데이터의 파티션 업데이트에 관련된 정보를 수신하고 - 상기 파티션 업데이트는 상기 첫 번째 레플리카의 마지막 동기화 이후 변형되었던 상기 첫 번째 레플리카의 모든 행들을 나타내고, 상기 수신자 컴포넌트는 상기 파티션 업데이트의 모든 행들에 세대 식별자를 할당하고, 상기 세대 식별자의 값은 상기 파티션 업데이트의 모든 행들에 대하여 동일하며, 파티션은 레플리카 데이터의 행들의 서브세트임 - ,

상기 결정 컴포넌트는 다른 목적지 데이터 콜렉션들에 의한 동기화 이전에 다른 목적지 데이터 콜렉션들 중 어느 것이 파티션 변화에 의해 영향을 받게 될지를 결정하고 - 상기 결정 컴포넌트는 결정을 위해 서버 테이블 상의 세트 기반 질의를 이용하고, 상기 서버 테이블들은 파티션 메타데이터 테이블, 현재 변화 메타데이터 테이블, 및 과거 변화 메타데이터 테이블을 포함하고, 상기 파티션 메타데이터 테이블은 각각의 목적지 레플리카에 관련된 각각의 파티션을 식별하고, 상기 현재 변화 메타데이터 테이블은 현재 각각의 파티션의 부분인 각각의 행을 식별하며, 상기 과거 변화 메타데이터 테이블은 과거에 각각의 파티션의 부분이었던 각각의 행을 식별함 - ,

상기 업데이트 컴포넌트는 상기 목적지 데이터 콜렉션들에 의한 동기화 시에, 상기 복수의 목적지 데이터 콜렉션들에 걸쳐 파티션 일관성을 도모하기 위하여, 상기 파티션 변화에 의해 영향을 받는 것으로 결정된 각각의 목적지 데이터 콜렉션은 그 목적지 데이터 콜렉션에 전파되지 않은 모든 변화들의 세대들로 업데이트되는 - 상기 업데이트 컴포넌트는 세대 식별자들과, 각각의 파티션에 관련된 각각의 세대 식별자를 식별하는 세대 파

티션 매핑 테이블 및 각각의 목적지 데이터 콜렉션에 전파되지 않은 각각의 세대를 식별하는 세대 메타데이터 테이블을 이용함 -,

컴퓨터 판독 가능 기록 매체.

청구항 45

제44항에 있어서, 상기 파티션 업데이트에 관련된 정보는 복제 메타데이터의 형태로 수신된 컴퓨터 판독 가능 기록 매체.

명 세 서

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

[0015] 본 발명은 네트워크 데이터 아키텍처(network data architecture)에 관한 것으로, 특히 이종 시스템(disparate system)에 걸친 그러한 네트워크 기반 데이터(network-based data)를 업데이트하는 것에 관한 것이다.

[0016] 인터넷과 같은 세계적인 통신 네트워크의 도래는 다양한 지리적 위치에 존재하는 개별적인 법인체 또는 지사(branch office)에 의해 전형적으로 이용되는 이종 데이터베이스의 형태로 법인 기업(들)에 대한 정보의 광범위한 보급을 용이하게 했다. 이러한 이질적인 데이터 소스를 동질적인 데이터베이스로 병합하는 것은 많은 시스템 프로세스의 중복을 초래하는 중대한 시스템 집약적 프로세스(system-intensive process)를 내포한다.

[0017] 이종 데이터베이스는 복제(replication)의 수단을 통해 집중될 수 있다 - 복제는 하나의 데이터베이스로부터 다른 것으로 데이터 및 데이터 개체를 복사하여 분배하고, 일관성을 위하여 데이터베이스 간의 정보를 동기화하는 프로세스를 말한다.

[0018] 병합 복제(merge replication)는 어려운 복제 타입이다. 병합 복제 능력은 접속이 끊기고 이동 중인(disconnected and mobile) 사용자가 오프라인으로 애플리케이션을 실행시키고, 메인 데이터베이스(main database)와 동기화하기 위하여 주기적으로 재접속하는 것을 가능하게 한다. 접속되었거나 접속이 되지 않은 동안에 소스(또는 게시자(publisher)로 불림) 및 목적지(또는 구독자(subscriber)로 불림) 상의 복제된 데이터에 자율적인 변화가 가능하고, 그 후에 사이트(site)가 접속된 경우에 사이트 간에 업데이트를 병합한다. 병합 복제를 이용하여, 서버는 소스 데이터베이스 및 목적지 데이터베이스의 증가하는 데이터 변화를 획득하고, 사전 설정된 규칙에 따르거나, 충돌(conflict)을 해결하기 위한 커스텀 해결자(custom resolver)를 사용함으로써 충돌을 조절한다.

[0019] 병합 복제는 전형적으로 소스 및/또는 목적지에서의 복제된 데이터의 자율적인 변화를 지원하기 위하여 사용된다. 데이터는 예정된 시간 또는 요구에 따라 서버들 간에 동기화된다. 업데이트는 하나 이상의 서버에서 독립적으로(예컨대 no commit protocol) 수행되는데, 그러므로 동일한 데이터가 소스 또는 하나 이상의 목적지에 의해 업데이트될 수 있다. 결과적으로, 데이터 변경이 병합되는 경우에 충돌이 발생할 수 있다. 병합 복제는 병합 소스가 설정되는 경우에 정의될 수 있는 충돌 해결(conflict resolution)을 위한 기본 및 커스텀 선택 사항(default and custom choices)을 포함한다. 충돌이 발생한 경우에, 병합 에이전트는 충돌 해결자(conflict resolver)를 호출하여 어떠한 데이터가 수용되고, 다른 목적지 사이트로 전파될 것인지를 결정한다. 병합 복제에 이용 가능한 옵션은 수평으로 또는 수직으로 소스 데이터를 필터링하는 것, 조인 필터(join filter) 및 동적 필터의 사용을 포함하는 것, 대안적인 동기화 상대방을 사용하는 것, 병합 성능을 향상시키기 위하여 동기화를 최적화하는 것, 동기화를 보증하기 위하여 복제된 데이터를 유효화하는 것 및 부가할 수 있는 구독 데이터베이스(attachable subscription database)를 이용하는 것을 포함한다.

[0020] 병합 복제는 수평, 동적 및 조인 필터를 지원하는데, 이상의 모든 필터는 관리자가 복제될 데이터의 파티션(또는 테이블)을 생성하는 것을 가능하게 한다. 복제된 데이터의 필터링은 적어도 네트워크를 통해 전송될 데이터 양의 최소화, 목적지 레플리카에서 요구되는 저장 공간의 양의 감소, 개인적인 목적지 레플리카 요구 사항에 기초한 데이터 소스 와 애플리케이션의 커스텀화 및 각각의 데이터 파티션은 상이한 목적지 레플리카로 전송될 수 있음으로 인한 충돌의 회피 및 완화를 허용한다. 병합 복제는 다수의 레플리카가 동일한 데이터를 업데이트하는 것을 허용할지라도, 레플리카가 해체 세트(disjoint set)를 수신하는 것과 같이 데이터를 필터링하는 것은

하나의 목적지로 두 개의 레플리카가 동일한 데이터 값을 업데이트하지 않음을 보증한다.

[0021] 전통적으로, 병합 복제는 소스 레플리카의 파티션을 목적지 레플리카와 일치하도록 유지하기 위한 기술을 지원했다. 그러나, 현존하는 알고리즘은 소스 및 목적지 레플리카 간에 동시 발생의 동기화 세션을 요구함으로써 시스템의 심각한 성능 저하를 가져왔다. 전통적인 서버에서, 병합 복제는 목적지 레플리카가 소스 레플리카의 데이터의 부집합만을 수신하도록 하기 위하여 매우 정교한 파티션화 기술(partitioning technique)을 지원해야 했다. 이러한 기술은 소스 레플리카에서 높은 주파수의 CPU를 요구하고, 따라서 목적지 데이터베이스 복제를 유지하기 위하여 요구되는 동시적인 동기화의 수를 증가시킴으로써 네트워크를 확장시키기 위한 병목 현상을 생성했다. 집약적인 하드웨어 및 소프트웨어 프로세서 기능은 목적지 레플리카의 파티션이 소스 레플리카의 그것과 일치하도록 유지하기 위하여 필요한 변화의 리스트를 준비하는 "파티션 계산(partition computation)" 알고리즘과 연관된다.

[0022] 전통적인 파티션 계산 접근법 하에서, 클라이언트가 서버와의 데이터베이스 동기화를 요청하는 경우, 서버의 파티션 계산 기능은 두 데이터 베이스 간의 차이를 결정하기 위하여 동기화되지 않은 행의 파티션 멤버쉽을 계산하는 것을 포함한다. 클라이언트 데이터베이스는 전형적으로 서버에 존재하는 전체 데이터베이스의 부집합이므로, 계산 기능은 목적지 레플리카로 잠재적으로 전파될 필요가 있는 소스 레플리카의 모든 현재 변화를 실질적으로 검토하는 것을 더 포함하고, 그러한 변화들 중의 어떤 것이 목적지 레플리카와 관련되는지 결정한다. 예컨대, 목적지 레플리카 필터 기준(destination replica filter criteria)에 따라 자격이 부여된 행은 목적지 레플리카에 속하고, 목적지 레플리카에 업데이트되거나 삽입되어야 한다. 업데이트를 거치고 더 이상 필터 기준을 만족하지 않는 행은 더 이상 목적지 레플리카에 속하지 않고, 따라서 삭제되어야 한다. 이는 소스 레플리카에서의 업데이트는 목적지 레플리카로의 삭제 동작으로 전파될 수 있음을 의미한다. 추가적으로, 업데이트를 거치고 이제 목적지 레플리카에 속하는 행은 확장되고, (조인 필터가 존재한다면) 조인된 테이블(joined table)로부터 관련된 행을 획득해야 할 필요가 있다.

[0023] "동기화 세션당 파티션 계산(per synchronization session partition computation)" 접근법의 성능 및 확장(scaling)의 관점에서 부정적인 측면은 다음과 같다. 파티션 업데이트가 존재하는지와 무관하게, 모든 변화에 대하여 전통적인 동기화 세션은 변화된 행이 목적지 레플리카에 속하는지를 평가해야 할 필요가 있으므로, 전통적인 시스템은 중복된 프로세싱으로 부담을 가지게 된다. 각각의 동기화 세션 이전에 행이 변화하면, 모든 추후 동기화 세션은 그 행에 대해 새로운 파티션 멤버쉽 메타데이터를 설정함으로써 그 행이 목적지 레플리카에 속하는지를 다시 평가해야 한다. 행의 이전의 파티션 멤버쉽에 대한 지식은 동기화 세션 간에 기억되지 않는다. 이는 각각의 동기화 세션에 대하여 중복된 프로세싱에 이르게 한다. 예로서 전통적인 시스템의 프로세스 집약적인 태양을 설명하기 위하여, 1000개의 목적지 레플리카를 가지는 네트워크를 고려한다. 행이 변화를 거쳤다고 소스에 의해 결정되고, 목적지들로 게시를 요구하면, 1000개의 목적지 레플리카는 보다 최신으로(more up-to-date) 동기화된다. 행이 다시 변하면, 또 다른 1000개의 동기화 세션이 뒤따르고, 이러한 프로세스는 행이 1000번 바뀔 때까지 반복된다. 결국, 모든 동기화 세션은 동일한 행에 대하여 조합하여 총계 100만번의 파티션 계산 기능을 실행할 것이다.

[0024] 모든 행 변화에 대하여, 각각의 목적지 레플리카는 자신의 평가를 수행해야 하므로, 전통적인 시스템은 서투른 병행성(concurrency) 동작을 보여준다. 소스 및 목적지 사이에 다수의 동시적인 동기화가 존재하는 경우, 동시에 실행되는 다수의 파티션 계산 인스턴스가 존재하고, 이는 소스 서버에서 높은 CPU 소비를 야기한다.

[0025] 목적지 레플리카의 파티션이 소스 레플리카의 파티션과 일치하도록 유지하기 위하여 필요로 하는 변화의 리스트를 준비하는 파티션 계산 알고리즘은 임의의 주어진 동기화 세션에서 실행되는 CPU 및 질의 프로세서 집약적인 기능이기 쉽다. 복제된 데이터의 세트를 정의하기 위하여 사용되는 필터링이 최적이 아니면, 동시에 발생하는 파티션 계산 기능의 수행은 소스 서버의 능력을 더욱 저하시킬 것이다.

[0026] 더 빠른 하드웨어 및 보다 효율적인 복제 설정(replication configuration)은 이러한 문제를 어느 정도까지 처리할 수 있다. 그러나, 완벽하게 디자인되고 조절된 애플리케이션에서도, 모든 게시자 서버(Publisher server)는 효과적으로 다룰 수 있는 동시적인 구독자 병합 프로세스(Subscriber merge process)의 수에 대하여 상한(upper limit)을 가지고 있다.

[0027] 이 문제에 대한 한 가지 해결책은 소스 병합 프로세스의 시간을 조절하여 프로세스들이 서로 교차되도록 하는 것이다. 변화를 병합하기 위하여 모든 목적지가 업무 일(work day)의 시작 또는 끝에 동시에 접속한다면, 병합 프로세스는 하루 중의 비절정 시점(off-peak time)에 수행되도록 요구될 수 있을 것이다. 언제 목적지가 병합 프로세스를 수행할 것인지에 대한 제어가 없다면, 동시에 병합이 허용되는 목적지의 수를 제한하기 위하여 상한

이 정의될 수 있다.

[0028] 소스 서버에 대한 요구를 감소시키는 다른 방법은 더 많은 소스로 프로세싱 부하를 분산시키는 것이다. 통상적인 기술은 재게시 계층 구조(republishing hierarchy)에서 여러 서버를 사용하는 것이다. 예컨대, 하나의 서버가 국가 내의 모든 판대 대리인에 대하여 소스로 동작하고 있다면, 둘 이상의 소스 서버가 이와 관련된 부하를 분산시키기 위하여 추가될 수 있는데, 중앙 게시자(Central Publisher)는 동부 및 서부 구독자(East and West Subscriber)에게 데이터를 게시하고, 동부 구독자는 동부의 판대 대리인에게 데이터를 재게시하며, 서부 구독자는 서부의 판대 대리인에게 데이터를 재게시한다.

[0029] 그러나, 전통적인 해결책 중의 어떤 것도 다수의 동기화와 연관된 문제를 처리하기 위한 비용 효과적(cost effective)이고 효율적인 아키텍처를 제공하지 않는다. 전통적인 시스템은 무관한 메타데이터를 네트워크를 통해 전파한다. 동기화 세션이 파티션을 계산한 후에 지속되는 멤버쉽 메타데이터(persisted membership metadata)가 존재하지 않으므로, 소스 레플리카의 모든 변화는 목적지 레플리카로의 전파와 관련된 것으로 간주될 것이다. 다수의 목적지 레플리카가 데이터를 소스 레플리카로 전파하는 경우에 고려될 변화의 리스트가 커지게 되므로, 이러한 접근법은 확장 문제에 기여한다. 추가적으로, 소스 레플리카가 목적지 레플리카와의 하나의 동기화 세션 동안 임의의 부가적인 메타데이터를 기록한다면, 다른 목적지 레플리카로 이 메타데이터를 전파해야 할 필요가 있을 것이다. 현재 접근법의 확장 특징(scaling characteristics)은 다소 바람직하지 않다. 당해 기술 분야에서 동기화 세션 동안 고려되고, 네트워크를 통해 전파될 메타데이터의 양을 최소화해야 할 필요성이 존재한다. 종래 기술에서 많은 동시 다발적인 동기화 세션에 대한 지원의 부족에 기초하여, 모든 동기화 세션의 실행 동안 값비싼 파티션 계산 기능을 제거해야할 필요성이 존재한다.

발명이 이루고자 하는 기술적 과제

[0030] 전통적인 해결책 중의 어떤 것도 다수의 동기화와 연관된 문제를 처리하기 위한 비용 효과적(cost effective)이고 효율적인 아키텍처를 제공하지 않는다. 전통적인 시스템은 무관한 메타데이터를 네트워크를 통해 전파한다. 동기화 세션이 파티션을 계산한 후에 지속되는 멤버쉽 메타데이터(persisted membership metadata)가 존재하지 않으므로, 소스 레플리카의 모든 변화는 목적지 레플리카로의 전파와 관련된 것으로 간주될 것이다. 다수의 목적지 레플리카가 데이터를 소스 레플리카로 전파하는 경우에 고려될 변화의 리스트가 커지게 되므로, 이러한 접근법은 확장 문제에 기여한다. 추가적으로, 소스 레플리카가 목적지 레플리카와의 하나의 동기화 세션 동안 임의의 부가적인 메타데이터를 기록한다면, 다른 목적지 레플리카로 이 메타데이터를 전파해야 할 필요가 있을 것이다. 현재 접근법의 확장 특징(scaling characteristics)은 다소 바람직하지 않다. 당해 기술 분야에서 동기화 세션 동안 고려되고, 네트워크를 통해 전파될 메타데이터의 양을 최소화해야 할 필요성이 존재한다. 종래 기술에서 많은 동시 다발적인 동기화 세션에 대한 지원의 부족에 기초하여, 모든 동기화 세션의 실행 동안 값비싼 파티션 계산 기능을 제거해야할 필요성이 존재한다.

발명의 구성 및 작용

[0031] 이하는 발명의 일부 태양에 대한 기본적인 이해를 제공하기 위하여 발명의 간단한 요약을 제시한다. 본 요약은 발명의 광범위한 개관이 아니다. 발명의 중심/주요 요소를 식별하거나, 발명의 범위를 묘사하려는 의도가 아니다. 유일한 목적은 이하 제시될 보다 상세한 설명에 대한 서문으로서 단순한 형태로 발명의 일부 개념을 제시하는 것이다.

[0032] 본 발명은 파티션 계산을 위한 신규한 아키텍처를 제공하고, 많은 수의 동시 다발적인 동기화 세션을 요구하는 배포를 지원하는 병합 복제를 통해 동시 다발적인 동기화 세션의 수가 크게 확장되는 것을 허용한다. 본 발명의 일 태양에 따라서, 목적지의 데이터 파티션이 소스의 파티션과 일치하도록 유지하기 위하여 새로운 알고리즘이 소개된다. 변화가 발생한 경우, 아키텍처가 동작하여 이러한 변화의 파티션 멤버쉽이 단순한 질의 세트를 사용하여 계산된다. 첫 번째 동기화의 파티션 멤버쉽의 사전 계산(pre-computation)은 소스 레플리카와 목적지 레플리카의 추후 동기화 세션 사이에서의 변화의 효율적인 전파를 허용한다.

[0033] 본 발명의 현저한 태양은 소스 레플리카와 목적지 레플리카 간의 모든 동기화 세션이 진행되는 동안 행의 파티션 멤버쉽을 계산하는 대신, 파티션 멤버쉽을 계산하는 부담이 행 변경(row modification)의 실제 시간에 초래된다는 것이다. 파티션 정보가 첫 번째 세션 업데이트 시점에 미리 계산(pre-computed)되고, 추후 동기화 세션 동안 지속되므로, 이러한 접근법은 매 동기화 세션 동안 파티션 계산을 제거하는 것을 허용한다.

[0034] 따라서, 이하 개시되고 청구되는 본 발명은, 일 태양에 있어, 데이터 레플리카를 사용하는 클라이언트/서버 환

경에서 동기화를 용이하게 하기 위한 아키텍처를 포함한다. 복수의 클라이언트가 서버에게 동기화를 요청하는 경우, 동기화를 위해 선택된 첫 번째 클라이언트는 데이터베이스의 파티션 멤버쉽 행이 계산되고, 지속되는 메타데이터(persisted metadata)를 사용하여 기록되는 방식으로 처리된다. 첫 번째 클라이언트를 업데이트 한 후에, 지속된 메타데이터에 의해 결정된 바와 같이, 업데이트에 의해 영향을 받게 될 잔존하는 클라이언트에 대해서만 동기화 프로세스가 계속된다.

[0035] 본 발명의 신규한 태양은 오직 처음으로 행이 변화된 경우에만 프로세싱이 수행되도록("동기화 세션 당 파티션 계산"과 유사) 허용하는데, 그 이후에는 파티션 멤버쉽 정보를 지속한다. 개시된 아키텍처와 반대로, 전통적인 아키텍처는 멤버쉽 정보를 "잊어버리고(forget)", 따라서 각각의 동기화 세션에 대하여 이러한 정보의 계산을 요구한다. 본 발명의 일 태양에 따라, 추후의 동기화 세션 동안 소스 레플리카에서의 변화는 변화가 잔존하는 목적지와 연관되었는지 평가하기 위하여 지속되는 멤버쉽 정보를 사용하여 목적지 레플리카로 전파된다. 더욱이, 동일한 행이 여러 번 변했다면, 관련된 업데이트가 없는 경우(아마도 업데이트에 대한 가장 통상적인 시나리오임)에도 첫 번째 이후에는 파티션 계산이 수행조차 되지 않는다. 이는 지속되는 멤버쉽 정보가 업데이트가 있을 때까지 유효하고, 추후의 동기화는 동일한 지속되는 멤버쉽 정보를 여전히 사용할 수 있기 때문이다.

[0036] 목적지에서 행이 업데이트를 거치는 경우, 그 행에 대한 멤버쉽 정보가 업데이트된다. 추가적으로, 업데이트를 거치고, 목적지 레플리카에 속하는 행은 (조인 필터가 존재한다면) 조인된 테이블로부터 관련된 행을 얻기 위하여 확장될 필요가 있을 수 있다. 계산의 중요한 태양은 그것이 단순한 세트 기반 질의(set-based queries)를 사용하여 수행된다는 점이다. 이 세트 기반 질의의 성능은 각각의 변화된 행이 모든 파티션의 작은 부집합, 또는 최상의 경우 잘 파티션된 데이터의 정확히 하나의 파티션에만 속하는 한도에서, 많은 파티션이 존재하더라도 매우 잘 확장된다.

[0037] 변화된 행에 대해 지속되는 멤버쉽 정보는 본질적으로, 변화된 행의 행 식별자(row identifier)와 행을 수신하도록 자격이 부여된 목적지 레플리카의 파티션 식별 사이의 매핑이다. 삽입 및 업데이트는 파티션 멤버쉽의 재평가를 필요로 하고, 또한 관련된 행을 얻기 위해 확장(expansion)이라 불리는 프로세스의 수행을 필요로 한다. 즉, 확장은 부모 행(parent row)이 업데이트됨으로써 변화된 파티션 멤버쉽을 가지는 자식 행(child row)을 처리하고, 그들의 파티션 식별자를 재평가해야 한다.

[0038] 삭제 및 업데이트에 있어서, 과거 멤버쉽에 대한 정보는 지속되어야 하는데, 다음 동기화 세션 동안 그 행이 목적지 레플리카로부터 삭제되어야 할 필요가 있는지 알기 위한 쉬운 방법이 존재하도록 하기 위함이다.

[0039] 전술한 그리고 관련된 목적의 달성으로서, 발명의 일정한 도시적인 태양이 다음의 기술 및 첨부된 도면과 관련하여 기술되었다. 그러나, 이러한 태양들은 발명의 원리가 사용될 수 있는 다양한 방법의 일부만을 나타내고, 본 발명은 모든 그러한 태양 및 그들의 균등물(equivalent)을 포함하려는 의도이다. 발명의 다른 이점 및 신규한 특징은 도면과 관련하여 고려되는 경우, 다음의 발명의 상세한 설명으로부터 명확해질 것이다.

[0040] 정의

[0041] 다음의 용어는 상세한 설명 전반에 사용되고, 본 발명의 다양한 태양의 이해를 돕기 위하여 용어의 정의가 제공된다.

[0042] 소스 레플리카(Source replica) : 변화가 발생한 데이터 세트(data set).

[0043] 목적지 레플리카(Destination replica) : 변화가 전파될 데이터 세트.

[0044] 부분 레플리카(Partial replica) : 소스 레플리카의 데이터의 부집합을 포함하는, 소스로부터 목적지로 게시되는(published) 데이터 세트.

[0045] 파티션(Partition) : 데이터의 부집합, 그리고 데이터의 동일한 부집합을 수신하는 모든 레플리카는 동일한 파티션에 존재하는 것으로 간주되고, 동일한 파티션 식별자를 할당받을 수 있다.

[0046] 파티션 재정렬(Partition realignment) : 목적지 레플리카의 행(row)의 파티션 멤버쉽(partition membership)이 변화하도록 야기하는 변경(modification). 예컨대, 파티션 레플리카의 행의 멤버쉽이 WHERE 구문을 사용하여 단정되면, 컬럼을 상이한 값으로 업데이트하는 임의의 행 변경은 파티션 재정렬을 구성한다. 예컨대, 필터 "where state='WA'"가 Customers 테이블에 사용되면, 행 내의 "state" 컬럼의 값의 변화는 행의 파티션 재정렬을 야기한다.

[0047] 이전 값(Before values) : 업데이트 동작 직전의 데이터의 값.

[0048] 이후 값(After values) : 업데이트 동작 직후의 데이터의 값.

[0049] 행 필터(Row filters) : 테이블 내의 행의 부집합이 소스로부터 목적지로 게시되는 것을 허용하는 필터. 행 필터는 질의의 WHERE 구문을 사용하고, 특정 기준(specific criteria)에 기초하여 파티션에 포함된 행을 제한한다.

[0050] 조인 필터(Join filters) : 하나의 테이블의 필터가 게시(publication) 내의 다른 테이블에 기초한 경우, 테이블 간 관계(cross-table relationship)가 복제 필터의 정의에 사용될 수 있도록 허용하는 필터. 조인 필터는 동기화 세션 동안 강화될 두 테이블간의 관계를 정의하는데, 이는 두 테이블간의 조인(join)을 지정하는 것과 유사하다. 조인 필터는 두 테이블에 이름을 부여하고, 두 테이블간의 관계를 나타내기 위한 조인 조건(join condition)을 지정한다. 조인 조건은 대게 TABLE1.COLUMN1 = TABLE2.COLUMN2 의 형태이다.

[0051] 동적 필터(Dynamic filters) : 목적지 레플리카로부터 값을 검색하기 위하여 함수를 사용하고, 그 값에 기초하여 데이터를 필터링하는 행 필터. 필터는 한번만 정의되지만, 결과 세트에 자격을 부여하는 것(qualifying resultant set)은 각각의 목적지 레플리카에 대하여 상이할 수 있고, 목적지 레플리카가 오직 자신의 필요에 따라 커스터마이즈된 데이터의 부집합만을 수신하는 것을 허용한다.

[0052] 동기화(Synchronization) : 소스 및 목적지 레플리카로부터 최종 수렴 상태(final convergent state)로 데이터 세트를 집중시키는 프로세스.

[0053] 동기화 앵커(Synchronization anchor) : 동기화를 벗어난(out-of-sync) 두 레플리카를 어떻게 할 것인지 결정하는 엔티티. 이는 대게 마지막 두 레플리카가 동기화되는 시점을 묘사하는 "논리적인 시계 엔티티(logical clock entity)"로 모델링된다.

[0054] 충돌 감지(Conflict detection) : 변경이 충돌 상태에 있는지 결정하기 위하여 소스 레플리카 및 목적지 레플리카에 있는 메타데이터를 질문하는 동기화 동안 발생하는 프로세스.

[0055] 충돌 해결(Conflict resolution) : 충돌이 발생한 경우, 충돌의 승자와 패자를 결정하는 동기화 동안 발생하는 프로세스.

[0056] 본 발명은 도면을 참조하여 기술되는데, 전체에 걸쳐 유사한 참조 번호는 유사한 요소를 언급하기 위하여 사용된다. 다음의 기술에서, 설명의 의도로 본 발명의 완벽한 이해를 제공하기 위해 많은 특정 세부 사항이 제시된다. 그러나, 본 발명은 이러한 특정 세부 사항 없이도 실시될 수 있음이 명백할 수 있다. 다른 예에서, 잘 알려진 구조 및 장치는 본 발명의 기술을 용이하게 하기 위하여 블록도 형식으로 도시된다.

[0057] 본 명세서에서 사용된 바와 같이, "컴포넌트(component)" 및 "시스템"은 컴퓨터 관련 엔티티(entity), 즉 하드웨어, 하드웨어 및 소프트웨어의 조합, 소프트웨어, 또는 실행 중인 소프트웨어를 언급하려는 의도이다. 예컨대, 컴포넌트는 프로세서 상에서 실행되는 프로세스, 프로세서, 개체, 실행 가능(executable), 실행 스레드(thread of execution), 프로그램 및/또는 컴퓨터가 될 수 있다(이에 제한되지 않음). 도시의 방법으로, 서버 상에서 실행되는 애플리케이션 및 서버 모두가 컴포넌트일 수 있다. 하나 이상의 컴포넌트가 프로세스 및/또는 실행 스레드 내에 존재할 수 있고, 컴포넌트는 하나의 컴퓨터에 로컬라이즈(localize)되고/되거나 둘 이상의 컴퓨터에 분산될 수 있다.

[0058] 도 1을 참조하면, 본 발명의 시스템 블록도가 도시되어 있다. 다음 논의는 시스템 동작 중에 소스 및 다수의 목적지가 이전에 동기화된 지점부터 시작한다. 각각의 목적지는 그들 각각의 데이터베이스에 변화를 발생시키고, 이제 데이터베이스의 동기화를 요청하기 위하여 소스와의 통신을 재설정한다.

[0059] 서버(또는 소스)(100)는 복수의 N 클라이언트(또는 목적지)로부터 동기화 요청(synchronization request)을 수신한다. 소스(100)는 동기화 요청을 처리하고, 모든 소스 시스템 동작을 제어하기 위하여 CPU(Central Processing Unit)(102)를 포함한다. CPU(102)는 복수의 목적지들에 의해 이용되는 데이터베이스 엔트리를 저장하기 위하여 소스(또는 마스터) 데이터베이스(104)와 인터페이스한다. 그러므로, 또한 소스 레플리카라고도 불리는 소스 데이터베이스(104)는 N 목적지로 잠재적으로 게시될 필요가 있는 모든 정보를 실질적으로 포함한다.

[0060] N 목적지로부터 다수의 동기화 요청을 수신한 후에, 첫 번째 목적지(106)는 동기화를 위하여 소스(100)에 의해 선택될 것이다. 첫 번째 목적지(106)는 목적지(106)와 관련된 모든 데이터베이스 엔트리를 저장하는 첫 번째

목적지 데이터베이스(108)를 포함하는데, 이는 마지막 동기화 이후에 행해진 가장 최근의 변화를 포함한다. 그러므로, 소스(100)의 마스터 데이터베이스(104)의 데이터와는 다른, 첫 번째 목적지 데이터베이스(108) 상에 저장된 데이터가 존재한다.

[0061] 첫 번째 목적지(106)를 선택한 후에, 소스(100)는 소스 데이터베이스(104)와 목적지 데이터베이스(108) 사이의 데이터 차이(또는 변화)를 확인하기 위하여 목적지 데이터베이스(108)와의 이전의 동기화 세션 후에 수행된 데이터베이스 업데이트의 세트를 결정한다. 소스(100)는 변화된 데이터의 파티션을 생성하는 파티션 계산 알고리즘(110; partition computation algorithm)을 포함하는데, 파티션은 첫 번째 목적지(106)와 관련된 변화들만을 정의한다. 이 파티션은 소스(100)에 멤버쉽 메타데이터(114; membership metadata)로써 지속되고, 이전의 동기화 프로세스 후에 변화된 목적지 데이터베이스(108)의 행 엔트리를 표시한다. 멤버쉽 메타데이터(114)는 잔존하는 목적지(2 내지 N)의 추후 동기화 세션과의 이용을 위하여 소스(100)에 저장된다.

[0062] 첫 번째 목적지가 첫 번째 레플리카 데이터(112)에 의해 소스(100)와 동기화된 후에, 첫 번째 목적지(106)의 동기화는 완료된다.

[0063] 첫 번째 목적지(106)의 업데이트된 데이터베이스 정보는 이제 동기화를 기다리고 있는 목적지들(2 내지 N)의 일부 또는 전부에 병합 복제에 의하여 전파될 필요가 있다. 본 발명의 일 태양에 따르면, 추후 목적지 동기화는 첫 번째 목적지(106)의 동기화 동안 수행된 모든 계산 동작을 수행할 필요가 없고, 첫 번째 동기화에 의해 지속된 멤버쉽 메타데이터(114)로부터 이익을 얻는다. 그러므로, 멤버쉽 메타데이터(114)는 목적지(2 내지 N)로의 다운로드를 위한 레플리카를 생성하기 위하여 이용된다. 전통적인 아키텍처는 모든 추후의 동기화에 대하여 파티션 멤버쉽을 재계산할 것을 요구하기 때문에, 이는 전통적인 아키텍처에 대하여 소스(100)에서의 프로세싱 시간(processing time)에 있어 많은 절약을 가져온다. 더욱 중요하게는, 값비싼 파티션 계산 비용은 그것이 동기화 동안에 반대되는 실제 업데이트 시에 실행되기 때문에 다수의 동기화 세션으로 양도된다. 이 기술은 다수의 동기화 세션이 소스 레플리카에 대하여 동시에 실행할 경우에 막대한 부하에서 더욱 확장된다.

[0064] 동작에 있어서, 두 번째 목적지 데이터베이스(118)를 가지는 두 번째 목적지(116)가 동기화를 위하여 소스(100)에 의해 선택된다. 소스(100)는 두 번째 목적지(116)와 연관된 필터 기준(filter criteria)을 획득하고, 정확히 두 번째 목적지(116)의 파티션 멤버쉽에 기초하여 어떤 변화가 두 번째 목적지(116)로의 전파와 관련되어 있는지 결정한다.

[0065] 소스(100)가 N 번째 목적지 데이터베이스(124)를 가지는 N 번째 목적지(122)와의 동기화를 위하여 멤버쉽 메타데이터(114)를 이용하는 방식으로, 동기화 프로세스는 잔존하는 목적지(N)에 대하여 계속된다. N 번째 목적지로의 다운로드를 위하여(요구된다면) 변화(126)의 세트를 생성하기 위한 멤버쉽 메타데이터에 대하여, N 번째 목적지(122)에 대한 필터 기준이 획득, 분석 및 적용된다. 주어진 레플리카에 지속되는 동기화 메타데이터가 종결 처리(cleanup)에 대하여 안전하다고 판단된 경우(예컨대 보유 기반 정책(retention based policy)에 근거하여 즉, 주어진 기간 내에 동기화하지 않은 레플리카는 추후의 동기화가 금지됨), 대응되는 파티션 멤버쉽 메타데이터도 종결 처리에 대하여 안전하다고 판단된다.

[0066] 기술(description)이 클라이언트/서버 환경과 관련하여 제공되지만, 본 발명은 동기화를 요구하는 임의의 동질적인 데이터 콜렉션(homogeneous data collection)에 적용된다. 마찬가지로, 본 발명은 피어 대 피어 컴퓨팅 환경(peer to peer computing environment)에 적용된다. 예컨대, 소스 데이터 콜렉션과 동기화를 요구하는 적어도 두 개의 목적지 데이터 콜렉션이 존재하는 경우, 데이터 콜렉션은 기술된 신규한 태양에 따라 동기화될 수 있다.

[0067] 도 2를 참조하면, 본 발명의 복제 프로세스의 흐름도가 도시되어 있다. 설명의 단순화를 위해, 일련의 동작으로 방법론이 도시 및 기술되고 있지만, 본 발명에 따른 일부 동작이 다른 순서 및/또는 도시 및 기술된 것으로부터의 다른 동작과 동시에 발생하는 것처럼, 본 발명은 동작의 순서에 제한되지 않음을 이해하고, 인식해야 한다. 예컨대, 당업자는 상태도에서와 같이 대안으로서 일련의 상호 관계된 상태 또는 이벤트로 방법론을 나타낼 수 있음을 이해하고 인식할 것이다. 추가적으로, 본 발명에 따른 방법론을 구현하기 위하여 도시된 모든 동작이 요구되지 않을 수 있다.

[0068] 200에서, 복수의 목적지(1 내지 N)는 소스에 동기화를 요청한다. 202에서, 소스는 동기화를 위한 첫 번째 목적지를 선택한다. 처음으로 동기화를 요청한 목적지와 싱크 업(synch-up)을 요청하는 목적지의 우선 순위 스키마(priority schema)를 이용하는 것을 포함하는(이에 제한되지 않음) 다수의 방법으로 선택 프로세스(selection process)가 결정될 수 있다. 첫 번째 목적지가 선택되면, 204에 표시된 바와 같이 소스는 현재 상태를 설정하

기 위하여 첫 번째 목적지 데이터베이스에서 변화된 행의 세트(set of changed rows)를 결정한다. 206에서, 소스는 잔존하는 목적지(2 내지 N)들 중의 선택된 목적지로 어떤 변화가 전파될 것인지 결정하기 위하여, 검사(examination)를 위한 파티션 계산 알고리즘을 이용하여 소스 데이터베이스와 첫 번째 목적지 데이터베이스 사이의 차이를 결정한다. 208에서, 파티션 계산 알고리즘은 하나 이상의 메타데이터 테이블의 형태로 첫 번째 멤버쉽 메타데이터를 만들고, 멤버쉽 메타데이터를 소스에 저장한다. 210에서, 첫 번째 파티션 레플리카는 업데이트를 위하여 첫 번째 목적지로 다운로드된다. 업데이트가 완료되면, 첫 번째 목적지의 동기화가 완료된다.

[0069] 212에서, 소스는 동기화를 위한 다음 목적지를 선택한다. 214에서, 첫 번째 목적지의 특정 데이터 세트를 위해 다음 목적지에 대하여 동기화가 요구되는지 결정하기 위해, 다음 목적지의 필터 기준이 소스에 의해 획득된다. 그 후, 216에서 소스는 다음 목적지(또는 두 번째 목적지)를 위한 두 번째 파티션 레플리카를 생성하기 위하여 필터 기준 및 첫 번째 멤버쉽 메타데이터 모두를 이용한다. 218에서, 두 번째 파티션 레플리카가 다운로드되고, 220에서 다음 목적지를 위한 동기화 프로세스의 이 부분을 완료하기 위하여 파티션 업데이트가 수행된다. 프로세스 사이클은동기화를 위한 다음 목적지를 선택하기 위하여 212의 입력으로 돌아간다.

[0070] 위의 프로세스는 동기화를 요구하는 모든 목적지가 첫 번째 목적지의 변화된 정보를 수신할 때까지 계속되고, 그 후에 동기화가 다른 모든 목적지 등에 대하여 목적지(N)까지 두 번째 목적지의 변화된 데이터에 대하여 계속된다.

[0071] 도 3을 참조하면, 목적지의 멤버쉽 메타데이터를 계산하기 위한 계산 컴포넌트(computation component)에 의해 이용되는 메타데이터 테이블(300; metadata table)의 상관 관계가 도시되어 있다. 목적지 데이터베이스의 변화 정보(change information)를 획득하기 위해 이용되는 여섯 개의 테이블이 대체로 존재한다. 파티션 메타데이터 테이블(302; PartitionsMetadata로 표시)은 세 개의 다른 테이블 - 현재 변화 메타데이터 테이블(304; CurrentChangesPartitionMapping로 표시), 과거 변화 메타데이터 테이블(306; PastChangesPartitionMapping로 표시) 및 세대 파티션 메타데이터 테이블(308; GenerationPartitionMapping로 표시) - 에 의해 매핑된다(mapped to). GenerationPartitionMapping 테이블(308)은 세대 메타데이터 테이블(310; GenerationMetadata로 표시)에 매핑되는데, 테이블(310)은 또한 행 메타데이터 테이블(312; RowMetadata로 표시)에 의하여 매핑된다. 특정한 구현에 따라 임의의 적절한 수의 테이블 및/또는 메타데이터가 사용될 수 있음을 인식해야 한다.

[0072] 레플리카에서 행의 파티션 멤버쉽(partition membership of row)을 계산하기 위하여 이용되는 세 개의 메타데이터는 목적지 레플리카의 파티션을 식별하는 개별 파티션(distinct partitions)(PartitionsMetadata 테이블(302) 사용), 행의 현재 파티션 멤버쉽(CurrentChangesPartitionMapping 테이블(304) 사용) 및 행의 과거 파티션 멤버쉽(PastChangesPartitionMapping 테이블(306) 사용)을 포함한다. 현재 및 과거의 멤버쉽 정보는 다른 목적지 레플리카로의 파티션 업데이트의 효과적인 전파를 가능하게 한다. 행이 이전에 속했던 파티션에 대해 계속항 알고 있는 것이 바람직한데, 이는 "삭제"의 전파를 가능하게 하기 때문이다. 삭제는 목적지 레플리카에서 더 이상 요구되지 않는 데이터(또는 행)이다.

[0073] PartitionsMetadata 테이블(302)은 필터 기능에 있어 관심있는 평가를 추적한다. 목적지 레플리카가 소스 레플리카와 동기화하고, 목적지 레플리카를 식별하는 개별 파티션이 파티션 테이블(302)에 존재하지 않는다면, "파티션 값"을 가지는 새로운 엔트리가 생성되고 새로운 partition_id 파라미터가 할당된다.

[0074] Employees에 대한 동적 필터 표현식(dynamic filter expression)이 "where TerritoryID=fn_EmployeeTerritory()"이면, fn_EmployeeTerritory라고 불리는 컬럼(column)이 PartitionsMetadata 테이블(302)에 추가된다. Employees 행에 변화가 생긴 경우, 변화된 행이 속하는 모든 파티션이 하나의 세트 기반 질의(set based query)를 사용하여 계산되는데, 위의 세트 기반 질의는 fn_EmployeeTeritory()가 PartitionsMetadata.fn_EmployeeTerritory로 대체된 조인 구문(join clause)으로 필터 표현식을 사용하여 Employees를 PartitionsMetadata 테이블(302)과 조인한다.

[0075] 각각의 변화된 행이 모든 partition_id의 작은 부집합 또는 최적의 경우로, 오직 하나의 partition_id로 잘 파티션된 데이터의 부집합에 속하는 한도에서, 다수의 등록된 partition_id가 존재할지라도, 이러한 세트 기반 질의의 성능은 매우 잘 확장된다.

[0076] CurrentChangesPartitionMapping 테이블(304)은 주어진 행의 자신과 관련된 파티션에 대한 현재 매핑을 추적한다. 그러므로, 테이블(304)은 테이블에 partition_id 컬럼을 포함하는데, 이는 PartitionsMetadata 테이블(302)에서 유래된 값이다. CurrentChangesPartitionMapping 테이블(304)의 row_id 컬럼은 복제에 의해 사용되는 주어진 행의 유일한 식별자를 포함한다.

[0077] PastChangesPartitionMapping 테이블(306)은 자신이 속하는 임의의 파티션에 대한 주어진 행의 임의의 과거 매핑을 추적한다. 그러므로, 테이블(306)은 파티션 테이블(302)에서 유래된 값인 partition_id 컬럼을 포함한다. 테이블(306)의 row_id 컬럼은 복제에 의해 사용되는 주어진 행의 유일한 식별자를 포함한다. 동기화 앵커 컬럼(synchronization anchor column; synch_anchor)은 행의 파티션 매핑이 언제 변화되었는지에 대한 정보를 논리적으로 포함한다. 파티션 업데이트 동안 동기화 앵커를 획득함으로써 동기화 프로세스가 소스(100)로부터 소스 및 목적지가 마지막으로 동기화된 시점 이후로 이러한 변화를 수신하지 않은 목적지들로 파티션 업데이트를 전파하는 것이 가능하게 된다. 샘플 데이터에서, sync_anchor 컬럼은 단순함을 위해 UTC(Universal Time Coordinate) 시간 또는 GMT(Greenwich Mean Time) 값의 형태로 값을 이용할 것이다.

[0078] 세대 파티션 테이블(308)은 변화의 그룹에 할당된 동기화 앵커인 generation_id 컬럼을 포함한다. 변화가 목적지로부터 소스(100)로 전파되는 경우, 소스(100)에서 변화들은 새로운 generation_id를 할당받는다. 이 세대의 일부인 행은 개별 파티션 P1에 속하므로, GenerationPartitionMapping 테이블(308)은 목적지의 partition_id P1으로의 매핑을 위한 partition_id 컬럼을 포함한다. 그러나, 일반적으로 세대는 하나 이상의 파티션 식별자에 매핑될 수 있음에 주의해야한다. 이는 partition_id에 대하여 "-1"의 특별한 값을 갖는 세대와는 상이하다. 다른 개별 파티션 P2에 속하는 상이한 목적지 레플리카가 소스(100)와 동기화하는 경우, 세대 매핑은 P1 세대를 제거한다. partition_id에 대한 -1의 특별한 값은 세대가 전역적이고 따라서 모든 파티션과 연관되어 있음을 나타낸다.

[0079] 세대 테이블(310)은 각각의 세대에 유일한 세대 ID를 할당하기 위한 generation_id 컬럼을 포함한다. 세대 파티션 테이블(308)은 이 테이블(310)을 매핑하고, 어떤 세대가 현재 레플리카로 전파되었는지 및 어떤 세대가 로컬 변화를 나타내는지를 추적하는 generation_id를 포함한다. 이로 인하여 동기화 프로세스가 주어진 세션에 대하여 고려될 필요가 있는 연관된 세대의 리스트를 추출하는 것이 가능하다.

[0080] RowMetadata 테이블(312)은 행별 기준으로(on a row-by-row basis) 복제 메타데이터를 추적하고, 시간을 나타내는 논리적인 시계를 사용하여 언제 행이 변화되었는지에 대한 정보를 포함한다. 또한, 테이블(312)은 컬럼에 대한 현재 버전의 정보와 함께, 어떠한 레플리카가 이러한 버전의 행에 기여했는지에 대한 정보를 포함한다. 세대 파티션 테이블(308) 및 RowMetadata 테이블(312) 모두는 세대 테이블(310)에 매핑된다.

[0081] 다음의 표 1은 상이한 메타데이터 테이블 및 각각의 기능을 요약한 것이다.

표 1

[0082] 메타데이터 테이블 및 요약

메타데이터 테이블 이름	목적
RowMetadata	행별 기준으로 복제 메타데이터를 추적하고 언제 행이 변화되었는지(시간을 나타내는 논리적인 시계를 사용하여)에 대한 정보 및 행 및 열 버전 벡터(row and column version vector)를 포함한다.
GenerationMetadata	어떤 생성이 현재의 레플리카로 전파되는 지 및 어떤 생성이 로컬 변화를 나타내는지를 추적하고, 동기화 프로세스가 주어진 세션에 대하여 고려될 필요가 있는 연관된 생성의 리스트를 추출하는 것을 가능하게 한다.
PartitionsMetadata	목적지 레플리카를 나타내는 모든 파티션 값에 대한 개별 엔트리를 포함한다.
GenerationPartitionMapping	GenerationMetadata 테이블 내의 어떠한 새성이 어떠한 파티션과 연관된 것인지에 대한 매핑 정보를 포함한다.
CurrentChangesPartitionMapping	현재 어떠한 변화가 어떠한 파티션과 연관된 것인지에 대한 메타데이터를 포함한다.
PastChangesPartitionMapping	이전의 어떠한 변화가 어떠한 파티션과 연관된 것인지에 대한 메타데이터를 포함한다.

[0083] 변화의 효율적인 전파를 위한 세대의 파티션화

[0084] 본 발명의 신규한 태양은 변화의 전파동안 사용되는 최적화를 도모한다. 병합 복제는 소스로부터 목적지 레플

리카로 전파되는 변화를 논리적으로 그룹화하기 위하여 일반적으로 "세대"의 개념을 사용한다. GenerationMetadata 테이블(310)은 어떠한 세대가 현재 레플리카로 전파되었는지 및 어떠한 세대가 로컬 변화를 나타내는지를 추적한다. 또한, 세대 파티션화로 인해 동기화 프로세스가 주어진 세션에 대하여 고려되어야 할 필요가 있는 연관된 세대 리스트를 추출할 수 있다.

[0085] 소스 레플리카에 변화가 발생한 경우, 소스는 테이블 상의 한 세트의 변화에 세대 값을 할당하는데, 세대 값은 논리적인 시계 엔티티(logical clock entity)이다. 그룹화 개념으로 인해 목적지 레플리카는 변화의 그룹이 간섭받았을 수도 있는 이전의 동기화 세션으로부터 수신된 것인지, 상이한 소스 레플리카와의 동기화 세션을 통한 것인지를 효율적으로 식별할 수 있다. 본질적으로, 현재 목적지 레플리카에서 결여된 세대 값의 리스트는 소스로부터 목적지로의 전파에 대하여 고려되는 것과 관련된 변화를 반영한다. 목적지 레플리카가 소스 레플리카로부터 예컨대 목적지 레플리카와 같은 데이터의 부집합을 수신하면, 레플리카의 파티션 기준을 만족하는 변화는 목적지로 전파되는 유일한 변화이다.

[0086] 그러나, 병합 복제의 현재 버전은 여전히 소스 레플리카에 존재하는 모든 세대에 대한 정보를, 이러한 세대가 목적지 레플리카와 연관되었거나 연관되지 않은 변화를 포함하는지와 무관하게 목적지 레플리카로 전파해야 할 필요가 있다. 이는 소스 레플리카에서 세대의 일부인 변화가 관심있는 파티션과 연관되어 있는지 또는 무관한지를 추적하는 메타데이터가 없기 때문이다.

[0087] 행의 파티션 멤버쉽을 식별하는 파티션 그룹은 소스로부터 목적지 레플리카로의 세대의 전파에 효율성을 허용한다. 변화의 세트가 이미 파티션 식별자에 매핑되었으므로, 행의 그룹인 세대는 또한 파티션 식별자에 매핑될 수 있다. 그러므로, 목적지 레플리카가 소스 레플리카에서 이용 가능한 특정한 파티션에 관심이 있는 경우, 목적지 레플리카와 무관한 파티션 내의 변화를 포함하는 세대 값은 연관된 변화의 세트를 계산함에 있어 신속하게 제거될 수 있다. 계산 효율성을 제공함에 부가하여, 또한 이 알고리즘은 더 나은 네트워크 성능 특징을 가지는데, 오직 목적지 파티션과 연관된 세대만이 네트워크를 통해 전파되기 때문이다.

[0088] 도 4를 참조하면, 필터링과 확장을 이용하는 샘플 업데이트 스키마(update schema)가 도시되어 있다. 예는 재배치될 종업원을 전제로 하고 있다. 업데이트 프로세스는 재배치될 종업원의 고객 데이터를 다른 종업원에게 재할당하는 것을 포함한다. 이러한 예에서, 클라이언트 정보는 적어도 고객 정보, 고객 주문 정보 및 고객 주문 세부 사항을 포함한다. 파티션 테이블은 재배치되는 종업원을 EmployeeID로 유일하게 식별하는 Employees 파티션 테이블(400)이다. 이러한 예에서, 테이블(400)은 적어도 FirstName 컬럼, LastName 컬럼 및 TerritoryID 컬럼의 세 개의 컬럼을 포함한다. Employees 파티션에 대한 동적 조인 필터 표현식이 "where TerritoryID=fn_EmployeeTerritory()"이면, fn_EmployeeTerritory라고 불리는 컬럼이 Employees 파티션 메타데이터 테이블(400)에 추가된다.

[0089] 새로운 파티션이 처음으로 동기화하기 위해 도착하면, 새로운 파티션은 그들 각각의 파티션 값을 이용하여 이 테이블(400)에 엔트리를 등록하고, 새로운 partition_id를 할당받는다. Employees 테이블(400)의 행에 변화가 생긴 경우, 변화된 행이 속하는 모든 파티션이, 조인 구문으로서 필터 표현식을 사용하여 종업원을 Employees 파티션 메타데이터 테이블(400)과 조인하는 하나의 세트 기반 질의를 사용하여 계산된다. fn_EmployeeTerritory() 함수의 호출은 PartitionsMetadata.fn_EmployeeTerritory로 대체된다.

[0090] 예컨대, Employees 파티션 테이블(400)의 TerritoryID 컬럼이 미국 내의 51개 개별 지역으로 평가되면, Employees 파티션 테이블(400)은 이들 51개 개별 값의 각각에 유일한 파티션 식별(unique partition identity)을 할당할 것이다. 목적지가 소스(100)와 동기화하고, 목적지 레플리카를 식별하는 개별 파티션이 Employees 테이블(400)에 존재하지 않는다면, 개별적인 "파티션 값(partition value)"를 이용하여 새로운 엔트리가 생성되고, 새로운 partition_id가 할당된다.

[0091] 다음 표 2는 종업원의 성과 이름 컬럼을 제외한 샘플 Employes PartitionsMetadata 테이블(400)이다. 마지막 컬럼은 샘플 데이터의 해석을 제공하고 있으나, 스키마의 필수적인 부분이 아니다.

표 2

[0092] 샘플 PartitionsMetadata 테이블

Partition_id	fn_EmployeeTerritory	테이블의 해석(스키마의 일부 아님)
1	"WA"	파티션 ID = 1 은 워싱톤 영역에 대응된다.
2	"CA"	파티션 ID = 2 은 캘리포니아 영역에 대응된다.
3	"OR"	파티션 ID = 3 은 오레곤 영역에 대응된다.
...	...	...

[0093] 각각의 변화된 행이 모든 partition_id의 부집합 또는 최적의 경우로, 오직 하나의 partition_id로 잘 파티션된 데이터의 부집합에 속하는 한도에서, 다수의 등록된 partition_id가 존재할지라도, 이러한 세트 기반 질의의 성능은 매우 잘 확장된다.

[0094] 위에서 나타난 바와 같이, 모든 클라이언트 정보는 종업원과 함께 전파될 필요가 있다. 동기화 프로세스 동안에 사용될 테이블 간 관계(cross table relationship)를 정의함으로써 이 프로세스를 촉진하기 위하여 조인 필터가 사용된다. 샘플 스키마에서, Employees 테이블(400)의 행의 파티션 멤버쉽은 Emplyees 테이블(400) 상의 "행 필터(row filter)" 정의를 사용하여 서술된다. Customers 테이블(404) 내의 행의 파티션 멤버쉽은 이러한 Employees 및 Customers 간의 조인 필터 정의(402)를 사용하여 서술되는 Employee 테이블(400) 내의 행의 멤버쉽에 기초한다. Orders 테이블(408) 내의 행의 멤버쉽은 Customers 및 Orders 간의 조인 필터 정의를 사용하여 서술되는 Customers 테이블(404) 내의 행의 멤버쉽에 기초한다. 마찬가지로, Order Details 테이블(412) 내의 행의 멤버쉽은 Orders 및 Order Details 간의 조인 필터 정의를 사용하여 서술되는 Orders 테이블(408) 내의 행의 멤버쉽에 기초한다.

[0095] 그러므로, 첫 번째 조인 필터(402)는 Employees 테이블(400)과 Customers 테이블(404) 간에 테이블 간 관계(Customers.EmployeeID=Employees.EmployeeID)를 정의한다. Customer 테이블(404)은 고객을 유일하게 식별하는 CustomerID와 연관되고, Employees 테이블(400)에 매핑되는 적어도 EmployeeID 컬럼과 이름, 주소, 우편 번호 및 연락 번호와 정보 같은 고객 계좌 정보와 관련된 다른 컬럼을 포함한다. 두 번째 조인 필터(406)는 Customers 테이블(404)과 Orders 테이블(408) 간에 테이블 관계(Orders.CustomerID=Customers.CustomerID)를 정의한다. Order 테이블(408)은 Customers 테이블(404)을 가지고 주문 정보를 유일하게 식별하는 OrderID와 연관되고, Customers 테이블(404)에 매핑되는 CustomerID 컬럼을 적어도 포함한다. 테이블(408)은 배송 정보, 세율(tax rate) 및 배송료(freight charges)와 같은 고객 주문과 관련된 정보의 다른 컬럼을 포함한다.

[0096] 세 번째 조인 필터(410)는 Orders 테이블(408)과 Orders Detail 테이블(412) 간에 테이블 관계(OrderDetails.OrderID=Orders.OrderID)를 정의한다. Order Details 테이블(412)은 Orders 테이블(408)을 가지고 주문 상세 정보(order details)를 유일하게 식별하는 OrderDetailID와 연관되고, Orders 테이블(408)에 매핑되는 OrderID 컬럼을 적어도 포함한다. 테이블(412)은 Order Details 테이블(412)을 Products 테이블(416)에 매핑시키는 ProductID 컬럼을 포함한다. 또한, 테이블(412)은 주문된 제품에 대한 정보를 제공하는 Quantity 컬럼 및 UnitPrice를 포함한다. 그러므로, Order Details 테이블(412)은 Products 테이블(416)로부터 UnitPrice 정보를 요구한다.

[0097] 네 번째 조인 필터(416)는 Order Details 테이블(412)과 Products 테이블(416) 간에 테이블 관계(OrderDetails.ProductID=Products.ProductID)를 정의한다. Products 테이블(416)은 유일한 ProductID와 연관되고, ProductName 및 UnitPrice를 위한 컬럼을 더 포함한다.

[0098] 그러므로, 조인 필터와 함께 확장 알고리즘(expansion algorithm)은 EmployeeID와 함께 고객이 구매한 제품, 구매한 제품의 주문 상세 정보, 상세 정보 와 특정 제품을 포함하는 주문, 및 제품을 주문한 고객과 관련된 모든 정보의 전파를 용이하게 한다.

[0099] 도 5에서 기술된 샘플 데이터를 사용하여, Customers 테이블(404) 상의 이러한 파티션 업데이트는 Customers 행에 대한 과거 및 현재 매핑이 재계산되도록 한다. 그 후, Customers 테이블(404) 및 Order 테이블(408) 간의 조인 필터(406)에 따라, 변화한 부모 파티션 멤버쉽을 가지는 모든 Order 행은 각각의 과거 및 현재 파티션 매핑을 재평가할 것이다. Orders 테이블(408) 및 Order Details 테이블(412) 간의 조인 필터(410)에 따라, 부모 파티션 멤버쉽이 변화한 모든 Order Details 행은 각각의 과거 및 현재 파티션 매핑을 재평가할 것이다. 최종

적으로, Order Details 테이블(412) 및 Products 테이블(416) 간의 조인 필터(414)에 따라, 변화한 부모 파티션 멤버쉽을 가지는 모든 Products 행은 각각의 과거 및 현재 파티션 매핑이 재평가될 것이다. 그 후, Produts 테이블(416)이 어떠한 자식 행을 가지지 않으므로, 위의 알고리즘은 종료된다. 확장 알고리즘이 종료된 경우, 메타데이터 테이블은 행의 파티션 멤버쉽을 정확히 반영한다. 소스 레플리카로부터 목적지 레플리카로 행이 전파되기 전에 추가적인 계산이 요구되지 않는다.

[0100] 이를 반영하여, 자식 테이블에 대하여 데이터베이스 뷰(database view)가 생성되고, 그들의 직접적인 부모에 대하여 참조 뷰(reference view)가 생성된다. 샘플 스키마에서, Customers에 대한 뷰는 Employees에 대한 뷰를 참조한다. 마찬가지로, Orders에 대한 뷰는 Customers에 대한 뷰를 참조한다. 마찬가지로, Order Details에 대한 뷰는 Orders에 대한 뷰를 참조한다. 본 예에서는 Employee 테이블(400)인 최상위 부모에 대한 뷰는 행의 파티션 멤버쉽을 평가하기 위하여 PartitionsMetadata 테이블(302)을 사용하고, 샘플에서는 필터링 컬럼(TerritoryID)의 값을 추가로 사용한다.

[0101] 설명을 위해, 다음의 뷰 정의가 샘플 스키마에 대하여 사용된다.

[0102] Employees 테이블을 위한 뷰 정의(view_partition_Employees)

[0103] select[Employees].*,partition_id=[PartitionMetadata].[partition_id] from Employees, PartitionsMetadata where PartitionsMetadata.fn_EmployeeTerritory=Emplyees.TerritoryID

[0104] Customers 테이블을 위한 뷰 정의(view_partition_Customers)

[0105] select[Customers].*,partition_id=[Employees].[partition_id] from Customers, [view_partition_Employees] Employees where Customers.EmployeeID=Employees.EmployeeID

[0106] Orders 테이블을 위한 뷰 정의(view_partition_Orders)

[0107] select[Orders].*,partition_id=[Customers].partition_id from [Orders], [view_partition_Customers] [Customers] where (Orders.CustomerID=Customers.CustomerID)

[0108] Order Details 테이블을 위한 뷰 정의(view_partition_OrderDetails)

[0109] select[Order_Details].*,partition_id=[Orders].partition_id from [Order_Details], [view_partition_Orders][Orders] where (Order_Details.OrderID=Orders.OrderID)

[0110] 자식 행의 파티션 멤버쉽은 변화된 행의 세트에 대한 뷰를 통해 행의 변화된 세트를 선택함으로서 결정된다. "이전" 값으로부터 계산된 모든 과거 파티션 매핑은 PastChangesPartitionMapping 테이블(306)에 지속되고, 모든 현재 파티션 매핑은 CurrentChangesPartitionMapping 테이블(304)에 지속된다.

[0111] 다음 표 3은 동기화 업데이트의 일부로서 종업원 Joe가 CA 구역에서 WA 구역으로 이동하는 변화를 반영한 샘플 CurrentChangesPartitionMapping 테이블(304)이다. 마지막 컬럼은 해석을 위한 것으로, 스키마의 필수적인 부분이 아니다.

표 3

[0112] 샘플 CurrentChangesPartitionMapping 테이블

row_id	partition_id	테이블의 해석(스키마의 일부 아님)
R1	1	R1 행은 Territory = “WA” 인 Employee "Joe"에 대응된다.
R2	2	R2 행은 Territory = “CA” 인 Employee "Mary"에 대응된다.
R3	3	R3 행은 Territory = “OR” 인 Employee "Jane"에 대응된다.
...	...	...

[0113] 다음은 표 4로서, Joe가 이전에 CA 구역에 할당되었음을 나타내는 샘플 PastChangesPartitionMapping 테이블(306)이다. 마지막 컬럼은 샘플 데이터의 해석을 제공하지만, 스키마의 필수적인 부분이 아니다.

표 4

[0114] 샘플 PastChangesPartitionMapping 테이블

Row_id	Partition_id	동기화 앵커	테이블의 해석(스키마의 일부 아님)
R1	2	2002년 10월 8일 화요일 오전 10시 27분	R1행은 2002년 10월 8일 화요일 오전 10시 27분까지 구역이 "CA"이었던 종업원 "Joe"에 대응된다.
R2	3	2002년 10월 7일 월요일 오전 12시 25분	R2행은 2002년 10월 7일 화요일 오전 12시 25분까지 구역이 "OR"이었던 종업원 “Mary"에 대응된다.
...	...	...	...

[0115] 도 5를 참조하면, 고객 데이터가 한 명의 종업원으로부터 다른 사람에게 재할당되는 예시적인 파티션 업데이트가 도시되어 있다. 현재 도시된 테이블 관계는 EmployeeID "Joe"에 대한 것이다. Employee1 테이블(500)은 "Joe"를 나타내는 EmployeeID 컬럼을 가지는 Employee1 행 및 Employee1 테이블(500)을 Employees PartitionsMetadata 테이블(400)로 역으로 링크시키는 엔트리 "WA"를 가지는 TerritoryID 컬럼을 기술한다. Employee1 테이블(500)에 대해 자식으로서, Customer1 테이블(502)은 "Alfred"의 CustomerID 컬럼 엔트리를 가지는 Customer1 행 및 Customer1 테이블(502)을 Employee1 테이블(500)로 역으로 링크시키는 "Joe"라는 EmployeeID 엔트리를 기술한다.

[0116] Customer1 테이블(502)은 첫 번째 Order_1 엔티티(504), 두 번째 Order_2 엔티티(506) 및 세 번째 Order_3 엔티티(508)의 세 개의 자식 엔티티를 가진다. Order_1 엔티티(504)는 "1"의 엔트리를 가지는 OrderID 컬럼 및 엔티티(504)를 부모 Customer1 테이블(502)로 역으로 링크시키는 "Alfred"의 엔트리를 가지는 CustomID 컬럼을 포함하는 것으로 Order_1 행을 기술한다. Order_2 엔티티(506)는 "2"의 엔트리를 가지는 OrderID 컬럼 및 엔티티(506)를 부모 Customer1 테이블(502)로 역으로 링크시키는 "Alfred"의 엔트리를 가지는 CustomID 컬럼을 포함하는 것으로 Order_2 행을 기술한다. Order_3 엔티티(508)는 "3"의 엔트리를 가지는 OrderID 컬럼 및 엔티티(508)를 부모 Customer1 테이블(502)로 역으로 링크시키는 "Alfred"의 엔트리를 가지는 CustomID 컬럼을 포함하는 것으로 Order_3 행을 기술한다.

[0117] 첫 번째 Order_1 엔티티(504)는 특정한 OrderID=1에 대하여 주문 상세 정보를 정의하는 네 개의 자식 OrderDetail 엔티티를 가진다. OrderDetail1 엔티티(510)는 엔티티(510)를 부모 엔티티(504)로 역으로 링크시키는 "1"의 엔트리를 가지는 OrderID 컬럼 및 도시되지 않은 다른 상세 정보 컬럼을 포함하는 것으로 OrderDetail1 행을 기술한다. OrderDetail2 엔티티(512)는 엔티티(512)를 부모 엔티티(504)로 역으로 링크시키는 "1"의 엔트리를 가지는 OrderID 컬럼 및 도시되지 않은 다른 상세 정보 컬럼을 포함하는 것으로 OrderDetail2 행을 기술한다. OrderDetail3 엔티티(514)는 엔티티(514)를 부모 엔티티(504)로 역으로 링크시키는 "1"의 엔트리를 가지는 OrderID 컬럼 및 도시되지 않은 다른 상세 정보 컬럼을 포함하는 것으로 OrderDetail3 행을 기술한다. OrderDetail4 엔티티(516)는 엔티티(516)를 부모 엔티티(504)로 역으로 링크시키는 "1"의 엔트리를 가지는 OrderID 컬럼 및 도시되지 않은 다른 상세 정보 컬럼을 포함하는 것으로 OrderDetail4 행을 기술한다.

[0118] Order_2 엔티티(506)는 OrderDetail5 엔티티(518)라는 하나의 자식 엔티티를 가지는데, 이는 엔티티(518)를 부모 엔티티(506)로 역으로 링크시키는 "2"의 엔트리를 가지는 OrderID 컬럼 및 도시되지 않은 다른 상세 정보 컬럼을 포함하는 것으로 OrderDetail5 행을 기술한다.

[0119] Order_3 엔티티(508)는 OrderDetail6 엔티티(520) 및 OrderDetail7 엔티티(522)라는 두 개의 자식 엔티티를 가진다. OrderDetail6 엔티티(520)는 엔티티(520)를 부모 엔티티(508)로 역으로 링크시키는 "3"의 엔트리를 가지는 OrderID 컬럼 및 도시되지 않은 다른 상세 정보 컬럼을 포함하는 것으로 OrderDetail6 행을 기술한다. OrderDetail7 엔티티(522)는 엔티티(522)를 부모 엔티티(508)로 역으로 링크시키는 "3"의 엔트리를 가지는 OrderID 컬럼 및 도시되지 않은 다른 상세 정보 컬럼을 포함하는 것으로 OrderDetail7 행을 기술한다.

[0120] Customer1 테이블(502) 내의 Customer1 행의 EmployeeID 컬럼은 "Joe"로부터 "Mary"로 업데이트 된다. Employee2 테이블(524)은 "Mary" 엔트리를 가지는 컬럼 및 "CA" 엔트리를 가지는 TerritoryID 컬럼을 포함하는 것으로 Employee2 행을 기술한다. 이 업데이트는 본질적으로 Customer_1 행 파티션 멤버쉽을 "Joe"로부터

"Mary"로 변화시킨다. 그러므로, Orders 행(Order_1, Order_2 및 Order_3)에 대한 파티션 멤버쉽 또한 "Joe"로부터 "Mary"로 변화할 것이다. 마찬가지로, OrderDetails 엔티티(510, 512, 514, 516, 518, 520 및 522) 내의 대응되는 Order Details 행은 이제 다른 파티션 즉 "Mary"에 속한다. 행의 파티션 멤버쉽을 부모 엔티티로부터 자식 엔티티들로 전파하기 위하여, 확장 프로세스가 수행된다.

[0121] 복제 프로세싱-변화 추적 및 변화 열거(change enumeration)

[0122] 영향을 받는 복제 프로세싱의 두 개의 중요한 단계는 변화 추적 및 변화 열거를 포함한다. 사용자 데이터베이스에서 변화가 발생한 경우, 사용자 데이터베이스에서의 변화 추적 메커니즘이 복제 메타데이터를 추적하여, 이들 변화가 추후 시점에 다른 레플리카와 동기화 되도록 한다. 변화 열거는 소스와 목적지 레플리카 간의 이전의 동기화 이후로 이 레플리카 내에서 발생했던 변화가 열거되는 동기화 프로세스의 한 단계이다.

[0123] 복제 프로세싱의 변화 조정(change reconciliation) 및 변화 응용(change application) 단계는 어떤 중대한 방법으로도 영향을 받지 않는다.

[0124] 행이 삽입, 업데이트 또는 삭제되는 경우 복제 메타데이터에 대한 변경에 대하여 변화 추적 메커니즘이 기술될 것이다. 행이 삽입된 경우, 그 행에 대한 현재 파티션 멤버쉽이 평가되고, 이 정보는 CurrentChangesPartitionMapping 테이블(304)에 지속된다. 주어진 행의 자식 행이 이미 존재하는 약간 드문 경우에 있어서, 새로운 행의 삽입은 자식 행의 파티션 멤버쉽이 확장 프로세스를 사용하여 평가되도록 한다.

[0125] 행이 비필터 컬럼 업데이트(non-filter column update)를 이용하여 업데이트되는 경우, 메타데이터 테이블에 주어진 행에 대한 메타데이터가 존재하지 않는다면, 그 행에 대한 현재 파티션 멤버쉽이 평가되고, 이 정보는 CurrentChangesPartitionMapping 테이블(304)에 지속된다. 메타데이터가 이미 존재한다면, 파티션 멤버쉽은 평가되지 않는데, 이는 파티션 멤버쉽이 이미 이전에 평가되었음을 나타내기 때문이다.

[0126] 행이 필터링된 컬럼 업데이트(filtered column update)를 이용하여 업데이트되는 경우, 업데이트 전에 변화 추적동안 이용 가능한 "이전" 값을 사용하여 그 행이 과거에 속했던 파티션이 평가된다. 자식 행의 파티션 멤버쉽은 부모 해의 파티션 멤버쉽에 의하여 영향을 받으므로, 자식 행에 대한 과거 파티션 매핑을 평가하기 위하여 확장 프로세스가 사용된다. 모든 과거 파티션 매핑은 PastChangesPartitionMapping 테이블(306)에 저장된다. 부가적으로, 파티션 업데이트가 발생한 논리적인 시간이 과거 변화 테이블(306)의 동기화 앵커 컬럼에 기록된다. 이로 인해 동기화 프로세스는 이전의 동기화 세션 동안 또는 다른 소스 레플리카와의 동기화를 통해 이러한 변화를 이미 거친 레플리카로의 파티션 업데이트의 전파를 약화시킬 수 있다. 과거의 파티션 매핑을 평가하는 것은 본질적으로 CurrentChangesPartitionMapping 테이블(304) 내의 임의의 엔트리를 정리(clean up)한다는 점에 주의해야 하는데, 이는 그러한 엔트리들이 파티션 업데이트로 인해 무효화되기 때문이다.

[0127] 행이 필터링된 컬럼 업데이트를 이용하여 업데이트되는 경우, 업데이트 후에 변화 추적 동안 이용 가능한 "이후" 값을 사용하여 그 행이 현재 속하는 파티션이 평가된다. 자식 행의 파티션 멤버쉽은 부모 행의 파티션 멤버쉽에 의해 영향을 받으므로, 자식 행에 대한 현재 파티션 매핑을 평가하기 위하여 확장 프로세스가 사용된다. 모든 현재 파티션 매핑은 CurrentChangesPartitionMapping 테이블(304)에 저장된다.

[0128] 삭제를 처리하는 경우, 삭제 전에 그 행이 이전에 속했던 파티션이 변화 추적 동안 이용 가능한 "이전" 값을 사용하여 평가된다. 자식 행의 파티션 멤버쉽은 부모 행의 파티션 멤버쉽에 의해 영향을 받으므로, 자식 행의 과거 파티션 매핑을 평가하기 위하여 확장 프로세스가 사용된다. 모든 과거 파티션 매핑은 PastChangesPartitionMapping 테이블(306)에 저장된다. 부가적으로, 파티션 업데이트가 발생하는 논리적인 시간은 테이블(306)의 동기화 앵커 컬럼에 기록된다. 이는 동기화 프로세스가 이전의 동기화 세션 동안 이 변화를 이미 경험한 레플리카로 파티션 업데이트를 중복적으로 전파하는 것을 방지한다. 행 및 그의 자식은 삭제 후에 파티션에 존재하지 않으므로, CurrentChangesPartitionMapping 테이블(304)에 어떤한 엔트리도 생성할 필요가 없다. 과거 파티션 매핑은 본질적으로 CurrentChangesPartitionMapping 테이블(304)내의 임의의 엔트리들을 정리함에 주의해야 하는데, 이는 그러한 엔트리들은 행 삭제 프로세스에 의하여 무효화되기 때문이다.

[0129] 파티션의 업데이트는 업데이트가 효율적인 방법으로 목적지 레플리카로 전파되도록 메타데이터를 업데이트하기 위한 변화 추적 메커니즘을 요구한다. 주요 태양은, 행의 파티션 멤버쉽이 변화한 경우, 행의 partition_id가 업데이트되어야 하고, 부모 행이 업데이트됨으로써 변화된 파티션 멤버쉽을 가지는 임의의 자식 행들은 자신들의 partition_id를 재평가해야 한다는 점이다. 이 정보는 확장 프로세스를 사용하여 획득되고, 이는 이하 제공

되는 예에서 이용된다.

[0130] 변화 열거 메커니즘은 소스 레플리카로부터 목적지 레플리카로 변화를 효율적으로 전파하기 위하여 파티션 멤버쉽 메타데이터를 사용한다. 변화 추적 메커니즘이 이미 파티션을 평가하고, 확장을 수행했으므로, 런타임에 변화 열거 복잡성(change enumeration complexity)은 크게 단순화된다.

[0131] 삭제 및 파티션 업데이트는 "과거 파티션 매핑 엔트리"를 가지는 변화의 세트에 기여한다. 이러한 변화의 세트는 동기화 앵커가 그 세션에 대해 결정된 동기화 앵커보다 더 최근이고, partition_id가 목적지 레플리카의 partition_id와 부합되는 행들을 PastChangesPartitionMapping 테이블(306)로부터 선택함으로써 열거된다. 이러한 행들은 삭제로써 목적지 레플리카로 전파될 것이다.

[0132] 파티션된 그리고 파티션되지 않은 삽입 및 업데이트는"현재 파티션 매핑 엔트리"를 가지는 변화의 세트에 기여한다. 이러한 변화의 세트는 동기화 앵커가 그 세션에 대해 결정된 동기화 앵커보다 더 최근이고, 목적지 레플리카의 partition_id와 부합하는 partition_id를 가지는 엔트리를 CurrentChangesPartitionMapping 테이블(304)에 가지는 그러한 행들을 RowMetadata 테이블(312)로부터 선택함으로써 열거된다. 이러한 행들은 업데이트로써 목적지 레플리카로 전파된다.

[0133] 주어진 파티션에 대한 변화가 열거되면, 변화는 현존하는 기술을 사용하여 충돌 감지 및 해결 메커니즘을 통해 전파된다. 마찬가지로, 주어진 파티션에 대한 변화가 열거되고, 충돌이 삭제 및 해결되면, 변화는 현존하는 기술을 사용하여 변화 응용을 통해 전파된다.

[0134] 세대 파티션화(generation partitioning)

[0135] 소스 레플리카에서 유지되는 바와 같이, 파티션에 대한 행의 매핑은 관련된 세대 리스트의 효율적인 계산이 소스 및 목적지 레플리카 간에 전파되는 것을 허용한다. 예컨대, GenerationPartitionMapping 테이블(308)은 generation_id 컬럼을 포함하는데, 이는 세대의 그룹에 할당된 동기화 앵커이다. 변화가 목적지 레플리카로부터 소스 레플리카로 전파되는 경우, 변화는 소스 레플리카에서 새로운 generation_id를 할당받는다. 이 세대의 일부인 행은 개별적인 파티션 P1에 속하므로, 그 세대는 목적지의 partitino_id P1에 매핑된다. 다른 개별 파티션 P2에 속하는 상이한 목적지 레플리카가 소스 레플리카와 동기화하는 경우, 세대 매핑은 P1 세대를 제거한다. partition_id에 대한 -1의 특별한 값은 세대가 전역이고, 따라서 모든 파티션에 관련됨을 나타낸다.

[0136] 다음은 표 5로서, 샘플 GenerationPartitionMapping 테이블(308)이다. 마지막 컬럼은 샘플 데이터의 해석을 제공하고, 스키마의 필수적인 일부가 아니다.

표 5

[0137] 샘플 GenerationPartitionMapping 테이블

generation_id	partition_id	테이블의 해석(스키마의 일부 아님)
G1	1	G1 세대는 Territory = “WA” 인 파티션에 대응된다
G2	2	G2 세대는 Territory = “CA” 인 파티션에 대응된다
G3	-1	G3 세대는 전역 세대이고, 모든 파티션으로 전파될 필요가 있다
...	...	...

[0138] 세대 파티션화는 본 발명의 파티션 그룹화 태양을 기반으로 확립된 최적화로 다음의 세가지 이점(무관한 변화의 효율적인 제거, 많은 목적지 레플리카가 소스 레플리카와 동기화하는 경우에도 동기화 세션의 예측 가능한 지속 기간 및 세대의 전파로 강화된 네트워크 성능)을 제공한다.

[0139] 소스 레플리카가 많은 목적지 레플리카와 각각의 개별 파티션을 이용하여 동기화하는 토폴로지에 있어, 세대 파티션화로 인해 동기화 프로세스가 효과적으로 무관한 변화를 제거할 수 있다. 예컨대, 목적지 레플리카로 데이터의 개별 부집합을 제공하는 하나의 소스 레플리카가 존재하는 토폴로지를 고려하는데, 목적지 레플리카 각각

은 한 명의 판매사원에 특정한 판매 정보를 포함한다. 처음에 1000 개의 레플리카가 10,000 개의 변화를 소스 레플리카로 전파하고, 10,000 개의 변화는 소스 레플리카에서 100 개의 개별 세대로 위치한다고 가정하면, 모든 10,000 개의 목적지 레플리카가 그들의 변화를 소스 레플리카로 전파하는 경우, 소스 레플리카에서 10만 세대로 집중되는 1000만 개의 변화가 존재한다. 결과적으로, GenerationPartitionMapping 테이블(308)은 100개의 세대의 세트를 가지는 10만 개의 엔트리를 포함하게 될 것이고, 이들 각각은 하나의 파티션에 매핑된다.

[0140] 개별 partitino_id=P1001을 가지는 첫 번째 목적지 레플리카가 소스 레플리카와 동기화하는 경우, 모든 세대가 파티션(P1 내지 P1000)에 대응되는 파티션으로 매핑되므로, 10만 개의 모든 세대(그러므로 1000만 개의 변화)는 partition_id=P1001을 가지는 첫 번째 목적지 레프리카와 무관할 것이다. 반면, partition_id=P500을 가지는 두 번째 목적지 레플리카가 소스 레플리카와 동기화한다면, 동기화는 (partition_id=P500을 가지는)두 번째 목적지 레플리카와 관련된 정확히 100 개의 세대를 열거하고, 따라서 두 번째 목적지 레플리카로 정확히 만 개의 변화를 전파할 것이다.

[0141] 세대 파티션화 최적화(generation partitioning optimization)의 다른 이점은 파티션과 관련된 변화의 수에 비례하여 목적지 레플리카에 의해 조사될 동기화 메타데이터의 양을 제어하므로, 소스 레플리카와 목적지 레플리카 간의 동기화 세션의 지속 기간이 예측 가능하다는 점이다. 예컨대, partition_id=P500을 가지는 목적지 레플리카가 오랜 기간 동안 소스 레플리카와 동기화되지 않았다고 가정한다. 병합 복제의 현재 버전에서는, 이 목적지 레플리카는 소스 레플리카에 축적된 모든 세대를 열거해야 하므로, 소스 레플리카와 마지막으로 동기화하는 경우 불리해진다. 즉, 예컨대 1000 개의 목적지 레플리카가 각각 100 개의 세대를 전파한다면, 관련된 그리고 무관한 세대의 세트는 총 10만 세트가 되는데, 이는 상당한 것이다. 10만 개의 세대를 고려하면, "동기화 세션당 파티션 계산(per synchronization session partition computation)" 알고리즘은 무관한 변화를 제거하고, 오직 관련된 변화만이 목적지 레플리카로 전파될 것이다.

[0142] 세대가 관련된 파티션 식별자로 파티션되는 시나리오를 고려한다. 본 발명의 태양에 따르면, 목적지 레플리카(예컨대 partition_id=P500)와의 동기화 세션은 불리해지지 않는데, 이는 관련된 세대를 즉시 식별할 수 있기 때문이다. 목적지 레플리카로의 무관한 세대 전파가 회피되므로, 세대 파티션화의 네트워크 특징에 대한 영향은 상당하다. 위의 예에 있어서, 이는 10만 개에서 100개를 제외한 즉 99,900 개의 세대(목적지 레플리카로의 전파가 필요하지 않은 세대)의 절약에 이르게 된다. 소스 레플리카에서 로컬 변화가 생길 때마다, 그 변화가 오직 하나의 파티션 또는 파티션의 세트들에 관련되었더라도, 이러한 로컬 변화를 전역 세대로 전파시키는 것이 더욱 바람직하고 효율적일 것이다. 이러한 사실이 기초하고 있는 전제는 소스 레플리카에서 생긴 변화는 가상적으로 모든 목적지 레플리카에 관련되어있다는 것으로, 반면 목적지 레플리카에서 생긴 변화는 동일한 partition_id를 공유하는 다른 목적지 레플리카에만 관련된다.

[0143] 본 발명의 태양은 다른 응용예를 가지고 있음을 인식해야 한다. 예컨대, 다수의 클라이언트 사용자에 의해 액세스될 수 있는 서버 팜(server farm)에서, 첫 번째 서버의 특정 사용에 대하여 발생한 파티션 변화는 사용자가 첫 번째 서버와 연관을 끊고, 두 번재 서버와 재연관되어야 하는지를 결정하기 위하여 이용될 수 있다. 데이터베이스 정보의 특정 필드에 생겨난 변화가 추적되면, 연관된 프로세스가 자동적으로 시작되는 것처럼, 사전에 결정된 기준에 따라 이러한 "연관(associating)" 프로세스가 자동적으로 수행될 수 있다.

[0144] 발명이 데이터베이스 변화와 연관되어 기술되었지만, 이에 제한되지 않고, 이종 소스에 걸쳐 데이터 동기화가 요구되는 임의의 환경에 응용될 수 있음을 인식해야 한다. 예컨대, 본 발명은 이름, 프로필 정보, 사용자 계정, 네트워크 사용 권한(network permission) 및 모든 사용자와 네트워크 상의 자원의 장치 주소가 조화될 수 있는 디렉토리 서비스(directory service)에 응용될 수 있다.

[0145] 도 6을 참조하면, 개시된 아키텍처를 실행하기 위해 동작 가능한 컴퓨터의 블록도가 도시되어 있다. 본 발명의 다양한 태양에 대한 부가적인 상황을 제공하기 위하여, 도 6 및 다음의 논의는 본 발명의 다양한 태양이 구현될 수 있는 적절한 컴퓨팅 환경(600)의 간략하고, 일반적인 설명을 제공함을 목적으로 한다. 발명이 하나 이상의 컴퓨터 상에서 실행될 수 있는 컴퓨터 실행 가능 명령어의 일반적인 상황에 대해서 기술되어 있지만, 당업자는 발명이 다른 프로그램 모듈과 조합하여 및/또는 하드웨어와 소프트웨어의 조합으로 구현될 수 있음을 인식할 것이다. 일반적으로, 프로그램 모듈은 특정한 업무를 수행하거나, 특별한 추상 데이터 차입을 구현하는 루틴(routines), 프로그램, 컴포넌트, 데이터 구조 등을 포함한다. 추가적으로, 당업자는 발명의 방법이 각각의 장치가 하나 이상의 연관된 장치와 효과적으로 결합될 수 있는 개인용 컴퓨터, 핸드 헬드 컴퓨팅 장치(hand-held computing device), 마이크로프로세서 기반 또는 프로그램 가능 소비자 전자 제품뿐만 아니라 단일 프로세서 또는 멀티프로세서 컴퓨터 시스템, 미니컴퓨터, 메임프레임 컴퓨터 등과 같은 다른 컴퓨터 시스템 설정과 함께 실

시될 수 있음을 인식할 것이다. 또한, 발명의 도시된 태양은 통신 네트워크를 통해 연결된 원격 프로세싱 장치에 의해 특정 업무가 수행되는 분산 컴퓨팅 환경에서 실시될 수 있다. 분산 컴퓨팅 환경에서, 프로그램 모듈은 지역 및 원격 메모리 저장 장치 모두에 위치할 수 있다.

[0146] 도 6을 참조하면, 본 발명의 다양한 태양을 구현하기 위한 예시적인 환경(600)은 컴퓨터(602)를 포함하는데, 컴퓨터(602)는 프로세싱 유닛(604), 시스템 메모리(606) 및 시스템 버스(608)를 포함한다. 시스템 버스(608)는 시스템 메모리(606)를 포함하는(이에 제한되지 않음) 시스템 컴포넌트를 프로세싱 유닛(604)에 결합시킨다. 프로세싱 유닛(604)은 다양한 상업적으로 이용 가능한 프로세서 가운데 임의의 것이 될 수 있다. 이중 마이크로프로세서(dual microprocessor) 및 다른 멀티 프로세서 아키텍처가 프로세싱 유닛(604)으로서 사용될 수 있다.

[0147] 시스템 버스(608)는 다양한 상업적으로 이용 가능한 버스 아키텍처 중의 임의의 것을 사용하는 메모리 버스 또는 메모리 제어기, 주변 버스(peripheral bus) 및 로컬 버스를 포함하는 버스 구조의 여러 타입 가운데 임의의 것이 될 수 있다. 시스템 메모리(606)는 ROM(610; Read Only Memory) 및 RAM(612; Random Access Memory)을 포함한다. 스타트 업(start-up)동안과 같이 컴퓨터(602) 내에서 요소 간에 정보를 전송하는 것을 돕는 기본적인 루틴을 포함하는 BIOS(Basic Input/Output System)가 ROM(610)에 저장된다.

[0148] 컴퓨터(602)는 하드 디스크 드라이브(614), (예컨대 이동식 디스크(618)로부터 판독하고 기록하기 위하여)자기 디스크 드라이브(616) 및 (예컨대 CD-ROM(622)를 판독하거나 다른 광 매체로부터 판독하거나 기록하기 위하여) 광 디스크 드라이브(620)를 더 포함한다. 하드 디스크 드라이브(614), 자기 디스크 드라이브(616) 및 광 디스크 드라이브(620)는 각각 하드 디스크 드라이브 인터페이스(624), 자기 디스크 드라이브 인터페이스(626) 및 광 드라이브 인터페이스(628)에 의해 시스템 버스(608)에 접속될 수 있다. 드라이브 및 그들의 연관된 컴퓨터 판독 가능 매체는 데이터, 데이터 구조, 컴퓨터 실행 가능 명령어 등의 비휘발성 저장을 제공한다. 컴퓨터(602)에 대하여, 드라이브 및 매체는 적절한 디지털 형태의 브로드캐스트 프로그래밍(broadcast programming)의 저장을 제공한다. 이상의 컴퓨터 판독 가능 매체의 설명은 하드 디스크, 이동식 자기 디스크 및 CD를 언급하고 있지만, 당업자는 집 드라이브(zip drives), 자기 카세트, 플래시 메모리 카드, 디지털 비디오 디스크, 카트리지 등과 같은 컴퓨터 판독 가능한 다른 타입의 매체가 예시적인 운영 환경에서 사용될 수 있고, 그러한 매체는 본 발명의 방법을 수행하기 위한 컴퓨터 실행 가능 명령어를 포함할 수 있음을 인식해야 한다.

[0149] 운영 시스템(630), 하나 이상의 애플리케이션 프로그램(632), 다른 프로그램 모듈(634) 및 프로그램 데이터(636)를 포함하는 다수의 프로그램 모듈이 드라이브 및 RAM(612)에 저장될 수 있다. 본 발명은 다양한 상업적으로 이용 가능한 시스템 또는 운영 시스템의 조합으로 구현될 수 있음을 인식해야 한다.

[0150] 사용자는 마우스(640)와 같은 위치 지정 도구 및 키보드(638)를 통해 컴퓨터(602)에 명령 및 정보를 입력할 수 있다. 다른 입력 장치(도시되지 않음)는 마이크로폰, 적외선 리모콘(IR remote control), 조이스틱, 게임 패드, 위성 접시, 스캐너 등을 포함할 수 있다. 이러한 그리고 다른 입력 장치는 종종 시스템 버스(608)에 결합된 직렬 포트 인터페이스(642)를 통해 프로세싱 유닛(604)에 접속되지만, 병렬 포트, 게임 포트, USB(Universal Serial Bus), 적외선 인터페이스 등과 같은 다른 인터페이스에 의해 접속된다. 또한, 모니터(644) 또는 다른 타입의 디스플레이 장치는 비디오 어댑터(646)와 같은 인터페이스를 통해 시스템 버스(608)에 접속될 수 있다. 모니터(644)에 부가하여, 컴퓨터는 전형적으로 스피커, 프린터 등과 같은 다른 주변 출력 장치(도시되지 않음)를 포함한다.

[0151] 컴퓨터(602)는 원격 컴퓨터(들)(648)와 같은 하나 이상의 원격 컴퓨터에 대한 논리적인 접속을 사용하는 네트워크 환경에서 동작할 수 있다. 원격 컴퓨터(들)(648)는 워크스테이션, 서버 컴퓨터, 라우터, 개인용 컴퓨터, 휴대용 컴퓨터, 마이크로프로세서 기반 오락용 기구, 피어 장치(peer device) 또는 다른 공용 네트워크 노드가 될 수 있고, 간결을 위해 오직 메모리 저장 장치(650)만이 도시되었지만, 전형적으로 컴퓨터(602)에 대해 기술된 많은 또는 모든 요소를 포함할 수 있다. 기술된 논리적인 접속은 LAN(652) 및 WAN(654)를 포함한다. 그러한 네트워크 환경은 사무실, 기업 규모 컴퓨터 네트워크(enterprise-wide computer network), 인트라넷 및 인터넷에서 일반적인 것이다.

[0152] LAN 네트워크 환경에서 사용되는 경우, 컴퓨터(602)는 네트워크 인터페이스 또는 어댑터(656)를 통해 로컬 네트워크(652)에 접속된다. WAN 네트워크 환경에서 사용되는 경우, 컴퓨터(602)는 전형적으로 모뎀(658)을 포함하거나, LAN 상의 통신 서버에 접속되거나, WAN(654)을 통한 통신을 설정하기 위한 인터넷과 같은 다른 수단을 가진다. 내장형 또는 외장형인 모뎀(658)은 직렬 포트 인터페이스(642)를 통해 시스템 버스(608)에 접속된다. 네트워크 환경에서, 컴퓨터(602)에 대하여 묘사된 프로그램 모듈 또는 그의 일부는 원격 메모리 저장 장치(650)에 저장될 수 있다. 도시된 네트워크 접속은 예시적이고, 컴퓨터들 간에 통신 링크를 설정하는 다른 수단이

사용될 수 있음을 인식할 것이다.

[0153] 도 7을 참조하면, 본 발명에 따른 샘플 컴퓨팅 환경(700)의 개략적인 블록도가 도시되어 있다. 시스템(700)은 하나 이상의 클라이언트(들)(702)를 포함한다. 클라이언트(들)(702)는 하드웨어 및/또는 소프트웨어(예컨대 스레드, 프로세스, 컴퓨팅 장치)가 될 수 있다. 예컨대, 클라이언트(들)(702)는 본 발명을 사용함으로써 쿠키(들) 및/또는 연관된 상황 정보를 보관할 수 있다. 또한, 시스템(700)은 하나 이상의 서버(들)(704)를 포함한다. 또한, 서버(들)(704)는 하드웨어 및/또는 소프트웨어(예컨대 스레드, 프로세스, 컴퓨팅 장치)가 될 수 있다. 예컨대, 서버(704)는 본 발명을 사용함으로써 변형을 수행하기 위하여 스레드를 보관할 수 있다. 클라이언트(702) 및 서버(704) 사이에서 하나의 가능한 통신은 둘 이상의 컴퓨터 프로세스 간에 전송되도록 적응된 데이터 패킷의 형태일 수 있다. 예컨대, 데이터 패킷은 쿠키(들) 및/또는 연관된 상황 정보를 보관할 수 있다. 시스템(700)은 클라이언트(들)(702)와 서버(들)(704) 사이의 통신을 용이하게 하기 위하여 사용될 수 있는 통신 프레임워크(706)를 포함한다. 클라이언트(들)(702)는 클라이언트(들)(702)(예컨대 쿠키(들) 및/또는 연관 상황 정보)에 로컬 정보를 저장하기 위하여 사용될 수 있는 하나 이상의 클라이언트 데이터 저장소(들)와 실시 가능하게 접속된다. 마찬가지로, 서버(들)(704)는 서버(704)에 로컬 정보를 저장하기 위하여 사용될 수 있는 하나 이상의 서버 데이터 저장소(들)와 실시 가능하게 접속된다.

[0154] 이상에서 기술된 것은 본 발명의 예를 포함한다. 물론, 본 발명을 기술하기 위하여 컴포넌트 또는 방법론의 모든 상상할 수 있는 조합을 기술하는 것은 불가능하지만, 당업자는 본 발명의 많은 추가적인 조합 및 변형이 가능하다는 것을 인식할 수 있다. 따라서, 본 발명은 첨부된 청구항의 사상 및 범위에 해당하는 모든 변형, 변경 및 변동을 포함하는 것을 의도한다. 추가적으로, 발명의 상세한 설명 및 청구항에서 "포함하다(include)"가 사용되는 범위에 있어서, 그러한 용어는 "포함하다(comprising)"가 청구항에서 번역 용어로 사용되는 경우 해석되는 것과 마찬가지로 "포함하다(comprising)"라는 용어와 유사한 정도로 포괄적인 의도이다.

발명의 효과

[0155] 본 발명은 파티션 계산을 위한 신규한 아키텍처를 제공하고, 많은 수의 동시 다발적인 동기화 세션을 요구하는 배포를 지원하는 병합 복제를 통해 동시 다발적인 동기화 세션의 수가 크게 확장되는 것을 허용한다. 본 발명의 일 태양에 따라서, 목적지의 데이터 파티션이 소스의 파티션과 일치하도록 유지하기 위하여 새로운 알고리즘이 소개된다. 변화가 발생한 경우, 아키텍처가 동작하여 이러한 변화의 파티션 멤버쉽이 단순한 질의 세트를 사용하여 계산된다. 첫 번째 동기화의 파티션 멤버쉽의 사전 계산(pre-computation)은 소스 레플리카와 목적지 레플리카의 추후 동기화 세션 사이에서의 변화의 효율적인 전파를 허용한다.

[0156] 본 발명의 현저한 태양은 소스 레플리카와 목적지 레플리카 간의 모든 동기화 세션이 진행되는 동안 행의 파티션 멤버쉽을 계산하는 대신, 파티션 멤버쉽을 계산하는 부담이 행 변경(row modification)의 실제 시간에 초래된다는 것이다. 파티션 정보가 첫 번째 세션 업데이트 시점에 미리 계산(pre-computed)되고, 추후 동기화 세션 동안 지속되므로, 이러한 접근법은 매 동기화 세션 동안 파티션 계산을 제거하는 것을 허용한다.

도면의 간단한 설명

[0001] 도 1은 본 발명의 시스템 블록도.

[0002] 도 2는 본 발명의 복제 프로세스의 흐름도.

[0003] 도 3은 목적지의 멤버쉽 메타데이터(membership metadata)를 계산하기 위한 계산 알고리즘(computation algorithm)에 의해 이용되는 메타데이터 테이블의 상관 관계(interrelationship)를 도시하는 도면.

[0004] 도 4는 필터링 및 확장(expansion)을 이용하는 샘플 업데이트 스키마(sample update schema)를 도시하는 도면.

[0005] 도 5는 고객 데이터가 한 명의 종업원으로부터 다른 사람에게 재할당되는 파티션 업데이트의 예를 도시하는 도면.

[0006] 도 6은 개시된 아키텍처를 실행하도록 동작하는 컴퓨터의 블록도.

[0007] 도 7은 본 발명에 따른 샘플 컴퓨팅 환경의 개략적인 블록도.

[0008] < 도면의 주요 부분에 대한 부호 설명 >

[0009] 102 : CPU

[0010] 104 : 소스 데이터베이스

[0011] 108 : 데이터베이스

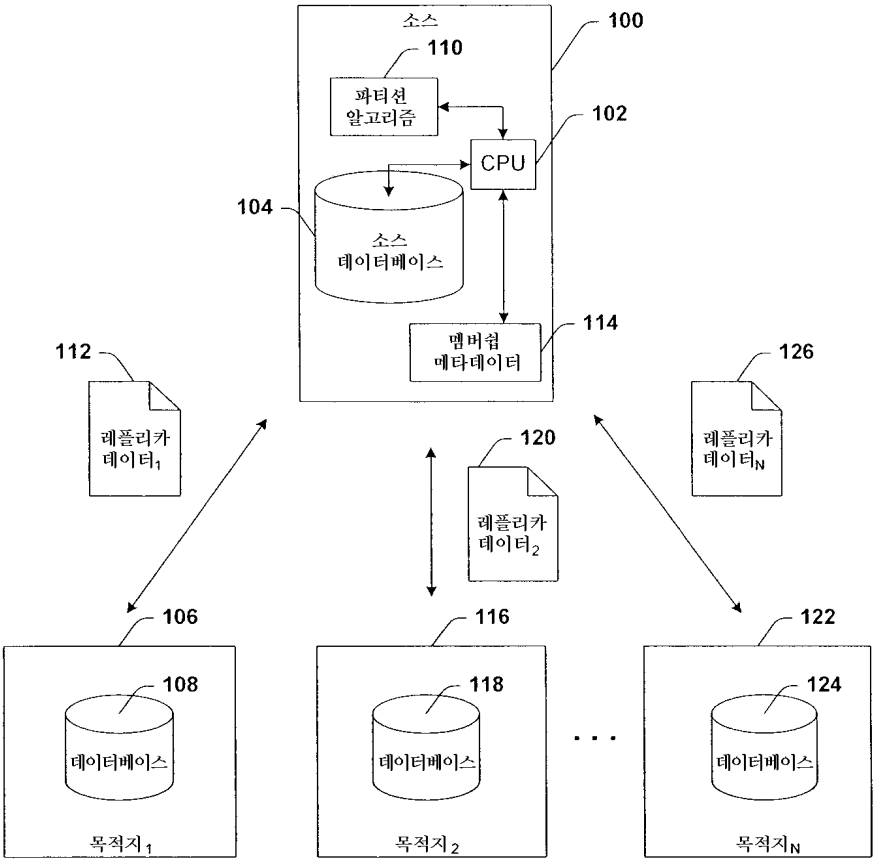
[0012] 110 : 파티션 알고리즘

[0013] 112 : 레플리카 데이터$_1$

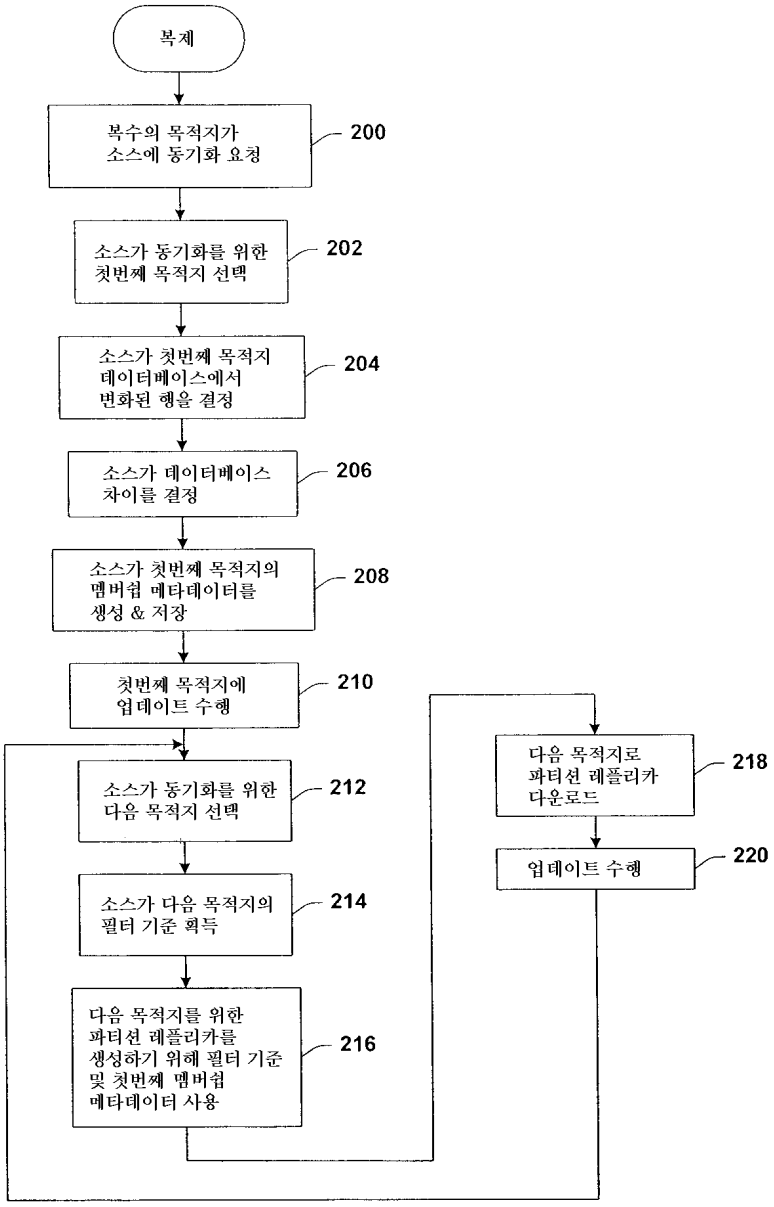
[0014] 114 : 멤버쉽 메타데이터

도면

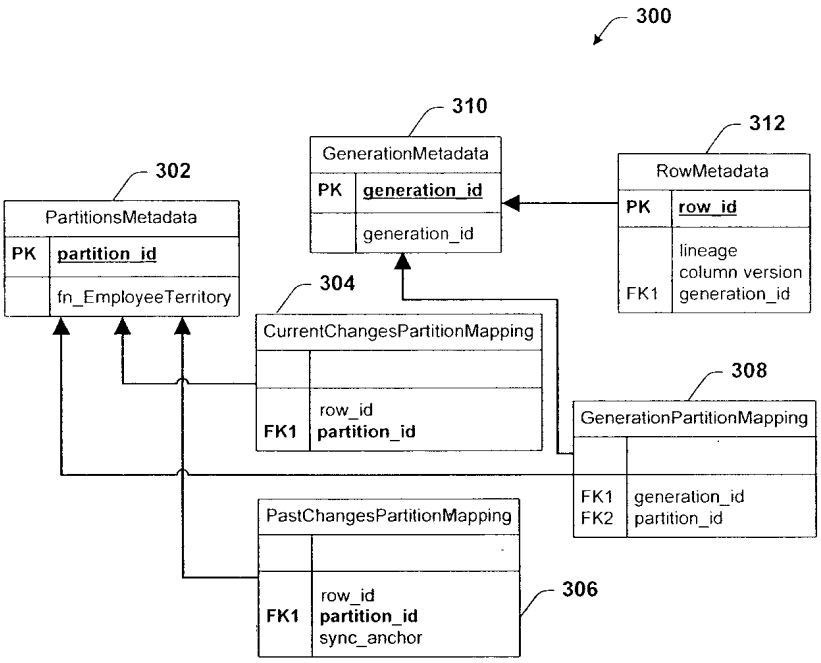
도면1



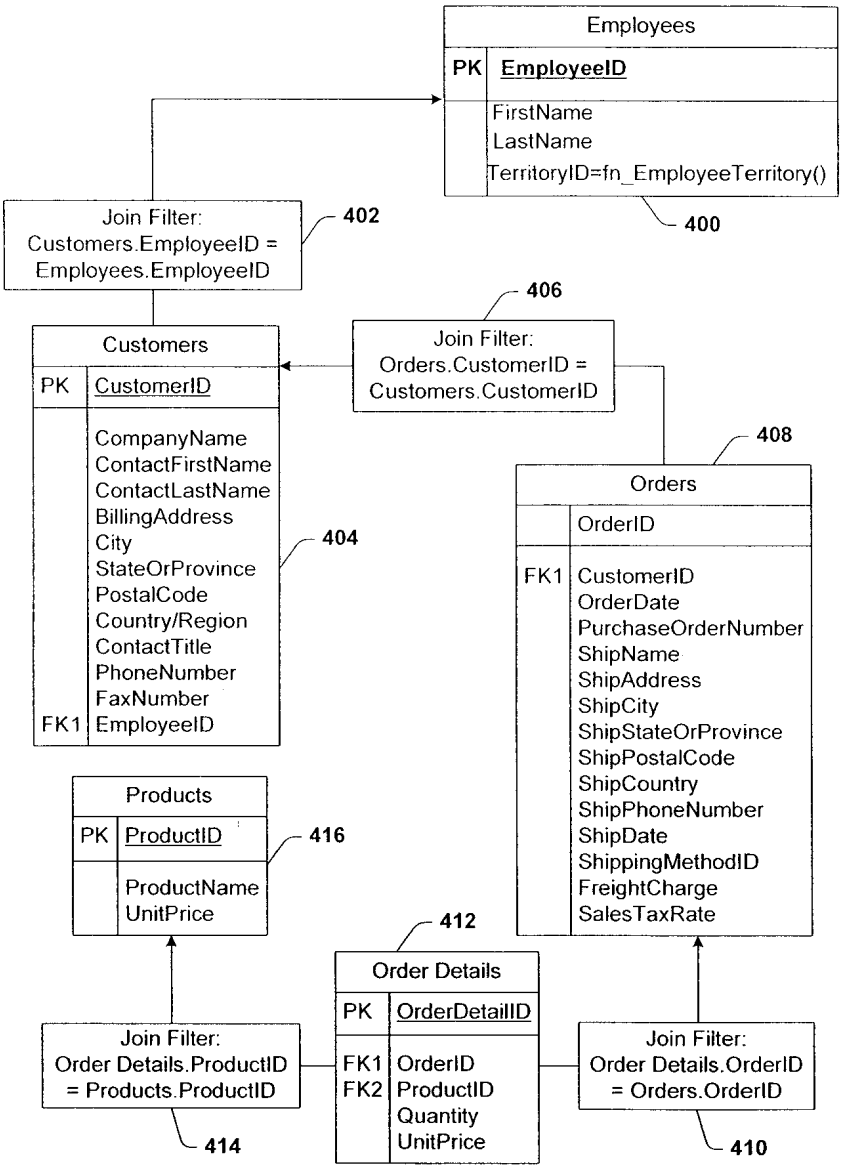
도면2


복제
복수의 목적지가 소스에 동기화 요청
200
소스가 동기화를 위한 첫번째 목적지 선택
202
소스가 첫번째 목적지 데이터베이스에서 변화된 행을 결정
204
소스가 데이터베이스 차이를 결정
206
소스가 첫번째 목적지의 멤버쉽 메타데이터를 생성 & 저장
208
첫번째 목적지에 업데이트 수행
210
소스가 동기화를 위한 다음 목적지 선택
212
소스가 다음 목적지의 필터 기준 획득
214
다음 목적지를 위한 파티션 레플리카를 생성하기 위해 필터 기준 및 첫번째 멤버쉽 메타데이터 사용
216
다음 목적지로 파티션 레플리카 다운로드
218
업데이트 수행
220

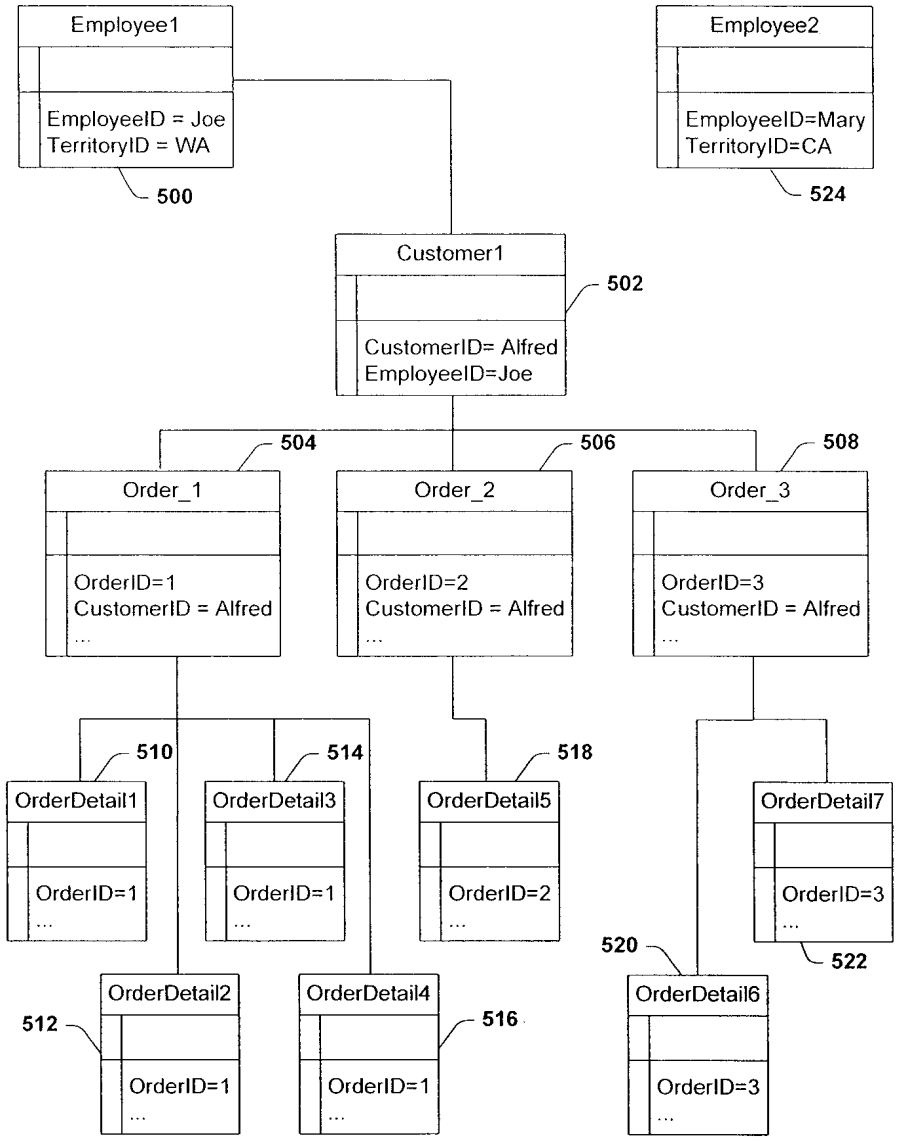
도면3


300
310
GenerationMetadata
PK generation_id
generation_id
312
RowMetadata
PK row_id
lineage
column version
FK1 generation_id
302
PartitionsMetadata
PK partition_id
fn_EmployeeTerritory
304
CurrentChangesPartitionMapping
row_id
FK1 partition_id
308
GenerationPartitionMapping
FK1 generation_id
FK2 partition_id
PastChangesPartitionMapping
row_id
FK1 partition_id
sync_anchor
306

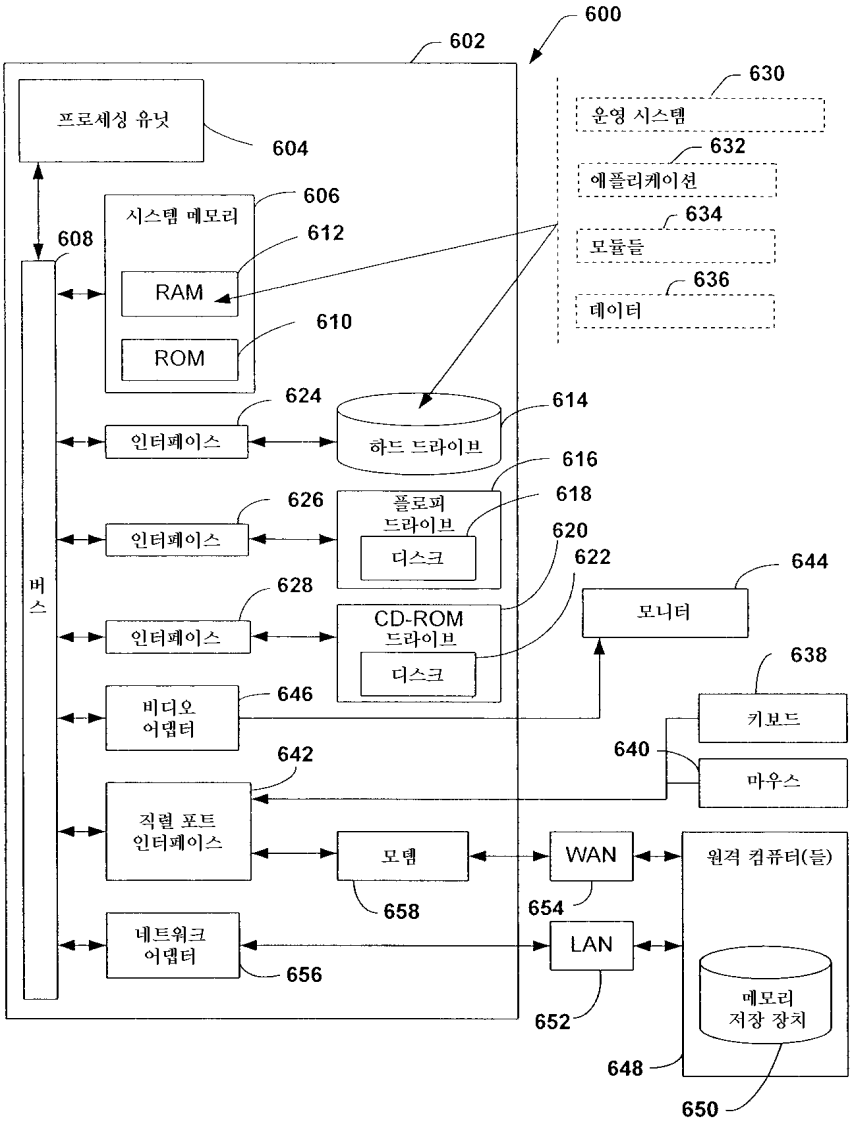
도면4


Employees
PK EmployeeID
FirstName
LastName
TerritoryID=fn_EmployeeTerritory()
400
Join Filter:
Customers.EmployeeID =
Employees.EmployeeID
402
406
Join Filter:
Orders.CustomerID =
Customers.CustomerID
Customers
PK CustomerID
CompanyName
ContactFirstName
ContactLastName
BillingAddress
City
StateOrProvince
PostalCode
Country/Region
ContactTitle
PhoneNumber
FaxNumber
FK1 EmployeeID
404
408
Orders
OrderID
FK1 CustomerID
OrderDate
PurchaseOrderNumber
ShipName
ShipAddress
ShipCity
ShipStateOrProvince
ShipPostalCode
ShipCountry
ShipPhoneNumber
ShipDate
ShippingMethodID
FreightCharge
SalesTaxRate
Products
PK ProductID
ProductName
UnitPrice
416
412
Order Details
PK OrderDetailID
FK1 OrderID
FK2 ProductID
Quantity
UnitPrice
Join Filter:
Order Details.ProductID
= Products.ProductID
414
Join Filter:
Order Details.OrderID
= Orders.OrderID
410

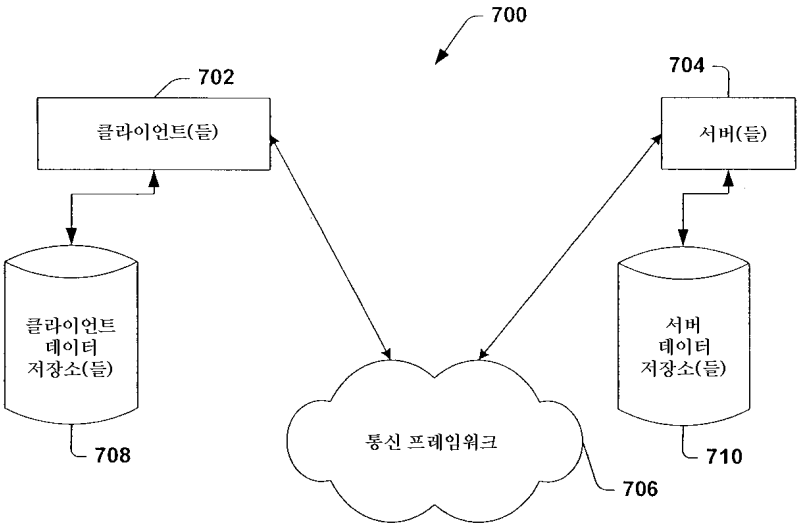
도면5


Employee1
EmployeeID = Joe
TerritoryID = WA
500
Employee2
EmployeeID=Mary
TerritoryID=CA
524
Customer1
CustomerID= Alfred
EmployeeID=Joe
502
504
506
508
Order_1
OrderID=1
CustomerID = Alfred
...
Order_2
OrderID=2
CustomerID = Alfred
...
Order_3
OrderID=3
CustomerID = Alfred
...
510
514
518
OrderDetail1
OrderID=1
...
OrderDetail3
OrderID=1
...
OrderDetail5
OrderID=2
...
OrderDetail7
OrderID=3
...
522
520
512
516
OrderDetail2
OrderID=1
...
OrderDetail4
OrderID=1
...
OrderDetail6
OrderID=3
...

도면6


600
602
630
운영 시스템
632
애플리케이션
634
모듈들
636
데이터
프로세싱 유닛
604
시스템 메모리
606
608
612
RAM
610
ROM
624
인터페이스
614
하드 드라이브
616
626
인터페이스
플로피 드라이브
618
디스크
620
622
644
모니터
628
인터페이스
CD-ROM 드라이브
디스크
638
키보드
646
비디오 어댑터
640
마우스
642
직렬 포트 인터페이스
모뎀
WAN
원격 컴퓨터(들)
658
654
네트워크 어댑터
656
LAN
메모리 저장 장치
652
648
650
버스

도면7


700
702
704
클라이언트(들)
서버(들)
클라이언트
데이터
저장소(들)
서버
데이터
저장소(들)
통신 프레임워크
708
710
706