(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2016년02월25일
(11) 등록번호 10-1597704
(24) 등록일자 2016년02월19일

(51) 국제특허분류(Int. Cl.)
G06F 9/46 (2006.01) *G06F 11/07* (2006.01)
(21) 출원번호 10-2012-7008873
(22) 출원일자(국제) 2010년09월23일
심사청구일자 2015년01월29일
(85) 번역문제출일자 2012년04월05일
(65) 공개번호 10-2012-0076354
(43) 공개일자 2012년07월09일
(86) 국제출원번호 PCT/US2010/049966
(87) 국제공개번호 WO 2011/038096
국제공개일자 2011년03월31일
(30) 우선권주장
12/638,588 2009년12월15일 미국(US)
61/245,862 2009년09월25일 미국(US)
(56) 선행기술조사문헌
JP2006107481 A
US05504900 A
US20050193056 A1
US20050216421 A1

(73) 특허권자
아브 이니티오 테크놀로지 엘엘시
미국 02421 매사추세츠주 렉싱턴 스프링 스트리트 201
(72) 발명자
두로스 브라이언 필
미국 10701 매사추세츠주 프래밍험 레이크뷰 로드 92
애터버리 매튜 달시
미국 02421 매사추세츠주 렉싱턴 제라드 테라스 6
(뒷면에 계속)
(74) 대리인
유미특허법인

전체 청구항 수 : 총 52 항 심사관 : 권현수

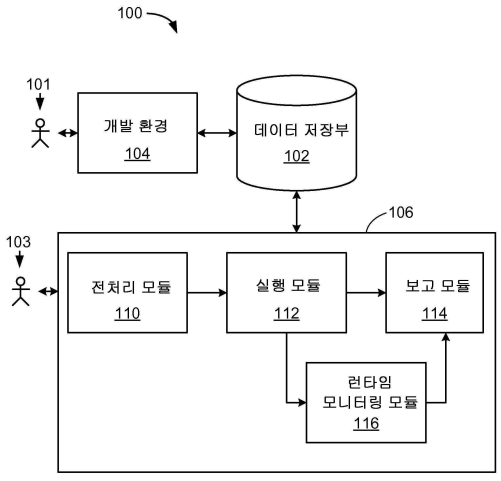
(54) 발명의 명칭 **그래프 기반 어플리케이션에서 트랜잭션 처리 방법**

(57) 요 약

그래프 기반의 연산은 그래프 컴포넌트를 나타내는 복수의 노드를 포함한다. 상기 복수의 노드는 그래프 컴포넌트 사이에서 데이터 흐름을 나타내는 하나 이상의 링크에 의해 연결된다. 상기 연산을 준비하는 단계는: 상기 연산에서 복수의 트랜잭션을 처리하기 위해, 적어도 제1 그래프 컴포넌트 세트를 식별하는 단계 및 제1 트랜잭션

(뒷면에 계속)

대 표 도 - 도1



을 하나 이상의 데이터 레코드 및 이 데이터 레코드에 대응하는 데이터 동작을 포함하는 제1 그룹과 연관짓고, 제2 트랜잭션을 하나 이상의 데이터 레코드 및 이 데이터 레코드에 대응하는 데이터 동작을 포함하는 제2 그룹과 연관짓는 단계를 포함한다. 적어도 상기 제1 및 제2 트랜잭션을 처리하기 위해 그래프 기반의 연산을 실행하는 단계는 상기 제1 그룹의 최종 데이터 레코드에 대응하는 데이터 동작이 상기 제1 그래프 컴포넌트 세트에 의해 실행된 후까지 상기 제2 그룹의 최초 데이터 레코드에 대응하는 데이터 동작이 상기 제1 그래프 컴포넌트 세트에 의해 실행되는 것을 지연시키는 단계를 포함한다.

(72) 발명자

스탠필 크레이그 더블유.

미국 01773 매사추세츠주 린콜린 헉클베리 힐 로드 43

홀리 조셉 스케핑턴 3세

미국 02478 매사추세츠주 벨몬트 힐크레스트 로드 11

브롬리 에이치. 마크

미국 03082 뉴햄프셔주 린드보로 1194 센터 로드

명 세 서

청구범위

청구항 1

그래프 컴포넌트를 나타내는 복수의 노드를 포함하는 그래프 기반의 연산을 준비하는 준비 단계; 및

상기 그래프 기반의 연산을 실행하는 실행 단계

를 포함하고,

상기 복수의 노드는 상기 그래프 컴포넌트 사이에서 데이터 흐름을 나타내는 하나 이상의 링크에 의해 연결되고,

상기 준비 단계는,

상기 연산에서 복수의 트랜잭션을 처리하기 위해, 적어도 다수의 그래프 컴포넌트의 제1 세트를 식별하는 단계; 및

제1 트랜잭션을 하나 이상의 데이터 레코드 및 이 데이터 레코드에 대응하는 데이터 동작을 포함하는 제1 그룹과 연관짓고, 제2 트랜잭션을 하나 이상의 데이터 레코드 및 이 데이터 레코드에 대응하는 데이터 동작을 포함하는 제2 그룹과 연관짓는 단계

를 포함하고,

상기 실행 단계는, 적어도 상기 제1 및 제2 트랜잭션을 처리하기 위해 상기 그래프 기반의 연산을 실행하는 단계로서, 상기 제1 그룹의 최종 데이터 레코드에 대응하는 데이터 동작이 상기 그래프 컴포넌트의 제1 세트에 의해 실행된 이후 까지 상기 제2 그룹의 최초 데이터 레코드에 대응하는 데이터 동작이 상기 그래프 컴포넌트의 제1 세트에 의해 실행되는 것을 지연시키는 단계를 포함하는,

컴퓨터에서 구현되는 방법.

청구항 2

제1항에 있어서,

상기 실행 단계는,

상기 제1 및 제2 트랜잭션에 대응하는 데이터 레코드 및 데이터 동작을 처리하고, 실패가 검출된 경우, 처리된 트랜잭션에 대응하는 모든 처리된 데이터 레코드 및 데이터 동작을 롤백(rolling back)시킴으로써 제1 트랜잭션 배치(batch)를 롤백시키는 것에 의해, 상기 제1 트랜잭션 배치로서 상기 제1 및 제2 트랜잭션을 실행시키는 단계를 더 포함하는,

컴퓨터에서 구현되는 방법.

청구항 3

제1항에 있어서,

상기 그래프 컴포넌트의 제1 세트에 대한 데이터 흐름의 시작으로서 적어도 출력 포트를 포함하는 제1 그래프 컴포넌트를 특정하는 단계; 및

상기 그래프 컴포넌트의 제1 세트에 대한 데이터 흐름의 종료로서 적어도 입력 포트를 포함하는 제2 그래프 컴포넌트를 특정하는 단계

를 더 포함하는

컴퓨터에서 구현되는 방법.

청구항 4

제1항에 있어서

하나 이상의 상기 트랜잭션을 처리하기 위해 상기 그래프 컴포넌트의 제1 세트를 동적으로 불러오는 단계

를 더 포함하는

컴퓨터에서 구현되는 방법.

청구항 5

제1항에 있어서,

제1 트랜잭션 배치 내의 실질적으로 모든 트랜잭션을 성공적으로 처리한 경우, 상기 제1 트랜잭션 배치를 완료(committing)하는 단계

를 더 포함하는

컴퓨터에서 구현되는 방법.

청구항 6

제5항에 있어서,

상기 제1 트랜잭션 배치를 완료하는 단계는,

실질적으로 동시에 상기 제1 트랜잭션 배치 내의 각각의 상기 트랜잭션에 대응하는 모든 데이터 동작을 완료하는 단계를 포함하는,

컴퓨터에서 구현되는 방법.

청구항 7

제2항에 있어서,

상기 실패의 검출은,

상기 제1 트랜잭션 배치의 트랜잭션 내에서 각각의 상기 데이터 레코드에 대응하는 상기 데이터 동작의 실패를 검출하는 것을 포함하는,

컴퓨터에서 구현되는 방법.

청구항 8

제2항에 있어서,

상기 실패를 검출한 후,

상기 제1 트랜잭션 배치의 처리되지 않은 트랜잭션 및 상기 제1 트랜잭션 배치의 처리 중에 완료되지 않고 처리된 트랜잭션을 상기 제1 트랜잭션 배치와는 다른 제2 트랜잭션 배치로서 재처리하는 단계

를 더 포함하는

컴퓨터에서 구현되는 방법.

청구항 9

제2항에 있어서,

상기 실패를 검출한 후,

상기 제1 트랜잭션 배치에 연관된 각각의 처리되거나 처리되지 않은 트랜잭션을 일련의 분산 배치(batch)로서 재처리하는 단계

를 더 포함하는

컴퓨터에서 구현되는 방법.

청구항 10

제2항에 있어서,

상기 실패를 검출하는 경우, 하나 이상의 실패된 트랜잭션을 실패 트랜잭션으로서 식별하는 단계;

상기 실패 트랜잭션을 제외한 상기 제1 트랜잭션 배치를 상기 제1 트랜잭션 배치와는 다른 제2 트랜잭션 배치로서 재처리하는 단계; 및

상기 실패 트랜잭션을 제3 트랜잭션 배치로서 재처리하는 단계

를 더 포함하는,

컴퓨터에서 구현되는 방법.

청구항 11

제2항에 있어서,

상기 제1 트랜잭션을 실행하는 과정에서,

상기 그래프 기반의 연산에 속하는 그래프 컴포넌트 세트에 의해 제1 데이터베이스 내의 하나 이상의 데이터 동작을 수행하는 단계

를 더 포함하는

컴퓨터에서 구현되는 방법.

청구항 12

제11항에 있어서,

상기 제1 트랜잭션을 실행하는 과정에서,

상기 제1 데이터베이스에 제1 세션을 설정하는 단계를 더 포함하며,

상기 제1 세션은 상기 제1 트랜잭션을 처리하는 하나 이상의 그래프 컴포넌트의 제1 세트에 의해 공유되는,

컴퓨터에서 구현되는 방법.

청구항 13

제2항에 있어서,

상기 제1 트랜잭션의 크기는 사용자에 의해 설정되는,

컴퓨터에서 구현되는 방법.

청구항 14

컴퓨터 프로그램을 저장하는 컴퓨터 판독가능한 매체에 있어서, 상기 컴퓨터 프로그램은 컴퓨터로 하여금,

그래프 컴포넌트를 나타내는 복수의 노드를 포함하는 그래프 기반의 연산을 준비하도록 하고;

상기 그래프 기반의 연산을 실행하도록 하는 명령어를 포함하고,

상기 복수의 노드는 상기 그래프 컴포넌트 사이에서 데이터 흐름을 나타내는 하나 이상의 링크에 의해 연결되고,

상기 준비는,

상기 연산에서 복수의 트랜잭션을 처리하기 위해, 적어도 다수의 그래프 컴포넌트의 제1 세트를 식별하는 것; 및

제1 트랜잭션을 하나 이상의 데이터 레코드 및 이 데이터 레코드에 대응하는 데이터 동작을 포함하는 제1 그룹과 연관짓고, 제2 트랜잭션을 하나 이상의 데이터 레코드 및 이 데이터 레코드에 대응하는 데이터 동작

을 포함하는 제2 그룹과 연관짓는 것

을 포함하며,

상기 실행은, 적어도 상기 제1 및 제2 트랜잭션을 처리하기 위해 상기 그래프 기반의 연산을 실행하는 것으로서, 상기 제1 그룹의 최종 데이터 레코드에 대응하는 데이터 동작이 상기 그래프 컴포넌트의 제1 세트에 의해 실행된 이후 까지 상기 제2 그룹의 최초 데이터 레코드에 대응하는 데이터 동작이 상기 그래프 컴포넌트의 제1 세트에 의해 실행되는 것을 지연시키는 것을 포함하는,

컴퓨터 판독가능한 매체.

청구항 15

그래프 컴포넌트를 나타내는 복수의 노드를 포함하는 그래프 기반의 연산을 준비하기 위한 준비 수단; 및

상기 그래프 기반의 연산을 실행하기 위한 실행 수단

을 포함하고,

상기 복수의 노드는 상기 그래프 컴포넌트 사이에서 데이터 흐름을 나타내는 하나 이상의 링크에 의해 연결되고,

상기 준비는,

상기 연산에서 복수의 트랜잭션을 처리하기 위해, 적어도 다수의 그래프 컴포넌트의 제1 세트를 식별하는 것; 및

제1 트랜잭션을 하나 이상의 데이터 레코드 및 이 데이터 레코드에 대응하는 데이터 동작을 포함하는 제1 그룹과 연관짓고, 제2 트랜잭션을 하나 이상의 데이터 레코드 및 이 데이터 레코드에 대응하는 데이터 동작을 포함하는 제2 그룹과 연관짓는 것

을 포함하고,

상기 실행 수단은, 적어도 상기 제1 및 제2 트랜잭션을 처리하기 위해 상기 그래프 기반의 연산을 실행하기 위한 수단으로서, 상기 제1 그룹의 최종 데이터 레코드에 대응하는 데이터 동작이 상기 그래프 컴포넌트의 제1 세트에 의해 실행된 이후 까지 상기 제2 그룹의 최초 데이터 레코드에 대응하는 데이터 동작이 상기 그래프 컴포넌트의 제1 세트에 의해 실행되는 것을 지연시키는 것을 포함하는,

컴퓨터 시스템.

청구항 16

컴퓨팅 시스템으로서,

적어도 하나의 프로세서를 포함하고, 상기 프로세서는:

그래프 컴포넌트를 나타내는 복수의 노드를 포함하는 그래프 기반의 연산을 준비하고;

상기 그래프 기반의 연산을 실행하도록 구성되고,

상기 복수의 노드는 상기 그래프 컴포넌트 사이에서 데이터 흐름을 나타내는 하나 이상의 링크에 의해 연결되며,

상기 준비는,

상기 연산에서 복수의 트랜잭션을 처리하기 위해, 적어도 다수의 그래프 컴포넌트의 제1 세트를 식별하는 것; 및

제1 트랜잭션을 하나 이상의 데이터 레코드 및 이 데이터 레코드에 대응하는 데이터 동작을 포함하는 제1 그룹과 연관짓고, 제2 트랜잭션을 하나 이상의 데이터 레코드 및 이 데이터 레코드에 대응하는 데이터 동작을 포함하는 제2 그룹과 연관짓는 것

을 포함하고,

상기 실행은, 적어도 상기 제1 및 제2 트랜잭션을 처리하기 위해 상기 그래프 기반의 연산을 실행하는

것으로서, 상기 제1 그룹의 최종 데이터 레코드에 대응하는 데이터 동작이 상기 그래프 컴포넌트의 제1 세트에 의해 실행된 이후 까지 상기 제2 그룹의 최초 데이터 레코드에 대응하는 데이터 동작이 상기 그래프 컴포넌트의 제1 세트에 의해 실행되는 것을 지연시키는 것을 포함하는,

컴퓨팅 시스템.

청구항 17

제14항에 있어서,

상기 그래프 기반의 연산을 실행하는 것은,

상기 제1 및 제2 트랜잭션에 대응하는 데이터 레코드 및 데이터 동작을 처리하고, 실패가 검출된 경우, 처리된 트랜잭션에 대응하는 모든 처리된 데이터 레코드 및 데이터 동작을 롤백시킴으로써 제1 트랜잭션 배치를 롤백시키는 것에 의해, 상기 제1 트랜잭션 배치로서 상기 제1 및 제2 트랜잭션을 실행하는 것을 더 포함하는, 컴퓨터 판독가능한 매체.

청구항 18

제14항에 있어서,

상기 컴퓨터 프로그램은 컴퓨터로 하여금:

상기 다수의 그래프 컴포넌트의 제1 세트에 대한 데이터 흐름의 시작으로서 적어도 출력 포트를 포함하는 제1 그래프 컴포넌트를 특정하도록 하고;

상기 다수의 그래프 컴포넌트의 제1 세트에 대한 데이터 흐름의 종료로서 적어도 입력 포트를 포함하는 제2 그래프 컴포넌트를 특정하도록 하는 명령어를 더 포함하는, 컴퓨터 판독가능한 매체.

청구항 19

제14항에 있어서,

상기 컴퓨터 프로그램은 컴퓨터로 하여금, 하나 이상의 상기 트랜잭션을 처리하기 위해 상기 다수의 그래프 컴포넌트의 제1 세트를 동적으로 불러오도록 하는 명령어를 더 포함하는, 컴퓨터 판독가능한 매체.

청구항 20

제14항에 있어서,

상기 컴퓨터 프로그램은 컴퓨터로 하여금, 제1 트랜잭션 배치 내의 실질적으로 모든 트랜잭션을 성공적으로 처리한 경우, 상기 제1 트랜잭션 배치를 완료하도록 하는 명령어를 더 포함하는, 컴퓨터 판독가능한 매체.

청구항 21

제20항에 있어서,

상기 제1 트랜잭션 배치를 완료하는 것은,

실질적으로 동시에 상기 제1 트랜잭션 배치 내의 각각의 상기 트랜잭션에 대응하는 모든 데이터 동작을 완료하는 것을 포함하는, 컴퓨터 판독가능한 매체.

청구항 22

제17항에 있어서,

상기 실패의 검출은,

상기 제1 트랜잭션 배치의 트랜잭션 내에서 각각의 상기 데이터 레코드에 대응하는 상기 데이터 동작의 실패를 검출하는 것을 포함하는, 컴퓨터 판독가능한 매체.

청구항 23

제17항에 있어서,

상기 컴퓨터 프로그램은 컴퓨터로 하여금, 상기 실패를 검출한 후, 상기 제1 트랜잭션 배치의 처리되지 않은 트랜잭션 및 상기 제1 트랜잭션 배치의 처리 중에 완료되지 않고 처리된 트랜잭션을 상기 제1 트랜잭션 배치와는 다른 제2 트랜잭션 배치로서 재처리하도록 하는 명령어를 더 포함하는, 컴퓨터 판독가능한 매체.

청구항 24

제17항에 있어서,

상기 컴퓨터 프로그램은 컴퓨터로 하여금, 상기 실패를 검출한 후, 상기 제1 트랜잭션 배치에 연관된 각각의 처리되거나 처리되지 않은 트랜잭션을 일련의 분산 배치로서 재처리하도록 하는 명령어를 더 포함하는, 컴퓨터 판독가능한 매체.

청구항 25

제17항에 있어서,

상기 컴퓨터 프로그램은 컴퓨터로 하여금:

상기 실패를 검출하는 경우, 하나 이상의 실패된 트랜잭션을 실패 트랜잭션으로서 식별하도록 하고;

상기 실패 트랜잭션을 제외한 상기 제1 트랜잭션 배치를 상기 제1 트랜잭션 배치와는 다른 제2 트랜잭션 배치로서 재처리하도록 하며;

상기 실패 트랜잭션을 제3 트랜잭션 배치로서 재처리하도록 하는 명령어를 더 포함하는, 컴퓨터 판독가능한 매체.

청구항 26

제17항에 있어서,

상기 컴퓨터 프로그램은 컴퓨터로 하여금, 상기 제1 트랜잭션을 실행하는 과정에서, 상기 그래프 기반의 연산에 속하는 그래프 컴포넌트 세트에 의해 제1 데이터베이스 내의 하나 이상의 데이터 동작을 수행하도록 하는 명령어를 더 포함하는, 컴퓨터 판독가능한 매체.

청구항 27

제26항에 있어서,

상기 컴퓨터 프로그램은 컴퓨터로 하여금, 상기 제1 트랜잭션을 실행하는 과정에서, 상기 제1 데이터베이스에 제1 세션을 설정하도록 하는 명령어를 더 포함하고, 상기 제1 세션은 상기 제1 트랜잭션을 처리하는 하나 이상의 그래프 컴포넌트의 제1 세트에 의해 공유되는, 컴퓨터 판독가능한 매체.

청구항 28

제17항에 있어서,

상기 제1 트랜잭션의 크기는 사용자에 의해 설정되는, 컴퓨터 판독가능한 매체.

청구항 29

제15항에 있어서,

상기 실행 수단은,

상기 제1 및 제2 트랜잭션에 대응하는 데이터 레코드 및 데이터 동작을 처리하고, 실패가 검출된 경우, 처리된 트랜잭션에 대응하는 모든 처리된 데이터 레코드 및 데이터 동작을 롤백시킴으로써 제1 트랜잭션 배치를 롤백시키는 것에 의해, 상기 제1 트랜잭션 배치로서 상기 제1 및 제2 트랜잭션을 실행하기 위한 수단을 더 포함하는, 컴퓨터 시스템.

청구항 30

제15항에 있어서,

상기 다수의 그래프 컴포넌트의 제1 세트에 대한 데이터 흐름의 시작으로서 적어도 출력 포트를 포함하는 제1 그래프 컴포넌트를 특정하기 위한 수단; 및

상기 다수의 그래프 컴포넌트의 제1 세트에 대한 데이터 흐름의 종료로서 적어도 입력 포트를 포함하는 제2 그래프 컴포넌트를 특정하기 위한 수단을 더 포함하는, 컴퓨터 시스템.

청구항 31

제15항에 있어서,

하나 이상의 상기 트랜잭션을 처리하기 위해 상기 다수의 그래프 컴포넌트의 제1 세트를 동적으로 불러오기 위한 수단을 더 포함하는, 컴퓨터 시스템.

청구항 32

제15항에 있어서,

제1 트랜잭션 배치 내의 실질적으로 모든 트랜잭션을 성공적으로 처리한 경우, 상기 제1 트랜잭션 배치를 완료하기 위한 수단을 더 포함하는, 컴퓨터 시스템.

청구항 33

제32항에 있어서,

상기 제1 트랜잭션 배치를 완료하는 것은, 실질적으로 동시에 상기 제1 트랜잭션 배치 내의 각각의 상기 트랜잭션에 대응하는 모든 데이터 동작을 완료하는 것을 포함하는, 컴퓨터 시스템.

청구항 34

제29항에 있어서,

상기 실패의 검출은, 상기 제1 트랜잭션 배치의 트랜잭션 내에서 각각의 상기 데이터 레코드에 대응하는 상기 데이터 동작의 실패를 검출하는 것을 포함하는, 컴퓨터 시스템.

청구항 35

제29항에 있어서,

상기 실패를 검출한 후, 상기 제1 트랜잭션 배치의 처리되지 않은 트랜잭션 및 상기 제1 트랜잭션 배치의 처리 중에 완료되지 않고 처리된 트랜잭션을 상기 제1 트랜잭션 배치와는 다른 제2 트랜잭션 배치로서 재처리하기 위한 수단을 더 포함하는, 컴퓨터 시스템.

청구항 36

제29항에 있어서,

상기 실패를 검출한 후, 상기 제1 트랜잭션 배치에 연관된 각각의 처리되거나 처리되지 않은 트랜잭션을 일련의 분산 배치로서 재처리하기 위한 수단을 더 포함하는, 컴퓨터 시스템.

청구항 37

제29항에 있어서,

상기 실패를 검출하는 경우, 하나 이상의 실패된 트랜잭션을 실패 트랜잭션으로서 식별하기 위한 수단;

상기 실패 트랜잭션을 제외한 상기 제1 트랜잭션 배치를 상기 제1 트랜잭션 배치와는 다른 제2 트랜잭션 배치로서 재처리하기 위한 수단; 및

상기 실패 트랜잭션을 제3 트랜잭션 배치로서 재처리하기 위한 수단을 더 포함하는, 컴퓨터 시스템.

청구항 38

제29항에 있어서,

상기 제1 트랜잭션을 실행하는 과정에서, 상기 그래프 기반의 연산에 속하는 그래프 컴포넌트 세트에 의해 제1 데이터베이스 내의 하나 이상의 데이터 동작을 수행하기 위한 수단을 더 포함하는, 컴퓨터 시스템.

청구항 39

제38항에 있어서,

상기 제1 트랜잭션을 실행하는 과정에서, 상기 제1 데이터베이스에 제1 세션을 설정하기 위한 수단을 더 포함하며, 상기 제1 세션은 상기 제1 트랜잭션을 처리하는 하나 이상의 그래프 컴포넌트의 제1 세트에 의해 공유되는, 컴퓨터 시스템.

청구항 40

제29항에 있어서,

상기 제1 트랜잭션의 크기는 사용자에 의해 설정되는, 컴퓨터 시스템.

청구항 41

제16항에 있어서,

상기 그래프 기반의 연산을 실행하는 것은,

상기 제1 및 제2 트랜잭션에 대응하는 데이터 레코드 및 데이터 동작을 처리하고, 실패가 검출된 경우, 처리된 트랜잭션에 대응하는 모든 처리된 데이터 레코드 및 데이터 동작을 롤백시킴으로써 제1 트랜잭션 배치를 롤백시키는 것에 의해, 상기 제1 트랜잭션 배치로서 상기 제1 및 제2 트랜잭션을 실행하는 것을 더 포함하는,

컴퓨팅 시스템.

청구항 42

제16항에 있어서,

상기 프로세서는:

상기 다수의 그래프 컴포넌트의 제1 세트에 대한 데이터 흐름의 시작으로서 적어도 출력 포트를 포함하는 제1 그래프 컴포넌트를 특정하고;

상기 다수의 그래프 컴포넌트의 제1 세트에 대한 데이터 흐름의 종료로서 적어도 입력 포트를 포함하는 제2 그래프 컴포넌트를 특정하도록 더 구성되는, 컴퓨팅 시스템.

청구항 43

제16항에 있어서,

상기 프로세서는, 하나 이상의 상기 트랜잭션을 처리하기 위해 상기 다수의 그래프 컴포넌트의 제1 세트를 동적으로 불러오도록 더 구성되는, 컴퓨팅 시스템.

청구항 44

제16항에 있어서,

상기 프로세서는, 제1 트랜잭션 배치 내의 실질적으로 모든 트랜잭션을 성공적으로 처리한 경우, 상기 제1 트랜잭션 배치를 완료하도록 더 구성되는, 컴퓨팅 시스템.

청구항 45

제44항에 있어서,

상기 제1 트랜잭션 배치를 완료하는 것은, 실질적으로 동시에 상기 제1 트랜잭션 배치 내의 각각의 상기 트랜잭션에 대응하는 모든 데이터 동작을 완료하는 것을 포함하는, 컴퓨팅 시스템.

청구항 46

제41항에 있어서,

상기 실패의 검출은,

상기 제1 트랜잭션 배치의 트랜잭션 내에서 각각의 상기 데이터 레코드에 대응하는 상기 데이터 동작의 실패를 검출하는 것을 포함하는, 컴퓨팅 시스템.

청구항 47

제41항에 있어서,

상기 프로세서는, 상기 실패를 검출한 후, 상기 제1 트랜잭션 배치의 처리되지 않은 트랜잭션 및 상기 제1 트랜잭션 배치의 처리 중에 완료되지 않고 처리된 트랜잭션을 상기 제1 트랜잭션 배치와는 다른 제2 트랜잭션 배치로서 재처리하도록 더 구성되는, 컴퓨팅 시스템.

청구항 48

제41항에 있어서,

상기 프로세서는, 상기 실패를 검출한 후, 상기 제1 트랜잭션 배치에 연관된 각각의 처리되거나 처리되지 않은 트랜잭션을 일련의 분산 배치로서 재처리하도록 더 구성되는, 컴퓨팅 시스템.

청구항 49

제41항에 있어서,

상기 프로세서는:

상기 실패를 검출하는 경우, 하나 이상의 실패된 트랜잭션을 실패 트랜잭션으로서 식별하고;

상기 실패 트랜잭션을 제외한 상기 제1 트랜잭션 배치를 상기 제1 트랜잭션 배치와는 다른 제2 트랜잭션 배치로서 재처리하며;

상기 실패 트랜잭션을 제3 트랜잭션 배치로서 재처리하도록 더 구성되는, 컴퓨팅 시스템.

청구항 50

제41항에 있어서,

상기 프로세서는, 상기 제1 트랜잭션을 실행하는 과정에서, 상기 그래프 기반의 연산에 속하는 그래프 컴포넌트 세트에 의해 제1 데이터베이스 내의 하나 이상의 데이터 동작을 수행하도록 더 구성되는, 컴퓨팅 시스템.

청구항 51

제50항에 있어서,

상기 프로세서는, 상기 제1 트랜잭션을 실행하는 과정에서, 상기 제1 데이터베이스에 제1 세션을 설정하도록 더 구성되며, 상기 제1 세션은 상기 제1 트랜잭션을 처리하는 하나 이상의 그래프 컴포넌트의 제1 세트에 의해 공유되는, 컴퓨팅 시스템.

청구항 52

제41항에 있어서,

상기 제1 트랜잭션의 크기는 사용자에 의해 설정되는, 컴퓨팅 시스템.

발명의 설명

기 술 분 야

[0001] 본 발명은 그래프 기반 어플리케이션에서 트랜잭션 처리 방법에 관한 것이다.

[0002] 관련 출원

[0003] 본 출원은 2009년 9월 25일에 출원된 미국특허출원 제61/245,862호 및 2009년 12월 15일에 출원된 미국특허출원 제12/638,588호의 우선권을 주장하며, 각각은 본 명세서에서 참조로 첨부된다.

배 경 기 술

[0004] 복합적인 연산(complex computations)은 방향성 그래프(directed graph)를 사용한 데이터 흐름(data flow)으로 표현할 수 있는데, 여기서 복합적인 연산의 각 컴포넌트(component)는 그래프의 정점(vertex)과 연관되며, 컴포넌트 간의 데이터 흐름은 그래프의 링크(아크, 에지)에 대응한다. 이러한 그래프 기반의 연산을 구현하는 시스템에 대해서는, "EXECUTING COMPUTATIONS EXPRESSED AS GRAPHS"란 명칭의 미국특허 5,966,072호에 개시되어 있다. 그래프 기반의 연산을 실행하기 위한 한가지 방법은 그래프의 여러 정점에 각각 연관된 다수의 프로세스를 실행하고, 그래프의 링크에 따라 이들 프로세스 간에 통신 경로를 구축하는 것이다. 예를 들어, 통신 경로는 TCP/IP 또는 UNIX 도메인 소켓을 사용하거나, 프로세스 간에 데이터를 전달할 수 있는 공유 메모리를 사용할 수 있다.

발명의 내용

과제의 해결 수단

[0005] 본 발명의 하나의 관점으로서, 전체적으로, 컴퓨터로 구현되는 방법은 그래프 컴포넌트를 나타내는 복수의 노드를 포함하는 그래프 기반의 연산을 준비하는 준비 단계 및 상기 그래프 기반의 연산을 실행하는 실행 단계를 포함한다. 상기 복수의 노드는 상기 그래프 컴포넌트 사이에서 데이터 흐름을 나타내는 하나 이상의 링크에 의해 연결된다. 상기 준비 단계는 상기 연산에서 복수의 트랜잭션을 처리하기 위해, 적어도 제1 그래프 컴포넌트 세트를 식별하는 단계 및 제1 트랜잭션을 하나 이상의 데이터 레코드 및 이 데이터 레코드에 대응하는 데이터 동작을 포함하는 제1 그룹과 연관짓고, 제2 트랜잭션을 하나 이상의 데이터 레코드 및 이 데이터 레코드에 대응하는 데이터 동작을 포함하는 제2 그룹과 연관짓는 단계를 포함한다. 상기 상기 실행 단계는, 적어도 상기 제1 및 제2 트랜잭션을 처리하기 위해 상기 그래프 기반의 연산을 실행하는 단계로서, 상기 제1 그룹의 최종 데이터 레코드에 대응하는 데이터 동작이 상기 제1 그래프 컴포넌트 세트에 의해 실행된 후까지 상기 제2 그룹의 최초 데이터 레코드에 대응하는 데이터 동작이 상기 제1 그래프 컴포넌트 세트에 의해 실행되는 것을 지연시키는 단계를 포함한다.

[0006] 실행 단계는 상기 제1 및 제2 트랜잭션에 대응하는 데이터 레코드 및 데이터 동작을 처리하고, 실패가 검출된 경우, 처리된 트랜잭션에 대응하는 모든 처리된 데이터 레코드 및 데이터 동작을 롤백(rolling back)시킴으로써 제1 트랜잭션 배치(batch)를 롤백시키는 것에 의해 상기 제1 트랜잭션 배치로서 상기 제1 및 제2 트랜잭션을 실행시키는 단계를 포함한다.

[0007] 상기 방법은 또한 상기 제1 그래프 컴포넌트 세트에 대한 데이터 흐름의 시작으로서 적어도 출력 포트를 포함하는 제1 그래프 컴포넌트를 특정하는 단계 및 상기 제1 그래프 컴포넌트 세트에 대한 데이터 흐름의 종료로서 적어도 입력 포트를 포함하는 제2 그래프 컴포넌트를 특정하는 단계를 더 포함할 수 있다.

[0008] 상기 방법은 하나 이상의 상기 트랜잭션을 처리하기 위해 상기 제1 그래프 컴포넌트 세트를 동적으로 불러오는 단계를 더 포함할 수 있다.

[0009] 상기 방법은 상기 제1 트랜잭션 배치 내의 실질적으로 모든 트랜잭션을 성공적으로 처리한 경우, 상기 제1 트랜잭션 배치를 완료(committing)하는 단계를 더 포함할 수 있다. 제1 트랜잭션 배치를 완료하는 단계는 실질적으로 동시에 상기 제1 트랜잭션 배치 내의 각각의 상기 트랜잭션에 대응하는 모든 데이터 동작을 완료하는 단계를 포함한다.

[0010] 실패의 검출은 상기 제1 트랜잭션 배치의 트랜잭션 내에서 각각의 상기 데이터 레코드에 대응하는 상기 데이터 동작의 실패를 검출하는 것을 포함한다.

[0011] 상기 방법은 실패를 검출한 후, 상기 제1 트랜잭션 배치의 처리되지 않은 트랜잭션 및 상기 제1 트랜잭션 배치의 처리 중에 완료되지 않고 처리된 트랜잭션을 상기 제1 트랜잭션 배치와는 다른 제2 트랜잭션 배치로서 재처리하는 단계를 포함할 수 있다.

[0012] 상기 방법은 상기 실패를 검출한 후, 상기 제1 트랜잭션 배치에 연관된 각각의 처리되거나 처리되지 않은 트랜잭션을 일련의 분산 배치(batch)로서 재처리하는 단계를 포함할 수 있다.

[0013] 상기 방법은 상기 실패를 검출하는 경우, 하나 이상의 실패된 트랜잭션을 실패 트랜잭션으로서 식별하는 단계, 상기 실패 트랜잭션을 제외한 상기 제1 트랜잭션 배치를 상기 제1 트랜잭션 배치와는 다른 제2 트랜잭션 배치로서 재처리하는 단계 및 상기 실패 트랜잭션을 제3 트랜잭션 배치로서 재처리하는 단계를 포함할 수 있다.

[0014] 상기 방법은 상기 제1 트랜잭션을 실행하는 과정에서, 상기 그래프 기반의 연산에 속하는 그래프 컴포넌트 세트에 의해 제1 데이터베이스 내의 하나 이상의 데이터 동작을 수행하는 단계를 포함할 수 있다. 상기 방법은 상기 제1 트랜잭션을 실행하는 과정에서, 상기 제1 데이터베이스에 제1 세션을 설정하는 단계를 더 포함할 수 있으며, 상기 제1 세션은 상기 제1 트랜잭션을 처리하는 하나 이상의 제1 그래프 컴포넌트 세트에 의해 공유될 수 있다.

[0015] 상기 제1 트랜잭션의 크기는 사용자에 의해 특정될 수 있다.

[0016] 본 발명의 다른 관점으로서, 컴퓨터 프로그램을 저장하는 컴퓨터 판독가능한 매체에 있어서, 상기 컴퓨터 프로그램은 컴퓨터로 하여금, 그래프 컴포넌트를 나타내는 복수의 노드를 포함하는 그래프 기반의 연산을 준비하는 준비 단계 및 상기 그래프 기반의 연산을 실행하는 실행 단계를 포함하는 컴퓨터에서 구현되는 방법을 실행시키는 명령어를 포함한다. 상기 복수의 노드는 상기 그래프 컴포넌트 사이에서 데이터 흐름을 나타내는 하나 이상의 링크에 의해 연결된다. 상기 준비 단계는 상기 연산에서 복수의 트랜잭션을 처리하기 위해, 적어도 제1 그래프 컴포넌트 세트를 식별하는 단계 및 제1 트랜잭션을 하나 이상의 데이터 레코드 및 이 데이터 레코드에 대응하는 데이터 동작을 포함하는 제1 그룹과 연관짓고, 제2 트랜잭션을 하나 이상의 데이터 레코드 및 이 데이터 레코드에 대응하는 데이터 동작을 포함하는 제2 그룹과 연관짓는 단계를 포함한다. 상기 상기 실행 단계는, 적어도 상기 제1 및 제2 트랜잭션을 처리하기 위해 상기 그래프 기반의 연산을 실행하는 단계로서, 상기 제1 그룹의 최종 데이터 레코드에 대응하는 데이터 동작이 상기 제1 그래프 컴포넌트 세트에 의해 실행된 후까지 상기 제2 그룹의 최초 데이터 레코드에 대응하는 데이터 동작이 상기 제1 그래프 컴포넌트 세트에 의해 실행되는 것을 지연시키는 단계를 포함한다.

[0017] 본 발명의 다른 관점으로서, 컴퓨터 시스템은 그래프 컴포넌트를 나타내는 복수의 노드를 포함하는 그래프 기반의 연산을 준비하는 준비 수단 및 상기 그래프 기반의 연산을 실행하는 실행 수단을 포함한다. 상기 복수의 노드는 상기 그래프 컴포넌트 사이에서 데이터 흐름을 나타내는 하나 이상의 링크에 의해 연결된다. 상기 준비 수단은 상기 연산에서 복수의 트랜잭션을 처리하기 위해, 적어도 제1 그래프 컴포넌트 세트를 식별하고, 제1 트랜잭션을 하나 이상의 데이터 레코드 및 이 데이터 레코드에 대응하는 데이터 동작을 포함하는 제1 그룹과 연관짓고, 제2 트랜잭션을 하나 이상의 데이터 레코드 및 이 데이터 레코드에 대응하는 데이터 동작을 포함하는 제2 그룹과 연관짓도록 구성된다. 상기 실행 수단은, 적어도 상기 제1 및 제2 트랜잭션을 처리하기 위해 상기 그래프 기반의 연산을 실행하는 수단으로서, 상기 제1 그룹의 최종 데이터 레코드에 대응하는 데이터 동작이 상기 제1 그래프 컴포넌트 세트에 의해 실행된 후까지 상기 제2 그룹의 최초 데이터 레코드에 대응하는 데이터 동작이 상기 제1 그래프 컴포넌트 세트에 의해 실행되는 것을 지연시키도록 구성된다.

[0018] 본 발명의 하나 이상의 실시예의 상세한 사항은 이하의 도면 및 상세란 설명에 제시된다. 이상 설명한 것 외의 본 발명의 다른 특징이나 장점은 이하의 도면, 상세한 설명과 청구범위로부터 명백하게 알 수 있을 것이다.

도면의 간단한 설명

[0019] 도 1은 그래프 기반 시스템의 블록도이다.

도 2 내지 4는 트랜잭션 처리를 나타내는 데이터 흐름 그래프의 실시예이다.

도 5는 트랜잭션 처리 시나리오의 실시예이다.

도 6은 데이터 흐름 그래프 준비 및 상기 데이터 흐름 그래프 내에서 트랜잭션 실행의 예시적인 실시예를 나타내는 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0020] 도 1을 참조하면, 그래프 기반의 연산을 실행하기 위한 연산 시스템(100)은 데이터 저장부(102)에 연결된 개발 환경(development environment, 104)과 데이터 저장부(102)에 연결된 런타임 환경(106)을 포함한다. 개발자(101)는 개발 환경(104)을 사용하여 어플리케이션을 구축한다. 어플리케이션은 하나 이상의 데이터 흐름 그래프와 연관되는데, 이러한 연산 그래프는 개발자가 개발 환경(104)을 사용한 결과로서 데이터 저장부(102)에 기록될 수 있는, 데이터 저장부(102) 내의 데이터 구조에 의해 특정된다. 데이터 구조는 예를 들어 데이터 흐름

그래프의 정점(컴포넌트 또는 데이터 세트를 나타냄)과 정점들 간의 링크(데이터 흐름을 나타냄)를 특정한다. 데이터 구조는 또한 다양한 특징의 컴포넌트, 데이터 세트, 및 그래프의 데이터 흐름을 포함할 수 있다. 데이터 처리 애플리케이션은, 예를 들어 하나 이상의 입력 데이터 세트로부터 처리용 컴포넌트의 그래프를 통해 하나 이상의 출력 데이터 세트로 흐르는 데이터에 대해 수행되는 연산을 수행하는 데이터 흐름 그래프와 연관될 수 있다.

[0021] 런타임 환경(106)은 UNIX 운영 시스템과 같은 적절한 운영 시스템의 제어하에서 하나 이상의 범용 컴퓨터에 대하여 호스트 역할을 할 수 있다. 예를 들어, 런타임 환경(106)은 다수의 중앙 처리 장치(CPU), 로컬(예컨대, SMP 컴퓨터와 같은 다중 프로세서 시스템) 또는 로컬 분산된(예컨데, 클러스터 또는 MPP로서 연결된 다중 프로세서), 원격 또는 원격 분산된(예컨데, LAN 또는 WAN 네트워크를 통해 연결된 다중 프로세서), 또는 이들의 조합을 이용하는 컴퓨터 시스템의 구성을 갖는 다중 노드(multiple-node) 병렬 컴퓨팅 환경을 포함할 수 있다. 그래프에서 다수의 컴포넌트를 동시에 실행시키는 것은 한가지 형태의 병렬 특성을 제공한다. 추가적인 병렬 특성은 그래프의 여러 컴포넌트를 여러 컴퓨팅 노드(예를 들어, 다른 CPU)에 분산시킴으로써 이루어질 수 있다. 그래프의 요소(예를 들어, 데이터 세트, 컴포넌트, 및 흐름)는, 추가의 병렬 특성을 런타임 환경(106)에 도입하기 위해, 직접적으로 또는 간접적으로 복제될 수 있다.

[0022] 사용자(103)는, 예를 들어 명령어 라인 또는 그래픽 인터페이스를 이용하여 런타임 환경(106)과 상호작용을 할 수 있다. 런타임 환경(106)은 소정의 데이터 흐름 그래프를 특정하는 저장된 그래프 데이터 구조를 판독하고, 컴포넌트의 연산을 수행하기 위한 프로세스(호스트 운영 시스템 내에서의 프로세스 또는 실행 쓰레드(thread of execution))와 같은 컴퓨팅 리소스를 할당 및 구성하기 위한 전처리 모듈(pre-execution module, 110)을 포함한다.

[0023] 런타임 환경(106)은 또한 전처리 모듈(110)에 의해 데이터 흐름 그래프에 할당된 프로세스의 실행을 스케줄링 및 제어하기 위한 실행 모듈(execution module, 112)을 포함한다. 실행 모듈(112)은, 데이터베이스 엔진, 데이터 저장장치, 또는 그래프 컴포넌트와 연관된 처리 과정 중에 액세스되는 다른 모듈과 같이, 시스템(100)에 연결된 외부 컴퓨팅 리소스와 상호작용할 수 있다.

[0024] 데이터 흐름 그래프의 실행 후, 또는 실행 중의 기설정된 간격으로, 보고 모듈(reporting module, 114)은 데이터 흐름 그래프의 개별 성분에 연관된 통계와 같이, 소정의 연산을 특징짓는 정보를 제공한다. 보고 모듈(114)에 의해 보고되는 정보의 일부는 데이터 흐름 그래프에 의해 생성되는 출력으로부터 얻어진다. 보고되는 정보의 일부는 데이터 흐름 그래프의 실행을 모니터링 하는 것으로부터 얻어진다.

[0025] 런타임 모니터링 모듈(RMM, 116)은 데이터 흐름 그래프에 할당된 하나 이상의 프로세스의 실행을 모니터하고 보고 모듈(114)에 정보를 제공한다. 상기 정보는 예를 들어, 각각의 컴포넌트를 구동하는데 전용된 중앙 처리 장치(CPU) 시간,또는 각 컴포넌트에 의해 처리되는 데이터의 양을 포함한다.

[0026] 도 2를 참조하면, 데이터 흐름 그래프(200)의 일 예는 데이터 흐름 그래프(200)의 컴포넌트 204A 내지 204I(전체적으로 204)에 의해 처리되는 하나 이상의 입력 데이터 레코드를 가지는 입력 데이터 세트(201)를 포함한다. 일 구현예에서, 입력 데이터 레코드(201)는 컴포넌트들(204) 중 하나 이상의 입력 포트로 들어가고, 출력 데이터 레코드(출력 데이터 레코드는 어떤 경우에는 입력 데이터 레코드(201)이거나 또는 입력 데이터 레코드(204)의 처리된 버전이다)는 전형적으로 컴포넌트들(204) 중 하나 이상의 출력 포트로 나온다. 일 구현예에서, 단일 입력 데이터 레코드(201)의 처리로 다수의 출력 데이터 레코드를 만들어낼 수 있다. 그래프(200)에서, 컴포넌트 204C, 204I, 및 255로부터의 출력 데이터 레코드는 출력 데이터 세트 206A 내지 206C(전체적으로 206)에 저장된다. 데이터 흐름 그래프(200)의 일 실시예는, 여기에 참조로 포함되는, 2006년 5월 16일 출원된 미국 특허 출원 제11/434,623에 보다 상세하게 설명되어 있다.

[0027] 업스트림(upstream) 컴포넌트(예를 들어, 204A)의 출력으로부터 다운스트림(downstream) 컴포넌트(예를 들어, 204B 또는 240)로의 레코드의 흐름은 일반적으로 임의의 데이터 흐름 링크(245) 상에서 동시에 발생하는 것이 허용된다. 이는 데이터 흐름 그래프(200) 내에서 상이한 컴포넌트들(204)이 상이한 레코드 상에서 동시에 과제를 수행하는 것을 허용하며, 이로써 데이터 흐름 그래프(200)에 의해 수행되는 전체적인 연산에 대한 파이프라인(pipeline) 병렬 특성의 한 형태가 제공된다. 다시 말하면, 연산의 주어진 일부가 주어진 레코드 세트 상에서 업스트림 컴포넌트(204)에 의해 수행되는 것과 병렬로 그 연산의 다른 일부가 다른 레코드 세트 상에서 다운스트림 컴포넌트에 의해 수행된다.

[0028] 데이터 흐름 그래프(200)는 뱅렬 처리 시스템상에서 실행될 수 있으며, 파일 조작 동작(생성, 삭제 및 재명명

등) 및 데이터 조작 동작(판독 및 기록 등)의 조합에 의해 다수의 외부 데이터 모음(파일 및 데이터베이스 등)을 수정할 수 있다. 이러한 외부 데이터 모음의 하나는 데이터베이스 시스템(209)이다. 데이터 흐름 그래프(200)의 컴포넌트들(204)은 데이터베이스 시스템(209)에 액세스하도록 구성된다. 이와 관련하여, 그래프(200)에 의해 처리되는 데이터 레코드는 데이터베이스 시스템(209) 상에서 수행되는 하나 이상의 데이터 동작과 연관된다.

[0029] 일 구현예에서, 컴포넌트들(204)은 이하에서 더욱 상세하게 설명되는 트랜잭션 특성을 제공하도록 구성된다. 예시적으로, 도 3을 참조하여, 현금 자동 입출금기(ATM)에서 고객의 세션으로부터 수신된 정보를 기반으로 하는 은행 고객의 레코드를 업데이트하기 위한 연산 그래프(300)를 생각해본다. 동작에서, 컴포넌트(315)는 ATM으로부터 정보를 판독하고 그 정보를 기반으로 데이터베이스 시스템(209)에서 실행되는 연관된 데이터 동작 및 데이터 레코드(312)를 생성한다. 도시된 바와 같이, 컴포넌트 "계좌 검증" 306a, "잔고 확인" 306b, 및 "잔고 갱신" 306c는 모두 동일한 데이터베이스 시스템(209)에 액세스한다. ATM에 입력되는 고객의 개인 식별 번호(PIN)에 관한 정보를 전달하는 제1 데이터 레코드(312a)는 "계좌 검증" 306a에서 처리된다. 또한, 일정한 금액이 인출되는 고객의 계좌를 지시하는 정보를 전달하는 제2 데이터 레코드(312b)는 "잔고 갱신" 306c에서 처리된다.

[0030] 만약 제1 데이터 레코드(312a)가 제2 데이터 레코드(312b)와 독립적으로 처리된다면, 데이터베이스(209)에 대한 독립적인 완료(commit)는 고객의 정보를 검증하기 전에 어쩌면 데이터베이스가 계좌의 잔고를 갱신하는 결과를 야기할 수 있다. 그렇게되면, 데이터베이스 시스템(209)에 들어있는 데이터는 요구되는 형태가 아닐 수 있다. 이러한 상황을 다루기 위해, 단일 유닛으로서 함께 완료되거나 또는 실패될(즉, 롤백(rolled back)) 필요가 있는 데이터 레코드(312) 및 연관된 동작들은 모든 레코드(312) 및 이 레코드(312)에 대응되는 동작들이 처리된 후에만 전체로서 완료되는 트랜잭션으로서 그룹화된다. 동일한 트랜잭션의 일부로서 처리되기 위해 함께 그룹화되는 레코드(312)는 "작업 단위(unit of work)"로 명명될 수 있다. 나아가, 상이한 트랜잭션은 선택적으로 트랜잭션 배치(batch)로 함께 그룹화될 수 있으며, 상기 트랜잭션은 상기 트랜잭션 배치 내의 트랜잭션들이 함께 완료되거나 또는 트랜잭션 배치 전체가 롤백되도록 하기 위해 하나의 단위로서 처리된다. 트랜잭션 배치의 크기 즉, 배치 내의 트랜잭션들의 수는 사용자에 의해 특정되거나, 또는 전처리 모듈(110, 도 1)에 의해 자동으로 결정될 수 있다. 나아가, 컴포넌트 315, 306a 내지 306d, 및 310은 이하에서 구체적으로 설명되는 트랜잭션 유닛에 대응하는 세트로 함께 그룹화될 수 있다. 일반적으로, 데이터 레코드(312)는 독립적이고 구별되는 유닛으로서 데이터베이스(209)에 대해 각각 완료되는 개별 트랜잭션들 또는 트랜잭션 배치들에서 트랜잭션 유닛에 의해 처리된다.

[0031] 본 실시예에서, 컴포넌트 306a 내지 306c는 각각 데이터베이스 시스템(209)에 접속하고, 예를 들어, 데이터를 수신하고, 변형하며, 및/또는 삭제하도록 구성된다. 일련의 데이터 레코드(312)는 "판독(read)" 컴포넌트(315)에 의해 입력으로서 수신된다. 데이터 레코드(312)의 처리 및 연관된 데이터 동작들이 데이터베이스 시스템(209)의 영구적인 변화를 야기할 때, 상기 변화를 "완료(commit)"라 한다. 이러한 완료된 변화는 컴퓨팅 시스템(100) 내에서의 모든 프로세스들에게 보여질 수 있게 이루어진다. 이와 관련하여, 완료 동작은 데이터 레코드(312) 및 연관된 데이터 동작에 의해 영향을 받은 변화를 데이터베이스 시스템(209) 내에서 영구적으로 만들기 위해 데이터 레코드(312) 및 연관된 동작에 대응하는 각각의 트랜잭션에 대하여 수행된다. 만약 실패하게 된다면, 데이터 레코드(312) 및 연관된 데이터 동작을 포함하는 실패한 트랜잭션은 단일 유닛으로서 롤백될 수 있으며(즉, 동작이 실행되기 전의 데이터베이스 시스템(209)의 이전 단계가 복구될 수 있다), 데이터 레코드(312) 및 연관된 동작은 폐기되거나, 또는 실패의 원인을 검토한 후 재처리될 수 있다. 실패는 다양한 원인에 의해 발생할 수 있다. 예를 들어, 실패는 메모리 할당 에러 또는 메모리 공간에서의 기록 충돌과 같은 시스템 실패일 수 있다. 실패는 또한 예를 들어 고갈된 계좌(depleted account)로부터 자금을 인출하려는 시도와 같은 근본적인 데이터 동작에서의 실패일 수 있다. 이러한 시나리오에서, 데이터 레코드(312) 및 연관된 데이터 동작을 포함하는 트랜잭션은 실패할 것이다.

[0032] 일반적으로, 데이터 레코드(312) 및 연관된 데이터 동작을 포함하는 모든 트랜잭션을 처리한 후 완료하는 것은 많은 비용을 소모할 수 있다. 전형적인 완료 동작에서, 데이터베이스 시스템(209)은 데이터 동작에 의해 만들어진 데이터베이스 시스템(209)에 대한 모든 수정사항을 물리적인 디스크(예를 들어, 데이터베이스 시스템(209)에 연관된 데이터를 저장하는 하드 디스크)에 기록한다. 일 실시예에서, 로그 파일(도시되지 않음)은 또한 업데이트 될 필요가 있으므로, 완료 동작을 수행하는데 관여되는 처리 사이클의 수가 증가하게 된다. 일반적으로, 완료 동작은 상당한 리소스와 처리 주기를 요구할 수 있다. 일반적으로, 다수의 트랜잭션 배치에 대한 완료 동작을 그룹화하는 것에 대하여 여기에 설명된 예시적인 기술들은 이러한 비용을 줄이고 자원을 보호할 수

있다.

[0033] 상기한 바와 같이, 구별된 트랙잭션에 연관된 작업의 단위로 데이터 레코드의 도입(incoming) 흐름을 처리하는 것은 다른 한편으로 이러한 트랜잭션에 대한 컴포넌트 처리 동작 세트 내에서 제공되는 파이프라인 병렬 특성을 억제함으로써 임의의 트랜잭션 특성(예를 들어, 분리(isolation), 이하에서 자세히 설명됨)을 가능하게 한다. 도 2를 참조하면, 트랜잭션은 전체적인 데이터 흐름 그래프(200)에 의해 정의되는 연산의 일부에 의해 수행될 수 있다. 예를 들어, 데이터베이스 시스템(209)과 연결되어 수행되는 트랜잭션은 데이터베이스 시스템(209)과 통신하는 컴포넌트 204D 내지 204G 만을 포함할 수 있다. 이러한 라인을 따라, 일 실시예에서, 세트(250)는 컴포넌트 204D 내지 204G를 포함하는 트랜잭션을 수행하기 위해 정의될 수 있다. 이러한 방법에서, 컴포넌트 204D 내지 204G의 세트(250)는 트랜잭션에 포함되는 데이터 레코드 상에서 총괄적으로 동작을 실행할 수 있다. 일 실시예에서, 각각 상이한 종류의 트랜잭션을 수행하는 하나 이상의 컴포넌트 세트 (250)가 존재할 수 있다. 세트(250)는 다양한 방법으로 설정될 수 있다. 예를 들어, 세트(250)는 그래프가 실행되기 이전에 설정될 수 있다. 일 실시예에서, 세트는 그래프의 스타트업(stratup) 동안에 설정될 수 있다. 컴포넌트(도시되지 않음)는 세트를 동적으로 불러오기 위해 사용자에 의해 정의될 수 있다. 일 구현예에서, 데이터 흐름 그래프(200)를 실행하는데 있어서, 전처리 모듈(110, 도 1)은 트랜잭션을 처리하는 세트(250)의 일부가 될 컴포넌트(204)를 식별할 수 있다.

[0034] 일 구현예에서, 세트(250)는 하나 이상의 특별한 컴포넌트(240, 255)를 사용함으로써 설정될 수 있으며, 상기 특별한 컴포넌트(240, 255)는 소정의 트랜잭션에 대한 트랜잭션의 시작과 끝의 범위를 정한다. 일 실시예에서, 컴포넌트(240, 255)는 그래프(200)가 사용자(도 1의 사용자(103))에 의해 전개되기 전에 설계자(예를 들어, 도 1의 개발자(101))에 의해 특정될 수 있다. 일 실시예에서, 컴포넌트(240, 255)는 그래프(200)를 수정할 능력을 갖는 최종 사용자(103)에 의해 특정될 수 있다. 도시된 바와 같이, 제1 컴포넌트, 예를 들어 적어도 하나의 출력 포트를 포함하는 "트랜잭션 시작"(BT, 240)는 트랜잭션에 대한 데이터 흐름(245)의 시작으로 특정될 수 있으며, 제2 컴포넌트, 예를 들어 적어도 하나의 입력 포트를 포함하는 "트랜잭션 완료"(ET, 255)는 트랜잭션에 대한 데이터 흐름(245)의 종료로 특정될 수 있다. 일 구현예에서, 트랜잭션의 데이터 레코드에 연관된 트랜잭션의 동작을 실행하는 컴포넌트들(204)의 세트(250)는 트랜잭션 유닛(TU)으로 명명된다. 처리되고 있는 단일 트랜잭션에 대한 데이터 레코드만이 TU로 흐르도록 허용하는 것을 보장함으로써, 파이프라인 병렬 특성은 TU 내에서 억제된다. 다음 트랜잭션에 대한 데이터 레코드가 TU에 의해 처리되기 위해 이전 트랜잭션이 완성된 후(예를 들어, 완료된 후) TU로 흐를 수 있다. BT 240과 ET 250은 이러한 조건을 집행하기 위해 통신할 수 있다. 데이터 흐름 그래프(200)는 다수의 TU를 포함할 수 있다.

[0035] 일 구현예에서, 스타트업에서, 그래프(200)는 세트(250) 내의 컴포넌트들(204)을 결정하기 위해 분석될 수 있다. 세트(250) 내의 컴포넌트들(204)은 단일 프로세스에서 실행될 수 있다. 이러한 접근에 있어서, 전처리 모듈(110, 도 1)은 그래프(200)의 컴포넌트들(204)을 각각 하나 이상의 컴포넌트(204)를 가지는 하나 이상의 세트로 분할한다. 하나 이상의 컴포넌트(204)를 가지는 각각의 세트는 상이한 프로세스에 할당될 수 있다. 따라서, 동일한 세트에 위치하는 하나 이상의 컴포넌트들(204)에 대하여, 이러한 컴포넌트들(204)에 의해 표현되는 연산은 동일한 프로세스에서 실행되기 위해 "함께 접힐(folded together)" 수 있다. 프로세스에서 하나 이상의 컴포넌트(204)는 연속으로 실행될 수 있다. 이러한 방법에서, 컴포넌트들이 별개의 프로세스에 존재한다면 상기 컴포넌트를 실행할 수 있게 하기 위해, 레코드는 하나의 컴포넌트(204)로부터 프로세스 내의 다음 컴포넌트로 전달된다.

[0036] 소정의 세트(250) 내의 컴포넌트들(204)에 대하여, BT 240과 ET 255는 상술한 내용에 따라 특정된다. BT 240에 대한 하나의 입력, 하나 이상의 ET 255로부터 선택적인 출력, 및 세트(250) 내의 컴포넌트로부터 다른 출력들이 존재할 수 있다. 일 실시예에서, BT 240의 업스트림(예를 들어, 컴포넌트 204A) 및 ET 255의 다운스트림 컴포넌트가 존재할 수 있다.

[0037] 일 구현예에서, 소정의 트랜잭션 처리에 잠재적으로 참여하는 컴포넌트(204)는 BT 240이 세트(250)의 일부라는 것을 나타내기 위해 "발견(discovery)" 프로세스를 사용할 수 있다. 예를 들어, 컴포넌트 204D 내지 204G는 자체적으로 세트(250)에 할당된 컴포넌트(204)로 등록함으로써, BT 240은 소정의 트랜잭션 처리에 포함되는 컴포넌트 204D 내지 204G를 알 수 있다.

[0038] 일 실시예에서, 새로운 트랜잭션은 BT 240이 데이터 레코드(201)를 수신할 때 개시되는 반면, 다른 나머지 트랜잭션은 활성화되지 않는다. 트랜잭션의 개시 시점에서, BT 240은 각각의 컴포넌트(204) 및 ET 255가 트랜잭션 처리를 개시할 것을 지시한다. 트랜잭션에 연관된 데이터 레코드(201)는 트랜잭션에서 데이터 흐름(245)의 일

부로서 컴포넌트 204D 내지 204G들 간에 전달된다. 트랜잭션 처리의 종료 시점에서, ET 255는 상술한 바와 같이 트랜잭션의 결과를 완료 또는 중단시킬 수 있다.

[0039] 나아가, 일 구현예에서, 세트(250) 내의 하나 이상의 컴포넌트들(204D 내지 204G)은 데이터베이스 시스템(209)에 액세스하기 위한 연결을 공유할 수 있다. 일반적으로, 세트(250) 내의 컴포넌트들(204D 내지 204G)을 사용하는 트랜잭션을 실행하는 과정에서, 세션은 데이터베이스 시스템(209)과 함께 설정될 수 있으며, 세트(250) 내의 컴포넌트들(204D 내지 204G)에 의해 공유될 수도 있다. 이러한 식으로, 트랜잭션 배치를 처리하는 콘텍스트(context)에서, 데이터베이스 시스템(209)에 관련된 모든 데이터 레코드 및 연관된 데이터 동작은 동일한 데이터베이스 세션 내에서 처리될 수 있다.

[0040] 연산 그래프 콘텍스트에서 신뢰성을 향상시키기 위해, 데이터 흐름 그래프(200)의 TU는 "ACID" 특성을 제공하는 트랜잭션을 구현하도록 구성되어야 한다. 즉, 시스템(100)은 소정의 트랜잭션의 데이터 레코드(201) 및 연관된 데이터 동작에 대하여 "원자성(atomicity)", "일관성(consistency)", "독립성(isolation)" 및 "내구성(durability)"을 보장한다. 예를 들어, 원자성을 충족시키기 위해, TU는 데이터베이스 시스템(209)과 상호 작용하여 단일 트랜잭션 내에서 실행되는 데이터베이스 시스템(209)에 대한 변화와 연관되는 모든 동작이 함께 완료되도록 한다. 예를 들어, 도 3의 실시예에서, 고객의 정보가 성공적으로 확인되었을 때에만 고객의 계좌에 대한 업데이트가 성공적으로 일어날 수 있다. 유사하게는, 하나의 계좌에서 다른 계좌로 자금을 이체하는 어플리케이션에 대하여, 원자성은 하나의 계좌로부터 이체가 성공적으로 이루어진다면, 대응하는 입금이 다른 계좌에도 만들어지는 것을 보장한다. 이러한 방법에서, 모든 변화는 데이터베이스 시스템(209) 내에서 수행되거나, 어떠한 변화도 수행되지 않을 수 있다.

[0041] 원자성 이외에도, 트랜잭션이 시작하고 완료될 때 시스템(100)은 데이터베이스(209) 내의 데이터가 일관된 상태로 있는 것을 보장한다. 이는 데이터베이스(209)가 "일관성"을 충족시키는 것을 보장한다. 예를 들어, 하나의 계좌에서 다른 계좌로 자금을 이체하는 어플리케이션에 대하여, 일관성은 두 계좌의 전체 자금의 가치가 각 트랜잭션의 시점과 종점에서 동일한 것을 보장한다.

[0042] 나아가, 시스템(100)은 트랜잭션의 중간 단계가 다른 트랜잭션에서 보이지 않도록 하는 것을 보장한다. 이는 시스템(100)이 독립성을 충족시키는 것을 보장한다. 예를 들어, 하나의 계좌에서 다른 계좌로 자금을 이체하는 어플리케이션에 대하여, 다른 트랜잭션은 하나의 계좌에서 다른 계좌로 이체된 자금을 볼 수 있으나, 양쪽 어느 것에도 속하지 않는다.

[0043] 최종적으로, 트랜잭션이 성공적으로 완성된 후, 시스템(100)은 데이터의 변화를 유지하며, 시스템 실패가 발생한 경우일지라도 이를 분실하지 않는다. 이는 시스템(100)이 "내구성"을 충족하는 것을 보장한다. 예를 들어, 하나의 계좌에서 다른 계좌로 자금을 이체하는 어플리케이션에 대하여, 내구성은 각각의 계좌에서 형성된 변화가 분실되지 않도록 하는 것을 보장한다.

[0044] 여러 가지 기술들이 트랜잭션 처리 중의 실패를 다루기 위해 사용될 수 있다. 만약 실패가 발생하면, 데이터베이스 시스템(209)에 대하여 부분적인 변화가 생성될 수 있으며, 이는 수정될 때까지 현재의 어플리케이션 또는 다른 어플리케이션에 의해 사용할 수 없는 데이터베이스 시스템(209) 내의 데이터를 만든다. 전형적으로, 이러한 현상은 데이터베이스 시스템(209) 내의 레코드가 연산 과정에서 수정되거나, 삭제되거나 생성되는 경우에 발생한다. 병렬 처리 시스템에서, 데이터베이스 시스템(209)이 종종 다수의 상이한 컴퓨팅 시스템 및 저장 유닛(예를 들어, 자기 디스크) 상에서 펼쳐지고, 작업이 "롤백"을 요구하게 하므로 상기 문제가 더 심화되며, 데이터의 상태는 저장 유닛의 수에 비례해서 증가한다.

[0045] 이러한 실패로부터 회복하기 위해, 현재(즉, 실패된)의 어플리케이션을 닫고, 어플리케이션 시작부터 어플리케이션에 의해 생성된 모든 변화를 원래대로 돌리거나("전체 롤백(full rollback)"), 시스템의 상태를 중간 "체크포인트"에 저장하고 상기 포인트로부터 실행을 재시작("부분 롤백(partial rollback)")할 필요가 있다.

[0046] 상기에서 제안한 바와 같이, 데이터 흐름 그래프(200)의 특징은 데이터 레코드(201) 및 데이터 레코드의 대응하는 데이터 동작이 다수의 트랜잭션에 연관될 수 있다는 것이다. 소정의 트랜잭션에 연관된 데이터 레코드(201) 및 대응하는 데이터 동작은 단일 유닛으로서 완료되거나(즉, 영구적이 됨) 또는 롤백(즉, 원래대로 돌림)될 수 있다.

[0047] 도 4를 참조하면, 데이터 흐름 그래프(400)는 그래프 컴포넌트 406a 내지 406d(전체적으로 406) 및 410a 내지 410c(전체적으로 401)을 포함한다. 도시된 바와 같이, 컴포넌트들(406)의 세트(402)는 트랜잭션을 처리하는 TU로 정의된다. 나아가 컴포넌트(406)는 데이터베이스 시스템(209)과 통신한다. 구현에 있어서, 임의의 컴포넌

트는, 예를 들어 컴포넌트 410은, 세트(402)에 포함되지 않으며, 따라서 세트(402)에 연관된 트랜잭션 처리에 관여하지 않는다.

[0048] 일 구현예에서, 입력 데이터 레코드(312)는 트랜잭션 배치 414a, 414b(전체적으로 414)로 함께 그룹화될 수 있으며, 각각은 TU(402)에서 처리되기 위해 "작업 단위"내의 레코드의 상이한 그룹에 대응한다. 데이터 레코드(312)를 트랜잭션(414)에 대한 작업 단위로 그룹화하는 것은 다양한 방법으로 실행될 수 있다. 예를 들어, 사용자(103)는 각각의 트랜잭션(414)에 대한 트랜잭션 크기(즉, 데이터 레코드(312)의 수)를 정의할 수 있다. 일 실시예에서, 전처리 모듈(110, 도 1)은 예를 들어, 데이터 레코드의 기설정된 특성을 기반으로 데이터 레코드를 자동으로 트랜잭션(414)으로 분할할 수 있다. 예를 들어, 만약 데이터 레코드(312)가 예를 들어, 계산 현금 레지스터(checkout cash register)로부터의 라인 항목 레코드 스트림이면, 각각의 레코드(312)는 예를 들어, "레지스터 번호", "트랜잭션 번호", "항목 번호", "항목 코드" 및 "가격" 등의 특징을 포함할 수 있다. 만약 계산 현금 레지스터에서 발생하는 각각의 판매가 단일 트랜잭션으로서 평가될 필요가 있다면, 모든 레코드(312)는 "레지스터 번호" 및/또는 "트랜잭션 번호"를 기반으로 단일 트랜잭션(414)으로 함께 그룹화될 수 있다.

[0049] 일 실시예에서, 상이한 트랜잭션 사이에 독립성을 제공하기 위해, 오직 하나의 트랜잭션, 예를 들어 트랜잭션 414a가 한번에 세트(402)의 컴포넌트들(406a 내지406d)에 의해 처리된다. 따라서, 트랜잭션 414a의 마지막 데이터 레코드가 마지막 컴포넌트 406d에 의해 처리된 후에만 트랜잭션 414b의 처음 데이터 레코드가 제1 컴포넌트 406a에 의해 처리될 수 있다.

[0050] 일 구현예에서, 완료 동작은 각 트랜잭션 후에 실행되지 않고, 지연되어 트랜잭션 배치에 대하여 한번에 실행된다. 다음의 실시예에서, "배치(batch)"는 다수의 트랜잭션 및 상기 트랜잭션의 대응하는 작업 단위의 배치를 말하는 것이며, 각각은 상기 트랜잭션에서 처리되기 위한 레코드로 구성된다. 도 5를 참조하면, 예시적인 시나리오는 작업 단위 510a 내지 510e의 배치(509)를 수신하고 처리하는 TU 내의 컴포넌트 세트(502)를 포함하는 데이터 흐름 그래프(500)의 일부를 나타낸다. 도시된 바와 같이, 세트(502)는 처리를 위해 작업 단위 510a 내지 510e의 입력을 포함하는 배치(509)를 기다린다. 배치(509)를 실행하는 과정에서, 컴포넌트 508a 내지 508c는 데이터베이스 시스템(209)과 공통 세션을 공유한다. 작업 단위 510a 및 510b에 대한 입력 트랜잭션은 세트(502)에 의해 미리 처리되어 상기 트랜잭션의 처리된 결과 514a 및 514b로 나타나며, 완성된 것으로(하지만 완료(committed)는 아님) 표시된다. 나아가, 입력된 작업 단위 510c는 작업 단위 512c로서 컴포넌트 508a에서 현재 처리되고 있다. 컴포넌트 508에 의해 처리되는 동안, 작업 단위 512c에 포함되는 레코드는 입력된 작업 단위 510c에 포함되는 레코드에 대하여 변할 수 있으나, 이들 두 레코드들은 시간상의 상이한 포인트에서 주어진 트랜잭션에 연관된 레코드에 대응한다. 상술한 바와 같이, 주어진 트랜잭션에 대하여 한번에 오직 하나의 작업 단위만이 TU의 컴포넌트에 의해 처리된다.

[0051] 만약 작업 단위 512c에 대한 트랜잭션의 처리 중에 실패가 발생한다면, 이전의 처리된 트랜잭션의 결과 514a 및 514b로 표현되는 변화는 롤백될 것이다. 처리된 트랜잭션의 결과 514a 및 514b에 대응하는 입력된 작업 단위 510a 및 510b는 세트(502)의 입력에서 여전히 사용가능하다. 일 실시예에서, 실패가 발생하면, 작업 단위 510a 내지 510e에 대응하는 트랜잭션은 배치(509)를 분리함으로써 재처리될 수 있다. 즉, 배치(509) 내에서 작업 단위 510a 내지 510e에 대응하고, 관련된 데이터 레코드 및 동작을 포함하는 각 트랜잭션은 분리되어 처리되며, 하나 이상의 개별적인 트랜잭션을 기반으로 하는 데이터베이스(209)에 대한 변화는 다음 트랜잭션을 처리하기 이전에 완료될 수 있다. 일 실시예에서, 실패한 트랜잭션은, 예를 들어 입력된 작업 단위 510c(또는 처리된 작업 단위 512c)에 대응하는 트랜잭션은 식별될 수 있으므로, 배치(509) 내에서 연관된 데이터 레코드 및 작업을 포함하는 트랜잭션은 실패한 트랜잭션, 및 (입력된 작업 단위 510c의) 연관된 데이터 레코드 와 동작 없이 새로운 배치(509)로서 재처리될 수 있다. 일 실시예에서, 실패한 트랜잭션은 단독으로 처리될 수 있으며, 배치(509) 내의 나머지 트랜잭션은 2개 이상의 더 작은 트랜잭션 배치로 분할되어 차례로 처리될 수 있다. 예를 들어, 작업 단위 510a 및 510b에 대응하는 트랜잭션은 제1 배치로서 처리될 수 있으며, 작업 단위 510d 및 510e에 대응하는 트랜잭션은 제2 배치로서 처리될 수 있다.

[0052] 도 6을 참조하면, 그래프 기반의 연산을 준비하고 연산에서 트랜잭션을 실행하기 위한 흐름도(600)가 도시된다. 하나 이상의 컴포넌트 세트(예를 들어, 도 4에서 컴포넌트들(406)의 세트(402))는 하나 이상의 트랜잭션을 처리하기 위해 식별된다 (단계 602). 상술한 바와 같이, 트랜잭션 유닛(TU)은 컴포넌트들(406)의 세트(402)를 동적으로 불러오도록 구성된 컴포넌트를 통해 식별되거나, 하나 이상의 특별한 컴포넌트(예를 들어, 도 2의 컴포넌트 240 및 255)를 사용하는 것을 통해 범위를 정할 수 있다.

[0053] 데이터 레코드 및 이 데이터 레코드의 대응하는 데이터 동작은 TU에 의해 처리되기 위해 트랜잭션 배치 내의 트

랜잭션에 연관된다 (단계 606). 예를 들어, 트랜잭션(414, 도 4)은 데이터 레코드(312) 및 이에 대응하는 동작에 연관된다. 일 실시예에서, 사용자(103)는 트랜잭션 배치 내의 각각의 트랜잭션(414)에 대한 트랜잭션 크기(즉, 데이터 레코드(312)의 수)를 특정할 수 있으며, 또는 전처리 모듈(110, 도 1)이 데이터 레코드(312) 및 대응하는 동작을 트랜잭션으로 자동으로 그룹화할 수 있다. 이후, 트랜잭션(414)은 컴포넌트 세트(402)에 의해 실행되기 위해 선택된다 (단계 610). 트랜잭션은 일 단위로서 각 트랜잭션에 대한 데이터 레코드(312)를 처리하는 것에 의해 실행될 수 있다 (단계 614).

[0054] 트랜잭션(414)을 처리하는 동안, 실패가 발생할 수 있다. 상술한 바와 같이, 데이터 레코드(312) 및 상기 데이터 레코드의 연관된 데이터 동작은 예를 들어, 시스템 실패 및/또는 동작 실패 등을 포함하는 다양한 이유에 의해 실패할 수 있다. 따라서, 트랜잭션(414)이 처리되는 동안, 트랜잭션(414)은 이와 같은 실패 및 다른 에러를 체크할 수 있다 (단계 618). 만약 실패가 검출되지 않는다면, 트랜잭션(414)에 연관된 데이터 레코드(312) 및 대응하는 데이터 동작의 처리 결과가 단일 유닛으로서 완료될 것이다 (단계 622). 만약 트랜잭션(414)의 처리에서 실패가 발생한다면, 대응하는 데이터베이스(예를 들어, 도 3의 데이터베이스(209))에 대해 생성된 변화는 모두 롤백될 필요가 있다 (단계 626). 나아가, 만약 트랜잭션(414) 실패가 함께 완료되는 다수의 트랜잭션들의 배치 내에서 처리된다면, 실패된 트랜잭션 배치는 분리될 수 있으며, 배치 내의 개별 트랜잭션에 대응하는 데이터 레코드(312) 및 연관된 데이터 동작은 실패한 트랜잭션을 포함하거나 포함하지 않은 상태에서 개별적으로 재처리될 수 있다 (단계 630). 일 구현예에서, 실패한 배치 내의 트랜잭션은 (각 트랜잭션(414)을 재처리한 후 데이터베이스(209)에 대해 완료된 변화를) 개별적으로 재처리되거나, 또는 상술한 다른 기술을 이용하여 재처리될 수 있다.

[0055] 본 명세서에서 설명하는 기술은 컴퓨터에서 실행되는 소프트웨어를 사용하여 구현될 수 있다. 예를 들어, 소프트웨어는 하나 이상의 프로세서, 하나 이상의 데이터 저장 시스템(휘발성 및 불휘발성 메모리 및/또는 저장 요소를 포함), 하나 이상의 입력 디바이스 또는 포트, 및 하나 이상의 출력 디바이스 또는 포트를 각각 포함하는 하나 이상의 프로그램화된 또는 프로그램가능한 컴퓨터 시스템(분산형, 클라이언트/서버, 또는 그리드(grid)와 같은 다양한 구조가 가능함)에서 실행되는 하나 이상의 컴퓨터 프로그램에서의 절차를 형성한다. 소프트웨어는, 예를 들어 연산 그래프의 구성 및 설계에 관련된 다른 서비스를 제공하는, 더 큰 프로그램의 하나 이상의 모듈을 구성할 수 있다. 그래프의 노드와 요소는 컴퓨터로 판독가능한 매체에 저장된 데이터 구조 또는 데이터 저장 공간에 저장된 데이터 모델에 부합하는 다른 구성의 데이터로서 구현될 수 있다.

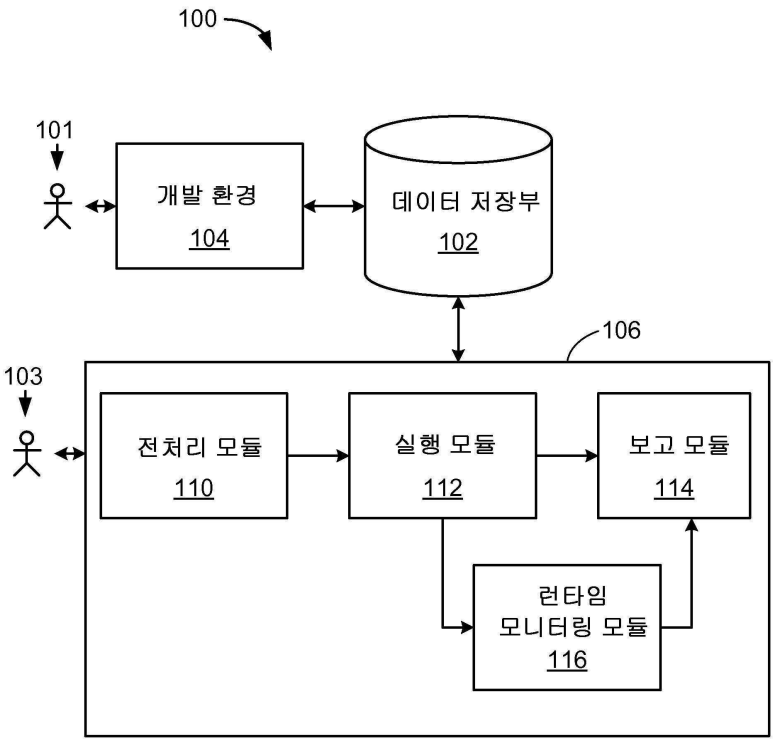
[0056] 소프트웨어는 범용 또는 전용의 프로그램가능한 컴퓨터로 판독가능한 CD-ROM과 같은 저장 매체에 제공되거나, 네트워크와 같은 통신 매체를 통해 실행가능한 컴퓨터로 전달(전파 신호로 부호화되는)될 수 있다. 모든 기능은, 전용의 컴퓨터상에서, 또는 코프로세서와 같은 전용의 하드웨어를 사용해서 수행될 수 있다. 소프트웨어는, 해당 소프트웨어에 의해 특정된 연산의 상이한 부분이 여러 컴퓨터에서 수행되는 분산 방식으로 구현되어도 된다. 이러한 각각의 컴퓨터 프로그램은, 본 명세서에서 설명하는 절차를 수행하도록 저장 매체 또는 디바이스가 컴퓨터 시스템에 의해 판독될 때에 컴퓨터를 구성 및 운영하기 위한, 범용 또는 전용의 프로그램가능한 컴퓨터에 의해 판독가능한 저장 매체 또는 디바이스(예를 들어, 고체 메모리 또는 매체, 자기 또는 광학 매체)에 저장되거나 다운로드되도록 하는 것이 바람직하다. 본 발명의 시스템은 컴퓨터 프로그램에 맞게 구성된, 컴퓨터로 판독가능한 저장 매체로서 구현될 수도 있는데, 이러한 저장 매체는 컴퓨터 시스템으로 하여금, 본 명세서에서 설명한 기능의 수행을 위해 특정되고 미리 정해진 방식으로 동작할 수 있도록 구성된다.

[0057] 본 발명에 대하여 많은 실시예를 설명하였다. 그렇지만, 본 발명의 범위를 벗어남이 없이 다양한 변형이 가능하다는 것을 알 수 있을 것이다. 예를 들어, 상기 설명한 단계들 중 몇몇은 반드시 그 순서대로 수행되지 않아도 되며, 설명된 것과 다른 순서대로 수행되어도 된다.

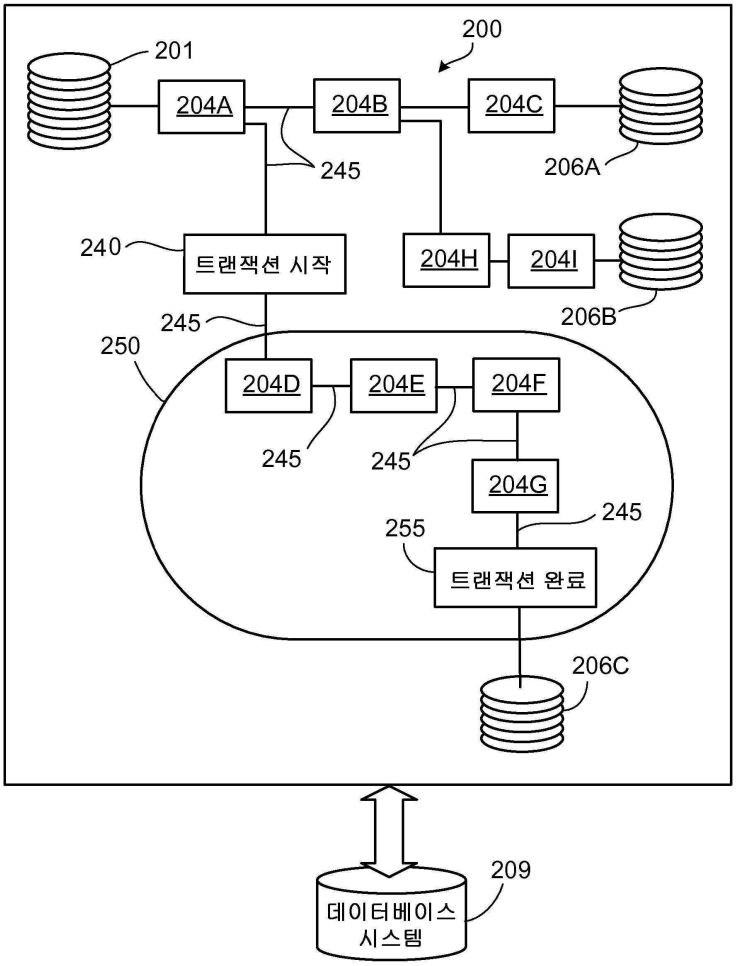
[0058] 이상의 설명은 청구범위에 의해 정해지는 본 발명의 범위를 제한하기 위한 것이 아니라 예시일 뿐이다. 예를 들어, 앞서 설명한 많은 기능적 단계들은 전체적인 과정에 실질적인 영향을 미치지 않으면서, 다른 순서로 수행되어도 된다. 다른 실시예는 이하의 청구범위에 포함된다.

도면

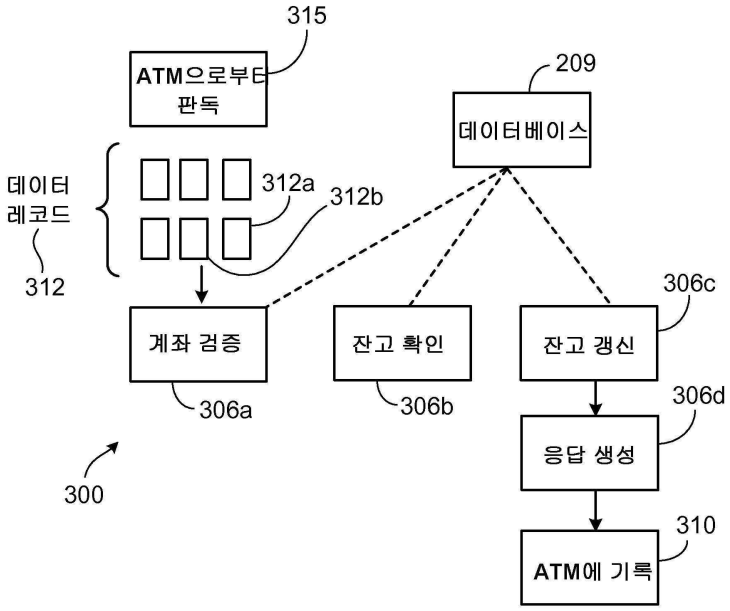
도면1


100
101
개발 환경
104
데이터 저장부
102
106
103
전처리 모듈
110
실행 모듈
112
보고 모듈
114
런타임
모니터링 모듈
116

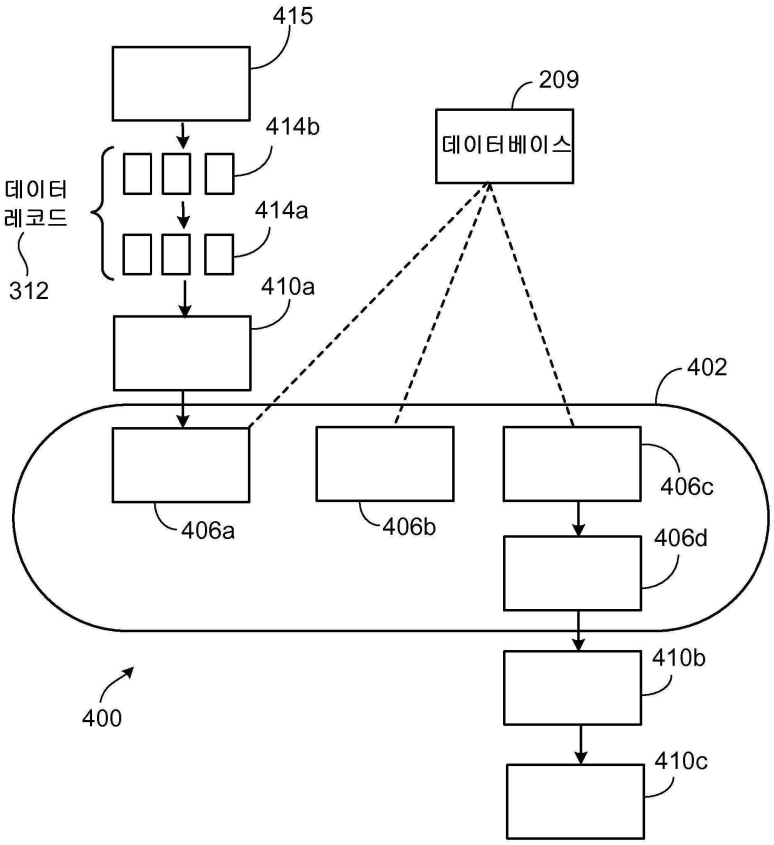
도면2


200
201
204A
204B
204C
206A
245
240
트랜잭션 시작
204H
204I
206B
245
250
204D
204E
204F
245
245
204G
245
255
트랜잭션 완료
206C
209
데이터베이스
시스템

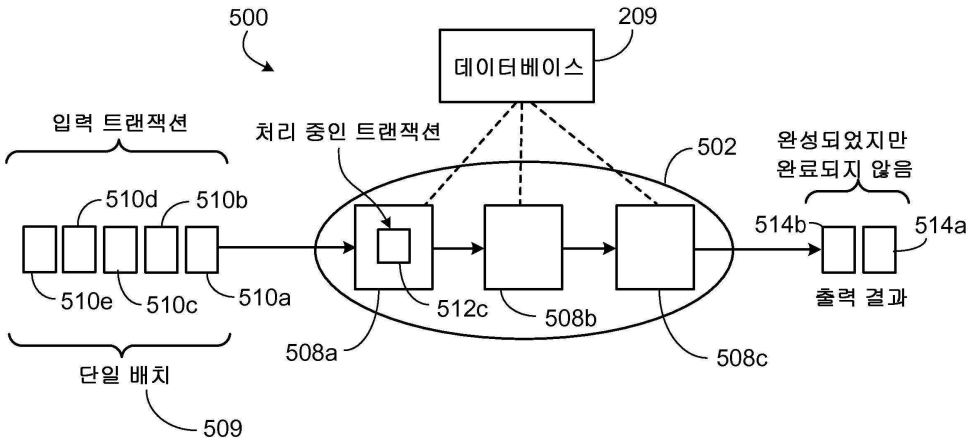
도면3


315
ATM으로부터
판독
209
데이터베이스
데이터
레코드
312
312a
312b
306c
계좌 검증
306a
잔고 확인
306b
잔고 갱신
306d
응답 생성
310
ATM에 기록
300

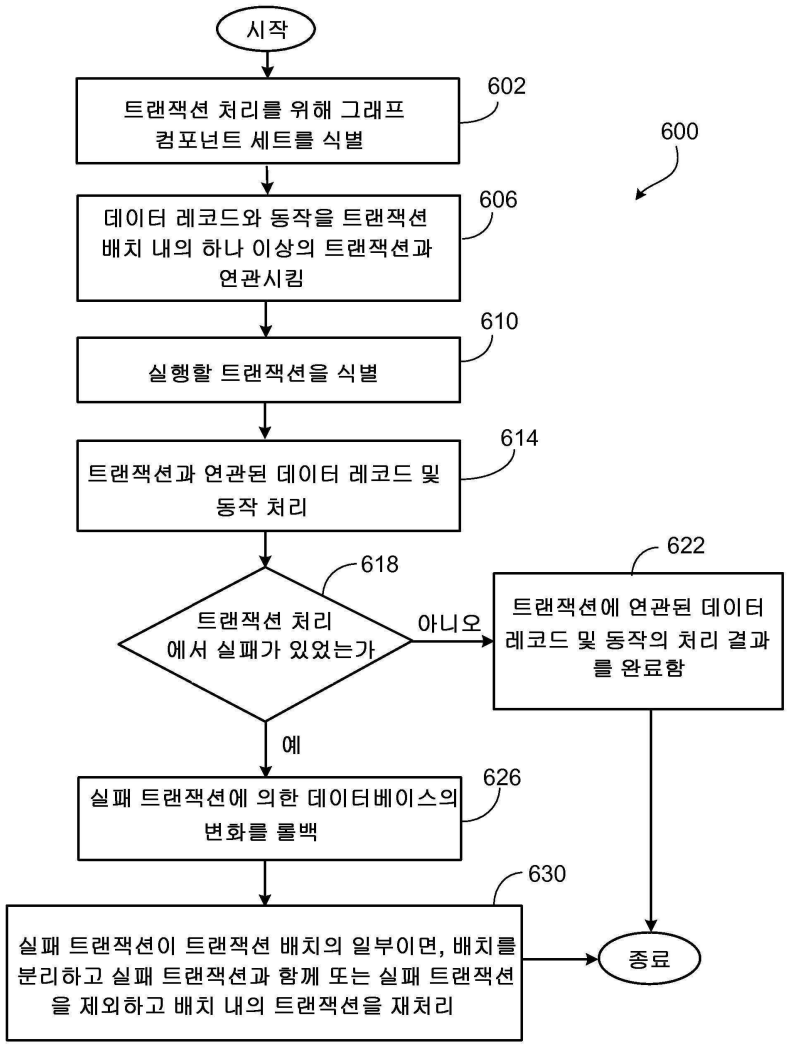
도면4


415
209
데이터베이스
414b
데이터
레코드
414a
312
410a
402
406a
406b
406c
406d
410b
410c
400

도면5


500
209
데이터베이스
입력 트랜잭션
처리 중인 트랜잭션
502
완성되었지만
완료되지 않음
510d
510b
514b
514a
510e
510c
510a
512c
508b
출력 결과
508a
508c
단일 배치
509

도면6


시작
트랜잭션 처리를 위해 그래프 컴포넌트 세트를 식별
602
600
데이터 레코드와 동작을 트랜잭션 배치 내의 하나 이상의 트랜잭션과 연관시킴
606
실행할 트랜잭션을 식별
610
트랜잭션과 연관된 데이터 레코드 및 동작 처리
614
618
622
트랜잭션 처리에서 실패가 있었는가
아니오
트랜잭션에 연관된 데이터 레코드 및 동작의 처리 결과를 완료함
예
실패 트랜잭션에 의한 데이터베이스의 변화를 롤백
626
630
실패 트랜잭션이 트랜잭션 배치의 일부이면, 배치를 분리하고 실패 트랜잭션과 함께 또는 실패 트랜잭션을 제외하고 배치 내의 트랜잭션을 재처리
종료