(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2017년07월18일
(11) 등록번호 10-1758670
(24) 등록일자 2017년07월11일

(51) 국제특허분류(Int. Cl.)
G06F 9/44 (2006.01) *G06F 11/07* (2006.01)
G06F 9/46 (2006.01) *G06N 99/00* (2010.01)
(52) CPC특허분류
G06F 9/4436 (2013.01)
G06F 11/0736 (2013.01)
(21) 출원번호 10-2016-7017122(분할)
(22) 출원일자(국제) 2008년07월25일
심사청구일자 2016년07월13일
(85) 번역문제출일자 2016년06월27일
(65) 공개번호 10-2016-0078523
(43) 공개일자 2016년07월04일
(62) 원출원 특허 10-2010-7003442
원출원일자(국제) 2008년07월25일
심사청구일자 2013년07월23일
(86) 국제출원번호 PCT/US2008/071206
(87) 국제공개번호 WO 2009/015342
국제공개일자 2009년01월29일
(30) 우선권주장
60/952,075 2007년07월26일 미국(US)
(56) 선행기술조사문헌
US5966072 B1
US20040107414 A1
JP2006504160 A

(73) 특허권자
아브 이니티오 테크놀로지 엘엘시
미국 02421 매사추세츠주 렉싱턴 스프링 스트리트 201
(72) 발명자
스탠필 크레이그 더블유.
미국 01773 매사추세츠주 린콜린 헉클베리 힐 로드 43
홀리 요셉 스케핑튼 3
미국 02478 매사추세츠주 벨몬트 힐크레스트 로드 11
(74) 대리인
유미특허법인

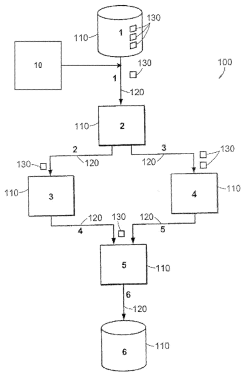
전체 청구항 수 : 총 60 항 심사관 : 최정권

(54) 발명의 명칭 **에러 핸들링이 가능한 그래프 기반의 트랜잭션 연산 처리 방법 및 시스템**

(57) 요 약

그래프 기반의 연산을 사용하여 트랜잭션을 처리하는 과정에는, 하나 이상의 일련의 연산 그래프 중의 어느 하나의 연산 그래프의 다수의 그래프 요소 중의 하나 이상이, 소정의 트랜잭션에 대해 수행해야 할 연산을 포함하는지를 판정하는 단계, 각각의 그래프 요소와 연관된 재사용가능한 연산 요소를 포함하는, 연산 그래프의 인스턴스를, 소정의 트랜잭션과 연관시키는 단계, 및 그래프를 실행하여 연산을 수행하는 단계를 포함한다.

대 표 도 - 도1



(52) CPC특허분류
G06F 11/079 (2013.01)
G06F 8/34 (2013.01)
G06F 9/466 (2013.01)
G06N 99/005 (2013.01)

명 세 서

청구범위

청구항 1

그래프 기반의 연산을 처리하기 위한 방법으로서,

정점을 연결하는 링크에 따라 작업 요소를 스케줄러에 의해 처리하는(360) 그래프 성분을 나타내는 상기 정점을 포함하는 그래프 내에서,

그래프 외부의 프로세스에 에러 정보를 제공하도록 구성된 하나 이상의 에러 핸들링 그래프 성분을 스케줄러에 의해 제공하는 단계(390); 및

스케줄러에 의해 데이터를 처리하는 단계로서, 처리 중에 에러가 발생한 그래프 성분에 응하여(370), 상기 작업 요소 중의 적어도 몇몇 작업 요소를, 상기 에러 핸들링 그래프 성분을 나타내는 정점에 대한 하나 이상의 링크에 따라 상기 에러 핸들링 그래프 성분으로 다이렉트(direct)하는 것(380)을 포함하여 상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트(redirect)하는 것을 포함하는, 데이터 처리 단계

를 포함하는, 그래프 기반의 연산 처리 방법.

청구항 2

제1항에 있어서,

상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것은, 하나 이상의 입력 대기행렬(input queue)로부터 작업 요소를 제거(remove)하는 것을 포함하는, 그래프 기반의 연산 처리 방법.

청구항 3

제1항에 있어서,

상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것은, 상기 에러 핸들링 그래프 성분으로 다이렉트된 상기 작업 요소를 처리하는 것을 포함하는, 그래프 기반의 연산 처리 방법.

청구항 4

제3항에 있어서,

상기 에러 핸들링 그래프 성분으로 다이렉트된 상기 작업 요소를 처리하는 것은, 에러가 생기기 전에 만들어진 데이터베이스에 대한 변경을 롤백(rolling back)하는 것을 포함하는, 그래프 기반의 연산 처리 방법.

청구항 5

제3항에 있어서,

상기 데이터를 처리하는 단계는, 상기 에러를 핸들링할 때 포함되지 않은 그래프 성분들에 대하여, 이러한 그래프 성분들로 다이렉트된 작업 요소를 폐기하는(discard) 단계를 포함하는, 그래프 기반의 연산 처리 방법.

청구항 6

제1항에 있어서,

스케줄러에 의해 서브 그래프(sub-graph)를 제공하는 단계로서, 상기 서브 그래프의 출력으로서 에러 코드를 제공하도록 구성된 에러 핸들링 서브 그래프 성분을 포함하는 상기 서브 그래프를 제공하는 단계를 더 포함하는, 그래프 기반의 연산 처리 방법.

청구항 7

제6항에 있어서,

상기 서브 그래프에 의해 제공된 출력이 상기 서브 그래프에서 에러가 발생했음을 나타내는 경우에, 스케줄러에 의해, 처리가 상기 에러 핸들링 그래프 성분으로 리다이렉트되는, 그래프 기반의 연산 처리 방법.

청구항 8

제1항에 있어서,

상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것은, 상기 에러가 발생한 그래프 성분으로부터 상기 에러 핸들링 그래프 성분으로, 상기 에러가 발생했을 때에 상기 그래프 성분이 처리한 작업 요소를 통신(communicate)하는 것을 포함하는, 그래프 기반의 연산 처리 방법.

청구항 9

제8항에 있어서,

상기 작업 요소는, 스케줄러에 의해, 상기 에러 핸들링 그래프 성분을 나타내는 정점에 대한 링크에 따라 통신되는, 그래프 기반의 연산 처리 방법.

청구항 10

제8항에 있어서,

상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것은, 상기 에러가 발생한 그래프 성분으로부터 상기 에러 핸들링 그래프 성분으로, 상기 에러에 관한 정보를 보고(report)하는 통신을 행하는 것을 포함하는, 그래프 기반의 연산 처리 방법.

청구항 11

제10항에 있어서,

상기 정보의 보고는, 스케줄러에 의해, 상기 에러가 발생한 그래프 성분과 상기 에러 핸들링 그래프 성분 사이에서의 암묵적 접속(implicit connection)에 따라 통신이 행해지는, 그래프 기반의 연산 처리 방법.

청구항 12

제11항에 있어서,

스케줄러에 의해, 상기 암묵적 접속을, 사용자의 요청에 응하여, 상기 에러가 발생한 그래프 성분을 나타내는 정점과 상기 에러 핸들링 그래프 성분을 표현하는 정점 사이에 명시적 링크(explicit link)로서 드러내는(reveal) 단계를 더 포함하는, 그래프 기반의 연산 처리 방법.

청구항 13

제1항에 있어서,

상기 에러 핸들링 그래프 성분을 제공하는 단계는, 복수의 에러 핸들링 그래프 성분을 제공하는 단계를 포함하며,

상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것은, 상기 에러가 발생한 그래프 성분으로부터 제공된 출력에 기초하여 에러 핸들링 그래프 성분을 선택하는 것을 포함하는, 그래프 기반의 연산 처리 방법.

청구항 14

제1항에 있어서,

상기 데이터를 처리하는 단계는, 처리 중에 그래프 성분에 에러가 발생하는 경우, 상기 에러의 원인이 된 작업 요소의 식별정보(identification)를 출력하는 단계를 더 포함하는, 그래프 기반의 연산 처리 방법.

청구항 15

제1항에 있어서,

상기 데이터를 처리하는 단계는,

상기 그래프의 제1 성분을 인에이블(enable)시키는 단계;

상기 에러 핸들링 그래프 성분을 디스에이블(disable)시키는 단계; 및

상기 에러 핸들링 그래프 성분 이외의 상기 제1 성분의 다운스트림 방향으로의 각각의 성분에 대하여, 상기 성분의 바로 업스트림 방향으로의 성분이 인에이블되어 있는 경우에, 상기 성분을 인에이블시키는 단계를 포함하는, 그래프 기반의 연산 처리 방법.

청구항 16

제15항에 있어서,

상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것은,

인에이블된 각각의 그래프 성분의 실행을 중단(stop)시키는 것;

상기 에러가 발생한 성분을 디스에이블시키는 것;

상기 에러 핸들링 그래프 성분을 인에이블시키는 것;

상기 에러 핸들링 그래프 성분의 다운스트림 방향이 아닌, 상기 에러가 발생한 성분의 다운스트림 방향의 성분을 디스에이블시키는 것; 및

상기 에러 핸들링 그래프 성분의 업스트림 방향의 성분을 인에이블시키는 것을 포함하는, 그래프 기반의 연산 처리 방법.

청구항 17

제1항에 있어서,

상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것은,

상기 에러가 제1 성분에서 발생한 경우에,

상기 에러가 제1 조건하에서 발생하였다면, 상기 제1 성분으로부터 상기 제1 성분의 업스트림 방향으로의 제1 에러 핸들링 그래프 성분으로 프로세스 흐름을 다이렉트하는 것을 포함하고,

상기 에러가 제2 조건하에서 발생하였다면, 상기 제1 성분으로부터 상기 제1 성분의 다운스트림 방향으로의 제2 에러 핸들링 그래프 성분으로 프로세스 흐름을 다이렉트하는 것을 포함하는, 그래프 기반의 연산 처리 방법.

청구항 18

제17항에 있어서,

상기 제1 조건은 카운터가 한계치 미만인 것인, 그래프 기반의 연산 처리 방법.

청구항 19

제17항에 있어서,

상기 제2 조건은 카운터가 한계치를 초과한 것인, 그래프 기반의 연산 처리 방법.

청구항 20

제17항에 있어서,

상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것은, 상기 에러에 앞서 결정되어 있던 일련의 그래프 성분을 인에이블시키는 것을 더 포함하는, 그래프 기반의 연산 처리 방법.

청구항 21

그래프 기반의 연산을 처리하기 위한 시스템으로서,

정점을 연결하는 링크에 따라 작업 요소를 처리하는 그래프 성분을 나타내는 상기 정점을 포함하는 그래프 내에서,

그래프 외부의 프로세스에 에러 정보를 제공하도록 구성된 하나 이상의 에러 핸들링 그래프 성분을 제공하기 위한 수단; 및

데이터를 처리하기 위한 수단으로서, 처리 중에 에러가 발생한 그래프 성분에 응하여, 상기 작업 요소 중의 적어도 몇몇 작업 요소를, 상기 에러 핸들링 그래프 성분을 나타내는 정점에 대한 하나 이상의 링크에 따라 상기 에러 핸들링 그래프 성분으로 다이렉트하는 것을 포함하여 상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것을 포함하는, 데이터를 처리하기 위한 수단

을 포함하는, 그래프 기반의 연산을 처리하기 위한 시스템.

청구항 22

그래프 기반의 연산을 처리하기 위한 컴퓨터 프로그램을 저장하는 컴퓨터 판독가능 매체로서, 상기 컴퓨터 프로그램은 컴퓨터 시스템으로 하여금:

정점을 연결하는 링크에 따라 작업 요소를 처리하는 그래프 성분을 나타내는 상기 정점을 포함하는 그래프 내에서,

그래프 외부의 프로세스에 에러 정보를 제공하도록 구성된 하나 이상의 에러 핸들링 그래프 성분을 제공하도록 하고,

데이터를 처리하도록 하되, 처리 중에 에러가 발생한 그래프 성분에 응하여, 상기 작업 요소 중의 적어도 몇몇 작업 요소를, 상기 에러 핸들링 그래프 성분을 나타내는 정점에 대한 하나 이상의 링크에 따라 상기 에러 핸들링 그래프 성분으로 다이렉트하는 것을 포함하여 상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것을 포함하는, 데이터를 처리하도록 하기 위한 명령을 포함하는, 컴퓨터 판독가능 매체.

청구항 23

제21항에 있어서,

상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것은, 하나 이상의 입력 대기행렬로부터 작업 요소를 제거하는 것을 포함하는, 그래프 기반의 연산을 처리하기 위한 시스템.

청구항 24

제21항에 있어서,

상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것은, 상기 에러 핸들링 그래프 성분으로 다이렉트된 상기 작업 요소를 처리하는 것을 포함하는, 그래프 기반의 연산을 처리하기 위한 시스템.

청구항 25

제24항에 있어서,

상기 에러 핸들링 그래프 성분으로 다이렉트된 상기 작업 요소를 처리하는 것은, 에러가 생기기 전에 만들어진 데이터베이스에 대한 변경을 롤백하는 것을 포함하는, 그래프 기반의 연산을 처리하기 위한 시스템.

청구항 26

제24항에 있어서,

상기 데이터를 처리하는 것은, 상기 에러를 핸들링할 때 포함되지 않은 그래프 성분들에 대하여, 이러한 그래프 성분들로 다이렉트된 작업 요소를 폐기하는 것을 포함하는, 그래프 기반의 연산을 처리하기 위한 시스템.

청구항 27

제21항에 있어서,

서브 그래프를 제공하기 위한 수단으로서, 상기 서브 그래프의 출력으로서 에러 코드를 제공하도록 구성된 에러 핸들링 서브 그래프 성분을 포함하는 상기 서브 그래프를 제공하기 위한 수단을 더 포함하는, 그래프 기반의 연산을 처리하기 위한 시스템.

청구항 28

제27항에 있어서,

상기 서브 그래프에 의해 제공된 출력이 상기 서브 그래프에서 에러가 발생했음을 나타내는 경우에, 처리가 상기 에러 핸들링 그래프 성분으로 리다이렉트되는, 그래프 기반의 연산을 처리하기 위한 시스템.

청구항 29

제21항에 있어서,

상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것은, 상기 에러가 발생한 그래프 성분으로부터 상기 에러 핸들링 그래프 성분으로, 상기 에러가 발생했을 때에 상기 그래프 성분이 처리한 작업 요소를 통신하는 것을 포함하는, 그래프 기반의 연산을 처리하기 위한 시스템.

청구항 30

제29항에 있어서,

상기 작업 요소는 상기 에러 핸들링 그래프 성분을 나타내는 정점에 대한 링크에 따라 통신되는, 그래프 기반의 연산을 처리하기 위한 시스템.

청구항 31

제29항에 있어서,

상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것은, 상기 에러가 발생한 그래프 성분으로부터 상기 에러 핸들링 그래프 성분으로, 상기 에러에 관한 정보를 보고하는 통신을 행하는 것을 포함하는, 그래프 기반의 연산을 처리하기 위한 시스템.

청구항 32

제31항에 있어서,

상기 정보의 보고는, 상기 에러가 발생한 그래프 성분과 상기 에러 핸들링 그래프 성분 사이에서의 암묵적 접속에 따라 통신이 행해지는, 그래프 기반의 연산을 처리하기 위한 시스템.

청구항 33

제32항에 있어서,

상기 암묵적 접속을, 사용자의 요청에 응하여, 상기 에러가 발생한 그래프 성분을 나타내는 정점과 상기 에러 핸들링 그래프 성분을 표현하는 정점 사이에 명시적 링크로서 드러내기 위한 수단을 더 포함하는, 그래프 기반의 연산을 처리하기 위한 시스템.

청구항 34

제21항에 있어서,

상기 에러 핸들링 그래프 성분을 제공하는 것은, 복수의 에러 핸들링 그래프 성분을 제공하는 것을 포함하며,

상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것은, 상기 에러가 발생한 그래프 성분으로부터 제공된 출력에 기초하여 에러 핸들링 그래프 성분을 선택하는 것을 포함하는, 그래프 기반의 연산을 처리하기 위한 시스템.

청구항 35

제21항에 있어서,

상기 데이터를 처리하는 것은, 처리 중에 그래프 성분에 에러가 발생하는 경우, 상기 에러의 원인이 된 작업 요소의 식별정보를 출력하는 것을 더 포함하는, 그래프 기반의 연산을 처리하기 위한 시스템.

청구항 36

제21항에 있어서,

상기 데이터를 처리하는 것은,

상기 그래프의 제1 성분을 인에이블시키는 것;

상기 에러 핸들링 그래프 성분을 디스에이블시키는 것; 및

상기 에러 핸들링 그래프 성분 이외의 상기 제1 성분의 다운스트림 방향으로의 각각의 성분에 대하여, 상기 성분의 바로 업스트림 방향으로의 성분이 인에이블되어 있는 경우에, 상기 성분을 인에이블시키는 것을 포함하는, 그래프 기반의 연산을 처리하기 위한 시스템.

청구항 37

제36항에 있어서,

상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것은,

인에이블된 각각의 그래프 성분의 실행을 중단시키는 것;

상기 에러가 발생한 성분을 디스에이블시키는 것;

상기 에러 핸들링 그래프 성분을 인에이블시키는 것;

상기 에러 핸들링 그래프 성분의 다운스트림 방향이 아닌, 상기 에러가 발생한 성분의 다운스트림 방향의 성분을 디스에이블시키는 것; 및

상기 에러 핸들링 그래프 성분의 업스트림 방향의 성분을 인에이블시키는 것을 포함하는, 그래프 기반의 연산을 처리하기 위한 시스템.

청구항 38

제21항에 있어서,

상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것은,

상기 에러가 제1 성분에서 발생한 경우에,

상기 에러가 제1 조건하에서 발생하였다면, 상기 제1 성분으로부터 상기 제1 성분의 업스트림 방향으로의 제1 에러 핸들링 그래프 성분으로 프로세스 흐름을 다이렉트하는 것을 포함하고,

상기 에러가 제2 조건하에서 발생하였다면, 상기 제1 성분으로부터 상기 제1 성분의 다운스트림 방향으로의 제2 에러 핸들링 그래프 성분으로 프로세스 흐름을 다이렉트하는 것을 포함하는, 그래프 기반의 연산을 처리하기 위한 시스템.

청구항 39

제38항에 있어서,

상기 제1 조건은 카운터가 한계치 미만인 것인, 그래프 기반의 연산을 처리하기 위한 시스템.

청구항 40

제39항에 있어서,

상기 제2 조건은 카운터가 한계치를 초과한 것인, 그래프 기반의 연산을 처리하기 위한 시스템.

청구항 41

제38항에 있어서,

상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것은, 상기 에러에 앞서 결정되어 있던 일련의 그래프 성분을 인에이블시키는 것을 더 포함하는, 그래프 기반의 연산을 처리하기 위한 시스템.

청구항 42

제22항에 있어서,

상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것은, 하나 이상의 입력 대기행렬로부터 작업 요소를 제거하는 것을 포함하는, 컴퓨터 판독가능 매체.

청구항 43

제22항에 있어서,

상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것은, 상기 에러 핸들링 그래프 성분으로 다이렉트된 상기 작업 요소를 처리하는 것을 포함하는, 컴퓨터 판독가능 매체.

청구항 44

제43항에 있어서,

상기 에러 핸들링 그래프 성분으로 다이렉트된 상기 작업 요소를 처리하는 것은, 에러가 생기기 전에 만들어진 데이터베이스에 대한 변경을 롤백하는 것을 포함하는, 컴퓨터 판독가능 매체.

청구항 45

제43항에 있어서,

상기 데이터를 처리하는 것은, 상기 에러를 핸들링할 때 포함되지 않은 그래프 성분들에 대하여, 이러한 그래프 성분들로 다이렉트된 작업 요소를 폐기하는 것을 포함하는, 컴퓨터 판독가능 매체.

청구항 46

제22항에 있어서,

상기 컴퓨터 시스템으로 하여금, 서브 그래프를 제공하되, 상기 서브 그래프의 출력으로서 에러 코드를 제공하도록 구성된 에러 핸들링 서브 그래프 성분을 포함하는 상기 서브 그래프를 제공하도록 하기 위한 명령을 더 포함하는, 컴퓨터 판독가능 매체.

청구항 47

제46항에 있어서,

상기 서브 그래프에 의해 제공된 출력이 상기 서브 그래프에서 에러가 발생했음을 나타내는 경우에, 처리가 상기 에러 핸들링 그래프 성분으로 리다이렉트되는, 컴퓨터 판독가능 매체.

청구항 48

제22항에 있어서,

상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것은, 상기 에러가 발생한 그래프 성분으로부터 상기 에러 핸들링 그래프 성분으로, 상기 에러가 발생했을 때에 상기 그래프 성분이 처리한 작업 요소를 통신하는 것을 포함하는, 컴퓨터 판독가능 매체.

청구항 49

제48항에 있어서,

상기 작업 요소는 상기 에러 핸들링 그래프 성분을 나타내는 정점에 대한 링크에 따라 통신되는, 컴퓨터 판독가능 매체.

청구항 50

제48항에 있어서,

상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것은, 상기 에러가 발생한 그래프 성분으로부터 상기

에러 핸들링 그래프 성분으로, 상기 에러에 관한 정보를 보고하는 통신을 행하는 것을 포함하는, 컴퓨터 판독가능 매체.

청구항 51

제50항에 있어서,

상기 정보의 보고는, 상기 에러가 발생한 그래프 성분과 상기 에러 핸들링 그래프 성분 사이에서의 암묵적 접속에 따라 통신이 행해지는, 컴퓨터 판독가능 매체.

청구항 52

제51항에 있어서,

상기 컴퓨터 시스템으로 하여금, 상기 암묵적 접속을, 사용자의 요청에 응하여, 상기 에러가 발생한 그래프 성분을 나타내는 정점과 상기 에러 핸들링 그래프 성분을 표현하는 정점 사이에 명시적 링크로서 드러내도록 하기 위한 명령을 더 포함하는, 컴퓨터 판독가능 매체.

청구항 53

제22항에 있어서,

상기 에러 핸들링 그래프 성분을 제공하는 것은, 복수의 에러 핸들링 그래프 성분을 제공하는 것을 포함하며,

상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것은, 상기 에러가 발생한 그래프 성분으로부터 제공된 출력에 기초하여 에러 핸들링 그래프 성분을 선택하는 것을 포함하는, 컴퓨터 판독가능 매체.

청구항 54

제22항에 있어서,

상기 데이터를 처리하는 것은, 처리 중에 그래프 성분에 에러가 발생하는 경우, 상기 에러의 원인이 된 작업 요소의 식별정보를 출력하는 것을 더 포함하는, 컴퓨터 판독가능 매체.

청구항 55

제22항에 있어서,

상기 데이터를 처리하는 것은,

상기 그래프의 제1 성분을 인에이블시키는 것;

상기 에러 핸들링 그래프 성분을 디스에이블시키는 것; 및

상기 에러 핸들링 그래프 성분 이외의 상기 제1 성분의 다운스트림 방향으로의 각각의 성분에 대하여, 상기 성분의 바로 업스트림 방향으로의 성분이 인에이블되어 있는 경우에, 상기 성분을 인에이블시키는 것을 포함하는, 컴퓨터 판독가능 매체.

청구항 56

제55항에 있어서,

상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것은,

인에이블된 각각의 그래프 성분의 실행을 중단시키는 것;

상기 에러가 발생한 성분을 디스에이블시키는 것;

상기 에러 핸들링 그래프 성분을 인에이블시키는 것;

상기 에러 핸들링 그래프 성분의 다운스트림 방향이 아닌, 상기 에러가 발생한 성분의 다운스트림 방향의 성분을 디스에이블시키는 것; 및

상기 에러 핸들링 그래프 성분의 업스트림 방향의 성분을 인에이블시키는 것을 포함하는, 컴퓨터 판독가능

매체.

청구항 57

제22항에 있어서,

상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것은,

상기 에러가 제1 성분에서 발생한 경우에,

상기 에러가 제1 조건하에서 발생하였다면, 상기 제1 성분으로부터 상기 제1 성분의 업스트림 방향으로의 제1 에러 핸들링 그래프 성분으로 프로세스 흐름을 다이렉트하는 것을 포함하고,

상기 에러가 제2 조건하에서 발생하였다면, 상기 제1 성분으로부터 상기 제1 성분의 다운스트림 방향으로의 제2 에러 핸들링 그래프 성분으로 프로세스 흐름을 다이렉트하는 것을 포함하는, 컴퓨터 판독가능 매체.

청구항 58

제57항에 있어서,

상기 제1 조건은 카운터가 한계치 미만인 것인, 컴퓨터 판독가능 매체.

청구항 59

제57항에 있어서,

상기 제2 조건은 카운터가 한계치를 초과한 것인, 컴퓨터 판독가능 매체.

청구항 60

제57항에 있어서,

상기 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 것은, 상기 에러에 앞서 결정되어 있던 일련의 그래프 성분을 인에이블시키는 것을 더 포함하는, 컴퓨터 판독가능 매체.

발명의 설명

기 술 분 야

[0001] 본 발명은 그래픽에 기반을 둔 연산의 실행에 관한 것이다.

[0002] 관련출원

[0003] 본 출원은 2007년 7월 26일에 제출된 미국출원 60/952,075에 대한 우선권을 주장하며, 상기 출원을 본 명세서에서 참조에 의해 원용한다.

배 경 기 술

[0004] 복잡한 연산(complex computations)은, 방향성 그래프(directed graph)를 사용하여 데이터 흐름(data flow)으로서 표현될 수 있는데, 이러한 연산의 성분은 그래프의 정점(vertex) 및 그래프의 링크(아크, 에지)에 대응하는 성분들 사이에서의 데이터 흐름과 연관되어 있다. 이러한 그래프 기반의 연산을 구현하는 시스템은, "EXECUTING COMPUTATIONS EXPRESSED AS GRAPHS"란 명칭의 미국특허 5,966,072호에 언급되어 있다. 그래프 기반의 연산을 실행하기 위한 한가지 방안은, 그래프의 여러 상이한 정점(vertex)과 각각 연관된 다수의 프로세스를 실행하고, 그래프의 링크에 따라 프로세스들 간에 통신 경로(communication paths: 전달 경로)를 확립하는 것이다. 예를 들어, 통신 경로는 TCP/IP 또는 UNIX(등록상표) 도메인 소켓(domain socket)을 사용하거나, 프로세스들 사이에서 데이터를 전달(pass)하기 위한 공유 메모리(shared memory)를 사용할 수 있다.

발명의 내용

[0005] 하나의 관점에서, 일반적으로, 그래프 기반의 연산을 사용하여 트랜잭션을 처리하기 위한 방법은, 하나 이상의 일련의 연산 그래프(computation graph) 중의 어느 하나의 연산 그래프의 다수의 그래프 요소(graph element) 중의 하나 이상이, 소정의 트랜잭션에 대해 수행해야 할 연산을 포함하는 지를 판정하는 단계와, 각각의 그래프

요소와 연관된 재사용가능한(reusable) 연산 요소(computation elements)를 포함하는, 연산 그래프의 인스턴스(instance)를, 소정의 트랜잭션과 연관(associate)시키는 단계(310), 및 그래프를 실행하여 연산을 수행하는 단계를 포함한다.

[0006] 본 발명의 관점에는 다음에 설명하는 특징들 중의 하나 이상이 포함될 수 있다.

[0007] 일련의 연산 그래프 내의 그래프의 적어도 몇몇 인스턴스는, 하나 이상의 연산 요소를 공유(share)한다.

[0008] 연산 요소는, 운영 체제 프로세스 및 프로세스 쓰레드(process thread) 중 하나 이상에 의해 실행되는 연산을 포함한다.

[0009] 그래프 요소는 연산 그래프의 정점(vertex)을 포함한다.

[0010] 연산 그래프의 인스턴스를 트랜잭션과 연관시키는 단계는, 연산 그래프 내의 각각의 그래프 요소에 대응하는 연산 요소를, 그래프 요소의 실행을 시작하기 전에, 연산 그래프의 인스턴스에 할당(assign)하는 단계를 포함한다.

[0011] 연산 그래프의 인스턴스를 트랜잭션과 연관시키는 단계는, 연산 그래프 내의 그래프 요소에 대응하는 연산 요소를, 연산 그래프의 인스턴스에 할당하는 단계를 포함하며, 연산 그래프의 인스턴스에 할당하는 단계는, 인스턴스에 이미 할당된 연산 요소를 사용하여 다른 그래프 요소를 실행한 후에 수행된다.

[0012] 그래프 요소 중의 둘 이상의 그래프 요소는 공통 리소스(common resource)를 사용하며, 그래프를 실행하여 연산을 수행하는 단계는, 공통 리소스를 사용하는 각각의 그래프 요소를 단일의 연산 요소에 할당하는 단계를 포함한다.

[0013] 단일의 연산 요소는, 그래프 요소가 연산 요소에 할당될 때에 이미 초기화되어 있다.

[0014] 공통 리소스는 데이터베이스(database)를 포함한다.

[0015] 공통 리소스는 특정의 포트(specific port)를 포함한다.

[0016] 트랜잭션을 처리하는 방법은 트랜잭션에 대한 요청(request)을 수신하는 단계를 더 포함한다.

[0017] 본 발명의 방법은, 연산 그래프가, 트랜잭션과 다른 제2 트랜잭션에 대해 수행될 연산과 연관되는지를 판정하는 단계, 제2 트랜잭션을 연산 그래프의 인스턴스와 다른 제2 인스턴스와 연관시키는 단계, 및 그래프의 제2 인스턴스를 실행하여, 제2 트랜잭션에 대한 연산을 수행하는 단계를 포함한다.

[0018] 연산 그래프의 상이한 인스턴스를 사용하여 수행되는 트랜잭션에 대한 연산은, 타임 인터리브드(time interleaved) 방식으로 수행된다.

[0019] 트랜잭션이 동시에 여러 개가 처리된다.

[0020] 트랜잭션은 대응하는 연산 그래프에 따라 처리되는 하나 이상의 작업 요소와 각각 연관된다.

[0021] 적어도 몇 개의 트랜잭션은, 대응하는 연산 그래프에 따라 처리되는 작업 요소와 각각 하나씩 연관된다.

[0022] 본 방법은, 연산 그래프의 적어도 몇몇의 다수의 인스턴스를 형성하는 단계를 더 포함한다.

[0023] 본 방법은, 트랜잭션 중의 하나의 트랜잭션에 대한 연산을 수행하는 중에 에러가 생겼음을 확인(identify)하는 단계와, 트랜잭션 중의 다른 하나의 트랜잭션에 대한 연산을 계속해서 수행하는 단계를 더 포함한다.

[0024] 다수의 상기 트랜잭션 중의 제1 트랜잭션에 대한 처리를 제1 시간에 시작하고, 다수의 트랜잭션 중의 제2 트랜잭션에 대한 처리를 제1 시간보다 늦은 제2 시간에 시작하며, 본 방법은, 제1 트랜잭션에 대한 연산의 수행을 완료하기 전에 제2 트랜잭션에 대한 연산의 수행을 완료하는 단계를 더 포함한다.

[0025] 다른 관점으로서, 일반적으로, 그래프 기반의 연산을 사용하여 트랜잭션을 처리하기 위한 시스템은, 하나 이상의 일련의 연산 그래프 중의 어느 하나의 연산 그래프의 다수의 그래프 요소 중의 하나 이상이, 소정의 트랜잭션에 대해 수행해야 할 연산을 포함하는 지를 판정하기 위한 수단, 각각의 그래프 요소와 연관된 재사용가능한 연산 요소를 포함하는, 연산 그래프의 인스턴스를, 소정의 트랜잭션과 연관시키기 위한 수단, 및 그래프를 실행하여 연산을 수행하기 위한 수단을 포함한다.

[0026] 다른 관점으로서, 일반적으로, 컴퓨터로 판독가능한 매체는, 그래프 기반의 연산을 사용하여 트랜잭션을 처리하기 위한 컴퓨터 프로그램을 기억한다. 컴퓨터 프로그램은, 컴퓨터 시스템으로 하여금, 하나 이상의 일련의 연산

그래프 중의 어느 하나의 연산 그래프의 다수의 그래프 요소 중의 하나 이상이, 소정의 트랜잭션에 대해 수행해야 할 연산을 포함하는 지를 판정하고, 각각의 그래프 요소와 연관된 재사용가능한 연산 요소를 포함하는, 연산 그래프의 인스턴스를, 소정의 트랜잭션과 연관시키며, 그래프를 실행하여 연산을 수행하도록 하기 위한 명령(instructions)을 포함한다.

[0027] 다른 관점으로서, 일반적으로, 그래프 기반의 연산을 처리하기 위한 방법은, 그래프 내에 그래프 성분(graph components)을 나타내는 정점을 포함하여, 정점을 연결(join)하는 링크에 따라 작업 요소(work elements)를 처리하는 단계, 그래프 외부의 프로세스에 에러 정보(error information)를 제공하도록 구성된 하나 이상의 에러 핸들링(error-handling) 그래프 성분을 제공하는 단계, 및 데이터를 처리하는 단계로서, 처리를 행하는 동안 에러를 인식한 그래프 성분에 따라, 작업 요소 중의 적어도 몇몇 작업 요소를, 에러 핸들링 성분을 나타내는 정점에 대한 하나 이상의 링크에 따라 에러 핸들링 성분으로 다이렉트(direct)하는 단계를 포함해서 에러 핸들링 그래프 성분으로 처리를 리다이렉트(redirect)하는 단계를 포함하는 데이터 처리 단계를 포함한다.

[0028] 본 발명의 관점은 다음과 같은 특징들을 하나 이상 포함할 수 있다.

[0029] 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 단계는, 하나 이상의 입력 대기행렬(input queue)로부터 작업 요소를 제거(remove)하는 단계를 포함한다.

[0030] 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 단계는, 에러 핸들링 그래프 성분으로 다이렉트된 작업 요소를 처리하는 단계를 포함한다.

[0031] 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 단계는, 에러가 생기기 전에 만들어진 데이터베이스에 대한 변경을 롤백(rolling back)하는 단계를 포함한다.

[0032] 데이터를 처리하는 단계는, 에러를 핸들링하는 데에 포함되지 않은 그래프 성분에 대하여, 그래프 성분들로 다이렉트된 작업 요소를 폐기하는(discard) 단계를 포함한다.

[0033] 서브 그래프(sub-graph)를 제공하며, 서브 그래프는 서브 그래프의 출력으로서 에러 코드(error code)를 제공하도록 구성된 에러 핸들링 서브 그래프 성분을 포함한다.

[0034] 서브 그래프에 의해 제공된 출력이 서브 그래프에서 에러가 발생했음을 나타내는 경우에, 에러 핸들링 그래프 성분으로 처리가 리다이렉트된다.

[0035] 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 단계는, 에러를 인식한 그래프로부터 에러 핸들링 그래프 성분으로, 에러가 발생했을 때에 그래프 성분이 처리한 작업 요소를 통신하는 단계를 포함한다.

[0036] 작업 요소는 에러 핸들링 성분을 나타내는 정점에 대한 링크에 따라 통신이 행해진다.

[0037] 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 단계는, 에러를 인식한 그래프 성분으로부터 에러 핸들링 그래프 성분까지 에러에 관한 정보를 보고(report)하는 통신을 행하는 단계를 포함한다.

[0038] 정보의 보고는, 에러를 인식한 그래프 성분과 에러 핸들링 성분 사이에서의 암묵적 접속(implicit connection)에 따라 통신이 행해진다.

[0039] 암묵적 접속을, 사용자의 요청에 따라, 에러를 인식한 그래프 성분을 나타내는 정점과 에러 핸들링 성분을 표현하는 정점 사이에 명시적 링크(explicit link)로서 나타낸다.

[0040] 에러 핸들링 그래프 성분을 제공하는 단계는, 다수의 에러 핸들링 그래프 성분을 제공하는 단계를 포함하며, 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 단계는, 에러를 인식한 그래프 성분으로부터 제공된 출력에 기초하여 에러 핸들링 그래프 성분을 선택하는 단계를 포함한다.

[0041] 데이터를 처리하는 단계는, 처리를 행하는 동안 그래프 성분이 에러를 인식하면, 에러의 원인이 된 작업 요소의 식별정보(identification)를 출력하는 단계를 포함한다.

[0042] 데이터를 처리하는 단계는, 그래프의 제1 성분을 인에이블(enable)시키는 단계, 에러 핸들링 성분을 디스에이블(disable)시키는 단계, 및 에러 핸들링 성분 외의 제1 성분의 다운스트림 방향으로의 각각의 성분에 대하여, 성분의 바로 업스트림 방향의 성분이 인에이블되어 있는 경우에, 상기 성분을 인에이블시키는 단계를 포함한다.

[0043] 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 단계는, 인에이블된 각각의 그래프 성분의 실행을 중단(stop)시키는 단계, 에러를 인식한 성분을 디스에이블시키는 단계, 에러 핸들링 성분을 인에이블시키는 단계, 다운스트림 방향의 에러 핸들링 성분이 아닌, 에러를 인식한 성분의 다운스트림 방향의 성분을 디스에이블시키

는 단계, 및 에러 핸들링 성분의 업스트림 방향의 성분을 인에이블시키는 단계를 포함한다.

[0044] 에러 핸들링 그래프 성분으로 연산을 리다이렉트하는 단계는, 에러가 제1 성분에서 발생한 경우에, 에러가 제1 조건하에서 발생하였다면, 제1 성분으로부터 제1 성분의 업스트림 방향으로의 제1 에러 핸들링 성분으로 프로세스 흐름을 다이렉트하는 단계를 포함하고, 에러가 제2 조건하에서 발생하였다면, 제1 성분으로부터 제1 성분의 다운스트림 방향으로의 제2 에러 핸들링 성분으로 프로세스 흐름을 다이렉트하는 단계를 포함한다.

[0045] 제1 조건은 카운터가 한계치 미만인 것이다.

[0046] 제2 조건은 카운터가 한계치를 초과한 것이다.

[0047] 에러 핸들링 그래프 성분으로 처리를 리다이렉트하는 단계는, 에러에 앞서 정해졌던 일련의 그래프 성분을 인에이블시키는 단계를 포함한다.

[0048] 다른 관점에서, 일반적으로, 그래프 기반의 연산을 처리하기 위한 시스템은, 그래프 내에 그래프 성분(graph components)을 나타내는 정점을 포함하여, 정점을 연결(join)하는 링크에 따라 작업 요소(work elements)를 처리하며, 그래프 외부의 프로세스에 에러 정보(error information)를 제공하도록 구성된 하나 이상의 에러 핸들링(error-handling) 그래프 성분을 제공하기 위한 수단, 및 데이터를 처리하는 수단으로서, 처리를 행하는 동안 에러를 인식한 그래프 성분에 따라, 작업 요소 중의 적어도 몇몇 작업 요소를, 에러 핸들링 성분을 나타내는 정점에 대한 하나 이상의 링크에 따라 에러 핸들링 성분으로 다이렉트(direct)하는 수단을 포함해서 에러 핸들링 그래프 성분으로 처리를 리다이렉트(redirect)하는 수단을 포함하는 데이터 처리 수단을 포함한다.

[0049] 다른 관점에서, 일반적으로, 컴퓨터로 판독가능한 매체는, 그래프 기반의 연산을 처리하기 위한 컴퓨터 프로그램을 기억한다. 컴퓨터 프로그램은, 컴퓨터 시스템으로 하여금, 그래프 내에 그래프 성분을 나타내는 정점을 포함하여, 정점을 연결하는 링크에 따라 작업 요소를 처리하며, 그래프 외부의 프로세스에 에러 정보를 제공하도록 구성된 하나 이상의 에러 핸들링 그래프 성분을 제공하고, 및 데이터의 처리를 위해, 처리를 행하는 동안 에러를 인식한 그래프 성분에 따라, 작업 요소 중의 적어도 몇몇 작업 요소를, 에러 핸들링 성분을 나타내는 정점에 대한 하나 이상의 링크에 따라 에러 핸들링 성분으로 다이렉트(direct)하는 것을 포함해서 에러 핸들링 그래프 성분으로 처리를 리다이렉트(redirect)하도록 하는 명령을 포함한다.

[0050] 본 발명의 다른 특징과 장점은 이하의 상세한 설명과 특허 청구의 범위로부터 명백하게 될 것이다.

도면의 간단한 설명

[0051] 도 1은 그래프 기반의 연산의 예를 나타내는 도면이다.

도 2는 작업 흐름을 처리하기 위한 시스템의 논리 블록도이다.

도 3a는 각각의 작업 흐름을 처리하기 위한 플로차트이다.

도 3b는 에러를 핸들링하기 위한 플로차트이다.

도 4a, 도 4b, 도 5, 도 6은 에러-핸들링 그래프의 예이다.

발명을 실시하기 위한 구체적인 내용

[0052] 1. 개요

[0053] 본 출원은 "Startup and Control of Graph-Based Computation"이란 명칭으로 2002년 10월 10일 출원된 미국특허출원 10/268,509호, 미국특허출원 10/268,509호의 계속 출원으로서 "Transactional Graph-Based Computation"이란 명칭으로 2007년 4월 10일 출원된 미국특허출원 11/733,579호에 관한 것으로서, 상기 2개의 특허문헌 모두 본 명세서에서 참조에 의해 원용한다.

[0054] 이하에 설명하는 시스템은 연산 그래프(computation graph)에 관하여 정의된 연산을 실행하기 위한 방법을 구현한다. 도 1을 보면, 연산 그래프(100)의 예는, 단방향성 링크(unidirectional links)(120)에 의해 연결된 다수의 정점(vertex)(110)을 포함한다. 도 1에 나타낸 예에서, 정점(110)은 1부터 6까지의 번호가 매겨져 있으며, 링크(120)도 1부터 6까지의 번호가 부여되어 있다. 연산 그래프(100)는, 트랜잭션 처리 시스템과 연관된 연산 그래프에 따라 처리되는 개별 트랜잭션(individual transactions)과 같은 일련의 직업 요소(work elements: 130)로 이루어진 작업 흐름(work flow)을 처리한다. 트랜잭션은 다수의 작업 요소를 포함하여 이루어질 수 있다. 각각의 정점은, 연산 중의 전체적인 연산 그래프에 의해 정의되는 부분과 연관되어 있다. 이 예에서, 정

점(1)은, 하나 이상의 트랜잭션과 연관된 일련의 최초의 작업 요소(130)에 관한 기억장소에 대한 액세스를 제공하며, 일련의 작업 요소를 출력 링크(1)를 통해 보낸다. 각각의 정점과 연관된 연산을 수행하는 프로세스는, 작업 요소(130)를 처리하고, 전형적으로는 해당 정점의 하나 이상의 출력 링크에 작업 요소를 생성한다.

[0055] 정점에 대한 프로세스는, 적어도 하나의 작업 요소가 각 정점의 입력에 대기행렬로 대기(queue)하게 되면, 실행할 준비가 된 것이다. 도 1에 나타낸 바와 같이, 작업 요소(130)는 링크(1)를 통해 전달되어, 정점(3)에서의 처리를 위해 하나의 작업 요소가 대기행렬로 대기하고, 정점(4)에서의 처리를 위해 2개의 작업 요소가 대기행렬로 대기한다. 이에 따라, 정점(3)과 정점(4)에서, 대기행렬로 대기하고 있는 작업 요소를 처리하는 프로세스를 실행할 준비가 된 것이다. 도시된 바와 같이, 정점(5)은 자신의 입력 중 하나, 즉 링크(4)에서 대기행렬로 대기하고, 다른 입력, 즉 링크(5)에서는 대기행렬로 대기하지 않는다. 따라서, 정점(5)과 연관된 프로세스는 실행할 준비가 되지 않은 것이다.

[0056] 다른 예로서, 작업 흐름에는, 다수의 트랜잭션으로부터의 작업 요소가 포함될 수 있다[즉, 하나 이상의 작업 요소의 제1 세트는 제1 트랜잭션에 대응하고, 하나 이상의 작업 요소의 제2 세트는 제2 트랜잭션에 대응한다]. 트랜잭션은, 세트(set)로서 처리될 모든 행동(actions)을 나타내는 작업 요소 세트를 포함함으로써, 하나의 행동이 실패하면, 아무것도 실행해서는 안 된다. 다수의 트랜잭션을 처리하기 위해 많은 예의 그래프가 사용될 수 있으며, 재사용가능한(reusable) 연산 요소[예를 들어, 운영 체제 프로세스]를 가진 그래프 성분의 연산을 수행함으로써, 필요에 따라, 개별적인 그래프 성분[연산 그래프의 정점으로 나타냄]의 많은 인스턴스(instance)가 만들어질 수 있다. 여러 상이한 트랜잭션을 여러 상이한 각각의 그래프의 인스턴스와 연관시킴으로써, 다수의 트랜잭션을 동시에 처리할 수 있다. 다수의 연산 요소를 그래프 인스턴스에 필요에 따라 할당(assign)할 수 있도록 함으로써, 연산 요소가 하나의 그래프 인스턴스에 의해 사용되도록 하고 다른 그래프에 의해 재사용되도록 하여, 효과적인 자원 할당이 실현될 수 있다. 이에 대해서는, 나중에 구체적으로 설명한다.

[0057] 도 2를 참조하면, 트랜잭션을 포함하는 작업 흐름을 처리하는 연산 그래프를 실행시키기 위한 시스템(200)은, 기억된 그래프 데이터 구조(210)를 포함한다. 이들 데이터 구조는, 그래프의 정점과 링크의 특성을 포함하는 연산 그래프의 사양(specifications)을 포함한다. 이들 데이터 구조의 부분은 전체 그래프를 로딩하지 않고도 액세스가 가능하다. 예를 들어, 새롭게 생성한 그래프 성분에 작업 요소를 할당하기 위해, 개별적인 그래프 성분의 사양이 로딩될 수 있다.

[0058] 시스템의 트랜잭션 서브스크립션(transaction subscription) 모듈(220)은, 기억된 그래프 데이터 구조(210)에서 규정한, 대응하는 연산 그래프를 사용하는 특정의 작업 흐름(232)을 처리하기 위한 커맨드를 포함하는, 그래프 성분[예를 들어, 도 10의 정점(10)으로 표현된 성분과 같은, 작업 요소를 반드시 처리하지 않고도 커맨드를 제공하는 성분]을 서브스크라이브(subscribe)하는 트랜잭션으로부터 제어 입력(222)을 수신한다. 트랜잭션 서브스크립션 모듈(220)은, 특정의 트랜잭션에 할당될 그래프 인스턴스를 인스턴스화(instantiate)하는 데에 사용할 수 있는 그래프 연산 처리 리소스(230)를 기록한다. 트랜잭션 서브스크립션 모듈(220)은, 그래프 연산 처리 리소스(230)를 사용하여 그래프 인스턴스를 어떻게 인스턴스화할지를 판정하기 위해 연산 그래프의 사양을 이용하는 스케줄러(scheduler)를 포함한다. 이 스케줄러는, 일반적으로 다수의 프로세스[또는 프로세스의 풀(pool)]로 이루어져 있으며, 스케줄러 내에서, 각 프로세스는 그래프 인스턴스 내의 소정의 그래프 성분을 인스턴스화하는 재사용가능한 연산 요소로서 작용한다. 그래프의 성분의 연산을 수행하기 위해 실행되는 프로세스는, 외부 데이터 및 프로세스(240)를 사용할 수 있다. 이 외부 데이터 및 프로세스는, 데이터베이스 엔진, 데이터 기억장치, 또는 연산 그래프의 정점과 연관된 처리 과정 중에 액세스되는 다른 모듈을 포함한다. 다른 예에서, 단일의 프로세스 또는 다수의 상이한 동작을 수행할 수 있는 일련의 프로세스는, 해당 인스턴스의 모든 동작을 처리하기 위한 그래프의 소정의 인스턴스에 한정된다.

[0059] 다른 예에서, 트랜잭션 서브스크립션 모듈(220)의 스케줄러는, 원격 프로시저 호출(RPC: remote procedure call) 프로세스를 사용한다. 스케줄러가 소정의 트랜잭션에 대한 작업 요소를 수신하면, 해당 작업 요소를, 트랜잭션과 연관된(즉, 트랜잭션에 할당된) 그래프 인스턴스의 적절한 성분에 할당한다. 해당 그래프 인스턴스에 할당된 프로세스는 해당 성분의 연산을 실행한다. 작업 요소와 연관된 데이터는, 그래프 인스턴스에 의해 이용가능하고 프로세스에 의해 액세스가능한 임시 장소에 기록된다. 트랜잭션 서브스크립션 모듈(220)이 해당 성분에 의해 행해졌다는 통보를, 스케줄러가 받으면, 스케줄러는 실행을 위한 모든 다운스트림(downstream) 그래프 성분을 스케줄링한다. 결국, 트랜잭션은 그래프 전체를 통해 진행될 것이며[그래프는 그래프 연산 처리 리소스(230)를 사용하여 실행됨], RPC 퍼블리시 프로세스(publish process)에 의해 출력될 것이다. 이 과정은 임시 장소에 축적된 데이터를 취하고, 취한 데이터를 적절한 출력 채널, 예컨대 도 1의 데이터베이스 출력(6)에 커밋(commit)한다. RPC 퍼블리시 프로세스는, PRC 서브스크라이브 프로세스(subscribe process)에 의해 다중화

(multiplex)되어, 트랜잭션에 최초에 수신되었던 소켓(socket)을 액세스할 수 있게 된다.

[0060] 일반적으로, 여러 상이한 트랜잭션이 각각 상이한 그래프 인스턴스에 의해 동시에 처리될 수 있다. 시스템(200)은, 트랜잭션 서브스크립션 모듈(220)을 통해, 연산 그래프의 인스턴스에 대한 리소스를 각 트랜잭션에 할당하고, 그래프 연산 처리 리소스(230)를 통해, 작업 흐름을 처리하기 위한 이들의 실행을 제어한다.

[0061] 2. 그래프 데이터 구조 (graph data structures)

[0062] 시스템(200)은, 제한된 리소스를 효율적으로 공유하는 것뿐만 아니라, 그래프 연산의 기동(startup)을 신속하게 하는 많은 특징을 포함한다.

[0063] 연산 그래프의 인스턴스에 의해 트랜잭션을 처리하기 전에, 트랜잭션 서브스크립션 모듈(220)은 기능적으로 공유된 메모리 내의 그래프 인스턴스에 대하여 런타임(runtime) 데이터 구조를 생성한다. 일실시예에서, 단일의 공유 메모리 세그먼트가 생성되는데, 그 내부에, 그래프 인스턴스에 대한 모든 런타임 데이터 구조가 생성된다.

[0064] 트랜잭션에 한정된 하나의 프로세스 또는 여러 프로세스는, 그래프의 정점과 연관되어 있으며, 이들 프로세스의 각각은, 공유 메모리 세그먼트를 어드레스 공간에 매핑(map)한다. 이들 프로세스는 그래프 인스턴스가 개별 트랜잭션에 대해 생성될 때 정점과 연관될 수 있으며, 또는 개별 그래프 성분의 인스턴스가 생성되거나 실행될 때까지 정점과 연관되지 않을 수도 있다. 프로세스는 트랜잭션을 처리하는 동안, 그래프 인스턴스에 대한 런타임 데이터 구조로부터 작업 요소를 판독(read)하고 런타임 데이터 구조에 작업 요소를 기입(write)한다. 즉, 그래프를 통해 전달되는 트랜잭션에 대한 데이터가 하나의 성분에서 다른 성분으로 전달되고, 하나 이상의 프로세스가 트랜잭션에 한정되어 있는 경우에는, 공유 메모리 세그먼트 내의 런타임 데이터 구조를 통해, 하나의 프로세스에서 다른 프로세스로 전달된다. 그래프의 각 성분에 대해 액세스가능한 메모리 공간 내의 소정의 트랜잭션에 대한 데이터를 포함하고, 각 성분을 해당 프로세스 또는 일련의 프로세스에서 실행함으로써, 성분들 사이에서 상태(state)가 공유될 수 있다. 다른 장점들 중에서, 이러한 구성에 의하면, 트랜잭션이 성공적으로 실행되었다는 것을 확인한 후에, 트랜잭션에 대한 성분을 실행하는 것과 연관된 모든 데이터베이스 동작이 한 번에 커밋(commit)될 수 있다.

[0065] 3. 프로세스 풀(process pools)

[0066] 앞서 언급한 바와 같이, 그래프 인스턴스(graph instance)의 성분을 실행하기 위한 그래프 연산 처리 리소스(230)는, 스케줄러에 의해 관리되고 할당된 프로세스 풀(process pools)을 사용하여 구현될 수 있다. 여러 상이한 타입의 연산의 각각에 대해, 프로세스 풀은, 해당 타입의 연산을 필요로 하는 그래프 성분을 사용하여 트랜잭션의 작업 흐름의 처리를 시작하기 전에 생성된다. 트랜잭션이 그래프 인스턴스에 할당될 때에, 그래프 인스턴스의 소정의 성분에 대한 연산을 수행하기 위해 특정 타입의 연산이 필요하게 되면, 스케줄러는 그래프 인스턴스에 의한 그리고 소정의 성분과의 사용을 위한 프로세스 풀의 멤버(member)를 할당한다. 프로세스 풀의 멤버가 트랜잭션의 처리 과정 동안 그래프 인스턴스와 계속 연관되며, 동일한 타입의 성분을 필요로 하는 그래프 인스턴스 내의 다른 성분에 대해 재사용될 수 있다. 이 프로세스는, 해당 타입의 성분을 필요로 하는 해당 트랜잭션에 대한 그래프 인스턴스 내의 마지막 성분의 업스트림(upstream)을 유지하는 작업 요소가 없으면, 해당 프로세스 풀로 다시 릴리즈될 수 있다. 대응하는 타입의 연산과 각각 연관되는 많은 상이한 프로세스 풀이 있을 수 있다. 풀 내의 프로세스는, 상이한 그래프 인스턴스 내의 소정 타입의 성분을 포함하는, 동일하거나 상이한 그래프 인스턴스 내의 성분과, 예를 들어 하나의 그래프 인스턴스 내의 여러 상이한 성분에 대해 사용될 수 있다.

[0067] 다른 실시예에서, 프로세스 풀 내의 각각의 프로세스는, 프로세스 풀을 관리(manage)하는, 트랜잭션 서브스크립션 모듈(220)에 의해 유발(invoke)되는 별개의 프로세스[예컨대, UNIX(등록상표) 프로세스]이다. 트랜잭션 서브스크립션 모듈(220)은, 각 프로세스 풀에 대해 별개의 작업 대기행렬(work queue)을 유지한다. 작업 대기행렬 내의 각각의 엔트리(entry)는 프로세스가 연산을 수행할 그래프 인스턴스의 특정의 정점을 식별한다.

[0068] 몇몇 프로세스는 고정 리소스(fixed resources)를 보존(reserve) 또는 소비(consume)한다. 이러한 프로세서의 예는, Oracle(등록상표) 데이터베이스와 같은 데이터베이스에 접속되는 프로세스이다. 리소스는 각각의 데이터베이스 접속을 형성 및 유지함으로써 소비되기 때문에, 액티브 상태인 이러한 프로세스의 수를 제한하는 것이 바람직하다. 그래프가 데이터베이스를 엑세스하는 다수의 성분을 포함하고 있으면, 소정의 트랜잭션을 위한 모든 데이터베이스 동작은 단일의 데이터베이스 프로세스에서 이루어지도록 하는 것이 바람직할 수 있다. 이를 위하여, 데이터베이스에의 접속을 각각 유지하고, 소정의 그래프 인스턴스가 필요로 할 수 있는 데이터베이스 기능(database function)을 각각 수행할 수 있는 일련의 프로세스를 구축할 수 있다. 소정의 트랜잭션에 그래프 인스턴스가 할당되면, 앞서 설명한 바와 같이, 일련의 프로세스 중의 하나의 프로세스가 전체 트랜잭션을 위한

해당 그래프 인스턴스에 할당되고, 모든 데이터베이스 성분이 해당 프로세스에 다중화된다. 트랜잭션의 작업 요소를 처리하기 위해 데이터베이스에 액세스하기 위한 프로세스를, 정점이 필요로 하게 되면, 해당 정점에 대해, 할당된 프로세스(데이터베이스와의 접속이 이미 구축되어 있음)가 연관된다. 이에 의하여, 해당 데이터베이스에의 접속을 필요로 했던 해당 프로세스의 초기화 단계(initialization step)의 오버헤드(overhead)를 피하게 되며, 소정의 트랜잭션에 대한 모든 데이터베이스 행동이, 해당 프로세스에 의해 처리될 수 있다. 다른 타입의 프로세스도 동일한 방식으로 처리될 수 있다.

[0069] 시스템(200)은, 정점이 프로세스와 연관될 때와 정점에 대한 연산이 개시될 때가 상이한, 정점에 대한 프로세스를 구성(configure)하는 여러 상이한 방식을 지원한다. 한가지 타입의 구성으로서, 프로세스는, 모든 입력된 작업 요소에서의 모든 데이터가 완전히 이용가능하게 될 때까지 정점과 연관되지 않는다. 작업 요소가 크면, 업스트림 정점(upstream vertex)에 의해 계산되고 이용가능하게 될 전체 작업 요소에 대한 시간이 많이 걸릴 수 있다. 이러한 타입의 구성에 의하면, 입력을 대기하고 있는 프로세스가 사용되는 것을 차단(block)함으로써, 해당 그래프 인스턴스 내의 다른 정점에 의해 사용될 수 있도록 한다.

[0070] 다른 타입의 구성은, 스트리밍(streaming) 방식을 사용한다. 프로세스는 정점과 연관되며, 각 입력의 적어도 시작부분이 사용가능하게 될 때에 개시된다. 각 입력의 나머지 부분은, 프로세스가 실행되는 동안에 사용가능하게 된다. 해당 입력이 충분히 신속하게 사용가능하게 되면, 프로세스는 입력에 대한 대기를 차단하지 않는다. 그러나, 입력이 사용가능하게 되지 않으면, 프로세스는 차단될 수 있다.

[0071] 4. 연산 제어(computation control)

[0072] 도 3a는 그래프 인스턴스를 사용하여 각 트랜잭션을 처리하는 프로세스(300)에 대한 플로차트이다. 트랜잭션 서브스크립션 모듈(220)(도 1 참조)이 트랜잭션의 처리에 대한 요청을 받으면, 먼저 연산 그래프(및 이에 대응하는 타입)가 트랜잭션의 처리에 적합한지 여부를 판정한다(단계 305). 예를 들어, 스케줄러는 소정의 연산 그래프가 트랜잭션에 대한 연산을 수행하기에 적합한지(예컨대, 적절한 성분을 포함하는지)를 판정한다. 트랜잭션 자체에서 이것을 특정할 수 있거나, 또는 트랜잭션 서브스크립션 모듈(220)이 특정의 트랜잭션 타입을 특정의 연산 그래프에 연관시키는 데이터에 대한 액세스를 포함할 수 있다. 다음에, 트랜잭션 서브스크립션 모듈(220)은, 해당 트랜잭션을 처리하는 데에 필요한 타입의 연산 그래프의 그래프 인스턴스(필요한 경우)를 생성하고(단계 310), 그 트랜잭션을 그래프 인스턴스에 연관시킨다. 이 프로세스의 일부분으로서, 트랜잭션 서브스크립션 모듈(220)은 그래프 인스턴스에 대한 런타임 데이터 구조의 공유 메모리 세그먼트의 일부를 할당하고, 해당 타입의 연산 그래프에 대한 그래프 템플릿(graph template)을 런타임 데이터 구조에 복제(copy)함으로써, 런타임 데이터 구조를 초기화한다. 그래프 템플릿의 사용 예는, 미국특허 7,167,850호에 구체적으로 개시되어 있으며, 이 특허문헌의 내용을 본 명세서에 참조에 의해 원용한다. 다른 예에서, 그래프 인스턴스는 이미 생성되어 있으며, 이 단계에서는 단지 현재의 트랜잭션에 할당될 뿐이다. 다음으로, 트랜잭션 서브스크립션 모듈(220)은, 이하 구체적으로 설명된 바와 같이, 스케줄러의 제어에 따라, 그래프 인스턴스를 실행한다(단계 320). 그래프 인스턴스는 재사용가능한 각각의 성분과 연관된(할당된) 연산 요소(예를 들어, 프로세스)를 포함한다. 트랜잭션의 전체 작업 흐름이 처리되었으면, 트랜잭션 서브스크립션 모듈(220)은 그래프의 실행 결과를 커밋(commit)하고(예를 들어, 변경사항을 출력 데이터베이스에 커밋함), 할당된 리소스와 연산 요소를 선택적으로 릴리즈(release)하며, 그래프 인스턴스에 대한 런타임 데이터 구조를 삭제(delete)함으로써, 공유 메모리 세그먼트의 일부를 다른 그래프 인스턴스에 대해 재사용할 수 있게 된다(단계 330).

[0073] 5. 변형 예

[0074] 앞서 언급한 바와 같이, 필요로 하게 될 트랜잭션이 있을 것으로 예상하고, 연산 그래프의 미리 인스턴스화된 인스턴스의 그래프 풀을 사전생성(pre-create)하는 것이 가능하다. 트랜잭션을 받고 그래프 인스턴스를 필요로 할 때에, 그 그래프 인스턴스를 그래프 풀로부터 사용가능하다면, 생성하는 것이 아니라, 풀로부터 할당된다. 이에 의하면, 트랜잭션에 대한 개시 비용(startup cost)을 더 감축할 수 있다. 트랜잭션에 대한 연산이 완료되면, 트랜잭션에 할당하고 임의의 동적으로 할당된 메모리를 자유롭게 하기 전에 변수들(variables)을 이들의 초기값으로 복원(restore)함으로써 리셋된다. 그래프 인스턴스가 리셋된 후에는 풀로 복귀한다.

[0075] 다른 실시예에서, 그래프 풀 내의 그래프 인스턴스의 수는 필요에 따라 증가될 수 있다. 예를 들어, 각 그래프의 인스턴스를 최소 개수로 해도 되고, 필요에 따라 더 많은 수를 생성해도 된다.

[0076] 상기 설명에서, 프로세스는 온디멘드(on-demand) 방식으로 그래프 내의 정점에 할당될 수 있다. 온디멘드 방식에서, 프로세스는, 특정의 그래프 인스턴스와 트랜잭션에 한정(bound)되어 있어도, 해당 정점에 대한 모든 입력

이 사용가능하게 될 때까지 정점과 연관되지 않는다. 다른 방식은, 트랜잭션이 그래프 인스턴스와 연관된 경우에, 프로세스를 정점에 연관시키고, 트랜잭션의 전체 작업 흐름이 처리될 때까지 그 연관을 유지하는 것이다.

[0077] 6. 애플리케이션

[0078] 상기 설명한 타입의 연산 그래프의 한가지 애플리케이션은, 은행 애플리케이션에서의 금융 트랜잭션(financial transactions)의 처리이다. 일반적으로, 여러 상이한 타입의 트랜잭션은 상이한 타입의 연산 그래프를 필요로 한다. 전형적인 연산 그래프는, 트랜잭션을 처리하는 데에 필요한 "백엔드"(backend) 서비스와 한가지 타입의 고객 트랜잭션(customer transaction)의 몇몇 조합과 연관된다. 예를 들어, 트랜잭션은 ATM 리퀘스트(requests), 은행직원(bank teller) 입력, 및 컴퓨터 또는 웹서버 간의 비투비(business-to-business) 트랜잭션이 될 수 있다. 여러 고객들은, 특히 은행들이 통합되고 고객들이 원래의 여러 은행으로부터 연합될 때에, 여러 가지 백엔드 시스템을 가질 수 있다. 이들의 계정(accounts)은, 모든 고객이 매입 은행(acquiring bank)의 고객인 경우에도, 매우 상이한 벡엔드 시스템에서 유지될 수 있다. 따라서, 그래프에서의 상이한 정점은 상이한 트랜잭션을 처리하는 데에 사용될 수 있다. 여러 서비스가 그래프 내의 정점과 연관될 수 있다. 예를 들어, 몇몇 정점은 잔고를 업데이트 하거나, 계좌에 돈을 입금하거나, 자금이 계좌에 유지되도록 계좌 홀드(account hold)를 수행하는 것과 같은 기능과 연관될 수 있다. 몇몇 구현 예에서, 프로세스의 정점으로의 동적인(on-the-fly) 배치(assignment)에 의해, 사용하지 않는 정점에 대한 프로세스를 아이들(idle) 상태로 하는 오버헤드(overhead)를 피할 수 있다.

[0079] 그래프 인스턴스를 트랜잭션 단위(per-transation basis)로 할당하는 장점은, 이에 의해, 그렇지 않았으면 직렬로 처리되었어야 했을, 데이터 스트림의 병렬화(parallelization)가 가능하다는 것이다. 상이한 트랜잭션에 할당된 그래프 인스턴스는 처음과 다른 순서로 종료할 수 있는데, 예를 들어, 제1 트랜잭션이 제2 트랜잭션보다 더 복합한 경우에 그렇게 할 수 있다. 이에 의하면, 직렬화된 시스템이 계속해서 제1 트랜잭션을 처리할 때에, 제2 그래프 인스턴스를 릴리즈할 수 있고, 제3 트랜잭션을 처리하는 데에 사용할 수 있게 된다.

[0080] 7. 에러 핸들링(error handling)

[0081] 그래프 인스턴스를 트랜잭션 단위로 할당하는 장점은, 그래프 인스턴스를 실행할 때의 에러에 기인한 실패(failures)가, 해당 트랜잭션에만 한정되며, 다른 그래프 인스턴스의 동시 처리에 영향을 미치지 않는다는 것이다. 전체 트랜잭션이 완료될 때까지 연산 그래프의 결과를 커밋하는 것을 지연(delay)시킴으로써, 에러가 발생한 경우, 데이터를, 시스템이 트랜잭션의 처리를 시작하기 전의 상태로 "롤백"(rolled-back)시킬 수 있다. 에러는 여러 가지 방식으로 핸들링될 수 있다.

[0082] 다른 실시예에서, "에러 핸들링" 성분은 그래프에 포함된다. 에러 핸들링 성분은, 완료를 위해 그래프를 실행할 필요가 없는 특별한 경우이다. 임의의 정점에서의 성분이 에러를 발생시키는 경우에, 전체 성분을 어보트(abort)하는 것이 아니라, 그래프의 실행을, 에러 핸들링 성분으로 리다이렉트(redirect)한다. 소정의 성분과 에러 핸들링 성분(성분의 출력 포트로부터 에러 핸들링 성분의 입력 포트까지의 작업 흐름을 포함) 간의 명백한 관계를, 예외 흐름(exception flow)이라고 한다. 스케줄러(scheduler)는 연산의 실패한 부분에 해당하는 작업 요소를 그래프 인스턴스로부터 제거하고, 에러 핸들링 성분은 그래프가 에러 메시지를 제공하기 위해 사용할 수 있는 출력을, 제공되는 프로세스로 출력한다. 에러 핸들링 성분은, 그 구현 방법에 따라, 예외 흐름 외의 다른 데이터 입력을 수신할 수 있다.

[0083] 도 3b는 그래프에서 발생하는 에러를 핸들링하고 그래프를 실행하기 위한 과정(350)의 예를 나타내는 플로차트이다. 스케줄러는, 링크에 따라 그래프 성분 내의 작업 요소의 작업 흐름을 처리하고(단계 360), 에러가 그래프 성분 내에서 발생했음을 스케줄러가 인식하게 되면(단계 370), 스케줄러는 에러 핸들링 성분에 대한 처리를 리다이렉트(redirect)한다. 이러한 리다이렉트에 대한 한가지 관점은, 해당 에러 핸들링 성분에 대한 임의의 예외 흐름에 따라 작업 요소를 에러 핸들링 성분으로 다이렉트(단계 380)하는 것이다. 이하 상세하게 설명하는 바와 같이, 예외 흐름의 처리에 의하면, 에러 핸들링 성분은 그래프의 외부 프로세스에, 그래프가 에러가 발생한 트랜잭션의 처리를 시작하기 전에 그래프 처리 상태를 나타내는 에러 정보를 제공할 수 있다(단계 390).

[0084] 그래프 내의 임의의 성분에 대해, 지정된(designated) 에러 핸들링 성분이 있다. 이것은 예외 흐름 출력 또는 다른 그래프 성분으로부터의 다른 에러 데이터 출력을 직접 수신하는 성분이 될 수 있거나, 또는 예외 흐름을 수신하는지 여부에 관계없이, 일련의 성분에 대한 지정된 에러 핸들링 성분으로서 정의될 수 있다. 다른 실시예에서, 예외 흐름은 도 4a 및 도 4b에 도시된 것과 같이 핸들링될 수 있다. 이 예에서, 그래프는 트랜잭션 연산을 수행하도록 지정되고, 서브스크라이브 성분(902)과 퍼블리시 성분(904)이 도시되어 있지만, 논트랜잭션(non-

transactional) 작업 흐름에 대한 그래프에서 동일한 기술이 사용될 수 있다. 도 4a에서, 스케줄러는 그래프(900)를 액티브시킨다. 제1 성분인, 서브스크라이브 성분(902)에서 시작해서, 다운스트림 방향의 임의의 비-예외 경로(non-exception path) 내의 각 성분을 "enabled"(인에이블)로 표시한다. 예외 경로(exception path)는, 예외가 생긴 경우에[예를 들어, 앞서 설명한 바와 같이 에러 핸들링 성분으로 이어지는 예외 흐름], 작업 요소의 흐름이나 다른 에러 데이터만을 수신하는 경로이다. 이것을 인에이블 전달(enablement propagation)이라고 한다. 다른 다운스트림 방향에서의 소정의 성분은, 임의의 입력이 인에이블된 업스트림 성분에 접속된 경우에, 인에이블된다. 즉, 레플리케이트(replicate) 성분(906), 리포맷(reformat) 성분(908), 호출 웹 서비스(call web service) 성분(910), 롤업(rollup) 성분(912), 퓨즈(fuse) 성분(914), 퍼블리시(publish) 성분(904)이 모두 인에이블되지만, 다운스트림 방향에 있으며, 임의의 인에이블 성분으로부터 비-예외 입력 흐름을 수신하지 않는, 에러 핸들러(error handler) 성분(916), 롤백(rollback) 성분(918) 및 에러 로그(error log) 성분(920)은 "disabled"(디스에이블) 상태를 유지한다.

[0085] 에러가 생기면, 스케줄러는 에러가 생긴 성분의 실행을 중단하고, 이미 실행을 하고 있는 임의의 다른 성분으로 하여금 완료하도록 하고, 임의의 관련된 데이터[예를 들어, 완료된 성분의 예외 흐름 출력, 즉 에러가 생긴 성분의 "에러 보고 출력"(error reporting output)]를 에러 핸들링 성분으로 전달(propagate)한다. 예를 들어, 호출 웹 서비스 성분(910)이 에러를 트리거(trigger)하면, 레플리케이트 성분(906)으로부터의 예외 흐름과, 호출 웹 서비스 성분(910)의 리젝트(reject) 포트(921)로부터의 에러 보고 출력이, 에러 핸들링 성분(916)의 입력(922) 및 입력(924)에 각각 제공된다. 에러 보고 출력 포트[그래프(900)에서 몇몇 성분의 바닥 부분의 포트로서 도시됨]는, 예컨대, 에러가 발생한 대상, 에러가 발생한 위치, 및 에러와 연관된 임의의 리젝트된 작업 흐름을 특징짓는 정보를 포함하는 임의의 에러에 관한 정보를 제공하는 데에 사용될 수 있다.

[0086] 본 예에서는, 레플리케이트 성분(906)에 대해 에러를 보고하는 출력 포트가 3개 있다. 리젝트 포트(921)는 에러를 발생시켰을 수도 있는 작업 흐름을 제공하거나, 에러와 관련된 몇몇 방식에 속할 수 있는 작업 요소를 제공한다. 에러 포트(error port)(923)는 에러에 관련된 정보를 설명하는 에러 메시지를 제공한다. 로그 포트(log port)(925)는 에러가 발생된 것을 로깅하는 정보를 선택적으로 제공할 수 있다. 로그 포트(925)는 또한 에러가 발생하지 않은 경우에서도 정상적인 실행 과정 동안의 이벤트에 관한 로그 정보(log information)를 제공할 수 있다. 본 예에서, 리젝트 포트(921)는 해당 포트를 사용할 필요가 있을 수 있는 성분들[예컨대, 호출 웹 서비스 성분(910)]에 접속된 것으로 명확히 도시되어 있다. 그러나, 에러 포트(923)와 로그 포트(925)는 접속된 것으로 명확히 도시되어 있지는 않지만, 에러 핸들링 성분(916)에 암묵적으로 접속(implicit connections)된 것을 알 수 있다. 예를 들어, 포트는 디벨로퍼(developer)에 의해 접속될 수 있으며, 인터페이스 컨트롤을 사용하여 숨겨질 수 있다. 다른 예에서, 시스템은, 디벨로퍼에 의해 오버라이드될 수 있는, 디폴트 에러 핸들링 성분에의 암묵적 접속을 자동으로 판정할 수 있다. 대형 및/또는 복잡한 그래프의 경우, 한 가지 타입 이상의 에러 보고 포트에 대한 이러한 "암묵적 와이어링"(implicit wiring)은, 디벨로퍼에 의해 그래프의 시각적 인식(visual comprehension)을 향상시키는데, 이것은 그래프에 기반을 둔 프로그래밍의 장점 중 하나이다. 다른 예에서, 포트가 다른 성분의 포트에 암묵적으로 접속된 것을 나타내기 위해 시각적 단서(visual cues)가 제공될 수 있다[예를 들어, 아이콘, 음영, 또는 채색한 포트]. 숨겨진 암묵적 작업 흐름 접속의 몇몇 또는 모두는 사용자의 요청에 따라 명시적인 링크(explicit links)로서 보여질 수도 있다[예를 들어, 버튼을 클릭하거나 포트 위를 이동시킴으로써].

[0087] 레플리케이트 성분(906)으로부터 출력된 예외 흐름은, 에러가 발생하기 전에 레플리케이트 동작이 완료된 경우, 입력(922)에 이미 대기행렬로 대기하고 있을 수 있다. 다음에, 스케줄러에 의해, 에러 핸들링 성분(본 예에서는, 도면부호 916)이, 에러 성분(본 예에서는, 도면부호 910)을 디스에이블시키고, 에러 핸들링 성분으로부터 인에이블 전달(enablement propagation)을 수행한다(본 예에서는, 918, 904, 920을 인에이블시킴). 디스에이블된 에러 성분의 다운스트림 방향의 임의의 성분도 또한, 해당 성분을 에러 핸들링 성분의 다운스트림 방향으로의 인에이블된 성분으로부터의 흐름을 수신하지 않는 한, 디스에이블된다(본 예에서는, 912 및 914를 디스에이블시킴). 마지막으로, 인에이블된 성분에 흐름을 제공하는 남은 모든 성분을 인에이블시킨다(본 예에서는, 906 및 902를 인에이블시킴).

[0088] 따라서, 이 과정의 결과는, 도 4b에 "<enabled>"(인에이블됨) 및 "<disabled>"(디스에이블됨)으로 표시되어 있다. 퍼블리시 성분(904)은, 그 출력에 에러 메시지가 있더라도, 에러 핸들러(916)에 의해 트랜잭션이 완료된 후, 흐름으로 다시 접속된다. 현재 디스에이블된 성분, 예컨대 리포맷 성분(908)로부터의 출력으로 이미 전달된 데이터는 폐기(discard)된다.

[0089] 앞서 설명한 바와 같이, 데이터는 예외 흐름의 일부로서 또는 다른 성분의 에러 보고 출력의 일부로서 에러 핸

들링 성분으로 흐를 수 있다. 에러가 발생하기 전에 이용가능한 데이터, 예를 들어 도 4b의 레플리케이트 모듈(906)로부터의 출력 데이터는, 그것이 한번이라도 있는 경우, 필요로 할 때까지, 에러 핸들러(916)에 대한 입력 대기행렬에 에스크로(escrow)로 유지된다. 그래프가 에러 없이 완료되면, 에러 핸들러(916)는 전혀 액티브되지 않으며, 데이터는 폐기된다. 에러가 발생하면, 에러 핸들러(916)는 응답을 하기 위해 수신한 모든 입력 데이터를 사용한다. 도 4b에 나타낸 것과 같은 다른 예에서는, 롤백 성분(918)이 사용된다. 레플리케이트 성분(906)으로부터의 입력 데이터는, 그래프가 트랜잭션의 처리를 시작하기 전에 어떤 상태였는지 에러 핸들러(916)에게 보고한다. 에러 핸들러(916)는 이것을 롤백 성분(918)에 출력하고, 롤백 성분(918)에서는, 다른 성분에 의해 변경된 임의의 데이터를 트랜잭션을 실행하기 전의 상태로 복원하는 데에 사용한다. 다음으로, 예외 흐름은, 에러를 로그하는 에러 로그(920)와 퍼블리시 성분(904)으로 진행함으로써, 에러가 보고될 수 있도록 하고, 그래프(900)로 전달된 모든 상위 레벨에 의해 적절히 핸들링되도록 할 수 있다. 임의의 성분으로부터 에러 핸들러(916)로의 예외 흐름에는 데이터가 포함될 수도 있다. 호출 웹 서비스 성분(910)으로부터의 에러 출력이나 임의의 다른 성분(도시 안 함)으로부터의 예외 흐름과 같이, 레플리케이트 성분(906)으로부터의 최초 데이터(original data) 외에, 에러 핸들러(916)에 대한 입력이 존재하면, 보다 구체적인 에러 메시지를 에러 로그 또는 퍼블리시 성분에 형성하는 데에 사용될 수 있다.

[0090] 도 5에 나타낸 것과 같은 다른 예로서, 그래프는, 자신의 에러 핸들링 성분(952)을 각각 가질 수 있는 서브 그래프(950)와 같은 서브 그래프로서 구현된 정점을 포함한다. 따라서, 하위 "그래프 레벨"이 서브 그래프인 정점을 갖는, 최상단 레벨 그래프를 가진 서브 그래프의 계층이 있을 수 있다.

[0091] 서브 그래프(950)의 임의의 성분(954, 956, 958, 960, 962)에서 에러가 발생하면, 프로세스 흐름은, 서브 그래프 에러 보고 포트(974)에 에러 보고 출력을 제공하는 에러 핸들링 성분(952)으로 경로 설정된다. 에러 핸들링 성분(952)의 범위(scope)는 서브 그래프(950)이다. 에러 핸들링 성분은, 다른 그래프 요소[예컨대, 요소(954)]로부터의 예외 흐름 또는 스스로 내포된(nested) 서브 그래프가 될 수 있는 다른 그래프 요소[예컨대, 요소(958)]의 에러 출력(959)으로부터의 출력을 수신하는 입력(966, 968)을 가질 수 있다. 다른 예에서, 에러 핸들링 성분이 다수의 입력을 갖는다면, 가장 최근에 데이터를 수신한 입력만 사용된다. 서브 그래프(950)의 모든 성분이 자신의 동작을 성공적으로 완료하면, 출력(작업 흐름)이 정상적인 서브 그래프 출력 포트(970)로 제공되고, 서브 그래프(950) 이후의 프로세스 흐름이 계속 정상적으로 된다. 에러가 발생하면, 에러 흐름 출력(972) 또는 에러 보고 출력(974)에서 핸들링 및 보고될 수 있다. 다른 예에서는, 에러가 표준 출력(970)으로 보고될 수도 있다.

[0092] 서브 그래프가 에러 핸들링 기능을 갖지 않는다면, 에러가 에러 핸들링 기능을 갖는 그래프 레벨에 도달할 때까지, 그 일부를 이루는 서브 그래프의 계층에서 상방으로 에러가 흐르고, 이 시점에서, 해당 레벨의 에러 핸들링 성분이 액티브된다.

[0093] 에러 핸들링 성분의 입력에서 에스크로된 데이터는 작업 흐름의 서브세트가 될 수 있으며, 트랜잭션과 연관된 모든 데이터가 될 수 있거나, 전체 데이터 흐름이 될 수 있다. 에러 핸들링 성분이 에러 출력 포트를 갖는 경우, 에러를 발생시킨 레코드나 에스크로된 데이터에 기초한 다른 에러 정보 또는 에러가 생긴 성분으로부터 수신한 입력을 출력할 것이다. 에러 핸들링 성분이, 이러한 포트를 갖지 않는다면, 그 출력 포트에 정상적인 출력으로서 오펜딩 레코드(offending record)를 출력하기만 하면 된다.

[0094] 서브 그래프가 에러 핸들링을 갖지 않는 경우, 해당 성분에서의 에러는, 에러가 에러 핸들링 기능을 갖는 그래프 레벨에 도달할 때까지, 그 일부를 이루는 서브 그래프의 계층에서 상방(upwards)으로 흐른다. 이 시점에서, 해당 레벨의 에러 핸들링 성분은 적절한 입력을 수신하고 적절한 에러 출력을 생성한다.

[0095] 에러 핸들링에 의해, 그래프 기반의 연산 프로세싱에서 정상적으로 회피하게 되는 순환적인 그래프 방식이 가능하게 된다. 예를 들어, 도 6에 나타낸 것과 같이, 그래프(1100)에서는, 에러 핸들러(1104)로부터 다운스트림 방향으로 연산 성분(1112)으로부터의 에러 출력(1116)이 해당 에러 핸들러(1104)로 다시 흐른다. 에러 핸들러(1104)는, 도 4a에 나타낸 바와 같이, 서브스크라이브 성분(1102)으로부터 입력을 수신하고, 롤백 성분(1106)에 출력을 제공한다. 롤백 성분(1106)은, 서브스크라이브 성분(1102)에 의한 에러 핸들러(1104)에 대한 데이터 입력에 기초하여, 데이터를, 실패한 연산이 시도되기 전의 상태로 되돌린다. 카운터(counter) 성분(1108)은 롤백 성분(1106)으로부터의 흐름을 수신하고, 수집 성분(1110)에 대한 흐름을 되돌리기 전의 값을 증분시킬 수 있다. 연산 성분(1112)은 카운터 성분(1108)으로부터 입력된 값을 여러 가지 방식으로 사용할 수 있다. 연산 성분은 해당 연산을 수행하기 전의 값을 참고(consult)할 수 있다. 예를 들어, 연산 성분이 자신의 동작에 관하여 어떤 것을 변경하여야 하는 지를 참고할 수 있다. 연산 성분은, 몇몇 임계의 시도 횟수가 이루어졌는지 여부를 알기

위해, 에러 이후의 카운터를 참고할 수 있다. 임계값이 초과되었으면, 에러 출력을 출력(1116)을 통해 다시 에러 핸들러(1104)로 되돌리지 않고, 해당 출력을 제2 에러 핸들러(1120)로 이어지는 제2 에러 출력(1118)으로 보낸다. 카운터 성분이 사용되지 않는 경우, 순환(cycle)을 해제(break)하고 그래프가 결국 완료되는 것을 보장하기 위해 몇몇 다른 기술이 사용될 수 있다.

[0096] 순환적인 그래프의 특징이 명확하게 되는(well-defined) 것을 보장하기 위해, 에러에 대해 인에이블되는 일련의 요소를, 앞서 설명한 바와 같이 필요에 따라 수행하는 것이 아니라, 그래프의 토폴로지(topology)에 기초하여 미리 정한다.

[0097] 다른 예로서, 에러 핸들링을 정확하게 수행하는 것을 보장하기 위해 다른 규칙이 사용된다. 예를 들어, 다른 구현 예로서, 에러 핸들링은 그래프 내의 하나의 성분의 하나의 예외 포트에 대해서만 트리거될 수 있다(동시에 발생하는 임의의 에러는 무시될 수 있다). 그래프 성분 또는 서브 그래프가, 에러 핸들링 성분에 링크되어 있으면, 임의의 에러에 대한 해당 성분을 사용하여야 한다. 그래프 성분 또는 서브 그래프가 에러 핸들링 성분에 링크되어 있지 않으면, 에러는 현재의 범위에 대해 일반적인 에러 핸들러에 의해 처리되어야 한다. 각각의 그래프 성분은 통상적으로 정확히 하나의 에러 핸들러와 연관된다. 이러한 규칙은 시스템의 요건에 따라 변경 또는 조합될 수 있다. 이러한 규칙은 각 트랜잭션에 대한 프로세스의 빈틈없는 제어가 필요한 경우에 유용할 수 있다.

[0098] 다른 예로서, 에러가 발생하면, 운영 체제는, 어떤 에러 핸들링 성분이, 에러가 생긴 성분과 연관되어 있는지를 판정하고, 해당 에러 핸들링 성부에 대하여 어떤 입력 흐름이 사용되어야 하는지를 판정한다. 다수의 입력이 존재한다면, 가장 최근에 데이터가 기입된 입력을 사용한다.

[0099] 에러 핸들링이, 앞서 설명한 바와 같이, 성분 또는 서브 그래프가 자신들의 에러를 핸들링하고 에러와 관련해서 진단 또는 작업을 하기 위해 다른 성분에 의해 사용될 수 있는 에러 코드를 생성하는 경우에는 액티브(active) 상태가 될 수 있으며, 그렇지 않으면 패시브(passive) 상태가 될 수 있다. 패시브 시스템에서, 에러가 생긴 그래프는 실패하게 되고, 운영 체제로 하여금, 예컨대 프로세스를 디버깅하기 위한 스택 덤프(stack dump)를 제공함으로써, 에러 핸들링을 제공하도록 한다.

[0100] 그래프의 각각의 성분은 에러에 개입하여 처리하기 위해 그래프로부터 특별한 요청을 필요로 하지 않는 스케줄러에 암묵적으로 접속된다. 스케줄러는 그래프 인스턴스로부터 에러에 관련된 데이터를 제거할 수 있으며, 다른 예에서는, 에러의 특성을 알 필요가 없다. 몇몇 경우에, 스케줄러는 그래프에 할당된 리소스를 단계별로 그들 각각의 풀로 돌려보낼 수 있으며, 이에 의해 그래프는 에러에 의해 영향을 받지 않았던 작업 요소의 처리를 완료할 수 있게 된다.

[0101] 8. 구현 (implememtation)

[0102] 본 발명은 하드웨어나 소프트웨어, 또는 이들의 조합(예컨대, 프로그램가능한 로직 어레이)에 의해 구현될 수 있다. 달리 특정하지 않는 한, 개시한 알고리즘은 임의의 특정 컴퓨터나 다른 장치에만 국한되는 것이 아니다. 특히, 다양한 범용의 머신이, 본 명세서에 따라 작성된 프로그램과 함께 사용되거나, 특정의 기능을 수행하기 위해 보다 특화된 장치(예를 들어, 집적 회로)를 구성하는 것이 더 편리하게 될 수 있다. 따라서, 본 발명은 하나 이상의 프로세서, 하나 이상의 데이터 기억 시스템(휘발성 및 불휘발성 메모리 및/또는 기억요소를 포함), 하나 이상의 입력 디바이스 또는 포트, 및 하나 이상의 출력 디바이스 또는 포트를 각각 포함하는 하나 이상의 프로그램된 또는 프로그램가능한(programmed or programmable) 컴퓨터 시스템(분산형, 클라이언트/서버, 또는 그리드와 같은 다양한 구조가 가능함)에서 실행되는 하나 이상의 컴퓨터 프로그램에서 구현될 수 있다. 프로그램 코드는 본 명세서에 개시된 기능을 수행하고 출력 정보를 생성하도록 입력 데이터에 적용된다. 출력 정보는, 주지의 방식에 따라, 하나 이상의 출력 디바이스에 적용된다.

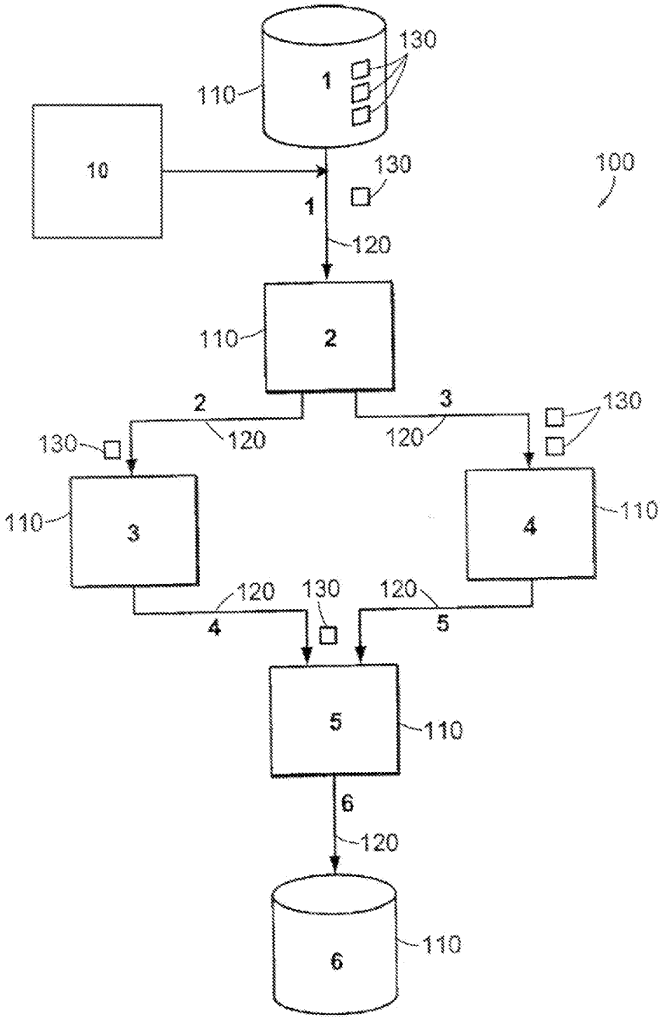
[0103] 이러한 각각의 프로그램은, 컴퓨터 시스템과 통신을 행하기 위해 임의의 바람직한 컴퓨터 언어(머신, 어셈블리, 또는 상위의 절차적, 논리적 또는 객체지향적 프로그래밍 언어를 포함)로 구현될 수 있다. 어떤 경우에나, 언어는 컴파일링 또는 인터프리팅된 언어가 될 수 있다.

[0104] 이러한 각각의 컴퓨터 프로그램은, 기억 매체 또는 디바이스가 컴퓨터 시스템에 의해 판독되어, 본 명세서에 개시된 과정을 수행할 때에, 컴퓨터를 구성하고 동작하는, 범용 또는 전용의 프로그램가능한 컴퓨터에 의해 판독가능한 기억 매체 또는 디바이스(예를 들어, 고체 메모리 또는 매체, 또는 자기, 광학 매체)에 바람직하게 기억 또는 다운로드된다. 본 발명의 시스템은, 컴퓨터 프로그램으로 구성된, 컴퓨터로 판독가능한 기억 매체로서 구현될 수 있다, 이 기억 매체는, 컴퓨터 시스템으로 하여금, 본 명세서에 개시한 기능을 수행하도록 특정되고 미리 정해진 방식으로 동작하도록 구성될 수 있다.

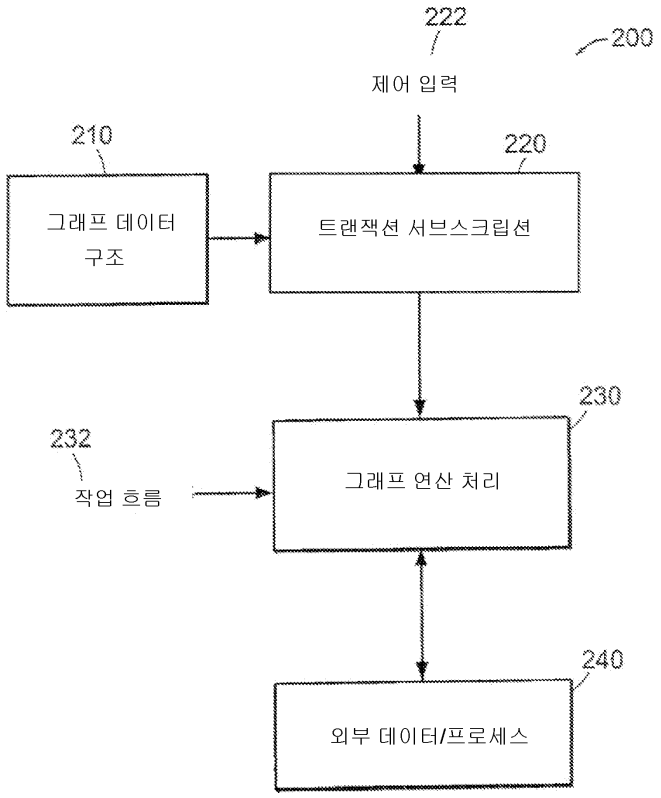
[0105] 이상의 설명은 본 발명의 예시일 뿐으로서 청구범위에 의해 한정되는 본 발명의 범위를 제한하기 위한 것이 아니다. 다른 실시예도 물론 특허 청구범위의 범위 내에 속한다.

도면

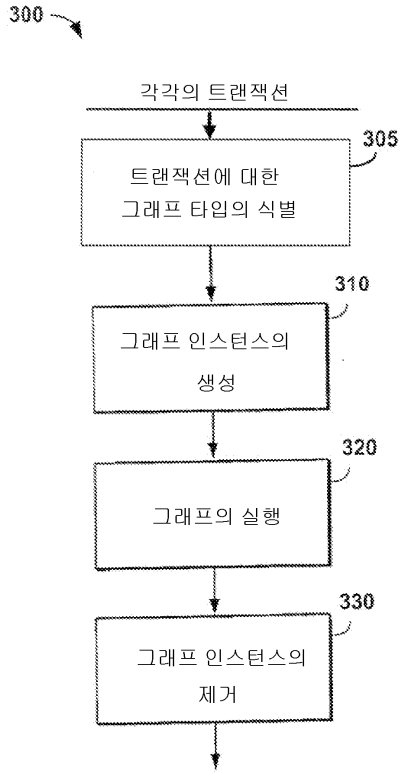
도면1



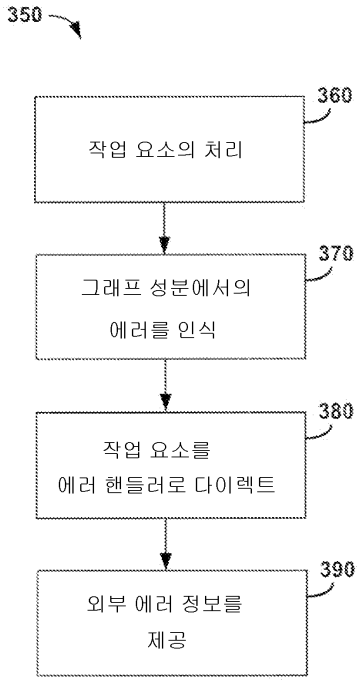
도면2


222
200
제어 입력
210
220
그래프 데이터
구조
트랜잭션 서브스크립션
230
232
작업 흐름
그래프 연산 처리
240
외부 데이터/프로세스

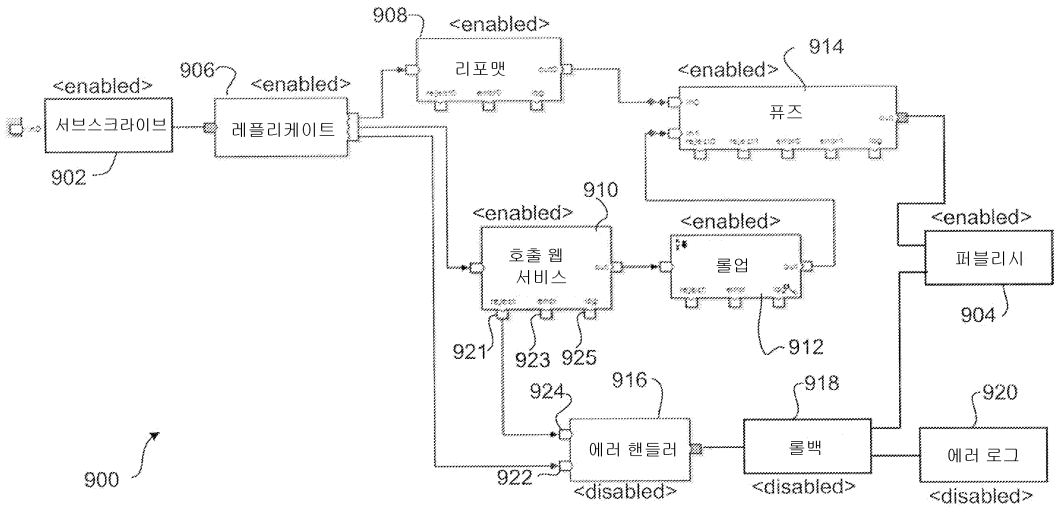
도면3a


300
각각의 트랜잭션
305
트랜잭션에 대한
그래프 타입의 식별
310
그래프 인스턴스의
생성
320
그래프의 실행
330
그래프 인스턴스의
제거

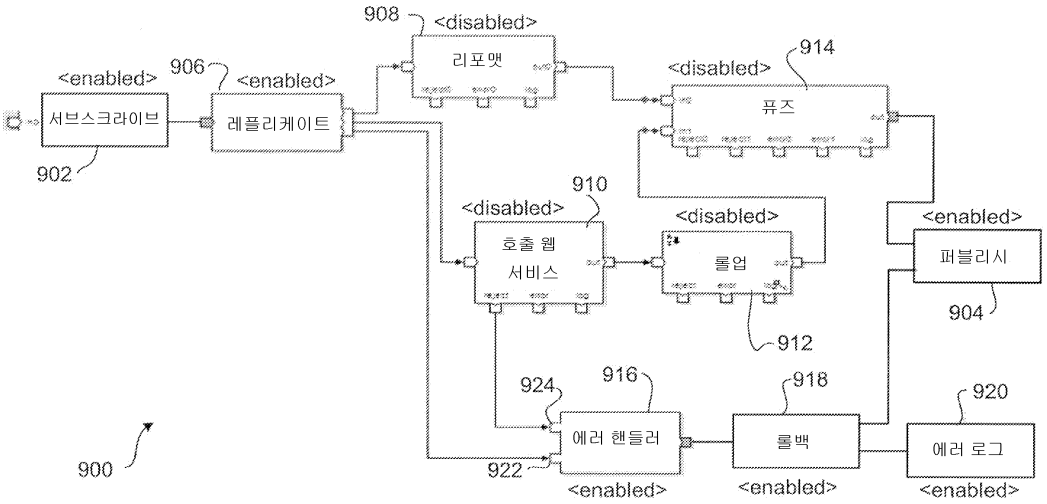
도면3b


350
360
작업 요소의 처리
370
그래프 성분에서의
에러를 인식
380
작업 요소를
에러 핸들러로 다이렉트
390
외부 에러 정보를
제공

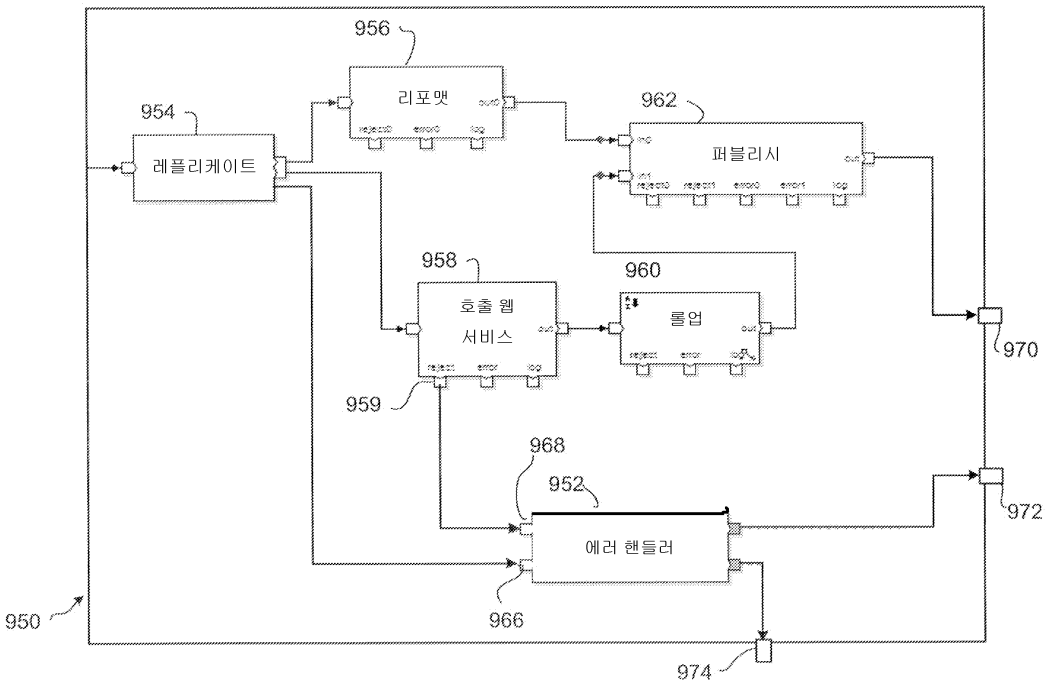
도면4a


908
<enabled>
리포맷
914
<enabled>
퓨즈
906
<enabled>
<enabled>
서브스크라이브
레플리케이트
902
910
<enabled>
호출 웹
서비스
<enabled>
롤업
<enabled>
퍼블리시
904
912
921
923
925
924
916
918
920
에러 핸들러
롤백
에러 로그
922
<disabled>
<disabled>
<disabled>
900

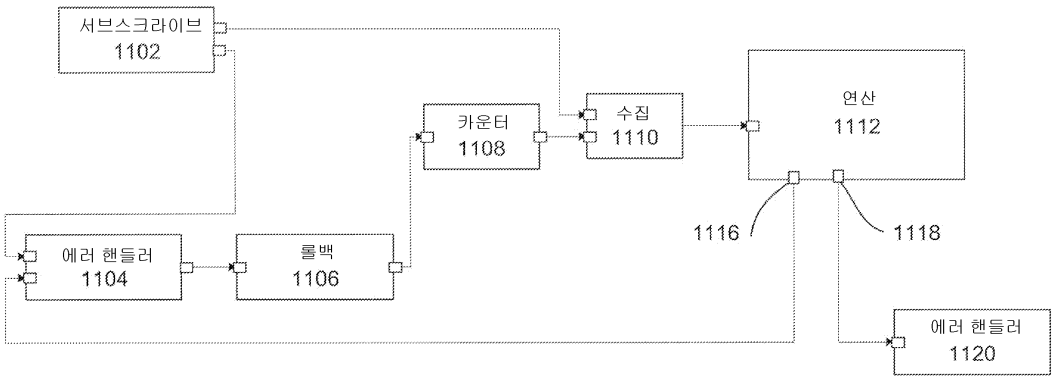
도면4b


908
<disabled>
리포맷
<enabled>
906
<enabled>
서브스크라이브
레플리케이트
902
<disabled>
914
퓨즈
910
<disabled>
호출 웹
서비스
<disabled>
롤업
912
<enabled>
퍼블리시
904
924
916
918
920
에러 핸들러
롤백
에러 로그
922
<enabled>
<enabled>
<enabled>
900

도면5


956
리포맷
954
962
레플리케이트
퍼블리시
958
960
호출 웹
서비스
롤업
970
959
968
952
972
에러 핸들러
950
966
974

도면6


서브스크라이브
1102
카운터
1108
수집
1110
연산
1112
에러 핸들러
1104
롤백
1106
1116
1118
에러 핸들러
1120