(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2010-0061720
(43) 공개일자 2010년06월08일

(51) Int. Cl.
G06F 7/00 (2006.01) *G06F 5/00* (2006.01)
(21) 출원번호 10-2010-7007441
(22) 출원일자(국제출원일자) 2008년09월19일
심사청구일자 없음
(85) 번역문제출일자 2010년04월06일
(86) 국제출원번호 PCT/US2008/076981
(87) 국제공개번호 WO 2009/039352
국제공개일자 2009년03월26일
(30) 우선권주장
60/973,979 2007년09월20일 미국(US)

(71) 출원인
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

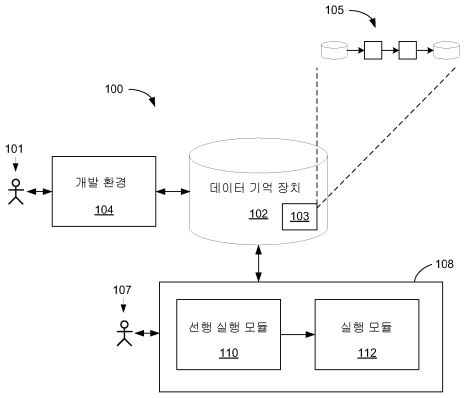
전체 청구항 수 : 총 24 항

(54) 그래프에 기초한 연산에서 데이터 흐름을 관리하는 방법 및 시스템

(57) 요 약

다수의 레벨을 포함하는 계층과 관련된 데이터 요소를 처리하는 방법은, 계층의 레벨과 각각 관련되며, 데이터 요소를 포함하는 다수의 흐름을 형성하는 과정과, 기 위한 수단; 흐름 중의 몇몇 흐름 내에, 계층의 레벨과 관련된 계층 구조 정보를 포함시키는 과정을 포함한다. 하나 이상의 레벨에 대하여, 흐름 중의 둘 이상이면서 모든 흐름보다는 적은 수의 흐름이, 레벨과 관련된 계층 구조 정보를 포함하도록 되어 있다.

대 표 도 - 도1a



특허청구의 범위

청구항 1

다수의 레벨(level)을 갖는 계층(hierarchy)과 관련된 데이터 요소(data element)를 처리하기 위한 방법으로서,

상기 계층의 각각의 레벨과 관련된, 상기 데이터 요소를 포함하여 이루어진 다수의 흐름(flow)을 형성하는 단계; 및

상기 흐름 중의 적어도 몇몇 흐름 내에, 상기 계층의 레벨과 관련된 계층 구조 정보(hierarchical structure information)를 포함시키는 단계

를 포함하며,

상기 레벨 중의 하나 이상의 레벨에 대하여, 상기 흐름 중의 둘 이상이면서 모든 흐름보다는 적은 수의 흐름이, 상기 레벨과 관련된 계층 구조 정보를 포함하도록 된 것을 특징으로 하는 데이터 요소 처리 방법.

청구항 2

제1항에 있어서,

상기 계층 구조 정보는, 하나 이상의 데이터 요소로 이루어진 인접해 있는 세트를 분리시키는 분리 요소(separation element)를 포함하며,

상기 분리 요소는 상기 계층의 레벨과 각각 관련되어 있고,

상기 레벨 중의 하나 이상의 레벨에 대하여, 상기 흐름 중의 둘 이상이면서 모든 흐름보다는 적은 수의 흐름이, 상기 레벨과 관련된 분리 요소를 포함하도록 된, 데이터 요소 처리 방법.

청구항 3

제2항에 있어서,

계층적 데이터 구조 내의 상기 데이터 요소의 표현과 다수의 흐름 내의 상기 데이터 요소의 표현을, 상기 분리 요소에 적어도 부분적으로 기초하여 변환(convert)시키는 변환 단계를 더 포함하는 데이터 요소 처리 방법.

청구항 4

제3항에 있어서,

상기 계층적 데이터 구조는 마크업 언어(markup language)에 따라 포맷되는 것인, 데이터 요소 처리 방법.

청구항 5

제4항에 있어서,

상기 마크업 언어는 확장가능한 마크업 언어(XML)인 것인, 데이터 요소 처리 방법.

청구항 6

제3항에 있어서,

상기 계층적 데이터 구조는 프로그래밍 언어에 따라 포맷되는, 데이터 요소 처리 방법.

청구항 7

제6항에 있어서,

상기 프로그래밍 언어는 PL/I, COBOL, 또는 IMS인 것인, 데이터 요소 처리 방법.

청구항 8

제3항에 있어서,

상기 변환 단계는, 상기 계층적 데이터 구조 내의 데이터 요소의 표현을, 다수의 흐름 내의 데이터 요소의 표현으로 변환시키는 단계를 포함하는, 데이터 요소 처리 방법.

청구항 9

제8항에 있어서,

상기 데이터 요소로 이루어진 다수의 흐름을 형성하는 단계는,

상기 계층적 데이터 구조로부터 제1 레벨과 관련된 제1 데이터 요소를 추출(extract)하는 단계;

상기 계층적 데이터 구조 중에서 상기 제1 데이터 요소에 대응하는 부분에 내포된(nested) 제2 레벨과 관련된 하나 이상의 데이터 요소를 추출하는 단계;

상기 제2 레벨과 관련된 상기 추출한 데이터 요소의 적어도 몇몇을, 상기 제2 레벨과 관련된 흐름에 포함시키는 단계; 및

상기 제1 데이터 요소를 상기 제1 레벨과 관련된 흐름 내에 포함시키고, 상기 제1 레벨과 관련된 대응하는 분리 요소를 상기 제2 레벨과 관련된 흐름 내에 포함시키는 단계를 포함하는, 데이터 요소 처리 방법.

청구항 10

제8항에 있어서,

상기 데이터 요소로 이루어진 다수의 흐름을 형성하는 단계는,

다수의 계층적 데이터 구조의 스트림을 채택(accept)하는 단계;

각각의 계층적 데이터 구조에 대하여,

상기 계층적 데이터 구조로부터 대응하는 레벨과 각각 관련된 데이터 요소를 추출(extract)하는 단계; 및

상기 계층적 데이터 구조 중의 부모(parent) 데이터 요소에 대응하는 부분에 내포된 하나 이상의 자식(child) 데이터 요소로 이루어진 소정 세트에 대하여, 상기 자식 데이터 요소의 세트와 분리 요소를, 상기 자식 데이터 요소에 대응하는 레벨과 관련 흐름에 포함시키며, 상기 부모 데이터 요소를 상기 부모 데이터 요소에 대응하는 레벨과 관련된 흐름에 포함시키는 단계

를 포함하는, 데이터 요소 처리 방법.

청구항 11

제10항에 있어서,

포함된 상기 분리 요소는 상기 부모 데이터 요소에 대응하는 레벨과 관련되어 있는 것인, 데이터 요소 처리 방법.

청구항 12

제3항에 있어서,

상기 변환 단계는, 상기 다수의 흐름 내의 데이터 요소의 표현을, 상기 계층적 데이터 구조 내의 데이터 요소의 표현으로 변환하는 단계를 포함하는, 데이터 요소 처리 방법.

청구항 13

제12항에 있어서,

제1 레벨과 관련된 흐름으로부터 제1 데이터 요소를 수신(receive)하는 단계;

제2 레벨과 관련된 흐름으로부터 제1 레벨과 관련된 분리 요소 및 하나 이상의 데이터 요소를 수신하는 단계;

상기 계층적 데이터 구조의 일부에 상기 제1 데이터 요소를 포함시키는 단계; 및

상기 계층적 데이터 구조 중의 상기 제1 데이터 요소를 포함하는 부분에 내포된 상기 제2 레벨과 관련된 흐름으

로부터 수신된 데이터 요소 중의 적어도 몇몇을 포함하는 단계

를 더 포함하는 데이터 요소 처리 방법.

청구항 14

제12항에 있어서,

상기 데이터 요소 처리 방법은 다수의 계층적 데이터 구조의 스트림을 생성하는 생성 단계를 더 포함하며,

상기 생성 단계는, 각각의 계층적 데이터 구조에 대하여,

소정 레벨과 관련된 데이터 요소를, 상기 소정 레벨과 관련된 흐름으로부터 수신하는 단계;

제1 레벨과 관련된 흐름으로부터 수신된 각각의 데이터 요소에 대하여,

상기 계층적 데이터 구조의 일부에 상기 수신한 데이터 요소를 포함시키는 단계;

다른 레벨과 관련된 대응하는 하나 이상의 데이터 요소로 이루어진 세트가 상기 다른 레벨과 관련된 흐름 내의 분리 요소에 기초하여 존재하는지 여부를 판정하는 단계; 및

상기 대응하는 수신한 데이터 요소를 포함하는 상기 계층적 데이터 구조 내에 내포된 각각의 대응하는 하나 이상의 데이터 요소로 이루어진 세트를 포함하는 단계를 포함하는, 데이터 요소 처리 방법.

청구항 15

제14항에 있어서,

상기 다른 레벨과 관련된 흐름 내의 분리 요소는 상기 제1 레벨과 관련되어 있는 것인, 데이터 요소 처리 방법.

청구항 16

제1항에 있어서,

상기 계층 구조 정보는 상기 계층의 레벨의 다수의 데이터 요소 중의 하나 이상의 데이터 요소를 고유하게 식별하는 적어도 몇몇 데이터 요소에 포함되는 하나 이상의 인덱스 값(index value)을 포함하며,

상기 인덱스 값은 상기 계층의 레벨과 각각 관련되고,

상기 레벨 중의 하나 이상의 레벨에 대하여, 상기 흐름 중의 둘 이상이면서 모든 흐름보다는 적은 수의 흐름이, 상기 레벨과 관련된 인덱스 값을 포함하는, 데이터 요소 처리 방법.

청구항 17

제16항에 있어서,

상기 데이터 요소 중 적어도 몇몇은 상기 계층에 따라 부모 데이터 요소를 고유하게 식별하는 인덱스 값을 포함하는, 데이터 요소 처리 방법.

청구항 18

제16항에 있어서,

계층적 데이터 구조 내의 데이터 요소의 표현과 다수의 흐름 내의 데이터 요소의 표현을, 상기 인덱스 값에 적어도 부분적으로 기초하여 변환하는 단계를 더 포함하는 데이터 요소 처리 방법.

청구항 19

제18항에 있어서,

상기 계층적 데이터 구조는 마크업 언어에 따라 포맷되는 것인, 데이터 요소 처리 방법.

청구항 20

제19항에 있어서,

상기 마크업 언어는 확장가능한 마크업 언어(XML)인 것인, 데이터 요소 처리 방법.

청구항 21

제18항에 있어서,

상기 계층적 데이터 구조는 프로그래밍 언어에 따라 포맷되는, 데이터 요소 처리 방법.

청구항 22

제21항에 있어서,

상기 프로그래밍 언어는 PL/I, COBOL, 또는 IMS인 것인, 데이터 요소 처리 방법.

청구항 23

다수의 레벨을 포함하는 계층과 관련된 데이터 요소를 처리하기 위한 시스템으로서,

상기 계층의 레벨과 각각 관련되며, 데이터 요소를 포함하는 다수의 흐름을 형성하기 위한 수단; 및

상기 흐름 중의 몇몇 흐름 내에, 상기 계층의 레벨과 관련된 계층 구조 정보를 포함시키기 위한 수단

을 포함하며,

상기 레벨 중의 하나 이상의 레벨에 대하여, 상기 흐름 중의 둘 이상이면서 모든 흐름보다는 적은 수의 흐름이, 상기 레벨과 관련된 계층 구조 정보를 포함하도록 된 것을 특징으로 하는 데이터 요소 처리 시스템.

청구항 24

그래프에 기초한 연산의 명세(specification)를 처리하기 위한 컴퓨터 프로그램이 기억되는, 컴퓨터로 판독가능한 매체로서,

상기 컴퓨터 프로그램은, 컴퓨터로 하여금,

상기 계층의 레벨과 각각 관련되며, 데이터 요소를 포함하는 다수의 흐름을 형성하고,

상기 흐름 중의 몇몇 흐름 내에, 상기 계층의 레벨과 관련된 계층 구조 정보를 포함시키도록 하기 위한 명령어를 포함하며,

상기 레벨 중의 하나 이상의 레벨에 대하여, 상기 흐름 중의 둘 이상이면서 모든 흐름보다는 적은 수의 흐름이, 상기 레벨과 관련된 계층 구조 정보를 포함하도록 된 것을 특징으로 하는 컴퓨터로 판독가능한 매체.

명 세 서

기 술 분 야

[0001] 본 발명은 그래프에 기초한 연산에서의 데이터 흐름의 관리에 관한 것이다.

[0002] 관련 출원

[0003] 본 출원은 2007년 9월 20일에 출원된 미국특허출원 60/973,979호의 우선권을 주장한다.

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, 이들 프로세스 간에 그래프의 링크에 따라 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

[0005] 본 발명의 하나의 관점으로서, 일반적으로, 다수의 레벨(level)을 갖는 계층(hierarchy)과 관련된 데이터 요소(data element)를 처리하기 위한 방법은, 계층의 각각의 레벨과 관련된, 데이터 요소를 포함하여 이루어진 다수의 흐름(flow)을 형성하는 단계; 및 흐름 중의 적어도 몇몇 흐름 내에, 계층의 레벨과 관련된 계층 구조 정보(hierarchical structure information)를 포함시키는 단계를 포함한다. 레벨 중의 하나 이상의 레벨에 대하여, 흐름 중의 둘 이상이면서 모든 흐름보다는 적은 수의 흐름이, 레벨과 관련된 계층 구조 정보를 포함하도록 되어 있다.

[0006] 본 발명의 관점은 다음에 설명하는 하나 이상의 특징을 포함한다.

[0007] 계층 구조 정보는, 하나 이상의 데이터 요소로 이루어진 인접해 있는 세트를 분리시키는 분리 요소(separation element)를 포함하며, 분리 요소는 계층의 레벨과 각각 관련되어 있고, 레벨 중의 하나 이상의 레벨에 대하여, 흐름 중의 둘 이상이면서 모든 흐름보다는 적은 수의 흐름이, 레벨과 관련된 분리 요소를 포함하도록 되어 있다.

[0008] 본 발명의 방법은 또한 계층적 데이터 구조 내의 데이터 요소의 표현과 다수의 흐름 내의 데이터 요소의 표현을, 분리 요소에 적어도 부분적으로 기초하여 변환(convert)시키는 변환 단계를 더 포함한다.

[0009] 계층적 데이터 구조는 마크업 언어(markup language)에 따라 포맷된다.

[0010] 마크업 언어는 확장가능한 마크업 언어(XML)이다.

[0011] 계층적 데이터 구조는 프로그래밍 언어에 따라 포맷된다.

[0012] 프로그래밍 언어는 PL/I, COBOL, 또는 IMS이다.

[0013] 앞서 설명한 변환 단계는, 계층적 데이터 구조 내의 데이터 요소의 표현을, 다수의 흐름 내의 데이터 요소의 표현으로 변환시키는 단계를 포함한다.

[0014] 데이터 요소로 이루어진 다수의 흐름을 형성하는 단계는, 계층적 데이터 구조로부터 제1 레벨과 관련된 제1 데이터 요소를 추출(extract)하는 단계; 계층적 데이터 구조 중에서 제1 데이터 요소에 대응하는 부분에 내포된(nested) 제2 레벨과 관련된 하나 이상의 데이터 요소를 추출하는 단계; 제2 레벨과 관련된 상기 추출한 데이터 요소의 적어도 몇몇을, 제2 레벨과 관련된 흐름에 포함시키는 단계; 및 제1 데이터 요소를 제1 레벨과 관련된 흐름 내에 포함시키고, 제1 레벨과 관련된 대응하는 분리 요소를 제2 레벨과 관련된 흐름 내에 포함시키는 단계를 포함한다.

[0015] 데이터 요소로 이루어진 다수의 흐름을 형성하는 단계는, 다수의 계층적 데이터 구조의 스트림을 채택(accept)하는 단계; 각각의 계층적 데이터 구조에 대하여, 계층적 데이터 구조로부터 대응하는 레벨과 각각 관련된 데이터 요소를 추출(extract)하는 단계; 및 계층적 데이터 구조 중의 부모(parent) 데이터 요소에 대응하는 부분에 내포된 하나 이상의 자식(child) 데이터 요소로 이루어진 소정 세트에 대하여, 자식 데이터 요소의 세트와 분리 요소를, 자식 데이터 요소에 대응하는 레벨과 관련 흐름에 포함시키며, 부모 데이터 요소를 부모 데이터 요소에 대응하는 레벨과 관련된 흐름에 포함시키는 단계를 포함한다.

[0016] 포함된 분리 요소는 부모 데이터 요소에 대응하는 레벨과 관련되어 있다.

[0017] 변환 단계는, 다수의 흐름 내의 데이터 요소의 표현을, 계층적 데이터 구조 내의 데이터 요소의 표현으로 변환하는 단계를 포함한다.

[0018] 본 발명의 방법은, 제1 레벨과 관련된 흐름으로부터 제1 데이터 요소를 수신(receive)하는 단계; 제2 레벨과 관련된 흐름으로부터 제1 레벨과 관련된 분리 요소 및 하나 이상의 데이터 요소를 수신하는 단계; 계층적 데이터 구조의 일부에 제1 데이터 요소를 포함시키는 단계; 및 계층적 데이터 구조 중의 제1 데이터 요소를 포함하는 부분에 내포된 제2 레벨과 관련된 흐름으로부터 수신된 데이터 요소 중의 적어도 몇몇을 포함하는 단계를 더 포함한다.

[0019] 본 발명의 데이터 요소 처리 방법은 다수의 계층적 데이터 구조의 스트림을 생성하는 생성 단계를 더 포함하며, 이 생성 단계는, 각각의 계층적 데이터 구조에 대하여, 소정 레벨과 관련된 데이터 요소를, 소정 레벨과 관련된 흐름으로부터 수신하는 단계; 제1 레벨과 관련된 흐름으로부터 수신된 각각의 데이터 요소에 대하여, 계층적 데이터 구조의 일부에 수신한 데이터 요소를 포함시키는 단계; 다른 레벨과 관련된 대응하는 하나 이상의 데이터

요소로 이루어진 세트가 다른 레벨과 관련된 흐름 내의 분리 요소에 기초하여 존재하는지 여부를 판정하는 단계; 및 대응하는 수신한 데이터 요소를 포함하는 계층적 데이터 구조 내에 내포된 각각의 대응하는 하나 이상의 데이터 요소로 이루어진 세트를 포함하는 단계를 포함한다.

[0020] 다른 레벨과 관련된 흐름 내의 분리 요소는 제1 레벨과 관련되어 있다.

[0021] 계층 구조 정보는 계층의 레벨의 다수의 데이터 요소 중의 하나 이상의 데이터 요소를 고유하게 식별하는 적어도 몇몇 데이터 요소에 포함되는 하나 이상의 인덱스 값(index value)을 포함하며, 인덱스 값은 계층의 레벨과 각각 관련되고, 레벨 중의 하나 이상의 레벨에 대하여, 흐름 중의 둘 이상이면서 모든 흐름보다는 적은 수의 흐름이 레벨과 관련된 인덱스 값을 포함한다.

[0022] 데이터 요소 중 적어도 몇몇은 계층에 따라 부모 데이터 요소를 고유하게 식별하는 인덱스 값을 포함한다.

[0023] 본 발명의 방법은, 계층적 데이터 구조 내의 데이터 요소의 표현과 다수의 흐름 내의 데이터 요소의 표현을, 인덱스 값에 적어도 부분적으로 기초하여 변환하는 단계를 더 포함한다.

[0024] 계층적 데이터 구조는 마크업 언어에 따라 포맷된다.

[0025] 마크업 언어는 확장가능한 마크업 언어(XML)이다.

[0026] 계층적 데이터 구조는 프로그래밍 언어에 따라 포맷된다.

[0027] 프로그래밍 언어는 PL/I, COBOL, 또는 IMS이다.

[0028] 본 발명의 다른 관점으로서, 다수의 레벨을 포함하는 계층과 관련된 데이터 요소를 처리하기 위한 시스템은, 계층의 레벨과 각각 관련되며, 데이터 요소를 포함하는 다수의 흐름을 형성하기 위한 수단; 및 흐름 중의 몇몇 흐름 내에, 계층의 레벨과 관련된 계층 구조 정보를 포함시키기 위한 수단을 포함하며, 레벨 중의 하나 이상의 레벨에 대하여, 흐름 중의 둘 이상이면서 모든 흐름보다는 적은 수의 흐름이, 레벨과 관련된 계층 구조 정보를 포함하도록 되어 있다.

[0029] 본 발명의 또 다른 관점으로서, 그래프에 기초한 연산의 명세(specification)를 처리하기 위한 컴퓨터 프로그램이 기억되는, 컴퓨터로 판독가능한 매체는, 컴퓨터 프로그램이, 컴퓨터로 하여금, 계층의 레벨과 각각 관련되며, 데이터 요소를 포함하는 다수의 흐름을 형성하고, 흐름 중의 몇몇 흐름 내에, 계층의 레벨과 관련된 계층 구조 정보를 포함시키도록 하기 위한 명령어를 포함하며, 레벨 중의 하나 이상의 레벨에 대하여, 흐름 중의 둘 이상이면서 모든 흐름보다는 적은 수의 흐름이, 레벨과 관련된 계층 구조 정보를 포함하도록 되어 있다.

[0030] 본 발명의 관점에는 다음에 설명하는 하나 이상의 장점을 포함한다.

[0031] 데이터 요소를 포함하여 이루어진 흐름 내에 계층 구조 정보를 포함시킴으로써, 그래프에 기초한 연산에서의 컴포넌트가, 컴포넌트를 서로 연결시키는 링크를 통해 여러 흐름에서의 데이터 요소들 간의 관계를 인식할 수 있다. 계층적 데이터 구조의 요소는, 데이터 구조 전체의 표현을 메모리에 기억시키지 않고도, 상기 관계를 보존하면서, 데이터 요소 내의 합성 키로부터 또는 분리 요소로부터 추출해낼 수 있다. 합성 키(synthetic key)는 계층으로부터의 관계에 기초하여 데이터 요소를 그룹화하는 것을 포함하여, 데이터 요소에 대해 행해지는 다양한 연산을 용이하게 하는데에 사용될 수 있다. 예를 들어, 합성 키는 계층 관계 중의 적어도 몇몇을 재구성하거나 데이터 요소를 재정렬시키기 위한 결합 연산(join operation)의 키 필드(key field)로서 사용될 수 있다. 흐름 내에 분리 요소를 배치함으로써, 컴포넌트는 최초의 데이터 구조로부터 계층 관계 중의 몇몇을 재구성할 수 있다. 몇몇 계층 구조는 흐름 내의 이산적인 데이터 요소에 의해 암시적으로(implicitly) 표현되기 때문에, 어떤 경우에는 분리 요소를 흐름 내에 선택적으로 삽입하는 것에 의해, 모든 데이터 요소에 합성 키를 삽입하는 것보다 그 계층 구조를 더 효과적으로 표현할 수 있다. 예를 들어, 소정 레벨과 관련된 분리 요소는 하위 레벨의 흐름에만 포함되도록 하고 동일 레벨이나 상위 레벨(하위 레벨이, 루트부터 시작해서 관련된 계층의 레벨에 대응하는 더 깊은 내포 구조를 나타내는)의 흐름에는 포함되지 않도록 할 수 있다. 마찬가지로, 합성 키를 사용하여 계층을 효과적으로 표현하기 위하여, 소정 레벨과 관련된 합성 키를, 해당 레벨의 부모(parent)를 갖는 데이터 요소의 흐름에만 포함되도록 하고, 다른 레벨의 부모를 갖는 데이터 요소를 갖는 흐름에는 포함되지 않도록 해도 된다.

[0032] 이상 설명한 것 외의 본 발명의 다른 특징이나 장점은 이하의 상세한 설명과 청구범위로부터 명백하게 알 수 있을 것이다.

도면의 간단한 설명

[0033] 도 1a는 그래프 기반의 연산을 수행하기 위한 시스템의 블록도이다.

도 1b는 연산 그래프를 나타낸 도면이다.

도 1c는 내포 문서의 계층을 표현하는 트리를 나타낸 도면이다.

도 2, 도 3a 및 4a는 연산 그래프를 나타낸 도면이다.

도 3b는 데이터를 처리하기 위한 과정을 나타내는 플로차트이다.

도 3c, 도 4b, 도 4c, 및 도 4d는 정규화 컴포넌트로부터의 흐름을 나타내는 도면이다.

도 4e는 데이터를 처리하기 위한 과정을 나타내는 플로차트이다.

발명을 실시하기 위한 구체적인 내용

[0034] 도 1a를 참조하면, 그래프 기반의 연산을 실행하기 위한 시스템(100)은 데이터 기억부(102)에 연결된 개발 환경(development environment: 104)과 데이터 기억부(102)에 연결된 런타임 환경(108)을 포함한다. 개발자(101)는 개발 환경(104)을 사용하여 애플리케이션을 구축한다. 애플리케이션은 하나 이상의 연산 그래프(computation graph)와 관련되는데, 이러한 연산 그래프는 개발자가 개발 환경(104)을 사용한 결과로서 데이터 기억부에 기록될 수 있는, 데이터 기억부(102) 내의 데이터 구조에 의해 특정된다. 연산 그래프(105)를 위한 데이터 구조(103)는, 예를 들어 연산 그래프의 정점(컴포넌트 또는 데이터 세트)과 정점들 간의 링크[작업 요소(work element)의 흐름(flow)을 나타냄]를 특정한다. 데이터 구조는 다양한 특징의 컴포넌트, 데이터 세트, 및 컴포넌트 그래프의 흐름["데이터플로 그래프"(dataflow graph)라고도 함]을 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 연산 그래프와 연관될 수 있다.

[0035] 런타임 환경(108)은 UNIX 운영 체제와 같은 적절한 운영 체제의 제어하에서 하나 이상의 범용 컴퓨터에 대하여 호스트 역할을 할 수 있다. 예를 들어, 런타임 환경(108)은 다수의 중앙 처리 장치(CPU), 로컬(예컨대, SMP 컴퓨터와 같은 다중 프로세서) 또는 국부적으로 분산된(클러스터 또는 MPP로서 연결된 다중 프로세서), 원격 또는 원격 분산된(LAN 또는 WAN을 통해 연결된 다중 프로세서), 또는 이들의 조합을 이용하는 컴퓨터 시스템의 구성을 갖는 다중 노드(multiple-node) 병렬 컴퓨팅 환경을 포함할 수 있다. 런타임 환경(108)에 의해 액세스되는 입력, 출력 또는 중간 데이터 세트는 병렬 파일 시스템[데이터 기억부(102), 또는 통신 링크를 통해 국부적으로 또는 원격으로 시스템(100)에 연결된 외부 데이터 기억장치]에 기억되는 병렬의 "다중 파일"(multifile)이 될 수 있다.

[0036] 그래프에서 다수의 컴포넌트를 동시에 실행시키는 것은 한가지 형태의 병렬 특성을 필요로 한다. 추가적인 병렬 특성은 그래프의 여러 컴포넌트를 여러 컴퓨팅 노드에 분산시킴으로써 이루어질 수 있다. 그래프의 요소(예를 들어, 데이터 세트, 컴포넌트, 및 흐름)는, 추가의 병렬 특성을 런타임 환경(108)에 도입하기 위해, 직접적으로 또는 간접적으로 복제될 수 있다.

[0037] 런타임 환경(108)은 기억된 스크립트로부터의 제어 입력 또는 사용자(107)로부터의 입력을 수신하여 연산을 실행 및 구성(configure)할 수 있다. 제어 입력은 기억된 그래프 데이터 구조로 특정된 대응하는 컴포넌트 그래프를 사용하여 특정의 데이터 세트를 처리하기 위한 명령(또는 명령어)을 포함할 수 있다. 사용자(107)는, 예를 들어 명령어 인터페이스 또는 그래프 인터페이스를 이용하여 런타임 환경(108)과 상호작용을 할 수 있다.

[0038] 런타임 환경(108)은 소정의 연산 그래프를 특정하는 기억된 그래프 데이터 구조를 판독하고, 컴포넌트의 연산을 수행하기 위한 프로세스[호스트 운영 체제 내에서의 실행의 프로세스 또는 쓰레드(thread)]와 같은 컴퓨팅 리소스를 할당 및 구성하기 위한 선행 실행 모듈(pre-execution module)(110)을 포함한다. 선행 실행 모듈(110)은 컴포넌트들 간의 링크를 구현하기 위한 컴포넌트간(inter-component) 통신 리소스(예를 들어, 파이프 또는 공유 메모리)를 할당하고, 아직 새로운 작업 요소를 받아들일(accept) 준비가 안 된 프로세스를 갖는 컴포넌트의 입력 포트에 도달하는 작업 요소를 위한 기억 공간을 할당한다. 컴포넌트들 사이에서 링크를 통해 작업 요소들을 전달하기 위한 리소스는 시스템(100)의 처리 및 기억 오버헤드의 원인이 된다.

[0039] 런타임 환경(108)은 선행 실행 모듈(110)에 의해 연산 그래프에 할당된 프로세스의 실행을 스케줄링 및 제어하기 위한 실행 모듈(execution module: 112)을 포함한다. 이 실행 모듈(112)은, 데이터베이스 엔진, 데이터 기억

장치, 또는 그래프 컴포넌트와 연관된 처리 과정 중에 액세스되는 다른 모듈과 같이, 시스템(100)에 연결된 외부 컴퓨팅 리소스와 상호작용을 행할 수 있다.

[0040] 도 1b를 참조하면, 연산 그래프(105)를 단순하게 나타낸 예로서, 이 연산 그래프(105)는 4개의 링크된 정점, 즉 입력 데이터 세트(120), 제1 컴포넌트(122), 제2 컴포넌트(124) 및 출력 데이터 세트(126)를 포함하며, 입력 데이터 세트(120)는 제1 컴포넌트(122)에 링크되어 있고, 제1 컴포넌트(122)는 제2 컴포넌트(124)에 링크되어 있으며, 제2 컴포넌트(124)는 출력 데이터 세트(126)에 링크되어 있다. 링크에 의해, 그래프(105)의 정점들 사이에서 작업 요소들이 이동할 수 있게 된다. 작업 요소들 중 일부는 데이터 요소(130)(예를 들어, 데이터베이스의 개별 레코드 또는 하나 이상의 레코드에 관련된 값들)에 대응하며, 작업 요소의 일부는 반드시 실제의 데이터를 나타내지 않고 데이터 요소를 처리하기 위해 사용되는 처리 요소(processing element: 132)에 대응한다. 예를 들어, 처리 요소(132)에는 체크 포인트(checkpoint) 요소, 연산 포인트(compute point) 요소, 또는 분리 요소(separation element) 등이 있을 수 있다. 입력 데이터 세트(120)와 출력 데이터 세트(126)는, 예를 들어 데이터베이스 시스템과 관련된 레코드 또는 트랜잭션(transaction) 처리 시스템과 관련된 트랜잭션을 나타낼 수 있다.

[0041] 도 1b에 나타낸 예에서, 컴포넌트(122, 124)는 전체 연산 그래프(105)에 의해 정의되는 연산의 일부와 관련되며, 링크를 통해 입력 포트에서 수신한 하나 이상의 입력 데이터 요소에 기초하여 연산을 수행하고, 이상의 데이터 요소를 링크를 통해 출력 포트로부터 제공한다. 출력 데이터 요소는, 예를 들어 입력 요소의 변형된 형태가 되거나, 출력 데이터 요소가 입력 데이터 요소 중의 어떤 것과 반드시 직접적인 대응 관계를 갖지 않을 수도 있다. 컴포넌트(122, 124)들 간의 작업 요소 중의 일부는 처리 요소(132)이다. 컴포넌트(124)가 수신한 처리 요소(132)가 체크 포인트 요소에 해당하면, 컴포넌트(124)는 체크 포인트 처리 과정을 수행하여, 그래프(105)로 하여금 실패가 생겼을 때 연산 과정 중의 해당 지점으로 재귀하도록 하기 위한 연산의 상태를 기억할 수 있다. 컴포넌트(124)가 수신한 처리 요소(132)가 연산 포인트이면, 컴포넌트(124)는 마지막 연산 포인트 이후에 수신한 일련의 데이터 요소에 기초하여 결과를 갱신하는 것과 같은 연산에서 미리 정해진 동작을 수행한다[이에 대해서는, 예를 들어 "Continuous flow compute point based data processing"이란 명칭의 미국특허 6,654,907호에 더 상세히 개시되어 있으며, 상기 특허문헌의 내용을 본 명세서에서 참조에 의해 원용하는 것으로 한다]. 연산 포인트는, 소정의 연산 중의 관련 부분에 대응하는 여러 흐름 중에서 위치를 표시(mark)함으로써 그래프 내의 컴포넌트에 대해 일종의 동기화를 제공하는 것이다.

[0042] 분리 요소(separation element)에 의해, 동기화(synchronization)가 가능하며, 컴포넌트들이 여러 흐름 내의 데이터 요소들 간의 관계를 인식할 수 있게 된다. 예를 들어, 분리 요소는 계층(hierarchical) 또는 내포(nested) 구조를 갖는 데이터와 같은 구조화된 데이터의 여러 부분과 연관된 데이터 요소의 그룹을 분리시키기 위한 작업 요소의 흐름 내에서 사용될 수 있다. 계층적 데이터 구조의 예는, 확장가능한 마크업 언어(XML: Extensible Markup Language)와 같은 마크업 언어에 따라 포맷된 텍스트를 포함하는 파일 또는 파일의 일부이다. XML 문서에서, 쌍을 이루는 오프닝 태그(opening tag)와 클로징 태그(closing tag)는 서로 내포(nest)될 수 있다. 계층(hierarchy)에서의 레벨은 문서 내의 태그의 내포의 깊이에 대응한다. 내포 구조를 갖는 데이터의 다른 예는, 연산 내에서 내포된 반복 루프(nested iterative loop)에 대한 인수(argument)의 내포된 리스트이다. 계층적 데이터는 PL/I 또는 COBOL과 같은 프로그래밍 언어나, IBM 정보 관리 시스템(IMS: Information Management System)과 같은 계층적 데이터베이스에서와 같은 계층적 데이터 구조에 따라 포맷될 수 있다.

[0043] 일부 경우에, 입력 데이터 세트의 데이터는 계층적으로 될 수 있으며, 컴포넌트는 데이터 구조 내에서의 데이터 값(예컨대, 흐름 내의 데이터 요소)뿐만 아니라, 데이터 값들 사이의 적어도 몇몇 관계를 수신할 필요가 있다. 이들 관계는 흐름 내의 계층 구조 정보[예컨대, 분리 요소 또는 합성 키(synthetic key)]를 포함함으로써 표현될 수 있다. 입력 데이터 세트로부터의 계층적 데이터 구조는 해당 데이터 구조로부터 데이터 값을 추출(extract)한 컴포넌트에 의해 수신될 수 있으며, 계층 구조 정보를 포함하는 하나 이상의 출력 흐름을 생성할 수 있다. 예를 들어, 데이터 요소는 다른 다운스트림(downstream) 컴포넌트로 하여금 최초의 데이터 구조로부터 적어도 몇몇의 계층적 관계를 구분해낼 수 있도록 하는 배열(arrangement)에서 분리 요소에 의해 분리될 수 있다. 이와 달리, 데이터 요소는 다른 다운스트림 컴포넌트로 하여금 최초의 데이터 구조로부터 적어도 몇몇의 계층적 관계를 구분해낼 수 있도록 하는 대리 키(surrogate key)를 포함할 수 있다. 이러한 컴포넌트["정규화 컴포넌트"(normalize component)라고 함]의 예에 대해서는 나중에 상세하게 설명한다. 일부의 경우에, 데이터 요소는 출력 데이터 세트에 기억되어야 하는 계층적 데이터 구조에 정렬될 필요가 있다. 컴포넌트는 하나 이상의 흐름으로부터 수신한 여러 상이한 데이터 요소가, 흐름 내에서 분리 요소에 적어도 부분적으로 기초하여 어떻게

정렬되어야 하는지를 판정할 수 있으며, 출력 데이터 세트를 위한 적절한 데이터 구조를 생성할 수 있다. 이러한 컴포넌트["역정규화 컴포넌트"(denormalized component)라고 함]의 예에 대해서는 나중에 상세하게 설명한다.

[0044] XML 문서와 같은 몇몇 계층적 데이터 구조는, 데이터 구조 내에서 요소에 액세스하기 위한 인터페이스를 제공하는 파서(parser)를 사용하여 처리될 수 있다. XML 문서의 경우, 파서에 대한 2가지 방법이 있는데, 그중 한가지는 W3C(World Wide Web Consortium)에서 개발한 DOM(Document Object Model) 사양이고, 다른 하나는 자바(Java)와 다른 프로그래밍 언어로 개발된 SAX(Simple API for XML)이다. 이 2가지 방법은 모두 구문 규칙에 맞는("well-formed") XML 문서의 구문(syntax)에 기초한다. 이러한 문서는, 계층을 나타내는 내포 구조 내에 임의 개수의 XML 요소를 포함할 수 있는 내용을 갖는 한 쌍의 루트(root) 오프닝 태그 및 클로징 태그를 포함한다.

[0045] XML 요소의 형태는 다음과 같다.

[0046] <tag_name attribute_name="value"...>...</tag_name>

[0047] 이 요소는 이름(본 예에서는 "tag_name")과 오프닝 태그에서 정의된 임의 개수의 속성을 갖는 오프닝 태그 및 클로징 태그에 의해 범위가 정해진다. 각각의 속성은 이름(본 예에서는 "attribute_name")과 인용부호(quote) 내의 값을 갖는다. 임의의 오프닝 태그 및 클로징 태그로 이루어진 쌍들 간의 내용(content)은 오프닝 태그 및 클로징 태그로 이루어진 쌍 내에 내포된 다른 XML 요소를 포함할 수 있다. 태그의 내포에 기초하여, 각각의 XML 문서는 트리(tree)로 표현될 수 있는 계층에 대응하는데, 이러한 트리는 루트 태그(root tag)에 대응하는 "루트 노드"(root node)와 자식 노드(child node)를 포함하고, 자식 노드의 일부 구조는 다른 태그 내의 XML 요소의 내포에 대응하는 각각의 부모 노드(parent node)에 연결되어 있다.

[0048] 예를 들어, 도 1c에 나타낸 트리 구조(140)는 이하의 XML 문서 내의 내포된 태그로부터 생긴 "계층"을 표현한다. 트리 구조(140)에는 레벨 1(루트 노드)에 하나의 노드가 있고, 3개의 레벨(각각의 레벨은 루트 노드로부터의 상이한 거리와 태그의 상이한 내포 깊이에 대응함)의 깊이로 펼쳐진 자식 노드가 있다. 각각의 노드에는 대응하는 속성 값이 붙여져 있다.

[0049] <top x="t1">

[0050]

[0051]

[0052]

[0053]

[0054]

[0055]

[0056]

[0057] </top>

[0058] SAX 방법과 DOM 방법은 상이한 장점을 갖는다. XML 문서를 파싱하는 DOM 방법에서는, 파서가 XML 문서의 내포 태그에 기초하여 노드들 간에 부모 관계를 가진 트리 데이터 구조를 생성한다. DOM 파서는 요소의 처리를 시작하기 전에 메모리 내에 이러한 트리 데이터 구조를 기억시키기 때문에, 이용가능한 메모리 용량에 의해 효과적으로 처리될 수 있는 XML 문서의 사이즈가 제한을 받게 될 수 있다. XML 문서를 파싱하기 위한 SAX 방법에서는, 파서가 XML 문서를 순차적으로 순회(traverse)하고, 문서의 요소가 태그에 기초하여 인식됨에 따라 애플리케이션에 대한 함수 호출을 생성한다. 이러한 함수 호출(function call)은 전체 XML 문서의 표시를 메모리에 기억시키지 않고도, 문서를 순회시킴에 따라 요소를 처리한다. 따라서, SAX 파서(XML 트리의 최대 깊이에 기초함)는 DOM 파서(XML 트리에서의 전체 노드 개수에 기초함)에 비해, 일반적으로 더 적은 용량의 메모리를 필요하게 된다. SAX 파서가, 특히 부피가 큰 문서에 대해 더 빨리 행해질 수 있는 반면에, DOM 파서는 더 유연한 처리가 가능해서, 문서가 처리될 때의 구조적인 관계를 유지할 필요 없이, 미리 정해진 트리 데이터 구조를 사용하여 수행될 수 있다.

[0059] 도 2는 XML 문서와 같은 계층적 데이터 구조를 처리하기 위한 연산 그래프(200)의 예를 나타낸다. 이하의 예에서는, 서술 XML 문서 구조를 단순하게 나타내었지만, 사용되는 기술은 다른 유형의 계층적 또는 내포 데이터에

도 적용될 수 있다. 입력 데이터 세트(202)는 XML 문서 또는 다수의 XML 문서의 스트림을 제공한다. 정규화 컴포넌트(204)는 XML 문서로부터 데이터 요소(예컨대, 속성, 태그 이름, 및/또는 태그가 붙여진 XML 요소의 내용)를 추출하고, 소정의 출력 포트에 대해서는, 해당 문서의 계층적 구조의 특정 레벨로부터 링크를 통해 데이터 요소의 흐름을 제공한다. 이 흐름은, 특정 레벨의 모든 데이터 요소, 또는 특정 레벨에서의 데이터 요소의 하위세트(예를 들어, 소정의 태그를 갖는 및/또는 소정의 태그를 가진 부모를 가진 해당 레벨에서의 모든 요소)를 포함할 수 있다.

[0060] 정규화 컴포넌트(204)는 다중의 처리 모드 중의 임의의 것에 따라 동작하도록 구성될 수 있다. "분리 요소 처리 모드"에서, 정규화 컴포넌트(204)는 어떤 데이터 요소가 소정의 출력 포트를 통해 전송되어야 하는지를 판정하고, 계층 구조에 따라 적절한 위치에 분리 요소를 삽입한다. 이에 대해서는 나중에 더 상세하게 설명한다. "합성 키 처리 모드"에서, 정규화 컴포넌트(204)는 어떤 데이터 요소를 소정의 출력 포트를 통해 전송해야 하는지를 판정하고, 각각의 데이터 요소를 합성 키(대리 키라고도 함)로 증분시켜서, 나중에 더 상세하게 설명하는 바와 같이, 내포된 계층 구조 내의 각자의 위치를 나타내도록 한다(예컨대, "부모" 데이터 요소를 식별한다). 합성 키는 나중에 결합 연산(join operation)에 사용되어, XML 문서의 최초의 내포된 계층 구조의 일부 또는 모두를 재구성한다. 위의 2가지 모드는 모두 전체 문서 구조의 복제를 메모리에 기억시키지 않고도, 입력 데이터 세트(202)로부터 XML 문서를 순차적으로 처리함으로써, SAX 파서의 몇몇 장점을 가질 수 있다.

[0061] 분리 요소 처리 모드는 몇몇 경우에 다른 장점을 갖는다. 분리 요소 처리 모드에서는, 데이터 요소를 그래프의 연산 컴포넌트(206, 208, 210)에 의해 처리하면서, 데이터 요소를 위한 더 효과적인 메모리 기억 요건을 제공하는 XML 문서를 재구성하기 위한 임의의 합성 키 값 또는 다른 정보로 데이터 요소를 증분시킬 필요가 없다. 분리 요소는 몇 개의 흐름(예를 들어, 트리의 깊이와 같은 개수의 흐름) 내에서 XML 문서의 트리 구조의 일부 또는 모두를 제공함으로써, DOM 파서의 장점을 제공할 수 있다.

[0062] 도 2에 나타낸 예에서, 입력 XML 문서는 계층적 구조의 3개의 다른 레벨에 대응하도록 내포된 3개의 다른 태그로 이루어진 단순한 구조를 가질 수 있다. 정규화 컴포넌트(204)는 3개의 다른 태그를 각각 갖는 데이터 요소를 3개의 연산 컴포넌트(206, 208, 210)에 각각 제공한다. 3개의 연산 컴포넌트는 수신한 데이터 요소에 대하여, 데이터 요소의 속성 및/또는 내용을 변형(transform)시키거나, 데이터 요소를 삽입 또는 삭제하는 것과 같은 변경이 가능하다. 처리된 데이터 요소는 역정규화 컴포넌트(220)로 전달되고, 역정규화 컴포넌트에서 출력 XML 문서를 구성해서 출력 데이터 세트(222)에 기억시킨다. 분리 요소 처리 모드에서, 연산 컴포넌트는 각각의 흐름 내의 데이터 요소 중에서 분리 요소의 정렬을 유지하면서 데이터 요소를 처리하는 것이 가능하다. 역정규화 컴포넌트(220)는 분리 요소를 사용하여, 최초의 XML 문서의 계층 구조에 대응하는 태그의 적절한 내포 정렬을 갖는 처리된 데이터 요소를 갖는 XML 문서를 구성할 수 있다. 다른 예에서, XML 문서에 대하여 중요한 변경이 이루어질 예정이면, 그래프는 분리 요소의 정렬을 변경하거나, 상이한 흐름의 레벨을 변경하는 동작을 수행하거나(적절한 때에 분리 요소를 제거 또는 삽입), 다수의 흐름을 조합하거나, 하나의 흐름을 다수의 흐름으로 분할하는 등이 가능한 다양한 추가의 컴포넌트를 포함할 수 있다.

[0063] 도 3a는 최상위(루트) 레벨인 "레벨 1"과 그 하위의 레벨로서 가능한 2가지 태그 중 하나의 태그가 붙은 요소를 갖는 "레벨 2"를 각각 갖는 일련의 XML 문서를 처리하기 위한 그래프(300)의 일부를 나타낸다. 다음에 나타내는 2개의 XML 문서는 입력 데이터 세트(302)에 의해 제공된 문서 스트림의 일부의 예이다.

[0064] `<top x="t1">`

[0065] `<a y="a1.1"></a>`

[0066] `<b z="b1.1"></b>`

[0067] `<a y="a1.2"></a>`

[0068] `</top>`

[0069] `<top x="t2">`

[0070] `<a y="a2.1"></a>`

[0071] `<b z="b2.1"></b>`

[0072] `<b z="a2.2"></b>`

[0073] `</top>`

[0074] 도 3b는 XML 문서의 계층 구조를 보존할 수 있는 적절한 분리 요소에 의해 분리된 문서로부터 속성 값을 가진 데이터 요소의 흐름을 제공하기 위해 분리 요소 처리 모드에서 XML 문서를 처리하기 위한, 정규화 컴포넌트(304)를 사용한 프러시저(330)의 예를 나타내는 플로차트이다. 이 프러시저(330)는 새로운 XML 요소의 오프닝 태그가 있을 때마다 "추출" 단계(332)에서 시작하는 재귀 프러시저(recursive procedure)로서 표현된다. 이 프러시저(330)는 처음에, 발견한 루트 XML 요소(본 예에서는, "레벨 1"과 관련된 "최상위" 태그를 갖는)에 적용된다. 이 컴포넌트(304)는 발견한 XML 요소의 임의의 태그가 없는 내용과 속성을 추출해서(332), 대응하는 흐름의 데이터 요소에 대한 적절한 형태로 임시로 기억시킨다. 컴포넌트(304)는 해당 문서에서 새로운 오프닝 태그 또는 클로징 태그가 될 수 있는 다음 항목을 찾는다(334). 발견한 XML 요소 내에서 내포된 새로운 오프닝 태그를 찾으면, 컴포넌트(304)는 이 프러시저(330)를 반복해서 적용한다. 클로징 태그를 발견하면, 컴포넌트(304)는 소정 레벨에 관련된 흐름 내의 클로징 태그의 소정 레벨과 관련된 보존된 데이터 요소를 출력하는 단계(336)로 진행한다. 컴포넌트(304)는 소정의 레벨보다 하위의 레벨(이 레벨이 있다는 전제하에)과 관련된 다른 각 흐름의 분리 요소를 출력(337)한다. 몇몇 경우에, 분리 요소는 클로징 태그의 소정 레벨의 표시를 포함함으로써, 상이한 레벨에서 클로징 태그에 기초하여 삽입되었던 흐름에서의 상이한 분리 요소를 구분할 수 있게 된다. 이어서, 컴포넌트(304)는 임의의 개방 프러시저 호출이 남아 있으면, 해당 재귀 루틴(recursion)을 빠져나오거나, 다음 오프닝 루트 태그(340)를 발견하기 위해 다음 단계로 진행한다.

[0075] 도 3c는 상기 2개의 XML 문서에 적용된 프러시저(330)의 결과를 나타낸다. 레벨 1과 관련된 흐름(361)은 "최상위" 태그를 갖는 최상위 레벨 XML 요소로부터의 속성 값을 데이터 요소에 제공한다. 레벨 2와 관련된 흐름(362)은 "a" 태그를 가진 XML 요소로부터의 속성 값을 데이터 요소에 제공한다. 레벨 2와 관련된 흐름(363)은 "b" 태그를 가진 XML 요소로부터의 속성 값을 데이터 요소에 제공한다. 본 예에서, 정규화 컴포넌트(304)는 레벨 1의 분리 요소를 각각의 클로징 최상위 레벨 태그에 대한 흐름(262, 263)의 각각에 삽입한다. "a"와 "b"에 대한 클로징 태그는 가장 낮은 레벨의 흐름(본 예에서는 레벨 2)과 관련되기 때문에, 이들 클로징 태그에 대해 출력되는 분리 요소는 없다.

[0076] 그래프(300: 도 3a 참조)는 어느 데이터 요소가 상위 레벨의 데이터 요소의 태그 내에 내포되었는지를 나타내는 표시로서 분리 요소를 인식하는 작업 요소의 각각의 흐름을 처리하도록 구성된 연산 컴포넌트(306, 308, 310)를 포함한다. 레벨 2의 흐름을 처리하는 연산 요소(306, 308)는 출력 흐름을 각각의 수집(collect) 컴포넌트(312, 314)에 제공한다. 이 수집 컴포넌트(312, 314)는 분리 요소를 제거하기 위해 레벨 2의 흐름을 처리하고, 레벨 1의 흐름을 제공하기 위해 대응하는 데이터 요소를 수집한다. 조합 컴포넌트(320)는 그래프(300)의 나머지 부분에서의 후속하는 처리를 위해, 모두 3개의 흐름을 처리하여 조합할 수 있다.

[0077] 도 4a는 최상위(루트) 레벨인 "레벨 1"과, 2번째 레벨인 "레벨 2"와, 3번째 레벨인 "레벨 3"을 각각 갖는 일련의 XML 문서를 처리하기 위한 그래프(400)의 일부를 나타낸다. 이하의 2개의 XML 문서는 입력 데이터 세트(402)에 의해 제공된 문서 스트림의 일부의 예이다.

[0078]
```
<top x="t1">
```

[0079]
```
    <a y="a1.1">
```

[0080]
```
        <b z="b1.1.1"></b>
```

[0081]
```
        <b z="b1.1.2"></b>
```

[0082]
```
    </a>
```

[0083]
```
    <a y="a1.2">
```

[0084]
```
        <b z="b1.2.1"></b>
```

[0085]
```
    </a>
```

[0086]
```
</top>
```

[0087]
```
<top x="t2">
```

[0088]
```
    <a y="a2.1">
```

[0089]
```
        <b z="b2.1.1"></b>
```

[0090]
```
    </a>
```

[0091] </top>

[0092] 도 4b는 상기 2개의 XML 문서에 적용된 프러시저(330)의 결과를 나타낸다. 레벨 1과 관련된 흐름(431)은 "top" 태그를 갖는 최상위 레벨의 XML 요소로부터의 속성 값을 데이터 요소에 제공한다. 레벨 2와 관련된 흐름(432)은 "a" 태그를 갖는 XML 요소로부터의 속성 값을 데이터 요소에 제공한다. 레벨 3과 관련된 흐름(433)은 "b" 태그를 갖는 XML 요소로부터의 속성 값을 데이터 요소에 제공한다. 본 예에서, 정규화 컴포넌트(404)는 레벨 1 분리 요소를, 각각의 클로징 최상위 레벨 태그에 대한 흐름(432, 433)의 각각에 삽입하고, 레벨 2의 분리 요소를 각각의 클로징 "b" 태그에 대한 흐름(433)에 삽입한다. "b"에 대한 클로징 태그는 최하위 레벨의 흐름(본 예에서는 레벨 3)에 관련되기 때문에, 이들 클로징 태그에 대해 출력되는 분리 요소는 없다.

[0093] 도 4a를 다시 참조하면, 3개의 연산 컴포넌트(406, 408, 410)는 흐름(433, 432, 431)을 각각 처리한다. 수집 컴포넌트(412)는 레벨 2의 분리 요소를 제거하기 위한 레벨 3의 흐름(431)을 처리하고, 레벨 2의 흐름을 제공하기 위해 대응하는 데이터 요소를 수집한다. 조합 컴포넌트(414)는 연산 컴포넌트(408)로부터 레벨 2의 흐름과 함께 결과로서 생긴 레벨 2의 흐름을 처리하여, 조합된 흐름을 수집 컴포넌트(416)에 제공할 수 있다. 수집 컴포넌트(416)는 레벨 1의 분리 요소를 제공하고, 조합 컴포넌트(420) 내에서 연산 컴포넌트(410)로부터 레벨 1의 흐름과 조합되는 레벨 1의 흐름을 제공한다.

[0094] XML 문서(또는 다른 계층적 데이터 구조)를 제공하도록 하나 이상의 수신된 흐름을 처리하기 위해 역정규화 컴포넌트에 의해 사용될 수 있는 프러시저는 정규화 컴포넌트에 의해 수행되는 것과 반대의 작용을 할 수 있다. 역정규화 컴포넌트는 흐름 내의 분리 요소를 다른 흐름 내의 대응하는 데이터 요소에 매칭시켜서, XML 요소를 내포시킬 방법을 정할 수 있다. 예를 들어, 레벨 L의 분리 요소에 이어 하나 이상의 데이터 요소가 있으면, 역정규화 컴포넌트는 다른 흐름에서 대응하는 레벨 L의 데이터 요소를 찾을 수 있으며, 하나 이상의 데이터 요소가 레벨 L의 데이터 요소에 대응하는 XML 요소 내에 내포될 수 있다.

[0095] 합성 키 처리 모드는 몇몇 경우에 장점을 갖는다. 분리 요소는 합성 키 처리 모드에서는 필요하지 않다. 소정의 데이터 요소는, 소정의 데이터 요소의 적어도 부모 데이터 요소를 고유하게 식별하는 합성 키를 포함한다. 예를 들어, 데이터 요소의 레코드 구조(record structure)는 합성 키 값을 기억하기 위한 여분의 필드를 포함할 수 있다. 각각의 자식 데이터 요소에서의 부모 데이터 요소를 식별하는 것이 계층적 구조에서의 관계를 재구성하는 데에 충분하지만, 합성 키는 해당 계층에서의 임의의 개수의 조상(ancestor)을 임의로 식별할 수 있다.

[0096] 합성 키 처리 모드는 전체 문서 구조의 복제본을 메모리에 기억시킬 필요없이, 입력 데이터 세트(202)로부터 XML 문서를 순차로 처리함으로써 SAX 파서의 장점을 갖는다. 합성 키는 몇몇 흐름(예를 들어, 트리의 깊이와 같은 수의 흐름) 내에서 XML 문서의 트리 구조의 일부 또는 모두를 제공함으로써 DOM 파서의 장점을 제공할 수 있다.

[0097] 합성 키는 순차적인(예컨대, 수치) 값이 될 수 있다. 몇몇 구현 예에서, 합성 키 값은 각각의 새로운 데이터 요소가 정규화 컴포넌트에 의해 처리됨에 따라 증가하는 순차적인 인덱스 값으로서 데이터 요소에 할당될 수 있다. 예를 들어, 정규화 컴포넌트는 인덱스 값을 할당하기 위한 전역 카운터(global counter)를 유지할 수 있다. 계층에서의 각각의 노드는 고유의 인덱스 값과 관련될 것이다. 이와 달리, 일부 예에서, 모든 합성 키를 부모 노드를 고유하게 식별하기 위해 고유하게 하지 않아도 된다. 각 데이터 요소의 레벨을 알고 있으면, 합성 키 값은 계층의 소정의 레벨에 대해 고유하게 될 수 있지만, 다른 레벨에서 반복해도 된다. 예를 들어, 각각의 레벨에 대해 카운터가 유지될 수 있으며, 인덱스 값이 해당 레벨에 대한 카운터를 이용하여 소정 레벨의 데이터 요소에 할당될 수 있다. 소정 레벨에 대한 데이터 요소가 고유의 인덱스 값을 가짐에 따라, 부모 데이터 요소는 알고 있는 부모 레벨과 인덱스 값을 조합하여 고유하게 식별될 수 있다.

[0098] 다음에 나타내는 XML 문서의 초기 부분은 합성 키 처리 모드에서 처리할 입력 데이터 세트(402)에 의해 제공되는 문서의 예이다. 이 문서는 "top" 태그를 가진 최상위(루트) 레벨, "Head" 및 "Body" 태그를 가진 제2 레벨, 각각의 부모 데이터 요소의 태그에 의존하는 태그를 가진 제3, 제4 및 제5 레벨을 포함한다.

[0099] <top>

[0100] <Head>

[0101] <Eye color="green" side="left"></Eye>

[0102] <Eye color="green" side="right"></Eye>

[0103] <Nose color="red"></Nose>

[0104] <Mouth state="open">Pizza</Mouth>

[0105] </Head>

[0106] <Body>

[0107] <Arm side="left">

[0108] <Hand state="open">

[0109] <Finger name="thumb"></Finger>

[0110] <Finger name="index"></Finger>

[0111] <Finger name="middle"></Finger>

[0112] <Finger name="ring"></Finger>

[0113] <Finger name="pinky"></Finger>

[0114] </Hand>

[0115] </Arm>

[0116] <Arm side="right">

[0117] <Hand state="fist">

[0118] <Finger name="thumb"></Finger>

[0119] <Finger name="index"></Finger>

[0120] <Finger name="middle"></Finger>

[0121] <Finger name="ring"></Finger>

[0122] <Finger name="pinky"></Finger>

[0123] </Hand>

[0124] </Arm>

[0125] ...

[0126] </Body>

[0127] </top>

[0128] 도 4c를 참조하면, 본 예에서, 정규화 컴포넌트(404)는 3가지 상이한 일련의 데이터 요소를 흐름(441, 442, 443)에 제공한다. 흐름(441)은 "Nose" 태그를 가진 데이터 요소를 포함한다. 흐름(442)은 "Arm" 태그를 가진 데이터 요소를 포함한다. 흐름(443)은 "Finger" 태그를 가진 데이터 요소를 포함한다. 본 예에서는 분리 요소는 필요하지 않다. 각각의 데이터 요소에는 동일 레벨에서 적어도 다른 데이터 요소 중에서 고유한 인덱스 값이 할당된다. 인덱스 값은 입력 데이터 세트(402)로부터 문서의 계층에 속하는 각각의 데이터 요소를 재구성하기 위해 사용될 수 있는데, 이는 각각의 데이터 요소가 자신의 부모 데이터 요소(부모 데이터 요소를 갖지 않는 루트 레벨의 데이터 요소를 제외함)의 인덱스 값을 포함하기 때문이다. 인덱스 값은 흐름들 간의 데이터의 결합 또는 조사를 위해 데이터 흐름을 수신하는 후속하는 컴포넌트에 의해 판독 및 사용될 수 있다.

[0129] 일례로, 소정의 데이터 요소의 레코드 구조는, 데이터 요소 자체에 할당된 인덱스 값에 대한 필드와, 부모 데이터 요소에 할당된 인덱스 값에 대한 필드를 포함한다. 도 4c에서, 각각의 데이터 요소에는 자신의 인덱스 값과 그 부모의 인덱스 값, 즉 "[own]/[parent]"가 붙여져 있다.흐름(441)에서의 데이터 요소는 소정의 "Nose" 데이터 요소의 부호 "Head" 데이터 요소를 고유하게 식별하는 인덱스 값을 포함한다. 흐름(442)에서의 데이터 요소는 소정의 "Arm" 데이터 요소의 부호 "Body" 데이터 요소를 고유하게 식별하는 인덱스 값을 포함한다. 흐름(443)에서의 데이터 요소는 소정의 "Finger" 데이터 요소의 부호 "Hand" 데이터 요소를 고유하게 식별하는 인덱스 값을 포함한다. 다른 예로서, 소정의 데이터 요소의 레코드 구조는 데이터 요소 자체에 할당된 인덱스 값에

대한 필드와, 루트 레벨까지 조상 데이터 요소에 할당된 인덱스 값에 대한 필드를 포함한다. 도 4d에서, 각각의 데이터 요소는 자신의 인덱스 값과 그 조상의 인덱스 값, 즉 "[own]/[parent]/.../[root]"가 붙여져 있다. 도 4c와 도 4d의 예에서, 각 데이터 요소의 부모 인덱스는 모든 흐름에서의 그 동일 인덱스를 포함할 필요없이, 해당 데이터 요소의 계층에서의 위치를 재구성하기에 충분하다. 이에 의해, 계층의 폭이 증가함에 따라 증가하지 않는 흐름 내에서 계층의 효과적인 표현이 가능하게 된다. 도 4c의 예에서, 각 데이터 요소에 대한 인덱스 필드의 사이즈는 계층의 사이즈와 일치한다. 도 4d의 예에서, 각 데이터 요소에 대한 인덱스 필드의 사이즈는 계층의 폭이 아니라 계층의 깊이에 따라 증가한다.

[0130] 정규화 컴포넌트(404)는 각 데이터 요소 출력 내에 포함될 임의 개수의 후손(descendent) 데이터 요소를 선택할 수 있는 능력을 흐름에 제공한다. 일부의 경우에, 소정의 노드와 관련된 데이터(예를 들어, 속성 값)만이 데이터 요소로서 출력된다. 일부의 경우에, 데이터 요소는 자체적으로 데이터 요소가 문서의 최초의 계층으로부터 후손 노드의 완전한 또는 부분적인 하위트리(sub-tree)를 포함하도록 하는 계층 구조를 포함한다. 예를 들어, 상기 문서의 경우에, "Arm" 태그의 레벨에 대응하는 데이터 요소는 "Hand" 및 "Finger" 태그에 대응하는 임의의 개수의 관련된 후손 노드에 대한 데이터를 포함한다. 데이터 요소 자체는 계층적 구조를 갖기 때문에, 후손 노드로부터 삽입된 데이터에 대해서는 합성 키가 할당될 필요가 없다.

[0131] 부모 또는 다른 조상 노드를 식별하는 인덱스 필드에 추가로, 데이터 요소는 부모 또는 다른 조상 노드로부터의 데이터를 임의선택적으로 포함할 수 있다. 예를 들어, 상기 문서의 경우, "Finger" 태그의 레벨에 대응하는 데이터 요소는 손가락 "이름"(name) 속성(예를 들어, name="pinky")뿐만 아니라, 손의 "상태"(state) 속성(예를 들어, state="first")과 팔의 "사이드"(side) 속성(예를 들어, side="right")을 포함할 수 있다. 일부의 경우에, 조상 노드로부터 정보를 포함함으로써 소정의 결합(join) 과정을 수행하지 않아도 된다.

[0132] 도 4e는 XML 문서의 계층적 구조가 보존될 수 있도록 하는 적절한 합성 키와 문서로부터의 속성 값을 데이터 요소의 흐름에 제공하기 위해 합성 키 처리 모드에서 XML 문서를 처리하기 위한, 정규화 컴포넌트(404)에 의해 사용되는 프러시저(450)의 예를 나타내는 플로차트이다. 이 프러시저(450)는 새로운 XML 요소의 오프닝 태그를 발견할 때마다 "추출" 단계(452)에서 시작하는 재귀 프러시저로서 표현된다. 이 프러시저(450)는 발견한 루트 XML 요소(본 예에서는, "레벨 1"과 관련된 "최상위" 태그를 가짐)에 먼저 적용된다. 컴포넌트(404)는 대응하는 흐름의 데이터 요소에 대한 적절한 포맷으로 발견한 XML 요소의 임의의 태그가 붙지 않은 내용과 속성을 추출(452)한 다음 임시로 기억한다. 컴포넌트(404)는 새로운 오프닝 태그 또는 클로징 태그가 될 수 있는, 해당 문서에서 다음 아이템을 찾는다(454). 새로운 오프닝 태그가 발견된 XML 요소 내에 내포된 것을 발견하면, 컴포넌트(454)는 프러시저(500)를 재귀적으로 적용한다. 클로징 태그를 발견하면, 컴포넌트(404)는 소정의 레벨과 관련된 흐름에서 클로징 태그의 소정의 레벨과 관련된 보존된 데이터 요소를 출력한다(456). 컴포넌트(404)는 부모 데이터 요소(존재한다는 가정하에)를 식별하는 데이터 요소에 인덱스 값을 기입한다(457). 이어서, 컴포넌트(404)는 임의의 오픈 프러시저 호출이 남아 있는 경우에, 재귀 루틴을 빠져나오거나, 다음 오프닝 루트 태그를 발견하기 위한 과정으로 진행한다(460).

[0133] XML 문서(또는 다른 계층적 데이터 구조)를 제공하도록 하나 이상의 수신된 흐름을 처리하기 위한 역정규화 컴포넌트에 의해 사용될 수 있는 프러시저는, 정규화 컴포넌트에 의해 수행되는 것과 반대의 작용을 할 수 있다. 역정규화 컴포넌트는 흐름 내에서의 합성 키 값을, 대응하는 다른 흐름에서의 합성 키 값에 매칭시켜서, XML 요소를 어떻게 내포시킬 것인지를 판정할 수 있다.

[0134] 본 명세서에서 설명하는 데이터 흐름 관리 방법은 컴퓨터에서 실행되는 소프트웨어를 사용하여 구현될 수 있다. 예를 들어, 소프트웨어는 하나 이상의 프로세서, 하나 이상의 데이터 기억 장치 시스템(휘발성 및 불휘발성 메모리 및/또는 기억 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 프러시저를 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 기억된 데이터 구조 또는 데이터 저장 공간에 기억된 데이터 모델에 부합하는 다른 구성의 데이터로서 구현될 수 있다.

[0135] 소프트웨어는 범용 또는 전용의 프로그램가능한 컴퓨터로 판독가능한 CD-ROM과 같은 기억 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 구현되어

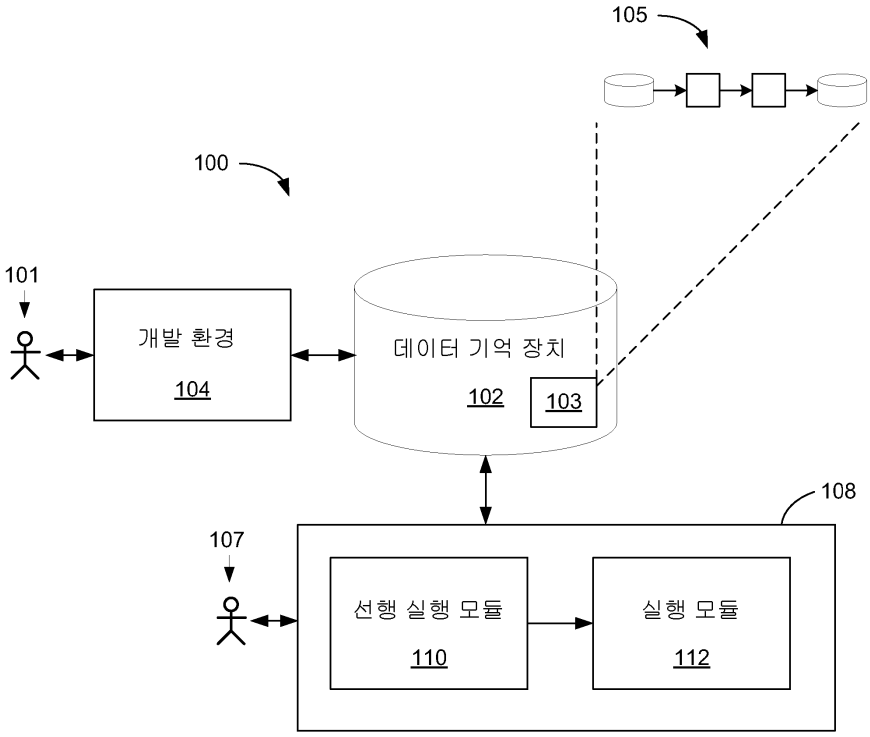
도 된다. 이러한 각각의 컴퓨터 프로그램은, 본 명세서에서 설명하는 프러시저를 수행하도록 기억 매체 또는 디바이스가 컴퓨터 시스템에 의해 판독될 때에 컴퓨터를 구성 및 운영하기 위한, 범용 또는 전용의 프로그램가능한 컴퓨터에 의해 판독가능한 기억 매체 또는 디바이스(예를 들어, 고체 메모리 또는 매체, 자기 또는 광학 매체)에 기억되거나 다운로드되도록 하는 것이 바람직하다. 본 발명의 시스템은 컴퓨터 프로그램에 맞게 구성된, 컴퓨터로 판독가능한 기억 매체로서 구현될 수도 있는데, 이러한 기억 매체는 컴퓨터 시스템으로 하여금, 본 명세서에서 설명한 기능의 수행을 위해 특정되고 미리 정해진 방식으로 동작할 수 있도록 구성된다.

[0136] 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.

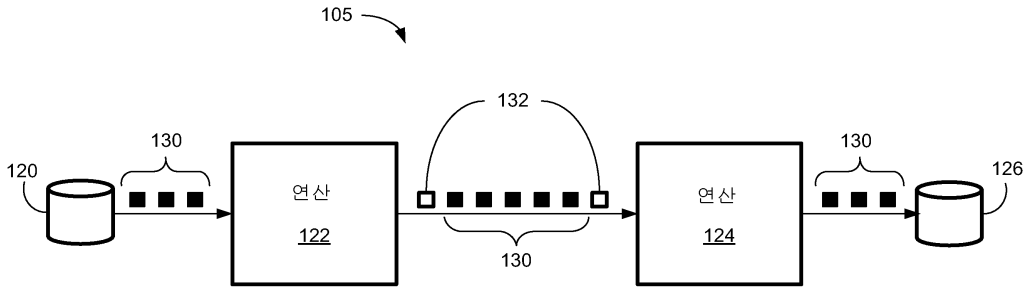
[0137] 이상의 설명은 청구범위에 의해 정해지는 본 발명의 범위를 제한하기 위한 것이 아니라 예시일 뿐이다. 예를 들어, 앞서 설명한 많은 기능적 단계들은 전체적인 과정에 실질적인 영향을 미치지 않으면서, 다른 순서로 수행되어도 된다. 다른 실시예는 이하의 청구범위에 포함된다.

도면

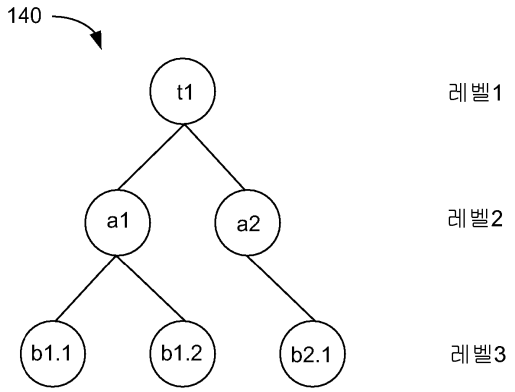
도면1a



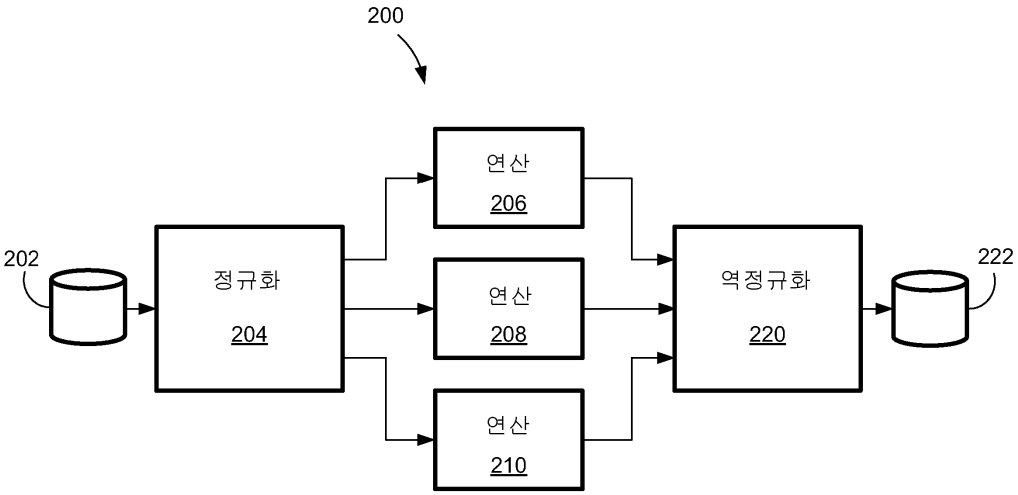
도면1b



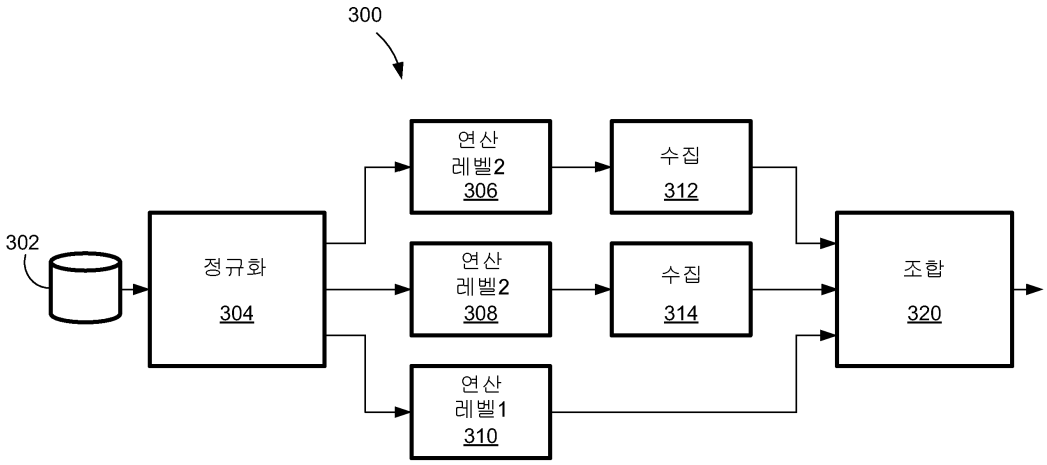
도면1c



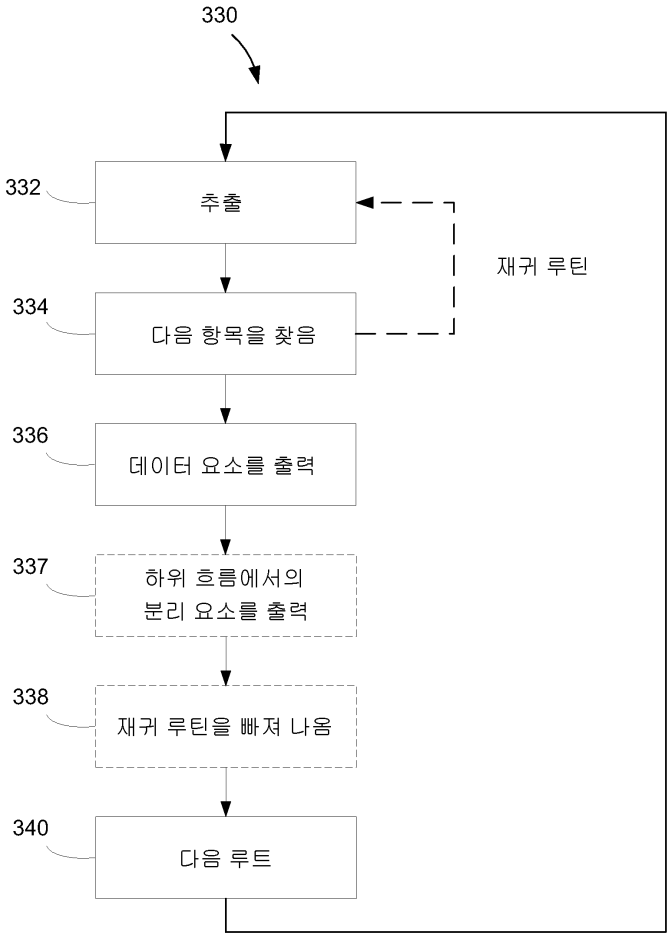
도면2



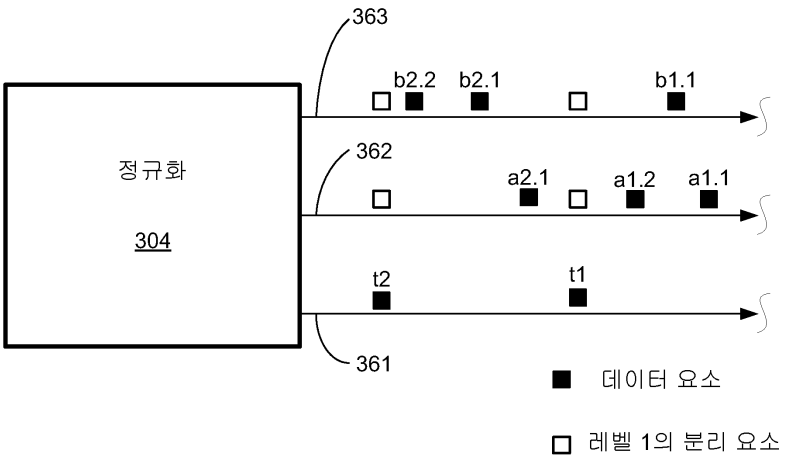
도면3a


300
302
정규화
304
연산
레벨2
306
연산
레벨2
308
연산
레벨1
310
수집
312
수집
314
조합
320

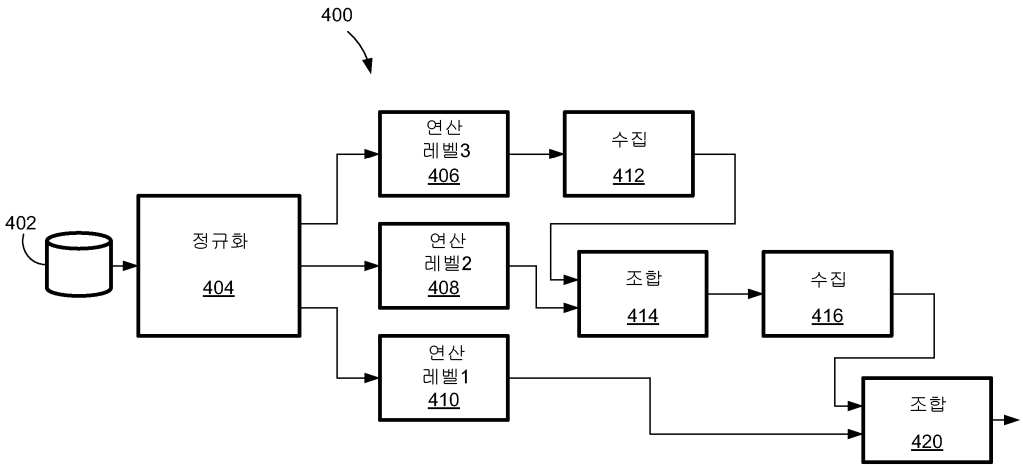
도면3b


330
332
추출
334
다음 항목을 찾음
재귀 루틴
336
데이터 요소를 출력
337
하위 흐름에서의
분리 요소를 출력
338
재귀 루틴을 빠져 나옴
340
다음 루트

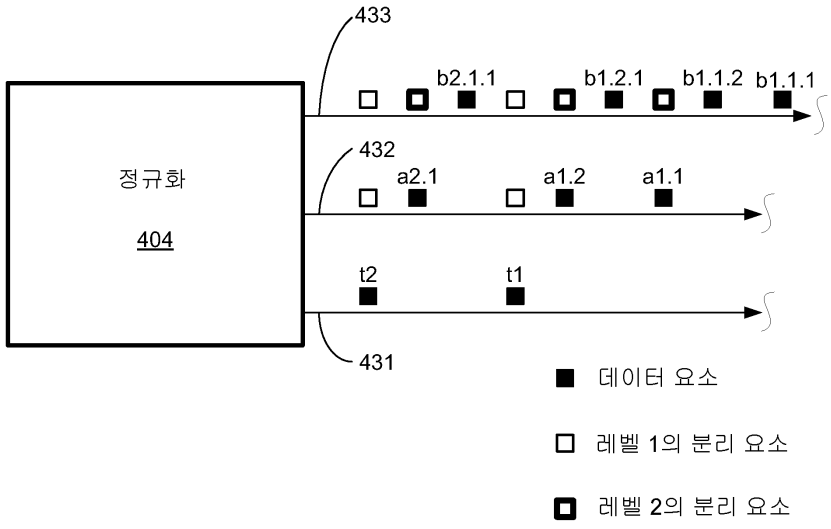
도면3c



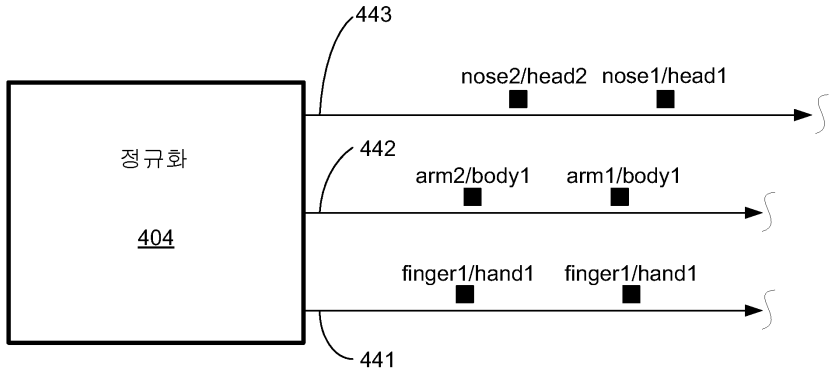
도면4a



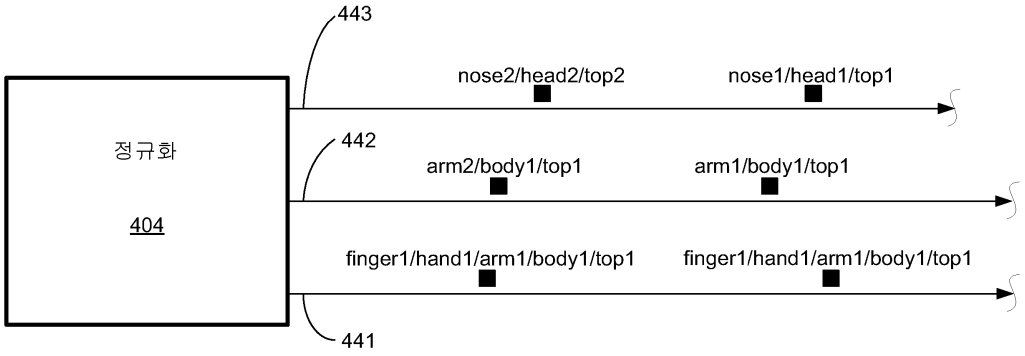
도면4b



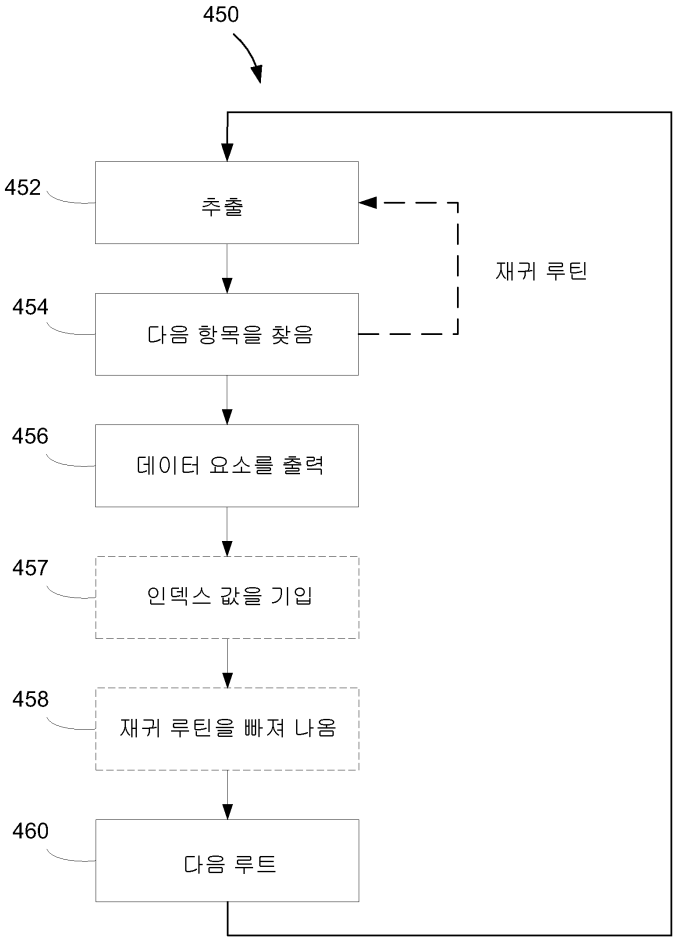
도면4c


443
nose2/head2
nose1/head1
정규화
404
442
arm2/body1
arm1/body1
finger1/hand1
finger1/hand1
441

도면4d


443
nose2/head2/top2
nose1/head1/top1
정규화
404
442
arm2/body1/top1
arm1/body1/top1
finger1/hand1/arm1/body1/top1
finger1/hand1/arm1/body1/top1
441

도면4e


450
452
추출
재귀 루틴
454
다음 항목을 찾음
456
데이터 요소를 출력
457
인덱스 값을 기입
458
재귀 루틴을 빠져 나옴
460
다음 루트