(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2013년01월21일
(11) 등록번호 10-1224680
(24) 등록일자 2013년01월15일

(51) 국제특허분류(Int. Cl.)
G06F 12/00 (2006.01) *G06F 9/00* (2006.01)
(21) 출원번호 10-2006-0002181
(22) 출원일자 2006년01월09일
심사청구일자 2011년01월05일
(65) 공개번호 10-2006-0095448
(43) 공개일자 2006년08월31일
(30) 우선권주장
11/155,428 2005년06월17일 미국(US)
60/657,533 2005년02월28일 미국(US)
(56) 선행기술조사문헌
KR1020040058177 A
KR1020040070090 A
KR1020040103470 A
KR1020040107444 A
전체 청구항 수 : 총 13 항

(73) 특허권자
마이크로소프트 코포레이션
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이
(72) 발명자
리친스, 잭 에스.
미국 98052 워싱턴주 레드몬드 원 마이크로소프트 웨이마이크로소프트 코포레이션 내
헌터, 제이슨 티.
미국 98052 워싱턴주 레드몬드 원 마이크로소프트 웨이마이크로소프트 코포레이션 내
아차리아, 스리니바스무르티 피
미국 98052 워싱턴주 레드몬드 원 마이크로소프트 웨이마이크로소프트 코포레이션 내
(74) 대리인
제일특허법인

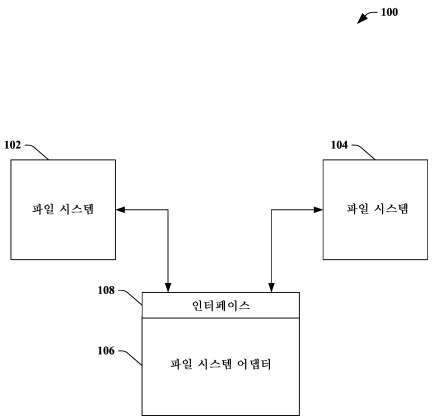
심사관 : 권오성

(54) 발명의 명칭 **데이터베이스 내부에 표현된 파일 시스템**

(57) 요 약

주제 발명은 2 이상의 이질적인 파일 시스템들을 적응시키는 것을 용이하게 하는 시스템 및/또는 방법을 제공한다. 파일 시스템 어댑터는 제1 파일 시스템과 제2 파일 시스템간에 일관성 및/또는 심리스 통합을 제공할 수 있다. 파일 시스템 어댑터는, 파일에 의해 강화된 파일-지지 항목을 이용해 항목 및 항목과 연관된 파일 스트림에 특성 중 하나 이상을 제공할 수 있다. 파일-지지 항목은, 제1 파일 시스템을 위한 API 및 제2 파일 시스템을 위한 API 중 하나 이상에 의한 조작을 허용할 수 있다.

대 표 도 - 도1



특허청구의 범위

청구항 1

이질적인 파일 시스템들간에 일관성(uniformity)을 제공하는 것을 용이하게 하는 컴퓨터 구현된 시스템으로서,

제1 파일 시스템 및 제2 파일 시스템으로부터 데이터를 수신하는 인터페이스;

상기 제1 파일 시스템을 위한 API 및 상기 제2 파일 시스템을 위한 API 중 적어도 하나에 의한 조작을 허용하기 위해, 적어도 하나의 특성(property)을 항목(item)에 제공하기 위한 파일 및 상기 항목과 연관된 파일 스트림에 의해 보강되는 파일-지원(file-backed) 항목을 이용함으로써 상기 제1 파일 시스템과 상기 제2 파일 시스템간에 일관성을 제공하는 파일 시스템 어댑터 - 상기 파일-지원 항목은 또한 상기 제1 파일 시스템에 관한 파일 스트림을 포함함 -; 및

상기 파일 스트림을 포함하지 않는 상기 파일-지원 항목의 복제본을 생성하는 고스트 컴포넌트

를 포함하는 컴퓨터 구현된 시스템.

청구항 2

제1항에 있어서,

상기 제1 파일 시스템은 연관된 파일 스트림을 갖춘 파일 및 디렉토리를 이용하여 애플리케이션을 위한 공간의 구조성을 증강시키는 16 비트 파일 시스템 및 32 비트 파일 시스템 중 적어도 하나인, 컴퓨터 구현된 시스템.

청구항 3

제1항에 있어서,

상기 제2 파일 시스템은 정보를 복합 유형(complex type)으로 표현하기 위해 항목, 서브-항목, 특성, 및 관계 중 적어도 하나를 정의하는 파일 저장 시스템인 컴퓨터 구현된 시스템.

청구항 4

제1항에 있어서,

디렉토리, 파일, 파일 스트림, 속성, 노드, 리프 노드, 및 보안 특징 중 적어도 하나를 판정하기 위해 상기 제1 파일 시스템을 분석하는 분석기 컴포넌트를 더 포함하는 컴퓨터 구현된 시스템.

청구항 5

제1항에 있어서,

상기 제1 파일 시스템의 리프 노드를 위한 파일-지원 항목을 생성하는 발생기 컴포넌트를 더 포함하는 컴퓨터 구현된 시스템.

청구항 6

제1항에 있어서,

상기 제1 파일 시스템 내의 상기 파일의 속성을 상기 파일-지원 항목과 연관짓는 속성 컴포넌트를 더 포함하고,

상기 속성은 상기 제2 파일 시스템에 의해 이용될, 컴퓨터 구현된 시스템.

청구항 7

제6항에 있어서,

상기 속성은 읽기 전용, 압축 파일, 및 보존 파일 중 적어도 하나를 표현할 수 있는 비트인 컴퓨터 구현된 시스템.

청구항 8

제1항에 있어서,

상기 파일-지원 항목과 상기 제1 파일 시스템 내의 파일 지원(file backing)간에 적어도 하나의 속성을 동기화시키는 메타데이터 핸들러 컴포넌트를 더 포함하는 컴퓨터 구현된 시스템.

청구항 9

제8항에 있어서,

상기 메타데이터 핸들러 컴포넌트는,

지원 파일 스트림에서 특성이 수정될 때, 새로운 값을 상기 제2 파일 시스템내의 상기 파일-지원 항목에서의 적합한 필드로 승급시키는 것; 또는

상기 제2 파일 시스템을 위한 API를 통해 특성이 변경될 때, 새로운 값을 상기 지원 파일 스트림으로 강등시키는 것

중 적어도 하나를 제공하는 컴퓨터 구현된 시스템.

청구항 10

제1항에 있어서,

자신과 연관된 상기 파일 스트림의 교환(swap)을 허용함으로써, 적어도 하나의 파일-지원 항목의 동기화를 제공하는 속성 터널링 컴포넌트를 더 포함하는 컴퓨터 구현된 시스템.

청구항 11

제10항에 있어서,

상기 파일 스트림은, 항목 ID들이 교환될 수 있는 파일 스트림 테이블 내에 존재하는 컴퓨터 구현된 시스템.

청구항 12

제1항에 있어서,

상기 파일-지원 항목은 시맨틱(semantic) 계층, 파일 스트림, 파일 속성, 엔티티 상태, 변경 단위들, 또는 항목 데이터 중 적어도 하나를 포함하는 컴퓨터 구현된 시스템.

청구항 13

제1항에 있어서,

연결된 항목들의 그래프를 주기없이 제공하는 DAG(directed acyclic graph) 컴포넌트를 더 포함하는 컴퓨터 구현된 시스템.

청구항 14

삭제

청구항 15

삭제

청구항 16

삭제

청구항 17

삭제

청구항 18

삭제

청구항 19

삭제

청구항 20

삭제

명 세 서

발명의 상세한 설명

발명의 목적

종래기술의 문헌 정보

[0020] 본 출원은 "FILE SYSTEM REPRESENTED INSIDE A DATABASE"라는 명칭으로 2005년 2월 28일에 출원된 미국 가 특허출원 제 60/657,533호에 대한 우선권을 주장한다. 그 출원의 전체 내용이 여기에 참조로써 포함되어 있다.

발명이 속하는 기술 및 그 분야의 종래기술

[0021] 컴퓨터 기술의 진보들(예를 들어, 마이크로프로세서 속도, 메모리 용량, 데이터 전송 대역폭, 소프트웨어 기능 등)은 대체로, 다양한 산업들에서의 증가된 컴퓨터 애플리케이션에 기여해 왔다. 대부분 서버들의 어레이로서 구성되는, 훨씬 더 강력한 서버 시스템들에는 통상적으로, 예를 들어, 월드 와이드 웹(World Wide Web)과 같은 외부 소스들에 기인하는 서비스 요청들이 제공된다.

[0022] 이용 가능한 전자 데이터의 양이 증가함에 따라, 사용자 친화적이며 빠른 데이터 서치들 및 검색(searches and retrieval)을 용이하게 하는 관리 가능한 방식으로 이러한 데이터를 저장하는 것이 더 중요해졌다. 오늘날, 흔한 접근 방법은 전자 데이터를 하나 이상의 데이터베이스에 저장하는 것이다. 일반적으로는, 예를 들어, 컴퓨터 프로그램이 데이터의 소정 부분들을 빠르게 서치하고 선택할 수 있도록 구성된 데이터를 갖춘 정보의 조직된 집합을 통상적인 데이터베이스라고 한다. 흔히, 데이터베이스내의 데이터는 하나 이상의 테이블들에 의해 조직된다. 이러한 도표들은 로우들 및 컬럼들의 어레이로서 정렬된다.

[0023] 또한, 테이블들은 한 세트의 레코드들을 구비할 수 있고, 레코드는 한 세트의 필드들을 포함한다. 레코드들은 보통 테이블내의 로우들로서 인덱싱되고 레코드 필드들은 통상적으로 컬럼들로서 인덱싱되어, 인덱스들의 로우/컬럼 쌍은 테이블내의 특정 데이터를 참조할 수 있다. 예를 들어, 로우는 상거래, 사람, 또는 프로젝트에 관한 완전한 데이터 레코드를 저장할 수 있다. 마찬가지로, 테이블의 컬럼들은 동일한 범용 데이터 포맷을 가진 로우들의 개별적인 부분들을 정의할 수 있는데, 이 경우, 컬럼들은 레코드들의 필드들을 정의할 수 있다.

[0024] 독립적인, 데이터의 개별적인 각 부분은 일반적으로 그다지 유용하지 않다. 데이터베이스 애플리케이션들은 데이터를 좀더 유용하게 하는데, 사용자들이 데이터를 조작하고 프로세싱하는 것을 돕기 때문이다. 데이터베이스 애플리케이션으로 인해, 사용자는 데이터를 비교하고, 정렬하며, 순서화하고, 머지하며, 분리하고, 서로 연결할 수 있고, 그에 따라, 데이터로부터 유용한 정보가 생성될 수 있다. 또한, 데이터베이스들을 이용해 사실상 무한한 저장 용량을 허용할 수 있을 정도로, 데이터베이스의 용량 및 호환성은 놀라운 정도로 증가되어 왔다. 더 나아가, 통상적인 데이터베이스 시스템들은 시간, 파일 확장자, 위치, 및 사이즈에 기초해 한정된 쿼리-능력(query-ability)을 제공한다. 예를 들어, 데이터베이스에 관련된 대량의 데이터를 서치하기 위해서, 통상적인 서치는 파일명, 파일 크기, 또는 생성 날짜로 한정되는데, 이러한 기술들은 불충분하며 부적합하다.

[0025] 최종-사용자들로부터의 데이터 생성이 계속적으로 증가함에 따라, 이러한 데이터를 찾아내고, 관련짓고, 저장하는 것에 관한 문제점들 및 어려움들이 최고조에 달하고 있다. 최종-사용자들은 문서들을 기입하고, 사진들을 저장하며, CD들로부터 음악을 리핑(ripping)하고, 이메일을 수신하며, 송신된 이메일의 복사본들을 보관한다. 예를 들어, 음악 CD를 제작하는 간단한 프로세스에서, 최종-사용자는 메가바이트의 데이터를 발생시킬 수 있다. CD로부터 음악을 리핑하고, 파일을 적당한 포맷으로 변환하며, 보석 상자 커버를 생성하고, CD 레이블을 디자인하는 모든 것은 데이터 생성을 요한다.

[0026] 사용자들 주위의 복잡한 요소들 뿐만 아니라, 개발자들도 데이터와 관련하여 유사한 쟁점들을 가진다. 개발자

들은 개인적인 애플리케이션들에서 고도로 발전된 기업 애플리케이션들에 이르기까지 무수히 많은 애플리케이션들을 생성하고 기입한다. 생성하고 그리고/또는 개발하는 동안, 개발자들은 대부분, 항상은 아니지만, 데이터를 수집한다. 이러한 데이터를 획득하는 동안, 데이터는 저장되어야 한다. 다시 말해, 데이터를 찾아내고, 관련짓고, 저장하는 것에 관한 문제점들 및 어려움들은 개발자와 최종-사용자 모두를 어렵게 한다.

[0027] 또한, 데이터베이스들에서의 기술적 진보들로 인해 많은 포맷들 및/또는 구성들이 초래되어 왔다. 데이터 포맷들 및/또는 구성들은 증가된 듀(due)를 갖는데, 이러한 변화들은 통합 어려움들을 제기할 수 있다. 다시 말해, 이질적인 파일 데이터베이스들은 통상적으로 개발 차이점들에 기초해 서로 잘 적응하지 못한다. 따라서, 종래의 시스템들 및 데이터베이스들과 관련된 상기한 예시적 결함들을 극복할 필요가 있다.

발명이 이루고자 하는 기술적 과제

[0028] 본 발명의 일부 태양들에 대한 기본적인 이해를 제공하기 위해, 다음에서는 본 발명의 간략화된 요약을 제시한다. 이 요약이 본 발명의 포괄적인 개요는 아니다. 이것이 본 발명의 중요하거나 필수적인 요소들을 식별하거나 본 발명의 범위를 서술하는 것도 아니다. 이것의 목적은 이후에 제시되는 좀더 상세한 설명에 대한 도입부로서 본 발명의 일부 개념들을 간략화된 형태로 제시하는 것일 뿐이다.

[0029] 주제 발명은, 이질적인 파일 시스템들에 적응하는 것을 용이하게 하는 시스템들 및/또는 방법들에 관한 것이다. 파일 시스템 어댑터는 제1 파일 시스템과 제2 파일 시스템간에 일관성을 제공할 수 있다. 일관성으로 인해, 제1 파일 시스템을 위한 애플리케이션 및/또는 API가 제2 파일 시스템에 구현될 수 있다. 파일 시스템 어댑터는, 그와 관련된 하나 이상의 특성을 채우기 위해 파일 및/또는 파일 스트림에 의존하는 제2 파일 시스템에 관한 항목일 수 있는 파일-지지 항목(file-backed item)을 구현할 수 있다. 파일-지지 항목은 제2 파일 시스템의 특성들 및/또는 메타데이터로써 항목을 채우기 위해 파일 및/또는 연관된 파일 스트림을 이용할 수 있다. 파일-지지 항목을 이용하는 것에 의해, 파일 시스템 어댑터는 제2 파일 시스템에 연관된 API 뿐만 아니라 제1 파일 시스템의 현재 및 이전 버전들에 관련된 기존 API에 의한 조작을 허용한다. 또한, 파일 시스템 어댑터는 제1 파일 시스템과 제2 파일 시스템간의 동기화를 제공할 수도 있다. 파일-지지 항목은, 속성(attribute), 보안 특징, 파일 스트림, 파일 등과 같은, 그러나 이에 한정되는 것은 아닌, 제1 파일 시스템의 특징들을 제2 파일 시스템에 통합할 수 있다는 것을 알 수 있을 것이다.

[0030] 주제 발명의 일 태양에 따르면, 파일 시스템 어댑터는 속성 컴포넌트를 포함할 수 있다. 속성 컴포넌트는 제1 파일 시스템과 연관된 파일 및/또는 파일 시스템에 연관된 속성들을 제2 파일 시스템에 제공할 수 있다. 파일은, 읽기 전용, 압축, 보존 파일 등과 같은, 그러나 이에 한정되는 것은 아닌, 하나 이상의 속성을 가질 수 있는데, 이 경우, 비트 및/또는 비트들이 이러한 속성들을 표현할 수 있다. 속성 컴포넌트로 인해, 제1 파일 시스템의 파일 및/또는 파일 스트림을 표현하는 파일-지지 항목은 이러한 속성들을 제2 파일 시스템에 통합할 수 있다.

[0031] 주제 발명의 다른 태양에 따르면, 파일 시스템 어댑터는, 하나 이상의 특성(property)이 항목, 파일-지지 항목, 및 지지 파일(backing file;예를 들어, 파일-지지 항목에 의해 표현되는 파일 시스템의 파일) 사이에 동기하여 머물 수 있게 하는 메타데이터 핸들러 컴포넌트를 포함할 수 있다. 특성은 지지 파일 스트림에서 변경될 수 있는데, 이 경우, 메타데이터 핸들러 컴포넌트는 새로운 값을 제2 파일 시스템내의 항목에서의 적절한 필드로 승격시킬 수 있다. 또한, 특성은 제2 파일 시스템과 연관된 API를 통해서도 변경될 수 있는데, 메타데이터 핸들러 컴포넌트는 새로운 값을 제1 파일 시스템내의 지지 파일 스트림으로 강등시킬 수 있다.

[0032] 주제 발명의 또 다른 태양에 따르면, 파일 시스템 어댑터는 파일-지지 항목의 특성에 동기화를 제공하는 고스트 컴포넌트(ghost component)를 포함할 수 있다. 또한, 속성 터널 컴포넌트(attribute tennel component)는 제2 파일 시스템내에서 리소스들의 명칭을 변경하거나 그리고/또는 보존하는 것을 용이하게 하기 위한 속성 터널링을 제공하는데 이용될 수 있다. 또한, 파일 시스템 어댑터는 주제 발명과 함께 방향 비순환 그래프(directed acyclic graph)를 이용하는 DAG 컴포넌트를 포함할 수 있다. 주제 발명의 다른 태양들에서는, 이질적인 파일 시스템들에 적응하는 것을 용이하게 하는 방법들이 제공된다.

[0033] 다음의 설명 및 첨부 도면들은 본 발명의 예시적인 소정 태양들을 상세하게 기술한다. 이들 태양들은, 본 발명의 원리들이 이용될 수 있는 다양한 방법들 중 몇가지만을 나타내지만, 주제 발명은 이러한 태양들 및 그들의 등가물들 모두를 포함하기 위한 것이다. 도면들과 함께 고려되는 본 발명에 대한 다음의 상세한 설명으로부터, 본 발명의 다른 이점들 및 신규한 사양들이 좀더 명백해질 것이다.

발명의 구성 및 작용

[0034] 본 명세서에서 이용되는 바와 같이, "컴포넌트", "시스템", "인터페이스" 등의 용어들은 하드웨어, (예를 들어, 실행 중인) 소프트웨어, 및/또는 펌웨어의 컴퓨터-관련 엔티티를 언급하기 위한 것이다. 예를 들어, 컴포넌트는 프로세서에서 실행 중인 프로세스, 프로세서, 오브젝트, 실행 가능한 것, 프로그램, 및/또는 컴퓨터일 수 있다. 예시로써, 서버에서 실행 중인 애플리케이션과 서버 모두가 컴포넌트일 수 있다. 하나 이상의 컴포넌트들이 프로세스내에 상주할 수 있고, 컴포넌트는 하나의 컴퓨터상에 국한되거나 그리고/또는 2 이상의 컴퓨터들 사이에서 분산될 수 있다.

[0035] 유사한 참조 번호들이 전체에 걸쳐 유사한 요소들을 언급하는데 사용되는 도면들을 참조하여, 주제 발명이 설명된다. 다음의 명세서에서는, 설명을 목적으로, 주제 발명에 대한 완전한 이해를 제공하기 위해 다수의 구체적인 세부 사항들이 기술된다. 그러나, 이러한 구체적 세부 사항들 없이도, 주제 발명이 실시될 수 있다는 것이 명백하다. 다른 경우들로서, 주제 발명의 설명을 용이하게 하기 위해, 주지의 구조들 및 장치들이 블록도 형태로 도시되어 있다.

[0036] 이제 도면들을 참조하면, 도 1은 파일 시스템을 이질적인 파일 시스템과 적응시키는 것을 용이하게 하는 시스템(100)을 도시한다. 파일 시스템(102)은, 파일 스트림 및/또는 디렉토리에 기초한 구조의 파일 저장 시스템일 수 있다. 파일 시스템(102)은 파일 스트림을 이용해, 애플리케이션 및/또는 API(application programming interface)가 파일 스트림을 강화하여 더 많은 구조를 제공하게 할 수 있다. 파일 시스템(104)은 좀더 복잡한 모델일 수 있는데, 이 경우, 항목, 서브-항목, 특성, 및 관계는, 파일 시스템(104)내의 정보 표현이 복합 유형들(complex types)의 인스턴스들이 되도록 정의된다. 항목은, 독립적으로 보안, 직렬화, 동기화, 복사, 백업/복구 등이 될 수 있는, 파일 시스템(104)에서의 일관성의 최소 단위로서 정의될 수 있다. 항목은 유형의 인스턴스인데, 이 경우, 파일 시스템(104)의 모든 항목들은 항목들의 전역적인 단일 범위로 저장될 수 있다. 파일 시스템(104)은 하나 이상의 항목 및/또는 컨테이너 구조에 기초할 수 있다. 파일 시스템(104)은, 파일들에 항목들로서 묻혀 있는 풍부한 메타데이터를 드러내는 저장 플랫폼일 수 있다. 파일 시스템(104)은 논의된 기능을 지원하기 위해 데이터 모델로써 표현되는 파일 저장 시스템일 수 있다는 것을 알 수 있어야 한다.

[0037] 파일 시스템 어댑터(106)는 파일 시스템(102)과 파일 시스템(104)간에 호환성 및/또는 일관성을 제공할 수 있다. 호환성 및/또는 일관성으로 인해, 파일 시스템(102)을 위한 애플리케이션 및/또는 API가 파일 시스템(104)에 구현될 수 있다. 파일 시스템 어댑터(106)는, 그와 연관된 하나 이상의 특성을 채우기 위해 파일 및/또는 파일 스트림에 의존하는 파일 시스템(104)에 관한 항목일 수 있는 파일-지지 항목을 구현할 수 있다. 파일-지지 항목은 파일 시스템(102)과 완전히 체계화된 파일 시스템(104;richly schematized file system) 및/또는 그와 관련된 데이터 모델간의 브리지로서 동작할 수 있다. 예를 들어, 파일 시스템(104)의 이미지 항목은 파일-지지 항목일 수 있는데, 이 경우, 이미지 항목은 파일 시스템(102)의 표준 이미지 포맷(예를 들어, JPEG, TIF, BMP 등)에 따른 파일에 의해 지지된다. 다시 말해, 파일-지지 항목은, 파일 시스템(104)의 특성들 및/또는 메타데이터로써 항목을 채우기 위해 (예를 들어, 파일 시스템(102)에 의해 이용되는) 파일 및/또는 연관된 파일 스트림을 이용할 수 있다. 파일-지지 항목을 이용하는 것에 의해, 파일 시스템 어댑터(106)는 파일 시스템(104)에 연관된 API 뿐만 아니라 파일 시스템(102)의 현재 및 이전 버전들에 관련된 기존 API에 의한 조작을 허용한다. 파일 시스템 어댑터(106)는 파일 시스템(102)과 파일 시스템(104)간에 동기화를 제공할 수도 있다. 특히, 파일-지지 항목은 지지 파일의 대응되는 데이터와 동기된 데이터를 제공할 수 있다. 파일-지지 항목은 파일 시스템(104)에, 속성, 보안 특징, 파일 스트림, 파일 등과 같은, 하지만 그에 한정되는 것은 아닌, 파일 시스템(102)의 특징들을 제공할 수 있다는 것을 알 수 있을 것이다.

[0038] 시스템(100)은, 파일 시스템 어댑터(106)를 실질적인 임의의 오퍼레이팅 시스템(들)으로 통합하기 위한 다양한 어댑터들, 커넥터들, 채널들, 통신 경로들 등을 제공하는 인터페이스 컴포넌트(108)를 더 포함한다. 또한, 인터페이스 컴포넌트(108)는 데이터 및 파일 시스템 어댑터(106)와의 상호 작용을 제공하는 다양한 어댑터들, 커넥터들, 채널들, 통신 경로들 등을 제공할 수 있다. 인터페이스 컴포넌트(108)가 파일 시스템 어댑터(106)로 통합되어 있기는 하지만, 이러한 구현이 그렇게 한정적인 것은 아니다. 예를 들어, 인터페이스 컴포넌트(108)는 시스템(100)과 관련하여 데이터를 송수신하기 위한 독립형 컴포넌트일 수도 있다.

[0039] 도 2는, 일관성을 제공하기 위해 2 이상의 이질적인 파일 시스템들을 적응시키는 것을 용이하게 하는 시스템(200)을 도시한다. 파일 시스템 어댑터(206)는 파일 시스템(202)과 파일 시스템(204)간에 일관성 및/또는 호환성을 제공할 수 있다. 그러나, 파일 시스템 어댑터(206)는 복수개의 이질적인 파일 시스템들에 대한 적응을 제공할 수도 있다는 것을 알 수 있을 것이다. 파일 시스템(202)은 하나 이상의 API 및/또는 애플리케이션을 이용

하는 파일 스트림 및 디렉토리 기반 파일 시스템일 수 있다. 예를 들어, 파일 시스템(202)에 관련된 API 및/또는 애플리케이션은, 이에 한정되는 것은 아니지만, 16 비트, 32 비트 등의 버전일 수 있다. 파일 시스템(204)은 항목을 일관성의 단위로서 이용하는 복합 구조일 수 있는데, 이 경우, 메타데이터, 특성들, 관계들은 복합 유형들의 인스턴스들로서 구현된다. 파일 시스템(204)은, 데이터의 형태를 설명할 수 있고, 데이터에 대한 소정의 시맨틱스적 일관성을 내포하기 위해 제한들을 선언할 수 있으며, 데이터 사이의 시맨틱스적 연관들을 정의할 수 있는 데이터 모델일 수 있다. 파일 시스템 어댑터(206)를 이용하는 것에 의해, 파일 시스템(202)에 관련된 API 및/또는 애플리케이션들은 파일 시스템(204)내에서 변경없이 이용될 수 있다. 파일 시스템(202), 파일 시스템(204), 및 파일 시스템 어댑터(206)는, 각각, 도 1의 파일 시스템(102), 파일 시스템(104), 및 파일 시스템 어댑터(106)와 실질적으로 유사할 수 있다.

[0040] 파일 시스템 어댑터(206)는 파일 시스템(202)을 분석하는 분석기 컴포넌트(210)를 포함할 수 있다. 분석기 컴포넌트(210)는, 파일 시스템(204)에 적응 가능하고 그리고/또는 호환 가능한 시스템을 제공하는 것을 용이하게 하기 위해, 그와 연관된 디렉토리들 및/또는 파일들을 판정할 수 있다. 분석기 컴포넌트(210)는 하나 이상의 리프 노드(leaf node)를 판정할 수 있다. 리프 노드는 파일-지지 항목으로서 표현될 수 있는 반면, 파일 시스템(202)의 모든 디렉토리들 및/또는 파일들은 파일 시스템(204)의 항목들로서 표현될 수 있다. 그에 따라, 리프 노드들이 아닌 디렉토리들 및/또는 파일들이 파일 시스템(204)에 존재할 수도 있지만, 그와 연관된 스트림은 존재하지 않는다.

[0041] 파일 시스템 어댑터(206)는, 파일-지지 항목을 생성하고 그리고/또는 발생시키는 발생기 컴포넌트(212)를 더 포함할 수 있다. 발생기 컴포넌트(212)는 파일 시스템(202)의 분석에 적어도 부분적으로 기초해 파일-지지 항목을 생성할 수 있다. 예를 들어, 발생기 컴포넌트(212)는 파일 시스템(202)에 연관된 임의의 리프 노드를 위한 파일-지지 항목을 생성할 수 있는데, 이 경우, 파일-지지 항목은 파일 시스템(202)으로부터의 파일 및/또는 파일 스트림에 의해 강화되어 있으므로, 파일-지지 항목은 파일 시스템(204)과 호환할 수 있다. 발생기 컴포넌트(212)는, 연관된(예를 들어, 지지) 파일 스트림을 가진 파일-지지 항목; 다른 항목들을 포함할 수 없는 파일-지지 항목(예를 들어, 이름 공간의 리프 노드들); (다음에서 논의되는) 고스팅될 수 있는 파일-지지 항목; 및 그를 위해 (다음에서 논의되는) 파일 속성이 유지되는 파일-지지 항목 중 하나 이상의 파일-지지 항목을 생성할 수 있다.

[0042] 파일-지지 항목은 파일 시스템(202)과 파일 시스템(204)간의 브리지로서 동작할 수 있다. 파일 시스템(204)의 임의 항목은, 1) 내용의 적어도 일부가 파일 시스템(202)의 파일 및/또는 파일 스트림을 형성하거나, 그리고/또는 2) 기초적인 파일의 내용이 파일 시스템(202)에 연관된 하나 이상의 API 및/또는 애플리케이션에 의해 조작되어야 한다면, 파일-지지 항목으로서 지정될 수 있다. 파일 시스템(204)의 파일-지지 항목은 항목과 연관된 메타데이터 특성 뿐만 아니라 항목에 연관된 파일 스트림을 저장할 수 있다. 파일 시스템(202) 폴더/디렉토리/파일 계층 구조 메커니즘들이, 파일-지지 항목들을 포함해 다른 항목들을 포함하는 컨테이너 항목들을 이용해 파일 시스템(204)에 표현될 수 있다는 것을 알 수 있을 것이다. 그에 따라, 파일 시스템(202)의 API를 이용하는 애플리케이션이 파일 시스템(204)의 저장 메커니즘과 심리스하게(seamlessly) 동작할 수 있다.

[0043] 도 3은 파일 스트림 연관형 파일 시스템 및 메타데이터 연관형 파일 시스템을 적응시키는 것을 용이하게 하는 시스템(300)을 도시한다. 파일 시스템 어댑터(306)는 파일 시스템(302)과 그에 대해 이질적인 파일 시스템(304)간에 일관적이고 그리고/또는 호환적인 상호 작용을 제공할 수 있다. 파일 시스템 어댑터(306)는 파일-지지 항목을, 하나 이상의 특성 및/또는 메타데이터를 파일 시스템(302)에 연관된 파일 및/또는 파일 스트림으로써 채울 수 있는 파일 시스템(304)내의 항목으로서 이용할 수 있다. 파일-지지 항목은 파일 시스템(302;예를 들어, 파일 포맷)과 파일 시스템(304;예를 들어, 복합 유형들의 완전히 체계화된 인스턴스들)간의 브리지로서 동작할 수 있다.

[0044] 파일 시스템 어댑터(306)는, 파일 시스템(302)과 연관된 파일 및/또는 파일 스트림에 연관된 속성들을, 파일 시스템(304)에 제공하는 속성 컴포넌트(310)를 포함할 수 있다. 예를 들어, 파일 시스템(302)은 파일 스트림 기반 저장 시스템을 이용하는데, 이 경우, 파일은, 읽기 전용, 압축, 보존 파일 등과 같은, 그러나 이에 한정되는 것은 아닌, 하나 이상의 속성을 가질 수 있고, 이 경우, 비트 및/또는 비트들이 이러한 속성들을 표현할 수 있다. 속성 컴포넌트(310)로 인해, 파일 시스템(302)의 파일 및/또는 파일 스트림을 표현하는 파일-지지 항목은 이러한 속성들을 파일 시스템(304)에 통합할 수 있다. 속성 컴포넌트(310)는 이러한 속성들을 개개의 파일-지지 항목을 가진 특성으로서 표현하거나 그리고/또는 대응시킬 수 있다는 것을 알 수 있을 것이다. 일례로서, 속성 컴포넌트(310)는 파일 시스템(302)으로부터의 호출을 통해 파일 스트림을 닫을 때 파일 속성에 대한 업데이트를 제공할 수 있다. 특성들은 (다음에서 논의되는) 파일 시스템(304)에 연관된 API를 통해 업데이트될 수

있다.

[0045] 속성 컴포넌트(310)는 부가적으로, 모든 항목들에 공통적인 속성들을 포함할 수 있는 내포 유형(nested type)인 항목 속성을 제공할 수 있다. 속성들은 파일 시스템(302) 및/또는 파일 시스템(304)과 연관된 API를 통해 동작중인 임의의 애플리케이션에 의해 설정될 수 있다. 다음의 표 1의 속성들이 파일 시스템(302)과의 애플리케이션 및/또는 호환성에 의해 이용될 일례로서 제공될 수 있다.

표 1

[0046]

이름	유형	설명
IsArchive	FileStorageSytem.Boolean	항목이 Archival을 위해 마킹되어 있다면, 참. 애플리케이션들이 이 특성을 설정하고 사용할 수 있다. FileStorageSytem은 이 특성의 시맨틱스를 해석/강제하지 않을 수도 있다. 이것은 디폴트로 참을 가진다.
IsHidden	FileStorageSytem.Boolean	항목이 은닉될 것으로 마킹되어 있다면, 참. 애플리케이션들은 이 속성을 사용해 항목을 나타내거나 은닉할 수 있다. FileStorageSytem은 이 특성의 시맨틱스를 해석/강제하지 않을 수도 있다. 이것은 디폴트로 거짓을 가진다.
IsReadOnly	FileStorageSytem.Boolean	항목이 읽기 전용으로 마킹된다. 애플리케이션들은 이들 항목들에 대한 업데이트들을 금지하기를 원할 수 있다. FileStorageSytem은 이 특성의 시맨틱스를 해석/강제하지 않을 수도 있다. 이것은 디폴트로 거짓을 가진다.
IsSystem	FileStorageSytem.Boolean	항목이 오퍼레이팅 시스템의 일부인지를 지시한다. 애플리케이션들은 이 특성을 업데이트할 수 있고 디스플레이 또는 다른 목적을 위해 이것을 해석하기를 원할 수 있다. FileStorageSytem은 이 특성의 시맨틱스를 해석/강제하지 않을 수도 있다. 이것은 디폴트로 거짓을 가진다.
IsTemporary	FileStorageSytem.Boolean	항목이 임시적인 저장을 위해 사용된다는 것을 지시한다. FileStorageSytem은 이 특성의 시맨틱스를 해석/강제하지 않을 수도 있다. 이것은 디폴트로 거짓을 가진다.

[0047] 파일 시스템 어댑터(306)는, 하나 이상의 특성이 항목, 파일-지지 항목, 및 지지 파일(예를 들어, 파일-지지 항목에 의해 표현되는 파일 시스템(302)의 파일) 사이에서 동기 상태를 유지할 수 있게 하는 메타데이터 핸들러 컴포넌트(312)를 포함할 수 있다. 예를 들어, 특성은 지지 파일 스트림에서 변경될 수 있는데, 이 경우, 메타데이터 핸들러 컴포넌트(312)는 새로운 값을 파일 시스템(304)내의 항목에서의 적절한 필드로 승격시킬 수 있다. 다른 예에서, 특성은 파일 시스템(304)에 연관된 API를 통해 변경될 수 있고; 메타데이터 핸들러 컴포넌트(312)는 새로운 값을 파일 시스템(302)내의 지지 파일 스트림으로 강등시킬 수 있다. 파일-지지 항목의 상이한 유형이 그와 연관된 상이한 메타데이터 핸들러 컴포넌트(312)를 가질 수 있다는 것을 알 수 있어야 한다.

[0048] 예를 들어, 승격(promotion)은, 파일이 애플리케이션에 의해 닫힌 후에, 메타데이터 핸들러 컴포넌트(312)에 의해 비동기적으로 수행될 수 있다. 한편, 강등은 비동기적일 수 있다. 특수한 항목 특성은, 승격된 특성들이 항목과 지지 파일 사이에서 동기인지 아닌지를 특정할 수 있다. 승격 상태는 항목 승격 오류 상태를 특정할 수 있다. 상태는 파일-지지 항목의 디폴트 스트림에서 파일이 닫히는 것과 비동기적으로 설정될 수 있다. 상태가 비-파일-지지 항목에 대해서는 NULL일 수 있다. 다음의 표 2는 승격 상태를 위해 가능한 값들을 도시한다.

표 2

[0049]

유형	값	상태	설명
Int32	0	READY	항목은 변경되었고; 승격은 성공적으로 실행되었다.
	1	STALE	항목은 변경되었고; 승격은 실행되지 않았다.
	2	ERROR_STALE	항목은 변경되었고; 승격은 실행되었으나 소정의 결정적인 이유로 실패하였다.
	3	ERROR_DELAYED	항목은 변경되었고; 승격은 실행되었으나 소정의 일시적인 이유로 실패하였다.

[0050] 파일 시스템 어댑터(306)가, 하나 이상의 항목 및/또는 파일-지지 항목이 저장되어 있는 항목 테이블을 이용할 수 있다는 것을 알 수 있을 것이다. 항목 테이블은 항목 테이블에 저장될 수 있는 개개 파일-지지 항목에 파일 스트림 및/또는 파일 속성을 연관짓는 파일 스트림 테이블 및/또는 파일 속성 테이블을 이용할 수 있다. 파일 스트림 테이블 및/또는 파일 속성 테이블은, (예를 들어, 하나 이상의 특성을 강화하거나 그리고/또는 채우기 위한 파일 스트림 및/또는 파일을 포함하는) 파일-지지 항목이 될 항목과 관련하여 개개 파일 스트림 및 파일 속성을 저장할 수 있다. 파일 속성들이 항목 테이블에 저장될 수 있다는 것을 알 수 있을 것이다. 파일 속성 시맨틱스는, 예를 들어, 업데이트 API에 의해 구현될 수 있다.

[0051] 또한, 전역적 항목 테이블은 파일-지지 항목을 위한 다음의 표 3의 추가물들을 포함할 수 있다.

표 3

[0052]

이름	유형	설명
(TableItem의 현재 컬럼들)		
LastAccessTime	FileStorageSystem.DateTime	파일 또는 디렉토리가 액세스된 최종 시간. 기본은 이 속성을 업데이트하지 않는 것이며; 필요에 따라 업데이트할 수 있다. 이것은 아주 높은 업데이트 빈도를 가지므로, 동기되지 않는다.
StreamSize	BigInt	스트림의 사이즈
AllocationSize	BigInt	파일 스트림을 위한 할당 사이즈
PromotionSize	Int32	승격 오류 플래그들

[0053] 예를 들어, 파일 스트림 테이블들은 다음의 표 4의 특성들을 가질 수 있다.

표 4

[0054]

이름	유형	설명
ItemId	널이 아닌 Uniqueidentifier	FileStorageSystem Item 식별 정보
RowId	널이 아닌 Uniqueidentifier ROWGUIDCOL 고유한 디폴트 (newid())	각각의 파일 스트림을 위한 고유한 식별자
Stream	varbinary(max) 널이 아닌 FILESTREAM	파일 스트림

[0055] 다음은 파일 스트림 저장을 이용하는 파일 시스템의 일례를 도시하는데, 이 경우, 32 비트 시스템이 구현되지만 이것이 주제 발명의 한정일 수는 없다. 다음의 표 5는 시스템(300) 및 파일 저장 시스템(FSS;예를 들어, 파일 시스템(304))에 의해 지원될 수 있는 32 비트 파일 시스템을 위한 속성들의 일례이다.

표 5

[0056]

32 비트 파일 속성	FSS의 이름	FSS 유형	디폴트 값	신뢰 레벨	32 비트 시맨틱스	FSS 시맨틱스
CreationTime	Created	DateTime	항목이 생성된 시간	Update API, FSA-Store APIs, WRITE_CONTROL	파일이 생성된 시간을 특정한다.	항목이 생성된 시간을 특정한다.
ChangeTime	Modified	DateTime	항목이 생성된 시간	Update API, FSA-Store APIs, WRITE_CONTROL	ChangeTime: 속성들, 파일 스트림 또는 디렉토리 내용들이 변경될 때 업데이트될 수 있다.	항목이 변경된 마지막 시간을 특정한다.
LastAccessTime	LastAccessTime	DateTime	항목이 생성된 시간	Update API, FSA-Store APIs, WRITE_CONTROL	파일 또는 디렉토리가 액세스된 새로운 시간을 특정한다. 액세스의 새로운 시간이 LastAccessTime 속성의 값보다 한시간 이상 크다면, 업데이트된다.	항목이 액세스된 마지막 시간을 특정한다. 액세스 시간이 LastAccessTime 속성의 값보다 한시간 이상 크다면, 업데이트된다.
FILE_ATTRIBUTE_HIDDEN	IsHidden	Bool	FALSE	WRITE_CONTROL	파일 또는 디렉토리가 은닉된다.	타깃 항목이 보통의 폴더 목록에 포함되지 않는다.
EndOfFile	StreamSize	Int32	0	FSS_Store API	절대값 EOF(end-of-file) 위치를 파일 시작으로부터의 바이트 오프셋으로서 특정한다. EOF까지의 바이트 오프셋을 특정한다.	절대값 EOF 위치를 지지 파일 스트림 시작으로부터의 바이트 오프셋으로서 특정한다. EOF까지의 바이트 오프셋을 특정한다.
AllocationSize	AllocationSize	Int32	0	FSS_Store API	파일 할당 사이즈를 바이트로 특정한다. 일반적으로, 이 값은 기초적인 물리 장치의 섹터 또는 클러스터 사이즈의 배수이다.	지지 파일 스트림 할당 사이즈를 바이트로 특정한다. 일반적으로, 이 값은 기초적인 물리 장치의 섹터 또는 클러스터 사이즈의 배이다.
FILE_ATTRIBUTE_ARCHIVE	IsArchive	Bool	TRUE	WRITE_CONTROL	파일 또는 디렉토리는 보존 파일 또는 디렉토리이다. 애플리케이션들은 이 속성을 사용해 백업 또는 삭제용 파일들을 마킹한다.	항목은 보존 파일 또는 디렉토리이다. 애플리케이션들은 이 속성을 사용해 백업 또는 삭제용 파일들을 마킹한다.
FILE_ATTRIBUTE_TEMPORARY	IsTemporary	Bool	FALSE	WRITE_CONTROL	Files: 파일은 임시적인 저장을 위해 사용 중이다.	항목은 임시적인 저장을 위해 사용 중이다.
File_ATTRIBUTE_READONLY	IsReadOnly	Bool	FALSE	WRITE_CONTROL	파일은 읽기-전용일 수 있다.	항목은 읽기-전용일 수 있다.

FILE_ATTRIBUTE_SYSTEM	IsSystem	Bool	FALSE	WRITE_CONTROL	파일 또는 디렉토리는 오퍼레이팅 시스템의 일부이거나 오퍼레이팅 시스템에 의해 배타적으로 사용된다.	항목은 오퍼레이팅 시스템의 일부이거나 오퍼레이팅 시스템에 의해 배타적으로 사용된다.
NumberOfLinks	Calculated Value				파일로의 하드 링크들의 수	루트 디렉토리가 0을 갖는 것을 제외하면, 항상 1이다.
DeletePending	Calculated Value				삭제 보류 중인 상태. TRUE는 파일 삭제가 요청되었다는 것을 지시한다.	32 비트를 통해 FBI에서 삭제가 보류 중이라면, TRUE. 그렇지 않으면 FALSE.
FILE_ATTRIBUTE_DEVICE	Calculated Value				Reserved: 사용하지 않는다.	항상 FALSE.
FILE_ATTRIBUTE_DIRECTORY	Calculated Value				핸들은 디렉토리를 식별한다.	파일-지지 항목들에 대해서는 FALSE. 그렇지 않으면 TRUE.
FILE_ATTRIBUTE_OFFLINE	IsGhosted	Bool	FALSE	이 플래그는 32 비트 API를 통해 설정할 수 없어야 한다.	이 속성은, 파일 데이터가 물리적으로 오프라인 저장 공간으로 이동되었다는 것을 지시한다. 이 속성은 Remote Storage, 계층 구조적 저장 관리 소프트웨어에 의해 사용된다.	항목이 고스팅되는지의 여부를 특정한다.

[0057] 상기 예를 계속하면, 다음의 표 6의 속성들은 부분적으로 지원될 수 있거나 그리고/또는 전체적으로 지원될 수 있다.

표 6

[0058]

32 비트 파일 속성	FSS의 이름	FSS 유형	디폴트 값	신뢰 레벨	시맨틱스	FSS 시맨틱스
LastWriteTime	Modified	DateTime	CreationTime	Update APIs, FSA-Store API, WRITE_CONTROL	파일 스트림이 변경되었거나 디렉토리의 목록 결과들이 변경된 마지막 시간을 특정한다.	항목이 변경된 마지막 시간을 특정한다.

[0059] 도 4는 파일 시스템을 이질적인 파일 시스템과 적응시키거나 그리고/또는 통합하는 것을 용이하게 하는 시스템(400)을 도시한다. 파일 시스템 어댑터(406)는 파일 시스템(402) 및 파일 시스템(404)에 일관성을 제공할 수 있다. 파일 시스템(402)은 파일 스트림에 기초할 수 있고 디렉토리, 폴더, 및 파일을 포함할 수 있다. 예를 들어, 파일 시스템(402)은 16 비트 파일 시스템, 32 비트 파일 시스템(예를 들어, FAT 파일 시스템)일 수 있는데, 이 경우, API(예를 들어, 16 비트 API, 32 비트 API)는 개개의 오퍼레이팅 시스템을 위해 이러한 파일 시스템을 이용한다. 파일 시스템(404)은 복합 유형 기반 시스템일 수 있고, 유형, 항목, 특성, 관계, 유형의 인스턴스, 컨테이너 등을 포함한다. 파일 시스템(404)은 정보 단위들을 연관된 메타데이터 및/또는 특성들을 갖춘 오브젝트들로서 표현한다는 것을 알 수 있을 것이다. 또한, 파일 시스템(402) 및 파일 시스템(404)은, 각각, 도 3, 도 2, 및 도 1의 파일 시스템들(302, 304, 202, 204, 102, 및 104)과 본질적으로 유사할 수 있다.

[0060] 파일 시스템 어댑터(406)는 파일 시스템(404)으로의 심리스 통합을 허용하기 위해 적어도 파일 시스템(402)의 파일 구조에 기초한 파일-지지 항목을 생성할 수 있다. 파일-지지 항목은 파일 시스템(404)에 의해 이용되는 특성들을 생성하거나 그리고/또는 첨부하기 위한 파일 및/또는 파일 스트림에 의해 채워지거나 그리고/또는 강화될 수 있다. 파일-지지 항목으로 인해, 파일 시스템(402)을 위한 API는, 파일-지지 항목이 그에 연관된 파일 및/또는 파일 스트림을 가진 유형이기 때문에, 파일 스트림을 필요로 하는 API 포맷에 상관없이, 파일 시스템(404)에 이용될 수 있다. 다시 말해, 파일-지지 항목은 파일 시스템(402)과 파일 시스템(404)간의 브리지이다.

[0061] 파일 시스템 어댑터(406)는, 파일-지지 항목의 특성에 동기화를 제공하는 고스트 컴포넌트(410)를 포함할 수 있다. 고스트 컴포넌트(410)는, 다른 파일-지지 항목의 레플리커(replica)이지만 파일 스트림을 포함하지 않는 고스팅된 항목을 생성할 수 있다. 또한, 고스트 컴포넌트(410)는, 고스팅된 항목을 생성하기 위한 그리고/또는 기존 항목을 고스팅된 항목으로 변환하기 위한 연산인 고스팅을 제공할 수 있다. 또한, 고스트 컴포넌트(410)는, 고스팅된 항목을 통상적인 항목으로 변환하기 위한 연산인 언고스팅(unghosting)을 제공할 수도 있다. 고스트 컴포넌트(410)는 파일 시스템 어댑터(406)로 통합되어 있지만, 고스트 컴포넌트(410)가 개별 컴포넌트 및/또는 독립형 유닛일 수도 있다는 것을 알 수 있을 것이다.

[0062] 다음의 표 7은 고스트 컴포넌트(410)의 능력들에 관련한 추가적인 설명을 제공할 수 있다.

표 7

[0063]

이름	방향	유형	설명
itemId	IN	Guid	Ghost/UnGhost로의 항목에 대한 Id.
isGhost	IN	BIT	항목이 고스팅되어야 할 것인지 아니면 언고스팅되어야 할 것인지를 특정한다.
concurrencyToken	IN	BIGINT	concurrencyToken은 항목에 대해 예상되는 병행성 토큰 값이다. 입력 값이 NULL이면, 점검이 수행되지 않는다. 디폴트 값은 NULL이다.

[0064] 일례에서, 고스트 컴포넌트(410)는 파일 시스템(404)을 위한 엔티티를 고스팅할 수 있다. 항목이 파일-지지 항목이 아니라면, 오류가 리턴될 수 있다. 항목이 이미 고스팅되었다면, 연산은 무연산(noop)이며 성공을 리턴한다. 고스트 컴포넌트(410)는 그 정보를 고스팅된 그 항목 트리의 모든 서브 엔티티들(예를 들어, 모든 매입 항목들, 관계들, 및/또는 확장자들)로 전파할 수 있고, 엔티티가 고스팅된 상태라는 것을 통지하기 위해 비트를 참으로 설정한다. 고스팅이 발생한 후에(예를 들어, 그것이 파일 스트림을 NULL로 설정한 후에), 고스팅된 항목을 위한 파일 스트림 내용들은 삭제될 수 있다.

[0065] 다른 예에서, 엔티티는 고스트 컴포넌트(410)에 의해 언고스팅될 수 있다. 항목이 파일-지지 항목이 아니라면, 오류가 리턴된다. 항목이 고스팅되지 않았다면, 연산은 비연산이며 성공을 리턴한다. 비트는, 엔티티가 고스팅되지 않은 엔티티 상태를 통지하기 위해 거짓으로 설정될 수 있다. 0 길이의 파일 스트림이 생성되어 파일 스트림 테이블에 추가될 수 있다. 언고스팅은 파일 스트림 내용들을 복구하지 않으며, 호출자는 (예를 들어, 바람직스럽게도, 실질적으로 유사한 트랜잭션의) 적절한 내용들로 파일 스트림을 명시적으로 채워야 한다. 일 인스턴스에서, 언고스트 연산을 위한 사용 패턴은, 트랜잭션 개방하기; 항목 언고스팅하기; 파일 스트림 내용들을 채우기/복구하기; 및 트랜잭션 실행하기일 수 있다.

[0066] 파일 시스템 어댑터(406)는 속성 터널 컴포넌트(ATC;412)를 포함할 수 있다. ATC(412)는 파일 시스템(404)내의 리소스들에 대한 명칭 변경 및/또는 보존을 용이하게 하기 위한 속성 터널링(attribute tunneling)을 제공한다. 파일 시스템(402)은 임시 파일을 이용하는데, 이 경우, 이러한 파일은 개방, 액세스, 및/또는 편집시에 생성된다. 파일이 파일 시스템(402)에 저장될 때, 임시 파일명 및 저장된 파일명은 교환된다. 예를 들어, 파일 food.doc가 파일 시스템(402)에 생성되어 액세스될 수 있다. 파일 시스템(402)내에 액세스할 때, 파일 f1.tmp가 생성될 수 있다. food.doc는 f2.tmp로 이동될 수 있는데, 이 경우, f1.tmp는 food.doc로 이동된다. 그 다음, 파일 f2.tmp는 파일 시스템(402)에서 삭제될 수 있다. 그러나, 정보의 단위들을 표현하기 위한 항목의 파일 시스템(404) 구조 및 이용에 기초해, 이름을 업데이트하기 위한 API 호출은 값비싼 리소스이다. 일반적으로, ATC(412)는, 연관된 파일 스트림이 2개의 파일-지지 항목들 사이에서 교환될 수 있게 한다. 예를 들어, 파일 스트림들의 테이블에 저장되어 있는 파일 스트림들이 ATC(412)에 의해 이용될 수 있는데, 이 경우, 2개 파일 스트림의 항목 ID가 교환될 수 있다. 파일 시스템 어댑터(406)에 통합되어 있는 것으로 도시되어 있지만, ATC(412)는 독립형 컴포넌트이거나, 임의의 적합한 컴포넌트에 통합되거나, 그리고/또는 이들의 조합일

수 있다.

[0067] 도 5는 복수개의 이질적인 파일 시스템들을 적응시키는 것을 용이하게 하기 위해 생성된 파일-지지 항목을 도시한다. 특히, 파일-지지 항목(502)은 파일 시스템(504)과 파일 시스템(506)간에 일관성을 제공할 수 있다. 파일 시스템(504)은, 각각, 도 4, 도 3, 도 2, 및 도 1의 파일 시스템들(402, 302, 202, 및 102)과 본질적으로 유사할 수 있다는 것을 알 수 있을 것이다. 또한, 파일 시스템(506)은, 각각, 도 4, 도 3, 도 2, 및 도 1의 파일 시스템들(404, 304, 204, 및 104)과 본질적으로 유사할 수 있다는 것을 알 수 있을 것이다.

[0068] 파일-지지 항목(502)은, (예를 들어, 업데이트 API 및/또는 앞서 도시된 메타데이터 핸들러 컴포넌트에 의해 구현되는) 시맨틱스를 제공하는 시맨틱 계층(508)을 제공할 수 있다. 파일-지지 항목(502) 또한 파일 스트림(510)을 포함한다. 파일 스트림(510)은 파일 시스템(504) 기술에 따른 것일 수 있다. 파일 시스템(504)에 연관된 애플리케이션들(예를 들어, 32 비트 애플리케이션 및/또는 API)은, 파일-지지 항목(502)을 이용해, 파일 시스템(506)에 저장되어 있는 파일들을 위한 파일 스트림들에서 직접적으로 동작할 수 있다. 캐시 관리, 스트리밍, 바이트-범위 로킹, 및 메모리-매핑과 같은, 그러나 이에 한정되는 것은 아닌, 많은 파일 시스템(504) 기술들이 파일 시스템(506)에 의해 재-구현될 필요는 없다. 기초적인 스트림들이 파일-지지 항목(502)에 연관된다. 또한, 파일 스트림(510)은 하나 이상의 ACL(access control list)에 의해 보호될 수 있다. 파일 스트림(510)은 다음의 표 8과 같은 파일 스트림 테이블로 표현될 수 있다.

표 8

[0069]

이름	유형	설명
ItemId	널이 아닌 Uniqueidentifier	FileStorageSystem Item의 식별 정보
RowId	널이 아닌 Uniqueidentifier 고유한 ROWGUIDCOL default(newid())	각각의 파일 스트림을 위한 고유한 식별자
Stream	varbinary(max) 널이 아닌 FILESTREAM	파일 스트림

[0070] 또한, 파일-지지 항목(502)은 파일 속성(512), 엔티티 상태(514), 변경 단위들(516), 및 항목 데이터(518)도 포함할 수 있다. 파일 속성(512)은 항목 테이블에 저장될 수 있는데, 이 경우, 파일 속성 시맨틱스는, 예를 들어, 업데이트 API에 의해 구현될 수 있다. 엔티티 상태(514)는 엔티티 상태 항목의 UDT(user-defined type) 필드에 저장된 속성들을 가질 수 있다. 엔티티 상태 속성들을 위한 시맨틱스는 (나타내지 않은) 업데이트 API 및/또는 메타데이터 핸들러 컴포넌트에 의해 구현될 수 있다. 또한, 파일-지지 항목(502)은 변경 단위들(516)을 이용해, 항목에 관련된 데이터를 표현하는 하나 이상의 단위 및/또는 항목 데이터(518)를 변경할 수 있다.

[0071] 일례에서, 다음의 표 9는 엔티티 상태(514)에 대한 하나 이상의 특징을 표현할 수 있다.

표 9

[0072]

이름	유형	설명
IsCompoundItemType	Boolean	엔티티가 Item 유형이고 스키마 선언에서 "복합 항목"으로 선언되면, 참. 엔티티가 항목 유형이지만 스키마 선언에서 복합 항목인 것으로 선언되지 않으면(예를 들어, 일반적인 컨테이너로 선언되면), 거짓. 엔티티가 항목 유형이 아니면(예를 들어, 엔티티가 링크, 항목 단편, 또는 확장자이면), 이 비트의 값은 그것의 소유자/소스 항목과 동일할 것이다. 이것은 디폴트로 참을 가진다.
RootCompoundItemId	Guid	이 엔티티가 포함되어 있는 최고 복합 Item의 ItemId를 저장한다. 이것이 복합 항목 계층 구조의 일부가 아니면, 값은 NULL이다(예를 들어, 이것은 "true generic container"이다). 이 엔티티가 최고의 Root 복합 항목이면, 값은 항목 Id와 동일할 수 있다. 이 엔티티가 복합 항목 계층 구조내에 포함되어 있다면, 값은 계층 구조상의 최고 RootCompound 항목의 항목 Id이다. 이 것은 디폴트로 거짓을 가진다.

IsRootFileBackedItem	Boolean	항목이 Root File-Backed 항목이면, 참. 항목이 File- Backed 항목이 아니거나 Root File-Backed 항목이 아니면(예를 들어, File-Backed 항목 트리에 포함되어 있는 항목이면), 거짓. 다른 모든 엔티티들(링크들, 확장자들 및 항목 파편들)에 대해, 이 플래그는 그들의 소유/소스 항목과 동일한 값을 가진다. 이것은 디폴트로 거짓을 가진다.
RootFileBackedItemId	Guid	이 엔티티가 포함되어 있는 Root File-Backed 항목의 ItemId를 저장한다. 이것이 File-Backed 항목의 일부가 아니면, 값은 Null이다. 이 엔티티가 Root File-Backed 항목이면, 값은 항목 Id와 동일할 수 있다. 이 엔티티가 File-Backed 항목 트리내에 포함되어 있다면, 값은 Root File- Backed 항목의 항목 Id이다.
IsGhost	Boolean	엔티티가 다른 항목의 고스팅된 레플리커(의 일부)이면, 참. 이것은 디폴트로 거짓을 가진다.

[0073] 도 6은 파일 스트림 연관형 파일 시스템 및 메타데이터 연관형 파일 시스템을 적응시키는 것을 용이하게 하는 시스템(600)을 도시한다. 파일 시스템 어댑터(606)는 파일 시스템(602;예를 들어, 32 비트 API를 이용하는 파일 스트림 기반 시스템) 및 파일 시스템(604;예를 들어, 정보의 단위들을 표현하기 위해 항목, 유형, 특성, 및 관계들을 이용하는 오브젝트 기반 시스템)간에 일관성 및/또는 심리스 통합을 제공할 수 있다. 파일 시스템(602)은, 각각, 도 5, 도 4, 도 3, 도 2, 및 도 1의 파일 시스템들(502, 402, 302, 202, 및 102)과 본질적으로 유사할 수 있다는 것을 알 수 있을 것이다. 또한, 파일 시스템(604)은, 각각, 도 5, 도 4, 도 3, 도 2, 및 도 1의 파일 시스템들(504, 404, 304, 204, 및 104)과 본질적으로 유사할 수 있다는 것을 알 수 있을 것이다.

[0074] DAG(directed acyclic graph) 컴포넌트(608)가 주제 발명과 함께 이용될 수 있다. DAG 컴포넌트(608)는 어떤 주기들도 갖추지 않은 방향 그래프를 이용할 수 있는데, 이 경우, 임의의 정점에 대해, 이러한 정점으로 시작하고 끝나는 방향 경로가 존재하지 않는다. 소스는 입력 에지들이 없는 정점인 한편, 싱크는 출력 에지들이 없는 정점이다. 유한한 DAG는 하나 이상의 소스 및 하나 이상의 싱크를 가진다. DAG의 길이는 길이(예를 들어, 최장 방향 경로의 에지들의 수)이다. 파일 시스템(602)이 DAG를 통합하지는 않지만, 파일 시스템(604)은 파일 시스템 어댑터(606)를 통해 DAG 컴포넌트(608)를 이용할 수 있다는 것을 알 수 있을 것이다. 독립형 유닛으로 도시되어 있지만, DAG 컴포넌트(608)는 파일 시스템 어댑터(606) 및/또는 임의의 적합한 다른 컴포넌트에 통합될 수 있다. DAG 컴포넌트(608)를 이용하는 것에 의해, 파일은 동시에 2개의 디렉토리들을 설명할 수 있다. 예를 들어, 파일이 디렉토리로부터 삭제되고 물리적으로 제거될 때, 파일은 이에 대해 여전히 부모 상태일 수 있다.

[0075] 도 7은 파일 시스템을 이질적인 파일 시스템에 적응시키는 것을 용이하게 하기 위해 지능을 이용하는 시스템(700)을 도시한다. 시스템(700)은, 모두가 선행 도면들에서 설명된 개개 컴포넌트들과 실질적으로 유사할 수 있는 파일 시스템(702), 파일 시스템(704), 파일 시스템 어댑터(706), 및 인터페이스(108)를 포함할 수 있다. 시스템(700)은 지능 컴포넌트(708)를 더 포함한다. 지능 컴포넌트(708)는 파일 시스템 어댑터(706)에 의해 이용되어 2 이상의 이질적인 파일 시스템들에 적응하는 것을 용이하게 할 수 있다. 예를 들어, 지능 컴포넌트(708)는 파일 시스템(702)의 파일 구조를 판정하는 것을 용이하게 하는데 이용될 수 있다.

[0076] 지능 컴포넌트(708)는 이벤트들 및/또는 데이터에 의해 포착되는 한 세트의 관찰들로부터 시스템, 환경, 및/또는 사용자의 상태들에 관한 추론 또는 추측을 제공할 수 있다. 추론은 특정 컨텍스트 또는 동작을 식별하는데 이용될 수 있거나, 예를 들어, 상태들에 대한 확률 분포를 발생시킬 수 있다. 추론은 확률적일 수 있는데, 즉, 관심있는 상태들에 대한 확률 분포의 계산은 데이터 및 이벤트들의 고려에 기초할 수 있다. 또한, 추론은 한 세트의 이벤트들 및/또는 데이터로부터 더 높은-레벨의 이벤트들을 구성하는데 이용되는 기술들을 의미할 수도 있다. 이러한 추론은, 이벤트들이 시간적으로 밀접하게 상관되어 있는지에 따라, 그리고 이벤트들 및 데이터가 하나 또는 수 개의 이벤트 및 데이터 소스들로부터 유래하는지에 따라, 한 세트의 관찰된 이벤트들 및/또는 저장된 이벤트 데이터로부터 새로운 이벤트들 또는 동작들의 구성을 초래할 수 있다. (예를 들어, SVM(support vector machines), 신경망들, 전문가 시스템들, 베이시안 신뢰 네트워크들, 퍼지 로직, 데이터 퓨전 엔진들 등) 다양한 분류 (명시적으로 그리고/또는 암시적으로 훈련된) 방식들 및/또는 시스템들이 주제 발명과 관련하여 자동화된 그리고/또는 추론된 동작을 수행하는 것에 이용될 수 있다.

[0077] 분류자는 입력 속성 벡터(x=(x1, x2, x3, x4, xn))를 입력이 클래스에 속하는 신뢰도로 매핑하는 함수, 즉, f(x) = confidence(class)이다. 이러한 분류는, 사용자가 자동적으로 수행되기를 원하는 동작을 진단하거나 추

론하기 위해 확률 및/또는 통계-기반 분석(예를 들어, 분석 유틸리티들 및 비용들로의 팩토링)을 이용할 수 있다. SVM은 이용될 수 있는 분류자의 일례이다. SVM은 가능한 입력 공간에서 초곡면을 찾아내는 것에 의해 동작하는데, 이러한 초곡면은 비-트리거링 이벤트들로부터 트리거링 기준들을 분리하고자 한다. 직관적으로, 이것이 인접한 데이터를 테스트하기에는 정확한 분류이지만, 데이터를 트레이닝하는 것과는 다르다. 다른 방향 및 비향향 모델 분류 접근 방법들로는, 예를 들어, 나이브 베이즈, 베이시안 네트워크들, 판정 트리들, 신경망들, 퍼지 로직 모델들, 및 독립성의 상이한 패턴들을 제공하는 확률적 분류 모델들을 들 수 있다. 또한, 여기에서 사용되는 분류는 우선 순위의 모델들을 개발하는데 이용되는 통계적 회귀를 포함한다.

[0078] 프레젠테이션 컴포넌트(710)는 사용자와 파일 시스템 어댑터(706)에 커플링되어 있는 임의 컴포넌트간의 상호작용을 용이하게 하기 위해 다양한 유형들의 사용자 인터페이스들을 제공할 수 있다. 도시된 바와 같이, 프레젠테이션 컴포넌트(710)는 파일 시스템 어댑터(706)와 함께 이용될 수 있는 별개의 엔티티이다. 그러나, 프레젠테이션 컴포넌트(710) 및/또는 유사한 뷰 컴포넌트들이 파일 시스템 어댑터(706) 및/또는 독립형 유닛으로 통합될 수 있다는 것을 알 수 있을 것이다. 프레젠테이션 컴포넌트(710)는 하나 이상의 GUI들(graphic user interfaces), 명령 라인 인터페이스들 등을 제공할 수 있다. 예를 들어, 사용자에게 데이터를 로드, 이식, 판독하기 위한 영역 또는 수단을 제공하는 GUI가 렌더링될 수 있고, GUI는 이러한 결과들을 제시하기 위한 영역을 포함할 수 있다. 이들 영역들은 대화 상자들, 정적 제어들, 드롭-다운 메뉴들, 리스트 박스들, 팝업 메뉴들, 편집 제어들, 콤보 박스들, 라디오 버튼들, 체크 박스들, 푸시 버튼들, 및 그래픽 박스들을 포함하는 공지의 텍스트 및/또는 그래픽 영역들을 구비할 수 있다. 또한, 네비게이션을 위한 수직 및/또는 수평 스크롤 바들 및 영역이 시청 가능한지를 판정하기 위한 툴바 버튼들과 같은, 프레젠테이션을 용이하게 하기 위한 유틸리티들이 이용될 수 있다. 예를 들어, 사용자는 파일 시스템 어댑터(706)에 커플링되어 있는 컴포넌트들 중 하나 이상과 상호 작용할 수 있다.

[0079] 사용자는, 예를 들어, 마우스, 롤러 볼, 키패드, 키보드, 펜, 및/또는 음성 활성화 장치와 같은, 다양한 장치들을 통해서도 정보를 선택하고 제공하기 위한 영역들과 상호 작용할 수 있다. 통상적으로는, 키보드상의 푸시 버튼 또는 엔터키와 같은 메커니즘이 서치를 개시하기 위해 정보를 후속적으로 입력하는데 이용될 수 있다. 그러나, 본 발명이 그렇게 한정적이지 않다는 것을 알 수 있을 것이다. 예를 들어, 체크 박스를 단지 하일라이팅하는 것도 정보 전달을 개시할 수 있다. 다른 예에서는, 명령 라인 인터페이스가 이용될 수 있다. 예를 들어, 명령 라인 인터페이스는 (예를 들어, 디스플레이상의 텍스트 메시지 및 음조를 통해) 텍스트 메시지를 제공함으로써 사용자에게 정보를 촉구할 수 있다. 그 다음, 사용자는, 인터페이스 프롬프트에서 제공되는 옵션에 대응되는 영숫자 입력 또는 프롬프트에서 제시된 질문에 대한 답변과 같은, 적합한 정보를 제공할 수 있다. 명령 라인 인터페이스가 GUI 및/또는 API와 관련하여 이용될 수도 있다는 것을 알 수 있을 것이다. 또한, 명령 라인 인터페이스는 하드웨어(예를 들어, 비디오 카드들) 및/또는 한정된 그래픽 지원 및/또는 낮은 대역폭의 통신 채널을 갖춘 디스플레이들(예를 들어, 흑백 및 EGA)과 관련하여 이용될 수도 있다.

[0080] 도 8 내지 도 10은 주제 발명에 따른 방법들을 도시한다. 설명의 간략화를 위해, 방법들은 일련의 단계들로서 도시되고 설명된다. 주제 발명이 도시된 단계들에 의해 그리고/또는 단계들의 순서에 의해 한정되지 않는다는 것을, 예를 들어, 단계들은, 여기에서 제시되고 설명되지 않은 다른 단계들과 함께, 다양한 순서들로 그리고/또는 동시에 발생할 수 있다는 것을 알 수 있을 것이다. 또한, 도시된 단계들 모두가 주제 발명에 따른 방법들을 구현하는데 필요한 것은 아닐 수도 있다. 또한, 당업자들이라면, 방법들이 상태도를 통해 상관되어 있는 일련의 상태들 또는 이벤트들로서 다르게 표현될 수도 있다는 것을 알 수 있을 것이다.

[0081] 도 8은 2 이상의 이질적인 파일 시스템들에 적응하는 것 및/또는 2 이상의 이질적인 파일 시스템들에 일관성을 제공하는 것을 용이하게 하는 방법(800)을 도시한다. 제1 파일 시스템은 파일 저장 시스템일 수 있는데, 이 경우, 구조는 파일 스트림 및/또는 디렉토리에 기초한다. 제1 파일 시스템은, 애플리케이션 및/또는 API가 파일 스트림을 강화하여 더 많은 구조를 제공할 수 있도록 하기 위해, 파일 스트림을 이용할 수 있다. 제2 파일 시스템은 좀더 복잡한 모델일 수 있는데, 이 경우, 항목, 서브-항목, 특성, 및 관계는 복합 유형들의 인스턴스들로서 제2 파일 시스템내의 정보를 표현할 수 있도록 정의된다. 항목은, 독립적으로 보안, 직렬화, 동기화, 복사, 백업/복구 등이 이루어질 수 있는, 제2 파일 시스템에서의 일관성의 최소 단위로서 정의될 수 있다. 항목은 유형의 인스턴스인데, 이 경우, 제2 파일 시스템의 모든 항목들은 항목들의 전역적인 단일 범위로 저장될 수 있다. 제2 파일 시스템은 하나 이상의 항목 및/또는 컨테이너 구조에 기초할 수 있다. 제2 파일 시스템은 파일들에 항목들로서 매입되어 있는 풍부한 메타데이터를 노출시키는 저장 플랫폼일 수 있다.

[0082] 참조 번호 802에서는, 파일 스트림에 기초한 제1 파일 시스템이 분석될 수 있다. 이러한 분석은 일관성을 제공하는데 이용될 파일 구조 및 내용을 판정할 수 있다. 참조 번호 804에서는, 분석에 적어도 부분적으로 기초해,

파일-지지 항목이 생성될 수 있다. 예를 들어, 분석은 32 비트 파일 저장 시스템 및 이러한 구조와 연관되어 있는 리프 노드들의 수를 판정할 수 있다. 그에 따라, 제1 파일 시스템에 연관된 각각의 리프 노드에 대해 파일-지지 항목이 생성될 수 있는 한편, 항목은 리프 노드가 아닌 파일 및/또는 디렉토리를 표현하는데 이용될 수 있다. 파일-지지 항목은, 그것의 특성들 중 일부 또는 전부를 채우기 위해 파일 및/또는 파일 스트림에 의존하는 제2 파일 시스템내의 항목일 수 있다. 파일-지지 항목은 제1 파일 시스템과 제2 파일 시스템간의 브리지로서 동작할 수 있다. 참조 번호 806에서는, 파일 스트림 및 파일을 이용하는 제1 파일 시스템 및 제1 파일 시스템에 관련될 수 있는 파일 및/또는 파일 스트림을 포함하는 파일-지지 항목을 통합하는 제2 파일 시스템에 적어도 부분적으로 기초해, 파일-지지 항목은 2개의 이질적인 파일 시스템들간에 일관성을 제공하는데 이용될 수 있다.

[0083] 도 9는 하나의 파일 시스템을 이질적인 파일 시스템에 적응시키기 위한 방법(900)을 도시한다. 참조 번호 902에서는, 파일 및/또는 파일 스트림을 그것의 구조로서 이용하는 제1 파일 시스템이 분석된다. 이러한 분석으로부터, 참조 번호 904에서는, 제1 파일 시스템의 리프 노드가 아닌 각각의 디렉토리 및/또는 파일에 대해 항목이 생성된다. 참조 번호 906에서는, 그에 대해 파일-지지 항목이 생성되는 리프 노드가 제1 파일 시스템에 존재하는지를 판정할 수 있다. 파일-지지 항목은 유형의 복합 인스턴스들을 이용해 정보 단위를 표현하는 제2 파일 시스템에 연관된 항목이다. 참조 번호 908에서는, 파일을 제2 파일 시스템의 항목으로 표현하기 위해, 파일-지지 항목이 파일, 파일 스트림, 속성, 보안 특징 등으로 강화되거나 그리고/또는 채워진다. 참조 번호 910에서는, 파일-지지 항목이 제1 파일 시스템을 제2 파일 시스템에 적응시키는데 이용된다.

[0084] 도 10은 파일 스트림 구조형 파일 시스템 및 메타데이터 연관형 파일 시스템을 적응시키는 것을 용이하게 하는 방법(1000)을 도시한다. 참조 번호 1002에서는, (예를 들어, 제1 파일 시스템이라고 하는) 파일 스트림 구조형 파일 시스템이 분석된다. 참조 번호 1004에서는, 제1 파일 시스템의 분석에 적어도 부분적으로 기초해, 파일-지지 항목이 발생될 수 있다. 파일-지지 항목은, 파일 및/또는 파일 스트림이 그것의 특성들 중 하나 이상을 강화할 수 있는, 제2 파일 시스템에 연관된 항목이다. 1006에서는, 파일-지지 항목을 이용해 제1 파일 시스템을 제2 파일 시스템에 적응시킬 수 있다.

[0085] 참조 번호 1008에서는, 제1 파일 시스템과 제2 파일 시스템간의 속성 무결성(attribute integrity) 및 동기화를 보장하기 위해, 고스팅이 제공될 수 있다. 예를 들어, 고스팅된 항목은, 그것이 다른 파일-지지 항목의 레플리커이지만 파일 스트림은 포함하지 않도록, 생성될 수 있다. 또한, 고스팅된 항목을 생성하기 위한 그리고/또는 기존 항목을 고스팅된 항목으로 변환하기 위한 연산인 고스팅이 제공될 수도 있다. 부가적으로, 고스팅된 항목을 통상적인 항목으로 변환하기 위한 연산인 언고스팅이 이용될 수도 있다. 참조 번호 1010에서는, 제1 파일 시스템으로서의 속성이 제2 파일 시스템에 연관된 파일-지지 항목과 통합되거나 그리고/또는 제2 파일 시스템에 연관된 파일-지지 항목과 함께 이용될 수 있다. 제1 파일 시스템은, 읽기 전용, 압축, 보존 파일 등일 수 있지만, 이에 한정되는 것은 아닌, 속성을 포함할 수 있는데, 이 경우, 비트 및/또는 비트들이 이러한 속성들을 표현할 수 있다. 속성들은 파일-지지 항목을 통해 제2 파일 시스템에 통합될 수 있다는 것을 알 수 있을 것이다.

[0086] 참조 번호 1012에서는, 속성 터널링 및/또는 DAG가 이용될 수 있다. 속성 터널링은 리소스들의 보존을 허용하기 위한 2 이상의 파일-지지 항목들 사이에서의 파일 스트림 교환 및/또는 제2 파일 시스템의 임시 파일들 사이에서의 명칭 변경을 제공할 수 있다. DAG는, 주기들이 없는 접속 항목들의 그래프를 제공할 수 있다. 참조 번호 1014에서는, 파일-지지 항목을 이용하는 것에 의해, 제1 파일 시스템으로부터의 API가 제2 파일 시스템에 이용될 수 있는데, 이는 심리스 통합을 제공한다. 다시 말해, 파일-지지 항목은 항목과 연관된 메타데이터 특성들 뿐만 아니라 항목과 연관된 파일 스트림을 포함한다. 파일-지지 항목들은 제2 파일 시스템과 연관된 API 뿐만 아니라 제1 파일 시스템의 API 부분에 의해서도 조작될 수 있다.

[0087] 주제 발명을 구현하기 위한 추가적인 맥락을 제공하기 위해, 도 11-도 12 및 다음의 논의는, 주제 발명의 다양한 태양들이 구현될 수 있는 적합한 컴퓨팅 환경에 대한 간략하고 일반적인 설명을 제공하기 위한 것이다. 로컬 컴퓨터 및/또는 원격 컴퓨터에서 실행되는 컴퓨터 프로그램의 컴퓨터-실행 가능 명령어들의 일반적인 맥락에서 본 발명을 상술하였지만, 당업자들이라면, 본 발명이 다른 프로그램 모듈들의 조합으로도 구현될 수 있다는 것을 알 수 있을 것이다. 일반적으로, 프로그램 모듈들은 특정한 태스크들을 수행하거나 특정한 추상적 데이터 형들을 구현하는 루틴들, 프로그램들, 오브젝트들, 컴포넌트들, 데이터 구조들 등을 포함한다.

[0088] 또한, 당업자들이라면, 본 방법들이, 단일-프로세서 또는 멀티-프로세서 컴퓨터 시스템들, 미니컴퓨터들, 메인프레임 컴퓨터들 뿐만 아니라 퍼스널 컴퓨터들, 핸드-헬드 컴퓨팅 장치들, 마이크로프로세서-기반 및/또는 프로그램 가능한 상용 전자 장치들 등을 포함하는, 이들 각각이 하나 이상의 연관 장치들과 통신할 수 있는, 다른

컴퓨터 시스템 구성들로도 실시될 수 있다는 것을 알 수 있을 것이다. 본 발명의 도시된 태양들은, 통신 네트워크를 통해 링크되어 있는 원격 프로세싱 장치들에 의해 소정 태스크들이 수행되는 분산 컴퓨팅 환경들에서 실시될 수도 있다. 그러나, 본 발명의, 전부는 아니지만, 일부 태양들은 독립형 컴퓨터들에서도 실시될 수 있다. 분산 컴퓨팅 환경에서, 프로그램 모듈들은 로컬 및/또는 원격 메모리 저장 장치들에 배치될 수 있다.

[0089] 도 11은, 주제 발명이 상호 작용할 수 있는 예시적-컴퓨팅 환경(1100)의 개략적인 블록도이다. 시스템(1100)은 하나 이상의 클라이언트(들)(1110)를 포함한다. 클라이언트(들)(1110)는 하드웨어 및/또는 소프트웨어(예를 들어, 스레드들, 프로세스들, 컴퓨팅 장치들)일 수 있다. 또한, 시스템(1100)은 하나 이상의 서버(들)(1120)를 포함한다. 서버(들)(1120)는 하드웨어 및/또는 소프트웨어(예를 들어, 스레드들, 프로세스들, 컴퓨팅 장치들)일 수 있다. 예를 들어, 서버들(1120)은 주제 발명을 이용하는 것에 의해 변환들을 수행하기 위한 하우스 스레드들(house threads)일 수 있다.

[0090] 클라이언트(1110)와 서버(1120)간의 가능한 일 통신은 2 이상의 컴퓨터 프로세스들 사이에서 전송되도록 적응되어 있는 데이터 패킷의 형태일 수 있다. 시스템(1100)은 클라이언트(들)(1110)와 서버(들)(1120) 사이에서의 통신을 용이하게 하는데 이용될 수 있는 통신 프레임워크(1140)를 포함한다. 클라이언트(들)(1110)는, 클라이언트(들)(1110)에 국한된 정보를 저장하는데 이용될 수 있는 하나 이상의 클라이언트 데이터 저장 장치(들)(1150)에 접속되어 있다. 마찬가지로, 서버(들)(1120)는, 서버들(1120)에 국한된 정보를 저장하는데 이용될 수 있는 하나 이상의 서버 데이터 저장 장치(들)(1130)에 접속되어 있다.

[0091] 도 12를 참조하면, 본 발명의 다양한 태양들을 구현하기 위한 예시적 환경(1200)은 컴퓨터(1212)를 포함한다. 컴퓨터(1212)는 프로세싱 유닛(1214), 시스템 메모리(1216), 및 시스템 버스(1218)를 포함한다. 시스템 버스(1218)는, 시스템 메모리(1216)를 포함하지만, 그에 한정되는 것은 아닌, 시스템 컴포넌트들을 프로세싱 유닛(1214)에 결합시킨다. 프로세싱 유닛(1214)은 이용 가능한 다양한 프로세서들 중 하나일 수 있다. 듀얼 마이크로프로세서들 및 다른 멀티프로세서 아키텍처들 또한 프로세싱 유닛(1214)으로서 이용될 수 있다.

[0092] 시스템 버스(1218)는 메모리 버스 또는 메모리 컨트롤러, 주변 장치 버스 또는 외장형 버스, 및/또는 ISA(Industry Standard Architecture) 버스, MCA(Micro Channel Architecture) 버스, EISA(Enhanced ISA) 버스, IDE(Intelligent Drive Electronics), VLD(VESA Local Bus), PCI(peripheral Component Interconnect), 카드 버스, USB(Universal Serial Bus), AGP(Advanced Graphics Port), PCMCIA(Personal Computer Memory Card International Association) 버스, Firewire(IEEE 1394), 및 SCSI(Small Computer System Interface)를 포함하지만, 이에 한정되는 것은 아닌, 이용 가능한 다양한 버스 아키텍처들 중 하나를 사용하는 로컬 버스를 포함하는 몇가지 유형의 버스 구조들 중 하나일 수 있다.

[0093] 시스템 메모리(1216)는 휘발성 메모리(1220) 및 비휘발성 메모리(1222)를 포함한다. 스타트-업(start-up) 동안과 같은 때에, 컴퓨터(1212)내의 요소들 사이에서 정보 전달을 돕는 기본적인 루틴들을 포함하는 BIOS(basic input/output system)는 비휘발성 메모리(1222)에 저장된다. 한정이 아닌 일례로써, 비휘발성 메모리(1222)는 ROM(read only memory), PROM(programmable ROM), EPROM(electrically programmable ROM), EEPROM(electrically erasable PROM), 또는 플래시 메모리를 포함할 수 있다. 휘발성 메모리(1220)는, 외장형 캐시 메모리로서 동작하는 RAM(random access memory)을 포함한다. 한정이 아닌 일례로써, RAM은 SRAM(static RAM), DRAM(dynamic RAM), SDRAM(synchronous DRAM), DDR SDRAM(double data rate SDRAM), ESDRAM(enhanced SDRAM), SLDRAM(Synchlink DRAM), RDRAM(Rambus direct RAM), DRDRAM(direct Rambus dynamic RAM), 및 RDRAM(Rambus dynamic RAM)과 같은, 많은 형태들로 이용될 수 있다.

[0094] 컴퓨터(1212)는 분리형/비분리형, 휘발성/비휘발성 컴퓨터 저장 매체들도 포함한다. 도 12는, 예를 들어, 디스크 저장 장치(1224)를 도시한다. 디스크 저장 장치(1224)는 자기 디스크 드라이브, 플로피 디스크 드라이브, 테이프 드라이브, 재즈 드라이브, 짚 드라이브, LS-100 드라이브, 플래시 메모리 카드, 또는 메모리 스틱과 같은 장치들을 포함하지만, 이에 한정되는 것은 아니다. 또한, 디스크 저장 장치(1224)는, CD-ROM(compact disk ROM device), CD-R 드라이브(CD recordable drive), CD-RW(CD rewritable) 드라이브, 또는 DVD-ROM(digital versatile disk ROM) 드라이브와 같은 광학 디스크 드라이브를 포함하지만, 이에 한정되는 것은 아닌, 다른 저장 매체들과 별도로 또는 결합하여 저장 매체들을 포함할 수 있다. 디스크 저장 장치들(1224)의 시스템 버스(1218)로의 접속을 용이하게 하기 위해, 통상적으로, 인터페이스(1226)와 같은 분리형 또는 비분리형 인터페이스가 사용된다.

[0095] 도 12는 사용자들과 적합한 동작 환경(1200)에서 설명된 기본적인 컴퓨터 리소스들간의 매개자로서 동작하는 소프트웨어를 설명한다는 것을 알 수 있을 것이다. 이러한 소프트웨어는 오퍼레이팅 시스템(1228)을 포함한다.

디스크 저장 장치(1224)에 저장될 수 있는 오퍼레이팅 시스템(1228)은 컴퓨터 시스템(1212)의 리소스들을 제어하고 할당하는 기능을 한다. 시스템 애플리케이션들(1230)은 시스템 메모리(1216)나 디스크 저장 장치(1224)에 저장되어 있는 프로그램 모듈들(1232) 및 프로그램 데이터(1234)를 통한 오퍼레이팅 시스템(1228)에 의한 리소스들의 관리를 이용한다. 주제 발명이 다양한 오퍼레이팅 시스템들 또는 오퍼레이팅 시스템들의 조합으로써 구현될 수 있다는 것을 알 수 있을 것이다.

[0096] 사용자는 입력 장치(들)(1236)를 통해 컴퓨터(1212)에 명령들 또는 정보를 입력한다. 입력 장치들(1236)은 마우스와 같은 포인팅 장치, 트랙볼, 스틸러스, 터치 패드, 키보드, 마이크로폰, 조이스틱, 게임 패드, 위성 안테나, 스캐너, TV 튜너 카드, 디지털 카메라, 디지털 비디오 카메라, 웹 카메라 등을 포함하지만, 이에 한정되는 것은 아니다. 이들 및 다른 입력 장치들은 인터페이스 포트(들)(1238)를 경유하고 시스템 버스(1218)를 통해 프로세싱 유닛(1214)에 접속된다. 인터페이스 포트(들)(1238)로는, 예를 들어, 직렬 포트, 병렬 포트, 게임 포트, 및 USB를 들 수 있다. 출력 장치(들)(1240)는 입력 장치(들)(1236)와 동일한 유형의 포트들 중 일부를 사용한다. 따라서, 예를 들어, USB 포트는 컴퓨터(1212)에 입력을 제공하는데 사용될 수 있고, 컴퓨터(1212)로부터의 정보를 출력 장치(1240)로 출력하는데도 사용될 수 있다. 출력 어댑터(1242)는, 다른 출력 장치들(1240) 중에, 모니터들, 스피커들, 및 프린터들과 같은, 특수한 어댑터들을 필요로 하는 일부 출력 장치들(1240)이 존재한다는 것을 도시하기 위해 제공된다. 출력 어댑터들(1242)은, 한정이 아닌 일례로써, 출력 장치(1240)와 시스템 버스(1218)간에 접속 수단을 제공하는 비디오 및 사운드 카드들을 포함한다. 원격 컴퓨터(들)(1244)와 같은 다른 장치들 및/또는 장치들의 시스템들은 입력 및 출력 능력들 모두를 제공한다는 것에 주목해야 한다.

[0097] 컴퓨터(1212)는, 원격 컴퓨터(들)(1244)와 같은, 하나 이상의 원격 컴퓨터들로의 논리적 접속들을 사용하는 네트워크 환경에서 동작할 수 있다. 원격 컴퓨터(들)(1244)는 퍼스널 컴퓨터, 서버, 라우터, 네트워크 PC, 워크스테이션, 마이크로프로세서 기반형 어플라이언스, 피어 장치, 또는 다른 공통 네트워크 노드 등일 수 있으며, 통상적으로 컴퓨터(1212)와 관련하여 상술된 요소들 중 많은 것을 또는 그 전부를 포함한다. 간략화를 위해, 원격 컴퓨터(들)(1244)에는 메모리 저장 장치(1246)만이 도시되어 있다. 원격 컴퓨터(들)(1244)는 네트워크 인터페이스(1248)를 통해 컴퓨터(1212)에 논리적으로 접속된 다음 통신 접속(1250)에 의해 물리적으로 접속된다. 네트워크 인터페이스(1248)는 LAN(local area network) 및 WAN(wide area network)과 같은 유선 및/또는 무선 통신 네트워크들을 포함한다. LAN 기술들로는 FDDI(Fiber Distributed Data Interface), CDDI(Copper Distributed Data Interface), 이더넷, 토큰 링 등을 들 수 있다. WAN 기술들로는 점 대 점 링크들, ISDN(Integrated Services Digital Networks)과 같은 회로 스위칭 네트워크들과 그것의 변형들, 패킷 스위칭 네트워크들, 및 DSL(Digital Subscriber Lines)을 들 수 있지만, 이에 한정되는 것은 아니다.

[0098] 통신 접속(들)(1250)은 네트워크 인터페이스(1248)를 버스(1218)에 접속시키는데 이용되는 하드웨어/소프트웨어를 의미한다. 명료한 설명을 위해 통신 접속(1250)이 컴퓨터(1212) 내부에 도시되어 있지만, 컴퓨터(1212) 외장형일 수도 있다. 네트워크 인터페이스(1248)로의 접속을 위해 필요한 하드웨어/소프트웨어는, 단지 일례로써, 보통의 전화기 등급의 모뎀들, 케이블 모뎀들 및 DSL 모뎀들을 포함하는 모뎀들, ISDN 어댑터들, 및 이더넷 카드들과 같은, 내장형 및 외장형 기술들을 포함한다.

[0099] 상술된 것은 주제 발명의 예들을 포함한다. 물론, 주제 발명을 설명하기 위해 컴포넌트들 또는 방법들의 가능한 모든 조합들을 설명하는 것은 불가능하지만, 당업자라면, 주제 발명의 추가적인 많은 조합들 및 변경들이 가능하다는 것을 알 수 있을 것이다. 따라서, 주제 발명은 첨부된 청구항들의 범위내에 해당되는 이러한 모든 변화들, 변형들, 및 변경들을 포함하는 것이다.

[0100] 특히 상술된 컴포넌트들, 장치들, 회로들, 시스템들 등에 의해 수행되는 다양한 펑크션들과 관련하여, 이러한 컴포넌트들을 설명하는데 사용되는 ("수단"에 대한 참조를 포함하는) 용어들은, 다르게 지시되지 않는다면, 본 발명의 여기에서 도시된 예시적 태양들에서의 펑크션을 수행하는, 개시된 구조와 구조적으로 등가가 아니라고 하더라도, 설명된 컴포넌트의 특정된 펑크션을 수행하는 (예를 들어, 기능적으로 등가인) 임의의 컴포넌트에 대응되는 것이다. 이러한 관점에서, 본 발명은, 본 발명의 다양한 방법들에 대한 단계들 및/또는 이벤트들을 수행하기 위한 컴퓨터-실행 가능 명령어들을 가진 시스템 뿐만 아니라 컴퓨터-판독 가능 매체를 포함한다는 것도 알 수 있을 것이다.

[0101] 또한, 본 발명의 특정 사양이 몇가지 구현들 중 오직 하나에 대해서만 개시되었을 수도 있지만, 이러한 사양은, 임의의 소정 또는 특정 애플리케이션을 위해 필요하며 유용할 수 있는 다른 구현들의 하나 이상의 다른 사양들과 조합될 수 있다. 또한, "포함한다" 및 "포함하는"이라는 용어들과 그것의 변형들이 상세한 설명 또는 청구항들에서 사용된다면, 이러한 용어들은 "포함하는"이라는 용어와 유사한 방식으로 포괄적인 의미로 사용된다.

발명의 효과

[0102] 따라서, 주제 발명에 따르면, 2 이상의 이질적인 파일 시스템들을 적응시키는 것을 용이하게 하는 시스템 및/또는 방법이 제공된다.

도면의 간단한 설명

[0001] 도 1은 파일 스트림 연관형 파일 시스템 및 메타데이터 연관형 파일 시스템에 적응하는 것을 용이하게 하는 예시적 시스템의 블록도를 도시한다.

[0002] 도 2는 일관성(uniformity)을 제공하기 위해 2 이상의 이질적인 파일 시스템들에 적응하는 것을 용이하게 하는 예시적 시스템의 블록도를 도시한다.

[0003] 도 3은 파일 스트림 연관형 파일 시스템 및 메타데이터 연관형 파일 시스템에 적응하는 것을 용이하게 하는 예시적 시스템의 블록도를 도시한다.

[0004] 도 4는 2 이상의 이질적인 파일 시스템들을 동기화하는 것을 용이하게 하는 예시적 시스템의 블록도를 도시한다.

[0005] 도 5는 하나의 파일 시스템을 이질적인 파일 시스템에 적응시키는 것을 용이하게 하는 파일-지지 항목의 블록도를 도시한다.

[0006] 도 6은 파일 스트림 연관형 파일 시스템 및 메타데이터 연관형 파일 시스템에 적응하는 것을 용이하게 하는 예시적 시스템의 블록도를 도시한다.

[0007] 도 7은 파일 스트림 연관형 파일 시스템 및 메타데이터 연관형 파일 시스템에 적응하는 것을 용이하게 하는 예시적 시스템의 블록도를 도시한다.

[0008] 도 8은 2 이상의 이질적인 파일 시스템들을 적응시켜 일관성을 제공하기 위한 예시적 방법을 도시한다.

[0009] 도 9는 하나의 파일 시스템을 이질적인 파일 시스템에 적응시키기 위한 예시적 방법을 도시한다.

[0010] 도 10은 파일 스트림 연관형 파일 시스템 및 메타데이터 연관형 파일 시스템에 적응하기 위한 예시적 방법을 도시한다.

[0011] 도 11은 주제 발명의 신규한 태양들이 이용될 수 있는 예시적인 네트워킹 환경을 도시한다.

[0012] 도 12는 주제 발명에 따라 이용될 수 있는 예시적인 동작 환경을 도시한다.

[0013] <도면의 주요 부분에 대한 부호의 설명>

[0014] 100, 200, 300, 400, 500, 600, 700 : 시스템

[0015] 102, 202, 302, 402, 504, 602, 702 : 제1 파일 시스템

[0016] 104, 204, 304, 404, 506, 604, 704 : 제2 파일 시스템

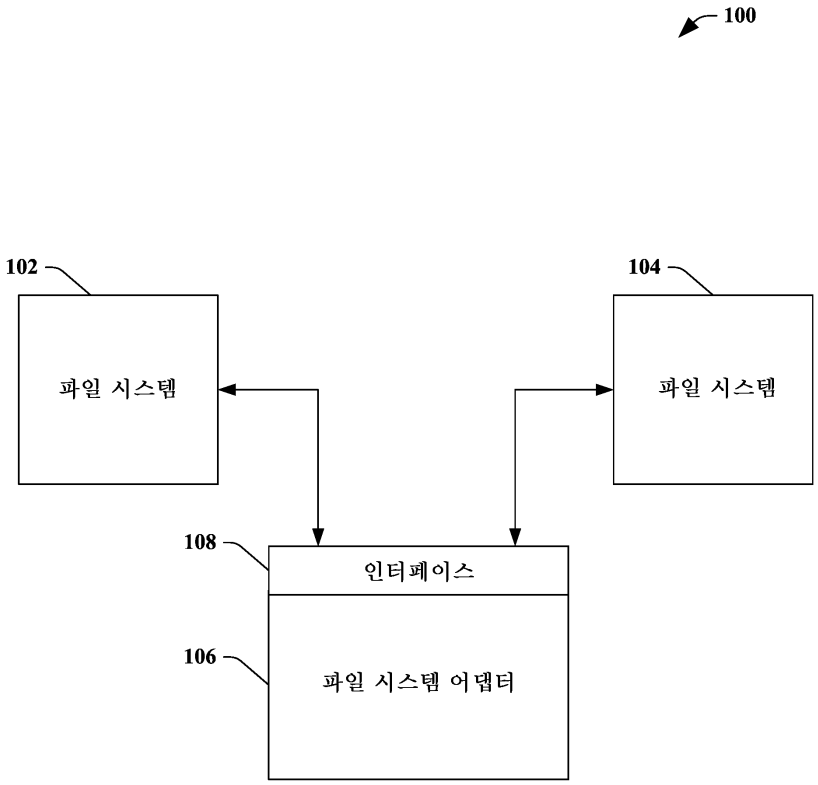
[0017] 106, 206, 306, 406, 606, 706 : 파일 시스템 어댑터

[0018] 108 : 인터페이스

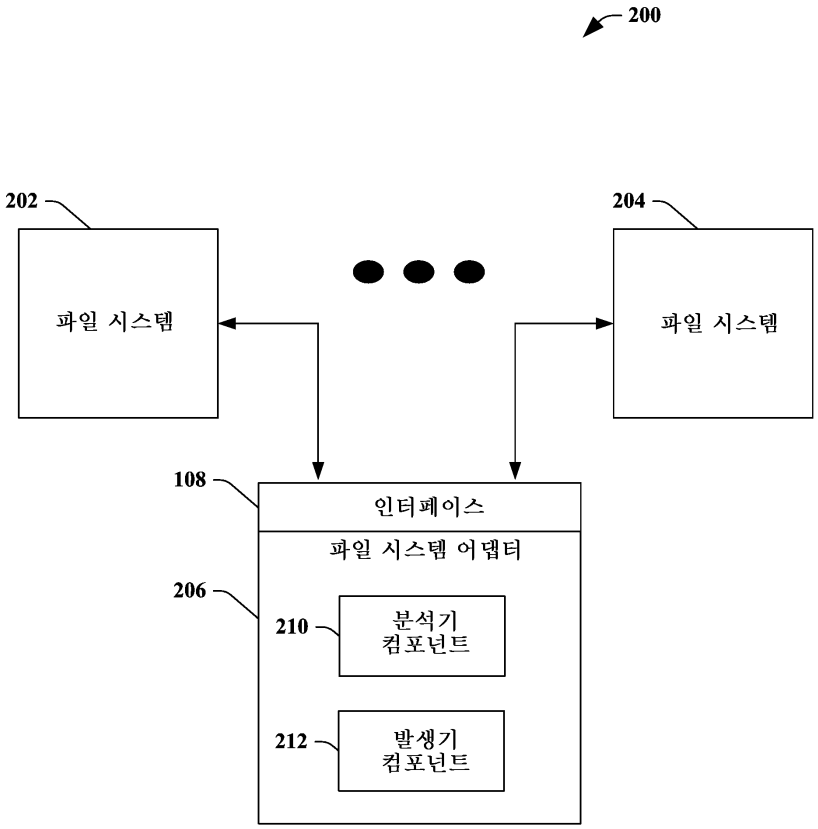
[0019] 502 : 파일-지지 항목

도면

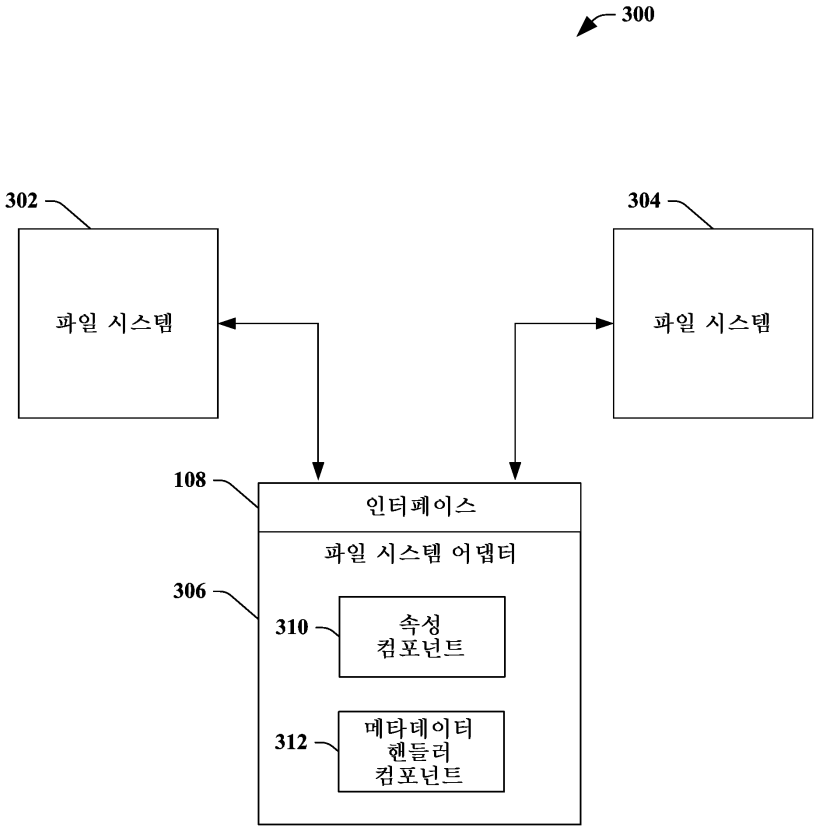
도면1



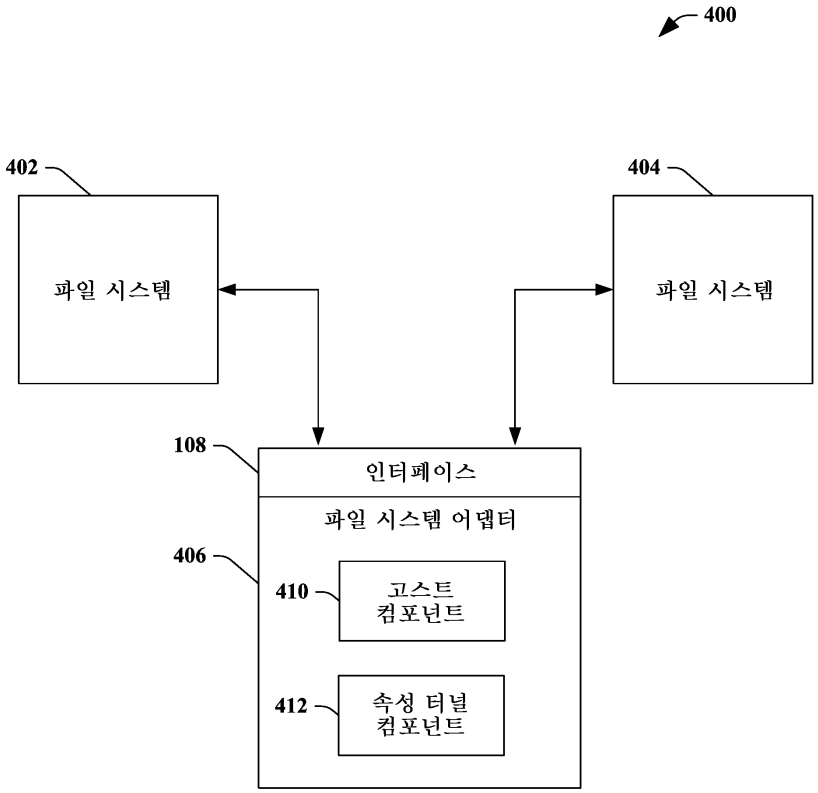
도면2


200
202
파일 시스템
204
파일 시스템
108
인터페이스
파일 시스템 어댑터
206
210
분석기
컴포넌트
212
발생기
컴포넌트

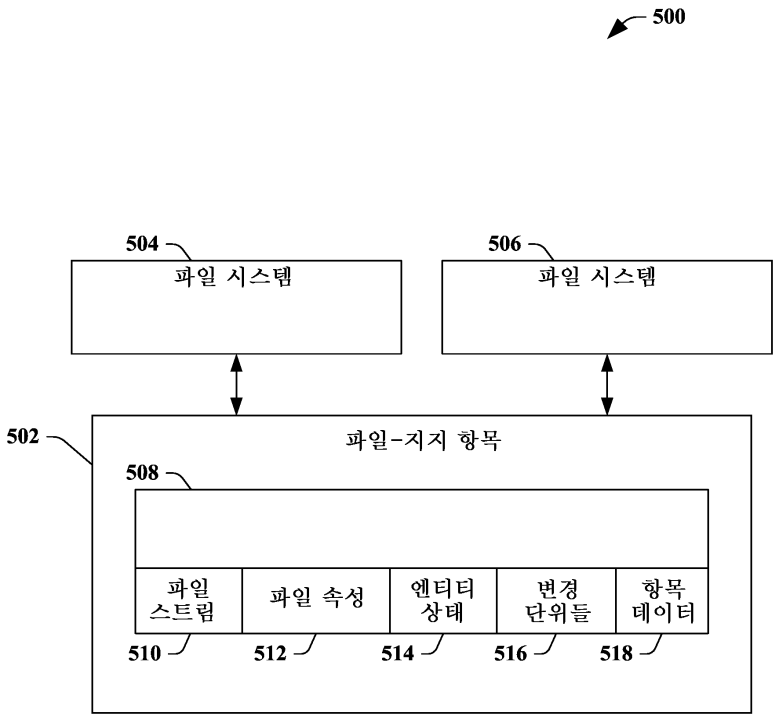
도면3


300
302
304
파일 시스템
파일 시스템
108
인터페이스
파일 시스템 어댑터
306
310
속성
컴포넌트
312
메타데이터
핸들러
컴포넌트

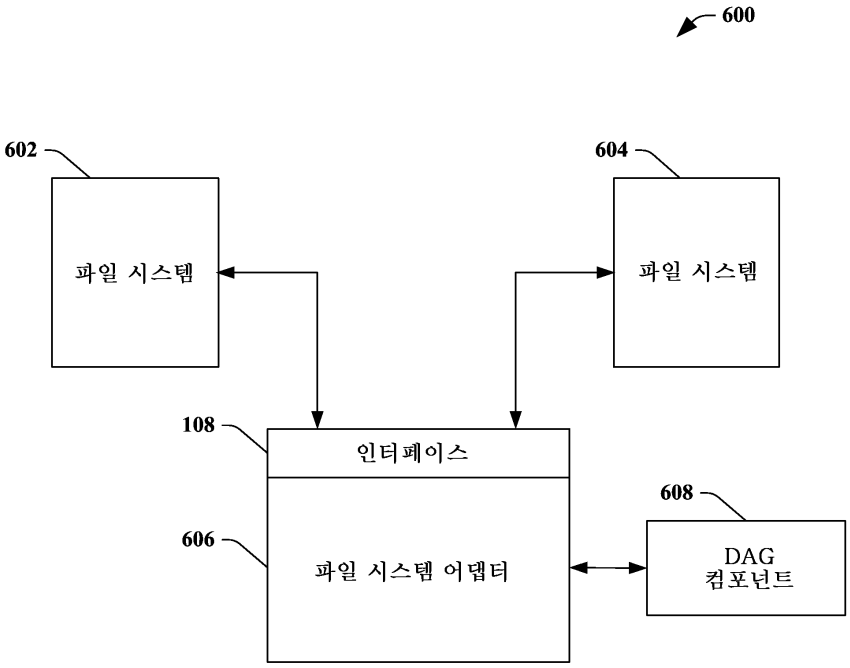
도면4


400
402
파일 시스템
404
파일 시스템
108
인터페이스
파일 시스템 어댑터
406
410
고스트 컴포넌트
412
속성 터널 컴포넌트

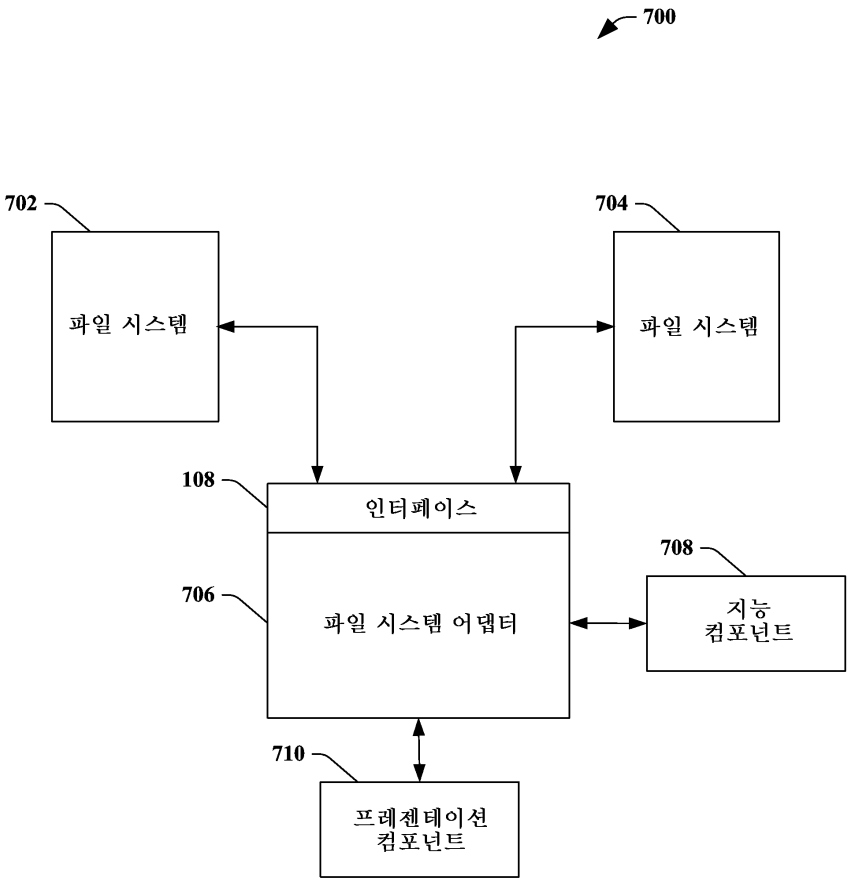
도면5


500
504
파일 시스템
506
파일 시스템
502
파일-지지 항목
508
파일 스트림
파일 속성
엔티티 상태
변경 단위들
항목 데이터
510
512
514
516
518

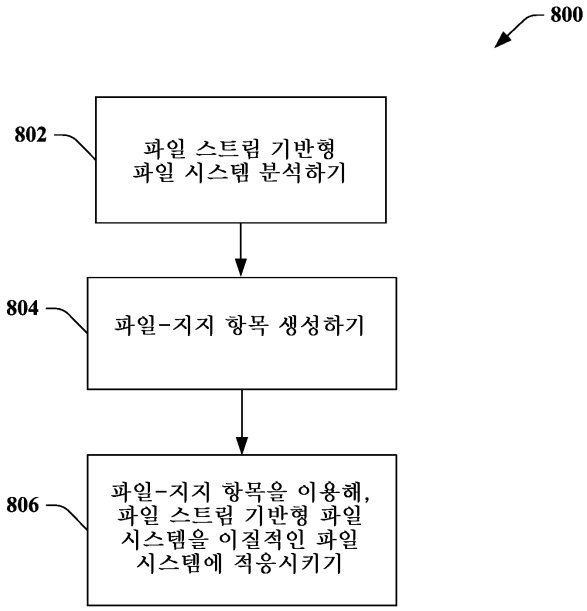
도면6


600
602
파일 시스템
604
파일 시스템
108
인터페이스
606
파일 시스템 어댑터
608
DAG
컴포넌트

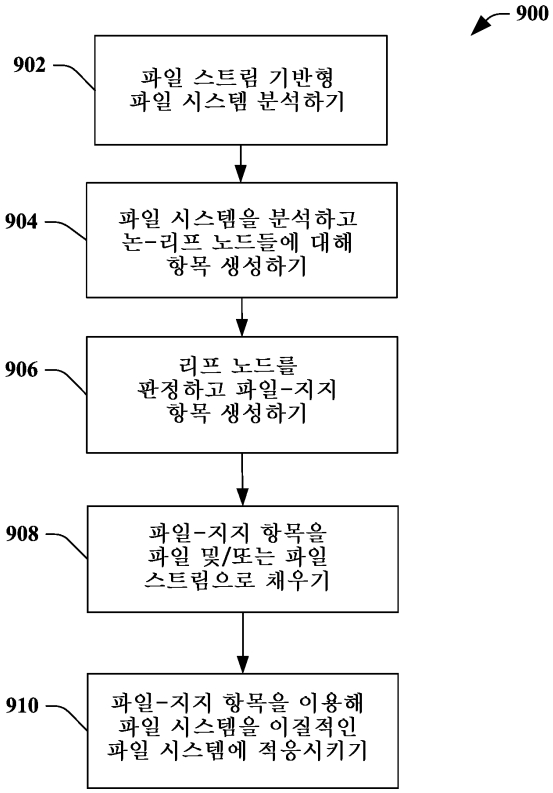
도면7


700
702
파일 시스템
704
파일 시스템
108
인터페이스
706
파일 시스템 어댑터
708
지능
컴포넌트
710
프레젠테이션
컴포넌트

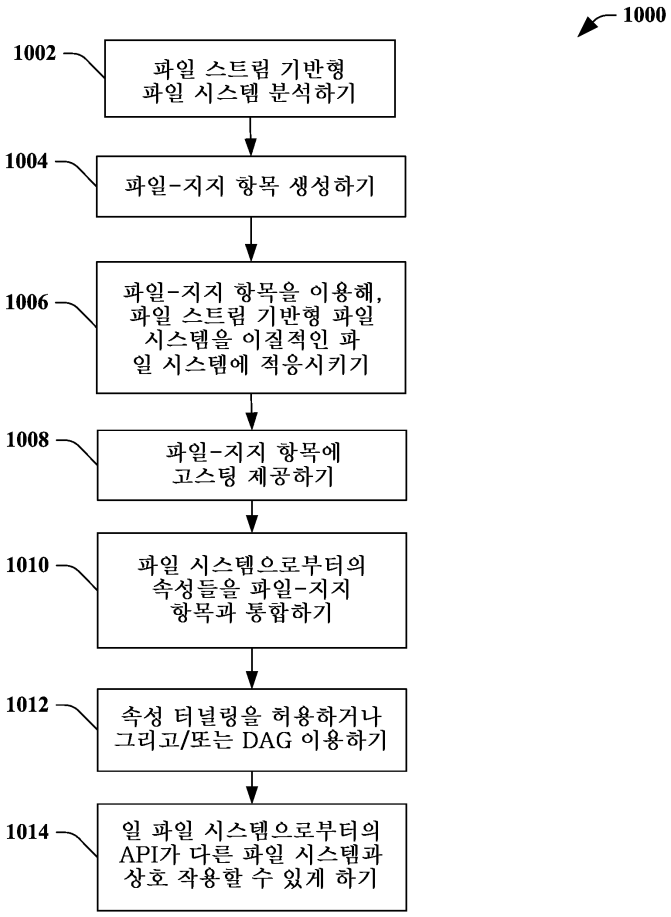
도면8


800
802
파일 스트림 기반형 파일 시스템 분석하기
804
파일-지지 항목 생성하기
806
파일-지지 항목을 이용해, 파일 스트림 기반형 파일 시스템을 이질적인 파일 시스템에 적응시키기

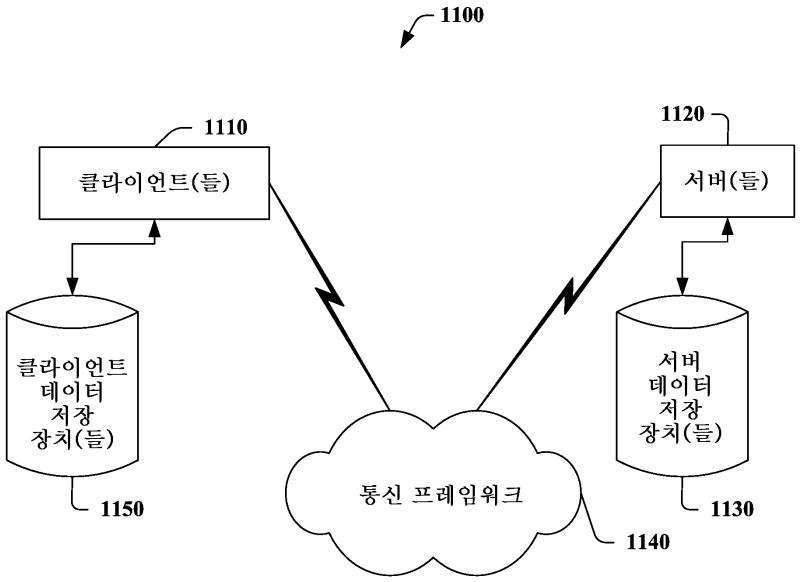
도면9


900
902
파일 스트림 기반형 파일 시스템 분석하기
904
파일 시스템을 분석하고 논-리프 노드들에 대해 항목 생성하기
906
리프 노드를 판정하고 파일-지지 항목 생성하기
908
파일-지지 항목을 파일 및/또는 파일 스트림으로 채우기
910
파일-지지 항목을 이용해 파일 시스템을 이질적인 파일 시스템에 적응시키기

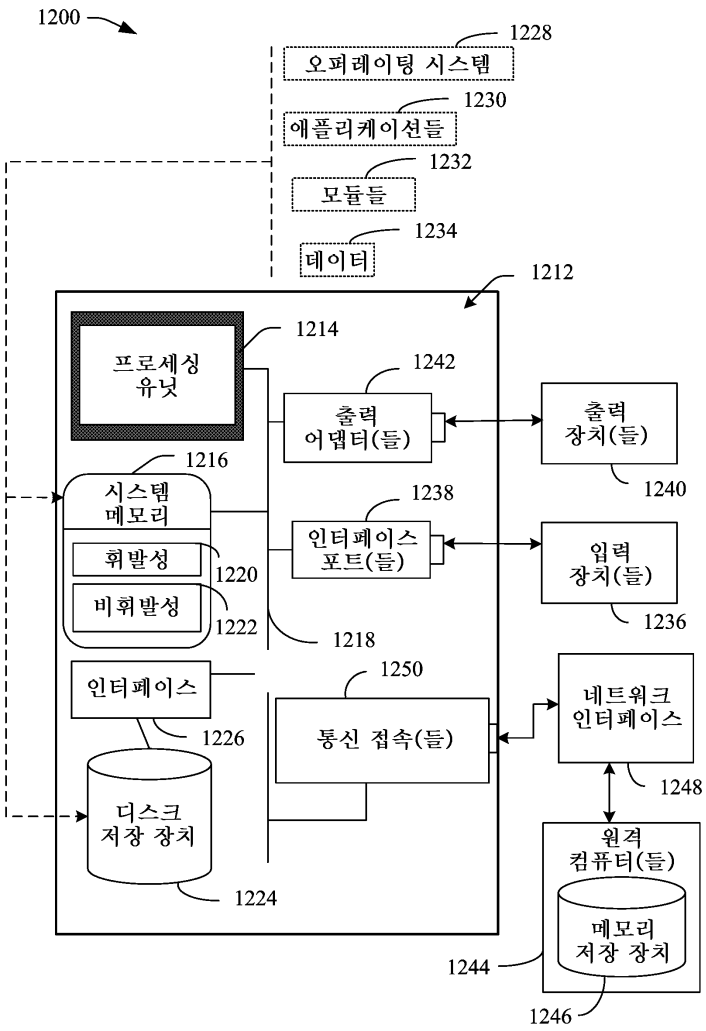
도면10


1000
1002
파일 스트림 기반형 파일 시스템 분석하기
1004
파일-지지 항목 생성하기
1006
파일-지지 항목을 이용해, 파일 스트림 기반형 파일 시스템을 이질적인 파일 시스템에 적응시키기
1008
파일-지지 항목에 고스팅 제공하기
1010
파일 시스템으로부터의 속성들을 파일-지지 항목과 통합하기
1012
속성 터널링을 허용하거나 그리고/또는 DAG 이용하기
1014
일 파일 시스템으로부터의 API가 다른 파일 시스템과 상호 작용할 수 있게 하기

도면11


1100
1110
클라이언트(들)
1120
서버(들)
클라이언트 데이터 저장 장치(들)
1150
통신 프레임워크
1140
서버 데이터 저장 장치(들)
1130

도면12


1200
1228
오퍼레이팅 시스템
1230
애플리케이션들
1232
모듈들
1234
데이터
1212
1214
프로세싱 유닛
1242
출력 어댑터(들)
출력 장치(들)
1240
1216
시스템 메모리
휘발성
1220
비휘발성
1222
1218
1238
인터페이스 포트(들)
입력 장치(들)
1236
인터페이스
1226
1250
통신 접속(들)
네트워크 인터페이스
1248
디스크 저장 장치
1224
원격 컴퓨터(들)
메모리 저장 장치
1244
1246