(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2013년01월21일
(11) 등록번호 10-1224703
(24) 등록일자 2013년01월15일

(51) 국제특허분류(Int. Cl.)
G06F 9/06 (2006.01) *G06F 9/00* (2006.01)
G06F 9/44 (2006.01)
(21) 출원번호 10-2006-0002250
(22) 출원일자 2006년01월09일
심사청구일자 2011년01월05일
(65) 공개번호 10-2006-0095451
(43) 공개일자 2006년08월31일
(30) 우선권주장
11/168,589 2005년06월28일 미국(US)
60/657,536 2005년02월28일 미국(US)
(56) 선행기술조사문헌
US20040044655 A1
US20030188198 A1

(73) 특허권자
마이크로소프트 코포레이션
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이
(72) 발명자
헌터, 제이슨 티.
미국 98052 워싱턴주 레드몬드 원 마이크로소프트 웨이마이크로소프트 코포레이션 내
두브하시, 케다르나스 에이.
미국 98052 워싱턴주 레드몬드 원 마이크로소프트 웨이마이크로소프트 코포레이션 내
스카리아, 사이몬
미국 98052 워싱턴주 레드몬드 원 마이크로소프트 웨이마이크로소프트 코포레이션 내
(74) 대리인
제일특허법인

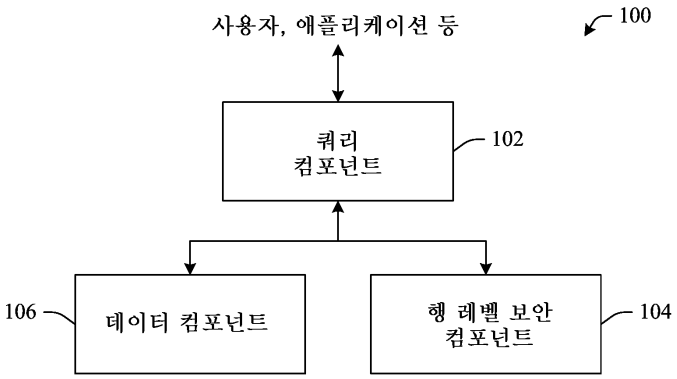
전체 청구항 수 : 총 16 항 심사관 : 복진요

(54) 발명의 명칭 **데이터 액세스를 용이하게 하는 시스템 및 방법**

(57) 요 약

접속점으로부터 스토어의 사용자별 추상화(a per user abstraction)를 생성하는 시스템이 개시되어 있다. 주체의 액세스 권한에 기초하여 계층 구조적으로 보안된 컨테인먼트의 뷰 세트를 필터링하는 것이 본 발명의 새로운 특징들 중 하나이다. 본 발명은 잠재적으로 이종의(heterogeneous) 보안 설명자들로 다수의 컨테이너 계층 구조들에 걸치는 이 집합체(aggregation)에 대해 작용할 수 있는 프리미티브들의 컬렉션을 제공할 수 있다. 이 모델은 도메인 내의 모든 액세스 가능한 아이템들을 발견하기 위해 컨테이너 계층 구조를 횡단(traverse)할 필요로 줄일 수 있다.

대 표 도 - 도1



특허청구의 범위

청구항 1

데이터 액세스를 용이하게 하는 시스템으로서,

소프트웨어 컴포넌트들을 실행하는 컴퓨터 프로세서를 포함하고,

상기 소프트웨어 컴포넌트는,

접속점(connection point)으로부터 데이터 스토어(data store)의 추상화(abstraction)를 생성하는 쿼리 컴포넌트(query component);

적어도 하나의 행 레벨 액세스 권한(row-level access permission)에 기초하여 상기 추상화를 제한하고, 보안 정책을 상기 데이터 스토어 내의 적어도 하나의 행과 관련시키며, 보안 설명자(security descriptor)를 보안 설명자 식별자(SDID)에 매핑하는 보안 설명자 테이블 및 상기 SDID를 상기 SDID의 해시 값(hash value)에 매핑하는 단일 인스턴스 테이블(single instance table)을 포함하여, 상기 단일 인스턴스 테이블 및 상기 보안 설명자 테이블이 함께, SHA-1 해시 알고리즘으로부터 상기 SDID로 이어서 바이너리로의 완전한 매핑을 제공하고 상기 테이블들이 단일 인스턴싱 체크(single instancing check)를 수행하는 데 사용되도록 하는, 행 레벨 보안 컴포넌트(row-level security component); 및

전파(propagation)가 적절한지를 판정하고, 필요하다면 계층 구조의 루트에서 상기 보안 정책을 설정하고 상기 계층 구조 내의 적어도 하나의 자식에게 상기 보안 정책을 전파하는 컴포넌트

를 포함하고,

상기 SDID는 상기 데이터 스토어의 행의 각각에 대해 저장되어, 사용자가 아이템을 생성하면 상기 보안 설명자가 상기 계층 구조 내의 부모로부터 상속되는 시스템.

청구항 2

제1항에 있어서,

상기 데이터 스토어는 계층 구조로 조직되고 상기 쿼리 컴포넌트는 상기 계층 구조를 초월(transcend)하는 시스템.

청구항 3

제1항에 있어서,

상기 소프트웨어 컴포넌트는,

액세스 제어 시행 정책(access control enforcement policy)과 관련하여 이용되는 신뢰할 수 있는 아이덴티티 설정 시스템(trustworthy identity establishment system)을 제공하는 컴포넌트를 더 포함하는 시스템.

청구항 4

제1항에 있어서,

상기 소프트웨어 컴포넌트는,

상기 제한된 추상화를 렌더링하는 렌더링 컴포넌트(rendering component)를 더 포함하는 시스템.

청구항 5

제1항에 있어서,

상기 행 레벨 보안 컴포넌트는 보안 정책을 상기 데이터 스토어 내의 적어도 하나의 행과 관련시키는 시스템.

청구항 6

제5항에 있어서,

상기 데이터 스토어 내의 각 행은 단일 객체(single object)를 포함하는 시스템.

청구항 7

제6항에 있어서,

상기 보안 정책은 액세스 제어 리스트(access control list:ACL) 및 보안 설명자 중 적어도 하나인 시스템.

청구항 8

제7항에 있어서,

상기 객체는 계층 구조적 조직으로 조직된 컨테이너(container)와 데이터 엘리먼트 중 적어도 하나인 시스템.

청구항 9

제8항에 있어서,

상기 보안 정책을 전파하는 컴포넌트는 상기 객체에 대한 유효한 보안 설명자를 계산하기 위해 상기 객체 및 부모의 보안 설명자를 지능적으로 이용하는 시스템.

청구항 10

제1항에 있어서,

상기 SDID는 상기 보안 설명자를 가리키는(point to) 정수 값인 시스템.

청구항 11

제1항에 있어서,

상기 해시 값은 SHA-1 해시 알고리즘을 통하여 생성되는 시스템.

청구항 12

제1항에 있어서,

상기 소프트웨어 컴포넌트는,

사용자가 자동적으로 수행되기를 원하는 액션을 예지하거나 추론하기 위해 확률 기반 분석 및 통계 기반 분석 중 하나 이상을 이용하는 인공 지능(AI) 컴포넌트를 더 포함하는 시스템.

청구항 13

데이터 스토어 내의 데이터에 대한 액세스 제어를 제공하는 방법으로서,

상기 데이터를 계층 구조적 조직으로 조직하는 단계;

상기 계층 구조적 조직을 초월(transcend)하는 단계;

상기 계층 구조적 조직의 루트에서 보안 정책을 설정하는 단계;

부모 보안 설명자에 적어도 일부분 기초하여 상기 계층 구조적 조직 내의 적어도 하나의 자식에게 상기 보안 정책을 지능적으로 전파하는 단계;

상기 데이터 스토어의 접속점 추상화를 생성하는 단계;

행 레벨 보안 정책에 적어도 일부분 기초하여 상기 추상화를 상기 데이터의 서브세트로 제한하기 위해 상기 행 레벨 보안 정책을 적용하는 단계 - 상기 행 레벨 보안 정책은 ACL과 보안 설명자 중 적어도 하나를 상기 데이터 스토어 내의 적어도 하나의 행과 관련시킴 -;

보안 설명자를 보안 설명자 식별자(SDID)에 매핑하는 단계;

SHA-1 해시 알고리즘으로부터 상기 SDID로 이어서 바이너리로 완전한 매핑이 수행되고 단일 인스턴싱 체크를 수행하기 위해 사용되도록 상기 SDID를 상기 SDID의 해시 값에 매핑하는 단계;

사용자가 아이템을 생성하면 상기 보안 설명자가 계층 구조 내의 부모로부터 상속되도록 상기 SDID를 상기 데이터 스토어의 행의 각각에 대해 저장하는 단계; 및

상기 사용자로의 디스플레이를 통해 상기 제한된 추상화를 렌더링하는 단계

를 포함하는 방법.

청구항 14

제13항에 있어서,

상기 행 레벨 보안 정책을 적용하는 것과 관련하여 이용되는 신뢰할 수 있는 아이덴티티 설정 시스템을 설정하는 단계를 더 포함하는 방법.

청구항 15

데이터 스토어 내의 데이터의 액세스 제어를 용이하게 하는 시스템으로서,

상기 데이터를 트리형 구조로 조직하기 위한 수단;

상기 트리형 구조를 초월(transcend)하기 위한 수단;

상기 트리형 구조의 루트에서 보안 정책을 설정하기 위한 수단;

상기 트리형 구조 내의 적어도 하나의 자식에게 상기 보안 정책을 지능적으로 전파하기 위한 수단;

부모의 보안 정책 및 상기 자식의 보안 정책에 적어도 일부분 기초하여 상기 전파된 보안 정책을 적용하기 위한 수단;

상기 데이터 스토어 내의 적어도 하나의 행과 관련되는 하나 이상의 보안 정책들에 적어도 일부분 기초하여 상기 데이터 스토어의 접속점 추상화를 필터링하기 위한 수단;

보안 설명자를 보안 설명자 식별자(SDID)에 매핑하기 위한 수단;

SHA-1 해시 알고리즘으로부터 상기 SDID로 이어서 바이너리로 완전한 매핑이 수행되고 단일 인스턴싱 체크를 수행하기 위해 사용되도록 상기 SDID를 상기 SDID의 해시 값에 매핑하기 위한 수단;

사용자가 아이템을 생성하면 상기 보안 설명자가 계층 구조 내의 부모로부터 상속되도록 상기 SDID를 상기 데이터 스토어의 행의 각각에 대해 저장하기 위한 수단; 및

상기 사용자로의 디스플레이를 통해 제한된 상기 추상화를 렌더링하기 위한 수단

을 포함하는 시스템.

청구항 16

제15항에 있어서,

행 레벨 보안 정책을 적용하는 것과 관련하여 이용되는 신뢰할 수 있는 아이덴티티 설정 시스템을 설정하기 위한 수단을 더 포함하는 시스템.

청구항 17

삭제

청구항 18

삭제

청구항 19

삭제

청구항 20

삭제

명 세 서

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

[0014] <관련 출원들에의 상호 참조>

[0015] 이 출원은 2005년 2월 28일에 출원된 "DISCOVERABILITY AND ENUMERATION MECHANISMS IN A HIERARCHICALLY SECURE SYSTEM"이라는 제명의 미국 가특허 출원 번호 60/657,536에 대한 우선권 이익을 주장한다. 상기 출원의 전체 내용은 본 명세서에 참고로 통합된다.

[0016] 저장 시스템들은 전통적으로 컨테인먼트 계층 구조(containment hierarchy)를 이용하여 저장 단위들(units of storage)을 조직한다. 이들 시스템에 따르면, 컨테이너 및 그에 따라, 본질적으로 컨테이너 내에 유지된 데이터 단위들은, 주체들(principals)에 대한 액세스의 제공을 용이하게 하기 위해 독립적으로 확보 가능(securable)하다. 종래의 시스템들은 주체들에 대해 액세스 가능하지 않은 컨테이너를 만날 때 데이터에의 액세스를 제한할 수 있는 횡단(traversal)을 통하여 발견 가능성(discoverability)을 제공한다.

[0017] 이들 시스템은 적어도 다음의 제한들로 고생한다. 하나의 제한은 주체가 그들이 액세스할 수 있는 데이터의 전체 세트(global set of data)를 시각화(visualize)할 수 없다는 것이다. 바꾸어 말하면, 데이터의 전체 세트를 렌더링(rendering)할 때, 그에 의해 사용자가 액세스할 수 없는 컨테이너와 만나면, 이 컨테이너의 콘텐츠(예컨대, 데이터 단위들)는 렌더링될 수 없을 것이다. 주체에 대해 액세스 제한들(access restrictions)이 가해진 컨테이너 내에 하위 폴더(sub-foler) 및 하위 컨테이너(sub-container)가 존재하는 상황을 생각해보자. 이런 시나리오에서, 주체는 적절한 권한들(adequate permissions)이 적소에 있다 하더라도 상기 하위 폴더의 콘텐츠를 시각화(예컨대, 발견) 또는 액세스할 수 없을 것이다. 이 제한적인 발견 가능성은 부모 폴더를 액세스하는 적절한 권한들의 부재 때문이다.

[0018] 종래의 시스템들의 또 하나의 제한은, 주체가 동시에 모든 데이터에 대해 작용할 수 없다는 것이다. 예를 들면, "주어진 노드를 루트로 하는 트리형 구조 내의 모든 데이터에 대한 FABRIKAM\alice에의 액세스 허가"와 같은 동작에 대한 제한(restriction)은 그 트리형 구조 내의 데이터의 일부에의 액세스를 제한할 제한들이 적소에 있을 수 있으므로 가능하지 않을 것이다. 일부 전통적인 시스템들에서, 그러한 동작은 시스템 컨텍스트에서보다는 사용자 컨텍스트에서 실행된다.

[0019] 일부 종래의 시스템들의 또 다른 제한은, 데이터를 액세스하려면 해당 저장 단위에 대한 액세스 권한 외에 해당 데이터 단위의 직계 부모(immediate parent)에의 접속점으로부터 모든 컨테이너들에 대해 적소에 적절한 권한들을 필요로 한다는 것이다. 바꾸어 말하면, 일부 시스템들에서는, 데이터의 직접 파일 경로가 알려져 있다 할지라도, 해당 데이터가 저장되어 있는 직계 부모에의 접속점으로부터 액세스 권한들이 존재하지 않는다면 해당 데이터를 액세스하는 권한이 제한될 수 있다.

[0020] 또 다른 제한은, 기존의 파일 시스템 모델 상에서의 유효한 열거(enumeration)를 위하여, 전통적인 저장 시스템들은 데이터와 메타데이터를 구별한다. 풍부한 최종 사용자 타입들에 대해서, 이러한 분리는 메타데이터와 데이터 간의 구별을 인지하는 데 어려움을 야기한다.

발명이 이루고자 하는 기술적 과제

[0021] 다음은 본 발명의 몇몇 양태들에 대한 기본적인 이해를 제공하기 위하여 본 발명의 간략한 개요를 소개한다. 이 개요는 본 발명의 광범위한 개관은 아니다. 이 개요는 본 발명의 기본적인/중요한 구성 요소들을 확인하기 위해 의도된 것이 아니다. 이 개요의 유일한 목적은 나중에 소개되는 보다 상세한 설명에 대한 서론으로서 본 발명의 일부 개념들을 간략한 형태로 소개하기 위한 것이다.

[0022] 여기에 개시되고 청구된 발명은, 그 하나의 양태에서, 접속점으로부터 스토어의 사용자별 추상화(a per user abstraction of a store from a connection point)를 생성하는 시스템을 포함한다. 이 추상화는 적용 가능한 권한들에 따라서 계층 구조적으로 안전한 저장 시스템에 유지된 데이터의 발견 가능성을 용이하게 할 수 있다.

주체의 액세스 권한들에 기초하여 계층 구조적으로 보안된 컨테인먼트 구조로부터 뷰 세트를 필터링하는 것이 본 발명의 새로운 특징들 중 하나이다. 본 발명은 잠재적으로 이종의(heterogeneous) 보안 정책들(예컨대, 보안 설명자들)로 다수의 컨테이너 계층 구조들에 걸치는(span) 이 집합체(aggregation)에 대해 작용할 수 있는 프리미티브들의 컬렉션(a collection of primitives)을 제공할 수 있다. 이 모델은 도메인 내의 모든 판독 액세스 가능한 아이템들을 발견하기 위해 컨테이너 계층 구조를 횡단(traverse)할 필요를 줄일 수 있다.

[0023] 또 다른 양태에서는, 사용자가 자동적으로 수행되기를 원하는 액션을 예지하거나 추론하기 위해 확률 및/또는 통계 기반 분석을 이용하는 인공 지능(AI) 컴포넌트가 제공된다.

[0024] 전술한 그리고 관련 목적들의 성취를 위하여, 본 발명의 특정한 예시적 양태들이 이하의 설명 및 첨부 도면들과 관련하여 여기에 설명된다. 이들 양태들은, 그러나, 본 발명의 원리들이 이용될 수 있는 여러 가지 방법들 중 소수만을 나타내고 본 발명은 모든 그러한 양태들 및 그들의 균등물들을 포함하도록 의도된다. 본 발명의 다른 이점 및 새로운 특징들은 이하의 발명의 상세한 설명을 도면들과 관련하여 고찰했을 때 분명해질 것이다.

[0025] 이제 도면들을 참조하여 본 발명을 설명한다. 도면들에서는 전반에 걸쳐서 유사한 구성 요소들을 가리키기 위해 유사한 참조 부호들이 이용된다. 이하의 설명에서는, 설명을 목적으로, 본 발명에 대한 충분한 이해를 제공하기 위하여 다수의 특정 상세들이 제시된다. 그러나, 본 발명이 이들 특정 상세들 없이도 실시될 수 있음은 명백할 것이다. 그 밖에, 본 발명의 설명을 용이하게 하기 위하여 잘 알려진 구조들 및 디바이스들은 블록도 형태로 도시되어 있다.

[0026] 이 출원에서 사용될 때, "컴포넌트" 및 "시스템"이라는 용어들은 컴퓨터 관련 엔티티(entity)로, 하드웨어나, 하드웨어와 소프트웨어의 조합이나, 소프트웨어나, 실행 중인 소프트웨어를 가리키도록 의도되어 있다. 예를 들면, 컴포넌트는 프로세서 상에서 실행 중인 프로세스, 프로세서, 객체, 실행 파일(executable), 실행 스레드(a thread of execution), 프로그램, 및/또는 컴퓨터일 수 있지만, 이에 제한되지는 않는다. 예시로서, 서버 상에서 실행 중인 애플리케이션과 서버 모두가 컴포넌트일 수 있다. 프로세스 및/또는 실행 스레드 내에 하나 이상의 컴포넌트들이 있을 수 있고, 컴포넌트는 하나의 컴퓨터 상에 로컬화되거나 및/또는 2 이상의 컴퓨터들 사이에 분산될 수 있다.

[0027] 여기에서 사용될 때, "추론하다(inter)" 또는 "추론(inference)"이라는 용어는 일반적으로 이벤트들 및/또는 데이터를 통하여 획득된 관찰들의 세트로부터 시스템, 환경, 및/또는 사용자의 상태들에 관하여 추론하거나 추측하는 프로세스를 가리킨다. 추론은 특정 컨텍스트 또는 액션을 식별하기 위해 이용될 수 있고, 또는 예를 들어 상태들에 걸친 확률 분포를 발생시킬 수 있다. 추론은 확률적일 수 있다. 즉, 데이터 및 이벤트에 대한 고찰에 기초하여 관심 있는 상태들에 걸친 확률 분포의 계산일 수 있다. 추론은 또한 이벤트들 및/또는 데이터의 세트로부터 보다 높은 레벨의 이벤트들을 구성하기 위해 이용되는 기법들을 가리킬 수 있다. 그러한 추론의 결과, 관찰된 이벤트들 및/또는 저장된 데이터의 세트로부터, 그 이벤트들이 시간적으로 매우 근접하게 상호 관련되어 있든 그렇지 않든 간에, 그리고 그 이벤트들 및 데이터가 하나의 이벤트 및 데이터 소스로부터 온 것이든 수 개의 이벤트 및 데이터 소스들로부터 온 것이든 간에, 새로운 이벤트들 또는 액션들이 구성된다.

[0028] 이 발명의 양태들은 컴퓨터 시스템들에 관한 것이고 보다 구체적으로는 계층 구조적으로 안전한 저장 시스템(들)에 유지된 데이터의 발견 가능성에 관한 것이다. 앞에서 설명된 바와 같이, 전통적인 저장 시스템들은 보안 관련 발견 가능성 메커니즘들과 관련하여 제한들을 갖고 있다. 이 때문에, 새로 나오는 데이터베이스 지향 파일 시스템들은 풍부한 쿼링(querying)을 지원할 수 있고 공통의 데이터 단위들(예컨대, 컨택트들)에 대해 체계화된 최종 사용자 타입들(schematized end-user types)을 제공할 수 있다. 이들 체계화된 최종 사용자 타입들은 데이터와 관련하여 애플리케이션들의 상호 운용성을 용이하게 하고 향상시킬 수 있다.

[0029] 본 발명은 데이터의 계층 구조적 표현을 고려한다. 보다 구체적으로, 이 발명은 데이터가 서로 다른 폴더들로 "버킷화(bucketize)"되고 그 후 서로 다른 컨테이너들에 놓일 수 있는 것을 고려한다. 사용자들은 이들 컨테이너를 이용하여 그들의 데이터를 조직(organize)할 수 있다. 예를 들면, 데이터는 그림, 음악, 문서 등과 같은 카테고리들로 조직(예컨대, 버킷화)될 수 있다. 게다가, 이들 카테고리들은 컨테이너들로 더 조직화됨으로써 데이터의 계층 구조적 표현을 확립할 수 있다. 예로서, 그림들 내에는, "나의 가족", "나의 휴가", "나의 결혼" 등의 그림들이 있을 수 있다. 또한, 계층 구조에 따라서 하위 카테고리들이 존재할 수 있다.

[0030] 이 계층 구조적 표현에 따르면, 본 발명은 보안 정책(예컨대, 보안 설명자)을 각 객체와 관련시키는 것을 용이하게 할 수 있다. 객체는 컨테이너 자체는 물론 컨테이너 내에 포함된 임의의 데이터 엘리먼트일 수도 있다는 것을 알 것이다. 또한, 각 객체는 테이블의 개개의 행(row)에 표현될 수 있다. 이 행 기반 표현은 후술되는

도면들에 대한 논의에서 더 잘 이해될 것이다.

[0031] 일 양태에서, 보안 설명자는 데이터 액세스를 위한 이들 객체들의 제공을 가능케 할 수 있다. 예로서, 본 발명의 일 양태에 따라서, 보안 정책은 그룹 "나의 가족" 내의 임의의 누군가에 의한 액세스를 허용하도록 "나의 휴가" 폴더를 설정하는 것을 용이하게 할 수 있다. 또한, "나의 휴가" 내에서 사용자는 하위 폴더(예컨대, "시애틀로의 여행(my trip to Seattle)")에 액세스하려는 "나의 가족" 내의 특정 맴버들에 대한 액세스를 더 제한할 수 있다.

[0032] 종래의 시스템들에 따르면, 데이터 스토어의 액세스 가능한 탐사는 사용자가 그에 대해 열거 액세스(enumeration access)를 갖고 있지 않은 폴더에 도달할 때 임의의 지점에서 끝난다. F1이 F2를 포함하고 F2가 F3를 포함하는 계층 구조를 생각해보자. 사용자가 아무런 권한도 허여되지 않은 F2에 도달하는 순간, 사용자는 F3 내의 데이터를 볼 수 있는 능력을 갖지 않을 것이다. 사용자가 F3에 액세스할 수 있다 할지라도, F3는 권한들이 적소에 있지 않은 F2 내에 포함되어 있기 때문에 종래의 시스템들은 발견 가능성을 금지할 것이다 - 이것이 제한이다. 본 발명은 사용자가 탐사(예컨대, 발견)하고 및/또는 렌더링하는 균일한 액세스를 가질 수 있게 함으로써 권한들이 허여되고 적소에 있는 데이터 스토어 내의 모든 데이터의 이용을 허용한다. 앞에서 설명된 바와 같이, 이 균일한 액세스는 데이터 스토어 내의 각 객체와 관련된 보안 정책을 통하여 용이하게 될 수 있다. 이해되겠지만, 각 보안 정책은 행 레벨 아이템에 관련될 수 있다.

[0033] 전통적인 파일 시스템들은 파일들을 검색하기 위해 2개의 액세스 모드를 이용한다. 첫째로, 이들 시스템들은 제한된 발견 방법을 용이하게 하고 이 방법에 의해 사용자는 그에 대해 적절한 보안 권한들이 존재하는 데이터 엘리먼트들을 발견할 수 있다. 다른 것은 직접 액세스 메커니즘이고 이것에 의해 사용자는 전체 경로가 알려져 있고 액세스할 권한이 적소에 있다면 파일을 액세스할 수 있다.

[0034] 상기 2개의 전혀 다른 모드들 외에, 본 발명은 보안 자격 증명(security credentials)에 기초하여 액세스 및 발견을 허용하는 쿼리 모드(예컨대, 데이터 스토어 필터링)인 제3 모드를 이용할 수 있다. 전통적인 시스템들과 달리, 본 발명은 정의된 특정 속성에 기초하여 모든 데이터를 쿼리할 뿐만 아니라 해당 데이터에 대해 작용하는 메커니즘을 제공할 수 있다. 이 발명에 의하면, 액세스 자격 증명이 적소에 있는 한, 데이터를 발견할 수 있고 그 데이터에 대해 원하는 대로 작용할 수 있다.

[0035] 그에 따라서, 본 발명은 트리형 구조(예컨대, 계층 구조적 데이터 조직)의 루트에서 설정되어 트리형 구조를 통하여 그 구조 내의 모든 자식들에게 전파될 수 있는 보안 정책(예컨대, 보안 설명자)을 가능케 할 수 있다. 전파된 보안 설명자는 부모 보안 정책, 자식 보안 정책, 및/또는 객체의 타입에 기초할 수 있다는 것을 이해해야 할 것이다. 보안 정책을 발생시키고 트리형 구조를 통하여 전파하는 것을 실행하는 로직이 이용될 수 있다. 뒤에서 설명되겠지만, 보안 정책을 전파하기 위해 규칙 기반 로직 및 인공 지능이 이용될 수 있다.

[0036] 사용자가 새로운 아이템을 생성하는 시나리오를 생각해보자. 이 시나리오에서는, 자식에게 상속되거나 결합되는 부모의 특정 보안 정책들(예컨대, 설명자들)이 있을 것이다. 일 양태에서, 사용자는 권한들을 갖는 폴더(예컨대, 컨테이너)를 가질 수 있고 객체가 생성될 때, 그 객체에 대한 권한들은 동일한 것으로 추정될 수 있다. 대안적으로, 새로 생성된 객체에 전파된 권한들은 그 객체에 대한 권한들은 물론 폴더에 대한 권한들 양쪽 모두에 기초하여 지능적으로 결정될 수 있다. 전술한 것은 새로운 혁신의 양태들에 따른 상속의 예들이다.

[0037] 전통적인 파일 시스템들에서는, 이런 전파가 불가능하다는 것을 알 것이다. 오히려, 종래의 시스템들에 따라서 권한들을 변경하기 위해서는, 관리자가 트리형 구조의 각 자식을 거쳐가면서 권한들을 적절하게 변경해야 한다. 그와 반대로, 이 발명의 양태들에 따르면, 루트 권한(root permission)이 변경(또는 확립)될 때, 그 권한은 자식을 포함한 트리형 구조의 모두에게 자동으로 전파될 수 있다.

[0038] 일부 전통적인 시스템들에서는, 업데이트 시에 "사용자의 컨텍스트"에서만 보안 권한들이 전파될 수 있다는 것을 주목하는 것이 중요하다. 권한들이 나중에 변경될 수 있는 상황들이 있기는 하지만, 종래의 시스템들은 이들 권한들을 자동으로 업데이트할 수 없다.

[0039] 본 발명은 "시스템의 컨텍스트"에서 권한들을 전파할 수 있다. 그러므로, 사용자가 개재하는 폴더(intervening folder)에 대한 권한들을 갖고 있지 않더라도, 만일 하위(sub), 하위-하위(sub-sub) 등의 트리형 구조에 대해 권한들이 적소에 있다면, 이들 권한들은 본 발명에 따라서 전파될 수 있다. 이런 양태는 전술한 F1, F2 및 F3 예를 고려함으로써 더 잘 이해될 것이다.

[0040] 그 예에서 계속해서, F2에 대해 권한들이 적소에 있지 않더라도, 만일 F3에 대해 권한들이 존재한다면, F1으로부터 F3로 권한들이 전파될 수 있다. 속성들(예컨대, 파일명, 사이즈, 생성된 날짜)과 데이터(예컨대, 파일의

콘텐츠)를 구별하는 기존의 파일 시스템들과 달리, 풍부한 데이터 시스템들에서는 속성과 데이터를 구별하는 것이 곤란하다. 그러므로, 데이터 엘리먼트가 속성이든 데이터이든 상관없이 "아이템들"은 생성되었고 "아이템" 별 토대로(on a per "item" basis) 액세스 권한들을 허여하기 위해 사용된다. 따라서, 본 발명과 관련하여, 시스템이 2개의 별개의 보안 권한들을 계속 알고 있을 필요가 없기 때문에 보안 모델의 관리는 특히 간단해질 수 있다. 오히려, 일 양태에서는, 아이템마다 2개의 "판독" 권한 및 2개의 "기입" 권한을 이용하기보다는 아이템마다 단 하나의 "판독" 또는 단 하나의 "기입" 권한만이 이용된다.

[0041] 그 결과, 본 발명은 사용자가 권한들이 적소에 있는 모든 데이터의 추상화를 보는 것을 이용하게 할 수 있다. 이 뷰들은 전체 스토어에 걸쳐서 정의되고 그 후 사용자에게 렌더링될 수 있다. 뷰는 접속점으로부터 볼 수 있는 아이템들과 허용된 보안 권한들의 세트의 교집합(intersection)으로 정의될 수 있다. 그 결과, 사용자는 사용자가 보거나 및/또는 액세스할 보안 권한들을 갖고 있는 접속점 아래의 아이템들을 보거나 및/또는 액세스할 수 있다.

발명의 구성 및 작용

[0042] 우선 도 1을 참조하면, 파일 스토어의 콘텐츠의 표현을 렌더링하는 것을 용이하게 하는 시스템(100)이 도시되어 있다. 일반적으로, 시스템(100)은 쿼리 컴포넌트(102) 및 행 레벨 보안 컴포넌트(104)를 포함한다. 동작시에, 쿼리 컴포넌트(120)는, 행 레벨 보안 컴포넌트(104)와 함께, 보안 정책 또는 권한을 만족시키는 데이터 컴포넌트(106) 내의 아이템들을 식별할 수 있다. 일단 식별되면, 결과의 데이터 세트는 사용자 및/또는 애플리케이션에게 렌더링될 수 있다. 예를 들면, 앞에서 설명된 바와 같이, 본 발명은 그 결과 세트를 디스플레이를 통하여 사용자에게 렌더링할 수 있다.

[0043] 이제 도 2를 참조하면, 행 레벨 보안 컴포넌트(104)에 대한 보다 상세한 블록도가 도시되어 있다. 특히, 행 레벨 보안 컴포넌트(104)는 보안 설명자 테이블(202) 및 단일 인스턴스 테이블(204)을 포함할 수 있다. 각각의 이들 테이블에 대해서는 뒤에서 더 상세히 설명될 것이다.

[0044] 보안 컴포넌트(104)는 행 레벨 보안의 실현을 제공할 수 있다. 사용자가 셰어(share)(예컨대, 데이터 컴포넌트(106))에 접속할 때, 각각의 데이터 타입들에 대한 암시적 뷰 정의들(implicit view definitions)이 접속 범위 내에서 정의될 수 있다. 본 발명에 컨텍스트를 추가하기 위하여, 아래에는 "컨택트(Contact)" 타입에 대한 예시적 뷰 정의가 있다.

[0045]
```
CREATE VIEW [System.Storage.Contacts.Store].[Contact] AS
```

[0046]
```
 SELECT ItemId, TypeId, NamespaceName, ContainerId,
```

[0047]
```
ItemSyncMetadata,
```

[0048]
```
  TREAT(Item AS [System.Storage.Contacts.Store].[Contact]) AS
```

[0049]
```
Item, PathHandle,
```

[0050]
```
  EntityState, ObjectSize, ChangeInformation, PromotionStatus
```

[0051]
```
 FROM [System.Storage.Store].[Table!Item]
```

[0052]
```
 WHERE Item IS OF ([System.Storage.Contacts.Store].[Contact])
```

[0053]
```
  AND (@@ITEM_DOMAIN_IS_ROOT = 1
```

[0054]
```
  OR (PathHandle >= @@ITEM_DOMAIN AND PathHandle <
```

[0055]
```
@@ITEM_DOMAIN_LIMIT))
```

[0056] 각 아이템은 엔티티 테이블들(202, 204) 내의 행으로서 저장된다. 상기 예시적 표현은 스토어 내의 아이템들의 전체적 범위(global scope)로부터 컨택트 타입들을 필터링하여 제거하는 것을 실행할 수 있다. 사용자가 대응하는 행 내의 보안 설명자들에 따라서 판독 가능한 아이템들만을 보려고 할 경우 액세스 제어의 디멘전은 이 필터링에 대해 암시적이다.

[0057] 이 예에서, 뷰 정의는 컨택트들인 아이템들로 뷰를 제한하는 상기 "WHERE" 절(clause)을 포함할 수 있다. 예의 나머지는 접속점으로부터 아이템들에 대한 액세스를 제한할 수 있다. 상기 뷰 정의는 보안 정의를 포함하지 않는다는 것을 이해해야 할 것이다.

[0058] 위에서 설명된 바와 같이, 보안 메커니즘은 테이블들(202, 204) 내에 저장된 행 레벨 보안의 기능이다. 이 메커니즘은 뷰의 하위 테이블 레벨(underlying table level)에서 적용되고 뷰에 대한 전파 효과를 갖는다. 보안이 행별 토대로(on a per row basis) 인에이블될 때, 사용자가 그에 대해 판독 액세스를 갖고 있지 않은 행들은 쿼리 컴포넌트(102)에 의해 제공되는 결과 세트에 나타나지 않는다.

[0059] 파일 시스템 모델에서, 각 "아이템"은 행에 있고, 각 행은 그와 관련된 보안을 갖는다. 행 레벨 보안 메커니즘(104)은 사용자가 판독 액세스를 갖고 있지 않은 행들에 대해 그 행들이 결과에 나타나는 것을 제한한다. 뷰는, 쿼리 컴포넌트(102)에 전달된 정의가 주어질 경우, (상기 예에서와 같이) 접속점에 적어도 일부 기초하여 렌더링(예컨대, 뷰잉(viewing))을 제한할 수 있다. 그러므로, 결과 세트는 이들 2개의 제한의 교집합일 수 있다. 이들 보안 메커니즘들은 쿼리 정의에 대해 암시적으로 나타날 수 있다는 것을 알 것이다. 그 결과, 사용자는 그 동작들 중 어느 것으로부터도 보호될 수 있다.

[0060] 본 발명은 테이블(예컨대, 204) 내의 각 행의 보안 설명자를 체크하는 단일 인스턴싱 메커니즘(single instancing mechanism)을 이용한다. 이 단일 인스턴싱 메커니즘은 시스템이 각 행에 걸쳐서 체크를 수행하는 것으로 보이게 할 수 있다. 행들에 걸친 보안 설명자들의 단일 인스턴싱은 이 메커니즘의 체크를 효과적으로 만들 수 있다. 보안 정책들(예컨대, 액세스 제어 리스트들)이 예시적 보안 설명자들 대신에 이용될 수 있다는 것을 알 것이다. 그러므로, 이들 부가적인 새로운 특징들은 이 발명 및 여기에 부속된 청구항들의 범위 내에 속하도록 의도되어 있음을 이해해야 할 것이다. 게다가, 위에서는 ACL들이 언급되었지만, 전혀 다른 보안 정책들(disparate security policies)을 이용하는 다른 양태들이 존재한다는 것을 이해해야 할 것이다. 이들 전혀 다른 정책들은 이 명세서 및 이에 부속된 청구항들의 범위 내에 속하도록 의도되어 있다.

[0061] 동작 시에는, 2개의 테이블들(202, 204)이 유지된다 - 보안 설명자들의 테이블(202) 및 보안 설명자의 해시(예컨대, SHA-1) 및 보안 설명자 ID(SDID) 간을 매핑하는 단일 인스턴스 테이블. 이 SDID는 고유한 값임을 알 것이다. 본 발명에 따르면, 단일 인스턴싱은, 스토어 내의 각각의 고유한 보안 설명자마다, 시스템이 SDID와 보안 설명자의 해시 간의 맵을 유지하는 메커니즘을 가리킨다.

[0062] 그러므로, 각 행마다, 보안 설명자를 저장하는 대신에, 그것에 대응하는 SDID가 저장된다. 일 양태에서는, 사용자가 아이템을 생성할 때, 사용자는 보안 설명자를 제공하거나 또는 그것을 빈 상태로 놓아두는 선택권을 갖는다. 만일 빈 상태로 되면, 보안 설명자는 생성되는 아이템으로부터의 부모로부터 상속될 수 있다. 사용자가 보안 설명자를 명시적으로 제공하기로 선택할 경우, 시스템은 명시적으로 정의된 설명자를 부모의 보안 설명자와 병합하여 설명자를 생성할 수 있다.

[0063] 일단 새로운 아이템에 대한 보안 설명자가 무엇일지에 대한 결정이 행해지면, 그것이 이미 존재하는지에 대한 결정이 행해질 것이다. 만일 그것이 존재하면, 기존의 것이 사용될 것이다. 만일 그것이 존재하지 않으면, 새로운 것이 저장될 것이다.

[0064] 보안 설명자가 존재하는지를 결정하기 위하여, 본 발명은 보안 설명자 대 보안 설명자의 해시(예컨대, SHA-1 해시)의 매핑을 포함하는 단일 인스턴스 테이블(204)을 참조한다. 그러므로, 동일한 보안 설명자를 갖는 또 다른 아이템이 존재하는지를 결정하기 위하여, 해당 보안 설명자의 해시가 계산된다. 그 후 시스템은 임의의 행들이 보안 설명자의 동일한 해시(예컨대, SHA-1)를 포함하는지를 알기 위해 행에 대해 단일 인스턴싱 테이블(204)을 쿼리한다. 만일 매치하는 것이 발견되면, 그것이 존재할 확률이 높다.

[0065] 다음으로, 보안 설명자가 존재하는지를 검증하기 위하여 실제 보안 설명자와의 비교가 행해진다. 만일 실제 보안 설명자가 동일하지 않다면, 시스템은 보안 설명자를 독립적으로 저장한다. 시스템은 비유일성(non-uniqueness)을 보장하기 위하여 해시 알고리즘(예컨대, SHA-1)에만 의지한다는 것을 알아야 할 것이다. 바꾸어 말하면, 만일 해싱된 값이 단일 인스턴스 테이블(204) 내의 해싱된 값과 매치하지 않으면, 보안 설명자가 존재하지 않는다는 결정이 행해질 수 있다.

[0066] 보안 설명자에 대한 3개의 속성이 있다 - 해시(보안 설명자의 바이너리에 기초하여 수학적으로 계산된 값), 보안 설명자 자체(바이너리), 및 SDID(보안 설명자를 가리키는 정수 값). 각 행마다, 시스템은 그에 대해 보안 설명자가 관련된 그 특정 행의 ID를 저장한다. 다음으로, 단일 인스턴스 테이블(204)에서, 시스템은 해시(예컨대, SHA-1)와 SDID 간을 매핑한다. 보안 설명자 테이블(202)에서, 시스템은 SDID와 바이너리 간을 매핑한다.

[0067] 그러므로, 단일 인스턴스 테이블(204) 및 보안 설명자 테이블(202)은 함께 SHA-1 해시로부터 SDID로 바이너리로의 완전한 매핑을 제공한다. 효과적으로, 이들 2개의 테이블(202, 204)은 단일 인스턴싱 체크를 수행하기 위해 이용될 수 있다.

[0068] 보안 설명자는 다음의 논리 형태를 가질 수 있다:

[0069] O : owner_sid

[0070] G : group_sid

[0071] D : dacl_flags(ace1)(ace2) ... (acen)

[0072] S : sacl_flags(ace1)(ace2) ... (acen)

[0073] 상기 예에서, O는 소유자를 식별하고, G는 그룹을 식별하고, D는 임의의 액세스 제어 리스트(DACL : Discretionary Access Control List)(명세서의 범위 내의 보안 설명자의 부분)를 식별하고, S는 시스템 액세스 제어 리스트(SACL)를 식별한다. DACL은 액세스 제어 엔트리들(ACE)의 컬렉션이고, 각 엔트리는 다음 형태를 취할 수 있다.

[0074] ace_type; ace_flags; rights; account_sid

[0075] 주어진 주체에게는 특정 아이템들에 대한 액세스가 허여되거나 거절될 수 있다. 따라서, 거절된 아이템들은 사용자 뷰들로부터 암시적으로 필터링 제거될 수 있다. 필터링 엔진 또는 쿼리 컴포넌트(102)는 임의의 컨테이너 의미(container semantics)에 대해 불가지의(agnostic) 스토어 내의 모든 아이템들을 스캔하여 균일한 세트를 생성할 수 있고 그에 의해 전통적인 파일 시스템들에서의 횡단의 제한들을 교묘하게 회피할 수 있다.

[0076] 이 2개의 내부 테이블들(202, 204)은 시스템에서의 저장 및 액세스 제어를 용이하게 하기 위해 이용될 수 있다. 예시적 양태에서, 시스템은 [System.Storage.Store].[Table!SecurityDescriptorSingleInstance] 테이블(204)(예컨대, 인스턴스 테이블) 및 Sys.security_descriptors 테이블(202)(예컨대, 보안 설명자 테이블)을 이용할 수 있다. Sys.security_descriptors 테이블(202)은 보안 설명자들의 카탈로그 뷰(catalog view)이다. 이들 설명자들은 SQL 서버에 의해 제공된 데이터 정의 언어(DLL) 프리미티브들을 이용하여 생성되거나 삭제될 수 있다. 단일 인스턴스 테이블(204)은 시스템에서의 중앙 처리 장치(CPU) 및 메모리 최적화의 키일 수 있다.

[0077] 일 양태에 따르면, 상당 수의 아이템들이 동일한 보안 정책 또는 설명자를 공유하는 것이 통상적일 수 있다. 일례에서, 액세스 제어 리스트(ACL)의 최대 사이즈는 64KB이고 따라서 주어진 보안 설명자는 128KB 정도일 수 있다. 각 아이템에 잠재적으로 높은 정도의 공통성(commonality)이 주어진 경우 이 사이즈의 값을 저장하는 것은 비효율적일 수 있다는 것을 알 것이다. 그러므로, 각각의 고유한 보안 설명자는 Sys.security_descriptors 테이블(202)에 저장될 수 있고 설명자와 그것의 SHA-1 해시 간의 매핑은 단일 인스턴스 테이블(204)에 유지될 수 있다. 앞에서 설명된 바와 같이, SHA-1은 출력들의 유일성(uniqueness)을 보장하지 않지만, 그것의 큰 출력 범위(예컨대, 2^160)를 가정하면 충돌은 극히 일어날 것 같지 않다. 인스턴스 테이블(204)은 자기 치유 성질(self-healing nature)을 가질 수 있으므로, 시스템이 손상(corruption) 또는 불일치(inconsistencies)로부터 자동 복구할 수 있음을 보장할 수 있다.

[0078] Item/Extension/Fragment/Link(아이템/확장/프로그램먼트/링크) 테이블들은 SECURITY 속성으로 마크될 수 있는 SDID에 대한 엔트리를 갖는다. 이것은 이들 테이블에 대한 모든 판독 액세스 및 이들 뷰들의 상부에 구축된 임의의 뷰들이 (FILE_READ_DATA | FILE_READ_ATTRIBUTES)을 요구하는 액세스 체크를 받도록 보증할 수 있다. ItemExtension, Link 및 ItemFragment 테이블들 내의 행들은 아이템 테이블 내의 대응하는 행과 동일한 보안 설명자를 갖는다.

[0079] 앞에서 설명된 메커니즘은 최근 생겨난 파일 시스템들에 대한 판독 경로에서의 인가 모델(authorization model)의 핵심에 있는 것으로 생각될 수 있다. 어느 인가 모델이든 본질적으로 인증 모델(authentication model)에 의지할 수 있다. 일례에서, 사용자가 스토어에 접속할 때, 사용자는 선호되는 오퍼레이팅 시스템 인증 메커니

즘(예컨대, NTLM(NT LAN 매니저), 커베로스(Kerberos))을 이용하여 인증(예컨대, 믿을 수 있는 것으로 간주)될 수 있다. 인증의 최종 결과는 파일 시스템에 액세스하고 있는 사용자를 나타내는 보안 토큰일 수 있다. 이 토큰은 그 후 주체에 대한 인가 판정을 하기 위해 이용될 수 있다.

[0080] 본 발명의 다른 양태에 따르면, 행 또는 레코드 레벨 보안(RLS)을 이용하여 안전하게 된 아이템들은 저장 서비스 계정(storage service account)으로부터도 보호될 수 있다. 보안 평가를 위하여, 서비스 계정은 임의의 다른 NT-브랜드 계정처럼 간주될 수 있다. 이것은 특히 균일한 보안 의미(uniform security semantics)를 보장할 수 있지만, 업데이트 경로에서의 교차 문제(intersecting problems)를 드러낼 수 있다. 예를 들면, 주어진 네임스페이스 이름을 갖는 아이템을 생성하려고 하는 사용자를 생각해보자. 최근 생겨난 파일 시스템들에서의 네임스페이스 이름들은 그들의 포함하는 폴더에서 고유한 것으로 보장되어, 모호하지 않은 명명 시스템(naming system)을 제공한다. 생성 동작들 중에, 시스템은 동일한 폴더 내에 동일한 네임스페이스 이름을 가진 다른 아이템들이 존재하지 않음을 보증함으로써 이 유일성을 보장한다.

[0081] 이런 시나리오에서, 서비스 계정에 대해 거절된 액세스 권한들을 가진 폴더 내에 이미 아이템이 존재할 수 있다. 이 발명은 서명 메커니즘을 이용함으로써 이 문제를 다룰 수 있다. 스토어에 대한 전체적 액세스(global access)를 필요로 하는 업데이트 프리미티브들은 "면제된 RLS(exempt RLS)" 권한이 허여된 인증서들로 서명될 수 있다. 그런 프리미티브의 컨텍스트 내로부터, 시스템은 스토어를 쿼리할 수 있고 이 경우 행 레벨 보안은 무시(bypass)될 것이다.

[0082] 앞에서 설명된 바와 같이, 전통적인 파일 시스템들은 횡단 의미(traversal semantics)를 가능케 하기 위하여 속성과 데이터 간의 구별을 행하였다. 발견 가능성 및 쿼리 기반 의미(query-based semantics)의 결핍은 액세스 제어 판정을 위하여 속성과 데이터가 구별되는 모델을 야기하였다. 본 발명은 타입 시스템에 대한 모든 또는 전무한 의미(all or nothing semantics)를 용이하게 함으로써 데이터 및 속성들에의 무결절성 액세스(seamless access)를 제공한다.

[0083] 다음은 예시적 파일 시스템 보안 모델에 대한 상세한 논의이다. 이하의 논의는 전혀 다른 시나리오들에서의 컴포넌트 기능을 설명한다. 이들 설명된 시나리오들은 단지 본 발명에 대한 컨텍스트를 제공하기 위해 제공된 것일 뿐 어떤 식으로든 본 발명이나 부속된 청구항들을 제한하기 위해 의도된 것이 아님을 알아야 할 것이다.

[0084] 우선 파일 시스템 보안 모델을 참조하면, 일 양태에서, 데이터는 파일 시스템에서의 최소의 무모순 단위(smallest unit of consistency)를 가리킬 수 있는 "아이템"으로서 스토어 내에 조직될 수 있다. "아이템"은 독립적으로 확보(secure)될 수 있고, 시리얼라이즈(serialize)될 수 있고, 동기화될 수 있고, 복사될 수 있고, 백업/복원될 수 있고 등등일 수 있다. 파일 시스템 아이템은 그 조상이 엔티티 타입인 타입 System.Storage.Item인 타입의 인스턴스로서 기술될 수 있음을 알 것이다. 파일 시스템 내의 모든 아이템들은 아이템들의 단일 전체적 범위(single global extent of items)에 저장될 수 있다. 또한, 각 아이템은 주어진 파일 시스템 스토어 내의 모든 아이템들에 대해 고유하도록 보장되는 고유한 식별자를 가질 수 있다.

[0085] 이제 도 3을 참조하면, 시스템(300)이 도시되어 있다. 시스템(300)은 이 보안 논의의 컨텍스트에 따른 것인 데 반하여 타입 시스템(302) 내의 아이템들은 일반적 컨테이너 타입들(generic container types)(304) 및 복합 아이템 타입들(compound item types)(306)의 인스턴스들로서 분류될 수 있다. 일반적 컨테이너들(304)은 폴더들 및 임의의 다른 계층 구조적 데이터 컬렉션 버킷들을 모델링하기 위해 이용될 수 있다. 복합 아이템 타입들(306)은 애플리케이션에 대한 단일 논리적 데이터 단위(single logical unit of data)를 모델링하기 위해 이용될 수 있다. 이 타입의 인스턴스들은 복사, 이동, 동기화 등과 같은 전형적인 데이터 연산들에 대한 모든 또는 전무한 의미(all or nothing semantics)를 제공할 수 있다. 후자의 예를 들면, 메일 메시지, 그림, 콘택트 등을 포함하지만, 이들에 제한되는 것은 아니다. 복합 아이템 타입들(306)의 인스턴스들(점선으로 표시됨)은 파일 지원 아이템들(file backed items)(308)(FBI들) 및 비파일 지원 아이템들(non-file backed items)(310)(nFBI들)로서 더 분류될 수 있다. Win32-브랜드 액세스는 의미론적으로 FBI들 및 일반적 컨테이너들에 제한된다는 것을 알 것이다.

[0086] 다음의 컨테인먼트 계층 구조(예컨대, 트리형 구조)는 아이템들에 적용된다. 일반적 컨테이너들(304) 및 복합 아이템들(306)은 일반적 컨테이너들을 포함한 임의의 다른 아이템 타입들을 포함할 수 있다. 이들 부가적인 일반적 컨테이너들 내의 아이템들도 독립적으로 확보될 수 있다. FBI들(308)은 다른 아이템들을 포함할 수 없고 따라서 계층 구조에서의 리프 노드들(leaf nodes)을 형성한다.

[0087] 이제 도 4를 참조하면, 파일 시스템(400)은 신뢰 경계(trust boundary)(402)의 양쪽에 2개의 주요 컴포넌트들 -

스토어 컴포넌트(404) 및 클라이언트 컴포넌트(406)를 포함할 수 있다는 것을 알 것이다. 예시된 바와 같이, 스토어 컴포넌트(404)는 1 내지 N의 객체 컴포넌트들을 포함할 수 있고, 여기서 N은 정수이다. 객체 컴포넌트들(1 내지 N)은 개별적으로 또는 집합적으로 객체 컴포넌트들(408)로 불릴 수 있다. 객체(408)의 저장 및 검색을 다루는 스토어 컴포넌트(404)는 스토어 컴포넌트(404)와 클라이언트 컴포넌트(406) 간의 신뢰 파일 시스템(trusted file system)을 형성할 수 있다.

[0088] 프로그래밍 의미(programming semantics)를 플랫폼에 제공하는 클라이언트 컴퓨터(406)는 통상적으로 사용자 프로세스들에서 실행한다. 사용자들은 접속 시에 인증될 수 있다는 것을 이해할 것이다. 검색된 객체들(406)(예컨대, 아이템들)은 클라이언트 공간에서 실체화(materialize)될 수 있다. 일 양태에서, 이들 객체들(408)에 대해 클라이언트에 의해 아무런 보안 체크나 액세스 제약도 실시되지 않는다. 본 발명에 따르면, 스토어 컴포넌트(404)는 프로그래밍 컨텍스트가 스토어 컴포넌트(404)에 지속될 때 (액세스 제어 컴포넌트(410)를 통하여) 액세스 제어를 실시할 수 있다. 다음은 사용자 인증에 대한 논의이다.

[0089] 파일 시스템(400)은 파일 시스템 스토어(404)에 포함된 아이템들(408)에 대하여 액션을 수행할 수 있는 보안 주체(security principal)의 개념을 제시할 수 있다. 본 발며의 양태들에서, 보안 주체는 사용자 또는 보안 그룹일 수 있다. 따라서, 보안 주체는 보안 식별자(SID)에 의해 표현될 수 있다.

[0090] 도 4에 예시된 바와 같이, 파일 시스템 서비스에의 접속은 액세스 제어 컴포넌트(410)에 의해 성공적으로 인증된 보안 주체의 컨텍스트에 있다. (예컨대, 액세스 제어 컴포넌트(410)를 통한) 파일 시스템 인증은 오퍼레이팅 시스템 인증 메커니즘의 파생물일 수 있음을 이해할 것이다. 예를 들면, 파일 시스템 인증은 SQL(구조화 쿼리 언어) 보안 모델에서 이용 가능한 윈도즈-브랜드 인증의 파생물일 수 있다. 예를 들면, SQL은 파일 시스템(400)에서 지원될 수 없는 SQL 인증이라 불리는 또 다른 기본 제공된(built-in) 인증 메커니즘을 제공한다는 것을 알 것이다.

[0091] 이 예에서 계속해서, 윈도즈-브랜드 사용자에 의해 시도된 접속은 커베로스, NTLM 등과 같은 인증 서비스들에 의해 제공된 윈도즈-브랜드를 이용(leveraging)하면서 파일 시스템(400)에 의해 인증될 수 있다. 이 예에서, 인증된 사용자는 스토어(404)에서의 인가 판정에 이용되는 SQL 내의 "공공" 역할("public" role)로 매핑된다. 일 양태에서, 빌트인 관리자(BA)는 BA에게 SQL 관리 권한을 허여하는 SQL 관리자들에 매핑될 것이다. 대안적 양태에서, 파일 시스템 관리는 오로지 파일 시스템 프리미티브들을 이용하여 구축될 수 있다. 그러므로, BA는 대안적 양태에서 SQL 관리자들의 일원이 아닐 것이다.

[0092] 인증의 최종 결과는 파일 시스템(400)을 액세스하는 주체를 나타내는 보안 토큰이다. 이 데이터 구조는 들어오는 주체의 SID는 물론 그 주체가 일원이 되는 모든 그룹들의 SID들을 포함할 수 있다. 게다가, 사용자가 보유한 모든 권한들은, 파일 시스템(300)에 접속하면서, 디폴트로, 인에이블될 수 있다. 아래의 논의에 이어서 더 잘 이해되겠지만, 이 토큰은 그 후 인가 판정을 행하기 위해 이용될 수 있다.

[0093] 이제 인가에 대한 논의로 넘어가서, 앞에서 설명된 바와 같이, 파일 시스템 인가는 셰어 레벨 보안 및 아이템 레벨 보안 상에서 구축될 수 있다. 이 설명에서 사용될 때, "셰어(share)"라는 용어는 스토어(410) 내의 아이템(408)의 별명을 가리킬 수 있다. 스토어(410)가 생성될 때, 루트 아이템의 별명으로 디폴트 셰어가 생성된다. 충분한 권한들을 가진 사용자들은 스토어(410) 내의 임의의 일반적 컨테이너(예컨대, 아이템(408))의 별명으로 셰어들을 생성할 수 있다.

[0094] 파일 시스템은 보편적 명명 규칙(universal naming convention) 경로들을 이용하여 로컬로(locally) 및 원격으로 네임스페이스를 공개할 수 있다. 그러므로 파일 시스템 클라이언트들은 셰어에 접속하고 그에 의해 그 접속점은 이름들의 상대적 계층 구조와 함께 파일 시스템 객체들(408)의 어드레싱 메커니즘을 구성한다.

[0095] 예로서, 사용자가 foo를 액세스하기 위해 루트 셰어에 접속하는 것을 가정해보자. 따라서 액세스는 \\MachineName\StoreName\RootShare\...\foo로서 나타날 것이다. 유사하게, AliceShare라 불리는 셰어에 접속한 사용자는 \\MachineName\AliceShare\...\foo로서 동일한 객체를 액세스할 것이다. 이 예에서, 아이템에 대한 유효한 권한은 접속된 셰어 및 아이템에 대한 보안 설명자의 기능일 수 있다. 전자는 셰어 레벨 보안을 정의하고 후자는 아이템 레벨 보안을 정의한다. 이들 보안 메커니즘들 각각은 물론 유효한 보안 설명자들을 구성하기 위한 규칙들에 대한 상세는 뒤에서 설명된다.

[0096] 셰어 레벨 보안의 논의에서 시작하여, 본 발명에 따른 파일 시스템 셰어들은 얼마간 윈도즈-브랜드 셰어들과 유사하다. 로컬 및 원격 액세스에 걸쳐서 균일한 의미(uniform semantics)를 제공하기 위하여, 생성된 모든 파일 시스템 셰어마다, 미러링 셰어(mirroring share)도 생성될 수 있다. 셰어들은 카탈로그 스토어 내에 아이템들

로서 저장될 수 있고 다음에 나오는 주제인 아이템 보안을 이용하여 확보될 수 있다. 이들 아이템들 및 셰어들에 대한 권한들은 로컬 및 원격 액세스 양쪽 모두에 대한 균일한 액세스 의미를 허여하는 것과 동일할 수 있다.

[0097] 디폴트 권한들은 아이템들과 관련하여 원하는 대로 허여될 수 있다. 예를 들면, 셰어 내의 전혀 다른 아이템들은 사용자 특징들(예컨대, 로컬 시스템 빌트인 관리자, 인증된, 대화식(interactive)...)와 관련하여 적용된 서로 다른 디폴트 권한들을 가질 수 있다.

[0098] 윈도즈-기반 셰어들과 유사하게, 셰어 보안 설명자에 대한 디폴트 값들은 LanManServer\DefaultSecurity\SrvsvcDefaultShareInfo에서의 레지스트리 설정을 이용하여 구성 가능하다.

[0099] 아이템 보안 메커니즘들은 보안 설명자들을 이용하여 액세스 제어를 실행할 수 있다. 따라서, 일 양태에서, 보안 설명자는 보안 설명자 정의 언어 문자열 포맷으로 API들(애플리케이션 프로그램 인터페이스들)에 의해 통신되어 보안 설명자 테이블(도 2의 202)인 Sys.Security_Descriptors의 VARBINARY 열(column) 아래 패킹된 바이너리 포맷(packed binary format)으로 데이터베이스에 저장될 수 있다.

[0100] 앞에서 설명된 새로운 보안 설명자 테이블인 Sys.Security_Descriptors는, 파일 시스템 기반 테이블들에서 외래 키(foreign key)로서 사용하기 위해 고유 ID(SDID)와 함께 패킹된 바이너리 설명자 테이블로서 저장된, 각각의 고유한 보안 설명자를 보유하기 위해 존재한다. 예를 들면, 보안 설명자 테이블은 다음과 같이 나타날 수 있다:

[0101]

SDID	보안 설명자 VARBINARY
55	XXXXXXXXXX
56	XXXXXXXXXX

[0102] 비록 상기 보안 설명자 테이블이 보안 설명자용으로 바이너리 표현을 이용하고는 있지만, 본 발명 및 이에 부속된 청구항들의 의미 및 범위를 일탈하지 않고서 임의의 적당한 표현이 이용될 수 있다는 것을 알아야 할 것이다.

[0103] 이제 보안 설명자들 및 관련 데이터이 표현 및 저장에 대한 논의로 돌아가서, 앞에서 설명된 바와 같이, 본 발명은 보안 설명자 관련 정보를 보유할 수 있는 2개의 내부 테이블들을 이용한다 - 보안 설명자 테이블(예컨대, sys.security_descriptors) 및 단일 인스턴스 테이블(예컨대, [System.Storage.Store].[Table!SecurityDescriptorSingleInstance]).

[0104] 이 예에서 계속해서, Sys.security_descriptors는 SQL에 의해 유지된 카탈로그 뷰이다. 이 바이너리는 SDID와 함께 대응하는 행에 저장된다.

[0105] 단일 인스턴스 테이블은 파일 시스템에 의해 유지될 수 있다. 그것은 바이너리 보안 설명자의 해시 대 전술한 Sys.security_descriptors 뷰 또는 테이블의 맵(map)을 포함한다. 일례에서는, SHA-1 해시가 이용될 수 있다. 일 양태에서, 만일 동일한 보안 설명자들을 가진 다수의 아이템들이 생성된다면, 양쪽 테이블에 단일 엔트리가 존재할 수 있다.

[0106] 위에서 언급된 바와 같이, 본 발명의 또 다른 새로운 특징은, 만일 단일 인스턴스 테이블이 손상되는 일이 있다면, 그것은 자기 치유 테이블(self-healing table)이므로 파괴될 수 있다. 바꾸어 말하면, 손상이 일어난다면, 단지 새로운 해시 값들을 생성하고 그것들을 적절한 SDID에 관련시키는 것으로 새로운 테이블이 생성될 수 있다.

[0107] 일 양태에서, Item/Extension/Fragment/Link 테이블들은 "보안" 속성으로 마크된 SDID에 대한 엔트리를 가질 수 있다. 이것은 이들 테이블에 대한 어떠한 판독 액세스든 그리고 이들 뷰들의 상부에 구축된 어떠한 뷰들이든 (FILE_READ_DATA | FILE_READ_ATTRIBUTES)을 요구하는 액세스 체크를 받도록 보증할 수 있다는 것을 이해할 것이다. ItemExtension, Link 및 ItemFragment 테이블은 Item 테이블과 동일한 보안 설명자를 가져야 한다는 것을 또한 이해할 것이다.

[0108] 도 5는 본 발명의 일 양태에 따라서 초기화의 방법을 예시한다. 설명을 간단히 할 목적으로, 예컨대, 흐름도 형태의, 여기에 도시된 하나 이상의 방법들은, 일련의 동작들(a series of acts)로 도시되고 설명되어 있기는

하지만, 본 발명은 동작들의 순서에 의해 제한되지 않는다는 것을 이해하고 알아야 할 것이다. 왜냐하면 일부 동작들은, 본 발명에 따라서, 여기에 도시되고 설명된 것과는 다른 순서로 및/또는 다른 동작들과 동시에 일어날 수 있기 때문이다. 예를 들면, 숙련된 당업자라면 대안적으로 방법이 상태도에서와 같은 일련의 상호 관련된 상태들 또는 이벤트들로서 표현될 수도 있다는 것을 이해하고 알 것이다. 더욱이, 본 발명에 따라서 방법을 구현하기 위해 예시된 모든 동작들이 필요하지 않을 수도 있다.

[0109] 작성 프로세스(build process) 중에 모델 데이터베이스를 작성하는 동안에 보안 데이터 구조들이 초기화된다. 액트(502)에서, 테이블들이 셋업된다. 일례에서, 테이블들을 셋업하는 것은 Sys.server_principals, Sys.database_principals, Sys.server_role_members 및 Sys.database_role_members를 셋업하는 것을 포함할 수 있다. 액트(504)에서, 단일 인스턴스 테이블이 생성된다. 일례에 따르면,

[0110] [System.Storage.Store].[Table!SecurityDescriptorSingleInstance]가 액트(504)에서 생성될 수 있다.

[0111] 액트(506)에서 루트 보안 설명자가 생성된다. 이 루트 보안 설명자는 스토어의 루트에 대응한다(예컨대, 관리자들이 완전한 제어를 가질 수 있다). 액트(508)에서, 아이템 레벨 보안 설명자들이 생성된다. 예를 들면, 액트(508)에서, 관리자들이 완전한 제어를 갖고 인증된 사용자들이 판독 액세스를 갖도록 톰스톤 아이템(tombstone items)에 대한 보안 설명자들이 생성될 수 있다. 액트(510)에서, 이들 엔트리들이 단일 인스턴스 테이블에 추가된다.

[0112] 파일 시스템은 ACL들의 상속을 지원할 수 있다. 예를 들면, 아이템 생성의 때로부터(예컨대, CreateItem 또는 CreateComplexItems), 공급된 보안 설명자(만일 있다면), 부모 보안 설명자, 아이템의 타입 및 호출자의 토큰(예컨대, NT-브랜드 토큰)을 이용하여 아이템에 대한 보안 설명자가 계산될 수 있다.

[0113] 이제 액세스 체크에 대한 논의를 참조하여, 모든 업데이트 API들은 [System.Storage.Store].[HasSecurityAccess]를 호출함으로써 적절한 액세스 체크를 수행한다. API는 호출자에게 보안 설명자(예컨대, 아이템, 레코드) 레벨뿐만 아니라 셰어 레벨 양쪽 모두에서 요구 권한(request permission)이 허여되도록 보증한다. 하나의 특정 양태에서, (부모의) 보안 설명자에 대해 수행된 액세스 체크는 셰어에 대해 수행된 것(DELETE)과 다르다(FILE_DELETE_CHILD). 다른 경우들에서, 그 2개의 액세스 체크는 일관될 수 있다.

[0114] 이 예에서 계속하여, 트리형 구조 전반에 걸친 ACL 전파는 (새로운 DACL 또는 SACL을 갖는) SetItemSecurity 또는 새로운 부모를 갖는 MoveItem이 호출될 때 수행될 수 있다. 호출자가 동작을 수행하는 것이 허용되도록 보증하기 위해 적절한 액세스 체크들이 수행된 후에, ACL 전파는 파일 시스템의 컨텍스트에서 실행될 수 있다. ACL들이 업데이트되는 서브트리형 구조(subtree-like structure)에 대해서는 아무런 액세스 체크도 행해지지 않는다.

[0115] 본 발명은 비동기식 및/또는 동기식 전파를 이용할 수 있다는 것을 알아야 할 것이다. 다음은 동기식 전파에 대한 논의이다. 서브트리형 구조의 루트는 복합 아이템들(Compound items)과는 아무런 관계가 없다는 것을 이해해야 할 것이다. 오히려, 서브트리형 구조의 루트는 SetItemSecurity 또는 MoveItem이 호출되는 노드를 기술하기 위한 일반적 용어이다.

[0116] 동기식 전파에 따르면, 루트 아이템에 대한 새로운 보안 설명자가 계산된다. 만일 DACL 또는 SACL이 업데이트되지 않으면, 아이템, 확장, 프래그먼트 및 링크 테이블들에 대해 SDID가 업데이트되고 시스템은 리턴한다. 전체 아이템 서브트리형 구조는 아이템에서 시작하여 잠겨진다(lock). 이 예에서는, 임의의 다른 테이블(확장, 프래그먼트, 링크)을 잠글 필요가 없다.

[0117] 다음으로, 상기 동작에서의 모든 아이템들을 포함하는 임시 테이블(temporary table)이 생성될 수 있다. 이 임시 테이블은 다음의 특징들을 가질 수 있다. 임시 테이블은 ContainerID, ItemID, 및 NewSdId를 가질 수 있다. 또한, 처음에, 서브트리형 구조의 루트를 제외한 모두에 대해 NewSdId는 널(NULL)일 수 있다.

[0118] 임시 테이블 내의 각 엔트리마다, 새로운 부모 SD, 아이템의 타입 및 기존 아이템 SD를 이용하여 새로운 SD가 계산될 수 있다. 이 예에서, CreatePrivateObjectSecurityEx(SEF_AVOID_PRIVILEGE_CHECK | SEF_AVOID_OWNER_CHECK)가 생성될 수 있다. 따라서, 임시 테이블은 새로운 부모 SD가 계산 완료되었고 아이템에 대한 새로운 SDID가 널인 행들을 처리할 때마다 한 레벨씩(level by level) 횡단(traverse)될 수 있다. 이 예에 따르면, 이것은 한 번에 한 레벨씩 테이블을 순회한다.

[0119] 반복 횟수는 0이다(예컨대, 트리형 구조의 깊이). 2개의 이슈가 고려될 수 있다. 첫째로, 새로운 보안 설명자

들의 계산이 고려될 수 있다. 둘째로, 모든 자식에 대한 보안 설명자들의 업데이트가 고려될 수 있다. 두 번째 시나리오에서, 이론적 한계는 0이다(예컨대, 자식의 수). 첫 번째 시나리오에서는, 그럴 필요는 없지만, 통상적으로 0이다(트리의 깊이). 만일 필요하다면, 새로운 보안 설명자자 생성될 수 있다(예컨대, 단일 인스턴스 및 Sys.security_descriptors 테이블들에서). 다음으로, 임시 테이블에서 임시 SDID 테이블이 업데이트된다. 마지막으로, 임시 테이블에서 계산된 데이터를 이용하여 아이템, 확장, 링크 및 프래그먼트가 업데이트될 수 있다.

[0120] 도 6은 선택문들(SELECT statements)에 대한 액세스 제어가 행 레벨 보안(Row Level Security)에 의해 실시되는 사용자 컨텍스트에서 마스터 테이블 뷰들(Master Table Views)을 쿼리하는 T/SQL 연산들이 동작하는 것을 예시한다. 또한, 파일 시스템 스토어 업데이트 API(File system Store Update API)에 대한 호출들이 사용자 컨텍스트에서 행해지지만 시스템 컨텍스트에서 실행된다. 그러므로 구현은 호출자에 대한 권한 체크들을 실시할 수 있다.

[0121] 도 7은 본 발명에 따라서 하나 이상의 특징들을 자동화하는 것을 용이하게 하는 인공 지능(AI)을 이용하는 시스템(700)을 예시한다. 본 발명(예컨대, 보안 정책들을 구현하는 것과 관련하여)은 각종 양태들을 수행하기 위해 각종 AI 기반 스킴들(AI-based schemes)을 이용할 수 있다. 예를 들면, 보안 설명자가 설정되어야 하는지를 결정하는 프로세스 및, 만일 그렇다면, 이용할 보안의 레벨은 자동 분류자(classifier) 시스템 및 프로세스를 통하여 용이하게 될 수 있다. 더욱이, 단일 인스턴스 및 보안 설명자 테이블들(도 2로부터의 202, 204)이 다수의 위치들에 멀리 떨어져서 위치하는 경우, 비교를 위해 어느 위치가 선택될 것인지를 결정하기 위해 분류자가 이용될 수 있다.

[0122] 분류자는 입력 속성 벡터, x=(x1,x2,x3,x4,xn)를 그 입력이 클래스에 속한다는 신뢰(confidence), 즉, f(x)=confidence(class)로 매핑하는 함수이다. 그러한 분류는 사용자가 자동적으로 수행되기를 원하는 액션을 예지하거나 추론하기 위해 확률 및/또는 통계 기반 분석을 이용할 수 있다.

[0123] 지원 벡터 머신(SVM : support vector machine)이 이용될 수 있는 분류자의 일례이다. SVM은 가능한 입력들의 공간 내에서, 비트리거링 이벤트들(non-triggering events)로부터 트리거링 기준(triggering criteria)을 분리시키려고 하는 하이퍼서피스(hypersurface)를 찾음으로써 동작한다. 직관적으로, 이것은 가까이 있지만 훈련 데이터와 동일하지 않은 테스팅 데이터에 대해 분류를 정확하게 한다. 다른 다이렉트(directed) 및 언다이렉트(undirected) 모델 분류 방법들은 예컨대, 나이브 베이스(naive Bayes), 베이스 네트워크(Bayesian networks), 판정 트리(decision trees), 신경망(neural networks), 퍼지 논리 모델(fuzzy logic models)을 포함하고, 독립의 상이한 패턴들을 제공하는 확률적 분류 모델들이 이용될 수 있다. 여기에서 사용되는 분류는 또한 우선 순위의 모델들을 개발하기 위해 이용되는 통계적 리그레션(statistical regression)을 포함한다.

[0124] 본 명세서로부터 쉽게 알 수 있겠지만, 본 발명은 명시적으로 훈련되는(예컨대, 일반적 훈련 데이터를 통하여) 것뿐만 아니라 암시적으로 훈련되는(예컨대, 사용자 행위를 관찰하거나, 외부로부터의 정보를 수신하는 것을 통하여) 분류자들을 이용할 수 있다. 예를 들면, SVM들은 분류자 생성자(classifier constructor 및 특징 선택 모듈 내에서 학습 및 훈련 단계(learning and training phase)를 통하여 구성된다. 따라서, 분류자(들)는 다수의 기능들을 자동으로 학습하고 수행하기 위해 이용된다. 그 기능들은 소정의 기준에 따라서 판정하는 것을 포함하지만 이에 제한되지는 않는다.

[0125] 이제 도 8을 참조하면, 개시된 아키텍처를 실행하도록 동작 가능한 컴퓨터의 블록도가 예시되어 있다. 본 발명의 각종 양태들에 대한 부가적인 컨텍스트를 제공하기 위하여, 도 8 및 이하의 논의는 본 발명의 각종 양태들이 구현될 수 있는 적당한 컴퓨팅 환경(800)에 대한 간략하고 일반적인 설명을 제공하도록 의도되어 있다. 위에서는 하나 이상의 컴퓨터들 상에서 실행될 수 있는 컴퓨터 판독 가능한 명령들의 일반적인 컨텍스트에서 본 발명이 설명되었지만, 숙련된 당업자라면 본 발명이 프로그램 모듈들과 조합하여 및/또는 하드웨어와 소프트웨어의 조합으로서 구현될 수도 있다는 것을 인지할 것이다.

[0126] 일반적으로, 프로그램 모듈들은 특정 작업을 수행하거나 또는 특정 추상(abstract) 데이터 타입들을 구현하는 루틴, 프로그램, 컴포넌트, 데이터 구조 등을 포함한다. 또한, 숙련된 당업자라면, 본 발명의 방법들이, 싱글프로세서 또는 멀티프로세서 컴퓨터 시스템, 미니컴퓨터, 메인프레임 컴퓨터는 물론, 퍼스널 컴퓨터, 핸드헬드 컴퓨팅 디바이스, 마이크로프로세서 기반 또는 프로그램 가능한 소비자 전자 기기 등을 포함하는 다른 컴퓨터 시스템 구성들을 이용하여 실시될 수도 있다는 것을 알 것이다. 위에 열거된 각각의 디바이스는 하나 이상의 관련 디바이스들에 동작 가능하게 결합될 수 있다.

[0127] 본 발명의 예시된 양태들은 또한 통신 네트워크를 통하여 연결되어 있는 원격 처리 디바이스들에 의해 특정 작업들이 수행되는 분산 컴퓨팅 환경들에서 실시될 수도 있다. 분산 컴퓨팅 환경에서는, 프로그램 모듈들은 로컬 메모리 기억 장치 및 원격 메모리 기억 장치 양쪽 모두에 위치할 수 있다.

[0128] 컴퓨터는 전형적으로 각종의 컴퓨터 판독 가능 매체를 포함한다. 컴퓨터 판독 가능 매체는 컴퓨터에 의해 액세스될 수 있는 임의의 이용 가능한 매체일 수 있고 휘발성 및 비휘발성 매체, 이동식(removable) 및 고정식(non-removable) 매체 모두를 포함한다. 한정이 아니라 예로서, 컴퓨터 판독 가능 매체는 컴퓨터 기억 매체 및 통신 매체를 포함할 수 있다. 컴퓨터 기억 매체는 컴퓨터 판독 가능 명령들, 데이터 구조들, 프로그램 모듈들 또는 그 밖의 데이터와 같은 정보의 저장을 위한 임의의 방법 또는 기술로 구현된 휘발성 및 비휘발성, 이동식 및 고정식 매체 모두를 포함한다. 컴퓨터 기억 매체는, RAM, ROM, EPROM, EEPROM, 플래시 메모리 또는 다른 메모리 기술, CD-ROM, 디지털 비디오 디스크(DVD) 또는 다른 광 디스크 기억 장치, 자기 카세트, 자기 테이프, 자기 디스크 기억 장치 또는 다른 자기 기억 장치들, 또는 원하는 정보를 저장하기 위해 이용될 수 있고 컴퓨터에 의해 액세스될 수 있는 임의의 다른 매체들을 포함하지만, 이들에 제한되지는 않는다.

[0129] 통신 매체는 전형적으로 컴퓨터 판독 가능 명령들, 데이터 구조들, 프로그램 모듈들 또는 다른 데이터를 반송파(carrier wave) 또는 다른 전송 메커니즘(transport mechanism)과 같은 변조된 데이터 신호로 구현하고, 임의의 정보 전달 매체를 포함한다. "변조된 데이터 신호"(modulated data signal)라는 용어는 그 신호 내에 정보를 인코딩하도록 설정되거나 변경된 하나 이상의 그것의 특징들을 갖는 신호를 의미한다. 한정이 아니라 예로서, 통신 매체는 유선 네트워크(wired network) 또는 직접 유선 접속(direct-wired connection)과 같은 유선 매체, 및 음향, RF, 적외선 및 다른 무선 매체와 같은 무선 매체를 포함한다. 상기한 것들 중 임의의 것들의 조합들도 컴퓨터 판독 가능 매체의 범위 내에 포함된다.

[0130] 다시 도 8을 참조하면, 본 발명의 각종 양태들을 구현하기 위한 예시적 환경(800)은 컴퓨터(802)를 포함하고, 컴퓨터(802)는 처리 장치(804), 시스템 메모리(806) 및 시스템 버스(808)를 포함한다. 시스템 버스(808)는 시스템 메모리(806)를 포함하지만 그것에 제한되지 않는 시스템 컴포넌트들을 처리 장치(804)에 연결시킨다. 처리 장치(804)는 각종의 상업적으로 입수 가능한 프로세서들 중 임의의 것일 수 있다. 듀얼 마이크로프로세서 및 다른 멀티프로세서 아키텍처들도 처리 장치(804)로서 이용될 수 있다.

[0131] 시스템 버스(808)는 몇몇 타입의 버스 구조 중 어느 하나일 수 있고, 이것은 또한 각종의 상업적으로 이용 가능한 버스 아키텍처들 중 어느 하나를 이용하여 (메모리 컨트롤러를 갖거나 갖지 않는) 메모리 버스, 주변 버스, 및 로컬 버스에 상호 접속할 수 있다. 시스템 메모리(806)는 판독 전용 메모리(ROM)(810) 및 랜덤 액세스 메모리(RAM)(812)를 포함한다. 기본 입출력 시스템(BIOS)이 ROM, EPROM, EEPROM과 같은 비휘발성 메모리(810)에 저장되고, 이 BIOS는 시동(start-up) 중과 같이, 컴퓨터(802) 내의 엘리먼트들 간에 정보를 전송하는 것을 돕는 기본 루틴들을 포함한다.

[0132] 컴퓨터(802)는 내부 하드 디스크 드라이브(HDD)(814)(예컨대, EIDE, SATA) - 이 내부 하드 디스크 드라이브(814)는 또한 외부 사용을 위하여 적당한 섀시(도시되지 않음) 내에 구성될 수도 있음 -, (예컨대, 이동식 디스켓(818)로부터 판독하거나 거기에 기록하기 위한) 자기 플로피 디스크 드라이브(FDD)(816), (예컨대, CD-ROM 디스크(822)를 판독하거나, DVD와 같은 다른 고용량 광학 매체로부터 판독하거나 거기에 기록하기 위한) 광 디스크 드라이브(820)를 더 포함한다. 하드 디스크 드라이브(814), 자기 디스크 드라이브(816) 및 광 디스크 드라이브(820)는 각각 하드 디스크 드라이브 인터페이스(824), 자기 디스크 드라이브 인터페이스(826) 및 광 드라이브 인터페이스(828)에 의해 시스템 버스(808)에 접속될 수 있다. 외부 드라이브 구현을 위한 인터페이스(824)는 USB(Universal Serial Bus) 및 IEEE 1394 기술들 중 적어도 하나 또는 양쪽 모두를 포함한다. 다른 외부 드라이브 접속 기술들도 본 발명의 사상 범위 내에 있다.

[0133] 드라이브들 및 그들의 관련 컴퓨터 판독 가능한 매체는 데이터, 데이터 구조, 컴퓨터 판독 가능한 명령 등의 비휘발성 저장을 제공한다. 컴퓨터(802)를 위하여, 드라이브들 및 매체는 적당한 디지털 포맷으로 임의의 데이터를 저장하는 편의를 제공한다. 상기 컴퓨터 판독 가능한 매체에 대한 설명은 HDD, 이동식 자기 디스켓, 및 CD 또는 DVD와 같은 이동식 광학 매체를 언급하고 있지만, 집 드라이브(zip drives), 자기 카세트, 플래시 메모리 카드, 카트리지 등과 같이, 컴퓨터에 의해 판독 가능한 다른 타입의 매체들도 예시적 운영 환경에서 사용될 수 있고, 또한 임의의 그러한 매체들은 본 발명의 방법들을 수행하기 위한 컴퓨터 판독 가능한 명령들을 포함할 수 있다는 것을 숙련된 당업자라면 알 것이다.

[0134] 드라이브들 및 RAM(812)에는, 운영 시스템(830), 하나 이상의 애플리케이션 프로그램(832), 기타 프로그램 모듈들(834), 및 프로그램 데이터(836)를 포함하는 다수의 프로그램 모듈들이 저장될 수 있다. 그 운영 시스템, 애

플리케이션들, 모듈들, 및/또는 데이터의 전부 또는 일부는 또한 RAM(812)에 캐싱될 수 있다. 본 발명은 각종의 상업적으로 입수 가능한 운영 시스템들 또는 운영 시스템들의 조합들로 구현될 수 있다는 것을 알아야 할 것이다.

[0135] 사용자는 하나 이상의 유무선 입력 장치들, 예컨대, 키보드(838) 및 마우스(840)와 같은 포인팅 디바이스를 통하여 컴퓨터(802) 내에 커맨드 및 정보를 입력할 수 있다. 다른 입력 장치들(도시되지 않음)은 마이크, IR 원격 제어기, 조이스틱, 게임 패드, 스타일러스 펜, 터치 스크린, 또는 그런 것들을 포함할 수 있다. 이들 및 다른 입력 장치들은 흔히 시스템 버스(808)에 연결되어 있는 입력 장치 인터페이스(842)를 통하여 처리 장치(804)에 접속되지만, 예컨대, 병렬 포트, IEEE 1394 시리얼 포트, 게임 포트, USB 포트, IR 인터페이스 등과 같은 다른 인터페이스들에 의해 접속될 수도 있다.

[0136] 모니터(844) 또는 다른 타입의 디스플레이 장치가 또한 비디오 어댑터(846)와 같은 인터페이스를 통하여 시스템 버스(808)에 접속된다. 모니터(844) 외에, 컴퓨터는 전형적으로 스피커, 프린터 등과 같은 다른 주변 출력 장치들(도시되지 않음)을 포함한다.

[0137] 컴퓨터(802)는 원격 컴퓨터(들)(848)와 같은 하나 이상의 원격 컴퓨터들에의 유선 및/또는 무선 통신을 통한 논리적 접속들을 이용하여 네트워킹된 환경에서 동작할 수 있다. 원격 컴퓨터(들)(848)는 워크 스테이션, 서버 컴퓨터, 라우터, 퍼스널 컴퓨터, 휴대형 컴퓨터, 마이크로프로세서 기반 오락 기기, 피어 디바이스(peer device) 또는 다른 통상의 네트워크 노드일 수 있고, 전형적으로 컴퓨터(802)와 관련하여 설명된 엘리먼트들의 다수 또는 전부를 포함하지만, 간결성을 위하여, 메모리/기억 장치(850)만이 예시되어 있다. 도시된 논리적 접속들은 근거리 네트워크(LAN)(852) 및/또는, 예컨대, 광역 네트워크(WAN)(854)와 같은 보다 큰 네트워크들에의 유무선 접속을 포함한다. 그러한 LAN 및 WAN 네트워킹 환경들은 사무실과 회사에서 흔한 것으로, 인트라넷과 같은 전사적 컴퓨터 네트워크(enterprise-wise computer networks)를 용이하게 하고, 이들 모두는 예컨대 인터넷과 같은 전세계적 통신 네트워크에 접속할 수 있다.

[0138] LAN 네트워킹 환경에서 사용될 경우, 컴퓨터(802)는 유선 및/또는 무선 통신 네트워크 인터페이스 또는 어댑터(856)를 통하여 로컬 네트워크(852)에 접속된다. 어댑터(856)는 LAN(852)에의 유선 또는 무선 통신을 용이하게 할 수 있고, LAN(852)은 또한 무선 어댑터(856)와 통신하기 위해 그 위에 배치된 무선 액세스 포인트(wireless access point)를 포함할 수 있다.

[0139] WAN 네트워킹 환경에서 사용될 경우, 컴퓨터(802)는 모뎀(858)을 포함할 수 있고, 또는 WAN(854) 상의 통신 서버에 접속되거나, 또는 예컨대 인터넷을 경유하여 WAN(854) 상에서 통신을 확립하기 위한 다른 수단들을 갖는다. 내장형 또는 외장형이고 유선 또는 무선 디바이스일 수 있는 모뎀(858)은 시리얼 포트 인터페이스(842)를 통하여 시스템 버스(808)에 접속된다. 네트워킹된 환경에서, 컴퓨터(802)와 관련하여 도시된 프로그램 모듈들 또는 그 일부는 원격 메모리/기억 장치(850)에 저장될 수 있다. 도시된 네트워크 접속들은 예시적인 것이고 컴퓨터들 간에 통신 링크를 확립하는 다른 수단들이 이용될 수도 있다는 것을 알아야 할 것이다.

[0140] 컴퓨터(802)는 예컨대, 프린터, 스캐너, 데스크탑 및/또는 휴대형 컴퓨터, 휴대 정보 단말기(PDA), 통신 위성, 무선으로 검출 가능한 태그와 관련된 임의의 장비 또는 위치(예컨대, 키오스크, 신문 판매점(news stand), 화장실), 및 전화기와 같이, 무선 통신으로 동작 가능하게 배치된 임의의 무선 디바이스들 또는 엔티티들과 통신하도록 동작 가능하다. 이것은 적어도 Wi-Fi 및 Bluetooth™ 무선 기술을 포함한다. 따라서, 통신은 종래의 네트워크에서와 같이 미리 정의된 구조이거나 또는 단순히 적어도 2개의 디바이스들 간의 애드 혹 통신(ad hoc communication)일 수 있다.

[0141] Wi-Fi, 즉 Wireless Fidelity는 집안의 침상이나, 호텔방의 침대나, 직장의 회의실로부터 무선으로 인터넷에 접속 가능하게 한다. Wi-Fi는 예컨대 컴퓨터와 같은 그러한 디바이스들이 실내에서 또는 밖에서, 기지국의 유효 범위 내에서는 어디서든 데이터를 송신하고 수신할 수 있게 하는 셀 전화(cell phone)에서 사용되는 것과 무선 기술이다. Wi-Fi 네트워크는 안전하고, 신뢰할 수 있고, 고속의 무선 접속성을 제공하기 위해 IEEE 802.11(a, b, g 등)이라 불리는 무선 기술을 이용한다. Wi-Fi 네트워크는 컴퓨터들을 서로 접속하고, 인터넷에 접속하고, (IEEE 802.3 또는 이더넷을 이용하는) 유선 네트워크들에 접속하기 위해 사용될 수 있다. Wi-Fi 네트워크는, 예를 들어, 비인가된(unlicensed) 2.4 및 5 GHz 라디오 대역에서, 11 Mbps(802.11a) 또는 54 Mbps(802.11b) 데이터 레이트로 동작하거나, 또는 양 대역을 포함하는(듀얼 밴드) 제품들과 함께 동작하고, 따라서 그 네트워크는 많은 사무실들에서 사용되는 기본 10BaseT 유선 이더넷 네트워크와 유사한 실세계 성능을 제공할 수 있다.

[0142] 이제 도 9를 참조하면, 본 발명에 따른 예시적 컴퓨팅 환경(900)의 개략 블록도가 예시되어 있다. 이 시스템

(900)은 하나 또는 그 이상의 클라이언트(들)(902)를 포함한다. 클라이언트(들)(902)는 하드웨어 및/또는 소프트웨어(예컨대, 스레드, 프로세스, 컴퓨팅 디바이스)일 수 있다. 클라이언트(들)(902)는 예를 들어 본 발명을 이용함으로써 쿠키(들) 및/또는 관련 상황 정보(contextual information)를 수용(house)할 수 있다.

[0143] 이 시스템(900)은 또한 하나 또는 그 이상의 서버(들)(904)를 포함한다. 서버(들)(904)도 또한 하드웨어 및/또는 소프트웨어(예컨대, 스레드, 프로세스, 컴퓨팅 디바이스)일 수 있다. 서버들(904)은 예를 들어 본 발명을 이용함으로써 변환(transformations)을 수행하는 스레드를 수용할 수 있다. 클라이언트(902)와 서버(904) 간의 하나의 가능한 통신은 2개 또는 그 이상의 컴퓨터 프로세스들 간에 전송되게 되어 있는 데이터 패킷의 형태로 될 수 있다. 데이터 패킷은 예를 들어 쿠키 및/또는 관련 상황 정보를 포함할 수 있다. 시스템(900)은 클라이언트(들)(902)와 서버(들)(904) 간의 통신을 용이하게 하기 위해 이용될 수 있는 통신 프레임워크(902)(예컨대, 인터넷과 같은 전세계적 통신 네트워크)를 포함한다.

[0144] 통신들은 유선(광섬유를 포함) 및/또는 무선 기술을 통하여 용이하게 될 수 있다. 클라이언트(들)(902)는 클라이언트들(902)에 로컬인 정보(예컨대, 쿠키(들) 및/또는 관련 상황 정보)를 저장하기 위해 이용될 수 있는 하나 또는 그 이상의 클라이언트 데이터 스토어(들)(908)에 동작 가능하게 접속된다. 이와 유사하게, 서버(들)(904)는 서버들(904)에 로컬인 정보를 저장하기 위해 이용될 수 있는 하나 또는 그 이상의 클라이언트 데이터 스토어(들)(910)에 동작 가능하게 접속된다. 이상에서 설명된 것은 본 발명의 예들을 포함한다. 물론, 본 발명을 설명하기 위하여 컴포넌트들 또는 방법들의 상상할 수 있는 모든 조합을 기술하는 것은 가능하지 않지만, 숙련된 당업자라면 본 발명의 다수의 추가적인 조합들 및 치환들이 가능하다는 것을 인지할 것이다. 따라서, 본 발명은 첨부된 청구항들의 의미 및 범위 내에 드는 그러한 모든 변경, 변형 및 변화들을 두루 포함하도록 의도되어 있다. 또한, 이 상세한 설명 또는 청구항들에서 "포함한다"(includes)는 용어가 사용되는 한은, 그러한 용어는 "comprising"(포함하는)이 청구항에서 연결어(transitional word)로서 사용될 때 해석되는 것처럼 그 "comprising"이라는 용어와 유사하게 포괄적인 것으로 의도된다.

발명의 효과

[0145] 본 발명은 접속점으로부터 스토어의 사용자별 추상화를 생성하는 시스템을 포함한다. 이 추상화는 적용 가능한 권한들에 따라서 계층 구조적으로 안전한 저장 시스템에 유지된 데이터의 발견 가능성을 용이하게 할 수 있다. 주체의 액세스 권한들에 기초하여 계층 구조적으로 보안된 컨테인먼트 구조로부터 뷰 세트를 필터링하는 것이 본 발명의 새로운 특징들 중 하나이다. 본 발명은 잠재적으로 이종의 보안 정책들(예컨대, 보안 설명자들)로 다수의 컨테이너 계층 구조들에 걸치는 이 집합체에 대해 작용할 수 있는 프리미티브들의 컬렉션을 제공할 수 있다. 이 모델은 도메인 내의 모든 판독 액세스 가능한 아이템들을 발견하기 위해 컨테이너 계층 구조를 횡단할 필요를 줄일 수 있다.

도면의 간단한 설명

[0001] 도 1은 본 발명의 일 양태에 따라서 계층 구조적으로 안전한 저장 시스템 내의 데이터의 발견 가능성을 용이하게 하는 시스템의 일반적인 컴포넌트 블록도를 예시한다.

[0002] 도 2는 본 발명의 일 양태에 따라서 단일 인스턴스 테이블(single instance table) 및 보안 설명자 테이블(security descriptor table)을 포함하는 시스템의 블록도를 예시한다.

[0003] 도 3은 일 양태에 따라서 일반적 컨테이너 타입들(generic container types) 및 복합 아이템 타입들(compound item types)의 인스턴스들로서 타입 시스템 내의 아이템들을 분류하는 시스템을 예시한다.

[0004] 도 4는 본 발명의 일 양태에 따라서 신뢰 경계(trust boundary)의 양쪽에 스토어 컴포넌트 및 클라이언트 컴포넌트를 갖는 시스템의 블록도를 예시한다.

[0005] 도 5는 본 발명의 일 양태에 따라서 초기화의 방법을 예시한다.

[0006] 도 6은 본 발명의 일 양태에 따라서 뷰들을 쿼리하는 연산들이 선택문들(selection statements)에 대한 액세스 제어가 행 레벨 보안에 의해 실시될 수 있는 사용자 컨텍스트에서 동작할 수 있음을 예시하는 관계도이다.

[0007] 도 7은 본 발명의 일 양태에 따라서 인공 지능 기반 메커니즘들을 이용하는 시스템의 블록도이다.

[0008] 도 8은 개시된 아키텍처를 실행하도록 동작 가능한 컴퓨터의 블록도를 예시한다.

[0009] 도 9는 본 발명에 따른 예시적 컴퓨팅 환경의 개략 블록도를 예시한다.

[0010] <도면의 주요 부분에 대한 부호의 설명>

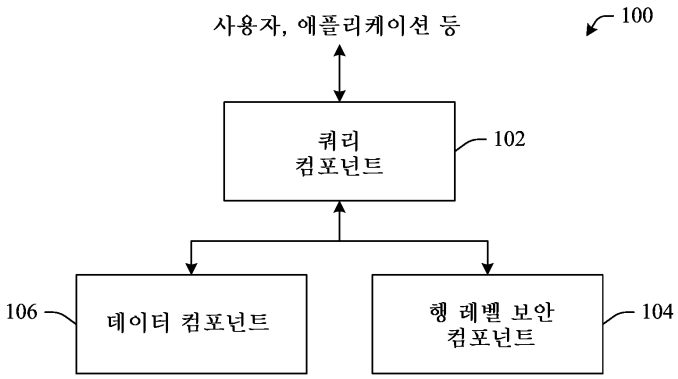
[0011] 102 : 쿼리 컴포넌트

[0012] 104 : 행 레벨 보안 컴포넌트

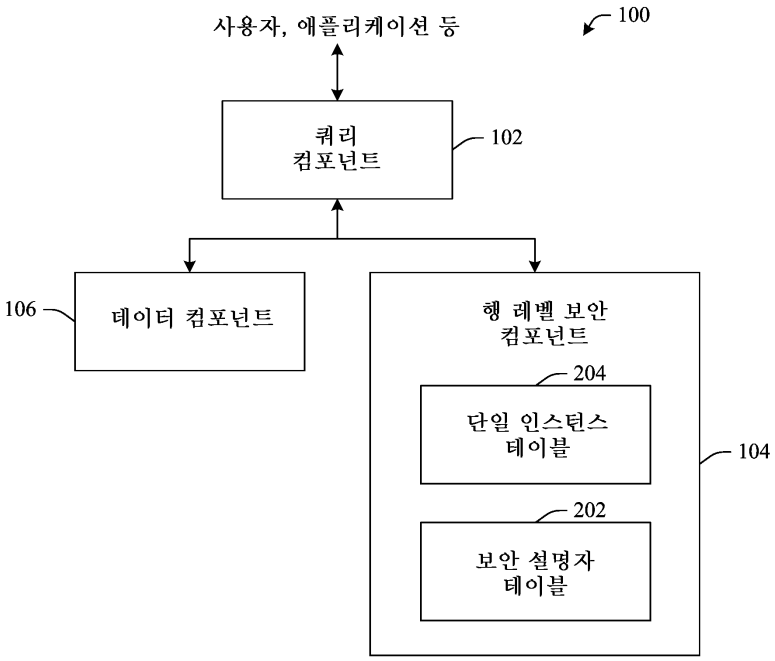
[0013] 106 : 데이터 컴포넌트

도면

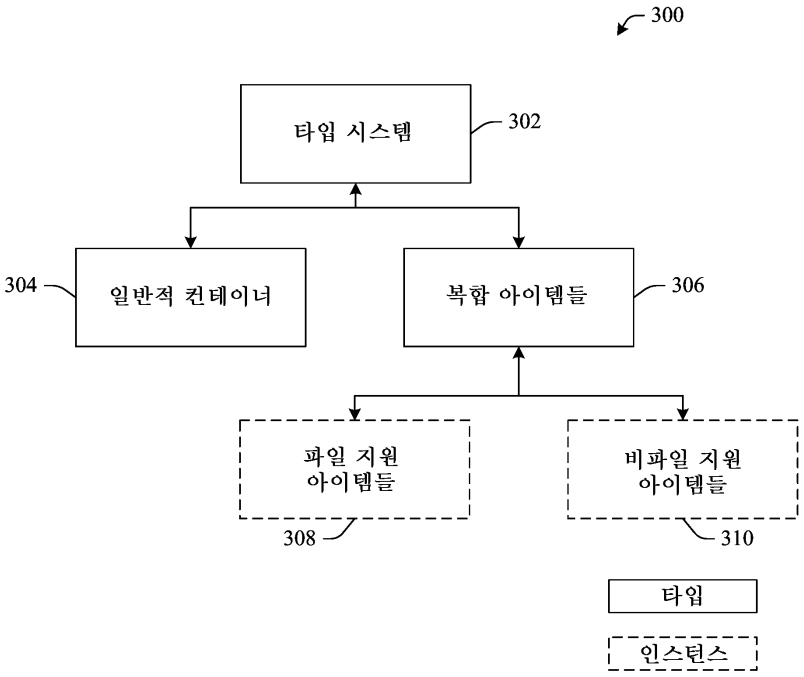
도면1



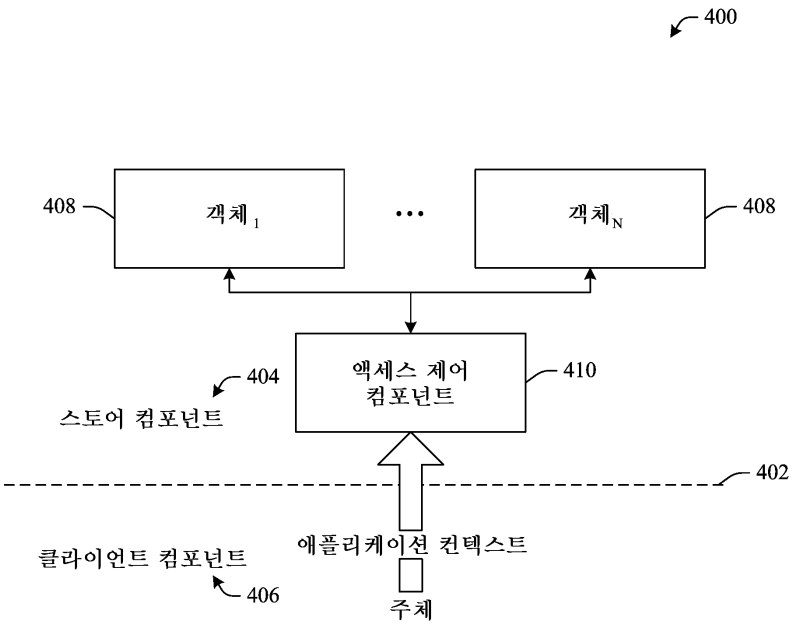
도면2



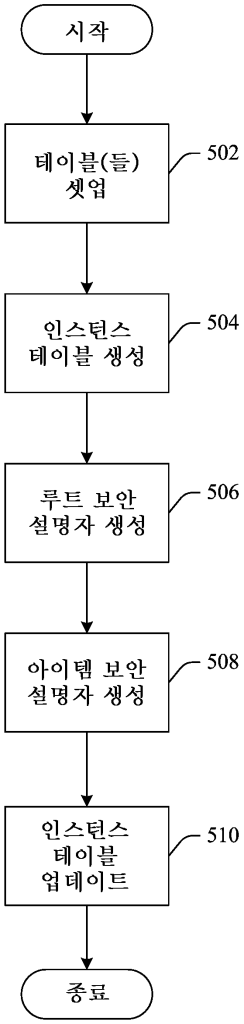
도면3


300
타입 시스템
302
304
일반적 컨테이너
복합 아이템들
306
파일 지원
아이템들
308
비파일 지원
아이템들
310
타입
인스턴스

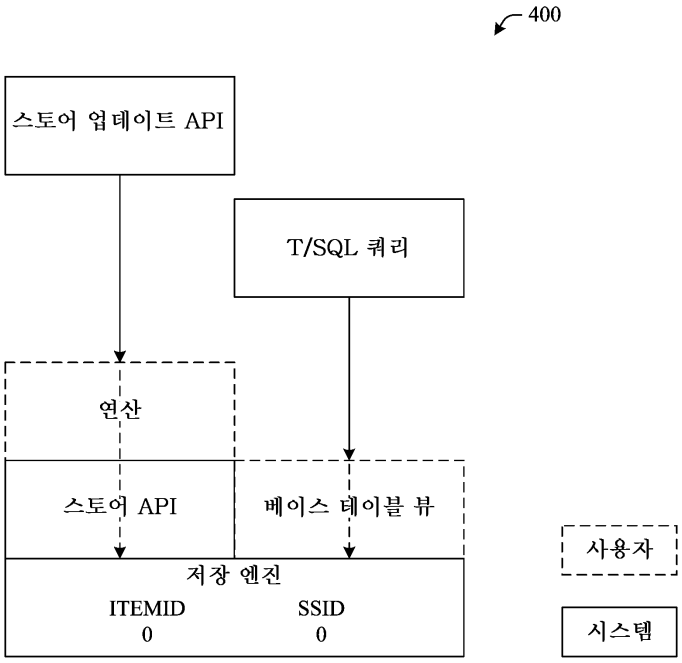
도면4


400
408
객체 1
객체 N
408
404
스토어 컴포넌트
액세스 제어
컴포넌트
410
402
클라이언트 컴포넌트
406
애플리케이션 컨텍스트
주체

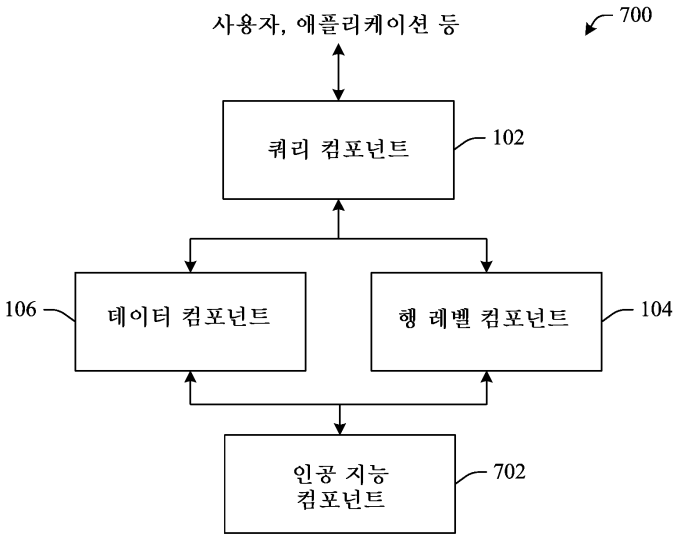
도면5


시작
테이블(들) 셋업
502
인스턴스 테이블 생성
504
루트 보안 설명자 생성
506
아이템 보안 설명자 생성
508
인스턴스 테이블 업데이트
510
종료

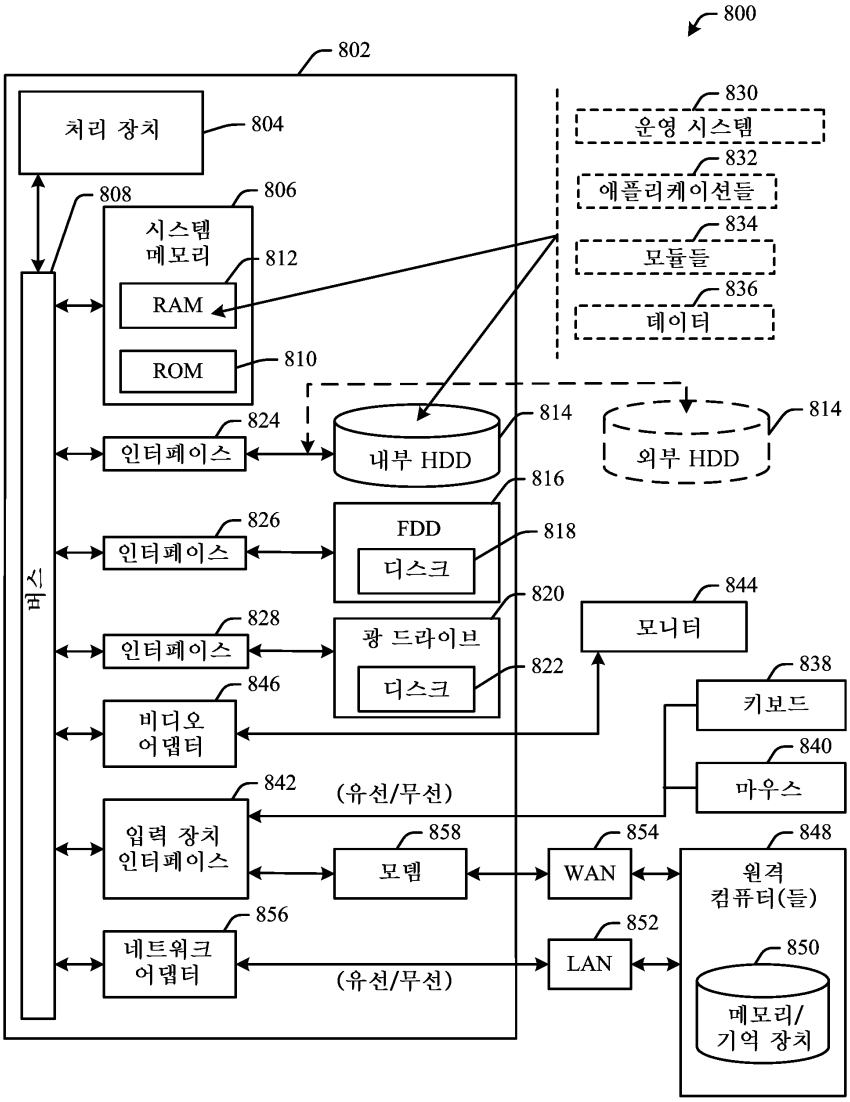
도면6


400
스토어 업데이트 API
T/SQL 쿼리
연산
스토어 API
베이스 테이블 뷰
저장 엔진
ITEMID
0
SSID
0
사용자
시스템

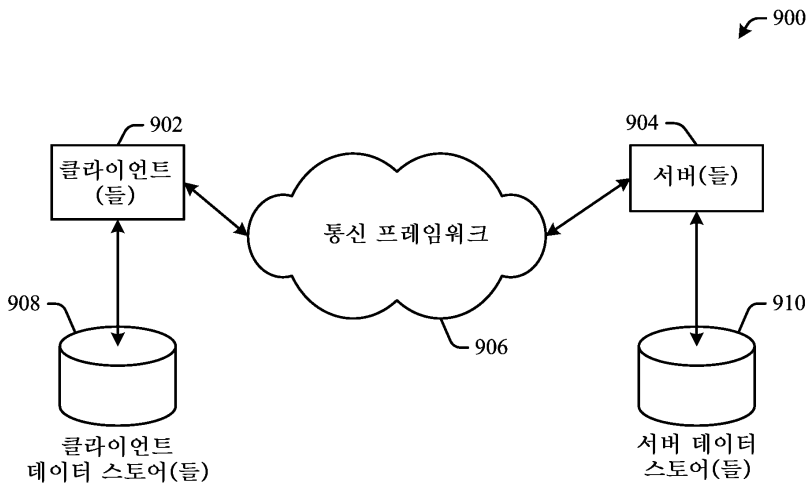
도면7


사용자, 애플리케이션 등
700
쿼리 컴포넌트
102
106
데이터 컴포넌트
행 레벨 컴포넌트
104
인공 지능
컴포넌트
702

도면8


800
802
처리 장치
804
830
운영 시스템
832
애플리케이션들
834
모듈들
836
데이터
808
806
시스템
메모리
812
RAM
ROM
810
824
인터페이스
814
내부 HDD
814
외부 HDD
816
826
인터페이스
FDD
818
디스크
820
844
모니터
828
인터페이스
광 드라이브
822
디스크
838
키보드
846
비디오
어댑터
840
마우스
842
(유선/무선)
입력 장치
인터페이스
858
모뎀
854
WAN
848
원격
컴퓨터(들)
856
852
네트워크
어댑터
LAN
850
(유선/무선)
메모리/
기억 장치
버스

도면9


900
902
클라이언트
(들)
904
서버(들)
통신 프레임워크
908
906
910
클라이언트
데이터 스토어(들)
서버 데이터
스토어(들)