(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 September 2008 (12.09.2008)

PCT

(10) International Publication Number
WO 2008/109244 A1

(51) International Patent Classification:
G06F 21/20 (2006.01)

(21) International Application Number:
PCT/US2008/054152

(22) International Filing Date:
15 February 2008 (15.02.2008)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/682,093 5 March 2007 (05.03.2007) US

(71) Applicant *(for all designated States except US)*: **MICROSOFT CORPORATION** [US/US]; One Microsoft Way, Redmond, Washington 98052-6399 (US).

(72) Inventors: **UNGUREANASU, Cezar**; One Microsoft Way, Redmond, Washington 98052-6399 (US). **ZYBURA, John H.**; One Microsoft Way, Redmond, Washington 98052-6399 (US). **LEIBMANN, Matthias**; One Microsoft Way, Redmond, Washington 98052-6399 (US). **GAJULA, Pallavi**; One Microsoft Way, Redmond, Washington 98052-6399 (US).

(81) **Designated States** *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: DYNAMIC COMPUTATION OF IDENTITY-BASED ATTRIBUTES

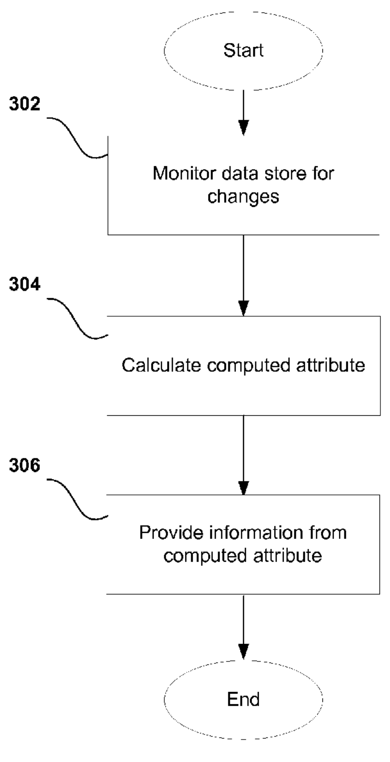

Start
302
Monitor data store for changes
304
Calculate computed attribute
306
Provide information from computed attribute
End

Fig. 3

(57) Abstract: Enterprise Identity Management systems control access to information derived from identity-related data stored in various data repositories. An identity-based management system can automatically and dynamically compute derived data when the source data changes. Rule-base tools can be used to compute derived data from arbitrary attribute-based datasets. Dynamic computation of identity-based attributes within information system servers allows data to be aggregated and normalized from multiple data sources deployed across an organization so that updated related information can be persisted and pushed to various servers in the organization.

DYNAMIC COMPUTATION OF IDENTITY-BASED ATTRIBUTES

Background

[0001] Enterprise Identity Management systems control access to information derived from identity-related data stored in various data repositories. Examples of the data include email distribution lists or system resource access rights based on identity-based attributes such as job function, office location, department, and reporting relationship. Maintaining such information often requires an administrator's direct input and a knowledge of the relationships of the data, which can be quite complex.

[0002] As identity attributes (such as users, employees, contractors, customers, and the like) join, leave, and change positions within the organization, their attributes frequently change. This usually requires that relational data be updated as needed to reflect these changed attributes. Conversely, derived data sets (such as new lists or entitlements) are populated based on current user attributes. The challenge of maintaining this derived data current becomes even more difficult as the both the source and the derived datasets become larger and/or more volatile and/or more distributed.

Summary

[0003] This summary is provided to introduce a selection of concepts in a simplified form that are further described below in the detailed description. This summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended as an aid in determining the scope of the claimed subject matter.

[0004] The present disclosure is directed to efficient means of computing derived data for an identity-based management system automatically and dynamically, such as when the source data changes. Computing derived data automatically and dynamically reduces IT costs through automation and timeliness, and increases security by ensuring entitlements are accurately assigned and reported. Dynamically computing the derived data can be applied to identity-based application domains such as attribute-based distribution list memberships regardless where that data is natively accessed.

[0005] Dynamic computation of identity-based attributes can be implemented as a general rule-based tool for computing data regardless of the types of attributes and attribute relationships used by any specific organization. The rule-base tool can be used to compute derived data from arbitrary attribute-based datasets.

[0006] The rule set can comprise rules that describe how disparate objects in an identity-based application store are related to one another, and rules which describe the actual computations over the attributes of related objects. The two types or rules allow high-order business logic to be described in a more modular and intuitive way. Productivity is typically improved through reuse of rules describing similar attribute computations which share the same conditions for object relationships.

[0007] Dynamic computation of identity-based attributes within information system servers allows data to be aggregated and normalized from multiple data sources deployed across the organization, so that not only is all the requisite data for the computation is persisted, but the dynamically computed results can be pushed to the system(s) to be updated with the result as the results are computed. Accordingly, systems (to where the computed results are pushed) do not require knowledge of the computation and necessary relationships of the data. The systems (to where the data is pushed) typically works with its known data types and attributes, while the computation, relationship building, and derivation are typically performed by the information system server. Incorporation of computation of identity-based attributes within an information system server can also be used to leverage the efficient relational query capabilities within an SQL server.

[0008] These and other features and advantages will be apparent from a reading of the following detailed description and a review of the associated drawings. It is to be understood that both the foregoing general description and the following detailed description are explanatory only and are not restrictive. Among other things, the various embodiments described herein may be embodied as methods, devices, or a combination thereof. Likewise, the various embodiments may take the form of an entirely hardware embodiment, an entirely software embodiment or an embodiment combining software and hardware aspects. The disclosure herein is, therefore, not to be taken in a limiting sense.

Brief Description of the Drawings

[0009] FIGURE 1 is an illustration of an example operating environment and system for dynamic computation of identity-based attributes.

[0010] FIGURE 2 is an illustration of dynamically calculated, identity-based attributes.

[0011] FIGURE 3 is a top-level illustration of a flow diagram for dynamic computation of identity attributes.

Detailed Description

[0012] As briefly described above, embodiments are directed to dynamic computation of identity-based attributes. With reference to FIGURE 1, one example system for managed code assemblies includes a computing device, such as computing device 100. Computing device 100 may be configured as a client, a server, a mobile device, or any other computing device that interacts with data in a network based collaboration system. In a very basic configuration, computing device 100 typically includes at least one processing unit 102 and system memory 104. Depending on the exact configuration and type of computing device, system memory 104 may be volatile (such as RAM), non-volatile (such as ROM, flash memory, etc.) or some combination of the two. System memory 104 typically includes an operating system 105, one or more applications 106, and may include program data 107 such that data store monitor 120, attribute computer 122, and cache 124 can be implemented (which are discussed below).

[0013] Computing device 100 may have additional features or functionality. For example, computing device 100 may also include additional data storage devices (removable and/or non-removable) such as, for example, magnetic disks, optical disks, or tape. Such additional storage is illustrated in FIGURE 1 by removable storage 109 and non-removable storage 110. Computer storage media may include volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information, such as computer readable instructions, data structures, program modules, or other data. System memory 104, removable storage 109 and non-removable storage 110 are all examples of computer storage media. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by computing device 100. Any such computer storage media may be part of device 100. Computing device 100 may also have input device(s) 112 such as keyboard, mouse, pen, voice input device, touch input device, etc. Output device(s) 114 such as a display, speakers, printer, etc. may also be included.

[0014] Computing device 100 also contains communication connections 116 that allow the device to communicate with other computing devices 118, such as over a network. Networks include local area networks and wide area networks, as well as other large scale networks including, but not limited to, intranets and extranets. Communication connection 116 is one example of communication media. Communication media may typically be embodied by computer readable instructions, data structures, program modules, or other data in a modulated data signal, such as a carrier wave or other transport mechanism, and includes any information delivery media. The term “modulated data signal” means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. The term computer readable media as used herein includes both storage media and communication media.

[0015] In accordance with the discussion above, computing device 100 system memory 104 (and processor 102, and related peripherals can be used to implement data store monitor 120, attribute computer 122, and cache 124. Data store monitor 120, attribute computer 122, and cache 124 in an embodiment can be used to implement dynamic computation of identity-based attributes (described below with reference to Figures 2-3). Data store monitor 120 can be used for detecting changes to identity-based attributes for structured data in a data store and changes to relationships amongst the structured data. Attribute computer 122 can be used for computing in response to a detected change a computed attribute for a query (that can be an identity function, for example). Cache 124 can be used for persisting the computed attribute and sending the computed attribute in response to a query that be the same (or substantially similar to) or different from the query for which the change was detected.

[0016] FIGURE 2 is an illustration of dynamically calculated, identity-based attributes. In an embodiment, an identity-based attribute computation can be directly associated with an attribute chosen to hold the result of the computation. The chosen attribute thus becomes dedicated to the result of the computation, and can be referred to as a "computed attribute." The value of a computed attribute typically only changes when the result of the computation itself changes.

[0017] A straightforward implementation could perform the actual computation on-demand whenever the computed attribute is accessed. However, to improve processing efficiency, another implementation can use a more sophisticated system whereby the result of the computation is cached. While this approach typically requires more physical storage, it typically reduces computational overhead since identity-based attribute computations are normally only executed when the inputs into the computation change.

[0018] In dynamic computation of identity-based attributes, the values of a computed attribute on a specific object are calculated from the attribute values present on a set of related objects (which may or may not include the object holding the computed attribute). In a server implementation, the object holding the computed attribute is called the "base object" (220), and the set of related objects are called "match objects" (210, 230, and 240). The base object itself may also be a match object.

[0019] After the base and match objects have been established, an arbitrary operation over the set of values from all match objects can be performed to produce a resultant set of values for the computed attribute. In general, any mathematical operation can be used for the computation, depending on the amount of flexibility desired for a given scenario.

[0020] In one embodiment, the operations can be restricted to only allow the relocation of values to different operations. (In other embodiments operations can be supported that modify the content of individual values, such as arithmetic or string manipulation operators.) The feature of identity-based attribute computations is typically implemented on top of "normalized" identity data store. In the embodiment, any value-level transformations are performed before the data enters the normalized identity store. (In other embodiments value-level transformation operators can be provided.)

[0021] In the embodiment, the transformation from match object values into computed attribute values is defined by listing the names of one or more attributes from the match objects which thus forms the computed value set. These attributes are called "value attributes." Thus the result of the computation is the union of all values from all value attributes from all match objects.

[0022] As illustrated in Figure 2, two relationships (250 and 260) and two value attributes are used to compute five values from the three match objects. Thus components used to define a computed attribute for the example comprises two lists. The first is the list of relationships, and the second is the list of value attributes. The definition of CA1 in this example can be given as follows in Table I:

Table I

CA1 Definition
Relationships: "Relationship 1", "Relationship 2"
Value Attributes: "VA1", "VA2"

[0023] In the embodiment, the definitions are stored in the DSML schema of the aggregate identity store, using custom XML elements to extend the DSML schema description format. An example extended DSML definition for CA1 can be written as:

```
<ms-dsml:computed-attribute-type id="CA1"    ms-
                                 dsml:indexed="true"
                                 ms-dsml:indexable="true">
        <dsml:name>CA1</dsml:name>
        <ms-dsml:value-attribute ref="#VA1" />
        <ms-dsml:value-attribute ref="#VA2" />
        <ms-dsml:relationship ref="#Relationship1" />
        <ms-dsml:relationship ref="#Relationship2" />
   </ms-dsml:computed-attribute-type>
```

[0024] The example DSML definition does not describe how the relationships are defined. Not defining the relationships in the DSML code illustrates an important modularity in the definitions of identity-based attribute computations. The conditions that define a relationship are normally established independently of the computations which use those relationships. The modularity allows the logic for a relationship to be reused in multiple computations. It also potentially allows different parties, with different areas of expertise, to author relationships and computations independently. The modularity of the code can thus offer a significant improvement in system manageability.

[0025] A relationship can be defined as any set of conditions that can be evaluated to determine the set of match objects for a given base object. In an embodiment, a relationship can be defined by three optional components: a “filter” on the match object, a “filter” on the base object, and a list of “search conditions.” A “filter” is an arbitrary test that can be applied to a single object to determine if it is a candidate for the relationship. A “search condition” is a test that can take as input both the match object and the base object, and determine if the two objects are related according to the specific attribute values on each. A relationship is determined to exist between two objects when the match filter test passes, when the base filter test passes, and when all search condition tests pass. If any of these tests has not been defined for a given relationship, then that test is considered to always pass. The definition of a relationship can be given as follows in Table II:

Table II

Relationship Definition
Match filter: <Arbitrary test on match object>
Base filter: <Arbitrary test on base object>
Search Condition 1: <Arbitrary condition between both objects>
Search Condition 2: <Arbitrary condition between both objects>
…
Search Condition N: <Arbitrary condition between both objects>

[0026] In an embodiment, a "filter" is defined as a mathematical combination of Boolean operators, conditional operators, constant values, and object attributes. A typical example for a filter might be "title starts-with 'VP' and buildingNumber = 22." This filter would only pass for objects whose "title" attribute has the string prefix "VP" and whose "buildingNumber" attribute has the numeric value 22.

[0027] A "search condition" can be primarily defined by two listing attributes, one called the "base attribute" and one called the "match attribute." A simple search condition is considered to "pass" for a given pair of base object and match object when the match attribute on the match object and the base attribute on the base object share at least one value in common.

[0028] However, there can be three additional options on a search condition that influence the matching logic. The first is an "inversion" flag which negates the result of the search condition. In other words, an inverted search condition passes for a given pair of objects usually only if the match attribute on the match object has no values in common with the base attribute on the base object. The second and third options are "transitivity" flags which can be independently set on either the match or the base object (discussed below). The definition of a search condition can be given as follows in Table III:

Table III

Search Condition Definition
Match attribute: <Name of attribute on match object>
Base attribute: <Name of attribute on base object>

```
Inverted: <True/False>
Match attribute transitive: <True/False>
Base attribute transitive: <True/False>
```

[0029] Normally, a search condition directly tests the immediate values of the match attribute against the immediate values of the base attribute. However, if the transitive flag is turned on for either attribute, then the value set that will be tested for that attribute will actually be the "transitive closure" of the immediate values of the attribute. The transitive closure operation is usually only defined for attributes of reference type, so it is normally invalid to set this flag for a match or base attribute which is not of a reference type. The actual transitive closure operation typically has the standard mathematical definition: the result is the set of all objects reachable through the specified reference attribute in any number of "hops."

[0030] For example, the standard "manager" reference-type attribute can be used. For a given user object, the immediate value of the attribute is a reference to the one person who is that user's manager. Additionally, the transitive closure of this attribute is the entire set of the user's manager, the user's manager's manager, and the like, all the way up the management chain of the user in question.

[0031] Like the computed attribute definitions discussed above, relationship definitions can be stored in an aggregated identity store schema using an extended DSML grammar. An example of a relationship definition in using an extended DSML grammar can be as follows:

```
<ms-dsml:relationship-type id="Relationship1">
     <dsml:name>Relationship1</dsml:name>
     <ms-dsml:match-filter>
          <ms-dsml:or-condition>
               <ms-dsml:and-condition>
                    <ms-dsml:relational-predicate
operator="not-contains">
                         <ms-dsml:left-operand>
                              <ms-dsml:mv-attribute
ref="department" />
                         </ms-dsml:left-operand>
                         <ms-dsml:right-operand>
                              <ms-dsml:constant-
value>Internal Affairs</ms-dsml:constant-value>
                         </ms-dsml:right-operand>
                    </ms-dsml:relational-predicate>
                </ms-dsml:and-condition>
            </ms-dsml:or-condition>
        </ms-dsml:match-filter>
        <ms-dsml:base-filter>
            <ms-dsml:or-condition>
                <ms-dsml:and-condition>
                    <ms-dsml:relational-predicate
operator="substring-any">
                        <ms-dsml:left-operand>
                            <ms-dsml:mv-attribute
ref="description" />
                        </ms-dsml:left-operand>
                        <ms-dsml:right-operand>
                            <ms-dsml:constant-
value>vendor</ms-dsml:constant-value>
                        </ms-dsml:right-operand>
                    </ms-dsml:relational-predicate>
                </ms-dsml:and-condition>
            </ms-dsml:or-condition>
        </ms-dsml:base-filter>
        <ms-dsml:search-condition ms-dsml:positive="false">
            <ms-dsml:base-attribute ms-
dsml:transitive="false" ref="#company" />
            <ms-dsml:match-attribute ms-
dsml:transitive="false" ref="#department" />
        </ms-dsml:search-condition>
        <ms-dsml:search-condition ms-dsml:positive="true">
            <ms-dsml:base-attribute ms-
dsml:transitive="true" ref="#manager" />
            <ms-dsml:match-attribute ms-
dsml:transitive="false" ref="#assistant" />
        </ms-dsml:search-condition>
</ms-dsml:relationship-type>
```

[0032] As noted above, relationship and computed attribute definitions frequently reference other attributes in the identity store schema. According to the given schema, any of these referenced attributes can be themselves computed attributes. Thus a computed attribute can be referenced as a value attribute in another computed attribute. A computed attribute can also be used as a match attribute or base attribute in a search condition. Additionally a computed attribute could be referenced in the match or base filters of a relationship.

[0033] These types of nested definitions are allowed to any degree of depth or complexity, such that typically none of the definitions are cyclically related. The dependencies are calculated between the various nested definitions, and calculations are performed in the proper order such that the dependencies are obeyed and all computation results are kept current.

[0034] FIGURE 3 is a top-level illustration of a flow diagram for dynamic computation of identity attributes. In operation 302, a data store is monitored for changes to identity-based attributes for structured data and changes to relationships amongst the structured data.

[0035] In operation 304, a computed attribute is dynamically computed for a first query. The calculation is performed in response to the detected change as detected by the data store monitoring above. The query can be persisted in a cache for replying to queries that are the same or similar to (sharing some identical components) the first query. The cache can be implemented in a server.

[0036] In operation 306, the information from the computed attribute is provided in response to a second query. In various embodiments, the second query can be the same as, a duplicate of, similar to, different from (and the like) as the first query. The second query is received after the computed attribute has been computed.

[0037] In various embodiments, the information from the computed attribute can be published (either in conjunction with, or separately from operation 307) by exporting the computed result to a connected system. The connected system can then use the exported results to query against the connected systems querying capabilities.

[0038] The above specification, examples and data provide a complete description of the manufacture and use of embodiments of the invention. Since many embodiments of the invention can be made without departing from the spirit and scope of the invention, the invention resides in the claims hereinafter appended.

WE CLAIM:

1. A computer-implemented method for dynamic computation of identity-based attributes, comprising:

[302] monitoring a data store for changes to identity-based attributes for structured data and changes to relationships amongst the structured data;

[304] in response to a detected change, dynamically computing a computed attribute for a first query; and

[306] in response to a second query, providing information from the computed attribute.

2. The method of claim 1 wherein the first and second queries are the same query.

3. The method of claim 1 wherein the second query is received after the computed attribute has been computed.

4. The method of claim 1 wherein the computed attribute is persisted in a cache.

5. The method of claim 4 wherein the computed attribute is distributed to another system.

6. The method of claim 1 wherein the structured data relationship is defined using a filter on a match object, a filter on an object, and a list of search conditions.

7. The method of claim 6 wherein one of the search conditions is an element of a set comprising an inversion element and a transitivity element.

8. The method of claim 7 wherein the components used to define the computed attribute are defined using a schema.

9. The method of claim 1 wherein the values of a computed attribute on a specific object are calculated from the attribute values present on a set of related objects.

10. The method of claim 1 wherein the operations used to compute a computed attribute allows the relocation of values to different operations.

11. The method of claim 10 wherein the operations that modify the content of individual values by using arithmetic and/or string manipulation operators.

12. The method of claim 1 wherein the components used to define the computed attribute comprise a first list of relationships and a second list of value attributes

13. The method of claim 1 wherein the data store is a normalized identity data store.

14. The method of claim 13 wherein value-level transformations are performed before the data enters the normalized identity store.

15. The method of claim 13 wherein value-level transformations are performed after the data enters the normalized identity store.

16. A system for event-based and/or state changes-based parsing of an input file, comprising:

[220] a data store monitor for detecting changes to identity-based attributes for structured data in a data store and changes to relationships amongst the structured data;

[222] an attribute computer for computing in response to a detected change a computed attribute for a first query that is an identity function; and

[224] a cache for persisting the computed attribute and sending the computed attribute in response to a second query.

17. The system of claim 16 wherein a computed attribute is referenced as a value attribute in another computed attribute.

18. A tangible medium comprising computer-executable instructions for:

monitoring a data store for changes to identity-based attributes for structured data and changes to relationships amongst the structured data;

in response to a detected change, dynamically computing a computed attribute for a first query that is an identity function; and

persisting the computed attribute.

19. The tangible medium of claim 18 further comprising providing information from the computed attribute in response to a second query.

20. The tangible medium of claim 18 further comprising using computed attribute as a match attribute or base attribute in a search condition.

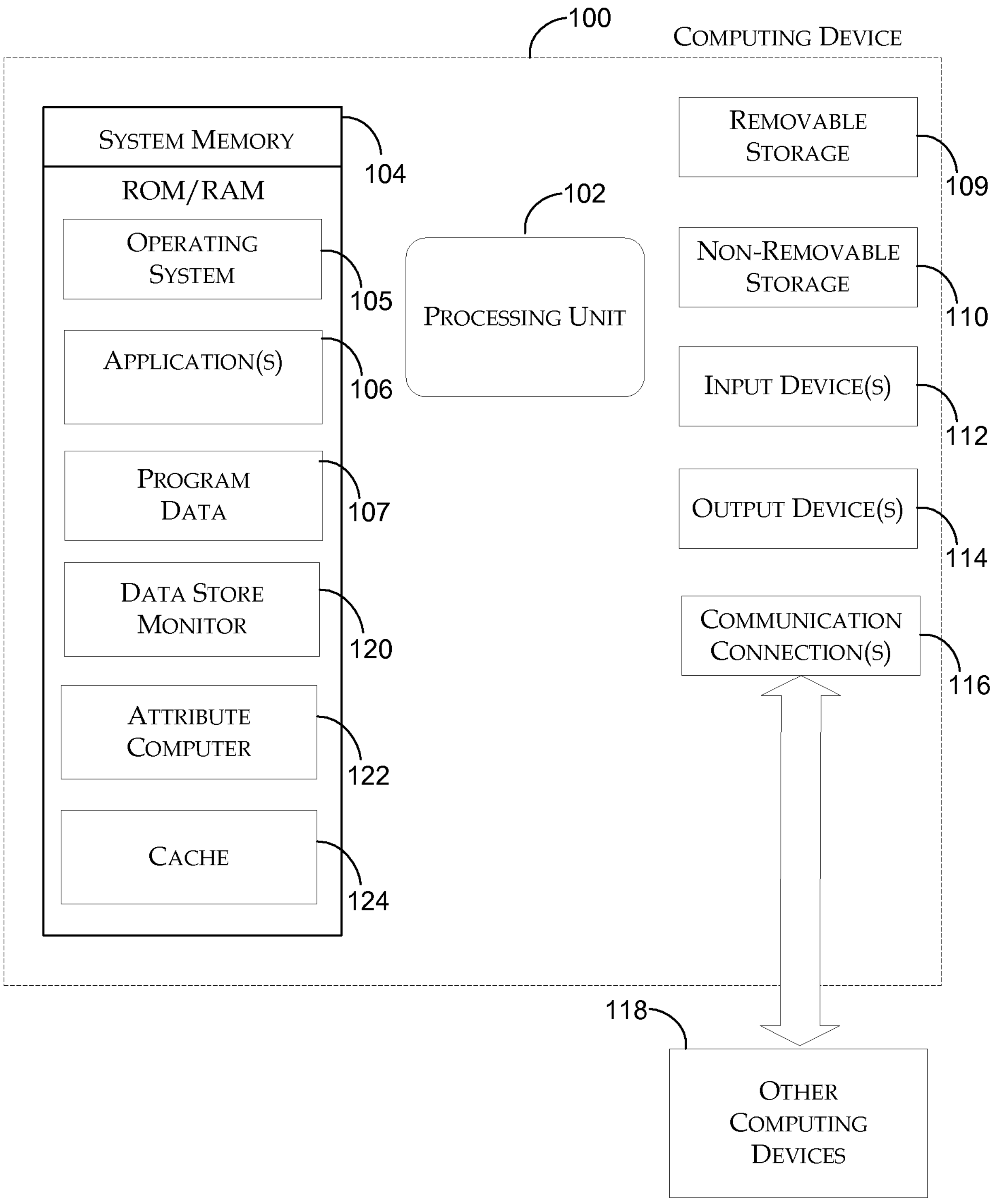
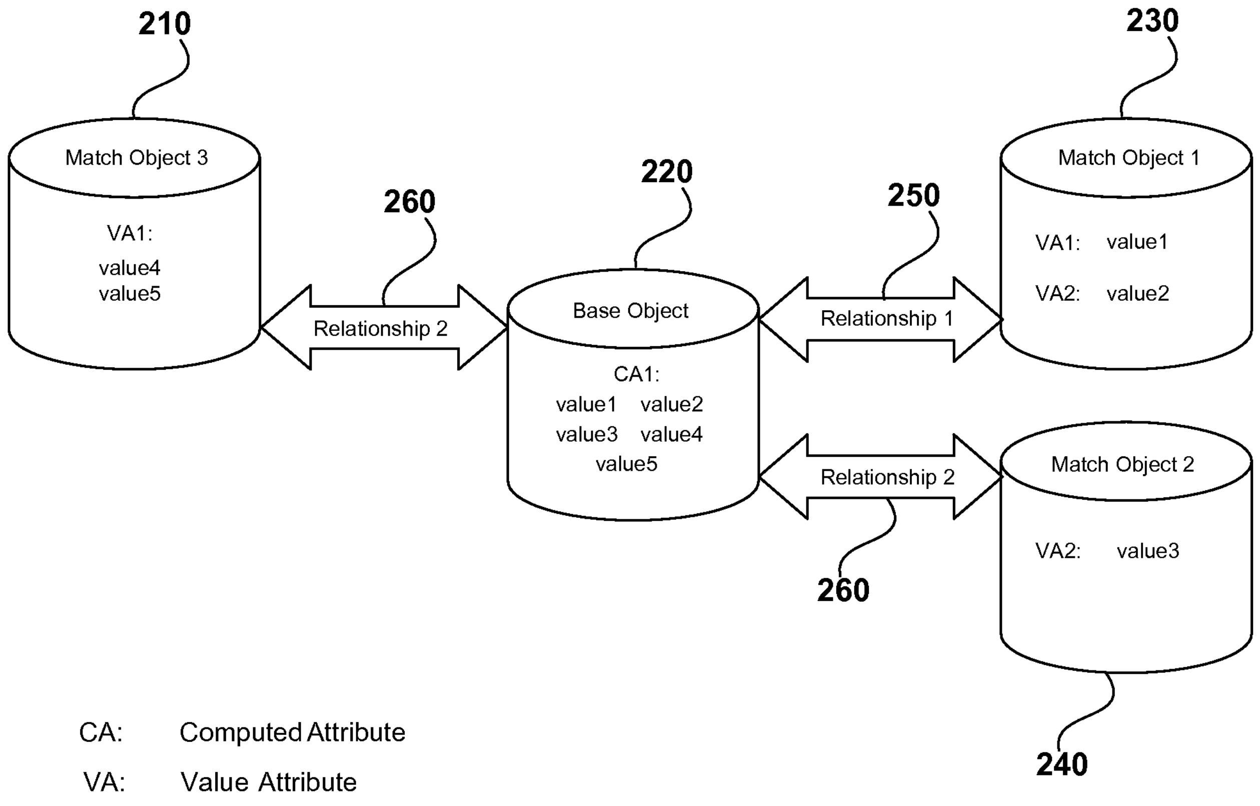

100
COMPUTING DEVICE
SYSTEM MEMORY
104
ROM/RAM
OPERATING SYSTEM
105
APPLICATION(S)
106
PROGRAM DATA
107
DATA STORE MONITOR
120
ATTRIBUTE COMPUTER
122
CACHE
124
102
PROCESSING UNIT
REMOVABLE STORAGE
109
NON-REMOVABLE STORAGE
110
INPUT DEVICE(S)
112
OUTPUT DEVICE(S)
114
COMMUNICATION CONNECTION(S)
116
118
OTHER COMPUTING DEVICES

Fig. 1



210
Match Object 3
VA1:
value4
value5
260
Relationship 2
220
Base Object
CA1:
value1 value2
value3 value4
value5
250
Relationship 1
Relationship 2
260
230
Match Object 1
VA1: value1
VA2: value2
Match Object 2
VA2: value3
240
CA: Computed Attribute
VA: Value Attribute

Fig. 2

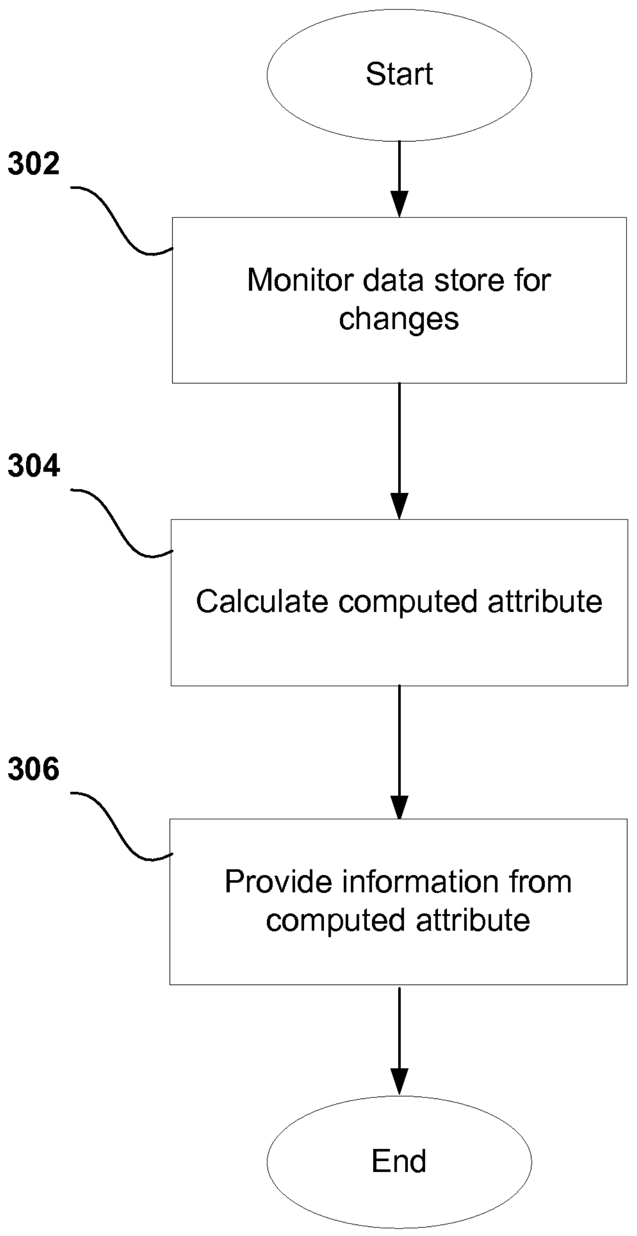

Start
302
Monitor data store for changes
304
Calculate computed attribute
306
Provide information from computed attribute
End

Fig. 3

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2008/054152

A. CLASSIFICATION OF SUBJECT MATTER

G06F 21/20(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 8 : G06F 11/30, 15/20, 17/30

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean Utility Models and applications for utility model since 1975
Japanses Utility Models and applications for utility model since 1975

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

eKIPASS(KIPO internal) "identity", "attribute", "update"

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2002-0144142 A1 (SHOHAT DALIA) 3 October 2002 See abstract, claims	1-20
A	US 6088679 A (BARKLEY JOHN) 11 July 2000 See abstract, claims	1-20
A	US 6202066 B1 (BARKLY JOHN, et al.) 13 March 2001 See abstract, claims	1-20
A	US 7171411 B1 (LEWIS NINA, et al.) 30 January 2007 See abstract, claims	1-20

☐ Further documents are listed in the continuation of Box C. ☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search 23 JULY 2008 (23.07.2008)	Date of mailing of the international search report **24 JULY 2008 (24.07.2008)**
Name and mailing address of the ISA/KR Korean Intellectual Property Office Government Complex-Daejeon, 139 Seonsa-ro, Seo-gu, Daejeon 302-701, Republic of Korea Facsimile No. 82-42-472-7140	Authorized officer LEE, Jong Ick Telephone No. 82-42-481-8373

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2008/054152

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2002-0144142 A1	03.10.2002	EP 1248178 B1	07.01.2004
US 6088679 A	11.07.2000	NONE	
US 6202066 B1	13.03.2001	NONE	
US 7171411 B1	30.01.2007	NONE	