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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
11 June 2020 (11.06.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/113700 A1

(51) International Patent Classification:
G06F 21/60 (2013.01)

(21) International Application Number:
PCT/CN2018/123072

(22) International Filing Date:
24 December 2018 (24.12.2018)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
201811483548.0 05 December 2018 (05.12.2018) CN

(71) Applicant: **BEIJING DIDI INFINITY TECHNOLOGY AND DEVELOPMENT CO., LTD.** [CN/CN]; Building 34, No. 8 Dongbeiwang West Road, Haidian District, Beijing 100193 (CN).

(72) Inventor: **ZHONG, Zhe**; Building 34, No. 8 Dongbeiwang West Road, Haidian District, Beijing 100193 (CN).

(74) Agent: **METIS IP (CHENGDU) LLC**; Room 808, 8th Floor, Building B7, Block D, Tianfu Jingrong Center, No. 99 West Section of Hupan Road, Tianfu New District, Chengdu, Sichuan 610213 (CN).

(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, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, 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, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ,

(54) Title: SYSTEMS AND METHODS FOR CONTROLLING A USER'S ACCESS TO AN OBJECT

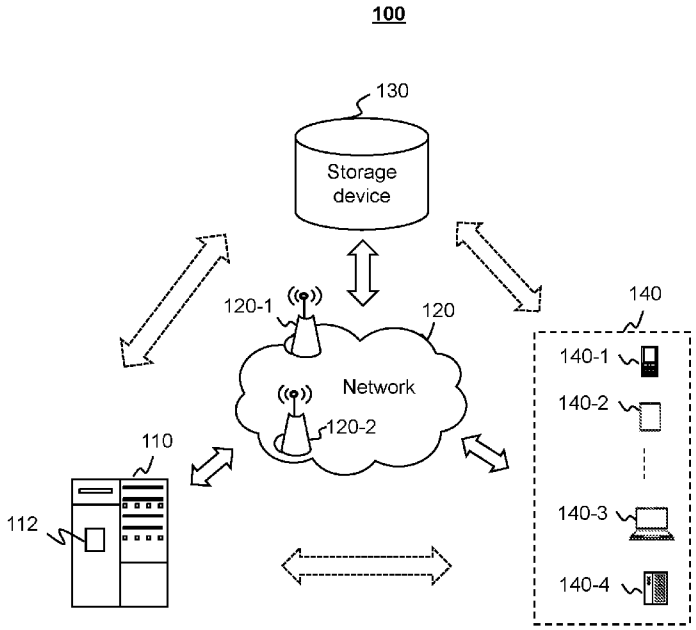

FIG. 1

(57) Abstract: The present disclosure relates to a system and method for controlling a user's access to an object. A request for access to the object may be obtained from a user. Access authority information of the user may be determined based on a mapping relation between one or more predefined access permissions and one or more users. The access authority information of the user may indicate one or more permissions. An access check may be performed to resolve the request. The one or more permissions may be included in a plurality of permissions that are stored in a storage of the one or more storage device as a data structure. The data structure may include at least one directed acyclic graph (DAG) structure.

WO 2020/113700 A1

[Continued on next page]

TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

SYSTEMS AND METHODS FOR CONTROLLING A USER'S ACCESS TO AN OBJECT

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to Chinese Patent Application No. 201811483548.0, filed on December 05, 2018, the contents of which are incorporated herein by reference.

TECHNICAL FIELD

[0002] This disclosure generally relates to the field of information security, and more particularly, relates to systems and methods for storing permission information and controlling a user's access to an object based on the permission information.

BACKGROUND

[0003] Permission management system is widely used in most of the installed centralized and distributed operating systems. When a user (or a user program) requests an access to an object, such as a resource (e.g., a file, a database) or perform an operation (e.g., read, write, edit, or delete) on a resource, the permission management system generally performs an access check to resolve the request based on permission information. The permission management system generally defines, and/or stores the permission information as a data structure to facilitate the management of the permission information. The design of the data structure can affect the refinement degree and flexibility of permission management (e.g., a management of inheritance relationships in the permission information). Refined permission control and multi-level permission inheritance relationships can reduce the cost of user operation and maintenance. Therefore, it is desirable to provide systems and methods for storing permission information as a more refined or organizable data structure, and controlling a user's access to an object based on the permission information more efficiently.

SUMMARY

[0004] According to one aspect of the present disclosure, a method for controlling a user's access to an object is provided. The method may be implemented on a computing device having one or more processors and one or more storage devices. The method may include obtaining a request for access to the object from the user. The method may also include determining access authority information of the user based on a mapping relation between one or more predefined access permissions and one or more users. The access authority information of the user may indicate one or more permissions. The method may further include performing an access check to resolve the request. The one or more permissions may be included in a plurality of permissions that are stored in a storage of the one or more storage device as a data structure. The data structure may include at least one directed acyclic graph (DAG) structure. Each DAG structure may include a plurality of nodes and a plurality of directed edges. Each node may represent a permission. Each directed edge may include a direction from a parent node to a child node among the plurality of nodes. The request may include node information of at least one key node to be checked. The method may also include determining whether the node information of the at least one key node matches one of the one or more permissions indicated by the access authority information. The method may further include granting the request for access to the object upon determination that the match exists.

[0005] In some embodiments, the each DAG structure may allow permission inheritance among the plurality of permissions. The parent node may have one or more permissions corresponding to one or more child nodes of the parent node.

[0006] In some embodiments, the access authority information of the user may further include one or more roles of the user associated with the one or more permissions.

[0007] In some embodiments, the object may include an operation on a resource.

[0008] In some embodiments, the at least one DAG structure may include a first DAG structure. The first DAG structure may include a plurality of first nodes of the plurality of nodes associated with functional operation permissions.

[0009] In some embodiments, the object may include a resource.

[0010] In some embodiments, the at least one DAG structure may include a second DAG structure. The second DAG structure may include a plurality of second nodes of the plurality of nodes associated with permissions to data scopes.

[0011] In some embodiments, the plurality of permissions may include at least one of a permission of granting a permission of the user to another user, a permission of creating a permission for another user, a permission of inquiring a permission of a user, a permission of modifying a permission of a user, and a permission of revoking a permission of a user.

[0012] In some embodiments, for each key node of the at least one key node, the method may also include traversing from the each key node to a root node of the at least one DAG structure. The method may further include determining whether one of the one or more permissions indicated by the access authority information is traversed. The method may still further include determining that the match exists upon determination that one of the one or more permissions indicated by the access authority information is traversed. The method may still further include determining that the match does not exist upon determination that none of the one or more permissions indicated by the access authority information is traversed.

[0013] In some embodiments, the method may also include verifying the node information of the at least one key node included in the request for access to the object.

[0014] In some embodiments, the method may also include refusing the request upon determination that the node information of the at least one key node included in the request for access to the object is not verified.

[0015] According to another aspect of the present disclosure, a system for controlling a user's access to an object is provided. The system may include at least one computer-readable storage medium storing a set of instructions; and at least one processor configured to communicate with the at least one computer-readable storage medium, wherein when executing the set of instructions, the at least one processor may be directed to cause the system to obtain a request for

access to the object from the user. The at least one processor may be directed to cause the system to determine access authority information of the user based on a mapping relation between one or more predefined access permissions and one or more users. The access authority information of the user may indicate one or more permissions. The at least one processor may be directed to cause the system to perform an access check to resolve the request. The one or more permissions may be included in a plurality of permissions that are stored in a storage of the one or more storage device as a data structure. The data structure may include at least one directed acyclic graph (DAG) structure. Each DAG structure may include a plurality of nodes and a plurality of directed edges. Each node may represent a permission. Each directed edge may include a direction from a parent node to a child node among the plurality of nodes. The request may include node information of at least one key node to be checked. The at least one processor may be directed to cause the system to: determining whether the node information of the at least one key node matches one of the one or more permissions indicated by the access authority information. The at least one processor may be directed to cause the system to granting the request for access to the object upon determination that the match exists.

[0016] According to another aspect of the present disclosure, a non-transitory computer readable medium is provided. The non-transitory computer readable medium may include at least one set of instructions for controlling a user's access to an object, wherein when executed by one or more processors of a computing device, the at least one set of instructions may cause the computing device to perform a method. The method may include obtaining a request for access to the object from the user. The method may also include determining access authority information of the user based on a mapping relation between one or more predefined access permissions and one or more users. The access authority information of the user may indicate one or more permissions. The method may further include performing an access check to resolve the request. The one or more permissions may be included in a plurality of permissions that are stored in a storage of the one or more

storage device as a data structure. The data structure may include at least one directed acyclic graph (DAG) structure. Each DAG structure may include a plurality of nodes and a plurality of directed edges. Each node may represent a permission. Each directed edge may include a direction from a parent node to a child node among the plurality of nodes. The request may include node information of at least one key node to be checked. The method may further include determining whether the node information of the at least one key node matches one of the one or more permissions indicated by the access authority information. The method may still further include granting the request for access to the object upon determination that the match exists.

[0017] According to another aspect of the present disclosure, a method for storing permission information is provided. The method may be implemented on a computing device having one or more processors and one or more storage devices. The method may include storing the permission information as a data structure. The data structure may include at least one directed acyclic graph (DAG) structure. The method may also include generating a plurality of nodes of the at least one DAG structure. Each node may represent a permission to access to an object. The method may further include generating a plurality of directed edges of the at least one DAG structure, each directed edge including a direction from a parent node to one or more child nodes of the parent node among the plurality of nodes. The method may still further include associating a user with one or more access authorities denoted by one or more nodes of the plurality of nodes.

[0018] In some embodiments, the method may also include associating one or more roles with the one or more nodes of the plurality of nodes. Any role of the one or more roles associated with the parent node may have one or more permissions corresponding to the one or more child nodes of the parent node. The method may further include associating the user with at least one role of the one or more roles. The user may have a permission corresponding to the at least one role.

[0019] In some embodiments, the method may also include adding, deleting or modifying one or more nodes of the plurality of nodes in the data structure.

[0020] In some embodiments, the method may also include selecting the first node for deletion from the data structure. The method may further include revoking an association between the first node and a corresponding role. The method may still further include revoking associations between all child nodes of the first node and corresponding roles. The method may still further include deleting the first node and all child nodes of the first node.

[0021] In some embodiments, the method may also include selecting the first node for deletion from the data structure. The method may further include revoking an association between the first node and a corresponding role. The method may further include selecting at least one child node of all child nodes of the first node for deletion from the data structure. The method may further include revoking an association between the at least one child node and at least one corresponding role; deleting the first node and the at least one child node of the first node. The method may further include associating the rest of child nodes of the first node with at least one node of the data structure.

[0022] In some embodiments, the method may also include selecting the second node for modification from the data structure. The second node may have a first parent node. The method may further include associating the second node with a second parent node. A user associated with a role of the one or more roles corresponding to the second node may have the permission corresponding to the second node. A user associated with a role of the one or more roles corresponding to the first parent node may have the permission corresponding to the second node. A user associated with a role of the one or more roles corresponding to the second parent node may have the permission corresponding to the second node.

[0023] In some embodiments, the method may also include revoking an association between the second node and the first parent node. The permission corresponding to the second node may be revoked from the user associated with the role of the one or more roles corresponding to the first parent node.

[0024] In some embodiments, the method may also include associating the third node with at least one parent node and/or at least one child node among the plurality

of nodes. The method may further include associating the third node with a permission; and associating the third node with a role.

[0025] According to another aspect of the present disclosure, a system for storing permission information is provided. The system may include at least one computer-readable storage medium storing a set of instructions; and at least one processor configured to communicate with the at least one computer-readable storage medium, wherein when executing the set of instructions, the at least one processor may be directed to cause the system to store the permission information as a data structure. The data structure may include at least one directed acyclic graph (DAG) structure. The at least one processor may be directed to cause the system to generate a plurality of nodes of the at least one DAG structure. Each node may represent a permission to access to an object. The at least one processor may be directed to cause the system to generate a plurality of directed edges of the at least one DAG structure. Each directed edge may include a direction from a parent node to one or more child nodes of the parent node among the plurality of nodes. The at least one processor may be directed to cause the system to associate a user with one or more access authorities denoted by one or more nodes of the plurality of nodes.

[0026] According to another aspect of the present disclosure, a non-transitory computer readable medium is provided. The non-transitory computer readable medium may include at least one set of instructions for storing permission information, wherein when executed by one or more processors of a computing device, the at least one set of instructions causes the computing device to perform a method. The method may include storing the permission information as a data structure. The data structure may include at least one directed acyclic graph (DAG) structure. The method may also include generating a plurality of nodes of the at least one DAG structure, each node representing a permission to access to an object. The method may further include generating a plurality of directed edges of the at least one DAG structure. Each directed edge may include a direction from a parent node to one or more child nodes of the parent node among the plurality of nodes. The method may still further include associating a user with one or more

access authorities denoted by one or more nodes of the plurality of nodes.

[0027] Additional features will be set forth in part in the description which follows, and in part will become apparent to those skilled in the art upon examination of the following and the accompanying drawings or may be learned by production or operation of the examples. The features of the present disclosure may be realized and attained by practice or use of various aspects of the methodologies, instrumentalities and combinations set forth in the detailed examples discussed below.

BRIEF DESCRIPTION OF THE DRAWINGS

[0028] The present disclosure is further described in terms of exemplary embodiments. These exemplary embodiments are described in detail with reference to the drawings. The drawings are not to scale. These embodiments are non-limiting exemplary embodiments, in which like reference numerals represent similar structures throughout the several views of the drawings, and wherein:

[0029] FIG. 1 is a schematic diagram illustrating an exemplary permission management system according to some embodiments of the present disclosure;

[0030] FIG. 2 is a schematic diagram illustrating exemplary components of a computing device according to some embodiments of the present disclosure;

[0031] FIG. 3 is a schematic diagram illustrating exemplary hardware and/or software components of an exemplary mobile device according to some embodiments of the present disclosure;

[0032] FIG. 4 is a block diagram illustrating an exemplary processing engine according to some embodiments of the present disclosure;

[0033] FIG. 5 is a flowchart illustrating an exemplary process for controlling a user's access to an object according to some embodiments of the present disclosure;

[0034] FIG. 6 is a flowchart illustrating an exemplary process for performing an access check to resolve a request according to some embodiments of the present disclosure;

[0035] FIG. 7 is a flowchart illustrating an exemplary process for determining

whether node information of at least one key node matches a permission indicated by access authority information according to some embodiments of the present disclosure;

[0036] FIG. 8 is a flowchart illustrating an exemplary process for storing permission information according to some embodiments of the present disclosure;

[0037] FIG. 9 is a flowchart illustrating an exemplary process for deleting a node in a data structure according to some embodiments of the present disclosure;

[0038] FIG. 10 is a flowchart illustrating an exemplary process for modifying a node in a data structure according to some embodiments of the present disclosure;

[0039] FIG. 11A is a schematic diagram illustrating an exemplary data structure according to some embodiments of the present disclosure;

[0040] FIG. 11B is a schematic diagram illustrating an exemplary data structure according to some embodiments of the present disclosure;

[0041] FIG. 12A is a schematic diagram illustrating an exemplary node modification in a data structure according to some embodiments of the present disclosure;

[0042] FIG. 12B is a schematic diagram illustrating an exemplary node modification in a data structure according to some embodiments of the present disclosure; and

[0043] FIG. 13 is a schematic diagram illustrating an exemplary process for controlling a user's access to an object according to some embodiments of the present disclosure.

DETAILED DESCRIPTION

[0044] In order to illustrate the technical solutions related to the embodiments of the present disclosure, brief introduction of the drawings referred to in the description of the embodiments is provided below. Obviously, drawings described below are only some examples or embodiments of the present disclosure. Those having ordinary skills in the art, without further creative efforts, may apply the present disclosure to other similar scenarios according to these drawings. Unless stated otherwise or obvious from the context, the same reference numeral in the drawings refers to the

same structure and operation.

[0045] As used in the disclosure and the appended claims, the singular forms “a,” “an,” and “the” include plural referents unless the content clearly dictates otherwise. It will be further understood that the terms “comprises,” “comprising,” “includes,” and/or “including” when used in the disclosure, specify the presence of stated steps and elements, but do not preclude the presence or addition of one or more other steps and elements.

[0046] Some modules of the system may be referred to in various ways according to some embodiments of the present disclosure, however, any number of different modules may be used and operated in a client terminal and/or a server. These modules are intended to be illustrative, not intended to limit the scope of the present disclosure. Different modules may be used in different aspects of the system and method.

[0047] According to some embodiments of the present disclosure, flowcharts are used to illustrate the operations performed by the system. It is to be expressly understood, the operations above or below may or may not be implemented in order. Conversely, the operations may be performed in inverted order, or simultaneously. Besides, one or more other operations may be added to the flowcharts, or one or more operations may be omitted from the flowchart.

[0048] Technical solutions of the embodiments of the present disclosure be described with reference to the drawings as described below. It is obvious that the described embodiments are not exhaustive and are not limiting. Other embodiments obtained, based on the embodiments set forth in the present disclosure, by those with ordinary skill in the art without any creative works are within the scope of the present disclosure.

[0049] Moreover, the systems and methods in the present disclosure may be applied to any application scenario in which permission management and/or access control is required. For example, the system or method of the present disclosure may be applied to different transportation systems including land, ocean, aerospace, or the like, or any combination thereof. The transportation systems may provide

transportation service for users using various vehicles. The vehicles of the transportation service may include a taxi, a private car, a hitch, a bus, a train, a bullet train, a high speed rail, a subway, a vessel, an aircraft, a spaceship, a hot-air balloon, a driverless vehicle, a bicycle, a tricycle, a motorcycle, or the like, or any combination thereof. The system or method of the present disclosure may be applied to a taxi hailing service, a chauffeur service, a delivery service, a carpooling service, a bus service, a take-out service, a driver hiring service, a shuttle service, a travel service, or the like, or any combination thereof. As another example, the system or method of the present disclosure may be applied to a navigation service, a shopping service, a house service, a location based service (LBS), or the like, or any combination thereof. The application scenarios of the system or method of the present disclosure may include a web page, a plug-in of a browser, a client terminal, a custom system, an internal analysis system, an artificial intelligence robot, or the like, or any combination thereof.

[0050] An aspect of the present disclosure is directed to systems and methods for controlling a user's access to an object. The systems and methods may obtain a request for access to the object from the user. The request may include node information of at least one key node to be checked. The systems and methods may determine access authority information of the user based on a mapping relation between one or more predefined access permissions and one or more users. The access authority information of the user may indicate one or more permissions. The one or more permissions may be included in a plurality of permissions that are stored in a storage as a data structure. The data structure may include at least one directed acyclic graph (DAG) structure. Each DAG structure may include a plurality of nodes and a plurality of directed edges. Each node may represent a permission. Each directed edge may include a direction from a parent node to a child node of the parent node. The systems and methods may further perform an access check to resolve the request. For example, the systems and methods may determine whether the node information of the at least one key node matches one of the one or more permissions indicated by the access authority information. The systems and

methods may grant the request for access to the object in response to a determination that the match exists.

[0051] Another aspect of the present disclosure is directed to systems and methods for storing permission information. The systems and methods may store the permission information as a data structure. The data structure may include at least one directed acyclic graph (DAG) structure. To determine the data structure, the systems and methods may generate a plurality of nodes of the at least one DAG structure. Each node may represent a permission to access to an object. The systems and methods may also generate a plurality of directed edges of the at least one DAG structure. Each directed edge may include a direction from a parent node to one or more child nodes of the parent node among the plurality of nodes. The systems and methods may further associate a user with one or more access authorities denoted by one or more nodes of the plurality of nodes.

[0052] Accordingly, the systems and methods may store permission information as a data structure (e.g., one or more DAG structures) to achieve multi-level permission inheritance, and control a user's access to the object based on the permission information more efficiently.

[0053] FIG. 1 is a schematic diagram illustrating an exemplary permission management system according to some embodiments of the present disclosure. The permission management system 100 may include a server 110, a network 120, a storage device 130, and/or a client terminal 140.

[0054] The server 110 may perform permission storage, permission processing and/or permission control for the permission management system 100. In some embodiments, the server 110 may be a single server or a server group. The server group may be centralized, or distributed (e.g., server 110 may be a distributed system). In some embodiments, the server 110 may be local or remote. For example, the server 110 may access information and/or data stored in the client terminal 140, and/or the storage device 130 via the network 120. As another example, the server 110 may be directly connected to the client terminal 140, and/or the storage device 130 to access stored information and/or data. In some

embodiments, the server 110 may be implemented on a cloud platform. Merely by way of example, the cloud platform may include a private cloud, a public cloud, a hybrid cloud, a community cloud, a distributed cloud, an inter-cloud, a multi-cloud, or the like, or any combination thereof. In some embodiments, the server 110 may be implemented on a computing device 200 having one or more components illustrated in FIG. 2 in the present disclosure or a mobile device 300 having one or more components illustrated in FIG. 3 in the present disclosure.

[0055] In some embodiments, the server 110 may include a processing engine 112. The processing engine 112 may process information and/or data to perform one or more functions described in the present disclosure. For example, the processing engine 112 may obtain a request for access to an object from a user. As another example, the processing engine 112 may verify node information of at least one key node included in a request. As still another example, the processing engine 112 may determine access authority information of a user based on a mapping relation between one or more predefined access permissions and one or more users. As still another example, the processing engine 112 may perform an access check to resolve the request. As still another example, the processing engine 112 may store permission information as a data structure. As still another example, the processing engine 112 may associate a user with one or more access authorities denoted by one or more nodes. As still another example, the processing engine 112 may associate one or more roles with one or more nodes. As still another example, the processing engine 112 may associate a user with at least one role of one or more roles associated with one or more nodes. As still another example, the processing engine 112 may add, delete or modify one or more nodes of a plurality of nodes in a data structure.

[0056] In some embodiments, the processing engine 112 may include one or more processing engines (e.g., single-core processing engine(s) or multi-core processor(s)). Merely by way of example, the processing engine 112 may include one or more hardware processors, such as a central processing unit (CPU), an application-specific integrated circuit (ASIC), an application-specific instruction-set

processor (ASIP), a graphics processing unit (GPU), a physics processing unit (PPU), a digital signal processor (DSP), a field-programmable gate array (FPGA), a programmable logic device (PLD), a controller, a microcontroller unit, a reduced instruction-set computer (RISC), a microprocessor, or the like, or any combination thereof.

[0057] The network 120 may facilitate the exchange of information and/or data. In some embodiments, one or more components in the permission management system 100 (e.g., the server 110, the storage device 130, and the client terminal 140) may send information and/or data to other component(s) in the permission management system 100 via the network 120. For example, the processing engine 112 may obtain a request for access to an object from the client terminal 140 via the network 120. As another example, the processing engine 112 may obtain a data structure from the storage device 130 via the network 120. In some embodiments, the network 120 may be any type of wired or wireless network, or a combination thereof. Merely by way of example, the network 120 may include a cable network, a wireline network, an optical fiber network, a telecommunications network, an intranet, the Internet, a local area network (LAN), a wide area network (WAN), a wireless local area network (WLAN), a metropolitan area network (MAN), a wide area network (WAN), a public telephone switched network (PSTN), a Bluetooth™ network, a ZigBee network, a near field communication (NFC) network, or the like, or any combination thereof. In some embodiments, the network 120 may include one or more network access points. For example, the network 120 may include wired or wireless network access points such as base stations and/or internet exchange points 120-1, 120-2, ..., through which one or more components of the permission management system 100 may be connected to the network 120 to exchange data and/or information.

[0058] The storage device 130 may store data and/or instructions. In some embodiments, the storage device 130 may store data obtained from the client terminal 140 and/or the processing engine 112. For example, the storage device 130 may store a request for access to an object received from the client terminal

140. As another example, the storage device 130 may store a data structure including permission information determined by the processing engine 112. In some embodiments, the storage device 130 may store data and/or instructions that the server 110 may execute or use to perform exemplary methods described in the present disclosure. For example, the storage device 130 may store instructions that the processing engine 112 may execute or use to verify node information of at least one key node included in a request. As another example, the storage device 130 may store instructions that the processing engine 112 may execute or use to determine access authority information of a user based on a mapping relation between one or more predefined access permissions and one or more users. As still another example, the storage device 130 may store instructions that the processing engine 112 may execute or use to perform an access check to resolve a request. As still another example, the storage device 130 may store instructions that the processing engine 112 may execute or use to store permission information as a data structure. As still another example, the storage device 130 may store instructions that the processing engine 112 may execute or use to add, delete or modify one or more nodes of a plurality of nodes in a data structure.

[0059] In some embodiments, the storage device 130 may include a mass storage, a removable storage, a volatile read-and-write memory, a read-only memory (ROM), or the like, or any combination thereof. Exemplary mass storage may include a magnetic disk, an optical disk, a solid-state drive, etc. Exemplary removable storage may include a flash drive, a floppy disk, an optical disk, a memory card, a zip disk, a magnetic tape, etc. Exemplary volatile read-and-write memory may include a random access memory (RAM). Exemplary RAM may include a dynamic RAM (DRAM), a double date rate synchronous dynamic RAM (DDR SDRAM), a static RAM (SRAM), a thyrisor RAM (T-RAM), and a zero-capacitor RAM (Z-RAM), etc. Exemplary ROM may include a mask ROM (MROM), a programmable ROM (PROM), an erasable programmable ROM (EPROM), an electrically-erasable programmable ROM (EEPROM), a compact disk ROM (CD-ROM), and a digital versatile disk ROM, etc. In some embodiments, the storage device 130 may be

implemented on a cloud platform. Merely by way of example, the cloud platform may include a private cloud, a public cloud, a hybrid cloud, a community cloud, a distributed cloud, an inter-cloud, a multi-cloud, or the like, or any combination thereof.

[0060] In some embodiments, the storage device 130 may be connected to the network 120 to communicate with one or more components in the permission management system 100 (e.g., the server 110, the client terminal 140). One or more components in the permission management system 100 may access the data or instructions stored in the storage device 130 via the network 120. In some embodiments, the storage device 130 may be directly connected to or communicate with one or more components in the permission management system 100 (e.g., the server 110, the client terminal 140). In some embodiments, the storage device 130 may be part of the server 110.

[0061] In some embodiments, the client terminal 140 may include a mobile device 140-1, a tablet computer 140-2, a laptop computer 140-3, or the like, or any combination thereof. In some embodiments, the mobile device 140-1 may include a smart home device, a wearable device, a mobile equipment, a virtual reality device, an augmented reality device, or the like, or any combination thereof. In some embodiments, the smart home device may include a smart lighting device, a control device of an intelligent electrical apparatus, a smart monitoring device, a smart television, a smart video camera, an interphone, or the like, or any combination thereof. In some embodiments, the wearable device may include a bracelet, footgear, glasses, a helmet, a watch, clothing, a backpack, a smart accessory, or the like, or any combination thereof. In some embodiments, the mobile equipment may include a mobile phone, a personal digital assistance (PDA), a gaming device, a navigation device, a point of sale (POS) device, a laptop, a desktop, or the like, or any combination thereof. In some embodiments, the virtual reality device and/or the augmented reality device may include a virtual reality helmet, a virtual reality glass, a virtual reality patch, an augmented reality helmet, augmented reality glasses, an augmented reality patch, or the like, or any combination thereof. For example, the

virtual reality device and/or the augmented reality device may include a Google Glass™, a RiftCon™, a Fragments™, a Gear VR™, etc.

[0062] It should be noted that the permission management system 100 is merely provided for the purposes of illustration, and is not intended to limit the scope of the present disclosure. For persons having ordinary skills in the art, multiple variations or modifications may be made under the teachings of the present disclosure. For example, the permission management system 100 may further include a database, an information source, or the like. As another example, the permission management system 100 may be implemented on other devices to realize similar or different functions. However, those variations and modifications do not depart from the scope of the present disclosure. In some embodiments, two or more components of the permission management system 100 may be integrated into a single component. For example, the storage device 130 may be integrated into the client terminal 140 as indicated by the bi-directional arrow in dotted lines linking the storage device 130 and the client terminal 140. As another example, the server 110 may be integrated into the client terminal 140 as indicated by the bi-directional arrow in dotted lines linking the server 110 and the client terminal 140. As still another example, the server 110, the storage 130, and the client terminal 140 may be integrated into a single component as indicated by the bi-directional arrow in dotted lines linking the storage device 130 and the client terminal 140, the bi-directional arrow in dotted lines linking the server 110 and the client terminal 140, and the bi-directional arrow in dotted lines linking the server 110 and the storage device 130. In some embodiments, one or more components may be omitted. For example, the network 120 may be omitted. As another example, the server 110 may be omitted, and the client terminal 140 may implement one or more functions of the server 110.

[0063] FIG. 2 is a schematic diagram illustrating exemplary components of a computing device on which the server 110, the storage device 130, and/or the client terminal 140 may be implemented according to some embodiments of the present disclosure. A particular system (e.g., the permission management system 100) may use a functional block diagram to explain the hardware platform containing one or

more user interfaces. The computer may be a computer with general or specific functions. Both types of the computers may be configured to implement any particular system (e.g., the permission management system 100) according to some embodiments of the present disclosure. Computing device 200 may be configured to implement any components that perform one or more functions disclosed in the present disclosure. For example, the computing device 200 may implement any component of the permission management system 100 as described herein. In FIGs. 1-2, only one such computer device is shown purely for convenience purposes. One of ordinary skill in the art would understood at the time of filing of this application that the computer functions relating to permission management as described herein may be implemented in a distributed fashion on a number of similar platforms, to distribute the processing load.

[0064] The computing device 200, for example, may include COM ports 250 connected to and from a network connected thereto to facilitate data communications. The computing device 200 may also include a processor (e.g., the processor 220), in the form of one or more processors (e.g., logic circuits), for executing program instructions. For example, the processor may include interface circuits and processing circuits therein. The interface circuits may be configured to receive electronic signals from a bus 210, wherein the electronic signals encode structured data and/or instructions for the processing circuits to process. The processing circuits may conduct logic calculations, and then determine a conclusion, a result, and/or an instruction encoded as electronic signals. Then the interface circuits may send out the electronic signals from the processing circuits via the bus 210.

[0065] The exemplary computing device may include the internal communication bus 210, program storage and data storage of different forms including, for example, a disk 270, and a read only memory (ROM) 230, or a random access memory (RAM) 240, for various data files to be processed and/or transmitted by the computing device. The exemplary computing device may also include program instructions stored in the ROM 230, RAM 240, and/or other type of non-transitory storage

medium to be executed by the processor 220. The methods and/or processes of the present disclosure may be implemented as the program instructions. The computing device 200 also includes an I/O component 260, supporting input/output between the computer and other components. The computing device 200 may also receive programming and data via network communications.

[0066] Merely for illustration, only one CPU and/or processor is illustrated in FIG. 2. Multiple CPUs and/or processors are also contemplated; thus operations and/or method steps performed by one CPU and/or processor as described in the present disclosure may also be jointly or separately performed by the multiple CPUs and/or processors. For example, if in the present disclosure the CPU and/or processor of the computing device 200 executes both step A and step B, it should be understood that step A and step B may also be performed by two different CPUs and/or processors jointly or separately in the computing device 200 (e.g., the first processor executes step A and the second processor executes step B, or the first and second processors jointly execute steps A and B).

[0067] FIG. 3 is a schematic diagram illustrating exemplary hardware and/or software components of an exemplary mobile device according to some embodiments of the present disclosure; on which the client terminal 140 may be implemented according to some embodiments of the present disclosure. As illustrated in FIG. 3, the mobile device 300 may include a communication platform 310, a display 320, a graphic processing unit (GPU) 330, a central processing unit (CPU) 340, an I/O 350, a memory 360, and a storage 390. The CPU 340 may include interface circuits and processing circuits similar to the processor 220. In some embodiments, any other suitable component, including but not limited to a system bus or a controller (not shown), may also be included in the mobile device 300. In some embodiments, a mobile operating system 370 (e.g., iOS™, Android™, Windows Phone™, etc.) and one or more applications 380 may be loaded into the memory 360 from the storage 390 in order to be executed by the CPU 340. The applications 380 may include a browser or any other suitable mobile apps for receiving and rendering information relating to a request or other information from

the permission management system on the mobile device 300. User interactions with the information stream may be achieved via the I/O devices 350 and provided to the processing engine 112 and/or other components of the permission management system 100 via the network 120.

[0068] In order to implement various modules, units and their functions described above, a computer hardware platform may be used as hardware platforms of one or more elements (e.g., a component of the sever 110 described in FIG. 2). Since these hardware elements, operating systems, and program languages are common, it may be assumed that persons skilled in the art may be familiar with these techniques and they may be able to provide information required in permission management according to the techniques described in the present disclosure. A computer with user interface may be used as a personal computer (PC), or other types of workstations or terminal devices. After being properly programmed, a computer with user interface may be used as a server. It may be considered that those skilled in the art may also be familiar with such structures, programs, or general operations of this type of computer device. Thus, extra explanations are not described for the figures.

[0069] FIG. 4 is a schematic diagram illustrating an exemplary processing engine according to some embodiments of the present disclosure. In some embodiments, the processing engine 112 may include an obtaining module 410, a verification module 420, an authority determination module 430, an access check module 440, a data structure determination module 450, an association module 460, and a modification module 470. The modules may be hardware circuits of at least part of the processing engine 112. The modules may also be implemented as an application or set of instructions read and executed by the processing engine 112. Further, the modules may be any combination of the hardware circuits and the application/instructions. For example, the modules may be part of the processing engine 112 when the processing engine 112 is executing the application or set of instructions.

[0070] The obtaining module 410 may be configured to obtain data and/or

information related to the permission management system 100. In some embodiments, the obtaining module 410 may obtain a request for access to an object from a user. In some embodiments, the object may include a resource. Accordingly, the request may be a request for access to a specific resource (e.g., a request for querying the resource, obtaining the resource, viewing the resource, or the like). In some embodiments, the object may include an operation on a resource. Accordingly, the request may be a request for performing an operation (e.g., reading, writing, publishing, subscribing, editing, adding, deleting, modifying, updating, or the like) on a resource. In some embodiments, the request may further include the user's identity information (e.g., an identification (ID), a telephone number, a name, etc.).

[0071] In some embodiments, the obtaining module 410 may obtain the data and/or information related to the permission management system 100 from a user terminal (e.g., the client terminal 140), the storage device 130, and/or an external data source (not shown). In some embodiments, the obtaining module 410 may obtain the data and/or information related to the permission management system 100 via the network 120.

[0072] The verification module 420 may be configured to verify node information of at least one key node included in a request. In some embodiments, the verification module 420 may verify the node information of the at least one key node based on a plurality of nodes in one or more DAG structures. More descriptions of a DAG structure and the key node may be found elsewhere in the present disclosure (e.g., FIG. 5 and the descriptions thereof). For example, the verification module 420 may determine whether the node information of the at least one key node is included in the one or more DAG structures. In response to a determination that the node information of the at least one key node is included in the one or more DAG structures, the verification module 420 may determine that the node information of the at least one key node included in the request is verified. In response to a determination that the node information of the at least one key nodes is not included in the one or more DAG structures, the verification module 420 may determine that

the node information of the at least one key node included in the request is not verified, and the request may be refused.

[0073] The authority determination module 430 may be configured to determine access authority information of a user. In some embodiments, the authority determination module 430 may determine access authority information of the user based on a mapping relation between one or more predefined access permissions and one or more users. In some embodiments, the mapping relation between one or more predefined access permissions and one or more users may be preset manually by a user (e.g., an administrator) of the permission management system 100, or may be determined by one or more components (e.g., the processing engine 112) of the permission management system 100 according to different situations. In some embodiments, the mapping relation between one or more predefined access permissions and one or more users may be stored in a storage device (e.g., the storage device 130) of the permission management system 100 or an external storage device, and the processing engine 112 may access the storage device and retrieve the mapping relation. In some embodiments, the authority determination module 430 may determine the access authority information of the user based on one or more roles of the user. More descriptions of the determination of the access authority information of the user may be found elsewhere in the present disclosure (e.g., FIG. 5 and the descriptions thereof).

[0074] The access check module 440 may be configured to perform an access check to resolve a request. In some embodiments, the access check module 440 may perform the access check by determining whether the node information of the at least one key node matches one of the one or more permissions indicated by the access authority information of the user. In response to a determination that the node information of the at least one key node matches one of the one or more permissions indicated by the access authority information, the access check module 440 may grant the request. In response to a determination that the node information of the at least one key node matches none of the one or more permissions indicated by the access authority information, the access check module

440 may refuse the request. More descriptions of the access check may be found elsewhere in the present disclosure (e.g., FIGs. 5, 6, 7 and the descriptions thereof).

[0075] The data structure determination module 450 may be configured to store permission information as a data structure. In some embodiments, the data structure determination module 450 may store permission information as a data structure in a storage device (e.g., the storage device 130) of the permission management system 100. In some embodiments, the data structure may include one or more directed acyclic graph (DAG) structures as described elsewhere in the present disclosure (e.g., FIGs. 5, 11A-12B and the descriptions thereof). In some embodiments, the data structure determination module 450 may generate a plurality of nodes and/or a plurality of directed edges of the DAG structure(s). More descriptions of the generation of a plurality of nodes and a plurality of directed edges of a DAG structure may be found elsewhere in the present disclosure (e.g., FIGs. 5, 8 and the descriptions thereof).

[0076] The association module 460 may be configured to determination an association between data and/or information related to the permission management system 100. In some embodiments, the association module 460 may associate a user with one or more access authorities denoted by one or more nodes of a data structure. In some embodiments, the association module 460 may associate one or more roles with one or more nodes of a data structure. In some embodiments, the association module 460 may associate a user with one or more roles. More descriptions of the associations may be found elsewhere in the present disclosure (e.g., FIG. 8 and the descriptions thereof). In some embodiments, the associations may be stored in a storage device (e.g., the storage device 130) of the permission management system 100, and the association module 460 may access the storage device and retrieve the associations in the management and/or control of the permissions.

[0077] The modification module 470 may be configured to modify data and/or information related to a data structure. In some embodiments, the modification module 470 may add one or more nodes in a data structure. In some

embodiments, the modification module 470 may delete one or more nodes in a data structure. In some embodiments, the modification module 470 may modify one or more nodes in a data structure. More descriptions for adding, deleting, and modifying one or more nodes in a data structure may be found elsewhere in the present disclosure (e.g., FIGs. 8, 9, 10 and the descriptions thereof).

[0078] It should be noted that the above description of the processing engine 112 is merely provided for the purposes of illustration, and not intended to limit the scope of the present disclosure. For persons having ordinary skills in the art, multiple variations and modifications may be made under the teachings of the present disclosure. However, those variations and modifications do not depart from the scope of the present disclosure. For example, the data structure determination module 450, the association module 460, and/or the modification module 470 may be omitted. In some embodiments, the processing engine 112 may be divided into two or more engines. For example, the processing engine 112 may be divided into an controlling engine and a storing engine, in which the controlling engine may include the obtaining module 410, the verification module 420, the authority determination module 430 and the access check module 440, and the storing engine may include the data structure determination module 450, the association module 460 and the modification module 470.

[0079] FIG. 5 is a flowchart illustrating an exemplary process for controlling a user's access to an object according to some embodiments of the present disclosure. In some embodiments, the process 500 may be implemented in the permission management system 100. For example, the process 500 may be stored in the storage device 130 and/or the storage (e.g., the ROM 230, the RAM 240, etc.) as a form of instructions, and invoked and/or executed by the server 110 (e.g., the processing engine 112 in the server 110, or the processor 210 of the processing engine 112 in the server 110).

[0080] In 510, the processing engine 112 (e.g., the obtaining module 410) may obtain a request for access to an object from a user. In some embodiments, the processing engine 112 may obtain the request from the client terminal 140 via the

network 120.

[0081] In some embodiments, the client terminal 140 may establish a communication (e.g., wireless communication) with the server 110, through an application (e.g., the application 380 in FIG. 3) installed in the client terminal 140 or a webpage in a browser via the network 120. In some embodiments, the application may be associated with a service (e.g., an online service). For example, the application may be associated with a taxi-hailing service, a carpooling service, a hitch service, a delivery service, an online shopping service, or the like. In some embodiments, the application may be associated with an enterprise information management tool. In some embodiments, the user may send the request to the processing engine 112 (e.g., the obtaining module 410) by, for example, selecting or confirming one or more options (e.g., action button(s), function menu(s), or the like) on an interface of the application. In some embodiments, the user may select or confirm the option(s) on the interface of the application through a touch screen of the client terminal 140. In some embodiments, the application installed in the client terminal 140 may direct the client terminal 140 to monitor, continuously or periodically, the request from the user, and automatically transmit the request to the processing engine 112 via the network 120.

[0082] The request may be a request for access to the object. In some embodiments, the object may include a resource. Accordingly, the request may be a request for access to a specific resource (e.g., a request for querying the resource, obtaining the resource, viewing the resource, or the like). For example, the request may be a request for viewing orders initiated in Beijing in a taxi-hailing service. As used herein, a resource may refer to information resources (e.g., database(s), file(s), data) or operational resources (e.g., program(s), process(es)) stored in one or more storage devices of the permission management system 100 or external data source(s). In some embodiments, the information resources may include information or data related to the management and/or operation of enterprise(s), for example, an order quantity, a revenue, a working report, staff information, or the like. In some embodiments, the object may include an operation on a resource.

Accordingly, the request may be a request for performing an operation (e.g., reading, writing, publishing, subscribing, editing, adding, deleting, modifying, updating, or the like) on a resource. For example, the user may initiate a request for modifying an order in a taxi-hailing service. In some embodiments, the request may further include the user's identity information (e.g., an identification (ID), a telephone number, a name, etc.).

[0083] In some embodiments, the user may have a permission (or authorization) for access to the object. In some embodiments, the user may have no permission (or authorization) for access to the object. In response to the request, the processing engine 112 may check whether the user have the permission (or authorization), as illustrated in operations 520-540. In response to a determination that the user has the permission, the access to the object may be allowed. In response to a determination that the user has no permission, the access to the object may be refused (e.g., the permission management system 100 may provide a prompt message notifying the user he/she has no permission).

[0084] In 520, the processing engine 112 (e.g., the verification module 420) may verify node information of at least one key node included in the request.

[0085] In some embodiments, the processing engine 112 may store a plurality of permissions as a data structure in one or more storage devices (e.g., the storage device 130) of the permission management system 100 as described elsewhere in the present disclosure (e.g., FIG. 8 and the descriptions thereof). In some embodiments, the permission(s) may include a data scope permission, and/or a functional operation permission. The data scope permission may be associated with data access rights and/or scopes of data access (also referred to as permission to data scopes). In some embodiments, the data scope permission may indicate whether a user is allowed to access a resource of a system (e.g., the permission management system 100). The functional operation permission may be associated with one or more data operation functions. In some embodiments, the functional operation permission may indicate whether a user is allowed to operate (e.g., read, write, publish, subscribe, edit, add, delete, modify, or update) data of the system. In

some embodiments, the plurality of permissions stored in the data structure may include a permission of granting a permission of the user to one or more other users, a permission of creating a permission for one or more other users, a permission of inquiring a permission of one or more users, a permission of modifying a permission of one or more users, and/or a permission of revoking a permission of one or more users. For example, user A may have a permission of granting and/or creating a permission of cancelling an order to user B in a taxi-hailing service.

[0086] In some embodiments, the data structure may include one or more directed acyclic graph (DAG) structures. As used herein, a DAG may refer to a directed graph with no directed cycles. The DAG structure may include a plurality of nodes and a plurality of directed edges (see FIGs. 11A-12B). The plurality of nodes in the DAG structure may be associated with one another through one or more child/parent relationships. The child/parent relationships between the nodes may be represented by the plurality of directed edges. Each directed edge may include a direction from a parent node to a child node of the parent node. For example, if a node A is connected to a node B via a directed edge A→B, then the node A is regarded as a parent node of the node B, and the node B is regarded as a child node of the node A. Each node of the plurality of nodes may have zero, one or more parent nodes. For example, as illustrated in FIG. 11A, node 1101a may have zero parent node. Node 1101b, node 1101c, node 1101d, node 1101e, and/or node 1101f may have one parent node. If node 1101c and node 1101e may further be connected via a directed edge directing from node 1101c to node 1101e, then node 1101e may have two parent nodes (e.g., node 1101b and node 1101c). Each node of the plurality of nodes may have zero, one or more child nodes. For example, node 1101d, node 1101e, and node 1101f may have zero child node. Node 1101c may have one child node. Node 1101a and/or node 1101b may have two child nodes. If a node has no parent node (e.g., node 1101a in FIG. 11A, node 1103a in FIG. 11B), that is, the node is the topmost node in the DAG structure, then the node may be also referred to as a root node in the DAG structure.

[0087] In some embodiments, each node may represent a permission. The DAG

structure allows permission inheritance among the plurality of permissions. In some embodiments, the parent node may have one or more permissions corresponding to one or more child nodes of the parent node. For example, assuming that user A has a permission corresponding to a parent node, user A has one or more permissions corresponding to one or more child nodes of the parent node. Specifically, as illustrated in FIG. 11A, if user A has a permission corresponding to node 1101b (e.g., manage order), and node 1101b has two child nodes (e.g., node 1101d, node 1101e), then user A may also have one or more permissions corresponding to node 1101d (e.g., cancel order) and node 1101e (e.g., modify order) node 1101b. In some embodiments, the DAG structure in the present disclosure may provide multi-level permission inheritance relationships. The multi-level permission inheritance may refer to that the plurality of permissions denoted by the plurality of nodes may be designed as multiple levels. A node in a higher level may have one or more permissions corresponding to one or more nodes in a lower level. For example, as illustrated in FIG. 11A, the DAG structure may have three levels. Node 1101a may be in a first level of the data structure. Node 1101b and node 1101c may be in a second level of the data structure. Node 1101d, node 1101e, and node 1101f may be in a third level of the data structure. If user A has a permission corresponding to node 1101a in the first level, then user A may have permissions corresponding to node 1101b and node 1101c in the second level, node 1101d, node 1101e, and node 1101f in the third level.

[0088] As illustrated above, the permission(s) may include a functional operation permission and/or a data scope permission. In some embodiments, the processing engine 112 may store a plurality of functional operation permissions and a plurality of data scope permissions as a single DAG structure. In the single DAG structure, a first portion of nodes may be associated with functional operation permissions, while a second portion of nodes may be associated with data scope permissions. In some embodiments, the processing engine 112 may store a plurality of functional operation permissions as a first DAG structure. The first DAG structure may include a plurality of first nodes associated with functional operation permissions (e.g., the

functional operation permissions shown in FIG. 11A). In some embodiments, the processing engine 112 may further store a plurality of data scope permissions as a second DAG structure. The second DAG structure may include a plurality of second nodes associated with permissions to data scopes (e.g., the data scope permissions shown in FIG. 11B).

[0089] In some embodiments, the request obtained in 510 may include node information of at least one key node (also referred to as first node information relative to second node information and third node information described below) to be checked. Each key node may correspond to a permission requested by the user and associated with a node in a DAG structure. For example, if the user initiates a request to modify an order in Beijing store of KFC, then there are two key nodes to be checked, including "modify an order" and "Beijing store".

[0090] In some embodiments, the processing engine 112 may verify the node information of the at least one key node based on the plurality of nodes in the one or more DAG structures. For example, the processing engine 112 may determine whether the node information of the at least one key node is included in the one or more DAG structures. In some embodiments, in response to a determination that the node information of the at least one key node is included in the one or more DAG structures, the processing engine 112 may determine that the node information of the at least one key node included in the request is verified, and the process 500 may proceed to operation 530. In some embodiments, in response to a determination that the node information of the at least one key nodes is not included in the one or more DAG structures, the processing engine 112 may determine that the node information of the at least one key node included in the request is not verified, and the request may be refused (accordingly, the process 500 may be terminated).

[0091] In 530, the processing engine 112 (e.g., the authority determination module 430) may determine access authority information of the user based on a mapping relation between one or more predefined access permissions and one or more users.

[0092] In some embodiments, the mapping relation between one or more predefined access permissions and one or more users may be preset manually by a user (e.g., an administrator) of the permission management system 100, or may be determined by one or more components (e.g., the processing engine 112) of the permission management system 100 according to different situations. In some embodiments, a user may have one or more predefined access permissions. In some embodiments, a predefined access permission may be authorized to one or more users. The access authority information of the user may refer to the predefined access permissions of the user.

[0093] In some embodiments, the mapping relation between one or more predefined access permissions and one or more users may be stored in a storage device (e.g., the storage device 130) of the permission management system 100 or an external storage device, and the processing engine 112 may access the storage device and retrieve the mapping relation.

[0094] In some embodiments, the processing engine 112 may determine the access authority information of the user based on one or more roles of the user. In some embodiments, a user may be assigned to (or associated with) one or more roles. In some embodiments, each role of the one or more roles may be associated with one or more permissions that are authorized to user(s) in the each role. For example, in an organization, one or more roles may be created for various job titles (e.g., a store manager, a store assistant). In some embodiments, certain permissions may be assigned to specific roles. If members or staffs (or other system users) is assigned to the specific roles, then the members or staffs may acquire the permissions associated with the specific roles. For example, if a user is assigned to a role of store manager, and one or more permissions (e.g., cancelling an order, and/or modifying an order) are associated with the role of store manager, then the user may have the permissions of cancelling an order and/or modifying an order. In some embodiments, the access authority information of the user may further include one or more roles of the user associated with the one or more permissions.

[0095] In 540, the processing engine 112 (e.g., the access check module 440) may

perform an access check to resolve the request. In some embodiments, the processing engine 112 may perform the access check by determining whether the node information of the at least one key node matches one of the one or more permissions indicated by the access authority information of the user.

[0096] For example, for each key node of the at least one key node, the processing engine 112 may traverse from the each key node to a root node of the at least one DAG structure. The processing engine 112 may determine whether one of the one or more permissions indicated by the access authority information is traversed. In response to a determination that one of the one or more permissions indicated by the access authority information is traversed, the processing engine 112 may determine that a match exists. If one or more matches between the one or more key nodes included in the request and one or more permissions indicated by the access authority information are all exist, the processing engine 112 may determine that the node information of the at least one key node matches one of the one or more permissions indicated by the access authority information. In response to a determination that none of the one or more permissions indicated by the access authority information is traversed, the processing engine 112 may determine that the match does not exist. That is, the processing engine 112 may determine that the node information of the at least one key node matches none of the one or more permissions indicated by the access authority information.

[0097] In response to a determination that the node information of the at least one key node matches one of the one or more permissions indicated by the access authority information, the processing engine 112 may grant the request. In response to a determination that the node information of the at least one key node matches none of the one or more permissions indicated by the access authority information, the processing engine 112 may refuse the request. More descriptions of the access check may be found elsewhere in the present disclosure (e.g., FIGs. 6, 7, and the descriptions thereof).

[0098] It should be noted that the above description is merely provided for the purposes of illustration, and not intended to limit the scope of the present disclosure.

For persons having ordinary skills in the art, multiple variations and modifications may be made under the teachings of the present disclosure. However, those variations and modifications do not depart from the scope of the present disclosure. In some embodiments, one or more operations may be added or omitted. For example, a storing operation may be added in process 500. In the storing operation, the processing engine 112 may store information and/or data associated with a data structure in a storage medium (e.g., the storage device 130), which is disclosed elsewhere in the present disclosure. As another example, operation 520 may be omitted. That is, in some embodiments, the processing engine 112 may perform the access check without verifying the node information of at least one key node included in the request. In some embodiments, the mapping relation between predefined access permission(s) and user(s) may include a first level mapping relation and a second level mapping relation. In some embodiments, the first level mapping relation may include an association relationship between user ID(s), role(s), and permission(s) to data scopes, and the second level mapping relation may include an association relationship between role(s) and functional operation permission(s). Additionally or alternatively, the first level mapping relation may include an association relationship between user ID(s), role(s), and functional operation permission(s), and the second level mapping relation may include an association relationship between role(s) and permission(s) to data scopes. In some embodiments, the processing engine 112 may obtain role information of the user and second node information corresponding to the access authority information of the user based on the first level mapping relation. In some embodiments, the processing engine 112 may obtain third node information corresponding to the role information. In some embodiments, the processing engine 112 may obtain the third node information corresponding to the role information based on a mapping relation (e.g., the second level mapping relation) between one or more predefined access permissions and one or more roles from role information stored in one or more storage devices of the permission management system 100 or an external storage device.

[0099] FIG. 6 is a flowchart illustrating an exemplary process for performing an access check to resolve a request according to some embodiments of the present disclosure. In some embodiments, the process 600 may be implemented in the permission management system 100. For example, the process 600 may be stored in the storage device 130 and/or the storage (e.g., the ROM 230, the RAM 240, etc.) as a form of instructions, and invoked and/or executed by the server 110 (e.g., the processing engine 112 in the server 110, or the processor 210 of the processing engine 112 in the server 110). In some embodiments, operation 540 may be performed according to one or more operations of process 600.

[0100] In 610, the processing engine 112 (e.g., the access check module 440) may determine whether the node information of the at least one key node matches a permission indicated by the access authority information (e.g., the access authority information determined in 530).

[0101] In some embodiments, the processing engine 112 may determine whether the node information of the at least one key node matches one of the one or more permissions indicated by the access authority information by traversing from each key node to a root node of at least one DAG structure. For example, the processing engine 112 may determine whether one of the one or more permissions indicated by the access authority information is traversed. In response to a determination that one of the one or more permissions indicated by the access authority information is traversed, the processing engine 112 may determine that a match exists. If one or more matches between one or more key nodes included in the request and one or more permissions indicated by the access authority information are all exist, the processing engine 112 may determine that the node information of the at least one key node matches the permission indicated by the access authority information. In response to a determination that no permission indicated by the access authority information is traversed, the processing engine 112 may determine that a match does not exit. Accordingly, the processing engine 112 may determine that the node information of the at least one key node does not match the permission indicated by the access authority information. More descriptions of determining whether the

node information of the at least one key node matches a permission indicated by the access authority information may be found elsewhere in the present disclosure (e.g., FIG. 7 and the descriptions thereof).

[0102] In response to a determination that the node information of the at least one key node matches the permission indicated by the access authority information, in 620, the processing engine 112 (e.g., the access check module 440) may grant the request. For example, if a user initiates a request to access to a resource, and the node information of the at least one key node matches a permission indicated by the access authority information of the user, then the processing engine 112 may grant the request, and accordingly, the user may be allowed to access to the resource. As another example, if the user initiates a request to perform an operation on a resource, and the node information of the at least one key node matches a permission indicated by the access authority information of the user, then the processing engine 112 may grant the request, and accordingly, the user may be allowed to perform the operation on the resource.

[0103] In response to a determination that the node information of the at least one key node does not match the permission indicated by the access authority information, in 630, the processing engine 112 (e.g., the access check module 440) may refuse the request. For example, if the user initiates a request to access to a resource, and the node information of the at least one key node does not match the permission indicated by the access authority information, then the processing engine 112 may refuse the request, and accordingly, the user may not be allowed to access to the resource. As another example, if the user initiates a request to perform an operation on a resource, and the node information of the at least one key node does not match the permission indicated by the access authority information, then the processing engine 112 may refuse the request, and accordingly, the user may not be allowed perform the operation on the resource.

[0104] It should be noted that the above description is merely provided for the purposes of illustration, and not intended to limit the scope of the present disclosure. For persons having ordinary skills in the art, multiple variations and modifications

may be made under the teachings of the present disclosure. However, those variations and modifications do not depart from the scope of the present disclosure.

[0105] FIG. 7 is a flowchart illustrating an exemplary process for determining whether node information of at least one key node matches a permission indicated by access authority information according to some embodiments of the present disclosure. In some embodiments, the process 700 may be implemented in the permission management system 100. For example, the process 700 may be stored in the storage device 130 and/or the storage (e.g., the ROM 230, the RAM 240, etc.) as a form of instructions, and invoked and/or executed by the server 110 (e.g., the processing engine 112 in the server 110, or the processor 210 of the processing engine 112 in the server 110). In some embodiments, operation 610 may be performed according to one or more operations of process 700.

[0106] In 710, the processing engine 112 (e.g., the access check module 440) may traverse from each key node to a root node of at least one DAG structure.

[0107] In some embodiments, for each key node, the processing engine 112 may traverse from the key node to one or more root nodes of the one or more DAG structures including the key node. In some embodiments, in each DAG structure, the processing engine 112 may traverse from the key node to the root node along one or more directed edges of the DAG structure. For example, if node C is the key node, node B is a parent node of node C, node A is a parent node of node B (i.e., node A→node B→node C), and node A is a root node of the DAG structure, then the processing engine 112 may traverse from node C to node B, and then from node B to node A. More descriptions of the traversing process may be found elsewhere in the present disclosure (e.g., FIGs. 12A and 12B, and the descriptions thereof).

[0108] In 720, the processing engine 112 (e.g., the access check module 440) may determine whether one permission indicated by access authority information is traversed.

[0109] In some embodiments, the processing engine 112 may determine whether a node associated with one of the one or more permissions indicated by the access authority information of the user is traversed. For example, if the processing engine

112 traverses from node C (i.e., the key node) to node B, and then to node A (i.e., the root node), and node B is associated with one of the one or more permissions indicated by the access authority information of the user, then the processing engine 112 may determine that the permission is traversed. More descriptions of the traversing process may be found elsewhere in the present disclosure (e.g., FIGs. 12A and 12B, and the descriptions thereof).

[0110] In response to a determination that one permission indicated by the access authority information is traversed, in 730, the processing engine 112 (e.g., the access check module 440) may determine that a match exists. If one or more matches between one or more key nodes included in the request and one or more permissions indicated by the access authority information are all exist, the processing engine 112 may grant the request.

[0111] In response to a determination that none permission indicated by the access authority information is traversed, in 740, the processing engine 112 (e.g., the access check module 440) may determine that a match does not exist. Accordingly, the processing engine 112 may refuse the request.

[0112] It should be noted that the above description is merely provided for the purposes of illustration, and not intended to limit the scope of the present disclosure. For persons having ordinary skills in the art, multiple variations and modifications may be made under the teachings of the present disclosure. However, those variations and modifications do not depart from the scope of the present disclosure.

[0113] FIG. 8 is a flowchart illustrating an exemplary process for storing permission information according to some embodiments of the present disclosure. In some embodiments, the process 800 may be implemented in the permission management system 100. For example, the process 800 may be stored in the storage device 130 and/or the storage (e.g., the ROM 230, the RAM 240, etc.) as a form of instructions, and invoked and/or executed by the server 110 (e.g., the processing engine 112 in the server 110, or the processor 210 of the processing engine 112 in the server 110).

[0114] In 810, the processing engine 112 (e.g., the data structure determination

module 450) may store permission information as a data structure. In some embodiments, the processing engine 112 may store permission information as a data structure in a storage device (e.g., the storage device 130) of the permission management system 100. The permission information may include information relating to a plurality of permissions and/or relationships (e.g., inheritance relationships) between the plurality of permissions. More descriptions of the permissions may be found elsewhere in the present disclosure (e.g., FIG. 5 and the descriptions thereof).

[0115] In some embodiments, the data structure may include one or more directed acyclic graph (DAG) structures as described elsewhere in the present disclosure (e.g., FIG. 5 and the descriptions thereof). In some embodiments, the processing engine 112 may generate a plurality of nodes of the DAG structure(s). Each node may represent a permission to access to an object. For example, the processing engine 112 may store functional operation permissions as a first DAG structure as illustrated in FIG. 11A. Each node in the first DAG structure may represent a permission to perform an operation on a resource. As another example, the processing engine 112 may store permissions to data scopes as a second DAG structure as illustrated in FIG. 11B. Each node in the second DAG structure may represent a permission to access to a resource. In some embodiments, the processing engine 112 may also generate a plurality of directed edges of the DAG structure(s). Each directed edge may include a direction from a parent node to one or more child nodes of the parent node among the plurality of nodes. In some embodiments, the DAG structure may allow or facilitate permission inheritance among the plurality of permissions. For example, a parent node may have one or more permissions corresponding to one or more child nodes of the parent node.

[0116] In 820, the processing engine 112 (e.g., the association module 460) may associate a user with one or more access authorities denoted by one or more nodes.

[0117] The one or more access authorities of the user may be preset manually by a user (e.g., an administrator) of the permission management system 100, or may be determined by one or more components (e.g., the processing engine 112) of the

permission management system 100 according to different situations. In some embodiments, the association between the user and the one or more access authorities denoted by the one or more nodes may be stored in a storage device of the permission management system 100, so that the processing engine 112 may access the storage device and retrieve the one or more access authorities of the user in the management and/or control of the permissions.

[0118] In 830, the processing engine 112 (e.g., the association module 460) may associate one or more roles with the one or more nodes of the plurality of nodes.

[0119] In some embodiments, the processing engine 112 may associate a role with one or more nodes of the plurality of nodes. In some embodiments, the processing engine 112 may associate one or more roles with a node. In some embodiments, any role of the one or more roles associated with a parent node may have one or more permissions corresponding to one or more child nodes of the parent node. For example, if node A is a parent node of node B, node C, and node D, and the processing engine 112 associates a role of store manager with node A, the role of store manager may have permissions corresponding to node B (e.g., cancel an order), node C (modify an order), and node D (e.g., review an order). Specifically, as illustrated in FIG. 11A, if the processing engine 112 associates a role of store manager with node 1101b (e.g., manage order), the role of store manager may also have permissions corresponding to node 1101d (e.g., cancel order) and node 1101e (e.g., modify order).

[0120] In some embodiments, the associations between the one or more roles and the one or more nodes may be stored in a storage device of the permission management system 100, the processing engine 112 may access the storage device and retrieve the associations between the one or more roles and the one or more nodes in the management and/or control of the permissions.

[0121] In 840, the processing engine 112 (e.g., the association module 460) may associate the user with at least one role of the one or more roles. In some embodiments, the user associated with a role may have a permission corresponding to the role. For example, if the processing engine 112 associates the user with a

role of store manager, and the role of store manager has a plurality of permissions (e.g., cancelling an order, and/or modifying an order), then the user may have the permissions (e.g., cancelling an order, and/or modifying an order) associated with the role of store manager.

[0122] In some embodiments, the processing engine 112 may add, delete, and/or modify one or more nodes of the plurality of nodes in the data structure. In some embodiments, the processing engine 112 may delete a first node of a parent node in the data structure. For example, the processing engine 112 may select the first node for deletion from the data structure. The processing engine 112 may revoke an association between the first node and a corresponding role. The processing engine 112 may revoke associations between all child nodes of the first node and corresponding roles. The processing engine 112 may delete the first node and all child nodes of the first node. More descriptions of the deletion of the first node in the data structure may be found elsewhere in the present disclosure (e.g., FIG. 9 and the descriptions thereof).

[0123] In some embodiments, the processing engine 112 may modify a second node of the one or more child nodes in the data structure. The processing engine 112 may select the second node for modification from the data structure. The second node may have a first parent node. The processing engine 112 may revoke an association between the second node and the first parent node. The processing engine 112 may associate the second node with a second parent node. More descriptions of the modification of the second node in the data structure may be found elsewhere in the present disclosure (e.g., FIG. 10 and the descriptions thereof).

[0124] In some embodiments, the processing engine 112 may add a third node in the data structure. For example, the processing engine 112 may associate the third node with one or more child nodes and/or one or more parent nodes among the plurality of nodes. The processing engine 112 may generate one or more directed edges between the third node and the one or more child nodes and/or the one or more parent nodes of the third node in the data structure. The processing engine

112 may associate the third node with a permission. The processing engine 112 may associate the third node with a role.

[0125] It should be noted that the above description is merely provided for the purposes of illustration, and not intended to limit the scope of the present disclosure. For persons having ordinary skills in the art, multiple variations and modifications may be made under the teachings of the present disclosure. However, those variations and modifications do not depart from the scope of the present disclosure. In some embodiments, one or more operations may be added or omitted. For example, operation 830 and/or operation 840 may be omitted. As another example, operation 820 may be omitted.

[0126] FIG. 9 is a flowchart illustrating an exemplary process for deleting a node in a data structure according to some embodiments of the present disclosure. In some embodiments, the process 900 may be implemented in the permission management system 100. For example, the process 900 may be stored in the storage device 130 and/or the storage (e.g., the ROM 230, the RAM 240, etc.) as a form of instructions, and invoked and/or executed by the server 110 (e.g., the processing engine 112 in the server 110, or the processor 210 of the processing engine 112 in the server 110).

[0127] In 910, the processing engine 112 (e.g., the modification module 470) may select a first node for deletion from a data structure (e.g., a DAG structure). The first node may be any node of a plurality of nodes of the data structure. In some embodiments, the first node may have one or more child nodes. In some embodiments, the first node may have one or more parent nodes.

[0128] In 920, the processing engine 112 (e.g., the modification module 470) may revoke an association between the first node and a corresponding role. For example, the processing engine 112 may delete the role corresponding to the first node. As another example, the processing engine 112 may associate the role corresponding to the first node with another node of the data structure.

[0129] In 930, the processing engine 112 (e.g., the modification module 470) may revoke associations between all child nodes of the first node and corresponding

roles. In some embodiments, the processing engine 112 may traverse from the first node to all child nodes of the first node through one or more directed edges, and revoke associations between all child nodes of the first node and the corresponding roles. For example, the processing engine 112 may delete the roles corresponding to all child nodes of the first node. As another example, the processing engine 112 may associate the roles corresponding to all child nodes with one or more other nodes of the data structure. Specifically, as illustrated in FIG. 11A, if a node 1101b is to be deleted, the processing engine 112 may revoke an association between node 1101b and a corresponding role. The processing engine 112 may also revoke an association between node 1101d and a corresponding role, and an association between node 1101e and a corresponding role.

[0130] In 940, the processing engine 112 (e.g., the modification module 470) may delete the first node and all child nodes of the first node. For example, the processing engine 112 may remove the first node and all child nodes of the first node from the data structure. In some embodiments, the processing engine 112 may remove one or more directed edges connecting between the first node and all child nodes of the first node.

[0131] It should be noted that the above description is merely provided for the purposes of illustration, and not intended to limit the scope of the present disclosure. For persons having ordinary skills in the art, multiple variations and modifications may be made under the teachings of the present disclosure. However, those variations and modifications do not depart from the scope of the present disclosure. In some embodiments, the processing engine 112 may delete the first node and one or more child nodes of all child nodes of the first node. For example, the processing engine 112 may select one or more child nodes of all child nodes of the first node for deletion. The processing engine 112 may revoke associations between the one or more selected child nodes and corresponding roles. The processing engine 112 may delete the one or more selected child nodes. The processing engine 112 may associate the rest of child nodes of the first node with one or more other nodes of the data structure. Specifically, as illustrated in FIG. 11A, if node 1101b and child node

1101d of the node 1101b are to be deleted, the processing engine 112 may revoke an association between node 1101b and a correspond role, and/or an association between node 1101d and a corresponding role. The processing engine 112 may also delete node 1101b, node 1101d, a directed edge 1102a connecting node 1101a and node 1101b, and a directed edge 1102c connecting node 1101b and node 1101d. The processing engine 112 may further associate node 1101e with one or more other nodes (e.g., node 1101a, node 1101c, or node 1101f) of the data structure by generating one or more directed edges between node 1101e and the one or more other nodes (e.g., node 1101a, node 1101c, or node 1101f).

[0132] FIG. 10 is a flowchart illustrating an exemplary process for modifying a node in a data structure according to some embodiments of the present disclosure. In some embodiments, the process 1000 may be implemented in the permission management system 100. For example, the process 1000 may be stored in the storage device 130 and/or the storage (e.g., the ROM 230, the RAM 240, etc.) as a form of instructions, and invoked and/or executed by the server 110 (e.g., the processing engine 112 in the server 110, or the processor 210 of the processing engine 112 in the server 110).

[0133] In 1010, the processing engine 112 (e.g., the modification module 470) may select a second node for modification from a data structure (e.g., a DAG structure). The second node may be any node of a plurality of nodes of the data structure. In some embodiments, the second node may have a first parent node. In some embodiments, the second node may have one or more child nodes.

[0134] In 1020, the processing engine 112 (e.g., the modification module 470) may revoke an association between the second node and the first parent node. For example, the processing engine 112 may delete a directed edge between the second node and the first parent node.

[0135] In 1030, the processing engine 112 (e.g., the modification module 470) may associate the second node with a second parent node. For example, the processing engine 112 may generate a directed edge between the second node and the second parent node. The directed edge may include a direction from the

second parent node to the second node. Accordingly, a user associated with a role corresponding to the second node has the permission corresponding to the second node. A user associated with a role corresponding to the second parent node has the permission corresponding to the second node. Accordingly, a user associated with a role corresponding to the first parent node does not have the permission corresponding to the second node. More descriptions of the modification of one or more nodes of a data structure may be found elsewhere in the present disclosure (e.g., FIGs. 12A and 12B, and the descriptions thereof).

[0136] It should be noted that the above description is merely provided for the purposes of illustration, and not intended to limit the scope of the present disclosure. For persons having ordinary skills in the art, multiple variations and modifications may be made under the teachings of the present disclosure. However, those variations and modifications do not depart from the scope of the present disclosure. In some embodiments, one or more operations may be omitted. For example, operation 1020 may be omitted. The processing engine 112 may associate the second node with the second parent node without revoking the association between the second node and the first parent node. Specifically, as illustrated in FIG. 11A, if node 1101e is to be modified, the processing engine 112 may associate node 1101e with node 1101c by generating a directed edge directing from node 1101c to node 1101e.

[0137] FIG. 11A is a schematic diagram illustrating an exemplary data structure according to some embodiments of the present disclosure.

[0138] As illustrated in FIG. 11A, the data structure may include a first DAG structure. The first DAG structure may include a plurality of nodes (e.g., node 1101a, node 1101b, node 1101c, node 1101d, node 1101e, and node 1101f) and a plurality of directed edges (e.g., edge 1102a, edge 1102b, edge 1102c, edge 1102d, and edge 1102e). Node 1101a may be a root node of the first DAG structure. Each node may represent a functional operation permissions. Taking a restaurant management system as an example, node 1101b may represent “manage order”, node 1101c may represent “manage dish”, node 1101d may present “cancel order”,

node 1101e may present "modify order", and node 1101f may present "sell or not sell dish". In some embodiments, a node may be regarded as a parent node of one or more other nodes of the data structure. Meanwhile, the node may be regarded as a child node of one or more other nodes of the data structure. For example, node 1101b may be a parent node of node 1101d and node 1101e. Node 1101b may also be a child node of node 1101a. Each directed edge may include a direction from a parent node to a child node of the parent node. For example, node 1101b may be a parent node of node 1101d and node 1101e, and node 1101d and node 1101e may be child nodes of node 1101b. Accordingly, directed edge 1102c connecting node 1101b and node 1101d including a direction from node 1101b to node 1101d, and directed edge 1102d connecting between node 1101b and node 1101e including a direction from node 1101b to node 1101e.

[0139] The first DAG structure may allow permission inheritance among the plurality of functional operation permissions. In some embodiments, the parent node may have one or more permissions corresponding to one or more child nodes of the parent node. For example, if the processing engine 112 associate a user with a permission corresponding to node 1101b (e.g., manage order), then the user may also have permissions corresponding to node 1101d (e.g., cancel order) and node 1101e (e.g., modify order). That is, the user may have permissions to manage order, cancel order and modify order in the restaurant management system. As another example, if the processing engine 112 associate a user with a permission corresponding to node 1101a, then the user may also have permissions corresponding to node 1101b and node 1101c. Further, the user may also have permissions corresponding to node 1101d, node 1101e, and node 1101f. That is, the user may have permissions to manage order, manage dish, cancel order, modify order, and sell or not sell dish in the restaurant management system.

[0140] FIG. 11B is a schematic diagram illustrating an exemplary data structure according to some embodiments of the present disclosure.

[0141] As illustrated in FIG. 11B, the data structure may include a second DAG structure. The second DAG structure may include a plurality of nodes (e.g., node

1103a, node 1103b, node 1103c, node 1103d, node 1103e, and node 1103f) and a plurality of directed edges (e.g., edge 1104a, edge 1104b, edge 1104c, edge 1104d, and edge 1104e). Node 1103a may be a root node of the second DAG structure. Each node may represent a permission to a data scope. For example, node 1103a may represent "KFC", node 1103b may represent "north china region", node 1103c may represent "central china region", node 1103d may present "Beijing store", node 1103e may present "Tianjin store", and node 1103f may present "Chengdu store". Each directed edge may include a direction from a parent node to a child node of the parent node. For example, node 1103b and node 1103c may be child nodes of node 1103a. Node 1103d and node 1103e may be child nodes of node 1103b. Node 1103f may be a child node of node 1103c.

[0142] The second DAG structure may allow permission inheritance among the plurality of permissions to data scopes. In some embodiments, the parent node may have one or more permissions corresponding to one or more child nodes of the parent node. For example, if the processing engine 112 associate a user with a permission corresponding to node 1103b (e.g., north china region), then the user may also have permissions corresponding to node 1103d (e.g., Beijing store) and node 1103e (e.g., Tianjin store). That is, the user may have permissions to access data associated with north china region, Beijing store and Tianjin store.

[0143] In some embodiments, the processing engine 112 may associate one or more roles with one or more nodes of the first DAG structure and/or one or more nodes of the second DAG structure. For example, the processing engine 112 may associate a role of "store assistant" with node 1101e (i.e., modify order) and node 1101c (i.e., manage dish), and may associate a role of "store manager" with node 1101b (i.e., manage order) and node 1101c (i.e., manage dish). Merely for illustration purpose, the processing engine 112 may determine that access authority information of user A is "role list: store assistant; node list: Beijing store, Tianjin store". In this situation, user A may have permissions to modify order and sell or not sell dish in Beijing store and Tianjin store. The processing engine 112 may determine that access authority information of user B is "role list: store manager;

node list: north china region". In this situation, user B may have permissions to manage order (including, e.g., cancelling order and modifying order) and manage dish (including e.g., selling or not selling dish) in the north china region (including, e.g., Beijing store and Tianjin store).

[0144] It should be noted that the above description is merely provided for the purposes of illustration, and not intended to limit the scope of the present disclosure. For persons having ordinary skills in the art, multiple variations and modifications may be made under the teachings of the present disclosure. However, those variations and modifications do not depart from the scope of the present disclosure. In some embodiments, a data structure may include one DAG structure. For example, the first DAG structure illustrated in FIG. 11A or the second DAG structure illustrated in FIG. 11B may be omitted. As another example, the first DAG structure illustrated in FIG. 11A and the second DAG structure illustrated in FIG. 11B may be integrated into a single DAG structure. In the single DAG structure, a first portion of nodes may be associated with functional operation permissions, while a second portion of nodes may be associated with data scope permissions. In some embodiments, the data structure may further include a plurality of DAG structures. For example, the data structure may include the first DAG structure as illustrated in FIG. 11A, the second DAG structure as illustrated in FIG. 11B, and a third DAG structure (not shown). The third DAG structure may allow role inheritance among a plurality of roles. The third DAG structure may include a plurality of nodes and a plurality of directed edges. Each node may represent a role. Taking a restaurant management system as an example, a first exemplary node may represent "store manager", and a second exemplary node may represent "store assistant". Further, the first exemplary node may be a parent node of the second exemplary node.

[0145] FIG. 12A and FIG. 12B are schematic diagrams illustrating exemplary node modification in a data structure according to some embodiments of the present disclosure.

[0146] As illustrated in FIG. 12A, the data structure may include a DAG structure. The DAG structure may include a plurality of nodes (e.g., node 1201a, node 1201b,

node 1201c, node 1201d, node 1201e, node 1201f, node 1201g, node 1201h, and node 1201i) and a plurality of directed edges. Each node may represent a permission to a data scope. For example, node 1201a may represent "corporation ID", node 1201b may represent "north china region ID", node 1201c may represent "south china region ID", node 1201d may represent "Beijing ID", node 1201e may represent "Tianjin ID", node 1201f may represent "Shanghai ID", node 1201g may represent "Hangzhou", node 1201h may represent "store A ID", and node 1201i may represent "store B ID". The processing engine 112 may associate user A with node 1201b, associate user B with node 1201d, associate user C with node 1201h, and associate user D with node 1201c, respectively. Accordingly, user A may have a permission to a data scope corresponding to node 1201b (e.g., "north china region ID"), user B may have a permission to a data scope corresponding to node 1201d (e.g., "Beijing ID"), user C may have a permission to a data scope corresponding to node 1201h (e.g., "store A ID"), and user D may have a permission to a data scope corresponding to node 1201c (e.g., "south china region ID"). Merely for illustration purpose, the processing engine 112 may modify node 1201d and node 1201e of the DAG structure, as illustrated in FIG. 12B. For example, the processing engine 112 may revoke an association between node 1201d and node 1201b, and associate node 1201d with node 1201c. The processing engine 112 may revoke an association between node 1201e and node 1201b, and associate node 1201e with node 1201c.

[0147] If user A initiates a request for access to store A denoted by node 1201h, the processing engine 112 may traverse from node 1201h to the root node (e.g., node 1201a) of the DAG structure. The processing engine 112 may determine whether the permission indicated by access authority information (e.g., permission corresponding to node 1201b) of user A is traversed. For example, the processing engine 112 may traverse from node 1201h to node 1201d, 1201c, and then to the root node 1201a. In response to a determination that the permission indicated by the access authority information (e.g., permission corresponding to node 1201b) of user A is not traversed, the processing engine 112 may refuse the request from user

A.

[0148] If user B initiates a request for access to store A denoted by node 1201h, the processing engine 112 may traverse from node 1201h to the root node (e.g., node 1201a) of the DAG structure. The processing engine 112 may determine whether the permission indicated by access authority information (e.g., permission corresponding to node 1201d) of user B is traversed. In response to a determination that the permission indicated by the access authority information (e.g., permission corresponding to node 1201d) of user B is traversed, the processing engine 112 may grant the request from user B.

[0149] If user C initiates a request for access to store A denoted by node 1201h, the processing engine 112 may traverse from node 1201h to the root node (e.g., node 1201a) of the DAG structure. The processing engine 112 may determine whether the permission indicated by access authority information (e.g., permission corresponding to node 1201h) of user C is traversed. In response to a determination that the permission indicated by the access authority information (e.g., permission corresponding to node 1201h) of user C is traversed, the processing engine 112 may grant the request from user C.

[0150] If user D initiates a request for access to store A denoted by node 1201h, the processing engine 112 may traverse from node 1201h to the root node (e.g., node 1201a) of the DAG structure. The processing engine 112 may determine whether the permission indicated by the access authority information (e.g., permission corresponding to node 1201c) of user D is traversed. In response to a determination that the permission indicated by the access authority information (e.g., permission corresponding to node 1201c) of user D is traversed, the processing engine 112 may grant the request from user D.

[0151] FIG. 13 is a schematic diagram illustrating an exemplary process for controlling a user's access to an object according to some embodiments of the present disclosure.

[0152] As illustrated in FIG. 13, a user may send a request for access to an object to the processing engine 112 via an application (e.g., the application 380 in FIG. 3)

installed in a terminal device (e.g., the client terminal 140) or a webpage in a browser of an operation system (e.g., a restaurant management service system) via the network 120. The request may include one or more key nodes. For example, the user may initiate a request for selling or not selling dish in a Beijing store by, for example, selecting or confirming one or more options (e.g., action button(s), function menu(s), or the like) on an interface of the application. The key nodes included in the request may be "Beijing store" and "sell or not sell dish". In some embodiments, the request may further include a user's identity information, for example, a user identification (ID) 10001 as illustrated in FIG. 13.

[0153] The processing engine 112 may obtain first node information corresponding to the one or more key nodes. In some embodiments, the processing engine 112 may obtain the first node information corresponding to the one or more key nodes from node information stored in one or more storage devices (e.g., the storage device 130) of the permission management system 100 or an external storage device. As illustrated in FIG. 13, the first node information may include "Beijing store" and "sell or not sell dish". In some embodiments, the processing engine 112 may further verify the first node information (e.g., in the data structures illustrated in FIGs. 11A and 11B) as described in connection with operation 520. In response to a determination that the first node information is verified, the processing engine 112 may return the first node information (also referred to as node set A) to the permission management system 100. As illustrated in FIG. 13, the node set A may refer to ["Beijing store", "sell or not sell dish"].

[0154] The processing engine 112 may obtain access authority information of the user. In some embodiments, the processing engine 112 may obtain the access authority information of the user based on a mapping relation between one or more predefined access permissions and one or more users from access authority information stored in one or more storage devices of the permission management system 100 or an external storage device. In some embodiments, the mapping relation between predefined access permission(s) and user(s) may include a first level mapping relation and a second level mapping relation. In some embodiments,

the first level mapping relation may include an association relationship between user ID(s), role(s), and permission(s) to data scopes, and the second level mapping relation may include an association relationship between role(s) and functional operation permission(s). Additionally or alternatively, the first level mapping relation may include an association relationship between user ID(s), role(s), and functional operation permission(s), and the second level mapping relation may include an association relationship between role(s) and permission(s) to data scopes.

[0155] In some embodiments, the processing engine 112 may obtain role information of the user and second node information (also referred to as node set C) corresponding to the access authority information of the user based on the first level mapping relation. In some embodiments, the processing engine 112 may obtain third node information (also referred to as node set B) corresponding to the role information. In some embodiments, the processing engine 112 may obtain the third node information corresponding to the role information based on a mapping relation (e.g., the second level mapping relation) between one or more predefined access permissions and one or more roles from role information stored in one or more storage devices of the permission management system 100 or an external storage device. The processing engine 112 may traverse from the nodes of the node set A to a root node of at least one DAG structure. The processing engine 112 may determine whether one of the nodes in the node set B and/or the node set C is traversed. Upon a determination that one of the nodes in the node set B and/or the node set C is traversed, the processing engine 112 may return “True” to the operation system. Accordingly, the request from the user may be granted. The user may be allowed to sell or not sell dish in Beijing store. Upon a determination that none of the nodes in the node set B and/or the node set C is traversed, the processing engine 112 may return “False” to the operation system. Accordingly, the request from the user may be refused. The user may not be allowed to sell or not sell dish in Beijing store.

[0156] Merely for illustration purpose, as illustrated in FIG. 13, the first level mapping relation may include “user ID 1001; role list: store assistant; node list:

Beijing store (see node 1103d in FIG. 11B), Tianjin store (see node 1103e in FIG. 11B)", and the second level mapping relation may include "role: store assistant; node list: modify order (see node 1101e in FIG. 11A), manage dish (see node 1101c in FIG. 11A). The processing engine 112 may return the second node information to the permission management system 100. As illustrated in FIG. 13, the role information may refer to ["store assistant"], and the node set C may refer to ["Beijing store", "Tianjin store"]. The processing engine 112 may return the third node information to the permission management system 100. As illustrated in FIG. 13, the node set B may refer to ["modify order", "manage dish"]. The processing engine 112 may traverse from the nodes of the node set A ["Beijing store", "sell or not sell dish"] to a root node of at least one DAG structure (e.g., the first DAG structure illustrated in FIG. 11A, the second DAG structure illustrated in FIG. 11B). For example, the processing engine 112 may traverse from "sell or not sell dish" (see node 1101f) to "manage dish" (see node 1101c), and then to the root node 1101a in the first DAG structure. The processing engine 112 may also traverse from "Beijing store" (see node 1103d) to "north china region" (see node 1103b), and then to the root node "KFC" (see node 1103a) in the second DAG structure. The processing engine 112 may determine whether one of the nodes in the node set B and/or the node set C is traversed. Specifically, as illustrated in FIG. 13, the processing engine 112 may determine whether one of the nodes in the node set B ["modify order", "manage dish"] is traversed in the first DAG structure. The processing engine 112 may also determine whether one of the nodes in the node set C ["Beijing store", "Tianjin store"] is traversed. In this example, "manage dish" (see node 1101c) of the node set B is traversed, and "Beijing store" (see node 1103d) of the node set C is traversed, and then the processing engine 112 may return "True" to the operation system. Accordingly, the request from the user 10001 may be granted, and the user 10001 may be allowed to sell or not sell dish in Beijing store.

[0157] As another example, if the first level mapping relation includes "user ID 1001; role list: store manager; node list: north China region (see node 1103b in FIG. 11B)", and the second level mapping relation includes "role: store manager; node list:

manage order (see node 1101b in FIG. 11A), manage dish (see node 1101c in FIG. 11A), then the role information may refer to ["store manager"], the node set C may refer to ["north China region"], and the node set B may refer to ["manage order", "manage dish"]. The processing engine 112 may traverse from the nodes of the node set A ["Beijing store", "sell or not sell dish"] to a root node of at least one DAG structure (e.g., the first DAG structure illustrated in FIG. 11A, the second DAG structure illustrated in FIG. 11B). For example, the processing engine 112 may traverse from "sell or not sell dish" (see node 1101f) to "manage dish" (see node 1101c), and then to the root node 1101a in the first DAG structure. The processing engine 112 may also traverse from "Beijing store" (see node 1103d) to "north china region" (see node 1103b), and then to the root node "KFC" (see node 1103a) in the second DAG structure. The processing engine 112 may determine whether one of the nodes in the node set B ["manage order", "manage dish"] is traversed in the first DAG structure. The processing engine 112 may also determine whether one of the nodes in the node set C ["north China region"] is traversed. In this example, "manage dish" (see node 1101c) of the node set B is traversed, and "north China region" (see node 1103b) of the node set C is traversed, and then the processing engine 112 may return "True" to the operation system. Accordingly, the request from the user 10001 may be granted, and the user 10001 may be allowed to sell or not sell dish in Beijing store.

[0158] As a further example, if the first level mapping relation includes "user ID 1001; role list: store assistant; node list: Chengdu store (see node 1103f in FIG. 11B)", and the second level mapping relation includes "role: store assistant; node list: modify order (see node 1101e in FIG. 11A), manage dish (see node 1101c in FIG. 11A), then the role information may refer to ["store assistant"], the node set C may refer to ["Chengdu store"], and the node set B may refer to ["modify order", "manage dish"]. The processing engine 112 may traverse from the nodes of the node set A ["Beijing store", "sell or not sell dish"] to a root node of at least one DAG structure (e.g., the first DAG structure illustrated in FIG. 11A, the second DAG structure illustrated in FIG. 11B). For example, the processing engine 112 may traverse from

"sell or not sell dish" (see node 1101f) to "manage dish" (see node 1101c), and then to the root node 1101a in the first DAG structure. The processing engine 112 may also traverse from "Beijing store" (see node 1103d) to "north china region" (see node 1103b), and then to the root node "KFC" (see node 1103a) in the second DAG structure. The processing engine 112 may determine whether one of the nodes in the node set B ["modify order", "manage dish"] is traversed in the first DAG structure. The processing engine 112 may also determine whether one of the nodes in the node set C ["Chengdu store"] is traversed. In this example, "manage dish" (see node 1101c) of the node set B is traversed, but "Chengdu store" (see node 1103f) of the node set C is not traversed. In some embodiments, the processing engine 112 may return "False" to the operation system. Accordingly, the request from the user 10001 may be refused, and the user 10001 may not be allowed to sell or not sell dish in Beijing store (but the user 10001 may be allowed to sell or not sell dish in Chengdu store).

[0159] It should be noted that the above description is merely provided for the purposes of illustration, and not intended to limit the scope of the present disclosure. For persons having ordinary skills in the art, multiple variations and modifications may be made under the teachings of the present disclosure. However, those variations and modifications do not depart from the scope of the present disclosure.

[0160] Having thus described the basic concepts, it may be rather apparent to those skilled in the art after reading this detailed disclosure that the foregoing detailed disclosure is intended to be presented by way of example only and is not limiting. Various alterations, improvements, and modifications may occur and are intended to those skilled in the art, though not expressly stated herein. These alterations, improvements, and modifications are intended to be suggested by this disclosure, and are within the spirit and scope of the exemplary embodiments of this disclosure.

[0161] Moreover, certain terminology has been used to describe embodiments of the present disclosure. For example, the terms "one embodiment," "an embodiment," and "some embodiments" mean that a particular feature, structure or characteristic described in connection with the embodiment is included in at least

one embodiment of the present disclosure. Therefore, it is emphasized and should be appreciated that two or more references to “an embodiment” or “one embodiment” or “an alternative embodiment” in various portions of this specification are not necessarily all referring to the same embodiment. Furthermore, the particular features, structures or characteristics may be combined as suitable in one or more embodiments of the present disclosure.

[0162] Further, it will be appreciated by one skilled in the art, aspects of the present disclosure may be illustrated and described herein in any of a number of patentable classes or context including any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof. Accordingly, aspects of the present disclosure may be implemented entirely hardware, entirely software (including firmware, resident software, micro-code, etc.) or combining software and hardware implementation that may all generally be referred to herein as a “module,” “unit,” “component,” “device,” or “system.” Furthermore, aspects of the present disclosure may take the form of a computer program product embodied in one or more computer readable media having computer readable program code embodied thereon.

[0163] A computer readable signal medium may include a propagated data signal with computer readable program code embodied therein, for example, in baseband or as part of a carrier wave. Such a propagated signal may take any of a variety of forms, including electro-magnetic, optical, or the like, or any suitable combination thereof. A computer readable signal medium may be any computer readable medium that is not a computer readable storage medium and that may communicate, propagate, or transport a program for use by or in connection with an instruction execution system, apparatus, or device. Program code embodied on a computer readable signal medium may be transmitted using any appropriate medium, including wireless, wireline, optical fiber cable, RF, or the like, or any suitable combination of the foregoing.

[0164] Computer program code for carrying out operations for aspects of the present disclosure may be written in any combination of one or more programming

languages, including an object oriented programming language such as Java, Scala, Smalltalk, Eiffel, JADE, Emerald, C++, C#, VB. NET, Python or the like, conventional procedural programming languages, such as the "C" programming language, Visual Basic, Fortran 2003, Perl, COBOL 2002, PHP, ABAP, dynamic programming languages such as Python, Ruby and Groovy, or other programming languages. The program code may execute entirely on the user's computer, partly on the user's computer, as a stand-alone software package, partly on the user's computer and partly on a remote computer or entirely on the remote computer or server. In the latter scenario, the remote computer may be connected to the user's computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the Internet using an Internet Service Provider) or in a cloud computing environment or offered as a service such as a Software as a Service (SaaS).

[0165] Furthermore, the recited order of processing elements or sequences, or the use of numbers, letters, or other designations therefore, is not intended to limit the claimed processes and methods to any order except as may be specified in the claims. Although the above disclosure discusses through various examples what is currently considered to be a variety of useful embodiments of the disclosure, it is to be understood that such detail is solely for that purpose, and that the appended claims are not limited to the disclosed embodiments, but, on the contrary, are intended to cover modifications and equivalent arrangements that are within the spirit and scope of the disclosed embodiments. For example, although the implementation of various components described above may be embodied in a hardware device, it may also be implemented as a software only solution, e.g., an installation on an existing server or mobile device.

[0166] Similarly, it should be appreciated that in the foregoing description of embodiments of the present disclosure, various features are sometimes grouped together in a single embodiment, figure, or description thereof for the purpose of streamlining the disclosure aiding in the understanding of one or more of the various

embodiments. This method of disclosure, however, is not to be interpreted as reflecting an intention that the claimed subject matter requires more features than are expressly recited in each claim. Rather, claim subject matter lie in less than all features of a single foregoing disclosed embodiment.

WHAT IS CLAIMED IS:

1. A method implemented on a computing device having one or more processors and one or more storage devices for controlling a user's access to an object, the method comprising:

obtaining a request for access to the object from the user;

determining access authority information of the user based on a mapping relation between one or more predefined access permissions and one or more users, the access authority information of the user indicating one or more permissions; and

performing an access check to resolve the request;

wherein

the one or more permissions are included in a plurality of permissions that are stored in a storage of the one or more storage device as a data structure, the data structure including at least one directed acyclic graph (DAG) structure, each DAG structure including a plurality of nodes and a plurality of directed edges, each node representing a permission, each directed edge including a direction from a parent node to a child node among the plurality of nodes;

the request includes node information of at least one key node to be checked; and

the performing an access check to resolve the request includes:

determining whether the node information of the at least one key node matches one of the one or more permissions indicated by the access authority information; and

upon determination that the match exists, granting the request for access to the object.

2. The method of claim 1, wherein the each DAG structure allows permission inheritance among the plurality of permissions, and the parent node has one or more

permissions corresponding to one or more child nodes of the parent node.

3. The method of claim 1 or 2, wherein the access authority information of the user further includes one or more roles of the user associated with the one or more permissions.

4. The method of any one of claims 1 to 3, wherein the object includes an operation on a resource.

5. The method of claim 4, wherein the at least one DAG structure includes a first DAG structure, the first DAG structure including a plurality of first nodes of the plurality of nodes associated with functional operation permissions.

6. The method of any one of claims 1 to 5, wherein the object includes a resource.

7. The method of claim 6, wherein the at least one DAG structure includes a second DAG structure, the second DAG structure including a plurality of second nodes of the plurality of nodes associated with permissions to data scopes.

8. The method of any one of claims 1-7, wherein the plurality of permissions include at least one of

a permission of granting a permission of the user to another user,
a permission of creating a permission for another user,
a permission of inquiring a permission of a user,
a permission of modifying a permission of a user, and
a permission of revoking a permission of a user.

9. The method of any one of claims 1-8, wherein the determining whether the node information of the at least one key node matches one of the one or more permissions indicated by the access authority information comprises:

for each key node of the at least one key node,

traversing from the each key node to a root node of the at least one DAG structure;

determining whether one of the one or more permissions indicated by the access authority information is traversed;

upon determination that one of the one or more permissions indicated by the access authority information is traversed, determining that the match exists; and

upon determination that none of the one or more permissions indicated by the access authority information is traversed, determining that the match does not exist.

10. The method of any one of claims 1 to 9, further comprising:

verifying the node information of the at least one key node included in the request for access to the object.

11. The method of claim 10, further comprising:

upon determination that the node information of the at least one key node included in the request for access to the object is not verified, refusing the request.

12. A system for controlling a user's access to an object, comprising:

at least one computer-readable storage medium storing a set of instructions; and

at least one processor configured to communicate with the at least one computer-readable storage medium, wherein when executing the set of instructions, the at least one processor is directed to cause the system to:

obtain a request for access to the object from the user;

determine access authority information of the user based on a mapping relation between one or more predefined access permissions and one or more users, the access authority information of the user indicating one or

more permissions; and

perform an access check to resolve the request;

wherein

the one or more permissions are included in a plurality of permissions that are stored in a storage of the one or more storage device as a data structure, the data structure including at least one directed acyclic graph (DAG) structure, each DAG structure including a plurality of nodes and a plurality of directed edges, each node representing a permission, each directed edge including a direction from a parent node to a child node among the plurality of nodes;

the request includes node information of at least one key node to be checked; and

the performing an access check to resolve the request includes:

determining whether the node information of the at least one key node matches one of the one or more permissions indicated by the access authority information; and

upon determination that the match exists, granting the request for access to the object.

13. The system of claim 12, wherein the each DAG structure allows permission inheritance among the plurality of permissions, and the parent node has one or more permissions corresponding to one or more child nodes of the parent node.

14. The system of claim 12 or 13, wherein the access authority information of the user further includes one or more roles of the user associated with the one or more permissions.

15. The system of any one of claims 12 to 14, wherein the object includes an operation on a resource.

16. The system of claim 15, wherein the at least one DAG structure includes a first DAG structure, the first DAG structure including a plurality of first nodes of the plurality of nodes associated with functional operation permissions.

17. The system of any one of claims 12 to 16, wherein the object includes a resource.

18. The system of claim 17, wherein the at least one DAG structure includes a second DAG structure, the second DAG structure including a plurality of second nodes of the plurality of nodes associated with permissions to data scopes.

19. The system of any one of claims 12-18, wherein the plurality of permissions include at least one of

a permission of granting a permission of the user to another user,

a permission of creating a permission for another user,

a permission of inquiring a permission of a user,

a permission of modifying a permission of a user, and

a permission of revoking a permission of a user.

20. The system of any one of claims 12-19, wherein to determine whether the node information of the at least one key node matches one of the one or more permissions indicated by the access authority information, the at least one processor is directed to cause the system to:

for each key node of the at least one key node,

traverse from the each key node to a root node of the at least one DAG structure;

determine whether one of the one or more permissions indicated by the access authority information is traversed;

upon determination that one of the one or more permissions indicated by the access authority information is traversed, determine that the match

exists; and

upon determination that none of the one or more permissions indicated by the access authority information is traversed, determine that the match does not exist.

21. The system of any one of claims 12 to 20, the at least one processor is directed to cause the system to:

verify the node information of the at least one key node included in the request for access to the object.

22. The system of claim 21, the at least one processor is directed to cause the system to:

upon determination that the node information of the at least one key node included in the request for access to the object is not verified, refuse the request.

23. A non-transitory computer readable medium, comprising at least one set of instructions for controlling a user's access to an object, wherein when executed by one or more processors of a computing device, the at least one set of instructions causes the computing device to perform a method, the method comprising:

obtaining a request for access to the object from the user;

determining access authority information of the user based on a mapping relation between one or more predefined access permissions and one or more users, the access authority information of the user indicating one or more permissions; and

performing an access check to resolve the request;

wherein

the one or more permissions are included in a plurality of permissions that are stored in a storage of the one or more storage device as a data structure, the data structure including at least one directed acyclic graph (DAG) structure, each DAG structure including a plurality of nodes and a

plurality of directed edges, each node representing a permission, each directed edge including a direction from a parent node to a child node among the plurality of nodes;

the request includes node information of at least one key node to be checked; and

the performing an access check to resolve the request includes:

determining whether the node information of the at least one key node matches one of the one or more permissions indicated by the access authority information; and

upon determination that the match exists, granting the request for access to the object.

24. A method implemented on a computing device having one or more processors and one or more storage devices for storing permission information, the method comprising:

storing the permission information as a data structure, the data structure including at least one directed acyclic graph (DAG) structure, including:

generating a plurality of nodes of the at least one DAG structure, each node representing a permission to access to an object; and

generating a plurality of directed edges of the at least one DAG structure, each directed edge including a direction from a parent node to one or more child nodes of the parent node among the plurality of nodes; and

associating a user with one or more access authorities denoted by one or more nodes of the plurality of nodes.

25. The method of claim 24, further comprising:

associating one or more roles with the one or more nodes of the plurality of nodes, wherein any role of the one or more roles associated with the parent node has one or more permissions corresponding to the one or more child nodes of the parent node; and

associating the user with at least one role of the one or more roles, wherein the user has a permission corresponding to the at least one role.

26. The method of claim 25, further comprising:

adding, deleting or modifying one or more nodes of the plurality of nodes in the data structure.

27. The method of claim 26, wherein deleting a first node of the parent node in the data structure comprises:

selecting the first node for deletion from the data structure;

revoking an association between the first node and a corresponding role;

revoking associations between all child nodes of the first node and corresponding roles; and

deleting the first node and all child nodes of the first node.

28. The method of claim 26, wherein deleting a first node of the parent node in the data structure comprises:

selecting the first node for deletion from the data structure;

revoking an association between the first node and a corresponding role;

selecting at least one child node of all child nodes of the first node for deletion from the data structure;

revoking an association between the at least one child node and at least one corresponding role;

deleting the first node and the at least one child node of the first node; and

associating the rest of child nodes of the first node with at least one node of the data structure.

29. The method of claim 26, wherein modifying a second node of the one or more child nodes in the data structure comprises:

selecting the second node for modification from the data structure, the second

node having a first parent node; and

associating the second node with a second parent node,

wherein

a user associated with a role of the one or more roles corresponding to the second node has the permission corresponding to the second node,

a user associated with a role of the one or more roles corresponding to the first parent node has the permission corresponding to the second node, and

a user associated with a role of the one or more roles corresponding to the second parent node has the permission corresponding to the second node.

30. The method of claim 29, the method further comprises:

revoking an association between the second node and the first parent node, wherein

the permission corresponding to the second node is revoked from the user associated with the role of the one or more roles corresponding to the first parent node.

31. The method of claim 26, wherein adding a third node in the data structure comprises:

associating the third node with at least one parent node and/or at least one child node among the plurality of nodes;

associating the third node with a permission; and

associating the third node with a role.

32. A system for storing permission information, comprising:

at least one computer-readable storage medium storing a set of instructions; and

at least one processor configured to communicate with the at least one

computer-readable storage medium, wherein when executing the set of instructions, the at least one processor is directed to:

store the permission information as a data structure, the data structure including at least one directed acyclic graph (DAG) structure, the at least one processor is directed to:

generate a plurality of nodes of the at least one DAG structure, each node representing a permission to access to an object; and

generate a plurality of directed edges of the at least one DAG structure, each directed edge including a direction from a parent node to one or more child nodes of the parent node among the plurality of nodes; and

associate a user with one or more access authorities denoted by one or more nodes of the plurality of nodes.

33. The system of claim 32, the at least one processor is directed to:

associate one or more roles with the one or more nodes of the plurality of nodes, wherein any role of the one or more roles associated with the parent node has one or more permissions corresponding to the one or more child nodes of the parent node; and

associate the user with at least one role of the one or more roles, wherein the user has a permission corresponding to the at least one role.

34. The system of claim 33, the at least one processor is directed to:

add, delete or modify one or more nodes of the plurality of nodes in the data structure.

35. The system of claim 34, wherein to delete a first node of the parent node in the data structure, the at least one processor is directed to:

select the first node for deletion from the data structure;

revoke an association between the first node and a corresponding role;

revoke associations between all child nodes of the first node and

corresponding roles; and

delete the first node and all child nodes of the first node.

36. The system of claim 34, wherein to delete a first node of the parent node in the data structure, the at least one processor is directed to:

select the first node for deletion from the data structure;

revoke an association between the first node and a corresponding role;

select at least one child node of all child nodes of the first node for deletion from the data structure;

revoke an association between the at least one child node and at least one corresponding role;

delete the first node and the at least one child node of the first node; and

associate the rest of child nodes of the first node with at least one node of the data structure.

37. The system of claim 34, wherein to modify a second node of the one or more child nodes in the data structure, the at least one processor is directed to:

select the second node for modification from the data structure, the second node having a first parent node; and

associate the second node with a second parent node,

wherein

a user associated with a role of the one or more roles corresponding to the second node has the permission corresponding to the second node,

a user associated with a role of the one or more roles corresponding to the first parent node has the permission corresponding to the second node, and

a user associated with a role of the one or more roles corresponding to the second parent node has the permission corresponding to the second node.

38. The system of claim 37, the at least one processor is directed to:

revoke an association between the second node and the first parent node, wherein

the permission corresponding to the second node is revoked from the user associated with the role of the one or more roles corresponding to the first parent node.

39. The system of claim 34, wherein to add a third node in the data structure, the at least one processor is directed to:

associate the third node with at least one parent node and/or at least one child node among the plurality of nodes;

associate the third node with a permission; and

associate the third node with a role.

40. A non-transitory computer readable medium, comprising at least one set of instructions for storing permission information, wherein when executed by one or more processors of a computing device, the at least one set of instructions causes the computing device to perform a method, the method comprising:

storing the permission information as a data structure, the data structure including at least one directed acyclic graph (DAG) structure, including:

generating a plurality of nodes of the at least one DAG structure, each node representing a permission to access to an object; and

generating a plurality of directed edges of the at least one DAG structure, each directed edge including a direction from a parent node to one or more child nodes of the parent node among the plurality of nodes; and

associating a user with one or more access authorities denoted by one or more nodes of the plurality of nodes.

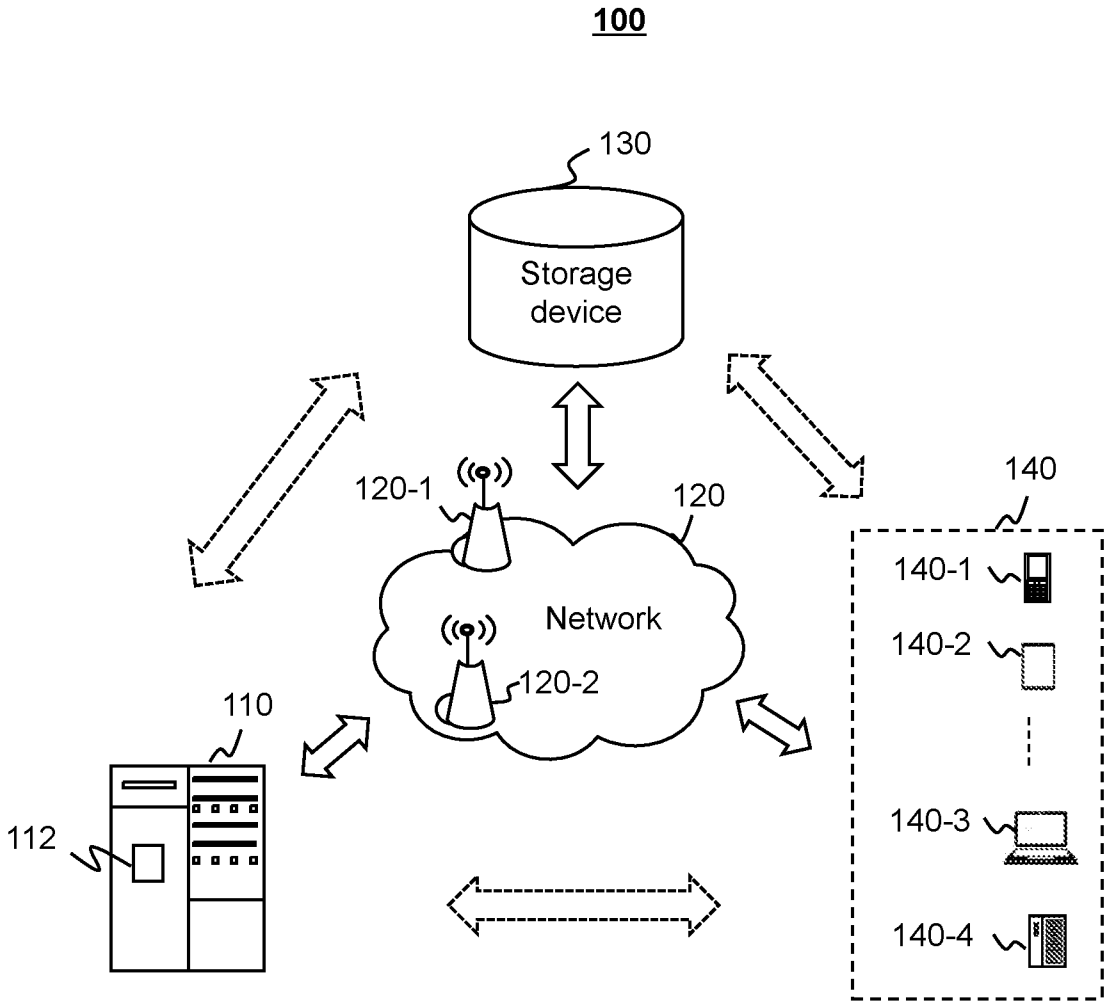
100
130
Storage device
120-1
120
140
Network
140-1
140-2
120-2
110
112
140-3
140-4

FIG. 1

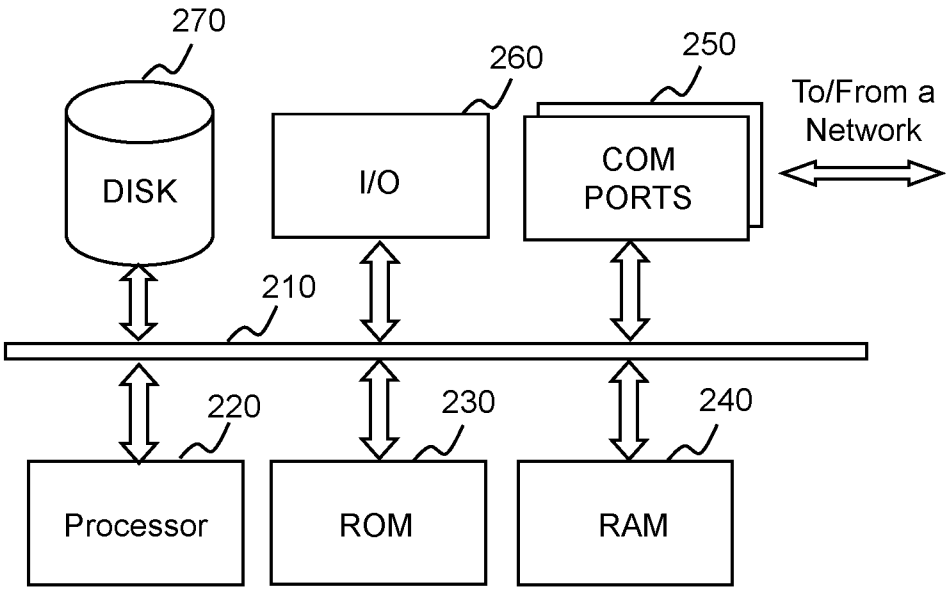
200
270
260
250
To/From a Network
DISK
I/O
COM PORTS
210
220
230
240
Processor
ROM
RAM

FIG. 2

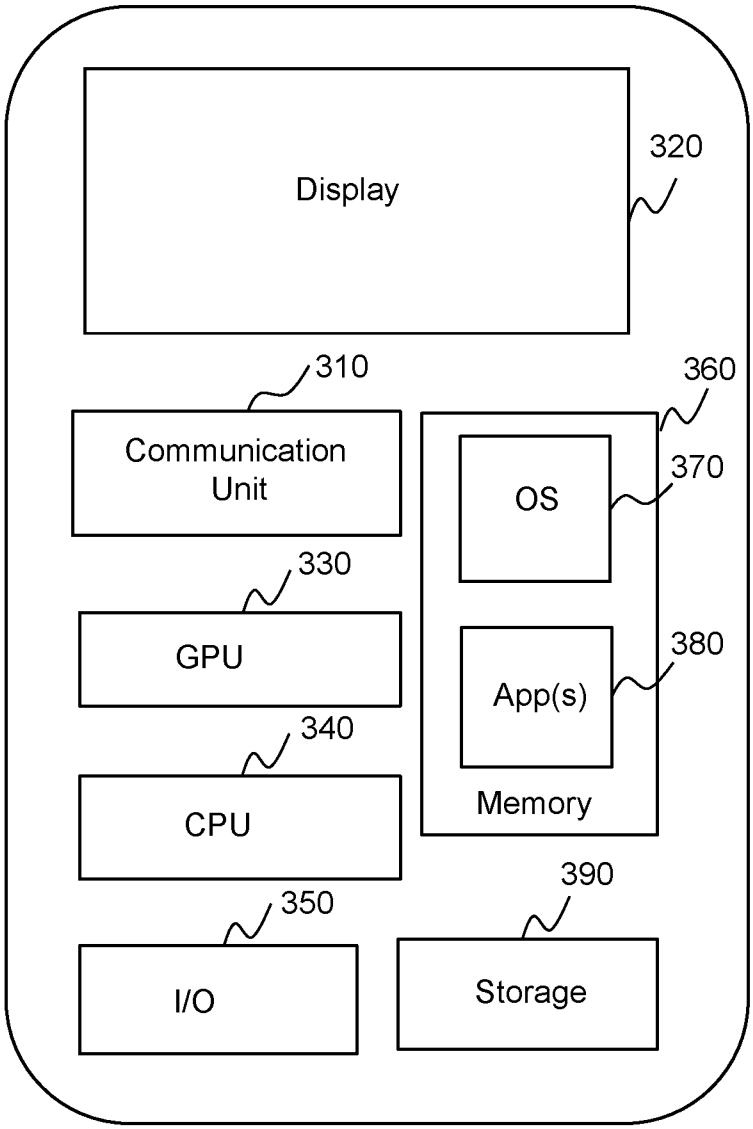
300
Display
320
310
Communication Unit
330
GPU
340
CPU
350
I/O
360
OS
370
App(s)
380
Memory
390
Storage

FIG. 3

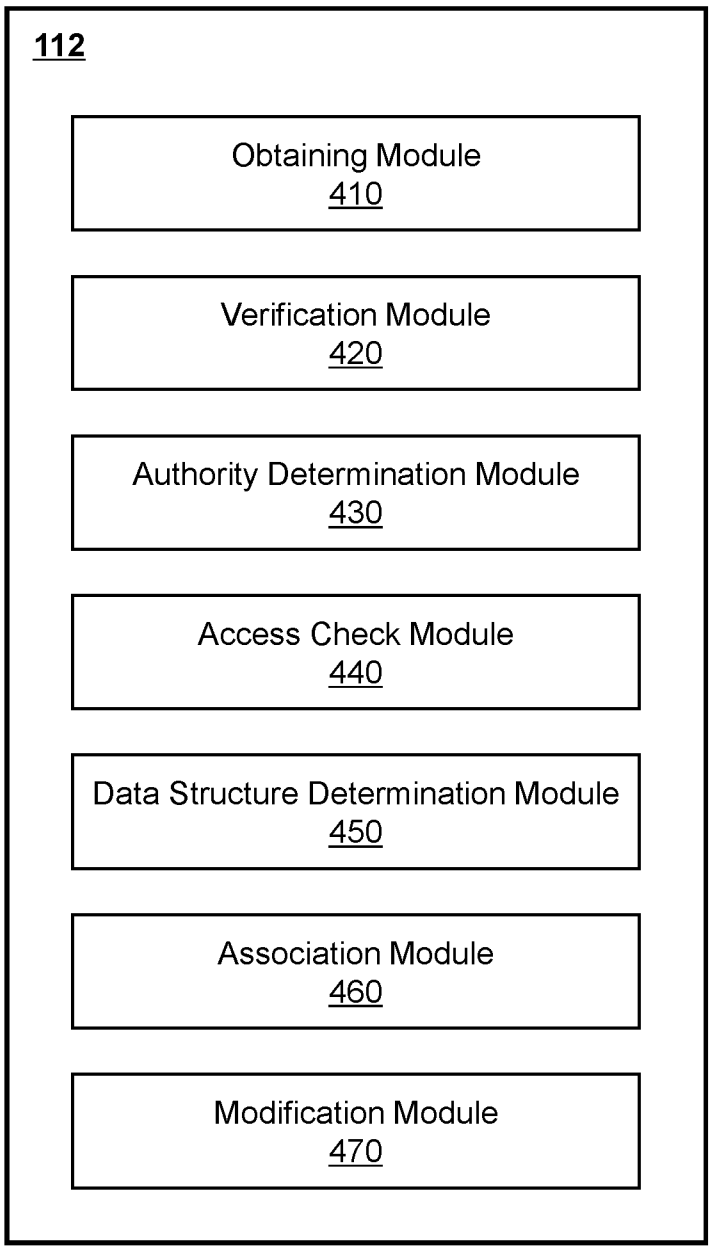
112
Obtaining Module
410
Verification Module
420
Authority Determination Module
430
Access Check Module
440
Data Structure Determination Module
450
Association Module
460
Modification Module
470

FIG. 4

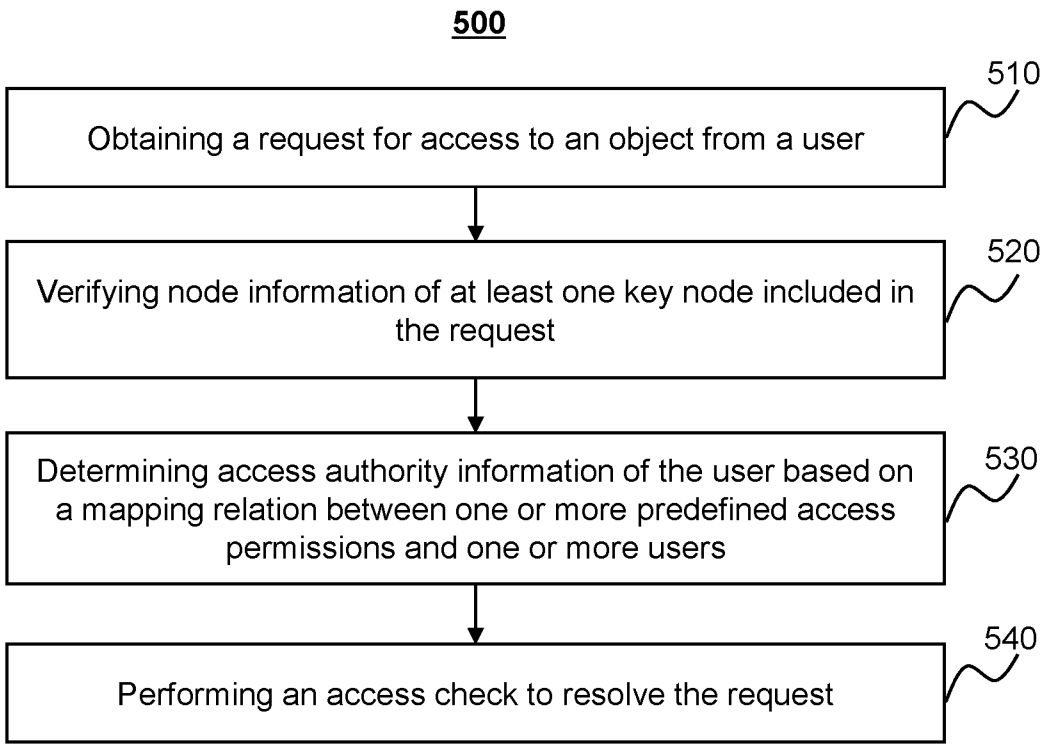
500
510
Obtaining a request for access to an object from a user
520
Verifying node information of at least one key node included in the request
530
Determining access authority information of the user based on a mapping relation between one or more predefined access permissions and one or more users
540
Performing an access check to resolve the request

FIG. 5

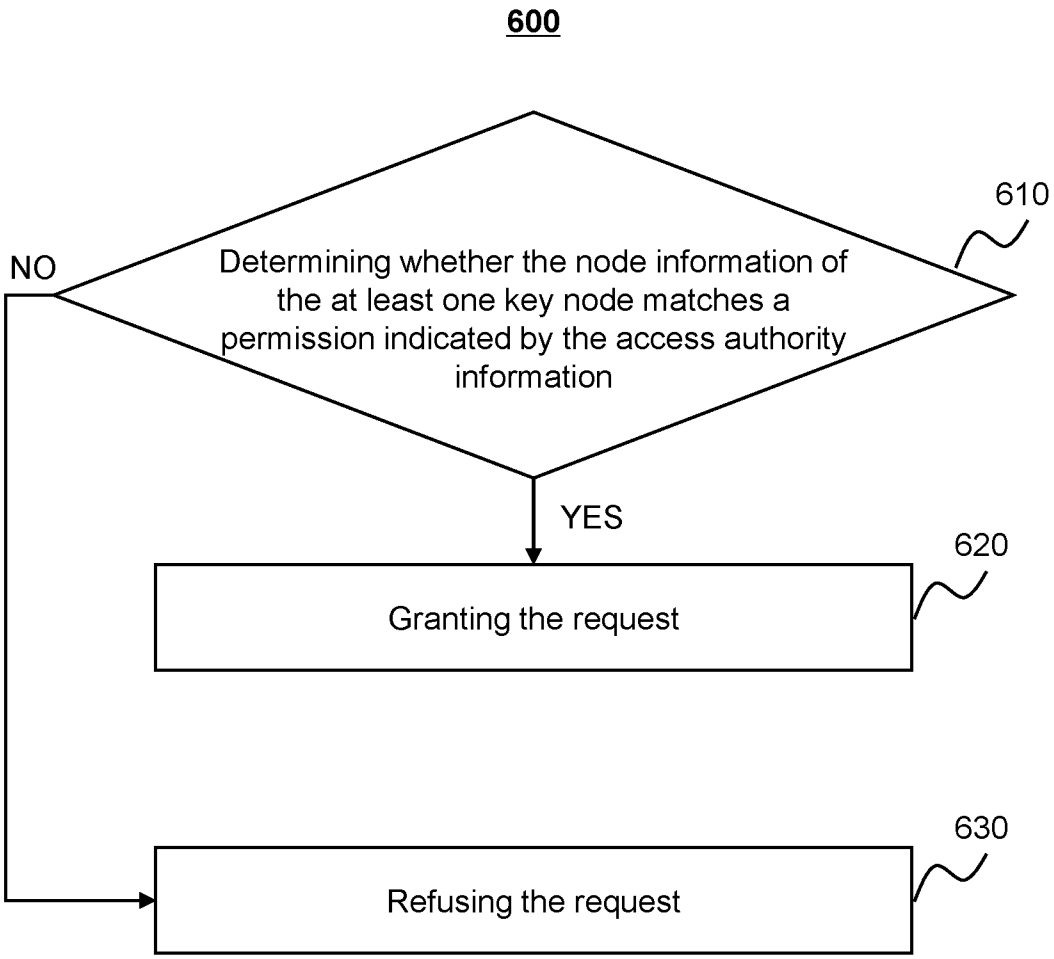
600
610
Determining whether the node information of the at least one key node matches a permission indicated by the access authority information
NO
YES
620
Granting the request
630
Refusing the request

FIG. 6

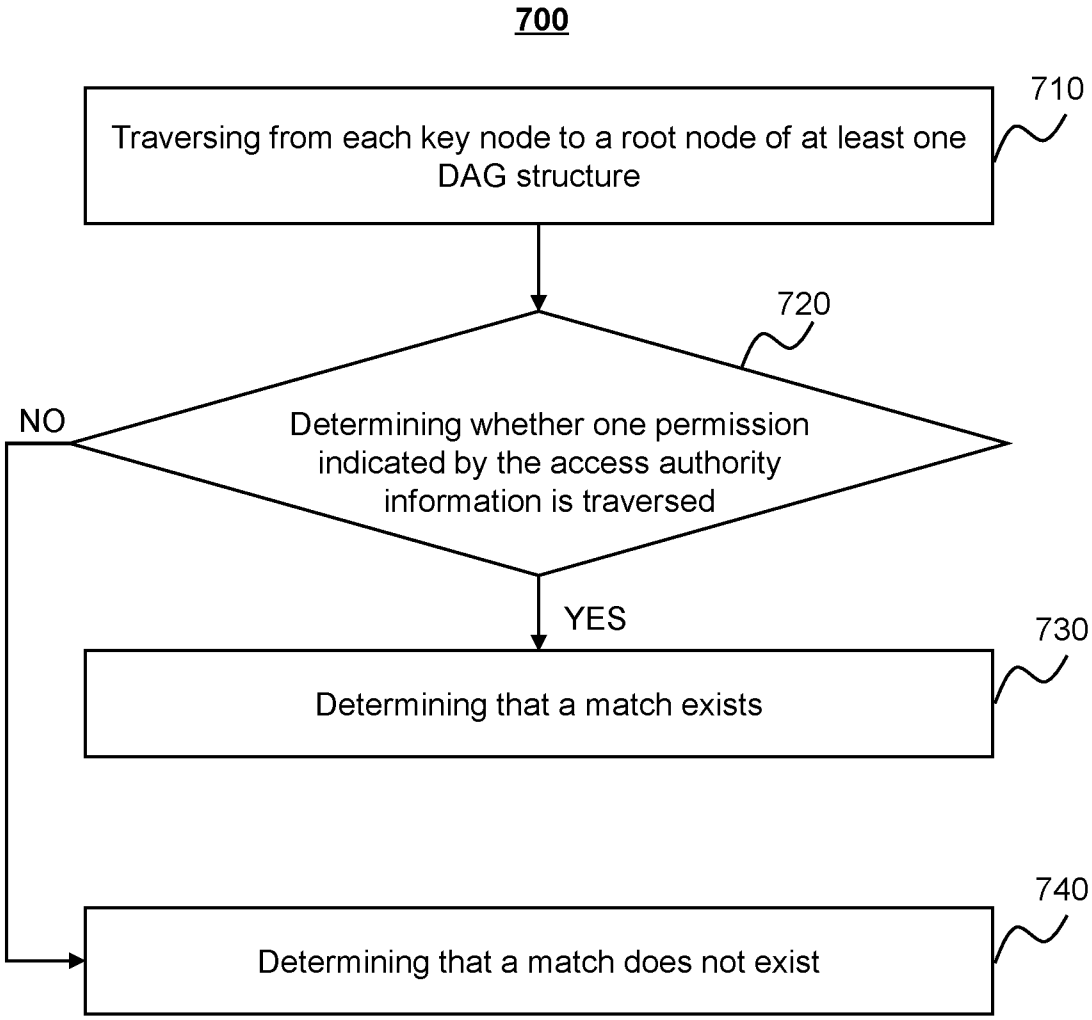
700
Traversing from each key node to a root node of at least one DAG structure
710
720
Determining whether one permission indicated by the access authority information is traversed
NO
YES
Determining that a match exists
730
Determining that a match does not exist
740

FIG. 7

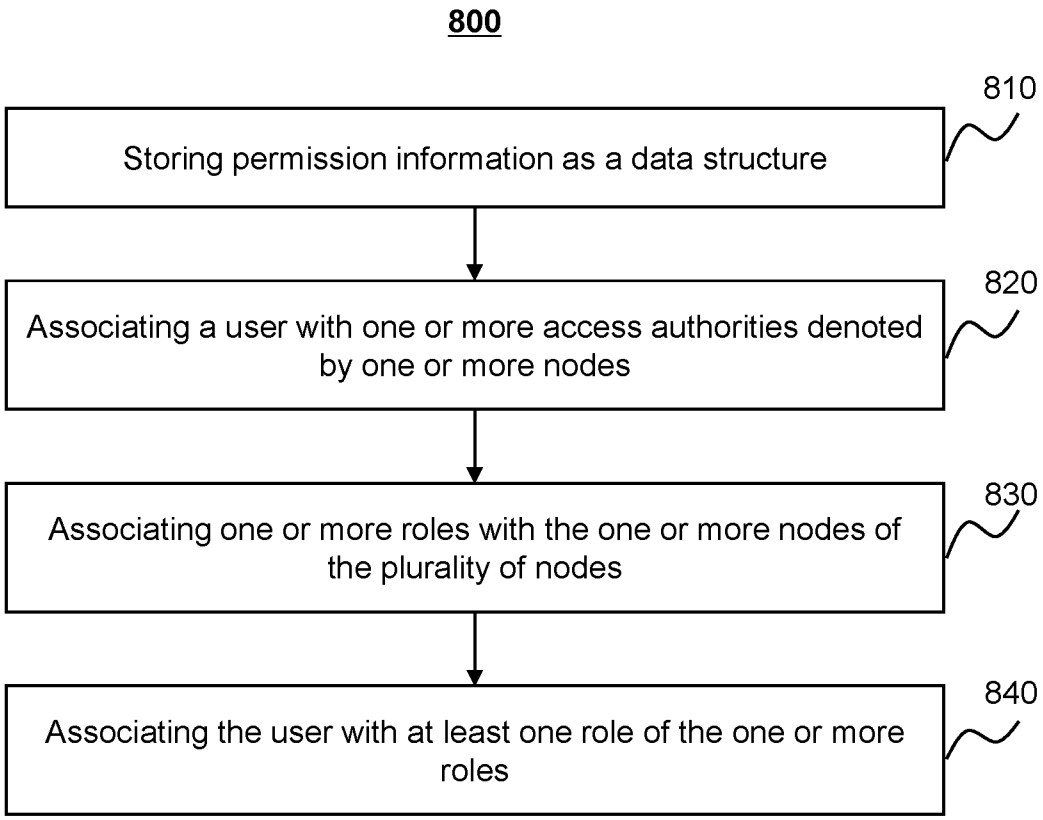
800
Storing permission information as a data structure
810
Associating a user with one or more access authorities denoted by one or more nodes
820
Associating one or more roles with the one or more nodes of the plurality of nodes
830
Associating the user with at least one role of the one or more roles
840

FIG. 8

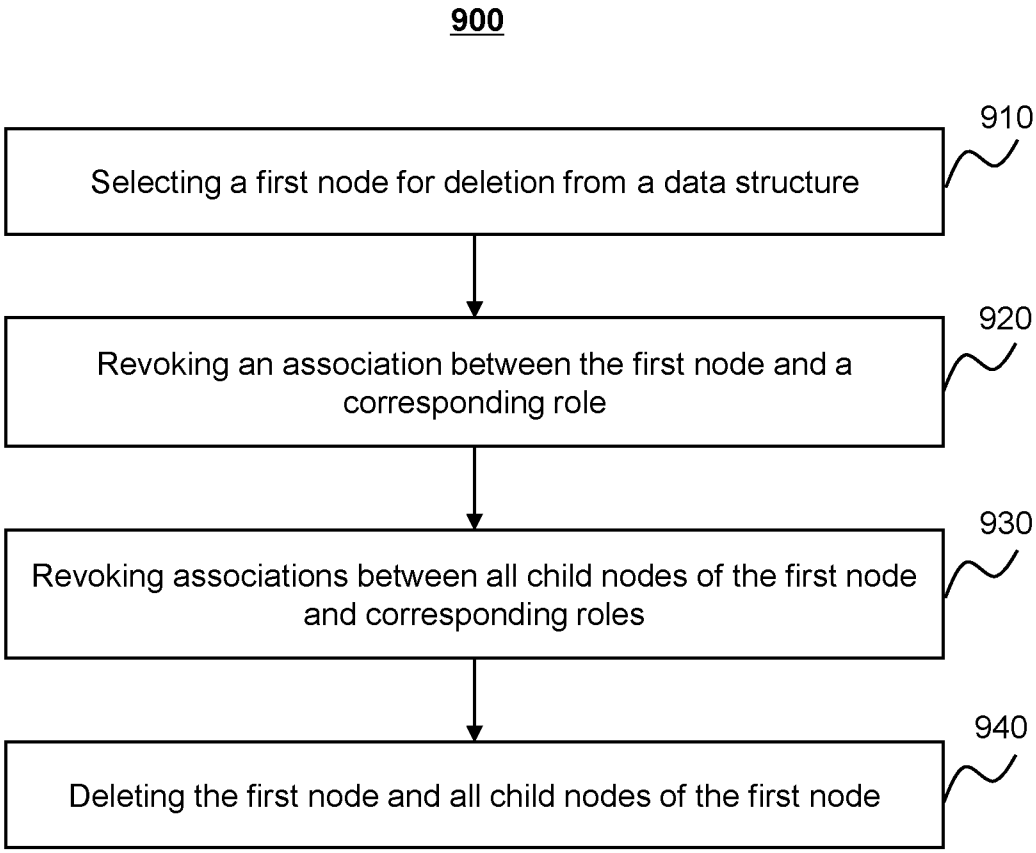
900
910
Selecting a first node for deletion from a data structure
920
Revoking an association between the first node and a corresponding role
930
Revoking associations between all child nodes of the first node and corresponding roles
940
Deleting the first node and all child nodes of the first node

FIG. 9

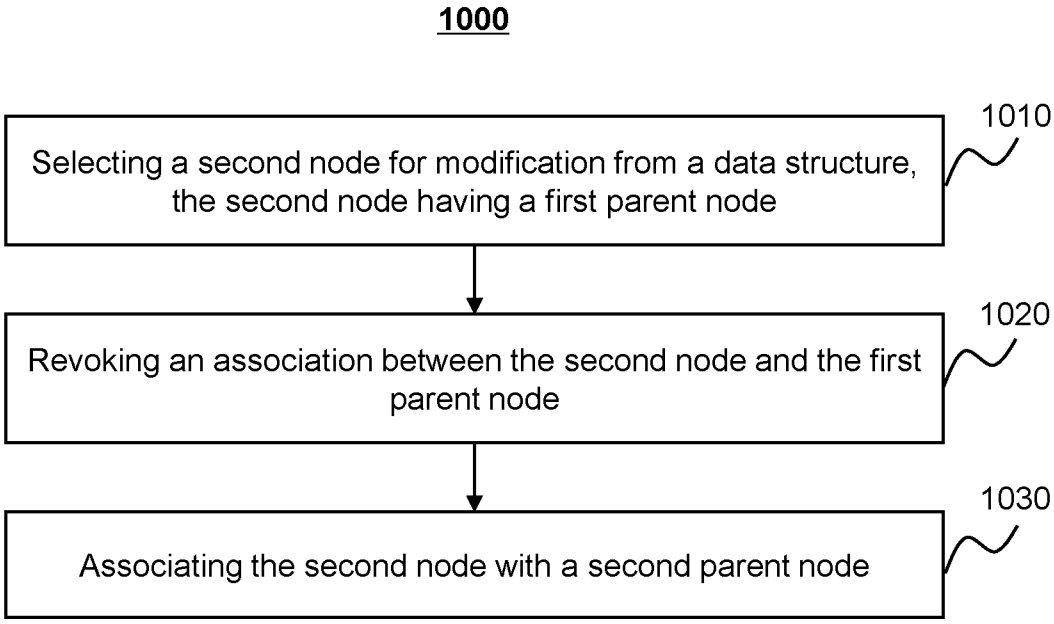
1000
Selecting a second node for modification from a data structure, the second node having a first parent node
1010
Revoking an association between the second node and the first parent node
1020
Associating the second node with a second parent node
1030

FIG. 10

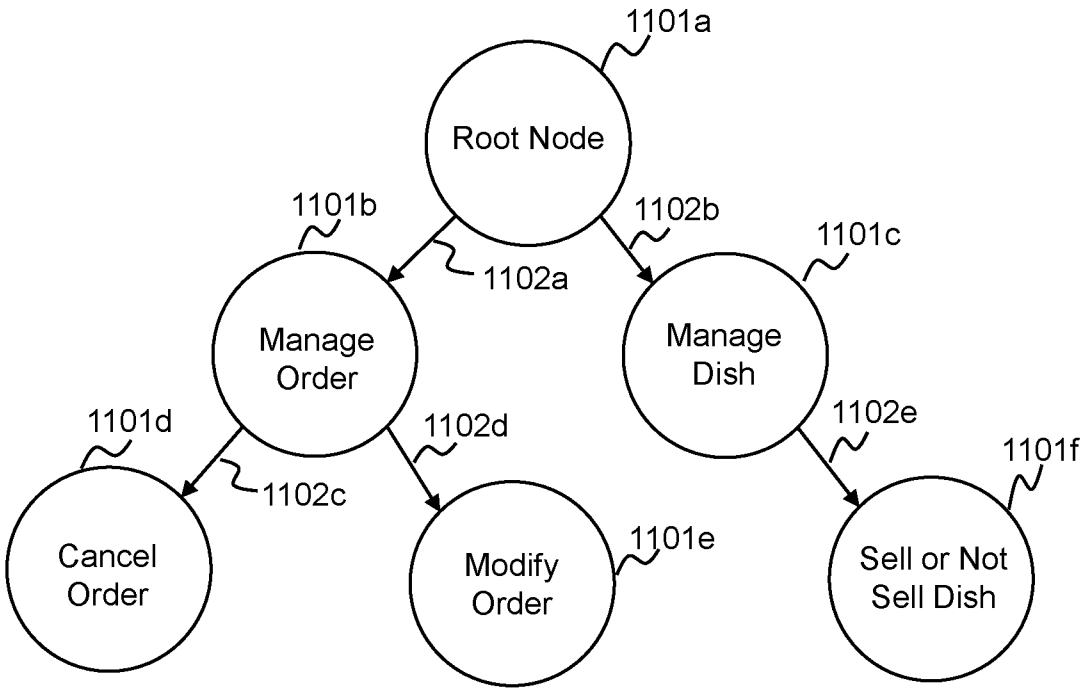

FIG. 11A

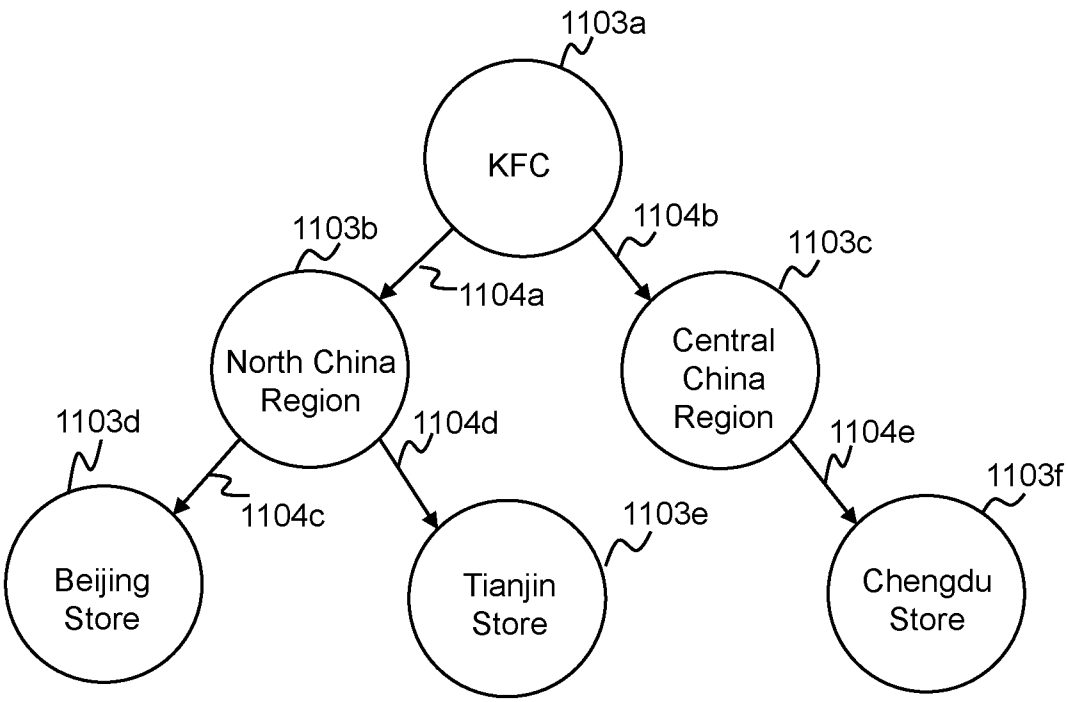

FIG. 11B

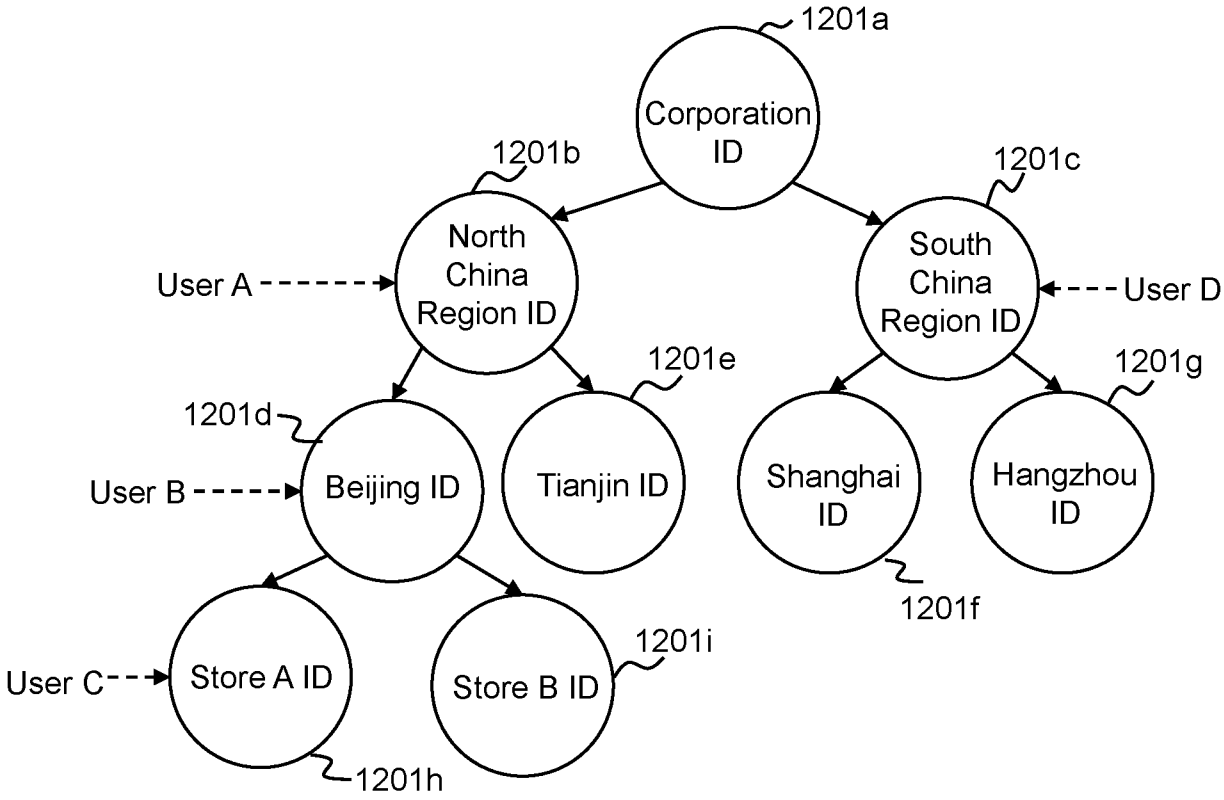

FIG. 12A

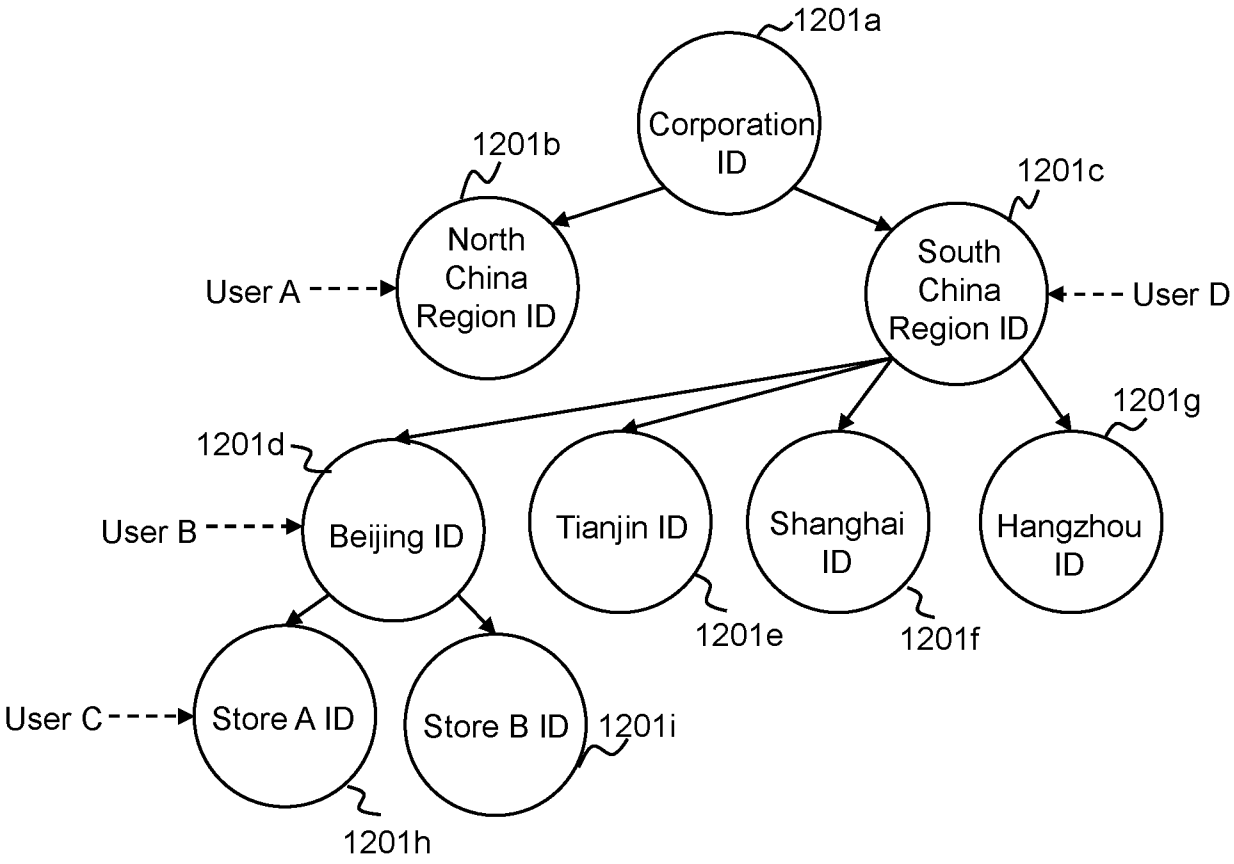

FIG. 12B

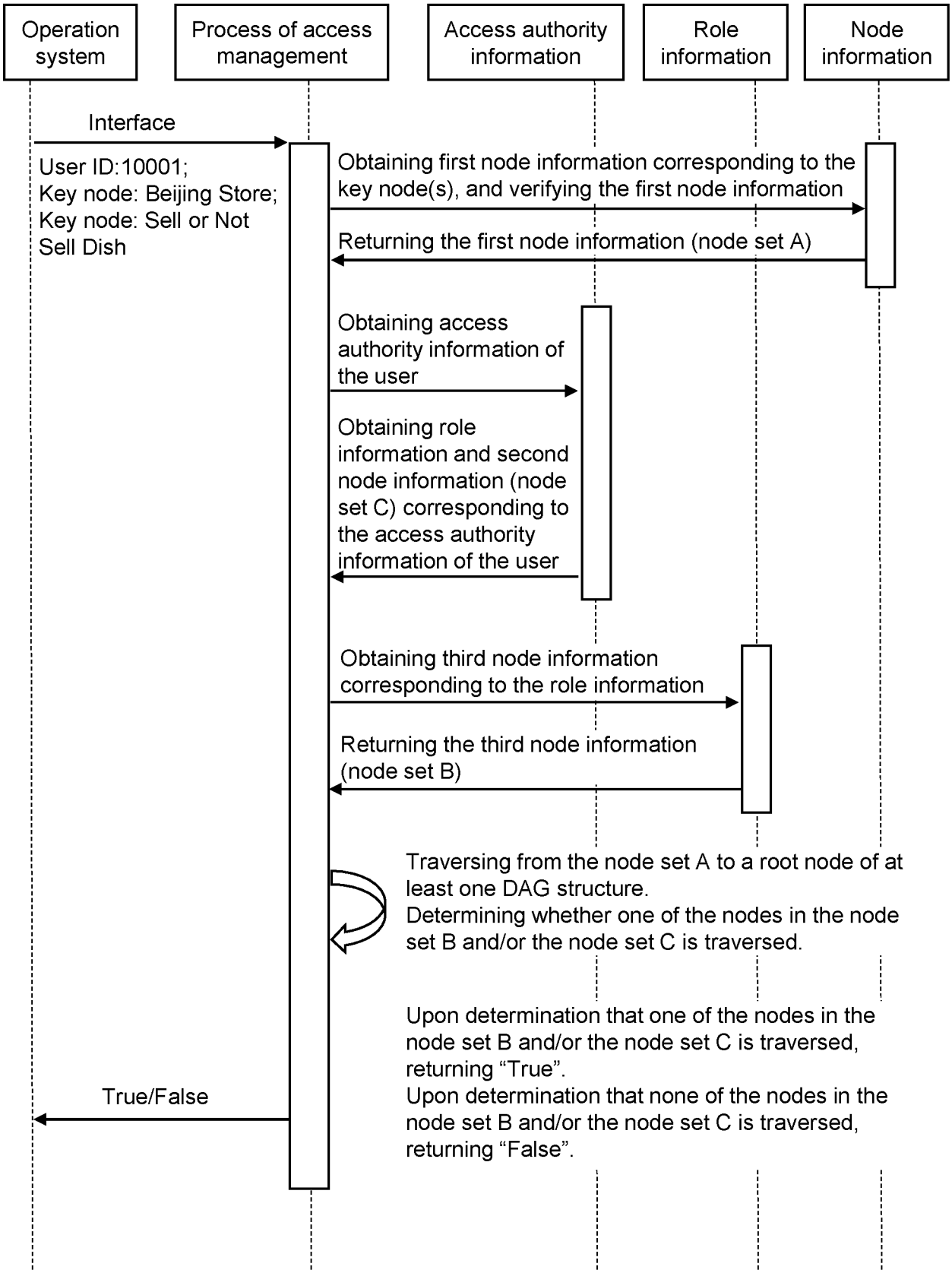
Operation system
Process of access management
Access authority information
Role information
Node information
Interface
User ID:10001;
Key node: Beijing Store;
Key node: Sell or Not Sell Dish
Obtaining first node information corresponding to the key node(s), and verifying the first node information
Returning the first node information (node set A)
Obtaining access authority information of the user
Obtaining role information and second node information (node set C) corresponding to the access authority information of the user
Obtaining third node information corresponding to the role information
Returning the third node information (node set B)
Traversing from the node set A to a root node of at least one DAG structure.
Determining whether one of the nodes in the node set B and/or the node set C is traversed.
Upon determination that one of the nodes in the node set B and/or the node set C is traversed, returning “True”.
Upon determination that none of the nodes in the node set B and/or the node set C is traversed, returning “False”.
True/False

FIG. 13

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2018/123072

A. CLASSIFICATION OF SUBJECT MATTER

G06F 21/60(2013.01)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)

G06F 21/-

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

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

CNTXT,CNABS, CNKI,SIPOABS,DWPI:access, request, permission, directed acyclic graph, DAG, check, add, delete, modify

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 8875230 B1 (MEDIDATA SOLUTIONS INC) 28 October 2014 (2014-10-28) column 1 line 10 to column 9 line 30, claims 1-24, figures 2-11	24-40
Y	US 8875230 B1 (MEDIDATA SOLUTIONS INC) 28 October 2014 (2014-10-28) column 1 line 10 to column 9 line 30, claims 1-24, figures 2-11	1-23
Y	CN 102542069 A (UNIVERSITY PEKING FOUNDER GROUP CORP ET AL.) 04 July 2012 (2012-07-04) paragraphs [0029]-[0066], claims 1-6, figures 1-3	1-23

☐ 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 another 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	Date of mailing of the international search report
04 September 2019	**12 September 2019**
Name and mailing address of the ISA/CN **National Intellectual Property Administration, PRC** **6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing 100088** **China** Facsimile No. **(86-10)62019451**	Authorized officer **SONG,Yunyun** Telephone No. **(86-10)62411661**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2018/123072

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
US	8875230	B1	28 October 2014	EP	3084590	A4	31 May 2017
				EP	3084590	B1	30 January 2019
				WO	2015094480	A1	25 June 2015
				EP	3084590	A1	26 October 2016
CN	102542069	A	04 July 2012	CN	102542069	B	25 March 2015