(19) **Bundesrepublik Deutschland**
Deutsches Patent- und Markenamt

(10) **DE 600 19 839 T2** 2005.10.06

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 039 380 B1**
(21) Deutsches Aktenzeichen: **600 19 839.1**
(96) Europäisches Aktenzeichen: **00 300 189.8**
(96) Europäischer Anmeldetag: **12.01.2000**
(97) Erstveröffentlichung durch das EPA: **27.09.2000**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **04.05.2005**
(47) Veröffentlichungstag im Patentblatt: **06.10.2005**

(51) Int Cl.[7]: **G06F 9/445**
G06F 17/30, H04L 12/24

(30) Unionspriorität:
239596 **29.01.1999** **US**

(84) Benannte Vertragsstaaten:
DE, FR, GB

(73) Patentinhaber:
Sun Microsystems, Inc., Palo Alto, Calif., US

(74) Vertreter:
HOFFMANN & EITLE, 81925 München

(72) Erfinder:
Traversat, Bernard A., San Francisco, California 94303, US; Slaughter, Gregory L., Palo Alto, California 95120, US; Saulpaugh, Tom, San Jose, California 95120, US

(54) Bezeichnung: **Verfahren zum Austauch von Daten zwischen einer Javasystemdatenbank und einem LDAP Verzeichnis**

Beschreibung

ALLGEMEINER STAND DER TECHNIK

[0001] Die vorliegende Erfindung betrifft im Allgemeinen Computersoftware und Computernetzwerkanwendungen. Im Besonderen betrifft sie Client-Server-Anwendungen und die Übertragung und Anordnung von Konfigurationsdaten unter Komponenten oder Speicherbereichen in einem Computernetzwerk.

[0002] Ein herkömmliches Computernetzwerk geht einher mit dem Verbinden einer Reihe von PCs (Personal Computer), die gängigerweise als Clients bezeichnet werden, mit einem oder mehreren Server-Computern. Client-Computer sind im Allgemeinen eigenständig und enthalten innerhalb ihres residenten Speichers viele der Informationen, die zum Ausführen von Benutzeranwendungen und Kommunizieren mit einem Netzwerk benötigt werden. Diese beinhalten Informationen über ihre eigene Konfiguration in Bezug auf Soft- und Hardwarefähigkeiten und -anforderungen. Ein Client-Computer greift auf Netzwerkserver typischerweise aus vielfältigen Gründen zu, wie z.B. zum Zugreifen auf Netzwerksoftwareanwendungen, E-Mail, Abrufen und Speichern von Informationen in einer Netzwerkdatenbank oder Zugreifen aufs Internet. Für einen bestimmten Client-Computer spezifische Informationen residieren jedoch im Allgemeinen auf jenem Client-Computer. Diese Informationen können beispielsweise die Speicherspezifikationen und Bustypen, zusätzliche Prozessoren und andere Hardwarespezifikationen beinhalten. Da Client-Computer relativ eigenständig sind und ihre eigenen Konfigurationsinformationen speichern (im Gegensatz zum Speichern derartiger Daten auf einem Server), ist die Aufgabe, Konfigurations- und Benutzerdaten zusätzlich zu Endbenutzer-Anwendungsdaten auf einem Client-Computer zu verwalten, zunehmend mühsam geworden.

[0003] Während es möglich ist, kleinere Aktualisierungen oder Problembehebungen für Anwendungen, die auf einem Netzwerkserver residieren, zu den Client-Computern zu verteilen, erfordert jedwede signifikante Aktualisierung oder Problembehebung oder die Installation einer neuen Anwendung, die sich auf jeden Client auswirkt, dass ein Netzwerkadministrator individuell auf jeden Client-Computer zugreift und ihn aktualisiert. Mit der wachsenden Zahl mit Netzwerken verbundener Computer, dich sich in einigen Unternehmen im Bereich von Zehntausenden bewegt, ist die Aufgabe des Installierens größerer Überarbeitungen oder Aktualisierungen für Anwendungssoftware oder allgemeine Konfigurationssoftware teuer, ineffizient und zeitaufwendig geworden. Darüber hinaus ist es, weil die meisten Client-Computer eigenständig sind, für Benutzer, die unterschiedliche Client-Computer an unterschiedlichen Standorten benutzen, schwierig, persönliche Voreinstellungen in Bezug auf Anwendungen und Konfigurationsdaten beizubehalten. Das heißt, dass ein Benutzer zwar persönliche Voreinstellungen als Standard auf seinem normalerweise benutzten Client-Computer installieren kann, es aber mühsam ist, diese Standardangaben auf anderen Client-Computern zu reproduzieren, ohne die Standardangaben auf jenen Computern zu ändern.

[0004] Wie oben beschrieben, ist in einer herkömmlichen Netzwerkkonfiguration der Prozess des Installierens neuer Software oder neuer Anwendungen ein statischer Prozess. In einer derartigen Konfiguration sind die Konfigurationsinformationen für jeden Client auf jeder Client-Maschine definiert. Somit definiert der Netzwerkadministrator jede Konfiguration statisch auf jedem Client. In einem herkömmlichen Computernetzwerk sind die Konfigurationsinformationen für jedes einzelne Subsystem oder jeden Client im einzelnen Client hardwarekodiert. Bei herkömmlichen Netzwerkkonfigurationen, die eigenständige, mit Netzwerkservern verbundene Clients verwenden, erfordert es die Anwendungspflege, wie z.B. das Installieren größerer Aktualisierungen von Software, bei der die Aktualisierung Zugriff auf die Konfiguration einen Subsystems voraussetzt, normalerweise außerdem, dass das Netzwerk zur Durchführung der Wartungsmaßnahme „heruntergefahren" wird.

[0005] Bei herkömmlichen Computernetzwerken, die mehrere Clients und einen Server aufweisen, wobei der Server Informationen enthält, die von einem Client aus verschiedenen Gründen benötigt werden, werden in vielen Fällen alle Daten auf dem Server, die vom Client benötigt werden oder für diesen relevant sind, vom Server zum Client bewegt. Dies kann typischerweise mit dem Bewegen großer Mengen von Daten einhergehen, von denen viele möglicherweise nicht benutzt werden oder für den Betrieb des Clients nicht nötig sind. Das Übertragen all dieser Daten zum Client ist ineffizient und steigert den Netzwerkverkehr. Darüber hinaus muss der Client ausreichend Arbeits- und Festspeicher aufweisen, um alle Informationen vom Server zu speichern, die diesen bestimmten Client betreffen. Beispielsweise können Geräte wie z.B. PDAs und Smart-Cards, die keine großen Speicherkapazitäten aufweisen, in residentem Speicher nicht alle notwendigen Informationen speichern, einschließlich der Konfigurationsinformationen, die für dieses bestimmte Gerät relevant sein könnten.

[0006] Eine weitere Komponente vieler herkömmlicher unternehmensweiter Netzwerke sind Verzeichnis- und Namensdienste. Derartige Dienste werden verwendet, um Konfigurations- und Abbildungsinformationen in Bezug auf Netzwerkbenutzer und Hardwarekomponenten zu speichern. Diese Informationen werden von Benutzern und Komponenten in ei-

nem Netzwerk benötigt, um gewisse Funktionen auszuführen, die Netzwerkdienste erfordern. Einige verbreitet genutzte Verzeichnisdienste sind DNS (Directory Name Service), der in der Windows NT®-Umgebung benutzte Dienst AD (Active Directory) und NIS in der Unix-Umgebung. Ein neuerer und leistungsfähigerer Verzeichnisdienst, der aktuellere Technologie verwendet, ist das Lightweight Directory Access Protocol oder LDAP, das in größerem Umfang in Unternehmensnetzwerken für Verzeichnisdienste benutzt wird. Fig. 1 ist ein Blockdiagramm, das darstellt, wie ein Client in einem LDAP-Verzeichnisdienst auf Daten zugreift. Ein Client-Computer **102** in einer Unternehmensumgebung **104** weist ein LDAP-Zugriffsmodul oder -Zusatzgerät **106** auf. Wenn Client **102** auf ein LDAP-Verzeichnis **108** zugreifen muss, tut er dies direkt mithilfe von Modul **106**, wie durch Linie **110** gezeigt. LDAP-Verzeichnis **108** ist in Wesentlichen ein Softwareserver, der eine Datenbank und einen (nicht gezeigten) Softwaredaemon aufweist. Das Datenbanksegment, das alle Konfigurations- und damit zusammenhängenden Daten enthält, kann funktional als eine Tabelle **112** beschrieben werden, die zwei Spalten besitzt. Eine Spalte enthält ein Attribut und die zweite Spalte enthält einen oder mehrere Ist-Werte. Das Attribut kann von beliebiger Datenkategorie sein und beispielsweise einen Benutzernamen, einen Anmeldenamen, eine Abteilung oder einen Hardware-identifikator repräsentieren. Ein Vorteil von Verzeichnis-diensten ist die Flexibilität, die sie Netzwerkadministratoren bieten, beliebige Typen von Daten zu speichern, auf die möglicherweise von Benutzern oder Komponenten im Netzwerk zugegriffen wird. Einem Attribut können ein oder mehrere Werte zugeordnet werden. Beispielsweise können einem Attribut, das benutzerspezifische Einstellungen repräsentiert, insgesamt mehrere Werte zugeordnet sein, wobei diese Werte im selben Werteintrag residieren und durch Trennzeichen beliebiger Art getrennt sind. Die Struktur und Organisation von LDAP-Verzeichnis **108** ist auf dem Feld der Computernetzwerkadministration wohl bekannt.

[0007] Auf die in diesen vorhandenen, als Legacy-Systeme bezeichneten Verzeichnisdiensten gespeicherten Informationen müsste von Client-Computern in einem Unternehmensnetzwerk zugegriffen werden können. Somit müsste jede Art von Konfigurations-Aufbewahrungsort, die die oben diskutierten Probleme bezüglich der Verwaltung von Anwendungs- und Konfigurationsdaten in expandierenden Netzwerken zu beheben sucht, Konfigurationsdaten auf Legacy-Systemen wie z.B. LDAP-Verzeichnisdiensten unterbringen oder Zugriff darauf haben. Ein Konfigurations-server, der fähig ist, Client-Computern auf effiziente Weise Konfigurations- und benutzerspezifische Daten zur Verfügung zu stellen, muss in der Lage sein, Daten mit vorhandenen Verzeichnisdiensten auszutauschen, die Konfigurationsdaten enthalten.

[0008] TRAVERSAT B: WOODWARD S: Java System Database Server, 31. Mai 1998 (1998-05-31) beschreibt die Architektur eines standardmäßigen Java-System-Datenbank-Servers (JSD-Servers). Die JSD ist weiter unten ausführlicher beschrieben. Dieses Dokument beschreibt die Verwendung einer Metaeigenschaft „.PERSISTANT.propName", die auf dem JSD-Server gespeichert ist, zum Halten der Abbildungsinformationen zwischen dem Schema eines JSD-Servers und eines LDAP-Servers (ebenfalls weiter unten ausführlicher beschrieben), um das Abrufen von Informationen vom LDAP-Server in die JSD zu gestatten.

[0009] Daher wäre es wünschenswert, ein System zu haben, das die verteilte Verwaltung von Client- und Benutzerkonfigurationen durch Speichern derartiger Konfigurationsinformationen an einem zentralen Aufbewahrungsort unterstützt. Dies würde es einem Netzwerkadministrator erlauben, Subsystem-Konfigurationen vom Server aus zu verwalten und alle Arten von Änderungen an Anwendungen von einem Server aus zu verteilen. Außerdem wäre es wünschenswert, dass der Zugriff des zentralen Aufbewahrungsortes auf Legacy-Systeme für Konfigurationsdaten zügig mit minimaler Overhead-Verarbeitung und für den Client-Computer transparent erfolgt.

[0010] Dementsprechend stellt die Erfindung unter einem Aspekt ein Verfahren des Abrufens von Werten von Konfigurationsdatenelementen von einem Lightweight Directory Access Protocol-Server (LDAP-Server) zu einem Java-basierten Konfigurationsserver bereit, wobei das Verfahren umfasst: Durchsuchen einer Speicherortabgleich-datei nach einer Übereinstimmung zwischen einem Pfad hoher Ebene in einem Java-basierten Konfigurationsserver und einer bestimmten LDAP-Adresse, Durchsuchen eines Abschnitts des LDAP-Servers nach einem oder mehreren Attributen von einem oder mehreren LDAP-Einträgen unter Verwendung der bestimmten LDAP-Adresse zur Bestimmung des Abschnitts des LDAP-Servers, der zu durchsuchen ist, Abrufen eines oder mehrerer Ist-Werte für das eine oder die mehreren Attribute und Abrufen von Werten für ein oder mehrere Schattenattribute, die dem einen oder den mehreren Attributen entsprechen, wobei die Schattenattribute anzeigen, wo in einem Java-basierten Konfigurations-serverschema ein bestimmter Ist-Wert zu platzieren ist, und Senden des einen oder der mehreren Ist-Werte zum Java-basierten Konfigurationsserver und Platzieren des einen oder der mehreren Ist-Werte im Java-basierten Konfigurationsserverschema als Konfigurationsdatenelemente basierend auf den Werten für das eine oder die mehreren Schattenattribute, sodass der eine oder die mehreren Ist-Werte Client-Computern in einer Java-Betriebssystemumgebung verfügbar gemacht werden.

[0011] Die Erfindung stellt Computerprogrammcode

nach Anspruch 8 und ein Computerprogrammprodukt nach Anspruch 9 bereit.

KURZBESCHREIBUNG DER ZEICHNUNGEN

[0012] Die Erfindung ist besser zu verstehen unter Hinzuziehung der nachstehenden Beschreibung in Verbindung mit den beiliegenden Zeichnungen, wobei

[0013] Fig. 1 ein Blockdiagramm ist, das darstellt, wie ein Client in einem LDAP-Verzeichnisdienst auf Daten zugreift.

[0014] Fig. 2A ein hierarchisches Blockdiagramm eines LDAP-Verzeichnisbaums ist.

[0015] Fig. 2B eine Veranschaulichung eines Namensschemas, das die hierarchische Struktur von Fig. 2A in einem LDAP-Verzeichnisdienst widerspiegelt, und zugeordneter Attribute ist.

[0016] Fig. 3 ein schematisches Diagramm ist, das Komponenten eines Computernetzwerks darstellt, das ein systemweites Datenschema gemäß einer Ausführungsform der vorliegenden Erfindung aufweist.

[0017] Fig. 4 ein Blockdiagramm ist, das eine Struktur eines JSD-Serverschemas gemäß einer Ausführungsform der vorliegenden Erfindung zeigt.

[0018] Fig. 5 ein schematisches Diagramm ist, das eine hierarchische Struktur eines Namensraumes MACHINE (Maschine) in einem Serverschema gemäß einer Ausführungs-form der vorliegenden Erfindung darstellt.

[0019] Fig. 6 ein schematisches Diagramm ist, das einen Namensraum USER (Benutzer) gemäß einer Ausführungsform der vor-liegenden Erfindung darstellt.

[0020] Fig. 7 ein Ablaufdiagramm eines Prozesses zum Zugreifen auf Daten in einem LDAP-Verzeichnis in einer Java-System-Datenbank-Umgebung gemäß einer Ausführungsform der vorliegenden Erfindung ist.

[0021] Fig. 8 ein Ablaufdiagramm eines Prozesses zum Initialisieren der Server-JSD ist, unter Verwenden eines LDAP-Verzeichnisdienstes, gemäß einer Ausführungsform der vorliegenden Erfindung.

[0022] Fig. 9A eine Darstellung ist, die ein Format eines benutzerspezifischen Konfigurationsdaten-Blattes im JSD-Server und einen Benutzereintrag einschließlich Attributen in einem LDAP-Verzeichnisserver zeigt, gemäß einer Ausführungsform der vorliegenden Erfindung.

[0023] Fig. 9B eine Darstellung eines Formats eines LDAP-Metaverzeichnisses ist, das im LDAP-Server enthalten ist, gemäß einer Ausführungsform der vorliegenden Erfindung.

[0024] Fig. 9C eine Darstellung eines Formats einer Abbildungskomponente eines Pfades hoher Ebene gemäß einer Ausführungsform der vorliegenden Erfindung ist.

[0025] Fig. 10 ein Ablaufdiagramm eines Prozesses zum Abrufen eines Konfigurationsdatenelements vom LDAP-Verzeichnis-dienst ist, wenn das Datenelement auf dem JSD-Server nicht verfügbar ist, gemäß einer Ausführungsform der vorliegenden Erfindung.

[0026] Fig. 11 ein Ablaufdiagramm eines Prozesses ist, in dem ein LDAP-Eintrag auf einen JSD-Eintrag abgebildet wird, gemäß einer Ausführungsform der vorliegenden Erfindung.

[0027] Fig. 12 ein Blockdiagramm eines typischen Computersystems ist, das für die Implementierung einer Ausführungsform der vorliegenden Erfindung geeignet ist.

AUSFÜHRLICHE BESCHREIBUNG

[0028] Ausführlich wird nun Bezug genommen auf eine bestimmte Ausführungsform dieser Erfindung. Ein Beispiel der bestimmten Ausführungsform ist in den beiliegenden Zeichnungen dargestellt. Obgleich die Erfindung in Verbindung mit einer bestimmten Ausführungsform beschrieben wird, versteht es sich, dass nicht beabsichtig ist, die Erfindung auf eine bestimmte Ausführungsform zu beschränken. Es ist im Gegenteil beabsichtigt, Alternativen, Modifikationen und Äquivalente abzudecken, wie sie im Umfang der Erfindung, wie durch die beigefügten Ansprüche definiert, enthalten sein können.

[0029] Verfahren und Systeme zum Abbilden eines Attributs oder Eintrags in einem LDAP-Verzeichnisdienst auf ein Konfigurationsserverschema sind in den verschiedenen Figuren beschrieben. In der beschriebenen Ausführungsform enthält der Konfigurationsserver eine Softwarekomponente, die als eine Java-System-Datenbank („JSD") bezeichnet wird. Die JSD der vorliegenden Erfindung ist ausführlicher in der US-Patentschrift 6,052,720 beschrieben, die am 14. Mai 1998 unter dem Titel „A Generic Schema for Storing Configuration Information on a Server Computer" (Generisches Schema zum Speichern von Konfigurationsinformationen auf einem Server-Computer) eingereicht wurde. Wie weiter unten ausführlicher beschrieben, ist das JSD-Serverschema eine Baumstruktur, die wohldefinierte Knoten und „Blätter" aufweist und eine oder mehrere Eigenschaften und zugehörige Datenwerte enthält. Somit kann

der Speicherort jedes einzelnen Datenelements im JSD-Serverschema als eine Reihe von Knotennamen übermittelt werden, die durch Schrägstriche getrennt sind. Die Fähigkeit des Definierens eines Pfades für jede Eigenschaft in einem JSD-Server, wie in Fig. 9A gezeigt, gestattet ein „Abbilden" eines LDAP-Verzeichnisdiensteintrags oder -attributs auf die JSD-Server-Eigenschaft der vorliegenden Erfindung.

[0030] Obgleich die Struktur und Verwendung von Netzwerk-Verzeichnisdiensten, die das Lightweight Directory Access Protocol (LDAP) – ein Protokoll eines offenen Standards, das über TCP/IP läuft – benutzen, auf dem Fachgebiet wohl bekannt sind, ist es hilfreich, einige bestimmte Merkmale von LDAP zu beschreiben, die für die Abbildungsmerkmale der vorliegenden Erfindung besonders relevant sind. Ein Netzwerk-LDAP-Verzeichnisdienst speichert typischerweise Konfigurationsdaten für ein Netzwerk, die im Allgemeinen deskriptiver und attributbasierter sind als Daten, die in herkömmlichen Datenbanken gespeichert sind. Im Allgemeinen werden sie viel häufiger gelesen als geschrieben. Verzeichnisse sind darauf abgestimmt, schnell auf Nachschlag- oder Suchvorgänge mit hohem Volumen reagieren zu können. Konfigurationsdaten werden allgemein als Daten beschrieben, die zum Konfigurieren eines Systems benutzt werden, wie z.B. eines Client-Computers, und als Daten, die sich auf Benutzerprofile beziehen, zum Einrichten einer Benutzerumgebung ungeachtet des Client-Computers, an dem sich ein Benutzer an einem Netzwerk anmeldet. Das JSD-Serverschema speichert ebenfalls Konfigurationsdaten, weist aber nicht die Skalierbarkeit eines LDAP-Servers auf. Somit ist es effizienter, große Datenmengen auf einem LDAP-Server zu speichern, wodurch sie für alle Benutzer im Netzwerk verfügbar gemacht werden, statt sie sämtlich auf dem JSD-Server zu speichern.

[0031] Ein Merkmal des LDAP-Servers sind die absoluten und relativen Namenskonventionen, die zum Definieren der Speicherorte von Datenelementen verwendet werden, die als Attribute oder Schlüssel bezeichnet werden. Von besonderer Bedeutung sind die absoluten Namen, die zum Auffinden der Attribute im LDAP-Verzeichnis benutzt werden. Ein absoluter Name beinhaltet eine Reihe „kennzeichnender Namen" (Distinguished Names), die den Knoten im JSD-Server ähnlich sind. Generell basiert das LDAP-Modell auf Einträgen. Ein Eintrag ist eine Zusammenstellung von Attributen. Ein derartiger Eintrag wird als ein kennzeichnender Name (Distinguished Name, „DN") bezeichnet. Ein Attribut kann einen oder mehrere Werte aufweisen und zu einem bestimmten Typ gehören. Typen sind typischerweise mnemonische Zeichenketten wie z.B. un oder u für „User Name" (Benutzername), ml for „Email address" (E-Mail-Adresse) oder o für „Organization" (Organisation). Jeder Typ weist eine Zusammenstellung von Attributen auf. Beispielsweise kann un Attribute wie z.B. unter anderem den üblichen Namen, den Nachnamen und den Anmeldenamen aufweisen. Der DN o kann die Attribute Mail und Fax aufweisen, denen jeweils ein oder mehrere Werte folgen.

[0032] Einträge in LDAP sind in einer hierarchischen baumähnlichen Struktur angeordnet, die typischerweise organisatorische Grenzen widerspiegelt. Beispielsweise erscheinen Einträge, die Länder repräsentieren, oft an der Spitze des Baumes unter dem DN c (Country), gefolgt z.B. von Einträgen (bu) (Business Unit), die Unternehmensbereiche repräsentieren, denen speziellere Einträge folgen, die alles von Druckern über Client-Computer bis zu Netzwerkbenutzern repräsentieren. Fig. 2A ist ein hierarchisches Blockdiagramm eines LDAP-Verzeichnisbaums. Eine detailliertere Beschreibung des offenen LDAP-Standards, seiner Datenorganisation und Verwendung kennzeichnender Namen enthalten der Request for Comments („RFC") Nummer 1777 mit dem Titel „The Lightweight Directory Access Protocol" (Das LDAP) von Wengyik Yeong, Tim Howes und Steve Kille, März 1995, und RFC 1779, „A String Representation of Distinguished Names" (Eine Zeichenketten-Darstellung kennzeichnender Namen) von Steve Kille, März 1995.

[0033] Ein Verzeichnisbaum **200**, der in Fig. 2A gezeigt ist, weist drei Knoten auf, die einem DN der höchsten Ebene für das Land entsprechen, das von dem Kürzel c repräsentiert wird. Ein Knoten **202** repräsentiert die USA. Das Kürzel c wird auch als ein Typ bezeichnet. Knoten **202** weist zwei vom ihm abstammende Knoten auf, **204** und **206**, die zum DN o gehören, der die Organisation repräsentiert. Knoten **204** stellt Netscape als eine Organisation dar. Netscape, das den Typ o aufweist, besitzt in diesem Beispiel zwei gezeigte Attribute, „mail" und „fax", gefolgt von deren jeweiligen Werten. Knoten **206** stellt eine andere Organisation, Sun, dar, mit zwei von dieser abstammenden Knoten. Knoten **206** kann – und sehr wahrscheinlich ist dies auch der Fall – eine Liste von Attributen (alle des Typs o) ähnlich Knoten **204** aufweisen, die jedoch nicht in der Figur gezeigt sind. Der DN der nächsten Ebene ist bu, der die Art des Geschäftsbereichs repräsentiert. Unter dem Sun-Knoten **206** gibt es zwei Knoten, beide des Typs bu. Ein Knoten **208** stellt den Unternehmensbereich „SunSoft" dar. SunSoft weist ebenfalls eine Zusammenstellung von Attributen auf, alle des Typs bu, die in Fig. 2 nicht gezeigt sind. Unter Knoten **208** befindet sich ein Knoten **210** für einen Benutzer Jonathan A. Smith, der als DN u repräsentiert wird. Der SunSoft-Unternehmensbereich kann viele Knoten ähnlich Knoten **210** aufweisen, die sämtliche Netzwerkbenutzer in diesem Unternehmensbereich darstellen. Der Typ u weist eine Liste von Attributen **212** auf. Jedem Attribut folgen ein oder mehrere Werte. In vielen Implementierungen von LDAP müssen diese Werte

entweder in Zeichenketten- oder binärer Form vorliegen. In dem in Fig. 2A gezeigten Beispiel gibt es unterhalb des DN u keine weiteren DNs. Ein Vorteil mit LDAP implementierter Verzeichnisdienste ist die Flexibilität zum Speichern aller Typen von Informationen, wie z.B. Daten, die Hardwarekomponenten betreffen, Benutzereinstellungen, Startdaten usw.

[0034] Ein LDAP-Verzeichnis beinhaltet typischerweise mehrere Kontexte. Wird ein neuer Kontext erstellt, wird ein absoluter Name oder eine Hierarchie von DNs, wie in Fig. 2B beschrieben, für diesen Kontext definiert. Ein Kontext wird definiert, um auf eine bestimmte Funktion zu zielen. Einige Beispiele für Kontexte sind „Starten des Clients", „benutzerspezifische Voreinstellungen", „Finanzberichte" oder „Daten der Abteilung Technik", um einige zu nennen. Kontexte können als Segmente hoher Ebene gesehen werden, die den Datenbank-Anteil eines LDAP-Verzeichnisses bilden. Beispielsweise werden Zugriffsbeschränkungen für Clients auf Daten in einem LDAP-Verzeichnis auf der Kontextebene gesetzt.

[0035] Für jeden Kontext wird (typischerweise von einem Netzwerkadministrator) ein Absolutnamensschema definiert, wenn ein Kontext erstellt wird. Das Absolutbenennungsschema oder die -konvention unter LDAP ist eine Liste von DNs. Fig. 2B ist eine Veranschaulichung eines Namensschemas und zugeordneter Attribute, das die hierarchische Struktur von Fig. 2A in einem LDAP-Verzeichnisdienst widerspiegelt. Die Benennungskonfiguration oder -hierarchie beginnt mit dem kennzeichnenden Namen „c=US" **214** auf der rechten Seite des Namensschemas. Der DN c kann einen beliebigen aus einer Anzahl von Werten aufweisen, die unterschiedliche Länder repräsentieren. Der nächste kennzeichnende Name o=Sun **216** repräsentiert die nächsttiefere Ebene in der Hierarchie. In diesem Beispiel repräsentiert DN o „Organisation" und kann Zeichenkettenwerte aufweisen, die andere Organisationen oder Firmen repräsentieren. Die DNs zur Linken gehen mehr ins Einzelne: bu repräsentiert Unternehmensbereich **218** und u repräsentiert einen Benutzer **220**. Jeder kennzeichnende Name oder Typ weist einen Satz von Attributen auf. Beispielsweise weist der kennzeichnende Name u=Jonathan A. Smith **220** einen Satz spezifischer Attribute **212** auf, die erstmals in Fig. 2A beschrieben sind. Die gesamte Zeichenkette von DNs **222** wird als eine absolute Adresse oder manchmal ein vollständiger DN bezeichnet.

[0036] Wie oben erwähnt, ist ein Anteil von Konfigurationsdaten für Client-Computer, Benutzer und andere Komponenten in einem Netzwerk im JSD-Serverschema gespeichert. Dies steht im Kontrast zu herkömmlichen Netzwerken, in denen Konfigurationsdaten für einen Client auf der Client-Maschine hardwarekodiert oder gespeichert sind. Das JSD-Serverschema gestattet es einem Netzwerkadministrator, Konfigurationsinformationen für jeden der Computer in einem Unternehmensnetz von einem zentralen Aufbewahrungsort aus zu verwalten. Somit können jedwede Software-Aktualisierungen, Versionsaktualisierungen oder Installationen neuer Anwendungen, die Kenntnis über und Zugriff auf eine Subsystem-Konfiguration (z.B. des Client-Computers) voraussetzen, vom zentralen Aufbewahrungsort aus implementiert und zum einzelnen Subsystem verteilt werden. Benutzer auf Client-Computern müssen beispielsweise Anwendungen nicht beenden, und darüber hinaus muss das Netzwerk nicht zur Wartung heruntergefahren werden, um die neue Aktualisierung oder Version der Anwendung zu installieren oder zu verteilen

[0037] Fig. 3 ist ein schematisches Diagramm, das Komponenten eines Computernetzwerks darstellt, das ein systemweites Datenschema gemäß einer Ausführungsform der vorliegenden Erfindung aufweist. In der beschriebenen Ausführungsform ist das systemweite Datenschema als eine Java-System-Datenbank oder JSD **301** implementiert, die aus einem Clientschema **303** besteht, das auf einer Client-Maschine **305** als Teil von Netzwerk **307** residiert. Ein JSD-Serverschema **311** residiert auf einem Server-Computer **309**, Teil von Netzwerk **307**. Wie oben erwähnt, ist es JSD-Serverschema **311** auf Server **309** („JSD-Server"), das mit einem LDAP-Verzeichnisdienst kommuniziert. Daher wird JSD-Serverschema **311** weiter unten ausführlicher beschrieben.

[0038] Fig. 4 ist ein Blockdiagramm, das eine Struktur eines JSD-Serverschemas gemäß einer Ausführungsform der vorliegenden Erfindung zeigt. Es zeigt Server-Computer **309** und Serverschema **311** aus Fig. 3 in detaillierterer Darstellung. An der Spitze eines n-fach verzweigten Baumes ist ein Rooteintragsknoten **401** vorhanden, der in der bevorzugten Ausführungsform mit „CONFIG" bezeichnet ist. In dem Serverschema gibt es zwei Namensräume. Bereich **403** repräsentiert einen Namensraum MACHINE, der einen Maschinenknoten **305** aufweist. Bereich **407** repräsentiert einen Namensraum USER, der einen Benutzerknoten **409** aufweist.

[0039] In der beschriebenen Ausführungsform beinhaltet der Namensraum MACHINE **403** drei Kategorien: platform (Plattform) **411**, identifier (Identifikator) **413** und profile (Profil) **415**. Unter platform **311** beispielsweise gibt es eine Reihe von Einträgen, die auf bestimmte Computerhersteller verweisen. In anderen Ausführungsformen kann der Namensraum MACHINE **403**, abhängig von der Plattform und den Anforderungen des Netzwerks, mehr oder weniger Unterkategorien aufweisen. Dies ist in Fig. 5 detaillierter dargestellt.

[0040] Fig. 5 ist ein schematisches Diagramm, das eine hierarchische Struktur eines Namensraumes

MACHINE **403** in Serverschema **311** darstellt. Unter Kategorie platform **411** sind generell herstellerspezifische Unterbereiche **501** und **503** vorhanden. Die Anzahl von Einträgen auf dieser Ebene kann beispielsweise von der Anzahl von Komponenten von unterschiedlichen Computerherstellern abhängen, die im Netzwerk verwendet werden. Unter einem bestimmten Hersteller wie z.B. Sun Microsystems gibt es zwei Einträge **505** und **507**, die auf ein bestimmtes, von Sun Microsystem hergestelltes Computermodell oder einen -typ verweisen. Beispielsweise gibt es unter Sun den Computertyp JDM1. Unter jedem Computermodell sind Blätter vorhanden, wie z.B. Knoten **509**, die Anwendungs-Konfigurationsdaten für diesen bestimmten Typ von Computer spezifizieren. Anwendungs-Konfigurationsdaten in in den Blatteinträgen enthalten alle möglichen Konfigurationen, die auf den bestimmten Computer anzuwenden sind, der im übergeordneten Eintrag (z.B. Knoten **507**) angegeben ist.

[0041] Unter der Kategorie identifier **413** sind Einträge vorhanden, die einen eindeutigen Identifikator für jeden Computer im Netzwerk **307** von Fig. 3 enthalten, wie z.B. in Knoten **511** gezeigt. In der beschriebenen Ausführungsform wird die Adresse der Medienzugangssteuerung (Media Access Control, MAC) für jeden Computer als ein eindeutiger Identifikator verwendet. Blatt **513** unter Client-Identifikator **511** enthält spezielle Anwendungs-Konfigurationsinformationen für diesen bestimmten Computer. Die Konfigurationsdaten **513** sind von den Konfigurationsdaten **509** unter Kategorie platform **411** insofern zu unterscheiden, dass die Daten **513** unter identifier **413** einen speziellen Computer betreffen, wie er von einem bestimmten Benutzer konfiguriert wurde. In der beschriebenen Ausführungsform gibt es (nicht gezeigte) Querverweise zwischen den eindeutigen Identifikatoren **413** unter der Identifikator-Kategorie und Einträgen unter Kategorie platform **411**. Das heißt, es gibt einen Verweis von einem speziellen Identifikator auf einen bestimmten Typ von Computer. Dies gestattet es dem Server festzustellen, auf welche Plattform oder welchen Typ von Computer sich ein bestimmter eindeutiger Identifikator bezieht.

[0042] Unter Kategorie profile **415** sind Einträge vorhanden, die bestimmte Kategorien oder Verwendungen von Computern im Netzwerk beschreiben. Die Konfigurationsinformationen für die einzelnen Profile, die beispielsweise Abteilungen innerhalb einer Firma beschreiben können, sind unter Kategorie profile **415** enthalten. Beispiele sind an den Knoten **515** und **517** als Profile für Finanzwesen und Vertrieb gezeigt. Unter Knoten Finanzwesen **515** befinden sich anwendungsspezifische Daten **519**, die Daten bezüglich des Profils Finanzwesen enthalten. Ähnlich den Verweisen von den eindeutigen Identifikatoren auf die platform-Einträge gibt es erforderlichenfalls auch einen Verweis vom speziellen Identifikator-Blatt **513** auf den profile-Eintrag des Knotens **519**. Das heißt: Weist ein bestimmter Computer ein gewisses Profil auf, z.B. ein Computer, der in der Buchführungsabteilung verwendet wird, oder ein Computer, der ausschließlich als Terminal am Empfang dient, gibt es einen Verweis vom Identifikator dieses Computers zum passenden profile-Eintrag.

[0043] Fig. 6 ist ein schematisches Diagramm, das einen Namensraum USER (Benutzer) gemäß einer Ausführungsform der vorliegenden Erfindung darstellt. In der beschriebenen Ausführungsform weist der Namensraum USER **407** zwei Kategorien auf, users (Benutzer) und groups (Gruppen). Kategorie users **417** repräsentiert Namen einzelner Benutzer im Netzwerk **307**, wie z.B. in den Knoten **601**, **603** und **605** gezeigt. Unter dem individuellen Knoten jedes Benutzers gibt es spezifische Konfigurationsdaten, die persönliche Voreinstellungen dieses individuellen Benutzers enthalten und im Blatt **607** gezeigt sind. Beispielsweise kann in einer Textverarbeitungs-Anwendung eine bestimmte Benutzervor-einstellung eine Standardschriftart und Formateinstellungen für Dokumente beinhalten, Diese Kategorie gestattet es einem individuellen Benutzer, einen beliebigen Computer im Netzwerk **307** zu benutzen, wobei die persönliche Konfiguration dieses Benutzers (also die persönlichen Einstellungen) diesem Computer bekannt sind. Wenn beispielsweise der Benutzer ein Textverarbeitungsprogramm aufruft, werden die Voreinstellungen des Benutzers als Standard genommen statt der Standardvorgaben des normalen Benutzers jenes Computers.

[0044] Die andere Kategorie im Namensraum USER ist die Kategorie groups. Diese Kategorie enthält Einträge bezüglich bestimmter Gruppen von Benutzern. Groups **419** kann vielfältige Kategorien enthalten, z.B. Abteilungen innerhalb einer Firma oder Kategorien, die Mitarbeiter einer Firma von anderen Mitarbeitern unterscheiden, wie z.B an den Knoten **609** und **611** gezeigt. In der beschriebenen Ausführungsform gibt es gegebenenfalls Verweiszeiger zwischen individuellen Benutzern **603** und **605** unter der Kategorie users **417** zu einer oder mehreren groups.

[0045] Fig. 7 ist ein Ablaufdiagramm eines Prozesses zum Zugreifen auf Daten in einem LDAP-Verzeichnis von einem Konfigurationsserver, der in einer Java-System-Datenbank-Umgebung betrieben wird, gemäß einer Ausführungsform der vorliegenden Erfindung. Das Ablaufdiagramm der Fig. 7 verwendet einen Konfigurationsserver, der mit einem JSD-Serverschema konfiguriert ist, wie oben in Fig. 3 – Fig. 6 beschrieben, um Client-Netzwerk- und Benutzerkonfigurationsdaten aufzubewahren. Im Schritt **702** importiert der JSD-Server Konfigurationsdaten vom LDAP-Verzeichnisserver beim Starten des JSD-Servers oder speichert sie im Cache-Speicher. In diesem Schritt geht das JSD-Serverschema gegebenenfalls

eine „Anfangspopulation" von benötigten Konfigurationsdaten vom LDAP-Verzeichnisdienst durch. Dieser Schritt ist in Fig. 8 detaillierter beschrieben. In Schritt **704** startet ein Client-Computer, und ein bestimmter Benutzer meldet sich an. Der Client-Computer greift auf den JSD-Server zu, um seine Konfigurationsdaten und Konfigurationsdaten für den bestimmten Benutzer abzurufen. Ein Protokoll für das Austauschen von Daten zwischen einem Client-JSD-Schema und dem Server-JSD-Schema ist ausführlicher in der US-Patentschrift 6,119,157 beschrieben, die am 14. Mai 1998 unter dem Titel „A Protocol for Exchanging Configuration Data in a Computer Network" (Protokoll zum Austauschen von Konfigurationsdaten in einem Computernetzwerk) eingereicht wurde.

[0046] Im Schritt **706** stellt der JSD-Server während normaler Benutzeraktivität und normalen Client-Betriebs fest, ob sich Daten, die von einem Client angefordert oder benötigt werden, im JSD-Serverschema befinden. Sind die Konfigurationsdaten, die vom Client benötigt werden, auf dem JSD-Server verfügbar, werden die Daten abgerufen, und die Steuerung geht zu Schritt **710** über, in dem die Daten zum Client übertragen werden. Sind sie auf dem JSD-Server nicht verfügbar, geht die Steuerung zu Schritt **708** über, in dem die benötigten Daten auf dem LDAP-Verzeichnisserver lokalisiert werden. Dieser Prozess ist in Fig. 10 detaillierter beschrieben. Ein Konfigurationsdatenelement kann in einem JSD-Server dadurch lokalisiert werden, dass der entsprechende Namensraum – entweder MACHINE oder USER – identifiziert und den Kategorien gefolgt wird, bis das korrekte Blatt erreicht ist, das das bestimmte Konfigurationsdatenelement enthält. Beispielsweise kann ein „Pfad" lauten:

[0047] CONFIG/MACHINE/identifier/(MAC-Adresse)/(spezielle Konfigurationseigenschaften)

[0048] Durch einen weiter unten beschriebenen Abbildungsprozess wird ein Teil dieses Pfades verwendet, um Daten vom LDAP-Server zum JSD-Serverschema zurückzugeben. Die Daten vom LDAP-Server werden abgerufen und zum JSD-Serverschema übertragen. Im Schritt **710** werden die Konfigurationsdaten zum Client übertragen, und der Prozess ist abgeschlossen.

[0049] Fig. 8 ist ein Ablaufdiagramm eines Prozesses zum Initialisieren eines JSD-Servers, der einen LDAP-Verzeichnisdienst verwendet, gemäß einer Ausführungsform der vorliegenden Erfindung. Es zeigt detaillierter Schritt **702** von Fig. 7. In Schritt **802** bestimmt der JSD-Server den passenden Kontext im LDAP-Verzeichnis, von dem Daten zu importieren sind, um den JSD-Server zu starten. In der beschriebenen Ausführungsform weist das LDAP-Verzeichnis einen Kontext unter dem Namen „JSD" oder einem anderen geeigneten Namen auf, der all die Daten enthält, die ein JSD-Server zum Starten benötigt, aber noch nicht hat. In der beschrieben Ausführungsform werden die meisten der Daten im JSD-Namensraum vom DAP-Server beschickt. In anderen Ausführungsformen kann ein Teil der USER-Daten bereits auf dem JSD-Server residieren. In anderen Ausführungsformen können JSD-Server-Startdaten über mehr als einen Kontext im LDAP-Verzeichnis verteilt sein. In noch anderen Ausführungsformen kann der JSD-Server bereit die meisten oder alle Start-Konfigurationsdaten aufweisen.

[0050] In Schritt **804** ruft der JSD-Server alle Einträge im Kontext „JSD" im LDAP-Verzeichnis ab. In der beschriebenen Ausführungsform ruft der JSD-Server alle Einträge im Kontext ab. In anderen Ausführungsformen können nach Bedarf vom JSD-Server mehr oder weniger Einträge in einem oder mehreren Kontexten abgerufen werden. In Schritt **806** werden die JSD-Pfade abgerufen, die den Attributen in den LDAP-Einträgen entsprechen. Dieser Prozess ist in Fig. 11 detaillierter beschrieben. In Schritt **808** werden die Konfigurationsdaten in dem JSD-Eintrag gespeichert, der in Schritt **806** ermittelt wurde, womit in dieser Phase der Prozess abgeschlossen ist.

[0051] Fig. 9A ist eine Darstellung, die ein Format eines benutzerspezifischen Konfigurationsdaten-Blattes im JSD-Server und einen Benutzereintrag einschließlich Attributen in einem LDAP-Verzeichnisserver zeigt, gemäß einer Ausführungsform der vorliegenden Erfindung. Die benutzerspezifischen Konfigurationsdaten im JSD-Server wurden erstmals als Knoten **607** in Fig. 6 beschrieben. Fig. 9A zeigt eine Liste von JSD-Eigenschaften **902** für Benutzer „Jon", der in Knoten **601** von Fig. 6 dargestellt ist. Liste **902** beinhaltet in Form eines Beispiels die Anmelde-ID (logon ID), Adresse (address), E-Mail-Adresse (email address), Benutzer-ID (user ID) und Abteilung des Benutzers. Die speziellen Benutzerbestätigungs-Datenelemente in Blatt **607** hängen von den spezifischen Bedürfnissen des Benutzers, des Netzwerks und der Anwendungen ab und sind nicht auf die Eigenschaften beschränkt, die in Liste **902** gezeigt sind.

[0052] Ebenfalls in Fig. 9A dargestellt ist ein LDAP-Eintrag **904** des Typs oder DNs u für den Benutzer Jonathan A. Smith. Er ist eine Erweiterung oder Modifikation von Eintrag **210**, der in Fig. 2A gezeigt ist. Attributliste **212** ist jetzt als eine erweiterte Attributliste **906** dargestellt, die zusätzliche „Schattenattribute" für Anmeldung (logon) **908**, Mail (mail) **910** und Computer (computer) **912** enthält. Diese Schattenattribute werden während des „Anfangspopulations"-Startens des JSD-Servers verwendet, ein Prozess, der in Fig. 11 detaillierter beschrieben ist. Kurz gesagt, gestatten es die Schattenattribute dem LDAP-Server, die Anfangsbeschickung des JSD-Serverschemas zügig durchzuführen, die erstmals in

den Schritten **806** und **808** von Fig. 8 beschrieben wurde. Jene Attribute im LDAP-Server, die über eine korrespondierende Eigenschaft im JSD-Serverschema verfügen, erhalten einen „Schatten". In der beschriebenen Ausführungsform enthält ein Schattenattribut den Namen der korrespondierenden JSD-Eigenschaft, den JSD-Pfad, mit dem die Eigenschaft aufgefunden werden kann, und den Namen der Java-Klasse, die der Eigenschaft zugeordnet ist.

[0053] In der beschriebenen Ausführungsform beginnt ein Schattenattribut mit einem Punkt (.). In anderen Ausführungsformen können andere Sonderzeichen oder Markierungen verwendet werden, um ein Schattenattribut zu kennzeichnen. Auf den Punkt folgt der Name des LDAP-Attributs, das einen Schatten erhält, z.B. „logon" oder „mail". Auf den LDAP-Attributnamen folgt der JSD-Eigenschaftsname. Beim Schattenattribut **908** lautet der Eigenschaftsname „logon_ID", welches die erste JSD-Eigenschaft in Liste **902** ist. Die Reihenfolge der Eigenschaften oder der Schattenattribute ist unerheblich und spielt in der beschriebenen Ausführungsform keine funktionale Rolle. Nach dem JSD-Eigenschaftsnamen steht der JSD-Pfad. In Attribut **908** liegt der Pfad im Namensraum USER und endet mit dem Benutzer „Jon". Auf den Pfadnamen folgt der Java-Klassenname für logon ID. In Attribut **912** ist die JSD-Eigenschaft client_ID, die einem Computer-Seriennummer-Typ-Attribut unter LDAP entspricht. Weil der Pfad für client_ID sich auf eine bestimmte Maschine bezieht, liegt er im Namensraum MACHINE.

[0054] Wie ausführlicher in Fig. 11 beschrieben, werden, wenn das JSD-Serverschema anfangs beschickt wird, die Schattenattribute verwendet, um zügig genau zu identifizieren, wo im JSD-Serverschema ein bestimmtes Konfigurationsdatenelement gespeichert sein sollte. Den Ist-Wert des Datenelements erhält man vom regulären „Nicht-Schatten"-Attribut im Eintrag. In der beschriebenen Ausführungsform werden Schatteneinträge manuell von einem Netzwerkcomputerverwalter erzeugt, der mit dem JSD-Serverschema vertraut ist. Eine Person mit Kenntnissen über das JSD-Schema und Java-Klassennamen ist in der Lage, jene Attribute unter LDAP zu identifizieren, die über korrespondierende Eigenschaften im JSD-Schema verfügen. Eine solche Person kann die Schattenattribute in die LDAP-Einträge einfügen. In der beschriebenen Ausführungsform kann der JSD-Server auf die Schattenattribut zugreifen, und der JSD-Server ist berechtigt, die Schattenattribute zu lesen.

[0055] Fig. 9B ist eine Darstellung eines Formats eines LDAP-Metaverzeichnisses, das im LDAP-Server enthalten ist, gemäß einer Ausführungsform der vorliegenden Erfindung. LDRP-Metaverzeichnis **914** ist eine Liste von LDAP-Typen oder DNs, wie z.B. c, bu und u, gefolgt von einer Liste jedes zugeordneten Attributs. Ein Metaverzeichnis wird verwendet, wenn der JSD-Server ein Konfigurationsdatenelement vom LDAP-Server abrufen muss. Es gestattet einem LDAP-Server, schnell zu ermitteln, ob ein bestimmter Typ ein gewisses Attribut aufweist. Es ist auch für andere Legacy-Systeme im Netzwerk nützlich, die Zugriff auf Daten im selben LDAP-Kontext benötigen. Diese Systeme können Metaverzeichnis **914** benutzen, um exakt zu ermitteln, welche Attribute eines bestimmten Typs benötigt werden, bevor auf den Eintrag zugegriffen wird. Beim Abrufen des Wertes von dem Eintrag kann das Legacy-System die Informationen, die es von Metaverzeichnis erhalten hat, dazu benutzen, nur Werte für jene Attribute zu extrahieren, nach denen vom Legacy-System gesucht wird. Im Wesentlichen benutzt es das Metaverzeichnis als eine Maskierung für den LDAP-Eintrag. Angesichts des Hinzufügens von Schattenattributen in den Einträgen ist diese Maskierungsfunktion insofern besonders nützlich, als sie die Wahrscheinlichkeit von Fehlern beim Abrufen von Werten für Attribute verringert, die über Schattenattribute verfügen.

[0056] Fig. 9C ist eine Darstellung eines Formats einer Abbildungskomponente eines Pfades hoher Ebene gemäß einer Ausführungsform der vorliegenden Erfindung. Eine Abbildungskomponente eines Pfades hoher Ebene **916** enthält eine Abbildung zwischen jedem JSD-Pfad hoher Ebene und einem korrespondierenden hierarchischen LDAP-Pfad, der aus kennzeichnenden Namen besteht. In der beschriebenen Ausführungsform beginnt ein Pfad hoher Ebene im JSD-Serverschema mit dem Rooteintrag, der mit CONFIG bezeichnet ist, gefolgt von einem der beiden primären Namensräume MACHINE oder USER, und endet mit einer der fünf Kategorien unter den beiden Namensräumen. Diese Abbildung wird verwendet, um die Suche einzuschränken, die im LDAP-Server erfolgt, wenn der JSD-Server, wie erstmals bei Schritt **708** in Fig. 7 gezeigt, ein Datenelement anfordert, über das er nicht verfügt. Ihre Rolle beim Abrufen von Datenelementen im LDAP-Server ist in Fig. 10 ausführlicher beschrieben. In der beschriebenen Ausführungsform ist die Abbildung von Pfaden hoher Ebene eine Komponente in einem Netzwerkcomputer-Verwaltungstool, das den LDAP-Verzeichnisdienst verwaltet und in der Lage ist, mit der JSD zu kommunizieren (also „JSD-bewusst" ist). Wenn daher die JSD ein bestimmtes Konfigurationsdatenelement vom LDAP-Server anfordert, greift sie zuerst auf die Abbildung der Pfade hoher Ebene zu, um zu ermitteln, welche „Verzweigung" der hierarchischen LDAP-Struktur mithilfe der standardmäßigen LDAP-Suchfunktionen zu durchsuchen ist. Das Abbilden zwischen JSD-Schemapfaden (die sämtlich wohldefiniert sind) und den DNs hoher Ebene im LDAP-Server wird von einem Netzwerkcomputeradministrator vorgenommen, der mit beiden Schemata vertraut ist.

[0057] Fig. 10 ist ein Ablaufdiagramm eines Prozesses zum Abrufen eines Konfigurationsdatenelements vom LDAP-Verzeichnisdienst, wenn das Datenelement auf dem JSD-Server nicht verfügbar ist, gemäß einer Ausführungsform der vorliegenden Erfindung. Fig. 10 beschreibt Schritt **708** von Fig. 7 detaillierter. In Schritt **1002** löst der JSD-Server eine Suche nach einer Übereinstimmung zwischen Pfaden hoher Ebene des JSD-Serverschemas und der hierarchischen LDAP-Struktur mithilfe von Tabelle **916** von Fig. 9C aus. Beispielsweise beinhaltet eine Benutzeranfrage auf einem Client-Computer nur den Nachnamen eines bestimmten Benutzers Jon Smith. Das JSD-Serverschema weist keine Eigenschaft auf, die nur den Nachnamen eines Benutzers bereitstellt. Es ermittelt dies durch Durchqueren des Pfades CONFIG/USER/users/Jon/, und Überprüfen der benutzerspezifischen Konfigurationsdaten auf eine Eigenschaft, die nur einen Nachnamen aufweist.

[0058] In Schritt **1002** erfolgt eine Suche in Pfadnamentabelle **916** von Fig. 9C nach dem Anteil hoher Ebene des Pfades CONFIG/USER/users. Ein korrespondierender LDAP-Pfadname hoher Ebene, der von einer Reihe von DNs repräsentiert wird, z.B. bu=SunSoft zu o=Sun zu c=US, wird identifiziert. In der beschriebenen Ausführungsform gibt es fünf mögliche JSD-Pfadnamen hoher Ebene: drei im Namensraum MACHINE (platform, identifier und Profile) und zwei im Namensraum USER (users und groups). Sobald ein korrespondierender LDAP-Pfadname hoher Ebene identifiziert ist, erfolgt eine Suche in der LDAP-Datenbank „unter" dem identifizierten Pfad hoher Ebene in Schritt **1004**. In obigem Beispiel wären dies sämtliche Daten unterhalb von bu=SunSoft. Die LDAP-Datenbank sucht nach dem Nachnamen-Attribut für Benutzer Jon Smith mithilfe des LDAP-Cursorsuchmechanismus unter LDAP. In der beschriebenen Ausführungsform verwendet sie Metaverzeichnis **914** aus Fig. 9B, um zu ermitteln, ob der Typ user ein Attribut aufweist, das dem Nachnamen entspricht.

[0059] In Schritt **1006** ruft der LDAP-Server das Attribut „ln" und dessen Wert ab, wie in Liste **906** in Fig. 9A gezeigt, und gibt die Daten an das JSD-Serverschema zurück. In Schritt **1008** erstellt das JSD-Serverschema eine Eigenschaft, die dem Attribut „ln" entspricht, und fügt den entsprechenden Wert ein. Diese Daten werden anschließend zum Client übertragen, der sie angefordert hat, und der Prozess ist abgeschlossen. In der beschriebenen Ausführungsform wird das Konfigurationsdatenelement, das anfangs nicht auf dem JSD-Server vorhanden war und vom LDAP-Server abgerufen wurde, nicht nur wie angefordert für den Client bereitgestellt, sondern auch verwendet, um den JSD-Server zu aktualisieren.

[0060] Fig. 11 ist ein Ablaufdiagramm eines Prozesses, in dem ein LDAP-Eintrag auf einen JSD-Eintrag abgebildet wird, gemäß einer Ausführungsform der vorliegenden Erfindung. Es beschreibt Schritt **806** von Fig. 8 detaillierter. Dieser Prozess verwendet die Schattenattribute, die erstmals in Fig. 9A in Attributliste **906** beschrieben sind. Man erinnere sich, dass Fig. 8 einen Prozess zum Initialisieren des JSD-Servers beim Starten beschrieb. In der beschriebenen Ausführungsform wird in diesem Prozess ein signifikanter Anteil der Daten des Namensraums USER im JSD-Schema mit Daten vom LDAP-Server beschickt. In anderen Ausführungsformen können einige der USER-Daten bereits auf dem JSD-Server resident sein, wie dies bei den Daten des Namensraums MACHINE der beschriebenen Ausführungsform der Fall ist. Weil das Volumen der Daten des Namensraums MACHINE erheblich geringer ist als das des Namensraums USER, ist es machbar und effizient, jene Informationen auf dem JSD-Server aufzubewahren. Die Daten des Namensraums USER residieren typischerweise in einem LDAP-Kontext (z.B. dem Kontext „JSD") und werden in den JSD-Server importiert oder in seinem Cache gespeichert.

[0061] In Schritt **1102** liest der LDAP-Server jeden Eintrag in dem oder den entsprechenden LDAP-Kontext(en). Wegen des „leichten" Designs und Protokolls von LDAP kann dies zügig erfolgen. In der beschriebenen Ausführungsform gibt es einen Kontext, der sämtliche Konfigurationsdaten enthält, die zum Beschicken des Namensraums USER im JSD-Serverschema benötigt werden. In Schritt **1104** unterscheidet oder separiert der LDAP-Server Schattenattribute und korrespondierende „Nicht-Schatten"-Attribute. Auch dies kann zügig erfolgen, weil die Schattenattribute ein eindeutiges Anfangssymbol aufweisen, beispielsweise einen Punkt, wie in den Attributen **908**, **910** und **912** von Fig. 9A gezeigt. In Schritt **1106** benutzt der LDAP-Server das Nicht-Schatten-Attribut, um den Ist-Wert des Attributs oder Eintrags, z.B. „logon:jsmith", und das korrespondierende Schattenattribut für Anweisungen darüber, wo im JSD-Serverschema der Wert zu platzieren ist, und den Objekttyp zu ermitteln, in dem der Wert zu speichern ist. In Schritt **1108** akzeptiert der JSD-Server den Wert und verwendet einen geeigneten Objekt-Constructor, um ein Objekt zu erzeugen, das dem Attributwert entspricht, und speichert das Objekt in einem Blatt am unteren Ende des JSD-Pfades, der im Schattenattribut bereitgestellt ist. Durch Verwenden des Schattenattributs für den JSD-Speicherort und des Nicht-Schatten-Attributs für den Wert ermöglicht LDAP dem JSD-Server das Vermeiden langer Startzeiten.

[0062] Die vorliegende Erfindung, die primär Software und Datenformate oder -strukturen betrifft, setzt verschiedene computerimplementierte Vorgänge ein, die in Computersystemen gespeicherte Daten betreffen. Diese Vorgänge beinhalten, ohne darauf beschränkt zu sein, jene, die physikalische Manipulati-

on physikalischer Größen erfordern. Üblicher-, aber nicht notwendigerweise nehmen diese Größen die Form elektrischer oder magnetischer Signale an, die in der Lage sind, gespeichert, übertragen, kombiniert, verglichen und anderweitig manipuliert zu werden. Die hierin beschriebenen Vorgänge, die einen Teil der Erfindung bilden, sind nützliche Maschinenvorgänge. Die vorgenommenen Manipulationen werden oft mit Begriffen wie z.B. Erzeugen, Identifizieren, Laufen, Ermitteln, Vergleichen, Ausführen, Herunterladen oder Erkennen bezeichnet. Manchmal ist es vornehmlich aus Gründen der üblichen Verwendung praktisch, diese elektrischen oder magnetischen Signale als Bits, Werte, Elemente, Variablen, Zeichen, Datenelemente oder dergleichen zu bezeichnen. Man sollte jedoch nicht vergessen, dass all diese und ähnliche Begriffe den entsprechenden physikalischen Größen zuzuordnen sind und hauptsächlich praktische Bezeichnungen sind, die auf diese Größen angewendet werden.

[0063] Die vorliegende Erfindung betrifft auch ein Gerät, ein System oder eine Vorrichtung zur Durchführung oben erwähnter Vorgänge, wie z.B. den JSD-Server und den LDAP-Server. Diese Systeme können speziell für die gewünschten Zwecke aufgebaut sein, oder sie können ein Mehrzweckcomputer sein, der von einem Computerprogramm selektiv aktiviert oder konfiguriert wird, das im Computer gespeichert ist oder unter einem bestimmten Protokoll arbeitet, wie z.B. dem Lightweight Directory Access Protocol (LDAP). Die oben präsentierten Prozesse und Datenformate sind nicht von Natur aus auf einen bestimmten Computer oder eine sonstige Rechnervorrichtung bezogen. Insbesondere können verschiedene Mehrzweckcomputer mit Programmen benutzt werden, die gemäß der Ausführungen hierin geschrieben wurden, oder es kann alternativ praktischer sein, ein spezialisierteres Computersystem aufzubauen, um die geforderten Vorgänge auszuführen, wie z.B. einen Server-Computer, der aufgebaut ist, um als ein JSD-Server betrieben zu werden.

[0064] Fig. 12 ist ein Blockdiagramm eines Mehrzweck-Computersystems **1200**, das zum Durchführen der Verarbeitung gemäß einer Ausführungsform der vorliegenden Erfindung geeignet ist. Fig. 12 stellt eine Ausführungsform eines Mehrzweck-Computersystems dar, das konfiguriert und erweitert werden kann, um als ein Server-Computer betrieben zu werden, wie z.B. als der LDAP-Verzeichnisserver oder der JSD-Server. Andere Computersystemarchitekturen und – konfigurationen können für das Durchführen der Verarbeitung der vorliegenden Erfindung benutzt werden. Computersystem **1200**, das aus verschiedenen Subsystemen gebildet ist, die weiter unten beschrieben werden, beinhaltet mindestens ein Mikroprozessor-Subsystem (auch als Central Processing Unit, CPU bezeichnet) **1202**. Das heißt, dass CPU **102** durch einen Einzelchip-Prozessor oder durch Mehrfach-Prozessoren implementiert werden kann. CPU **102** ist ein digitaler Mehrzweck-Prozessor, der den Betrieb des Computersystems **1200** steuert. Mithilfe von Befehlen, die aus dem Speicher abgerufen werden, steuert CPU **1202** den Empfang und die Manipulation von Eingangsdaten und gegebenenfalls die Ausgabe und Anzeige von Daten auf Ausgabegeräten.

[0065] Über einen Speicherbus **1208** ist CPU **1202** bidirektional mit einem ersten Primärspeicher **1204** gekoppelt, typischerweise einem Schreib-Lese-Speicher (Random Access Memory, RAM), und unidirektional mit einem zweiten Primärspeicherbereich **1206**, typischerweise einem Lese-Speicher (Read-Only Memory, ROM). Wie auf dem Fachgebiet wohl bekannt ist, kann Primärspeicher **1204** als ein allgemeiner Speicherbereich und als ein Scratch-Pad-Speicher verwendet werden und kann auch benutzt werden, um Eingangsdaten und verarbeitete Daten zu speichern. Er kann außerdem Programmbefehle und -daten speichern, beispielsweise in der Form eines erweiterten LDAP-Attributs oder -Eintrags oder in der Form einer JSD-Hierarchie, wie in den obigen Figuren beschrieben. Dieses ist zusätzlich zu anderen Daten und Befehlen für Prozesse, die auf CPU **1202** laufen, und wird typischerweise für den schnellen Transfer von Daten und Befehlen in einer bidirektionalen Weise über den Speicherbus **1208** verwendet. Wie ebenfalls auf dem Fachgebiet wohl bekannt ist, beinhaltet Primärspeicher **1206** typischerweise grundlegende Betriebsbefehle, Programmcode, Daten und Objekte, die von der CPU **1202** verwendet werden, um ihre Funktionen auszuführen. Die Primärspeichergeräte **1204** und **1206** können beliebige geeignete computerlesbare Speichermedien enthalten, wie weiter unten beschrieben, abhängig davon, ob beispielsweise Datenzugriff bidirektional oder unidirektional sein muss. CPU **1202** kann auch häufig benötigte Daten direkt und zügig aus einem Cache-Speicher **1210** abrufen und in diesem speichern.

[0066] Ein entfernbares Massenspeichergerät **1212** stellt zusätzliche Datenspeicherkapazität für das Computersystem **1200** bereit und ist entweder bidirektional oder unidirektional über einen Peripheriebus **1214** mit CPU **1202** gekoppelt. Beispielsweise leitet ein bestimmtes Massenspeichergerät, das üblicherweise als ein CD-ROM bekannt ist, typischerweise Daten unidirektional zur CPU **1202**, wohingegen eine Diskette Daten bidirektional zur CPU **1202** leiten kann. Die Daten im Namensraum MACHINE im JSD-Server können beispielsweise im entfernbaren Massen-speichergerät **1212** gespeichert werden. Speicher **1212** kann auch computerlesbare Medien wie z.B. Magnetband, Flash-Speicher, in einer Trägerwelle aufgenommene Signale, PC-CARDS, portable Massenspeichergeräte, polografische Speichergeräte und andere Speichergeräte beinhalten. Ein

fester Massenspeicher **1216** stellt ebenfalls zusätzliche Datenspeicherkapazität bereit und ist über Peripheriebus **1214** bidirektional mit CPU **1202** gekoppelt. Das gängigste Beispiel für Massenspeicher **1216** ist ein Festplattenlaufwerk. Generell ist der Zugriff auf diese Medien langsamer als der Zugriff auf Primärspeicher **1204** und **1206**. Massenspeicher **1212** und **1216** speichern im Allgemeinen zusätzliche Programmbefehle, Daten und dergleichen, die typischerweise nicht in aktivem Gebrauch durch die CPU **1202** sind. Man wird verstehen, dass die Informationen, die innerhalb von Massenspeicher **1212** und **1216** aufbewahrt werden, falls erforderlich, in standardisierter Weise als Teil von Primärspeicher **1204** (z.B. RAM) als virtueller Speicher integriert werden können.

[0067] Zusätzlich dazu, CPU **1202** Zugriff auf Speicher-Subsysteme bereitzustellen, wird der Peripheriebus **1214** benutzt, ebenfalls Zugriff auf andere Subsysteme und Geräte bereitzustellen. In der beschriebenen Ausführungsform beinhalten diese einen Anzeigemonitor **1218** und Adapter **1220**, ein Druckergerät **1222**, eine Netzwerkschnittstelle **1224**, eine Zusatz-Ein-und-Ausgangsgeräte-Schnittstelle **1226**, eine Soundkarte **1228** und Lautsprecher **1230** und, falls benötigt, andere Subsysteme.

[0068] Die Netzwerkschnittstelle **1224** gestattet es, CPU **1202**, wie gezeigt, mit einem anderen Computer, einem Computernetzwerk oder einem Telekommunikationsnetz unter Verwendung einer Netzwerkverbindung zu koppeln. Es ist zu erwarten, dass die CPU **1202** durch die Netzwerkschnittstelle **1224** Informationen, z.B. Konfigurationsdaten, Anforderungen von Konfigurationsdaten oder Programmbefehle von einem anderen Netzwerk empfangen könnte oder im Verlauf der Durchführung der oben beschriebenen Verfahrensschritte Informationen an ein anderes Netzwerk ausgeben könnte. Informationen, die oft als eine Folge von Befehlen dargestellt sind, die auf einer CPU auszuführen sind, können beispielsweise in der Form eines Computerdatensignals, das in einer Trägerwelle aufgenommen ist, von einem anderen Netzwerk empfangen und an dieses ausgegeben werden. Eine Schnittstellenkarte oder ein ähnliches Gerät und geeignete Software, die von CPU **1202** implementiert sind, können benutzt werden, um das Computersystem **1200** mit einem externen Netzwerk zu verbinden und Daten gemäß Standardprotokollen zu übertragen. Das heißt, Verfahrens-Ausführungsformen der vorliegenden Erfindung können ausschließlich auf CPU **1202** ausgeführt werden oder können über ein Netzwerk wie z.B. das Internet, Intranet-Netzwerke oder lokale Netzwerke hinweg, wie z.B. in einem unternehmensweiten Netzwerk, in Verbindung mit einer entfernten CPU durchgeführt werden, die einen Teil der Verarbeitung übernimmt. Zusätzliche (nicht gezeigte) Massenspeichergeräte können auch über Netzwerkschnittstelle **1224** mit CPU **1202** verbunden werden.

[0069] Zusatz-E/A-Geräte-Schnittstelle **1226** repräsentiert allgemeine und angepasste Schnittstellen, die es der CPU **1202** gestatten, Daten an andere Geräten wie z.B Mikrofone, berührungsempfindliche Anzeigen, Transducerkartenlesegeräte, Bandlesegeräte, Sprach- oder Handschrifterkennungsgeräte, biometrische Lesegeräte, Kameras, portable Massenspeichergeräte und andere Computer zu senden und, typischer, von diesen zu empfangen

[0070] Ebenfalls mit der CPU **1202** gekoppelt ist über einen lokalen Bus **1234** ein Tastaturcontroller **1232** zum Empfangen von Eingaben von einer Tastatur **1236** oder einem Zeigegerät **1238** und Senden decodierter Symbole von der Tastatur **1236** oder dem Zeigegerät **1238** zur CPU **1202**. Das Zeigegerät kann eine Maus, ein Griffel, ein Trackball oder ein Tablett sein und ist nützlich für die Interaktion mit einer grafischen Benutzeroberfläche.

[0071] Darüber hinaus betreffen Ausführungsformen der vorliegenden Erfindung ferner Computerspeicherprodukte einschließlich eines computerlesbaren Mediums, das Programmcode zur Durchführung verschiedener computerimplementierter Vorgänge enthält. Das computerlesbare Medium ist ein beliebiges Datenspeichergerät, das Daten speichern kann, die danach von einem Computersystem gelesen werden können. Bei den Medien und dem Programmcode kann es sich um jene handeln, die speziell für die Zwecke der vorliegenden Erfindung entwickelt und aufgebaut wurden, wie z.B. Abrufen von Konfigurationsdaten vom LDAP-Server, oder sie können von der Art sein, die dem Durchschnittsfachmann auf dem Gebiet der Computersoftware wohl bekannt ist. Zu den Beispielen computerlesbarer Medien zählen, ohne darauf beschränkt zu sein, alle oben erwähnten Medien: magnetische Medien wie z.B. Festplatten, Disketten und Magnetband, optische Medien wie z.B. CDs, magneto-optische Medien wie z.B. Floptical-Disks und speziell konfigurierte Hardwaregeräte wie z.B. anwendungsspezifische integrierte Schaltkreise (Applicationspecific integrated Circuits, ASICs), speicherprogrammierbare Geräte (PLDs) und ROM- und RAM-Geräte. Das computerlesbare Medium kann auch als ein Datensignal, das in einer Trägerwelle aufgenommen ist, über ein Netzwerk gekoppelter Computersysteme verteilt werden, sodass der computerlesbare Code in verteilter Weise gespeichert und ausgeführt wird. Zu den Beispielen für Programmcode zählen sowohl Maschinencode, wie er beispielsweise von einem Compiler erzeugt wird, oder Dateien, die Code höheren Niveaus enthalten, der mithilfe eines Interpreters ausgeführt werden kann.

[0072] Der Fachmann wird verstehen, dass die oben beschriebenen Hardware- und Softwareele-

mente von standardmäßiger Konzeption und ebensolchem Aufbau sind. Andere Computersysteme, die für die Verwendung mit der Erfindung geeignet sind, können zusätzliche oder weniger Subsysteme beinhalten. Darüber hinaus sind Speicherbus **1208**, Peripheriebus **1214** und lokaler Bus **1234** beispielhaft für alle Verbindungsschemata, die zum Verknüpfen der Subsysteme dienen. Beispielsweise könnte ein lokaler Bus verwendet werden, um die CPU mit dem festen Massenspeicher **1216** und Anzeigeadapter **1220** zu verbinden. Das in Fig. 12 gezeigte Computersystem ist nur ein Beispiel eines Computersystems, das für die Verwendung mit der Erfindung geeignet ist.

[0073] Andere Computerarchitekturen, die unterschiedliche Konfigurationen von Subsystemen aufweisen, wie z.B. jene von Server-Computern, können auch genutzt werden.

[0074] Obgleich die vorgenannte Erfindung zum Zwecke der Verdeutlichung und Verständlichkeit recht ausführlich beschrieben worden ist, ist offensichtlich, dass gewisse Änderungen und Modifikationen innerhalb des Umfangs der beigefügten Ansprüche in die Praxis umgesetzt werden können. Außerdem ist zu beachten, dass es alternative Wege der Implementierung sowohl des Prozesses als auch der Vorrichtung der vorliegenden Erfindung gibt. Beispielsweise können, obgleich die Erfindung anhand von Konfigurationsdaten beschrieben ist, andere Arten von Nicht-Konfigurations-Daten ebenfalls in der Java-System-Datenbank gespeichert und von LDAP-Verzeichnisdiensten in Netzwerken auf diese zugegriffen werden, in denen die LDAP-Datenbank Daten des Nicht-Konfigurations-Typs speichert. In einem anderen Beispiel können einige der Konfigurationsdaten des Namensraums USER ständig auf dem JSD-Server residieren, weshalb diese Daten nicht vom LDAP-Server beim Starten des JSD-Servers importiert oder im Cache gespeichert werden müssen. In noch einem anderen Beispiel kann ein anderer Konfigurations-Aufbewahrungsort als das beschriebene JSD-Serverschema benutzt werden, um Konfigurationsdaten zu speichern. Das Konzept der Schattenattribute und der Abbildung von Pfaden hoher Ebene kann in Verbindung mit einem anderen Typ von Konfigurations-Aufbewahrungsort verwendet werden, Dement-sprechend sind die vorliegenden Ausführungsformen als beispielhaft und nicht einschränkend anzusehen, und die Erfindung ist nicht auf die hierin angegebenen Einzelheiten zu beschränken, sondern kann innerhalb des Umfangs und der Äquivalente der beigefügten Ansprüche modifiziert werden.

Patentansprüche

1. Verfahren des Abrufens von Werten von Konfigurationsdatenelementen von einem Lightweight Directory Access Protocol-Server (LDAP-Server) (**108**) zu einem Javabasierten Konfigurationsserver (**301**), wobei das Verfahren umfasst:
Durchsuchen (**1002**) einer Speicherortabgleichdatei (**916**) nach einer Übereinstimmung zwischen einem Pfad hoher Ebene in einem Java-basierten Konfigurationsserver und einer bestimmten LDAP-Adresse,
Durchsuchen (**1004**) eines Abschnitts des LDAP-Servers nach einem oder mehreren Attributen von einem oder mehreren LDAP-Einträgen unter Verwendung der bestimmten LDAP-Adresse zur Bestimmung des Abschnitts des LDAP-Servers, der zu durchsuchen ist
Abrufen (**1006**) eines oder mehrerer Ist-Werte für das eine oder die mehreren Attribute und Abrufen von Werten für ein oder mehrere Schattenattribute, die dem einen oder den mehreren Attributen entsprechen, wobei die Schattenattribute anzeigen, wo in einem Java-basierten Konfigurationsserverschema ein bestimmter Ist-Wert zu platzieren ist, und
Senden (**1008**) des einen oder der mehreren Ist-Werte zum Java-basierten Konfigurationsserver und Platzieren des einen oder der mehreren Ist-Werte im Java-basierten Konfigurationsserverschema als Konfigurationsdatenelemente basierend auf den Werten für das eine oder die mehreren Schattenattribute, sodass der eine oder die mehreren Ist-Werte Client-Computern in einer Java-Betriebssystemumgebung verfügbar gemacht werden.

2. Verfahren nach Anspruch 1, wobei das Durchsuchen (**1002**) einer Speicherortabgleichdatei (**916**) ferner das Zugreifen auf ein Netzwerkcomputer-Verwaltungstool umfasst, welches die Speicherortabgleichdatei enthält.

3. Verfahren nach Anspruch 1, wobei das Durchsuchen (**1004**) eines Abschnitts des LDAP-Servers ferner das Aufrufen einer LDAP-Suchfunktion und das Übergeben eines LDAP-Attributs als ein Parameter umfasst.

4. Verfahren nach Anspruch 1, wobei die Speicherortabgleichdatei eine Mehrzahl von Pfaden im Java-basierten Konfigurationsserver und eine entsprechende Mehrzahl von LDAP-Adressen beinhaltet.

5. Verfahren nach Anspruch 1, das ferner das Unterscheiden der Attribute von den Schattenattributen (**1104**) umfasst.

6. Verfahren nach Anspruch 1, wobei die Schattenattribute einen Pfad bzgl. der Konfigurationsdatenbank und einen Eigenschaftsnamen enthalten, der der Konfigurationsdatenbank zugeordnet ist.

7. Verfahren nach Anspruch 6, wobei das Schattenattribut ferner einen Klassennamen enthält.

8. Computerprogrammcode, der von einem

Computer ausführbar ist, um das Verfahren nach einem der Ansprüche 1 bis 7 bereitzustellen.

9. Computerprogrammprodukt, das ein computerlesbares Medium umfasst, wobei ein computerlesbares Medium Computerprogrammcode nach Anspruch 8 beinhaltet.

Es folgen 14 Blatt Zeichnungen

Anhängende Zeichnungen

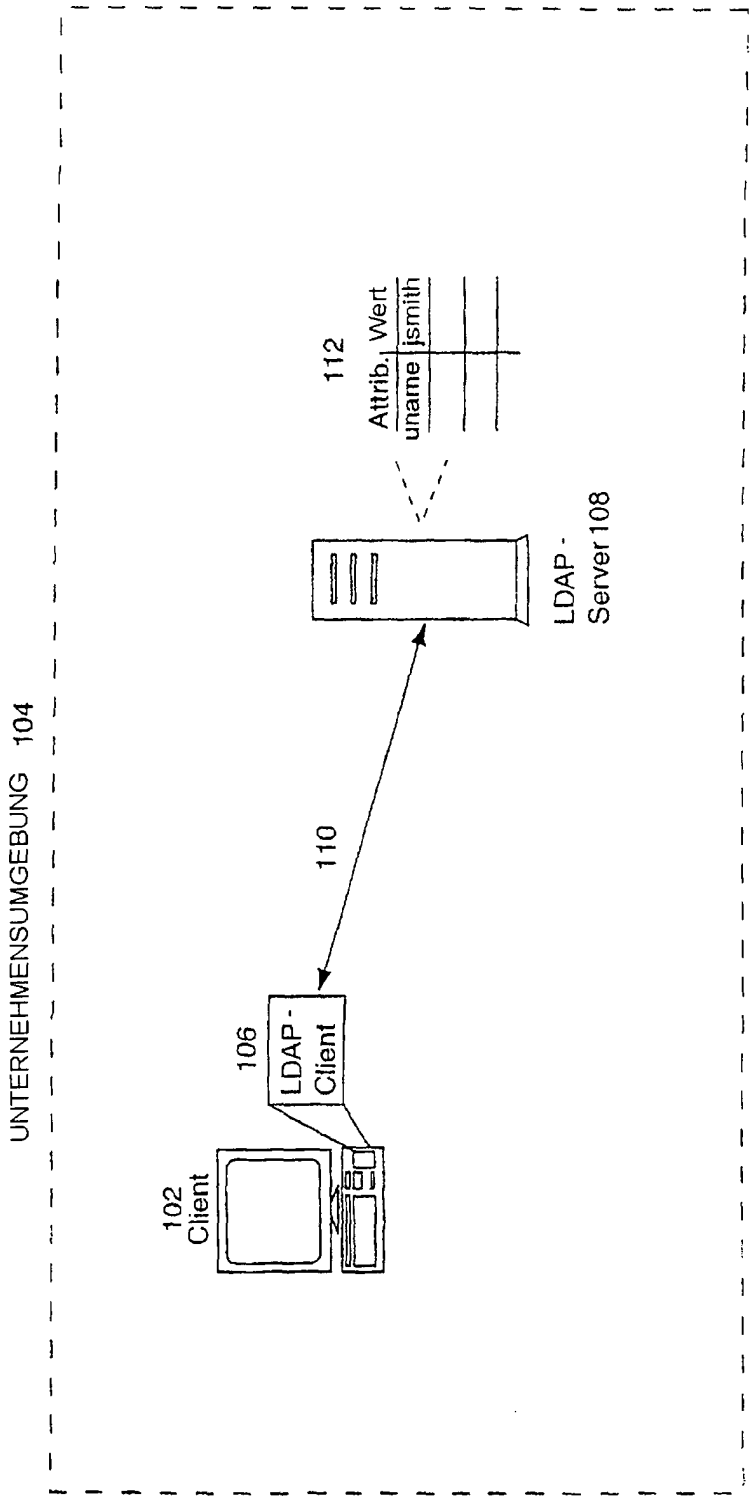

FIG. 1 Stand der Technik

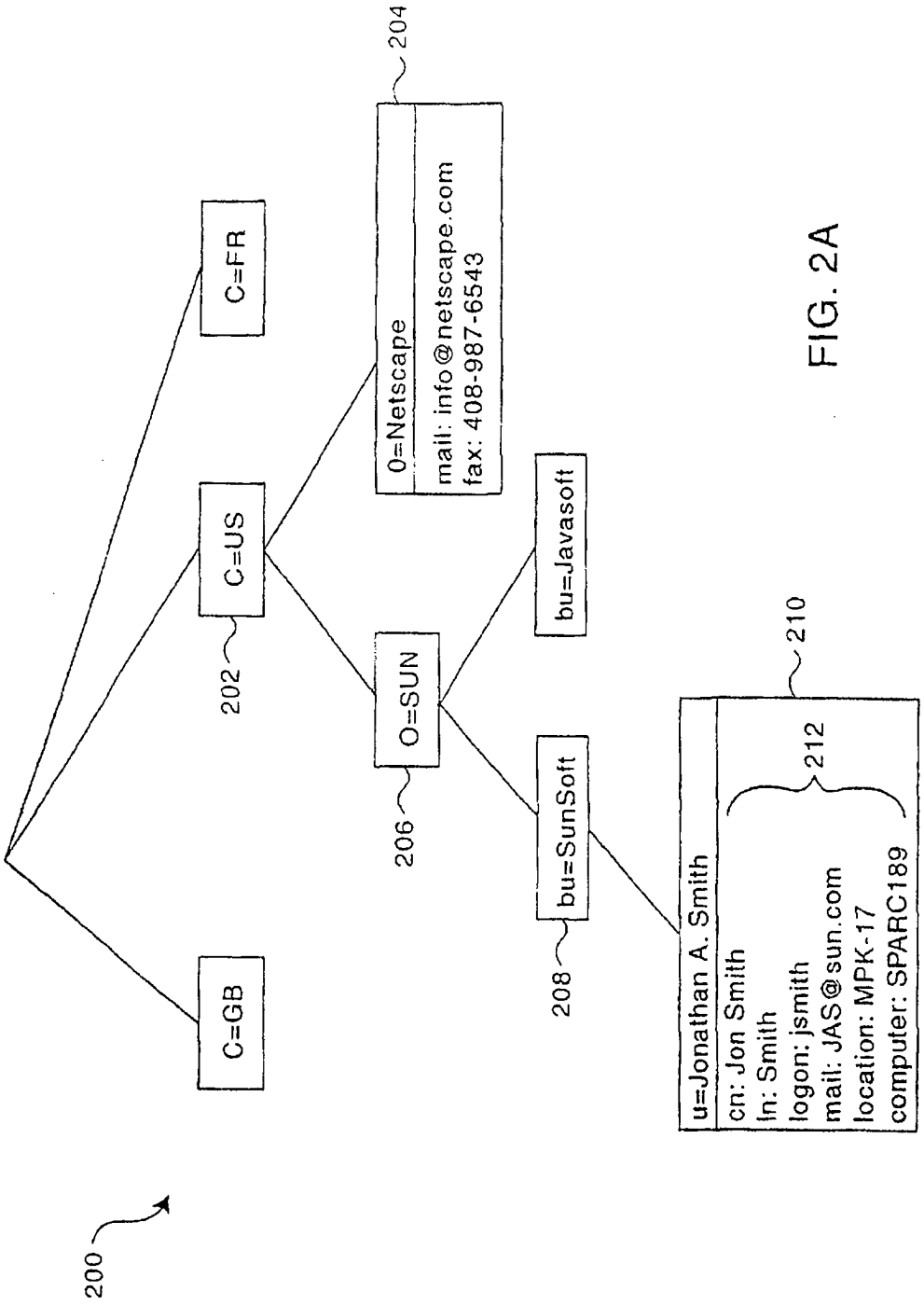
200
C=GB
C=US
C=FR
202
O=SUN
206
bu=SunSoft
bu=Javasoft
208
0=Netscape
mail: info@netscape.com
fax: 408-987-6543
204
u=Jonathan A. Smith
cn: Jon Smith
ln: Smith
logon: jsmith
mail: JAS@sun.com
location: MPK-17
computer: SPARC189
210
212

FIG. 2A

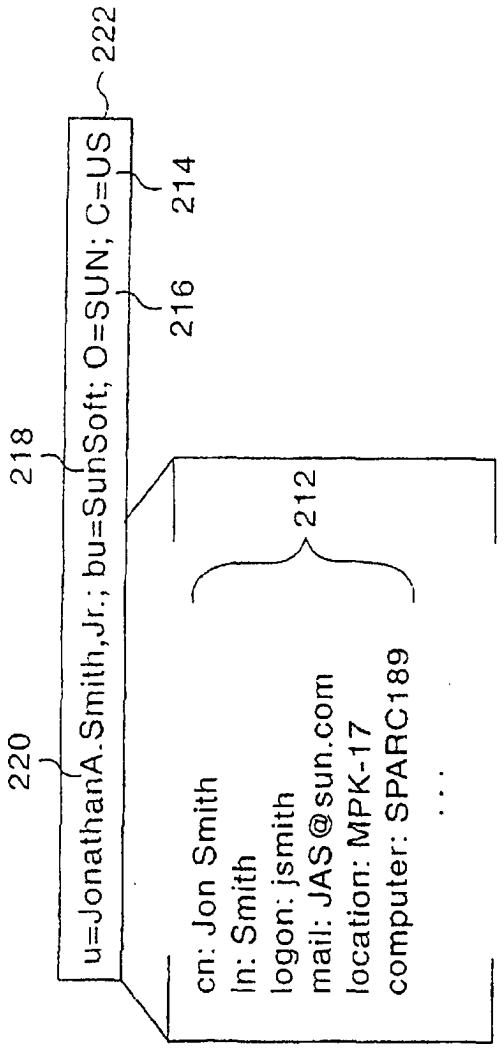
220
218
u=JonathanA.Smith,Jr.; bu=SunSoft; O=SUN; C=US
222
214
216
212
cn: Jon Smith
ln: Smith
logon: jsmith
mail: JAS@sun.com
location: MPK-17
computer: SPARC189
. . .

FIG. 2B

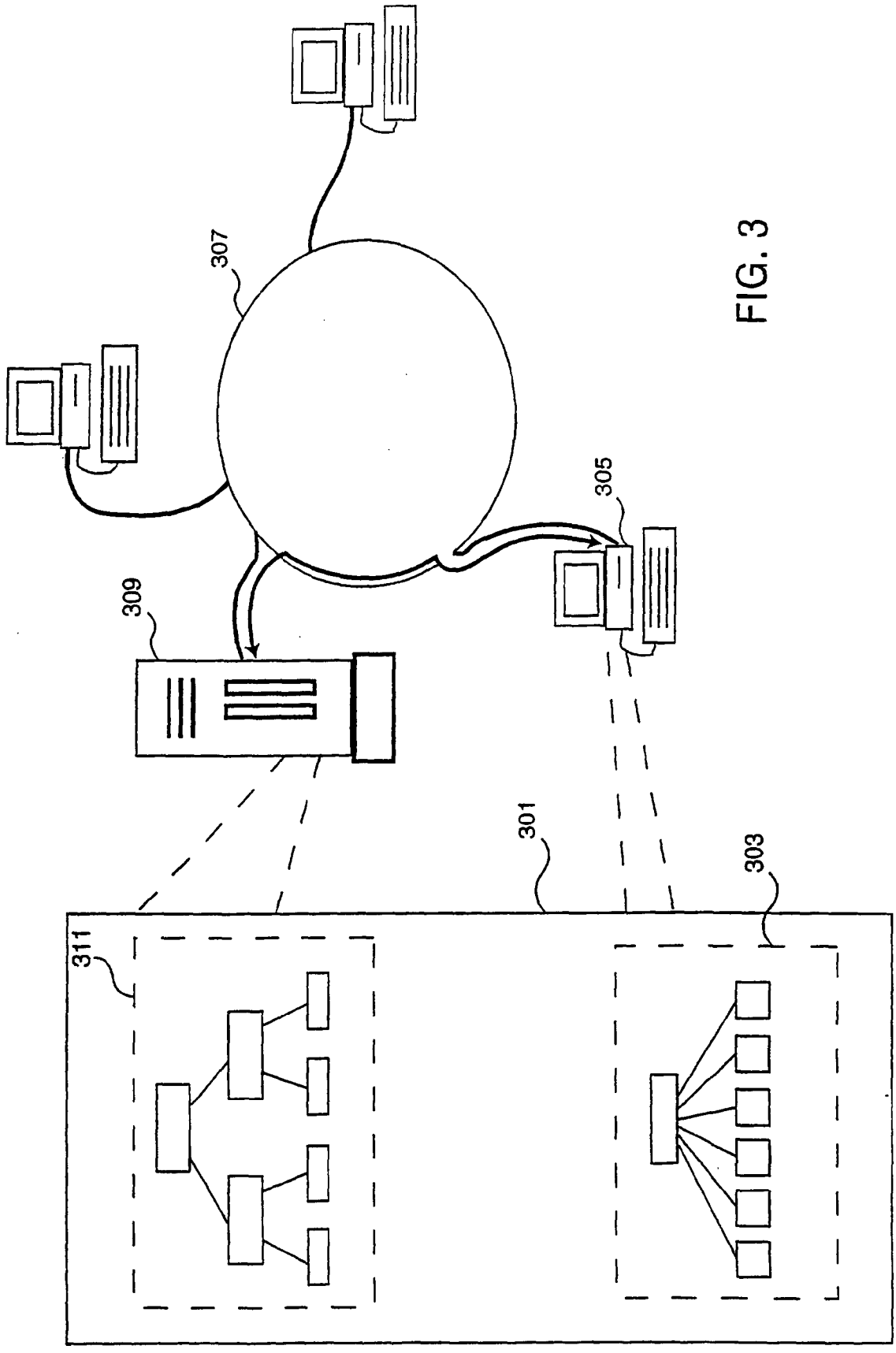

FIG. 3

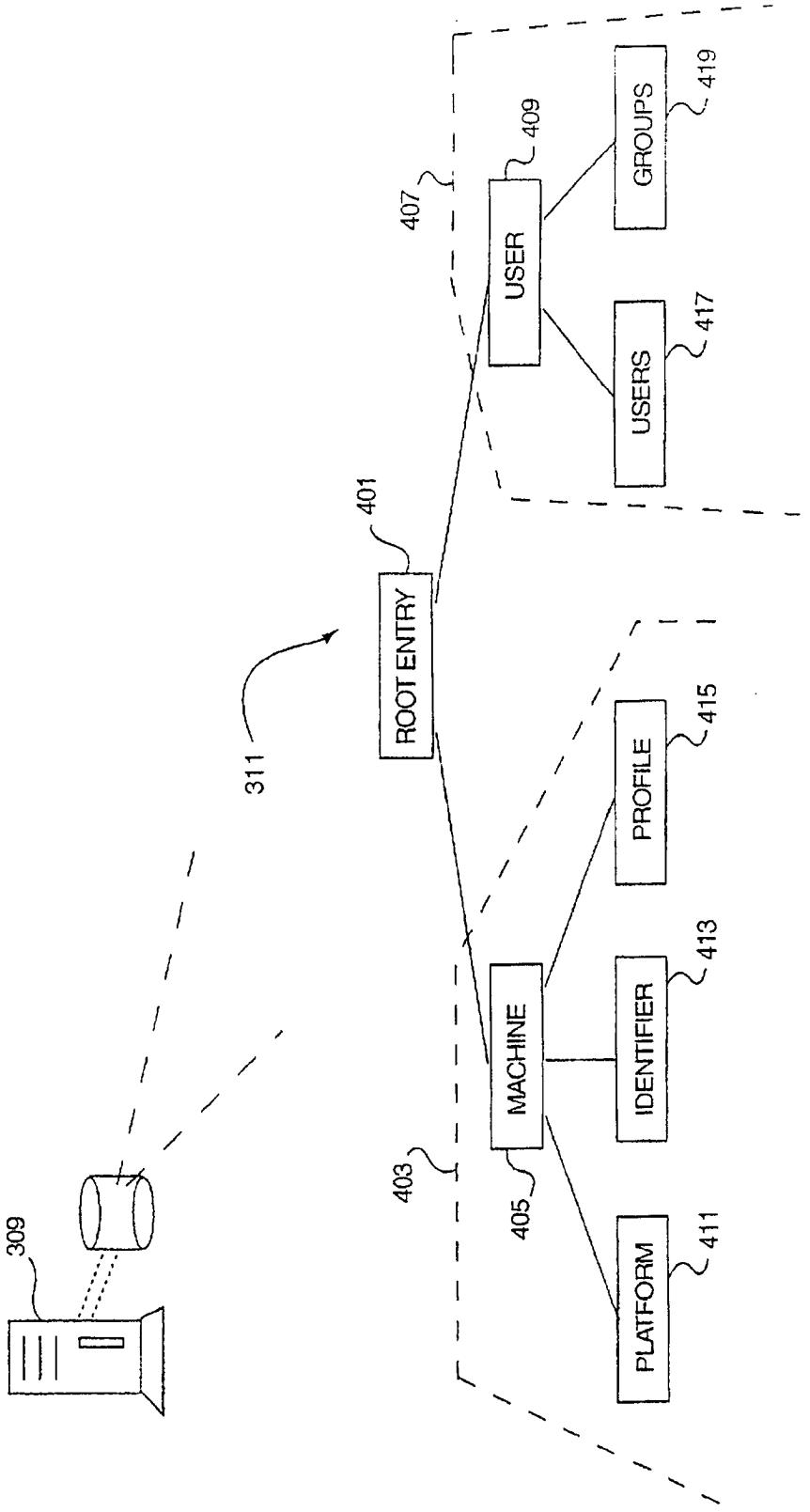
309
311
401
ROOT ENTRY
403
405
MACHINE
407
409
USER
411
PLATFORM
413
IDENTIFIER
415
PROFILE
417
USERS
419
GROUPS

FIG. 4

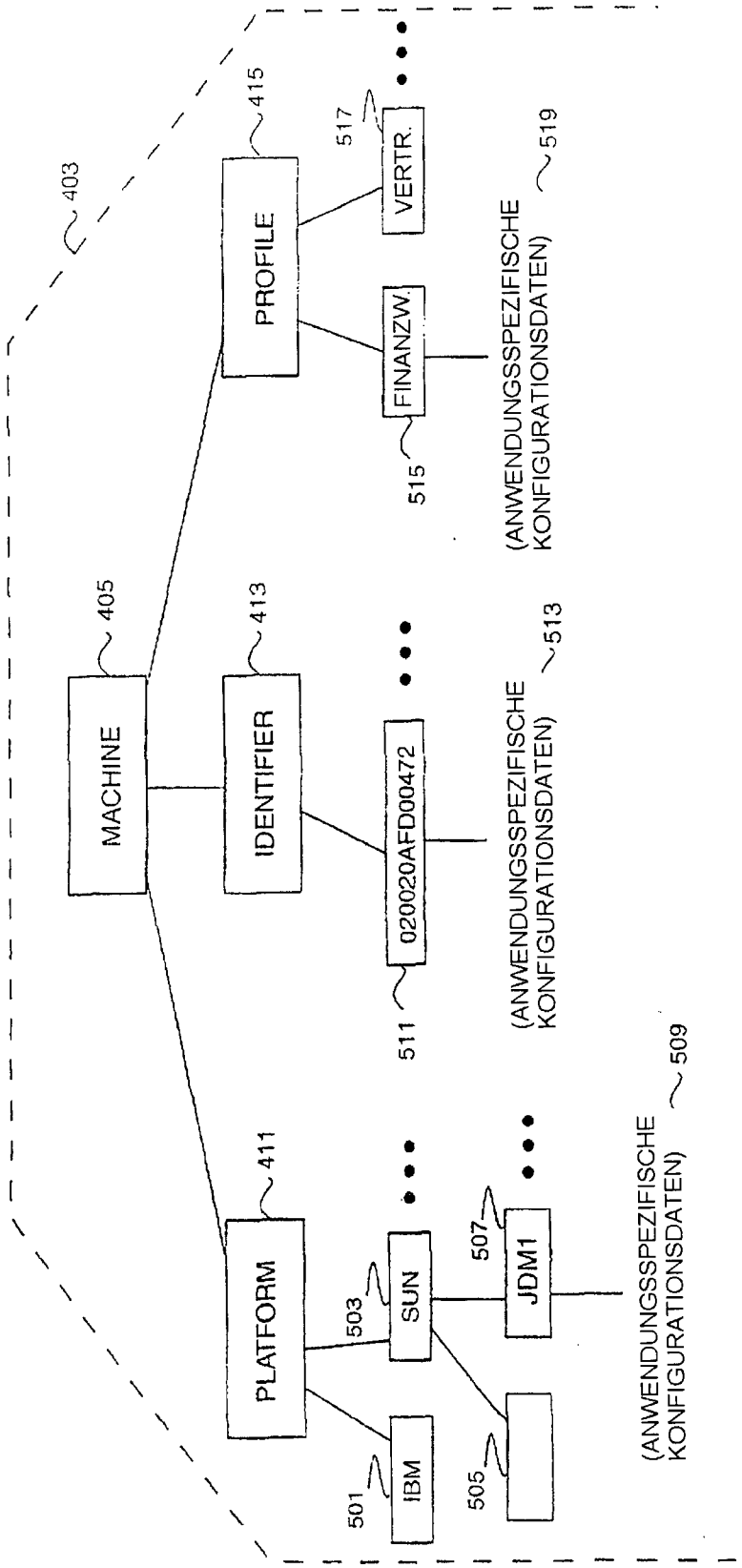
403
MACHINE
405
PLATFORM
411
IDENTIFIER
413
PROFILE
415
IBM
501
SUN
503
020020AFD00472
511
FINANZW.
515
JDM1
507
VERTR.
517
(ANWENDUNGSSPEZIFISCHE KONFIGURATIONSDATEN)
509
(ANWENDUNGSSPEZIFISCHE KONFIGURATIONSDATEN)
513
(ANWENDUNGSSPEZIFISCHE KONFIGURATIONSDATEN)
519
505

FIG. 5

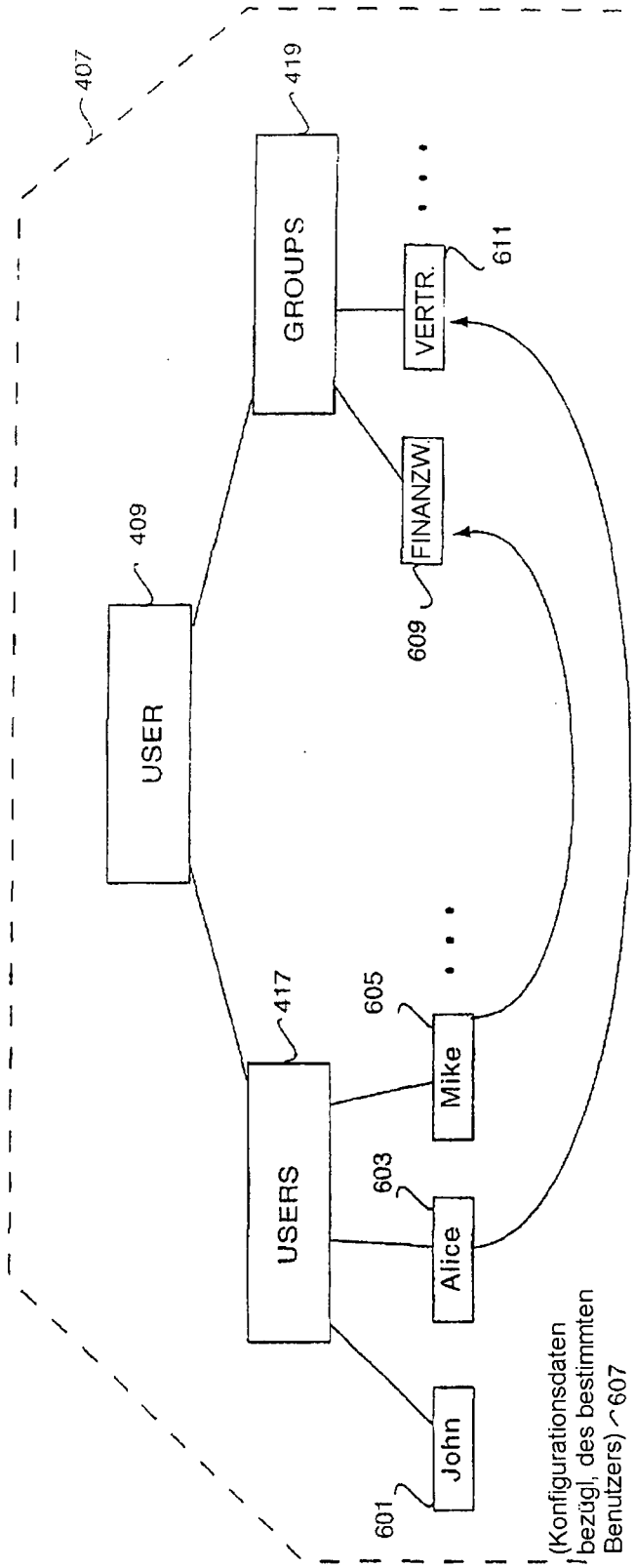
407
419
GROUPS
VERTR.
611
FINANZW.
609
409
USER
417
USERS
605
Mike
603
Alice
John
601
(Konfigurationsdaten bezügl. des bestimmten Benutzers) 607

FIG. 6

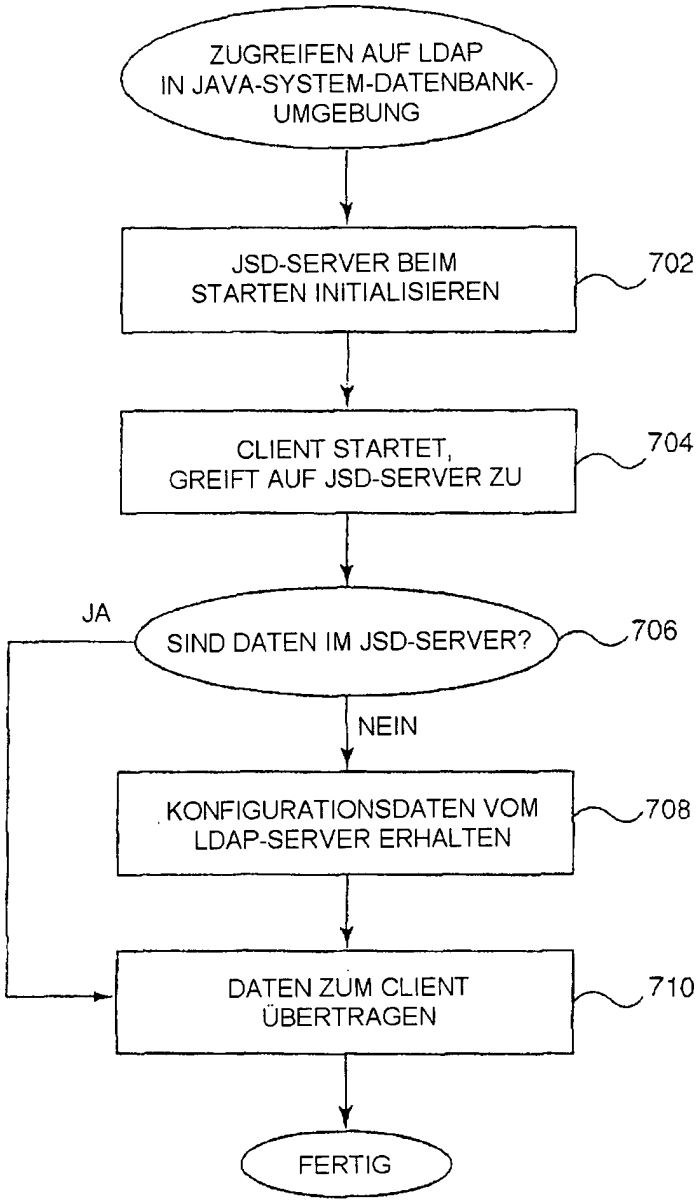
ZUGREIFEN AUF LDAP IN JAVA-SYSTEM-DATENBANK-UMGEBUNG
JSD-SERVER BEIM STARTEN INITIALISIEREN
702
CLIENT STARTET, GREIFT AUF JSD-SERVER ZU
704
JA
SIND DATEN IM JSD-SERVER?
706
NEIN
KONFIGURATIONSDATEN VOM LDAP-SERVER ERHALTEN
708
DATEN ZUM CLIENT ÜBERTRAGEN
710
FERTIG

FIG. 7

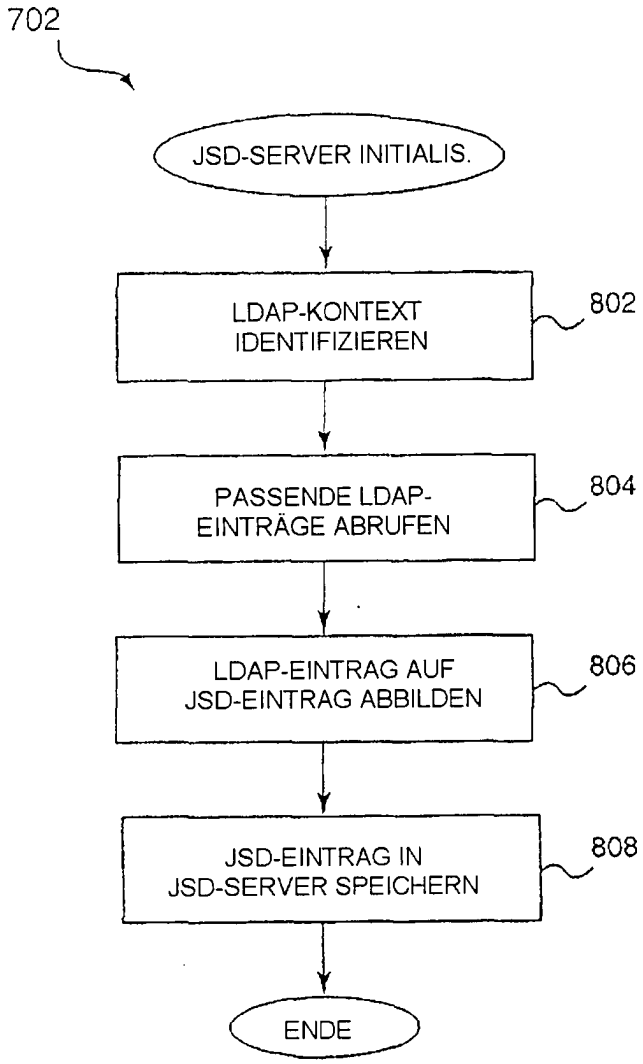
702
JSD-SERVER INITIALIS.
LDAP-KONTEXT IDENTIFIZIEREN
802
PASSENDE LDAP-EINTRÄGE ABRUFEN
804
LDAP-EINTRAG AUF JSD-EINTRAG ABBILDEN
806
JSD-EINTRAG IN JSD-SERVER SPEICHERN
808
ENDE

FIG. 8

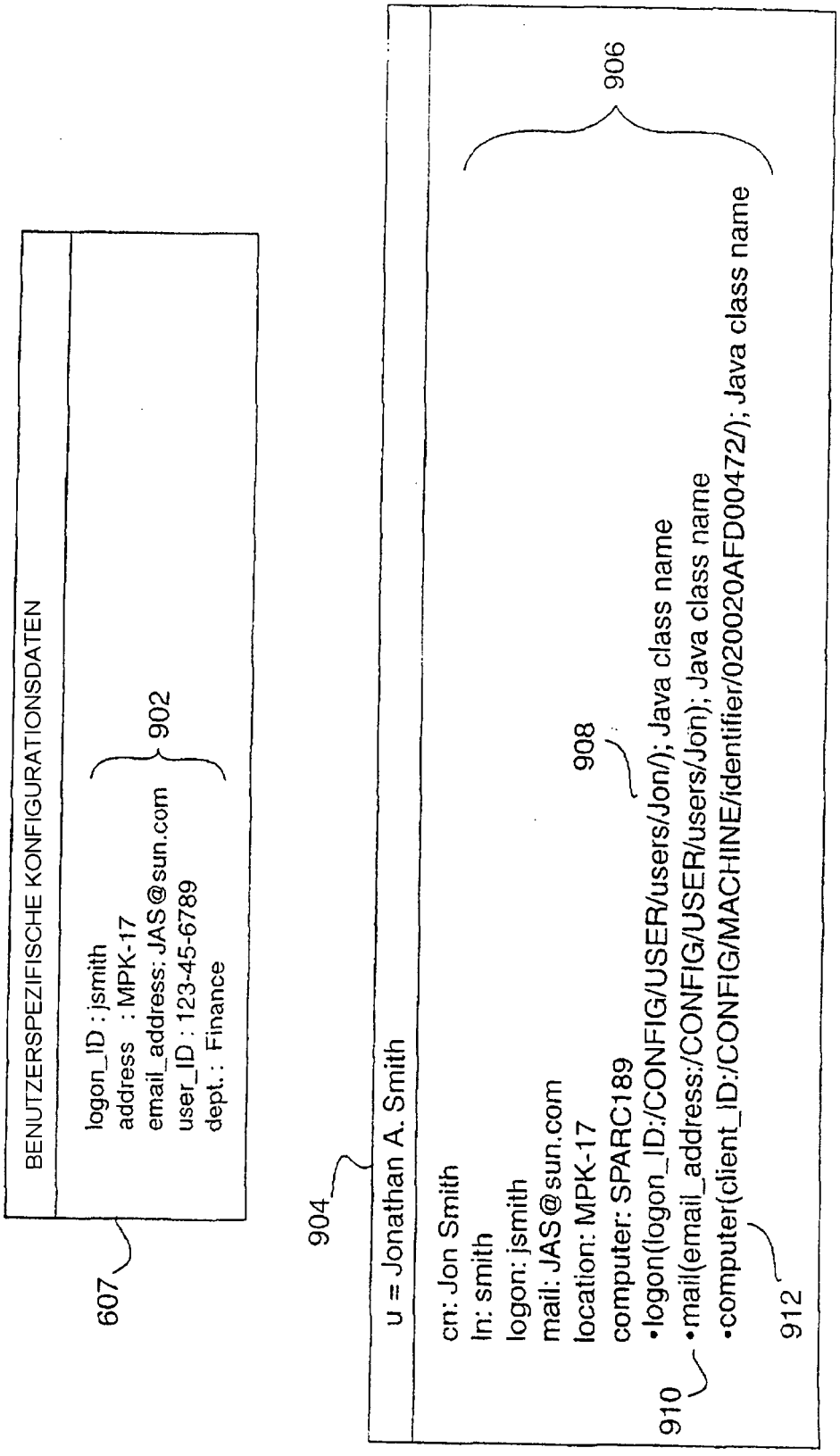
BENUTZERSPEZIFISCHE KONFIGURATIONSDATEN
logon_ID : jsmith
address : MPK-17
email_address: JAS@sun.com
user_ID : 123-45-6789
dept.: Finance
902
607
904
u = Jonathan A. Smith
cn: Jon Smith
ln: smith
logon: jsmith
mail: JAS@sun.com
location: MPK-17
computer: SPARC189
908
906
•logon(logon_ID:/CONFIG/USER/users/Jon/); Java class name
•mail(email_address:/CONFIG/USER/users/Jon); Java class name
•computer(client_ID:/CONFIG/MACHINE/identifier/020020AFD00472/); Java class name
910
912

FIG. 9A

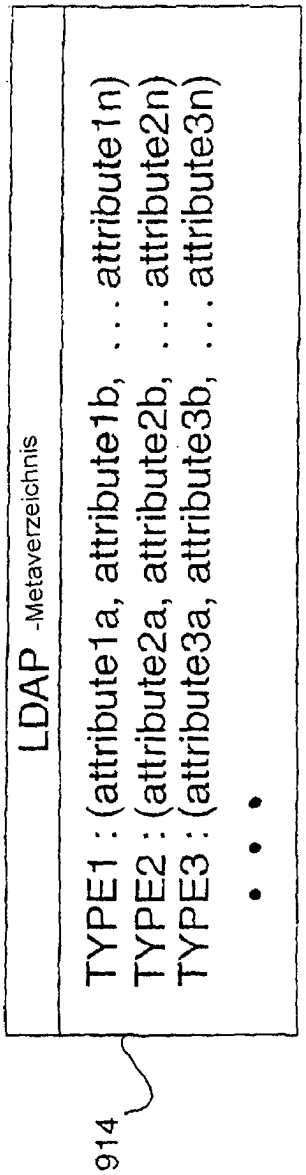
LDAP -Metaverzeichnis
TYPE1 : (attribute1a, attribute1b, . . . attribute1n)
TYPE2 : (attribute2a, attribute2b, . . . attribute2n)
TYPE3 : (attribute3a, attribute3b, . . . attribute3n)
• • •
914

FIG. 9B

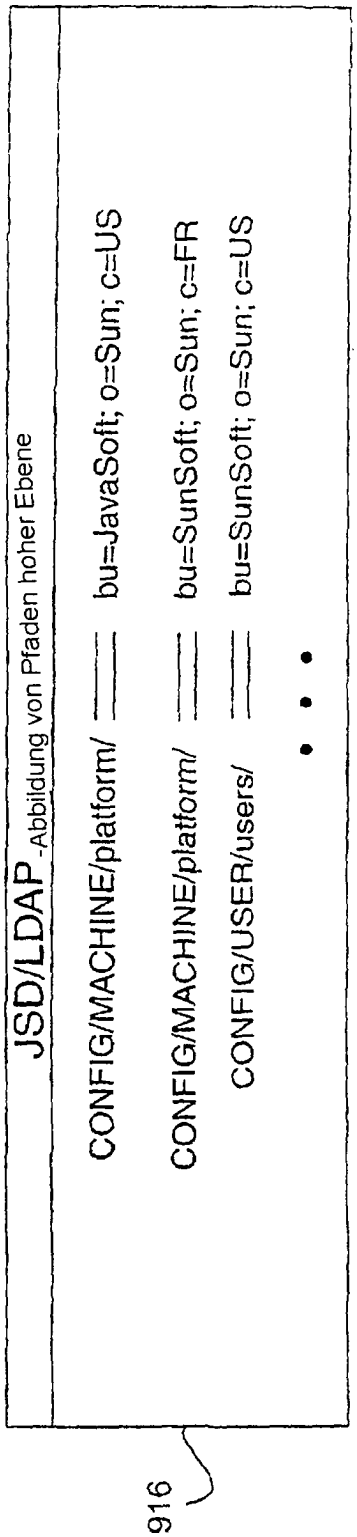
JSD/LDAP -Abbildung von Pfaden hoher Ebene
CONFIG/MACHINE/platform/ ==== bu=JavaSoft; o=Sun; c=US
CONFIG/MACHINE/platform/ ==== bu=SunSoft; o=Sun; c=FR
CONFIG/USER/users/ ==== bu=SunSoft; o=Sun; c=US
• • •
916

FIG. 9C

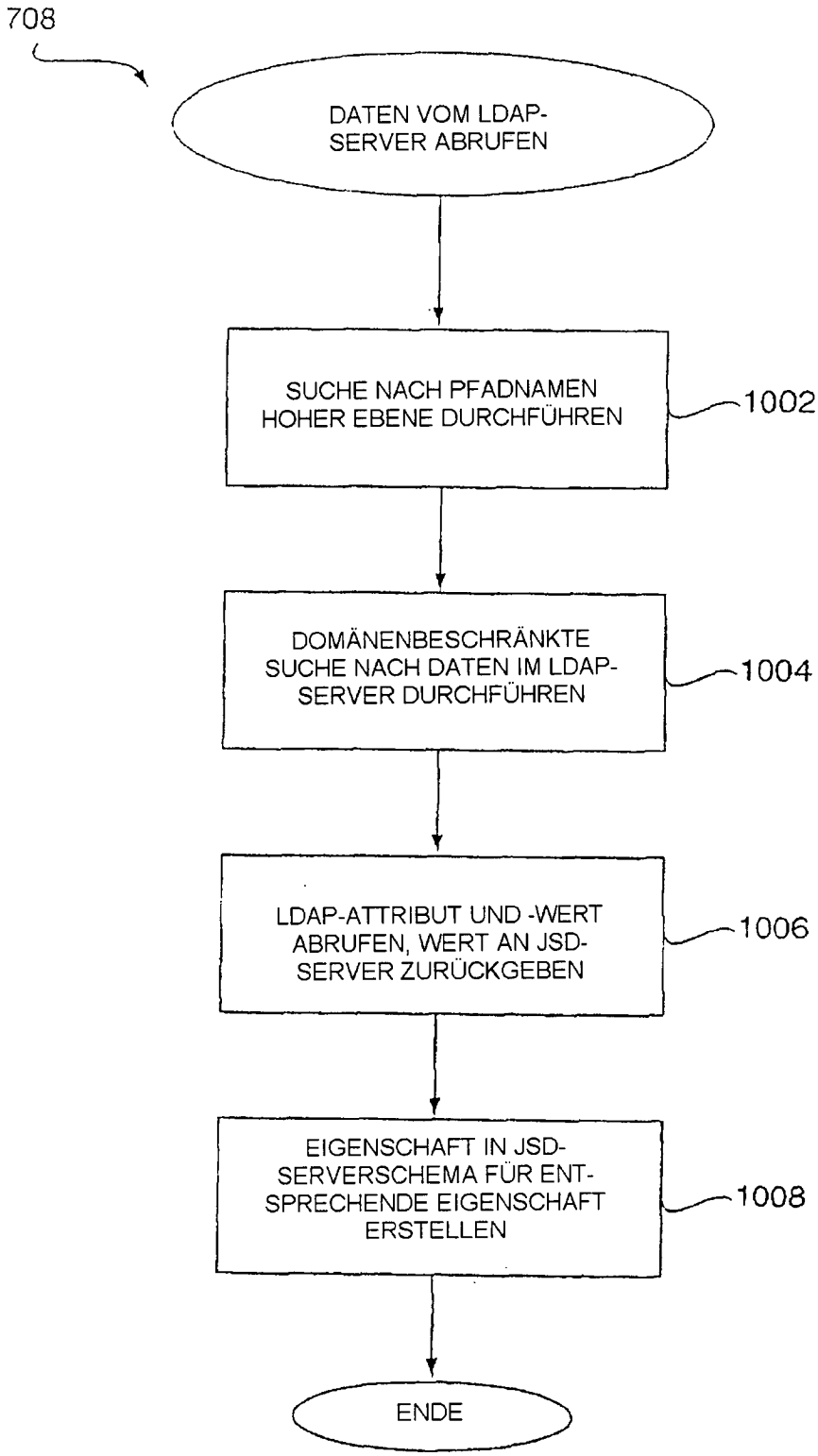
708
DATEN VOM LDAP-SERVER ABRUFEN
SUCHE NACH PFADNAMEN HOHER EBENE DURCHFÜHREN
1002
DOMÄNENBESCHRÄNKTE SUCHE NACH DATEN IM LDAP-SERVER DURCHFÜHREN
1004
LDAP-ATTRIBUT UND -WERT ABRUFEN, WERT AN JSD-SERVER ZURÜCKGEBEN
1006
EIGENSCHAFT IN JSD-SERVERSCHEMA FÜR ENTSPRECHENDE EIGENSCHAFT ERSTELLEN
1008
ENDE

FIG. 10

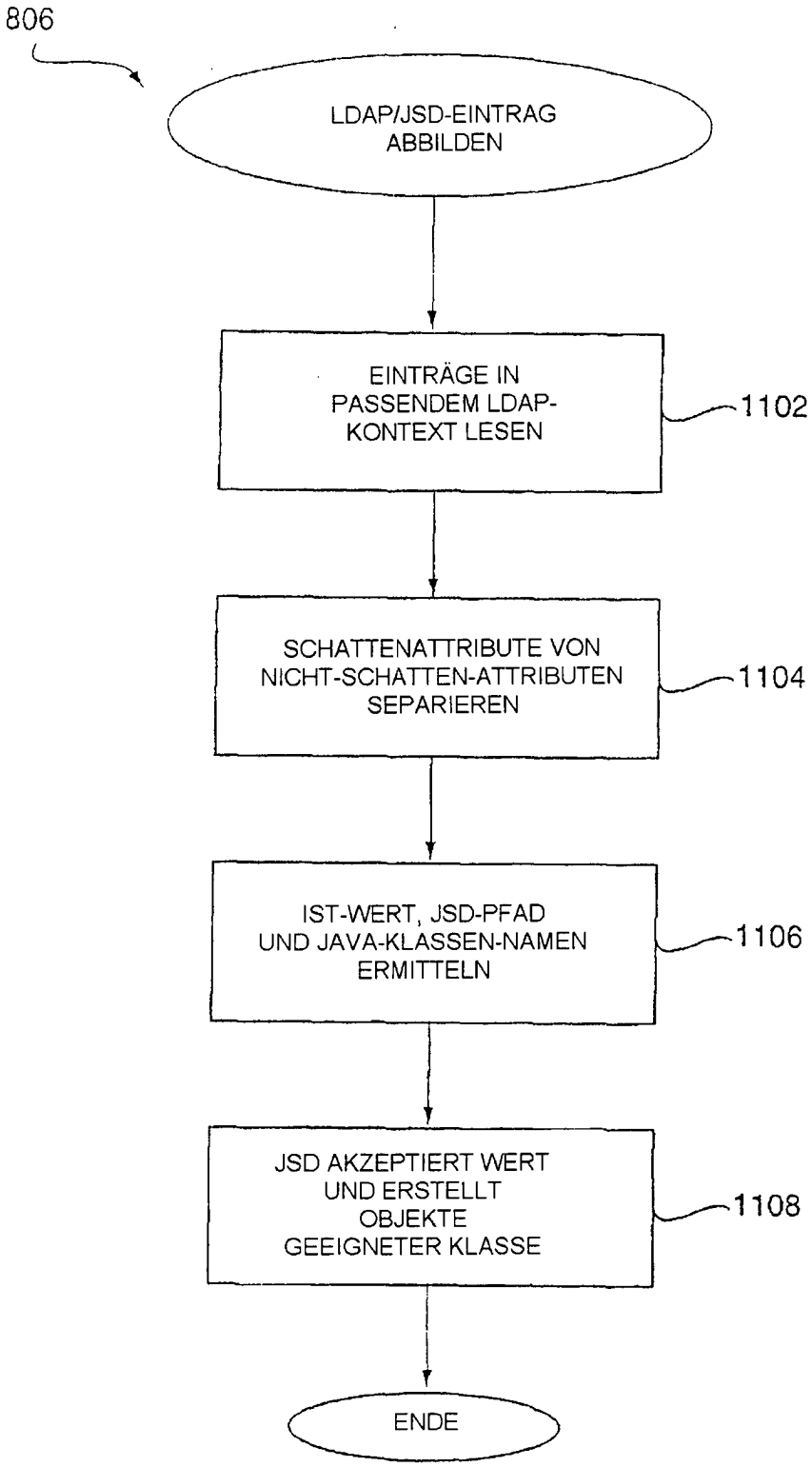
806
LDAP/JSD-EINTRAG ABBILDEN
EINTRÄGE IN PASSENDEM LDAP-KONTEXT LESEN
1102
SCHATTENATTRIBUTE VON NICHT-SCHATTEN-ATTRIBUTEN SEPARIEREN
1104
IST-WERT, JSD-PFAD UND JAVA-KLASSEN-NAMEN ERMITTELN
1106
JSD AKZEPTIERT WERT UND ERSTELLT OBJEKTE GEEIGNETER KLASSE
1108
ENDE

FIG. 11

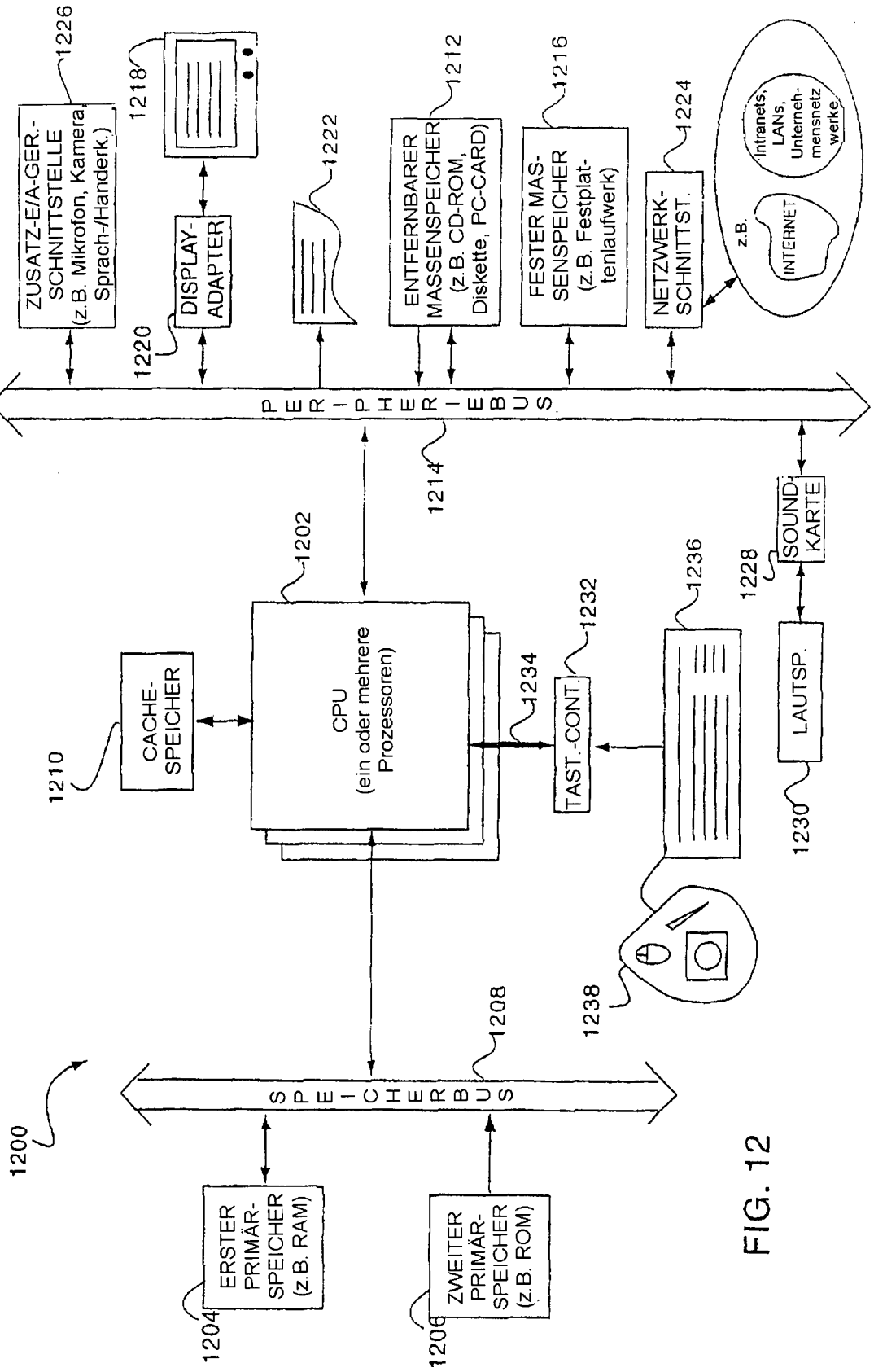
1200
SPEICHERBUS
1208
ERSTER PRIMÄR-SPEICHER (z.B. RAM)
1204
ZWEITER PRIMÄR-SPEICHER (z.B. ROM)
1206
1210
CACHE-SPEICHER
1202
CPU (ein oder mehrere Prozessoren)
1234
TAST.-CONT.
1232
1236
1238
1228
SOUND-KARTE
LAUTSP.
1230
PERIPHERIEBUS
1214
1220
DISPLAY-ADAPTER
1218
1222
ZUSATZ-E/A-GER.-SCHNITTSTELLE (z.B. Mikrofon, Kamera, Sprach-/Handerk.)
1226
ENTFERNBARER MASSENSPEICHER (z.B. CD-ROM, Diskette, PC-CARD)
1212
FESTER MASSENSPEICHER (z.B. Festplattenlaufwerk)
1216
NETZWERK-SCHNITTST.
1224
z.B.
INTERNET
Intranets, LANs, Unternehmensnetzwerke

FIG. 12