(19) Deutsches Patent- und Markenamt

(10) **DE 10 2014 204 830 A1** 2014.09.18

(12) # Offenlegungsschrift

(21) Aktenzeichen: **10 2014 204 830.3**
(22) Anmeldetag: **14.03.2014**
(43) Offenlegungstag: **18.09.2014**

(51) Int Cl.: ***G06F 17/30*** (2006.01)

(30) Unionspriorität:

61/801,297	**15.03.2013**	**US**
14/099,661	**06.12.2013**	**US**

(71) Anmelder:
Palantir Technologies, Inc., Palo Alto, Calif., US

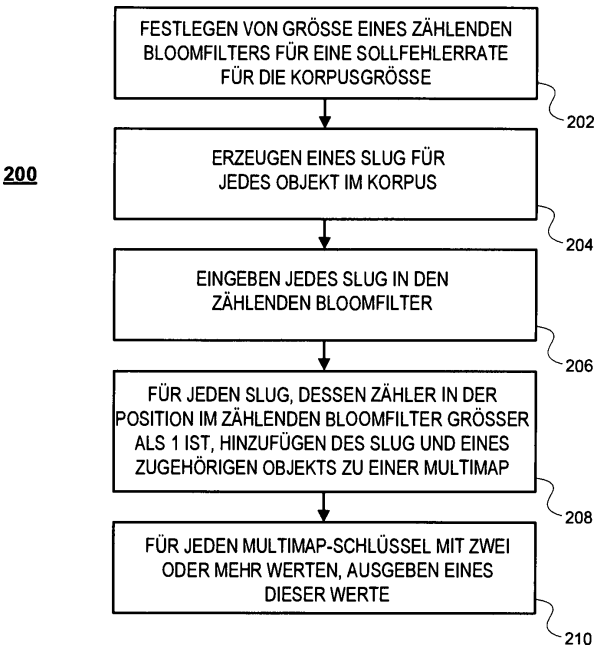
(74) Vertreter:
Dendorfer & Herrmann Patentanwälte Partnerschaft mbB, 80335 München, DE

(72) Erfinder:
Elliot, Mark, Palo Alto, Calif., US; Chang, Allen, Palo Alto, Calif., US

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

(54) Bezeichnung: **Computerimplementierte Systeme und Verfahren zum Vergleichen und Assoziieren von Objekten**

(57) Zusammenfassung: Computerimplementierte Systeme und Verfahren zum Vergleichen und Assoziieren von Objekten werden offenbart. Bei einigen Ausführungsformen wird ein Verfahren bereitgestellt, um ein erstes Objekt mit einem oder mehreren Objekten innerhalb einer Mehrzahl von Objekten zu assoziieren, wobei jedes Objekt eine erste Mehrzahl von Eigenschaften aufweist, jede Eigenschaft Daten aufweist, die ein Kennzeichen einer durch das Objekt repräsentierten Entität widerspiegeln, wobei die assoziierten Objekte übereinstimmende Daten in entsprechenden Eigenschaften für eine zweite Mehrzahl von Eigenschaften aufweisen. Das Verfahren kann beinhalten, dass, für jedes Objekt innerhalb der Mehrzahl von Objekten und für das erste Objekt, folgendes ausgeführt wird: Erzeugen eines Slug für das Objekt, wobei der Slug die zweite Mehrzahl von Eigenschaften von dem Objekt aufweist; und Eingeben des Slug für das Objekt in einen Bloomfilter. Weiter kann das Verfahren beinhalten, dass, für eine Position im Bloomfilter, die dem Slug für das erste Objekt entspricht, eine Assoziierung zwischen Objekten erzeugt wird, deren Slugs der Position entsprechen, falls die Slugs für diese Objekte übereinstimmen.



Beschreibung

[0001] Diese Anmeldung beansprucht die Priorität der provisorischen US-Patentanmeldung Nr. 61/801,297, eingereicht am 15. März 2013, und der US-Patentanmeldung Nr. 14/099,661, eingereicht am 6. Dezember 2013, wobei deren Offenbarungen hiermit durch Bezugnahme vollinhaltlich in das vorliegende Dokument aufgenommen werden.

[0002] Zahlreiche Organisationen, einschließlich Industrie- und Regierungsinstanzen, erkennen, dass wichtige Schlüsse gezogen werden können, wenn riesige Datensätze analysiert werden können, um Verhaltensmuster zu identifizieren, die Gefahren für die öffentliche Sicherheit nahelegen oder illegale Handlungen beweisen. Diese Analysen beinhalten häufig, dass Daten, die mit einer interessierenden Person oder Sache assoziiert sind, mit anderen Daten abgeglichen werden, die mit derselben Person oder Sache assoziiert sind, um zu bestimmen, dass dieselbe Person oder Sache an mehreren Taten beteiligt war, die Sicherheitsbedenken oder strafrechtliche Bedenken hervorrufen.

[0003] Dabei kann jedoch die Qualität des analytischen Ergebnisses, das von einer Verwendung von technisch fortschrittlichen Analysewerkzeugen herrührt, durch die Qualität der Daten, die das Werkzeug verwendet, eingeschränkt sein. Für gewisse Typen von Analysen muss eine akzeptable Fehlerrate buchstäblich oder fast Null sein, damit ein aus den Daten gezogener analytischer Schluss gut fundiert ist. Ein Erzielen dieser Null oder fast Null betragenden Fehlerrate für Datensätze, die einen zwei- oder dreistelligen Millionenwert von einzelnen Datensätzen umfassen, kann problematisch sein. Aktuelle Datenvergleichswerkzeuge sind nicht gut geeignet, um diese Probleme zu lösen.

[0004] Die zuvor erläuterten Probleme sind besonders heftig für Analysen, die Daten beinhalten, welche in Bezug zu einem Identifizieren von Personen oder Sachen für Untersuchungen betreffend die öffentliche Sicherheit stehen. Beispielsweise ist bei Analysewerkzeugen zum Identifizieren potentieller Sicherheitsbedrohungen die akzeptable Fehlerrate generell nicht größer als Null, da der Preis, dass irrtümlich auf ein Vorliegen einer Sicherheitsbedrohung erkannt wird (d. h. „falsch-positiv”) oder dass ein Nicht-Aufspüren einer Sicherheitsbedrohung zugelassen wird (d. h. „falsch-negativ”), unannehmbar hoch ist. Daher ist es erforderlich, dass zur Unterstützung der öffentlichen Sicherheit dienende Werkzeuge Daten, die mit interessierenden Personen oder Sachen assoziiert sind, zu Daten, in mit derselben Person oder Sache in Beziehung stehen, in korrekter Weise in Bezug setzen.

[0005] Es gibt einige Werkzeuge, um einen genauen Datenvergleich vorzunehmen, jedoch sind diese mit Datensätzen, die Millionen von einzelnen Einträgen enthalten, zur Rechnerauswertung praktisch nicht zu verwenden. Beispielsweise besteht eine Lösung, um zu bestimmen, ob zwei spezielle Objekte mit derselben interessierenden Person oder Sache assoziiert sind, darin, jedes Element eines Objekts mit einem entsprechenden Element im zweiten Objekt zu vergleichen. Beispielsweise kann, für Objekte, die M Elemente enthalten, ein erstes Element im ersten Objekt mit einem entsprechenden ersten Element im zweiten Objekt verglichen werden, und entsprechende Vergleiche können für jedes der verbleibenden M – 1 Elemente vorgenommen werden, die den ersten und zweiten Objekten gemeinsam sind. Falls die Elemente in jedem Objekt insgesamt geeignet sind, um die repräsentierte Person oder Sache mit Gewissheit eindeutig zu identifizieren, und entsprechende Elemente in den ersten und zweiten Objekten übereinstimmen, kann in begründeter Weise daraus geschlossen werden, dass die Objekte dieselbe Person oder Sache widerspiegeln. Als Alternative könnte jedes Objekt in eine einzelne Zeichenkette umgewandelt (serialisiert) werden, welche die Inhalte eines jeden zu vergleichenden Elementes widerspiegelt. Danach könnte eine Zeichenkette, die aus dem einen Objekt erzeugt wird, mit einer Zeichenkette, die aus einem anderen Objekt erzeugt wird, als eine Form eines Objektvergleichs verglichen werden.

[0006] Für gewisse Datensätze ist es möglich, dass die zuvor beschriebenen Lösungsansätze wenig Speicher oder Systemressourcen verbrauchen, da die Objekte oder ihre serialisierten Zeichenketten auf Plattenspeicher anstatt im Hauptspeicher gespeichert werden können. Rasch können jedoch die zuvor beschriebenen Lösungsansätze bei großen oder nicht-trivialen Datensätzen praktisch nicht mehr einsetzbar sein. Mit steigender Anzahl von zu vergleichenden Objekten steigt die Anzahl von Vergleichen, und somit die Verarbeitungszeit der Vergleiche exponentiell an; d. h. proportional zu $n^2/2$, wobei n die Anzahl von zu vergleichenden Objekten repräsentiert. Somit kann ein Vergleich von 500 Objekten unter Verwendung eines serialisierten Lösungsansatzes, dessen Bearbeitungszeit näherungsweise als die Zeit angenommen werden kann, um 125.000 Zeichenkettenvergleiche durchzuführen, mittels Rechnerauswertung zu bewältigen sein. Jedoch kann ein Vergleich von 100 Millionen (100M) Datensätzen unter Verwendung dieses Lösungsansatzes, dessen Berechnungszeit näherungsweise als die Zeit zur Durchführung von 5 Billiarden (5e15) Zeichenkettenvergleichen angenommen werden kann, mittels Rechnerauswertung schwer zu bewältigen sein. Zusätzlich kann ein Lesen von Zeichenketten von einem Plattenspeicher, anstatt diese von einem Arbeitsspeicher zu lesen, zusätzliche Berechnungszeit hinzufügen.

[0007] Eine weitere Lösung zum Identifizieren von übereinstimmenden Objekten in einem Korpus von Objekten besteht darin, jedes Objekt in einer Multimap zu speichern. Diese Multimap ist ein assoziatives Datenfeld, das mehrere Werte für jeden Schlüssel speichert. Ein Importieren der Objekte in die Multimap führt dazu, dass Objekte mit den gleichen Elementdaten in einem einzigen Eintrag der Multimap gespeichert werden. Somit erfolgt durch eine Verwendung einer Multimap ein Assoziieren identischer Objekte.

[0008] Ein Nachteil bei einer Verwendung einer Multimap für Objektvergleiche besteht darin, dass die Multimap typischerweise im Hauptspeicher gespeichert wird, aufgrund von den Algorithmus betreffenden Überlegungen, die ein Organisieren von Schlüsseln in der Multimap betrifft, und daher muss eine Objektvergleichseinrichtung ausreichend Hauptspeicher aufweisen, um eine den gesamten Korpus umfassende Multimap im Arbeitsspeicher zu halten. Daher kann eine Multimap-Lösung für Datensätze von 100M Objekten oder mehr unmöglich sein. Ähnliche Nachteile bestehen für jeden Lösungsansatz bei Anwendung auf weitere Objektvergleichsprobleme, beispielsweise ein effizientes Identifizieren von unikalen (einmalig vorkommenden) Objekten in einem Korpus von Objekten und ein effizientes Vergleichen eines einzelnen Objekts mit allen Objekten in einem Korpus von Objekten.

[0009] Keine der Lösungen ist für Datensätze gangbar, die annähernd oder mehr als 100M Objekte umfassen. Dabei sind jedoch Objekt-Datensätze, die 100M oder mehr Objekte umfassen, heutzutage nicht ungewöhnlich. Daher sind die hier beschriebenen Probleme ziemlich real, und es besteht ein Bedarf nach verbesserten Objektvergleichseinrichtungen.

[0010] Die vorliegende Erfindung ist in den unabhängigen Ansprüchen wiedergegeben. Die abhängigen Ansprüche betreffen optionale Merkmale einiger Ausführungsformen der Erfindung.

[0011] Nachfolgend wird Bezug genommen auf die anliegenden Zeichnungen, die beispielhafte Ausführungsformen der vorliegenden Anmeldung darstellen, wobei:

[0012] Fig. 1 ein Ablaufdiagramm eines beispielhaften Prozesses zum Vergleichen eines Zielobjekts mit zumindest einigen Objekten in einem Korpus zeigt, gemäß einigen Ausführungsformen der vorliegenden Offenbarung.

[0013] Fig. 2 ein Ablaufdiagramm eines beispielhaften Prozesses zum Vergleichen aller Objekte in einem Korpus mit allen anderen Objekten im Korpus zeigt, um Übereinstimmungen innerhalb des Korpus zu bestimmen, gemäß einigen Ausführungsformen der vorliegenden Offenbarung.

[0014] Fig. 3 ein Ablaufdiagramm eines beispielhaften Prozesses zum Vergleichen aller Objekte in einem Korpus mit allen anderen Objekten im Korpus zeigt, um unikale Objekte innerhalb des Korpus zu bestimmen, gemäß einigen Ausführungsformen der vorliegenden Offenbarung.

[0015] Fig. 4 eine beispielhafte Rechnerumgebung zeigt, innerhalb der Ausführungsformen der vorliegenden Offenbarung implementiert werden können.

[0016] Nachfolgend wird detailliert Bezug genommen auf die Ausführungsformen, wobei Beispiele für diese in den anliegenden Zeichnungen dargestellt sind. Wann immer möglich, werden gleiche Bezugszeichen in allen Zeichnungen verwendet, um auf gleiche oder ähnliche Teile Bezug zu nehmen.

[0017] Ausführungsformen der vorliegenden Offenbarung können die Nachteile herkömmlicher Objektvergleichseinrichtungen vermeiden, dadurch dass computerimplementierte Systeme und Verfahren bereitgestellt werden, um Objekte in einer Weise zu vergleichen, die einen größeren Berechnungsdurchsatz und einen akzeptablen Speicherverbrauch ermöglicht, ohne dass eine Vergleichsgenauigkeit verringert wird, sowie für Datensatzgrößen, die zuvor praktisch nicht, oder nicht auf akzeptablem Berechnungsdurchsatzniveau zu bewältigen waren.

[0018] Ausführungsformen der vorliegenden Offenbarung betreffen eine Klasse von Berechnungsproblemen, die sich auf einen Objektvergleich beziehen. Ein Element dieser Klasse beinhaltet einen effizienten Objektvergleich eines speziellen Objekts mit einem Korpus von Objekten. Ein weiteres Element dieser Klasse beinhaltet einen effizienten Vergleich von jedem Objekt in einem Korpus mit allen anderen Objekten im Korpus. Ein zusätzliches Element dieser Klasse beinhaltet eine effiziente Identifizierung von unikalen Objekten in einem Korpus von Objekten.

[0019] Die folgende detaillierte Beschreibung beginnt mit einem allgemeinen Überblick eines Objektvergleichs. Einige Beispiele von zu vergleichenden oder zu analysierenden Objekten werden geliefert. Die Beschreibung erläutert dann eine beispielhafte Ausführungsform, die sich mit der zuvor erörterten ersten Problemklasse befasst (d. h. einem effizienten Vergleichen eines einzigen Objekts mit allen Objekten in einem Korpus). Die Beschreibung erweitert dann die Lösung für die erste Problemklasse, um die zuvor erläuterte zweite Problemklasse anzusprechen (d. h. einen effizienten Vergleich eines jeden Objekts in einem Korpus mit allen anderen Objekten im Korpus). Die detaillierte Beschreibung offenbart dann eine Lösung für die dritte Problemklasse (d. h. eine ef-

fiziente Identifikation von unikalen Objekten in einem Korpus von Objekten). Eine Einführung zu Objekten und ein Überblick von Objektvergleichen folgt.

[0020] Mehrere Typen von Objekten existieren auf dem Gebiet der Informatik. Ein auf dem Gebiet der Informatik gut bekannter Objekttyp ist ein Objekt im objektorientierten Sinn. Wikipedia beschreibt ein Objekt dieses Typs als einen Satz von Elementen (d. h. Datenstrukturen) und Verfahren, die Funktionen ähnlich sind. Ohne notwendigerweise dieser grob vereinfachenden Beschreibung beizupflichten, sind Ausführungsformen, welche die hier erörterten Objektvergleichslösungen implementieren, kompatibel zu einem Vergleichen von Objekten dieses Typs.

[0021] Ein weiterer Objekttyp auf dem Gebiet der Informatik ist eine Datenstruktur, welche die Eigenschaften einer Person oder einer Sache widerspiegelt, die für eine spezielle Aufgabe oder Datenverarbeitungsumgebung relevant ist. Bei einigen Ausführungsformen werden diese Eigenschaften durch Zeichenketten (Strings) widergespiegelt. Bei weiteren Ausführungsformen können Eigenschaften durch Zeichenketten, Ganzzahlen, reelle Zahlen, Zeit- oder Datumsangaben, Binärwerte, Strukturen im Sinn der C-Programmierung, Variablenaufzählungen und/oder weitere Datenformen widergespiegelt werden. Bei einigen Ausführungsformen können Eigenschaften jedweden Objekttyps vor einem Vergleichen in Zeichenketten umgewandelt werden. Bei weiteren Ausführungsformen können möglicherweise einige Eigenschaften Zeichenketten sein oder können möglicherweise in Zeichenketten umgewandelt werden, hingegen können möglicherweise andere Eigenschaften keine Zeichenketten sein und können möglicherweise nicht in Zeichenketten umgewandelt werden. Die Ausführungsformen der vorliegenden Offenbarung können mit Zeichenkette- oder Nicht-Zeichenkette-Eigenschaften arbeiten.

[0022] Außerdem ist die Vorstellung von einer „Datenstruktur" in diesem Kontext sehr flexibel. Der Begriff „Datenstruktur" kann einen beliebigen Typ von strukturierten Daten widerspiegeln, und zwar von in einer Datenbank gespeicherter Information (mit Tabellenspalten, die Elemente in einem Objekt oder einer Datenstruktur widerspiegeln, und Tabellenzeilen, die Instanzen des Objekts oder der Datenstruktur widerspiegeln), und weiter zu formatiertem Text in einer Textdatei (beispielsweise Daten in einer XML-Struktur), bis hin zu Daten, die in einem ablaufenden Computerprogramm gespeichert sind. Demgemäß umfassen, da eine Datenstruktur die zuvor beschriebenen Typen von strukturierten Daten grob umfasst, Objekte ebenfalls grob diese Typen von strukturierten Daten. Außerdem sind die hier erläuterten Objektvergleichslösungen ebenfalls kompatibel mit einem Vergleichen von Objekten dieser Typen.

[0023] Bei einigen Ausführungsformen beinhaltet ein effektiver Objektvergleich, dass man in Betracht zieht, welche Eigenschaften der zu vergleichenden Objekte für ein Durchführen des Vergleichs relevant sind, da die Entitäten (z. B. Personen oder Sachen), die durch diese Objekte widergespiegelt werden, in unterschiedlichen Umgebungen unterschiedliche relevante Eigenschaften haben können. Beispielsweise kann ein Objekt Eigenschaften eines Automobils speichern, die für ein Kraftfahrzeugsamt eines Staates relevant sein können, und zwar durch Speichern der folgenden Information: Fahrzeugidentifizierungsnummer (VIN), Herstellungsjahr, Marke, Modell, Ablaufdatum der Fahrzeugzulassung und eine direkte oder indirekte Angabe der Person, die Eigentümer des Fahrzeugs ist.

[0024] Für Automobile, die auf einer Auktions-Website wie beispielsweise eBay verkauft werden, können jedoch die relevanten Eigenschaften eines Automobils sich von denen unterscheiden, die für das Kraftfahrzeugsamt des Staates relevant sind. Beispielsweise kann eine Datenstruktur zum Speichern von Eigenschaften eines Automobils, das für einen Verkauf auf eBay gelistet ist, beinhalten: VIN, Jahr, Marke, Modell, Kilometerstand, Zustand des Automobils, minimales Auktionsgebot und eine direkte oder indirekte Angabe der Person, die das zum Verkauf stehende Fahrzeug eingestellt hat. Somit können Eigenschaften einer Entität (z. B. eine Person oder Sache), die für die eine Umgebung relevant ist, sich von Eigenschaften der Entität unterscheiden, die für eine andere Umgebung relevant sind. Demgemäß können Eigenschaften eines Objekts, die während eines Objektvergleichs in der einen Umgebung berücksichtigt werden, sich von denen unterscheiden, die während eines Objektvergleichs in einer zweiten Umgebung berücksichtigt werden.

[0025] Bei einigen Ausführungsformen kann ein effektiver Datenvergleich auch beinhalten, dass berücksichtigt wird, welche Eigenschaften darauf abzielen, eine Entität (z. B. eine Person oder Sache) von anderen Instanzen der Entität zu unterscheiden. Beispielsweise sollte, gemäß Auslegung, eine VIN für ein Automobil unikal (einmalig vorkommend) für dieses Automobil sein. Jedoch können gelegentlich Situationen auftreten, bei denen eine VIN nicht unikal für ein spezielles Automobil ist. Derartige Situationen können von absichtlichen Fehlern oder zufälligen Fehlern herrühren. Ein Beispiel eines absichtlichen Fehlers ist ein Versuch einer betrügerischen Registrierung eines gestohlenen Fahrzeuges unter einer vorgeblichen VIN. Ein Beispiel eines zufälligen Fehlers tritt auf, wenn ein mit Smog-Überprüfung befasster Angestellter eine VIN inkorrekt in einen Computer an einer Smog-Überprüfungsstation eingibt, was zu einem Smog-Überprüfungs-Datensatz mit inkorrekter VIN führt, die anschließend an eine Datenbank eines Staates weitergegeben wird. Datenfehler kom-

men bei realen Datenverarbeitungsumgebungen vor, und daher führen einige Ausführungsformen der vorliegenden Offenbarung ein Minimieren oder Eliminieren von Fehlern durch, und zwar dadurch, dass Objekte mittels einer Kombination aus mehreren Objekteigenschaften identifiziert werden, anstatt Objekte mittels Verwendung einer einzelnen Objekteigenschaft zu identifizieren.

[0026] Bei einigen Ausführungsformen werden eine oder mehrere Identifizierungseigenschaften eines Objekts aus dem Objekt extrahiert und in einer Datenstruktur gespeichert. Diese Datenstruktur wird als „Slug" bezeichnet; sie enthält Information, die ausreichend sein kann, um eine Entität (z. B. eine Person oder Sache) mit einem gewissen Grad an Informationsredundanz eindeutig zu identifizieren, um ein Erfassen von Fehlern in den Eigenschaften im Slug zu ermöglichen. Bei einigen Ausführungsformen weist der Slug eine Verkettung von Zeichenketten auf, die durch ein Trennzeichen getrennt sind. Bei einigen Ausführungsformen ist das Trennzeichen ein NULL-Zeichen, hingegen kann bei weiteren Ausführungsformen das Trennzeichen ein Zeichen sein, dass ansonsten nicht in der verketteten Zeichenkette vorhanden ist. Bei einigen Ausführungsformen können die verketteten Zeichenketten durch eine Trenn-Zeichenkette (z. B. „--") anstelle eines Trennzeichens begrenzt sein. Bei Ausführungsformen, die eine Trenn-Zeichenkette verwenden, kann die Trenn-Zeichenkette eine beliebige Zeichenkette sein, die ansonsten nicht in den Zeichenketten vorkommt, die verkettet wurden. Bei weiteren Ausführungsformen weist der Slug eine Datenstruktur wie beispielsweise ein Objekt, ein Datenfeld (Array), eine Struktur oder ein assoziatives Datenfeld (assoziatives Array) auf.

[0027] Beispielsweise kann bei einer Ausführungsform ein Slug für ein Automobil Eigenschaften enthalten, die eine VIN, eine Marke, ein Modell und eine Jahresangabe für das Automobil beinhalten. Dadurch dass die Eigenschaften Marke, Modell und Jahr für das Automobil im Slug enthalten ist, wird eine Fähigkeit für ein Erfassen von Fehlern in der VIN-Eigenschaft bereitgestellt, da die VIN-Eigenschaft nicht die einzige Objekteigenschaft ist, die verglichen wird. Damit Slugs, die mit zwei Automobilen assoziiert sind, bei Vorliegen eines Fehlers in der VIN-Eigenschaft des einen Automobil-Objekts, übereinstimmen, muss ein Automobil-Objekt mit der gleichen VIN-Eigenschaft wie die fehlerhafte VIN ebenfalls die gleichen Marken-, Modell- und Jahr-Eigenschaften haben.

[0028] Die Chancen dieser zufälligen Übereinstimmung mehrerer Eigenschaften zwischen zwei oder mehr Objekten kann verschwindend gering sein. Daher sollte ein Einbeziehen eines gewissen Grades an Informationsredundanz irrtümliche Übereinstimmungen bei einem Objektvergleich vermeiden oder zumindest beträchtlich verringern, in Bezug auf Objektvergleiche, bei denen lediglich eine einzelne Eigenschaft zwischen Objekten verglichen wird, ungeachtet der Tatsache, dass beabsichtigt wurde, dass die einzelne Eigenschaft ihre zugehörige Entität (z. B. Person oder Sache) eindeutig identifiziert.

[0029] Beispielhafte Ausführungsformen werden nachfolgend beschrieben, die das zuvor erläuterte erste Problem lösen, d. h. ein effizientes Vergleichen eines speziellen Objekts (nachfolgend als „Zielobjekt" bezeichnet) mit allen Objekten in einem Korpus. Die offenbarten Ausführungsformen verwenden einen Bloomfilter, um Slugs zu identifizieren, die mit Objekten im Korpus assoziiert sind, die nicht mit dem Slug für das Zielobjekt übereinstimmen. Diese Schnellerkennung wird dadurch durchgeführt, dass Slugs verworfen werden, die mit einer Position im Bloomfilter assoziiert werden, welche von der Position verschieden ist, die mit dem Slug für das Zielobjekt assoziiert ist.

[0030] Bloomfilter haben die Eigenschaft, dass zwei Slugs, die in unterschiedliche Positionen im Bloomfilter fallen, mit Gewissheit unterschiedliche Eigenschaften haben und somit unterschiedliche Objekte widerspiegeln. Daher stimmt, wenn der Slug für das Zielobjekt nicht in die gleiche Position wie der Slug für ein spezielles Objekt im Korpus fällt, das Zielobjekt nicht mit dem speziellen Objekt im Korpus überein und kann somit bei derartigen Ausführungsformen von einer zukünftigen Berücksichtigung ausgeschlossen werden.

[0031] Fig. 1 zeigt ein Ablaufdiagramm eines beispielhaften Prozesses **100** zum Vergleichen eines Zielobjekts mit zumindest einigen Objekten in einem Korpus, gemäß einigen Ausführungsformen der vorliegenden Offenbarung. Bei einigen Ausführungsformen ist das Zielobjekt, das mit zumindest einigen Objekten im Korpus verglichen werden soll, Teil des Korpus. Bei diesen Ausführungsformen wird ein Vergleich zwischen dem Zielobjekt und allen anderen Objekten in dem Korpus durchgeführt. Bei weiteren Ausführungsformen ist das Objekt, das mit zumindest einigen Objekten im Korpus verglichen werden soll, nicht Teil des Korpus. Bei diesen weiteren Ausführungsformen wird ein Vergleich zwischen dem Zielobjekt und allen Objekten in dem Korpus durchgeführt.

[0032] Wie dargestellt, wird bei Schritt **102** die Größe eines Bloomfilters festgelegt und dieser unter Berücksichtigung der Fehlerrate erstellt, welche sich für die zu bearbeitende Korpusgröße ergibt. Beispielsweise kann ein Vergrößern der Anzahl von Positionen in einem Bloomfilter darauf abzielen, die Fehlerrate für eine spezifische Korpusgröße zu verringern, hingegen kann ein Verringern der Anzahl von Positionen in einem Bloomfilter darauf abzielen, die Fehlerrate für eine spezifische Korpusgröße zu vergrößern. Methoden zur Größenbestimmung eines Bloomfilters, um

eine Sollfehlerrate für eine spezifische Korpusgröße zu erzielen, sind in der Technik allgemein bekannt, und daher werden diese Methoden hier nicht erörtert.

[0033] Bei Schritt **104** wird ein Slug für das Zielobjekt (d. h. das Objekt, gegen das alle Objekte im Korpus verglichen werden) erzeugt. Überlegungen, welche Eigenschaften für ein Objekt in einen Slug einzuschließen sind, wurden zuvor erörtert. Bei Schritt **106** wird eine Bloomfilter-Position bestimmt, die dem Slug für das Zielobjekt entspricht. Bei einigen Ausführungsformen kann eine Bloomfilter-Position für einen Slug dadurch bestimmt werden, dass der Slug in einen Bloomfilter eingegeben wird und der Bloomfilter angewiesen wird, die Position zu offenbaren, in welcher der Slug hinzugefügt wurde.

[0034] Bei weiteren Ausführungsformen kann eine Bloomfilter-Position für einen Slug dadurch bestimmt werden, dass der Slug als eine Eingabe für eine Softwarefunktion präsentiert wird, die mit dem Bloomfilter assoziiert ist, ohne den Slug im Bloomfilter zu speichern. Bei weiteren Ausführungsformen kann eine Position für einen Slug dadurch bestimmt werden, dass der Slug in eine Softwarefunktion eingegeben wird, die einen Positionsauswahlalgorithmus für einen Bloomfilter widerspiegelt, bei Abwesenheit einer Verwendung eines tatsächlichen und/oder vollständigen Bloomfilters, und die Bloomfilter-Position als Ausgabewert dieser Softwarefunktion erhalten wird. Bei weiteren Ausführungsformen können andere Lösungsansätze verwendet werden, um von einem Slug eine Bloomfilter-Position zu liefern. Auf diese Lösungsansätze zum Identifizieren eines Bloomfilters für einen Slug, gemäß den zuvor erläuterten Ausführungsformen, wird insgesamt in Schritten **106**, **108** Bezug genommen. Die bestimmte Bloomfilter-Position wird verwendet, um Übereinstimmungen des Slug-Vergleichs zu identifizieren, von denen einige „falsch-positiv" sein können, wobei der nachstehend erläuterte Bloomfilter verwendet wird.

[0035] Bei Schritt **108** wird ein Slug für jedes Objekt im Korpus erzeugt. Bei Schritt **110** wird eine Bloomfilter-Position für jedes Objekt im Korpus bestimmt. Bei einigen Ausführungsformen kann eine Bloomfilter-Position für ein Objekt dadurch bestimmt werden, dass der Slug des Objekts in den Bloomfilter eingegeben wird und der Bloomfilter angewiesen wird, die Position zu offenbaren, zu welcher der Slug hinzugefügt wurde.

[0036] Nach Abschluss von Schritt **110** spiegeln Slugs, die der bei Schritt **108** identifizierten Position entsprechen, Übereinstimmungen mit dem Slug für das Zielobjekt wider. Einige dieser Übereinstimmungen können jedoch falsch-positive Übereinstimmungen sein, anstelle von echten Übereinstimmungen. Daher filtern Schritte **112** und **114** diese falsch-positiven Übereinstimmungen mittels Verwendung einer Multimap aus.

[0037] In Schritt **112** wird, für jeden Slug, der einem Objekt im Korpus zugehörig ist und dessen Position im Bloomfilter die gleiche Position wie die des Slug für das Zielobjekt ist, der einem Objekt im Korpus zugehörige Slug und sein zugehöriges Objekt im Korpus einer Multimap hinzugefügt. Beim Hinzufügen des Slug und dessen zugehörigen Objekts zur Multimap repräsentiert der Slug den Schlüssel zu der Multimap, und das Objekt im Korpus repräsentiert den Wert zu der Multimap. Diese Multimap wird verwendet, um falsch-positive Ergebnisse aus der Verarbeitung zu entfernen. In Schritt **114** wird das Verfahren durch Auswählen der echt-positiven Übereinstimmungen, die in der Multimap identifiziert sind, abgeschlossen. Diese nicht-falsch-positiven Übereinstimmungen können aus der Multimap abgerufen werden, dadurch dass, mit dem Slug für das Zielobjekt als Schlüssel, Daten aus der Multimap ausgelesen werden.

[0038] Bei einigen Ausführungsformen kann der Prozess **100** über mehrere Prozessoren verteilt werden. Beispielsweise kann ein Bloomfilter auf jedem von mehreren Prozessoren vorhanden sein, und Schritte **102** bis **114** können auf jedem der mehreren Prozessoren ausgeführt werden. Der Korpus von Objekten kann unter den verschiedenen Prozessoren so verteilt werden, dass alle Objekte durch einen Prozessor verarbeitet werden, jedoch kein Objekt durch mehr als einen Prozessor verarbeitet wird. Bei derartigen Ausführungsformen führt jeder der mehreren Prozessoren ein Ausgeben eines Teils der mit dem Zielobjekt übereinstimmenden Objekte im Korpus durch.

[0039] Beispielhafte Ausführungsformen werden nachfolgend beschrieben, die das zuvor erläuterte zweite Problem lösen, d. h. ein effizientes Vergleichen aller Objekte mit allen Objekten in einem Korpus. Diese Ausführungsformen verwenden einen zählenden Bloomfilter, um ein schnelles Identifizieren von Slugs vorzunehmen, die mit Objekten im Korpus, welche nicht mit dem Slug für das Zielobjekt übereinstimmen, assoziiert sind. Zählende Bloomfilter sind in der Technik allgemein bekannt, und daher wird ihre Struktur und ihr Aufbau hier nicht erörtert.

[0040] Speziell könnte, falls eine Position im zählenden Bloomfilter einen Wert von Null oder Eins hat, nachdem Slugs für alle Objekte in dem Korpus in den Bloomfilter eingegeben wurden, kein Objekt, dessen Slug mit dieser Position assoziiert ist, mit einem anderen Slug übereinstimmen, und daher werden diese Slugs von weiterer Berücksichtigung ausgeschlossen. Diese Slugs können ausgeschlossen werden, da für Fachleute klar ist, dass Bloomfilter falsch-positive Ergebnisse haben können, sie jedoch keine falsch-

negativen Ergebnisse haben können. Daher spiegelt ein zählender Bloomfilter, dessen Zählwert geringer als Zwei ist, eine genaue Bestimmung wider, dass keine Übereinstimmung zwischen Slugs vorhanden ist, die mit dieser Position assoziiert sind, da jegliche Übereinstimmung einen Zählwert von mindestens Zwei erzeugen würde. Jedoch können falsch-positive Ergebnisse unter Objekten vorkommen, deren Slugs mit der gleichen Bloomfilter-Position assoziiert sind, und daher können falsch-positive Ergebnisse mittels einer zusätzlichen Verarbeitung entfernt werden, wie später noch erläutert wird.

[0041] Fig. 2 zeigt ein Ablaufdiagramm eines beispielhaften Prozesses **200** zum Vergleichen aller Objekte in einem Korpus mit allen anderen Objekten in dem Korpus, um Übereinstimmungen innerhalb des Korpus zu bestimmen, gemäß einigen Ausführungsformen der vorliegenden Offenbarung. Wie dargestellt, wird in Schritt **202** die Größe eines zählenden Bloomfilters festgelegt und dieser unter Berücksichtigung der Fehlerrate erstellt, die für die zu verarbeitende Korpusgröße resultiert. Beispielsweise kann ein Vergrößern der Anzahl von Positionen in einem zählenden Bloomfilter darauf abzielen, die Fehlerrate für eine spezifische Korpusgröße zu verringern, hingegen kann ein Verringern der Anzahl von Positionen in einem zählenden Bloomfilter darauf abzielen, die Fehlerrate für eine spezifische Korpusgröße zu vergrößern. Methoden zur Größenbestimmung eines zählenden Bloomfilters, um eine Sollfehlerrate für eine spezifische Korpusgröße zu erzielen, sind in der Technik allgemein bekannt, und daher werden diese Methoden hier nicht erörtert.

[0042] Bei einigen Ausführungsformen kann der zählende Bloomfilter einen N-Bit-Zähler aufweisen, und diese Zähler können als Zwei-Bit-Zähler implementiert sein (d. h. N = 2). Bei weiteren Ausführungsformen können diese Zähler Ein-Bit-Zähler oder Zähler aus mehr als zwei Bit sein. Bei noch weiteren Ausführungsformen können diese Zähler Sättigungszähler sein; d. h. diese Zähler zählen bis zu einem Maximalwert hoch und überschreiten diesen Wert dann nicht.

[0043] In Schritt **204** wird ein Slug für jedes Objekt im Korpus erzeugt. In Schritt **206** wird jeder Slug in den zählenden Bloomfilter eingegeben, was bewirkt, dass ein Zähler in einer einem Slug entsprechenden Position hochgezählt wird. Nach Abschluss von Schritt **206** spiegeln Positionen, deren Zähler einen Wert größer als Eins aufweisen, einen oder mehrere übereinstimmende Slugs wider. Einige dieser Übereinstimmungen können jedoch falsch-positive Übereinstimmungen, anstelle von echten Übereinstimmungen sein. Daher filtern Schritte **208** und **210** die falsch-positiven Übereinstimmungen unter Verwendung einer Multimap aus.

[0044] In Schritt **208** werden, für Slugs, die mit Positionen im zählenden Bloomfilter assoziiert sind, deren Zähler einen Wert größer als Eins aufweisen, der Slug und sein zugehöriges Objekt einer Multimap hinzugefügt. Beim Hinzufügen des Slug und seines zugehörigen Objekts zur Multimap repräsentiert der Slug den Schlüssel zu der Multimap, und das Objekt im Korpus repräsentiert den Wert zu der Multimap. Diese Multimap wird verwendet, um falsch-positive Ergebnisse aus der Verarbeitung zu entfernen. In Schritt **210** wird der Prozess **200** abgeschlossen, durch Ausgeben eines Wertes für jeglichen Schlüssel in der Multimap, der zwei oder mehr Werte aufweist. Die ausgegebenen Werte spiegeln Objekte wider, deren Slugs mit Slugs von mindestens einem anderen Objekt im Korpus übereinstimmten. Somit identifizieren die ausgegebenen Objekte solche Objekte, bei denen ausgewählte Eigenschaften, wie in einem Slug eines Objekts widergespiegelt, in eindeutiger Weise mit mindestens einem weiteren Objekt im Korpus übereinstimmen.

[0045] Bei einigen Ausführungsformen kann der Prozess **200** über mehrere Prozessoren verteilt werden. Beispielsweise kann ein zählender Bloomfilter auf jedem von mehreren Prozessoren vorhanden sein, und Schritte **202**, **204** und **206** können auf jedem der mehreren Prozessoren ausgeführt werden. Der Korpus von Objekten kann unter den verschiedenen Prozessoren so verteilt werden, dass alle Objekte durch einen Prozessor verarbeitet werden, jedoch kein Objekt durch mehr als einen Prozessor verarbeitet wird. Bei derartigen Ausführungsformen erfolgt, vor einem Ausführen von Schritt **208**, ein Aufsummieren von Zählern für jede Position im zählenden Bloomfilter mit Zählern für die gleiche Position in zählenden Bloomfiltern auf anderen Prozessoren. Danach fährt der Prozess **200** mit einem Ausführen von Schritten **208** und **210** auf einem einzelnen Prozessor fort.

[0046] Beispielhafte Ausführungsformen werden nachfolgend beschrieben, die das zuvor erläuterte dritte Problem lösen, d. h. ein effizientes Identifizieren von unikalen Objekten in einem Korpus. Diese Ausführungsformen verwenden einen zählenden Bloomfilter und eine Multimap, um ein rasches Identifizieren von unikalen Objekten durchzuführen. Wenn Slugs für alle Objekte im Korpus in den zählenden Bloomfilter eingegeben werden, spiegelt jede Position mit einem Zählwert von Eins ein unikales Objekt wider, da Bloomfilter keine falsch-negativen Ergebnisse erzeugen. Zusätzlich könnten, in dem Maße, dass Positionen Zählwerte von Zwei oder mehr aufweisen, diese Zählwerte falsch-positive Ergebnisse widerspiegeln. Daher ermöglicht eine Multimap eine Bestimmung, ob die Übereinstimmungen, die in den Zählwerten widergespiegelt sind, falsch- oder echt-positiv sind.

[0047] Fig. 3 zeigt ein Ablaufdiagramm eines beispielhaften Prozesses **300** zum Vergleichen aller Objekte in einem Korpus mit allen anderen Objekten in dem Korpus, um unikale Objekte innerhalb des Korpus zu bestimmen, gemäß einigen Ausführungsformen der vorliegenden Offenbarung. Wie dargestellt, wird in Schritt **302** die Größe eines zählenden Bloomfilters festgelegt und dieser unter Berücksichtigung der Fehlerrate erstellt, die für die zu verarbeitende Korpusgröße resultiert. Beispielsweise kann ein Vergrößern der Anzahl von Positionen in einem zählenden Bloomfilter darauf abzielen, die Fehlerrate für eine spezifische Korpusgröße zu verringern, hingegen kann ein Verringern der Anzahl von Positionen in einem zählenden Bloomfilter darauf abzielen, die Fehlerrate für eine spezifische Korpusgröße zu vergrößern. Methoden zur Größenbestimmung eines zählenden Bloomfilters, um eine Sollfehlerrate für eine spezifische Korpusgröße zu erzielen, sind in der Technik allgemein bekannt, und daher werden diese Methoden hier nicht erörtert.

[0048] Bei einigen Ausführungsformen kann der zählende Bloomfilter einen N-Bit-Zähler aufweisen und diese Zähler können als Zwei-Bit-Zähler implementiert sein (d. h. N = 2). Bei weiteren Ausführungsformen können diese Zähler Ein-Bit-Zähler oder Zähler aus mehr als zwei Bit sein. Bei noch weiteren Ausführungsformen können diese Zähler Sättigungszähler sein; d. h. diese Zähler zählen bis zu einem Maximalwert hoch und überschreiten diesen Wert dann nicht.

[0049] In Schritt **304** wird ein Slug für jedes Objekt im Korpus erzeugt. In Schritt **306** wird jeder Slug in den zählenden Bloomfilter eingegeben, was bewirkt, dass ein Zähler in einer einem Slug entsprechenden Position hochgezählt wird. Wie zuvor erläutert, wird, nachdem Slugs für alle Objekte im Korpus in den zählenden Bloomfilter eingegeben wurden, durch jegliche Position mit einem Zählwert von Eins ein unikales Objekt im Korpus widergespiegelt, da der zählende Bloomfilter keine falsch-negativen Ergebnisse erzeugt. Daher wird in Schritt **308** für jeden Slug, dessen Zählwert im zählenden Bloomfilter Eins ist, das dem Slug zugehörige Objekt als unikales Objekt im Korpus ausgegeben.

[0050] Nach Abschluss von Schritt **308** spiegeln Positionen, deren Zählwerte einen Wert größer als Eins aufweisen, einen oder mehrere übereinstimmende Slugs wider; d. h. Slugs, die nicht unikal sind. Einige dieser Übereinstimmungen können jedoch falsch-positive Übereinstimmungen anstelle von echten Übereinstimmungen sein, aufgrund der Beschaffenheit von Bloomfiltern, wie zuvor erläutert wurde. Daher werden in Schritten **310** und **312** die falsch-positiven Übereinstimmungen unter Verwendung einer Multimap ausgefiltert.

[0051] In Schritten **310** und **312** wird bestimmt, ob der zählende Bloomfilter das Vorhandensein von weiteren unikalen Objekten maskiert, da der Bloomfilter falsch-positive Ergebnisse zulässt. In Schritt **310** wird, für jeden Slug, dessen assoziierte Position einen Zählwert größer als Eins aufweist, der Slug als Schlüssel zu einer Multimap eingegeben, und das dem Slug zugehörige Objekt wird als Wert zu diesem Schlüssel eingegeben. In Schritt **312** endet der Prozess, nachdem jeweils der Wert in der Multimap für Schlüssel ausgegeben wird, die lediglich einen einzigen Wert haben. Unikale Objekte im Korpus werden durch die Sammlung von Objekten, die von Schritt **308** ausgegeben werden, und die Sammlung von Objekten, die durch Schritt **312** ausgegeben werden, reflektiert, da Ersterer Objekte widerspiegelt, deren Slugs der alleinige Slug in einer Position im zählenden Bloomfilter waren, und die daher unikal unter Slugs waren, die mit Objekten im Korpus assoziiert sind, wohingegen Letzterer Slugs widerspiegelt, die falsch-positive Ergebnisse im zählenden Bloomfilter waren, für die jedoch mittels der Multimap Eindeutigkeit hergestellt wurde.

[0052] Bei einigen Ausführungsformen kann der Prozess **300** über mehrere Prozessoren verteilt werden. Beispielsweise kann ein zählender Bloomfilter auf jedem von mehreren Prozessoren vorhanden sein, und Schritte **302**, **304** und **306** können auf jedem der mehreren Prozessoren ausgeführt werden. Der Korpus von Objekten kann unter den verschiedenen Prozessoren so verteilt werden, dass alle Objekte durch einen Prozessor verarbeitet werden, jedoch kein Objekt durch mehr als einen Prozessor verarbeitet wird. Bei derartigen Ausführungsformen erfolgt, vor einem Ausführen von Schritt **308**, ein Aufsummieren von Zählern für jede Position im zählenden Bloomfilter mit Zählern für die gleiche Position in zählenden Bloomfiltern auf anderen Prozessoren. Danach fährt der Prozess **300** mit einem Ausführen von Schritten **308**, **310** und **312** auf einem einzelnen Prozessor fort.

[0053] Fig. 4 zeigt eine beispielhafte Rechnerumgebung, in der die Ausführungsformen der vorliegenden Offenbarung implementiert werden können.

[0054] Computersystem **400** beinhaltet einen Bus **402** oder einen anderen Kommunikationsmechanismus für ein Weiterleiten von Information, und einen Hardware-Prozessor **404**, der mit dem Bus **402** zum Verarbeiten von Information verbunden ist. Bei einigen Ausführungsformen kann der Hardwareprozessor **404** beispielsweise ein Universal-Mikroprozessor sein, oder es kann ein Mikroprozessor mit reduziertem Befehlssatz sein.

[0055] Das Computersystem **400** beinhaltet auch einen Hauptspeicher **406**, beispielsweise ein RAM (Direktzugriffsspeicher) oder eine andere dynamische

Speichervorrichtung, die mit dem Bus **402** verbunden ist, um Information und Anweisungen zu speichern, die durch den Prozessor **404** auszuführen sind. Der Hauptspeicher **406** kann auch verwendet werden, um temporäre Variablen oder andere Zwischeninformationen zu speichern, während eines Ausführens von Anweisungen durch den Prozessor **404**. Derartige Anweisungen machen, wenn sie in nicht-transitorischen Speichermedien gespeichert werden, die dem Prozessor **404** zugänglich sind, das Computersystem **400** zu einer Spezialmaschine, die kundenspezifisch angepasst ist, um die in den Anweisungen festgelegten Operationen auszuführen.

[0056] Bei einigen Ausführungsformen beinhaltet das Computersystem **400** weiter ein ROM (Nur-Lese-Speicher) **408** oder eine andere statische Speichervorrichtung, die mit dem Bus **402** verbunden ist, um statische Information und Anweisungen für den Prozessor **404** zu speichern. Eine Speichervorrichtung **410**, wie beispielsweise eine Magnetplatte oder eine optische Platte, ist vorgesehen und mit dem Bus **402** verbunden, um Information und Anweisungen zu speichern.

[0057] Das Computersystem **400** kann über den Bus **402** mit einer Anzeige **412** verbunden sein, beispielsweise einer Kathodenstrahlröhre (CRT) oder einem LCD-Bildschirm, um einem Benutzer eines Computers Information anzuzeigen. Eine Eingabevorrichtung **414**, die alphanumerische und weitere Tasten beinhaltet, ist mit dem Bus **402** verbunden, um Information und ausgewählte Befehle an den Prozessor **404** zu übermitteln. Ein weiterer Typ von Benutzereingabevorrichtung ist eine Cursor-Steuerung **416**, wie beispielsweise eine Maus, ein Trackball oder Cursor-Richtungstasten, um dem Prozessor **404** Richtungsinformation und ausgewählte Befehle mitzuteilen und eine Cursorbewegung auf der Anzeigeeinrichtung **412** zu steuern. Diese Eingabevorrichtung hat typischerweise zwei Freiheitsgrade in zwei Achsen, einer ersten Achse (beispielsweise x) und einer zweiten Achse (beispielsweise y), was der Vorrichtung ermöglicht, Positionen in einer Ebene zu bezeichnen.

[0058] Im Computersystem **400** können die hier beschriebenen Prozesse und Verfahren unter Verwendung von kundenspezifischen festverdrahteten Logikschaltungen, einem oder mehreren ASICs oder FPGAs, Firmware und/oder Programmlogik implementiert sein, die in Kombination mit dem Computersystem bewirken oder programmieren, dass das Computersystem **400** eine Spezialmaschine ist. Bei einigen Ausführungsbeispielen werden die hier beschriebenen Prozesse und Verfahren durch das Computersystem **400** reagierend darauf durchgeführt, dass der Prozessor **404** eine oder mehrere Sequenzen von einer oder mehreren im Hauptspeicher **406** enthaltenen Anweisungen ausführt. Derartige Anweisungen können in den Hauptspeicher **406** aus einem anderen Speichermedium, beispielsweise der Speichervorrichtung **410**, eingelesen werden. Eine Ausführung der im Hauptspeicher **406** enthaltenen Anweisungssequenzen veranlasst den Prozessor **404**, die hier beschriebenen Prozessschritte auszuführen. Bei weiteren Ausführungsbeispielen können festverdrahtete Schaltkreise verwendet werden, anstelle von Software-Anweisungen oder in Kombination mit diesen.

[0059] Der Begriff „Speichermedien", wie hier verwendet, bezieht sich auf jegliche nicht-transitorische Medien, die Daten und/oder Anweisungen speichern, welche eine Maschine veranlassen, in spezifischer Weise zu arbeiten. Derartige Speichermedien können nicht-flüchtige Medien und/oder flüchtige Medien beinhalten. Nicht-flüchtige Medien schließen beispielsweise optische oder magnetische Platten ein, beispielsweise die Speichervorrichtung **410**. Flüchtige Medien schließen einen dynamischen Speicher wie beispielsweise den Hauptspeicher **406** ein. Übliche Formen von Speichermedien beinhalten beispielsweise eine Diskette, eine flexible Platte, eine Festplatte, ein Halbleiterlaufwerk, ein Magnetband oder ein beliebiges anderes magnetisches Datenspeichermedium, eine CD-ROM, ein beliebiges anderes optisches Datenspeichermedium, ein beliebiges physisches Medium mit Lochmustern, ein RAM, ein PROM und ein EPROM, ein Flash-EPROM, ein NVRAM, und einen beliebigen anderen Speicherchip oder -kassette.

[0060] Speichermedien sind verschieden von Übertragungsmedien, können jedoch in Verbindung mit diesen verwendet werden. Übertragungsmedien nehmen an einer Übertragung von Information zwischen Speichermedien teil. Beispielsweise beinhalten Übertragungsmedien Koaxialkabel, Kupferdraht und Lichtwellenleiter, einschließlich der Drähte, die der Bus **402** beinhaltet. Übertragungsmedien können auch die Form von Schall- oder Lichtwellen annehmen, beispielsweise solche, die bei Funk- und Infrarot-Datenkommunikationen erzeugt werden.

[0061] Verschiedene Formen von Medien können bei der Ausführung einer oder mehrerer Sequenzen von einer oder mehreren Anweisungen beteiligt sein, die durch den Prozessor **404** auszuführen sind. Beispielsweise können die Anweisungen anfänglich auf einer Magnetplatte oder einem Halbleiterlaufwerk (Solid-State-Drive) eines entfernt befindlichen Computers gespeichert sein. Der ferne Computer kann die Anweisungen in seinen dynamischen Speicher laden und die Anweisungen über eine Telefonleitung unter Verwendung eines Modem senden. Ein beim Computersystem **400** befindliches Modem kann die Daten über die Telefonleitung empfangen und einen Infrarot-Sender verwenden, um die Daten in ein Infrarotsignal umzuwandeln. Ein Infrarot-Detektor kann

die im Infrarot-Signal beförderten Daten empfangen, und geeignete Schaltungen können die Daten auf den Bus **402** legen. Der Bus **402** befördert die Daten zum Hauptspeicher **406**, aus dem der Prozessor **404** die Anweisungen abruft und ausführt. Die vom Hauptspeicher **406** empfangenen Anweisungen können optional in der Speichervorrichtung **410** gespeichert werden, entweder vor oder nach einem Ausführen durch den Prozessor **404**.

[0062] Das Computersystem **400** beinhaltet auch eine Kommunikationsschnittstelle **418**, die mit dem Bus **402** verbunden ist. Die Kommunikationsschnittstelle **418** stellt eine Zweiweg-Datenkommunikationsverbindung zu einem Netzwerk-Verbindungsglied **420** bereit, das mit einem lokalen Netzwerk **422** verbunden ist. Beispielsweise kann eine Kommunikationsschnittstelle **418** eine ISDN-Karte (ISDN = Integrated Services Digital Network), ein Kabelmodem, ein Satellitenmodem oder ein Modem sein, um eine Datenkommunikationsverbindung zu einem entsprechenden Typ von Telefonleitung bereitzustellen. Als weiteres Beispiel kann eine Kommunikationsschnittstelle **418** eine LAN-Karte (LAN = Local Area Network) sein, um eine Datenkommunikationsverbindung zu einem kompatiblen LAN bereitzustellen. Es können auch Drahtlosverbindungen implementiert werden. Bei einer beliebigen derartigen Implementierung sendet und empfängt eine Kommunikationsschnittstelle **418** elektrische, elektromagnetische oder optische Signale, die digitale Datenströme tragen, welche verschiedene Typen von Information repräsentieren.

[0063] Das Netzwerk-Verbindungsglied **420** stellt typischerweise eine Datenkommunikation zu anderen Datengeräten über eines oder mehrere Netzwerke bereit. Beispielsweise kann das Netzwerk-Verbindungsglied **420** über ein lokales Netzwerk **422** eine Verbindung zu einem Host-Computer **424** oder zu Datenanlagen bereitstellen, die durch einen Internetdienstanbieter (ISP) **426** betrieben werden. Der ISP **426** stellt seinerseits Datenkommunikationsdienste über das weltweite Paketdaten-Kommunikationsnetz bereit, das jetzt üblicherweise als „Internet" **428** bezeichnet wird. Sowohl das lokale Netz **422** als auch das Internet **428** verwenden elektrische, elektromagnetische oder optische Signale, die digitale Datenströme tragen können. Die über die verschiedenen Netzwerke laufenden Signale und die Signale, die über das Netzwerk-Verbindungsglied **420** und über die Kommunikationsschnittstelle **418** laufen, welche die digitalen Daten zum Computersystem **400** hin und von diesem weg befördern, sind beispielhafte Formen von Übertragungsmedien.

[0064] Das Computersystem **400** kann Nachrichten und Daten, einschließlich Programmcode, senden und empfangen, und zwar über das/die Netzwerk(e), das Netzwerk-Verbindungsglied **420** und die Kommunikationsschnittstelle **418**. In dem Beispiel des Internet könnte ein Server **430** einen angeforderten Code für ein Anwendungsprogramm über das Internet **428**, den ISP **426**, das lokale Netzwerk **422** und die Kommunikationsschnittstelle **418** senden. Der empfangene Code kann durch den Prozessor **404** ausgeführt werden, und zwar unverändert wie empfangen, und/oder er kann in der Speichervorrichtung **410** oder einem anderen nicht-flüchtigen Speicher für eine spätere Ausführung gespeichert werden.

Patentansprüche

1. Verfahren zum Assoziieren eines ersten Objekts mit einem oder mehreren Objekten innerhalb einer Mehrzahl von Objekten, wobei jedes Objekt eine erste Mehrzahl von Eigenschaften aufweist, jede Eigenschaft Daten aufweist, die ein Kennzeichen einer durch das Objekt repräsentierten Entität widerspiegeln, wobei die assoziierten Objekte übereinstimmende Daten in entsprechenden Eigenschaften für eine zweite Mehrzahl von Eigenschaften aufweisen, wobei das Verfahren die folgenden Operationen umfasst, die durch einen oder mehrere Prozessoren ausgeführt werden:
Ausführen, für jedes Objekt innerhalb der Mehrzahl von Objekten und für das erste Objekt, des Folgenden:
Erzeugen eines Slug für das Objekt, wobei der Slug die zweite Mehrzahl von Eigenschaften von dem Objekt aufweist; und
Eingeben des Slug für das Objekt in einen Bloomfilter; und
Erzeugen, für eine Position im Bloomfilter, die dem Slug für das erste Objekt entspricht, einer Assoziierung zwischen Objekten, deren Slugs der Position entsprechen, falls die Slugs für diese Objekte übereinstimmen.

2. Verfahren nach Anspruch 1, weiter umfassend:
Festlegen der Größe des Bloomfilters, für eine vorbestimmte Fehlerrate und Anzahl von Objekten innerhalb der Mehrzahl von Objekten.

3. Verfahren nach Anspruch 1 oder Anspruch 2, weiter umfassend:
Auslesen der Mehrzahl von Objekten aus mindestens einer Datenbank.

4. Verfahren nach einem der Ansprüche 1 bis 3, das weiter umfasst, dass das Bestimmen, ob die Slugs in der Position, die dem Slug für das erste Objekt entspricht, mit dem Slug für das erste Objekt übereinstimmen, die folgenden Operationen umfasst, die durch einen oder mehrere Prozessoren ausgeführt werden:
Eingeben, für jeden Slug in der Position, die dem Slug für das erste Objekt entspricht, des jeweiligen Slug und dessen zugehörigen Objekts in eine Multimap, wobei der Slug für das jeweilige Objekt ein Schlüssel

zu der Multimap ist und sein zugehöriges Objekt ein Wert zu der Multimap ist; und
Assoziieren von Objekten in der Multimap, deren Schlüssel mit dem Slug für das erste Objekt übereinstimmt.

5. Verfahren zum Assoziieren von Objekten innerhalb einer oder mehreren Gruppen von Objekten innerhalb einer Mehrzahl von Objekten, wobei jedes Objekt eine erste Mehrzahl von Eigenschaften aufweist, jede Eigenschaft Daten aufweist, die ein Kennzeichen einer durch das Objekt repräsentierten Entität widerspiegeln, wobei die assoziierten Objekte innerhalb einer Gruppe von Objekten übereinstimmende Daten in entsprechenden Eigenschaften für eine zweite Mehrzahl von Eigenschaften aufweisen, wobei das Verfahren die folgenden Operationen umfasst, die durch einen oder mehrere Prozessoren ausgeführt werden:
Ausführen, für jedes Objekt innerhalb der Mehrzahl von Objekten, des Folgenden:
Erzeugen eines Slug für das Objekt, wobei der Slug die zweite Mehrzahl von Eigenschaften von dem Objekt aufweist; und
Eingeben des Slug für das Objekt in einen zählenden Bloomfilter;
Eingeben, für jeden erzeugten Slug, des Slug und dessen zugehörigen Objekts in eine Multimap, falls eine Position im zählenden Bloomfilter, die dem Slug entspricht, einen Zählwert größer als 1 aufweist, wobei der Slug ein Schlüssel zu der Multimap ist und das Objekt ein Wert zu der Multimap ist; und
Assoziieren der Objekte, die als Werte für jeden Schlüssel der Multimap gespeichert sind, mit zwei oder mehr zugehörigen Werten.

6. Verfahren nach Anspruch 5, weiter umfassend:
Festlegen der Größe des zählenden Bloomfilters, für eine vorbestimmte Fehlerrate und Anzahl von Objekten innerhalb der Mehrzahl von Objekten.

7. Verfahren nach Anspruch 5 oder Anspruch 6, weiter umfassend:
Auslesen der Mehrzahl von Objekten aus mindestens einer Datenbank.

8. Verfahren nach einem der Ansprüche 5 bis 7, wobei jeder Eintrag im zählenden Bloomfilter einen 2-Bit-Zähler aufweist.

9. Verfahren nach Anspruch 8, wobei jeder 2-Bit-Zähler ein Sättigungszähler ist.

10. Verfahren nach einem der Ansprüche 1 bis 9, wobei der Slug eine Verkettung von zwei oder mehr Zeichenketten aufweist, die durch einen Abgrenzer zwischen jeder verketteten Zeichenkette getrennt sind.

11. Verfahren nach Anspruch 10, wobei der Abgrenzer ein Zeichen aufweist, das ansonsten nicht in den Zeichenketten vorhanden ist, die verkettet wurden.

12. Verfahren nach Anspruch 10, wobei der Abgrenzer eine Abfolge von zwei oder mehr Zeichen aufweist und die Abfolge von zwei oder mehr Zeichen nicht in irgendeiner der zwei oder mehr Zeichenketten vorhanden ist, die verkettet wurden.

13. Verfahren nach einem der Ansprüche 10 bis 12, wobei die Anzahl von Eigenschaften in der ersten Mehrzahl von Eigenschaften gleich groß wie die Anzahl von Eigenschaften in der zweiten Mehrzahl von Eigenschaften ist.

14. Verfahren nach einem der Ansprüche 10 bis 12, wobei die Anzahl von Eigenschaften in der ersten Mehrzahl von Eigenschaften größer als die Anzahl von Eigenschaften in der zweiten Mehrzahl von Eigenschaften ist.

15. System, aufweisend:
eine Speichervorrichtung, die einen Satz von Anweisungen speichert; und mindestens einen Prozessor, der den Satz von Anweisungen ausführt, um Operationen durchzuführen, welche die Operationen nach einem der Ansprüche 1 bis 14 beinhalten.

16. Computerlesbares Medium, das Anweisungen speichert, die, wenn sie durch mindestens einen Prozessor ausgeführt werden, den mindestens einen Prozessor veranlassen, Operationen durchzuführen, welche die Operationen nach einem der Ansprüche 1 bis 14 beinhalten.

Es folgen 4 Seiten Zeichnungen

Anhängende Zeichnungen

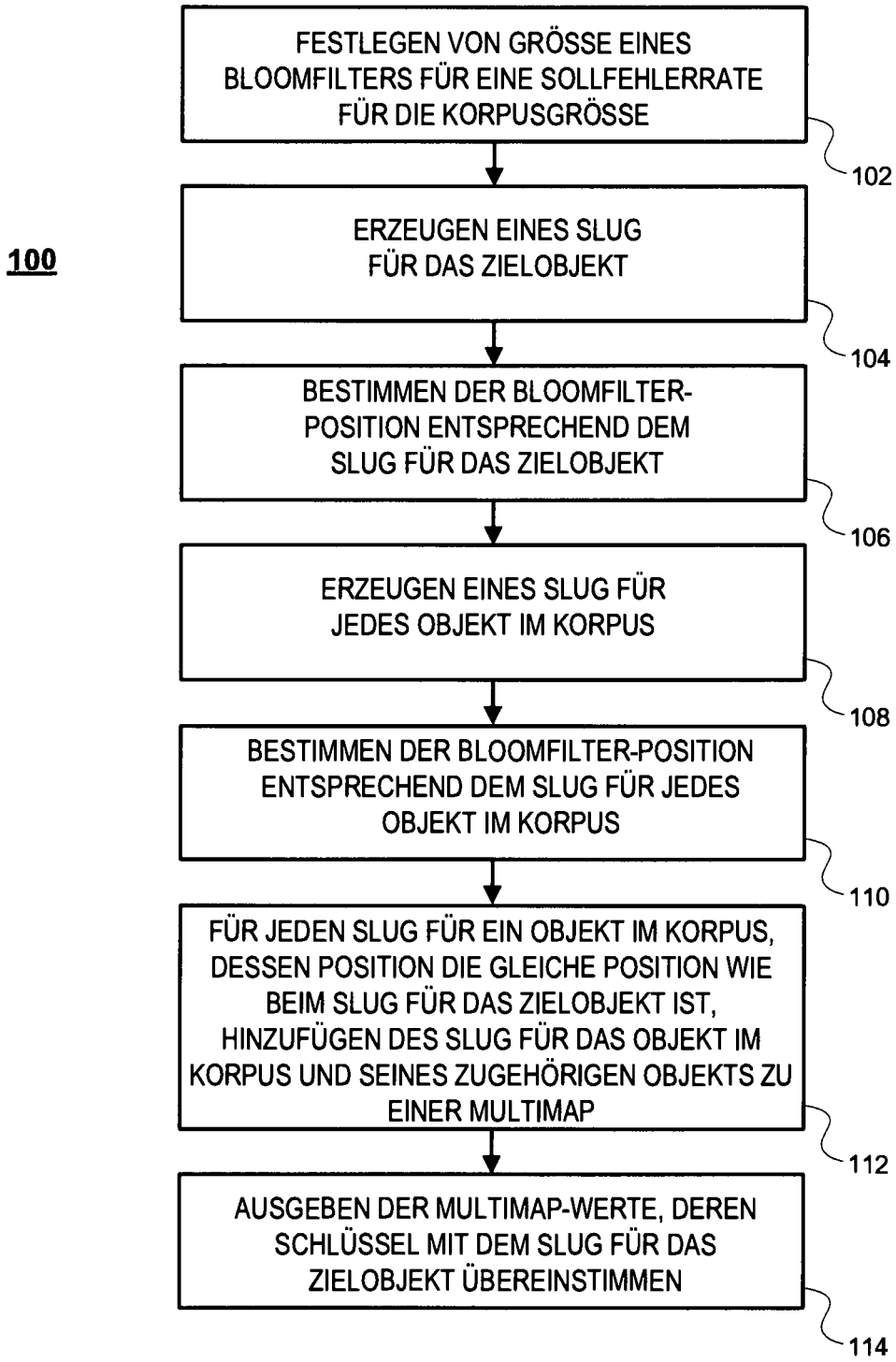


FIG. 1

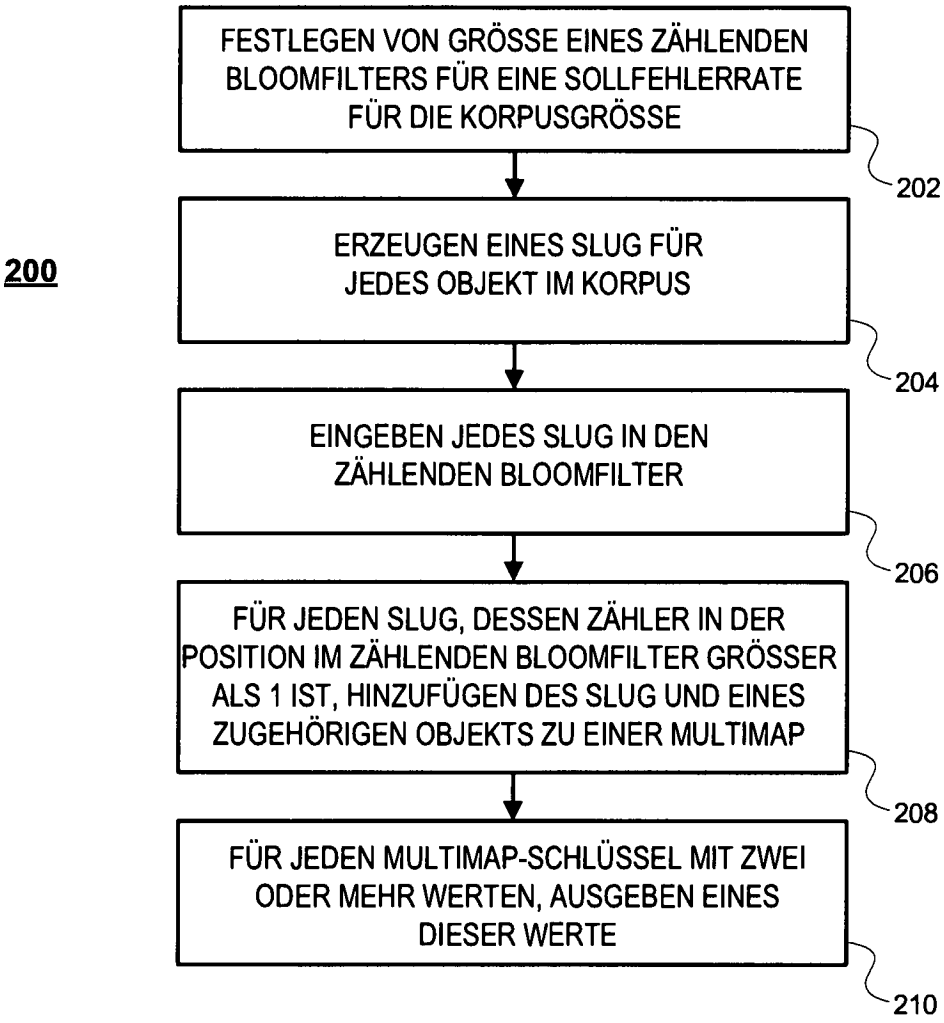

200
FESTLEGEN VON GRÖSSE EINES ZÄHLENDEN BLOOMFILTERS FÜR EINE SOLLFEHLERRATE FÜR DIE KORPUSGRÖSSE
202
ERZEUGEN EINES SLUG FÜR JEDES OBJEKT IM KORPUS
204
EINGEBEN JEDES SLUG IN DEN ZÄHLENDEN BLOOMFILTER
206
FÜR JEDEN SLUG, DESSEN ZÄHLER IN DER POSITION IM ZÄHLENDEN BLOOMFILTER GRÖSSER ALS 1 IST, HINZUFÜGEN DES SLUG UND EINES ZUGEHÖRIGEN OBJEKTS ZU EINER MULTIMAP
208
FÜR JEDEN MULTIMAP-SCHLÜSSEL MIT ZWEI ODER MEHR WERTEN, AUSGEBEN EINES DIESER WERTE
210

FIG. 2

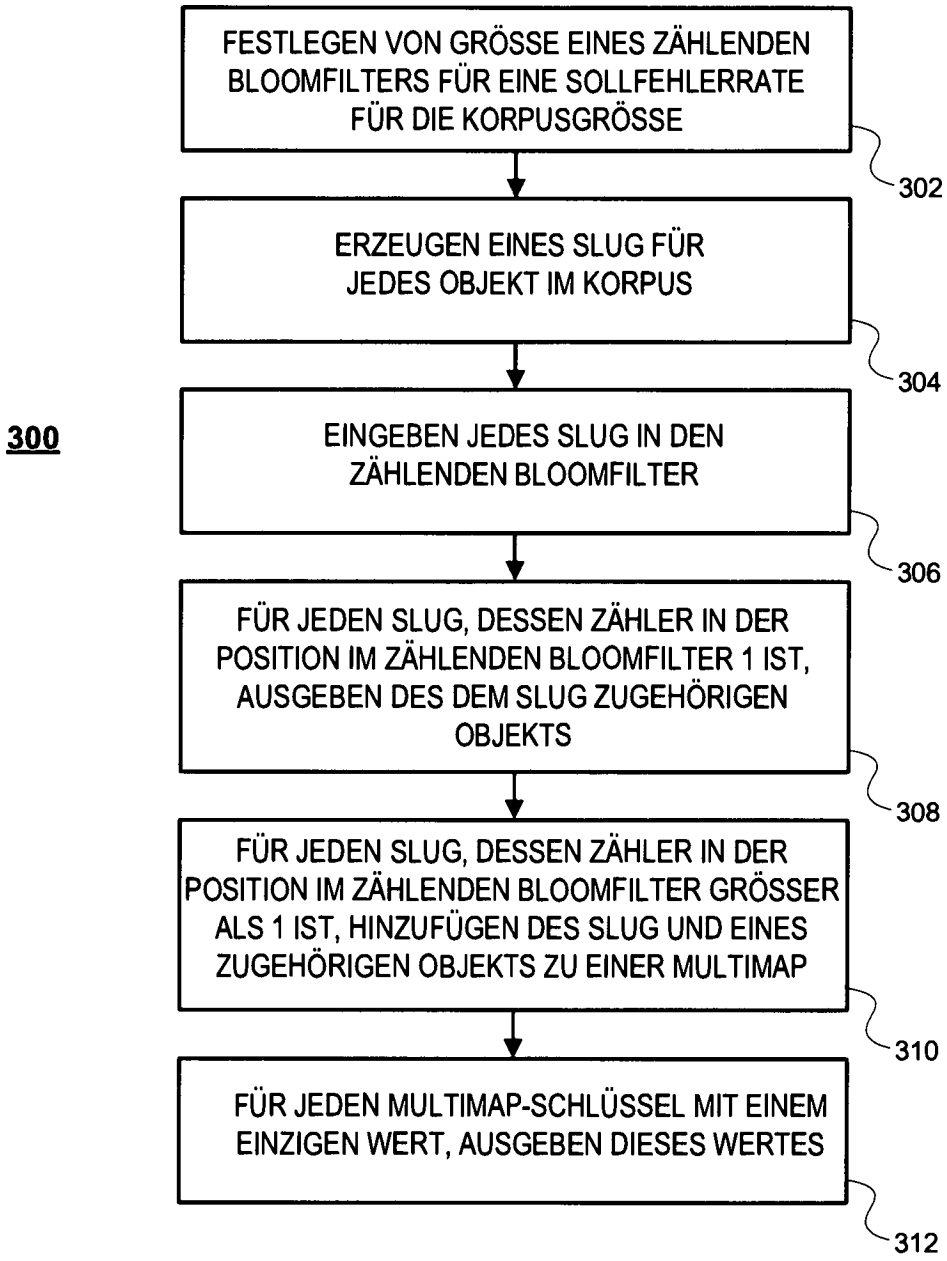

300
FESTLEGEN VON GRÖSSE EINES ZÄHLENDEN BLOOMFILTERS FÜR EINE SOLLFEHLERRATE FÜR DIE KORPUSGRÖSSE
302
ERZEUGEN EINES SLUG FÜR JEDES OBJEKT IM KORPUS
304
EINGEBEN JEDES SLUG IN DEN ZÄHLENDEN BLOOMFILTER
306
FÜR JEDEN SLUG, DESSEN ZÄHLER IN DER POSITION IM ZÄHLENDEN BLOOMFILTER 1 IST, AUSGEBEN DES DEM SLUG ZUGEHÖRIGEN OBJEKTS
308
FÜR JEDEN SLUG, DESSEN ZÄHLER IN DER POSITION IM ZÄHLENDEN BLOOMFILTER GRÖSSER ALS 1 IST, HINZUFÜGEN DES SLUG UND EINES ZUGEHÖRIGEN OBJEKTS ZU EINER MULTIMAP
310
FÜR JEDEN MULTIMAP-SCHLÜSSEL MIT EINEM EINZIGEN WERT, AUSGEBEN DIESES WERTES
312

FIG. 3

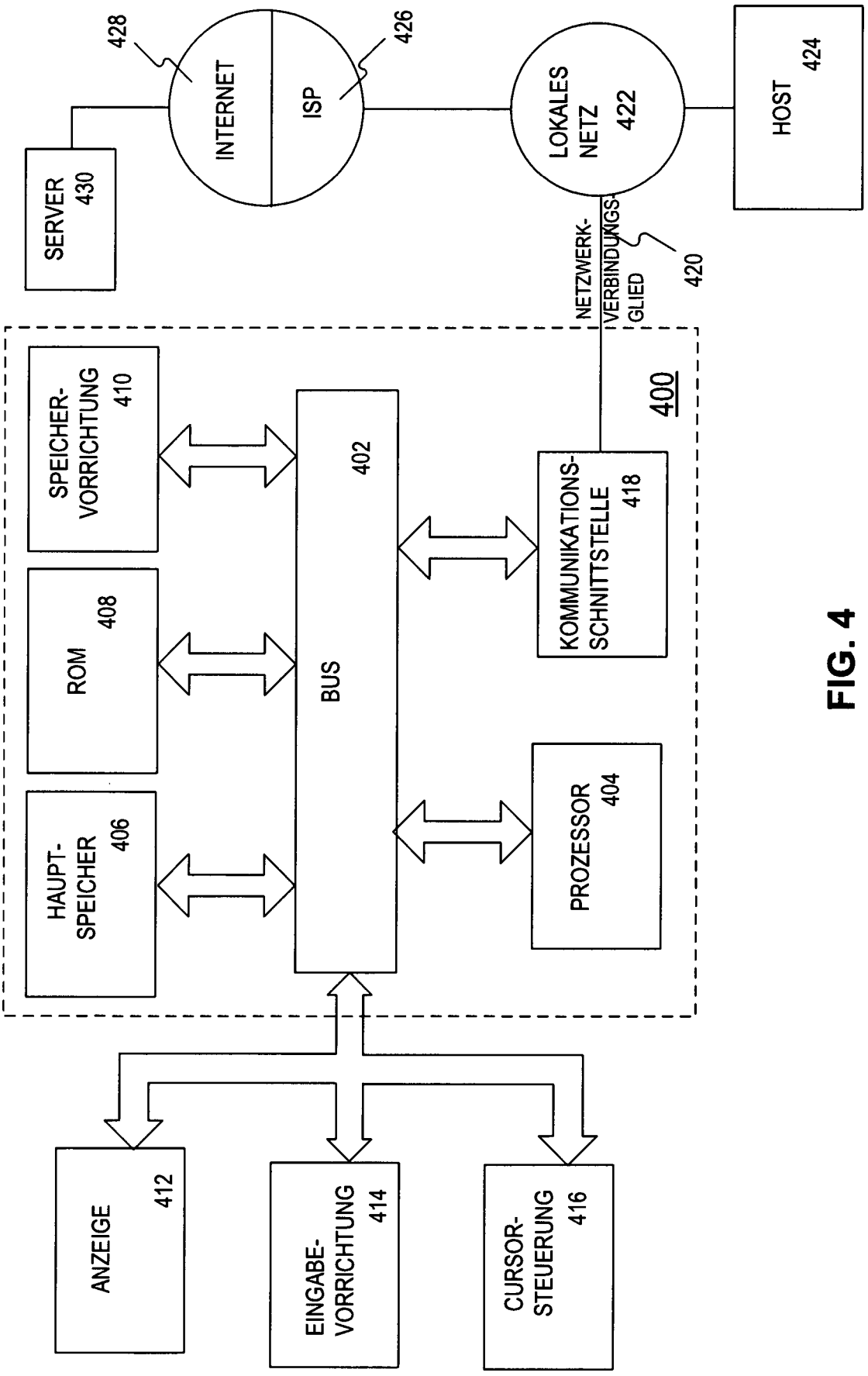

428
INTERNET
ISP
426
LOKALES NETZ 422
HOST 424
SERVER 430
NETZWERK-VERBINDUNGS-GLIED
420
400
SPEICHER-VORRICHTUNG 410
402
KOMMUNIKATIONS-SCHNITTSTELLE 418
ROM 408
BUS
HAUPT-SPEICHER 406
PROZESSOR 404
ANZEIGE 412
EINGABE-VORRICHTUNG 414
CURSOR-STEUERUNG 416

FIG. 4