



(10) **DE 11 2020 004 815 T5** 2022.06.15

(12) **Veröffentlichung**

der internationalen Anmeldung mit der
(87) Veröffentlichungs-Nr.: **WO 2021/070013**
in der deutschen Übersetzung (Art. III § 8 Abs. 2
IntPatÜbkG)
(21) Deutsches Aktenzeichen: **11 2020 004 815.2**
(86) PCT-Aktenzeichen: **PCT/IB2020/059141**
(86) PCT-Anmeldetag: **30.09.2020**
(87) PCT-Veröffentlichungstag: **15.04.2021**
(43) Veröffentlichungstag der PCT Anmeldung
in deutscher Übersetzung: **15.06.2022**

(51) Int Cl.: **G06F 16/835** (2019.01)
G06F 9/50 (2006.01)
G06N 5/04 (2006.01)

(30) Unionspriorität:
16/594,391 **07.10.2019** **US**

(71) Anmelder:
**INTERNATIONAL BUSINESS MACHINES
CORPORATION, Armonk, NY, US**

(74) Vertreter:
**LifeTech IP Spies & Behrndt Patentanwälte PartG
mbB, 80687 München, DE**

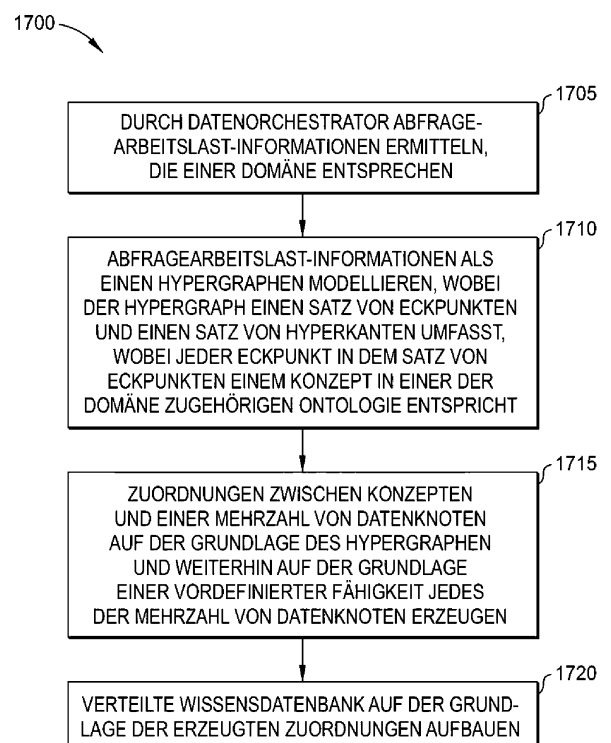
(72) Erfinder:
**Quamar, Abdul, San Jose, CA, US; Ozcan, Fatma,
San Jose, CA, US; Alotaibi, Rana, San Jose, CA,
US; Lei, Chuan, San Jose, CA, US**

Prüfungsantrag gemäß § 44 PatG ist gestellt.

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen.

(54) Bezeichnung: **ONTOLOGIEGESTÜTZTER DATENSPEICHER FÜR VERTEILTE WISSENSDATENBANKEN**

(57) Zusammenfassung: Techniken für eine verteilte Datenplatzierung werden bereitgestellt. Abfragearbeitslast-Informationen, die einer Domäne entsprechen, werden von einem Datenorchestrator ermittelt, und die Abfragearbeitslast-Informationen werden als ein Hypergraph modelliert, wobei der Hypergraph einen Satz von Eckpunkten und einen Satz von Hyperkanten umfasst, wobei jeder Eckpunkt in dem Satz von Eckpunkten einem Konzept in einer der Domäne zugehörigen Ontologie entspricht. Zuordnungen zwischen Konzepten und einer Mehrzahl von Datenknoten auf der Grundlage des Hypergraphen und weiterhin auf der Grundlage vordefinierter Fähigkeiten jedes der Mehrzahl von Datenknoten werden erzeugt. Eine verteilte Wissensdatenbank wird auf der Grundlage der erzeugten Zuordnungen eingerichtet.



Beschreibung

TECHNISCHES GEBIET

[0001] Die vorliegende Offenbarung bezieht sich auf Wissensdatenbanken und insbesondere auf Verwenden von Ontologien für eine leistungsfähige Datenplatzierung in verteilten Wissensdatenbanken.

HINTERGRUND

[0002] Immer mehr Unternehmen nutzen Wissensdatenbanken (Knowledge Bases, KBs), um ihre Analysen zu erweitern und die Entscheidungsfindung sowie die Effizienz und Wirksamkeit ihrer Systeme zu verbessern. Oft sind die KBs im Bereich des Unternehmens relativ spezialisiert. Finanzinstitute beispielsweise stützen sich auf KBs mit umfangreichem Finanzwissen, z.B. Daten im Zusammenhang mit staatlichen Vorschriften für Finanzmärkte. Umgekehrt können Unternehmen des Gesundheitswesens KBs mit einer beträchtlichen Menge an Daten unterhalten, die aus der medizinischen Literatur stammen. Es besteht ein erheblicher Bedarf an einer umfassenden Domänenspezialisierung, an leistungsfähigen Systemen und Techniken zum Verwalten dieser KBs. Bestehende Systeme, die die Domänenontologie, die eine entitätszentrierte Sicht auf das Domänenschema bereitstellt, nicht kennen, sind nicht in der Lage, Abfragen der KB wirksam zu verwalten und weiterzuleiten.

[0003] Darüber hinaus können KBs auf eine Reihe von Datenstandorten mit unterschiedlichen Kapazitäten und Kosten verteilt werden, um die Operationen zu verbessern. Bestehende Architekturen, z.B. föderierte Datenbanken, stützen sich auf einen zentralen Mediator, der Daten aus jeder dieser Quellen zusammenfasst. Dies ist ineffizient und schlecht skalierbar. Außerdem verstehen bestehende Systeme die zugrunde liegende Ontologie und die Fähigkeiten der einzelnen Datenquellen nicht und können Abfragen daher nicht effizient weiterleiten. Dies mindert die Effizienz solcher Systeme und erfordert erhebliche Datenverarbeitungsressourcen zum Beantworten typischer Abfragen.

KURZDARSTELLUNG

[0004] Gemäß einem Aspekt der vorliegenden Erfindung wird ein Verfahren bereitgestellt, das aufweist: Ermitteln von Abfragearbeitslast-Informationen, die einer Domäne entsprechen, durch einen Datenorchestrator. Das Verfahren umfasst weiterhin Modellieren der Abfragearbeitslast-Informationen als einen Hypergraphen, wobei der Hypergraph einen Satz von Eckpunkten und einen Satz von Hyperkanten umfasst, wobei jeder Eckpunkt in dem Satz von Eckpunkten einem Konzept in einer der Domäne zugehörigen Ontologie entspricht. Darüber hinaus

umfasst das Verfahren Erzeugen von Zuordnungen zwischen Konzepten und einer Mehrzahl von Datenknoten auf der Grundlage des Hypergraphen und weiterhin auf der Grundlage vordefinierter Fähigkeiten jedes der Mehrzahl von Datenknoten und Einrichtungen einer verteilten Wissensdatenbank auf der Grundlage der erzeugten Zuordnungen. Das Verfahren ermöglicht es dem Datenorchestrator auf vorteilhafte Weise, Daten auf der Grundlage einer vorhandenen Arbeitslast und der Fähigkeiten der einzelnen Datenknoten effizient in einer verteilten Umgebung zu platzieren und zu speichern. Dies verringert den für ein Speichern der Daten erforderlichen Rechenaufwand und verbessert weiterhin die Reaktionsfähigkeit des Systems, indem es eine leistungsfähige Zuordnung der Daten bereitstellt.

[0005] Gemäß einer anderen Ausführungsform der vorliegenden Offenbarung umfasst Ermitteln der Abfragearbeitslast Empfangen eines Satzes von früheren ontologischen Abfragen. Ein Verfahren gemäß dieser Ausführungsform umfasst Erzeugen eines ersten Satzes von Konzepten, auf die durch eine erste Abfrage in dem Satz von früheren ontologischen Abfragen zugegriffen wird, und Erzeugen eines zweiten Satzes von Operationen, die durch die erste Abfrage durchgeführt werden. Eine erste Gesamtabfrage wird erzeugt, indem eine Gruppe von Abfragen aus dem Satz früherer ontologischer Abfragen mit entsprechenden übereinstimmenden ersten Sätzen identifiziert wird und auf der Grundlage entsprechender zweiter Sätze für jede Abfrage in der identifizierten Gruppe von Abfragen ein zusammengefasster Satz von Operationen ermittelt wird. Die erste Gesamtabfrage wird dann dem zusammengefassten Satz von Operationen und Konzepten zugeordnet, der in den entsprechenden übereinstimmenden ersten Sätzen widergespiegelt wird. In einer solchen Ausführungsform bietet der Datenorchestrator eine Verbesserung gegenüber bestehenden Systemen, indem er frühere Gesamtabfragen wirksam zusammenfasst, um einen guten Speicherplan für die Daten zu ermitteln, der dem erwarteten Bedarf entspricht. Auch dies verbessert die Effizienz und verringert unnötige Rechenzeit während der Laufzeit.

[0006] Gemäß einer weiteren Ausführungsform der vorliegenden Offenbarung umfasst Modellieren der Abfragearbeitslast-Informationen als einen Hypergraphen Erzeugen eines Eckpunkts für jedes Konzept in der Ontologie und Erzeugen einer ersten Hyperkante für die erste Gesamtabfrage, wobei die erste Hyperkante einen ersten Satz von Eckpunkten in dem Hypergraphen verbindet, wobei der erste Satz von Eckpunkten den Konzepten entspricht, die in den übereinstimmenden ersten Sätzen widergespiegelt werden. In einer solchen Ausführungsform umfasst das Verfahren weiterhin Kennzeichnen der ersten Hyperkante mit dem zusammengefassten

Satz von Operationen. Eine solche Ausführungsform ermöglicht es dem Datenorchestrator auf vorteilhafte Weise, Arbeitslasten in Graphform wirksam darzustellen, wodurch der Orchestrator die Daten besser und effizienter auswerten kann, um bessere Platzierungsentscheidungen zu treffen. Dadurch wird die Laufzeitleistung erheblich verbessert.

[0007] Gemäß einer weiteren Ausführungsform der vorliegenden Offenbarung weist das Erzeugen der Zuordnungen Erzeugen eines ersten Clusters für eine erste Operation auf, die in dem Hypergraphen enthalten ist. Die Ausführungsform umfasst anschließend Identifizieren eines ersten Satzes von Konzepten, die durch eine erste Hyperkante in dem Hypergraphen verbunden sind, und Identifizieren eines ersten Satzes von Operationen, die durch die erste Hyperkante angegeben werden. Nachdem festgestellt wurde, dass der erste Satz von Operationen die erste Operation umfasst, umfasst das Verfahren Zuweisen des ersten Satzes von Konzepten zu dem ersten Cluster. Ein Vorteil einer solchen Ausführungsform besteht darin, dass sie ein effizientes Auswerten des Hypergraphen ermöglicht und zu einer hochzuverlässigen Datenplatzierung führt, die zur Laufzeit minimale Versetzungen erfordert.

[0008] Gemäß einer anderen Ausführungsform der vorliegenden Offenbarung umfasst Erzeugen der Zuordnungen weiterhin Zuordnen des ersten Satzes von Konzepten zu einem oder mehreren Datenknoten durch Identifizieren eines Satzes von Datenknoten, die die erste Operation durchführen können, und Zuordnen jedes Konzepts im ersten Satz von Konzepten zu jedem Datenknoten in dem identifizierten Satz von Datenknoten. Dies hat den Vorteil, dass das System Datenzuordnungen auf der Grundlage von Fähigkeiten der Knoten und unter Berücksichtigung früherer Arbeitslasten erzeugen kann. Dadurch erhöht sich die Wahrscheinlichkeit, dass das System in der Lage ist, auf künftige Anfragen mit möglichst kurzer Latenzzeit und möglichst geringer Inanspruchnahme von Ressourcen zu antworten.

[0009] Gemäß einer weiteren Ausführungsform der vorliegenden Offenbarung umfasst Erzeugen der Zuordnungen Identifizieren eines ersten Satzes von Konzepten, die durch eine erste Hyperkante in dem Hypergraphen verbunden sind, Identifizieren eines ersten Satzes von Operationen, die durch die erste Hyperkante angegeben werden, und Ermitteln eines kleinstmöglichen Satzes von Datenknoten, die in der Lage sind, den ersten Satz von Operationen gemeinsam durchzuführen. Das Verfahren umfasst dann Erzeugen eines Clusters, der den ersten Satz von Konzepten umfasst, und Kennzeichnen des Clusters mit dem kleinstmöglichen Satz von Datenknoten. Eine solche Ausführungsform bietet Verbesserungen gegenüber bestehenden Lösungen, indem sie die Replikation von Daten im System auf ein Mindest-

maß beschränkt, was die Speicherkosten senkt und zusätzlich die Latenzzeit und die Inanspruchnahme von Ressourcen verringert, die durch das Versetzen von Daten im System entstehen.

[0010] Gemäß einer weiteren Ausführungsform der vorliegenden Offenbarung weist das Erzeugen der Zuordnungen weiterhin Zuordnen jedes Konzepts im ersten Satz von Konzepten zu jedem Datenknoten im kleinstmöglichen Satz von Datenknoten auf. Dies verringert ebenfalls die Speicherkosten und die Latenzzeit des Systems.

[0011] Gemäß einer weiteren Ausführungsform der vorliegenden Offenbarung umfasst das Einrichten der verteilten Wissensdatenbank für jedes entsprechende Konzept in der Ontologie Identifizieren eines entsprechenden Datenknotens, der durch die Zuordnungen angegeben wird, Identifizieren von Daten, die dem jeweiligen Konzept entsprechen, und Erleichtern des Speicherns der identifizierten Daten in dem entsprechenden Datenknoten. Eine solche Ausführungsform hat den Vorteil, dass der Orchestrator Zuordnungen wirksam erzeugen und Daten effizient in den entsprechenden Knoten platzieren kann, was den Rechenaufwand sowohl vor als auch während der Laufzeit verringert.

[0012] Gemäß einer anderen Ausführungsform der vorliegenden Offenbarung kann jede der oben beschriebenen Ausführungsformen durch ein durch einen Computer lesbares Speichermedium implementiert werden. Das durch einen Computer lesbare Speichermedium enthält einen Computerprogrammcode, der, wenn er durch einen oder mehrere Computerprozessoren ausgeführt wird, eine Operation durchführt. In Ausführungsformen kann die durchgeführte Operation einer beliebigen Kombination der obigen Verfahren und Ausführungsformen entsprechen.

[0013] Gemäß einer anderen Ausführungsform der vorliegenden Offenbarung kann jede der oben beschriebenen Ausführungsformen durch ein System implementiert werden. Das System umfasst einen oder mehrere Computerprozessoren und einen Arbeitsspeicher, der ein Programm enthält, das, wenn es von dem einen oder mehreren Computerprozessoren ausgeführt wird, eine Operation durchführt. In Ausführungsformen kann die durchgeführte Operation einer beliebigen Kombination der obigen Verfahren und Ausführungsformen entsprechen.

Figurenliste

Fig. 1 veranschaulicht eine Architektur, die so konfiguriert ist, dass sie gemäß einer hier offenbarten Ausführungsform Daten für eine

Wissensdatenbank platziert und ontologische Abfragen weiterleitet.

Fig. 2 veranschaulicht einen Arbeitsablauf zum Verarbeiten und Weiterleiten von ontologischen Abfragen gemäß einer hier offenbaren Ausführungsform.

Die **Fig. 3A** und **Fig. 3B** zeigen eine beispielhafte Ontologie, die gemäß einer hier offenbaren Ausführungsform zum Auswerten und Weiterleiten von Abfragen verwendet werden kann.

Die **Fig. 4A** und **Fig. 4B** veranschaulichen gemäß einer hier offenbaren Ausführungsform einen Arbeitsablauf zum Parsen und Weiterleiten einer beispielhaften ontologischen Abfrage.

Fig. 5 ist ein Blockschaubild, das gemäß einer hier offenbaren Ausführungsform einen Abfrageverarbeitungsorchestrator veranschaulicht, der so konfiguriert ist, dass er ontologische Abfragen weiterleitet.

Fig. 6 ist ein Ablaufplan, der gemäß einer hier offenbaren Ausführungsform ein Verfahren zum Verarbeiten und Weiterleiten von ontologischen Abfragen veranschaulicht.

Fig. 7 ist ein Ablaufplan, der gemäß einer hier offenbaren Ausführungsform ein Verfahren zum Verarbeiten von ontologischen Abfragen veranschaulicht, um Abfrageblöcke effizient weiterzuleiten.

Fig. 8 ist ein Ablaufplan, der gemäß einer hier offenbaren Ausführungsform ein Verfahren zum Auswerten potenzieller Abfrageweiterleitungspläne veranschaulicht, um eine ontologische Abfrage effizient weiterzuleiten.

Fig. 9 ist ein Ablaufplan, der gemäß einer hier offenbaren Ausführungsform ein Verfahren zum Weiterleiten einer ontologischen Abfrage veranschaulicht.

Fig. 10 veranschaulicht einen Arbeitsablauf zum Auswerten von Arbeitslasten und Speichern von Daten einer Wissensdatenbank gemäß einer hier offenbaren Ausführungsform.

Fig. 11 zeigt einen beispielhaften Hypergraphen, der gemäß einer hier offenbaren Ausführungsform zum Auswerten von Arbeitslasten und Speichern von Daten einer Wissensdatenbank verwendet wird.

Fig. 12 ist ein Blockschaubild, das gemäß einer hier offenbaren Ausführungsform einen Datenplatzierungsorchestrator veranschaulicht, der so konfiguriert ist, dass er Arbeitslasten auswertet und Daten speichert.

Fig. 13 ist ein Ablaufplan, der gemäß einer hier offenbaren Ausführungsform ein Verfahren zum Auswerten und Zusammenfassen der

Arbeitslast von Abfragen veranschaulicht, um Entscheidungen über die Datenplatzierung zu unterstützen.

Fig. 14 ist ein Ablaufplan, der gemäß einer hier offenbaren Ausführungsform ein Verfahren zum Modellieren ontologischer Arbeitslasten veranschaulicht, um Entscheidungen über Datenplatzierungen zu unterstützen.

Fig. 15 ist ein Ablaufplan, der gemäß einer hier offenbaren Ausführungsform ein Verfahren zum Auswerten eines Hypergraphen veranschaulicht, um Entscheidungen über Datenplatzierungen zu treffen.

Fig. 16 ist ein Ablaufplan, der gemäß einer hier offenbaren Ausführungsform ein Verfahren zum Auswerten eines Hypergraphen veranschaulicht, um Entscheidungen über Datenplatzierungen zu treffen.

Fig. 17 ist ein Ablaufplan, der gemäß einer hier offenbaren Ausführungsform ein Verfahren zum Zuordnen von ontologischen Konzepten zu Speickerknoten veranschaulicht.

AUSFÜHRLICHE BESCHREIBUNG

[0014] Ausführungsformen der vorliegenden Offenbarung stellen eine ontologiesteuerte Architektur bereit, die Unterstützung für eine Vielfalt von Abfragetypen unter Verwendung mehrerer Datenspeicher oder -knoten bereitstellt, von denen jeder unterschiedliche Fähigkeiten aufweisen kann. Ausführungsformen der vorliegenden Offenbarung ermöglichen auf vorteilhafte Weise effizientes Verarbeiten und Weiterleiten von Abfragen an die verschiedenen Datenspeicher auf der Grundlage einer bestehenden Ontologie, wodurch die Latenzzeit des Systems verbessert und die für Identifizieren und Zurücksenden relevanter Daten erforderlichen Datenverarbeitungsressourcen verringert werden. In Ausführungsformen können die hier beschriebenen Techniken verwendet werden, um eine Vielfalt von Abfrageanwendungen zu unterstützen, z.B. Schnittstellen für natürliche Sprache und Konversation. Die hier beschriebenen Systeme können einen transparenten Zugriff auf die zugrunde liegenden Backend-Speicher über eine abstrakte Ontologieabfragesprache (ontology query language, OQL) bereitstellen, die es den Benutzern ermöglicht, ihren Informationsbedarf in Bezug auf die Domänenontologie auszudrücken.

[0015] Um eine höhere Leistung für verschiedene Abfragetypen bereitzustellen, optimiert das System in einer Ausführungsform zunächst das Platzieren der KB-Daten in verschiedenen Speichern, indem es Teilsätze der Daten entsprechend ihren Fähigkeiten in geeigneten Backend-Speichern platziert. In einigen Ausführungsformen parst und kompiliert das System während der Laufzeit die OQL-Abfrage

und übersetzt sie in die Abfragesprachen und/oder APIs der verschiedenen Backend-Datenknoten. In einer Ausführungsform setzt das hier beschriebene System auch einen Abfrageorchestrator ein, der die Abfrage entsprechend der Platzierung der relevanten Daten an einen oder mehrere Backend-Speicher weiterleitet. Dieses effiziente Weiterleiten verringert die Latenzzeit, die erforderlich ist, um Ergebnisse zu erzeugen und an die anfordernde Entität zurückzusenden.

[0016] In Ausführungsformen können die abzufragenden KB-Daten beliebige Informationen umfassen, z.B. strukturierte, unstrukturierte und/oder halbstrukturierte Daten. Zur Unterstützung einer umfassenden Domänenspezialisierung nutzen die hier offenbarten Ausführungsformen Domänenontologien. In einer Ausführungsform definiert die Domänenontologie insbesondere nur die Entitäten und ihre Beziehungen auf der Ebene der Metadaten, ohne Informationen auf Instanzebene bereitzustellen. In einer Ausführungsform stellt die Domänenontologie somit das Domänenschema bereit, während die Daten auf Instanzebene in verschiedenen Backend-Datenspeichern (auch als Datenorte und Datenknoten bezeichnet) gespeichert werden. In Ausführungsformen können die Datenknoten eine beliebige Speicherarchitektur umfassen, z.B. relationale Datenbank(en), Dokumentenspeicher mit invertiertem Index, JSON-Speicher (JSON = JavaScript Object Notation), Graphdatenbank(en) und Ähnliches.

[0017] In einigen Ausführungsformen kann das System die KB-Daten in jedes beliebige Backend versetzen, speichern und indizieren, das die erforderlichen Fähigkeiten für die zu unterstützenden Abfragetypen bereitstellt. Anders ausgedrückt muss das System die Daten nicht in bestehende Betriebsdatenspeicher einbinden. Dies unterscheidet sich deutlich von föderierten Datenbanken und mediatorgestützten Ansätzen. In einigen Ausführungsformen führt das System einen fähigkeitsgestützten Datenplatzierungsschritt durch, bei dem Daten einer Wissensdatenbank, die einer gegebenen Ontologie entsprechen, in der Vielfalt von Backend-Ressourcen gespeichert werden. In mindestens einer Ausführungsform ist die Mehrfachspeicherarchitektur so konfiguriert, dass Datenversetzungen zwischen verschiedenen Datenspeichern minimiert werden, da Datenversetzungen nicht nur Kosten für die Datenübertragung verursachen, sondern auch hohe Kosten für die Datenkonvertierung mit sich bringen. Eine Möglichkeit, die Kosten für Datenversetzungen zu minimieren, besteht darin, die Daten in allen Backend-Knoten zu replizieren, um sicherzustellen, dass Abfragen von einem einzelnen Speicher ohne Datenversetzungen beantwortet werden können. Dies erfordert jedoch auch eine unnötige Duplizierung und verfehlt den Zweck einer Mehrfachspei-

cherarchitektur, die unterschiedlichen Fähigkeiten mehrerer Datenspeicher zu nutzen. In Ausführungsformen der vorliegenden Offenbarung können die KB-Daten in einer beliebigen Anzahl von Backend-Ressourcen gespeichert werden, und das System leitet Abfragen intelligent weiter, um den (die) am besten geeigneten Datenspeicher zu wählen und die Kosten für Datenversetzungen zu minimieren.

[0018] In einigen hier beschriebenen Ausführungsformen stellt die Systemarchitektur geeignete Abstraktionen zum Abfragen der Daten bereit, ohne zu wissen, wie die Daten in mehreren Datenspeichern gespeichert und indiziert sind. In einer Ausführungsform wird zu diesem Zweck eine Ontologieabfragesprache (OQL) eingeführt. Die OQL wird auf der Grundlage der Domänenontologie für das Unternehmen ausgedrückt. Benutzer des Systems müssen nur die Domänenontologie kennen, in der die Entitäten und ihre Beziehungen definiert sind. In Ausführungsformen versteht das System die Zuordnungen der verschiedenen Ontologiekonzepte zu den Backend-Datenspeichern sowie ihren entsprechenden Schemata und stellt Abfrageübersetzer von der OQL in die Zielabfragesprache des zugrunde liegenden Systems bereit.

[0019] In Ausführungsformen können bei jeder gegebenen Abfrage ein oder mehrere Datenknoten beteiligt sein. Ausführungsformen der vorliegenden Offenbarung stellen Techniken bereit, mit denen die beteiligten Speicher identifiziert und die entsprechenden Teilabfragen erzeugt werden, um das endgültige Abfrageergebnis zu berechnen. In einigen Ausführungsformen verwendet das System keinen Mediatoransatz, bei dem eine globale Abfrage in mehrere Teilabfragen unterteilt wird und die endgültigen Ergebnisse im Mediator zusammengestellt werden. Stattdessen werden ein oder mehrere Datenspeicher als Mediator verwendet, und der (die) Backend-Datenspeicher schließt (schließen) die Abfrageantworten ab. In einer Ausführungsform wird die relationale Datenbank zum Beispiel verwendet, um die Abfrageantwort abzuschließen, da der relationale Speicher wahrscheinlich in der Lage ist, Verknüpfungsoperationen schnell auszuführen.

[0020] Fig. 1 zeigt eine Architektur 100, die gemäß einer hier offenbarten Ausführungsform so konfiguriert ist, dass sie ontologische Abfragen weiterleitet. In der dargestellten Ausführungsform empfängt ein Abfrageverarbeitungsorchestrator 115 ontologische Abfragen 105, die mithilfe einer OQL erstellt wurden, die ein entsprechendes Ontologieschema 110 aufweist. Die ontologischen Abfragen 105 werden vom Abfrageverarbeitungsorchestrator 115 verarbeitet, um relevante Daten zu identifizieren und zurückzusenden. In Ausführungsformen leitet der Abfrageverarbeitungsorchestrator 115 die Abfragen an ein Backend 125 weiter, das eine Vielfalt von Datenkno-

ten 130A bis N aufweist. In der dargestellten Ausführungsform wird dieses Weiterleiten zumindest teilweise auf der Grundlage eines Satzes von Konzeptzuordnungen 140 und Fähigkeiten 135 durchgeführt.

[0021] In der dargestellten Ausführungsform sind die ontologischen Abfragen 105 auf der Grundlage einer OQL formatiert, die zur Darstellung von Abfragen verwendet wird, die mit dem Satz von Konzepten und Beziehungen im Ontologieschema 110 arbeiten. In einer Ausführungsform kann die OQL Abfragen ausdrücken, die Aggregationen, Verbunde, verschachtelte Teilabfragen und Ähnliches umfassen. In einigen Ausführungsformen kann die OQL auch Volltext- und feldspezifische Suchprädikate sowie Pfadabfragen ausdrücken. Eine OQL-Abfrage besteht im Allgemeinen aus einem einzelnen Abfrageblock oder einem Verbund mehrerer OQL-Abfrageblöcke, wobei jeder OQL-Abfrageblock aus einer SELECT-Klausel und einer FROM-Klausel besteht. In einigen Ausführungsformen kann ein OQL-Block auch eine WHERE-Klausel, eine GROUP-BY-Klausel, eine ORDER-BY-Klausel, eine FETCH-FIRST-Klausel und/oder eine HAVING-Klausel enthalten. In Ausführungsformen arbeitet die OQL-Abfrage mit dem Satz von Tupeln, der durch das kartesische Produkt der in der FROM-Klausel genannten Konzepte gebildet wird.

[0022] In einer Ausführungsform beschreibt das Ontologieschema 110 die Entitäten und ihre Beziehungen in einer KB auf einer semantischen Ebene, unabhängig davon, wie die Daten tatsächlich in Backend-Ressourcen gespeichert sind. In einer Ausführungsform beschreibt das Ontologieschema 110 die für die Domäne relevanten Entitäten, die den verschiedenen Entitäten möglicherweise zugehörigen Eigenschaften und die möglichen Beziehungen zwischen verschiedenen Entitäten. Das Ontologieschema 110 kann ein umfassendes und aussagekräftiges Datenmodell bereitstellen, das eine Vielfalt von realen Beziehungen zwischen Entitäten erfasst, z.B. funktionale Beziehungen, Vererbung, Verbunde und Ähnliches. In einigen Ausführungsformen enthält das Ontologieschema 110 keine Instanzdaten. Das Ontologieschema 110 definiert somit die Entitäten und Beziehungen, enthält jedoch keine Daten, die sich auf bestimmte Entitäten oder Beziehungen in der KB beziehen. So kann das Ontologieschema 110 beispielsweise definieren, dass eine Entität „company“ mehrere Eigenschaften wie „name“ und „address“ haben kann, aber das Ontologieschema 110 enthält keine Daten für bestimmte Instanzen, z.B. ein Unternehmen mit dem Namen „Main Street Grocer“ und der Adresse „123 Main Street“. Stattdessen werden diese Instanzdaten separat im Backend 125 gehalten.

[0023] In einer Ausführungsform geben die Konzeptzuordnungen 140 die Entsprechung zwischen

dem logischen Schema, das durch das Ontologieschema 110 dargestellt wird, und dem physischen Schema der zugrunde liegenden Datenknoten 130 im Backend 125 an. Angenommen, das Ontologieschema 110 ist zum Beispiel als $O(C, R, P)$ definiert, wobei $C = \{c_n \mid 1 \leq n \leq N\}$ einen Satz von Konzepten (auch als Entitäten bezeichnet), $R = \{r_k \mid 1 \leq k \leq K\}$ einen Satz von Beziehungen zwischen den Konzepten/Entitäten und $P = \{p_m \mid 1 \leq m \leq M\}$ einen Satz von Dateneigenschaften bezeichnet. In einer Ausführungsform besteht jede Beziehung zwischen zwei oder mehreren Konzepten, während jede Dateneigenschaft einem Merkmal eines Konzepts entspricht. In der dargestellten Ausführungsform geben die Konzeptzuordnungen 140 die Zuordnung zwischen Konzepten, Beziehungen und Eigenschaften (definiert im Ontologieschema 110) und den zugrunde liegenden Datenknoten 130 an. In einer solchen Ausführungsform geben die Konzeptzuordnungen 140 somit jeden Datenknoten 130 an, der ein gegebenes Konzept (z.B. eine Entität), eine Beziehung und/oder eine Eigenschaft speichert. Die Konzeptzuordnungen 140 können beispielsweise angeben, dass Instanzen der Entität „company“ in den Datenknoten 130A und 130M gespeichert werden, während Daten, die sich auf einen bestimmten Beziehungstyp zwischen den Entitäten „company“ beziehen, im Datenknoten 130B gespeichert werden.

[0024] In einer Ausführungsform geben die Fähigkeiten 135 die Operationen und Fähigkeiten an, die von jedem Datenknoten 130 bereitgestellt werden. In einigen Ausführungsformen werden die Fähigkeiten eines Datenknotens 130 in Form von Ansichten ausgedrückt, die alle möglichen Abfragen (Ansichtsdefinitionen) auflisten, die von dem Datenspeicher bearbeitet/beantwortet werden können. Dieser Ansatz ist zwar flexibel, aber nicht skalierbar, da die Anzahl der Ansichtsdefinitionen sehr groß sein kann, was möglicherweise zu dem Problem des Umschreibens von Abfragen unter Verwendung einer unendlichen Anzahl von Ansichten führt. Um die Skalierbarkeit zu verbessern, beschreiben einige Ausführungsformen der vorliegenden Offenbarung die Fähigkeiten der Backend-Datenknoten 130 anhand der von ihnen unterstützten Operationen (z.B. join (verknüpfen), group by (gruppieren nach), aggregation (Aggregation), Fuzzytext-Abgleich, Pfadausdrücke usw.), anstatt alle möglichen Abfragen aufzuführen, die von den Datenspeichern beantwortet werden können. Zusätzlich stellt mindestens eine Ausführungsform der vorliegenden Offenbarung eine detailliertere Beschreibung jeder unterstützten Operation bereit, indem ein Mechanismus verwendet wird, um alle zugehörigen Einschränkungen auszudrücken. In einer solchen Ausführungsform könnte z.B. eine Aggregationsfunktion vom Typ MAX nur für numerische Typen unterstützt werden.

[0025] In einer Ausführungsform geben die Fähigkeiten 135 für jeden gegebenen Datenknoten 130 somit den Satz von Operationen, die der Knoten ausführen kann (sowie in einigen Ausführungsformen relevante Einschränkungen dieser Fähigkeit). In einer Ausführungsform wertet der Abfrageverarbeitungsorchestrator 115 die Fähigkeiten 135 und Konzeptzuordnungen 140 aus, um einen oder mehrere Datenknoten 130 auszuwählen, an die die empfangene ontologische Abfrage 105 weitergeleitet werden soll. Dazu identifiziert der Abfrageverarbeitungsorchestrator 115 in einer Ausführungsform Datenknoten 130, die entweder die benötigten Daten enthalten (z.B. auf der Grundlage der Konzeptzuordnungen 140) oder die benötigte(n) Operation(en) ausführen können (z.B. auf der Grundlage der Fähigkeiten 135) oder beides. Der Abfrageverarbeitungsorchestrator 115 kann dann Teilabfragen für jeden ausgewählten Datenknoten 130 erzeugen.

[0026] In Ausführungsformen verwendet der Abfrageverarbeitungsorchestrator 115 einen Satz von Übersetzern 120A bis N, um die Teilabfragen nach Bedarf zu übersetzen. In der dargestellten Ausführungsform verfügt jeder Typ von Datenknoten 130 über einen entsprechenden Übersetzer 120. In einigen Ausführungsformen empfängt jeder Übersetzer 120A bis N die gesamte oder einen Teil einer OQL-Abfrage und erzeugt eine entsprechende Abfrage in der Sprache und/oder Syntax des entsprechenden Datenknotens 130A bis N. Der Übersetzer 120A kann beispielsweise SQL-Abfragen für eine im Datenknoten 130A enthaltene relationale Datenbank erzeugen, während der Übersetzer 120B Graphabfragen für einen im Datenknoten 130B enthaltenen Graphspeicher ausgibt.

[0027] In einer Ausführungsform stützen sich die Übersetzer 120 auf Schemazuordnungen, die die in der Domänenontologie dargestellten Konzepte und Beziehungen den entsprechenden Schemaobjekten im physischen Zielschema zuordnen. Bei einem relationalen Backend-Datenknoten 130 können die Schemazuordnungen beispielsweise eine Entsprechung zwischen (1) den Konzepten in der Ontologie und den Tabellen im relationalen Schema; (2) Dateneigenschaften oder Attributen von Konzepten in der Ontologie und den Tabellenspalten im physischen Schema; und/oder (3) den Beziehungen zwischen den Konzepten in der Ontologie und den Primärschlüssel/Fremdschlüssel-Beschränkungen zwischen den Tabellen, die den Konzepten in der Datenbank entsprechen, bereitstellen. In ähnlicher Weise können die Schemazuordnungen für einen JSON-Dokumentenspeicher die in der Ontologie dargestellten Konzepte, Dateneigenschaften und Beziehungen den entsprechenden Feldpfaden im JSON-Dokument zuordnen.

[0028] In einigen Ausführungsformen bearbeiten die Übersetzer 120 auch spezielle Konzepte und Beziehungen, die in einer Ontologie dargestellt werden können, z.B. Verbund, Vererbung und Traversierungen zwischen Konzepten, die in der Regel Verknüpfungsbedingungen zwischen den Ontologiekonzepten darstellen. Je nach physischem Datenlayout werden diese in entsprechende Operationen übersetzt, die von den Datenknoten 130 des Backend 125 unterstützt werden.

[0029] In Ausführungsformen empfängt jeder Datenknoten 130 Teilabfragen und erzeugt eine Antwort für den Abfrageverarbeitungsorchestrator 115. Wenn der Datenknoten 130 lokal über alle erforderlichen Daten verfügt und in der Lage ist, die angegebene(n) Operation(en) auszuführen, führt der Knoten die Abfrage aus und sendet Ergebnisse zurück. In einigen Ausführungsformen kann die Teilabfrage angeben, dass der Datenknoten 130 eine Abfrage an einen oder mehrere andere Datenknoten 130 übertragen soll, um Daten abzurufen, die zum Ausführen der Abfrage benötigt werden. In einigen Ausführungsformen beispielsweise werden die Verknüpfungsoperationen vorzugsweise von einem relationalen Datenknoten durchgeführt, da relationale Systeme in der Regel eine geringe Latenzzeit für Verknüpfungsoperationen aufweisen. Hinzu kommt, dass einige Operationen nur an bestimmten Knoten möglich sind. Angenommen, der Datenknoten 130A ist zum Beispiel der einzige Knoten, der Verknüpfungsoperationen ausführen kann, die zu verknüpfenden Daten existieren jedoch nur auf dem Datenknoten 130B. In einer Ausführungsform empfängt der Datenknoten 130A eine Teilabfrage, die ihn anweist, die relevanten Daten zu verknüpfen, sowie eine oder mehrere andere Teilabfragen, die an den Datenknoten 130B weitergeleitet werden, um die Daten abzurufen.

[0030] In einer solchen Ausführungsform bereitet der Abfrageverarbeitungsorchestrator 115 somit übersetzte Teilabfragen für den Datenknoten 130B vor und übermittelt sie an den Datenknoten 130A. Dadurch kann der Datenknoten 130A sie einfach an den Datenknoten 130B weiterleiten. Die Daten werden dann vom Datenknoten 130B an den Datenknoten 130A zurückgesendet, der die Operation ausführt und die Ergebnisse an den Abfrageverarbeitungsorchestrator 115 zurücksendet. In einer anderen Ausführungsform kann der Abfrageverarbeitungsorchestrator 115 die Teilabfragen an den Datenknoten 130B übertragen und die sich ergebenden Daten zum Verarbeiten an den Datenknoten 130A weiterleiten. In einer weiteren Ausführungsform führt der Abfrageverarbeitungsorchestrator 115 die Verknüpfung lokal durch.

[0031] In der dargestellten Ausführungsform umfasst die Architektur 100 weiterhin einen Daten-

platzierungsorchestrator 150, der ermittelt, welche(r) Datenknoten 130A bis N zum Speichern der Daten in der Wissensdatenbank verwendet werden soll(en). Wie dargestellt, empfängt der Datenplatzierungsorchestrator 150 eine ähnliche Angabe zu den Fähigkeiten 155 der Datenknoten 130 sowie eine Angabe zu der Abfragearbeitslast 160. In einer Ausführungsform gibt die Abfragearbeitslast 160 den durchschnittlichen oder erwarteten Satz von Abfragen für die Wissensdatenbank an. In einer Ausführungsform wird die Abfragearbeitslast 160 beispielsweise durch Beobachten von Benutzerinteraktionen mit der Wissensdatenbank im Laufe der Zeit erzeugt. Das System kann dann diese Interaktionen zusammenfassen, um die durchschnittliche, erwartete oder typische Arbeitslast für das System zu ermitteln. In einigen Ausführungsformen umfasst die Abfragearbeitslast 160 eine Angabe darüber, welche(s) Konzept(e) gemeinsam abgefragt wird (werden), welche Operation(en) auf jedes Konzept angewandt wird (werden) und Ähnliches.

[0032] In Ausführungsformen ermittelt der Datenplatzierungsorchestrator 150 auf der Grundlage der Knotenfähigkeiten 155 und der bekannten Abfragearbeitslast 160 für jedes Konzept/jede Entität im Ontologieschema 110, welche(r) Datenknoten 130 Daten auf Instanzebene für das Konzept speichern soll(en). Beispielsweise kann der Datenplatzierungsorchestrator 150 auf der Grundlage der Operationen, die ein gegebener Datenknoten 130A durchführen kann, und weiterhin auf der Grundlage der Abfragearbeitslast 160 feststellen, dass der Datenknoten 130A alle Daten für eine Entität „company“ und eine Entität „public metric“ speichern soll, da sie häufig zusammen abgefragt werden. In ähnlicher Weise kann der Datenplatzierungsorchestrator 150 auf der Grundlage eines Feststellens, dass die Abfragen häufig Durchführen von Fuzzymatch-Operationen bei den „document“-Daten umfassen und der Datenknoten 130B Fuzzymatch-Operationen unterstützt, feststellen, dass Instanzen eines Konzepts „document“ im Datenknoten 130B platziert werden.

[0033] Diese intelligente Datenplatzierung kann spätere Datenübertragungen während der Laufzeit verringern. In Ausführungsformen kann der Datenplatzierungsorchestrator 150 zu Beginn (z.B. um die ersten Platzierungen bereitzustellen) und/oder in regelmäßigen Abständen während der Laufzeit (z.B. um die Platzierungsentscheidungen auf der Grundlage der Entwicklung der Abfragearbeitslast 160 im Laufe der Zeit zu überarbeiten) eingesetzt werden. Auch wenn die Operationen des Abfrageverarbeitungsorchestrators 115 und des Datenplatzierungsorchestrators 150 aus Gründen der Klarheit des Konzepts als einzelne Komponenten dargestellt werden, können sie in Ausführungsformen kombiniert oder in einer beliebigen Anzahl von Komponenten verteilt werden.

[0034] Wie dargestellt, gibt der Datenplatzierungsorchestrator 150 einen Satz von Platzierungsentscheidungen 165 an das Backend 125 und/oder an einen oder mehrere Zwischendienste wie z.B. einen Extraktions-, Transformations- und Ladedienst (extract, transform, and load, ETL) aus. Die Daten auf Instanzebene in der Wissensdatenbank werden dann auf der Grundlage dieser Auswahl in den entsprechenden Datenknoten 130 gespeichert. Die Konzeptzuordnungen 140 spiegeln die aktuelle Platzierung der Daten wider. Wenn der Datenplatzierungsorchestrator 150 verwendet wird, um die Datenplatzierung auf der Grundlage aktualisierter Abfragearbeitslasten 160 zu überarbeiten, werden die Konzeptzuordnungen 140 in einer Ausführungsform in ähnlicher Weise aktualisiert. Im Folgenden wird in **Fig. 2** bis **Fig. 9** der Abfrageverarbeitungsorchestrator 115 näher beschrieben, wobei davon ausgegangen wird, dass die Daten bereits platziert wurden. Der Datenplatzierungsorchestrator 150 und verschiedene Techniken zur Gewährleistung einer effizienten Datenplatzierung werden in den **Fig. 10** bis **Fig. 17** näher beschrieben.

[0035] **Fig. 2** veranschaulicht einen Arbeitsablauf 200 zum Verarbeiten und Weiterleiten von ontologischen Abfragen gemäß einer hier offenbarten Ausführungsform. Wie dargestellt, beginnt der Arbeitsablauf 200, wenn eine OQL-Abfrage 205 empfangen wird. Die OQL-Abfrage 205 wird einem OQL-Parser 210 bereitgestellt, der die Abfrage parst, um die Bedeutung der Abfrage zu ermitteln. Wie durch Pfeil 215 angezeigt, wird diese geparste OQL-Abfrage dann einem Abfragegraphmodellkonstrukteur 220 (Query Graph Model Constructor, QGM-Konstrukteur) bereitgestellt. In einer Ausführungsform erzeugt der QGM-Konstrukteur 200 eine logische Darstellung der Abfrage in Form eines Abfragegraphmodells. Dadurch wird die Komplexität der Abfragekompilierung und -optimierung verringert. Ein Beispiel für ein Abfragegraphmodell wird im Folgenden mit Bezug auf die **Fig. 4A** und **Fig. 4B** näher beschrieben. In einer Ausführungsform erfasst das QGM den Datenfluss und die Abhängigkeiten in einer Abfrage mithilfe von Operatorfeldern, z.B. SELECT, GROUP BY, SETOP und Ähnliches. In Ausführungsformen können die Operationen in einem Feld untereinander frei umgeordnet werden, aber die Feldgrenzen werden beim Erzeugen der Abfrageausführungspläne beachtet. Die Abfrageausführungspläne müssen somit der Reihenfolge der Abfragefelder folgen.

[0036] In einer Ausführungsform wird der Datenfluss zwischen den Abfragefeldern durch Quantoren dargestellt. Dieses Format ermöglicht es dem System, Abfrageäquivalenzen zu analysieren und Umschreiboptimierungen anzuwenden. In einer Ausführungsform ermöglicht die QGM-Darstellung der Abfrage dem System, sich auf Optimieren des Datenflusses zwischen den verschiedenen Datenspeichern wäh-

rend der Abfrageausführung zu konzentrieren, während die Wahl des tatsächlichen physischen Ausführungsplans dem zugrunde liegenden Datenknoten 130 überlassen bleibt, der für das Ausführen des Abfragefelds oder -fragments zuständig ist. In Ausführungsformen wird das QGM 225 verwendet, um einen optimierten Mehrfachspeicher-Ausführungsplan zu erstellen, der die Datenversetzung und -transformation in verschiedenen Backends auf ein Mindestmaß beschränkt.

[0037] In einigen Ausführungsformen umfasst das QGM 225 am unteren Ende einen Satz von Quantoren, die die Eingabekonzeptsätze für die Abfrage bereitstellen, und jeder Abfrageblock im QGM hat eine Kopfzeile und einen Hauptteil. Der Hauptteil jedes Blocks umfasst einen Satz von Prädikaten, die die Satzoperationen beschreiben, die an den Eingabekonzeptsätzen (z.B. Verknüpfungen) durchgeführt werden sollen, und die Ausdrücke in der Kopfzeile beschreiben, wie die Ausgabeigenschaften des Ergebniskonzepts berechnet werden sollen. Anders ausgedrückt enthält der Hauptteil jedes Feldes in einer Ausführungsform einen Satz von Prädikaten (auch als Operationen bezeichnet), die jeweils auf die Eingabequantoren für das Feld anzuwenden sind. In einigen Ausführungsformen wird ein Prädikat/eine Operation, das (die) sich auf einen einzelnen Quantor bezieht, als lokales Prädikat klassifiziert, während ein Prädikat/eine Operation, das (die) sich auf mehrere Quantoren bezieht, ein Verknüpfungsprädikat ausdrückt.

[0038] Im dargestellten Arbeitsablauf 200 wird das QGM 225 an eine Operatorplatzierungskomponente 230 weitergeleitet, die den Prozess der Abfrageweiterleitung fortsetzt. Wie dargestellt, empfängt die Operatorplatzierungskomponente 230 weiterhin den Satz von Konzeptzuordnungen 235 und Knotenfähigkeiten 240 und annotiert die Abfrageblöcke im QGM 225 auf der Grundlage der Fähigkeiten und Konzeptzuordnungen. In einer Ausführungsform geht die Operatorplatzierungskomponente 230 das QGM von unten nach oben durch und annotiert jede Operation in der Abfrage mit dem Satz möglicher Speicher, in denen die Operation durchgeführt werden kann. In einigen Ausführungsformen werden Annotationen für lokale und Verknüpfungsprädikate unterschiedlich erzeugt. Es sei daran erinnert, dass alle Ausdrücke der Kopfzeile, die einem einzelnen Quantor zugehörig sind, in einigen Ausführungsformen wie lokale Prädikate behandelt werden.

[0039] Wenn der Abfrageblock ein lokales Prädikat/eine lokale Operation (z.B. einen einzelnen Quantor) umfasst, ermittelt die Operatorplatzierungskomponente 230 in einer Ausführungsform, ob der Quantor ein Grundkonzept ist. Ist dies der Fall, wird das Prädikat mit einer Annotation versehen, die angibt, welche(r) Datenknoten 130 sowohl (i) das

Konzept enthält (enthalten) als auch (ii) das Prädikat ausführen kann (können). Wenn der Quantor aus einem anderen QGM-Block stammt (d.h., er wird von einem anderen Block berechnet), annotiert die Operatorplatzierungskomponente 230 in einer Ausführungsform das Prädikat mit dem Satz von Datenspeichern, die (i) das QGM-Feld dieses Quantors ausführen und (ii) in der Lage sind, das Prädikat auszuführen. Wenn keiner der Datenknoten 130, die die Eingabe für dieses Prädikat enthalten, in der Lage ist, das Prädikat auszuführen, wird es in einigen Ausführungsformen von der Operatorplatzierungskomponente 230 mit dem Satz von Speichern annotiert, die in der Lage sind, das Prädikat auszuführen.

[0040] Wenn das Prädikat beispielsweise eine Fuzzy-Suche enthält, die Daten jedoch nur im relationalen Backend-Datenknoten 130 gespeichert sind, kann die Operatorplatzierungskomponente 230 das Prädikat mit einem Dokumentenspeicher annotieren, der in der Lage ist, diese Fuzzy-Suche zu berechnen, auch wenn die Daten dort nicht gespeichert sind. Es sei darauf hingewiesen, dass die Daten in solchen Fällen während der Ausführung der Abfrage versetzt werden müssen.

[0041] Umfasst der Abfrageblock ein Verknüpfungsprädikat/eine Verknüpfungsoperation, prüft die Operatorplatzierungskomponente 230 in einigen Ausführungsformen den Verknüpfungstyp und das Verknüfungsprädikat. In einer Ausführungsform ist jedes Verknüfungsprädikat zwei oder mehr Quantoren zugehörig: einem für jede der Verknüpfungseingaben. In Ausführungsformen identifiziert die Operatorplatzierungskomponente 230 diese Quantoren auf der Grundlage dessen, ob es sich bei den identifizierten Quantoren um berechnete Eingaben oder um Grundkonzepte handelt und woher der Quantor stammt (z.B. ob er lokal berechnet und/oder gespeichert wird oder ob er von einem anderen Knoten empfangen wird).

[0042] Für den Fall, dass sich alle Quantoren auf Grundkonzepte erstrecken, wertet die Operatorplatzierungskomponente 230 den Satz von Datenknoten 130 aus, in dem sich die Grundkonzepte befinden, und ermittelt für jeden solchen Speicher, ob der Speicher Verknüpfungsoperationen unterstützt. Die Operatorplatzierungskomponente 230 annotiert dann die Verknüpfungsoperation mit einem Hinweis auf die Speicher, die eines oder mehrere der benötigten Konzepte enthalten und den Typ der Verknüpfungsoperation unterstützen. Wenn einer der Speicher ein relationaler Knoten ist, annotiert die Operatorplatzierungskomponente 230 die Operation in einigen Ausführungsformen mit einem Hinweis auf diesen relationalen Datenknoten 130 anstelle der übrigen Knoten.

[0043] Wenn beide Quantoren von anderen QGM-Abfrageblöcken erzeugt werden, annotiert die Operatorplatzierungskomponente 230 in einigen Ausführungsformen die Verknüpfungsoperation mit den Datenknoten 130, die den Typ der Verknüpfungsoperation unterstützen (in einigen Ausführungsformen nur solche, die relationale Speicher umfassen). In einer Ausführungsform annotiert die Operatorplatzierungskomponente 230 die Operation weiterhin mit einer Angabe der Datenspeicher, die die Quantoren erzeugen, wenn sie in der Lage sind, die Verknüpfung (JOIN) auszuführen.

[0044] Wenn einer der Quantoren ein Grundkonzept ist und ein anderer von einem anderen QGM-Abfrageblock berechnet wird, ist die Entscheidung über die Platzierung der Verknüpfungsoperation in einer weiteren Ausführungsform ähnlich wie in der obigen Beschreibung, wenn die Quantoren beide von anderen QGM-Blöcken berechnet werden. In einer Ausführungsform annotiert die Operatorplatzierungskomponente 230 die Verknüpfungsoperation mit dem Satz von Datenspeichern, die den Verknüpfungstyp unterstützen und für mindestens eine der Verknüpfungseingaben lokal sind.

[0045] Wie dargestellt, wird dieses annotierte QGM 245 dann an die Blockplatzierungskomponente 250 weitergeleitet. In einer Ausführungsform nutzt die Blockplatzierungskomponente 250 die von der Operatorplatzierungskomponente 230 erzeugten Annotationen, um die Platzierungsoptionen für die Abfragefelder in der Abfrage zu ermitteln.

[0046] In einer Ausführungsform wird Ermitteln der Platzierungsoptionen für die Abfragefelder „select“ als ein Problem des Ermitteln der Mindestsatzabdeckung modelliert. Die Blockplatzierungskomponente 250 ermittelt somit die Mindestanzahl von Datenknoten 130, die erforderlich sind, um die Platzierung aller Operatoren innerhalb des gegebenen Auswahlfeldes auszuführen. In einigen Ausführungsformen verwendet die Blockplatzierungskomponente 250 eine Greedy-Heuristik zum Durchführen der Prädikatspeichergruppierung, die garantiert, dass jedes Prädikat in einem einzigen Speicher platziert wird und die Gesamtzahl der vom Abfragefeld abgedeckten Speicher minimiert wird. Sobald jedes Prädikat mit dem entsprechenden Speicher annotiert ist, in dem es platziert werden soll, ermittelt die Blockplatzierungskomponente 250, ob der Abfrageblock aufgeteilt werden muss.

[0047] Wenn alle Prädikate im Abfragefeld „select“ im selben Datenknoten 130 platziert sind, wird das Abfragefeld in einer Ausführungsform mit der Annotation versehen, dass es in diesem Speicher platziert ist. Wenn die Prädikate im Abfragefeld „select“ dagegen in mehr als einem Datenknoten 130 platziert sind (was bedeutet, dass das Abfragefeld von mehreren

Speichern verarbeitet werden muss), stellt die Blockplatzierungskomponente 250 fest, dass der Block in mehrere Abfragefelder aufgeteilt werden muss. In diesem Fall teilt die Blockplatzierungskomponente 250 den Block dann so auf, dass jeder sich ergebende (Teil-)Block Prädikate umfasst, die einem einzelnen Datenknoten zugewiesen sind.

[0048] In einigen Ausführungsformen ermittelt die Blockplatzierungskomponente 250 für Abfragefelder „group by“ die Platzierung auf der Grundlage der Speicher, die den Typ der Aggregationsoperation unterstützen, und des Speichers, der das vorangehende Abfragefeld verarbeitet, das dem Feld „group by“ Daten zuführt. Wenn der Speicher, der diese Eingaben zuführt, auch die Funktionen „group by“ und „aggregation“ unterstützt, platziert die Blockplatzierungskomponente 250 in einer Ausführungsform das ausgewählte Feld „group by“ im selben Speicher. Wenn dieser Speicher diese Operationen jedoch nicht unterstützt, annotiert die Blockplatzierungskomponente 250 das Feld mit einer Liste der Datenknoten 130, die die Operation „group by“ ausführen können. Es sei darauf hingewiesen, dass eine solche Platzierung eine Datenversetzung vom zuführenden Speicher zu dem Speicher erfordern würde, der das Feld verarbeiten kann.

[0049] Sobald die Blockplatzierungskomponente 250 jeden Abfrageblock mit allen möglichen Platzierungen annotiert hat (z.B. alle Datenknoten 130, die den Block ausführen können), ermittelt die Kostenkomponente 255 die Kosten jeder Kombination von Platzierungen. In einer Ausführungsform konzentriert sich das Kostenmodell für die Abfrageweiterleitung nur auf die Kosten für die Datenübertragung und -konvertierung, um zwischen alternativen Abfrageausführungsplänen zu wählen, da die Kosten für die Datenversetzung und -transformation wahrscheinlich höher ausfallen als die Gesamtkosten für jeden Ausführungsplan in der Mehrfachspeicherumgebung. Zumindest in einigen Ausführungsformen werden daher die Kosten für das Ausführen eines gegebenen Ausführungsplans ermittelt, indem die Kosten der Datenversetzung für jedes Paar aus Quell- und Zieldatenspeicher im Abfrageausführungsplan zusammengefasst werden. Es sei darauf hingewiesen, dass in einigen Ausführungsformen verschiedene Operationen, z.B. Verknüpfungs- und Graphoperationen, mit sehr unterschiedlicher Leistung auf verschiedenen Datenknoten 130 ausgeführt werden können. Obwohl Verknüpfungsoperationen beispielsweise von einer Vielfalt von Speichern (z.B. JSON-Speichern) unterstützt werden können, stellen relationale Datenbanken in der Regel die beste Leistung für Verknüpfungsoperationen bereit.

[0050] In einigen Ausführungsformen berücksichtigt das Kostenmodell weiterhin diese Abweichung bei den Ausführungskosten der Operationen für ver-

schiedene Speicher. In anderen Ausführungsformen umfasst das Kostenmodell diese Faktoren nicht. In einer solchen Ausführungsform werden solche Fälle mithilfe eines deklarativen Mechanismus behandelt, der die Fähigkeiten der zugrunde liegenden Speicher ausdrückt. Um beispielsweise das Ausführen einer Verknüpfungsoperation für einen JSON-Speicher zu unterbinden, kann das System die Verknüpfungsfähigkeit von JSON-Speichern in der Fähigkeitsbeschreibung vollständig unterdrücken oder Einschränkungen hinzufügen, um ihre Anwendbarkeit für bestimmte Datentypen einzuschränken.

[0051] In einer Ausführungsform führt die Kostenkomponente 255 alle möglichen Platzierungskombinationen in allen Abfrageblöcken auf und erzeugt für jede dieser Kombinationen einen Ausführungsplan, indem sie bestimmte Abfragefelder zu Gruppen zusammenfasst. In einer Ausführungsform umfasst jede Gruppe im Abfrageplan Abfragefelder, die sowohl (i) im QGM aufeinander folgen (z.B. durch einen einzigen Zwischenschritt getrennt sind) als auch (ii) von demselben Datenknoten 130 verarbeitet werden können. Die Kostenkomponente 255 verbindet dann diese Gruppen von Blöcken auf der Grundlage des QGM-Ablaufs und definiert Kanten, die den Datenfluss zwischen den Datenknoten 130 darstellen. In einer Ausführungsform erzeugt die Kostenkomponente 255 weiterhin einen Satz von Datenversetzungsdeskriptoren für jeden Ausführungsplan auf der Grundlage der Kanten zwischen Gruppen im erzeugten Plan.

[0052] In einigen Ausführungsformen verwendet die Kostenkomponente 255 mithilfe dieser Versetzungsdeskriptoren für jeden Plan das oben beschriebene Datenversetzungskostenmodell, um die Kosten für jeden Ausführungsplan zu ermitteln, und wählt denjenigen mit den geringsten Kosten aus. In einer Ausführungsform ermittelt die Kostenkomponente 255 beispielsweise für jeden möglichen Ausführungsplan die Kosten für jeden der entsprechenden Versetzungsdeskriptoren. Dies kann Latenzzeiten, Datenverarbeitungskosten und Ähnliches umfassen. Die Werte können dann in jedem Ausführungsplan zusammengefasst werden, um zu ermitteln, welche (r) Ausführungsplan(pläne) die geringsten Kosten aufweist (aufweisen). Wie dargestellt, wird dieser Plan mit den geringsten Kosten ausgewählt und verwendet, um eine oder mehrere übersetzte Abfragen 260 für die Datenknoten 130 zu erzeugen.

[0053] Die **Fig. 3A** und **Fig. 3B** zeigen eine beispielhafte Ontologie 300, die gemäß einer hier offenbarten Ausführungsform zum Auswerten und Weiterleiten von Abfragen verwendet werden kann. In der dargestellten Ausführungsform wird jedes Konzept 305A bis I durch eine Ellipse dargestellt, während jede Eigenschaft 310A bis N durch ein abgerundetes Rechteck dargestellt wird. Weiterhin werden Pfeile

verwendet, um die Zugehörigkeiten zwischen den Konzepten 305 und den Eigenschaften 310 darzustellen, während die fettgedruckten Pfeile die Beziehungen zwischen den Konzepten 305 darstellen. Zusätzlich wird jeder Beziehungspfeil auf der Grundlage des Beziehungstyps gekennzeichnet. Wie in der Abbildung dargestellt, ist beispielsweise das Konzept „Company“ 305B eine Teilklasse des Konzepts „PublicCompany“ 305D. Weiterhin ist das Konzept „Company“ 305B mehreren Eigenschaften 310 zugehörig, einschließlich einer Eigenschaft „name“ 310E und einer Eigenschaft „identifizier“ 310D, bei denen es sich um Zeichenketten handelt.

[0054] In einer Ausführungsform entsprechen die Daten in der Wissensdatenbank wie oben beschrieben der Ontologie 300. Anders ausgedrückt definiert die Ontologie 300 die Konzepte und Entitäten in der Wissensdatenbank sowie die Beziehungen zwischen den Entitäten und die jedem Konzept zugehörigen Eigenschaften. Die Ontologie 300 umfasst insbesondere keine Instanzdaten (z.B. Daten über ein bestimmtes Unternehmen), sondern definiert die Struktur der Daten. In einigen Ausführungsformen können die Konzepte 305, Eigenschaften 310 und/oder Beziehungen in einer beliebigen Anzahl von Datenknoten 130 verteilt sein. In einer Ausführungsform werden die Daten zumindest teilweise auf der Grundlage des Datentyps in den Datenknoten 130 platziert.

[0055] Wenn die Konzeptzuordnungen angeben, dass ein gegebenes Konzept 305 in einem gegebenen Datenknoten 130 gespeichert ist, werden in einer solchen Ausführungsform alle Instanzdaten, die diesem Konzept 305 entsprechen, in dem Datenknoten 130 gespeichert. Wenn die Zuordnung beispielsweise angibt, dass ein Datenknoten 130A das Konzept „PublicMetric“ 305F umfasst, kann der Abfrageverarbeitungsorchestrator 115 Daten über eine beliebige Instanz von „PublicMetric“ (z.B. eine beliebige Maßzahl für ein beliebiges Unternehmen) von dem Datenknoten 130A abrufen. Wenn demnach eine empfangene Abfrage Zugreifen auf „PublicMetric“-Daten erfordert, leitet der Abfrageverarbeitungsorchestrator 115 zumindest einen Teil der Abfrage an den Datenknoten 130A (oder an einen anderen Knoten, der ebenfalls das Konzept 305F unterstützt) weiter.

[0056] Die **Fig. 4A** und **Fig. 4B** veranschaulichen gemäß einer hier offenbarten Ausführungsform einen Arbeitsablauf 400 zum Parsen und Weiterleiten einer beispielhaften ontologischen Abfrage. In der dargestellten Ausführungsform wird die Eingabe 405 empfangen und ausgewertet, um relevante Daten von den Datenknoten 130 abzurufen. In der dargestellten Ausführungsform ist die Eingabe 405 ein Text in natürlicher Sprache (z.B. von einem Benutzer). In verschiedenen Ausführungsformen

kann die Eingabe 405 jedoch auch eine Abfrage oder andere Daten umfassen. Weiterhin kann die Eingabe 405 von einer beliebigen Anzahl von Quellen empfangen werden, darunter von automatisierten Anwendungen, benutzerorientierten Anwendungen, direkt von einem Benutzer und Ähnliches. In einigen Ausführungsformen wird die Eingabe 405 als Teil eines Chatbots oder einer anderen interaktiven Anwendung empfangen, die es dem Benutzer ermöglicht, die Wissensdatenbank zu durchsuchen und zu erkunden.

[0057] Im dargestellten Beispiel handelt es sich bei der Eingabe 405 um den Satz „Show me the total revenue of all companies that filed technology patents in the last 5 years“ (Gesamtumsatz aller Unternehmen anzeigen, die in den letzten 5 Jahren Technologiepatente angemeldet haben). Im dargestellten Arbeitsablauf 400 wird die Eingabe 405 geparkt und mit einer oder mehreren Techniken zum Verarbeiten natürlicher Sprache (natural language processing, NLP), z.B. semantische Analyse, Schlüsselwortsuche, Stimmungsanalyse, Absichtsanalyse und Ähnliches, ausgewertet. So kann das System auf der Grundlage der Eingabe 405 eine OQL-Abfrage 410 erzeugen. In einigen Ausführungsformen ist die Eingabe 405 natürlich selbst eine OQL-Abfrage.

[0058] Wie dargestellt, umfasst die entsprechende OQL-Abfrage 410 für die Eingabe 405 eine Operation „select“ und eine Operation „group by“ sowie eine Angabe zu den relevanten Tabellen oder Konzepten und eine „where“-Klausel, die die Einschränkungen der Abfrage angibt. Diese OQL-Abfrage 410 wird dann vom Abfrageverarbeitungsorchestrator 115 geparkt, um ein OGM 435A zu erzeugen, bei dem es sich um eine logische Darstellung der Abfrage handelt und das eine Reihe von Abfrageblöcken umfasst. In der dargestellten Ausführungsform umfasst das OGM einen Abfrageblock „select“ 415B und einen Abfrageblock „group by“ 415A.

[0059] Wie dargestellt, umfasst das QGM 435A im unteren Bereich einen Satz von Quantoren 430A bis E, die die Eingabekonzeptsätze für die Abfrage bereitstellen. Dazu gehören in der dargestellten Ausführungsform „PublicMetricData“, „PublicMetric“, „PublicCompany“, „Document“ und „Companyinfo“. In einer Ausführungsform umfasst jeder Abfrageblock 415 im QGM 435A eine Kopfzeile 420 und einen Hauptteil 425. Der Hauptteil 425 beschreibt im Allgemeinen die Satzoperationen, die bei den Eingabekonzeptsätzen durchgeführt werden sollen (z.B. Verknüpfungen), während die Ausdrücke in der Kopfzeile 420 beschreiben, wie die Ausgabeeigenschaften des Ergebniskonzepts berechnet werden sollen. In der dargestellten Ausführungsform spezifiziert die Kopfzeile 420B des Abfrageblocks „select“ 415B beispielsweise die Ausgabeeigenschaften

„oPMD.value“, „oPMD.year_calendar“ und „oCI.id“. Der Hauptteil 425B enthält einen Satz von Prädikaten, die auf die Eingabequantoren 430A bis E anzuwenden sind. Wie oben beschrieben, wird in einigen Ausführungsformen ein Prädikat, das sich auf einen einzelnen Quantor bezieht, als lokales Prädikat bezeichnet, während ein Prädikat, das sich auf mehrere Quantoren bezieht, ein Verknüpfungsprädikat ausdrückt.

[0060] Wie in **Fig. 4B** dargestellt, berücksichtigt der Abfrageverarbeitungsorchestrator 115, dass der Abfrageblock „select“ 415B Prädikate umfasst, die von verschiedenen Datenknoten 130 ausgeführt werden müssen. Obwohl die meisten Prädikate im Abfrageblock 415B von einem relationalen Datenspeicher ausgeführt werden können, hat der Abfrageverarbeitungsorchestrator 115 insbesondere ermittelt, dass die Prädikate „oD -> companyInfo = oCI“ und „oD.selfMATCH(„Tech Patent Filed“)“ Operationen entsprechen, die mit einer elastischen Suche durchgeführt werden, die der relationale Datenbankknoten nicht unterstützt. Daher hat der Abfrageverarbeitungsorchestrator 115 den Abfrageblock 415B aufgeteilt, indem er diese Prädikate in einen neuen Abfrageblock 415C abtrennt. Wie dargestellt, dient dieser Abfrageblock 415C als Eingabe für den Abfrageblock 415B.

[0061] In der dargestellten Ausführungsform hat der Abfrageverarbeitungsorchestrator 115 durch Aufteilen des Abfrageblocks 415B sichergestellt, dass jeder Block vollständig in einem einzelnen Datenknoten 130 ausgeführt werden kann. Beispielsweise können sowohl der Abfrageblock 415A als auch der Abfrageblock 415B in einem relationalen Datenknoten 130 ausgeführt werden, während der Abfrageblock 415C von einem Datenknoten 130 ausgeführt werden muss, der eine elastische Suche unterstützt. In einer Ausführungsform identifiziert oder erzeugt der Abfrageverarbeitungsorchestrator 115 daher eine Teilabfrage, die den Abfrageblöcken 415A und 415B entspricht, übersetzt die Teilabfrage in die entsprechende Sprache und/oder das entsprechende Format des relationalen Knotens und überträgt die übersetzte Teilabfrage an den relationalen Knoten.

[0062] In der dargestellten Ausführungsform identifiziert oder erzeugt der Abfrageverarbeitungsorchestrator 115 weiterhin eine Teilabfrage, um den Abfrageblock 415C auszuführen, und übersetzt sie in eine Sprache und/oder ein Format, die (das) vom Knoten der elastischen Suche unterstützt wird. In einer Ausführungsform überträgt der Abfrageverarbeitungsorchestrator 115 diese Teilabfrage zusätzlich an den relationalen Datenspeicher, der als Aggregator fungieren soll. Der relationale Knoten kann dann die Teilabfrage an den elastischen Knoten weiterleiten und seine eigene Teilabfrage unter Verwendung der

vom elastischen Knoten ausgegebenen Ergebnisse ausführen.

[0063] Fig. 5 ist ein Blockschaubild, das gemäß einer hier offenbarten Ausführungsform einen Abfrageverarbeitungsorchestrator 115 veranschaulicht, der so konfiguriert ist, dass er ontologische Abfragen weiterleitet. Der Abfrageverarbeitungsorchestrator 115 ist in Ausführungsformen zwar als physische Einheit dargestellt, er kann jedoch unter Verwendung virtueller Einheit(en) und/oder in einer Anzahl von Einheiten (z.B. in einer Cloud-Umgebung) implementiert werden. Wie dargestellt, umfasst der Abfrageverarbeitungsorchestrator 115 einen Prozessor 510, einen Arbeitsspeicher 515, einen Speicher 520, eine Netzwerkschnittstelle 525 und eine oder mehrere E/A-Schnittstellen 530. In der dargestellten Ausführungsform ruft der Prozessor 510 die im Arbeitsspeicher 515 gespeicherten Programmanweisungen ab und führt sie aus, und er speichert die im Speicher 520 befindlichen Anwendungsdaten und ruft diese ab. Der Prozessor 510 steht im Allgemeinen für eine einzelne CPU und/oder GPU, mehrere CPUs und/oder GPUs, eine einzelne CPU und/oder GPU mit mehreren Verarbeitungskernen und Ähnliches. Der Arbeitsspeicher 515 ist allgemein stellvertretend für einen Direktzugriffsspeicher enthalten. Beim Speicher 520 kann es sich um eine beliebige Kombination von Festplattenlaufwerken, flashgestützten Speichereinheiten und Ähnliches handeln, die feste und/oder wechselbare Speichereinheiten umfassen können, z.B. Festplattenlaufwerke, wechselbare Speicherkarten, Caches, optische Speicher, NAS-Speicher (Network Attached Storage, NAS) oder Speicherbereichsnetzwerke (Storage Area Networks, SAN).

[0064] In einigen Ausführungsformen sind Ein- und Ausgabeeinheiten (z.B. Tastaturen, Monitore usw.) über die E/A-Schnittstelle(n) 530 angeschlossen. Weiterhin kann der Abfrageverarbeitungsorchestrator 115 über die Netzwerkschnittstelle 525 mit einer oder mehreren anderen Einheiten und Komponenten zum Datenaustausch verbunden sein (z.B. über das Netzwerk 580, das das Internet, ein oder mehrere lokale Netzwerke und Ähnliches umfassen kann). Wie dargestellt, sind der Prozessor 510, der Arbeitsspeicher 515, der Speicher 520, die Netzwerkschnittstelle(n) 525 und die E/A-Schnittstelle(n) 530 zum Datenaustausch über einen oder mehrere Busse 575 verbunden. Außerdem ist der Abfrageverarbeitungsorchestrator 115 über das Netzwerk 580 mit einer Anzahl von Datenknoten 130A bis N zum Datenaustausch verbunden. In Ausführungsformen können die Datenknoten 130A bis N natürlich direkt mit dem Abfrageverarbeitungsorchestrator 115 verbunden sein, auf sie kann über ein lokales Netzwerk zugegriffen werden, sie können in den Abfrageverarbeitungsorchestrator 115 integriert sein und Ähnliches. Der Abfrageverarbeitungsorchestrator 115 ist

zwar in der dargestellten Ausführungsform nicht enthalten, in einigen Ausführungsformen ist er jedoch weiterhin mit dem Datenplatzierungsorchestrator 150 zum Datenaustausch verbunden.

[0065] In der dargestellten Ausführungsform umfasst der Speicher 520 die Ontologie 560, die Fähigkeitsdaten 565 und die Datenzuordnungen 570. Die Ontologie 560, die Fähigkeitsdaten 565 und die Datenzuordnungen 570 sind in Ausführungsformen zwar als im Speicher 520 befindlich dargestellt, sie können jedoch an jedem geeigneten Speicherort und auf jede geeignete Weise gespeichert werden. In einer Ausführungsform gibt die Ontologie 560 wie oben beschrieben die Entitäten oder Konzepte an, die für die Domäne, in der der Abfrageverarbeitungsorchestrator 115 arbeitet, relevant sind, sowie die potenziellen Eigenschaften jedes Konzepts/jeder Entität und die Beziehungen zwischen und unter den Entitäten/Konzepten. In mindestens einer Ausführungsform enthält die Ontologie 560 keine Daten auf Instanzebene. Stattdessen werden die in der KB enthaltenen Daten in den Datenknoten 130A bis N gespeichert.

[0066] In einer Ausführungsform geben die Fähigkeitsdaten 565 wie oben beschrieben die Fähigkeiten der einzelnen Datenknoten 130A bis N an. Dies kann z.B. eine Angabe darüber umfassen, welche Operation(en) jeder Datenknoten 130A bis N unterstützt, sowie eine oder mehrere damit einhergehende Einschränkungen dieser Unterstützung. Die Fähigkeitsdaten 565 können zum Beispiel angeben, dass der Datenknoten 130A Verknüpfungsoperationen („join“) unterstützt, allerdings nur für ganzzahlige Datentypen. In Ausführungsformen geben die Datenzuordnungen 570 wie oben beschrieben den/die Datenknoten 130 an, in dem/denen Instanzdaten für jede Entität/jedes Konzept gespeichert sind, die/das in der Ontologie 560 definiert ist. Die Datenzuordnungen 570 können beispielsweise angeben, dass alle Instanzen des Konzepts „Company“ in den Datenknoten 130A und 130B gespeichert sind, während alle Instanzen eines Konzepts „Document“ in den Datenknoten 130B und 130N gespeichert sind.

[0067] In der dargestellten Ausführungsform umfasst der Arbeitsspeicher 515 eine Abfrageanwendung 535. Die Abfrageanwendung 535 ist zwar in Ausführungsformen als Software dargestellt, die sich im Arbeitsspeicher 515 befindet, sie kann jedoch mit Hardware, Software oder einer Kombination aus Hardware und Software implementiert werden. Wie dargestellt, umfasst die Abfrageanwendung 535 eine Parsingkomponente 540, eine Weiterleitungskomponente 545 und einen Satz von Übersetzern 120. Obwohl sie aus Gründen der Klarheit des Konzepts in Ausführungsformen als einzelne Komponenten dargestellt sind, können die Operationen der Parsingkomponente 540, der

Weiterleitungskomponente 545 und des (der) Übersetzer(s) 120 kombiniert oder in einer beliebigen Anzahl von Komponenten verteilt werden.

[0068] In einer Ausführungsform empfängt die Parsingkomponente 540 OQL-Abfragen und parst sie, um ihre Bedeutung zu ermitteln und eine logische Darstellung der Abfrage (z.B. ein QGM) zu erzeugen. Wie oben beschrieben, umfasst die logische Darstellung in einigen Ausführungsformen einen Satz von Abfrageblöcken, wobei jeder Abfrageblock ein oder mehrere Prädikate spezifiziert, die die an den Eingabequantoren des Blocks durchzuführenden Operationen definieren. In Ausführungsformen können diese Quantoren ein oder mehrere Grundkonzepte sein, die in einem oder mehreren Datenknoten 130 gespeichert sind, und/oder Daten, die von einem anderen Abfrageblock berechnet werden. In einer Ausführungsform ermöglicht diese logische Darstellung der Abfrageanwendung 535, die Struktur der Abfrage und den Datenfluss zwischen den Blöcken zu verstehen und zu interpretieren. Dies ermöglicht der Weiterleitungskomponente 545, die Abfrage (oder Teilabfragen davon) effizient weiterzuleiten.

[0069] In der dargestellten Ausführungsform empfängt die Weiterleitungskomponente 545 die logische Darstellung (z.B. das Abfragegraphmodell) von der Parsingkomponente 540 und wertet sie aus, um einen oder mehrere Datenknoten 130 zu identifizieren, an die die Abfrage weitergeleitet werden soll. In einigen Ausführungsformen umfasst dies Analysieren der Prädikate, die in jedem Abfrageblock in der logischen Darstellung enthalten sind. In einer Ausführungsform kennzeichnet oder annotiert die Weiterleitungskomponente 545 jedes Prädikat in einem gegebenen Abfrageblock auf der Grundlage des (der) Datenknoten(s) 130, der (die) das Prädikat ausführen kann (können). In einigen Ausführungsformen umfasst dies den (die) Datenknoten 130, der (die) den (die) relevanten Quantor(en) enthält (enthalten) und/oder in der Lage ist (sind), die angegebene(n) Operation(en) durchzuführen. Weiterhin kann in mindestens einer Ausführungsform die Weiterleitungskomponente 545 nach Verarbeiten jedes Prädikats in einem Block die Annotationen auswerten, um einen Platzierungsort für den Block zu ermitteln. Die Weiterleitungskomponente 545 ermittelt somit, ob der Abfrageblock in einem einzelnen Datenknoten 130 platziert werden kann. Ist dies der Fall, so weist die Weiterleitungskomponente 545 in einer Ausführungsform den Block diesem Speicher zu. Wenn der Block nicht von einem einzelnen Datenknoten 130 verarbeitet werden kann, teilt die Weiterleitungskomponente 545 den Block in einer Ausführungsform weiterhin in zwei oder mehr Abfrageblöcke auf und wiederholt für jeden von ihnen den Weiterleitungsprozess.

[0070] Sobald jeder Abfrageblock einem einzelnen Datenknoten 130 zugewiesen wurde, erzeugt (erzeugen) der (die) Übersetzer 120 in einer Ausführungsform eine oder mehrere entsprechende übersetzte Abfragen. In einer Ausführungsform entspricht jeder Übersetzer 120 einer bestimmten Architektur des Datenknotens 130 und ist so konfiguriert, dass er OQL-Abfragen (oder Teilabfragen) in eine geeignete Abfrage für die entsprechende Architektur des Datenspeichers übersetzt. In einer solchen Ausführungsform kann das System z.B. einen ersten Übersetzer 120 umfassen, um die OQL in eine Abfrage für einen relationalen Speicher (z.B. SQL) zu übersetzen, einen zweiten Übersetzer 120, um die OQL in eine JSON-Abfrage zu übersetzen, einen dritten Übersetzer 120, um in eine Abfrage für eine Dokumentensuche zu übersetzen, und einen vierten Übersetzer 120, um die OQL in eine Graphabfrage zu übersetzen.

[0071] In Ausführungsformen wird die Abfrage (oder werden die Teilabfragen) auf der Grundlage des Datenknotens 130, der die Abfrage oder Teilabfrage ausführt, an den entsprechenden Übersetzer 120 weitergeleitet. In einigen Ausführungsformen wird dann jede Abfrage (oder Teilabfrage) direkt an diesen Datenknoten 130 übertragen (z.B. über eine Anwendungsprogrammierschnittstelle oder API). In einigen Ausführungsformen kann die Abfrageanwendung 535 den Datenfluss auf der Grundlage der erzeugten logischen Darstellung ermitteln und die Abfragen entsprechend übertragen, wenn für das Ausführen der Abfrage Daten zwischen den Datenknoten 130 versetzt werden müssen (z.B., um Daten aus zwei oder mehr Speichern zu verknüpfen). Die Abfrageanwendung 535 ermittelt somit anhand des QGM, welche(r) Speicher Daten von anderen Speichern empfangen muss (müssen), und überträgt die erforderlichen Abfragen an diese verknüpfenden Speicher. Wenn beispielsweise ein Datenknoten 130A Daten vom Datenknoten 130N empfangen und eine oder mehrere Operationen damit ausführen soll, kann die Abfrageanwendung 535 eine erste Abfrage an den Datenknoten 130A senden, um Daten von diesem Knoten abzurufen, sowie eine zweite Abfrage, die für den Datenknoten 130N konfiguriert ist. Der Datenknoten 130A kann dann mit der bereitgestellten Abfrage selbst den Datenknoten 130N abfragen.

[0072] Fig. 6 ist ein Ablaufplan, der gemäß einer hier offenbarten Ausführungsform ein Verfahren 600 zum Verarbeiten und Weiterleiten von ontologischen Abfragen veranschaulicht. Das Verfahren 600 beginnt in Block 605, wo ein Abfrageverarbeitungsorchestrator 115 eine ontologische Abfrage empfängt, die in einer Wissensdatenbank ausgeführt werden soll. In einer Ausführungsform wird die ontologische Abfrage relativ zu einer vordefinierten Domänenontologie ausgedrückt, wobei die Ontolo-

gie die Konzepte, Eigenschaften und Beziehungen spezifiziert, die für die Domäne relevant sind. In Block 610 parst der Abfrageverarbeitungsorchestrator 115 die empfangene Abfrage und erzeugt daraus eine logische Darstellung. In einigen Ausführungsformen erzeugt der Abfrageverarbeitungsorchestrator 115 eine Abfragegraph-Modusdarstellung der Abfrage. In Ausführungsformen umfasst die logische Darstellung einen oder mehrere Abfrageblöcke, die das (die) Konzept(e), die der Abfrage zugrunde liegt (liegen), die bei den Daten durchzuführende(n) Operation(en) sowie den Datenfluss bezeichnen.

[0073] Das Verfahren 600 fährt dann mit Block 615 fort, in dem der Abfrageverarbeitungsorchestrator 115 jeden Abfrageblock in der logischen Darstellung auf der Grundlage einer vordefinierten Konzeptzuordnung und/oder Datenspeicherfähigkeiten einem oder mehreren Datenspeichern zuordnet. In einer Ausführungsform ermittelt der Abfrageverarbeitungsorchestrator 115 beispielsweise für jeden Abfrageblock, welche(s) Konzept(e) relevant ist (sind) (z.B. auf welche Daten der Abfrageblock zugreift). Anhand der Konzeptzuordnungen kann der Abfrageverarbeitungsorchestrator 115 dann den (die) Datenspeicher identifizieren, der (die) in der Lage ist (sind), die benötigten Daten bereitzustellen. In einer Ausführungsform ermittelt der Abfrageverarbeitungsorchestrator 115 außerdem für jeden Abfrageblock, welche Operation(en) erforderlich ist (sind). Mithilfe der vordefinierten Speicherfähigkeiten kann der Abfrageverarbeitungsorchestrator 115 dann identifizieren, welche(r) Datenknoten in der Lage ist (sind), die erforderliche(n) Operation(en) durchzuführen. Der Abfrageverarbeitungsorchestrator 115 ordnet dann den Abfrageblock bzw. die Abfrageblöcke den (dem) Datenknoten zu, wobei er darauf achtet, das Versetzen von Daten zwischen den Speichern zu minimieren.

[0074] In Block 620 wählt der Abfrageverarbeitungsorchestrator 115 einen der zugeordneten Abfrageblöcke aus. Das Verfahren 600 geht dann weiter zu Block 625, wo der Abfrageverarbeitungsorchestrator 115 eine entsprechende übersetzte Abfrage für den Abfrageblock auf der Grundlage des zugeordneten Datenspeichers erzeugt. In einer Ausführungsform umfasst dies Identifizieren oder Erzeugen einer Teilabfrage aus der empfangenen Abfrage, um den ausgewählten Abfrageblock auszuführen. Der Abfrageverarbeitungsorchestrator 115 ermittelt dann die Konfiguration des Datenknotens, der den Abfrageblock ausführt. In einer Ausführungsform ermittelt der Abfrageverarbeitungsorchestrator 115 somit die Sprache und/oder das Format der Abfrage, für deren Verarbeitung der Knoten konfiguriert ist. In einer anderen Ausführungsform identifiziert der Abfrageverarbeitungsorchestrator 115 den Übersetzer, der dem zugeordneten Datenspeicher entspricht. Der Abfrageverarbeitungsorchestrator 115 verwendet

dann diesen Übersetzer, um eine entsprechende Abfrage für den Speicher zu erzeugen.

[0075] Das Verfahren 600 geht dann weiter zu Block 630, wo der Abfrageverarbeitungsorchestrator 115 ermittelt, ob es noch mindestens einen zusätzlichen Abfrageblock gibt, der noch nicht verarbeitet wurde. Ist dies der Fall, kehrt das Verfahren 600 zu Block 620 zurück. Andernfalls geht das Verfahren 600 weiter zu Block 635, wo der Abfrageverarbeitungsorchestrator 115 die übersetzte(n) Abfrage(n) an einen oder mehrere Datenknoten überträgt. In einer Ausführungsform überträgt der Abfrageverarbeitungsorchestrator 115 jede Abfrage an den entsprechenden Knoten. In einer anderen Ausführungsform überträgt der Abfrageverarbeitungsorchestrator 115 die Abfragen auf der Grundlage des Datenflusses in der logischen Darstellung. Beispielsweise kann der Abfrageverarbeitungsorchestrator 115 mehrere Abfragen an einen einzelnen Knoten übertragen, sodass der Knoten die Abfragen an den (die) entsprechenden Speicher weiterleiten kann, um die Daten abzurufen. Der Knoten kann dann als Mediator fungieren, um die Ergebnisse zu vervollständigen.

[0076] Durch Verwenden eines der Datenknoten als Mediator kann der Abfrageverarbeitungsorchestrator 115 auf vorteilhafte Weise seinen Rechenaufwand verringern und die Skalierbarkeit des Systems erweitern. In Block 640 empfängt der Abfrageverarbeitungsorchestrator 115 die vollständigen Abfrageergebnisse von dem als Mediator fungierenden Datenspeicher (oder von dem einzelnen Datenspeicher, an den Abfragen gesendet wurden, wenn die empfangene Abfrage in einer einzigen Backend-Ressource ausgeführt werden kann). Der Abfrageverarbeitungsorchestrator 115 sendet dann die Ergebnisse an die anfordernde Entität zurück.

[0077] Fig. 7 ist ein Ablaufplan, der gemäß einer hier offenbarten Ausführungsform ein Verfahren 700 zum Verarbeiten von ontologischen Abfragen veranschaulicht, um Abfrageblöcke effizient weiterzuleiten. Das Verfahren 700 stellt in einigen Ausführungsformen zusätzliche Details zum Weiterleiten von Abfrageblöcken bereit. Das Verfahren 700 beginnt in Block 705, wo der Abfrageverarbeitungsorchestrator 115 wie oben beschrieben einen oder mehrere Abfrageblöcke erzeugt, um die empfangene Abfrage darzustellen. In einer Ausführungsform bestimmt jeder Abfrageblock einen oder mehrere Quantoren, der (die) die Eingabedaten für den Block spezifiziert (spezifizieren). Diese Quantoren können Grundkonzepten (z.B. Instanzdaten in der Wissensdatenbank) und/oder vorläufig berechneten Daten (z.B. Daten, die aus einem Speicher abgerufen und in irgendeiner Weise verarbeitet und/oder transformiert wurden) entsprechen. In einer Ausführungsform spezifiziert jeder Abfrageblock zusätzlich

die Operation(en), die an den Quantoren durchgeführt werden soll (sollen).

[0078] In Block 710 wählt der Abfrageverarbeitungsorchestrator 115 einen der erzeugten Abfrageblöcke aus. Weiterhin identifiziert der Abfrageverarbeitungsorchestrator 115 in Block 715 den Satz von Datenknoten, die in der Lage sind, die durch den ausgewählten Block definierte(n) Operation(en) auszuführen. In einer Ausführungsform geschieht dies, indem der Abfrageverarbeitungsorchestrator 115 auf vordefinierte Fähigkeitsdefinitionen zugreift, die den Satz von Operationen, die jeder Datenspeicher ausführen kann, sowie alle entsprechenden Einschränkungen dieser Fähigkeit definieren. Das Verfahren 700 geht dann weiter zu Block 720, wo der Abfrageverarbeitungsorchestrator 115 den Satz von Knoten identifiziert, die in der Lage sind, den (die) im Abfrageblock aufgeführten Quantor(en) auszuführen. Der Abfrageverarbeitungsorchestrator 115 identifiziert somit für jeden Quantor, der einem Grundkonzept entspricht, Knoten, die das Grundkonzept speichern.

[0079] In einer Ausführungsform identifiziert der Abfrageverarbeitungsorchestrator 115 für jeden Quantor, der von einem anderen Abfrageblock berechnet wird, den oder die Datenknoten, der (die) diesem Abfrageblock zugewiesen wurde(n). Es sei daran erinnert, dass der Abfrageverarbeitungsorchestrator 115 in Ausführungsformen Abfrageblöcke Knoten zuordnet, indem er vom unteren Ende des QGM nach oben geht. Wenn also ein erster Block von Daten abhängt, die in einem zweiten Block berechnet wurden, wurde der zweite Block notwendigerweise ausgewertet und hat einen oder mehrere Datenknoten zugewiesen bekommen, bevor der Abfrageverarbeitungsorchestrator 115 mit der Auswertung des ersten Abfrageblocks beginnt.

[0080] Das Verfahren 700 geht dann weiter zu Block 725, wo der Abfrageverarbeitungsorchestrator 115 ermittelt, ob es mindestens einen Datenknoten gibt, der sowohl die Quantoren bereitstellen als auch die Operationen ausführen kann. In einer Ausführungsform umfasst dies Ermitteln, ob es Überschneidungen zwischen den identifizierten Sätzen gibt. Handelt es sich beispielsweise bei den Quantoren durchweg um Grundkonzepte, gibt die Überschneidung zwischen den beiden Sätzen den Satz der Knoten an, die in der Lage sind, alle erforderlichen Operationen auszuführen und alle Konzepte zu speichern. Umfassen die Quantoren Elemente, die von anderen Abfrageblöcken berechnet wurden, gibt der überschneidende Satz Knoten an, die die Operationen durchführen können und die (potenziell) den (die) anderen Block (Blöcke) ausführen.

[0081] In einer Ausführungsform geht das Verfahren 700 im Fall einer Überschneidung zwischen den Sät-

zen weiter zu Block 730, wo der Abfrageverarbeitungsorchestrator 115 den ausgewählten Abfrageblock mit den identifizierten Knoten in der Überschneidung annotiert. Diese Annotation gibt an, dass die spezifizierten Knoten in der Lage sind, den Abfrageblock auszuführen, bedeutet aber kein endgültiges Zuweisen des Blocks zu den (dem) Knoten. Wenn ein Abfrageblock mehrere Knoten in den Annotationen umfasst, führt der Abfrageverarbeitungsorchestrator 115 wie weiter unten näher erläutert in einer Ausführungsform eine Kostenanalyse durch, um jede Alternative zu bewerten und so zu versuchen, die Datenübertragungen zwischen den Speichern auf ein Mindestmaß zu beschränken. Das Verfahren 700 geht dann weiter zu Block 740.

[0082] Wenn es keine Überschneidung zwischen den Sätzen gibt, stellt der Abfrageverarbeitungsorchestrator 115 in der dargestellten Ausführungsform fest, dass es keinen einzigen Datenknoten gibt, der den Abfrageblock ausführen kann. Wenn alle Operationen von einem einzigen Speicher durchgeführt werden können, annotiert der Abfrageverarbeitungsorchestrator 115 in einigen Ausführungsformen den Block mit dem (den) Speicher(n), der (die) die Operationen des Blocks durchführen kann (können). Ein oder mehrere andere Speicher können dann zugewiesen werden, um die Quantoren bereitzustellen. In der dargestellten Ausführungsform geht das Verfahren 700 dann weiter zu Block 735, wo der Abfrageverarbeitungsorchestrator 115 den ausgewählten Abfrageblock in zwei oder mehr Blöcke aufteilt, um Abfrageblöcke zu erzeugen, die vollständig in einem einzigen Knoten ausgeführt werden können. In einer Ausführungsform umfasst Aufteilen des ausgewählten Abfrageblocks Identifizieren eines oder mehrerer Teilsätze des (der) Quantors (Quantoren) und/oder der Operation(en), die von einem einzigen Speicher ausgeführt werden kann (können). So kann der Abfrageverarbeitungsorchestrator 115 beispielsweise Teilsätze der spezifizierten Prädikate identifizieren, die die gleichen Annotationen aufweisen (z.B. Prädikate, die vom selben Speicher ausgeführt werden können). Der Abfrageblock kann dann aufgeteilt werden, indem die Prädikate auf der Grundlage des Teilsatzes, zu dem sie gehören, in entsprechende Felder aufgeteilt werden. Wenn beispielsweise ein Abfrageblock sowohl herkömmliche relationale Datenbankoperationen als auch eine Fuzzymatch-Operation erfordert, kann der Abfrageverarbeitungsorchestrator 115 bestimmen, dass der (die) Fuzzymatch(es) in ein separates Feld gestellt werden soll(en). Dies kann es dem Abfrageverarbeitungsorchestrator 115 ermöglichen, jeden aufgeteilten Block einem einzelnen Datenspeicher zuzuweisen (z.B. können die relationalen Operationen einem relationalen Speicher und die Fuzzymatch-Operationen einem anderen Backend, das sie durchführen kann, zugewiesen werden).

[0083] In der dargestellten Ausführungsform werden diese neu erzeugten Abfrageblöcke in die Warteschlange gestellt, um wie die bestehenden Blöcke ausgewertet zu werden. Das Verfahren 700 geht dann weiter zu Block 740. In Block 740 ermittelt der Abfrageverarbeitungsorchestrator 115, ob es mindestens einen zusätzlichen Abfrageblock gibt, der noch nicht ausgewertet wurde. Ist dies der Fall, kehrt das Verfahren 700 zu Block 710 zurück, um die Blöcke iterativ zu durchlaufen. Der Abfrageverarbeitungsorchestrator 115 durchläuft somit weiterhin iterativ die Abfrageblöcke und teilt die Felder gegebenenfalls auf, bis alle Abfrageblöcke mit mindestens einem Datenknoten annotiert sind.

[0084] Wenn alle Abfrageblöcke mit Annotationen versehen wurden, geht das Verfahren 700 weiter zu Block 745, wo der Abfrageverarbeitungsorchestrator 115 die Abfrage ausführt. In einigen Ausführungsformen umfasst dies Auswerten der alternativen Kombinationen von Zuweisungen, um Datenübertragungen auf ein Mindestmaß zu beschränken, wie weiter unten ausführlicher erläutert wird. Wenn die Annotationen eines oder mehrerer Abfrageblöcke einen einzigen Datenknoten angeben, weist der Abfrageverarbeitungsorchestrator 115 den Block in einigen Ausführungsformen einfach diesem angegebenen Knoten zu, da es keine Alternativen gibt, die in Betracht gezogen werden können.

[0085] Fig. 8 ist ein Ablaufplan, der gemäß einer hier offenbarten Ausführungsform ein Verfahren 800 zum Auswerten potenzieller Abfrageweiterleitungspläne veranschaulicht, um eine ontologische Abfrage effizient weiterzuleiten. In der dargestellten Ausführungsform beginnt das Verfahren 800, nachdem die Abfrageblöcke mit ihren potenziellen Zuweisungen annotiert worden sind (z.B. mit Knoten, die in der Lage sind, den gesamten Block auszuführen und alle benötigten Quantoren bereitzustellen). Das Verfahren 800 beginnt in Block 805, wo der Abfrageverarbeitungsorchestrator 115 eine der möglichen Kombinationen von Knotenzuweisungen auswählt. In einer Ausführungsform erzeugt der Abfrageverarbeitungsorchestrator 115 somit alle möglichen Kombinationen von Speichern für die Blöcke auf der Grundlage der entsprechenden Annotationen. Zu diesem Zweck kann der Abfrageverarbeitungsorchestrator 115 iterativ für jeden Block eine andere Option auswählen, bis alle Auswahlmöglichkeiten erzeugt worden sind.

[0086] Angenommen, ein erster Block ist mit „Knoten A“ und „Knoten B“ annotiert, während ein zweiter Block mit „Knoten A“ annotiert ist. In einer Ausführungsform stellt der Abfrageverarbeitungsorchestrator 115 fest, dass die möglichen Weiterleitungspläne Zuweisen des ersten Blocks und des zweiten Blocks zu „Knoten A“ oder Zuweisen des ersten Blocks zu „Knoten B“ und des zweiten Blocks zu „Knoten A“

umfassen. In Block 805 wählt der Abfrageverarbeitungsorchestrator 115 eine der identifizierten Kombinationen zum Auswerten aus. Das Verfahren 800 geht dann weiter zu Block 810.

[0087] In Block 810 identifiziert der Abfrageverarbeitungsorchestrator 115 die Datenversetzung(en), die im Rahmen des ausgewählten Plans erforderlich ist (sind). In Ausführungsformen werden die Versetzungen auf der Grundlage der Speicherauswahlen im Plan und/oder des Abfragegraphmodells ermittelt. In Fortführung des obigen Beispiels kann der Abfrageverarbeitungsorchestrator 115 feststellen, dass der Plan, der beide Blöcke dem „Knoten A“ zuweist, keine Versetzung erfordert, da beide Abfrageblöcke in einem einzigen Knoten gruppiert werden können. Da die Blöcke im Modell zusammenhängend sind (z.B. direkt verbunden/getrennt durch einen einzigen Zwischenschritt, ohne dass andere Blöcke dazwischenliegen) und demselben Knoten zugewiesen sind, werden sie somit zusammen gruppiert, und es ist kein Versetzen von Daten erforderlich. Im Gegensatz dazu erfordert Zuweisen des ersten Blocks zu „Knoten B“ eine Übertragung von Daten zwischen „Knoten A“ und „Knoten B“, da das QGM angibt, dass Daten vom zweiten Block zum ersten Block fließen und die Blöcke verschiedenen Speichern zugewiesen sind.

[0088] Das Verfahren 800 geht dann weiter zu Block 815, wo der Abfrageverarbeitungsorchestrator 115 eine der identifizierten Datenübertragungen auswählt, die für den ausgewählten Abfrageplan erforderlich sind. In Block 820 ermittelt der Abfrageverarbeitungsorchestrator 115 die Berechnungskosten für die ausgewählte Versetzung. In einer Ausführungsform wird diese Ermittlung auf der Grundlage eines vordefinierten Kostenmodells durchgeführt. In einigen Ausführungsformen kann das Modell für jedes geordnete Datenknotenpaar die Berechnungskosten für das Übertragen von Daten vom ersten zum zweiten Knoten angeben. Die Kosten können z.B. Latenzzeit umfassen, die gegebenenfalls für die eigentliche Übertragung der Daten und/oder die Transformation der Daten erforderlich ist, damit der Zielknoten mit ihnen arbeiten kann, sowie Verarbeitungszeit und/oder den Speicherbedarf für die Übertragung/Transformation und Ähnliches.

[0089] In Block 825 ermittelt der Abfrageverarbeitungsorchestrator 115, ob die ausgewählte Kombination zusätzliche Datenübertragungen erfordert. Ist dies der Fall, kehrt das Verfahren 800 zu Block 815 zurück. Andernfalls geht das Verfahren 800 weiter zu Block 830, wo der Abfrageverarbeitungsorchestrator 115 die Gesamtkosten des ausgewählten Plans berechnet, indem er die einzelnen Kosten für jede Versetzung zusammenrechnet. In Block 835 ermittelt der Abfrageverarbeitungsorchestrator 115, ob es mindestens einen alternativen Plan gibt, der noch

nicht ausgewertet wurde. Ist dies der Fall, kehrt das Verfahren 800 zu Block 805 zurück. Andernfalls geht das Verfahren 800 weiter zu Block 840, wo der Abfrageverarbeitungsorchestrator 115 die Pläne auf der Grundlage ihrer Gesamtkosten in eine Rangfolge bringt. In einer Ausführungsform wählt der Abfrageverarbeitungsorchestrator 115 den Plan aus, der die geringsten ermittelten Kosten aufweist. Der Abfrageverarbeitungsorchestrator 115 führt dann den Plan aus, indem er die Teilabfragen auf der Grundlage der Knotenzuweisungen für den Plan mit den Mindestkosten übersetzt und weiterleitet.

[0090] Fig. 9 ist ein Ablaufplan, der gemäß einer hier offenbarten Ausführungsform ein Verfahren 900 zum Weiterleiten einer ontologischen Abfrage veranschaulicht. Das Verfahren 900 beginnt in Block 905, wo ein Abfrageverarbeitungsorchestrator 115 eine ontologische Abfrage empfängt. In Block 910 erzeugt der Abfrageverarbeitungsorchestrator 115 auf der Grundlage der ontologischen Abfrage einen oder mehrere Abfrageblöcke, von denen jeder eine oder mehrere Operationen und einen oder mehrere Quantoren angibt, die den Datenfluss zwischen Abfrageblöcken darstellen. Das Verfahren 900 geht dann weiter zu Block 915, wo der Abfrageverarbeitungsorchestrator 115 für jeden der einen oder mehrerer Abfrageblöcke auf der Grundlage des einen oder mehrerer Quantoren und der einen oder mehrerer Operationen mindestens einen Datenknoten identifiziert. Weiterhin wählt der Abfrageverarbeitungsorchestrator 115 in Block 920 einen oder mehrere der identifizierten Datenknoten auf der Grundlage vordefinierter Kostenkriterien aus. Zusätzlich überträgt der Abfrageverarbeitungsorchestrator 115 dann in Block 925 eine oder mehrere Teilabfragen an den einen oder mehrere ausgewählte Datenknoten.

[0091] Fig. 10 veranschaulicht einen Arbeitsablauf 1000 zum Auswerten von Arbeitslasten und Speichern von Daten einer Wissensdatenbank gemäß einer hier offenbarten Ausführungsform. Wie bereits beschrieben, benötigen Unternehmensanwendungen oft Unterstützung für verschiedene Abfragetypen, je nach Arbeitslast der Abfrage. Um diese verschiedenen Abfragetypen zu unterstützen, verwenden Ausführungsformen der vorliegenden Offenbarung mehrere Backend-Speicher, z.B. relationale Datenbanken, Dokumentenspeicher, Graphspeicher und Ähnliches. In Ausführungsformen der vorliegenden Offenbarung kann das System die Daten der Wissensdatenbank in jedes beliebige Backend, das die erforderlichen Fähigkeiten für die unterstützten Abfragetypen bereitstellt, versetzen, speichern und indizieren. Mit dieser Flexibilität beim Organisieren der Daten kann die anfängliche Datenplatzierung in mehreren Backend-Speichern eine entscheidende Rolle bei der effizienten Ausführung von Abfragen spielen.

[0092] Um eine effiziente Ausführung während der Laufzeit mit minimalem Replikationsaufwand zu erreichen, stellen einige Ausführungsformen der vorliegenden Offenbarung eine Datenaufbereitungs- und -ladephase bereit, die offline ist und eine intelligente Datenplatzierung umfasst. Im Allgemeinen umfasst Aufnehmen, Platzieren und Laden von Daten in die mehreren Backend-Speicher eine Reihe von Operationen. Zu Beginn werden Daten für die domänenspezifische Wissensdatenbank aus einer Vielfalt unterschiedlicher Quellen aufgenommen, die strukturierte, halbstrukturierte und unstrukturierte Daten umfassen. In einigen Ausführungsformen besteht die erste Phase des Aufnehmens von Daten aus diesen Datenquellen aus einem Datenaufbereitungs-/-zusammenstellungsprozess, der Informationsextraktion, Entitätsauflösung, Datenintegration und -transformation umfasst. Die in diesem Schritt erzeugten Ausgabedaten, die mit der Domänenontologie übereinstimmen, werden dann dem Datenplatzierungsorchestrator 150 zugeführt, um eine geeignete Datenplatzierung für die mehreren Backend-Datenspeicher zu erzeugen. In einigen Ausführungsformen wurden die Daten bereits zusammengestellt und können bereits vor der Verwendung des Datenplatzierungsorchestrators 150 beim Antworten auf Abfragen verwendet worden sein. Sobald eine zufriedenstellende Datenplatzierung ermittelt wurde, platziert ein Datenlademodul die Instanzdaten gemäß dem Datenplatzierungsplan in den entsprechenden Datenspeichern.

[0093] In einigen Ausführungsformen kann das Versetzen von Daten während der Abfrageausführung vermieden werden, indem der gesamte Satz von Daten in allen Datenknoten repliziert wird. Diese Lösung führt jedoch zu einer umfangreichen Replikation und erhöht den Speicherplatzbedarf drastisch. Darüber hinaus stellen in vielen Ausführungsformen nicht alle Speicher alle für die Abfragen erforderlichen Funktionen bereit, und selbst eine vollständige Replikation kann ein Versetzen von Daten nicht vollständig verhindern. Um unnötige Speicherkosten und Datenversetzungen auf ein Mindestmaß zu beschränken, stellen einige Ausführungsformen der vorliegenden Offenbarung fähigkeitsgestützte Datenplatzierungstechniken bereit, die Daten Datenspeichern zuweisen und dabei sowohl die erwartete Arbeitslast als auch die Fähigkeiten der Backend-Datenspeicher (z.B. in Bezug auf die Operationen, die sie an den gespeicherten Daten durchführen können) berücksichtigen.

[0094] In Ausführungsformen entscheidet der Datenplatzierungsorchestrator 150 über die Datenplatzierung auf der Ebene von Abfrageoperationen auf der Grundlage der Konzepte der Domänenontologie, die das Schema der in der Wissensdatenbank gespeicherten Daten darstellt. In einigen Ausführungsformen identifiziert der Orchestrator verschie-

dene und sich möglicherweise überschneidende Teilsätze der Ontologie auf der Grundlage einer gegebenen Arbeitslast im Zusammenhang mit der Wissensdatenbank sowie der Fähigkeiten der zugrunde liegenden Speicher und gibt eine Zuordnung zwischen den identifizierten Teilsätzen der Daten und den Zieldatenspeichern aus, in denen die Daten gespeichert werden sollen.

[0095] In der dargestellten Ausführungsform beginnt der Arbeitsablauf 1000, wenn ein Satz von OQL-Abfragen 1005 empfangen wird. In einigen Ausführungsformen handelt es sich bei den OQL-Abfragen 1005 um zuvor übertragene Abfragen für die Wissensdatenbank. In einer Ausführungsform ist die Wissensdatenbank beispielsweise ein bereits vorhandener Datenbestand, den Benutzer und Anwendungen abfragen und untersuchen können. In einer solchen Ausführungsform können die OQL-Abfragen 1005 den Abfragen entsprechen, die Benutzer, Anwendungen und andere Entitäten zuvor beim Interagieren mit der Wissensdatenbank übermittelt haben.

[0096] In Ausführungsformen stellen die OQL-Abfragen 1005 im Allgemeinen die durchschnittliche, typische, erwartete und/oder historische Arbeitslast der Wissensdatenbank dar. Anders ausgedrückt sind die OQL-Abfragen 1005 im Allgemeinen repräsentativ für die Abfragen, die die Wissensdatenbank während der Laufzeitoperationen empfängt (oder von denen erwartet wird, dass sie sie empfängt), und können dazu verwendet werden, die Sätze von Konzepten zu identifizieren, die häufig zusammen abgefragt werden, sowie Operationen, die in der Regel mit jedem Konzept durchgeführt werden, und Ähnliches. Wie dargestellt, werden diese repräsentativen OQL-Abfragen 1005 dem OQL-Abfrageanalysator 1010 bereitgestellt, der sie analysiert und auswertet, um eine Gesamtarbeitslast 1015 zu erzeugen.

[0097] In einer Ausführungsform drückt der OQL-Abfrageanalysator 1010 die OQL-Abfragen 1005 als einen Satz von Konzepten und Beziehungen mit den entsprechenden Operationen aus, die bei ihnen ausgeführt werden. Auf dieser Grundlage erzeugt der Analysator die Gesamtarbeitslast 1015. Anders ausgedrückt spiegelt die Gesamtarbeitslast 1015 eine Zusammenfassung der bereitgestellten OQL-Abfragen 1005 wider, die eine eingehendere Analyse ermöglicht, um Muster in den Daten zu identifizieren. In einer Ausführungsform umfasst die Gesamtarbeitslast 1015 einen Satz von Gesamtabfragen, die auf der Grundlage der OQL-Abfragen 1005 erzeugt wurden. In einigen Ausführungsformen handelt es sich bei den Gesamtabfragen insbesondere nicht um vollständige formatierte Abfragen, die in der Wissensdatenbank ausgeführt werden könnten. Stattdessen geben die Gesamtabfragen Cluster von Kon-

zepten an, die während der Laufzeit wahrscheinlich zusammen abgefragt werden, zusammen mit Sätzen von Operationen, die wahrscheinlich auf jeden Cluster von Konzepten angewendet werden.

[0098] In einigen Ausführungsformen nimmt der OQL-Abfrageanalysator 1010 den Satz von OQL-Abfragen 1005, die in Bezug auf die Domänenontologie ausgedrückt werden, als Eingabe und erzeugt für jede Abfrage zwei Sätze. In einer solchen Ausführungsform ist der erste Satz ein Satz von Konzepten, auf den die Abfrage zugreift, während der zweite Satz ein Satz von Operationen (z.B. Verknüpfung, Aggregation usw.) ist, den die Abfrage bei diesen Konzepten durchführt. Um die Darstellung der Gesamtarbeitslast 1015 der gegebenen OQL-Abfragen 1005 zu erzeugen, gruppiert der OQL-Abfrageanalysator 1010 in mindestens einer Ausführungsform die Abfragen, die auf denselben Satz von Konzepten in einer Gruppe zugreifen und erstellt dann einen Satz, der die zugehörigen Operationen jeder Abfrage in der Gruppe kombiniert. Im Folgenden wird dies noch ausführlicher beschrieben.

[0099] In der dargestellten Ausführungsform wird die Gesamtarbeitslast 1015 dann einem Hypergraph-Modellierer 1020 bereitgestellt, der die bereitgestellten Gesamtabfragen auswertet, um einen Hypergraphen 1025 zu erzeugen. In einer Ausführungsform ist der Hypergraph 1025 ein Graph, der einen Satz von Eckpunkten und einen Satz von Kanten (auch als Hyperkanten bezeichnet) umfasst, wobei jede Kante mit einer beliebigen Anzahl von Eckpunkten verbunden sein kann. In einigen Ausführungsformen entspricht jeder Eckpunkt im Hypergraphen 1025 einem Konzept aus der Domänenontologie, und jede Hyperkante entspricht einer Gesamtabfrage aus der Gesamtarbeitslast 1015. Zum Beispiel erstreckt sich jede Hyperkante über den Satz von Konzepten, der durch die entsprechende Gesamtabfrage angegeben wird. In einer Ausführungsform wird jede Hyperkante weiterhin mit einer Angabe des Satzes von Operationen annotiert oder gekennzeichnet, die durch die entsprechende Gesamtabfrage angegeben werden.

[0100] Wie dargestellt, wird der Hypergraph 1025 zusammen mit dem Ontologieschema 110 und den Knotenfähigkeiten 155 von der Datenplatzierungskomponente 1030 ausgewertet. In einer Ausführungsform gruppiert die Datenplatzierungskomponente 1030 die Konzepte und Beziehungen im Hypergraphen 1025 auf der Grundlage der Abfrageoperationen in sich potenziell überschneidende Teilsätze. Die diesen Teilsätzen entsprechenden Daten können dann auf der Grundlage der von ihnen unterstützten Operationen auf einzelnen Backend-Datenknoten platziert werden. In einigen Ausführungsformen werden die Entscheidungen über die Datenplatzierung nach der Granularität der identifi-

zierten Ontologieteilsätze getroffen, wobei alle Daten eines beliebigen Ontologiekonzepts in ihrer Gesamtheit platziert werden. Anders ausgedrückt führen die Platzierungsentscheidungen 165 nicht zu einer horizontalen Aufteilung der Konzepte in verschiedenen Speichern. Wenn die Datenplatzierungskomponente 1030 beispielsweise das Konzept „company“ in einem ersten Datenknoten platziert, werden alle Daten auf Instanzebene, die dem Konzept „company“ entsprechen, in dem ersten Datenknoten platziert.

[0101] In einigen Ausführungsformen verfolgt die ontologiegestützte Datenplatzierungskomponente 1030 einen zweistufigen Ansatz für die Datenplatzierung. Zunächst führt die Datenplatzierungskomponente 1030 Algorithmen zur Graphanalyse im Hypergraphen 1025 aus, der die Gesamtarbeitslast 1015 darstellt, um die Konzepte im Domänenontologieschema 110 auf der Grundlage der Ähnlichkeit der Operationen, die bei diesen Konzepten durchgeführt werden, zu gruppieren. Anschließend werden die Daten, die diesen identifizierten Gruppen oder Teilsätzen der Ontologie entsprechen, auf der Grundlage ihrer jeweiligen Fähigkeiten den zugrunde liegenden Datenspeichern zugeordnet, wobei der Umfang der erforderlichen Replikation auf ein Mindestmaß beschränkt wird. In Ausführungsformen verringert die sich ergebende fähigkeitsgestützte Datenplatzierung die Datenversetzung (und die Datentransformation) für eine gegebene Arbeitslast zum Zeitpunkt der Abfrageverarbeitung auf ein Mindestmaß, wodurch die Effizienz der Abfrageverarbeitung in einer Mehrfachspeicherumgebung erheblich gesteigert wird. Wie dargestellt, ist die endgültige Ausgabe der Datenplatzierungskomponente 1030 eine Zuordnung von Konzepten zu Speichern (z.B. Platzierungsentscheidungen 165), die die Ontologiekonzepte den entsprechenden Datenknoten zuordnet.

[0102] Zwar wird dies im dargestellten Arbeitsablauf 1000 nicht gezeigt, das System verwendet aber in Ausführungsformen die Platzierungsentscheidungen 165, um die Daten dann in den verschiedenen Backend-Ressourcen zu speichern. In einer Ausführungsform erfolgt dies, indem das System einen Extraktions-, Transformations- und Ladedienst (ETL) aufruft, der alle Transformationen oder Konvertierungen durchführt, die erforderlich sind, damit die Daten in dem/den entsprechenden Datenknoten 130 gespeichert werden können.

[0103] In einigen Ausführungsformen wird der Arbeitsablauf 1000 dazu verwendet, die Datenplatzierung regelmäßig neu zu bewerten und zu verfeinern, um die Effizienz des Systems zu erhalten. Zum Beispiel kann das System in Zeiten geringer Systemauslastung (z.B. außerhalb der Geschäftszeiten) den Arbeitsablauf 1000 auf der Grundlage eines aktuali-

sierten Satzes von OQL-Abfragen 1005 (z.B. einschließlich der nach der letzten Entscheidung über die Datenplatzierung empfangenen Abfragen) aufrufen, um zu ermitteln, ob die Platzierungen aktualisiert werden sollen, um die sich entwickelnde Arbeitslast zu berücksichtigen. Dadurch kann die Effizienz des Systems verbessert werden, da verhindert wird, dass Datenstandorte nicht mehr aktuell sind.

[0104] Fig. 11 zeigt einen beispielhaften Hypergraphen 1100, der gemäß einer hier offenbarten Ausführungsform zum Auswerten von Arbeitslasten und Speichern von Daten einer Wissensdatenbank verwendet wird. In der dargestellten Ausführungsform ist jedes Konzept 1105A bis G als Ellipse und jede Hyperkante 1110A bis D als gestrichelte Linie dargestellt, die die Konzepte 1105, denen sie entspricht, einkreist. Zum Beispiel verknüpft die Hyperkante 1110A die Konzepte 1105A („Company“), 1105B („PublicCompany“), 1105C („PublicMetric“) und 1105D („PublicMetricData“). Wie dargestellt, kann jede Hyperkante 1110 sowohl unzusammenhängende Teilsätze der Konzepte 1105 (z.B. überschneiden sich die Hyperkanten 1110A und 1110C nicht) als auch sich überschneidende Teilsätze (z.B. überschneiden sich die Hyperkanten 1110A und 1110B in Bezug auf das Konzept 1105A „Company“) umfassen.

[0105] Außerdem ist jede Hyperkante 1110 wie dargestellt mit den entsprechenden Operationen 1115A bis C für die Kante gekennzeichnet. Wie oben beschrieben, geben die Kennzeichnungen in einer Ausführungsform den Satz von Operationen an, die auf die durch die Hyperkante 1110 verbundenen Konzepte 1105 angewendet werden können oder angewendet wurden. Im dargestellten Beispiel ist die Hyperkante 1110A beispielsweise den Operationen 1115A („Join“), 1115B („Aggregation“) und 1115C („Fuzzy“-Matching) zugehörig. Insbesondere kann jede Operation 1115 einer beliebigen Anzahl von Hyperkanten 1110 und/oder Konzepten 1105 zugehörig sein. In der dargestellten Ausführungsform ist die Operation „Join“ 1115A den Hyperkanten 1110A und 1110B zugehörig, die Operation „Aggregation“ 1115B ist den Hyperkanten 1110A und 1110D zugehörig und die Operation „Fuzzy“ 1115C ist den Hyperkanten 1110A, 1110D und 1110C zugehörig.

[0106] Fig. 12 ist ein Blockschaubild, das gemäß einer hier offenbarten Ausführungsform einen Datenplatzierungsorchestrator 150 veranschaulicht, der so konfiguriert ist, dass er Arbeitslasten auswertet und Daten speichert. Der Datenplatzierungsorchestrator 150 ist in Ausführungsformen zwar als physische Einheit dargestellt, er kann jedoch unter Verwendung virtueller Einheit(en) und/oder in einer Anzahl von Einheiten (z.B. in einer Cloud-Umgebung) implementiert werden. Wie dargestellt, umfasst der Datenplat-

zierungsorchestrator 150 einen Prozessor 1210, einen Arbeitsspeicher 1215, einen Speicher 1220, eine Netzwerkschnittstelle 1225 und eine oder mehrere E/A-Schnittstellen 1230. In der dargestellten Ausführungsform ruft der Prozessor 1210 die im Arbeitsspeicher 1215 gespeicherten Programmweisungen ab und führt sie aus, und er speichert die im Speicher 1220 befindlichen Anwendungsdaten und ruft diese ab. Der Prozessor 1210 steht im Allgemeinen für eine einzelne CPU und/oder GPU, mehrere CPUs und/oder GPUs, eine einzelne CPU und/oder GPU mit mehreren Verarbeitungskernen und Ähnliches. Der Arbeitsspeicher 1215 ist allgemein stellvertretend für einen Direktzugriffsspeicher enthalten. Beim Speicher 1220 kann es sich um eine beliebige Kombination von Festplattenlaufwerken, flashgestützten Speichereinheiten und Ähnliches handeln, die feste und/oder wechselbare Speichereinheiten umfassen können, z.B. Festplattenlaufwerke, wechselbare Speicherkarten, Caches, optische Speicher, NAS-Speicher (Network Attached Storage, NAS) oder Speicherbereichsnetzwerke (Storage Area Networks, SAN).

[0107] In einigen Ausführungsformen sind Ein- und Ausgabeeinheiten (z.B. Tastaturen, Monitore usw.) über die E/A-Schnittstelle(n) 1230 angeschlossen. Weiterhin kann der Datenplatzierungsorchestrator 150 über die Netzwerkschnittstelle 1225 mit einer oder mehreren anderen Einheiten und Komponenten zum Datenaustausch verbunden sein (z.B. über das Netzwerk 1280, das das Internet, ein oder mehrere lokale Netzwerke und Ähnliches umfassen kann). Wie dargestellt, sind der Prozessor 1210, der Arbeitsspeicher 1215, der Speicher 1220, die Netzwerkschnittstelle(n) 1225 und die E/A-Schnittstelle(n) 1230 zum Datenaustausch über einen oder mehrere Busse 1275 verbunden. Außerdem ist der Datenplatzierungsorchestrator 150 über das Netzwerk 1280 mit den Datenknoten 130A bis N zum Datenaustausch verbunden. In Ausführungsformen können die Datenknoten 130A bis N natürlich direkt mit dem Datenplatzierungsorchestrator 150 verbunden sein, über ein lokales Netzwerk zugänglich sein, in den Datenverarbeitungsorchestrator 150 integriert sein und Ähnliches. Der Datenplatzierungsorchestrator 150 ist zwar in der dargestellten Ausführungsform nicht enthalten, in einigen Ausführungsformen ist er jedoch weiterhin mit dem Abfrageverarbeitungsorchestrator 115 zum Datenaustausch verbunden.

[0108] In der dargestellten Ausführungsform umfasst der Speicher 1220 eine Kopie der Domänentontologie 1260, die Fähigkeitsdaten 1265 und die Datenzuordnungen 1270. Die Ontologie 1260, die Fähigkeitsdaten 1265 und die Datenzuordnungen 1270 sind in Ausführungsformen zwar als im Speicher 1220 befindlich dargestellt, sie können jedoch an jedem geeigneten Speicherort und auf jede geeignete Weise gespeichert werden. In einer Ausführungsform entsprechen die Ontologie 1260, die Fähigkeitsdaten 1265 und die Datenzuordnungen 1270 der Ontologie 560, den Fähigkeitsdaten 565 und den Datenzuordnungen 570, die oben unter Bezugnahme auf den Abfrageverarbeitungsorchestrator 115 beschrieben wurden.

Wie bereits beschrieben, kann die Ontologie 1260 beispielsweise die Entitäten oder Konzepte angeben, die für die Domäne relevant sind, sowie die potenziellen Eigenschaften jedes Konzepts/jeder Entität und die Beziehungen zwischen den Entitäten/Konzepten, ohne Daten auf Instanzebene zu umfassen. In ähnlicher Weise geben die Fähigkeitsdaten 1265 wie oben beschrieben die Fähigkeiten der einzelnen Datenknoten 130A bis N an. Dies kann z.B. eine Angabe darüber umfassen, welche Operation(en) jeder der Datenknoten 130A bis N unterstützt, sowie eine oder mehrere damit einhergehende Einschränkungen dieser Unterstützung.

[0109] In Ausführungsformen werden die Datenzuordnungen 1270 von der Platzierungsanwendung 1235 erzeugt; sie geben den (die) Datenknoten 130 an, in dem (denen) Instanzdaten für jede Entität/jedes Konzept gespeichert ist (sind), die/das in der Ontologie 1260 definiert ist. Die Datenzuordnungen 1270 können beispielsweise angeben, dass alle Instanzen des Konzepts „Company“ in den Datenknoten 130A und 130B gespeichert sind, während alle Instanzen eines Konzepts „Document“ in den Datenknoten 130B und 130N gespeichert sind.

[0110] In der dargestellten Ausführungsform umfasst der Arbeitsspeicher 1215 eine Platzierungsanwendung 1235. Die Platzierungsanwendung 1235 ist zwar in Ausführungsformen als Software dargestellt, die sich im Arbeitsspeicher 1215 befindet, sie kann jedoch mit Hardware, Software oder einer Kombination aus Hardware und Software implementiert werden. Wie dargestellt, umfasst die Platzierungsanwendung 1235 eine Zusammenfassungskomponente 1240, eine Hypergraphkomponente 1245 und eine Platzierungskomponente 1250. Obwohl sie aus Gründen der Klarheit des Konzepts in Ausführungsformen als einzelne Komponenten dargestellt sind, können die Operationen der Zusammenfassungskomponente 1240, der Hypergraphkomponente 1245 und der Platzierungskomponente 1250 kombiniert oder in einer beliebigen Anzahl von Komponenten verteilt werden.

[0111] In einer Ausführungsform empfängt die Zusammenfassungskomponente 1240 frühere OQL-Abfragen (oder Beispielabfragen), die die vorherige, aktuelle, erwartete, typische, durchschnittliche, voraussichtliche und/oder allgemeine Arbeitslast des Systems widerspiegeln. Die Zusammenfassungskomponente 1240 wertet dann die bereitgestellten Abfragen aus, um die Arbeitslast

in Form von Gesamtabfragen zusammenzufassen. In Ausführungsformen geben die Gesamtabfragen im Allgemeinen Sätze von Konzepten an, die zusammen mit den bei den Konzepten durchgeführten Operationen abgefragt werden, entsprechen aber nicht einer tatsächlichen Abfrage, die ausgeführt werden könnte. Die Funktionalität der Zusammenfassungskomponente 1240 wird im Folgenden unter Bezugnahme auf **Fig. 13** ausführlicher beschrieben.

[0113] In Ausführungsformen empfängt die Hypergraphkomponente 1245 die Informationen zur Gesamtarbeitslast von der Zusammenfassungskomponente 1240 und parst sie, um einen Hypergraphen zu erzeugen, der die Abfragearbeitslast darstellt. Wie oben beschrieben, umfasst der Hypergraph in einigen Ausführungsformen einen Eckpunkt für jedes Konzept in der Ontologie 1260. Zusätzlich sind in Ausführungsformen die Eckpunkte der Konzepte durch Hyperkanten verbunden, die die Gesamtabfragearbeitslast darstellen. Die Operationen der Hypergraphkomponente 1245 werden im Folgenden mit Bezug auf **Fig. 14** ausführlicher beschrieben.

[0114] In einer Ausführungsform wertet die Platzierungskomponente 1250 den Hypergraphen aus, um Datenplatzierungsentscheidungen zu erzeugen (z.B. die Datenzuordnungen 1270). Diese Entscheidungen können dann zum Weiterleiten von Daten an die entsprechenden Knoten verwendet werden. In einer Ausführungsform durchläuft das System zum Beispiel iterativ die Wissensdatenbank, um die Daten zu platzieren. Für jedes Datenelement kann das System das (die) Konzept(e) ermitteln, dem (denen) es entspricht, den (die) entsprechenden Datenknoten 130 suchen und die Daten in dem (den) angegebenen Knoten speichern. In einigen Ausführungsformen führt das System auch alle Transformationen oder Umwandlungen durch, die für den Zielspeicher geeignet sind. Die Funktionalität der Platzierungskomponente 1250 wird im Folgenden unter Bezugnahme auf die **Fig. 15** und **Fig. 16** ausführlicher beschrieben.

[0115] **Fig. 13** ist ein Ablaufplan, der gemäß einer hier offenbarten Ausführungsform ein Verfahren 1300 zum Auswerten und Zusammenfassen der Arbeitslast von Abfragen veranschaulicht, um Entscheidungen über die Datenplatzierung zu unterstützen. Das Verfahren 1300 beginnt in Block 1305, wo ein Datenplatzierungsorchestrator 150 eine oder mehrere Abfragen empfängt, die die Arbeitslast der Wissensdatenbank darstellen. In Block 1310 wählt der Datenplatzierungsorchestrator 150 einen der zugeordneten Abfrageblöcke zum Auswerten aus. Das Verfahren 1300 geht dann weiter zu Block 1315, wo der Datenplatzierungsorchestrator 150 das (die) Konzept(e) identifiziert, das (die) von der ausgewählten Abfrage betroffen sind. In ähnlicher Weise identifiziert der Datenplatzierungsorchestrator

150 in Block 1320 die Operation(en), die durch die Abfrage aufgerufen wird (werden).

[0116] In einer Ausführungsform ordnet der Datenplatzierungsorchestrator 150 den ermittelten Satz von Konzepten, die durch die Abfrage spezifiziert wurden, dem Satz von Operationen zu, die durch die Abfrage aufgerufen wurden. In einigen Ausführungsformen werden die Operation-Konzept-Paarungen auf einer granularen Konzept/Operations-Ebene ermittelt. In einigen anderen Ausführungsformen ermittelt der Datenplatzierungsorchestrator 150 nicht, welche Operationen mit welchen Konzepten verknüpft sind, sondern er verknüpft den Satz von Operationen als Ganzes mit dem Satz von Konzepten als Ganzes. Der Datenplatzierungsorchestrator 150 ignoriert somit die tatsächlichen Ausführungsdetails/das gewünschte Ergebnis der Abfrage und definiert die Sätze der relevanten Konzepte und Operationen auf der Grundlage der beteiligten Daten/Operationen und nicht der besonderen Anwendung der Transformationen.

[0117] Das Verfahren 1300 geht dann weiter zu Block 1325, wo der Datenplatzierungsorchestrator 150 ermittelt, ob es mindestens eine zusätzliche Abfrage gibt, die noch ausgewertet werden muss. Ist dies der Fall, kehrt das Verfahren 1300 zu Block 1310 zurück. Andernfalls geht das Verfahren 1300 weiter zu Block 1330. In der Gruppe 1330 gruppiert der Datenplatzierungsorchestrator 150 die empfangenen Abfragen auf der Grundlage des entsprechenden Satzes von Konzepten für jede Abfrage. In einer Ausführungsform umfasst dies Gruppieren oder Clustern aller Abfragen, die einen übereinstimmenden Satz von Konzepten spezifizieren. Der Datenplatzierungsorchestrator 150 kann zum Beispiel alle Abfragen, die sowohl ein Konzept „Company“ als auch ein Konzept „PublicMetric“ spezifizieren, in einer ersten Gruppe zusammenfassen. Insbesondere würden in einer solchen Ausführungsform Abfragen, die nur das Konzept „Company“ oder nur das Konzept „PublicMetric“ spezifizieren, in unterschiedliche Gruppen platziert werden. Ebenso würde eine Abfrage, die „Company“ und „PublicMetric“ spezifiziert, aber auch ein Konzept „Document“ umfasst, nicht in die erste Gruppe platziert werden.

[0118] In einer Ausführungsform gruppiert der Datenplatzierungsorchestrator 150 somit Abfragen, die einen genau übereinstimmenden Satz von Konzepten spezifizieren. Abfragen, die zusätzliche oder weniger Konzepte spezifizieren, werden in andere Gruppen platziert. In einer Ausführungsform wird der Satz von Konzepten, der der jeweiligen Gruppe entspricht, verwendet, um eine entsprechende Gesamtabfrage zu bilden. In der dargestellten Ausführungsform fasst der Datenplatzierungsorchestrator 150 die empfangenen früheren Abfragen somit zusammen, indem er die Abfragen, die auf identische

Konzepte zugreifen, unabhängig von den Operationen, die jede Abfrage durchführt, in einem Cluster gruppiert.

[0119] Das Verfahren 1300 geht dann weiter zu Block 1335, wo der Datenplatzierungsorchestrator 150 eine dieser definierten Abfragegruppen auswählt. In Block 1340 wählt der Datenplatzierungsorchestrator 150 eine der Abfragen aus, die der ausgewählten Gruppe zugehörig sind. Zusätzlich ordnet der Datenplatzierungsorchestrator 150 in Block 1345 die entsprechenden Operationen, die durch die ausgewählte Abfrage spezifiziert werden, der Gesamtabfrage zu, die die ausgewählte Abfragegruppe darstellt. In einer Ausführungsform fügt der Datenplatzierungsorchestrator 150 eine Operation, die bereits in der Gesamtabfrage enthalten ist, nicht erneut hinzu. In einer Ausführungsform sind die Sätze der Operationen, die der Gesamtabfrage zugehörig sind, somit binäre Werte, die entweder das Vorhandensein oder das Nichtvorhandensein einer bestimmten Operation angeben.

[0120] In Ausführungsformen können die der Gesamtabfrage zugehörigen Operationen eine beliebige Granularität aufweisen. In einigen Ausführungsformen definiert der Datenplatzierungsorchestrator 150 beispielsweise die Gesamtabfrage auf der Ebene der Operation (z.B. „join“), ohne Rücksicht auf den speziellen Operationstyp (z.B. „inner join“), die Einschränkungen der Operation oder die Besonderheiten der Operation (z.B. den Datentyp wie „string join“ oder „integer join“). In anderen Ausführungsformen werden diese Details in die Beschreibung der Gesamtabfrage einbezogen, um mehr Details für die nachfolgende Verarbeitung bereitzustellen.

[0121] Das Verfahren 1300 geht dann weiter zu Block 1350, wo der Datenplatzierungsorchestrator 150 ermittelt, ob es mindestens eine zusätzliche Abfrage in der ausgewählten Gruppe gibt. Ist dies der Fall, kehrt das Verfahren 1300 zu Block 1340 zurück. Ist dies nicht der Fall, ermittelt der Datenplatzierungsorchestrator 150 in Block 1355, ob es mindestens eine weitere Gruppe von Abfragen gibt, die noch ausgewertet werden müssen. Ist dies der Fall, kehrt das Verfahren 1300 zu Block 1335 zurück. Andernfalls geht das Verfahren 1300 weiter zu Block 1360. In Block 1360 speichert der Datenplatzierungsorchestrator 150 die Gesamtarbeitslast so, dass sie verwendet und ausgewertet werden kann, um einen Hypergraphen zu erzeugen.

[0122] Fig. 14 ist ein Ablaufplan, der gemäß einer hier offenbarten Ausführungsform ein Verfahren 1400 zum Modellieren ontologischer Arbeitslasten veranschaulicht, um Entscheidungen über Datenplatzierungen zu unterstützen. Das Verfahren 1400 beginnt in Block 1405, wo der Datenplatzierungs-

orchestrator 150 eines der Konzepte der Ontologie auswählt. In Block 1410 erzeugt der Datenplatzierungsorchestrator 150 einen Hypergrapheckpunkt für dieses ausgewählte Konzept. Das Verfahren 1400 geht dann weiter zu Block 1415, wo der Datenplatzierungsorchestrator 150 ermittelt, ob es noch weitere Konzepte in der Ontologie gibt, die noch keinen Eckpunkt im Hypergraphen haben. Ist dies der Fall, kehrt das Verfahren 1400 zu Block 1405 zurück. Andernfalls geht das Verfahren 1400 weiter zu Block 1420.

[0123] In Block 1420 wählt der Datenplatzierungsorchestrator 150 eine der Gesamtabfragen aus. Wie oben beschrieben, gibt jede Gesamtabfrage in einer Ausführungsform einen Satz von Konzepten und einen entsprechenden Satz von Operationen an, die auf die Konzepte in der vorherigen Arbeitslast angewendet wurden. In Block 1425 erzeugt der Datenplatzierungsorchestrator 150 eine Hyperkante, die jedes der von der ausgewählten Gesamtabfrage angegebenen Konzepte verknüpft. Das Verfahren 1400 geht dann weiter zu Block 1430, wo der Datenplatzierungsorchestrator 150 die neu erzeugte Hyperkante mit einem Hinweis auf die von der ausgewählten Gesamtabfrage angegebenen Operationen kennzeichnet. Auf diese Weise kann der Datenplatzierungsorchestrator 150 anschließend den Hypergraphen auswerten, um Beziehungen und Nutzungsmuster für die Wissensdatenbank zu identifizieren.

[0124] Das Verfahren 1400 geht dann weiter zu Block 1435, wo der Datenplatzierungsorchestrator 150 ermittelt, ob es mindestens eine zusätzliche Gesamtabfrage gibt, die noch ausgewertet und in den Hypergraphen aufgenommen werden muss. Ist dies der Fall, kehrt das Verfahren 1400 zu Block 1420 zurück. Andernfalls geht das Verfahren 1400 weiter zu Block 1440, wo der Datenplatzierungsorchestrator 150 den erzeugten Hypergraphen zur späteren Verwendung speichert.

[0125] Die Fig. 15 und Fig. 16 zeigen Ablaufpläne, die gemäß einer hier offenbarten Ausführungsform Verfahren zum Auswerten eines Hypergraphen veranschaulichen, um Entscheidungen über Datenplatzierungen zu treffen. Das Verfahren 1500, das mit Bezug auf Fig. 15 beschrieben wird, veranschaulicht eine Ausführungsform einer auf Operationen gestützten Clustering-Technik zum Erzeugen von Konzeptzuordnungen, während das Verfahren 1600, das weiter unten mit Bezug auf Fig. 16 beschrieben wird, eine Ausführungsform einer Mindestabdeckungstechnik veranschaulicht.

[0126] In einer Ausführungsform gruppiert die auf Operationen gestützte Clustering-Technik Konzepte auf der Grundlage der Operationen, die sie durchlaufen. In einer solchen Ausführungsform erzeugt der

Datenplatzierungsorchestrator 150 für jede Beschreibung einer Operation im Hypergraphen einen entsprechenden Cluster. Der Datenplatzierungsorchestrator 150 durchläuft dann iterativ den Satz von Operationsbeschreibungen, die jeder Hyperkante zugehörig sind, und weist für jede dieser Operationsbeschreibungen alle Konzepte, die von der Hyperkante abgedeckt werden, dem Cluster der entsprechenden Operation zu. In einer Ausführungsform weist der Datenplatzierungsorchestrator 150 nach Zusammenstellen der Konzepte jeden Konzeptcluster einem Satz von Datenknoten zu, und zwar so, dass jeder Knoten eine Fähigkeitsbeschreibung aufweist, die mit der Operationsbeschreibung des Clusters übereinstimmt (z.B. in der Lage ist, die entsprechende Operation des Clusters durchzuführen). In einem auf Operationen gestützten System schließlich erzeugt der Datenplatzierungsorchestrator 150 eine Zuordnung, die jedes Konzept in jedem Cluster dem entsprechenden Satz identifizierter Datenspeicher zuordnet.

[0127] Als Beispiel für eine Ausführungsform der auf Operationen gestützten Clustering-Technik sei der in **Fig. 11** bereitgestellte Hypergraph 1100 genannt. Zu Beginn erzeugt der Datenplatzierungsorchestrator 150 einen Cluster C für jede Operation 1115A bis C (z.B. C_{Join} , C_{Agg} und C_{Fuzzy}). Das System ermittelt dann für jede Hyperkante den Satz der ihr zugehörigen Operationen. Für jede angegebene Operation weist das System die von der Hyperkante 1110 spezifizierten Konzepte dem entsprechenden Cluster zu. In Fortsetzung des obigen Beispiels enthält C_{Join} die Konzepte 1105A, 1105B, 1105C und 1105D der Hyperkante 1110A sowie die Konzepte 1105E und 1105F der Hyperkante 1110B. Weiterhin enthält C_{Agg} die Konzepte 1105A, 1105B, 1105C und 1105D der Hyperkante 1110A sowie das Konzept 1105G der Hyperkante 1110D. Schließlich enthält C_{Fuzzy} die Konzepte 1105A, 1105B, 1105C und 1105D der Hyperkante 1110A sowie das Konzept 1105H der Hyperkante 1110C.

[0128] In vielen Ausführungsformen überschneiden sich diese auf Operationen gestützten Cluster erheblich. Zu beachten ist zum Beispiel, dass die Konzepte 1105A, 1105B, 1105C und 1105D in jedem Cluster enthalten sind. Um die Zuordnungen abzuschließen, identifiziert der Datenplatzierungsorchestrator 150 in einer Ausführungsform für jeden Cluster alle Datenknoten 130, die die entsprechende Operation durchführen können. Der Datenplatzierungsorchestrator 150 ordnet dann alle Konzepte 1105 des Clusters allen identifizierten Datenknoten 130 zu. In einigen Ausführungsformen kann die auf Operationen gestützte Technik zwar bei der Abfrageverarbeitung ein Versetzen von Daten auf ein Mindestmaß beschränken oder verringern, indem sie Daten in allen Speichern platziert, die die entsprechenden Operationen unterstützen, doch führt dies zu einem

gewissen Replikationsaufwand, da derselbe Cluster von Konzepten in mehreren Speichern platziert werden kann, wenn diese die Fähigkeit haben, die Operation des Clusters auszuführen.

[0129] In einigen Implementierungen werden Ausführungsformen der Mindestabdeckungstechnik verwendet, um den Replikationsaufwand weiter zu verringern. In einer Ausführungsform sind Ausführungsformen der Mindestabdeckung vorteilhafter als auf Operationen gestützte Techniken, indem sie das Ausmaß der Datenreplikation weiterhin minimieren und gleichzeitig die Datenversetzung bei der Abfrageverarbeitung auf ein Mindestmaß beschränken. In einer Ausführungsform nutzt diese Technik einen Mindestabdeckungsalgorithmus, um die Mindestanzahl von Datenspeichern zu ermitteln, die erforderlich sind, um den vollständigen Satz von Operationen zu unterstützen, der für jede Hyperkante im Abfragearbeitslast-Hypergraphen erforderlich ist. In einigen Ausführungsformen verkleinert die Technik der Mindestabdeckung die Spanne jeder Hyperkante in dem Satz von Datenspeichern, die den Satz von Operationen ausführen, die für die Hyperkante erforderlich sind.

[0130] In einer beispielhaften Ausführungsform der der Mindestabdeckungstechnik ermittelt der Datenplatzierungsorchestrator 150 für jede Hyperkante im Hypergraphen die Mindestanzahl von Datenknoten, die alle angegebenen Operationen abdecken. Wenn z.B. alle Operationen von einem einzigen Datenknoten 130A ausgeführt werden können, umfasst der kleinste Satz nur diesen Knoten. Wenn eine oder mehrere der Operationen nicht von dem Datenknoten 130A ausgeführt werden können, werden ein anderer oder mehrere andere Datenknoten 130B bis N zu dem kleinsten Satz hinzugefügt, bis alle Operationen ausgeführt sind. Sobald der kleinste Satz für eine Hyperkante ermittelt ist, wird jedes Konzept in der Hyperkante jedem der Knoten in dem entsprechenden kleinsten Satz zugeordnet.

[0131] Als Beispiel für eine Ausführungsform der Mindestabdeckungs-Clustering-Technik sei der in **Fig. 11** bereitgestellte Hypergraph 1100 genannt. Angenommen, die Datenknoten 130 umfassen einen ersten Datenknoten 130A, der so konfiguriert ist, dass er die Operationen 1115A und 1115B unterstützt, einen zweiten Datenknoten 130B, der so konfiguriert ist, dass er die Operationen 1115B und 1115C unterstützt, und einen dritten Datenknoten 130C, der nur die Operation 1115B unterstützt. Für die Hyperkante 1110A kann der Datenplatzierungsorchestrator 150 feststellen, dass kein Knoten allein alle drei angegebenen Operationen unterstützen kann, wohl aber der Satz von Datenknoten 130A und 130B. Da diese beiden Knoten die gesamte Hyperkante unterstützen können, ist es nicht not-

wendig, den Datenknoten 130C zu dem Satz hinzuzufügen.

[0132] In ähnlicher Weise wird die Hyperkante 1110B dem Datenknoten 130A zugewiesen (dem einzigen Knoten, der so konfiguriert ist, dass er Verknüpfungsoperationen bereitstellt), während die Hyperkante 1110C dem Datenknoten 130B zugewiesen wird (dem einzigen Knoten, der so konfiguriert ist, dass er Fuzzymatching bereitstellt). Schließlich kann die Hyperkante 1110D entweder den Datenknoten 130A und 130B oder den Datenknoten 130B und 130C zugewiesen werden. In einigen Ausführungsformen trifft der Datenplatzierungsorchestrator 150 zwischen diesen ansonsten gleichwertigen Alternativen auf der Grundlage anderer Kriterien, z.B. der Latenzzeit oder der Rechenressourcen der jeweiligen Alternativen, vordefinierter Präferenzen und Ähnlichem, eine Auswahl. Der Datenplatzierungsorchestrator 150 ordnet dann die Konzepte 1105 jeder Hyperkante 1110 dem (den) zugewiesenen Datenknoten 130 zu.

[0133] Fig. 15 ist ein Ablaufplan, der gemäß einer hier offenbarten Ausführungsform ein auf Operationen gestütztes Verfahren 1500 zum Auswerten eines Hypergraphen veranschaulicht, um Entscheidungen über Datenplatzierungen zu treffen. Das Verfahren 1500 beginnt in Block 1505, wo der Datenplatzierungsorchestrator 150 eine der durch den Hypergraphen angegebenen Operationen auswählt. In Block 1510 erzeugt der Datenplatzierungsorchestrator 150 einen Cluster für die ausgewählte Operation. Das Verfahren 1500 geht dann weiter zu Block 1515, wo der Datenplatzierungsorchestrator 150 ermittelt, ob es mindestens eine zusätzliche Operation gibt, die im Hypergraphen enthalten ist und der noch kein Cluster zugehörig ist. Ist dies der Fall, kehrt das Verfahren 1500 zu Block 1505 zurück. Andernfalls geht das Verfahren 1500 weiter zu Block 1520.

[0134] In Block 1520 wählt der Datenplatzierungsorchestrator 150 zum Analysieren eine der Hyperkanten im Hypergraphen aus. In Block 1525 identifiziert der Datenplatzierungsorchestrator 150 das (die) Konzept(e) und die Operation(en), das/die der ausgewählten Kante zugehörig ist (sind). Das Verfahren 1500 geht dann weiter zu Block 1530, wo der Datenplatzierungsorchestrator 150 eine dieser angegebenen Operationen auswählt. Darüber hinaus identifiziert der Datenplatzierungsorchestrator 150 in Block 1535 den entsprechenden Cluster für die ausgewählte Operation und fügt alle von der ausgewählten Hyperkante angegebenen Konzepte zu diesem Cluster hinzu. Das Verfahren 1500 geht dann weiter zu Block 1540, wo der Datenplatzierungsorchestrator 150 ermittelt, ob die ausgewählte Kante mindestens eine zusätzliche noch zu verarbeitende Operation angibt. Ist dies der Fall, kehrt das Verfahren 1500 zu Block 1530 zurück.

[0135] Wenn der ausgewählten Hyperkante keine weiteren Operationen zugehörig sind, geht das Verfahren 1500 weiter zu Block 1545, wo der Datenplatzierungsorchestrator 150 ermittelt, ob der Hypergraph mindestens eine weitere Kante umfasst, die noch nicht ausgewertet wurde. Ist dies der Fall, kehrt das Verfahren 1500 zu Block 1520 zurück. Andernfalls geht das Verfahren 1500 weiter zu Block 1550, wo der Datenplatzierungsorchestrator 150 den (die) Operationscluster den (die) entsprechenden Datenknoten zuordnet, der (die) für Durchführen jeder Operation konfiguriert ist (sind). In einer Ausführungsform identifiziert der Datenplatzierungsorchestrator 150 beispielsweise für jeden Cluster den Satz von Datenknoten, die die entsprechende Operation durchführen können. In einer Ausführungsform ordnet der Datenplatzierungsorchestrator 150 dann jedes Konzept im Cluster jedem Knoten in dem identifizierten Satz von Datenknoten zu. Der Datenplatzierungsorchestrator 150 kann diese Zuordnungen verwenden, um die Daten in der Wissensdatenbank auf die Vielfalt von Datenknoten zu verteilen.

[0136] Fig. 16 ist ein Ablaufplan, der gemäß einer hier offenbarten Ausführungsform ein auf Mindestabdeckung gestütztes Verfahren 1600 zum Auswerten eines Hypergraphen veranschaulicht, um Entscheidungen über Datenplatzierungen zu treffen. Das Verfahren 1600 beginnt in Block 1605, wo der Datenplatzierungsorchestrator 150 einen der Hyperkanten im Hypergraphen auswählt. In Block 1610 identifiziert der Datenplatzierungsorchestrator 150 die entsprechenden Konzepte und Operationen, die der ausgewählten Kante zugehörig sind. Weiterhin ermittelt der Datenplatzierungsorchestrator 150 in Block 1615 den kleinsten Satz von Datenknoten, die in der Lage sind, alle angegebenen Operationen gemeinsam auszuführen.

[0137] In einer Ausführungsform geschieht dies dadurch, dass der Datenplatzierungsorchestrator 150 jede Kombination von Datenknoten iterativ auswertet, um zu ermitteln, ob die Kombination die angegebenen Operationen ausführt. Es wird somit ermittelt, ob jede Operation, die durch die ausgewählte Kante angegeben wird, von mindestens einem Datenknoten in der Kombination durchgeführt werden kann. Ist dies nicht der Fall, kann die Kombination verworfen und ein anderer Knoten oder eine andere Kombination für die Prüfung ausgewählt werden (oder ein anderer Knoten kann der aktuellen Kombination hinzugefügt werden). Der Datenplatzierungsorchestrator 150 kann ermitteln, ob jede Kombination vollständig und zufriedenstellend ist, d.h., ob sie alle erforderlichen Operationen durchführen kann, und die Kombination(en) mit der geringsten Anzahl von Datenknoten identifizieren, da diese Kombinationen wahrscheinlich zu den geringsten Datenversetzungen während der Laufzeit führen

werden. Wenn zwei oder mehr Kombinationen gleich klein sind, kann der Datenplatzierungsorchestrator 150 in einer Ausführungsform vordefinierte Kriterien oder Präferenzen verwenden, um die beste Kombination auszuwählen. So können beispielsweise vordefinierte Regeln angegeben, dass Kombinationen, die mindestens einen relationalen Datenspeicher umfassen, gegenüber Kombinationen ohne relationale Datenspeicher bevorzugt werden. Als weiteres Beispiel kann eine Regel Gewichtungen oder Prioritäten für einen oder mehrere bestimmte Speichertypen und/oder bestimmte Datenknoten 130 angeben. In einer solchen Ausführungsform kann der Datenplatzierungsorchestrator 150 diese Gewichtungen für jede Kombination u.a. zusammenfassen, um zu ermitteln, welche Speicher verwendet werden sollen. Die ausgewählte Kante wird dann mit einem Hinweis auf den ermittelten Satz von Datenknoten gekennzeichnet.

[0138] Sobald der kleinste Satz von Datenknoten ermittelt wurde, geht das Verfahren 1600 weiter zu Block 1620, wo der Datenplatzierungsorchestrator 150 ermittelt, ob es mindestens eine zusätzliche Hyperkante im Hypergraphen gibt. Ist dies der Fall, kehrt das Verfahren 1600 zu Block 1605 zurück. Andernfalls geht das Verfahren 1600 weiter zu Block 1625. In Block 1625 wählt der Datenplatzierungsorchestrator 150 einen der verfügbaren Datenknoten im System aus. In Block 1630 identifiziert der Datenplatzierungsorchestrator 150 alle Hyperkanten im Hypergraphen mit einer Kennzeichnung, die den ausgewählten Datenknoten umfasst. Der Datenplatzierungsorchestrator 150 gruppiert oder clustert dann diese Hyperkanten (oder die Konzepte, die jede von ihnen umfasst), um eine Gruppe/einen Cluster von Konzepten zu bilden, die in dem ausgewählten Knoten gespeichert werden sollen. Das Verfahren 1600 geht dann weiter zu Block 1635, wo der Datenplatzierungsorchestrator 150 ermittelt, ob es mindestens einen zusätzlichen Datenknoten im System gibt, dem noch keine Gruppe/kein Cluster zugewiesen wurde. Ist dies der Fall, kehrt das Verfahren 1600 zu Block 1625 zurück.

[0139] Andernfalls geht das Verfahren 1600 weiter zu Block 1640, wo der Datenplatzierungsorchestrator 150 für jeden Datenknoten alle im entsprechenden Cluster enthaltenen Konzepte dem Speicher zuordnet. Der Datenplatzierungsorchestrator 150 kann diese Zuordnungen anschließend verwenden, um die Daten in der Wissensdatenbank auf die Vielfalt von Datenknoten zu verteilen.

[0140] Fig. 17 ist ein Ablaufplan, der gemäß einer hier offenbarten Ausführungsform ein Verfahren 1700 zum Zuordnen von ontologischen Konzepten zu Speicherstellen veranschaulicht. Das Verfahren 1700 beginnt in Block 1705, wo ein Datenplatzierungsorchestrator 150 Informationen zur Abfragear-

beitslast ermittelt, die einer Domäne entsprechen. In Block 1710 modelliert der Datenplatzierungsorchestrator 150 die Abfragearbeitslast-Informationen als einen Hypergraphen, wobei der Hypergraph einen Satz von Eckpunkten und einen Satz von Hyperkanten umfasst, wobei jeder Eckpunkt in dem Satz von Eckpunkten einem Konzept in einer der Domäne zugehörigen Ontologie entspricht. Das Verfahren 1700 geht dann weiter zu Block 1715, wo der Datenplatzierungsorchestrator 150 Zuordnungen zwischen Konzepten und einer Mehrzahl von Datenknoten auf der Grundlage des Hypergraphen und weiterhin auf der Grundlage vordefinierter Fähigkeiten jedes der Mehrzahl von Datenknoten erzeugt. Zusätzlich baut der Datenplatzierungsorchestrator 150 in Block 1720 auf der Grundlage der erzeugten Zuordnungen eine verteilte Wissensdatenbank auf.

[0141] Die Beschreibungen der verschiedenen Ausführungsformen der vorliegenden Erfindung wurden zum Zwecke der Veranschaulichung vorgestellt, sollen jedoch nicht erschöpfend oder auf die Ausführungsformen beschränkt sein. Für Fachleute ist offensichtlich, dass viele Änderungen und Abwandlungen möglich sind, ohne vom Anwendungsbereich der beschriebenen Ausführungsformen abzuweichen. Die hier verwendete Terminologie wurde gewählt, um die Grundgedanken der Ausführungsformen, die praktische Anwendung oder technische Verbesserung gegenüber Technologien auf dem Markt bestmöglich zu erläutern oder es Fachleuten zu ermöglichen, die hierin beschriebenen Ausführungsformen zu verstehen.

[0142] Im Vorangegangenen und/oder im Folgenden wird auf in dieser Offenbarung dargestellte Ausführungsformen Bezug genommen. Es versteht sich jedoch, dass der Anwendungsbereich der vorliegenden Offenbarung nicht auf bestimmte beschriebene Ausführungsformen beschränkt ist. Stattdessen ist eine beliebige Kombination der vorherigen und/oder folgenden Merkmale und Elemente unabhängig davon, ob sie mit unterschiedlichen Ausführungsformen im Zusammenhang steht, denkbar, um die in Betracht gezogenen Ausführungsformen auszuführen und in die Praxis umzusetzen. Eine bestimmte Ausführungsform schränkt den Anwendungsbereich der vorliegenden Offenbarung unabhängig davon nicht ein, ob ein bestimmter Vorteil durch eine bestimmte Ausführungsform erzielt wird, obwohl hier offenbarte Ausführungsformen der Erfindung Vorteile gegenüber anderen möglichen Lösungen und/oder gegenüber dem Stand der Technik erzielen können. Daher sind die vorhergehenden und/oder folgenden Aspekte, Merkmale, Ausführungsformen und Vorteile lediglich veranschaulichend und sind nicht als Elemente oder Einschränkungen der beigefügten Ansprüche gedacht, ausgenommen in Fällen, in denen dies in einem Anspruch bzw. in Ansprüchen ausdrücklich angegeben ist. Ebenso ist der Bezug

auf „die Erfindung“ nicht als Verallgemeinerung eines hier offenbarten Erfindungsgegenstands auszulegen und nicht als Element oder Einschränkung der beigefügten Ansprüche anzusehen, ausgenommen in Fällen, in denen dies in einem Anspruch bzw. in Ansprüchen ausdrücklich angegeben ist.

[0143] Aspekte der vorliegenden Offenbarung können daher die Form einer kompletten Hardware-Ausführung, einer kompletten Software-Ausführung (darunter Firmware, residente Software, Mikrocode usw.) oder eine Ausführungsform haben, bei der Hardware- und Software-Aspekte kombiniert sind, die allgemein hier als „Schaltkreis“, „Modul“ oder „System“ bezeichnet werden können.

[0144] Bei der vorliegenden Erfindung kann es sich um ein System, ein Verfahren und/oder ein Computerprogrammprodukt handeln. Das Computerprogrammprodukt kann (ein) durch einen Computer lesbare(s) Speichermedium (oder-medien) umfassen, auf dem/denen durch einen Computer lesbare Programmanweisungen gespeichert ist/sind, um einen Prozessor dazu zu veranlassen, Aspekte der vorliegenden Erfindung auszuführen.

[0145] Bei dem durch einen Computer lesbaren Speichermedium kann es sich um eine physische Einheit handeln, die Anweisungen zur Verwendung durch eine Einheit zur Ausführung von Anweisungen behalten und speichern kann. Bei dem durch einen Computer lesbaren Speichermedium kann es sich zum Beispiel um eine elektronische Speichereinheit, eine magnetische Speichereinheit, eine optische Speichereinheit, eine elektromagnetische Speichereinheit, eine Halbleiterspeichereinheit oder jede geeignete Kombination daraus handeln, ohne auf diese beschränkt zu sein. Zu einer nicht erschöpfenden Liste spezifischerer Beispiele des durch einen Computer lesbaren Speichermediums gehören die Folgenden: eine tragbare Computerdiskette, eine Festplatte, ein Direktzugriffsspeicher (RAM), ein Nur-Lese-Speicher (ROM), ein löschbarer programmierbarer Nur-Lese-Speicher (EPROM bzw. Flash-Speicher), ein statischer Direktzugriffsspeicher (SRAM), ein tragbarer Kompaktspeicherplatte-Nur-Lese-Speicher (CD-ROM), eine DVD (digital versatile disc), ein Speicher-Stick, eine Diskette, eine mechanisch codierte Einheit wie zum Beispiel Lochkarten oder gehobene Strukturen in einer Rille, auf denen Anweisungen gespeichert sind, und jede geeignete Kombination daraus. Ein durch einen Computer lesbares Speichermedium soll in der Verwendung hierin nicht als flüchtige Signale an sich aufgefasst werden, wie zum Beispiel Funkwellen oder andere sich frei ausbreitende elektromagnetische Wellen, elektromagnetische Wellen, die sich durch einen Wellenleiter oder ein anderes Übertragungsmedium ausbreiten (z.B. einen Lichtwellenlei-

ter durchlaufende Lichtimpulse) oder durch einen Draht übertragene elektrische Signale.

[0146] Hier beschriebene, durch einen Computer lesbare Programmanweisungen können von einem durch einen Computer lesbaren Speichermedium auf jeweilige Datenverarbeitungs-/Verarbeitungseinheiten oder über ein Netzwerk wie zum Beispiel das Internet, ein lokales Netzwerk, ein Weitverkehrsnetzwerk und/oder ein drahtloses Netzwerk auf einen externen Computer oder eine externe Speichereinheit heruntergeladen werden. Das Netzwerk kann Kupferübertragungskabel, Lichtwellenübertragungsleiter, drahtlose Übertragung, Router, Firewalls, Vermittlungseinheiten, Gateway-Computer und/oder Edge-Server aufweisen. Eine Netzwerkadapterkarte oder Netzwerkschnittstelle in jeder Datenverarbeitungs-/Verarbeitungseinheit empfängt durch einen Computer lesbare Programmanweisungen aus dem Netzwerk und leitet die durch einen Computer lesbaren Programmanweisungen zur Speicherung in einem durch einen Computer lesbaren Speichermedium innerhalb der entsprechenden Datenverarbeitungs-/Verarbeitungseinheit weiter.

[0147] Bei durch einen Computer lesbaren Programmanweisungen zum Ausführen von Arbeitsschritten der vorliegenden Erfindung kann es sich um Assembler-Anweisungen, ISA-Anweisungen (Instruction-Set-Architecture), Maschinenanweisungen, maschinenabhängige Anweisungen, Mikrocode, Firmware-Anweisungen, zustandssetzende Daten oder entweder Quellcode oder Objektcode handeln, die in einer beliebigen Kombination aus einer oder mehreren Programmiersprachen geschrieben werden, darunter objektorientierte Programmiersprachen wie Smalltalk, C++ o.ä. sowie herkömmliche prozedurale Programmiersprachen wie die Programmiersprache „C“ oder ähnliche Programmiersprachen. Die durch einen Computer lesbaren Programmanweisungen können vollständig auf dem Computer des Benutzers, teilweise auf dem Computer des Benutzers, als eigenständiges Software-Paket, teilweise auf dem Computer des Benutzers und teilweise auf einem entfernt angeordneten Computer oder vollständig auf dem entfernt angeordneten Computer oder Server ausgeführt werden. In letzterem Fall kann der entfernt angeordnete Computer mit dem Computer des Benutzers durch eine beliebige Art Netzwerk verbunden sein, darunter ein lokales Netzwerk (LAN) oder ein Weitverkehrsnetzwerk (WAN), oder die Verbindung kann mit einem externen Computer hergestellt werden (zum Beispiel über das Internet unter Verwendung eines Internet-Diensteanbieters). In einigen Ausführungsformen können elektronische Schaltungen, darunter zum Beispiel programmierbare Logikschaltungen, vor Ort programmierbare Gatter-Anordnungen (FPGA, field programmable gate arrays) oder programmierbare Logikanordnungen (PLA, programmable logic

arrays) die durch einen Computer lesbaren Programmanweisungen ausführen, indem sie Zustandsinformationen der durch einen Computer lesbaren Programmanweisungen nutzen, um die elektronischen Schaltungen zu personalisieren, um Aspekte der vorliegenden Erfindung durchzuführen.

[0148] Aspekte der vorliegenden Erfindung sind hier unter Bezugnahme auf Ablaufpläne und/oder Blockschaltbilder bzw. Schaubilder von Verfahren, Vorrichtungen (Systemen) und Computerprogrammprodukten gemäß Ausführungsformen der Erfindung beschrieben. Es wird darauf hingewiesen, dass jeder Block der Ablaufpläne und/oder der Blockschaltbilder bzw. Schaubilder sowie Kombinationen von Blöcken in den Ablaufplänen und/oder den Blockschaltbildern bzw. Schaubildern mittels durch einen Computer lesbare Programmanweisungen ausgeführt werden können.

[0149] Diese durch einen Computer lesbaren Programmanweisungen können einem Prozessor eines Universalcomputers, eines Spezialcomputers oder einer anderen programmierbaren Datenverarbeitungsvorrichtung bereitgestellt werden, um eine Maschine zu erzeugen, sodass die über den Prozessor des Computers bzw. der anderen programmierbaren Datenverarbeitungsvorrichtung ausgeführten Anweisungen ein Mittel zur Umsetzung der in dem Block bzw. den Blöcken der Ablaufpläne und/oder der Blockschaltbilder bzw. Schaubilder festgelegten Funktionen/Schritte erzeugen. Diese durch einen Computer lesbaren Programmanweisungen können auch auf einem durch einen Computer lesbaren Speichermedium gespeichert sein, das einen Computer, eine programmierbare Datenverarbeitungsvorrichtung und/oder andere Einheiten so steuern kann, dass sie auf eine bestimmte Art funktionieren, sodass das durch einen Computer lesbare Speichermedium, auf dem Anweisungen gespeichert sind, einen Herstellungsartikel aufweist, darunter Anweisungen, welche Aspekte der/des in dem Block bzw. den Blöcken des Ablaufplans und/oder der Blockschaltbilder bzw. Schaubilder angegebenen Funktion/Schritts umsetzen.

[0150] Die durch einen Computer lesbaren Programmanweisungen können auch auf einen Computer, eine andere programmierbare Datenverarbeitungsvorrichtung oder eine andere Einheit geladen werden, um das Ausführen einer Reihe von Prozessschritten auf dem Computer bzw. der anderen programmierbaren Vorrichtung oder anderen Einheit zu verursachen, um einen auf einem Computer ausgeführten Prozess zu erzeugen, sodass die auf dem Computer, einer anderen programmierbaren Vorrichtung oder einer anderen Einheit ausgeführten Anweisungen die in dem Block bzw. den Blöcken der Ablaufpläne und/oder der Blockschaltbilder bzw.

Schaubilder festgelegten Funktionen/Schritte umsetzen.

[0151] Die Ablaufpläne und die Blockschaltbilder bzw. Schaubilder in den Figuren veranschaulichen die Architektur, die Funktionalität und den Betrieb möglicher Ausführungen von Systemen, Verfahren und Computerprogrammprodukten gemäß verschiedenen Ausführungsformen der vorliegenden Erfindung. In diesem Zusammenhang kann jeder Block in den Ablaufplänen oder Blockschaltbildern bzw. Schaubildern ein Modul, ein Segment oder einen Teil von Anweisungen darstellen, die eine oder mehrere ausführbare Anweisungen zur Ausführung der bestimmten logischen Funktion(en) aufweisen. In einigen alternativen Ausführungen können die in dem Block angegebenen Funktionen in einer anderen Reihenfolge als in den Figuren gezeigt stattfinden. Zwei nacheinander gezeigte Blöcke können zum Beispiel in Wirklichkeit im Wesentlichen gleichzeitig ausgeführt werden, oder die Blöcke können manchmal je nach entsprechender Funktionalität in umgekehrter Reihenfolge ausgeführt werden. Es ist ferner anzumerken, dass jeder Block der Blockschaltbilder bzw. Schaubilder und/oder der Ablaufpläne sowie Kombinationen aus Blöcken in den Blockschaltbildern bzw. Schaubildern und/oder den Ablaufplänen durch spezielle auf Hardware beruhende Systeme umgesetzt werden können, welche die festgelegten Funktionen oder Schritte durchführen, oder Kombinationen aus Spezial-Hardware und Computeranweisungen ausführen.

[0152] Ausführungsformen der Erfindung können Endbenutzern über eine Cloud-Computing-Infrastruktur bereitgestellt werden. Der Begriff Cloud-Computing bezeichnet allgemein das Bereitstellen skalierbarer Datenverarbeitungsressourcen als Dienst über ein Netzwerk. Genauer ausgedrückt kann Cloud-Computing als Datenverarbeitungsfähigkeit definiert werden, die eine Abstraktion zwischen der Datenverarbeitungsressource und der ihr zugrunde liegenden technischen Architektur (z.B. Server, Speicher, Netzwerke) bereitstellt, die einen benutzerfreundlichen, bedarfsorientierten Netzwerkzugriff auf einen gemeinsam genutzten Pool von konfigurierbaren Datenverarbeitungsressourcen ermöglicht, die mit minimalem Verwaltungsaufwand oder Interaktion mit dem Dienstanbieter schnell bereitgestellt und freigegeben werden können. Daher ermöglicht das Cloud-Computing einem Benutzer unabhängig von den zugrunde liegenden physischen Systemen (oder den Standorten dieser Systeme), die zum Bereitstellen der Datenverarbeitungsressourcen verwendet werden, auf virtuelle Datenverarbeitungsressourcen (z.B. Speicher, Daten, Anwendungen und selbst vollständige virtualisierte Datenverarbeitungssysteme) in der „Cloud“ zuzugreifen.

[0153] In der Regel werden Cloud-Computing-Ressourcen einem Benutzer auf der Grundlage der Kosten pro Nutzung (pay-per-use) bereitgestellt, bei der Benutzer nur Kosten für die tatsächlich genutzten Datenverarbeitungsressourcen (z.B. eine Größe des von einem Benutzer verwendeten Speicherplatzes oder eine Anzahl virtualisierter Systeme, die der Benutzer instanziiert hat) in Rechnung gestellt werden. Ein Benutzer kann über das Internet jederzeit und von überall auf beliebige der Ressourcen zugreifen, die sich in der Cloud befinden. Im Zusammenhang mit der vorliegenden Erfindung kann ein Benutzer auf Anwendungen (z.B. den Abfrageverarbeitungsorchestrator) oder zugehörige Daten zugreifen, die in der Cloud zur Verfügung stehen. Der Abfrageverarbeitungsorchestrator 115 könnte zum Beispiel auf einem Datenverarbeitungssystem in der Cloud ausgeführt werden und Abfragen und Backend-Ressourcen auswerten. In einem solchen Fall könnte der Abfrageverarbeitungsorchestrator 115 Abfragen weiterleiten und Backend-Ressourcen und/oder Fähigkeitskonfigurationen an einem Speicherort in der Cloud speichern. Dies ermöglicht einem Benutzer, von einem beliebigen Datenverarbeitungssystem, das an ein mit der Cloud verbundenes Netzwerk (z.B. das Internet) angeschlossen ist, auf Daten zuzugreifen.

[0154] Obwohl das Vorstehende auf Ausführungsformen der vorliegenden Erfindung gerichtet ist, können andere und weitere Ausführungsformen der Erfindung entwickelt werden, ohne von deren grundlegendem Umfang abzuweichen, und deren Umfang wird durch die nachfolgenden Ansprüche festgelegt.

Patentansprüche

1. Verfahren, das aufweist:

Ermitteln von Abfragearbeitslast-Informationen, die einer Domäne entsprechen, durch einen Datenorchestrator;
Modellieren der Abfragearbeitslast-Informationen als einen Hypergraphen, wobei der Hypergraph einen Satz von Eckpunkten und einen Satz von Hyperkanten aufweist, wobei jeder Eckpunkt in dem Satz von Eckpunkten einem Konzept in einer der Domäne zugehörigen Ontologie entspricht;
Erzeugen von Zuordnungen zwischen Konzepten und einer Mehrzahl von Datenknoten auf Grundlage des Hypergraphen und weiterhin auf Grundlage vordefinierter Fähigkeiten jedes der Mehrzahl von Datenknoten; und
Einrichten einer verteilten Wissensdatenbank auf Grundlage der erzeugten Zuordnungen.

2. Verfahren nach Anspruch 1, wobei das Ermitteln der Abfragearbeitslast-Informationen aufweist:
Empfangen eines Satzes von früheren ontologischen Abfragen;
Erzeugen eines ersten Satzes von Konzepten, auf

die durch eine erste Abfrage in dem Satz von früheren ontologischen Abfragen zugegriffen wird;
Erzeugen eines zweiten Satzes von Operationen, die durch die erste Abfrage durchgeführt werden; und

Erzeugen einer ersten Gesamtabfrage durch:
Identifizieren einer Gruppe von Abfragen aus dem Satz früherer ontologischer Abfragen mit entsprechenden übereinstimmenden ersten Sätzen;
Ermitteln auf Grundlage entsprechender zweiter Sätze für jede Abfrage in der identifizierten Gruppe von Abfragen eines Gesamtsatzes von Operationen; und
Zuordnen der ersten Gesamtabfrage zu dem zusammengefassten Satz von Operationen und Konzepten, der sich in den entsprechenden übereinstimmenden ersten Sätzen widerspiegelt.

3. Verfahren nach Anspruch 2, wobei das Modellieren der Abfragearbeitslast-Informationen als einen Hypergraphen aufweist:

Erzeugen eines Eckpunkts für jedes Konzept in der Ontologie;
Erzeugen einer ersten Hyperkante für die erste Gesamtabfrage, wobei die erste Hyperkante einen ersten Satz von Eckpunkten in dem Hypergraphen verbindet, wobei der erste Satz von Eckpunkten den Konzepten entspricht, die in den übereinstimmenden ersten Sätzen widerspiegelt werden; und
Kennzeichnen der ersten Hyperkante mit dem zusammengefassten Satz von Operationen.

4. Verfahren nach Anspruch 1, wobei das Erzeugen der Zuordnungen aufweist:

Erzeugen eines ersten Clusters für eine erste Operation, die in dem Hypergraphen enthalten ist;
Identifizieren eines ersten Satzes von Konzepten, die durch eine erste Hyperkante im Hypergraphen verbunden sind;
Identifizieren eines ersten Satzes von Operationen, die durch die erste Hyperkante angegeben werden; und
nach einem Feststellen, dass der erste Satz von Operationen die erste Operation aufweist, Zuweisen des ersten Satzes von Konzepten zu dem ersten Cluster.

5. Verfahren nach Anspruch 4, wobei das Erzeugen der Zuordnungen weiterhin Zuordnen des ersten Satzes von Konzepten zu einem oder mehreren Datenknoten aufweist durch:

Identifizieren eines Satzes von Datenknoten, die dazu angepasst sind, die erste Operation durchzuführen; und
Zuordnen jedes Konzepts im ersten Satz von Konzepten zu jedem Datenknoten in dem identifizierten Satz von Datenknoten.

6. Verfahren nach Anspruch 1, wobei das Erzeugen der Zuordnungen aufweist:

Identifizieren eines ersten Satzes von Konzepten, die durch eine erste Hyperkante im Hypergraphen verbunden sind;
 Identifizieren eines ersten Satzes von Operationen, die durch die erste Hyperkante angegeben werden;
 Bestimmen eines kleinsten Satzes von Datenknoten, die dazu angepasst sind, gemeinsam den ersten Satz von Operationen durchzuführen;
 Erzeugen eines Clusters, der den ersten Satz von Konzepten aufweist; und
 Kennzeichnen des Clusters mit dem kleinsten Satz von Datenknoten.

7. Verfahren nach Anspruch 6, wobei das Erzeugen der Zuordnungen weiterhin Zuordnen jedes Konzepts im ersten Satz von Konzepten zu jedem Datenknoten im kleinstmöglichen Satz von Datenknoten aufweist.

8. Verfahren nach Anspruch 1, wobei das Einrichten der verteilten Wissensdatenbank für jedes entsprechende Konzept in der Ontologie aufweist:
 Identifizieren eines entsprechenden Datenknotens, der durch die Zuordnungen angegeben wird;
 Identifizieren von Daten, die dem entsprechenden Konzept entsprechen; und
 Ermöglichen des Speicherns der identifizierten Daten in dem entsprechenden Datenknoten.

9. Durch einen Computer lesbares Speichermedium, das einen Computerprogrammcode enthält, der, wenn er durch einen oder mehrere Computerprozessoren ausgeführt wird, eine Operation durchführt, die aufweist:
 Ermitteln von Abfragearbeitslast-Informationen, die einer Domäne entsprechen, durch einen Datenorchestrator;
 Modellieren der Abfragearbeitslast-Informationen als einen Hypergraphen, wobei der Hypergraph einen Satz von Eckpunkten und einen Satz von Hyperkanten aufweist, wobei jeder Eckpunkt in dem Satz von Eckpunkten einem Konzept in einer der Domäne zugehörigen Ontologie entspricht;
 Erzeugen von Zuordnungen zwischen Konzepten und einer Mehrzahl von Datenknoten auf Grundlage des Hypergraphen und weiterhin auf der Grundlage vordefinierter Fähigkeiten jedes der Mehrzahl von Datenknoten; und
 Einrichten einer verteilten Wissensdatenbank auf Grundlage der erzeugten Zuordnungen.

10. Durch einen Computer lesbares Speichermedium nach Anspruch 9, wobei das Ermitteln der Abfragearbeitslast-Informationen aufweist:
 Empfangen eines Satzes von früheren ontologischen Abfragen;
 Erzeugen eines ersten Satzes von Konzepten, auf die durch eine erste Abfrage in dem Satz von früheren ontologischen Abfragen zugegriffen wird;
 Erzeugen eines zweiten Satzes von Operationen,

die durch die erste Abfrage durchgeführt werden; und
 Erzeugen einer ersten Gesamtabfrage durch:
 Identifizieren einer Gruppe von Abfragen aus dem Satz früherer ontologischer Abfragen mit entsprechenden übereinstimmenden ersten Sätzen;
 Ermitteln auf der Grundlage entsprechender zweiter Sätze für jede Abfrage in der identifizierten Gruppe von Abfragen eines Gesamtsatzes von Operationen; und
 Zuordnen der ersten Gesamtabfrage zu dem zusammengefassten Satz von Operationen und Konzepten, der sich in den entsprechenden übereinstimmenden ersten Sätzen entspricht.

11. Durch einen Computer lesbares Speichermedium nach Anspruch 10, wobei das Modellieren der Abfragearbeitslast-Informationen als einen Hypergraphen aufweist:
 Erzeugen eines Eckpunkts für jedes Konzept in der Ontologie;
 Erzeugen einer ersten Hyperkante für die erste Gesamtabfrage, wobei die erste Hyperkante einen ersten Satz von Eckpunkten in dem Hypergraphen verbindet, wobei der erste Satz von Eckpunkten den Konzepten entspricht, die in den übereinstimmenden ersten Sätzen widerspiegelt werden; und
 Kennzeichnen der ersten Hyperkante mit dem zusammengefassten Satz von Operationen.

12. Durch einen Computer lesbares Speichermedium nach Anspruch 9, wobei das Erzeugen der Zuordnungen aufweist:
 Erzeugen eines ersten Clusters für eine erste Operation, die in dem Hypergraphen enthalten ist;
 Identifizieren eines ersten Satzes von Konzepten, die durch eine erste Hyperkante im Hypergraphen verbunden sind;
 Identifizieren eines ersten Satzes von Operationen, die durch die erste Hyperkante angegeben werden; und
 nach einem Feststellen, dass der erste Satz von Operationen die erste Operation umfasst, Zuweisen des ersten Satzes von Konzepten zu dem ersten Cluster.

13. Durch einen Computer lesbares Speichermedium nach Anspruch 12, wobei das Erzeugen der Zuordnungen weiterhin Zuordnen des ersten Satzes von Konzepten zu einem oder mehreren Datenknoten aufweist durch:
 Identifizieren eines Satzes von Datenknoten, die in der Lage sind, die erste Operation durchzuführen; und
 Zuordnen jedes Konzepts im ersten Satz von Konzepten zu jedem Datenknoten in dem identifizierten Satz von Datenknoten.

14. Durch einen Computer lesbares Speichermedium nach Anspruch 9, wobei das Erzeugen der

Zuordnungen aufweist:

Identifizieren eines ersten Satzes von Konzepten, die durch eine erste Hyperkante im Hypergraphen verbunden sind;
 Identifizieren eines ersten Satzes von Operationen, die durch die erste Hyperkante angegeben werden;
 Bestimmen eines kleinsten Satzes von Datenknoten, die dazu angepasst sind, gemeinsam den ersten Satz von Operationen durchzuführen;
 Erzeugen eines Clusters, der den ersten Satz von Konzepten aufweist; und
 Kennzeichnen des Clusters mit dem kleinsten Satz von Datenknoten.

15. Durch einen Computer lesbares Speichermedium nach Anspruch 9, wobei das Einrichten der verteilten Wissensdatenbank für jedes jeweilige Konzept in der Ontologie aufweist:
 Identifizieren eines entsprechenden Datenknotens, der durch die Zuordnungen angegeben wird;
 Identifizieren von Daten, die dem entsprechenden Konzept entsprechen; und
 Ermöglichen des Speicherns der identifizierten Daten in dem entsprechenden Datenknoten.

16. System, das aufweist:
 einen oder mehrere Computerprozessoren; und
 einen Arbeitsspeicher, der ein Programm enthält, das, wenn es von dem einen oder den mehreren Computerprozessoren ausgeführt wird, eine Operation durchführt, wobei die Operation aufweist:
 Ermitteln von Abfragearbeitslast-Informationen, die einer Domäne entsprechen, durch einen Datenorchestrator;
 Modellieren der Abfragearbeitslast-Informationen als einen Hypergraphen, wobei der Hypergraph einen Satz von Eckpunkten und einen Satz von Hyperkanten aufweist, wobei jeder Eckpunkt in dem Satz von Eckpunkten einem Konzept in einer der Domäne zugehörigen Ontologie entspricht;
 Erzeugen von Zuordnungen zwischen Konzepten und einer Mehrzahl von Datenknoten auf Grundlage des Hypergraphen und weiterhin auf Grundlage vordefinierter Fähigkeiten jedes der Mehrzahl von Datenknoten; und
 Einrichten einer verteilten Wissensdatenbank auf Grundlage der erzeugten Zuordnungen.

17. System nach Anspruch 16, wobei das Ermitteln der Abfragearbeitslast-Informationen aufweist:
 Empfangen eines Satzes von früheren ontologischen Abfragen;
 Erzeugen eines ersten Satzes von Konzepten, auf die durch eine erste Abfrage in dem Satz von früheren ontologischen Abfragen zugegriffen wird;
 Erzeugen eines zweiten Satzes von Operationen, die durch die erste Abfrage durchgeführt werden; und
 Erzeugen einer ersten Gesamtabfrage durch:
 Identifizieren einer Gruppe von Abfragen aus dem

Satz früherer ontologischer Abfragen mit entsprechenden übereinstimmenden ersten Sätzen;
 Ermitteln auf der Grundlage entsprechender zweiter Sätze für jede Abfrage in der identifizierten Gruppe von Abfragen eines Gesamtsatzes von Operationen; und
 Zuordnen der ersten Gesamtabfrage zu dem zusammengefassten Satz von Operationen und Konzepten, der sich in den entsprechenden übereinstimmenden ersten Sätzen widerspiegelt.

18. System nach Anspruch 17, wobei das Modellieren der Abfragearbeitslast-Informationen als einen Hypergraphen aufweist:
 Erzeugen eines Eckpunkts für jedes Konzept in der Ontologie;
 Erzeugen einer ersten Hyperkante für die erste Gesamtabfrage, wobei die erste Hyperkante einen ersten Satz von Eckpunkten in dem Hypergraphen verbindet, wobei der erste Satz von Eckpunkten den Konzepten entspricht, die in den übereinstimmenden ersten Sätzen widerspiegelt werden; und
 Kennzeichnen der ersten Hyperkante mit dem zusammengefassten Satz von Operationen.

19. System nach Anspruch 16, wobei das Erzeugen der Zuordnungen aufweist:
 Erzeugen eines ersten Clusters für eine erste Operation, die in dem Hypergraphen enthalten ist;
 Identifizieren eines ersten Satzes von Konzepten, die durch eine erste Hyperkante im Hypergraphen verbunden sind;
 Identifizieren eines ersten Satzes von Operationen, die durch die erste Hyperkante angegeben werden; und
 nach einem Feststellen, dass der erste Satz von Operationen die erste Operation umfasst, Zuweisen des ersten Satzes von Konzepten zu dem ersten Cluster.

20. System nach Anspruch 16, wobei das Erzeugen der Zuordnungen aufweist:
 Identifizieren eines ersten Satzes von Konzepten, die durch eine erste Hyperkante im Hypergraphen verbunden sind;
 Identifizieren eines ersten Satzes von Operationen, die durch die erste Hyperkante angegeben werden;
 Bestimmen eines kleinsten Satzes von Datenknoten, die in der Lage sind, gemeinsam den ersten Satz von Operationen durchzuführen;
 Erzeugen eines Clusters, der den ersten Satz von Konzepten aufweist; und
 Kennzeichnen des Clusters mit dem kleinsten Satz von Datenknoten.

Es folgen 19 Seiten Zeichnungen

Anhängende Zeichnungen

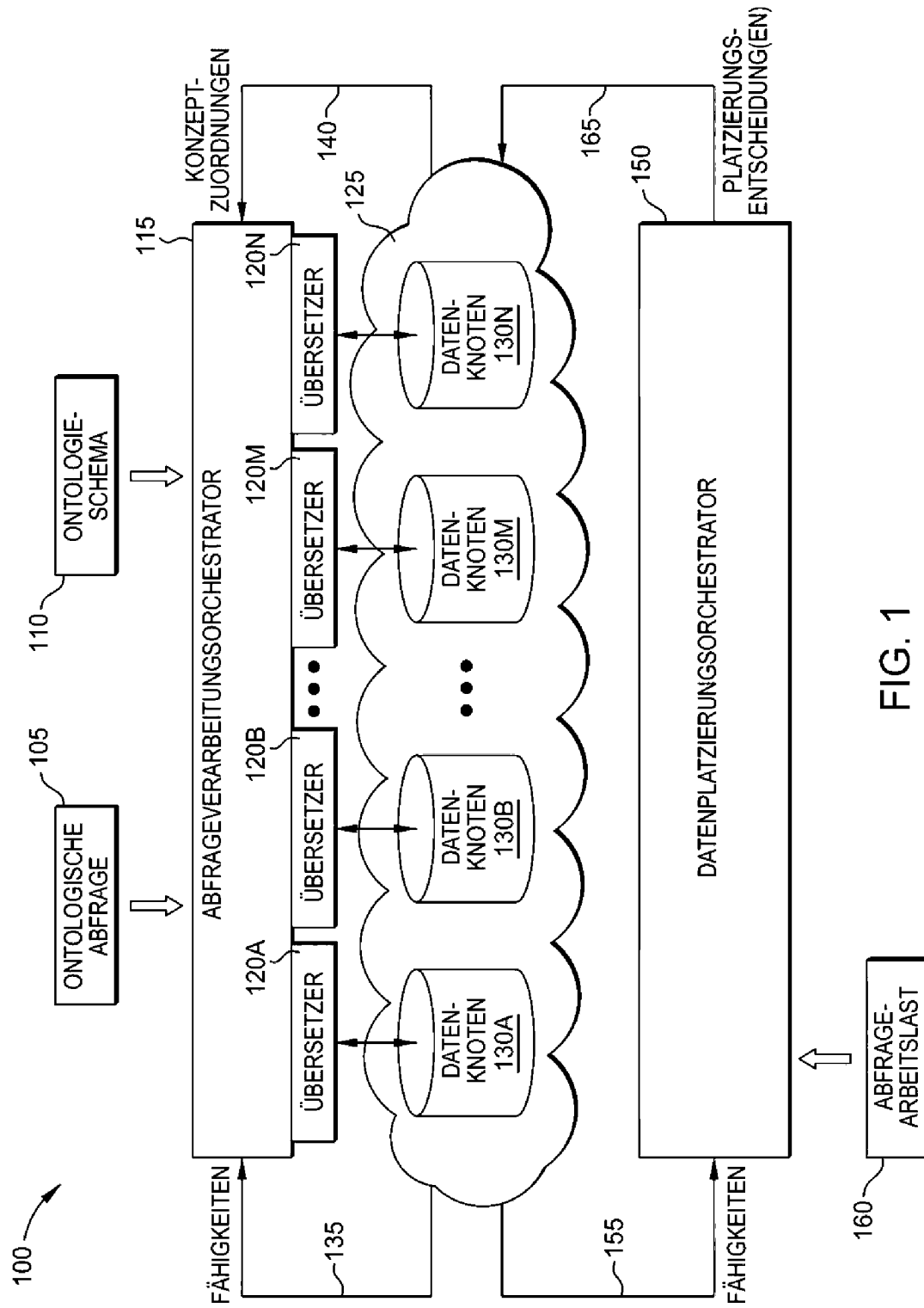


FIG. 1

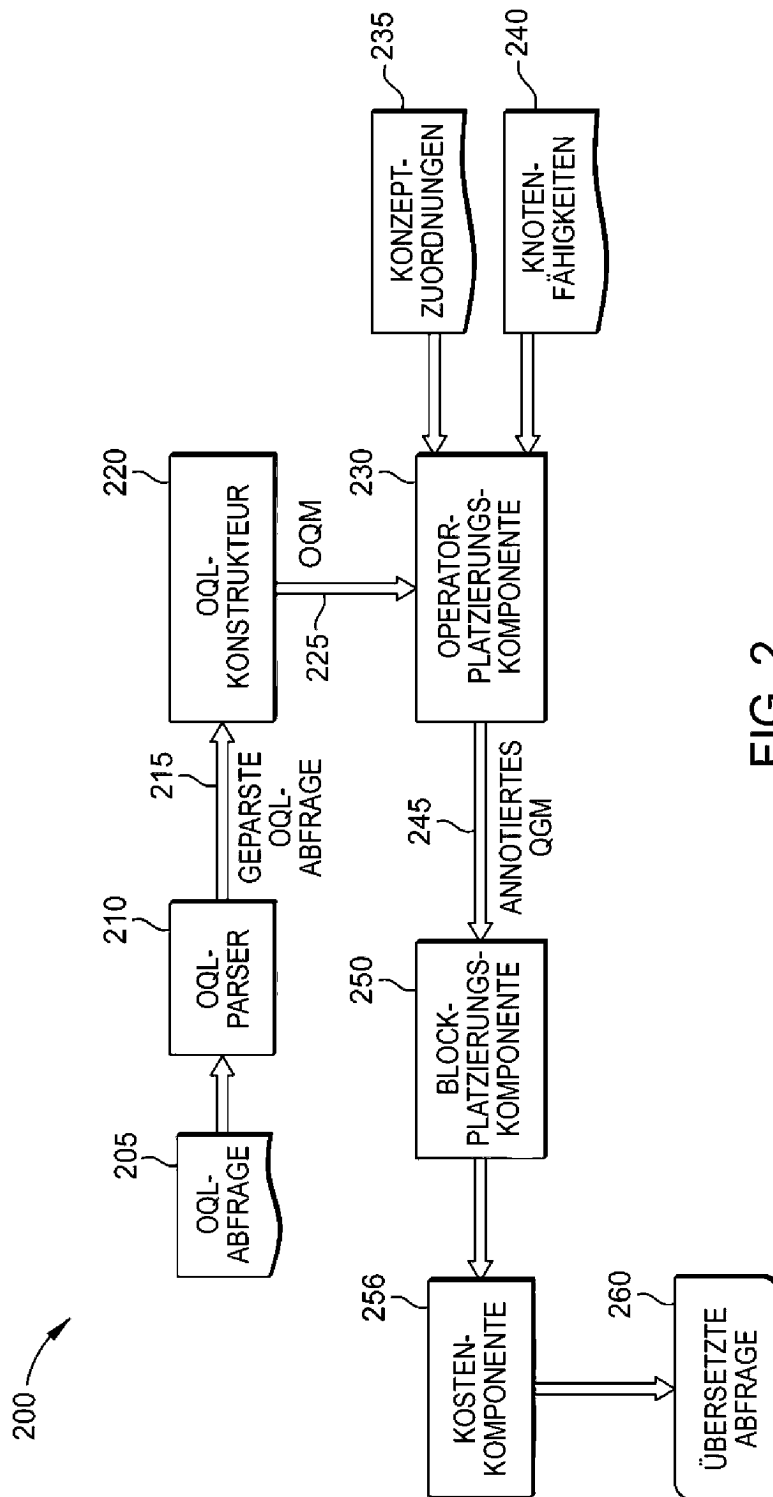


FIG. 2

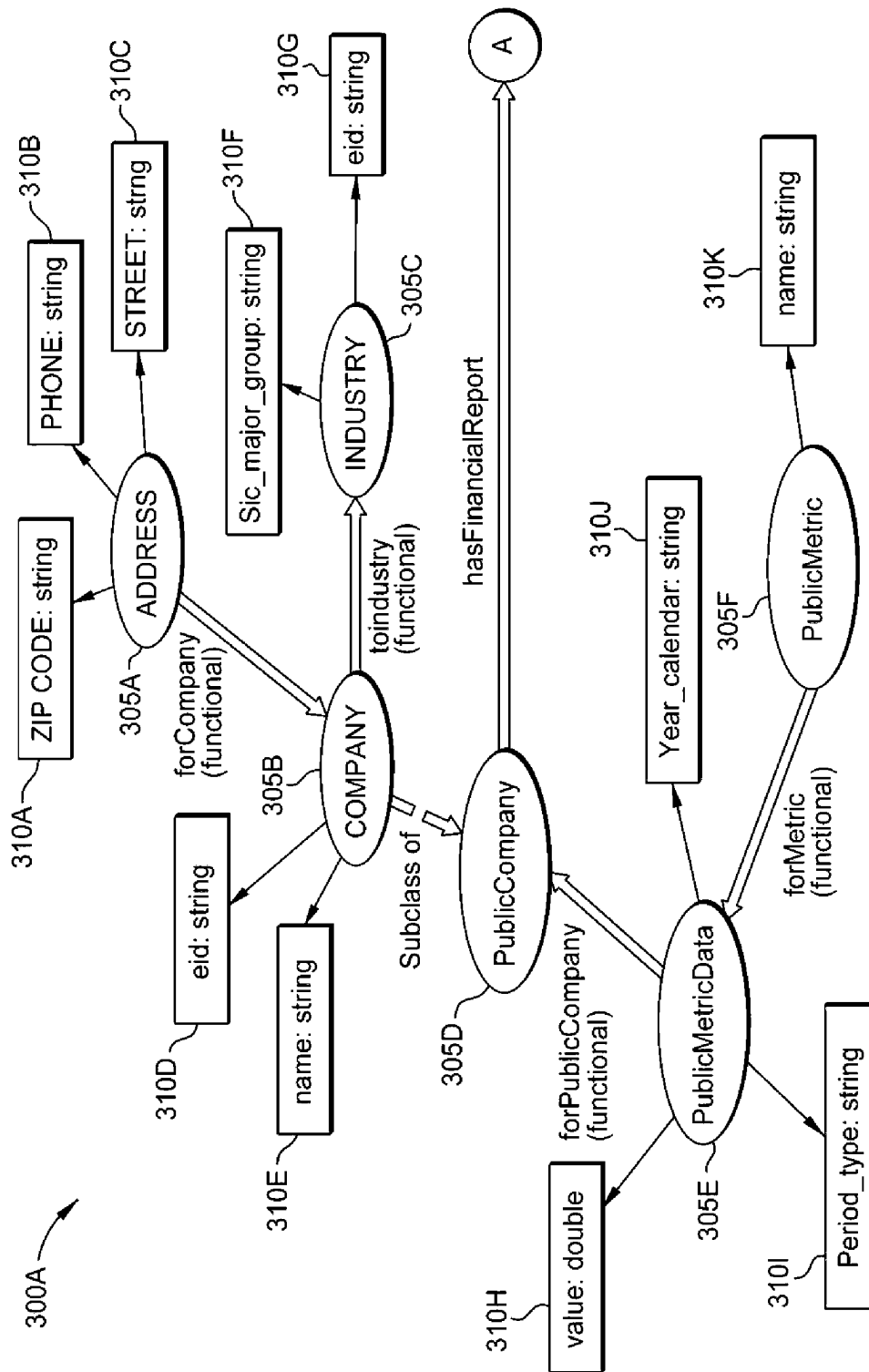


FIG. 3A

300B

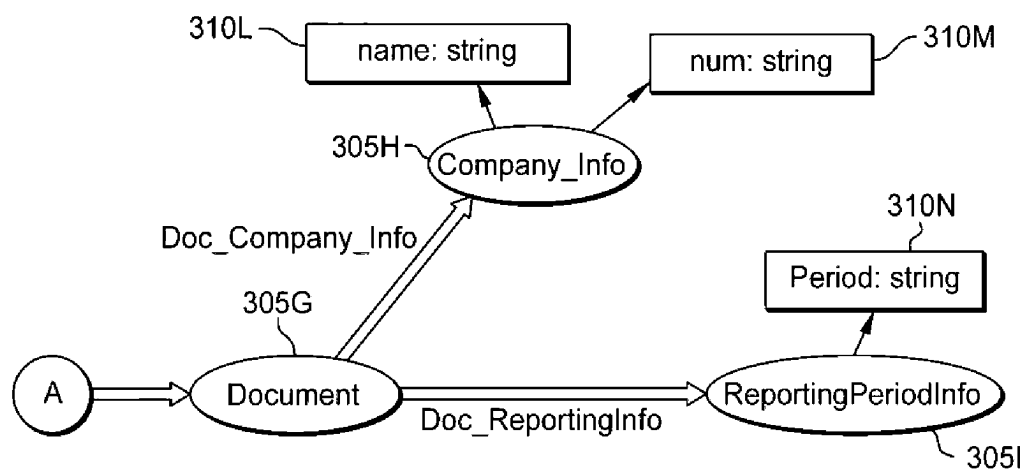


FIG. 3B

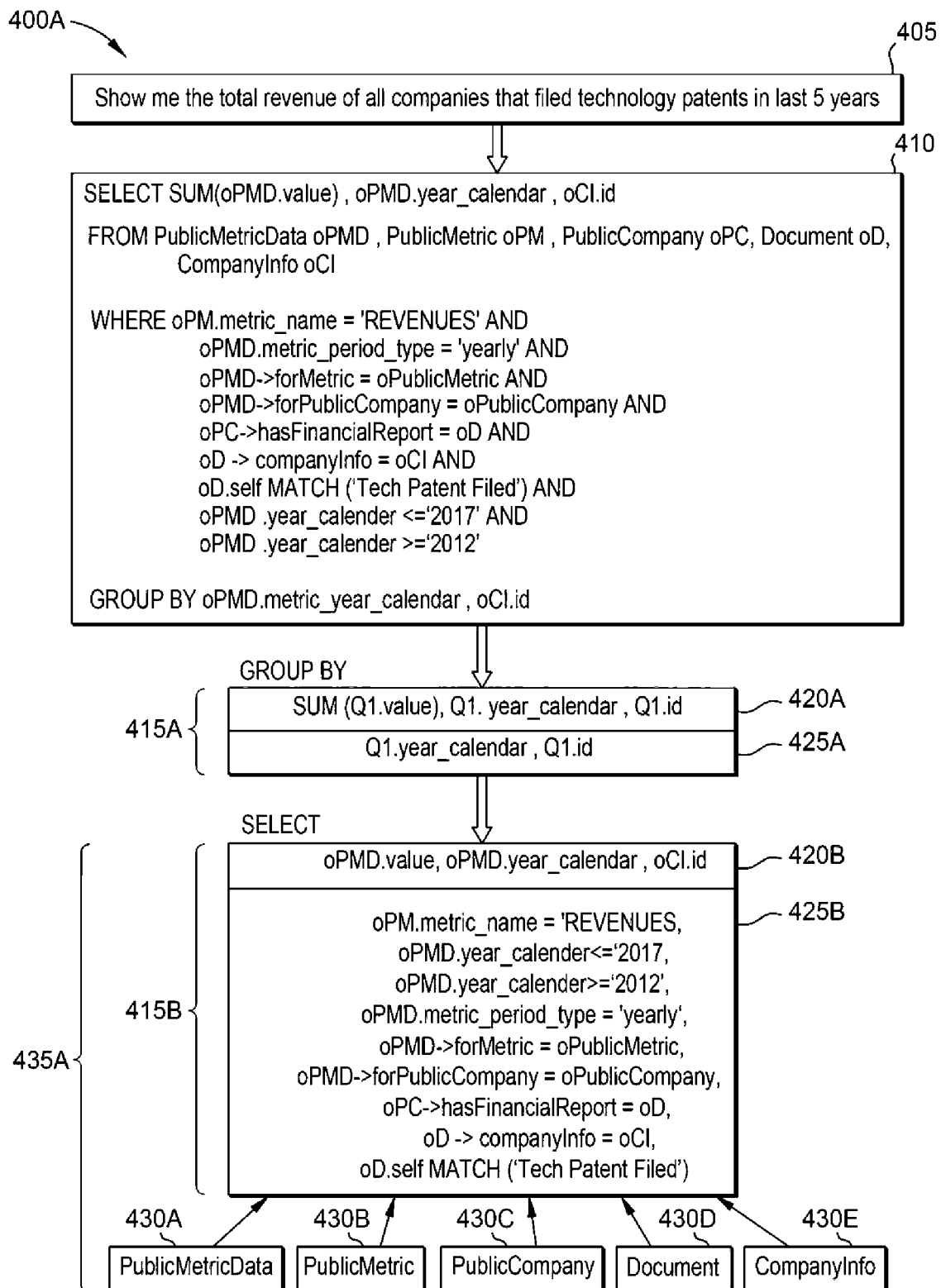


FIG. 4A

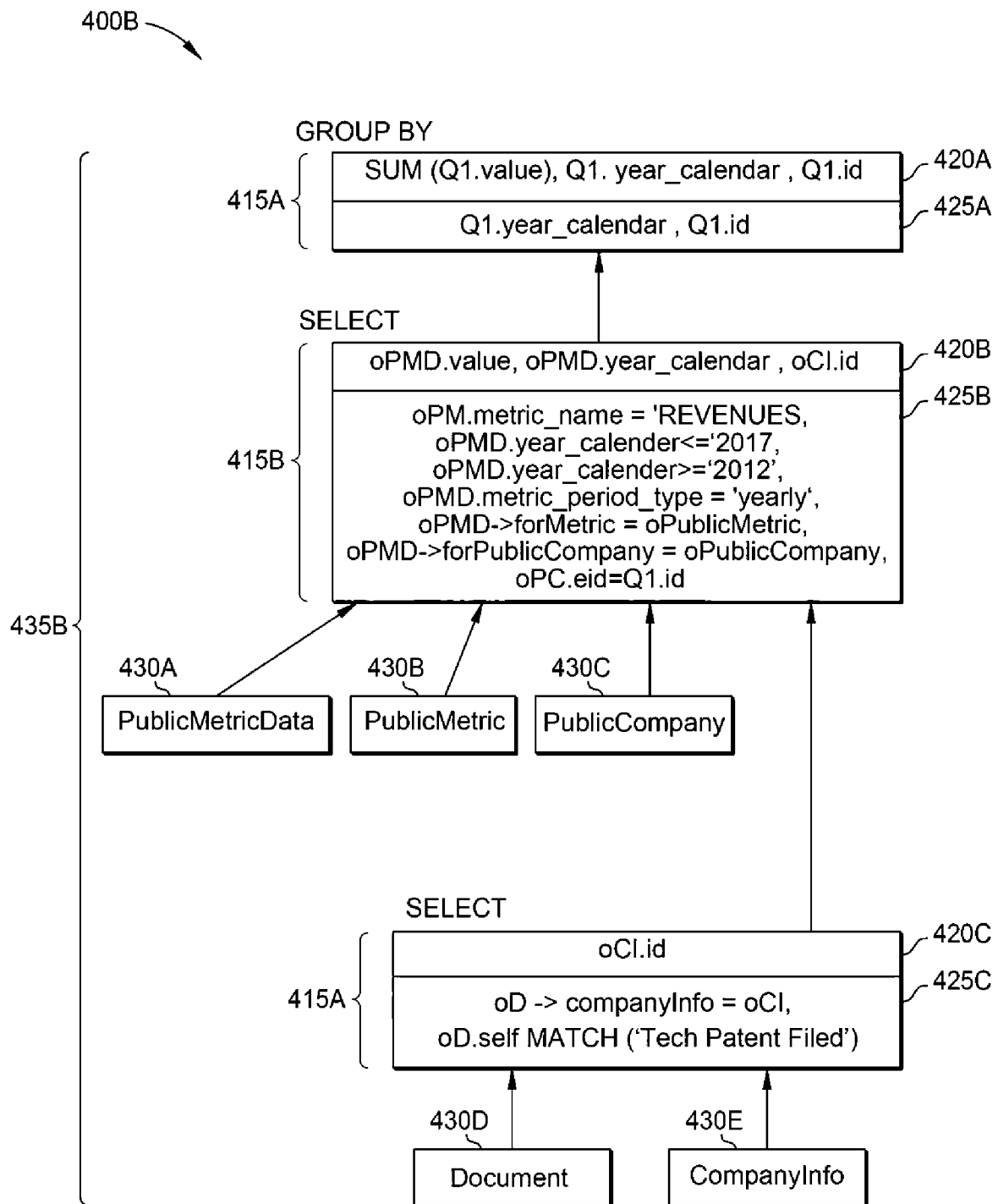


FIG. 4B

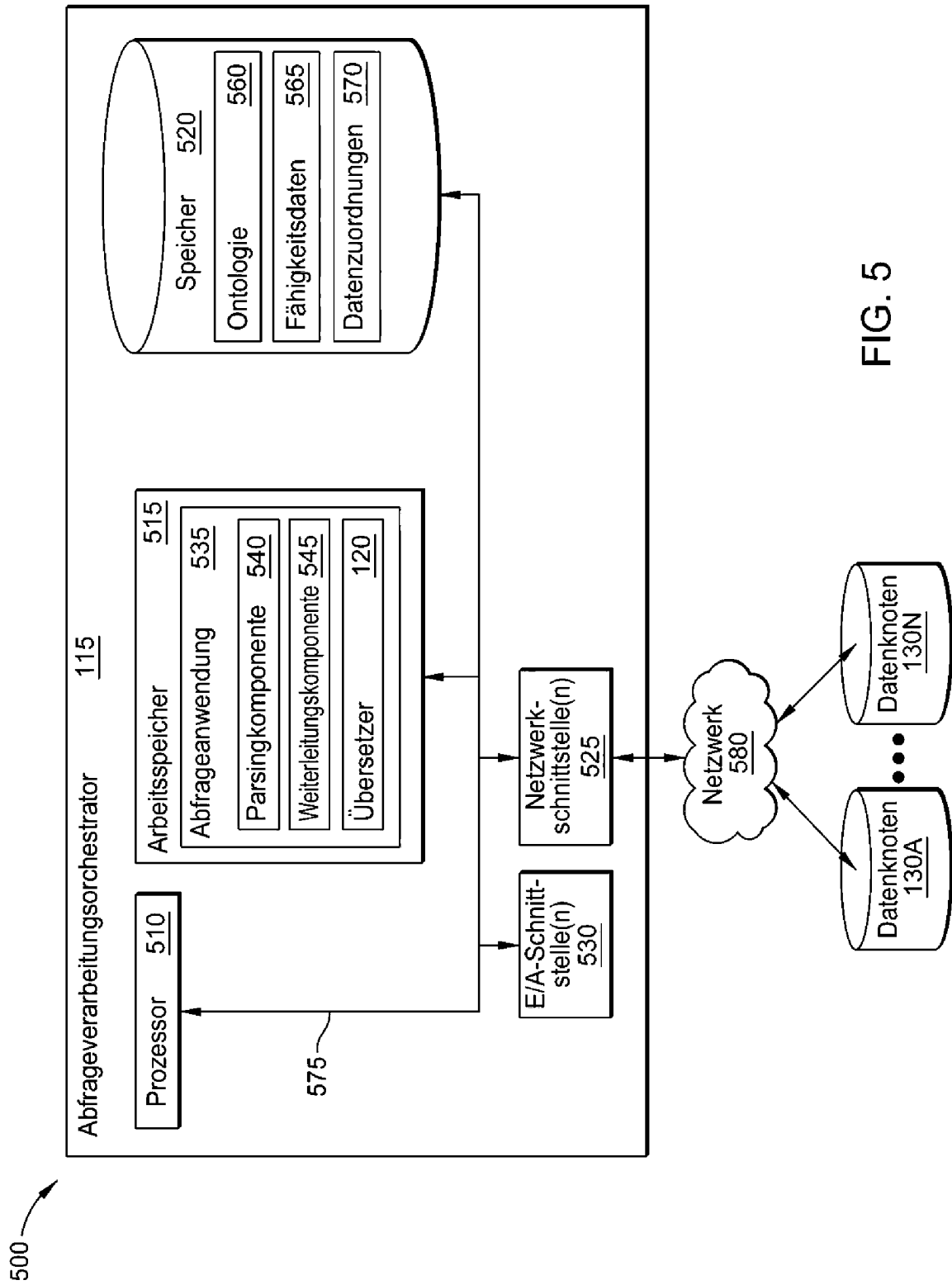


FIG. 5

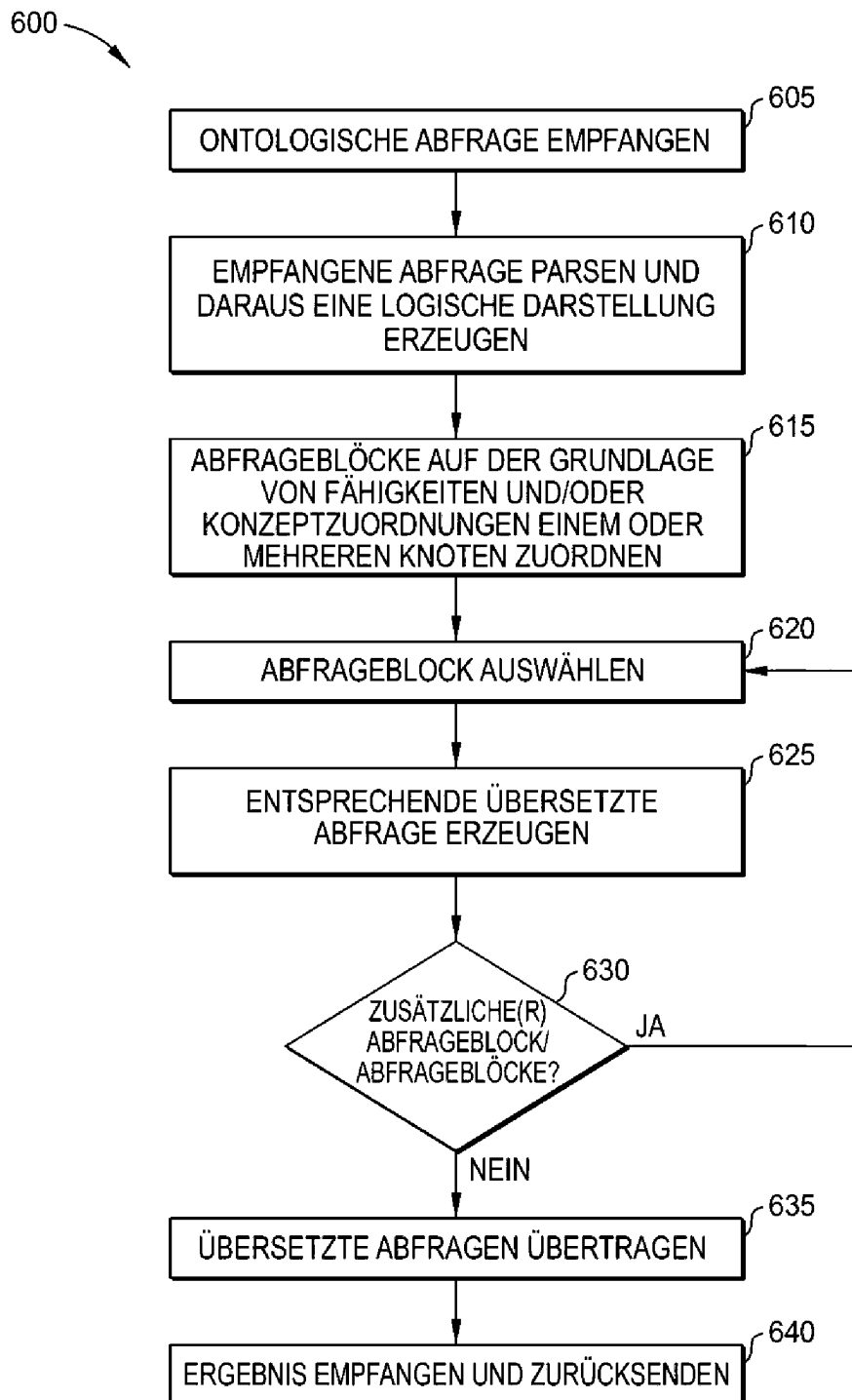


FIG. 6

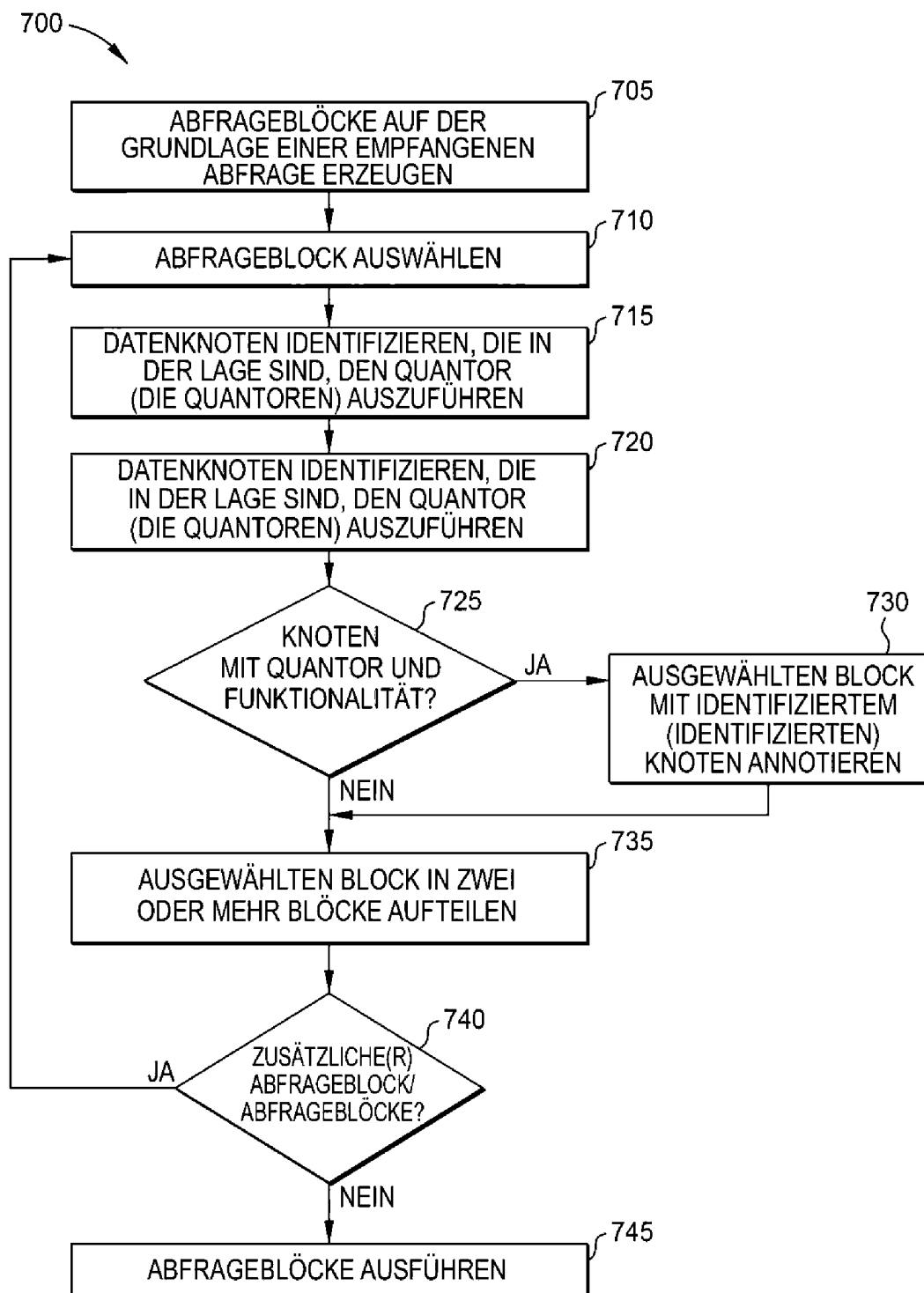


FIG. 7

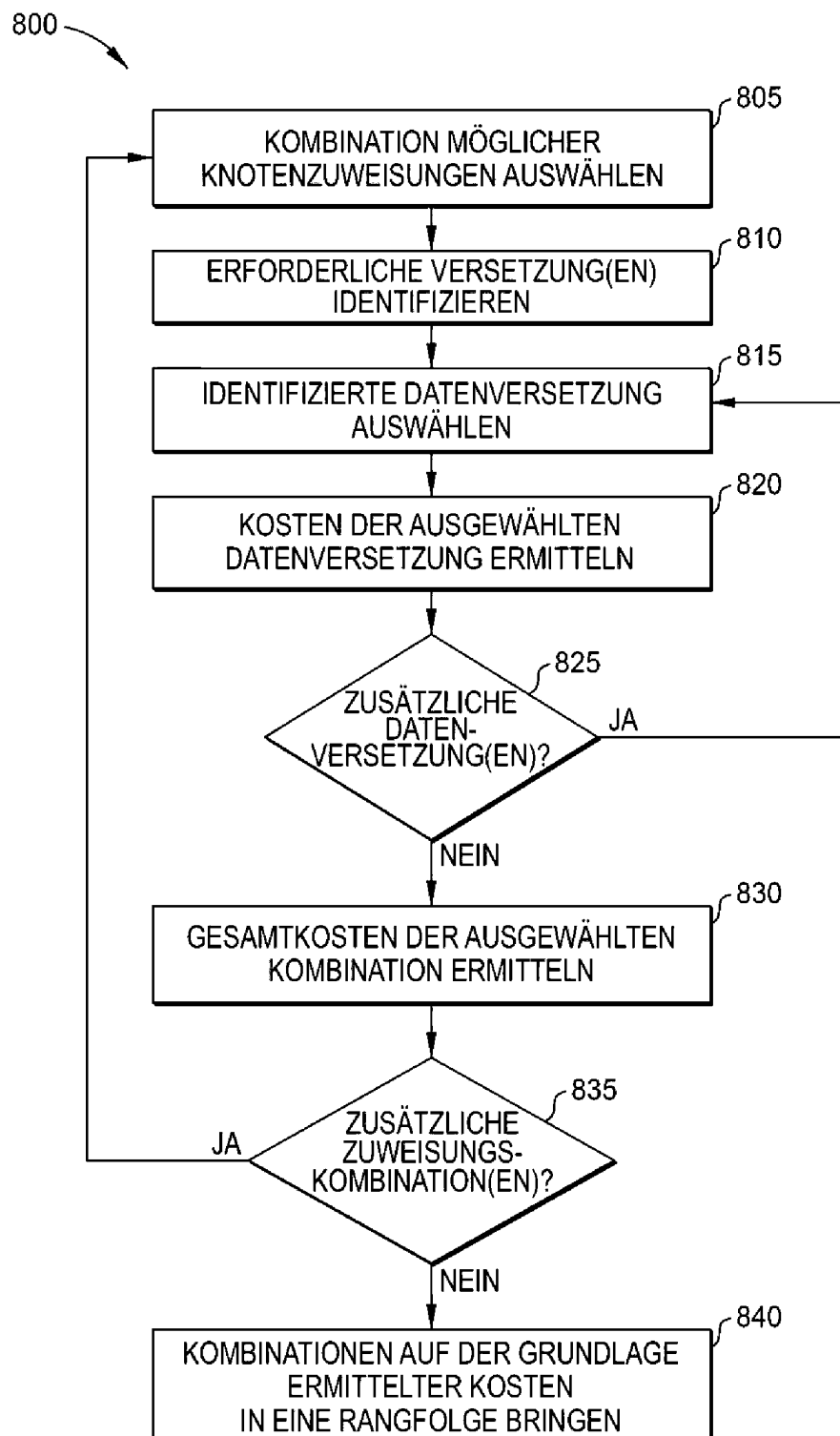


FIG. 8

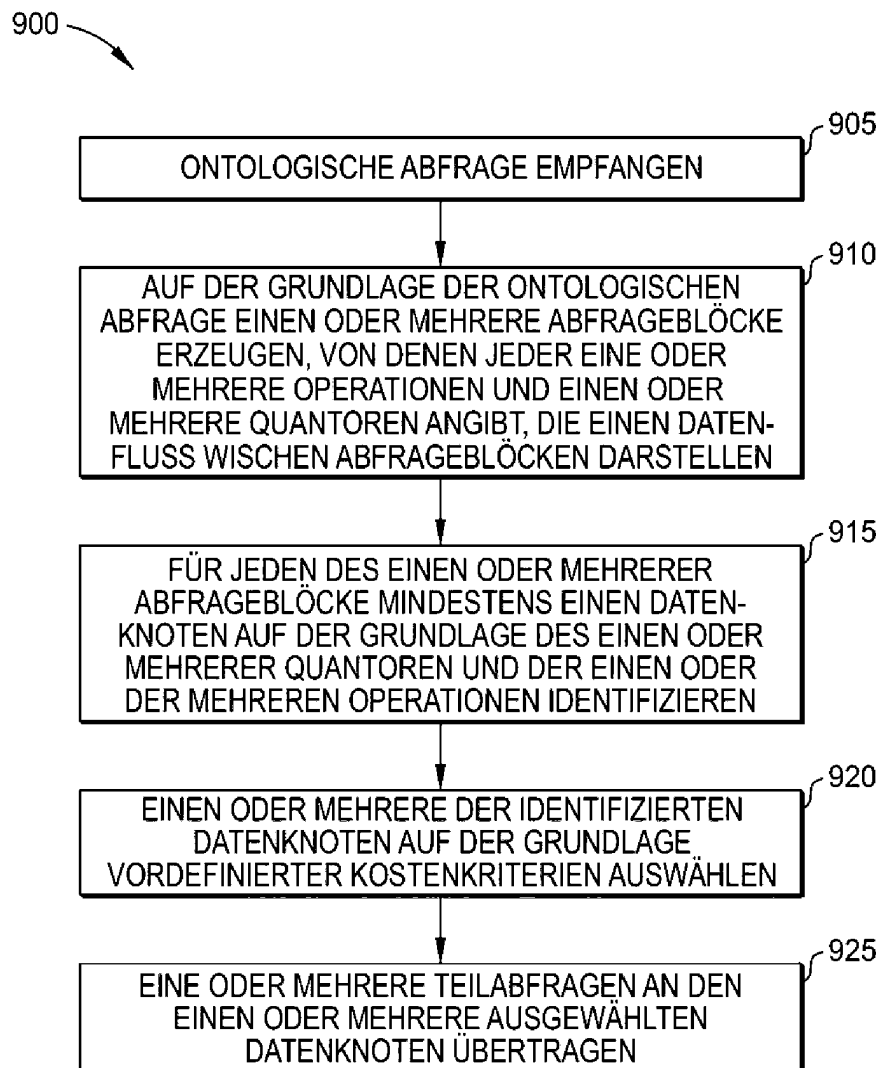


FIG. 9

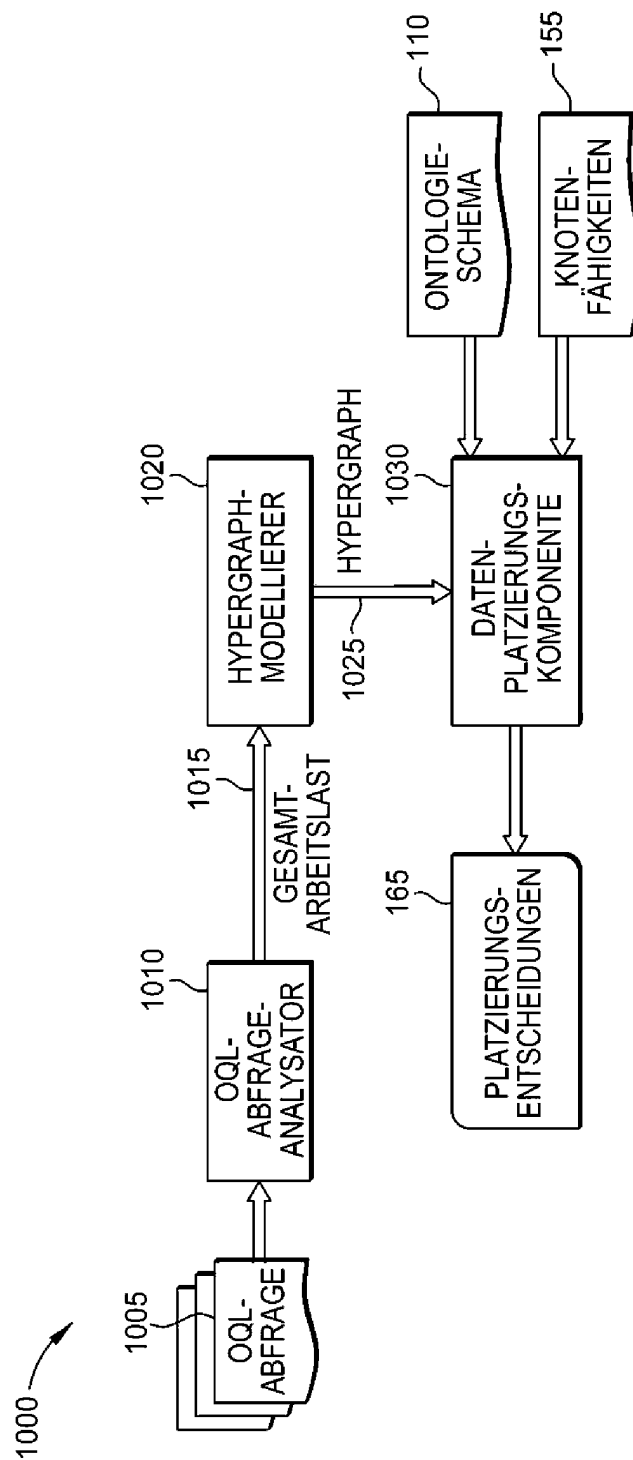


FIG. 10

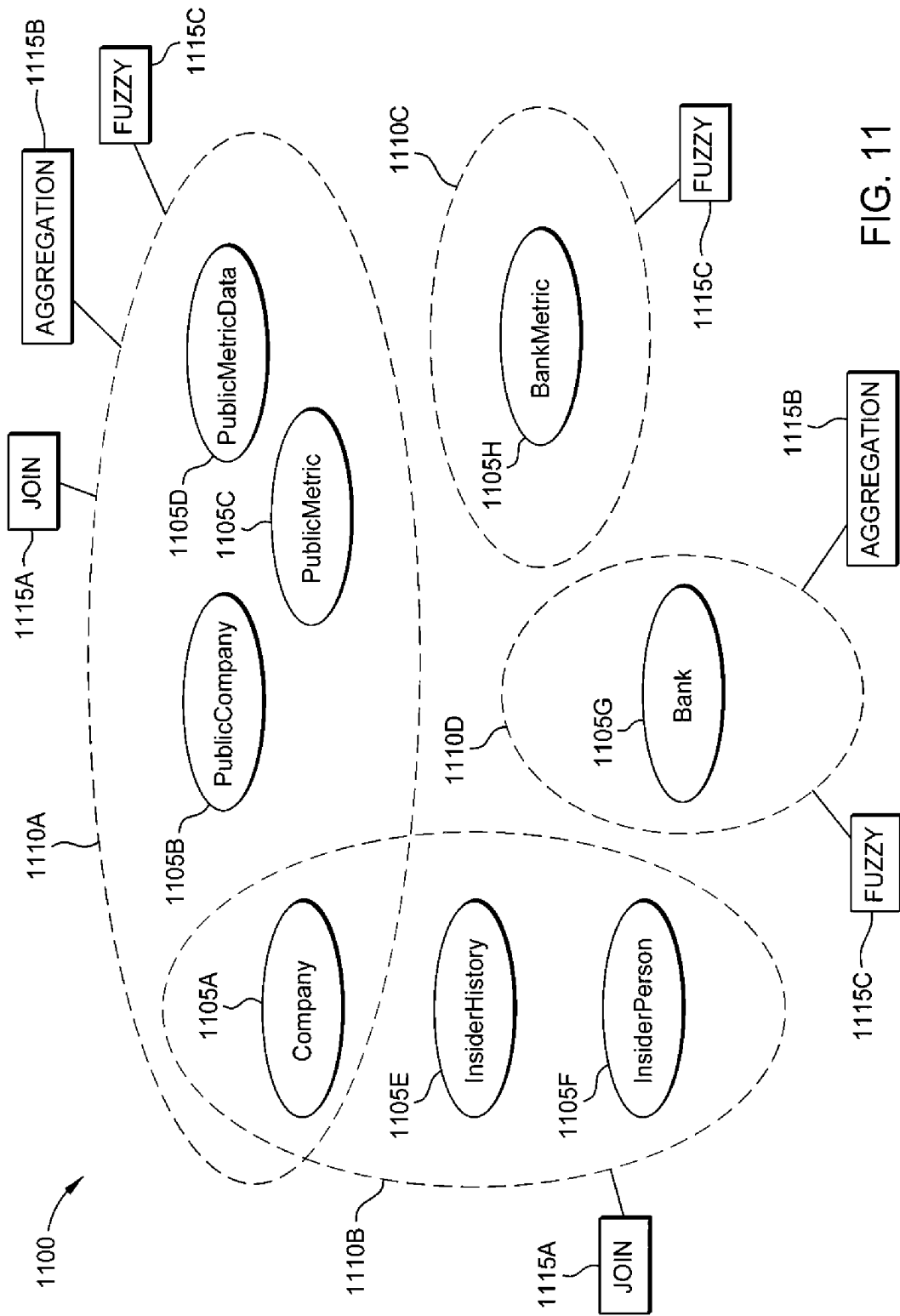


FIG. 11

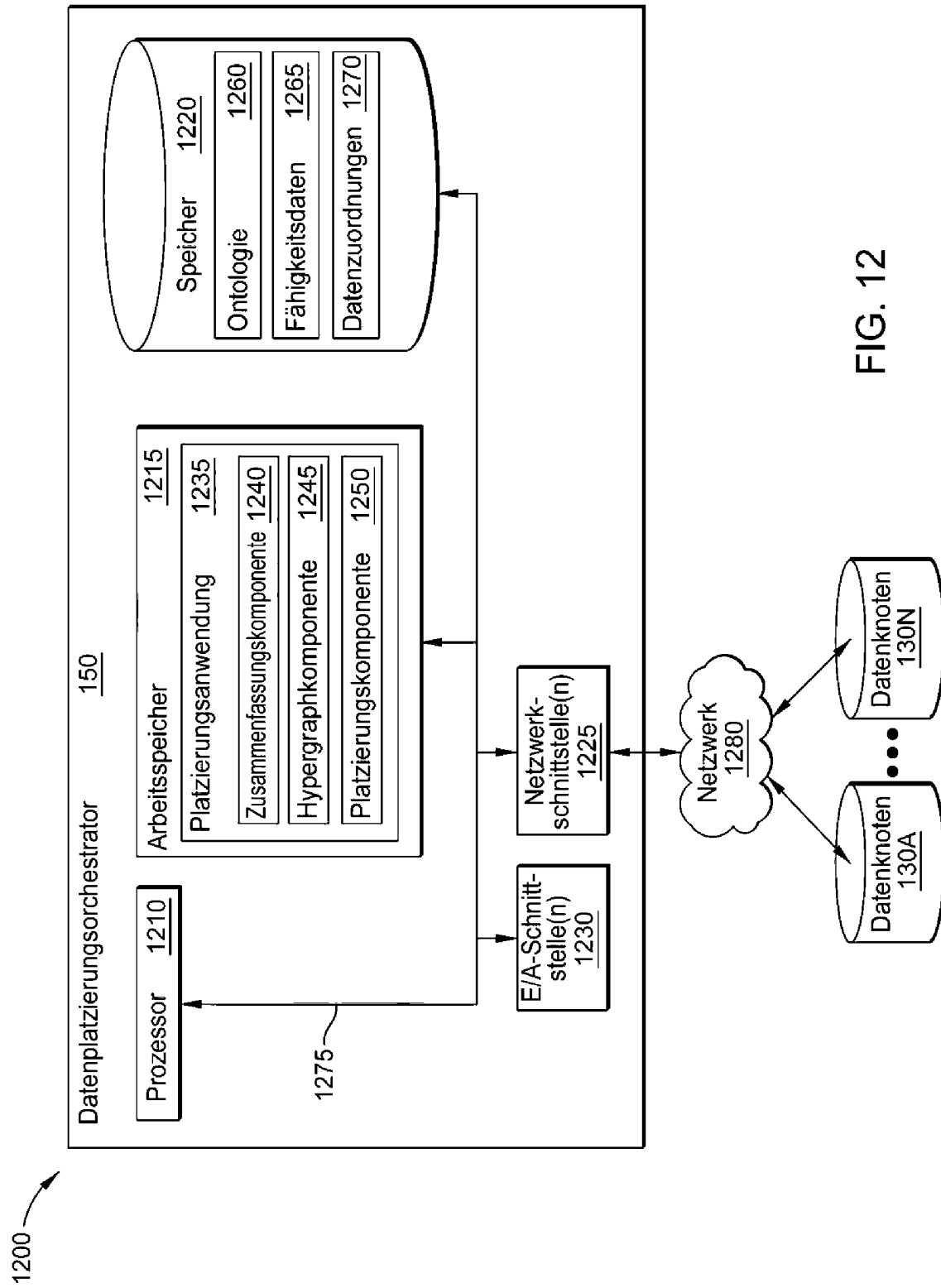


FIG. 12

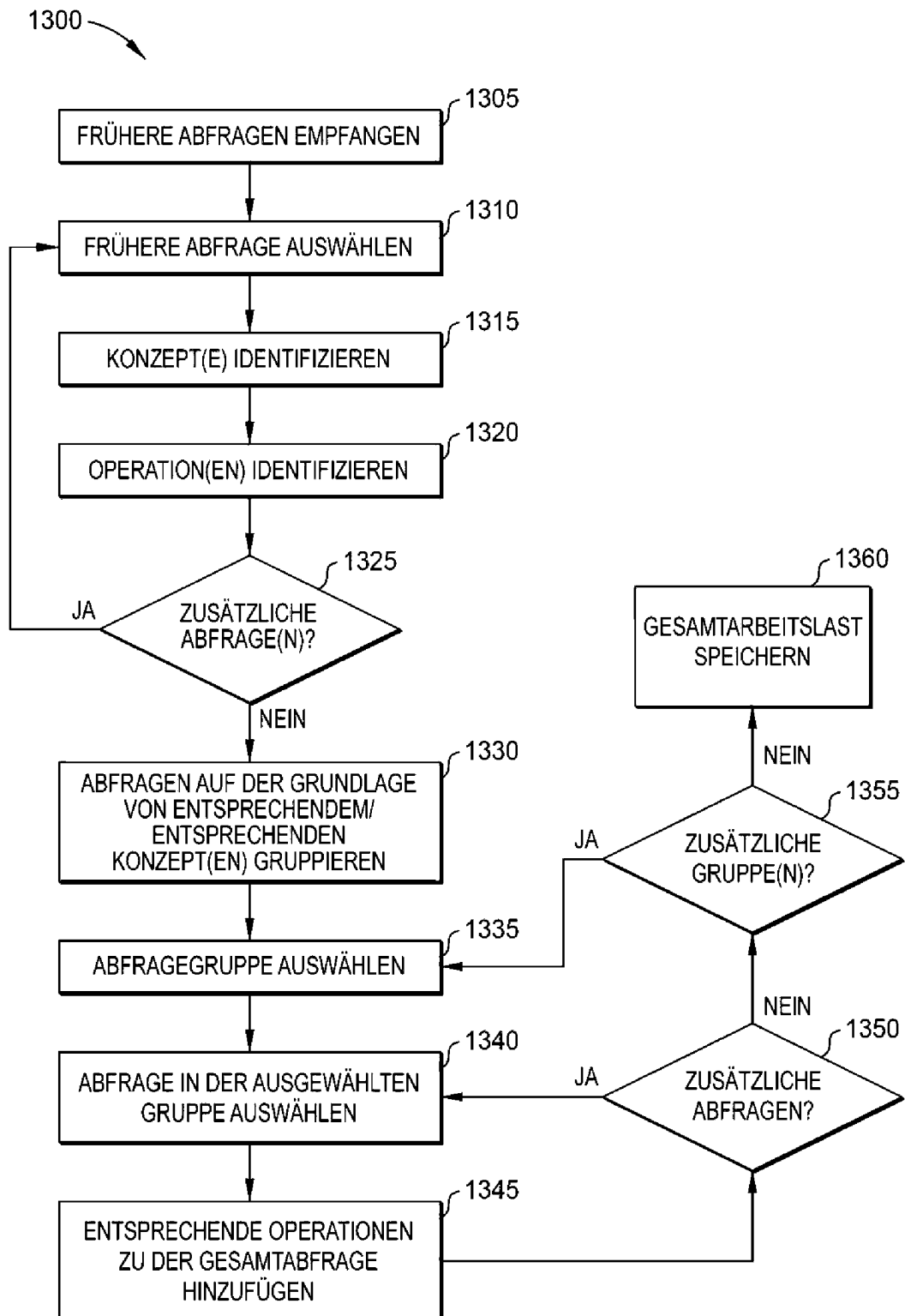


FIG. 13

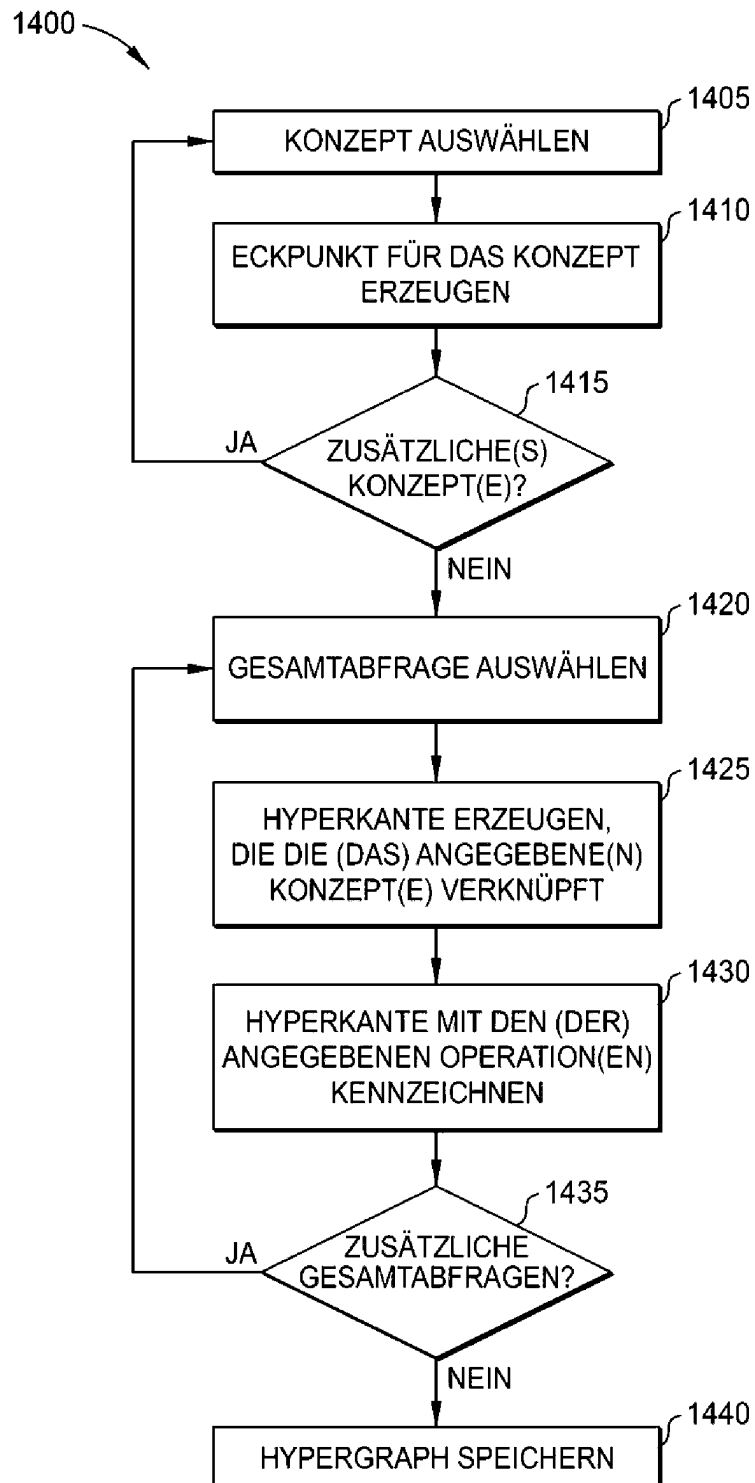


FIG. 14

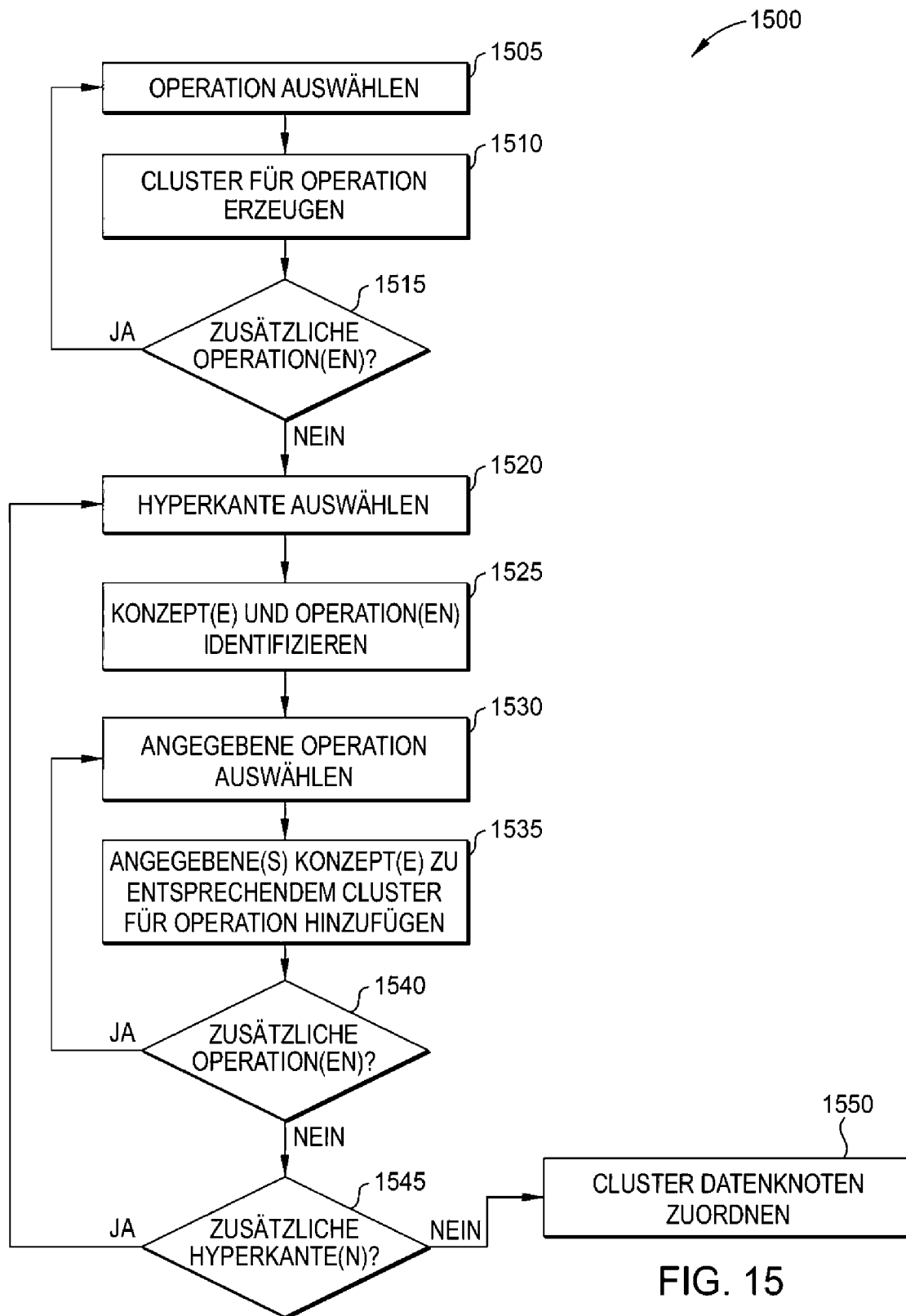


FIG. 15

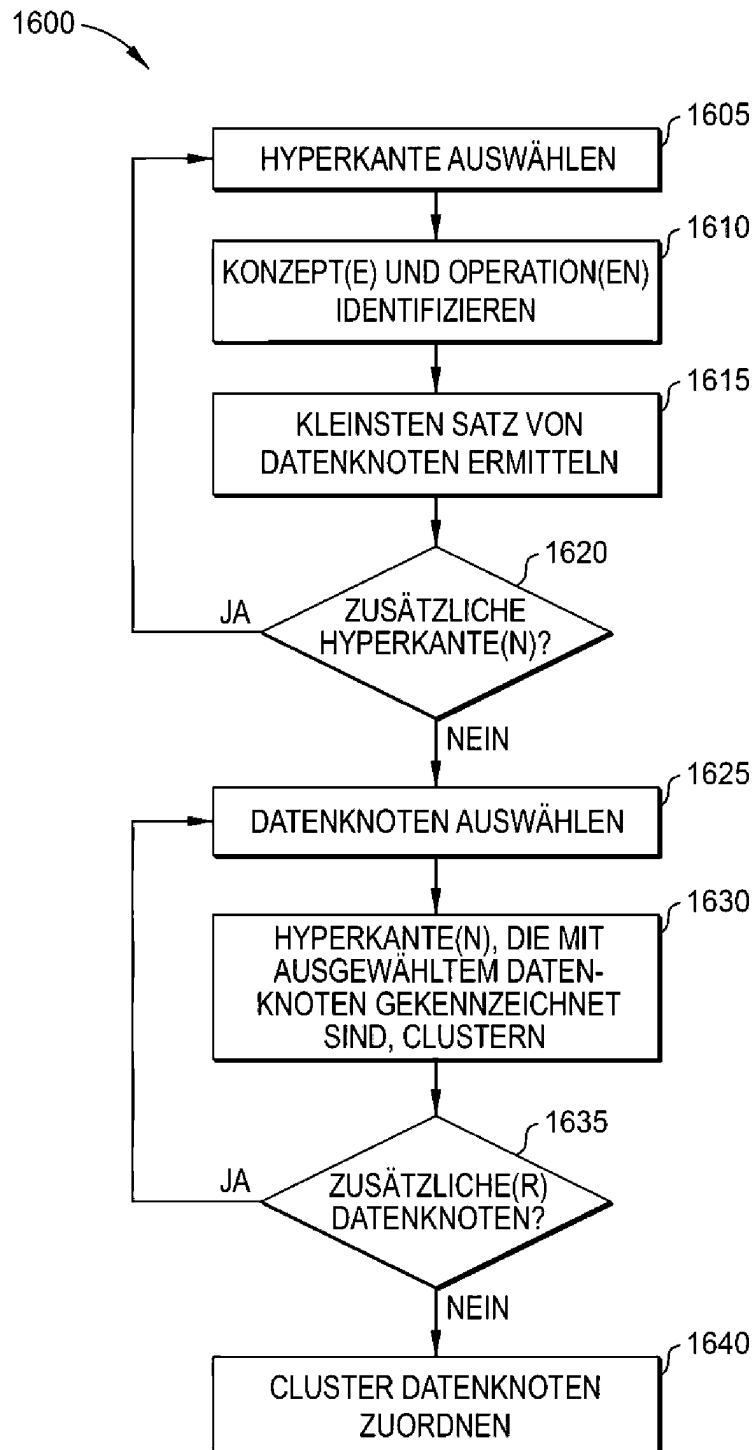


FIG. 16

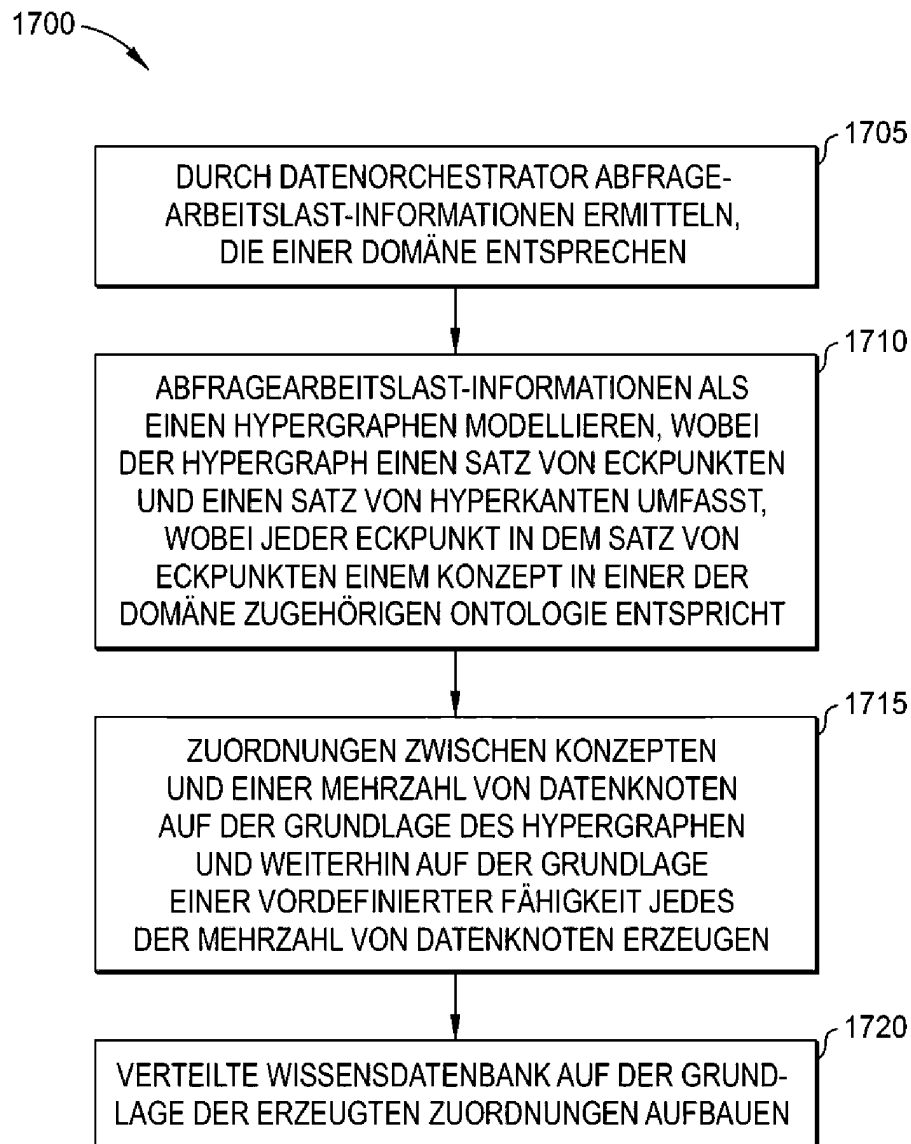


FIG. 17