(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2015-99586
(P2015-99586A)
(43)公開日　平成27年5月28日(2015.5.28)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 17/30	(2006.01)	G06F	17/30	350C	
G06F 12/00	(2006.01)	G06F	17/30	110C	
		G06F	12/00	545A	
		G06F	17/30	210D	

審査請求 未請求　請求項の数 14　ＯＬ　（全 26 頁）

(21)出願番号　特願2014-207897 (P2014-207897)
(22)出願日　平成26年10月9日 (2014.10.9)
(31)優先権主張番号　13193377.2
(32)優先日　平成25年11月18日 (2013.11.18)
(33)優先権主張国　欧州特許庁 (EP)

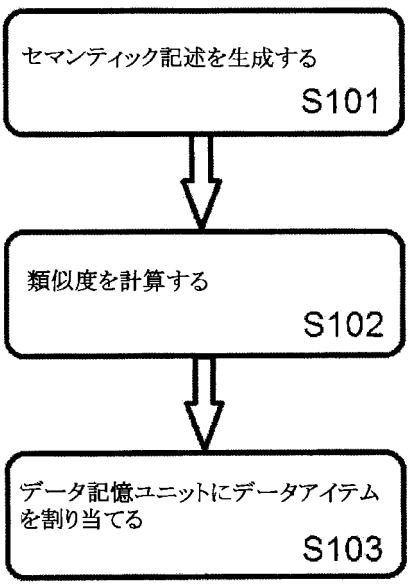
(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号
(74)代理人　100107766
弁理士　伊東　忠重
(74)代理人　100070150
弁理士　伊東　忠彦
(74)代理人　100192636
弁理士　加藤　隆夫
(72)発明者　リー・ヴィヴィアン
イギリス国，アールジー４２　２キュービー，ブラックネル　バークシャー，ハウエル　クローズ，４番

最終頁に続く

(54)【発明の名称】データ集約のためのシステム、装置、プログラム、及び方法

(57)【要約】
【課題】　実施形態は、複数のデータ記憶ユニットの間にデータアイテムを分散する方法、装置、プログラム、及びシステムを提供する。
【解決手段】　データアイテムは、複数のデータソースからのデータの集約である。当該方法は、複数のデータソースの各々のセマンティック記述を生成するステップと、前記複数のデータソースからの各データソース対について、前記データソース対のセマンティック記述間の類似度を計算するステップと、割り当て中のデータアイテムのデータソースと前記データ記憶ユニットに既に割り当てられたデータアイテムの該又は各データソースとの間の計算された類似度に依存して、データ記憶ユニットにデータアイテムを割り当てるステップと、を有する。
【選択図】　図１

本発明を具現化する処理のフローを示す図



【特許請求の範囲】
【請求項１】
複数のデータ記憶ユニットの間にデータアイテムを分散する方法であって、前記データアイテムは複数のデータソースからのデータの集約であり、前記方法は、
前記複数のデータソースの各々のセマンティック記述を生成するステップと、
前記複数のデータソースからの各データソース対について、前記データソース対の前記セマンティック記述間の類似性を計算するステップと、
割り当て中のデータアイテムのデータソースと前記データ記憶ユニットに既に割り当てられたデータアイテムの各データソースとの間の前記計算された類似度に依存して、データ記憶ユニットにデータアイテムを割り当てるステップと、
を有する方法。
【請求項２】
前記データソースの各々のセマンティック記述を生成するステップは、前記データソースから重み付き用語のリストとして最重要用語を抽出するステップであって、前記重み付き用語のリストは、前記生成されたセマンティック記述である、請求項１に記載の方法。
【請求項３】
用語頻度方法により、前記データソース内の用語の中の前記最重要用語は識別され、及び前記抽出された最重要用語の各々の属性となる前記重みは計算される、請求項２に記載の方法。
【請求項４】
前記類似度は、前記データソース対の前記生成されたセマンティック記述間のコサイン類似性を計算することにより得られる値である、請求項３に記載の方法。
【請求項５】
前記計算された類似度に依存してデータ記憶ユニットにデータアイテムを割り当てるステップは、割り当てられるべきデータアイテムを、前記割り当てられるべきデータアイテムのデータソースに対して最も高い類似度を有するデータソースからのデータアイテムを最も高い割合で格納するデータ記憶ユニットに、割り当てるステップを有する、請求項１に記載の方法。
【請求項６】
前記計算された類似度に依存してデータ記憶ユニットにデータアイテムを割り当てるステップは、同じデータソースからのデータアイテムのグループを、前記割り当てられるべきデータアイテムのデータソースに対して最も高い類似度を有するデータソースからのデータアイテムを最も高い割合で格納し及び前記データアイテムのグループのための十分な記憶空間を有するデータ記憶ユニットに、割り当てるステップを有する、請求項５に記載の方法。
【請求項７】
各データソースについて、前記データソースからの１又は複数のデータアイテムを格納している各データ記憶ユニットの識別子と、各識別されたデータ記憶ユニットに格納されたデータソースからのデータアイテムの割合の指示と、のレコードを維持するステップ、
を更に有する請求項５に記載の方法。
【請求項８】
前記データアイテムは、統一データフォーマットで格納され、任意で、前記統一データフォーマットはＲＤＦトリプルフォーマットである、請求項１に記載の方法。
【請求項９】
データソースからデータを読み出し、及び前記読み出したデータを前記統一データフォーマットを有するデータアイテムとして格納するために準備するステップ、
を更に有する請求項１に記載の方法。
【請求項１０】
割り当て中のデータアイテムを受信するために選択されたデータ記憶ユニットが既に２つのデータソースからのデータアイテムを格納し、且つ前記２つのデータソースのセマン

ティック記述間の類似度が前記データソースの各々及び前記割り当て中のデータアイテムのデータソースのセマンティック記述間の類似度のうち高い方より小さい場合、前記２つのデータソースのうちの、割り当て中のデータアイテムのデータソースに対して低い類似度を有する１つのデータソースからのデータアイテムは、前記データ記憶ユニットから削除され他の場所に割り当てられる、請求項１に記載の方法。

【請求項１１】

前記データアイテムは、それぞれ、前記複数のデータソースの間で個々のデータアイテムのデータソースを識別する識別子を有する、請求項１に記載の方法。

【請求項１２】

複数のデータ記憶ユニットの間にデータアイテムを分散する装置であって、前記データアイテムは複数のデータソースからのデータの集約であり、前記装置は、

前記複数のデータソースの各々のセマンティック記述を生成するよう構成される記述生成モジュールと、

前記複数のデータソースからの各データソース対について、前記データソース対の前記セマンティック記述間の類似性を計算するよう構成される類似性計算モジュールと、

割り当て中のデータアイテムのデータソースと前記データ記憶ユニットに既に割り当てられたデータアイテムの各データソースとの間の前記計算された類似度に依存して、データ記憶ユニットにデータアイテムを割り当てるよう構成される割り当てモジュールと、

を有する装置。

【請求項１３】

請求項１２に記載の装置と、

前記複数のデータ記憶ユニットと、

を有するデータ記憶システム。

【請求項１４】

コンピュータ装置により実行されると、前記コンピュータ装置に、複数のデータ記憶ユニットの間にデータアイテムを分散する処理を実行させるコンピュータプログラムであって、前記データアイテムは複数のデータソースからのデータの集約であり、前記処理は、

前記複数のデータソースの各々のセマンティック記述を生成するステップと、

前記複数のデータソースからの各データソース対について、前記データソース対の前記セマンティック記述間の類似性を計算するステップと、

割り当て中のデータアイテムのデータソースと前記データ記憶ユニットに既に割り当てられたデータアイテムの各データソースとの間の前記計算された類似度に依存して、データ記憶ユニットにデータアイテムを割り当てるステップと、

を有する、コンピュータプログラム。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、セマンティックウェブにおける用途の分野に属し、特に、異なる異種データソースからのデータの、統一データビューを提供するデータベースへの集約に関する。

【背景技術】

【０００２】

現在のビッグデータ時代では、データ集約は、データ分析において極めて重要な役割を果たす。データ集約は、ビッグデータ分析ツールが、種々のフォーマットの異種ソースからデータを集めること及びそれらのデータを１つの統一ビューに統合することを助ける。データ集約は、多数の接続されたサブプロセスを有する。例えば、データ集約は、外部データソースからデータを読み出すステップを含み得る。外部データソースからのデータフォーマットは構造化から準構造化を通じ、非構造化データまでに渡り得る。データ集約は、非統一データ型へのデータフォーマット変換を含むデータを処理するステップを更に有しても良い。例えば、ＲＤＦデータ型は、ビッグデータアプリケーションにおいて柔軟なデータ構造を提供する。最後に、データ集約方法は、フォーマット化データをデータ記憶

装置に書き込むステップを有しても良い。
【発明の概要】
【発明が解決しようとする課題】
【０００３】
本発明は、複数のデータ記憶ユニットの間にデータアイテムを分散する方法、装置、プログラム、及びシステムを提供する。
【課題を解決するための手段】
【０００４】
現在、データを書き込むために利用可能な技術は、データ記憶ユニットへのデータアイテムのランダム割り当て、及び／又はデータが記憶装置に書き込まれた後のデータの再割り当てを含む。ランダム書き込みは、実装が容易であるが、結果として生じるデータアイテムの拡散は、データベースクエリの効率的な処理に貢献しない。適応型ロケータ技術は、クエリ性能を向上するが、データ使用によりトリガされる再割り当てに依存する。
【０００５】
実施形態は、複数のデータ記憶ユニットの間にデータアイテムを分散する方法を含み、前記データアイテムは複数のデータソースからのデータの集約であり、前記方法は、前記複数のデータソースの各々のセマンティック記述を生成するステップと、前記複数のデータソースからの各データソース対について、前記データソース対の前記セマンティック記述間の類似性を計算するステップと、割り当て中のデータアイテムのデータソースと前記データ記憶ユニットに既に割り当てられたデータアイテムの各データソースとの間の前記計算された類似度に依存して、データ記憶ユニットにデータアイテムを割り当てるステップと、を有する。
【０００６】
データ集約は、ビッグデータ分析において重要な役割を果たす。エンドユーザ及びアプリケーションに全ての利用可能データのより包括的なビューを構築するために、データ集約は、別個のソースからの外部データの収集と、ユーザに統一ビューを提供するデータアイテムとしての（例えば、仮想データベースとしての）それらの格納と、を有する処理である。しかしながら、最初にデータアイテムをデータ記憶装置に書き込むとき、データの局所性が考慮されない場合、クエリ応答時間は、特にデータベースが急速に増大している場合、ボトルネックになり、格納されたデータアイテムへのアクセスを禁止し得る。本発明の実施形態は、外部データ及び既存データのデータ記憶ユニットへの割り当てを改善し、新しいデータが集約されたデータセットに含まれるときでも、記憶ユニットを交差するグラフ走査を最小化でき、クエリがデータ空間全体に亘って効率的に評価できるようにする。
【０００７】
有利なことに、本発明の実施形態は、データベースクエリの効率を向上するメカニズムを提供する（データベースは、データ記憶ユニットに格納されたデータアイテムである）。データソース間の類似性に依存して、データアイテムをデータ記憶ユニットに割り当てるステップは、一緒に問い合わせられる特定の可能性を有するデータアイテムを同じデータ記憶ユニット上にグループ化するための基礎を提供し、したがってデータ記憶ユニットの境界を跨るグラフ走査を低減し、したがって、クエリ処理の効率を向上する。さらに、実施形態は、データアイテムが一緒に問い合わせられる可能性を評価するためにデータアイテムのアクセス履歴に頼らないので、当該方法は、データベースに新たに追加されるデータアイテムにも適用できる。
【０００８】
データソースは、特定のＵＲＬ又はＵＲＬのグループ（つまり、データソースは１又は複数のプレフィックスにより指定され、該１又は複数のプレフィックスを含むＵＲＬはデータソースに含まれると考えられる）、文書、文書コーパス、データ記憶場所若しくはアドレス（又はそれらのグループ）、又は特定の他の形式の読み取り可能な符号化された情報、のうちの１又は任意の組合せであっても良い。データ又はデータアイテムがデータソ

ースからのものとして参照される場合、データ又はデータアイテムはデータソースから読み出された情報を符号化していると考えられても良い。データソースは、外部データソースであっても良く、データの起源がデータベース又は実施形態のデータ記憶システムの外部にあることを示す。複数のデータ記憶ユニットの間に割り当てられ及び格納されるデータアイテムは、データベース又は集約データベースとして参照されても良い。

【０００９】

データアイテムは、符号化された形式のドメインに関する知識であり、１より多いデータソースからのドメインに関する知識を含む。データアイテムは、同じデータソースからのデータアイテムは、互いに時間的に隣接するデータ記憶ユニットに割り当てられ及び書き込まれるように、データソース毎にデータ記憶ユニットに割り当てられ及び書き込まれても良い。データアイテムは、データ記憶ユニットにグループ化されて割り当てられ及び書き込まれても良い。ここで、各グループは、異なる個々のデータソースからのデータアイテムを表す。例えば、データアイテムは、最初にデータ記憶ユニットに書き込まれた新しいデータアイテムであっても良い。データ記憶ユニットに格納されるデータアイテムは、データベースを形成する。割り当てられるべきデータアイテムは、データベースにとって新しいデータアイテムであっても良く、したがって、該データアイテムが割り当てられるデータ記憶ユニットは該データアイテムの最初のデータ記憶ユニット位置である。

【００１０】

割り当ては、新しいデータアイテムが最初にデータ記憶ユニットのうちの１つに書き込まれる前に（又はその処理の一部として）、該新しいデータアイテムに対して実行されても良い。

【００１１】

生成、計算、及び割り当ては、必ずしも固定的順序で実行される又はそれぞれデータアイテムがデータ記憶ユニットに割り当てられる度に実行される必要はない。例えば、生成及び計算は、新しいデータソースからのデータアイテムが割り当てられる度に実行されても良い。一方、データアイテムが、データアイテムが既に割り当てられている又はデータ記憶ユニットに既に格納されているデータソースから割り当てられるとき、セマンティック記述は既に生成され、及び類似度は既に計算されているので、これらのステップを繰り返す必要はない。データソースは、該データソースからのいかなるデータアイテムもデータ記憶ユニットに既に割り当てられていない又は格納されていない場合、新しいデータソースであると考えられても良い。

【００１２】

方法は、各データソースについて、データソース識別情報及びセマンティック記述を含む情報の登録を維持するステップを更に有しても良い。情報は、ソースＵＲＬ、タイトル、及びデータ型を更に有しても良い。登録は、メタデータレジストリであっても良い。代替で又は追加で、及び登録／メタデータレジストリ若しくはその他の一部として、方法は、計算された類似度のレコードを維持するステップを有しても良い。例えば、レコードは、各データソースが行及び列に対応する正方行列の形式であっても良い。したがって、行列内の各エントリは、該エントリの行に対応するデータソースと該エントリの列に対応するデータソースとの間の類似度に対応する（必ずしも異なるデータソースである必要はない）。

【００１３】

データアイテムは、データアイテムに埋め込まれる個々のデータソースの識別子を有する形式であっても良い。識別子は、複数のデータソースの間でデータソースを識別するために十分である。識別子は、メタデータとして含まれ、又は特定の他の方法でデータアイテム内のデータから導出可能であっても良い。

【００１４】

複数のデータ記憶ユニットは、少なくとも、それらが全てエントリサーバ又はエントリノードのような単一のアクセスポイントから通信可能であるよう相互接続される。しかしながら、データ記憶ユニットは、それらが異なる管理ユニットを有する場合、異なり、し

たがって、それらはデータアイテムの場所を特定するために別個に問い合わせられなければならない。データ記憶ユニットは、それぞれ、異なる記憶サーバであっても良く、又は１又は複数のサーバ内の異なる記憶ユニットであり、別個の管理ユニットを介してアクセス可能であっても良い。各データ記憶ユニットは、複数のデータアイテムのうちの一部を格納する。データ記憶ユニットは、ハードディスクのようなディスクであっても良い。

【００１５】

方法は、データアイテムを、それらが割り当てられたデータ記憶ユニットに書き込むステップを更に有しても良い。データアイテムは、割り当ての直後に書き込まれても良い。或いは、データアイテムのグループ若しくは多数のデータアイテムの割り当てが実行され、次にデータアイテムが纏めて書き込まれても良い。

【００１６】

任意で、前記データソースの各々のセマンティック記述を生成するステップは、前記データソースから重み付き用語のリストとして最重要用語を抽出するステップであって、前記重み付き用語のリストは、前記生成されたセマンティック記述である。

【００１７】

最重要用語（用語はワード又はフレーズである）を抽出するステップは、データソース内の用語の重要性を表現する数値スコアを計算するステップと、テキスト記述内で識別された最重要用語の各々をそれらの個々の数値スコアと共に含む用語のリスト又はベクトルをコンパイルするステップと、を有しても良い。関連する数値スコアは、個々の用語の重みを表す。

【００１８】

有利なことに、ベクトル又はリストは、セマンティック記述を格納し及び評価するフォーマットを表す。該フォーマットは、記憶を容易にし、高速且つ正確な類似度計算を可能にする。例えば、ベクトルは、幾何学的に互いに評価できる。実施形態は、コサイン比較を用いて２つの用語のベクトルを比較しても良い。このようなベクトルを比較する例示的な手順は、コサイン比較を利用する潜在的セマンティック分析である。

【００１９】

実施形態は、ベクトル又はリスト内の各用語のデータソースのセマンティック記述に対する関連を定量化するために、抽出された最重要用語の各々を数値的に又は特定の他のメトリックを用いてスコア付けするステップを有しても良い。用語の重要性を定量化することは、用語を重み付けすることを可能にする。したがって、それら用語の、類似度計算のためにそれらが比較される別のセマンティック記述のベクトル又はリストからの用語に対する類似性の程度は、評価中の２つの用語の結合重みに依存して、２つの概念間の類似度の全体評価にとって多かれ少なかれ重要性を保有する。

【００２０】

任意で、データソース内の用語の中の最重要用語は、情報検索技術により識別される。例えば、用語頻度方法により、最重要用語は識別され、抽出された最重要用語の各々の属性となる重みが計算されても良い。

【００２１】

用語頻度方法は、文書又はデータソース内の用語の重要性を評価するために計算的に安価な技術である。実施形態は、自然言語処理のような特定の前処理を有しても良い。自然言語処理は、トークン化及び語幹解釈を有しても良い。

【００２２】

用語の数値スコア付けの任意の実装として、数値スコアは、用語頻度－逆文書頻度評価に基づいても良い。ここで、用語の重要性は、集合的な複数のデータソースのようなより大きな文書コーパス内での該用語の使用頻度に対するデータソース内での該用語の使用頻度を測定することにより評価される。

【００２３】

用語頻度－逆文書頻度は、実施するのが容易であり、用語の有用性を計算する方法を効率的に処理する。

【００２４】
コンパイルされると、セマンティック記述は、幾何学的比較のような比較により比較されても良い。
【００２５】
有利なことに、２つのセマンティック記述の幾何学的比較は、２つのデータソース間の類似度を評価する計算効率の良い方法を提供する。セマンティック記述対の幾何学的比較は、一方のセマンティック記述内の各用語の他方のセマンティック記述内の各用語との比較を有しても良い。この比較は、場合によっては２つの用語の個々のセマンティック記述内の該２つの用語の重要性重み付けに従って重み付けされる、該２つの用語間の類似性を定量化するスコアを生成する。次に、２つのセマンティック記述間の類似性の全体的評価は、例えば個々の用語の比較の結果を加算し、平均化し、重みを平均化し、又は結合する特定の他の処理により計算され得る。
【００２６】
特定の実装では、用語頻度は、データソース内の最重要用語を識別し及び重みを属性とする方法として用いられても良い。ここで、最重要用語は、セマンティック記述を形成する。単純な用語頻度は、例えば、外部データソースが個々に組み込まれ、したがって複数の文書要約技術が有効でない実装において適した方法である。さらに、外部データソースは大きく変化する可能性があり、（複数のスキャンラウンドを必要とする技術のような）より複雑な要約技術が全体的システム性能に悪影響を与えるかも知れない。本発明の実施形態では、データが外部データソースから読み出され集約データベースの統一フォーマットに変換される間に、セマンティック記述生成が実行されても良い。したがって、これは、システム性能の有意な低下を生じない。類似性計算は、より多くの外部データソースがシステムに入るとき増大すが、この増大は線形である。したがって、計算の複雑性は同じままであり、コストの増大はデータ変換処理と調和しないものではなく、ボトルネックは回避される。
【００２７】
類似度は、データソース対の生成されたセマンティック記述間のコサイン類似性を計算することにより得られる値であっても良い。コサイン類似性は、各用語比較について０と１の間のスコアを生成し、０は完全に異なり、１は同じである。コサイン類似性は、本発明の方法の部分を形成する幾何学的比較の一例である。
【００２８】
任意で、前記計算された類似度に依存してデータ記憶ユニットにデータアイテムを割り当てるステップは、割り当てられるべきデータアイテムを、前記割り当てられるべきデータアイテムのデータソースに対して最も高い類似度を有するデータソースからのデータアイテムを最も高い割合で格納するデータ記憶ユニットに、割り当てるステップを有する。
【００２９】
有利なことに、類似度計算は、割り当てられるべきアイテムのデータソースに最も類似するデータソースを識別するために、単純な方法で利用できる。意味的に類似するデータソースからのデータアイテムは同じグラフ操作（例えば、グラフ走査）でアクセス又は問い合わせられる可能性が高いと論理的に想定できるので、データ記憶ユニット境界を跨るグラス走査の数は、意味的に類似するデータソースからのデータアイテムを同じデータ記憶ユニットに格納することにより低減できる。
【００３０】
データアイテムが、データアイテムが既にデータベースに存在するデータソースからデータベースに追加される状況では、データアイテムは、デフォルトの記憶位置と同じデータソースからのデータアイテムを格納するデータ記憶ユニットに追加されても良い。周期的に又はデータソースからの所定数のデータアイテムが追加された後に、データソースのセマンティック記述は再び生成され、相応して類似性計算は更新されても良い。
【００３１】
前記計算された類似度に依存してデータ記憶ユニットにデータアイテムを割り当てるス

テップは、同じデータソースからのデータアイテムのグループを、前記割り当てられるべきデータアイテムのデータソースに対して最も高い類似度を有するデータソースからのデータアイテムを最も高い割合で格納し及び前記データアイテムのグループのための十分な記憶空間を有するデータ記憶ユニットに、割り当てるステップを有しても良い。

【００３２】

同じデータソースからのデータアイテム間のグループ化を保存するために、新しいデータソース（又はより正確には、新しいデータソースからのデータアイテム）がデータベースに追加されるとき、同じデータソースからのデータアイテムの一部又は全部は、一緒に割り当てられる。記憶空間は、データアイテムのグループを割り当てるべきデータ記憶ユニットを選択する際に検討材料になっても良い。上述のような実施形態は、同じデータソースからのデータアイテムを同一場所に配置するメカニズムを提供し、同時に、データアイテムが同じデータ記憶ユニットに格納されるデータソース間の類似性を最適化し、記憶空間を考慮に入れる。

【００３３】

実施形態は、各データソースについて、データソースからの１又は複数のデータアイテムを格納している該又は各データ記憶ユニットの識別子と、該又は各識別されたデータ記憶ユニットに格納されたデータソースからのデータアイテムの割合の指示とのレコードを維持するステップを更に有しても良い。

【００３４】

このようなレコードの維持は、特定のデータソースからのデータアイテムが位置する場所を（データ記憶ユニットの識別情報の観点から）識別するために利用され又は活用できる情報を提供する。既存のデータアイテムの再割り当ては、このようなレコード内に格納される情報に基づき実行されても良い。割合の指示は、分数、百分率、十進数、又はデータアイテムの絶対数として表されても良い（これらから、同じデータソースのレコード内の他のエントリに基づき、割合が導出できる）。

【００３５】

実施形態は、データアイテムを格納すべき統一データフォーマットを用いても良い。有利なことに、複数のデータソースからのデータの集約は、物理的位置の点だけでなくデータフォーマットの観点からも互いに区別される場合が多く、クエリ効率が良く及び管理の容易な形式でデータを再書き込みする機会を提供する。このような利益を達成するために、統一データフォーマットは有利である。データソースは、任意のフォーマットのデータを有しても良い。また、これらのデータは、セマンティック記述の生成の前に、又は本発明を実施する方法の特定の他の段階で、読み出され及び統一データフォーマットに変換されても良い。データフォーマットは、特定の位置に現れる又は特定のタグ若しくは他の識別子により識別されるデータの重要性若しくは型が全てのデータアイテムに渡り共通である点で、統一されていると考えられても良い。

【００３６】

統一データフォーマットを用いるために、実施形態は、データソースからデータを読み出し、統一データフォーマットを有するデータアイテムとして格納するために該読み出したデータを準備するために処理を実行するステップを更に有しても良い。

【００３７】

データソースは、複数のデータアイテムの統一データフォーマットと同じフォーマットでデータを格納しても良い。しかしながら、幾つかの例では、データソース内のデータのフォーマットは、統一データフォーマットと異なる。したがって、統一データフォーマットを有するデータアイテムとして記憶するためにデータソースからのデータを準備するために、特定の追加処理が必要である。このような処理は、当該方法の生成するステップ及び計算するステップに適するフォーマットであるデータソースからのデータを生じる点で、有利であり得る。

【００３８】

例示的な統一データフォーマットは、ＲＤＦトリプルフォーマットである。有利なこと

に、ＲＤＦトリプルのようなＲＤＦ（Resource Description Framework、リソース記述フレームワーク）データ型は、リソース間の有意義なリンクを提供し及びしたがって迅速に解釈できる柔軟なデータ構造を提供する。ＲＤＦは、意味的に有意義な方法でデータアイテムをマークするフレームワークを提供する。

【００３９】

関係型データベースは、データを行と列で格納する。行及び列は、データを格納する前に定める必要のあるテーブルを構成する。テーブルの定義及びこれらのテーブルに含まれるデータ間の関係は、スキーマと称される。関係型データベースは、固定スキーマを用いる。グラフデータベースは、データをノード及びアークの形式で格納することにより、関係型データベースの重要な拡張を表す。ここで、ノードはエンティティ又はインスタンスを表し、アークは任意の２個のノード間の特定種類の関係を表す。無向グラフでは、ノードＡからノードＢへのアークは、ノードＢからノードＡへのアークと同じであると考えられる。有向グラフでは、２つの方向は別のアークとして扱われる。

【００４０】

グラフデータベースは、概して２つの主な種類に分類できる広範な種類の異なるアプリケーションで用いられる。第１の種類は、知的意思決定支援及び自己学習のようなクラス記述子の大規模な集合体（「知識ベースアプリケーション」と称される）を有する複雑な知識ベースシステムを有する。第２の種類は、社会的データ及びビジネスインテリジェンスのようなトランザクションデータに対するグラフ検索の実行を含むアプリケーション（「トランザクションデータアプリケーション」と称される）を有する。多くのアプリケーションは、両方の種類を表し得る。しかしながら、大部分のアプリケーションは、主に知識ベース又はトランザクションデータアプリケーションのいずれかで特徴付けられ得る。グラフデータベースは、種々の分野の膨大な構造化又は非構造化データを格納できる大規模な「意味ネットワーク」を維持するために用いることができる。意味ネットワークは、知識表現の形式として用いられ、コンセプトを表すノード及びコンセプト間の意味関係を表すアークを有する有向グラフである。

【００４１】

幾つかの種類のグラフ表現がある。グラフデータは、多次元アレイとして又は他のシンボルにリンク付けされたシンボルとしてメモリに格納されても良い。別の形式のグラフ表現は、各々指定された種類のオブジェクトの有限シーケンス又は順序付きリストである「タプル」の使用である。ｎ個のオブジェクトを含むタプルは、「ｎタプル」として知られる。ここで、ｎは零より大きい任意の非負整数である。長さ２のタプル（２タプル）は、通常、ペアと呼ばれる。３タプルはトリプルと呼ばれ、４タプルはクワドラプルと呼ばれ、以降同様である。

【００４２】

ＲＤＦ（Resource Description Framework）は、概念記述又は意味ネットワークの標準である情報のモデル化のための一般的方法である。膨大な量のデータを格納し検索可能にするために、データは複数のサーバに保持されなければならない。データの追加、削除及び検索は、分散システムのために特注されたアルゴリズム及びデータ構造を用いて協調的方法で行われなければならない。コンピュータ的に効率的にデータの検索、保守及び操作を可能にするような方法でグラフデータを格納することが望ましい。

【００４３】

任意で、トリプルは、ＲＤＦ（Resource Description Framework）トリプルであっても良い。本願明細書を通じて、「ＲＤＦトリプル」への特定の参照が行われるとき、それはＲＤＦ標準に準拠するトリプルの例示的形式であることが理解されるべきである。さらに、「トリプル」への参照は、問題のトリプルがＲＤＦトリプルである可能性を有する。同様に、本願明細書のいずれかの箇所で議論されるＲＤＦプロセッサは、ＡＰＩラッパと格納されたデータアイテムとの間の相互作用のために用いられるプロセッサの例である。

【００４４】

ＲＤＦ（Resource Description Framework）は、概念記述又は意味ネットワークの標

準である情報のモデル化のための一般的方法である。意味ネットワークにおける情報のモデル化の標準化は、共通の意味ネットワークで動作するアプリケーション間の相互接続性を可能にする。ＲＤＦは、ＲＤＦスキーマ（ＲＤＦＳ）をＲＤＦ内の語彙を記述するための言語として提供することにより、一義的な形式意味論と共に語彙を保持する。

【００４５】

トリプルは、グラフデータを複数の主語－述語－目的語の表現として特徴付けることにより、グラフデータのエンコードを提供する。この文脈では、主語及び述語は、グラフデータのグラフノードであり、オブジェクト、インスタンス又はコンセプトのようなエンティティであり、述語は、主語と目的語の間の関係の表現である。述語は、目的語への特定の種類のリンクを提供することにより、主語に関する何かを断言する。例えば、主語は、（例えば、ＵＲＩを介して）ウェブリソースを示しても良く、述語はリソースの個々の特性、特徴又は状況を示し、目的語は、該特性、特徴又は状況のインスタンスを示す。言い換えると、トリプルステートメントの集合は、元来、方向性グラフデータを表す。ＲＤＦ標準は、このようなトリプルの形式化された構造を提供する。

【００４６】

任意で、トリプルの１又は複数の要素のうちの各々は（要素は、述語、目的語又は主語である）、ＵＲＩ（Uniform Resource Identifier）である。ＲＤＦ及び他のトリプルの形式は、識別するものの概念（つまり、オブジェクト、リソース又はインスタンス）を前提として、ＵＲＩのようなウェブ識別子を用い、それら識別される「もの」を簡易な特性及び特性値の観点で記述する。トリプルの観点では、そのトリプルのウェブリソースの具体化において、主語はエンティティを記述するウェブリソースを特定するＵＲＩであっても良く、述語は特性の種類（例えば、色）を特定するＵＲＩであっても良く、目的語は問題のエンティティに起因する特性の種類の特定のインスタンスを指定するＵＲＩであっても良い。ＵＲＩの使用は、トリプルに、個々の特性及び値と同様に、リソースを表すノード及びアークのグラフのようなリソースに関する簡易なステートメントを表すことを可能にする。ＲＤＦグラフは、SPARQLプロトコル及びＲＤＦクエリ言語（SPARQL）を用いて問い合わせることができる。SPARQLは、World Wide Web ConsortiumのRDF Data Access Working Group (DAWG)により標準化され、主要なセマンティックウェブ技術と考えられている。SPARQLは、クエリがトリプルのパターン、連結、分離、任意のパターンを有することを許容する。

【００４７】

実施形態では、格納されたデータは、新しいデータ、例えばデータベース内に以前に表されていない外部データソースからのデータ、の到着により効率的に除去されても良い。任意で、割り当て中のデータアイテムを受信するために選択されたデータ記憶ユニットが既に２つのデータソースからのデータアイテムを格納し、且つ前記２つのデータソースのセマンティック記述間の類似度が前記データソースの各々及び前記割り当て中のデータアイテムのデータソースのセマンティック記述間の類似度のうち高い方より小さい場合、前記２つのデータソースのうちの、割り当て中のデータアイテムのデータソースに対して低い類似度を有する１つのデータソースからのデータアイテムは、前記データ記憶ユニットから削除され他の場所に割り当てられる。

【００４８】

２つのデータソースのセマンティック記述間の類似度が、データソースの各々及び割り当てられるべきデータアイテムのデータソースのセマンティック記述間の類似度のうち高いものより低い場合、割り当て中のデータアイテムを受信するために異なるデータ記憶ユニットが選択される。

【００４９】

有利なことに、この方法でデータアイテムを再割り当てすることは、同一場所に配置されるデータアイテムのデータソース間の全体的な類似度を増大させ得る。

【００５０】

データアイテムは、データアイテムのデータソースを識別できる方法で格納された要素

を有しても良い。例えば、前記データアイテムは、それぞれ、前記複数のデータソースの間で個々のデータアイテムのデータソースを識別する識別子を有しても良い。

【００５１】

有利なことに、アイテムのデータソースを識別するステップは、異なるデータソースからのデータアイテムの場所を（データ記憶ユニットの観点から）追跡させ又は究明可能にする。次に、統計が生成され、新しいデータアイテムの割り当て又は既存のデータアイテムの再割り当てのような意志決定を可能にする。識別子は、メタデータ片であっても良く、又はデータアイテムの内容から特定の他の方法で導出可能であっても良い。

【００５２】

本発明の別の態様の実施形態は、複数のデータ記憶ユニットの間にデータアイテムを分散する装置を含み、前記データアイテムは複数のデータソースからのデータの集約であり、前記装置は、前記複数のデータソースの各々のセマンティック記述を生成するよう構成される記述生成モジュールと、前記複数のデータソースからの各データソース対について、前記データソース対の前記セマンティック記述間の類似性を計算するよう構成される類似性計算モジュールと、割り当て中のデータアイテムのデータソースと前記データ記憶ユニットに既に割り当てられたデータアイテムの各データソースとの間の前記計算された類似度に依存して、データ記憶ユニットにデータアイテムを割り当てるよう構成される割り当てモジュールと、を有する。

【００５３】

本発明の別の態様の実施形態は、コンピュータ装置により実行されると、前記コンピュータ装置に本発明を実施する方法を実行させる、コンピュータプログラムを有する。

【００５４】

本発明の更なる態様の実施形態は、１又は複数のコンピューティング装置により実行されると、前記１又は複数のコンピューティング装置を本発明を具現化する装置として及び／又はデータ記憶システムとして機能させる、コンピュータプログラム又はコンピュータプログラムスーツを含む。

【００５５】

このようなコンピューティング装置は、例えばサーバであっても良い。記憶機能に加えて、コンピューティング装置は、処理動作を実行し、データ記憶ユニットの分散型ネットワーク内の他のデータ記憶ユニットと及び／又は中央制御部と通信するよう構成される。

【００５６】

実施形態は、複数のデータ記憶ユニットの間にデータアイテムを分散する、本発明を実施する装置を有するシステムの形式で提供されても良い。当該システムは、複数のデータ記憶ユニットも有する。データ記憶ユニットは、それぞれ、コンピューティングリソースであっても良い。例えば、それらは、それぞれ、プロセッサ、メモリ及び／又はネットワークインタフェースカード、マザーボード、入力／出力装置のような追加構成要素に加えて、記憶ユニットを有しても良い。

【００５７】

さらに、本発明の実施形態は、コンピューティング装置の分散型ネットワークにより実行されると、コンピューティング装置の分散型ネットワークに、本発明を具現化するシステムとして機能させる、コンピュータプログラムスーツを有しても良い。

【００５８】

さらに、本発明の実施形態は、記憶ユニットの分散型ネットワークを有するコンピューティング装置のシステムにより実行されると、前記システムに、本発明を具現化する方法を実行させるコンピュータプログラム又はコンピュータプログラムスーツを有する。

【００５９】

態様（ソフトウェア／方法／装置）が別個に議論されたが、１つの態様に関連して議論されたその特徴及び影響は、他の態様にも等しく適用できる。したがって、方法の特徴が議論される場合、装置の実施形態はその特徴を実行する又は適切な機能を提供するよう構成されるユニット又は装置を有すること、及びプログラムは該プログラムが実行されるコ

ンピューティング装置に前記方法の特徴を実行させるものと解釈される。
【００６０】
上述の態様のいずれにおいても、種々の特徴は、ハードウェアで、又は１又は複数のプロセッサで動作するソフトウェアモジュールとして実装されても良い。一態様の特徴は、他の態様のいずれにも適用できる。
【００６１】
本発明は、上述の任意の方法を実行するコンピュータプログラム又はコンピュータプログラムプロダクト、及び上述の任意の方法を実行するプログラムを格納しているコンピュータ可読媒体も提供する。本発明を実施するコンピュータプログラムは、コンピュータ可読媒体に格納されてもよい。或いは、例えば、インターネットウェブサイトから提供されるダウンロード可能なデータ信号のような信号形式又は任意の他の形式であってもよい。
【００６２】
本発明の実施形態は、どのアルゴリズム又は他の分散メカニズムが最初に及びデータ再割り当て手順の一部としてデータアイテムを割り当てるべきかの基礎としてセマンティック記述を利用する方法、システム、装置、及びソフトウェアを提供する。
【００６３】
本発明の特定の実施形態は、添付の図面を参照して以下に更に詳細に議論される。
【図面の簡単な説明】
【００６４】
【図１】本発明を具現化する処理のフローを示す。
【図２】本発明を具現化するデータ集約装置の例示的なアーキテクチャを示す。
【図３】類似性行列とＫｅｙ＿Ｎｏｄｅ－ＩＤテーブルとの間の相互作用を更に説明するシステムアーキテクチャを示す。
【図４】本発明を具現化する処理が本発明を具現化する装置によりどのように実行され得るかを示す。
【発明を実施するための形態】
【００６５】
図１は、本発明を具現化する一連の処理を示す。処理が提示される順序は、前のステップの出力に対する後続のステップの依存を表す。しかしながら、出力は後続ステップの複数の反復で再利用されても良い。したがって、例えば、必ずしもセマンティック記述を生成し、データアイテムがデータ記憶ユニットに割り当てられる度に類似度を計算する必要はない。以下の説明では、用語データベースは、複数のデータ記憶ユニットの間に格納される複数のデータアイテムを表すために用いられる。
【００６６】
図１に示す個々の処理は、本発明を具現化する方法を形成する。方法は、コンピュータにより実施され、専用ハードウェアで実施され、又は特定の他の方法で実施されても良い。方法がコンピュータにより実施される場合、方法は、（ノード又はサーバとも表される）単一のコンピュータにより、又は複数の相互接続された若しくはネットワーク接続された共同的に動作するコンピュータにより、実施されても良い。例えば、異なる個々の処理Ｓ１０１～Ｓ１０３は、別個のコンピュータにより実行されても良い。請求項１の方法を実施するコンピュータは、ヘッドサーバ、ヘッドノード、エントリサーバ、エントリノード、又はデータベース若しくはデータ記憶システム内の制御機能を有する特定の他のコンピュータであっても良い。
【００６７】
図１に示す処理Ｓ１０１～Ｓ１０３は、本発明を具現化する方法を形成する。それらは、他の処理を組み入れられ又は結合されて、本発明を具現化する更なる方法を形成しても良い。
【００６８】
個々の処理Ｓ１０１～Ｓ１０３は、必ずしも順次実行されず、時間的に重なり合い又は同時に実行されても良い。例えば、既存のセマンティック記述は、類似度を計算するステ

ップＳ１０２内で、同時に１又は複数の新しいデータソースのセマンティック記述を生成するステップＳ１０１内で、部分的に利用できる。データベースは、第１のデータソースからの全てのデータアイテムが準備されデータベースに書き込まれ、次に次のデータソースへと、データソース毎にデータソース上にコンパイルされても良い。第１のデータソースのセマンティック記述は比較する目的で生成されるが、第２のデータソースからのデータアイテムがデータベースへの書き込みの準備が整うまで、類似度の計算Ｓ１０２はできない。第１のデータソースからのデータアイテムは、ランダムに割り当てられるデータ記憶ユニットに書き込まれても良く、単一のデータ記憶ユニットに又は可能な限り少数のデータ記憶ユニットに一緒にグループ化されても良い。

【００６９】

ステップＳ１０１で、セマンティック記述は、複数のデータソースの各々について生成される。該複数のデータソースからのデータは、データベース内に集約される。ステップＳ１０１は、記述生成モジュールにより実行されても良い。記述生成モジュールは、コンピュータプログラムの機能モジュールであっても良く、したがって出力（データソースのセマンティック記述）を生成するために入力（データソースからのリードデータ）に基づき命令を実行するコンピュータハードウェアを有する。代替で、記述生成モジュールは、専用ハードウェアであっても良い。新しいデータソースからのデータアイテムがデータベースに追加される度に、データソースのセマンティック記述が生成される。セマンティック記述は、データソースの内容の意味の要約である。セマンティック記述は、一旦生成されると、例えばデータソースからの新しいデータアイテムがデータベースへの追加のために読み出されるようなイベントによりトリガされるとき、又はシステムアイドル時間に、更新されても良い。一旦生成されると、セマンティック記述は、個々のデータソースの識別子と一緒に格納されても良い。

【００７０】

セマンティック記述のフレームワーク又は構造は、実装に依存する。例示的な形式は、それぞれ用語の重要度の定量的表現に関連付けられるデータソース内の最重要用語のリスト又はベクトルを有する。固定数の用語が各データソースのセマンティック記述に含まれても良い。この固定数は、予め定められ、データアイテムがデータベースに含まれる全てのデータソースについて一定であっても良い。

【００７１】

セマンティック記述を生成する全体処理Ｓ１０１に含まれる処理ステップは、実装に依存する。例示的なステップは、データソースを読み出すステップ又はデータソースに含まれる情報を読み出すステップと、読み出したデータ又は情報を分析に適するフォーマットに構造化するために、該読み出したデータ又は情報に対して前処理を実行するステップと、セマンティック重要性を有しない共通ワードを除去するステップと、例えば頻度に基づく分析方法により最重要用語を識別するために読み出したデータ又は情報を分析するステップと、のうちの１又は複数を含む。

【００７２】

ステップＳ１０２で、データアイテムがデータベースに書き込まれるべきデータソースの対の間の類似度が計算される。２以上のデータソースが存在する実施形態では、２以上のデータソースの中から少なくとも１つの及び可能な場合には全部の対の類似度が計算される。ステップＳ１０２は、新しいデータソースからのデータアイテムがデータベースに追加される度に実行されても良い（新しいデータソースと１又は複数の既存のデータソースとの間の類似度を計算する）。この意味で、類似度は、データベースと共に時間をかけて構築される。

【００７３】

類似度を計算するステップＳ１０２は、複数のデータソースの中から各データソース対について、検討されている複数のデータソースの中からの各データソース対について、データソース対のセマンティック記述間の類似度を計算するステップを有する。検討されている複数のデータソースは、データアイテムがデータベース内で集約される全てのデータ

ソースであっても良く、又はこれらのデータソースのサブセット、例えばセマンティック記述を生成するのに適するフォーマットのデータソースであっても良い。

【００７４】

データソース対間の類似度を計算するステップは、対のセマンティック記述を比較することにより実行される。したがって、データソース対間の類似度を計算するステップは、データソース対のセマンティック記述間の類似度を計算するステップと等価である。

【００７５】

ステップＳ１０２は、類似性計算モジュールにより実行されても良い。類似性計算モジュールは、コンピュータプログラムの機能モジュールであっても良く、したがって出力（データソース対間の類似度を表す値のような定量化）を生成するために入力（データソース対のセマンティック記述）に基づき命令を実行するコンピュータハードウェアを有する。類似度は、実施形態では、上限と下限との間、例えば０と１の間の任意の個数の離散値を取り得る値のようなデータにより表されても良い。値の可能な離散値の個数（つまり精度）は、実装の詳細に依存する。しかし、例えば値は単一の小数位に丸め込まれても良い。

【００７６】

セマンティック記述の対の間の類似度を計算するステップは、対のうちの一方のセマンティック記述内のどの用語が対のうちの他方のセマンティック記述にも現れるかを識別すること、及びこのような用語が識別される度に対の類似度をある量だけ増大することに基づいても良い。例えば、増大する量は、個々のセマンティック記述内の用語に関連付けられる個々の重み（セマンティック記述内の用語の重要性を表す数値）の積に比例しても良い。セマンティック記述は、用語の同義語及び／又は複合語を含むよう拡張されても良い。したがって、類似度は、両者のセマンティック記述が用語及び該用語の１又は複数の同義語若しくは複合語を含むリストからの１つを含むとき、増大される。

【００７７】

１又は複数のデータソース対の類似度が計算されると、類似度は格納され、より詳細には、類似度を表す値が格納される。データソース対の間の類似度は、類似度の計算された特定のデータソース対の属性となるように格納される。例えば、正方行列は、列と行が各データソースに含まれるようコンパイルされても良い。したがって、行列内の各エントリは、データソース対に対応し、エントリは、該対の間の類似度（又は代表値）である。

【００７８】

処理Ｓ１０３で、データアイテムは、データ記憶ユニットに割り当てられる。ステップＳ１０３は、割り当てモジュールにより実行されても良い。割り当てモジュールは、コンピュータプログラムの機能モジュールであっても良く、したがって、出力（１又は複数のデータアイテムを格納すべきデータ記憶ユニットの選択）を生成するために、入力（割り当てられるべき１又は複数のデータアイテム、及び該１又は複数のデータアイテムのデータソースとデータ記憶ユニットに既に格納されたデータアイテムのデータソースとの間の計算された類似度）に基づき命令を実行するコンピュータハードウェアを有する。割り当てモジュールは、１又は複数のデータアイテムを選択されたデータ記憶ユニットに書き込むよう構成されても良い。代替で、割り当てモジュールは、データアイテム書き込みモジュールに、各データアイテムが割り当てられるデータ記憶ユニットを通知しても良い。次に、データ書き込みユニットは、データアイテムのデータ記憶ユニットへのデータ書き込み動作を実行する。

【００７９】

データ記憶ユニットにデータアイテムを割り当てるステップＳ１０３は、割り当て中のデータアイテムのデータソースとデータ記憶ユニットに既に割り当てられたデータアイテムの各データソースとの間の計算された類似度に依存して、データ記憶ユニットにデータアイテムを割り当てるステップを有する。データアイテムを割り当てるべきデータ記憶ユニットを選択するために用いられる特定のアルゴリズムは、実装に依存する。割り当て中のデータアイテムのデータソースとデータ記憶ユニットに既に格納されたデータアイテム

との間の類似度は、データ記憶ユニットが割り当て目標として検討される優先順位を決定する。アルゴリズムは、割り当て中のデータアイテムのデータソースに最も類似するデータソースからのデータアイテムを格納するデータ記憶ユニット又は複数のデータ記憶ユニットを単に発見しても良い。代替で、特定の形式の集約又は統合は、格納されているデータアイテムのデータソースが割り当て中のデータアイテムのデータソースに平均してどれだけ類似しているかについて各データ記憶ユニットにスコアを付けるために用いられても良い。そして、最高平均及びデータアイテム又は複数のデータアイテムのための十分な記憶空間を有するデータ記憶装置が選択される。優先順位が決定されると、利用可能記憶空間又は妥当なデータソースからのデータアイテムの割合のような追加因子は、差別因子として又はタイブレイクのために考慮される。例えば、割り当てられるべきデータアイテムのデータソースに最も類似するデータソースは、格納された類似度から識別され、データアイテムは、識別されたデータソースからのデータを格納するデータ記憶ユニットに割り当てられても良い。例えば、識別されたデータソースから、最も多くのデータアイテムを格納するデータ記憶ユニットへである。データアイテムは、それら個々のデータソースからのデータアイテムを格納するデータ記憶ユニットに優先的に割り当てられても良い。しかし、データソースからデータベースへが無いとき、又は記憶空間が無いとき、データアイテムは、最も類似するデータソースからのデータアイテムを格納するデータ記憶ユニットに割り当てられる。

【００８０】

任意的に、データアイテムは、グループ化されて割り当てられる。例えば、共通データソースからのデータアイテムセットは、割り当てのために準備される。例えば、データソースは、データベース内にデータアイテムが現在存在しないデータソースであっても良い。データアイテムセットは、最も類似するデータソースからデータアイテムセットの共通データソースへのデータアイテムの最大の割合を格納するデータ記憶ユニットに優先的に割り当てられる。必要条件又は割り当てルールは、セットは１より多いデータ記憶ユニットに分けられない、であっても良い。したがって、このようなセットが最も類似するデータソースからのデータアイテムの最大割合を格納するデータ記憶ユニットに格納できない場合、セット全体を格納するために、それらは十分な空間を有する最も類似するデータソースからのデータアイテムの最高割合を格納するデータ記憶ユニットに割り当てられる。

【００８１】

図２は、データ集約装置１０の例示的なアーキテクチャを示す。この及び他の例及び実施形態の議論において、ＲＤＦトリプルは、データアイテムの例示的な形式として用いられる。実施形態は、他の形式のデータアイテムに適用でき、ＲＤＦトリプルの例は、多くの可能なデータアイテムフォーマットのうちの１つである。データソースは、外部データソースとして参照されても良い。外部データソースは、データソースが、データアイテムが割り当て中である複数のデータ記憶ユニットの外部にあることを示す。

【００８２】

図２に示すデータ集約装置１０は、以下のコンポーネント：リーダ１２、プロセッサ１４、ライタ１６、及びメタデータレジストリ１８を有する。コンポーネントは、専用ハードウェアコンポーネントとして、又はプログラムの部分を形成するモジュールとして、又はコンピュータにより実施され得る方法の中の処理として、実現されても良い。データ集約装置１０は、データ記憶装置２０と通信するよう動作する。データ記憶装置は、集約されたデータの統一ビューの表現として単一のコンポーネントとして示される。しかしながら、データ記憶装置２０を提供するハードウェアは、複数の個々のデータ記憶ユニットである。

【００８３】

図３は、類似性行列とＫｅｙ＿Ｎｏｄｅ－ＩＤテーブルとの間の相互作用を更に説明するシステムアーキテクチャを示す。図３は、データ記憶装置２０が複数のデータ記憶ユニットにより提供されることを強調する。例えば、各データ記憶ユニットは、別個のハードディスク、記憶ノード、又はサーバであっても良い。図４は、リーダ１２の外部データソ

ース３０との相互作用も強調する。外部データソース３０とリーダ１２との間の矢印は、外部データソース３０からリーダ１２により読み出されているデータを表す。

【００８４】

データ集約装置１０は、本願明細書のどこかで言及される複数のデータ記憶ユニットの間にデータアイテムを分散させる装置の例である。

【００８５】

リーダ１２は、データ集約装置１０と複数の外部データソース（external data source：ＥＤＳ）３０との間の接続を確立するよう構成される。リーダ１２は、データソースからデータを検索し、必要な場合には該データをシステムの解析可能なドキュメントフォーマットに変換するよう構成される。リーダ１２は、データソースからメタデータを抽出し、識別子をデータソースの属性にする（attribute）（データソースへの識別子の帰属は、代替でプロセッサ１４により実行されても良い）。ステップＳ１で、例えば１又は複数のソースＵＲＬ、タイトル、及びデータ型、並びに場合により識別子も含み得る抽出されたメタデータは、リーダ１２によりメタデータレジストリ１８に書き込まれる。

【００８６】

プロセッサ１４は、本願明細書のどこかで言及される記述生成モジュールの例である。プロセッサ１４は、外部データソースから読み出したデータをデータアイテムがデータ記憶ユニット２０に格納されるデータフォーマットに変換するよう、例えば外部データソースからのデータをＲＤＦトリプルに変換するよう構成される。さらに、プロセッサは、外部データソースのセマンティック記述を生成するよう構成される。データ変換及びセマンティック記述の生成は、並列に実行されても良い。ステップＳ２で、各データソースのセマンティック記述（semantic description：ＳＤ）は、記述されるデータソースの識別子と関連付けられてメタデータレジストリ１８に書き込まれる。

【００８７】

ライタ１６は、類似性計算モジュール及び本願明細書のどこかで言及される割り当てモジュールの例である。ステップＳ３で、ライタ１６は、メタデータレジストリ１８からセマンティック記述を検索し、複数のデータソースの中の各データソース対について、データソース対のセマンティック記述間の類似度を計算するよう構成される。例えば、新しい外部データソースからのデータは、リーダ１２により読み出され、プロセッサ１４により処理されても良い。データベース内のデータアイテムを既に有し、メタデータレジストリ１８に格納された又は別個のレコードとして格納されたデータソース間の類似度のレコード／テーブル／行列は、ステップＳ４でライタ１６により更新される。図３のメタデータレジストリ１８、類似性行列１７、及びＫｅｙ－Ｎｏｄｅ　ＩＤ１９は、全て単一の記憶コンポーネントに格納されても良い。

【００８８】

新しいデータソースと各外部データソースとの間の類似度は、計算され記録される。次に、ライタは、計算された類似度に依存して、新しいデータソースのようなデータソースからのデータアイテムを、データ記憶装置２０の間のデータ記憶ユニットに割り当てる。そうするために、ステップＳ５で、どのノード（記憶ユニット）が割り当て中のデータアイテムのデータソースに最も類似すると考えられるデータソースからのデータアイテムを格納しているかを識別するために、ｎｏｄｅ＿ＩＤ情報は、ライタ１６によりＫｅｙ－Ｎｏｄｅ＿ＩＤテーブル１９から検索される。さらに、ステップＳ６で、ライタは、新しいデータアイテムが割り当てられたとき、Ｋｅｙ－Ｎｏｄｅ＿ＩＤテーブル１９を更新しても良い。したがって、Ｋｅｙ－Ｎｏｄｅ＿ＩＤテーブル１９は、各データソースについて、データソースからの１又は複数のデータアイテムを格納している該又は各データ記憶ユニットの識別子と、該又は各識別されたデータ記憶ユニットに格納されたデータソースからのデータアイテムの割合の指示との最新のデータレコードを維持する。レコードは、ノードＩＤがデータ記憶ユニットの識別子であり及びｋｅｙがデータアイテムのデータソースの識別子であるＫｅｙ－Ｎｏｄｅ＿ＩＤテーブル１９であっても良い。このようなテーブルでは、データアイテム毎のエントリ、データソース毎のエントリ、又は各ノードから

のデータを格納するデータ記憶ユニット毎のエントリがある。ライタ１６は、新しいデータソースからのデータアイテムを割り当て、及び最適化のためにデータアイテムを再割り当てするよう構成されても良い。これらの処理の各々は図３のステップＳ７により表される。

【００８９】

メタデータレジストリ１８は、外部データソースに関する情報を保持し、データ集約装置１０のコンポーネントによりアクセス可能である。表１は、メタデータレジストリに格納され得る例示的なデータを示す。セマンティック記述は、説明の目的のために省略される。

［表１］例示的なメタデータレジストリ１８であり、列は外部データソースに関連する情報を示す。

【表１】

ID	ソース URL	タイトル	型	セマンティック記述
EDS_1	File://mydatabase/example	Financial	RDBMS	
EDS_2	File:///mypath/mydocument	...	Word	
EDS_3	http://dbpedia.org	dbpedia	Web Data	
...		...	...	
EDS_{new}		Finance Stock Exchange		

【００９０】

表１では、ＩＤは、メタデータレジストリ１８又はデータ集約装置１０の特定の他のコンポーネントにより各外部データソースについて生成される識別子であり、複数の外部データソース間でユニークである。例えば、このＩＤは、どの外部データソースにこれらのＲＤＦトリプルが属するかを示すために、データソースからＲＤＦトリプルの各々に埋め込まれても良い。このＩＤは、データ割り当て／再割り当て処理でデータの再割り当てのために用いられても良い。ソースＵＲＬは、外部データソースの絶対パスを表す。ソースＵＲＬは、ファイルシステム、ウェブサイト、データベースパス、又は特定の他のロケータであっても良い。タイトルは、外部データソースの名称であり、文書の場合にはファイル名、又はウェブサイトを表す短い名称であっても良い。型は、外部データソースの型、例えばＲＤＢＭＳデータベース、ファイル、又はウェブリソースを示す。セマンティック記述は、外部データソースのセマンティック要約であり、一連の重み付けされた用語の形式であっても良い。ソースＵＲＬ、タイトル、及び型の値は、外部データソースが検索されるとき、リーダ１２により外部データベースから直接抽出されても良い。セマンティック記述は、外部データソース種類、例えばＲＤＢＭＳ／文書／ウェブデータから、統一データ型、例えばＲＤＦへのデータ変換と並列して、プロセッサ１４により生成されても良い。

【００９１】

１つの実施形態の選択肢として、各データソースのセマンティック記述は、同じ用語を有しても良い。各セマンティック記述の中の各用語は、重み付け、又はセマンティック記述が関連するデータソース内での用語の出現頻度の特定の他の形式の定量的指示（例えば、頻度は、全てのデータソースを集合的に表すｉｄｆを有するｔｆ－ｉｄｆであっても良い）に関連付けられる。例えば、既存のセマンティック記述は、新しいセマンティック記述が生成されるとき、新しい用語を含むよう更新されても良い。さらに、重み付け又は既

存のセマンティック記述の定量的指示は、新しいセマンティック記述が生成されるのに応答して、更新されても良い。

【００９２】

データソースの例示的なセマンティック記述は、データソースから抽出され／読み出された重み付けされた用語のリストであり、外部データソースのセマンティック要約を提供する。テキスト要約技術が利用可能であり、データソースのセマンティック記述を生成する際に用いられ／利用されても良い。本例では説明を目的として、用語頻度（term－frequency：ＴＦ）方法が適用される（用語頻度は、ワード又はフレーズがコレクション又はコーパス内の文書に対してどれ位重要かを反映する数的統計値であり、情報検索技術の例である）。ＴＦに基づくデータ要約は、データソースのセマンティック記述を形成するために全てのデータソースから重み付けされた用語のリスを抽出するステップを有する。基本的な例示的アルゴリズムは、例えば、文書内でどれが最重要用語かを識別するステップ及びそれらをセマンティック記述として場合によってはそれらの相対的重要度を示す値と一緒に抽出するステップの前に、文書（データソース）内の用語の生の頻度を用いて、各用語がどれ位重要かを評価する。生の頻度は、用語ｔが文書ｄ内で発生する回数である。用語頻度を計数する前に、外部データソースから読み出されたデータは、記述生成モジュールによる前処理を行われても良い。前処理は、自然言語処理（Natural language processing：ＮＬＰ）技術を含んでも良く、トークン化、語幹解釈、ストップワード除去、等のうちの１又は複数を含む。ここで、語幹解釈は、例えばワード「fishing」、「fished」、「fish」を基語「fish」に縮めることを表す。ストップワード除去は、「a」、「the」、「of」等のようなあまり意味の無い語を取り除く。ストップワードは、ストップワードリストを参照することにより識別されても良い。ストップワードリストは、データ集約装置１０に格納され、又はリモートに格納され記述生成モジュールによりアクセス可能にされても良い。

【００９３】

記述生成モジュールがデータソースのセマンティック記述（セマンティック要約とも表される）を生成できる前に、ストップワードのリストが定められても良い。ストップワードは、他のワードより高い頻度で用いられるワードであり、したがって、セマンティック記述内の用語として選択される場合、あまり意味が無くあまり重要でないく又は無意味であり、例えばa,、and、are、as、the、of、will、等である。ストップワードの定義は、手動で実行され得る。或いは、コレクション頻度－各用語が文書内に現れる合計回数－を見付けるために、外部データソースの事前スキャンが実行され、次に文書のドメインと無関係に最も頻度の高い用語がストップワードと考えられても良い。代替で、ストップワードの汎用リストは、外部ソースから得られても良い。ストップワードリストが準備できた後に、セマンティック記述を形成するために重み付けされた用語を見付けるために、外部データは更に処理される準備が整う。例えば、ストップワードを除いてデータソース内で最高出現頻度を有する用語は、セマンティック記述に含まれても良い。

【００９４】

実装の詳細に依存して、外部データソースは、以下の型に分類され、記述生成モジュールにより異なる方法で処理されても良い。

【００９５】

－型１：テキスト文書／ウェブベーステキスト

－型２：表形式、例えばＲＤＢＭＳ／Ｅｘｃｅｌ

型１のデータでは、文書は、文書内で頻繁に現れる用語の数（及びどれがストップワードでないか）を識別することにより分析されても良い。これらの用語は、基本的に文書からのキーワードであり、したがって、セマンティック記述に含まれる。

【００９６】

型２のデータでは、データソースのセマンティック記述の生成は、データがＲＤＦトリプルに変換された後に、それらがデータ記憶装置２０に書き込まれる前に、実行されても良い。例えば、前の表データを符号化する変換されたＲＤＦトリプルは、タートル（turt

le）ファイルに一時的に書き込むことができる。次に、用語頻度処理は、型１の文書データと同様の方法で処理できる。これらのタートルファイルは、ＲＤＦトリプルがＲＤＦ記憶内に格納された後に保持され又は削除されても良い。タートルファイルは、文書、又は表形フォーマットからのデータを用語頻度分析が実行できる形式で表す他のテキストに基づくエンティティの例である。

【００９７】

外部データソースからのデータが集約データベースに含まれるために外部データソースと異なるデータソースへの変換を必要とする場合、上述のセマンティック記述生成処理は、変換処理と並列に実行されても良い。例えば、外部データソースからのデータは、処理１４により、ＲＤＦトリプルとして再フォーマット化されても良い。記述生成及び再フォーマット化を並列に実行することは、性能オーバヘッドを低減する。例えば、データは、一度読み出され、同じリードデータが用語頻度目的で分析されるのと同時にＲＤＦトリプルフォーマットに変換され得る。生成されたセマンティック記述の各々は、例えばメタデータレジストリ１８に格納される。

【００９８】

セマンティック記述生成モジュールは、用語頻度に基づくもの以外のテキスト要約ツールを用いても良い。さらに、他の標準的なＮＬＰ前処理、例えばトークン化及び語幹解釈、並びに追加処理は、このようなツールにより自動的に実行され得る。

【００９９】

さらに、セマンティック記述に含まれる最重要用語に関連する数値スコア又は重みを生成又は変更するために、潜在的セマンティック分析が実行されても良い。

【０１００】

潜在的セマンティック分析は、文書のコレクション内の用語の重要性を定量化する技術である。この場合、文書のコレクションは、データアイテムが由来する集合的データソースである。潜在的セマンティック分析は、用語に適用される重み付けの精度を向上する。したがって、類似する用語（例えば、複合語又は同義語）は識別され、グループとして類似する用語の重要性が重み付けにおいて考慮される。例えば、潜在的セマンティック分析は、用語の行列及び用語が現れるデータソースを集めるステップと、（文書コーパス全体又はテキスト記述全体の中の用語の使用頻度に対するデータソース内の用語の使用頻度についての）ｔｆ－ｉｄｆ重み付けを行列に移植するステップと、低ランク近似を見付けるために、行列にランク低下を適用するステップと、を有しても良い。

【０１０１】

任意で、データソースのセマンティック記述を生成するステップは、最重要用語の各々の同義語を見付け及び同義語をセマンティック記述に含めるステップを有しても良い。

【０１０２】

各用語の同義語をセマンティック記述に含めるステップは、別の一般化度を提供する。一般化度は、ローカルな命名の習慣の影響を減じ、セマンティック記述のドメイン又はオントロジ特異性を低減するのを助ける。同義語を見付けるステップは、セマンティック記述内の各用語をクエリ内の引数として及びクエリが同義語で応答されるべきであることを指定して、（場合によってはアプリケーションプログラミングインタフェースを介して）ローカル又はリモートに格納されたシソーラス又は辞書に問い合わせるステップと、セマンティック記述に未だ存在しない結果をセマンティック記述に追加するステップと、を有しても良い。

【０１０３】

セマンティック記述対を比較するとき、用語がセマンティック記述対の間で共通であると見付かったとき、計算中の類似度が増大される量は、個々のセマンティック記述内の用語に関連する個々の重み（セマンティック記述内の用語の重要性を表す数値）の積に比例しても良い。セマンティック記述は、用語の同義語及び／又は複合語を含むよう拡張されても良い。したがって、類似度は、両者のセマンティック記述が用語及び該用語の１又は複数の同義語若しくは複合語を含むリストからの１つを含むとき、増大される。

【０１０４】
類似性計算モジュールにより実行されるデータソース対の間の類似度の計算又は類似性のスコア付けは、意味的に類似する外部データソースからのデータアイテムをデータ記憶ユニット（サーバノード）に、同じデータ記憶ユニット内の同一場所に配置される傾向のある順序で（又はそのような場所に）格納する際に利用され得るスコア／値を生成する。意味的に類似するデータソースからのデータアイテムを同じデータ記憶ユニットに格納することは、範囲クエリのようなクエリに対する応答を生成するために必要なデータ記憶ユニットアクセス（ディスクアクセス）の数を低減する。例えば、Ｗｉｋｉｐｅｄｉａ及びＤＢｐｅｄｉａは共に知識ベースであり、ユーザ又はアプリケーションがＷｉｋｉｐｅｄｉａ内の知識を探している場合、ＤＢｐｅｄｉａも検索される可能性が非常に高い。同様に、データソースが財務バランスシートを含む場合、該データソースは、金融株式情報をアクセスするクエリ内でアクセスされる可能性が高い。

【０１０５】
異なるデータソース間の類似度を計算するために、標準的なコサイン類似性指標のような指標が用いられても良い。コサイン類似性は、２つのベクトル間の角度のコサインを測定する、内積空間の２つのベクトル間の類似度（量／レベル）を測定する方法である。２つのデータソース間の類似度の計算に、又はより具体的には、２つのデータソースの生成されたセマンティック記述間の類似度の計算に適用されるとき、コサイン類似性は、２つのセマンティック記述間の類似性（度）を計算する数学的方法である。類似度であるコサイン類似値の範囲は、２つのデータソースの場合、（ｔｆ－ｉｄｆ重みにより与えられる）用語頻度は負にならないので、０から１である。２つの外部データソースＥＤＳｉ及びＥＤＳｎｅｗが与えられる場合、セマンティック記述類似性は、２つのセマンティック記述対の間のコサイン類似性により、次式のように定められる。

【数１】

$$\mathrm{similarity}(SD_i, SD_{new}) = \cos(S_{SD_i} S_{SD_{new}}) = \frac{S_{SD_i} \cdot S_{SD_{new}}}{|S_{SD_i}||S_{SD_{new}}|}$$

【０１０６】
全てのセマンティック記述対の間の計算された類似性スコアは、記録され、例えば以下に示すように類似性行列１７に格納される。

【０１０７】
データソースのセマンティック記述を生成する処理、及びデータソースと他のデータソースとの間の類似度を計算する処理は、新しいデータソースからのデータが集約データベースに入力される度にトリガされても良い。

【０１０８】
以下の説明のための例では、初期データ記憶装置は空であり、つまり、最初のデータソースからのデータアイテムが追加されるとき、データ集約システム内では最初に類似度計算がない。最初のデータソースはＥＳＤ１として参照され、そのセマンティック記述は生成され及び記録され、ＳＤ１として参照される。２番目の外部データソースＥＤＳ２が読み出され、及びそのセマンティック記述が生成されるとき、類似度計算が開始し、ＳＤ１をＳＤ２と比較し、類似性結果を類似性行列１７に書き込む。次回、新しいデータソースＥＤＳ３が読み出されるとき、新しいデータソースのセマンティック記述が生成され、新しいデータソースＥＤＳ３と既存のデータソースＥＤＳ１及びＥＤＳ２の各々との間の類似度が計算される。したがって、類似性行列１７は、次第に拡張される。例示的な類似性行列１７は表２に示される。行列は対称性を有し、対角線上のセル内では１に等しい値を有する。

［表２］セマンティック記述類似性行列

【表２】

	SD1	SD2	SD3	…	SDnew
SD1	1	0.6	0.4	…	0.7
SD2	0.6	1	0.5	…	0.3
SD3	0.4	0.5	1	…	0.4
…	…	…	…	1	…
SDnew	0.7	0.3	0.4	…	1

【０１０９】
図２のアーキテクチャでは、類似性計算は、ライタにより実行されるデータ割り当てのための前処理として、ライタ１６により実行される。類似性計算の結果は、例えば表２のような類似性行列１７としてメタデータレジストリ１８に格納される。
【０１１０】
纏めると、外部データソースは読み出され、必要な場合には、そのデータは集約データベースの統一データフォーマットに変換される。データを変換する間、又はその他の場合、外部データソースのデータは、スキャンされ、任意的に既存データとのデータ整合が実行される。スキャンしている間、例えば、単純な用語頻度（ＴＦ）に基づく要約を実行することにより、データのセマンティック記述が生成される。生成されたセマンティック記述内の用語に帰属する重みを正規化するような更なる処理が実行されても良い。
【０１１１】
計算された類似度に基づき、新しい外部データソースＥＤＳｎｅｗからのデータアイテムは、割り当てモジュール又はライタ１６により、（識別子又はＮｏｄｅ＿ＩＤを指定することにより）選択されたデータ記憶ユニットに割り当てることができる。ＥＤＳｎｅｗと既存のデータソースＥＤＳｉとの間の類似度に依存してＥＤＳｎｅｗからデータ記憶ユニットへのデータアイテムを割り当てるステップは、意味的に類似するデータソースからのデータアイテムを同一場所に配置させる。意味的に類似するデータセット（及びデータセット内のデータアイテム）がデータベース操作において一緒に利用される可能性がより高いことに基づき、交差データ記憶ユニット操作を最小限にすることで、性能利益を生じる。
【０１１２】
マッピングテーブルは、各データソースについて、データソースからの１又は複数のデータアイテムを格納している該又は各データ記憶ユニットの識別子と、該又は各識別されたデータ記憶ユニットに格納されたデータソースからのデータアイテムの割合の指示とのレコードを維持し得る。データアイテムの割合の指示は、絶対的又は相対的であっても良い。したがって、割合の値がレコードから引き出せない場合でも、データ記憶ユニットは、特定のデータソースからの最も多いデータアイテムを格納するものから最も少ないものを格納するものへの順序でランク付けできる。Ｋｅｙ－Ｎｏｄｅ＿ＩＤマッピングテーブル１９は、このようなレコードの例である。特定のＥＤＳｉのサーバＮｏｄｅ＿ＩＤを見付けるために、Ｋｅｙ－Ｎｏｄｅ＿ＩＤマッピングテーブル１９は、ライタ１６により維持されても良い。
【０１１３】
ハッシュテーブルは、Ｋｅｙ－Ｎｏｄｅ＿ＩＤマッピングテーブル１９を実装するために用いられても良い。Ｋｅｙ－Ｎｏｄｅ＿ＩＤマッピングテーブル１９により提供される機能は、複数のデータソースの各々からのデータアイテムの位置情報のレコード（データ記憶ユニット識別子の観点で）を維持することである。ＥＤＳからのデータアイテムが割り当てられる度に、又は既存のＥＤＳからのデータアイテムが再割り当てされる度に、このテーブルは、これらの変化を反映するために相応して更新される。ハッシュテーブルの例は、表３に示される。
［表３］Ｋｅｙ－Ｎｏｄｅ＿ＩＤテーブル１９

【表３】

Key	Node_ID List
EDS1	[N1, N3....]
EDS2	[N1, N5...]
EDS 3	[N2, N7...]
...	...
EDS new	

【０１１４】
上記のテーブルでは、キーはデータソース識別子（ＩＤ）であり、メタデータレジストリで用いられるキーと同じである。また、Ｎｏｄｅ＿ＩＤリストは、各識別されたデータソースからのデータアイテムを格納するデータ記憶ユニットの識別子のリストである。この特定の例では、ノードＩＤ（データ記憶ユニット識別子）は、格納されるデータソースからのデータアイテムの割合の順で格納される。第１のＮｏｄｅ＿ＩＤは、ＥＤＳｉからのデータアイテムの最も高い割合を格納するサーバノードである。以下同様である。サーバＮｏｄｅ＿ＩＤは、通常、データ／データセグメントＩＤを記憶装置／サーバノードにマッピングするハッシュ関数を通じて、ライタ１６のようなデータ局所性メカニズムにより提供されても良い。
【０１１５】
ＥＤＳ２が新しい外部データソースＥＤＳｎｅｗに関して最も類似する外部データソースであると考えられる場合、対応するＮｏｄｅ＿ＩＤは、ＥＤＳｎｅｗからのデータアイテムのための可能な割り当て目標として、ハッシュテーブルから読み出される。生じ得る幾つかの異なる状況、及びそれらを処理するための可能なルールは、以下に提示される。
【０１１６】
ＥＤＳ２の大部分が格納されるサーバノード（本例ではＮ１）は、ＥＤＳｎｅｗからの１つのデータアイテム又は１つのデータアイテムグループを割り当てるために好適なデータ記憶ユニットとして選択される。
【０１１７】
したがって、ライタは、ＥＤＳｎｅｗからのデータアイテムを格納するとき、Ｎ１を優先する。
【０１１８】
ｓｉｚｅ（ＥＤＳｎｅｗ）＞ａｖａｉｌａｂｌｅ＿ｃａｐａｃｉｔｙ（Ｎ１）、つまり、ＥＤＳｎｅｗからのデータアイテムが一緒に格納されるべきである（これは実装オプションである）が、Ｎ１に十分な記憶空間がない場合、
適用できる場合、ＥＤＳ２の残り部分の最大割合を保持しているサーバノードが考慮される。
【０１１９】
次に最も類似するデータソースが選択され、上述の処理が繰り返される（つまり、データソースからのデータアイテムは、（データ記憶ユニットが）データアイテムのための十分な記憶空間を有する最も類似するデータソースからの大部分のデータアイテムを格納しているデータ記憶ユニットに割り当てられる）。
【０１２０】
したがって、データソースからのデータアイテムを割り当てるべきデータ記憶ユニットを選択する際に、割り当てモジュールは、先ず、最も類似するデータソースを見付け、データアイテムが全て割り当てられるまで、次に最も類似するデータソースへと進む。データソースからのデータアイテムが１より多いデータ記憶ユニットを占有するとき、該データ記憶ユニットに格納された該データソースからのデータアイテムの割合の降順は、データ記憶ユニットが割り当て目標として考慮される順を決定する。割り当て中のデータアイ

テムは、分割不可能なグループであっても良い。この場合、グループ全体のために十分な空間を有するデータ割り当て目標として考慮される最初のデータ記憶ユニットは、グループ全体を割り当てられる。代替で、グループは分割できる。したがって、データアイテムは、各々が更なる利用可能な記憶空間を有しなくなるまで、割り当て目標に割り当てられ、次に次の割り当て目標が満たされる。

【０１２１】

割り当てモジュールは、図２のライタ１６により又は特定の他のコンポーネントにより提供される機能であっても良く、データ記憶ユニット間を区別するために、データアイテムを格納しているデータソースと割り当てられるべきデータアイテムのデータソースとの間の類似度を用いることにより、データアイテムを割り当てるべきデータ記憶ユニットを選択しても良い。割り当てモジュールは、データ記憶ユニット間を区別するために、データアイテムを有するデータソースと割り当てられるべきデータアイテムのデータソースとの間の類似度により決定される順序で、データ記憶ユニットを割り当て目標として選択しても良い。割り当て目標が選択されると、割り当てられるべきデータアイテムは、（割り当てられるべきデータアイテムが単一のデータ記憶ユニットに格納されるべきか又は分割可能かに依存して）割り当てられるべきデータアイテムの一部又は全部を格納するために十分な空間が存在する場合、次の割り当て目標が考慮される前に、該選択された割り当て目標に割り当てられる。例えば＜割り当てモジュールがＥＤＳｎｅｗからのデータアイテムを書き込むために目標サーバを選択しようとするとき、先ず、ＳＤｎｅｗ及び残りのデータソース（又はそれらのセマンティック記述）の中の類似性比較を有する列を探すことにより、類似性行列１７を調べ、最高の類似性を有するＥＤＳｉを発見する。例えば、以下の通りである。

［表４］ＳＤｎｅｗに対する計算された類似度を含む類似性行列１７から抽出した列

【表４】

	SDnew
SD1	0.7
SD2	0.3
SD3	0.4
…	…
SDnew	1

【０１２２】

便宜のため、類似性リストに対して強制的にソートが行われても良い。あまり分散していないデータセットを優先することにより、タイブレイクが行われても良い。例えば、セグメントがＭ個のサーバノードに跨るデータセットは、Ｋ個のサーバノードに広がっているデータセットに比べて低いランクであると考えられる（ここで、Ｍ＞Ｋ）。

【０１２３】

表４に示した特定の例では、ＳＤ１は、ＳＤｎｅｗのセマンティック記述と最も近いセマンティック記述を有する。例えば次の通りである。

【０１２４】

ｓｉｍｉｌａｒｉｔｙ（ＳＤ１，ＳＤｎｅｗ）＝０．７

次に、ライタ１６は、Ｋｅｙ－Ｎｏｄｅ＿ＩＤテーブルからＥＳＤ１のＮｏｄｅ＿ＩＤ情報を検索し、ＥＳＤｎｅｗからのデータアイテムをＥＳＤ１が位置するのと同じノード、本例ではＮ１に割り当てる。

【０１２５】

割り当て目標がＥＤＳｎｅｗからのデータアイテムを収容できない又はＥＤＳｎｅｗからの全てのデータアイテムを収容できない状況では、次の割り当て目標を選択する必要があっても良い。

【０１２６】

ライタ１６は、割り当て中のデータアイテムのための記憶空間を生成するために、既存のデータアイテムを再割り当てするよう構成されても良い。例えば、割り当て目標として選択されたデータ記憶ユニットが２つのデータソースからのデータアイテムを格納し、２つのデータソース間の類似度がデータソースのうちの１つと割り当てられるできデータアイテムのデータソースとの間の類似度より小さい場合、割り当て等れるべきデータアイテムのデータソースに対して最も低い類似度を有する、２つのデータソースのうちの一方からのデータアイテムは、（例えば、ライタにより、現在の割り当ての後に、）データ記憶ユニットから削除され他の場所に割り当てられる。

【０１２７】

上述の例を続けることにより再割り当て手順を説明する。表３から分かるように、ＥＤＳ１及びＥＤＳ２の両方からのデータアイテムは最初にＮ１に置かれる。Ｎ１がＥＤＳｎｅｗからのデータアイテムを置くために十分な空間を有しない場合、ライタ１６は、他のＥＤＳｉが同じデータ記憶ユニットに配置されているかを調べ、ＥＤＳ２を見付ける。ＥＤＳ１とＥＤＳ２との間の類似性について類似性行列１７を更にさかのぼると、

ｓｉｍｉｌａｒｉｔｙ（ＳＤ１，ＳＤ２）＝０．５

本例では、ｓｉｍｉｌａｒｉｔｙ（ＳＤ１，ＳＤｎｅｗ）＞ｓｉｍｉｌａｒｉｔｙ（ＳＤ１，ＳＤ２）であり、したがって、ＥＤＳ２は、次に最も近い外部データソースに再割り当てされるべきである。同じ類似性行列１７を用いて、ライタ１６は、ｓｉｍｉｌａｒｉｔｙ（ＳＤ２，ＳＤ３）＝０．５が次の選択肢であることを発見する。したがって、ＥＤＳ２はＮ２に再割り当てされる。この再割り当ては、同じ外部データソースＩＤ（メタデータレジストリ内でＫｅｙ値として示される）を有する全てのＲＤＦトリプルをＮ２にコピーし、次にＮ１から元のデータセットを削除することにより達成される。

【０１２８】

代替で、ｓｉｍｉｌａｒｉｔｙ（ＳＤ１，ＳＤｎｅｗ）＜ｓｉｍｉｌａｒｉｔｙ（ＳＤ１，ＳＤ２）の場合、ＥＳＤｎｅｗからのデータアイテムは、次のＮｏｄｅ＿ＩＤに割り当てられる。次のＮｏｄｅ＿ＩＤは、次に最も近い外部データソースが位置する場所であり、例えばＳＤ３が位置するＮ２である

位置が決定されると、Ｋｅｙ－Ｎｏｄｅ＿ＩＤテーブルは相応して更新される。

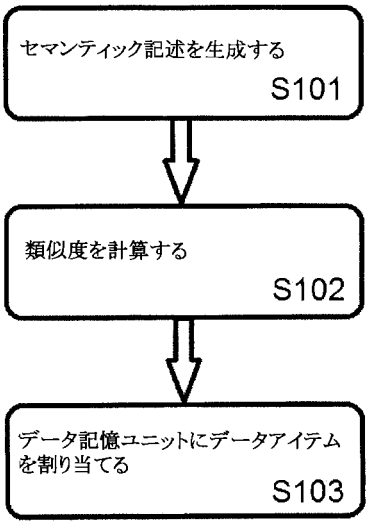
【符号の説明】

【０１２９】

１０　データ集約装置
１２　リーダ
１４　プロセッサ
１６　ライタ
１８　メタデータレジストリ
２０　データ記憶装置Ｎ＿１～Ｎ＿ｎ
３０　外部データソース＿１～ｎ

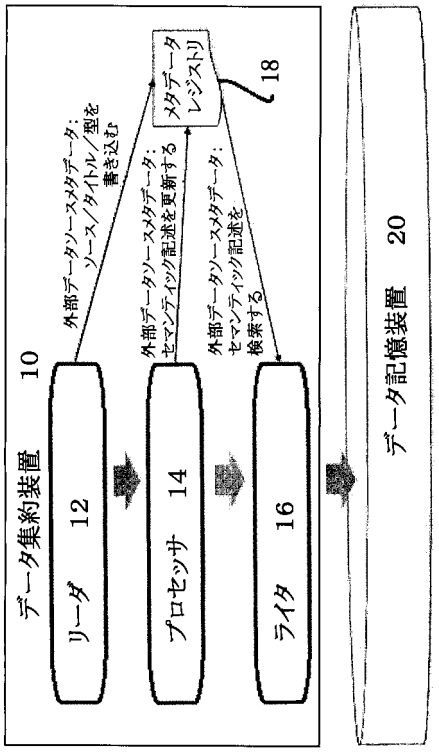
【図１】

本発明を具現化する処理のフローを示す図



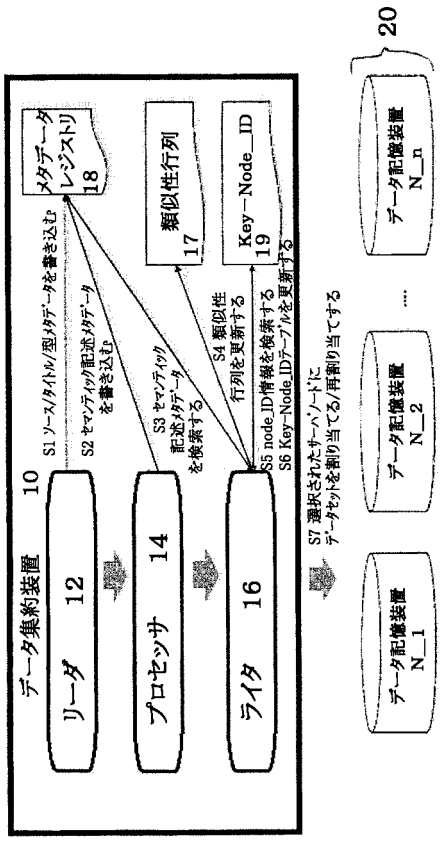
【図２】

本発明を具現化するデータ集約装置の例示的なアーキテクチャを示す図



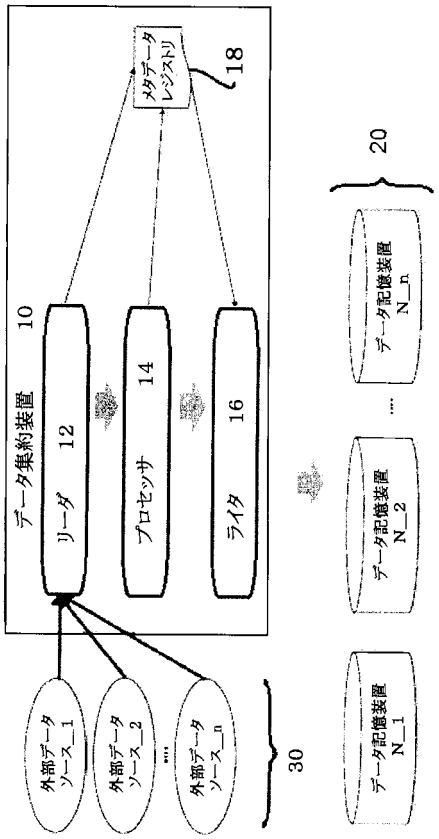
【図３】

類似性行列とKey__Node－IDテーブルとの間の相互作用を更に説明するシステムアーキテクチャを示す図



【図４】

本発明を具現化する処理が本発明を具現化する装置によりどのように実行され得るかを示す図



フロントページの続き

(72)発明者　ヒュー・ボ
イギリス国，エスオー２２　５イージー，ウィンチェスター，フルフロッド・コート，フラット６号