(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6434162号
(P6434162)
(45)発行日　平成30年12月5日(2018.12.5)　(24)登録日　平成30年11月16日(2018.11.16)

(51)Int.Cl.　F I
G06F 17/30 (2006.01)　G06F 17/30 350C
G06F 17/30 414A
G06F 17/30 412

請求項の数 16　(全 39 頁)

(21)出願番号　特願2017-547261(P2017-547261)
(86)(22)出願日　平成27年10月28日(2015.10.28)
(86)国際出願番号　PCT/JP2015/080446
(87)国際公開番号　WO2017/072890
(87)国際公開日　平成29年5月4日(2017.5.4)
審査請求日　平成30年1月26日(2018.1.26)

(73)特許権者　000003078
株式会社東芝
東京都港区芝浦一丁目１番１号
(73)特許権者　301063496
東芝デジタルソリューションズ株式会社
神奈川県川崎市幸区堀川町７２番地３４
(74)代理人　110002147
特許業務法人酒井国際特許事務所
(72)発明者　浜田　伸一郎
東京都港区芝浦一丁目１番１号　株式会社東芝内
(72)発明者　小野　聡一郎
東京都港区芝浦一丁目１番１号　株式会社東芝内

最終頁に続く

(54)【発明の名称】データ管理システム、データ管理方法およびプログラム

(57)【特許請求の範囲】
【請求項１】
蓄積するデータの特徴ベクトルである事例ベクトルに類似する周辺ベクトルを生成し、生成した前記周辺ベクトルに対応する前記事例ベクトルを特定するための索引情報を構築する索引構築部と、
任意の特徴ベクトルであるクエリベクトルを指定した検索要求に応じて、前記索引情報を用いて、前記クエリベクトルと完全一致する前記周辺ベクトルに対応する前記事例ベクトルを特定し、特定した前記事例ベクトルに基づく検索結果を出力する検索部と、を備え、
前記索引構築部は、少なくとも、前記周辺ベクトルを格納する第１列と、該周辺ベクトルに対応する前記事例ベクトルに関する情報を格納する第２列とを列要素に持つテーブルと、該テーブルにおける前記第１列に対する索引とを含む前記索引情報を構築し、
前記検索部は、前記索引を用いて、前記クエリベクトルと完全一致する前記周辺ベクトルに対応する前記テーブルのレコードを求め、求めたレコードの前記第２列に格納された情報に基づいて前記事例ベクトルを特定する、データ管理システム。
【請求項２】
前記テーブルのデータ構造として、前記第１列に格納される前記周辺ベクトルをキーとし、前記第２列に格納される情報を値とする連想配列または連続メモリ配置型配列を用いる、請求項１に記載のデータ管理システム。
【請求項３】

前記索引構築部は、前記第１列および前記第２列に加えてさらに、前記周辺ベクトルの前記事例ベクトルに対する類似度を格納する第３列を列要素に持つ前記テーブルと、該テーブルにおける前記第１列および前記第３列に対する複合索引とを含む前記索引情報を構築し、

前記検索部は、前記複合索引を用いて、前記クエリベクトルと完全一致する前記周辺ベクトルであって、前記類似度が条件を満たす前記周辺ベクトルに対応する前記テーブルのレコードを求め、求めたレコードの前記第２列に格納された情報に基づいて前記事例ベクトルを特定する、請求項１に記載のデータ管理システム。

【請求項４】

前記テーブルのデータ構造として、前記第１列に格納される前記周辺ベクトルおよび前記第３列に格納される前記類似度をキーとし、前記第２列に格納される情報を値とする連想配列または連続メモリ配置型配列を用いる、請求項３に記載のデータ管理システム。

【請求項５】

蓄積するデータの特徴ベクトルである事例ベクトルに類似する周辺ベクトルを生成し、生成した前記周辺ベクトルに対応する前記事例ベクトルを特定するための索引情報を構築する索引構築部と、

任意の特徴ベクトルであるクエリベクトルを指定した検索要求に応じて、前記索引情報を用いて、前記クエリベクトルと完全一致する前記周辺ベクトルに対応する前記事例ベクトルを特定し、特定した前記事例ベクトルに基づく検索結果を出力する検索部と、を備え、

前記索引構築部は、前記周辺ベクトルを格納する第１列と、該周辺ベクトルの前記事例ベクトルに対する類似度を格納する第２列とを列要素に持つ第１テーブルと、該第１テーブルのレコードの行ＩＤを格納する第１列と、該レコードの前記周辺ベクトルに対応する前記事例ベクトルに関する情報を格納する第２列とを列要素に持つ第２テーブルと、前記第１テーブルにおける前記第１列および前記第２列に対する複合索引とを含む前記索引情報を構築し、

前記検索部は、前記複合索引を用いて、前記クエリベクトルと完全一致する前記周辺ベクトルであって、前記類似度が条件を満たす前記周辺ベクトルに対応する前記第１テーブルのレコードの行ＩＤを求め、求めた行ＩＤが格納された前記第２テーブルのレコードの第２列に格納された情報に基づいて前記事例ベクトルを特定する、データ管理システム。

【請求項６】

蓄積するデータの特徴ベクトルである事例ベクトルに類似する周辺ベクトルを生成し、生成した前記周辺ベクトルに対応する前記事例ベクトルを特定するための索引情報を構築する索引構築部と、

任意の特徴ベクトルであるクエリベクトルを指定した検索要求に応じて、前記索引情報を用いて、前記クエリベクトルと完全一致する前記周辺ベクトルに対応する前記事例ベクトルを特定し、特定した前記事例ベクトルに基づく検索結果を出力する検索部と、を備え、

前記索引構築部は、前記周辺ベクトルの値と該周辺ベクトルの前記事例ベクトルに対する類似度の条件とに従って、前記周辺ベクトルに対応する前記事例ベクトルに関する情報を探索する複合索引を前記索引情報として構築し、

前記検索部は、前記複合索引を用いて、前記クエリベクトルと完全一致する前記周辺ベクトルであって、前記類似度が条件を満たす前記周辺ベクトルに対応する前記事例ベクトルを特定する、データ管理システム。

【請求項７】

蓄積するデータの特徴ベクトルである事例ベクトルに類似する周辺ベクトルを生成し、生成した前記周辺ベクトルに対応する前記事例ベクトルを特定するための索引情報を構築する索引構築部と、

任意の特徴ベクトルであるクエリベクトルを指定した検索要求に応じて、前記索引情報を用いて、前記クエリベクトルと完全一致する前記周辺ベクトルに対応する前記事例ベク

トルを特定し、特定した前記事例ベクトルに基づく検索結果を出力する検索部と、を備え、

前記検索部は、前記検索要求が出力件数の指定を含む場合に、前記事例ベクトルに対する前記周辺ベクトルの類似度の条件を厳しい方から段階的に変化させながら、前記クエリベクトルと完全一致する前記周辺ベクトルに対応する前記事例ベクトルを特定する処理を、特定した前記事例ベクトルの総数が指定された前記出力件数以上になるまで繰り返し、特定した前記事例ベクトルの総数が指定された前記出力件数以上になると前記処理を停止して、特定した前記事例ベクトルに基づく前記出力件数に近い件数の検索結果を出力する、データ管理システム。

【請求項８】

蓄積するデータの特徴ベクトルである事例ベクトルに類似する周辺ベクトルを生成し、生成した前記周辺ベクトルに対応する前記事例ベクトルを特定するための索引情報を構築する索引構築部と、

任意の特徴ベクトルであるクエリベクトルを指定した検索要求に応じて、前記索引情報を用いて、前記クエリベクトルと完全一致する前記周辺ベクトルに対応する前記事例ベクトルを特定し、特定した前記事例ベクトルに基づく検索結果を出力する検索部と、を備え、

前記検索部は、前記索引情報を用いた検索方式と、元の特徴ベクトルまたは縮約ベクトル空間に写像された特徴ベクトルを用いて線形検索する検索方式と、を含む複数の検索方式のうち、前記事例ベクトルを含むデータテーブルのレコード数に応じて選択された検索方式により前記事例ベクトルの検索を行う、データ管理システム。

【請求項９】

蓄積するデータの特徴ベクトルである事例ベクトルに類似する周辺ベクトルを生成し、生成した前記周辺ベクトルに対応する前記事例ベクトルを特定するための索引情報を構築する索引構築部と、

任意の特徴ベクトルであるクエリベクトルを指定した検索要求に応じて、前記索引情報を用いて、前記クエリベクトルと完全一致する前記周辺ベクトルに対応する前記事例ベクトルを特定し、特定した前記事例ベクトルに基づく検索結果を出力する検索部と、を備え、

前記検索部は、前記索引情報を用いた検索方式と、元の特徴ベクトルまたは縮約ベクトル空間に写像された特徴ベクトルを用いて線形検索する検索方式と、を含む複数の検索方式のうちの任意の検索方式を多段階で組み合わせて、前記事例ベクトルの検索を行う、データ管理システム。

【請求項１０】

前記索引構築部は、前記事例ベクトルを縮約ベクトル空間に写像した縮約事例ベクトルに類似する縮約周辺ベクトルを生成し、生成した前記縮約周辺ベクトルに対応する前記事例ベクトルを特定するための前記索引情報を構築し、

前記検索部は、前記クエリベクトルを前記索引構築部と共通の縮約ベクトル空間に写像した縮約クエリベクトルと完全一致する前記縮約周辺ベクトルに対応する前記事例ベクトルを特定する、請求項１乃至９のいずれか一項に記載のデータ管理システム。

【請求項１１】

前記事例ベクトルから前記縮約事例ベクトルへの写像、および、前記クエリベクトルから前記縮約クエリベクトルへの写像に、ＬＳＨ（Locality-Sensitive Hashing）技術を用いる、請求項１０に記載のデータ管理システム。

【請求項１２】

前記ＬＳＨ技術として、前記事例ベクトルを各次元が二値のみを取る二値ベクトルである前記縮約事例ベクトルに変換するとともに、前記クエリベクトルを各次元が二値のみを取る二値ベクトルである前記縮約クエリベクトルに変換するビットワイズＬＳＨを用いる、請求項１１に記載のデータ管理システム。

【請求項１３】

前記索引構築部は、二値ベクトルである前記縮約周辺ベクトルを生成し、生成した前記縮約周辺ベクトルを二進数と見立てて整数として格納した前記索引情報を構築し、
前記検索部は、二値ベクトルである前記縮約クエリベクトルを、二進数と見立てて整数として用いて前記縮約クエリベクトルと完全一致する前記縮約周辺ベクトルに対応する前記事例ベクトルを特定する、請求項１２に記載のデータ管理システム。
【請求項１４】
前記ＬＳＨ技術として、前記事例ベクトルを整数ベクトルである前記縮約事例ベクトルに変換するとともに、前記クエリベクトルを整数ベクトルである前記縮約クエリベクトルに変換する直積量子化ＬＳＨを用いる、請求項１１に記載のデータ管理システム。
【請求項１５】
データ管理システムにおいて実行されるデータ管理方法であって、
蓄積するデータの特徴ベクトルである事例ベクトルに類似する周辺ベクトルを生成し、生成した前記周辺ベクトルに対応する前記事例ベクトルを特定するための索引情報を構築する索引構築ステップと、
任意の特徴ベクトルであるクエリベクトルを指定した検索要求に応じて、前記索引情報を用いて、前記クエリベクトルと完全一致する前記周辺ベクトルに対応する前記事例ベクトルを特定し、特定した前記事例ベクトルに基づく検索結果を出力する検索ステップと、を含み、
前記索引構築ステップでは、少なくとも、前記周辺ベクトルを格納する第１列と、該周辺ベクトルに対応する前記事例ベクトルに関する情報を格納する第２列とを列要素に持つテーブルと、該テーブルにおける前記第１列に対する索引とを含む前記索引情報を構築し、
前記検索ステップでは、前記索引を用いて、前記クエリベクトルと完全一致する前記周辺ベクトルに対応する前記テーブルのレコードを求め、求めたレコードの前記第２列に格納された情報に基づいて前記事例ベクトルを特定する、データ管理方法。
【請求項１６】
コンピュータに、
蓄積するデータの特徴ベクトルである事例ベクトルに類似する周辺ベクトルを生成し、生成した前記周辺ベクトルに対応する前記事例ベクトルを特定するための索引情報を構築する索引構築部の機能と、
任意の特徴ベクトルであるクエリベクトルを指定した検索要求に応じて、前記索引情報を用いて、前記クエリベクトルと完全一致する前記周辺ベクトルに対応する前記事例ベクトルを特定し、特定した前記事例ベクトルに基づく検索結果を出力する検索部の機能と、を実現させるためのプログラであって、
前記索引構築部は、少なくとも、前記周辺ベクトルを格納する第１列と、該周辺ベクトルに対応する前記事例ベクトルに関する情報を格納する第２列とを列要素に持つテーブルと、該テーブルにおける前記第１列に対する索引とを含む前記索引情報を構築し、
前記検索部は、前記索引を用いて、前記クエリベクトルと完全一致する前記周辺ベクトルに対応する前記テーブルのレコードを求め、求めたレコードの前記第２列に格納された情報に基づいて前記事例ベクトルを特定する、プログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明の実施形態は、データ管理システム、データ管理方法およびプログラムに関する。
【背景技術】
【０００２】
近年、情報通信技術の進展に伴って、多種多様なデータの収集や蓄積が可能となり、ビッグデータ分析やビッグメディア解析などといった大規模データを対象とする情報処理技術が注目を浴びている。こうした大規模データを取り扱うシステムでは、データ規模の加

速度的な拡大に伴う計算量の肥大化がサービス低下に繋がるため、いかに計算量を削減できるかが重要な課題となっている。

【０００３】

データベース検索などのデータ検索では、画像や音楽などのメディア検索を高速に行う方法として、多次元の特徴ベクトルを用いた類似検索が行われる。この類似検索では、特徴ベクトル間の類似度計算を含むベクトル近傍検索、すなわち、ある特徴ベクトル（以下、これを「クエリベクトル」という）に近い特徴ベクトル群を、検索対象となる特徴ベクトル（以下、これを「事例ベクトル」という）群の中から見つけ出す処理が計算量の多くを占めている。このため、ベクトル近傍検索の計算量を削減してデータ検索の実行時間を短縮できるようにすることが求められている。

【先行技術文献】

【特許文献】

【０００４】

【特許文献１】特開２０００－３５９６５号公報

【特許文献２】特開２００１－５２０２４号公報

【発明の概要】

【発明が解決しようとする課題】

【０００５】

本発明が解決しようとする課題は、ベクトル近傍検索の計算量を削減してデータ検索の実行時間を短縮できるデータ管理システム、データ管理方法およびプログラムを提供することである。

【課題を解決するための手段】

【０００６】

実施形態のデータ管理システムは、索引構築部と、検索部と、を備える。索引構築部は、蓄積するデータの特徴ベクトルである事例ベクトルに類似する周辺ベクトルを生成し、生成した前記周辺ベクトルに対応する前記事例ベクトルを特定するための索引情報を構築する。検索部は、任意の特徴ベクトルであるクエリベクトルを指定した検索要求に応じて、前記索引情報を用いて、前記クエリベクトルと完全一致する前記周辺ベクトルに対応する前記事例ベクトルを特定し、特定した前記事例ベクトルに基づく検索結果を出力する。前記索引構築部は、少なくとも、前記周辺ベクトルを格納する第１列と、該周辺ベクトルに対応する前記事例ベクトルに関する情報を格納する第２列とを列要素に持つテーブルと、該テーブルにおける前記第１列に対する索引とを含む前記索引情報を構築し、前記検索部は、前記索引を用いて、前記クエリベクトルと完全一致する前記周辺ベクトルに対応する前記テーブルのレコードを求め、求めたレコードの前記第２列に格納された情報に基づいて前記事例ベクトルを特定する。

【図面の簡単な説明】

【０００７】

【図１】図１は、第１実施形態のデータ管理システムの概要を示すシステム構成図である。

【図２】図２は、データテーブルの具体例を示す図である。

【図３】図３は、メディアデータが静止画である場合のデータ登録器による処理手順の一例を示すフローチャートである。

【図４】図４は、メディアデータが動画である場合のデータ登録器による処理手順の一例を示すフローチャートである。

【図５】図５は、索引構築器の構成例を示すブロック図である。

【図６】図６は、ＬＳＨ即値テーブルの一例を示す図である。

【図７】図７は、ＬＳＨ即値索引情報生成器による処理手順の一例を示すフローチャートである。

【図８】図８は、ＬＳＨ近傍展開テーブルの一例を示す図である。

【図９】図９は、ＬＳＨ近傍展開テーブルを２つに分割することで正規化した例を示す図

である。
【図１０】図１０は、データベース複合索引の一例を示す図である。
【図１１】図１１は、ＬＳＨ近傍展開索引情報生成器による処理手順の一例を示すフローチャートである。
【図１２】図１２は、連想配列と連続メモリ配置型配列を説明する図である。
【図１３】図１３は、検索器の構成例を示すブロック図である。
【図１４】図１４は、ベクトル類似性判定部の入出力関係を示す図である。
【図１５】図１５は、ベクトル類似性判定部による処理手順の一例を示すフローチャートである。
【図１６】図１６は、厳密検索器による処理手順の一例を示すフローチャートである。
【図１７】図１７は、線形ＬＳＨ検索器による処理手順の一例を示すフローチャートである。
【図１８】図１８は、データベース索引ＬＳＨ検索器による処理手順の一例を示すフローチャートである。
【図１９】図１９は、データベース索引ＬＳＨ検索＋厳密検索器による処理手順の一例を示すフローチャートである。
【図２０】図２０は、データベース索引ＬＳＨ検索＋線形ＬＳＨ検索器による処理手順の一例を示すフローチャートである。
【図２１】図２１は、第２実施形態の検索器の構成例を示すブロック図である。
【図２２】図２２は、クエリ摂動型ＬＳＨ検索器による処理手順の一例を示すフローチャートである。
【図２３】図２３は、第３実施形態の索引構築器の構成例を示すブロック図である。
【図２４】図２４は、ＰＱＬＳＨ近傍展開テーブルの一例を示す図である。
【図２５】図２５は、ＰＱＬＳＨ近傍展開索引情報生成器による処理手順の一例を示すフローチャートである。
【図２６】図２６は、第３実施形態の検索器の構成例を示すブロック図である。
【図２７】図２７は、データベース索引ＰＱＬＳＨ検索器による処理手順の一例を示すフローチャートである。
【図２８】図２８は、データ管理システムのハードウェア構成例を示すブロック図である。

【発明を実施するための形態】
【０００８】
以下、実施形態のデータ管理システム、データ管理方法およびプログラムを、図面を参照して詳細に説明する。
【０００９】
実施形態のデータ管理システムは、大規模データを効率よく管理・検索するためのシステムである。データベース管理システムにみられるような大規模データを管理する従来のシステムは、一般的に、下位層としてディスクアクセスなどを最適化するデータ配置機構、上位層として検索条件に基づいて大規模データを高速検索するための索引機構が搭載されている。索引のアルゴリズムには、Ｂツリーなどの木構造アルゴリズムや、一般的なハッシュアルゴリズムが主に用いられている。
【００１０】
データベースで利用できる検索条件は、多くの場合、実数・整数・文字列・日付などの基本型（以下、これらを総称して「ＤＢ基本型」と呼ぶ）に関する四則演算・集合演算などで構成された論理式である。ベクトルやビット列（すなわち二値ベクトル）同士の類似性については、一部の例外を除き、通常、検索条件として用いることができない。その背景には、ベクトルの類似度計算を含む検索条件を効率化する索引が考案されていないことがある。
【００１１】
上述の一部の例外とは、低次元ベクトルを対象とした類似度計算である。多くの著名な

データベース管理システムでは、主に時空間データを管理する用途を想定して、２～４次元程度の低次元のベクトルの類似度計算を検索条件に含めることができるようになっている。この類似度計算の実現には、空間木（空間分割法）と呼ばれる索引手法が用いられており、これにより高速な検索が可能である。ただしベクトルが低次元でない場合は、索引サイズの肥大化によって高速化の効果が失われ、実行時間が通常の線形検索と変わらなくなることが知られている。実施形態で想定する特徴ベクトルは、例えば機械学習用途を想定した数百～数億次元の高次元ベクトルであるため、空間木による方法を用いても高速な検索は実現できない。

【００１２】

しかし、高次元ベクトルの完全一致を高速照合することは可能である。この場合、２つのベクトルの各次元の要素に対する等号条件をＡＮＤで結合した条件で検索することと等価である。すなわち、この検索条件はスカラ演算のみで構成されており、ベクトルの類似度計算は含まれていないため、Ｂツリーなどの索引を利用できる。

【００１３】

以上のように、従来の一般的なデータベース管理システムは、ＤＢ基本型で構成された検索条件での大規模データの高速検索が可能であるが、低次元でないベクトル間の類似度計算を含むベクトル近傍検索を高速実行することはできない。そこで、以下に示す実施形態では、ベクトル近傍検索の計算量を削減することでベクトル近傍検索の高速実行を可能とし、データ検索の実行時間を短縮できる新規なデータ管理システムを提案する。

【００１４】

なお、ベクトル間の類似度を表す指標としては内積と距離がある。ベクトル間の内積値を求める内積計算とベクトル間の距離を求める距離計算は、意味としても計算量としてもほぼ同じである。２つのベクトルが類似している場合、内積値は大きくなるが、距離は小さくなるという点が異なるだけである。つまり、ベクトル間の内積値が大きいことと、ベクトル間の距離が小さいことは、ともにベクトル間の類似度が高いことを意味する。以下では、ベクトル間の類似度を距離で表すものとして説明するが、距離を内積に置き換えてもよい。この場合、ベクトル間の距離が小さいほど、つまり、ベクトル間の類似度が高いほど、内積値が大きくなるものと考えればよい。

【００１５】

ベクトル近傍検索の計算量を削減する方法として、ＬＳＨ（Locality-Sensitive Hashing）を用いることが考えられる。ＬＳＨは、与えられたベクトル群を、離散値のみを取る縮約ベクトル空間に写像する技術である（例えば、下記の参考文献１参照）。このＬＳＨによる写像は、写像前の空間におけるベクトル間の距離の相対的な大きさが、写像後の空間においてもよく保存されているという性質がある。したがって、写像前のベクトル空間でベクトル間距離を計算する代わりに、写像後のベクトル空間でベクトル間距離を計算することで、計算を効率化することができる。ただし、距離の大小関係を完全に保存するわけではないため、得られるのは近似解である。

参考文献１：Anshumali Shrivastava and Ping Li, “Asymmetric LSH (ALSH) for Sublinear Time Maximum Inner Product Search (MIPS)”, Advances in Neural Information Processing Systems, 2014.

【００１６】

ＬＳＨアルゴリズムとしては、これまでに、例えば、下記の参考文献２に示されるSimHash（Random projection）や、下記の参考文献３に示されるSpectral Hashingなど、様々なアルゴリズムが考案されている。

参考文献２：Moses S. Charikar, “Similarity Estimation Techniques from Rounding Algorithms”, Proceedings of the 34th Annual ACM Symposium on Theory of Computing, pp. 380?388, doi：10, 1145／509907, 509965.

参考文献３：Yair Weiss, Antonio Torralba and Rob Fergus, “Spectral Hashing”, Advances in neural information processing systems, 2009.

【００１７】

ＬＳＨアルゴリズムは、大きく、ビットワイズ方式と直積量子化（Product Quantization）方式とに二分できる。ビットワイズ方式のアルゴリズムは、与えられたベクトルに対する写像結果として二値ベクトルを出力する。一方、直積量子化方式のアルゴリズムは、与えられたベクトルに対する写像結果として整数ベクトル（整数の要素からのみ構成される多値ベクトル）を出力する。また、直積量子化方式のアルゴリズムの場合、ベクトルの各次元ごとに、距離に対する重みに相当する情報（各次元の変化に対してベクトルの距離がどれくらい変化するか）も合わせて生成される。

【００１８】

上述のように、ＬＳＨによって写像されたベクトルを用いることで、ベクトル間距離の計算を効率化することができる。特に二値ベクトルの場合、距離計算処理に、xorなどのビット演算命令、あるいはpopcountなどの専用ＣＰＵ命令を用いることができるため、大幅な計算量削減が可能である。

【００１９】

しかし、ＬＳＨを用いたとしても、ベクトル間距離の計算そのものを回避することはできない。従来のデータベース管理システムでは、上述のように、低次元でないベクトルの距離計算を伴う検索処理に対して有効な索引アルゴリズムが考案されていないため、線形検索を行う必要があり、計算量オーダーをサブリニアにすることができない。その結果、検索対象となる事例ベクトルの数が例えば１億など大規模な場合には、大きな検索時間を要することとなる。そこで、低次元でないベクトルの距離計算を伴う検索処理に対して有効な索引アルゴリズムを導入した新規なデータ管理システムを提案する。

【００２０】

ここで、実施形態の基本原理を説明する。実施形態のデータ管理システムは、まず事前処理として、各事例ベクトルのそれぞれについて、事例ベクトルに類似するベクトル、つまり事例ベクトルに対する距離が所定値以下のベクトル（以下、これを「周辺ベクトル」と呼ぶ）群を生成するとともに、周辺ベクトルから事例ベクトルを特定するための索引情報を構築し、事例ベクトルを含むデータと索引情報とをデータベースに保存する。そして、検索時には、索引情報に基づいて、クエリベクトルと完全一致する周辺ベクトルに対応する事例ベクトルを特定し、特定した事例ベクトルに基づいて、検索要求に対する検索結果を出力する。

【００２１】

事前処理で事例ベクトルに類似する周辺ベクトルを合理的に生成するには、取り扱うベクトルの次元数および各要素値集合のカーディナリティがいずれも小さい必要がある。この条件が満たされなければ、周辺ベクトル数が爆発し、結果として検索時間を短縮できない。この条件を達成するために、上述したＬＳＨを用いる。ＬＳＨにより生成されるベクトルの要素は、ビットワイズ方式の場合は二値、直積量子化方式の場合は多値を持つ離散値を取り、ＬＳＨの設定にもよるが、多くの場合、要素値集合のカーディナリティおよび次元数が小さい離散ベクトル（縮約ベクトル）が生成される。したがって、基本原理として説明した上述の方法を実行する前に、ＬＳＨを用いて事例ベクトル群およびクエリベクトルを縮約ベクトルに変換しておき、これらを用いて、基本原理として説明した上述の方法による検索を行えばよい。ただし、ＬＳＨを用いるため、検索結果は近似解となる。

【００２２】

なお、事例ベクトル群やクエリベクトルが元々上記条件を満たすベクトルであるならば、ＬＳＨを用いたベクトル変換を事前に行う必要はなく、基本原理として説明した上述の方法のみを実行すればよい。

【００２３】

また、基本原理として説明した上述の方法では、事例ベクトルに対する距離が所定値以下の周辺ベクトルを生成したが、クエリベクトルに対する距離が所定値以下の周辺ベクトルを生成する構成としてもよい。この場合、検索時に、クエリベクトルに対する距離が所定値以下の周辺ベクトルを生成して、周辺ベクトルと完全一致する事例ベクトルを特定し、特定した事例ベクトルに基づいて、検索要求に対する検索結果を出力する。

【００２４】
以下では、ベクトル間距離の計算が検索条件に含まれるクエリを処理してメディア検索を行うデータ管理システムへの適用例について具体的に説明する。メディア検索の応用題材としては、顔画像（映像含む）検索を取り上げる。顔画像検索は、システムに予め登録された画像・映像などのメディア群の中から、クエリとして与えられた画像・映像に含まれる顔と似た顔を含む画像・映像の箇所を見つけ出す処理である。なお、ここでは応用題材として顔画像検索を取り上げるが、実施形態のデータ管理システムは、例えば、音楽検索、物体画像検索、シーン画像検索、テキスト意味検索、センサパタン検索、株価パタン検索、電力使用パタン検索など、様々なメディア検索やセンサデータ検索に応用することができる。
【００２５】
＜第１実施形態＞
まず、第１実施形態のデータ管理システムについて説明する。本実施形態では、写像処理にはビットワイズＬＳＨを用いるものとし、写像後の二値ベクトル（ハッシュ）間の距離としてハミング距離を用いるものとする。ただし、上述したように、取り扱う事例ベクトルおよびクエリベクトルが元々低次元の二値ベクトルであるならば、ビットワイズＬＳＨによる写像は不要であり、後述の事例ハッシュ（縮約事例ベクトル）として事例ベクトルをそのまま用い、クエリハッシュ（縮約クエリベクトル）としてクエリベクトルをそのまま用いればよい。また、この場合、後述の周辺ハッシュ（縮約周辺ベクトル）として、事例ベクトルを元に生成される周辺ベクトルを用いればよい。
【００２６】
図１は、第１実施形態のデータ管理システムの概要を示すシステム構成図である。本実施形態のデータ管理システムは、図１に示すように、データ登録器１００と、索引構築器２００と、検索器３００と、データベース４００とを備える。
【００２７】
データ登録器１００は、顔を含む画像や映像などのメディアデータ群１０を外部から受け取り、受け取ったメディアデータ群１０を対象として顔に関する解析を行い、顔特徴に関する特徴ベクトル（事例ベクトル）群を含むデータテーブル２０をデータベース４００内に生成する。
【００２８】
索引構築器２００は、索引構築命令文３０を外部から受け取ると、受け取った索引構築命令文３０に対応するデータテーブル２０をデータベース４００から取り出し、索引構築命令文３０の命令に従って、顔特徴ベクトルの索引として用いる索引情報４０をデータベース４００内に生成する。
【００２９】
検索器３００は、顔特徴に関するクエリベクトルを含む拡張ＳＱＬ５０（検索要求の一例）を外部から受け取ると、データベース４００から該当するデータテーブル２０および索引情報４０を取得し、拡張ＳＱＬ５０に含まれるクエリベクトルと類似する顔特徴ベクトルが含まれる検索結果データセット６０を出力する。
【００３０】
まず、データ登録器１００の詳細を説明する。図２は、データ登録器１００がデータベース４００内に生成するデータテーブル２０の具体例を示す図である。データ登録器１００が外部から受け取るメディアデータ群１０に含まれるメディアデータは、dir/a.jpg、dir/b.jpg、dir/a.mpgの３種類のファイルであるものとする。このうち、dir/a.jpg、dir/b.jpgは静止画ファイルであり、dir/a.mpgは動画ファイルである。
【００３１】
データ登録器１００は、外部からメディアデータ群１０を受け取ると、このメディアデータ群１０に含まれるメディアデータそれぞれについて、以下に示す処理を実行する。
【００３２】
図３は、メディアデータが静止画である場合のデータ登録器１００による処理手順の一

例を示すフローチャートである。データ登録器１００は、メディアデータが静止画である場合、例えば以下のステップＳ１０１～ステップＳ１０５の処理を実行して、メディアデータをデータテーブル２０に登録する。

【００３３】

ステップＳ１０１：データ登録器１００は、入力された静止画に対して顔領域抽出処理を行う。この結果、入力された静止画から、顔が写っている可能性の高い画像領域（顔画像領域）がすべて抽出される。なお、顔領域抽出処理は既存技術を用いればよいため、ここでは詳細な説明を省略する。

【００３４】

ステップＳ１０２：データ登録器１００は、ステップＳ１０１で抽出した顔画像領域を順に取り出す。

【００３５】

ステップＳ１０３：データ登録器１００は、ステップＳ１０２で取り出した顔画像領域に対して特徴生成処理を行って特徴ベクトル（事例ベクトル）を得る。なお、特徴生成処理には既存技術を用いればよいため、ここでは詳細な説明を省略する。

【００３６】

ステップＳ１０４：データ登録器１００は、ステップＳ１０３で得られた特徴ベクトル、入力された静止画に与えられたメディアデータ名、顔画像領域の座標の３つの情報を含むレコードをデータテーブル２０に追加する。

【００３７】

ステップＳ１０５：データ登録器１００は、ステップＳ１０１で抽出した顔画像領域をすべて取り出したか否かを判定する。そして、判定の結果がＮｏであればステップＳ１０２に戻って以降の処理を繰り返し、判定の結果がＹｅｓであれば処理を終了する。

【００３８】

図４は、メディアデータが動画である場合のデータ登録器１００による処理手順の一例を示すフローチャートである。データ登録器１００は、メディアデータが動画である場合、例えば以下のステップＳ２０１～ステップＳ２１２の処理を実行して、メディアデータをデータテーブル２０に登録する。

【００３９】

ステップＳ２０１：データ登録器１００は、入力された動画から先頭フレーム画像を取り出す。

【００４０】

ステップＳ２０２：データ登録器１００は、ステップＳ２０１で取り出した先頭フレーム画像に対して顔領域抽出処理および特徴生成処理を行う。この結果、顔が写っている可能性の高い画像領域（顔画像領域）が確信度付ですべて抽出され、それぞれの顔画像領域から特徴ベクトルが得られる。

【００４１】

ステップＳ２０３：データ登録器１００は、入力された動画から次のフレーム画像を取り出す。

【００４２】

ステップＳ２０４：データ登録器１００は、ステップＳ２０３で取り出したフレーム画像に対して、ステップＳ２０２と同様の顔領域抽出処理および特徴生成処理を行う。

【００４３】

ステップＳ２０５：データ登録器１００は、当該フレームについて、前フレームとの間で顔追跡処理を行う。顔追跡処理とは、フレーム間で画像と領域座標が類似する顔画像領域のペアを見つけ出し、これらを同一被写体と解釈する処理である。なお、顔追跡処理は既存技術を用いればよいため、ここでは詳細な説明を省略する。

【００４４】

ステップＳ２０６：データ登録器１００は、ステップＳ２０５の顔追跡処理の結果、同一被写体と判定された顔画像領域を同一グループとして束ねる。

【００４５】
ステップＳ２０７：データ登録器１００は、次のフレームがあるか否かを判定する。そして、判定の結果がＹｅｓであればステップＳ２０３に戻って以降の処理を繰り返し、Ｎｏであればステップ２０８に進む。
【００４６】
ステップＳ２０８：データ登録器１００は、以上の処理により生成されたグループを順に取り出す。
【００４７】
ステップＳ２０９：データ登録器１００は、ステップＳ２０８で取り出したグループに含まれるフレームのうち、最も早く出現したフレームの出現時刻である第１出現時刻と、最も遅く出現したフレームの出現時刻である第２出現時刻とを取得する。
【００４８】
ステップＳ２１０：データ登録器１００は、ステップＳ２０８で取り出したグループに含まれるフレームのうち最も確信度の高いフレームの出現時刻である第３出現時刻と、特徴ベクトルと、顔画像領域の座標とを取得する。
【００４９】
ステップＳ２１１：データ登録器１００は、ステップＳ２０９で取得した第１出現時刻および第２出現時刻と、ステップＳ２１０で取得した第３出現時刻、特徴ベクトル、および顔画像領域の座標と、入力された動画に与えられたメディアデータ名との６つの情報を含むレコードをデータテーブル２０に追加する。
【００５０】
ステップＳ２１２：データ登録器１００は、生成されたグループをすべて取り出したか否かを判定する。そして、判定の結果がＮｏであればステップＳ２０８に戻って以降の処理を繰り返し、判定の結果がＹｅｓであれば処理を終了する。
【００５１】
次に、索引構築器２００の詳細を説明する。索引構築器２００がデータベース４００内に生成する索引情報４０は、特徴ベクトルを列に持つデータテーブル２０に対するベクトル近傍検索を高速化するための補助情報である。索引構築器２００は、入力された索引構築命令文３０のタイプに応じて、ＬＳＨ即値索引情報とＬＳＨ近傍展開索引情報との２種類の索引情報４０を生成することができる。この索引情報４０には、後述するテーブルデータとデータベース索引（Ｂツリーなど）が含まれる。
【００５２】
索引構築器２００は、データテーブル２０の特徴ベクトルを格納する同一の行に対してハッシュのビット長の指定が異なる複数の索引構築命令文３０を実行することにより、複数の索引情報４０を構築することができる。例えば、ビット長が短いハッシュを用いた索引情報４０とビット長が長いハッシュを用いた索引情報４０との２種類を構築しておけば、精度より速度を重視した検索を行う場合は前者の索引情報４０を用い、速度より精度を重視した検索を行う場合は後者の索引情報４０を用いるといったように、検索時に用途に応じた使い分けができるようになる。
【００５３】
図５は、索引構築器２００の構成例を示すブロック図である。索引構築器２００は、図５に示すように、索引構築命令タイプ分類器２１０と、ＬＳＨ即値索引情報生成器２２０と、ＬＳＨ近傍展開索引情報生成器２３０とを備える。
【００５４】
索引構築命令タイプ分類器２１０は、索引構築器２００に入力された索引構築命令文３０を、そのタイプに応じて、ＬＳＨ即値索引情報作成器２２０とＬＳＨ近傍展開索引情報生成器２３０とのいずれかに受け渡すスイッチャである。
【００５５】
索引構築器２００に入力される索引構築命令文３０は、例えば、以下のような構成の命令文である。ただしxxxx部には何らかの値が入る。

Table:xxxx,Column:xxxx,Algo:xxxx,BitLen:xxxx,HammingDist:xxxx

【００５６】

上記構成の索引構築命令文３０は、Table項目で、索引情報４０の対象となるデータテーブル２０のテーブル名が指定され、Column項目で、索引情報４０の対象となるデータテーブル２０内の列名が指定される。この列名が示すデータテーブル２０内の列には、特徴ベクトルが格納されている必要がある。

【００５７】

また、Algo項目では、索引情報４０の生成に用いるＬＳＨアルゴリズムのアルゴリズム名が指定される。アルゴリズム名は、システム内に実装されているＬＳＨアルゴリズムの中から選ばれる。例えば、システム内にSimHashという名前のＬＳＨアルゴリズムとSpectralHashという名前のＬＳＨアルゴリズムのみが実装されているとすれば、これらのいずれかのアルゴリズム名が指定される。本実施形態では、上述したように、ビットワイズＬＳＨを用いるため、ビットワイズＬＳＨのアルゴリズム名が指定される。

【００５８】

Bitlen項目では、Algo項目で選んだＬＳＨアルゴリズムが出力するハッシュのビット長が指定される。HammingDist項目では、ＬＳＨ近傍展開索引情報生成器２３０において扱われるハミング距離の上限（後述の周辺ハッシュの範囲）が指定される。例えばHammingDist項目で“２”が指定されている場合、ＬＳＨ近傍展開索引情報生成器２３０において扱われるハミング距離は“０”と“１”と“２”である。なお、HammingDist項目は省略してもよい。

【００５９】

索引構築命令タイプ分類器２１０は、以上のように構成される索引構築命令文３０が索引構築器２００に入力されると、この索引構築命令文３０を、ＬＳＨ即値索引情報生成器２２０とＬＳＨ近傍展開索引情報生成器２３０のいずれかに渡す。例えば、索引構築命令タイプ分類器２１０は、索引構築器２００に入力された索引構築命令文３０にHammingDist項目がなければ、この索引構築命令文３０をＬＳＨ即値索引情報生成器２２０に渡し、HammingDist項目があれば、この索引構築命令文３０をＬＳＨ近傍展開索引情報生成器２３０に渡す。

【００６０】

ＬＳＨ即値索引情報生成器２２０は、索引構築命令タイプ分類器２１０から索引構築命令文３０を受け取った場合に、この索引構築命令文３０に従って、ＬＳＨ即値テーブルと、このＬＳＨ即値テーブルのHashValue列に対するデータベース索引とを含むＬＳＨ即値索引情報４０Ａを生成する。

【００６１】

図６は、ＬＳＨ即値テーブルの一例を示す図である。ＬＳＨ即値テーブルは、図６に示すように、HashValue列（型：blob）のみからなるテーブルデータである。ＬＳＨ即値索引情報生成器２２０は、索引構築命令文３０で指定されたデータテーブル２０の各レコードに対して、指定された列に格納される特徴ベクトルを順に取り出し、指定されたＬＳＨアルゴリズムを用いて、指定された出力ビット長のハッシュを生成し、得られた各ハッシュのみを持つレコードをＬＳＨ即値テーブルに登録する。

【００６２】

なお、ハッシュは本質的には二値ベクトルであるが、本実施形態ではハッシュをブール値の配列として保存するのではなく、二進数と見立てて整数として保存する。すなわち、ＬＳＨ即値テーブルのHashValue列は、整数または整数配列（計算機アーキテクチャのビット数上限を超えた場合の処置）を格納するblobである。これにより、保存領域サイズを削減することができる。

【００６３】

なお、このＬＳＨ即値テーブルをデータテーブル２０に組み込む構成としてもよい。このような構成とすることにより、検索全体の処理においてサブクエリ処理を１回減らすことができる。ＬＳＨ即値テーブルをデータテーブル２０に組み込むためには、上述のデー

タ登録器１００がデータテーブル２０を生成する際に、このデータテーブル２０にハッシュを格納する列を予め用意しておく必要がある。また、上述したように、データテーブル２０の特徴ベクトルを格納する同一の行に対して複数の索引情報４０を構築できるようにするためには、データテーブル２０内にビット長が異なる複数種類のハッシュを格納する複数の列を確保しておく必要がある。このような構成とした場合、ＬＳＨ即値索引情報生成器２２０は、特徴ベクトルを元に生成したハッシュをＬＳＨ即値テーブルに登録する代わりに、データテーブル２０の該当する列に格納すればよい。

【００６４】

ＬＳＨ即値索引情報生成器２２０は、以上のようなＬＳＨ即値テーブルの生成と合せて、ＬＳＨ即値テーブルのHashValue列に対するデータベース索引を構築し、これらをＬＳＨ即値索引情報４０Ａとしてデータベース４００内に保存する。

【００６５】

図７は、ＬＳＨ即値索引情報生成器２２０による処理手順の一例を示すフローチャートである。ＬＳＨ即値索引情報生成器２２０は、索引構築命令タイプ分類器２１０から索引構築命令文３０を受け取ると、例えば以下のステップＳ３０１～ステップＳ３０７の処理を実行し、ＬＳＨ即値テーブルとデータベース索引とを含むＬＳＨ即値索引情報４０Ａをデータベース４００に保存する。

【００６６】

ステップＳ３０１：ＬＳＨ即値索引情報生成器２２０は、データベース４００内に、名前：HashValue・型：blobの列のみを持つテーブル（空のＬＳＨ即値テーブル）を生成する。なお、テーブル名の形式は「AAAA_BBBB_CCCC_DDDD」とする。ただし、AAAAは処理対象のデータテーブル２０のテーブル名、BBBBは処理対象の列の列名、CCCCはＬＳＨアルゴリズムのアルゴリズム名、DDDDはビット長とする。例えば、aTable_feat_SimHash_64といったテーブル名が生成したテーブルに与えられる。

【００６７】

ステップＳ３０２：ＬＳＨ即値索引情報生成器２２０は、索引構築命令文３０で指定されたデータテーブル２０から、索引構築命令文３０で指定された列の特徴ベクトルを順に取り出す。

【００６８】

ステップＳ３０３：ＬＳＨ即値索引情報生成器２２０は、ステップＳ３０２で取り出した特徴ベクトルに対して、索引構築命令文３０で指定されたＬＳＨアルゴリズムを用いて、索引構築命令文３０で指定された出力ビット長のハッシュを生成する。

【００６９】

ステップＳ３０４：ＬＳＨ即値索引情報生成器２２０は、HashValue列の値としてステップＳ３０３で得られた事例ハッシュを持つレコードを、ＬＳＨ即値テーブルに追加する。

【００７０】

ステップＳ３０５：ＬＳＨ即値索引情報生成器２２０は、処理対象のデータテーブル２０から特徴ベクトルをすべて取り出したか否かを判定する。そして、判定の結果がＮｏであればステップＳ３０２に戻って以降の処理を繰り返し、ＹｅｓであればステップＳ３０６に進む。

【００７１】

ステップＳ３０６：ＬＳＨ即値索引情報生成器２２０は、以上の処理を経て生成されたＬＳＨ即値テーブルのHashValue列に対するデータベース索引（Ｂツリーなど）を構築し、データベース４００に保存する。

【００７２】

ステップＳ３０７：ＬＳＨ即値索引情報生成器２２０は、ハッシュの生成に用いたＬＳＨアルゴリズムのモデルパラメタセットなど（例えばSimHashにおける射影ベクトル群など）があれば、テーブル名と同じファイル名（例えばaTable_feat_SimHash_64）でデータベース４００に保存する。

【００７３】
ＬＳＨ近傍展開索引情報生成器２３０は、索引構築命令タイプ分類器２１０から索引構築命令文３０を受け取った場合に、この索引構築命令文３０に従って、ＬＳＨ近傍展開テーブルと、このＬＳＨ近傍展開テーブルのHashValue列およびHammingDistance列に対するデータベース複合索引（コンポジットインデックス）とを含むＬＳＨ近傍展開索引情報４０Ｂを生成する。
【００７４】
図８は、ＬＳＨ近傍展開テーブルの一例を示す図である。ＬＳＨ近傍展開テーブルは、図８に示すように、HashValue列（型：blob）、HammingDistance列（型：integer）およびDataRowID列（型：integer）からなるテーブルデータである。ＬＳＨ近傍展開索引情報生成器２３０は、索引構築命令文３０で指定されたデータテーブル２０から特徴ベクトルを取り出し、各特徴ベクトル（事例ベクトル）について、索引構築命令文３０で指定されたＬＳＨアルゴリズムおよびビット長に従ってハッシュ（縮約事例ベクトル）を生成する。これを「事例ハッシュ」と呼ぶ。次に、ＬＳＨ近傍展開索引情報生成器２３０は、生成した各事例ハッシュについて、索引構築命令文３０で指定されたハミング距離以内のハッシュ（縮約周辺ベクトル）群を生成する。ここで生成されるハッシュを「周辺ハッシュ」と呼ぶ。そして、ＬＳＨ近傍展開索引情報生成器２３０は、例えば、周辺ハッシュと、周辺ハッシュの事例ハッシュからのハミング距離と、事例ハッシュの元になる事例ベクトルが属するデータテーブル２０のレコードの行ＩＤとの３つ組からなるレコードを、ＬＳＨ近傍展開テーブルに登録する。
【００７５】
なお、本実施形態では、後述のデータベース索引ＬＳＨ検索処理において、検索条件として用いるハミング距離を０から順次インクリメントして検索結果を追加する処理を行う。このため、ＬＳＨ近傍展開テーブルの第２列にHammingDistance列を設け、周辺ハッシュの事例ハッシュからのハミング距離をこのHammingDistance列に格納している。しかし、検索時にＬＳＨ近傍展開テーブルの距離上限を検索条件として用いて検索を行う構成とすることも可能である。この場合、ＬＳＨ近傍展開テーブルにHammingDistance列を設ける必要はない。このような構成とした場合は、後述のデータベース索引ＬＳＨ検索処理において、出力する検索結果データセットの件数をハミング距離を利用して制御することはできなくなるが、ＬＳＨ近傍展開テーブルのサイズを小さくできる効果がある。また、このような構成とした場合は、ＬＳＨ近傍展開テーブルのHashValue列およびHammingDistance列に対するデータベース複合索引を構築する代わりに、ＬＳＨ近傍展開テーブルのHashValue列に対するデータベース索引を構築すればよい。
【００７６】
また、図８に示したＬＳＨ近傍展開テーブルには、HashValue列に格納される周辺ハッシュの値として重複した値が多く含まれているため、正規化を行うことでコンパクト化することができる。
【００７７】
図９は、図８に示したＬＳＨ近傍展開テーブルを２つに分割することで正規化した例を示す図である。ＬＳＨ近傍展開テーブルを図９のような構成とした場合、後述のデータベース索引ＬＳＨ検索処理において、HashValueとHammingDistanceに関する検索条件を用いて図９（ａ）のテーブルの行ＩＤを確定し、当該行ＩＤを元に生成した図９（ｂ）のテーブルのTable1_RowIDに関する検索条件を用いてDataRowIDを見つければよい。なお、図９（ｂ）のテーブルの第１列であるTable1_RowIDが図９（ａ）のテーブルの対応する行の行ＩＤを格納しており、このリンクを用いて図９（ａ）のテーブルと図９（ｂ）のテーブルとを結合（JOIN）すれば、図８に示したＬＳＨ近傍展開テーブルになる。
【００７８】
また、図９（ａ）のテーブルと図９（ｂ）のテーブルのそれぞれを、連想配列を用いて実装してもよい。連想配列として実装する場合、図９（ａ）の連想配列のキーはHashValueとHammingDistanceを組み合わせたものであり、値は１つ以上の行ＩＤである。また、図

９（ｂ）の連想配列のキーはTable1_RowIDであり、値は１つ以上のDataRowIDである。
【００７９】
図１０は、ＬＳＨ近傍展開テーブルのHashValue列およびHammingDistance列に対するデータベース複合索引の一例を示す図である。このデータベース複合索引は、図１０に示すように、HashValueおよびHammingDistanceの値に沿って分岐する木構造となっており、葉に該当するＬＳＨ近傍展開テーブルの行ＩＤが記載されている。このデータベース複合索引を用いることで、検索時間を０（Ｎ）から０（ｌｏｇ（Ｎ））に短縮させることができる。
【００８０】
なお、データベース複合索引の構成を少し変更することで、ＬＳＨ近傍展開テーブルを不要とすることもできる。具体的には、データベース複合索引の葉に、ＬＳＨ近傍展開テーブルの行ＩＤを格納する代わりにDataRowIDの値を格納する構成とする。これにより、DataRowIDしか検索できなくなるが、ＬＳＨ近傍展開テーブルをデータベース４００に保存しておく必要がなくなる。この場合、ＬＳＨ近傍展開索引情報４０Ｂはデータベース複合索引のみとなる。
【００８１】
図１１は、ＬＳＨ近傍展開索引情報生成器２３０による処理手順の一例を示すフローチャートである。ＬＳＨ近傍展開索引情報生成器２３０は、索引構築命令タイプ分類器２１０から索引構築命令文３０を受け取ると、例えば以下のステップＳ４０１～ステップＳ４１０の処理を実行し、ＬＳＨ近傍展開テーブルとデータベース複合索引とを含むＬＳＨ近傍展開索引情報４０Ｂをデータベース４００に保存する。
【００８２】
ステップＳ４０１：ＬＳＨ近傍展開索引情報生成器２３０は、データベース４００内に、名前：HashValue・型：blobの列と、名前：HammingDistance・型：integerの列と、名前：DataRowID・型：integerの列とを持つテーブル（空のＬＳＨ近傍展開テーブル）を生成する。なお、テーブル名の形式は「AAAA_BBBB_CCCC_DDDD」とする。ただし、AAAAは処理対象のデータテーブル２０のテーブル名、BBBBは処理対象の列の列名、CCCCはＬＳＨアルゴリズムのアルゴリズム名、DDDDはビット長とする。例えば、aTable_feat_PQ_3といったテーブル名が生成したテーブルに与えられる。
【００８３】
ステップＳ４０２：ＬＳＨ近傍展開索引情報生成器２３０は、索引構築命令文３０で指定されたデータテーブル２０から、索引構築命令文３０で指定された列の特徴ベクトルを順に取り出す。
【００８４】
ステップＳ４０３：ＬＳＨ近傍展開索引情報生成器２３０は、ステップＳ４０２で取り出した特徴ベクトル（事例ベクトル）に対して、索引構築命令文３０で指定されたＬＳＨアルゴリズムを用いて、索引構築命令文３０で指定されたビット長の事例ハッシュを生成する。
【００８５】
ステップＳ４０４：ＬＳＨ近傍展開索引情報生成器２３０は、ステップＳ４０３で得られた事例ハッシュからの距離が指定距離以下の周辺ハッシュをすべて生成する。
【００８６】
ステップＳ４０５：ＬＳＨ近傍展開索引情報生成器２３０は、ステップＳ４０４で得られた周辺ハッシュを順に取り出す。
【００８７】
ステップＳ４０６：ＬＳＨ近傍展開索引情報生成器２３０は、ステップＳ４０５で取り出した周辺ハッシュ、この周辺ハッシュの事例ハッシュからのハミング距離、事例ハッシュの元になる事例ベクトルが属するデータテーブル２０内のレコードの行ＩＤの３つ組からなるレコードを、ＬＳＨ近傍展開テーブルに追加する。
【００８８】

ステップＳ４０７：ＬＳＨ近傍展開索引情報生成器２３０は、ステップＳ４０４で得られた周辺ハッシュをすべて取り出したか否かを判定する。そして、判定の結果がＮｏであればステップＳ４０５に戻って以降の処理を繰り返し、ＹｅｓであればステップＳ４０８に進む。

【００８９】

ステップＳ４０８：ＬＳＨ近傍展開索引情報生成器２３０は、処理対象のデータテーブル２０から特徴ベクトルをすべて取り出したか否かを判定する。そして、判定の結果がＮｏであればステップＳ４０２に戻って以降の処理を繰り返し、ＹｅｓであればステップＳ４０９に進む。

【００９０】

ステップＳ４０９：ＬＳＨ近傍展開索引情報生成器２３０は、以上の処理を経て生成されたＬＳＨ近傍展開テーブルのHashValue列およびHammingDistance列に対するデータベース複合索引（コンポジットインデックス）を構築し、データベース４００に保存する。

【００９１】

ステップＳ４１０：ＬＳＨ近傍展開索引情報生成器２３０は、事例ハッシュの生成に用いたＬＳＨアルゴリズムのモデルパラメタセットなどがあれば、テーブル名と同じファイル名（例えばaTable_feat_PQ_3）でデータベース４００に保存する。

【００９２】

ここで、上記ステップＳ４０４における周辺ハッシュの生成方法の具体例について説明する。以下では、周辺ハッシュの起点となる事例ハッシュのビット列が“０１０”であるものとし、ハミング距離２以内の周辺ハッシュを列挙するという設定例で説明する。

【００９３】

まず、ハミング距離０の周辺ハッシュは、事例ハッシュと同じハッシュである。すなわち、事例ハッシュのビット列が“０１０”であれば、この事例ハッシュに対してハミング距離０の周辺ハッシュは“０１０”のみである。

【００９４】

次に、ハミング距離１の周辺ハッシュは、事例ハッシュの任意のビットを１つだけ反転させたものである。事例ハッシュのビット長が３であれば、反転させるビットの選択方法は、第１ビットのみ、第２ビットのみ、第３ビットのみの３種類である。事例ハッシュのビット列が“０１０”であれば、各選択方法でビット反転を行った結果は“１１０”、“０００”および“０１１”である。これら３つのハッシュが、事例ハッシュ“０１０”に対してハミング距離が１の周辺ハッシュである。

【００９５】

次に、ハミング距離２の周辺ハッシュは、事例ハッシュの任意のビットを２つ反転させたものである。事例ハッシュのビット長が３であれば、反転させるビットの選択方法は、第１ビット＋第２ビット、第１ビット＋第３ビット、第２ビット＋第３ビットの３種類である。事例ハッシュのビット列が“０１０”であれば、各選択方法でビット反転を行った結果は“１００”、“１１１”および“００１”である。これら３つのハッシュが、事例ハッシュ“０１０”に対してハミング距離が２の周辺ハッシュである。

【００９６】

以上をまとめると、事例ハッシュ“０１０”に対するハミング距離０の周辺ハッシュは“０１０”、ハミング距離１の周辺ハッシュは“１１０”、“０００”および“０１１”、ハミング距離２の周辺ハッシュは“１００”、“１１１”および“００１”である。このように、低コストな処理で周辺ハッシュをすべて生成することができる。

【００９７】

なお、以上の方法により事例ハッシュに対する周辺ハッシュを生成する際、ビットの重要度などを元に、反転を許可するビットを制限する仕組みを備えてもよい。例えば、上記参考文献３のように主成分分析をベースとしたハッシュ生成法の場合、各ビットに対応する固有値から当該ビットの弁別性（重要度）を割り出すことができる。事例ハッシュにおける弁別性の高いビットを反転して生成した周辺ハッシュは、弁別性が失われた周辺ハッ

シュとなるため、このような周辺ハッシュをＬＳＨ近傍展開テーブルに追加してもヒットする確率が低いと考えられる。したがって、このような弁別性の高いビットは反転させないように制限を加えるようにしてもよい。このように、反転を許可するビットを制限することにより、ＬＳＨ近傍展開テーブルの記憶領域サイズを小さくすることができる。

【００９８】

ＬＳＨ近傍展開テーブルは、検索キーとなる列と検索結果となる列が決まっているため、連想配列（キーバリューストア、C++標準ライブラリであればstd::mapやstd::unordered_mapなど）で簡潔に実装することができる。

【００９９】

すなわち、検索結果はDataRowIDであるため、連想配列の値としてDataRowIDの可変長の連続メモリ配置型配列（C++標準ライブラリであればstd::vectorなど）を用いる。検索条件はHashValueとHammingDistanceのANDであるため、これら２つの値を用いてユニークになる連想配列のキーを生成する。図８に例示したＬＳＨ近傍展開テーブルの場合、HashValue列は二進数表記で“０００”～“１１１”の値を取り、HammingDistanceは“０”～“１”の値を取るので、キー値を以下の算出式で決定するなどが考えられる。

キー値＝HashValue列の値×２＋HammingDistance列の値

【０１００】

図１２は、連想配列と連続メモリ配置型配列を説明する図である。連想配列は、キーから値へ対応付ける方法として、図１２（ａ）に示すように、データベース索引のように木やハッシュなどのデータ構造を内部に持つ。上記のキーが定義域においてほぼ充当されているならば、連想配列の代わりに、図１２（ｂ）に示すような連続メモリ配置型配列（C++標準ライブラリであればstd::vectorなど）を用いてもよい。連続メモリ配置型配列として確保する要素数は、HashValueとHammingDistanceの全組合せ数とする。連続メモリ配置型配列を用いたデータ構造は、連想配列を用いたデータ構造と比べ、キーの充当率が高い場合に限りメモリ使用量が低減され、さらに処理速度が高速化される。

【０１０１】

次に、検索器３００の詳細を説明する。この検索器３００の機能概要および特徴は以下の通りである。この検索器３００では、拡張ＳＱＬ５０を用いて、ベクトル類似性に関する０個以上の条件を含む論理条件でのレコード検索ができる。これにより、データベース利用者は、一般的なデータ照合と特徴ベクトル類似性判定とを条件として結び付けた論理式での検索が行えるようになる。

【０１０２】

また、ベクトル類似性判定（ベクトル近傍検索）の処理は、検索対象のレコード数に応じて、５種類の方式を内部的に切り替えることができる。この切り替えは事前に設定されたシステム設定に基づいて行われる。５種類の方式には速度、精度、件数制御の有無について長短がある。これら５種類の方式の１つであるデータベース索引ＬＳＨ検索は、クエリに対して、検索時にベクトル距離計算を行わずデータベース索引のみを用いてベクトル類似性判定（ベクトル近傍検索）を実現する方式である。この方式は、検索時間を大幅に削減する効果を持つ。

【０１０３】

図１３は、検索器３００の構成例を示すブロック図である。検索器３００は、拡張ＳＱＬ５０を入力として受け取り、拡張ＳＱＬ５０に従った検索処理を実行して検索結果データセット６０を出力するものであり、図１３に示すように、検索条件処理部３１０と、検索出力部３２０とを備える。検索条件処理部３１０は、内部に様々な検索条件に対する処理を行う機能モジュールを持つ。このうち、特に本実施形態に特徴的なベクトル類似性に関する検索条件に対する処理を行う機能モジュールがベクトル類似性判定処理部３３０である。また、それ以外の検索条件に対する処理を行う機能モジュールを、他の検索条件処理部３４０と総称する。検索条件処理部３１０は、これらベクトル類似性判定処理部３３０の検索結果と他の検索条件処理部３４０との集合演算を行い、検索条件全体の処理を完成させる。検索出力部３２０は、検索条件処理部３１０の処理の結果に基づいてデータベ

ース４００からレコードを取り出し、検索結果データセット６０として出力する。
【０１０４】
なお、検索器３００による検索処理の大半は従来のＳＱＬの検索処理と同様であり、検索条件処理部３１０がＳＱＬにおけるfrom命令部（データソースの特定処理）およびwhere命令部（検索条件処理）にあたり、検索出力部３２０がＳＱＬにおけるselect命令部にあたる。本実施形態における固有の処理は、検索条件処理部３１０の中の１つであるベクトル類似性判定処理部３３０が実行する処理である。このため、以下ではベクトル類似性判定処理部３３０を中心に説明し、従来技術をそのまま適用できる部分については適宜説明を省略する。
【０１０５】
まず、検索器３００が入力として受け取る拡張ＳＱＬ５０について説明する。拡張ＳＱＬ５０は、従来のＳＱＬに、ベクトル類似性に関する条件にあたるvnn関数を記述できるよう拡張した問合せ言語である。vnn関数は、以下の例のように、where命令内で一般的なＤＢ基本型データへの条件群と組合せ可能な項として用いる。
select * from aTable where vnn(feat_simhash_64,(10,20,30),10) and annualIncome > 10000000
【０１０６】
vnn関数の仕様は以下の通りである。
入力：vnn関数が検索対象とするデータセットを入力とする。この入力は、検索条件処理部３１０が内部的に生成してvnn関数を呼び出す際に渡す。通常の構文であれば、from命令で指定されたテーブルの全レコード群である。
第１引数：第１引数は、事例ベクトル（検索対象とする特徴ベクトル）を格納する列名と、その列に適用されているＬＳＨアルゴリズムのアルゴリズム名と、その列に適用されているＬＳＨアルゴリズムの出力ビット長（後述の第３実施形態ではより一般化してオプションパラメタとしている）とをアンダースコアで結んだテキストである。
第２引数：第２引数は、クエリベクトルである。
第３引数：第３引数は、出力上位件数（出力対象とするベクトルの上位件数）である。ただし、クエリベクトルとの距離の小さいベクトルを上位とみなすものとする。
出力：検索条件処理部３１０（from, whereなどの処理）に対して、引数で示された条件に合致する検索結果を返す。
なお、ここでは出力条件にあたる第３引数として出力上位件数を与えるものとしているが、これに代えて距離上限を第３引数として与えるようにしてもよい。
【０１０７】
図１４は、ベクトル類似性判定部３３０の入出力関係を示す図である。ベクトル類似性判定部３３０は、図１４に示すように、検索対象データセット７０を入力として受け取り、vnn関数で表現された検索条件に基づくベクトル類似性判定（ベクトル近傍検索）処理を行って、検索結果８０を返す。本実施形態では、ベクトル類似性判定部３３０による検索方式として、厳密検索（Strict Search）と、線形ＬＳＨ検索（Linear LSH Search）と、データベース索引ＬＳＨ検索（DB-Indexed LSH Search）と、データベース索引ＬＳＨ検索＋厳密検索と、データベース索引ＬＳＨ検索＋線形ＬＳＨ検索との５種類が用意されている。
【０１０８】
厳密検索は、元のクエリベクトルをそのまま用いて、クエリベクトルに類似する事例ベクトルを線形検索する検索方式である。線形ＬＳＨ検索は、ＬＳＨ即値索引情報４０Ａを用いて、クエリハッシュに完全一致する事例ハッシュに対応する事例ベクトルを線形検索する検索方式である。データベース索引ＬＳＨ検索は、ＬＳＨ近傍展開索引情報４０Ｂを用いて、クエリハッシュに完全一致する周辺ハッシュに対応する事例ベクトルを検索する検索方式である。データベース索引ＬＳＨ検索＋厳密検索は、データベース索引ＬＳＨ検索により件数を絞り込んだ後に厳密検索を行う２段階絞込方式の検索方式である。データベース索引ＬＳＨ検索＋線形ＬＳＨ検索は、データベース索引ＬＳＨ検索により件数を絞

り込んだ後に線形ＬＳＨ検索を行う２段階絞込方式の検索方式である。これら５種類の方式による検索は、それぞれ、図１３に示した厳密検索器３３１、線形ＬＳＨ検索器３３２、データベース索引ＬＳＨ検索器３３３、データベース索引ＬＳＨ検索＋厳密検索器３３４およびデータベース索引ＬＳＨ検索＋線形ＬＳＨ検索器３３５により行われる。

【０１０９】

また、本実施形態では、ベクトル類似性判定処理部３３０が、事前に与えられた検索方式設定情報９０（図１３参照）に基づいて、上記の５種類の検索方式のいずれかを検索対象のレコード数に応じて選択するものとする。検索方式設定情報９０は、以下のような複数行からなる文法となっている。件数条件が上から照合され、マッチした検索方式が採用される。

件数条件：検索方式
件数条件：検索方式
・・・
otherwise：検索方式

【０１１０】

件数条件には、“<=NNNN”（ただしNNNNは任意の正数）または“otherwise”を記述することができる。“<=NNNN”は検索対象となる事例ベクトルの数がNNNN件以下の場合にマッチする。“otherwise”はあらゆる件数にマッチする。“otherwise”は検索方式設定情報９０内に必ず１行含まれている必要がある。

【０１１１】

検索方式には、“Strict”（厳密検索）、“LinearLSH”（線形ＬＳＨ検索）、“DBIndexedLSH”（データベース索引ＬＳＨ検索）、“DBIndexedLSH:NNNN/Strict”（データベース索引ＬＳＨ検索＋厳密検索）および“DBIndexedLSH:NNNN/LinearLSH”（データベース索引ＬＳＨ検索＋線形ＬＳＨ検索）の５種類を記述することができる。NNNNには任意の正数を記述することができる。

【０１１２】

検索方式設定情報９０の記述例を以下に示す。以下に例示する検索方式設定情報９０は、検索対象となる事例ベクトルの数が１００００件以下なら厳密検索方式が採用され、１０００００件以下なら線形ＬＳＨ検索が採用され、それより多い件数ならデータベース索引ＬＳＨ検索＋線形ＬＳＨ検索方式が採用されることを示している。

<=10000：Strict
<=100000：LinearLSH
Otherwise：DBIndexedLSH:100000/LinearLSH

【０１１３】

図１５は、ベクトル類似性判定部３３０による処理手順の一例を示すフローチャートである。ベクトル類似性判定部３３０は、検索対象データセット７０を入力として受け取ると、例えば以下のステップＳ５０１～ステップＳ５０７の処理を実行して、検索結果８０を出力する。

【０１１４】

ステップＳ５０１：ベクトル類似性判定部３３０は、入力された検索対象データセット７０のレコード件数（検索対象となる事例ベクトルの数）をカウントする。

【０１１５】

ステップＳ５０２：ベクトル類似性判定部３３０は、検索方式設定情報９０を参照し、件数条件がステップＳ５０１でカウントしたレコード件数とマッチする検索方式を取得する。取得した検索方式がStrictであればステップＳ５０３に進み、LinearLSHであればステップＳ５０４に進み、DBIndexedLSHであればステップＳ５０５に進み、DBIndexedLSH:NNNN/Strict（ただしNNNNは任意の正数）であればステップＳ５０６に進み、DBIndexedLSH:NNNN/LinearLSH（ただしNNNNは任意の正数）であればステップＳ５０７に進む。

【０１１６】

ステップＳ５０３：ベクトル類似性判定部３３０は、厳密検索器３３１を用いて検索結

果８０を取得し、取得した検索結果８０を出力する。
【０１１７】
ステップＳ５０４：ベクトル類似性判定部３３０は、線形ＬＳＨ検索器３３２を用いて検索結果８０を取得し、取得した検索結果８０を出力する。
【０１１８】
ステップＳ５０５：ベクトル類似性判定部３３０は、データベース索引ＬＳＨ検索器３３３を用いて検索結果８０を取得し、取得した検索結果８０を出力する。
【０１１９】
ステップＳ５０６：ベクトル類似性判定部３３０は、データベース索引ＬＳＨ検索＋厳密検索器３３４を用いて検索結果８０を取得し、取得した検索結果８０を出力する。
【０１２０】
ステップＳ５０７：ベクトル類似性判定部３３０は、データベース索引ＬＳＨ検索＋線形ＬＳＨ検索器３３５を用いて検索結果８０を取得し、取得した検索結果８０を出力する。
【０１２１】
なお、以上の例では検索対象となる事例ベクトルの数に応じて検索方式を切り替えるようにしているが、切り替えるべき検索方式の指定として、ビット長など様々なパラメタ設定も含ませるといった拡張を行うようにしてもよい。例えば１６ビットなど短いビット長（すなわち１件当たりの処理時間は小さいが低精度）の線形ＬＨＳ検索で事例ベクトルの数を１００００００件に絞り込んだ後、４０９６ビットなど長いビット長（すなわち１件当たりの処理時間は大きいが高精度）の線形ＬＨＳ検索で順位付けするなどのプランを作ることができる。その場合の検索方式設定情報９０の記述例を以下に示す（@以降にパラメタ設定が列挙される）。
<=100000：LinearLSH@4096
otherwise：LinearLSH@16:100000/LinearLSH@4096
【０１２２】
以下では、それぞれの検索方式による検索処理の具体例を説明する。
【０１２３】
図１６は、厳密検索器３３１による処理手順の一例を示すフローチャートである。厳密検索器３３１は、例えば以下のステップＳ６０１～ステップＳ６０８の処理を実行して、検索結果８０を出力する。
【０１２４】
ステップＳ６０１：厳密検索器３３１は、空の検索結果リストを生成する。
【０１２５】
ステップＳ６０２：厳密検索器３３１は、vnn関数の第２引数として指定されたクエリベクトルを得る。
【０１２６】
ステップＳ６０３：厳密検索器３３１は、検索対象データセット７０からレコードを順に取り出す。
【０１２７】
ステップＳ６０４：厳密検索器３３１は、ステップＳ６０３で取り出したレコードにおいてvnn関数の第１引数として指定された事例ベクトルを格納する列に格納されている事例ベクトル群とクエリベクトルとのユークリッド距離を算出する。
【０１２８】
ステップＳ６０５：厳密検索器３３１は、ステップＳ６０３で取り出したレコードの行ＩＤとステップＳ６０４で算出したユークリッド距離との組を検索結果リストに追加する。ただし、検索結果リストの要素がユークリッド距離昇順で並ぶような位置に挿入する。
【０１２９】
ステップＳ６０６：厳密検索器３３１は、検索結果リストに含まれる要素数がvnn関数の第３引数として指定された出力上位件数よりも多い場合、出力上位件数以降の要素群を

破棄する。
【０１３０】
ステップＳ６０７：厳密検索器３３１は、検索対象データセット７０からレコードをすべて取り出したか否かを判定する。そして、判定の結果がＮｏであればステップＳ６０３に戻って以降の処理を繰り返し、Ｙｅｓであればステップ６０８に進む。
【０１３１】
ステップＳ６０８：厳密検索器３３１は、検索結果リストに含まれる行ＩＤの集合を検索結果８０として出力し、処理を終了する。
【０１３２】
図１７は、線形ＬＳＨ検索器３３２による処理手順の一例を示すフローチャートである。線形ＬＳＨ検索器３３２は、例えば以下のステップＳ７０１～ステップＳ７０９の処理を実行して、検索結果８０を出力する。
【０１３３】
ステップＳ７０１：線形ＬＳＨ検索器３３２は、空の検索結果リストを生成する。
【０１３４】
ステップＳ７０２：線形ＬＳＨ検索器３３２は、vnn関数の引数として指定されたクエリベクトル、ＬＳＨアルゴリズムのアルゴリズム名および出力ビット長を取得して、クエリベクトルのハッシュ（縮約クエリベクトル）を生成する。以下、これを「クエリハッシュ」と呼ぶ。
【０１３５】
ステップＳ７０３：線形ＬＳＨ検索器３３２は、検索対象データセット７０のテーブル名、vnn関数の引数として指定された事例ベクトルを格納する列、ＬＳＨアルゴリズムのアルゴリズム名および出力ビット長を取得して、これらを元に決定されるテーブル名を持つＬＳＨ即値テーブルを取得する。
【０１３６】
ステップＳ７０４：線形ＬＳＨ検索器３３２は、ステップＳ７０３で取得したＬＳＨ即値テーブルを用いて、検索対象データセット７０の各レコードに対応する事例ハッシュを順に取り出す。
【０１３７】
ステップＳ７０５：線形ＬＳＨ検索器３３２は、ステップＳ７０４で取り出した事例ハッシュとステップＳ７０２で生成したクエリハッシュとのハミング距離を算出する。
【０１３８】
ステップＳ７０６：線形ＬＳＨ検索器３３２は、ステップＳ７０４で取り出した事例ハッシュに対応する検索対象データセット７０のレコードの行ＩＤとステップＳ７０５で算出したハミング距離との組を検索結果リストに追加する。ただし、検索結果リストの要素がハミング距離昇順で並ぶような位置に挿入する。
【０１３９】
ステップＳ７０７：線形ＬＳＨ検索器３３２は、検索結果リストに含まれる要素数がvnn関数の第３引数として指定された出力上位件数よりも多い場合、出力上位件数以降の要素群を破棄する。
【０１４０】
ステップＳ７０８：線形ＬＳＨ検索器３３２は、検索対象データセット７０の各レコードに対応する事例ハッシュをすべて取り出したか否かを判定する。そして、判定の結果がＮｏであればステップＳ７０４に戻って以降の処理を繰り返し、ＹｅｓであればステップＳ７０９に進む。
【０１４１】
ステップＳ７０９：線形ＬＳＨ検索器３３２は、検索結果リストに含まれる行ＩＤの集合を検索結果８０として出力し、処理を終了する。
【０１４２】
なお、上記のステップＳ７０２において生成するクエリハッシュは本質的には二値ベク

トルであるが、ＬＳＨ即値テーブルでの保存形式に合わせて、二進数と見立て整数として扱う。これにより、記憶領域を節約できるほか、整数同士の比較となるため照合処理が高速になる。

【０１４３】

図１８は、データベース索引ＬＳＨ検索器３３３による処理手順の一例を示すフローチャートである。データベース索引ＬＳＨ検索器３３３は、例えば以下のステップＳ８０１～ステップＳ８１０の処理を実行して、検索結果８０を出力する。

【０１４４】

ステップＳ８０１：データベース索引ＬＳＨ検索器３３３は、空の検索結果セットを生成する。

【０１４５】

ステップＳ８０２：データベース索引ＬＳＨ検索器３３３は、vnn関数の引数として指定されたクエリベクトル、ＬＳＨアルゴリズムのアルゴリズム名および出力ビット長を取得して、クエリハッシュを生成する。

【０１４６】

ステップＳ８０３：データベース索引ＬＳＨ検索器３３３は、検索対象データセット７０のテーブル名、vnn関数の引数として指定された事例ベクトルを格納する列、ＬＳＨアルゴリズムのアルゴリズム名および出力ビット長を取得して、これらを元に決定されるテーブル名を持つＬＳＨ近傍展開テーブルを取得する。

【０１４７】

ステップＳ８０４：データベース索引ＬＳＨ検索器３３３は、ハミング距離条件変数に０を割り当てる。

【０１４８】

ステップＳ８０５：データベース索引ＬＳＨ検索器３３３は、ステップＳ８０３で取得したＬＳＨ近傍展開テーブルに対して、HammingDistance列の値がハミング距離条件変数の現在値と一致し、かつ、HashValue列の値がステップＳ８０２で生成したクエリハッシュと一致するレコード群を検索し、得られた行ＩＤのセットを検索結果セットに追加する。

【０１４９】

ステップＳ８０６：データベース索引ＬＳＨ検索器３３３は、検索結果セットから重複要素を除去する。

【０１５０】

ステップＳ８０７：データベース索引ＬＳＨ検索器３３３は、検索結果セットの要素数がvnn関数の第３引数として指定された出力上位件数以上となっているか否かを判定する。そして、判定の結果がＮｏであればステップＳ８０８に進み、ＹｅｓであればステップＳ８１０に進む。

【０１５１】

ステップＳ８０８：データベース索引ＬＳＨ検索器３３３は、ハミング距離条件変数の値を１追加する。

【０１５２】

ステップＳ８０９：データベース索引ＬＳＨ検索器３３３は、ハミング距離条件変数の値がＬＳＨ近傍展開テーブルの距離上限以下か否かを判定する。そして、判定の結果がＹｅｓであればステップＳ８０５に戻って以降の処理を繰り返し、ＮｏであればステップＳ８１０に進む。

【０１５３】

ステップＳ８１０：データベース索引ＬＳＨ検索器３３３は、検索結果セットを検索結果８０として出力し、処理を終了する。

【０１５４】

なお、上記のステップＳ８０２において生成するクエリハッシュは本質的には二値ベクトルであるが、ＬＳＨ近傍展開テーブルでの保存形式に合わせて、二進数と見立て整数と

して扱う。これにより、記憶領域を節約できるほか、整数同士の比較となるため照合処理が高速になる。また、ＬＳＨ近傍展開テーブルの距離上限が１００などの大きな値の場合は、上記のステップＳ８０８での増分値を１０など大きめの値とし、ステップＳ８０５でのハミング距離条件に範囲を設けるようにしてもよい。こうすれば反復回数が減るため、検索処理全体を高速化できる。

【０１５５】

図１９は、データベース索引ＬＳＨ検索＋厳密検索器３３４による処理手順の一例を示すフローチャートである。データベース索引ＬＳＨ検索＋厳密検索は、データベース索引ＬＳＨ検索により検索を行った後、その結果について厳密検索による検索を行う２段階絞込方式である。データベース索引ＬＳＨ検索＋厳密検索器３３４は、例えば以下のステップＳ９０１～ステップＳ９０４の処理を実行して、検索結果８０を出力する。

【０１５６】

ステップＳ９０１：データベース索引ＬＳＨ検索＋厳密検索器３３４は、検索方式設定情報９０に記載された検索方式名「DBIndexedLSH:NNNN/Strict」のNNNN部を読み取る。これが、データベース索引ＬＨＳ検索における出力上位件数目標である。

【０１５７】

ステップＳ９０２：データベース索引ＬＳＨ検索＋厳密検索器３３４は、出力上位件数目標の設定でデータベース索引ＬＨＳ検索器３３３を呼び出し、検索結果セットを得る。

【０１５８】

ステップＳ９０３：データベース索引ＬＳＨ検索＋厳密検索器３３４は、データベース索引ＬＨＳ検索器３３３から取得した検索結果セットを検索対象データセット７０として厳密検索器３３１を呼び出し、検索結果リストを得る。

【０１５９】

ステップＳ９０４：データベース索引ＬＳＨ検索＋厳密検索器３３４は、厳密検索器３３１から取得した検索結果リストに含まれる行ＩＤの集合を検索結果８０として出力し、処理を終了する。

【０１６０】

図２０は、データベース索引ＬＳＨ検索＋線形ＬＳＨ検索器３３５による処理手順の一例を示すフローチャートである。データベース索引ＬＳＨ検索＋線形ＬＳＨ検索は、データベース索引ＬＳＨ検索により検索を行った後、その結果について線形ＬＳＨ検索による検索を行う２段階絞込方式である。データベース索引ＬＳＨ検索＋線形ＬＳＨ検索器３３５は、例えば以下のステップＳ１００１～ステップＳ１００４の処理を実行して、検索結果８０を出力する。

【０１６１】

ステップＳ１００１：データベース索引ＬＳＨ検索＋線形ＬＳＨ検索器３３５は、検索方式設定情報９０に記載された検索方式名「DBIndexedLSH:NNNN/Strict」のNNNN部を読み取る。これが、データベース索引ＬＨＳ検索における出力上位件数目標である。

【０１６２】

ステップＳ１００２：データベース索引ＬＳＨ検索＋線形ＬＳＨ検索器３３５は、出力上位件数目標の設定でデータベース索引ＬＨＳ検索器３３３を呼び出し、検索結果セットを得る。

【０１６３】

ステップＳ１００３：データベース索引ＬＳＨ検索＋線形ＬＳＨ検索器３３５は、データベース索引ＬＨＳ検索器３３３から取得した検索結果セットを検索対象データセット７０として線形ＬＳＨ検索器３３２を呼び出し、検索結果リストを得る。

【０１６４】

ステップＳ１００４：データベース索引ＬＳＨ検索＋線形ＬＳＨ検索器３３５は、線形ＬＳＨ検索器３３２から取得した検索結果リストに含まれる行ＩＤの集合を検索結果８０として出力し、処理を終了する。

【０１６５】

ここで、上述した５種類の検索方式の選択基準について説明する。データ件数をＮ、特徴ベクトルの次元数をＤとするとき、厳密検索、線形ＬＳＨ検索、データベース索引ＬＳＨ検索の３つ検索方式の計算コスト（積算相当回数）は以下となる。

厳密検索：Ｎ×Ｄ

線形ＬＳＨ検索：Ｎ（ＬＳＨアルゴリズムの出力ビット長を６４以下とした場合）

データベース索引ＬＳＨ検索：α（データベース索引コスト）

ここで、αはＮよりはるかに小さいため、上記３つの検索方式の中ではデータベース索引ＬＳＨ検索が最も低コストである。

【０１６６】

精度については、厳密検索が最も高く、次に線形ＬＳＨ検索が高く、データベース索引ＬＳＨ検索が最も低い。近似的手法であるＬＳＨを用いない厳密検索が最も高精度なのは当然だが、仮に同じＬＳＨアルゴリズムを用いていても、線形ＬＳＨ検索よりもデータベース索引ＬＳＨ検索のほうが精度が低くなる理由は、データベース索引ＬＳＨ検索では大きなビット長、ハミング距離を扱えないためである。データベース索引ＬＳＨ検索は、ハミング距離以下のすべてのハッシュをレコードとして展開するＬＳＨ近傍展開テーブルを内部参照する。ハミング距離をＨ、ビット長をＬとするとき、レコード数Ｓは下記式（１）で表される。

【数１】

$$S = N\sum_{h}^{H} {}_{L}C_{h} \qquad \cdots(1)$$

【０１６７】

この式（１）から分かるように、大きなハミング距離、ビット長のＬＳＨを扱うと、レコード数が爆発するという問題がある。このため、小さなハミング距離、ビット長のＬＳＨを扱うことになるが、その場合、距離分解能が低下し、実際に出力する件数を要求された出力件数に正確に合わせることが困難となる。この問題は、線形ＬＳＨ検索、データベース索引ＬＳＨ検索など、ＬＳＨを用いるいずれの検索手法でも起こりえるが、特に小さなハミング距離、ビット長のＬＳＨしか扱えないデータベース索引ＬＳＨ検索において顕著である。この欠点を解決するには、データベース索引ＬＳＨ検索＋厳密検索や、データベース索引ＬＳＨ検索＋線形ＬＳＨ検索のように、出力件数を制御しやすい検索方式を２段階目に持つ２段階絞込方式を用いるようにし、１段階目のデータベース索引ＬＳＨ検索では多めの出力件数で出力し、２段階目の検索で出力件数を要求に正確に合わせるようにすることが有効である。

【０１６８】

以上のことから、検索速度を許容範囲内に収めつつなるべく高い精度を得るためには、検索対象データセット７０の件数が小規模の場合は厳密検索、検索対象データセット７０の件数が中規模の場合は線形ＬＳＨ検索、検索対象データセット７０の件数が大規模の場合は、データベース索引ＬＳＨ検索＋厳密検索、あるいはデータベース索引ＬＳＨ検索＋線形ＬＳＨ検索を用いることが有効である。

【０１６９】

以上、具体的な例を挙げながら詳細に説明したように、本実施形態のデータ管理システムでは、事前処理によってＬＳＨ近傍展開索引情報４０Ｂを含む索引情報４０を生成し、検索時にはこの索引情報４０を用いて、クエリベクトルに完全一致する周辺ベクトルに対応する事例ベクトルを特定するといった計算量の少ない方法でデータテーブル２０に対するベクトル近傍検索を行うようにしている。したがって、従来の類似検索において計算量の多くを占めていたベクトル近傍検索の計算量を削減して、データ検索の実行時間を短縮することができる。

【０１７０】

＜第２実施形態＞

次に、第２実施形態のデータ管理システムについて説明する。本実施形態は、検索時に

クエリベクトルに対する距離が所定値以下の周辺ベクトルを生成し、周辺ベクトルと完全一致する事例ベクトルを見つける構成とすることで、メモリ使用量の削減およびそれに伴うＬＳＨの精度向上を実現したシステムである。

【０１７１】

上述した第１実施形態では、データベース索引ＬＳＨ検索を実現するために、内部でＬＳＨ近傍展開テーブルやデータベース複合索引を用いている。これらの構造データは、上述したように、ハッシュのビット長およびハミング距離上限が大きいとサイズが肥大化する。しかし、データベース４００の記憶領域として用いるディスクなどの記憶容量には限界があるため、大きなビット長およびハミング距離上限は扱えない場合がある。そして、十分に大きなビット長やハミング距離が扱えない場合、検索精度の低下を起こす可能性がある。

【０１７２】

本実施形態では、第１実施形態のように検索対象となる各事例ベクトルのハッシュ（二値ベクトル）である事例ハッシュの近傍にあるハッシュ（二値ベクトル）である周辺ハッシュ群を列挙する代わりに、クエリベクトルのハッシュ（二値ベクトル）の近傍にあるハッシュ（二値ベクトル）群を周辺ハッシュ群として列挙する方式を取る。この方式の場合、１つのベクトルに対して列挙したハッシュ群を記憶するだけでよいため、記憶容量の限界という問題は解消され、検索精度の低下を起こすリスクが回避される。ただし、第１実施形態と比較してクエリが複雑となるため、第１実施形態よりも検索速度は低下する。

【０１７３】

本実施形態のデータ管理システムにおける基本的な枠組みは第１実施形態と同様である。ただし、本実施形態では、検索時にＬＳＨ近傍展開索引情報４０Ｂを用いないため、索引構築器２００（図５参照）は、ＬＳＨ即値索引情報４０Ａを生成するＬＳＨ即値索引情報生成器２２０のみを備えた構成となる。また、検索時にクエリハッシュを起点とした周辺ハッシュを生成するため、検索器３００のベクトル類似性判定処理部３３０における検索方式が第１実施形態とは異なる。以下では、第１実施形態からの主な変更点となる本実施形態の検索方式について説明する。

【０１７４】

図２１は、本実施形態の検索器３００Ａの構成例を示すブロック図である。第１実施形態との違いは、ベクトル類似性判定処理部３３０Ａが、図１３に示したデータベース索引ＬＳＨ検索器３３３、データベース索引ＬＳＨ検索＋厳密検索器３３４およびデータベース索引ＬＳＨ検索＋線形ＬＳＨ検索器３３５に代えて、クエリ摂動型ＬＳＨ検索器３３６、クエリ摂動型ＬＳＨ検索＋厳密検索器３３７およびクエリ摂動型ＬＳＨ検索＋線形ＬＳＨ検索器３３８を備える点である。

【０１７５】

クエリ摂動型ＬＳＨ検索＋厳密検索器３３７およびクエリ摂動型ＬＳＨ検索＋線形ＬＳＨ検索器３３８については、２段階絞込方式の前段の検索方式がデータベース索引ＬＳＨ検索からクエリ摂動型ＬＳＨ検索に代わる他は第１実施形態のデータベース索引ＬＳＨ検索＋厳密検索器３３４およびデータベース索引ＬＳＨ検索＋線形ＬＳＨ検索器３３５と同様である。このため、以下では、クエリ摂動型ＬＳＨ検索＋厳密検索器３３７およびクエリ摂動型ＬＳＨ検索＋線形ＬＳＨ検索器３３８の説明は省略し、クエリ摂動型検索を行うクエリ摂動型ＬＳＨ検索器３３６についてのみ説明する。

【０１７６】

図２２は、クエリ摂動型ＬＳＨ検索器３３６による処理手順の一例を示すフローチャートである。クエリ摂動型ＬＳＨ検索器３３６は、例えば以下のステップＳ１１０１～ステップＳ１１１２の処理を実行して、検索結果８０を出力する。

【０１７７】

ステップＳ１１０１：クエリ摂動型ＬＳＨ検索器３３６は、空の検索結果セットを生成する。

【０１７８】

ステップＳ１１０２：クエリ摂動型ＬＳＨ検索器３３６は、vnn関数の引数として指定されたクエリベクトル、ＬＳＨアルゴリズムのアルゴリズム名および出力ビット長を取得して、クエリハッシュを生成する。

【０１７９】

ステップＳ１１０３：クエリ摂動型ＬＳＨ検索器３３６は、検索対象データセット７０のテーブル名、vnn関数の引数として指定された事例ベクトルを格納する列、ＬＳＨアルゴリズムのアルゴリズム名および出力ビット長を取得して、これらを元に決定されるテーブル名を持つＬＳＨ即値テーブルを取得する。

【０１８０】

ステップＳ１１０４：クエリ摂動型ＬＳＨ検索器３３６は、ハミング距離条件変数に０を割り当てる。

【０１８１】

ステップＳ１１０５：クエリ摂動型ＬＳＨ検索器３３６は、ステップＳ１１０２で生成したクエリハッシュからの距離がハミング距離条件変数の現在地と一致する周辺ハッシュをすべて生成する。

【０１８２】

ステップＳ１１０６：クエリ摂動型ＬＳＨ検索器３３６は、ステップＳ１１０５で生成した周辺ハッシュを順に取り出す。

【０１８３】

ステップＳ１１０７：クエリ摂動型ＬＳＨ検索器３３６は、ステップＳ１１０３で取得したＬＳＨ即値テーブルから、ステップＳ１１０６で取り出した周辺ハッシュと同じ値を持つレコード群をすべて取り出し、各レコードの行ＩＤを検索結果セットに追加する。

【０１８４】

ステップＳ１１０８：クエリ摂動型ＬＳＨ検索器３３６は、ステップＳ１１０５で生成した周辺ハッシュをすべて取り出したか否かを判定する。そして、判定の結果がＮｏであればステップＳ１１０６に戻って以降の処理を繰り返し、判定の結果がＹｅｓであればステップＳ１１０９に進む。

【０１８５】

ステップＳ１１０９：クエリ摂動型ＬＳＨ検索器３３６は、検索結果セットから重複要素を除去する。

【０１８６】

ステップＳ１１１０：クエリ摂動型ＬＳＨ検索器３３６は、検索結果セットの要素数がvnn関数の第３引数として指定された出力上位件数以上となっているか否かを判定する。そして、判定の結果がＮｏであればステップＳ１１１１に進み、ＹｅｓであればステップＳ１１１２に進む。

【０１８７】

ステップＳ１１１１：クエリ摂動型ＬＳＨ検索器３３６は、ハミング距離条件変数の値を１追加する。

【０１８８】

ステップＳ１１１２：クエリ摂動型ＬＳＨ検索器３３６は、検索結果セットを検索結果８０として出力し、処理を終了する。

【０１８９】

以上説明したように、本実施形態のデータ管理システムでは、クエリベクトルに類似する周辺ベクトルを生成し、この周辺ベクトルに完全一致する事例ベクトルを特定するといった計算量の少ない方法でデータテーブル２０に対するベクトル近傍検索を行うようにしている。したがって、第１実施形態と同様に、従来の類似検索において計算量の多くを占めていたベクトル近傍検索の計算量を削減して、データ検索の実行時間を短縮することができる。

【０１９０】

また、本実施形態では、第１実施形態と比較して記憶容量に対する制約が緩和されるこ

とで、比較的大きなビット長およびハミング距離上限を扱うことが可能になるため、ビット長およびハミング距離上限の制限に伴って検索精度の低下を起こすといったリスクが回避される。

【０１９１】

＜第３実施形態＞

次に、第３実施形態のデータ管理システムについて説明する。本実施形態は、ベクトルの写像に用いるＬＳＨの方式が第１実施形態とは異なる。すなわち、第１実施形態では写像処理にビットワイズＬＳＨを用いていたが、本実施形態では、写像処理に直積量子化ＬＳＨを用いる。

【０１９２】

直積量子化ＬＳＨは、与えられたベクトルを整数ベクトル（整数の要素からのみ構成される多値ベクトル）に変換する。この整数ベクトルに対して第１実施形態で示したような索引手法を導入することで、検索時におけるベクトル距離計算を排除することができる。なお、第１実施形態と同様に、取り扱う事例ベクトルおよびクエリベクトルが元々低次元の整数ベクトルであるならば、直積量子化ＬＳＨによる写像は不要であり、後述の事例ハッシュとして事例ベクトルをそのまま用い、クエリハッシュとしてクエリベクトルをそのまま用いればよい。また、この場合、後述の周辺ハッシュとして、事例ベクトルを元に生成される周辺ベクトルを用いればよい。

【０１９３】

直積量子化ＬＳＨの代表的なアルゴリズムは、下記の参考文献４に記載されている。直積量子化ＬＳＨとしては、この参考文献４に記載のアルゴリズムを含め、様々なアルゴリズムが提案されている。

参考文献４：Herve Jegou, Matthijs Douze and Cordelia Schmid, “Product quantization for nearest neighbor search”, Pattern Analysis and Machine Intelligence, IEEE Transactions on 33.1（2011）：117-128.

【０１９４】

直積量子化ＬＳＨによるハッシュ生成の代表的な手順は以下の通りである。

（１）まず、与えられた事例ベクトル群が属する空間から、（好ましくは互いに排他的な）部分空間を複数取得する。

（２）各部分空間において、事例ベクトル群を予め定めた数のクラスタに分類する。

（３）各事例ベクトルについて、各部分空間のクラスタ番号を並べた整数ベクトルを生成する。これが写像後のベクトルである。

（４）各クラスタの分布などを用いて、写像後のベクトル間の差異から、写像前のベクトルの距離を近似的に算出するための距離モデルを生成する。

【０１９５】

上記の手順（４）で生成される距離モデルはアルゴリズムによって異なるが、いずれの距離モデルであっても、ハッシュである２つの整数ベクトルが与えられたときに、元のベクトル空間での距離を概算できる性質を持つ。また多くのアルゴリズムでは、クラスタ番号が隣接する場合、元のベクトル空間におけるクラスタの分布領域も近接している。次元数は変えず、元のベクトル空間において格子状に並んだ最近傍の点で代表させる格子ベクトル量子化は、最も単純な直積量子化ＬＳＨアルゴリズムである。

【０１９６】

本実施形態のデータ管理システムにおける基本的な枠組みは第１実施形態と同様である。本実施形態の第１実施形態からの変更点は、整数ベクトルをハッシュとして扱うこと、および、ベクトル間距離としてハミング距離を用いる代わりに直積量子化ＬＳＨが提供する距離モデルによって概算できる距離を用いるようにすることである。以下では、これらの変更点について説明する。

【０１９７】

図２３は、本実施形態の索引構築器２００Ｂの構成例を示すブロック図である。本実施形態の索引構築器２００Ｂは、図２３に示すように、第１実施形態の索引構築器２００に

おけるＬＳＨ即値索引情報生成器２２０（図５参照）に代えてＰＱＬＳＨ即値索引情報生成器２４０を備えるとともに、第１実施形態の索引構築器２００におけるＬＳＨ近傍展開索引情報生成器２３０（図５参照）に代えてＰＱＬＳＨ近傍展開索引情報生成器２５０を備える。

【０１９８】

ＰＱＬＳＨ即値索引情報生成器２４０は、索引構築命令タイプ分類器２１０から索引構築命令文３０を受け取った場合に、この索引構築命令文３０に従って、ＰＱＬＳＨ即値テーブルと、このＰＱＬＳＨ即値テーブルのHashValue列に対するデータベース索引とを含むＰＱＬＳＨ即値索引情報４０Ｃを生成する。ＰＱＬＳＨ即値索引情報生成器２４０が生成するＰＱＬＳＨ即値索引情報４０Ｃは、ＰＱＬＳＨ即値テーブルのHashValue列に格納される値が二値ベクトルではなく直積量子化ＬＳＨアルゴリズムを用いて生成された整数ベクトルとなる点を除き、第１実施形態で説明したＬＳＨ即値索引情報４０Ａと同様である。このため、ＰＱＬＳＨ即値索引情報生成器２４０については詳細な説明を省略する。

【０１９９】

ＰＱＬＳＨ近傍展開索引情報生成器２５０は、索引構築命令タイプ分類器２１０から索引構築命令文３０を受け取った場合に、この索引構築命令文３０に従って、ＰＱＬＳＨ近傍展開テーブルと、このＰＱＬＳＨ近傍展開テーブルのHashValue列およびDistance列に対するデータベース複合索引（コンポジットインデックス）とを含むＰＱＬＳＨ近傍展開索引情報４０Ｄを生成する。

【０２００】

図２４は、ＰＱＬＳＨ近傍展開テーブルの一例を示す図である。ＰＱＬＳＨ近傍展開テーブルは、図２４に示すように、HashValue列（型：blob）、Distance列（型：real）およびDataRowID列（型：integer）からなるテーブルデータである。この図２４に示す例では、要素値 {0,1,2} を取る長さ２の整数ベクトルのハッシュを用い、距離上限を１．０とした。Distance列の値は、直積量子化ＬＳＨアルゴリズムの提供する距離モデルによって概算される距離である。本例では、ハッシュの第１要素値が１変化すると距離が０．４、第２要素値が１変化すると距離が１．０変化するという距離モデルを用いるものとした。

【０２０１】

ＰＱＬＳＨ近傍展開索引情報生成器２５０は、索引構築命令文３０で指定されたデータテーブル２０から特徴ベクトルを取り出し、各特徴ベクトル（事例ベクトル）について、索引構築命令文３０で指定されたＬＳＨアルゴリズムおよび距離モデルを用いて整数ベクトルである事例ハッシュを生成する。次に、ＰＱＬＳＨ近傍展開索引情報生成器２５０は、生成した各事例ハッシュについて、索引構築命令文３０で指定された距離以下の周辺ハッシュ群を生成する。そして、ＰＱＬＳＨ近傍展開索引情報生成器２５０は、例えば、周辺ハッシュと、周辺ハッシュの事例ハッシュからの距離と、事例ハッシュの元になる事例ベクトルが属するデータテーブル２０のレコードの行ＩＤとの３つ組からなるレコードを、ＰＱＬＳＨ近傍展開テーブルに登録する。

【０２０２】

なお、ＰＱＬＳＨ近傍展開テーブルのHashValue列およびDistance列に対するデータベース複合索引は、図１０に示した第１実施形態のデータベース複合索引とほとんど同じであり、図１０のHammingDistanceの値に沿った分岐が、Distanceの値に沿った分岐に変わるだけである。

【０２０３】

図２５は、ＰＱＬＳＨ近傍展開索引情報生成器２５０による処理手順の一例を示すフローチャートである。ＰＱＬＳＨ近傍展開索引情報生成器２５０は、索引構築命令タイプ分類器２１０から索引構築命令文３０を受け取ると、例えば以下のステップＳ１２０１～ステップＳ１２１０の処理を実行し、ＰＱＬＳＨ近傍展開テーブルとデータベース複合索引とを含むＰＱＬＳＨ近傍展開索引情報４０Ｄをデータベース４００に保存する。

【０２０４】

ステップＳ１２０１：ＰＱＬＳＨ近傍展開索引情報生成器２５０は、データベース４００内に、名前：HashValue・型：blobの列と、名前：Distance・型：realの列と、名前：DataRowID・型：integerの列とを持つテーブル（空のＰＱＬＳＨ近傍展開テーブル）を生成する。なお、テーブル名の形式は「AAAA_BBBB_CCCC_DDDD」とする。ただし、AAAAは処理対象のデータテーブル２０のテーブル名、BBBBは処理対象の列の列名、CCCCは直積量子化ＬＳＨアルゴリズムのアルゴリズム名、DDDDはモデル（アルゴリズムに関する設定や分析結果）名とする。例えば、aTable_feat_PQ_confAといったテーブル名が生成したテーブルに与えられる。

【０２０５】

ステップＳ１２０２：ＰＱＬＳＨ近傍展開索引情報生成器２５０は、索引構築命令文３０で指定されたデータテーブル２０から、索引構築命令文３０で指定された列の特徴ベクトルを順に取り出す。

【０２０６】

ステップＳ１２０３：ＰＱＬＳＨ近傍展開索引情報生成器２５０は、ステップＳ１２０２で取り出した特徴ベクトル（事例ベクトル）に対して、索引構築命令文３０で指定された直積量子化ＬＳＨアルゴリズムおよびモデルを用いて、事例ハッシュを生成する。

【０２０７】

ステップＳ１２０４：ＰＱＬＳＨ近傍展開索引情報生成器２５０は、ステップＳ１２０３で得られた事例ハッシュからの距離が指定距離以下の周辺ハッシュをすべて生成する。ただし、起点となる事例ハッシュと各周辺ハッシュとの距離は、直積量子化ＬＳＨアルゴリズムのモデルを用いて概算する。

【０２０８】

ステップＳ１２０５：ＰＱＬＳＨ近傍展開索引情報生成器２５０は、ステップＳ１２０４で得られた周辺ハッシュを順に取り出す。

【０２０９】

ステップＳ１２０６：ＰＱＬＳＨ近傍展開索引情報生成器２５０は、ステップＳ１２０５で取り出した周辺ハッシュ、起点となった事例ハッシュからの距離、事例ハッシュの元になる事例ベクトルが属するデータテーブル２０内のレコードの行ＩＤの３つ組からなるレコードを、ＰＱＬＳＨ近傍展開テーブルに追加する。

【０２１０】

ステップＳ１２０７：ＰＱＬＳＨ近傍展開索引情報生成器２５０は、ステップＳ１２０４で得られた周辺ハッシュをすべて取り出したか否かを判定する。そして、判定の結果がＮｏであればステップＳ１２０５に戻って以降の処理を繰り返し、ＹｅｓであればステップＳ１２０８に進む。

【０２１１】

ステップＳ１２０８：ＰＱＬＳＨ近傍展開索引情報生成器２５０は、処理対象のデータテーブル２０から特徴ベクトルをすべて取り出したか否かを判定する。そして、判定の結果がＮｏであればステップＳ１２０２に戻って以降の処理を繰り返し、ＹｅｓであればステップＳ１２０９に進む。

【０２１２】

ステップＳ１２０９：ＰＱＬＳＨ近傍展開索引情報生成器２５０は、以上の処理を経て生成されたＰＱＬＳＨ近傍展開テーブルのHashValue列およびDistance列に対するデータベース複合索引（コンポジットインデックス）を構築し、データベース４００に保存する。

【０２１３】

ステップＳ１２１０：ＰＱＬＳＨ近傍展開索引情報生成器２５０は、事例ハッシュの生成に用いた直積量子化ＬＳＨアルゴリズムのモデルを、テーブル名と同じファイル名（例えばaTable_feat_PQ_confA）でデータベース４００に保存する。

【０２１４】

なお、上記のステップＳ１２０４で周辺ハッシュを生成する方法は第１実施形態と似た

方法を用いるが、本実施形態では、ハッシュのどの要素を変更するかだけでなく、どれだけ値を変更するかも決定する必要がある。これには、例えばダイクストラ法などの最良優先探索などを用いればよい。

【０２１５】

図２６は、本実施形態の検索器３００Ｂの構成例を示すブロック図である。第１実施形態との違いは、ベクトル類似性判定処理部３３０Ｂが、図１３に示した線形ＬＳＨ検索器３３２、データベース索引ＬＳＨ検索器３３３、データベース索引ＬＳＨ検索＋厳密検索器３３４およびデータベース索引ＬＳＨ検索＋線形ＬＳＨ検索器３３５に代えて、線形ＰＱＬＳＨ検索器３５１、データベース索引ＰＱＬＳＨ検索器３５２、データベース索引ＰＱＬＳＨ検索＋厳密検索器３５３およびデータベース索引ＰＱＬＳＨ検索＋線形ＰＱＬＳＨ検索器３５４を備える点である。

【０２１６】

線形ＰＱＬＳＨ検索器３５１は、扱うベクトルが第１実施形態からの実装変更に伴って二値ベクトルから整数ベクトルに変わる他は、第１実施形態の線形ＬＳＨ検索器３３２と同様である。また、データベース索引ＰＱＬＳＨ検索＋厳密検索器３５３およびデータベース索引ＰＱＬＳＨ検索＋線形ＰＱＬＳＨ検索器３５４については、２段階絞込方式の前段の検索方式がデータベース索引ＬＳＨ検索からデータベース索引ＰＱＬＳＨ検索に代わる他は、第１実施形態のデータベース索引ＬＳＨ検索＋厳密検索器３３４およびデータベース索引ＬＳＨ検索＋線形ＬＳＨ検索器３３５と同様である。このため、以下では、線形ＰＱＬＳＨ検索器３５１、データベース索引ＰＱＬＳＨ検索＋厳密検索器３５３およびデータベース索引ＰＱＬＳＨ検索＋線形ＰＱＬＳＨ検索器３５４の説明は省略し、データベース索引ＰＱＬＳＨ検索を行うデータベース索引ＰＱＬＳＨ検索器３５２についてのみ説明する。なお、線形ＬＳＨ検索については、第１実施形態から変更せずにそのまま用いることも可能である。

【０２１７】

図２７は、データベース索引ＰＱＬＳＨ検索器３５２による処理手順の一例を示すフローチャートである。データベース索引ＰＱＬＳＨ検索器３５２は、例えば以下のステップＳ１３０１～ステップＳ１３１０の処理を実行して、検索結果８０を出力する。

【０２１８】

ステップＳ１３０１：データベース索引ＰＱＬＳＨ検索器３５２は、空の検索結果セットを生成する。

【０２１９】

ステップＳ１３０２：データベース索引ＰＱＬＳＨ検索器３５２は、vnn関数の引数として指定されたクエリベクトル、直積量子化ＬＳＨアルゴリズムのアルゴリズム名およびオプションパラメタを取得して、クエリハッシュを生成する。

【０２２０】

ステップＳ１３０３：データベース索引ＰＱＬＳＨ検索器３５２は、検索対象データセット７０のテーブル名、vnn関数の引数として指定された事例ベクトルを格納する列、直積量子化ＬＳＨアルゴリズムのアルゴリズム名およびオプションパラメタを取得して、これらを元に決定されるテーブル名を持つＰＱＬＳＨ近傍展開テーブルを取得する。

【０２２１】

ステップＳ１３０４：データベース索引ＰＱＬＳＨ検索器３５２は、距離条件変数に０を割り当てる。

【０２２２】

ステップＳ１３０５：データベース索引ＰＱＬＳＨ検索器３５２は、ステップＳ１３０３で取得したＰＱＬＳＨ近傍展開テーブルに対して、Distance列の値が距離条件変数の現在値以下であり、かつ、HashValue列の値がステップＳ１３０２で生成したクエリハッシュと一致するレコード群を検索し、得られた行ＩＤのセットを検索結果セットに追加する。

【０２２３】

ステップＳ１３０６：データベース索引ＰＱＬＳＨ検索器３５２は、検索結果セットから重複要素を除去する。

【０２２４】

ステップＳ１３０７：データベース索引ＰＱＬＳＨ検索器３５２は、検索結果セットの要素数がvnn関数の第３引数として指定された出力上位件数以上となっているか否かを判定する。そして、判定の結果がＮｏであればステップＳ１３０８に進み、ＹｅｓであればステップＳ１３１０に進む。

【０２２５】

ステップＳ１３０８：データベース索引ＰＱＬＳＨ検索器３５２は、距離条件変数を一定量追加する。

【０２２６】

ステップＳ１３０９：データベース索引ＰＱＬＳＨ検索器３５２は、ハミング距離条件変数の値がＰＱＬＳＨ近傍展開テーブルの距離上限以下か否かを判定する。そして、判定の結果がＹｅｓであればステップＳ１３０５に戻って以降の処理を繰り返し、ＮｏであればステップＳ１３１０に進む。

【０２２７】

ステップＳ１３１０：データベース索引ＰＱＬＳＨ検索器３５２は、検索結果セットを検索結果８０として出力し、処理を終了する。

【０２２８】

以上、具体的な例を挙げながら詳細に説明したように、本実施形態のデータ管理システムでは、事前処理によってＰＱＬＳＨ近傍展開索引情報４０Ｄを含む索引情報４０を生成し、検索時にはこの索引情報４０を用いて、クエリベクトルに完全一致する周辺ベクトルに対応する事例ベクトルを特定するといった計算量の少ない方法でデータテーブル２０に対するベクトル近傍検索を行うようにしている。したがって、第１実施形態と同様に、従来の類似検索において計算量の多くを占めていたベクトル近傍検索の計算量を削減して、データ検索の実行時間を短縮することができる。

【０２２９】

＜補足説明＞

上述した実施形態のデータ管理システムは、一例として、一般的なコンピュータとしてのハードウェアを用いた実行環境で動作するプログラムによる実装が可能である。この場合、本実施形態のデータ管理システムにおける上述の各機能的な構成要素（データ登録器１００、索引構築器２００（索引構築器２００Ｂ）、検索器３００（検索器３００Ａ、検索器３００Ｂ））は、ハードウェアとソフトウェア（プログラム）との協働により実現される。また、データベース４００は、プログラムによってアクセス可能な任意のメモリ資源によって実現される。

【０２３０】

図２８は、データ管理システムのハードウェア構成例を示すブロック図である。データ管理システムは、例えば図２８に示すように、ＣＰＵ（Central Processing Unit）１００１などのプロセッサ回路、ＲＯＭ（Read Only Memory）１００２やＲＡＭ（Random Access Memory）１００３などの記憶装置、表示パネルや各種操作デバイスが接続される入出力Ｉ／Ｆ１００４、ネットワークに接続して通信を行う通信Ｉ／Ｆ１００５、各部を接続するバス１００６などを備えた、通常のコンピュータを利用したハードウェア構成とすることができる。

【０２３１】

また、上述した構成のハードウェア上で実行されるプログラムは、例えば、インストール可能な形式または実行可能な形式のファイルでＣＤ－ＲＯＭ（Compact Disk Read Only Memory）、フレキシブルディスク（ＦＤ）、ＣＤ－Ｒ（Compact Disk Recordable）、ＤＶＤ（Digital Versatile Disc）などのコンピュータで読み取り可能な記録媒体に記録されてコンピュータプログラムプロダクトとして提供される。また、上述した構成のハードウェア上で実行されるプログラムを、インターネットなどのネットワークに接続

されたコンピュータ上に格納し、ネットワーク経由でダウンロードさせることにより提供するように構成してもよい。また、上述した構成のハードウェア上で実行されるプログラムをインターネットなどのネットワーク経由で提供または配布するように構成してもよい。また、上述した構成のハードウェア上で実行されるプログラムを、ＲＯＭ１００２などに予め組み込んで提供するように構成してもよい。

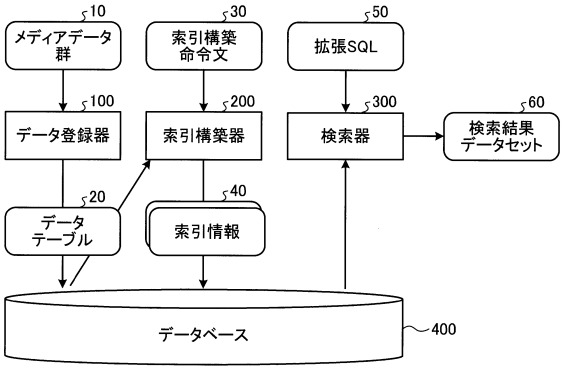
【０２３２】

上述した構成のハードウェア上で実行されるプログラムは、実施形態のデータ管理システムの各機能的な構成要素を含むモジュール構成となっており、例えば、ＣＰＵ１００１（プロセッサ回路）が上記記録媒体からプログラムを読み出して実行することにより、上述した各部がＲＡＭ１００３（主記憶）上にロードされ、ＲＡＭ１００３（主記憶）上に生成されるようになっている。なお、実施形態のデータ管理システムの各機能的な構成要素やデータベース４００は、複数のコンピュータに跨って実現される構成であってもよい。また、上述の機能的な構成要素の一部または全部を、ＡＳＩＣ（Application Specific Integrated Circuit）やＦＰＧＡ（Field-Programmable Gate Array）などの専用のハードウェアを用いて実現することも可能である。

【０２３３】

以上、本発明の実施形態を説明したが、ここで説明した実施形態は、例として提示したものであり、発明の範囲を限定することは意図していない。ここで説明した新規な実施形態は、その他の様々な形態で実施されることが可能であり、発明の要旨を逸脱しない範囲で、種々の省略、置き換え、変更を行うことができる。ここで説明した実施形態やその変形は、発明の範囲や要旨に含まれるとともに、請求の範囲に記載された発明とその均等の範囲に含まれる。

【図１】

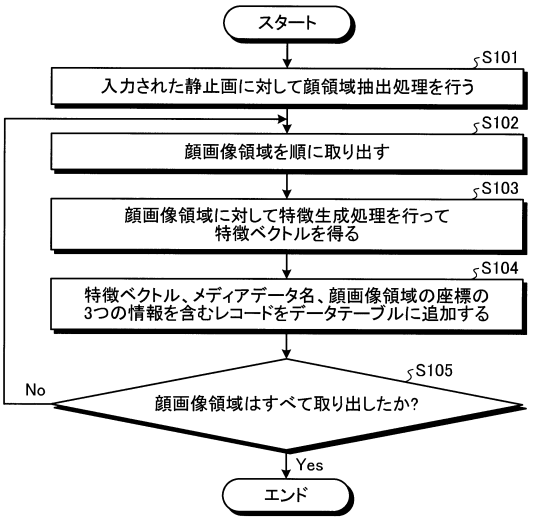


【図２】

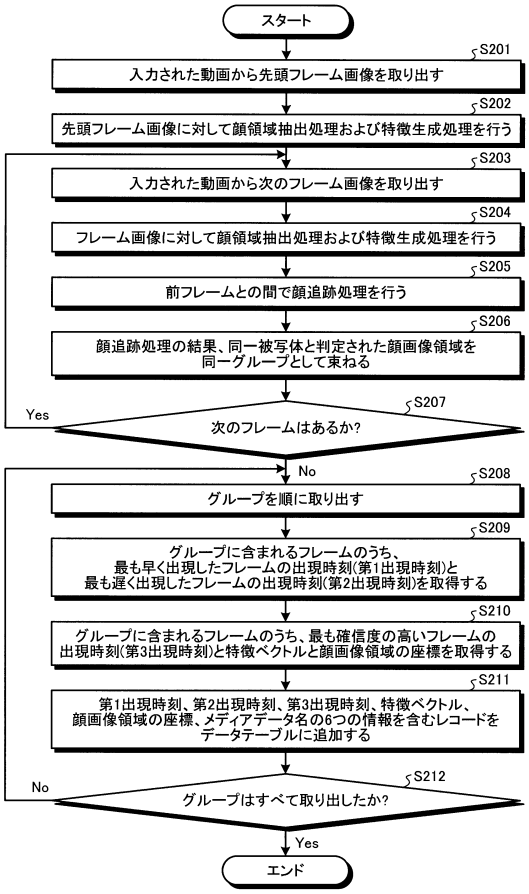
20

file	feat	region	begin_time	end_time	best_time
dir/a.jpg	(32,44,33)	(10,10,300,300)			
dir/a.jpg	(76,11,74)	(300,300,600,600)			
dir/b.jpg	(76,12,64)	(400,400,800,800)			
dir/a.mpg	(10,33,42)	(250,280,450,650)	22:04:10:29	22:05:00:12	22:04:30:28
dir/a.mpg	(80,93,12)	(80,80,200,200)	32:24:21:12	33:04:40:31	32:28:22:29
⋮	⋮	⋮	⋮	⋮	⋮

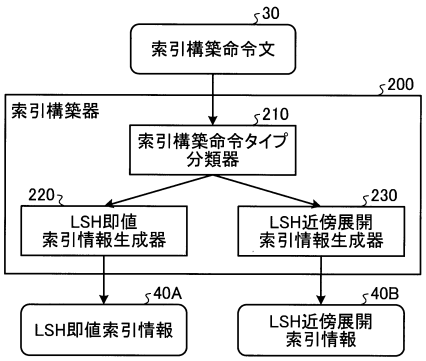
【図３】



【図４】



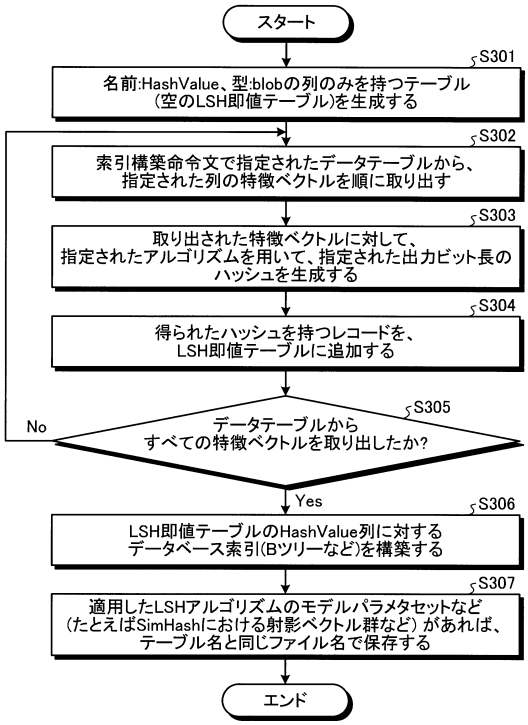
【図５】



【図６】

HashValue
100
010
000
001
011
⋮

【図７】



【図８】

HashValue	HammingDistance	DataRowID
100	0	1
000	1	1
110	1	1
101	1	1
010	0	2
110	1	2
000	1	2
011	1	2
000	0	3
100	1	3
010	1	3
001	1	3
001	0	4
101	1	4
011	1	4
000	1	4
011	0	5
111	1	5
001	1	5
010	1	5
⋮	⋮	⋮

【図９】

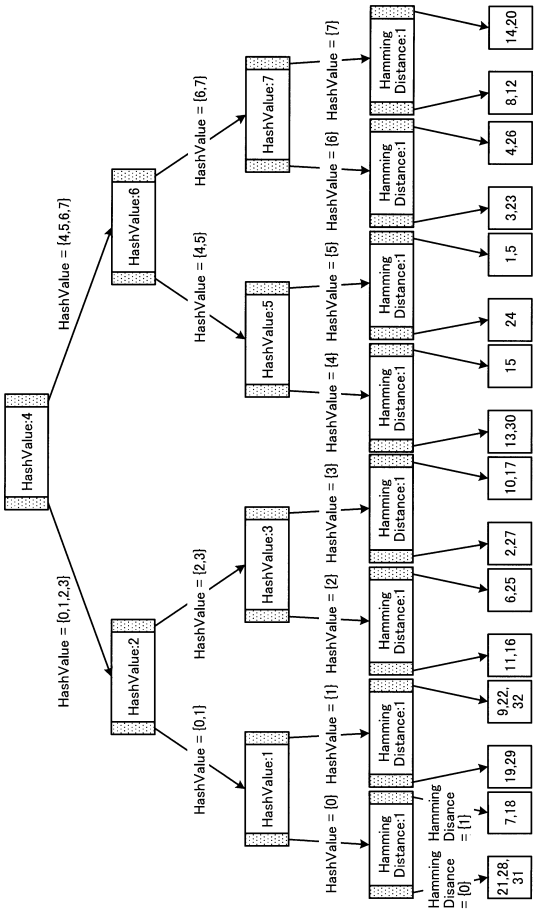
(a)

HashValue	HammingDistance
000	0
000	1
001	0
001	1
010	0
010	1
011	0
011	1
100	0
100	1
101	0
101	1
110	0
110	1
111	0
111	1
⋮	⋮

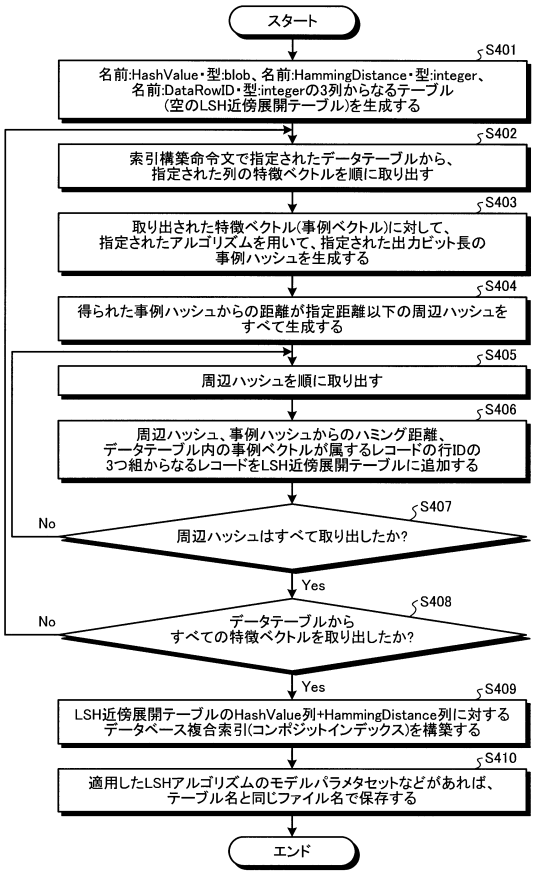
(b)

Table1_RowID	DataRowID
1	3
2	1
2	2
2	4
3	4
4	3
4	5
5	2
6	3
6	5
7	5
8	2
8	4
9	1
10	3
12	1
12	4
14	1
14	2
16	5
⋮	⋮

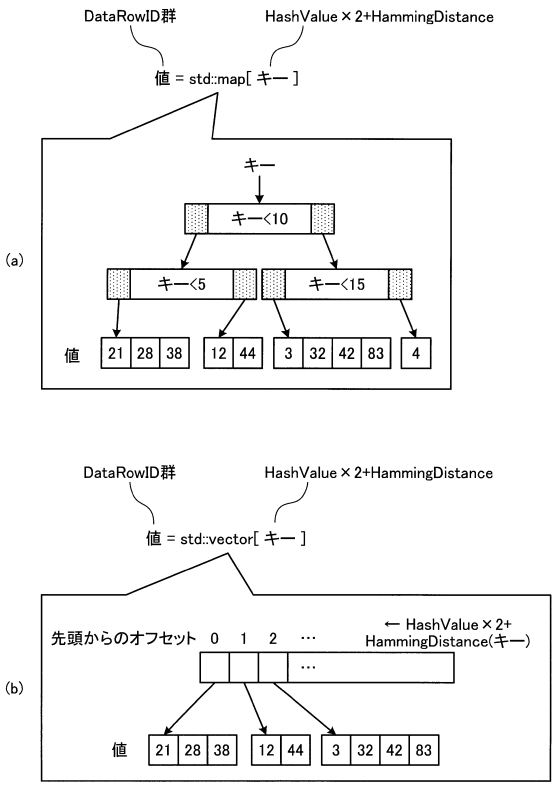
【図１０】



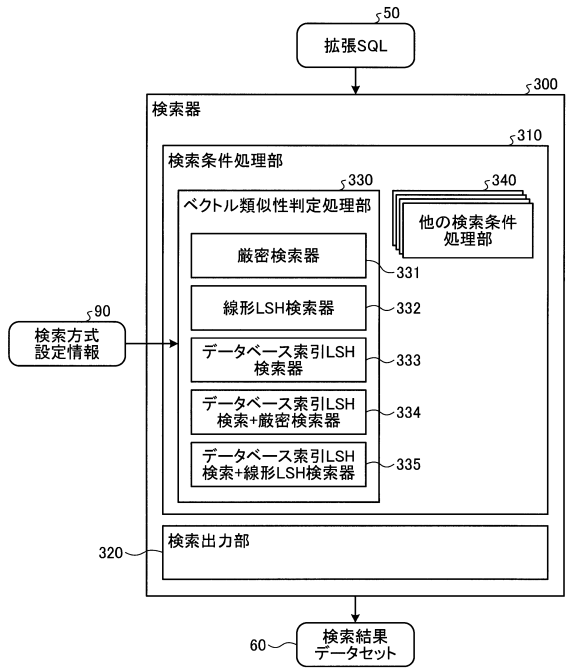
【図１１】



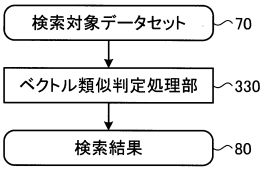
【図１２】



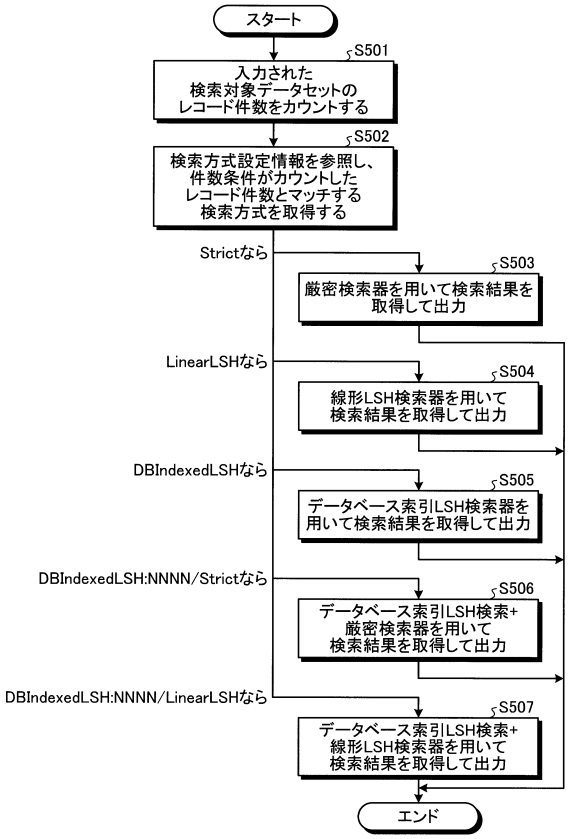
【図１３】



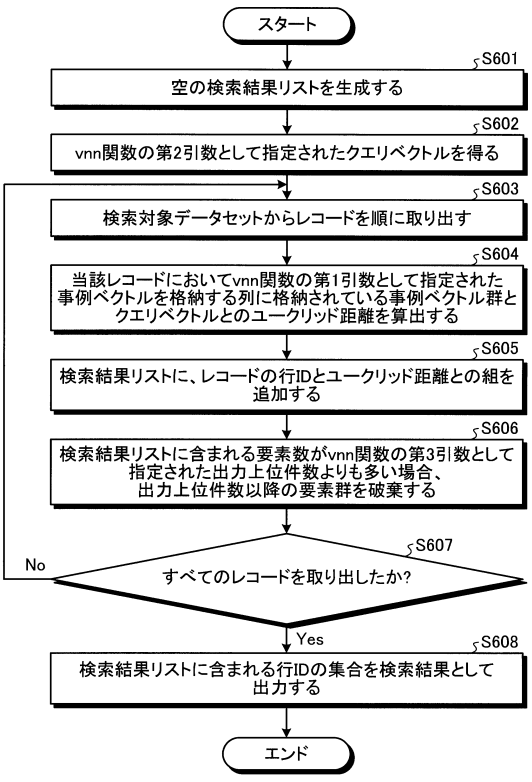
【図１４】



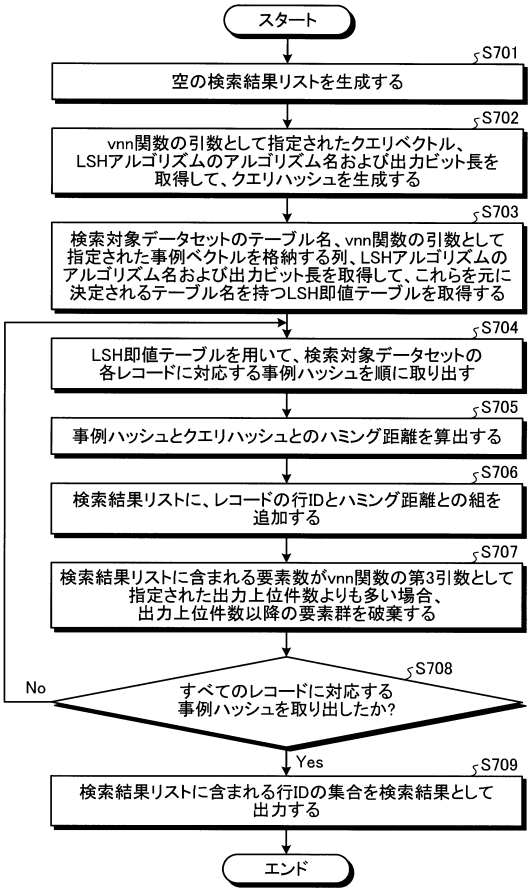
【図１５】



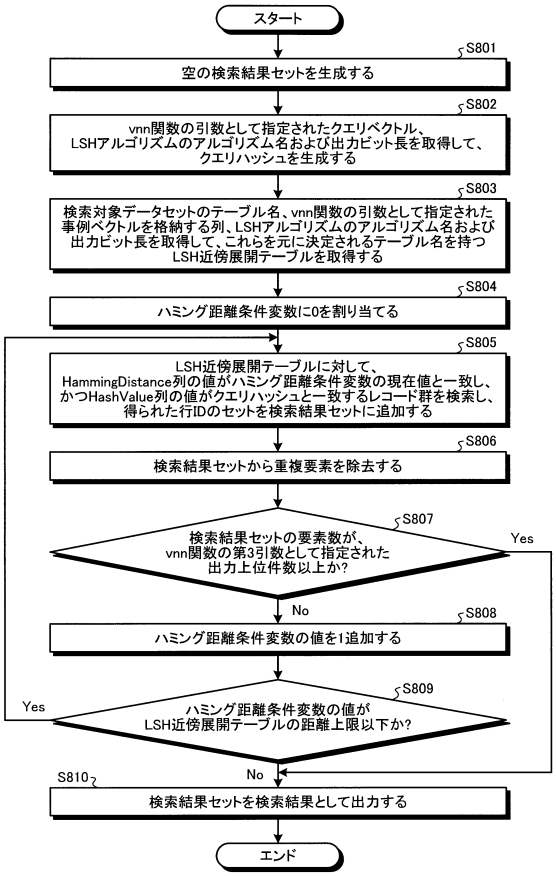
【図１６】



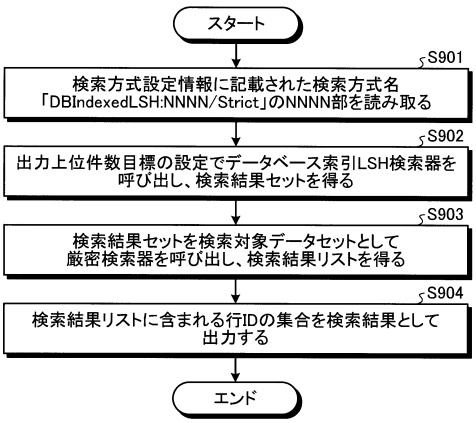
【図１７】



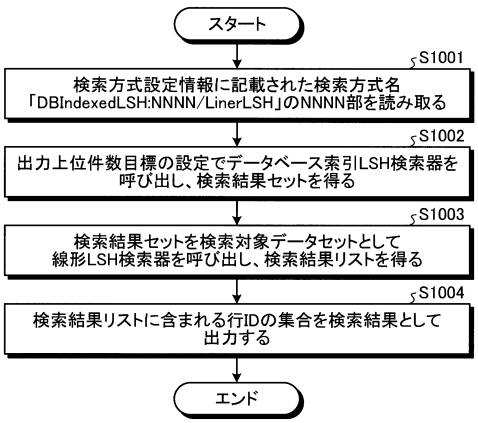
【図１８】



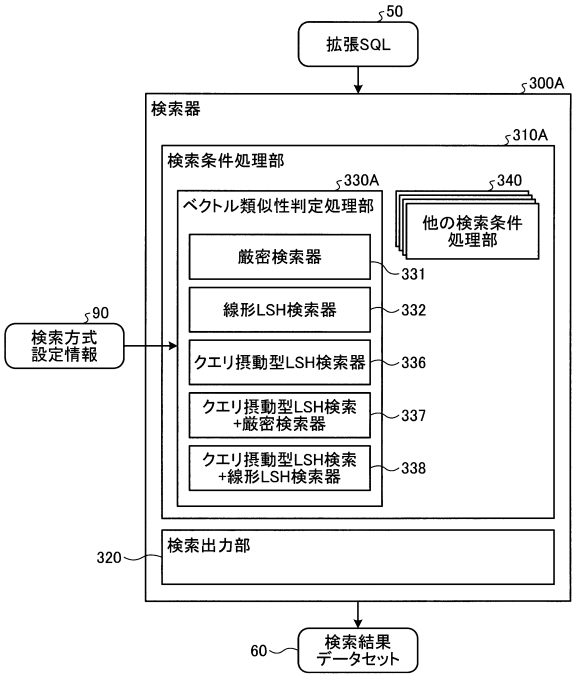
【図１９】



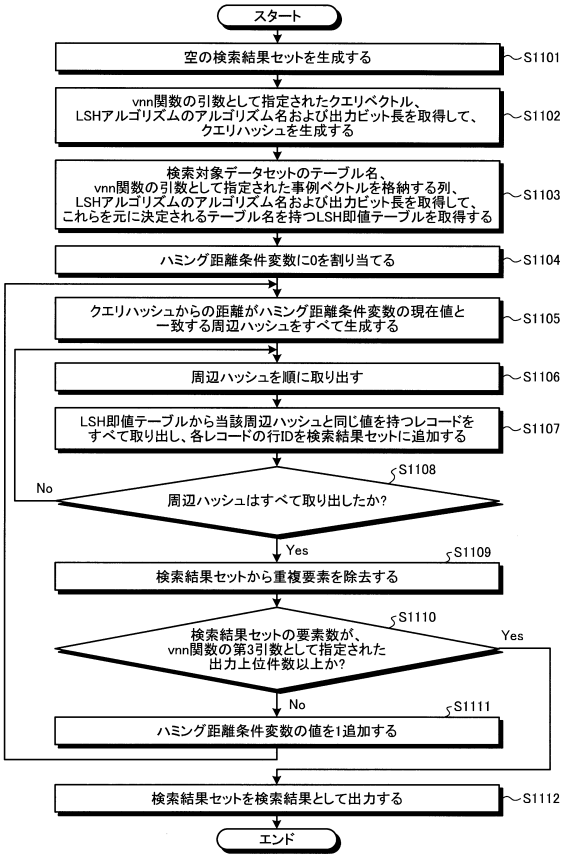
【図２０】



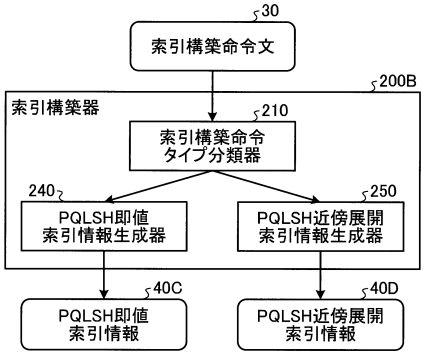
【図２１】



【図２２】



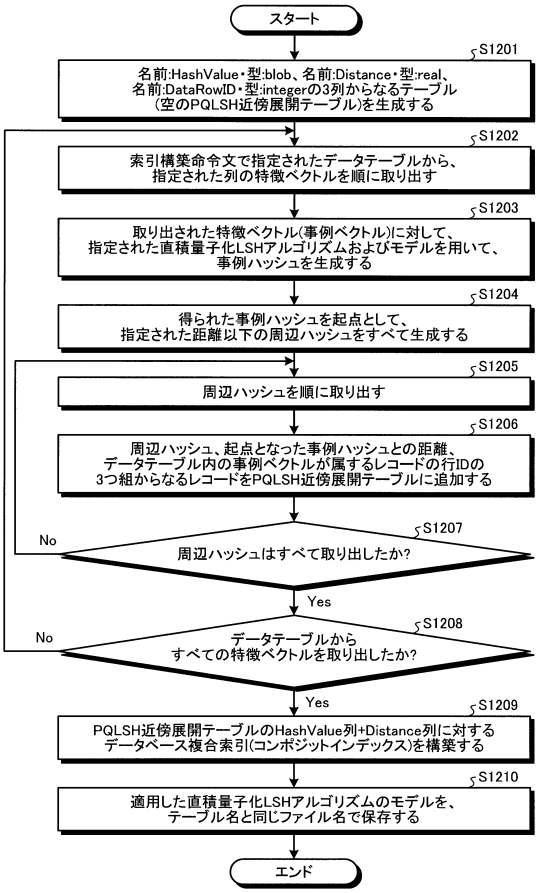
【図２３】



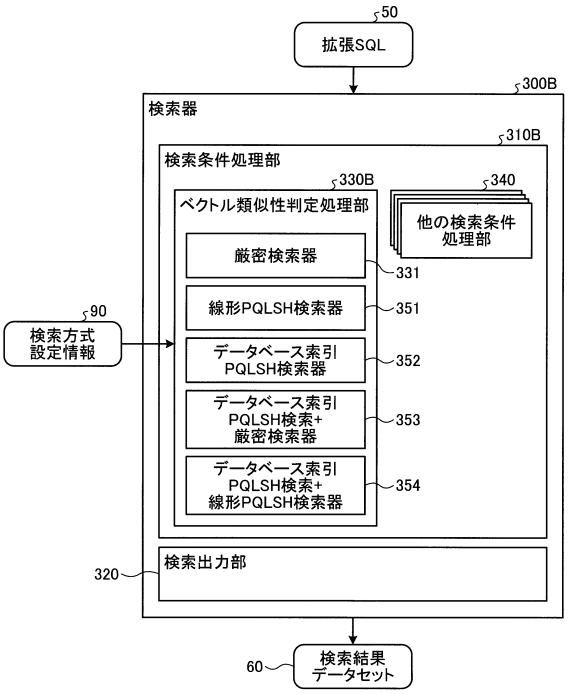
【図２４】

HashValue	Distance	DataRowID
10	0	1
00	0.4	1
20	0.4	1
11	1	1
20	0	2
10	0.4	2
00	0.8	2
21	1	2
⋮	⋮	⋮

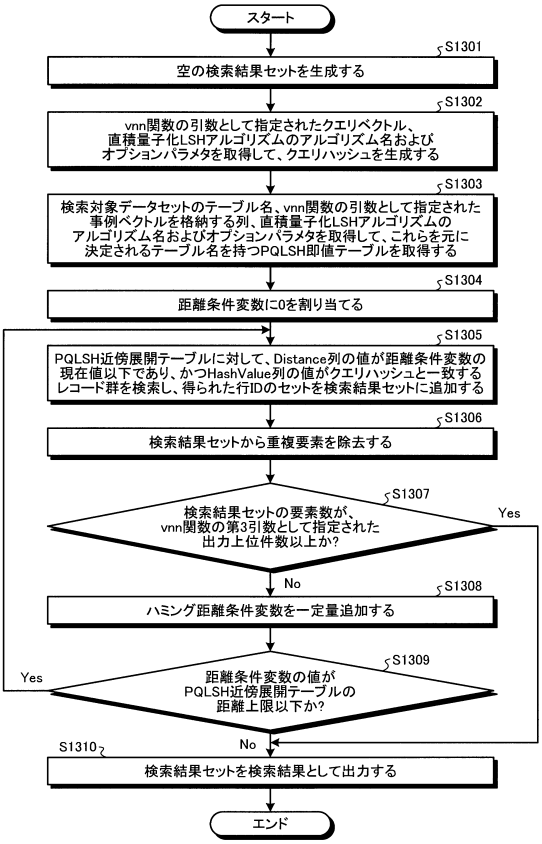
【図２５】



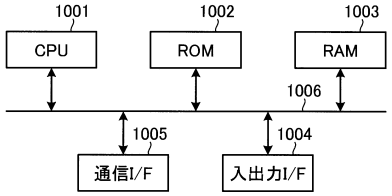
【図２６】


50
拡張SQL
300B
検索器
310B
検索条件処理部
330B
ベクトル類似性判定処理部
340
他の検索条件処理部
厳密検索器
331
線形PQLSH検索器
351
データベース索引 PQLSH検索器
352
データベース索引 PQLSH検索+ 厳密検索器
353
データベース索引 PQLSH検索+ 線形PQLSH検索器
354
90
検索方式設定情報
320
検索出力部
60
検索結果データセット

【図２７】


スタート
S1301
空の検索結果セットを生成する
S1302
vnn関数の引数として指定されたクエリベクトル、直積量子化LSHアルゴリズムのアルゴリズム名およびオプションパラメタを取得して、クエリハッシュを生成する
S1303
検索対象データセットのテーブル名、vnn関数の引数として指定された事例ベクトルを格納する列、直積量子化LSHアルゴリズムのアルゴリズム名およびオプションパラメタを取得して、これらを元に決定されるテーブル名を持つPQLSH即値テーブルを取得する
S1304
距離条件変数に0を割り当てる
S1305
PQLSH近傍展開テーブルに対して、Distance列の値が距離条件変数の現在値以下であり、かつHashValue列の値がクエリハッシュと一致するレコード群を検索し、得られた行IDのセットを検索結果セットに追加する
S1306
検索結果セットから重複要素を除去する
S1307
検索結果セットの要素数が、vnn関数の第3引数として指定された出力上位件数以上か?
Yes
No
S1308
ハミング距離条件変数を一定量追加する
S1309
距離条件変数の値がPQLSH近傍展開テーブルの距離上限以下か?
Yes
No
S1310
検索結果セットを検索結果として出力する
エンド

【図２８】


1001
CPU
1002
ROM
1003
RAM
1006
1005
通信I/F
1004
入出力I/F

フロントページの続き

(72)発明者　湯浅　真由美
東京都港区芝浦一丁目１番１号　株式会社東芝内
(72)発明者　長田　邦男
東京都港区芝浦一丁目１番１号　株式会社東芝内

審査官　早川　学

(56)参考文献　国際公開第２０１３／１２９５８０（WO，Ａ１）
国際公開第２００８／０２６４１４（WO，Ａ１）
BABENKO, Artem et al., The Inverted Multi-Index, 2012 IEEE Conference on Computer Visi
on and Pattern Recognition, IEEE, ２０１２年　６月２１日, pp.3069-3076

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　　１７／３０