

(19) 日本国特許庁 (JP)

(12) 特 許 公 報 (B2)

(11) 特許番号

特許第6662169号

(P6662169)

(45) 発行日 令和2年3月11日 (2020.3.11)

(24) 登録日 令和2年2月17日 (2020.2.17)

(51) Int.Cl.

F I

G 0 6 F 16/31 (2019.01)

G 0 6 F 16/31

請求項の数 9 (全 34 頁)

(21) 出願番号	特願2016-83168 (P2016-83168)	(73) 特許権者	000005223
(22) 出願日	平成28年4月18日 (2016.4.18)		富士通株式会社
(65) 公開番号	特開2017-194753 (P2017-194753A)		神奈川県川崎市中原区上小田中4丁目1番1号
(43) 公開日	平成29年10月26日 (2017.10.26)	(74) 代理人	110002147
審査請求日	平成31年1月15日 (2019.1.15)		特許業務法人酒井国際特許事務所
		(72) 発明者	片岡 正弘
			神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内
		(72) 発明者	松村 量
			神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内
		(72) 発明者	小澤 崇記
			神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

最終頁に続く

(54) 【発明の名称】 符号化プログラム、符号化方法、符号化装置、検索プログラム、検索方法および検索装置

(57) 【特許請求の範囲】

【請求項 1】

コンピュータに、

符号化対象のファイルを、静的辞書および動的辞書を用いて符号化する符号化処理と、
前記静的辞書および動的辞書に登録された単語の、前記ファイルにおける存否を示す、
ファイル軸および単語軸を有するインデックス情報を、前記ファイル軸方向および前記単語軸方向のそれぞれを、それぞれ所定の底を用いて折り返すことによりハッシュ化することにより生成するインデックス情報生成処理と、

前記ファイルが更新された際に、更新後のファイルに対応したインデックス情報の、更新前のインデックス情報に対する更新情報を、前記ファイル軸方向または前記単語軸方向

について生成する更新情報生成処理と
を行わせる符号化プログラム。

【請求項 2】

前記更新情報生成処理は、前記ファイルが追加された際に、追加されたファイルに対応したインデックス情報の、追加前のインデックス情報に対する更新情報を、前記ファイル軸方向または前記単語軸方向について生成し、生成した更新情報をハッシュ化されたインデックス情報に更新する

ことを特徴とする請求項 1 に記載の符号化プログラム。

【請求項 3】

前記インデックス情報生成処理は、さらに、前記符号化対象のファイルの存否を示す第

2 インデックス情報を、前記ファイル軸方向を所定の底でハッシュ化して折り返して生成し、

前記更新情報生成処理は、前記ファイルが削除された際に、前記第2インデックス情報の、前記ファイルの削除前の前記第2インデックス情報に対する更新情報を生成し、生成した更新情報をハッシュ化された前記第2インデックス情報に更新する

ことを特徴とする請求項1に記載の符号化プログラム。

【請求項4】

前記ファイル軸方向および前記単語軸方向の軸の大きさは、レジスタの大きさに合わせたビット数である

ことを特徴とする請求項1に記載の符号化プログラム。

【請求項5】

コンピュータが、

符号化対象のファイルを、静的辞書および動的辞書を用いて符号化し、

前記静的辞書および動的辞書に登録された単語の、前記ファイルにおける存否を示す、ファイル軸および単語軸を有するインデックス情報を、前記ファイル軸方向および前記単語軸方向のそれぞれを、それぞれ所定の底を用いて折り返すことによりハッシュ化することにより生成し、

前記ファイルが更新された際に、更新後のファイルに対応したインデックス情報の、更新前のインデックス情報に対する更新情報を、前記ファイル軸方向または前記単語軸方向について生成する

処理を実行する符号化方法。

【請求項6】

符号化対象のファイルを、静的辞書および動的辞書を用いて符号化する符号化部と、

前記静的辞書および動的辞書に登録された単語の、前記ファイルにおける存否を示す、ファイル軸および単語軸を有するインデックス情報を、前記ファイル軸方向および前記単語軸方向のそれぞれを、それぞれ所定の底を用いて折り返すことによりハッシュ化することにより生成する生成部と、

前記ファイルが更新された際に、更新後のファイルに対応したインデックス情報の、更新前のインデックス情報に対する更新情報を、前記ファイル軸方向または前記単語軸方向について生成する更新部と、

を有することを特徴とする符号化装置。

【請求項7】

コンピュータに、

検索対象の単語を受け付けると、受け付けた単語に対する、ファイルにおける存否を示すインデックス情報であってハッシュ化されたインデックス情報を復元し、

復元した第1インデックス情報と、ファイルの存否を示す第2インデックス情報とをAND演算し、前記単語が存在するファイルを検索する

処理を実行させることを特徴とする検索プログラム。

【請求項8】

コンピュータが、

検索対象の単語を受け付けると、受け付けた単語に対する、ファイルにおける存否を示すインデックス情報であってハッシュ化されたインデックス情報を復元し、

復元した第1インデックス情報と、ファイルの存否を示す第2インデックス情報とをAND演算し、前記単語が存在するファイルを検索する

処理を実行することを特徴とする検索方法。

【請求項9】

検索対象の単語を受け付けると、受け付けた単語に対する、ファイルにおける存否を示すインデックス情報であってハッシュ化されたインデックス情報を復元する復元部と、

復元した第1インデックス情報と、ファイルの存否を示す第2インデックス情報とをAND演算し、前記単語が存在するファイルを検索する検索部と、

を有することを特徴とする検索装置。

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、符号化プログラムなどに関する。

【背景技術】

【0002】

複数のファイルの圧縮を行う際には、複数のファイルのいずれが所定の文字情報を含むかを示すインデックス情報が生成される技術が知られている（例えば、特許文献1参照）。インデックス情報は、複数のファイルのいずれが検索対象の文字情報を含むか否かを示すインデックスとして用いられる。文字情報とは、例えば、1グラムの文字コードが接続する文字列のことをいう。

10

【0003】

一方、単語に対応付けられたポインタテーブル型のインデックス情報を生成する技術が知られている（例えば、非特許文献1参照）。かかる技術について、図1を参照して説明する。図1は、ポインタテーブル型インデックスの生成処理の参考例を示す図である。図1に示すように、かかる技術では、文書ファイルごとに単語を抽出し、文書IDと単語IDとその出現位置に対応付けられたインデックス情報を生成し、それを収集し、単語IDについてソートする。これにより、単語IDをもとに文書IDと出現位置とを対応づけた転置インデックスに転換する。

20

【先行技術文献】

【特許文献】

【0004】

【特許文献1】国際公開第2013/038527号

【特許文献2】特開平10-261969号公報

【特許文献3】特開平8-30633号公報

【特許文献4】特開平10-240754号公報

【非特許文献】

【0005】

【非特許文献1】西田圭介著「Googleを支える技術」技術評論社、2008年4月25日

30

【非特許文献2】関口宏司著「ApacheLucene入門」技術評論社、2006年6月25日

【発明の概要】

【発明が解決しようとする課題】

【0006】

しかしながら、ファイルの更新に応じて、複数のファイルのいずれが所定の単語を含むかを示すインデックス情報を容易に更新できないという問題がある。

【0007】

例えば、複数のファイルのいずれが所定の文字情報を含むかを示すインデックス情報が生成される技術では、文字情報についてのインデックス情報であり、そもそも単語についてのインデックス情報ではない。また、インデックスの基本部分は圧縮されているものの、ファイルの更新に応じて追加される更新部分は圧縮されておらず、インデックスのサイズの拡大に対して、領域のメンテナンスが必要である。したがって、かかる技術では、ファイルの更新に応じて、複数のファイルのいずれが所定の単語を含むかを示すインデックス情報を容易に更新できない。

40

【0008】

一方、単語に対応付けられたポインタテーブル型のインデックス情報を生成する従来技術では、文書ファイルごとに含まれる単語が異なるので、文書ファイルが更新されると、新語や未知語が含まれることがあり、再度、インデックスの生成処理、収集処理、ソート処理および転置処理が繰り返される。したがって、かかる従来技術では、複数の文書ファイルに含まれる単語の単語IDをもとにしたポインタテーブル型インデックスを容易に更

50

新できない。

【0009】

1つの側面では、ファイルの更新に応じて、複数のファイルに関するインデックス情報をサイズを変更せずに容易に更新することを目的とする。

【課題を解決するための手段】

【0010】

第1の案では、符号化プログラムは、コンピュータに、符号化対象のファイルを、静的辞書および動的辞書を用いて符号化し、前記静的辞書および動的辞書に登録された単語の、前記ファイルにおける存否を示す、ファイル軸および単語軸を有するインデックス情報を、前記ファイル軸方向および前記単語軸方向のそれぞれを、それぞれ所定の底を用いて折り返すことによりハッシュ化することにより生成し、前記ファイルが更新された際に、更新後のファイルに対応したインデックス情報の、更新前のインデックス情報に対する差分情報を、前記ファイル軸方向または前記単語軸方向について生成する、処理を実行させる。

10

【発明の効果】

【0011】

1つの態様によれば、ファイルの更新に応じて、複数のファイルに関するインデックス情報をサイズを変更せずに容易に更新することができる。

【図面の簡単な説明】

【0012】

20

【図1】図1は、ポインタテーブル型インデックスの生成処理の参考例を示す図である。

【図2】図2は、実施例1に係るビットマップ型インデックスの更新処理の一例を示す図である。

【図3】図3は、ファイル削除によるビットマップ型インデックスの更新処理の一例を示す図である。

【図4】図4は、実施例1に係るビットマップ型インデックスの一例を示す図である。

【図5】図5は、実施例1に係るビットマップ型インデックスの生成処理の一例を示す図である。

【図6】図6は、実施例1に係る動的辞書の一例を示す図である。

【図7】図7は、実施例1に係るビットマップ型インデックスのY軸方向のハッシュ化の一例を示す図である。

30

【図8】図8は、実施例1に係る検索処理の一例を示す図である。

【図9】図9は、実施例1に係るハッシュ化ビットマップ復元処理の一例を示す図である。

。

【図10】図10は、実施例1に係る符号化装置の構成を示す機能ブロック図である。

【図11】図11は、実施例1に係るインデックス生成処理のフローチャートの一例を示す図である。

【図12】図12は、実施例1に係る符号化処理のフローチャートの一例を示す図である。

。

【図13】図13は、実施例1に係るインデックス更新処理のフローチャートの一例を示す図である。

40

【図14】図14は、実施例1に係る検索装置の構成を示す機能ブロック図である。

【図15】図15は、実施例1に係る検索処理のフローチャートの一例を示す図である。

【図16】図16は、実施例2に係るビットマップ型インデックスの階層化の一例を示す図である。

【図17】図17は、実施例2に係る削除ビットマップの階層化の一例を示す図である。

【図18】図18は、実施例2に係るビットマップ型インデックスの更新処理の一例を示す図である。

【図19】図19は、階層化されたセグメント群を用いた検索処理の一例を示す図である。

。

50

【図 2 0】図 2 0 は、階層化されたセグメント群を実装したコンピュータシステムの構成例を示す図である。

【図 2 1】図 2 1 は、コンピュータのハードウェア構成例を示す図である。

【図 2 2】図 2 2 は、コンピュータで動作するプログラムの構成例を示す図である。

【図 2 3】図 2 3 は、実施形態のシステムにおける装置の構成例を示す図である。

【発明を実施するための形態】

【0 0 1 3】

以下に、本願の開示する符号化プログラム、符号化方法、符号化装置、検索プログラム、検索方法および検索装置の実施例を図面に基づいて詳細に説明する。なお、この実施例によりこの発明が限定されるものではない。

【実施例 1】

【0 0 1 4】

〔実施例 1 に係るビットマップ型インデックスの更新処理の一例〕

図 2 は、実施例 1 に係るビットマップ型インデックスの更新処理の一例を示す図である。図 2 上図に示すように、ビットマップ型インデックス B I におけるインデックス生成処理は、ビットマップ型インデックス B I のサイズ縮小を実現するために、ビットマップ型インデックス B I からハッシュ化インデックスを生成する。すなわち、インデックス生成処理は、隣接した複数のハッシュ値（底）を基に、2 次元（単語の軸とファイルの軸）にハッシュ化を適用したハッシュ化インデックスを生成する。そして、ファイルが更新されると、インデックス更新処理は、更新後のファイルに対応したインデックス情報の、更新前のインデックス情報に対する差分情報を単語の軸方向またはファイルの軸方向について生成する。なお、ここでいう「更新」とは、ファイル内の単語を変更すること、ファイルを追加することおよびファイルを削除することを含む。単語の軸をハッシュ化する際に用いられる底、ファイルの軸をハッシュ化する際に用いられる底は、共通であっても良いし、共通でなくても良い。

【0 0 1 5】

ここでいうビットマップ型インデックス B I とは、全文検索のためのインデックスであり、単語を指定するポインタと単語の各対象ファイルでの存否を示すビットを連結したビット列である。すなわち、ビットマップ型インデックス B I とは、ファイルに含まれる単語について、ファイルごとの存否をインデックス化したビットマップのことをいう。検索処理時には、このビットマップを、ビットの O N ・ O F F に応じて検索対象の単語を含むか否かを示すインデックスとして用いることができる。単語を指定するポインタとしては、例えば、単語の符号が採用される。単語の符号は、静的コードおよび単語コードのことをいい、単語 I D（identification）と同義である。なお、単語を指定するポインタは、例えば、単語そのものを用いても良い。すなわち、ビットマップ型インデックス B I は、単語を指定するポインタで示される単語ごとのビットマップを纏めたものである。図 2 に示すように、ビットマップ型インデックス B I の X 軸はファイル I D（Identification）を表し、ビットマップ型インデックス B I の Y 軸は単語 I D を表す。すなわち、ビットマップ型インデックス B I は、単語 I D が示す単語の、複数のファイル I D が示すファイルごとの存否を表す。

【0 0 1 6】

一例として、ファイルの軸（X 軸）方向のビットマップのハッシュ化について説明する。ある単語 I D に対応するビットマップ b 1 が示されている。ある単語 I D が示す単語がファイルに存在する場合には、当該単語のファイルの存否として 2 進数の「1」が設定され、当該単語がファイルに存在しない場合には、当該単語のファイルの存否として 2 進数の「0」が設定される。なお、ビットマップ型インデックス B I の詳細の説明は、後述する。

【0 0 1 7】

例えば、図 2 上図に示すように、インデックス生成処理は、単語 I D に対応するビットマップそれぞれについてハッシュ関数を適用した複数のハッシュ化ビットマップを生成す

10

20

30

40

50

る。ここでは、インデックス生成処理は、32ビットレジスタを想定し、一例として29と31のハッシュ値（底）を基に、各ビットマップをハッシュ化する。具体的には、インデックス生成処理は、1つの底のハッシュ化ビットマップについて、単語IDに対応するビットマップの各ビットの位置を底で割った余りの位置に、当該ビットマップの各ビットの値を設定する。ビットマップの各ビットの位置は、それぞれのファイルIDに対応する。一例として、インデックス生成処理は、底29のハッシュ化ビットマップh11について、ビットマップb1の各ビットの位置を底29で割った余りの位置に、ビットマップb1の各ビットの値を設定する。ビットマップb1の35ビット目の位置のビット値「1」は、ハッシュ化ビットマップh11の6ビット目に設定される。ビットマップb1の42ビット目の位置のビット値「1」は、ハッシュ化ビットマップh11の13ビット目に設定される。インデックス生成処理は、底31のハッシュ化ビットマップh12について、ビットマップb1の各ビットの位置を底31で割った余りの位置に、ビットマップb1の各ビットの値を設定する。ビットマップb1の35ビット目の位置のビット値「1」は、ハッシュ化ビットマップh12の4ビット目に設定される。ビットマップb1の42ビット目の位置のビット値「1」は、ハッシュ化ビットマップh12の11ビット目に設定される。すなわち、インデックス生成処理は、ビットマップの0ビット目からの各ビットを順番にハッシュ化ビットマップの0ビット目から設定し、（底－1）ビット目まで設定する。そして、インデックス生成装置は、再度折り返してハッシュ化ビットマップの0ビット目から既にハッシュ化ビットマップに設定された値とOR演算した値を設定する。この結果、複数の単語が示す各単語IDについてのビットマップにおいて、ファイルID1～mまでのインデックス情報が格納される。なお、ハッシュ値（底）は、以降、29と31であるとして説明する。

【0018】

図2下図は、図2上図の状態から新たなファイルm+1が追加された状態を示している。ビットマップ型インデックスBIには、複数の単語が示す単語IDごとにファイルm+1についてのビットが設定される。

【0019】

一例として、図2上図で示したある単語IDに対応するビットマップb1'が示されている。ある単語IDが示す単語がファイルに存在する場合には、当該単語のファイルの存否として2進数の「1」が設定され、当該単語がファイルに存在しない場合には、当該単語のファイルの存否として2進数の「0」が設定される。ここでは、ビットマップb1'の44ビット目に、当該単語がファイルに存在するとして「1」が設定されるとする。

【0020】

例えば、図2下図に示すように、インデックス更新処理は、単語IDに対応するビットマップの更新ビットについてハッシュ関数を適用した複数のハッシュ化ビットマップを更新する。具体的には、インデックス更新処理は、1つの底のハッシュ化ビットマップについて、単語IDに対応するビットマップの更新ビットの位置を底で割った余りの位置に、更新ビットの値と既にハッシュ化ビットマップに設定された値とをOR演算した値に更新する。一例として、インデックス更新処理は、底29のハッシュ化ビットマップh11'について、ビットマップb1'の更新ビットの位置を底29で割った余りの位置に、更新ビットの値「1」と既に設定された値「0」とをOR演算した値「1」を更新する。ビットマップb1'の44ビット目の位置のビット値「1」は、ハッシュ化ビットマップh11'の15ビット目に設定される。インデックス更新処理は、底31のハッシュ化ビットマップh12'について、ビットマップb1'の更新ビットの位置を底31で割った余りの位置に、更新ビットの値「1」と既に設定された値「0」とをOR演算した値「1」を更新する。ビットマップb1'の44ビット目の位置のビット値「1」は、ハッシュ化ビットマップh12'の13ビット目に設定される。この結果、ファイルが追加されても、インデックス更新処理は、追加されたファイルにおける単語の存否情報を対象にして、ハッシュ化ビットマップを更新するので、ビットマップの差分だけを更新できる。

【0021】

[ファイル削除によるビットマップ型インデックスの更新処理の一例]

図3は、ファイル削除によるビットマップ型インデックスの更新処理の一例を示す図である。図3に示すように、ビットマップ型インデックスBIにおけるインデックス生成処理は、ビットマップ型インデックスBIのサイズ縮小を実現するために、削除ビットマップdbからハッシュ化ビットマップhdbを生成する。

【0022】

ここでいう削除ビットマップdbとは、対象ファイルの存在または削除をビット列で示したインデックスである。なお、削除ビットマップdbにおいてON(=1)に該当するビットに対応するファイルIDの対象ファイルは、存在していることを意味する。一方、OFF(=0)になると、当該対象ファイルは削除されたことを意味する。これにより、

10

【0023】

一例として、削除ビットマップdbが示されている。あるファイルが存在する場合には、当該ファイルの存否として2進数の「1」が設定され、存在しない場合には、当該ファイルの存否として2進数の「0」が設定される。

【0024】

例えば、図3上図に示すように、インデックス生成処理は、削除ビットマップdbについてハッシュ関数を適用した複数のハッシュ化ビットマップを生成する。具体的には、インデックス生成処理は、1つの底のハッシュ化ビットマップについて、削除ビットマップdbの各ビットの位置を底で割った余りの位置に、当該削除ビットマップdbの各ビットの値を設定する。一例として、インデックス生成処理は、底29のハッシュ化ビットマップhdb1について、削除ビットマップdbの各ビットの位置を底29で割った余りの位置に、削除ビットマップdbの各ビットの値を設定する。削除ビットマップdbの5ビット目の位置のビット値「1」は、ハッシュ化ビットマップhdb1の5ビット目に設定される。インデックス生成処理は、底31のハッシュ化ビットマップhdb2について、削除ビットマップdbの各ビットの位置を底31で割った余りの位置に、削除ビットマップdbの各ビットの値を設定する。削除ビットマップdbの5ビット目の位置のビット値「1」は、ハッシュ化ビットマップhdb2の5ビット目に設定される。すなわち、インデックス生成処理は、削除ビットマップの0ビット目からの各ビットを順番にハッシュ化ビットマップの0ビット目から設定し、(底-1)ビット目まで設定する。そして、インデックス生成装置は、再度折り返してハッシュ化ビットマップの0ビット目から既にハッシュ化ビットマップに設定された値とOR演算した値を設定する。この結果、削除ビットマップdbにおいて、ファイルID1~mまでのインデックス情報が格納される。

20

30

【0025】

図3下図は、図3上図の状態からファイル6が削除された状態を示している。削除ビットマップdb'には、ファイル6についてのビットに2進数の「0」が設定される。

【0026】

例えば、図3下図に示すように、インデックス更新処理は、削除ビットマップdb'の更新ビットについてハッシュ関数を適用した複数のハッシュ化ビットマップを更新する。具体的には、インデックス更新処理は、1つの底のハッシュ化ビットマップについて、削除ビットマップdb'の更新ビットの位置を底で割った余りの位置に、更新ビットの値「0」と既にハッシュ化ビットマップhdb1'に設定された値とをAND演算した値に更新する。一例として、インデックス更新処理は、底29のハッシュ化ビットマップhdb1'について、削除ビットマップdb'の更新ビットの位置を底29で割った余りの位置に、更新ビットの値「0」と既に設定された値「1」とをAND演算した値「0」を更新する。削除ビットマップdb'の5ビット目の位置のビット値「0」は、ハッシュ化ビットマップhdb1'の5ビット目に設定される。同様に、インデックス更新処理は、底31のハッシュ化ビットマップhdb2'について、削除ビットマップdb'の更新ビット

40

50

の位置を底 3 1 で割った余りの位置に、更新ビットの値「0」と既に設定された値「1」とをAND演算した値「0」を更新する。削除ビットマップd b'の5ビット目の位置のビット値「0」は、ハッシュ化ビットマップh d b 2'の5ビット目に設定される。

【0027】

[ビットマップ型インデックスの一例]

次に、実施例1に係るビットマップ型インデックスBIの一例を、図4を参照して説明する。図4は、実施例1に係るビットマップ型インデックスの一例を示す図である。図4に示すように、ビットマップ型インデックスBIは、超高頻度の単語、高頻度の単語および低頻度の単語に係る圧縮付号(単語IDに対応)ごとにビットマップを対応付ける。ビットマップとは、超高頻度の単語、高頻度の単語および低頻度の単語がいずれの圧縮ファイルに含まれるかを表す符号ビット列である。ビットマップの各ビットが、各圧縮ファイルに超高頻度の単語、高頻度の単語および低頻度の単語が含まれているか否かを表す。

10

【0028】

ビットマップ型インデックスBIは、例えば、32種類の超高頻度単語ごと、8K(8000)種類の高頻度の単語および16K(16000)種類の低頻度単語ごとにビットマップを対応付ける。超高頻度単語とは、出現頻度集計用のファイル群において各単語の出現頻度を集計した場合に、出現頻度の高い単語を表す。例えば、超高頻度単語は、頻度集計用のファイル群での出現頻度が上位32位までの単語である。高頻度単語は、頻度集計用のファイル群での出現頻度が上位8000位までの単語である。また、低頻度単語は、頻度集計用のファイル群での出現頻度の順位が24000位未満であって、符号化するファイルから抽出された数値文字列または単語である。単語の一例として、専門用語、新語および未知語が挙げられる。ここでいう専門用語とは、ある特定の学問の分野や業界等の間で通用する単語であり、符号化するファイルの中で繰り返し出現する特長がある単語のことをいう。新語とは、流行語等の新しく作られた単語であり、符号化するファイルの中で繰り返し出現する特長がある単語のことをいう。未知語とは、専門用語でなく、新語でない単語であり、符号化するファイルの中で繰り返し出現する特長がある単語のことをいう。

20

【0029】

例えば、ビットマップ型インデックスBIの有効行1行目は、圧縮符号が示す単語「the」のビットマップが「・・・1101」となっている。ビットマップ型インデックスBIの有効行1行目のビットマップは、「the」の圧縮符号が含まれるファイルを表す。ビットマップ「・・・1101」は、1ビット目に「1」が格納されているのでファイル1に「the」が含まれ、2ビット目に「0」が格納されているのでファイル2に「the」が含まれず、3ビット目に「1」が格納されているのでファイル3に「the」が含まれることを表す。また、ビットマップ「・・・1101」は、4ビット目に「1」が格納されているのでファイル4に「the」が含まれていることを表す。なお、ビットマップ「・・・1101」は、ファイル5以降の他の各ファイルに「the」が含まれるか否かについても表す。

30

【0030】

[実施例1に係るビットマップ型インデックスの生成処理]

40

ここで、実施例1に係るビットマップ型インデックスBIの生成処理を、図5を参照して説明する。図5は、実施例1に係るビットマップ型インデックスの生成処理の一例を示す図である。図5に示すように、まず、符号化処理は、特定のファイルに含まれる単語を静的辞書S0および動的辞書D0を用いて符号化する際に、静的辞書S0に登録されない単語のうち外部辞書E0に含まれる単語を抽出する。符号化処理は、抽出された単語の外部辞書E0における符号と、動的に割り当てられる動的符号とを対応付けて動的辞書D0に登録する。

【0031】

ここでいう静的辞書S0とは、一般的な英語辞典、国語辞典や教科書などを基にして、文書中に出現する単語の出現頻度を特定し、出現頻度のより高い単語に対して、より短い

50

符号を割り当てた辞書のことをいう。静的辞書 S 0 には、それぞれの単語に対応する符号である静的コードがあらかじめ登録されている。

【0032】

ここでの動的辞書 D 0 とは、静的辞書 S 0 に登録されず、且つ、外部辞書 E 0 に登録されている単語と、外部辞書 E 0 における符号と、動的に付された動的コードとを対応付けた辞書である。静的辞書 S 0 に登録されていない単語には、一例として、出現頻度の低い単語（低頻度単語）が挙げられる。かかる単語（低頻度単語）には、専門単語、新語、未知語などがある。なお、動的辞書 D 0 には、静的辞書 S 0 に登録されていない単語の出現順に、動的コードに対応付けられた単語がバッファ部に登録される。動的辞書 D 0 に関する詳しい説明は後述する。

10

【0033】

ここでの外部辞書 E 0 とは、静的辞書 S 0 に登録されていない単語を専門ごとに単語コードと対応付けた辞書である。外部辞書 E 0 には、一例として、専門辞書、新語辞書および未知語辞書などがある。専門辞書には、専門単語が記憶される。新語辞書には、新語が記憶される。未知語辞書には、未知語が記憶される。低頻度単語に付される動的コードは、同一の単語であっても符号化対象のファイルごとに異なるが、外部辞書 E 0 における単語コードは、同一の単語であれば符号化対象のファイルごとに共通化される。

【0034】

符号化処理の一例を、以下に説明する。例えば、符号化処理は、ファイル F 1 内の符号化対象のファイル f 3 を記憶領域にロードする。なお、ファイル f 3 の「3」は、ファイル I D が「3」であることを示す。

20

【0035】

符号化処理は、符号化対象のファイル f 3 を記憶領域から読み出し、読み出したファイル f 3 に対して字句解析を行う。ここでの字句解析とは、符号化されていない状態のファイルを単語に分割することをいう。

【0036】

符号化処理は、静的辞書 S 0 と単語の文字列とを比較して、静的辞書 S 0 に単語の文字列に対応する符号があるか否かを判定する。なお、かかる判定処理は、静的辞書 S 0 を用いて符号化可能な単語の文字列を特定するビットフィルタと単語の文字列とを比較して、単語の文字列がビットフィルタにヒットするか否かを判定しても良い。ここでのビットフィルタとは、静的辞書 S 0 を用いて符号化可能な単語の文字列を特定するフィルタのことをいう。符号化処理は、静的辞書 S 0 に単語の文字列に対応する符号がある場合には、静的辞書 S 0 に基づいて単語の文字列を、当該単語の文字列に対応する符号（静的コード）に符号化し、静的コードを出力する。

30

【0037】

符号化処理は、静的辞書 S 0 に単語の文字列に対応する符号がない場合には、外部辞書 E 0 から、単語の文字列に対応する符号（単語コード）および当該単語が登録された外部辞書 E 0 の I D を取得する。なお、符号化処理は、外部辞書 E 0 における符号を、自装置にて割り当てても良いし、単語コードを割り当てるマスタ装置に問い合わせても良い。実施例では、符号化処理は、外部辞書 E 0 における符号を、自装置にて割り当てる場合とする。

40

【0038】

符号化処理は、単語の文字列と、外部辞書 E 0 における符号（単語コード）と、動的に付された動的コードとを動的辞書 D 0 に登録する。符号化処理は、動的辞書 D 0 に基づいて、登録した単語の文字列を、当該登録した単語の文字列に対応する単語コードに符号化し、単語コードを出力する。なお、符号化処理は、単語の文字列が動的辞書 D 0 に登録されている場合には、動的辞書 D 0 に基づいて、当該単語の文字列を、当該単語の文字列に対応する単語コードに符号化し、単語コードを出力すれば良い。

【0039】

符号化処理は、静的コードおよび動的辞書 D 0 に基づいた単語コードそれぞれを、単語

50

の出現順に、符号化データc 3のエリアに出力する。符号化データc 3のエリアは、ファイルf 3に対応する符号化ファイルF 1内の符号化データのエリアである。

【0040】

符号化対象のファイルf 3の符号化処理が完了すると、インデックス生成処理は、複数の単語コードそれぞれについて、ファイルf 3内の存否をビットマップ型インデックスB Iに設定する。

【0041】

一例として、「M i c k e y」の単語ID「A 0 0 2 h」に対応するビットマップが示されている。単語ID「A 0 0 2 h」が示す単語「M i c k e y」がファイルに存在する場合には、当該単語のファイルの存否としてON、すなわち2進数の「1」が設定され、当該単語がファイルに存在しない場合には、当該単語のファイルの存否としてOFF、すなわち2進数の「0」が設定される。ここでは、ファイルf 3には、単語「M i c k e y」が存在するので、「M i c k e y」の単語ID「A 0 0 2 h」に対応するビットマップには、ファイルID「3」に対応するビットに「1」が設定される。

10

【0042】

そして、インデックス生成処理は、ビットマップ型インデックスB Iからハッシュ化インデックスH Iを生成する。すなわち、インデックス生成処理は、隣接した複数のハッシュ値（底）を基に、例えばファイルの軸（X軸）にハッシュ化を適用したハッシュ化インデックスを生成する。なお、ハッシュ化インデックスを生成する方法は、図2上図の説明と同様であるので、その説明を省略する。

20

【0043】

[動的辞書の一例]

図6は、実施例1に係る動的辞書の一例を示す図である。図6に示される動的辞書D 0は、バッファ部D 1とアドレステーブルD 2とを含む。バッファ部D 1は、文字列を記憶する。アドレステーブルD 2は、動的コードと、格納位置と、データ長と、外部辞書IDと、単語コードとを対応付けて保持する。動的コードは、あらかじめ定められた固定長のコードであり、単語の文字列が登録された順に割り当てられる。ここでは、動的コードは、16進数「F」で始まる固定長3バイトのコードである。格納位置は、バッファ部D 1に格納された文字列の位置を示す。データ長は、バッファ部D 1に格納された文字列の長さ（バイト長）を示す。外部辞書IDは、単語が登録されている外部辞書E 0のIDを示す。単語コードは、単語に割り当てられたコードであり、例えば圧縮符号である。かかる単語コードは、動的辞書D 0に登録された単語であっても、同じ単語であれば、複数のファイルで共通して用いられる。

30

【0044】

例えば、単語の文字列「M i c k e y」に動的コードが割り当てられる場合について説明する。符号化処理は、単語の文字列「M i c k e y」をバッファ部D 1に格納する。符号化処理は、単語の文字列を格納した格納位置および格納したデータ長をアドレステーブルD 2に登録する。さらに、符号化処理は、単語の文字列「M i c k e y」に割り当てられた単語コードおよび外部辞書IDをアドレステーブルD 2に登録する。ここでは、符号化処理は、格納位置として「16」、データ長として「6」、外部辞書IDとして「53」、単語コードとして「C D 0 0 1 0 h」をアドレステーブルD 2に登録する。

40

【0045】

符号化処理は、単語の文字列を、当該単語の文字列に対応付けられたアドレステーブルD 2の単語コードに符号化する。ここでは、符号化処理は、単語の文字列「M i c k e y」を当該単語の文字列に対応付けられた動的コード「A 0 0 2 h」に符号化する。

【0046】

[Y軸のハッシュ化の一例]

ここで、ビットマップ型インデックスB Iの単語IDの軸（Y軸）方向のビットマップのハッシュ化について、図7を参照して説明する。図7は、実施例1に係るビットマップ型インデックスのY軸方向のハッシュ化の一例を示す図である。図7に示すように、イン

50

デックス生成処理は、ファイルIDに対応するビットマップにハッシュ関数を適用した複数のハッシュ化ビットマップを生成する。ここでは、インデックス生成処理は、出現頻度に応じて、Y軸を3分割し、3分割したそれぞれのビットマップを、異なる底の組合せでハッシュ化する。一例として、インデックス生成処理は、超高頻度単語では、 α_{y1} と β_{y1} のハッシュ値（底）を基に、Y軸方向の超高頻度単語に対応するビットマップをハッシュ化する。インデックス生成処理は、高頻度単語では、 α_{y2} と β_{y2} のハッシュ値（底）を基に、Y軸方向の高頻度単語に対応するビットマップをハッシュ化する。インデックス生成処理は、低頻度単語では、 α_{y3} と β_{y3} のハッシュ値（底）を基に、Y軸方向の低頻度単語に対応するビットマップをハッシュ化する。

【0047】

10

そして、インデックス更新処理は、ファイルIDに対応するビットマップの更新ビットについてハッシュ関数を適用した複数のハッシュ化ビットマップを更新する。例えば、ファイルに低頻度単語が追加されると、インデックス更新処理は、ファイルのIDに対応するビットマップのうち低頻度単語のビットマップの更新ビットの値を、底 α_{y3} のハッシュ化ビットマップおよび底 β_{y3} のハッシュ化ビットマップに更新する。すなわち、インデックス更新処理は、ハッシュ化ビットマップ $h21'$ について、ビットマップ $b2'$ のうち低頻度単語に対応するビットマップの更新ビットの位置を底 α_{y3} で割った余りの位置に、更新ビットの値「1」と既に設定された値「0」とをOR演算した値「1」を更新する。インデックス更新処理は、底 β_{y3} のハッシュ化ビットマップ $h22'$ について、ビットマップ $b2'$ のうち低頻度単語のビットマップの更新ビットの位置を底 β_{y3} で割った余りの位置に、更新ビットの値「1」と既に設定された値「0」とをOR演算した値「1」を更新する。この結果、ファイルに未知語が追加されても、インデックス更新処理は、追加された未知語の存否情報を対象にして、ハッシュ化ビットマップを更新するので、ビットマップの差分だけを更新できる。

20

【0048】

[実施例に係る検索処理の一例]

図8は、実施例1に係る検索処理の一例を示す図である。図8に示すように、検索処理は、検索単語の入力を受け付けると、受け付けた検索単語が示す単語IDに対する複数のハッシュ化ビットマップをハッシュ化インデックスHIから抽出する。ここでは、検索単語として「Sherlock」と「Mickey」が入力された場合とする。すると、検索処理は、検索単語として受け付けられた「Sherlock」が示す単語ID「A000h」に対する複数のハッシュ化ビットマップ $h5$ をハッシュ化インデックスHIから抽出する。複数のハッシュ化ビットマップ $h5$ には、底29のハッシュ化ビットマップ $h51$ と底31のハッシュ化ビットマップ $h52$ とが含まれる。また、検索処理は、検索単語として受け付けられた「Mickey」が示す単語ID「A002h」に対する複数のハッシュ化ビットマップ $h4$ をハッシュ化インデックスHIから抽出する。複数のハッシュ化ビットマップ $h4$ には、底29のハッシュ化ビットマップ $h41$ と底31のハッシュ化ビットマップ $h42$ とが含まれる。

30

【0049】

検索処理は、抽出された単語IDに対する複数のハッシュ化ビットマップおよびハッシュ化された削除ビットマップを復元する。なお、複数のハッシュ化ビットマップの復元処理は、後述する。復元結果は、単語IDに対応するビットマップで表わされる。ここでは、検索処理は、単語ID「A002_h」に対する複数のハッシュ化ビットマップ $h4$ を復元し、ビットマップ $b4$ を復元結果として出力する。検索処理は、単語ID「A000h」に対する複数のハッシュ化ビットマップ $h5$ を復元し、ビットマップ $b5$ を復元結果として出力する。検索処理は、ハッシュ化された削除ビットマップ hdb を復元し、削除ビットマップ db を復元結果として出力する。

40

【0050】

また、検索処理は、単語ID「A002h」のビットマップ $b4$ と、単語ID「A000h」のビットマップ $b5$ と、反転された削除ビットマップ db と、のAND演算を行う

50

。検索処理は、AND結果のビットがON（「1」）であるファイルIDを検索結果として出力する。ここでは、ファイルIDが3であるファイル3が検索結果として出力される。これにより、検索処理は、検索対象のハッシュ化ビットマップを復元し、復元されたビットマップだけを用いることで、検索キーが示す単語が、どのファイルに存在するかを、高速に絞り込むことができる。

【0051】

[実施例1に係るハッシュ化ビットマップ復元処理の一例]

図9は、実施例1に係るハッシュ化ビットマップ復元処理の一例を示す図である。図9に示すように、ハッシュ化ビットマップ復元処理は、単語IDに対応するハッシュ化ビットマップから、ハッシュ化を展開したビットマップへ復元する。ハッシュ化ビットマップ復元処理は、検索単語を構成する単語がどのファイルに存在するかを検索する際に実行される。

10

【0052】

検索処理は、ハッシュ化ビットマップh4（図8参照）の複数のハッシュ化ビットマップh41、h42をそれぞれビットマップに展開する（第1の復元処理）。ここでは、検索処理は、1つの底のハッシュ化ビットマップの復元先のビットマップについて、底に整数（0～）を乗算して得られた値にハッシュ化ビットマップの各ビットの位置を加算した位置に、ハッシュ化ビットマップの各ビットの値を設定する。一例として、検索処理は、底29のハッシュ化ビットマップh41の復元先のビットマップb41について、底29に「0」を乗算した値にハッシュ化ビットマップh41の各ビットの位置を加算した位置に、ハッシュ化ビットマップh41の各ビットの値を設定する。検索処理は、底29のハッシュ化ビットマップh41の復元先のビットマップb41について、底29に「1」を乗算した値にハッシュ化ビットマップh41の各ビットの位置を加算した位置に、ハッシュ化ビットマップh41の各ビットの値を設定する。検索処理は、復元先の底29のビットマップb41の最大ビットの位置のビットの値が設定されるまで繰り返す。同様に、検索処理は、底31のハッシュ化ビットマップh42の復元先のビットマップb42について、底31に「0」を乗算した値にハッシュ化ビットマップh42の各ビットの位置を加算した位置に、ハッシュ化ビットマップh42の各ビットの値を設定する。検索処理は、底31のハッシュ化ビットマップh42の復元先のビットマップb42について、底31に「1」を乗算した値にハッシュ化ビットマップh42の各ビットの位置を加算した位置に、ハッシュ化ビットマップh42の各ビットの値を設定する。検索処理は、復元先のビットマップb42の最大ビットの位置のビットの値が設定されるまで繰り返す。

20

30

【0053】

検索処理は、第1の復元処理で復元されたそれぞれのビットマップの対応する位置のビットをAND演算する（第2の復元処理）。ここでは、検索処理は、底29のハッシュ化ビットマップh41から復元されたビットマップb41と、底31のハッシュ化ビットマップh42から復元されたビットマップb42とをAND演算する。検索処理は、AND結果のビットマップb4を復元結果として出力する。

【0054】

[実施例1に係る符号化装置の構成]

40

次に、図10を参照して、実施例1に係る符号化装置100の構成について説明する。図10は、実施例1に係る符号化装置の構成を示す機能ブロック図である。図10に示すように、符号化装置100は、制御部110と記憶部120とを有する。

【0055】

制御部110は、図2、図3および図5に示したインデックス生成処理、符号化処理およびインデックス更新処理を実行する処理部である。制御部110は、各種の処理手順を規定したプログラムや制御データを格納するための内部メモリを有し、これらによって種々の処理を実行する。そして、制御部110は、例えば、ASICやFPGAなどの集積回路の電子回路に対応する。または、制御部110は、CPUやMPUなどの電子回路に対応する。また、制御部110は、ファイルリード部111、符号化部112、インデッ

50

クス生成部 113 およびインデックス更新部 114 を有する。

【0056】

記憶部 120 は、例えばフラッシュメモリや FRAM（登録商標）などの不揮発性の半導体メモリ素子などの記憶装置に対応する。記憶部 120 は、静的辞書 121、動的辞書 122、外部辞書 123 およびハッシュ化インデックス 124 を有する。なおハッシュ化インデックス 124 の構成は、各単語 ID に対応するハッシュ化ビットマップを纏めた構成（図 5 参照）と同様であるので、その説明を省略する。

【0057】

静的辞書 121 は、一般的な英語辞典、国語辞典や教科書などを基にして、文書中出现する単語の出現頻度を特定し、出現頻度のより高い単語に対して、より短い符号を割り

10

【0058】

動的辞書 122 は、静的辞書 121 に登録されていない単語と動的に付された動的コードおよび単語コードとを対応付けた辞書である。なお、動的辞書 122 は、図 5 および図 6 の動的辞書 D0 に対応する。動的辞書 122 の説明は、図 6 と同様であるので、その説明を省略する。

【0059】

外部辞書 123 は、静的辞書 121 に登録されていない単語を専門ごとに単語コードと対応付けた辞書である。外部辞書 123 には、例えば、専門辞書、新語辞書および未知語辞書が含まれる。一例として、外部辞書 123 は、辞書ごとに、単語の文字列および単語

20

【0060】

ファイルリード部 111 は、符号化対象のファイル F1 を記憶領域に読み出す。ファイルリード部 111 は、ファイル F1 を記憶領域から読み出し、読み出したファイル F1 に対して字句解析を行う。ファイルリード部 111 は、字句解析した結果の各単語を順次符号化部 112 に出力する。

【0061】

符号化部 112 は、ファイルリード部 111 から出力された単語を符号化する。

【0062】

例えば、符号化部 112 は、ファイルリード部 111 から出力された対象の単語の文字列が静的辞書 121 に登録されているか否かを判定する。一例として、符号化部 112 は、対象の単語の文字列が静的辞書 121 のビットフィルタにヒットするか否かを判定する。符号化部 112 は、対象の単語の文字列が静的辞書 121 に登録されている場合には、当該単語の文字列を静的辞書 121 に基づいて符号化する。一例として、符号化部 112 は、静的辞書 121 に基づいて、単語の文字列を、当該単語の文字列に対応する静的コード（単語コード）に符号化する。符号化部 112 は、符号化された単語コードをインデックス生成部 113 に出力する。

30

【0063】

そして、符号化部 112 は、対象の単語の文字列が静的辞書 121 に登録されていない場合には、当該単語の文字列を動的辞書 122 に基づいて符号化する。一例として、符号化部 112 は、単語の文字列が動的辞書 122 のバッファ部 D1 に既に格納されているか否かを判定する。符号化部 112 は、単語の文字列が動的辞書 122 のバッファ部 D1 に既に格納されていない場合には、当該単語の文字列に対応する単語コードおよび当該単語が登録された辞書の ID を外部辞書 123 から取得する。符号化部 112 は、単語コードおよび単語が登録された辞書の ID を、アドレステーブル D2 の新たな動的コードに対応するレコードに格納する。加えて、符号化部 112 は、単語の文字列をバッファ部 D1 に格納するとともに、アドレステーブル D2 の新たな動的コードに対応するレコードに、当該単語の文字列を格納した格納位置および格納したデータ長を格納する。符号化部 112 は、単語の文字列を、当該単語の文字列に対応付けられた、アドレステーブル D2 の単語

40

50

コードに符号化する。符号化部 1 1 2 は、符号化された単語コードをインデックス生成部 1 1 3 に出力する。

【0 0 6 4】

インデックス生成部 1 1 3 は、それぞれ符号化された単語コード群に基づいて、ビットマップを生成する。例えば、インデックス生成部 1 1 3 は、符号化部 1 1 2 から出力された単語コードに対応するビットマップの、ファイル F 1 に対応するビットをハッシュ化し「1」を設定する。インデックス生成部 1 1 3 は、ファイル F 1 に未処理の単語がある場合には、次の単語の符号化処理を符号化部 1 1 2 に実行させる。インデックス生成部 1 1 3 は、ファイル F 1 に未処理の単語がない場合には、他の符号化対象のファイルがあれば、他の符号化対象のファイルの符号化処理を符号化部 1 1 2 に実行させる。

10

【0 0 6 5】

また、インデックス生成部 1 1 3 は、複数のハッシュ値（底）を基に、ハッシュ化インデックス 1 2 4 を生成する。例えば、インデックス生成部 1 1 3 は、複数の単語 I D（単語コード）に対応するビットマップそれぞれについて、複数のハッシュ化ビットマップを生成する。一例として、インデックス生成部 1 1 3 は、複数の単語 I D に対応するビットマップそれぞれについて、2つのハッシュ値（底）に基づいたそれぞれのハッシュ化ビットマップを生成する。すなわち、インデックス生成部 1 1 3 は、ビットマップの 0 ビット目からの各ビットを順番にハッシュ化ビットマップの 0 ビット目から設定し、（底 - 1）ビット目まで設定する。そして、インデックス生成部 1 1 3 は、再度折り返してハッシュ化ビットマップの 0 ビット目から既にハッシュ化ビットマップに設定された値と OR 演算した値を設定する。そして、インデックス生成部 1 1 3 は、全ての単語 I D に対応するビットマップに対してハッシュ化ビットマップを生成すると、生成されたハッシュ化ビットマップを纏めたハッシュ化インデックス 1 2 4 を記憶部 1 2 0 に格納する。

20

【0 0 6 6】

インデックス更新部 1 1 4 は、ファイルが追加された場合には、追加されたファイルのファイル I D に対応する、ビットマップの追加ビットについて、ハッシュ化ビットマップを更新する。

【0 0 6 7】

例えば、インデックス更新部 1 1 4 は、ハッシュ化インデックス 1 2 4 において、追加ファイルのファイル I D に関係するビットを設定する。一例として、インデックス更新部 1 1 4 は、削除ビットマップについては、追加ファイルのファイル I D に対して「0」（OFF）のビットを設定し、削除ファイルについては、削除ファイル I D に対して「1」（ON）のビットを設定する。また、インデックス更新部 1 1 4 は、追加ファイルに含まれる単語の単語コード（単語 I D）に対応するビットマップの、当該追加ファイルのファイル I D に対応するビットを「1」に更新する。インデックス更新部 1 1 4 は、単語 I D に対応するビットマップの更新ビットについて、2つのハッシュ値（底）に基づいたそれぞれのハッシュ化ビットマップを更新する。一例として、底が α 、 β であるとする。すると、インデックス更新部 1 1 4 は、底 α のハッシュ化ビットマップについて、更新ビットの位置を α で割った余りの位置に、更新ビットの値「1」と既に設定された値「0」とを OR 演算した値「1」を更新する。インデックス更新部 1 1 4 は、底 β のハッシュ化ビットマップについて、更新ビットの位置を β で割った余りの位置に、更新ビットの値「1」と既に設定された値「0」とを OR 演算した値「1」を更新する。同様に、インデックス更新部 1 1 4 は、削除ビットマップの更新ビットについて、2つのハッシュ値（底）に基づいたそれぞれのハッシュ化ビットマップを更新する。これにより、インデックス更新部 1 1 4 は、ファイルの追加に応じて、複数のファイルのいずれが所定の単語を含むかを示すハッシュ化インデックス 1 2 4 に差分だけを更新できる。

30

40

【0 0 6 8】

また、インデックス更新部 1 1 4 は、ファイルが削除された場合には、削除されたファイルのファイル I D に対応する、ハッシュ化ビットマップを更新する。例えば、インデックス更新部 1 1 4 は、削除ビットマップについて、削除ファイルのファイル I D に対して

50

「1」のビットを更新する。そして、インデックス更新部114は、削除ビットマップの更新ビットについて、2つのハッシュ値（底）に基づいたそれぞれのハッシュ化ビットマップを更新する。これにより、インデックス更新部114は、ファイルの削除に応じて、複数のファイルのいずれが削除であるかを示す削除のハッシュ化ビットマップに差分だけを更新できる。

【0069】

[実施例1に係るインデックス生成処理の処理手順]

次に、実施例1に係るインデックス生成処理の処理手順について、図11を参照して説明する。図11は、実施例1に係るインデックス生成処理のフローチャートの一例を示す図である。

10

【0070】

図11に示すように、符号化部112は、前処理を実行する（ステップS11）。例えば、符号化部112は、各種記憶領域を記憶部120に確保する。そして、符号化部112は、符号化対象の複数の対象ファイルのうち1つの対象ファイルの文字列を読み出し、読出用の記憶領域に格納する（ステップS12）。

【0071】

符号化部112は、読み出し用の記憶領域から、1つの対象ファイルの単語の文字列を読み出す（ステップS13）。例えば、符号化部112は、1つの対象ファイルに対して字句解析を行い、字句解析した結果の単語の文字列を先頭から読み出す。

【0072】

20

符号化部112は、読み出した単語を符号化する（ステップS14）。なお、単語の符号化処理の説明は、後述する。そして、符号化部112は、単語の符号化処理から出力された単語コードを、対象ファイルに対応する符号化データ用の記憶領域に書き込む（ステップS15）。

【0073】

続いて、インデックス生成部113は、単語コードに対応するビットマップをハッシュ化し「1」を書き込む（ステップS16）。例えば、インデックス生成部113は、底 α のハッシュ化ビットマップについて、ビットマップの各ビットの位置を底 α で割った余りの位置に、各ビットの値を設定する。インデックス生成部113は、底 β のハッシュ化ビットマップについて、ビットマップの各ビットの位置を底 β で割った余りの位置に、各ビットの値を設定する。すなわち、インデックス生成部113は、ビットマップの0ビット目からの各ビットを順番にハッシュ化ビットマップの0ビット目から設定し、（底-1）ビット目まで設定したら、再度折り返して0ビット目からOR演算した結果を設定する。すなわち、インデックス生成部113は、単語コードに対応するビットマップの、対象ファイルにおけるファイルIDに対応するビットに存在することを示す「1」を設定する。

30

【0074】

符号化部112は、対象ファイルの終点か否かを判定する（ステップS17）。対象ファイルの終点でないと判定した場合には（ステップS17；No）、符号化部112は、対象ファイルの次の単語を読み出すべく、ステップS13に移行する。

【0075】

40

一方、対象ファイルの終点であると判定した場合には（ステップS17；Yes）、符号化部112は、符号化されていない対象ファイルがあるか否かを判定する（ステップS18）。符号化されていない対象ファイルがあると判定した場合には（ステップS18；Yes）、符号化部112は、次の対象ファイルを符号化すべく、ステップS12に移行する。

【0076】

符号化されていない対象ファイルがないと判定した場合には（ステップS18；No）、インデックス生成部113は、生成されたハッシュ化ビットマップを纏めたハッシュ化インデックス124を記憶部120に格納する（ステップS19）。そして、インデックス生成部113は、インデックス生成処理を終了する。

50

【0077】

[実施例1に係る符号化処理の処理手順]

次に、実施例1に係る符号化処理の処理手順について、図12を参照して説明する。図2は、実施例1に係る符号化処理のフローチャートの一例を示す図である。なお、符号化部112は、単語の文字列を受け取ったものとする。

【0078】

図12に示すように、単語の文字列を受け取った符号化部112は、静的辞書121に当該単語の文字列が登録済みであるか否かを判定する(ステップS31)。静的辞書121に単語の文字列が登録済みであると判定した場合には(ステップS31; Yes)、符号化部112は、静的辞書121に登録されている静的コード(単語コード)を出力する(ステップS32)。そして、符号化部112は、符号化処理を終了する。

10

【0079】

一方、静的辞書121に単語の文字列が登録済みでないと判定した場合には(ステップS31; No)、符号化部112は、動的辞書122を参照する(ステップS33)。符号化部112は、動的辞書122に単語の文字列が登録済みであるか否かを判定する(ステップS34)。例えば、符号化部112は、単語の文字列が動的辞書122のバッファ部D1に既に格納されているか否かを判定する。

【0080】

動的辞書122に単語の文字列が登録済みであると判定した場合には(ステップS34; Yes)、符号化部112は、ステップS37に移行する。

20

【0081】

一方、動的辞書122に単語の文字列が登録済みでないと判定した場合には(ステップS34; No)、符号化部112は、当該単語に対する単語コードを割り当てる(ステップS35)。例えば、符号化部112は、外部辞書123から、単語の文字列に対応する符号および当該単語が登録された外部辞書123のIDを取得する。そして、符号化部112は、取得された符号を単語コードとして割り当てる。

【0082】

符号化部112は、割り当てられた単語コードを動的辞書122に登録する(ステップS36)。例えば、符号化部112は、割り当てられた単語コードおよび単語が登録された辞書のIDを、アドレステーブルD2の新たな動的コードに対応するレコードに格納する。加えて、符号化部112は、単語の文字列をバッファ部D1に格納するとともに、アドレステーブルD2の新たな動的コードに対応するレコードに、当該単語の文字列を格納した格納位置および格納したデータ長を格納する。そして、符号化部112は、ステップS37に移行する。

30

【0083】

ステップS37において、符号化部112は、動的辞書122に登録されている単語コードを出力する(ステップS37)。例えば、符号化部112は、単語の文字列を、当該単語の文字列に対応付けられた、アドレステーブルD2の単語コードに符号化する。符号化部112は、符号化した単語コードを出力する。そして、符号化部112は、符号化処理を終了する。

40

【0084】

[実施例1に係るインデックス更新処理の処理手順]

次に、実施例1に係るインデックス更新処理の処理手順について、図13を参照して説明する。図13は、実施例1に係るインデックス更新処理のフローチャートの一例を示す図である。

【0085】

図13に示すように、インデックス更新部114は、ファイルが追加されたか否かを判定する(ステップS41)。ファイルが追加されたと判定した場合には(ステップS41; Yes)、インデックス更新部114は、削除ビットマップに対応するハッシュ化ビットマップの、追加ファイルのファイルIDのビットに「0」をデフォルトとして設定する

50

(ステップS 4 2)。

【0086】

続いて、インデックス更新部114は、追加ファイルの単語の文字列を読み出す(ステップS 4 3)。例えば、インデックス更新部114は、追加ファイルに対して字句解析を行い、字句解析した結果の単語の文字列を先頭から読み出す。

【0087】

インデックス更新部114は、読み出した単語を符号化する(ステップS 4 4)。なお、単語の符号化処理の説明は、図12で説明したので、その説明を省略する。そして、インデックス更新部114は、単語の符号化処理から出力された単語コードを、追加ファイルに対応する符号化データ用の記憶領域に書き込む(ステップS 4 5)。

10

【0088】

続いて、インデックス更新部114は、単語コードに対応するハッシュ化ビットマップの、追加ファイルのファイルIDのビットに「1」を更新する(ステップS 4 6)。例えば、インデックス更新部114は、単語コードに対応する底 α のハッシュ化ビットマップについて、追加ファイルIDに対応する、ビットマップのビット位置を α で割った余りの位置に、更新ビットの値「1」と既に設定された値「0」とをOR演算した値「1」を更新する。インデックス更新部114は、単語コードに対応する底 β のハッシュ化ビットマップについて、追加ファイルIDに対応する、ビットマップのビット位置を β で割った余りの位置に、更新ビットの値「1」と既に設定された値「0」とをOR演算した値「1」を更新する。

20

【0089】

インデックス更新部114は、追加ファイルの終点か否かを判定する(ステップS 4 7)。追加ファイルの終点でないと判定した場合には(ステップS 4 7; No)、インデックス更新部114は、追加ファイルの次の単語を読み出すべく、ステップS 4 3に移行する。

【0090】

一方、追加ファイルの終点であると判定した場合には(ステップS 4 7; Yes)、インデックス更新部114は、インデックス更新処理を終了する。

【0091】

ステップS 4 1において、ファイルが追加されていないと判定した場合には(ステップS 4 1; No)、インデックス更新部114は、ファイルが削除されたか否かを判定する(ステップS 4 8)。ファイルが削除されていないと判定した場合には(ステップS 4 8; No)、インデックス更新部114は、判定処理を繰り返すべく、ステップS 4 1に移行する。

30

【0092】

一方、ファイルが削除されたと判定した場合には(ステップS 4 8; Yes)、インデックス更新部114は、削除ビットマップに対応するハッシュ化ビットマップの、削除ファイルのファイルIDのビットに「1」を更新する(ステップS 4 9)。例えば、インデックス更新部114は、削除ビットマップに対応する底 α のハッシュ化ビットマップについて、削除ファイルIDに対応する、ビットマップのビット位置を α で割った余りの位置に、更新ビットの値「1」と既に設定された値とをOR演算した値「1」を更新する。インデックス更新部114は、削除ビットマップに対応する底 β のハッシュ化ビットマップについて、削除ファイルIDに対応する、ビットマップのビット位置を β で割った余りの位置に、更新ビットの値「1」と既に設定された値とをOR演算した値「1」を更新する。そして、インデックス更新部114は、インデックス更新処理を終了する。

40

【0093】

[実施例1に係る検索装置の構成]

次に、図14を参照して、実施例1に係る検索処理を実行する検索装置200の構成について説明する。図14は、実施例1に係る検索装置の構成を示す機能ブロック図である。図14に示すように、検索装置200は、制御部210と記憶部220とを有する。

50

【0094】

制御部210は、図8に示した検索処理を実行する処理部である。制御部210は、各種の処理手順を規定したプログラムや制御データを格納するための内部メモリを有し、これらによって種々の処理を実行する。そして、制御部210は、例えば、ASICやFPGAなどの集積回路の電子回路に対応する。または、制御部210は、CPUやMPUなどの電子回路に対応する。また、制御部210は、検索キー受付部211、復元部212、検索処理部213および検索結果出力部214を有する。

【0095】

記憶部220は、例えばフラッシュメモリやFRAMなどの不揮発性の半導体メモリ素子などの記憶装置に対応する。記憶部220は、ハッシュ化インデックス221と復元ビットマップ222および絞込みビットマップ223を有する。なお、ハッシュ化インデックス221と復元ビットマップ222の構成は、各単語IDに対応するハッシュ化ビットマップとビットマップの構成（図2参照）と同様であるので、その説明を省略する。また、絞込みビットマップ223の構成は、各単語とAND結果に対応するビットマップの構成（図8参照）と同様であるので、その説明を省略する。

10

【0096】

検索キー受付部211は、検索キーを受け付ける。例えば、検索キー受付部211は、検索キーとして検索対象の単語を受け付ける。

【0097】

復元部212は、検索対象の単語に対応する複数のハッシュ化ビットマップを復元する。例えば、復元部212は、検索対象の単語の単語IDに対応する複数のハッシュ化ビットマップをそれぞれビットマップに展開する（第1の復元処理）。一例として、底が α と β であるとする。すると、復元部212は、底 α のハッシュ化ビットマップの復元先のビットマップについて、底 α に整数（0～）を乗算した値にハッシュ化ビットマップの各ビットの位置を加算した位置に、ハッシュ化ビットマップの各ビットの値を設定する。復元部212は、底 β のハッシュ化ビットマップの復元先のビットマップについて、底 β に整数（0～）を乗算した値にハッシュ化ビットマップの各ビットの位置を加算した位置に、ハッシュ化ビットマップの各ビットの値を設定する。すなわち、復元部212は、ハッシュ化ビットマップの0ビット目からの各ビットを順番に復元先のビットマップの0ビット目から設定し、（底-1）ビット目まで処理する。そして、復元部212は、再度折り返してハッシュ化ビットマップの0ビット目からハッシュ化ビットマップの値を設定する。

20

30

【0098】

そして、復元部212は、第1の復元処理で展開されたそれぞれのビットマップの対応する位置のビットをAND演算する（第2の復元処理）。一例として、復元部212は、底 α のハッシュ化ビットマップを展開したビットマップと、底 β のハッシュ化ビットマップを展開したビットマップとをAND演算する。そして、復元部212は、AND結果のビットマップを復元結果として復元ビットマップ222に保持する。なお、検索対象の単語が複数ある場合には、復元部212は、検索対象の単語ごとにハッシュ化ビットマップを復元し、復元結果であるビットマップを復元ビットマップ222に保持する。

【0099】

また、復元部212は、ハッシュ化された削除ビットマップを復元する。一例として、復元部212は、ハッシュ化された削除ビットマップについて、第1の復元処理および第2の復元処理を実行し、削除ビットマップを復元結果として復元ビットマップ222に保持する。

40

【0100】

検索処理部213は、検索対象の単語が存在するファイルを検索する。例えば、検索処理部213は、復元部212によって復元ビットマップ222に保持されたビットマップと反転した削除ビットマップとのAND演算を行い、AND結果を絞込みビットマップ223に保持する。

【0101】

50

検索結果出力部 214 は、検索処理部 213 によって保持された絞込みビットマップ 223 を用いて、検索結果を出力する。例えば、検索結果出力部 214 は、絞込みビットマップ 223 のビットが「1」を示すファイル ID を検索結果として出力する。

【0102】

[検索処理のフローチャート]

図 15 は、実施例 1 に係る検索処理のフローチャートの一例を示す図である。

【0103】

まず、検索キー受付部 211 は、検索対象の単語を受け付けたか否かを判定する（ステップ S51）。検索対象の単語を受け付けていないと判定した場合には（ステップ S51；No）、検索キー受付部 211 は、検索対象の単語を受け付けるまで、判定処理を繰り返す。

10

【0104】

一方、検索対象の単語を受け付けたと判定した場合には（ステップ S51；Yes）、復元部 212 は、ハッシュ化インデックス 221 を記憶部 220 から読み出す（ステップ S51A）。復元部 212 は、読み出したハッシュ化インデックス 221 から、検索対象の単語が示す単語 ID に対応する、底 α 、底 β の各ハッシュ化ビットマップを選択する（ステップ S52）。

【0105】

復元部 212 は、選択されたハッシュ化ビットマップを復元する（ステップ S53）。なお、かかる復元処理は、底 α および底 β におけるハッシュ化ビットマップごとに行われる。すなわち、復元部 212 は、底 α のハッシュ化ビットマップについて、第 1 の復元処理を行う。復元部 212 は、底 β のハッシュ化ビットマップについて、第 1 の復元処理を行う。

20

【0106】

そして、復元部 212 は、底 α のビットマップと底 β のビットマップとを AND 演算する（ステップ S54）。すなわち、復元部 212 は、第 1 の復元処理によって復元された底 α のビットマップおよび底 β のビットマップを用いて、第 2 の復元処理を行う。復元部 212 は、AND 結果を復元ビットマップ 222 に保持する。

【0107】

復元部 212 は、未処理の単語があるか否かを判定する（ステップ S55）。未処理の単語があると判定した場合には（ステップ S55；Yes）、復元部 212 は検索対象の次の単語を処理すべく、ステップ S52 に移行する。

30

【0108】

一方、未処理の単語がないと判定した場合には（ステップ S55；No）、復元部 212 は、ハッシュ化された削除ビットマップを復元する（ステップ S56）。なお、かかる復元処理は、底 α および底 β におけるハッシュ化削除ビットマップごとに行われる。すなわち、復元部 212 は、底 α のハッシュ化削除ビットマップについて、第 1 の復元処理を行う。復元部 212 は、底 β のハッシュ化削除ビットマップについて、第 1 の復元処理を行う。

【0109】

そして、復元部 212 は、底 α の削除ビットマップと底 β の削除ビットマップとを AND 演算する（ステップ S57）。すなわち、復元部 212 は、第 1 の復元処理によって復元された底 α のビットマップおよび底 β のビットマップを用いて、第 2 の復元処理を行う。復元部 212 は、AND 結果を復元ビットマップ 222 に保持する。

40

【0110】

そして、検索処理部 213 は、検索対象の単語のビットマップと削除ビットマップとを AND 演算する（ステップ S58）。例えば、演算処理部 213 は、復元ビットマップ 222 に保持されたビットマップおよび反転した削除ビットマップをビットごとに AND 演算する。演算処理部 213 は、AND 結果を絞込みビットマップ 223 に保持する。

【0111】

50

そして、検索結果出力部 214 は、AND 結果のビットが「1」を示すファイル ID を検索結果として出力する（ステップ S59）。例えば、検索結果出力部 214 は、絞込みビットマップ 223 に保持された AND 結果を用いて、ビットが「1」を示すファイル ID を検索結果として出力する。

【0112】

これにより、符号化装置 100 は、符号化対象のファイルを、静的辞書 121 および動的辞書 122 を用いて符号化する。符号化装置 100 は、静的辞書 121 および動的辞書 122 に登録された単語の、ファイルにおける存否を示すインデックス情報を、ファイルの軸方向および単語の軸方向をそれぞれ所定の底でハッシュ化して折り返して生成する。符号化装置 100 は、ファイルが更新された際に、更新後のファイルに対応したインデックス情報の、更新前のインデックス情報に対する差分情報を、ファイルの軸方向または単語の軸方向について生成する。かかる構成によれば、符号化装置 100 は、ファイルの更新に応じて、ファイルにおける存否を示すインデックス情報を容易に更新することができる。すなわち、符号化装置 100 は、ファイルの更新に応じて、更新前のインデックス情報に対する更新後の差分情報を生成するので、インデックス情報に差分だけを更新することができる。例えば、符号化装置 100 は、ファイルが追加された場合には、追加されたファイルにおける単語の存否情報だけを追加前後のインデックス情報に対する差分情報として生成するので、インデックス情報にこの差分情報だけを更新することができる。さらに、ファイルに新語や流行語と呼ばれる単語が含まれていたとしても、符号化装置 100 は、インデックス情報を再生成しなくても、差分情報を更新することで、インデックス情報を更新できる。

【実施例 2】

【0113】

ところで、実施例 1 では、符号化装置 100 は、複数のビットマップ型インデックス BI を対象として複数のビットマップ型インデックス BI を基に、それぞれハッシュ化インデックス HI を生成する。そして、符号化装置 100 は、ファイルが更新されると、更新ファイルを担当するハッシュ化インデックス HI に対して、更新後のファイルに対応したビットマップの、更新前のビットマップに対する差分情報を当該ハッシュ化インデックス HI に更新する。そして、符号化装置 100 は、複数のビットマップ型インデックス BI を階層化しても良い。

【0114】

そこで、実施例 2 では、符号化装置 100 は、複数のビットマップ型インデックス BI を階層化する場合を説明する。

【0115】

[実施例 2 に係るビットマップ型インデックスの階層化]

図 16 は、実施例 2 に係るビットマップ型インデックスの階層化の一例を示す図である。図 16 に示すように、ビットマップ型インデックス BI は、所定のファイル数 n 単位で分割されている。 n は例えば 256 である。分割されたそれぞれを、セグメントと称す。

【0116】

セグメント $sg0(1)$ は、符号化対象のファイル $f1 \sim fn$ までのビットマップ型インデックス BI を有するセグメントである。セグメント $sg0(2)$ は、ファイル $f(n+1) \sim f(2n)$ までのビットマップ型インデックス BI を有するセグメントである。セグメント $sg0(3)$ は、ファイル $f(2n+1) \sim f(3n)$ までのビットマップ型インデックス BI を有するセグメントである。

【0117】

セグメントは初期状態では、セグメント $sg0(1)$ のみが存在し、ファイル追加によりファイル数が n を超えると、セグメント $sg0(2)$ が生成され、ファイル数が $2n$ を超えると、セグメント $sg0(3)$ が生成される。

【0118】

各セグメントのビットマップ型インデックス BI は、単語が同じであるビットマップを

有するが、担当するファイル I D が異なる。各セグメントのビットマップ型インデックス B I において担当するファイル I D は、自セグメントが保持する符号化ファイルのファイル I D となる。例えば、セグメント s g 0 (1) のビットマップ型インデックス B I では、各単語のビットマップについてファイル I D 1 ~ n の存否を示すビット列を有することになる。セグメント s g 0 (m) のビットマップ型インデックス B I では、各単語のビットマップについてファイル I D (m - 1) n ~ m n の存否を示すビット列を有することになる。

【 0 1 1 9 】

動的辞書 D 0 は、静的辞書 S 0 に登録されず、且つ、外部辞書 E 0 に登録されている単語と、外部辞書 E 0 における符号と、動的に付された動的コードとを対応付けた辞書である。すなわち、符号化装置 1 0 0 は、静的辞書 S 0 に単語の文字列に対応する符号がない場合には、外部辞書 E 0 から、単語の文字列に対応する符号（単語コード）および当該単語コードが登録された外部辞書 E 0 の I D を取得する。符号化装置 1 0 0 は、単語の文字列と、外部辞書 E 0 から取得された符号（単語コード）と、動的に付された動的コードとを動的辞書 D 0 に登録する。なお、動的辞書 D 0 は、実施例 1 ではファイル単位に生成されたが、ファイル単位に生成されても、セグメント単位に生成されても良い。図 1 6 では、動的辞書 D 0 は、セグメント単位に生成される場合である。

10

【 0 1 2 0 】

ここで、セグメント数が増加すると、単語 I D ごとにビットマップが冗長化する。冗長化した場合、単語 I D ごとに、全ファイル数について存否を示すビットの O N / O F F を確認する必要があるが、存在しない（O F F の）箇所について確認作業をするのは無駄である。そこで、単語 I D ごとに、セグメントが m + 1 個生成されたら、m 個単位でビットマップを集約することにする。

20

【 0 1 2 1 】

図 1 6 では、例えば、単語 x が示す単語 I D x のビットマップを上位階層に集約する場合を例に挙げて説明する。なお、「s g X (Y)」の X は、階層番号を示しており、Y はセグメント番号を示している。したがって、s g X (Y) の場合は、第 X 階層の Y 番目のセグメントとなる。また、これまで説明してきたセグメント s g 0 (1) ~ s g 0 (m) は、基本階層である第 0 階層のセグメントとなる。

【 0 1 2 2 】

まず、基本階層である第 0 階層から上位階層である第 1 階層への集約例について説明する。集約のルールとしては、対象階層のセグメントのビットマップであるビット列がオール 0、すなわち、単語 x がそのセグメント内の対象ファイル群に存在しない場合、上位階層のインデックスとして「0」に集約する。一方、対象階層のセグメントのビットマップであるビット列に 1 個でも「1」がある場合、すなわち、単語 x がそのセグメント内の対象ファイル群の少なくともいずれか 1 つに存在する場合、上位階層のインデックスとして「1」に集約する。

30

【 0 1 2 3 】

例えば、セグメント s g 0 (1) については、「1」が存在するため、上位階層のセグメント s g 1 (1) には、「1」が設定される。同様に、セグメント s g 0 (m) については、オール 0 であるため、上位階層のセグメント s g 1 (1) には、「0」が設定される。この上位階層のセグメント s g 1 (1) のビットマップのビット位置は、下位のセグメント s g 0 (1) ~ s g 0 (m) の位置を示している。このように、上位階層のセグメントのビットの値により下位のセグメントのビットマップの状態を特定することができる。

40

【 0 1 2 4 】

また、このような集約は第 0 階層と第 1 階層との間だけではなく、最上位階層のセグメント数が m 個になると、新たに上位階層のセグメントが生成されることとなる。例えば、第 1 階層でセグメントがセグメント s g 1 (m) まで生成されると、第 2 階層のセグメント s g 2 (1) が生成されることとなる。なお、図 1 6 では、第 2 階層までの例を示した

50

が、データの規模が増加するにしたがって、第3階層以上の階層に集約されることになる。

【0125】

図16では、基本階層の各セグメントは、ハッシュ化された状態で保持される。すなわち、符号化装置100は、自己が担当するセグメントについて、セグメントが示すビットマップ型インデックスBIからハッシュ化インデックスHIを生成して、生成して得られたハッシュ化インデックスHIを保持する。そして、符号化装置100は、ファイルが追加されると、追加ファイルを担当するハッシュ化インデックスHIに対して、追加後のファイルに対応したビットマップの、追加前のビットマップに対する差分情報を当該ハッシュ化インデックスHIに更新する。更新の際に、符号化装置100は、更新される単語x

10

【0126】

[実施例2に係る削除ビットマップの階層化]

図17は、実施例2に係る削除ビットマップの階層化の一例を示す図である。図17に示すように、削除ビットマップdbについても、図16と同様セグメント単位で上位階層に集約されることになる。

【0127】

また、削除ビットマップbdについても、基本階層の各セグメントは、ハッシュ化された状態で保持される。すなわち、符号化装置100は、自己が担当するセグメントについて、削除ビットマップbdからハッシュ化ビットマップhbdを生成して、生成して得られたハッシュ化ビットマップhbdを保持する。そして、符号化装置100は、ファイルが削除されると、削除ファイルを担当するハッシュ化ビットマップhbdに対して、削除前後の差分情報を更新する。更新の際に、符号化装置100は、該当するハッシュ化ビットマップhbdのビットが全て「0」となったタイミングで、上位階層のインデックスとして「0」に集約する。一方、符号化装置100は、ファイルが追加されると、追加ファイルを担当するハッシュ化ビットマップhbdに対して、追加前後の差分情報を更新する。更新の際に、符号化装置100は、該当するハッシュ化ビットマップhbdに、最初に「1」が設定されるタイミングで、上位階層のインデックスとして「1」に集約する。

20

【0128】

[ビットマップ型インデックスの更新処理の一例]

図18は、実施例2に係るビットマップ型インデックスの更新処理の一例を示す図である。図18上図に示すように、セグメントsg0(1)が登録済みであり、単語ID1～単語IDxについてのビットマップ型インデックスBIにおいて、ファイルID1～nまでのハッシュ化されたインデックス情報が格納されている。

30

【0129】

図18下図は、図18上図の状態から新たなファイルn+1が追加された状態を示している。図18上図での最後尾のセグメントであるセグメントsg0(1)ではファイルn+1を格納できないため、新たなセグメントsg0(2)が設定され、セグメントsg0(2)にファイルn+1が保存されることになる。

40

【0130】

セグメントsg0(2)のビットマップ型インデックスBIには、単語IDごとにファイルn+1についてのビットが設定される。図18下図の例では、単語ID1、単語ID2については「1」、単語IDxについては「0」が設定されている。そして、符号化装置100は、一例として29と31のハッシュ値(底)を基に、新たなセグメントsg0(2)の各ビットマップをハッシュ化する。具体的には、ビットマップb6の0ビット目が、ファイルIDn+1に対応する。すると、符号化装置100は、底29のハッシュ化ビットマップh61について、ビットマップb6の0ビットの位置を底29で割った余りの位置に、当該ビットマップの0ビットの値を設定する。符号化装置100は、底31のハッシュ化ビットマップh62について、ビットマップb6の0ビットの位置を底31で

50

割った余りの位置に、当該ビットマップの0ビットの値を設定する。この結果、ファイルID $n+1$ のファイルが追加されても、符号化装置100は、追加されたファイルにおける単語の存否情報を対象にして、ハッシュ化ビットマップを更新するので、ビットマップの差分だけを更新できる。

【0131】

[階層化されたセグメント群を用いた検索処理の一例]

図19は、階層化されたセグメント群を用いた検索処理の一例を示す図である。図19では、説明を単純化するため、第0階層のセグメントのファイル数 n を4とし、 m を4とする。したがって、第0階層セグメント $sg0(1) \sim sg0(16)$ まで存在することになるが、図示されていないセグメントについては説明を省略する。また、図19では、最上位階層を第2階層として説明する。また、図19において、実線矢印はAND結果にしたがって下位階層のセグメントを指定していることを示しており、点線矢印は実際には指定されないが、指定されたセグメントと対比するため図示している。また、図19では、検索文字列として「Sherlock△Mickey」が入力された場合とする。なお、 $P(s)$ は、単語 s の単語IDを示すものとする。また、図19における削除マップは、削除ビットマップの略称であり、「0」である場合には削除されたことを示し、「1」である場合には削除されていないことを示す。

【0132】

(A)において、最上位階層である第2階層のセグメント $sg2(1)$ について、単語「Sherlock」の単語IDを示す $P(Sherlock)$ のビットマップと、単語「Mickey」の単語IDを示す $P(Mickey)$ のビットマップと、集約削除マップと、のAND演算を行う。AND結果は「1100」となり、下位階層である第1階層のセグメント $sg1(1)$ 、 $sg1(2)$ に、単語「Sherlock」および「Mickey」が存在する可能性があることがわかる。

【0133】

また、最上位階層である第2階層のセグメント $sg2(1)$ について、下位セグメント番号3、4についてはAND結果が「0」である。したがって、セグメント $sg1(3)$ 、 $sg1(4)$ の各AND演算を行うまでもなく、セグメント $sg1(3)$ 、 $sg1(4)$ の各AND結果がオール0になることがわかる。

【0134】

(B)では、第2階層のAND結果から第1階層のセグメント $sg1(1)$ 、 $sg1(2)$ が指定されたため、セグメント $sg1(1)$ 、 $sg1(2)$ について、(A)と同様AND演算を実行する。これにより、セグメント $sg1(1)$ からはセグメント $sg0(1)$ が指定され、セグメント $sg1(2)$ からはセグメント $sg0(5)$ が指定されることになる。セグメント $sg1(2)$ に着目すると、下位セグメント番号6～8についてはAND結果が「0」である。したがって、セグメント $sg0(6) \sim sg0(8)$ の各AND演算を行うまでもなく、セグメント $sg0(6) \sim sg0(8)$ の各AND結果がオール0になることがわかる。

【0135】

(C)では、第1階層のAND結果から第0階層のセグメント $sg0(1)$ 、 $sg0(5)$ が指定されたため、セグメント $sg0(1)$ 、 $sg0(5)$ について、(A)、(B)と同様AND演算を実行する。すなわち、検索装置200は、単語「Sherlock」の単語IDに対するハッシュ化ビットマップを復元し、 $P(Sherlock)$ のビットマップを復元結果として出力する。検索装置200は、単語「Mickey」の単語IDに対するハッシュ化ビットマップを復元し、 $P(Mickey)$ のビットマップを復元結果として出力する。検索装置200は、ハッシュ化された削除ビットマップを復元し、削除ビットマップを復元結果として出力する。そして、検索装置200は、セグメント $sg0(1)$ について、 $P(Sherlock)$ のビットマップと、 $P(Mickey)$ のビットマップと、削除ビットマップと、のAND演算を行う。検索装置200は、セグメント $sg0(5)$ について、 $P(Sherlock)$ のビットマップと、 $P(Micke$

y) のビットマップと、削除ビットマップと、のAND演算を行う。これにより、セグメントsg0(1)からはファイルID3が指定され、セグメントsg0(5)からはファイルID19が指定されることになる。したがって、ファイルf3、f19に、単語「Sherlock」および「Mickey」の両方が存在することがわかる。

【0136】

[階層化されたセグメント群を実装したコンピュータシステムの構成例]

図20は、階層化されたセグメント群を実装したコンピュータシステムの構成例を示す図である。図20では、m個のセグメントを1アーカイブファイルとする。なお、「AX(Y)」は、アーカイブファイルの符号であり、Xは階層番号を示しており、Yはアーカイブ番号を示している。したがって、AX(Y)の場合は、第X階層のY番目のアーカイブファイルとなる。例えば、アーカイブファイルA0(1)は、第0階層のセグメントsg0(1)～sg0(m)の集合である。

10

【0137】

図20の例では、マスタサーバMSは、第1階層以上のアーカイブファイルを格納している。また、スレーブサーバS1、S2、・・・、S(2m+1)、・・・は、マスタサーバMSにより割り当てられた1個のアーカイブファイルを格納している。なお、個々のスレーブサーバは、例えば、それぞれ符号化装置100および検索装置200に対応する。図20のアーカイブファイルの割り当ては1個であり、マスタサーバMSが第1階層以上のアーカイブファイルをすべて担当する必要はなく、他のサーバに分散させても良い。また、スレーブサーバS1、S2、・・・、S(2m+1)、・・・についても1アーカイブファイルだけではなく、複数のアーカイブファイルを担当することとしても良い。

20

【0138】

これにより、符号化装置100および検索装置200を含むコンピュータシステムは、例えば専門用語、新語や未知語に対する符号(単語コード)をもとに、符号化および階層化を行うことで、容易にスレーブサーバ全体のパフォーマンスを向上させることができる。すなわち、コンピュータシステムは、スケールアウトすることができる。具体的には、マスタサーバMSは、担当しているアーカイブファイル内のAND結果を参照して、負荷が均一となるように、セグメントにスレーブサーバを割り当てることができる。

【0139】

[実施例に関連する他の態様]

30

以下、上述の実施形態における変形例の一部を説明する。下記の変形例のみでなく、本発明の本旨を逸脱しない範囲の設計変更は適宜行われうる。

【0140】

また、実施例に係る符号化装置100は、32ビットレジスタを想定し、29と31のハッシュ値(底)を基に、各ビットマップをハッシュ化すると説明した。しかしながら、29および31のハッシュ値(底)は、一例であって、これに限定されない。2つのハッシュ値(底)は、複数のファイル内のそれぞれの単語の種類の数に応じて決定されれば良い。例えば、単語の種類数が10000であるとすると、一方の底で割った余りと他方の底で割った余りから表わされる2次元の行列が約10000となるように、2つの底が選択される。2つの底は、隣接した素数であれば良い。選択される2つの素数は、行列の数が10000の場合、一例として、97と101である。つまり、最小公倍数が約10000となる2次元のマトリックス空間の中で、ある単語について一方のハッシュおよび他方のハッシュで求められる余りの組は、他の単語について求められる余りの組と衝突しない(重複しない)であろうという推測に基づくものである。

40

【0141】

また、実施例では、符号化装置100が、複数の単語IDに対応するビットマップそれぞれについて、2つのハッシュ値(底)に基づいたそれぞれのハッシュ化ビットマップを生成すると説明した。符号化装置100は、ハッシュ化ビットマップを生成する際、ハッシュの衝突(ハッシュノイズ)を検知する場合がある。例えば、超高頻度の単語は、複数のファイルに存在するため、超高頻度の単語に対応するビットマップの複数位置のビット

50

値が「1」に設定される。すると、ビットマップがハッシュ化されると、ハッシュ化ビットマップの同じ位置に「1」が重複して設定されることがある。超高頻度の単語の一例として、「the」や「on」が挙げられる。そこで、符号化装置100は、ハッシュノイズに対して、ハッシュの衝突監視を行い、0/1比率の測定やビットマップの分割により、ハッシュノイズの低減化を行えば良い。例えば、符号化装置100は、ハッシュ化ビットマップのいずれか1つで連続して衝突が発生した場合に、衝突が発生したハッシュ化ビットマップに対応する単語IDのビットマップの存否情報を用いて存否(1/0)の比率を集計する。符号化装置100は、「1」の比率が閾値より大きい場合には、衝突が発生したハッシュ化ビットマップに対応する単語IDのビットマップを分割する。具体的には、符号化装置100は、衝突が発生したハッシュ化ビットマップに対応する単語IDのビットマップの偶数番目の位置のビットを抽出し、新たにビットマップを生成する。加えて、符号化装置100は、衝突が発生したハッシュ化ビットマップに対応する単語IDのビットマップの奇数番目の位置のビットを抽出し、新たにビットマップを生成する。そして、符号化装置100は、分割した新たなビットマップを、分割先として例えば低頻度単語の領域に格納する。符号化装置100は、2つのハッシュ化ビットマップのいずれか1つに対し分割先を設定する。そして、符号化装置100は、ビットマップを分割後に、分割先の各ビットマップに対して、2つのハッシュ値(底)に基づいたそれぞれのハッシュ化ビットマップを生成する。これにより、符号化装置100は、ハッシュ化ビットマップのデータが衝突する場合であっても、ハッシュ化前のビットマップの偶数番目のデータと奇数番目のデータとを分割してそれぞれハッシュ化することで、データの衝突を回避することが可能となる。

10

20

【0142】

また、実施例に係る符号化装置100は、隣接した複数のハッシュ値(底)を基に、2次元(単語の軸とファイルの軸)にハッシュ化を適用したハッシュ化インデックスを生成すると説明した。しかしながら、符号化装置100は、ファイルの軸に代えてブロックの軸としても良い。すなわち、単語IDの存否情報は、ブロック単位であるとしても良い。

【0143】

また、実施例に示した処理手順、制御手順、具体的名称、各種のデータやパラメータを含む情報については、特記する場合を除いて任意に変更することができる。

【0144】

[ハードウェア構成]

下記に、上述の実施形態に用いられるハードウェア及びソフトウェアについて説明する。図21は、コンピュータ1のハードウェア構成例を示す図である。コンピュータ1は、例えば、プロセッサ301、RAM(Random Access Memory)302、ROM(Read Only Memory)303、ドライブ装置304、記憶媒体305、入力インターフェース(I/F)306、入力デバイス307、出力インターフェース(I/F)308、出力デバイス309、通信インターフェース(I/F)310、SAN(Storage Area Network)インターフェース(I/F)311およびバス312などを含む。それぞれのハードウェアはバス312を介して接続されている。

40

【0145】

RAM302は読み書き可能なメモリ装置であって、例えば、SRAM(Static RAM)やDRAM(Dynamic RAM)などの半導体メモリ、またはRAMでなくてもフラッシュメモリなどが用いられる。ROM303は、PROM(Programmable ROM)なども含む。ドライブ装置304は、記憶媒体305に記録された情報の読み出しか書き込みかの少なくともいずれか一方を行なう装置である。記憶媒体305は、ドライブ装置304によって書き込まれた情報を記憶する。記憶媒体305は、例えば、ハードディスク、SSD(Solid State Drive)などのフラッシュメモリ、CD(Compact Disc)、DVD(Digital Versatile Disc)、ブルーレイディスクなどの記憶媒体である。また、例えば、コ

50

ンピュータ1は、複数種類の記憶媒体それぞれについて、ドライブ装置304及び記憶媒体305を設ける。

【0146】

入力インターフェース306は、入力デバイス307と接続されており、入力デバイス307から受信した入力信号をプロセッサ301に伝達する回路である。出力インターフェース308は、出力デバイス309と接続されており、出力デバイス309に、プロセッサ301の指示に応じた出力を実行させる回路である。通信インターフェース310はネットワーク3を介した通信の制御を行なう回路である。通信インターフェース310は、例えばネットワークインターフェースカード(NIC)などである。SANインターフェース311は、ストレージエリアネットワークによりコンピュータ1と接続された記憶装置との通信の制御を行なう回路である。SANインターフェース311は、例えばホストバスアダプタ(HBA)などである。

10

【0147】

入力デバイス307は、操作に応じて入力信号を送信する装置である。入力信号は、例えば、キーボードやコンピュータ1の本体に取り付けられたボタンなどのキー装置や、マウスやタッチパネルなどのポインティングデバイスである。出力デバイス309は、コンピュータ1の制御に応じて情報を出力する装置である。出力デバイス309は、例えば、ディスプレイなどの画像出力装置(表示デバイス)や、スピーカなどの音声出力装置などである。また、例えば、タッチスクリーンなどの入出力装置が、入力デバイス307及び出力デバイス309として用いられる。また、入力デバイス307及び出力デバイス309は、コンピュータ1と一体になっていても良いし、コンピュータ1に含まれず、例えば、コンピュータ1に外部から接続する装置であっても良い。

20

【0148】

例えば、プロセッサ301は、ROM303や記憶媒体305に記憶されたプログラムをRAM302に読み出し、読み出されたプログラムの手順に従って制御部110、210の処理を行なう。その際にRAM302はプロセッサ301のワークエリアとして用いられる。記憶部120、220の機能は、ROM303および記憶媒体305がプログラムファイル(後述のアプリケーションプログラム24、ミドルウェア23およびOS22など)やデータファイル(例えば、静的辞書121、動的辞書122、外部辞書123、ビットマップ型インデックス部124、ハッシュ化インデックス125など)を記憶し、RAM302がプロセッサ301のワークエリアとして用いられることによって実現される。プロセッサ301が読み出すプログラムについては、図22を用いて説明する。

30

【0149】

図22は、コンピュータで動作するプログラムの構成例を示す図である。コンピュータ1において、図22に示すハードウェア群HW21(301~312)の制御を行なうOS(オペレーティング・システム)22が動作する。OS22に従った手順でプロセッサ301が動作して、ハードウェア群HW21の制御・管理が行なわれることにより、アプリケーションプログラムAP24やミドルウェアMW23に従った処理がハードウェア群21で実行される。さらに、コンピュータ1において、ミドルウェアMW23またはアプリケーションプログラムAP24が、RAM302に読み出されてプロセッサ301により実行される。

40

【0150】

プロセッサ301が、符号化機能が呼び出された場合に、ミドルウェア23またはアプリケーションプログラム24の少なくとも一部に基づく処理を行なうことにより、(それらの処理をOS22に基づいてハードウェア群21を制御して)制御部110の機能が実現される。プロセッサ301が、検索機能が呼び出された場合に、ミドルウェア23またはアプリケーションプログラム24の少なくとも一部に基づく処理を行なうことにより、(それらの処理をOS22に基づいてハードウェア群21を制御して)制御部210の機能が実現される。符号化機能および検索機能は、アプリケーションプログラム24自体に含まれても良いし、アプリケーションプログラム24に従って呼び出されることで実行さ

50

れるミドルウェア 2 3 の一部であっても良い。

【0151】

図 2 3 は、実施形態のシステムにおける装置の構成例を示す図である。図 2 3 のシステムは、コンピュータ 1 a、コンピュータ 1 b、基地局 2 およびネットワーク 3 を含む。コンピュータ 1 a は、無線または有線の少なくとも一方により、コンピュータ 1 b と接続されたネットワーク 3 に接続している。

【0152】

符号化装置 1 0 0 と検索装置 2 0 0 とは、図 2 3 に示すコンピュータ 1 a とコンピュータ 1 b とのいずれに含まれても良い。コンピュータ 1 b が符号化装置 1 0 0 の機能を含み、コンピュータ 1 a が検索装置 2 0 0 の機能を含んでも良いし、コンピュータ 1 a が符号化装置 1 0 0 の機能を含み、コンピュータ 1 b が検索装置 2 0 0 の機能を含んでも良い。また、コンピュータ 1 a とコンピュータ 1 b との双方が、符号化装置 1 0 0 の機能および検索装置 2 0 0 の機能を備えても良い。

【符号の説明】

【0153】

- 1 0 0 符号化装置
- 1 1 0 制御部
- 1 1 1 ファイルリード部
- 1 1 2 符号化部
- 1 1 3 インデックス生成部
- 1 1 4 インデックス更新部
- 1 2 0 記憶部
- 1 2 1 静的辞書
- 1 2 2 動的辞書
- 1 2 3 外部辞書
- 1 2 4 ハッシュ化インデックス
- 2 0 0 検索装置
- 2 1 0 制御部
- 2 1 1 検索キー受付部
- 2 1 2 復元部
- 2 1 3 検索処理部
- 2 1 4 検索結果出力部
- 2 2 0 記憶部
- 2 2 1 ハッシュ化インデックス
- 2 2 2 復元ビットマップ
- 2 2 3 絞込みビットマップ

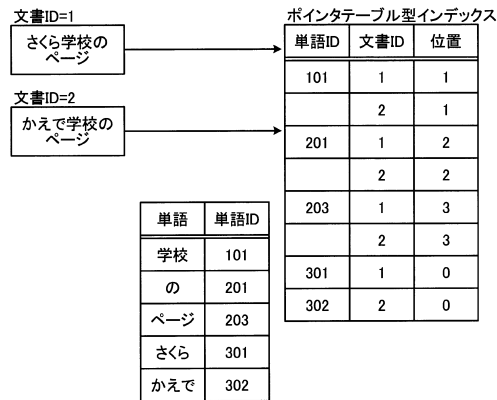
10

20

30

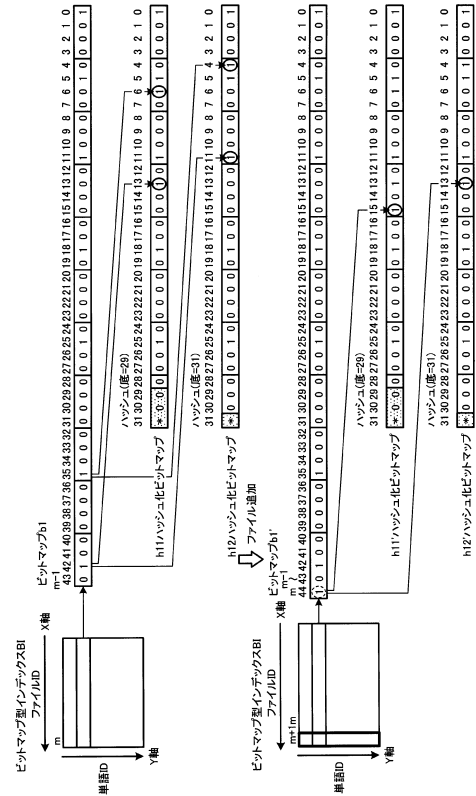
【図 1】

ポインタテーブル型インデックスの生成処理の参考例を示す図



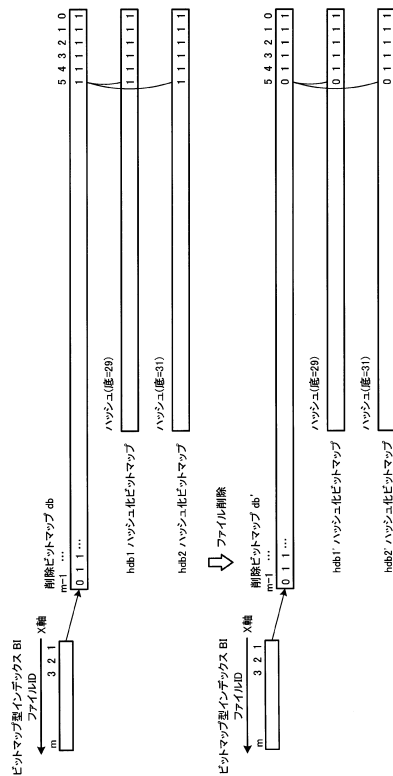
【図 2】

実施例1に係るビットマップ型インデックスの更新処理の一例を示す図



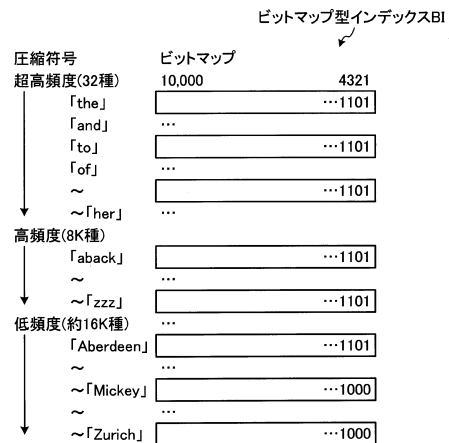
【図 3】

ファイル削除によるビットマップ型インデックスの更新処理の一例を示す図



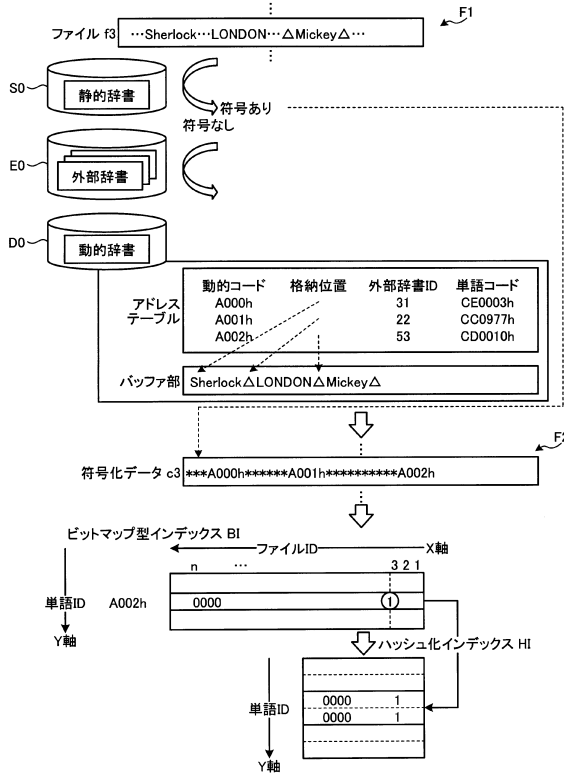
【图 4】

実施例1に係るビットマップ型インデックスの一例を示す図



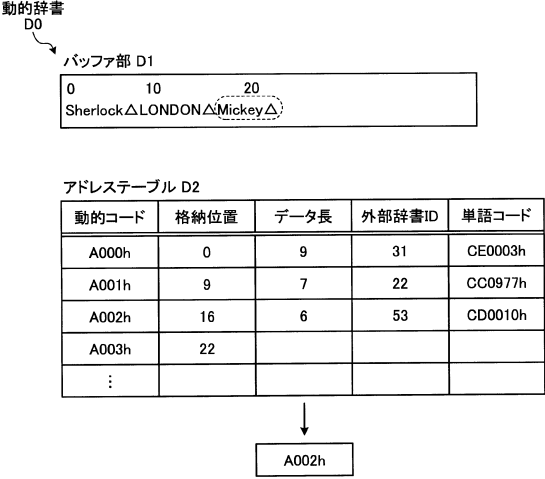
【図 5】

実施例1に係るビットマップ型インデックスの生成処理の一例を示す図



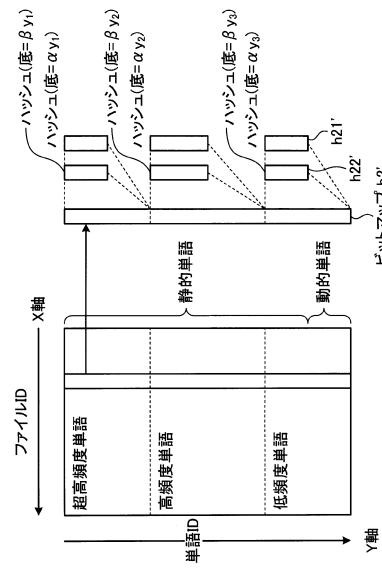
【図 6】

実施例1に係る動的辞書の一例を示す図



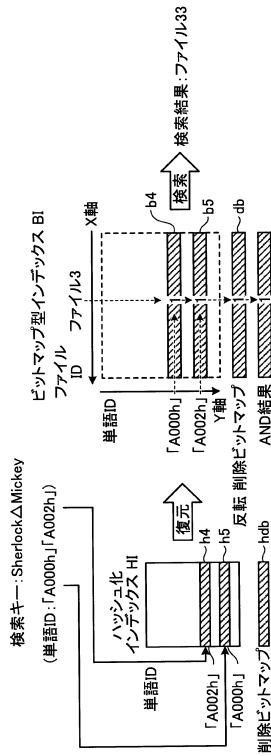
【図 7】

実施例1に係るビットマップ型インデックスのY軸方向のハッシュ化の一例を示す図

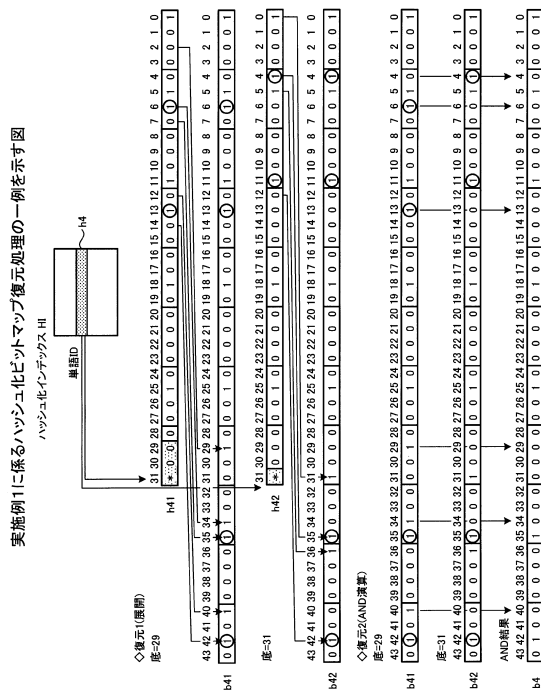


【図 8】

実施例1に係る検索処理の一例を示す図

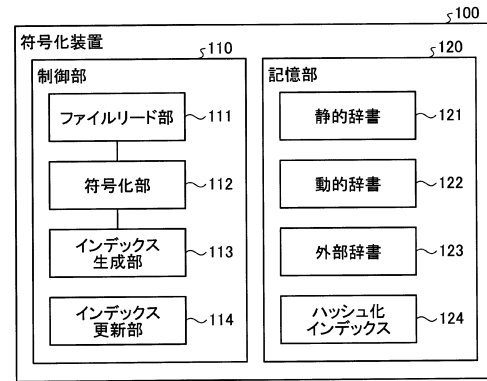


【図 9】



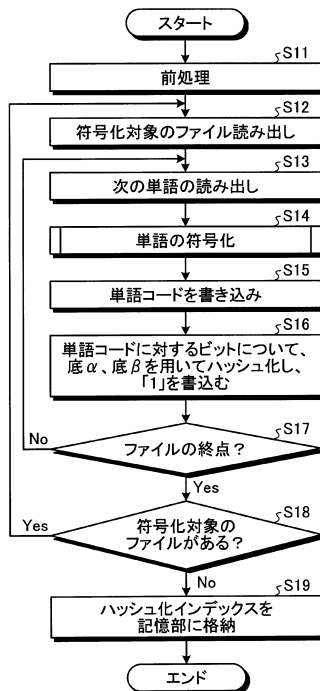
【図 10】

実施例1に係る符号化装置の構成を示す機能ブロック図



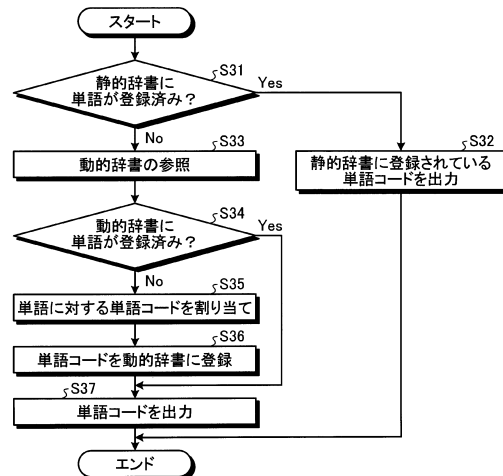
【図 11】

実施例1に係るインデックス生成処理のフローチャートの一例を示す図



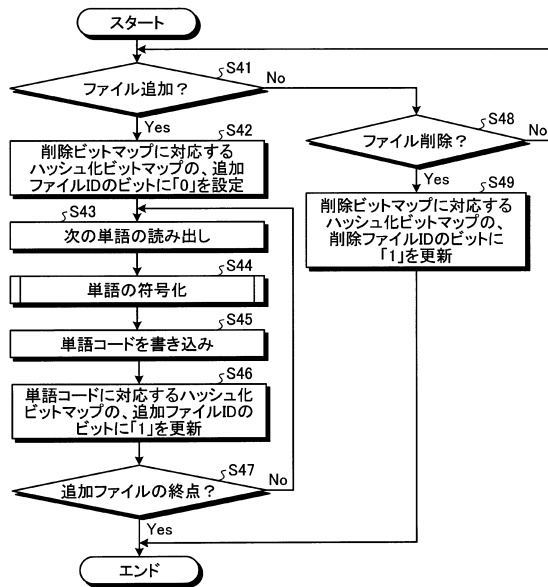
【図 12】

実施例1に係る符号化処理のフローチャートの一例を示す図



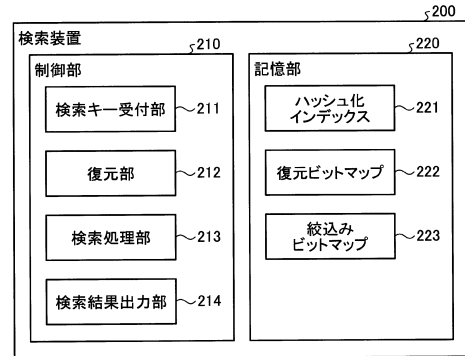
【図 13】

実施例1に係るインデックス更新処理のフローチャートの一例を示す図



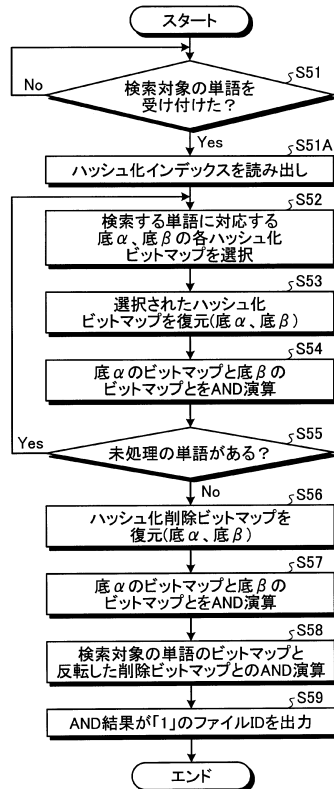
【図 14】

実施例1に係る検索装置の構成を示す機能ブロック図



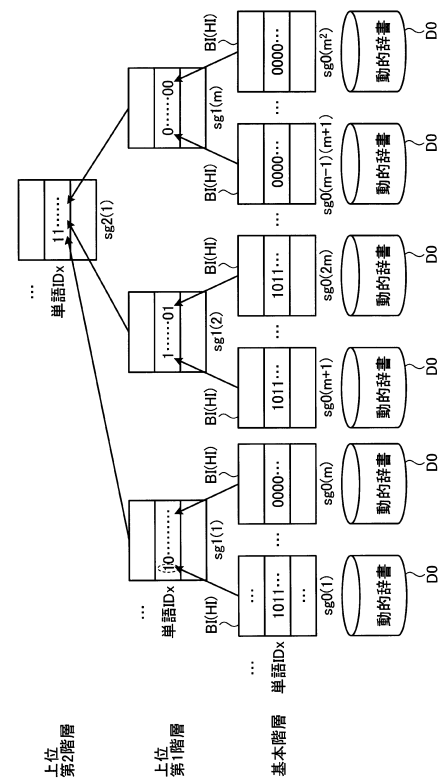
【図 15】

実施例1に係る検索処理のフローチャートの一例を示す図



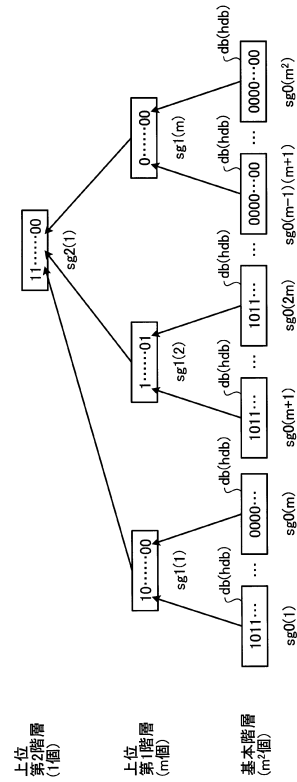
【図 16】

実施例2に係るビットマップ型インデックスの階層化の一例を示す図



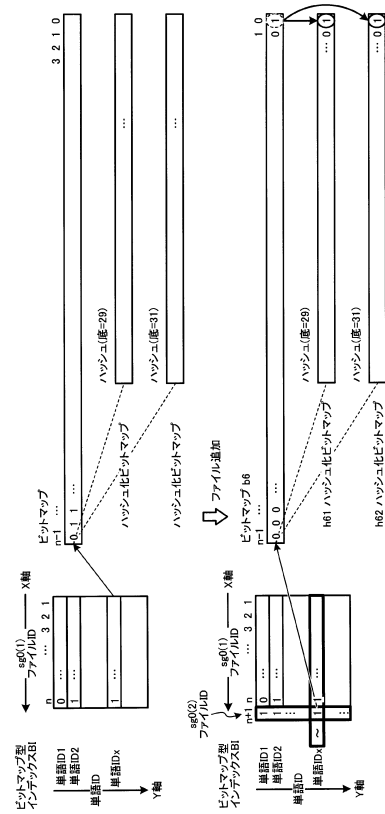
【図 17】

実施例2に係る削除ビットマップの階層化の一例を示す図



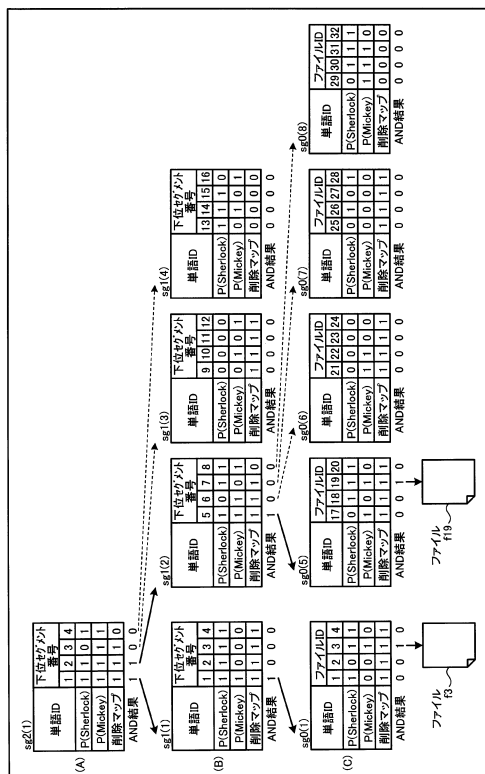
【図 18】

実施例2に係るビットマップ型インデックスの更新処理の一例を示す図



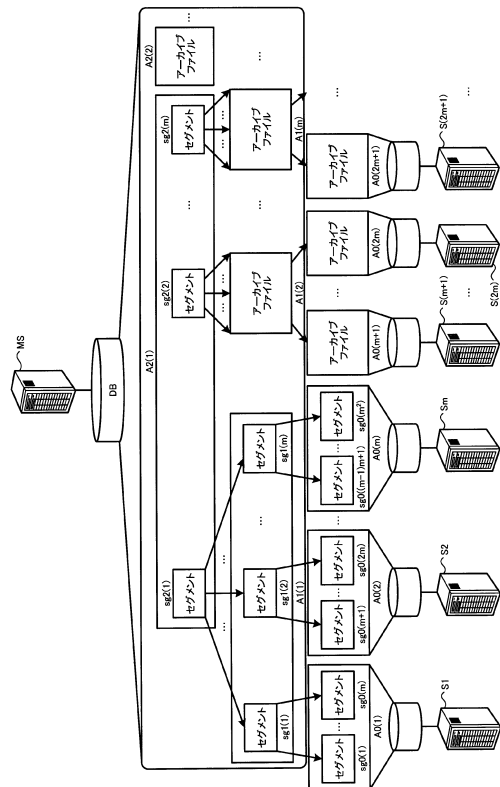
【図 19】

階層化されたセグメント群を用いた検索処理の一例を示す図

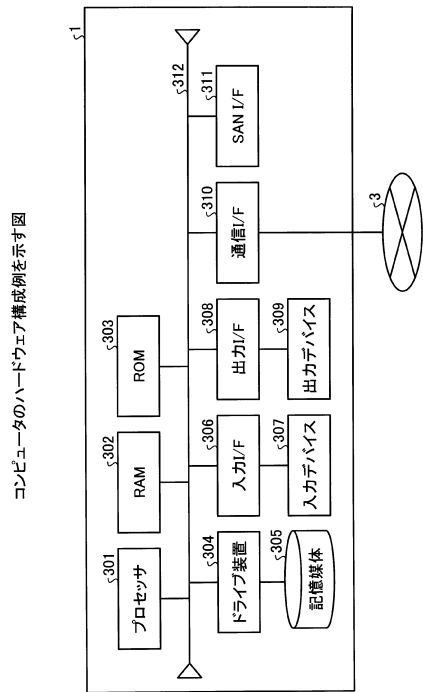


【図 20】

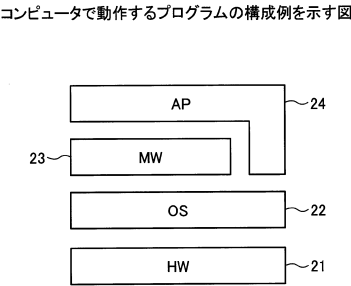
階層化されたセグメント群を実装したコンピュータシステムの構成例を示す図



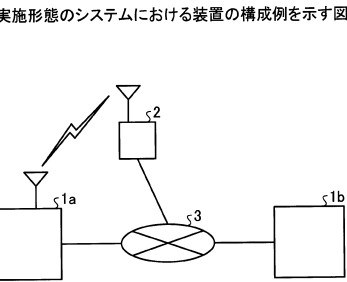
【図 2 1】



【図 2 2】



【図 2 3】



フロントページの続き

審査官 早川 学

(56)参考文献 国際公開第2013/038527 (WO, A1)
特開2008-269643 (JP, A)

(58)調査した分野(Int.Cl., DB名)
G06F 16/31