

(19) 世界知的所有権機関
国際事務局(43) 国際公開日
2009年7月9日 (09.07.2009)

PCT

(10) 国際公開番号
WO 2009/084145 A1

- (51) 国際特許分類:
G06F 17/30 (2006.01) G06F 12/00 (2006.01)
- (21) 国際出願番号: PCT/JP2008/003123
- (22) 国際出願日: 2008年10月30日 (30.10.2008)
- (25) 国際出願の言語: 日本語
- (26) 国際公開の言語: 日本語
- (30) 優先権データ:
特願 2007-338413
2007年12月28日 (28.12.2007) JP
- (71) 出願人 (米国を除く全ての指定国について): 株式会社エスグランツ (S.GRANTS CO., LTD.) [JP/JP]; 〒2610004 千葉県千葉市美浜区高洲三丁目5番3棟1210号 Chiba (JP).
- (72) 発明者: および
- (75) 発明者/出願人 (米国についてのみ): 新庄敏男 (SHINJO, Toshio) [JP/JP]; 〒2610004 千葉県千葉市

美浜区高洲三丁目5番3棟1210号株式会社エスグランツ内 Chiba (JP). 國分光裕 (KOKUBUN, Mitsuhiro) [JP/JP]; 〒2610004 千葉県千葉市美浜区高洲三丁目5番3棟1210号株式会社エスグランツ内 Chiba (JP).

(74) 代理人: ▲徳▼永民雄 (TOKUNAGA, Tamio); 〒2020012 東京都西東京市東町3丁目2番19号 Tokyo (JP).

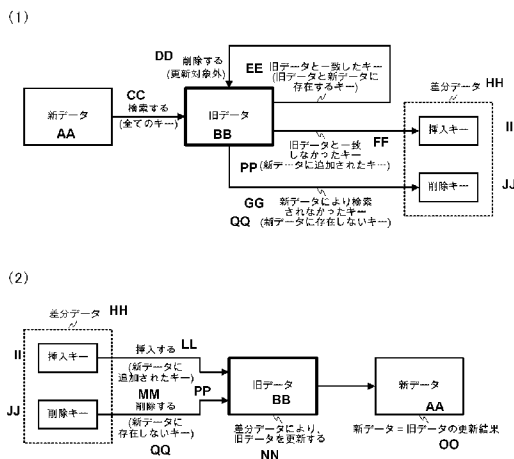
(81) 指定国 (表示のない限り、全ての種類の国内保護が可能): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

[続葉有]

(54) Title: DATABASE INDEX KEY UPDATE METHOD AND PROGRAM

(54) 発明の名称: データベースのインデックスキー更新方法及びプログラム

[図3A]



AA NEW DATA
BB OLD DATA
CC SEARCH (ALL THE KEYS)
DD DELETE (OTHER THAN THOSE TO BE UPDATED)
EE KEY MATCHED WITH OLD DATA (KEY EXISTING IN OLD DATA AND NEW DATA)
FF KEY NOT MATCHED WITH OLD DATA
GG KEY NOT SEARCHED BY NEW DATA
PP (KEY ADDED TO NEW DATA)
QQ (KEY NOT EXISTING IN NEW DATA)
HH DIFFERENCE DATA
II INSERT KEY
JJ DELETE KEY
LL INSERT
MM DELETE
NN UPDATE OLD DATA BY DIFFERENCE DATA
OO NEW DATA = OLD DATA UPDATE RESULT

(57) Abstract: When an index key is supplied for a new database to replace an index key of an existing database, it is possible to effectively update the index key of the database. An index key of new data is used to perform deletion from a coupled node tree containing an index key of old data as a leaf node index key and to create difference data between the new data formed by an insert key and a delete key and the old data. New data is created by executing deletion and insertion of the delete key and the insert key extracted from the difference data on the coupled node tree containing the index key of the old data as the leaf node index key.

(57) 要約: 既存のデータベースのインデックスキーに対してそれと置き換えられる新しいデータベース用のインデックスキーが供給されたときに、効率よくデータベースのインデックスキーの更新を行うことを可能とする。旧データのインデックスキーをリーフノードのインデックスキーとして格納したカップルドノードツリーから新データのインデックスキーにより削除処理を行って挿入キーと削除キーからなる新データと旧データの差分データを作成し、旧データのインデックスキーをリーフノードのインデックスキーとして格納したカップルドノードツリーに、差分データから取り出した削除キーと挿入キー削除処理と挿入処理を行って新データを作成する。



(84) 指定国 (表示のない限り、全ての種類の広域保護が可能): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), ヨーロッパ (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, NO, PL, PT, RO, SE,

SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

添付公開書類:
— 国際調査報告書

明 細 書

データベースのインデックスキー更新方法及びプログラム

技術分野

- [0001] 本発明は、データベースのインデックスキーの更新方法、及びそれに関連した、差分データ作成方法と装置、インデックスキーの差分データ更新方法と装置、さらにプログラムに関する。

背景技術

- [0002] 近年、社会の情報化が進展し、大規模なデータベースが各所で利用されるようになってきている。このような大規模なデータベースからレコードを検索するには、各レコードの記憶されたアドレスと対応づけられたレコード内の項目をインデックスキーとして検索をし、所望のレコードを探し出すことが通例である。また、全文検索における文字列も、文書のインデックスキーと見なすことができる。

そして、それらのインデックスキーはビット列で表現されることから、データベースの検索はビット列の検索に帰着されるということができる。

- [0003] 本発明の背景技術の１つとして、まず上述のビット列の検索処理について説明する。

ビット列の検索処理手法については、種々のものが知られている。それらのなかでも、ビット列の検索を高速に行うために、ビット列を記憶するデータ構造を種々に工夫することが従来から行われている。このようなものの一つとして、パトリシアツリーという木構造が知られている。

- [0004] 図１は、上述の従来の検索処理に用いられているパトリシアツリーの一例を示すものである。パトリシアツリーのノードは、インデックスキー、検索キーの検査ビット位置、左右のリンクポインタを含んで構成される。明示はされていないが、ノードにはインデックスキーに対応するレコードにアクセスするための情報が含まれていることは勿論である。

- [0005] 図１の例では、インデックスキー“１０００１０”を保持するノード１７

50aがルートノードとなっており、その検査ビット位置は0である。ノード1750aの左リンク1740aにはノード1750bが接続され、右リンク1741aにはノード1750fが接続されている。

ノード1750bの保持するインデックスキーは“010011”であり、検査ビット位置2030bは1である。ノード1750bの左リンク1740bにはノード1750cが、右リンク1741bにはノード1750dが接続されている。ノード1750cが保持するインデックスキーは“000111”、検査ビット位置は3である。ノード1750dが保持するインデックスキーは“011010”、検査ビット位置は2である。

[0006] ノード1750cから実線で接続された部分はノード1750cの左右のリンクポインタを示すものであり、点線の接続されていない左ポインタ1740cは、その欄が空欄であることを示している。点線の接続された右ポインタ1741cの点線の接続先は、ポインタの示すアドレスを表しており、今の場合ノード1750cを右ポインタが指定していることを表している。

ノード1750dの右ポインタ1741dはノード1750d自身を指しており、左リンク1740dにはノード1750eが接続されている。ノード1750eの保持するインデックスキーは“010010”、検査ビット位置は5である。ノード1750eの左ポインタ1740eはノード1750bを、右ポインタ1741eはノード1750eを指している。

[0007] また、ノード1750fの保持するインデックスキーは“101011”であり、検査ビット位置1730fは2である。ノード1750fの左リンク1740fにはノード1750gが、右リンク1741fにはノード1750hが接続されている。

ノード1750gの保持するインデックスキーは“100011”であり、検査ビット位置1730gは5である。ノード1750gの左ポインタ1740gはノード1750aを、右ポインタ1741gはノード1750gを指している。

ノード1750hの保持するインデックスキーは“101100”であり

、検査ビット位置 1730h は 3 である。ノード 1750h の左ポインタ 1740h はノード 1750f を、右ポインタ 1741h はノード 1750h を指している。

[0008] 図 1 の例では、ルートノード 1750a からツリーを降りるにしたがって、各ノードの検査ビット位置が大きくなるように構成されている。

ある検索キーで検索を行うとき、ルートノードから順次各ノードに保持される検索キーの検査ビット位置を検査していき、検査ビット位置のビット値が 1 であるか 0 であるか判定を行い、1 であれば右リンクをたどり、0 であれば左リンクをたどる。そして、リンク先のノードの検査ビット位置がリンク元のノードの検査ビット位置より大きくなければ、すなわち、リンク先が下方でなく上方に戻れば（図 1 において点線で示されたこの逆戻りのリンクをバックリンクという）、リンク先のノードのインデックスキーと検索キーの比較を行う。比較の結果、等しければ検索成功であり、等しくなければ検索失敗であることが保証されている。

[0009] 上記のように、パトリシアツリーを用いた検索処理では、必要なビットの検査だけで検索できること、キー全体の比較は 1 回ですむことなどのメリットがあるが、各ノードからの 2 つのリンクが必ずあることにより記憶容量が増大することや、バックリンクの存在による判定処理の複雑化、バックリンクにより戻ることによって初めてインデックスキーと比較することによる検索処理の遅延及び追加削除等データメンテナンスの困難性などの欠点がある。

[0010] これらのパトリシアツリーの欠点を解消しようとするものとして、例えば下記特許文献 1 に開示された技術がある。下記特許文献 1 に記載されたパトリシアツリーにおいては、下位の左右のノードは連続した領域に記憶することによりポインタの記憶容量を削減するとともに、次のリンクがバックリンクであるか否かを示すビットを各ノードに設けることにより、バックリンクの判定処理を軽減している。

しかしながら、下記特許文献 1 に開示されたものにおいても、1 つのノードは必ずインデックスキーの領域とポインタの領域を占めること、下位の左

右のノードを連続した領域に記憶するようにしてポインタを１つとしたため、例えば図１に示したパトリシアツリーの最下段の部分である左ポインタ１７４０c、右ポインタ１７４１h等の部分にもノードと同じ容量の記憶領域を割り当てる必要があるなど、記憶容量の削減効果はあまり大きいものではない。また、バックリンクによる検索処理の遅延の問題や追加削除等の処理が困難であることも改善されていない。

[0011] 次に、本発明の背景技術の別の１つとして、データベースの更新処理について説明する。

データベースシステムの機能として、データの更新機能は必須である。データベースに保管されるデータ量が増大するのに伴って、例えばバッチ処理により既存のデータベースに対して大量のデータの追加や削除を行おうとすると、その作業に要する時間が長くなるという不都合が生じている。

データベースのバッチ更新の態様としては、既存のデータベースに対する追加、変更あるいは削除データによりデータベースを更新する態様と、既存のデータベースを新しいデータベースでそっくり置き換える態様が存在する。

[0012] 後者のデータベース更新の態様は、例えばデータベースのデータの供給者が、データベースの更新が必要になったときにデータベースの更新データを供給するのではなく、データを更新済みの新しい版のデータベースを改めて供給する場合に採用される。しかし、このようなデータベースの更新の態様では、更新作業中はデータベースの利用ができないことから、この更新作業に長時間を要するのでは不便である。

また、データベースは、データベース本体のデータ部分とデータベース本体からデータを検索するためのインデックスから構成されるのが通例である。そこで、データベースの更新の態様には、インデックス部分を更新する態様があり、この部分の更新においても、更新後のインデックス全体が供給され、更新前のインデックスとそっくり入れ替える場合がある。

[0013] 例えばカーナビゲーションシステムの地図データのインデックスを更新す

る場合には、地図データの販売業者から購入した新しい地図データのインデックスデータを、カーナビゲーションシステムの販売業者あるいは車のディーラーのセンターから、カーナビゲーションシステムを搭載した各車に配布し、搭載されたカーナビゲーションシステムごとに、インデックスデータを更新する。このように、新しいインデックスデータ全体を配布するため、配布するデータ量が多く、また各カーナビゲーションシステムにおける地図データの更新時間も長いものとなる。

特許文献1：特開2001-357070号公報

図面の簡単な説明

- [0014] [図1]従来の検索で用いられるパトリシアツリーの一例を示す図である。
- [図2A]配列に格納されたカップルドノードツリーの構成例を説明する図である。
- [図2B]カップルドノードツリーのツリー構造を概念的に示す図である。
- [図3A]本発明の原理を説明する図である。
- [図3B]本発明を実施するためのハードウェア構成例を説明する図である。
- [図3C]差分データ作成装置の機能ブロック構成を説明する図である。
- [図3D]差分データ更新装置の機能ブロック構成を説明する図である。
- [図4]ビット列検索の基本動作を示したフローチャートである。
- [図5]カップルドノードツリーに格納されたインデックスキーの最小値を求める処理を示したフローチャートである。
- [図6]カップルドノードツリーに格納されたインデックスキーを昇順に取り出す処理を示したフローチャートである。
- [図7A]挿入処理の前段である検索処理の処理フローを示す図である。
- [図7B]挿入するノード対のための配列要素を準備する処理を説明する処理フロー図である。
- [図7C]ノード対を挿入する位置を求め、ノード対の各ノードの内容を書き込んで挿入処理を完成させる処理フローを示す図である。
- [図7D]ルートノードの挿入処理を含むインデックスキーを追加する場合のノ

ード挿入処理全体を説明する処理フロー図である。

[図8A]削除処理の前段である検索処理の処理フローを示す図である。

[図8B]削除処理の後段の処理フローを説明する図である。

[図9]新旧データによる差分データ作成処理の概要を説明する図である。

[図10]新旧データによる差分データ作成の処理フローを説明する図である。

[図11]旧データによりカップルドノードツリーを作成する処理フローを説明する図である。

[図12]差分データに更新データを格納する処理フローを説明する図である。

[図13]差分ツリーに残ったキーを削除キーとして差分データを作成する処理フローを説明する図である。

[図14]差分データにより旧データを新データに更新する処理の概要を説明する図である。

[図15]差分データにより旧データを新データに更新する処理フローを説明する図である。

発明の開示

[0015] そこで、本発明の課題は、既存のデータベースのインデックスキーに対してそれと置き換えられる新しいデータベース用のインデックスキーが供給されたときに、効率よくデータベースのインデックスキーの更新を行うことができるデータベース更新処理の手法を提供することである。

[0016] 本出願人は、特願2006-187827において、ルートノードと、隣接した記憶領域に配置されるブランチノードとリーフノードまたはブランチノード同士またはリーフノード同士のノード対からなるビット列検索に用いるツリーであって、ルートノードはツリーの始点を表すノードであって、該ツリーのノードが1つのときはリーフノード、ツリーのノードが2つ以上のときは前記ブランチノードであり、前記ブランチノードは、ビット列検索を行う検索キーの弁別ビット位置とリンク先のノード対の一方のノードの位置を示す位置情報を含み、前記リーフノードは検索対象のビット列からなるインデックスキーを含むカップルドノードツリーを用いたビット列検索を提案

した。

上記出願においては、与えられたインデックスキーの集合からカップルドノードツリーを生成する方法と、カップルドノードツリーから単一のインデックスキーを検索する手法等の、カップルドノードツリーを用いた基本的な検索手法が示されている。

[0017] また、ビット列の検索には、最小値、最大値を求める、ある範囲の値のものを求める等の各種の検索要求が存在する。そこで、本出願人は、特願2006-293619において、カップルドノードツリーの任意の部分木に含まれるインデックスキーの最大値／最小値を求める手法等を提案した。

本発明は、このカップルドノードツリーを応用した高速なデータベースのインデックスキー更新手法を実現することを目的とする。

[0018] 本発明によれば、インデックスキーの旧データと新データの差分データを作成し、差分データにより旧データを更新して新データを作成する。

その際、旧データあるいは新データのインデックスキーをリーフノードのインデックスキーとして格納したカップルドノードツリーである差分ツリーを取得し、差分ツリーから新データあるいは旧データのインデックスキーを検索キーとして検索を行い、検索キーと一致したインデックスキーは差分ツリーから削除し、旧データ中のインデックスキーと一致しない検索キーは差分データの挿入キーあるいは削除キーとし、新データあるいは旧データの全てのインデックスキーによる検索が終了した後に差分ツリーに残った旧データのインデックスキーを差分データの削除キーあるいは挿入キーとして差分データを作成する。

差分データによる旧データの新データへの更新においては、旧データのインデックスキーをリーフノードのインデックスキーとして格納したカップルドノードツリーである更新ツリーを取得し、差分データから取り出した削除キーを更新ツリーから削除し、差分データから取り出した挿入キーを更新ツリーに挿入することにより新データを作成する。

[0019] 本発明によれば、カップルドノードツリーを用いることにより、高速で差

分データを作成することができる。そして、インデックスキーを更新するサイトに配布するのは、新データ全体ではなく差分データなので、配布するデータ量を削減することができる。また、インデックスキーの更新にもカップルドノードツリーを用いることにより、高速なデータ更新が可能となる。

発明を実施するための最良の形態

[0020] 最初に、本出願人により先の上記出願において提案された、本発明で利用するカップルドノードツリーについて、カップルドノードツリーを配列に格納する例を説明する。ブランチノードが保持するリンク先の位置を示すデータとして、記憶装置のアドレス情報とすることもできるが、ブランチノードあるいはリーフノードのうち占有する領域の記憶容量の大きい方を格納可能な配列要素からなる配列を用いることにより、ノードの位置を配列番号で表すことができ、位置情報の情報量を削減することができる。

[0021] 図2Aは、配列に格納されたカップルドノードツリーの構成例を説明する図である。

図2Aを参照すると、ノード101が配列100の配列番号10の配列要素に配置されている。ノード101はノード種別102、弁別ビット位置103及び代表ノード番号104で構成されている。ノード種別102は0であり、ノード101がブランチノードであることを示している。弁別ビット位置103には1が格納されている。代表ノード番号104にはリンク先のノード対の代表ノードの配列番号20が格納されている。なお、以下では表記の簡略化のため、代表ノード番号に格納された配列番号を代表ノード番号ということもある。また、代表ノード番号に格納された配列番号をそのノードに付した符号あるいはノード対に付した符号で表すこともある。

[0022] 配列番号20の配列要素には、ノード対111の代表ノードであるノード[0]112が格納されている。そして隣接する次の配列要素（配列番号20+1）に代表ノードと対になるノード[1]113が格納されている。ノード[0]112のノード種別114には0が、弁別ビット位置115には3が、代表ノード番号116には30が格納されている。またノード[1]

1 1 3 のノード種別 1 1 7 には 1 が格納されており、ノード [1] 1 1 3 がリーフノードであることを示している。インデックスキー 1 1 8 には、“0 0 0 1” が格納されている。パトリシアツリーについて先に述べたと同様に、リーフノードにインデックスキーと対応するレコードにアクセスする情報が含まれることは当然であるが、表記は省略している。

[0023] なお、代表ノードをノード [0] で表し、それと対になるノードをノード [1] で表すことがある。また、ある配列番号の配列要素に格納されたノードを、その配列番号のノードということがあり、ノードの格納された配列要素の配列番号を、ノードの配列番号ということもある。

配列番号 3 0 及び 3 1 の配列要素に格納されたノード 1 2 2 とノード 1 2 3 からなるノード対 1 2 1 の内容は省略されている。

[0024] ノード [0] 1 1 2、ノード [1] 1 1 3、ノード 1 2 2、及びノード 1 2 3 の格納された配列要素にそれぞれ付された 0 あるいは 1 は、検索キーで検索を行う場合にノード対のどちらのノードにリンクするかを示すものである。前段のブランチノードの弁別ビット位置にある検索キーのビット値である 0 か 1 を代表ノード番号に加えた配列番号のノードにリンクする。

したがって、前段のブランチノードの代表ノード番号に、検索キーの弁別ビット位置のビット値を加えることにより、リンク先のノードが格納された配列要素の配列番号を求めることができる。

なお、上記の例では代表ノード番号をノード対の配置された配列番号のうち小さい方を採用しているが、大きいほうを採用することも可能であることは明らかである。

[0025] 図 2 B は、カップルドノードツリーのツリー構造を概念的に示す図である。図示の 6 ビットのインデックスキーは、図 1 に例示されたパトリシアツリーのものと同じである。

符号 2 1 0 a で示すのがルートノードである。図示の例では、ルートノード 2 1 0 a は配列番号 2 2 0 に配置されたノード対 2 0 1 a の代表ノードとしている。

ツリー構造としては、ルートノード 2 1 0 a の下にノード対 2 0 1 b が、その下層にノード対 2 0 1 c とノード対 2 0 1 f が配置され、ノード対 2 0 1 f の下層にはノード対 2 0 1 h とノード対 2 0 1 g が配置されている。ノード対 2 0 1 c の下にはノード対 2 0 1 d が、さらにその下にはノード対 2 0 1 e が配置されている。

各ノードの前に付された 0 あるいは 1 の符号は、図 2 A において説明した配列要素の前に付された符号と同じである。検索キーの弁別ビット位置のビット値に応じてツリーをたどり、検索対象のリーフノードを見つけることになる。

[0026] 図示された例では、ルートノード 2 1 0 a のノード種別 2 6 0 a は 0 でブランチノードであることを示し、弁別ビット位置 2 3 0 a は 0 を示している。代表ノード番号は 2 2 0 a であり、それはノード対 2 0 1 b の代表ノード 2 1 0 b の格納された配列要素の配列番号である。

ノード対 2 0 1 b はノード 2 1 0 b と 2 1 1 b で構成され、それらのノード種別 2 6 0 b、2 6 1 b はともに 0 であり、ブランチノードであることを示している。ノード 2 1 0 b の弁別ビット位置 2 3 0 b には 1 が格納され、リンク先の代表ノード番号にはノード対 2 0 1 c の代表ノード 2 1 0 c の格納された配列要素の配列番号 2 2 0 b が格納されている。

[0027] ノード 2 1 0 c のノード種別 2 6 0 c には 1 が格納されているので、このノードはリーフノードであり、したがって、インデックスキーを含んでいる。インデックスキー 2 5 0 c には “0 0 0 1 1 1” が格納されている。一方ノード 2 1 1 c のノード種別 2 6 1 c は 0、弁別ビット位置 2 3 1 c は 2 であり、代表ノード番号にはノード対 2 0 1 d の代表ノード 2 1 0 d の格納された配列要素の配列番号 2 2 1 c が格納されている。

ノード 2 1 0 d のノード種別 2 6 0 d は 0、弁別ビット位置 2 3 0 d は 5 であり、代表ノード番号にはノード対 2 0 1 e の代表ノード 2 1 0 e の格納された配列要素の配列番号 2 2 0 d が格納されている。ノード 2 1 0 d と対になるノード 2 1 1 d のノード種別 2 6 1 d は 1 であり、インデックスキー

2 5 1 dには“0 1 1 0 1 0”が格納されている。

ノード対2 0 1 eのノード2 1 0 e、2 1 1 eのノード種別2 6 0 e、2 6 1 eはともに1であり双方ともリーフノードであることを示し、それぞれのインデックスキー2 5 0 e、2 5 1 eにはインデックスキーとして“0 1 0 0 1 0”と“0 1 0 0 1 1”が格納されている。

[0028] ノード対2 0 1 bのもう一方のノードであるノード2 1 1 bの弁別ビット位置2 3 1 bには2が格納され、リンク先の代表ノード番号にはノード対2 0 1 fの代表ノード2 1 0 fの格納された配列要素の配列番号2 2 1 bが格納されている。

ノード対2 0 1 fのノード2 1 0 f、2 1 1 fのノード種別2 6 0 f、2 6 1 fはともに0であり双方ともブランチノードである。それぞれの弁別ビット位置2 3 0 f、2 3 1 fには5、3が格納されている。ノード2 1 0 fの代表ノード番号にはノード対2 0 1 gの代表ノード2 1 0 gの格納された配列要素の配列番号2 2 0 fが格納され、ノード2 1 1 fの代表ノード番号にはノード対2 0 1 hの代表ノードであるノード[0] 2 1 0 hの格納された配列要素の配列番号2 2 1 fが格納されている。

ノード対2 0 1 gのノード2 1 0 g、2 1 1 gのノード種別2 6 0 g、2 6 1 gはともに1であり双方ともリーフノードであることを示し、それぞれのインデックスキー2 5 0 g、2 5 1 gには“1 0 0 0 1 0”と“1 0 0 0 1 1”が格納されている。

また同じくノード対2 0 1 hの代表ノードであるノード[0] 2 1 0 hとそれと対をなすノード[1] 2 1 1 hのノード種別2 6 0 h、2 6 1 hはともに1であり双方ともリーフノードであることを示し、それぞれのインデックスキー2 5 0 h、2 5 1 hには“1 0 1 0 1 1”と“1 0 1 1 0 0”が格納されている。

[0029] 以下、上述のツリーからインデックスキー“1 0 0 0 1 0”を検索する処理の流れを簡単に説明する。弁別ビット位置は、左から0、1、2、・・・とする。

まず、ビット列“100010”を検索キーとしてルートノード210aから処理をスタートする。ルートノード210aの弁別ビット位置230aは0であるので、検索キー“100010”の弁別ビット位置が0のビット値をみると1である。そこで代表ノード番号の格納された配列番号220aに1を加えた配列番号の配列要素に格納されたノード211bにリンクする。ノード211bの弁別ビット位置231bには2が格納されているので、検索キー“100010”の弁別ビット位置が2のビット値をみると0であるから、代表ノード番号の格納された配列番号221bの配列要素に格納されたノード210fにリンクする。

ノード210gのノード種別260gは1でありリーフノードであることを示しているので、インデックスキー250gを読み出して検索キーと比較すると両方とも“100010”であって一致している。このようにしてカップルドノードツリーを用いた検索が行われる。

[0030] 次に、図2Bを参照してカップルドノードツリーの構成の意味について説明する。

カップルドノードツリーの構成はインデックスキーの集合により規定される。図2Bの例で、ルートノード210aの弁別ビット位置が0であるのは、図2Bに例示されたインデックスキーに0ビット目が0のものと1のものがあるからである。0ビット目が0のインデックスキーのグループはノード210bの下に分類され、0ビット目が1のインデックスキーのグループはノード211bの下に分類されている。

ノード211bの弁別ビット位置が2であるのは、ノード211h、210h、211g、210gに格納された0ビット目が1のインデックスキーの1ビット目がすべて0で等しく、2ビット目で初めて異なるものがあるという、インデックスキーの集合の性質を反映している。

[0031] 以下0ビット目の場合と同様に、2ビット目が1であるものはノード211f側に分類され、2ビット目が0であるものはノード210f側に分類される。

そして2ビット目が1であるインデックスキーは3ビット目の異なるものがあるのでノード2 1 1 fの弁別ビット位置には3が格納され、2ビット目が0であるインデックスキーでは3ビット目も4ビット目も等しく5ビット目で異なるのでノード2 1 0 fの弁別ビット位置には5が格納される。

ノード2 1 1 fのリンク先においては、3ビット目が1のものと0のものがそれぞれ1つしかないことから、ノード2 1 0 h、2 1 1 hはリーフノードとなり、それぞれインデックスキー2 5 0 hと2 5 1 hに“1 0 1 0 1 1”と“1 0 1 1 0 0”が格納されている。

[0032] 仮にインデックスキーの集合に“1 0 1 1 0 0”の代わりに“1 0 1 1 0 1”か“1 0 1 1 1 0”が含まれていたとしても、3ビット目までは“1 0 1 1 0 0”と等しいので、ノード2 1 1 hに格納されるインデックスキーが変わるだけで、ツリー構造自体は変わることはない。しかし、“1 0 1 1 0 0”に加えて“1 0 1 1 0 1”が含まれていると、ノード2 1 1 hはブランチノードとなり、その弁別ビット位置は5になる。追加されるインデックスキーが“1 0 1 1 1 0”であれば、弁別ビット位置は4となる。

[0033] 以上説明したように、カップルドノードツリーの構造は、インデックスキーの集合に含まれる各インデックスキーの各ビット位置のビット値により決定される。

そしてさらにいえば、異なるビット値となるビット位置ごとにビット値が“1”のノードとビット値が“0”のノードとに分岐していることから、ノード[1]側とツリーの深さ方向を優先させてリーフノードをたどると、それらに格納されたインデックスキーは、ノード2 1 1 hのインデックスキー2 5 1 hの“1 0 1 1 0 0”、ノード2 1 0 hのインデックスキー2 5 0 hの“1 0 1 0 1 1”、・・・、ノード2 1 0 cのインデックスキー2 5 0 cの“0 0 0 1 1 1”となり降順にソートされている。

すなわち、カップルドノードツリーにおいては、インデックスキーはソートされてツリー上に配置されている。

[0034] 検索キーで検索するときはインデックスキーがカップルドノードツリー上

に配置されたルートをたどることになり、例えば検索キーが“101100”であればノード211hに到達することができる。また、上記説明からも想像がつくように、“101101”か“101110”を検索キーとした場合でもノード211hにたどり着き、インデックスキー251hと比較することにより検索が失敗したことが分かる。

また、例えば“100100”で検索した場合でも、ノード210a、211b、210fのリンク経路では検索キーの3ビット目と4ビット目は使われることがなく、“100100”の5ビット目が0なので、“100010”で検索した場合と同様にノード210gに到達することになる。このように、カップルドノードツリーに格納されたインデックスキーのビット構成に応じた弁別ビット位置を用いて分岐が行われる。

[0035] 次に、本発明の原理を説明する。

図3Aは、本発明の原理を模式的に示す図である。図3Aの(1)に示すのは、旧データと新データから差分データを作成する原理を説明する図であり、図3Aの(2)に示すのは、差分データにより旧データを新データに更新する原理を説明する図である。

[0036] 図3Aの(1)に示すように、差分データを作成するには、新データに含まれる全てのキーで旧データに含まれるキーを検索する。旧データと一致したキー、すなわち旧データと新データに存在するキーについては、更新の対象外であるから、旧データから削除する。

旧データとは一致しなかった検索キー、すなわち新データに追加されたキーは、挿入キーとして差分データに取り込まれる。新データにより検索されず旧データに残存したキー、すなわち新データに存在しないキーは、削除キーとして差分データに取り込まれる。

なお、上述の説明では、新データに含まれる全てのキーで旧データに含まれるキーを検索する、としたが、差分データを作成するのであるから、旧データに含まれる全てのキーで新データに含まれるキーを検索する、として差分データを作成してもよいことは明らかである。その場合は、新データに残

存したキーが挿入キーとして差分データに取り込まれ、新データと一致しなかった旧データが削除キーとして差分データに取り込まれる。

[0037] 図3Aの(2)に示すように、差分データにより旧データを新データに更新するには、挿入キーを新データに追加されたキーとして旧データに挿入し、削除キーにより新データに存在しないキーを旧データから削除する。

[0038] 図3Bは、本発明を実施するためのハードウェア構成例を説明する図である。図に示すように、本発明を実施するためのハードウェアは、差分データ作成装置300と差分データ更新装置400a~400xで構成される。差分データ更新装置400a~400xの数は任意である。

[0039] 本発明の差分データ作成装置300による差分データ作成は中央処理装置302及びキャッシュメモリ303を少なくとも備えたデータ処理装置301によりデータ格納装置308を用いて実施される。カップルドノードツリーが配置される配列309と検索中にたどるノードが格納された配列要素の配列番号を記憶する探索経路スタック310、旧データを格納する旧データ格納領域321、新データを格納する新データ格納領域322及び作成済みの差分データを格納する差分データ格納領域320を有するデータ格納装置308は、主記憶装置305または外部記憶装置306で実現することができ、あるいは通信装置307を介して接続された遠方に配置された装置を用いることも可能である。

[0040] 図3Bの例示では、主記憶装置305、外部記憶装置306及び通信装置307が一本のバス304によりデータ処理装置301に接続されているが、接続方法はこれに限るものではない。また、主記憶装置305をデータ処理装置301内のものとすることもできるし、探索経路スタック310を中央処理装置302内のハードウェアとして実現することも可能である。あるいは、旧データ格納領域321、新データ格納領域322及び差分データ格納領域320は外部記憶装置306に、探索経路スタック310を主記憶装置305に持つなど、使用可能なハードウェア環境、インデックスキー集合の大きさ等に応じて適宜ハードウェア構成を選択できることは明らかである。

。

また、特に図示されてはいないが、処理の途中で得られた各種の値を後の処理で用いるためにそれぞれの処理に応じた一時記憶装置が用いられることは当然である。

[0041] 差分データ作成装置 300 において作成された差分データは、差分データ更新装置 400 a ~ 400 x に送られ、それぞれの差分データ格納領域 420 a ~ 420 x に格納され、それぞれのインデックス格納領域 421 a ~ 421 x に格納された旧データを新データに更新するために用いられる。特に図示はしていないが、差分データ更新装置 400 a ~ 400 x においても、データ処理装置や、差分データ格納領域、インデックス格納領域及びその他の記憶領域を有するデータ格納装置が備えられている。

なお、以下においては、上述の例えば差分データ格納領域 320 に格納された差分データを差分データ 320 と表記するように、あるデータ格納領域に格納されるデータ自体にデータ格納領域の符号を付して説明する場合がある。

[0042] 図 3 C は、差分データ作成装置 300 の機能ブロック構成を説明する図である。

図に示すように、差分データ作成装置 300 は、差分ツリー取得手段 330 と差分データ作成手段 331 の機能ブロックを含む。これらの機能ブロックは、図 3 B に例示するハードウェアと以下において説明する処理フローを実行するソフトウェアにより実現される。

差分ツリー取得手段 330 は、旧データあるいは新データのインデックスキーをカップルドノードツリーのリーフノードのインデックスキーとして格納した、差分ツリーを取得する。

差分データ作成手段 331 は、差分ツリーのルートノードを検索開始ノードとして新データあるいは旧データの全てのインデックスキーを検索キーとして検索を行い、検索キーと一致した検索結果キーであるインデックスキーは差分ツリーから削除し、検索結果キーであるインデックスキーと一致しな

い検索キーを差分データの挿入キーあるいは削除キーとする。そして、新データあるいは旧データの全てのインデックスキーによる検索が終了した後に差分ツリーに削除されずに残ったインデックスキーを差分データの削除キーあるいは挿入キーとして差分データを作成する。

[0043] 図3Dは、差分データ更新装置の機能ブロック構成を説明する図である。

図に示すように、差分データ更新装置400は、更新ツリー取得手段422と新旧インデックスキー更新手段423の機能ブロックを含む。これらの機能ブロックは、図3Bに示す差分データ更新装置400a～400xについて説明したハードウェアと以下において説明する処理フローを実行するソフトウェアにより実現される。

更新ツリー取得手段422は、旧データのインデックスキーをカップルドノードツリーのリーフノードのインデックスキーとして格納した更新ツリーを取得する。

新旧インデックスキー更新手段423は、差分データ作成手段331が作成した差分データから取り出した削除キーを更新ツリーから削除し、差分データから取り出した挿入キーを更新ツリーに挿入することにより新データのインデックスキーをリーフノードのインデックスキーとして格納した新データカップルドノードツリーを作成し、新データカップルドノードツリーに基づいて、旧データを新データに更新する。

[0044] 次に、上述の出願において、本出願人により提案されたカップルドノードツリーを用いた基本的な検索処理、カップルドノードツリーにおける挿入削除処理、及びカップルドノードツリーに含まれるインデックスキーの最小値を求める処理等の応用処理の一部について、本発明を理解するために必要な範囲で紹介する。

[0045] 図4は、本出願人による出願である上記特願2006-293619で提案されたビット列検索の基本動作を示したフローチャートである。

まず、ステップS401で、検索開始ノードの配列番号を取得する。取得された配列番号に対応する配列は、カップルドノードツリーを構成する任意

のノードを格納したものである。検索開始ノードの指定は、後に説明する各種応用検索において行われる。

取得された検索開始ノードの配列番号は、図示しない検索開始ノード設定エリアに設定されるが、この検索開始ノード設定エリアは、先に述べた「処理の途中で得られた各種の値を後の処理で用いるためにそれぞれの処理に応じた一時記憶装置」の一つである。以下の説明では、「図示しない検索開始ノード設定エリアに設定する」のような表現に変えて、「検索開始ノードの配列番号を得る。」、「検索開始ノードとして設定する」あるいは単に「検索開始ノードに設定する」のように記述することもある。

[0046] 次に、ステップS 4 0 2で、探索経路スタック3 1 0に取得された配列番号を格納し、ステップS 4 0 3で、その配列番号に対応する配列要素を参照すべきノードとして読み出す。そして、ステップS 4 0 4で、読み出したノードから、ノード種別を取り出し、ステップS 4 0 5で、ノード種別がブランチノードであるか否かを判定する。

ステップS 4 0 5の判定において、読み出したノードがブランチノードである場合は、ステップS 4 0 6に進み、ノードから弁別ビット位置についての情報を取り出し、更に、ステップS 4 0 7で、取り出した弁別ビット位置に対応するビット値を検索キーから取り出す。そして、ステップS 4 0 8で、ノードから代表ノード番号を取り出して、ステップS 4 0 9で、検索キーから取り出したビット値と代表ノード番号とを加算し、新たな配列番号として、ステップS 4 0 2に戻る。

[0047] 以降、ステップS 4 0 5の判定においてリーフノードと判定されてステップS 4 1 0に進むまで、ステップS 4 0 2からステップS 4 0 9までの処理を繰り返す。ステップS 4 1 0で、リーフノードからインデックスキーを取り出して、処理を終了する。

[0048] 図5は、本出願人による出願である上記特願2 0 0 6－2 9 3 6 1 9で提案されたカップルドノードツリー（部分木を含む）に格納されたインデックスキーの最小値を求める処理を示したフローチャートである。先に述べたよ

うなインデックスキーのツリー上の配置から、インデックスキーの最小値を求める処理は検索開始ノードからノード〔0〕をリーフノードに至るまでツリー上でたどることに相当する。

[0049] まず、ステップS 5 0 1の検索開始ノードの配列番号の取得からステップS 5 0 5のノード種別の判定までは、それぞれ上述の図4のステップS 4 0 1からステップS 4 0 5の処理と同様である。

ステップS 5 0 5のノード種別の判定においてノード種別がブランチノードと判定されると、ステップS 5 0 6に進み、ノードから配列の代表ノード番号を取り出し、ステップS 5 0 7で、取り出した代表ノード番号に値“0”を加算してその結果を新たな配列番号とし、ステップS 5 0 2に戻る。以降、ステップS 5 0 5においてそのノードがリーフノードと判定されるまでステップS 5 0 2からステップS 5 0 7までの処理を繰り返し、ステップS 5 0 8で、リーフノードからインデックスキーを取り出し、処理を終了する。

[0050] 上記の図5に示す処理においては、ノード〔0〕をたどるため、代表ノード番号に一律“0”を加算している。すなわち、図5の処理によれば、リンク先のノードは、ノード対のうち必ずノード〔0〕とし、より小さい値のインデックスキーが格納されているノードのほうに分岐している。これにより、ツリー構造が先に述べたように順列構成であるカップルドノードツリーの最小のインデックスキーを取り出すことができる。

なお、図示はしていないが、図5のステップS 5 0 7において、代表ノード番号に値“0”を加算する代わりに値“1”を加算すればノード〔1〕をたどることになり、カップルドノードツリーの最大のインデックスキーを取り出すことができる。

[0051] 図6は、カップルドノードツリーに格納されたインデックスキーを昇順に取り出す処理を示したフローチャートである。インデックスキーを昇順に取り出す処理は、ノード対をなすノードのうちノード〔0〕側及びツリーの深さを優先させてノードから順次リーフノードをたどり、各リーフノードから

インデックスキーを取り出していくことに相当する。

- [0052] まず、ステップS 6 0 1で、検索開始ノードの配列番号にルートノードの配列番号を設定し、ステップS 6 0 2で、上記図5を参照して説明した最小のインデックスキーを求める処理を実行して、カップルドノードツリーに含まれるインデックスキーの中で最小のインデックスキーを取得する。そして、ステップS 6 0 3で、取得したインデックスキーを取り出し、ステップS 6 0 4に進む。
- [0053] ステップS 6 0 4で、探索経路スタック3 1 0のスタックポインタ（以下、ポインタということもある。）がルートノードの配列番号を指しているか否かを判定する。スタックポインタが指す配列番号がルートノード以外である場合は、ステップS 6 0 5に進む。そして、ステップS 6 0 5で、探索経路スタック3 1 0からスタックポインタの指す配列番号を取り出してから、スタックポインタの値を1減らす。
- [0054] ステップS 6 0 6で、ステップS 6 0 5で取り出した配列番号から、その配列番号のノードがノード対のどちらの配列要素に格納されているかのノード位置を得る。例えばノード[0]については偶数番号の配列の配列要素に格納する等により、配列番号からノード位置を求めることができる。そして、ステップS 6 0 7で、ステップS 6 0 6で得たノード位置がノード[1]側であるか否かを判定する。ステップS 6 0 7においてノード[1]側と判定された場合は、ステップS 6 0 4に戻り、スタックポインタの指す配列番号のノードがノード[0]あるいはルートノードとなるまで、ステップS 6 0 4からステップS 6 0 7までの処理を繰り返す。
- [0055] ステップS 6 0 7においてノード[0]側と判定されると、ステップS 6 0 8に進み、配列番号に1を加算し、そのノードと対をなすノード[1]の配列番号を得る。そして、ステップS 6 0 9で、検索開始ノードにステップS 6 0 8で得たノード[1]の配列番号を設定し、ステップS 6 1 0で、検索開始ノードをルートノードとする部分木の最小のインデックスキーを求める処理を実行する。ステップS 6 1 0の処理は、ステップS 6 0 2と同様で

あり、図5に示す最小値検索処理が用いられる。

ステップS610で最小のインデックスキーを求めると、ステップS603に戻り、求めたインデックスキーを取り出し、以降ステップS604においてポインタがルートノードの配列番号を指していると判定されるまで、同様の処理を繰り返す。

[0056] このように、探索経路スタック310のポインタが指す配列番号を参照し、探索経路スタック310に格納されている配列番号のノード[0]と対をなすノード[1]について、そのノードを検索開始ノードとしてそのノード配下で最小のインデックスキーを求める。ステップS602で最小値が求められた段階においては、探索経路スタック310のポインタは、カップルドノードツリーの最小のインデックスキーを含むノードの配列番号を指している。続いて探索経路スタック310のポップ動作を行い、取り出した配列番号のノードのうちノード[1]を検索開始ノードとして該検索開始ノードをルートノードとする部分木のインデックスキーの最小値を求める処理を行い、探索経路スタック310のポップ動作の結果カップルドノードツリーのルートノードの配列番号が取り出されるまでポップ動作と最小値検索処理を繰り返す。

[0057] 探索経路スタック310には、先にステップS602においてルートノード配下のノードについて最小値を求める処理を行っていることにより、リンク経路についての配列番号が順次格納されている。このため、探索経路スタック310のポインタを1減らして新たにポインタが指す配列番号のノードのうち、ノード[0]について、そのノードと対をなすノード[1]を求め、ノード[1]のノード配下について順次最小値検索の処理を行っていくと、インデックスキーは昇順に取り出されることとなる。

先に最小値検索を変形して最大値検索が可能であることを述べたのと同様に、最小値検索に代えて最大値検索を行い、ノード[1]の判定をノード[0]の判定に置き換える等により、インデックスキーを降順に取り出すことが可能であることは、当業者に明らかである。

[0058] 次に、図 7 A～図 7 Dにより本出願人による出願である上記特願 2 0 0 6-1 8 7 8 2 7で提案されたカップルドノードツリーにおけるノード挿入処理を説明する。図 7 A～図 7 Cが通常の挿入処理を説明するものであり、図 7 Dはルートノードの挿入処理を説明するものである。ルートノードの挿入処理と通常の挿入処理により、カップルドノードツリーが生成されることから、ノード挿入処理の説明はカップルドノードツリーの生成処理の説明でもある。

[0059] 図 7 Aは挿入処理の前段である検索処理の処理フローを示す図であり、図 4に示した検索処理において、ルートノードを検索開始ノードとし、挿入キーを検索キーとしたものに相当する。

[0060] ステップ S 7 0 1において、検索開始ノードの配列番号を設定するエリアにルートノードの配列番号を設定し、ステップ S 7 0 2において、検索キーに挿入キーを設定する。

次にステップ S 7 1 0において、図 4に示す検索処理を実行し検索結果のインデックスキーを得て、ステップ S 7 1 1に進む。

ステップ S 7 1 1において挿入キーとインデックスキーを比較し、等しければ挿入キーは既にカップルドノードツリーに存在するのであるから、挿入は失敗となり、処理を終了する。等しくなければ次の処理、図 7 Bのステップ S 7 1 2以下の処理に進む。

[0061] 図 7 Bは、挿入するノード対のための配列要素を準備する処理を説明する処理フロー図である。

ステップ S 7 1 2において、配列から空きのノード対を求め、そのノード対のうち代表ノードとなるべき配列要素の配列番号を取得する。

ステップ S 7 1 3に進み、挿入キーとステップ S 7 1 0で得たインデックスキーの大小を比較し、挿入キーが大きいときは値 1 を小さいときは値 0 のブール値を得る。

ステップ S 7 1 4に進み、ステップ S 7 1 2で得た代表ノードの配列番号にステップ S 7 1 3で得たブール値を加算した配列番号を得る。

ステップS 7 1 5に進み、ステップS 7 1 2で得た代表ノードの配列番号にステップS 7 1 3で得たブール値の論理否定値を加算した配列番号を得る。

- [0062] ステップS 7 1 4で得た配列番号は、挿入キーをインデックスキーとして持つリーフノードが格納される配列要素の配列番号であり、ステップS 7 1 5で得た配列番号は、そのリーフノードと対を成すブランチノードあるいはリーフノードが格納される配列要素のものである。

つまり、前段の検索処理で得られたリーフノードに格納されたインデックスキーと挿入キーの大小により、挿入されるノード対のうちどちらのノードに挿入キーを保持するリーフノードが格納されるかが決定される。

- [0063] 例えば図2Bのカップルドノードツリーに“0 1 1 0 1 1”を挿入する場合、検索結果のインデックスキーはノード2 1 1 dに格納された“0 1 1 0 1 0”になる。挿入キー“0 1 1 0 1 1”とノード2 1 1 dに格納されたインデックスキー“0 1 1 0 1 0”の大小比較によりブール値が求められ、今の例では挿入キーが大きいのでブール値1が得られ、挿入されるノード対の代表ノード番号に1を加えた配列要素に挿入キーを保持するリーフノードが格納される。一方、インデックスキー“0 1 1 0 1 0”は、大小比較で得られたブール値を論理反転した値を代表ノード番号に加算した配列番号の配列要素に格納される。

その際、インデックスキー“0 1 1 0 1 0”と挿入キー“0 1 1 0 1 1”とは5ビット目で異なることから、ノード2 1 1 dは、弁別ビット位置を5とし、代表ノード番号を挿入されたノード対の代表ノードの配列番号とするブランチノードとなる。

- [0064] また図2Bのカップルドノードツリーに“0 1 1 0 0 1”を挿入しようとする場合も、検索結果のインデックスキーはノード2 1 1 dに格納された“0 1 1 0 1 0”になる。この場合には挿入キーが小さいのでブール値0が得られ、挿入されるノード対の代表ノード番号に0を加えた配列要素に挿入キーを保持するリーフノードが格納される。そして、インデックスキー“0 1

1010”と挿入キー“011001”とは4ビット目で異なることから、ノード211dは、弁別ビット位置を4とし、代表ノード番号を挿入されたノード対の代表ノードの配列番号とするブランチノードとなる。次に図7CのステップS716以下の処理に進む。

[0065] 図7Cは図7Bで準備された配列にノードを格納するとともにその挿入位置を求め、既存のノードの内容を変更して挿入処理を完成させる処理フローを示す図である。

ステップS716～ステップS723までの処理は、挿入するノード対のカップルドノードツリー上の位置を求める処理であり、ステップS724以下の処理は各ノードにデータを設定して挿入処理を完成させる処理である。

[0066] ステップS716で、挿入キーとステップS710で得たインデックスキーのビット列比較を例えば排他的論理和で行い、差分ビット列を得る。

ステップS717に進み、ステップS716で得た差分ビット列から、上位0ビット目から見た最初の不一致ビットのビット位置を得る。この処理は、例えばプライオリティエンコーダを有するCPUではそこに差分ビット列を入力し、不一致のビット位置を得ることができる。また、ソフト的にプライオリティエンコーダと同等の処理を行い最初の不一致ビットのビット位置を得ることも可能である。

[0067] 次にステップS718に進み、探索経路スタックのスタックポインタがルートノードの配列番号を指しているか判定する。指していればステップS724に移行し、指していなければステップS719に進む。

ステップS719において、探索経路スタックのスタックポインタを1つ戻してそこにスタックされている配列番号を取り出す。

ステップS720に進み、ステップS719で取り出した配列番号の配列要素を配列からノードとして読み出す。

ステップS721に進み、ステップS720で読み出したノードから、弁別ビット位置を取り出す。

次にステップS722に進み、ステップS721で取り出した弁別ビット

位置がステップS 7 1 7で得たビット位置より上位の位置関係か判定する。ここで上位の位置関係とは、ビット列のより左側の位置、すなわちビット位置の値が小さい位置であることとする。

ステップS 7 2 2の判定結果が否定であれば、ステップS 7 1 8に戻り、ステップS 7 1 8での判定が肯定になるかステップS 7 2 2での判定が肯定になるまで繰り返す。ステップS 7 2 2での判定が肯定になると、ステップS 7 2 3で経路探索スタックのスタックポインタを1つ進め、ステップS 7 2 4以下の処理に移行する。

[0068] 上記ステップS 7 1 6～ステップS 7 2 3で説明した処理は、挿入するノード対の挿入位置を決定するために、挿入するインデックスキーと検索により取得されたインデックスキーの間でビット列比較を行い、ビット列比較で異なるビット値となる先頭の（最上位の）ビット位置と探索経路スタックに格納されているブランチノードの弁別ビット位置との相対的位置関係を調べ、弁別ビット位置が上位となるブランチノードの次のブランチノードのリンク先を挿入するノード対の挿入位置とするものである。

[0069] 例えば図2Bのカップルドノードツリーに“1 1 1 0 0 0”を挿入するとき、検索結果のインデックスキーはノード2 1 0 hに格納された“1 0 1 0 1 1”になる。挿入キー“1 1 1 0 0 0”とノード2 1 0 hに格納されたインデックスキー“1 0 1 0 1 1”のビット列比較により異なるビット値となる最上位のビット位置1が得られる。得られたビット位置1と経路探索スタックに積まれた配列番号の配列要素に格納されたブランチノードの弁別ビット位置との位置関係を弁別ビット位置が上位になるまでを順次経路探索スタック逆にたどると、ルートノード2 1 0 aに至る。そこで探索経路スタックのポインタを1つ進め、ノード2 1 1 bの配列番号を得る。挿入キー“1 1 1 0 0 0”はノード2 1 1 bのリンク先に挿入される。

[0070] また、経路探索スタック逆にたどりルートノードに至っても、ルートノードの弁別ビット位置が、先に求めたビット列比較で異なるビット値となる最上位のビット位置より上位のビット位置でないということは、そのカップル

ドノードツリーのインデックスキーの上位ビットで、ルートノードの弁別ビット位置より上位のビットの値は全て等しい場合である。そして、挿入するインデックスキーにおいて、初めてルートノードの弁別ビット位置より上位のビットの値に異なるビット値のものがあるということである。したがって、挿入するノード対はルートノードの直接のリンク先となり、ルートノードの弁別ビット位置は、既存のインデックスキーと異なる値である挿入キーの最上位ビットの位置に変わる。

[0071] 次に、ステップS 7 2 4以下の各ノードにデータを設定して挿入処理を完成させる処理について説明する。

ステップS 7 2 4では探索経路スタックからスタックポインタの指す配列番号を取り出す。

ステップS 7 2 5において、ステップS 7 1 4で得た配列番号の指す配列要素のノード種別に1（リーフノード）を、インデックスキーに挿入キーを書き込む。

ステップS 7 2 6に進み、配列からステップS 7 2 4で得た配列番号の配列要素を読み出す。

次にステップS 7 2 7において、ステップS 7 1 5で得た配列番号の配列要素にステップS 7 2 6で読み出した内容を書き込む。

最後にステップS 7 2 8において、ステップS 7 2 4で得た配列番号の指す配列要素のノード種別に0（ブランチノード）を、弁別ビット位置にステップS 7 1 7で得たビット位置を、代表ノード番号にステップS 7 1 2で得た配列番号を書き込み、処理を終了する。

[0072] 上述の図2Bのカップルドノードツリーに“1 1 1 0 0 0”を挿入する例では、取得された空ノード対のノード[0]にノード2 1 1 bの内容を書き込み（ステップS 7 2 7）、ノード[1]を挿入キー“1 1 1 0 0 0”を保持するリーフノードとする（ステップS 7 2 5）。そして、ノード2 1 1 bの弁別ビット位置にビット列比較により異なるビット値となる最上位のビット位置1を格納し、代表ノード番号には取得されたノード対の代表ノードが格納

される配列要素の配列番号が格納される（ステップS 7 2 8）。

[0073] 図7 Dは、本出願人による出願である上記特願2 0 0 6－1 8 7 8 2 7で提案されたルートノードの挿入処理を含むインデックスキーを追加する場合のノード挿入処理全体を説明する処理フロー図である。

[0074] ステップS 1 0 1において、取得することを求められたカップルドノードツリーのルートノードの配列番号が登録済みであるか判定される。登録済みであれば、図7 A～図7 Cを用いて説明した通常の挿入処理が行われる。

[0075] ステップS 1 0 1での判定が登録済みでなければ、まったく新しいカップルドノードツリーの登録、生成が始まることになる。

まず、ステップS 1 0 2において、配列から空きのノード対を求め、そのノード対のうち代表ノードとなるべき配列要素の配列番号を取得する。次にステップS 1 0 3において、ステップS 1 0 2で得た配列番号に0を加えた配列番号を求める。（実際には、ステップS 1 0 2で取得した配列番号に等しい。）。さらにステップS 1 0 4において、ステップS 1 0 3で得た配列番号の配列要素に、挿入するルートノードのノード種別に1（リーフノード）とインデックスキーに挿入キーを書き込み、ステップS 1 0 5で、ステップS 1 0 2で取得したルートノードの配列番号を登録して処理を終了する。

[0076] 先にも述べたように、インデックスキーの集合があるとき、そこから順次インデックスキーを取り出し、図7 D及び図7 A～図7 Cの処理を繰り返すことにより、インデックスキーの集合に対応した本発明のカップルドノードツリーを構築することができることは明らかである。

[0077] 次に図8 A、図8 Bを参照して、本出願人による出願である上記特願2 0 0 6－1 8 7 8 2 7で提案されたカップルドノードツリーに係るインデックスキーの集合から、特定のインデックスキーを削除する処理フローを説明する。

[0078] 図8 Aは、削除処理の前段である検索処理の処理フローを示す図であり、図4に示した検索処理において、ルートノードを検索開始ノードとし、挿入キーを検索キーとしたものに相当する。

[0079] ステップS 8 0 1において、検索開始ノードの配列番号を設定するエリアにルートノードの配列番号を設定し、ステップS 8 0 2において、検索キーに削除キーを設定する。

次にステップS 8 1 0において、図 4 に示す検索処理を実行し検索結果のインデックスキーを得て、ステップS 8 1 1に進む。

図 8 A のステップS 8 1 1において削除キーとインデックスキーを比較し、等しくなければ削除するインデックスキーはカップルドノードツリーに存在しないのであるから、削除は失敗となり、処理を終了する。等しければ次の処理、図 8 B のステップS 8 1 2以下の処理に進む。

[0080] 図 8 B は、削除処理の後段の処理フローを説明する図である。

まず、ステップS 8 1 2で探索経路スタックに2つ以上の配列番号が格納されているか判定する。2つ以上の配列番号が格納されていないということは、言い換えれば1つだけで、その配列番号はルートノードの格納された配列要素のものである。その場合はステップS 8 1 8に移行し、ステップS 8 0 1で得たルートノードの配列番号に係るノード対を削除する。次にステップS 8 1 9に進み、登録されていたルートノードの配列番号を削除して処理を終了する。

[0081] ステップS 8 1 2において探索経路スタックに2つ以上の配列番号が格納されていると判定されたときはステップS 8 1 3に進み、ステップS 8 0 8で得た代表ノード番号にステップS 8 0 7で得たビット値を反転した値を加算した配列番号を得る。この処理は、削除対象のインデックスキーが格納されたリーフノードと対をなすノードの配置された配列番号を求めるものである。

次にステップS 8 1 4において、ステップS 8 1 3で得た配列番号の配列要素の内容を読み出し、ステップS 8 1 5において探索経路スタックのスタックポインタを1つ戻して配列番号を取り出す。

[0082] 次にステップS 8 1 6に進み、ステップS 8 1 4で読み出した配列要素の内容をステップS 8 1 5で得た配列番号の配列要素に上書きする。この処理

は、削除対象のインデックスキーが格納されたリーフノードへのリンク元であるブランチノードを上記リーフノードと対をなすノードに置き換えるものである。

最後にステップS 8 1 7においてステップS 8 0 8で得た代表ノード番号に係るノード対を削除して処理を終了する。

以上、カップルドノードツリーに関する本発明の前提となる技術を説明したが、必要とあれば、上述の特許出願の明細書及び図面の記載を参照されたい。

[0083] 次に、図9～図13を参照して、本発明に係る差分データを作成する処理について説明する。

図9は、本発明の一実施形態における旧データと新データにより差分データを作成する処理の概要を説明する図である。

[0084] 図9に示す例では、旧データ321はキー321a“010010”、321b“010011”、321c“100011”、321d“101011”及び321e“101100”で構成され、新データ322はキー322a“010010”、322b“010110”、322c“100010”、322d“100011”及び322e“101100”で構成されている。

また、差分データ320の各更新データ380a～380dは、更新種別381と更新キー382の項目で構成されている。更新データ380a～380dの各更新キー382は、図に示すようにキー321d、321b、322c、322bである。

[0085] 旧データであるキー321a～321eは、図9の(A)に示す挿入処理により、更新前の差分ツリー309-1に挿入される。なお、図9の(A)に示すS1001は、後記図10に示す差分データ作成の処理フローのステップS1001での処理に対応することを示すものである。以下の図9の(B)～(E)についても同様である。

[0086] 次に、図9の(B)に示すように、新データのキー322a～322eに

よる削除処理が図 10 に示すフローのステップ S 1005 で行われる。削除処理の結果、更新後の差分ツリー 309-2 には旧データのキー 321 d と 321 b が削除されずに残っている。

旧データに同一のデータが存在せず、図 9 の (C) に示すように図 10 のフローのステップ S 1006 で削除に失敗したと判定された新データのキー 322 c と 322 b は、図 9 の (D) に示すように、図 10 のステップ S 1007 で挿入キーとして差分データ格納領域 320 に格納される。図 9 に示すように、挿入キーの値は更新キー 382 に格納され、更新種別 381 には “i” が格納され、更新キーに格納された 322 c、322 b が挿入キーであることが示されている。

[0087] 一方、更新後の差分ツリー 309-2 に残った旧データのキー 321 d と 321 b は、図 9 の (E) に示すように、図 10 のステップ S 1009 で差分ツリー 309-2 から取り出され、削除キーとして差分データ格納領域 320 に格納される。図 9 に示すように、削除キーの値は更新キー 382 に格納され、更新種別 381 には “d” が格納され、更新キーに格納された 321 d、321 b が削除キーであることが示されている。

[0088] なお、上述の説明では、旧データから差分ツリー 309-1 を生成し、新データにより削除処理を行うものとした。しかし先に述べたように、旧データと新データの差分を作成するのであるから、逆に新データから差分ツリーを生成し、旧データにより削除処理を行って差分データを作成できることは明らかである。その場合には、更新後の差分ツリーに残った新データが挿入キーとなり、削除に失敗した旧データが削除キーとなることも当業者には明らかである。

[0089] 図 10 は旧データと新データにより差分データを作成する処理フローを説明する図である。以下、図 10 を参照して新旧データによる差分データ作成の処理フローを説明する。

図 10 に示すように、ステップ S 1001 において、旧データの全てのキーを差分ツリーに挿入する。この処理の詳細は、後に図 11 を参照して行う

。なお、旧データがすでにカップルドノードツリーに格納されている、すなわちデータベースのインデックスがカップルドノードツリーのデータ構造を有しているものであれば、ステップ 1001 に代えてその登録されたルートノードの配列番号を取得するだけでよい。また、カップルドノードツリーを配列に格納しない場合であれば、ルートノードの位置を示す位置情報を取得すればよい。

[0090] 次に、ステップ S 1002 において、新データを設定する。この新データの設定は、新データの供給元から供給された新データを、例えば図 3 B 及び図 9 に示す新データ格納領域 322 に格納することによって行われるが、供給元から供給された新データを格納した記憶媒体から直接キーを読み出すことも可能である。

[0091] 次にステップ S 1003 に進み、全ての新データが処理済であるか判定する。全ての新データが処理済であればステップ S 1008 に移行し、処理済でなければステップ S 1004 に進む。

ステップ S 1004 では、ステップ S 1002 で設定された新データからキーを取り出す。次のステップ S 1005 で、該取り出したキーを削除キーとして差分ツリーから削除する。この削除処理は、図 8 A 及び図 8 B に示すものである。

次にステップ S 1006 において、ステップ S 1005 での削除が成功したかを判定する。削除に成功すれば、ステップ S 1003 に戻り、失敗すればステップ S 1007 において、削除キーを挿入キーとして差分データを作成し、ステップ S 1003 に戻る。ステップ S 1007 の処理の詳細は、後に図 12 を参照して説明する。

[0092] 全ての新データが処理済となってステップ S 1003 から分岐したステップ S 1008 においては、差分ツリーが登録されているかを判定する。登録されていなければ処理を終了し、登録されていればステップ S 1009 で差分ツリーの全てのキーを取り出し、削除キーとして差分データを作成して処理を終了する。ステップ S 1009 の詳細は、後に図 13 を参照して説明す

る。

なお、図 10 で説明してフローはあくまで 1 つの例であり、先に図 9 による説明で述べたように新データにより差分ツリーを生成する方法や処理の順番を一部替える等の種々の変形が可能であることは当業者にとって自明である。

[0093] 図 11 は、図 10 に示す旧データと新データにより差分データを作成する処理フローのステップ S 1001 で実行される、旧データによりカップルドノードツリー（差分ツリー）を作成する処理フローを説明する図である。

[0094] 最初にステップ S 1101 において、旧データを設定する。この旧データの設定は、旧データの供給元から供給された旧データを保管しておき、例えば図 3 B 及び図 9 に示す旧データ格納領域 321 に格納することによって行われるが、供給元から供給された旧データを格納した記憶媒体から直接キーを読み出すことも可能である。

[0095] 次にステップ S 1102 で旧データから全てのキーを取り出し済みか判定し、取り出し済みであれば処理を終了する。

旧データから全てのキーを取り出し済みでなければ、ステップ S 1103 において旧データからキーを取り出し、ステップ S 1104 に進んで取り出したキーを挿入キーとして、図 7 A ~ 図 7 D に示す処理により、差分ツリーにキーを挿入する。

ステップ S 1104 の挿入処理の後、ステップ S 1102 の判定処理に戻るループ処理を、旧データから全てのキーが取り出し済みとなるまで繰り返す。

[0096] 図 12 は、差分データに更新データを格納する処理フローを説明する図である。図 12 に示す処理フローは、図 10 に示す旧データと新データにより差分データを作成する処理フローのステップ S 1007 で実行される、削除キーを挿入キーとして差分データを作成する処理に適用可能なものである。

また、図 12 に示す処理フローは、図 10 に示す処理フローのステップ S 1009 で実行される差分データ作成処理にも適用可能なものとなっており

、データの更新が挿入であるか削除であるかの情報と、差分データの更新キーとなるキーの値が与えられるものとしている。また、ワークエリアとして、図示しない更新種別と更新キーからなる更新データ設定エリアを使用するものとする。

[0097] 図に示すようにステップS 1 2 0 1において、更新データ設定エリアの更新種別に、データの更新が挿入であるか削除であるかの情報に基づいて、更新種別を設定する。図9に示す例では、呼出元がステップS 1 0 0 7であり、データ更新が挿入であれば、更新種別に“i”が設定される。

次にステップS 1 2 0 2において、更新データ設定エリアの更新キーに、更新キーとなるキーの値を設定する。

次にステップS 1 2 0 3において、ステップS 1 2 0 1とステップS 1 2 0 2で更新データ設定エリアに設定された更新データを差分データ格納領域に格納して処理を終了する。

[0098] 図13は、図10に示す旧データと新データにより差分データを作成する処理フローのステップS 1 0 0 9で実行される、差分ツリーの全てのキーを取り出し、削除キーとして差分データを作成する処理フローを説明する図である。

ステップS 1 3 0 1において、検索開始ノードの配列番号として、差分ツリーのルートノードの配列番号を設定する。次に検索開始ノードより図5に示す最小値検索を行い、インデックスキーの最小値を求める。

次にステップS 1 3 0 3に進み、ステップS 1 3 0 2で求めた最小値を削除キーとして差分データを作成する。このステップ1303の処理は、更新種別を削除とし、更新キーの値を直前の最小値検索で求めた最小値として図12に示す差分データに更新データを格納する処理により実現される。

[0099] 以下、ステップS 1 3 0 3、ステップS 1 3 0 4～ステップS 1 3 1 0に示すように、差分ツリーから最小値検索を繰り返して昇順でインデックスキーを取り出し、取り出した最小値を削除キーとして差分データを作成するステップS 1 3 0 3を繰り返す。

したがって、ステップS 1 3 0 4～ステップS 1 3 1 0の処理は、図6に示すインデックスキーを昇順に取り出す処理フローのステップS 6 0 4～ステップS 6 1 0の処理と同様であるので、説明を省略する。

なお、図10に示すステップS 1 0 0 9で実行される処理では、差分ツリーに残された全てのキーを取り出せばよいのであるから、図13に示す昇順取り出しに限らず、先に述べた降順取り出しを採用することも可能である。

以上の図10～図13を参照して説明した処理により、旧データと新データにより差分データが作成される。

[0100] 次に、図14、図15を参照して、本発明に係る差分データにより旧データを新データに更新する処理について説明する。

図14は、本発明の一実施形態における差分データにより旧データを新データに更新する処理の概要を説明する図である。図14に示す例は、図9において例示した旧データと新データにより作成された差分データにより、図3Bに例示する差分データ更新装置400aにおいて旧データを新データに更新するものとしている。

[0101] したがって、インデックス格納領域421aには、図9に示す旧データと同様に、旧データのキー321a “010010”、321b “010011”、321c “100011”、321d “101011”及び321e “101100”が格納されている。また同様に、差分データ格納領域420aにも、図9に示す差分データ格納領域に格納された更新データと同一のデータが格納されている。

[0102] 旧データであるキー321a～321eは、図14の(A)に示す挿入処理により、更新前の更新ツリー409a-1に挿入される。なお、図14の(A)に示すS1501は、後記図15に示す差分データにより旧データを新データに更新する処理フローのステップS1501での処理に対応することを示すものである。以下の図14の(B)～(D)についても同様である。

[0103] 次に、図 1 4 の (B) に示すように、差分データ格納領域 4 2 0 a に格納された更新データのうち、更新種別 2 8 1 の値が “d” である更新データ 3 8 0 a、3 8 0 b の更新キー 2 8 2 であるキー 3 2 1 b とキー 3 2 1 d による削除処理が図 1 5 に示すフローのステップ S 1 5 0 8 で行われる。

[0104] さらに、図 1 4 の (C) に示すように、差分データ格納領域 4 2 0 a に格納された差分データのうち、更新種別 2 8 1 の値が “i” である更新データ 3 8 0 c、3 8 0 d の更新キー 2 8 2 であるキー 3 2 2 b とキー 3 2 2 c による挿入処理が図 1 5 に示すフローのステップ S 1 5 0 7 で行われる。

その結果、更新前の更新ツリー 4 0 9 a - 1 は更新後の更新ツリー 4 0 9 a - 2 に更新される。更新ツリー 4 0 9 a - 2 には、旧データに含まれていたキー 3 2 1 a、3 2 1 c 及び 3 2 1 e と挿入されたキー 3 2 2 b と 3 2 2 c が含まれている。

[0105] 次に、図 1 4 の (D) に示すように、更新後の更新ツリー 4 0 9 a - 2 からキーを取り出して新データ格納領域 4 2 2 a に格納する処理が図 1 5 に示すステップ S 1 5 0 9 で行われ、旧データが新データに更新される。新データ格納領域 4 2 2 a には、キー 3 2 1 a “0 1 0 0 1 0”、3 2 2 b “0 1 0 1 1 0”、3 2 2 c “1 0 0 0 1 0”、3 2 1 c “1 0 0 0 1 1” 及び 3 2 1 e “1 0 1 1 0 0” が格納されている。なお、新データ格納領域 4 2 2 a はインデックス格納領域 4 2 1 a と同一の領域とし、旧データに新データを上書きすることができる。

[0106] なお、上述の説明では、更新ツリー 4 0 9 a - 1 に対して削除処理を実行した後に挿入処理を行うものとしたが、どのような順番で削除挿入を実行しても結果は同一である。

また、差分データ 4 2 0 a のうち、挿入キーのみ取り出して旧データ 4 2 1 a とともに更新前の更新ツリー 4 0 9 a - 1 を作成し、その後削除処理のみによって更新後の更新ツリー 4 0 9 a - 2 を作成する等種々の変形が可能である。

[0107] 図 1 5 は、差分データにより旧データを新データに更新する処理フローを

説明する図である。

図 15 に示すように、ステップ S 1501 において、旧データの全てのキーを更新ツリーに挿入する。この処理は、図 11 を参照して説明した旧データによりカップルドノードツリーを作成する処理により、実現される。先に述べたのと同様に、旧データがすでにカップルドノードツリーに格納されている、すなわちデータベースのインデックスがカップルドノードツリーのデータ構造を有しているものであれば、ステップ 1501 に代えてその登録されたルートノードの配列番号あるいはルートノードの位置を示す位置情報を取得するだけでよい。さらにその場合、データベースのインデックスがカップルドノードツリーのデータ構造を有しているのであるから、後記ステップ S 1509 の処理が不要になることも明らかである。

[0108] 次に、ステップ S 1502 において、差分データを設定する。この差分データの設定は、差分データ作成装置により差分データを作成した供給元から供給された差分データを、例えば図 3 B 及び図 14 に示す差分データ格納領域 420a ~ 420x に格納することによって行われるが、供給元から供給された新データを格納した記憶媒体から直接キーを読み出すことも可能である。

[0109] 次にステップ S 1503 に進み、全ての更新データが処理済であるか判定する。全ての更新データが処理済であればステップ S 1509 に移行し、処理済でなければステップ S 1504 に進む。

ステップ S 1504 では、ステップ S 1502 で設定された差分データの更新データを取り出し、そこから更新キーを取り出す。

[0110] 次のステップ S 1505 で、更新データから更新種別を取り出し、次にステップ S 1506 において、ステップ S 1505 で取り出した更新種別を判定する。

更新種別が削除であればステップ S 1507 に進み、挿入であればステップ S 1508 に進む。

[0111] ステップ S 1507 では、図 8 A、図 8 B に示す削除処理により、更新キ

ーを削除キーとして更新ツリーより削除して、ステップS 1 5 0 3に戻る。

ステップS 1 5 0 8では、図7 A～図7 Dに示す挿入処理により、更新キーを挿入キーとして更新ツリーに挿入して、ステップS 1 5 0 3に戻る。

[0112] 全ての更新データが処理済となってステップS 1 5 0 3から分岐したステップS 1 5 0 9においては、図6に示す昇順取り出し処理により、更新ツリーの全てのキーを新データとして取り出し、処理を終了する。

[0113] 上述の説明では、更新データの更新種別により削除処理と挿入処理を切り分けたが、図10に示す差分データ作成処理において、差分データファイルを削除キーのファイルと挿入キーのファイルに分割して作成すれば、それぞれのファイルごとに削除処理と挿入処理を実行することになる。また、図14の説明で述べたように、旧データと挿入キーのファイルの全てのキーを更新ツリーに挿入する処理を行い、削除キーのファイルに格納された削除キーにより更新ツリーの削除処理を実行して旧データを新データに更新することも可能である。

[0114] 以上詳細に説明したところから明らかなとおり、本発明によればカップルドノードツリーを活用することにより、データベースのインデックスを旧データのものから新データに効率よく更新することができる。

また、本発明の実施の形態は上記に限ることなく種々の変形が可能であることは当業者に明らかである。さらに、本発明のインデックスキーの更新方法を、コンピュータに実行させるプログラムによりコンピュータで実行可能なことは明らかである。

したがって、上記プログラム、及びプログラムを記憶したコンピュータ読み取り可能な記憶媒体は、本発明の実施の形態に含まれる。

請求の範囲

- [1] データベースのインデックスキーの旧データに対して該旧データと置き換えられる新しいインデックスキーの新データが供給されたときに、前記インデックスキーの更新を行うデータベースのインデックスキー更新方法において、

前記旧データあるいは新データのインデックスキーを、

ルートノードと、隣接した記憶領域に配置されるブランチノードとリーフノードまたはブランチノード同士またはリーフノード同士のノード対、からなるビット列検索に用いるツリーであって、前記ルートノードは、ツリーの始点を表すノードであって、該ツリーのノードが1つのときは前記リーフノード、ツリーのノードが2つ以上のときは前記ブランチノードであり、前記ブランチノードは、ビット列検索を行う検索キーの弁別ビット位置とリンク先のノード対の一方のノードである代表ノードの位置を示す位置情報を含み、前記リーフノードは検索対象のビット列からなるインデックスキーを含み、前記ツリーの任意のノードを検索開始ノードとして前記ブランチノードにおいて、該ブランチノードに含まれる弁別ビット位置の検索キーのビット値に応じてリンク先のノード対の代表ノードかあるいはそれと隣接した記憶領域に配置されたノードにリンクすることを順次前記リーフノードに至るまで繰り返すことにより、前記リーフノードに格納されたインデックスキーを、前記検索開始ノードをルートノードとする前記ツリーの任意の部分木の前記検索キーによる検索結果である検索結果キーとするように構成された、

カップルドノードツリーのリーフノードのインデックスキーとして格納した、差分ツリーを取得する差分ツリー取得ステップと、

前記差分ツリーのルートノードを前記検索開始ノードとして前記新データあるいは旧データの全てのインデックスキーを検索キーとして検索を行い、検索キーと一致した前記検索結果キーであるインデックスキーは差分ツリーから削除し、前記検索結果キーであるインデックスキーと一致しない検索キーを差分データの挿入キーあるいは削除キーとし、前記新データあるいは旧

データの全てのインデックスキーによる検索が終了した後に前記差分ツリーに削除されずに残ったインデックスキーを差分データの削除キーあるいは挿入キーとして差分データを作成する差分データ作成ステップと、

前記旧データのインデックスキーをリーフノードのインデックスキーとして格納した前記カップルドノードツリーである更新ツリーを取得する更新ツリー取得ステップと、

前記差分データから取り出した削除キーを前記更新ツリーから削除し、前記差分データから取り出した挿入キーを前記更新ツリーに挿入することにより新データのインデックスキーをリーフノードのインデックスキーとして格納した新データカップルドノードツリーを作成し、該新データカップルドノードツリーに基づいて、前記旧データを前記新データに更新する新旧インデックスキー更新ステップと、

を備えることを特徴とするインデックスキー更新方法。

- [2] データベースのインデックスキーの旧データに対して該旧データと置き換えられる新しいインデックスキーの新データが供給されたときに、前記インデックスキーの更新を行うデータベースのインデックスキー更新に用いる前記旧データと前記新データの差分データ作成方法において、

前記旧データあるいは新データのインデックスキーを、

ルートノードと、隣接した記憶領域に配置されるブランチノードとリーフノードまたはブランチノード同士またはリーフノード同士のノード対、からなるビット列検索に用いるツリーであって、前記ルートノードは、ツリーの始点を表すノードであって、該ツリーのノードが1つのときは前記リーフノード、ツリーのノードが2つ以上のときは前記ブランチノードであり、前記ブランチノードは、ビット列検索を行う検索キーの弁別ビット位置とリンク先のノード対の一方のノードである代表ノードの位置を示す位置情報を含み、前記リーフノードは検索対象のビット列からなるインデックスキーを含み、前記ツリーの任意のノードを検索開始ノードとして前記ブランチノードにおいて、該ブランチノードに含まれる弁別ビット位置の検索キーのビット値

に応じてリンク先のノード対の代表ノードかあるいはそれと隣接した記憶領域に配置されたノードにリンクすることを順次前記リーフノードに至るまで繰り返すことにより、前記リーフノードに格納されたインデックスキーを、前記検索開始ノードをルートノードとする前記ツリーの任意の部分木の前記検索キーによる検索結果である検索結果キーとするように構成された、

カップルドノードツリーのリーフノードのインデックスキーとして格納した、差分ツリーを取得する差分ツリー取得ステップと、

前記差分ツリーのルートノードを前記検索開始ノードとして前記新データあるいは旧データの全てのインデックスキーを検索キーとして検索を行い、検索キーと一致した前記検索結果キーであるインデックスキーは差分ツリーから削除し、前記検索結果キーであるインデックスキーと一致しない検索キーを差分データの挿入キーあるいは削除キーとし、前記新データあるいは旧データの全てのインデックスキーによる検索が終了した後に前記差分ツリーに削除されずに残ったインデックスキーを差分データの削除キーあるいは挿入キーとして差分データを作成する差分データ作成ステップと、

を備えることを特徴とする差分データ作成方法。

[3] 請求項 2 記載の差分データ作成方法において、

前記カップルドノードツリーは、配列に記憶され、前記リンク先のノード対の代表ノードの位置を示す情報は、そのノードが格納された前記配列の配列要素の配列番号であることを特徴とする差分データ作成方法。

[4] 請求項 3 記載の差分データ作成方法において、

前記差分ツリー取得ステップは、前記旧データあるいは新データから最初に取り出したインデックスキーを含むリーフノードを該差分ツリーのルートノードとした後、それ以後前記旧データあるいは新データから取り出した前記インデックスキーを前記検索キーとして該差分ツリーから該当するリーフノードを検索するとともに、該リーフノードに至るまでたどったリンク経路のブランチノード及び該リーフノードが格納された配列要素の配列番号をスタックに順次格納し、前記検索キーと前記該当するリーフノードに含まれる

インデックスキーの間で大小比較とビット列比較を行い、ビット列比較で異なるビット値となる先頭のビット位置と前記スタックに格納されているブランチノードの弁別ビット位置との相対的位置関係により挿入されるインデックスキーを含むリーフノードともう一方のノードからなるノード対の挿入位置を決定し、前記大小関係により挿入するインデックスキーを含むリーフノードを前記挿入されるノード対のどちらのノードとするかを決定するインデックスキーの挿入処理を繰り返すものであることを特徴とする差分データ作成方法。

- [5] データベースのインデックスキーの旧データと該旧データと置き換えられる新しいインデックスキーの新データの差分データを用いて前記インデックスキーの更新を行うインデックスキーの差分データ更新方法において、前記旧データのインデックスキーを、
- ルートノードと、隣接した記憶領域に配置されるブランチノードとリーフノードまたはブランチノード同士またはリーフノード同士のノード対、からなるビット列検索に用いるツリーであって、前記ルートノードは、ツリーの始点を表すノードであって、該ツリーのノードが1つのときは前記リーフノード、ツリーのノードが2つ以上のときは前記ブランチノードであり、前記ブランチノードは、ビット列検索を行う検索キーの弁別ビット位置とリンク先のノード対の一方のノードである代表ノードの位置を示す位置情報を含み、前記リーフノードは検索対象のビット列からなるインデックスキーを含み、前記ツリーの任意のノードを検索開始ノードとして前記ブランチノードにおいて、該ブランチノードに含まれる弁別ビット位置の検索キーのビット値に応じてリンク先のノード対の代表ノードかあるいはそれと隣接した記憶領域に配置されたノードにリンクすることを順次前記リーフノードに至るまで繰り返すことにより、前記リーフノードに格納されたインデックスキーを、前記検索開始ノードをルートノードとする前記ツリーの任意の部分木の前記検索キーによる検索結果である検索結果キーとするように構成された、
- カップルドノードツリーのリーフノードのインデックスキーとして格納した

前記カップルドノードツリーである更新ツリーを取得する更新ツリー取得ステップと、

前記旧データあるいは新データのインデックスキーを前記カップルドノードツリーのリーフノードのインデックスキーとして格納した差分ツリーのルートノードを前記検索開始ノードとして前記新データあるいは旧データの全てのインデックスキーを検索キーとして検索を行い、検索キーと一致した前記検索結果キーであるインデックスキーは差分ツリーから削除し、前記検索結果キーであるインデックスキーと一致しない検索キーを差分データの挿入キーあるいは削除キーとし、前記新データあるいは旧データの全てのインデックスキーによる検索が終了した後に前記差分ツリーに削除されずに残ったインデックスキーを差分データの削除キーあるいは挿入キーとして作成した差分データから取り出した削除キーを前記更新ツリーから削除し、前記差分データから取り出した挿入キーを前記更新ツリーに挿入することにより新データのインデックスキーをリーフノードのインデックスキーとして格納した新データカップルドノードツリーを作成し、該新データカップルドノードツリーに基づいて、前記旧データを前記新データに更新する新旧インデックスキー更新ステップと、

を備えることを特徴とするインデックスキーの差分データ更新方法。

[6] 請求項 5 記載のインデックスキーの差分データ更新方法において、

前記カップルドノードツリーは、配列に記憶され、前記リンク先のノード対の代表ノードの位置を示す情報は、そのノードが格納された前記配列の配列要素の配列番号であることを特徴とするインデックスキーの差分データ更新方法。

[7] 請求項 6 記載のインデックスキーの差分データ更新方法において、

前記更新ツリー取得ステップは、前記旧データから最初に取り出したインデックスキーを含むリーフノードを該更新ツリーのルートノードとした後、それ以後前記旧データから取り出した前記インデックスキーを前記検索キーとして該更新ツリーから該当するリーフノードを検索するとともに、該リー

フノードに至るまでたどったリンク経路のブランチノード及び該リーフノードが格納された配列要素の配列番号をスタックに順次格納し、前記検索キーと前記該当するリーフノードに含まれるインデックスキーの間で大小比較とビット列比較を行い、ビット列比較で異なるビット値となる先頭のビット位置と前記スタックに格納されているブランチノードの弁別ビット位置との相対的位置関係により挿入されるインデックスキーを含むリーフノードともう一方のノードからなるノード対の挿入位置を決定し、前記大小関係により挿入するインデックスキーを含むリーフノードを前記挿入されるノード対のどちらのノードとするかを決定するインデックスキーの挿入処理を繰り返すものであることを特徴とするインデックスキーの差分データ更新方法。

- [8] 請求項 7 記載のインデックスキーの差分データ更新方法において、
前記更新ツリー取得ステップは、前記旧データのインデックスキーに加えて、前記差分データの挿入キーを取り出して前記インデックスキーの挿入処理を繰り返すものであり、

前記新旧インデックスキー更新ステップは、前記差分データから取り出した削除キーを前記更新ツリーから削除することにより新データのインデックスキーをリーフノードのインデックスキーとして格納した新データカップルドノードツリーを作成し、該新データカップルドノードツリーに基づいて、前記旧データを前記新データに更新するものであることを特徴とするインデックスキーの差分データ更新方法。

- [9] 請求項 5 記載のインデックスキーの差分データ更新方法において、
前記旧データのインデックスキーはカップルドノードツリーに格納されており、

前記更新ツリー取得ステップは、該カップルドノードツリーのルートノードの位置を示す位置情報を取得するものであり、

前記新旧インデックスキー更新ステップは、前記新データカップルドノードツリーを作成することで前記旧データを前記新データに更新する、
ことを特徴とするインデックスキーの差分データ更新方法。

[10] データベースのインデックスキーの旧データに対して該旧データと置き換えられる新しいインデックスキーの新データが供給されたときに、前記インデックスキーの更新を行うデータベースのインデックスキー更新に用いる前記旧データと前記新データの差分データ作成装置において、

前記旧データあるいは新データのインデックスキーを、

ルートノードと、隣接した記憶領域に配置されるブランチノードとリーフノードまたはブランチノード同士またはリーフノード同士のノード対、からなるビット列検索に用いるツリーであって、前記ルートノードは、ツリーの始点を表すノードであって、該ツリーのノードが1つのときは前記リーフノード、ツリーのノードが2つ以上のときは前記ブランチノードであり、前記ブランチノードは、ビット列検索を行う検索キーの弁別ビット位置とリンク先のノード対の一方のノードである代表ノードの位置を示す位置情報を含み、前記リーフノードは検索対象のビット列からなるインデックスキーを含み、前記ツリーの任意のノードを検索開始ノードとして前記ブランチノードにおいて、該ブランチノードに含まれる弁別ビット位置の検索キーのビット値に応じてリンク先のノード対の代表ノードかあるいはそれと隣接した記憶領域に配置されたノードにリンクすることを順次前記リーフノードに至るまで繰り返すことにより、前記リーフノードに格納されたインデックスキーを、前記検索開始ノードをルートノードとする前記ツリーの任意の部分木の前記検索キーによる検索結果である検索結果キーとするように構成された、

カップルドノードツリーのリーフノードのインデックスキーとして格納した、差分ツリーを取得する差分ツリー取得手段と、

前記差分ツリーのルートノードを前記検索開始ノードとして前記新データあるいは旧データの全てのインデックスキーを検索キーとして検索を行い、検索キーと一致した前記検索結果キーであるインデックスキーは差分ツリーから削除し、前記検索結果キーであるインデックスキーと一致しない検索キーを差分データの挿入キーあるいは削除キーとし、前記新データあるいは旧データの全てのインデックスキーによる検索が終了した後に前記差分ツリー

に削除されずに残ったインデックスキーを差分データの削除キーあるいは挿入キーとして差分データを作成する差分データ作成手段と、

を備えることを特徴とする差分データ作成装置。

- [11] データベースのインデックスキーの旧データと該旧データと置き換えられる新しいインデックスキーの新データの差分データを用いて前記インデックスキーの更新を行うインデックスキーの差分データ更新装置において、前記旧データのインデックスキーを、

ルートノードと、隣接した記憶領域に配置されるブランチノードとリーフノードまたはブランチノード同士またはリーフノード同士のノード対、からなるビット列検索に用いるツリーであって、前記ルートノードは、ツリーの始点を表すノードであって、該ツリーのノードが1つのときは前記リーフノード、ツリーのノードが2つ以上のときは前記ブランチノードであり、前記ブランチノードは、ビット列検索を行う検索キーの弁別ビット位置とリンク先のノード対の一方のノードである代表ノードの位置を示す位置情報を含み、前記リーフノードは検索対象のビット列からなるインデックスキーを含み、前記ツリーの任意のノードを検索開始ノードとして前記ブランチノードにおいて、該ブランチノードに含まれる弁別ビット位置の検索キーのビット値に応じてリンク先のノード対の代表ノードかあるいはそれと隣接した記憶領域に配置されたノードにリンクすることを順次前記リーフノードに至るまで繰り返すことにより、前記リーフノードに格納されたインデックスキーを、前記検索開始ノードをルートノードとする前記ツリーの任意の部分木の前記検索キーによる検索結果である検索結果キーとするように構成された、カップルドノードツリーのリーフノードのインデックスキーとして格納した前記カップルドノードツリーである更新ツリーを取得する更新ツリー取得手段と、

前記旧データあるいは新データのインデックスキーを前記カップルドノードツリーのリーフノードのインデックスキーとして格納した差分ツリーのルートノードを前記検索開始ノードとして前記新データあるいは旧データの全

てのインデックスキーを検索キーとして検索を行い、検索キーと一致した前記検索結果キーであるインデックスキーは差分ツリーから削除し、前記検索結果キーであるインデックスキーと一致しない検索キーを差分データの挿入キーあるいは削除キーとし、前記新データあるいは旧データの全てのインデックスキーによる検索が終了した後に前記差分ツリーに削除されずに残ったインデックスキーを差分データの削除キーあるいは挿入キーとして作成した差分データから取り出した削除キーを前記更新ツリーから削除し、前記差分データから取り出した挿入キーを前記更新ツリーに挿入することにより新データのインデックスキーをリーフノードのインデックスキーとして格納した新データカップルドノードツリーを作成し、該新データカップルドノードツリーに基づいて、前記旧データを前記新データに更新する新旧インデックスキー更新手段と、

を備えることを特徴とするインデックスキーの差分データ更新装置。

- [12] 請求項 11 記載のインデックスキーの差分データ更新装置において、
前記旧データのインデックスキーはカップルドノードツリーに格納されており、

前記更新ツリー取得手段は、該カップルドノードツリーのルートノードの位置を示す位置情報を取得するものであり、

前記新旧インデックスキー更新手段は、前記新データカップルドノードツリーを作成することで前記旧データを前記新データに更新する、

ことを特徴とするインデックスキーの差分データ更新装置。

- [13] データベースのインデックスキーの旧データに対して該旧データと置き換えられる新しいインデックスキーの新データが供給されたときに、前記インデックスキーの更新を行うデータベースのインデックスキー更新に用いられるコンピュータ上で動作するプログラムにおいて、

前記旧データあるいは新データのインデックスキーを、

ルートノードと、隣接した記憶領域に配置されるブランチノードとリーフノードまたはブランチノード同士またはリーフノード同士のノード対、から

なるビット列検索に用いるツリーであって、前記ルートノードは、ツリーの始点を表すノードであって、該ツリーのノードが1つのときは前記リーフノード、ツリーのノードが2つ以上のときは前記ブランチノードであり、前記ブランチノードは、ビット列検索を行う検索キーの弁別ビット位置とリンク先のノード対の一方のノードである代表ノードの位置を示す位置情報を含み、前記リーフノードは検索対象のビット列からなるインデックスキーを含み、前記ツリーの任意のノードを検索開始ノードとして前記ブランチノードにおいて、該ブランチノードに含まれる弁別ビット位置の検索キーのビット値に応じてリンク先のノード対の代表ノードかあるいはそれと隣接した記憶領域に配置されたノードにリンクすることを順次前記リーフノードに至るまで繰り返すことにより、前記リーフノードに格納されたインデックスキーを、前記検索開始ノードをルートノードとする前記ツリーの任意の部分木の前記検索キーによる検索結果である検索結果キーとするように構成された、

カップルドノードツリーのリーフノードのインデックスキーとして格納した、差分ツリーを取得する差分ツリー取得ステップと、

前記差分ツリーのルートノードを前記検索開始ノードとして前記新データあるいは旧データの全てのインデックスキーを検索キーとして検索を行い、検索キーと一致した前記検索結果キーであるインデックスキーは差分ツリーから削除し、前記検索結果キーであるインデックスキーと一致しない検索キーを差分データの挿入キーあるいは削除キーとし、前記新データあるいは旧データの全てのインデックスキーによる検索が終了した後に前記差分ツリーに削除されずに残ったインデックスキーを差分データの削除キーあるいは挿入キーとして差分データを作成する差分データ作成ステップと、

をコンピュータに実行させることを特徴とするプログラム。

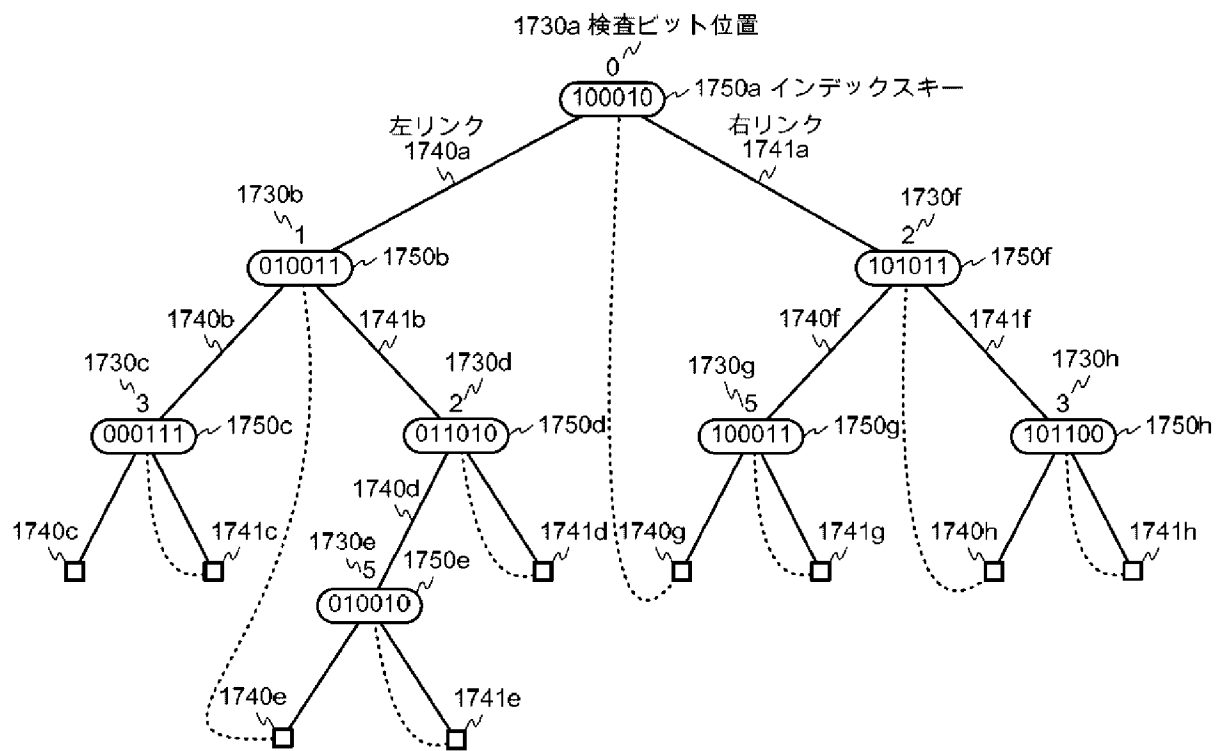
- [14] データベースのインデックスキーの旧データに対して該旧データと置き換えられる新しいインデックスキーの新データが供給されたときに、前記インデックスキーの更新を行うデータベースのインデックスキー更新に用いられるコンピュータ上で動作するプログラムにおいて、

前記旧データのインデックスキーを、ルートノードと、隣接した記憶領域に配置されるブランチノードとリーフノードまたはブランチノード同士またはリーフノード同士のノード対、からなるビット列検索に用いるツリーであって、前記ルートノードは、ツリーの始点を表すノードであって、該ツリーのノードが1つのときは前記リーフノード、ツリーのノードが2つ以上のときは前記ブランチノードであり、前記ブランチノードは、ビット列検索を行う検索キーの弁別ビット位置とリンク先のノード対の一方のノードである代表ノードの位置を示す位置情報を含み、前記リーフノードは検索対象のビット列からなるインデックスキーを含み、前記ツリーの任意のノードを検索開始ノードとして前記ブランチノードにおいて、該ブランチノードに含まれる弁別ビット位置の検索キーのビット値に応じてリンク先のノード対の代表ノードかあるいはそれと隣接した記憶領域に配置されたノードにリンクすることを順次前記リーフノードに至るまで繰り返すことにより、前記リーフノードに格納されたインデックスキーを、前記検索開始ノードをルートノードとする前記ツリーの任意の部分木の前記検索キーによる検索結果である検索結果キーとするように構成されたカップルドノードツリーのリーフノードのインデックスキーとして格納した、更新ツリーを取得する更新ツリー取得ステップと、

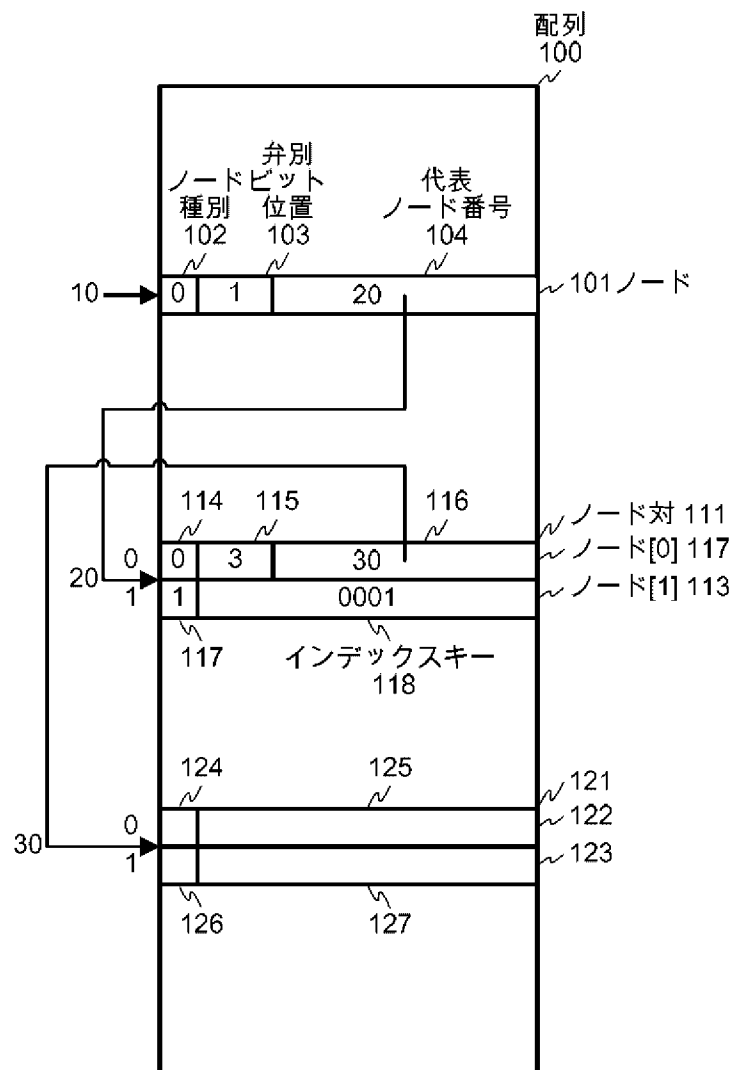
請求項13記載の前記差分データ作成ステップで作成された差分データから取り出した削除キーを前記更新ツリーから削除し、前記差分データから取り出した挿入キーを前記更新ツリーに挿入することにより新データのインデックスキーをリーフノードのインデックスキーとして格納した新データカップルドノードツリーを作成し、該新データカップルドノードツリーに基づいて、前記旧データを前記新データに更新する新旧インデックスキー更新ステップと、

をコンピュータに実行させることを特徴とするプログラム。

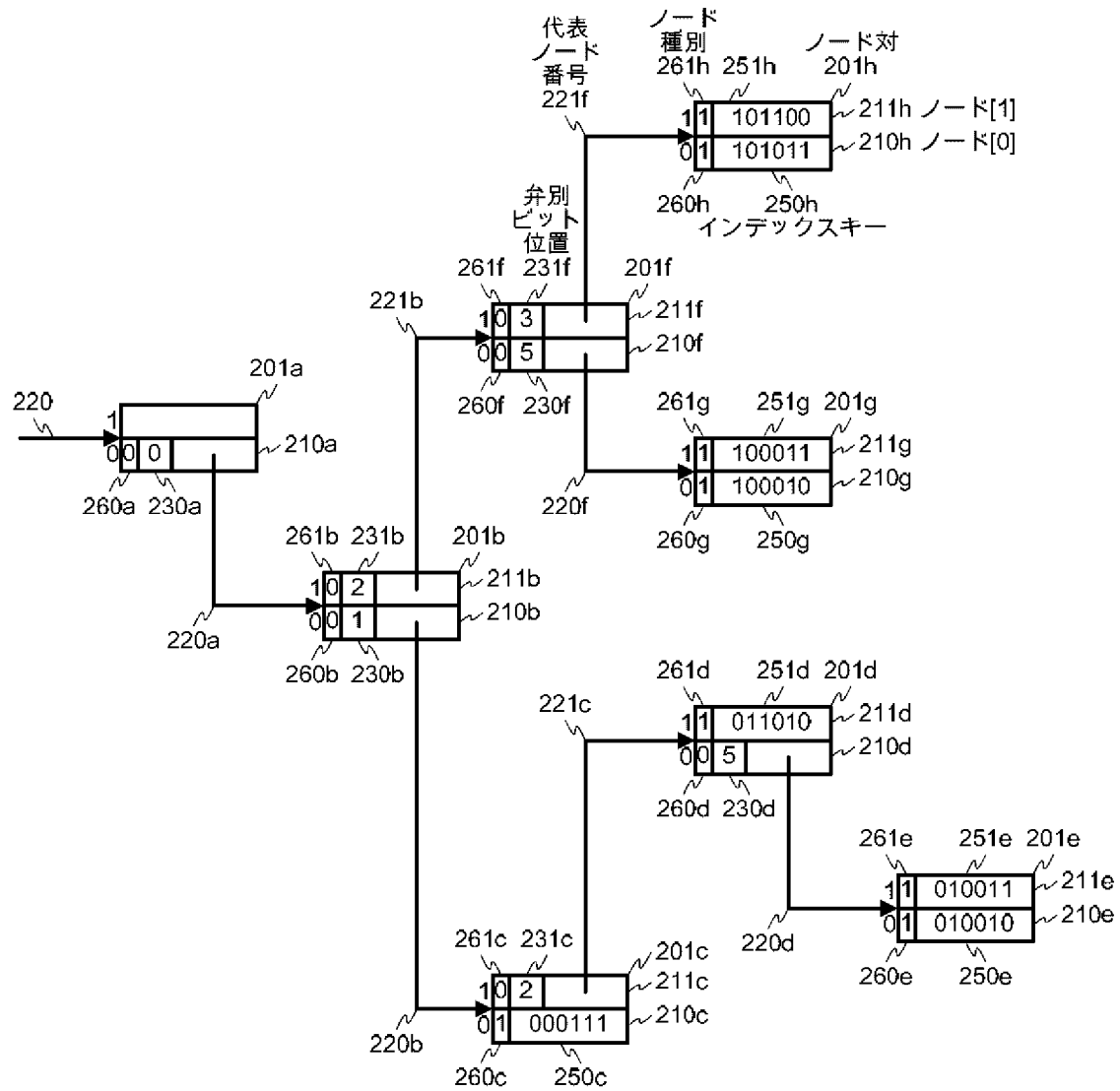
[図1]



[図2A]

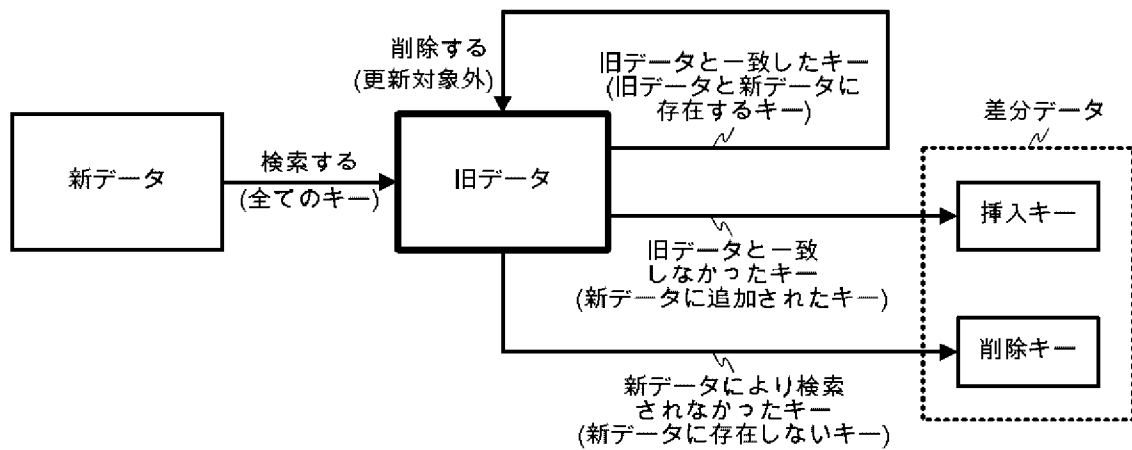


[図2B]

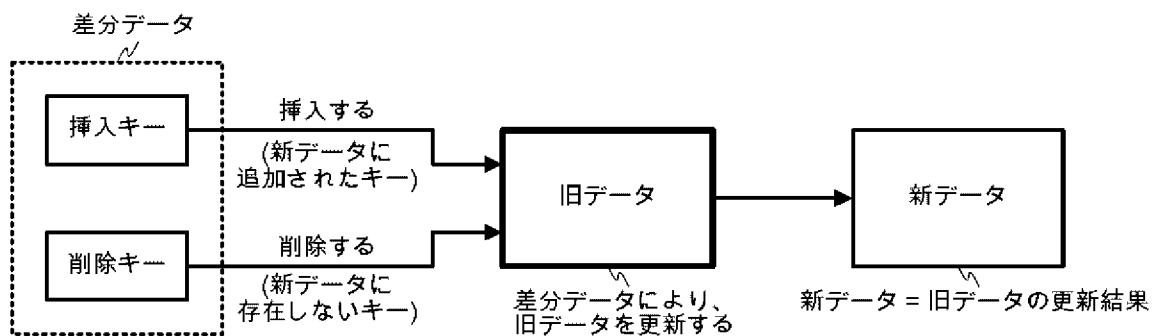


[図3A]

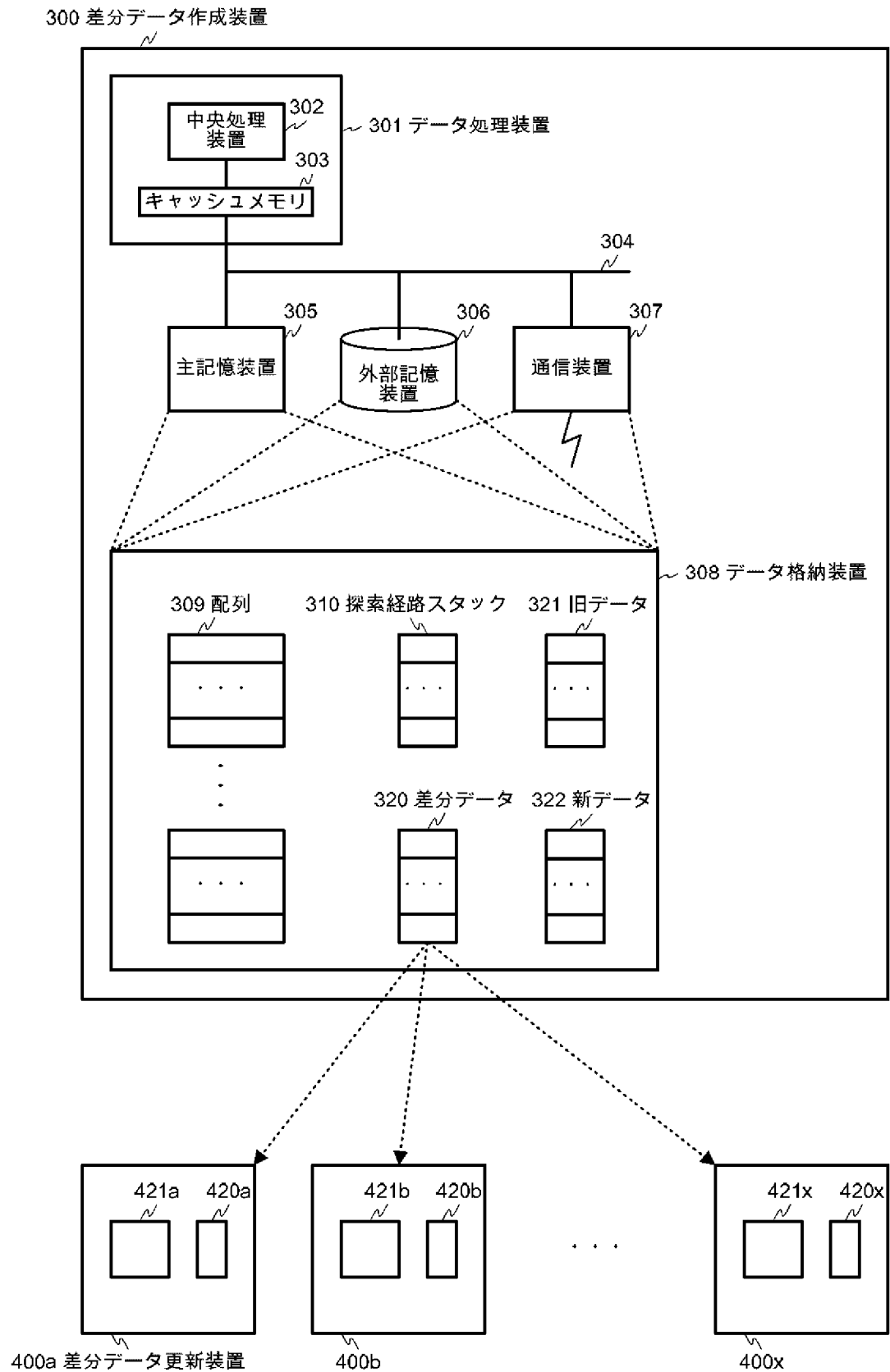
(1)



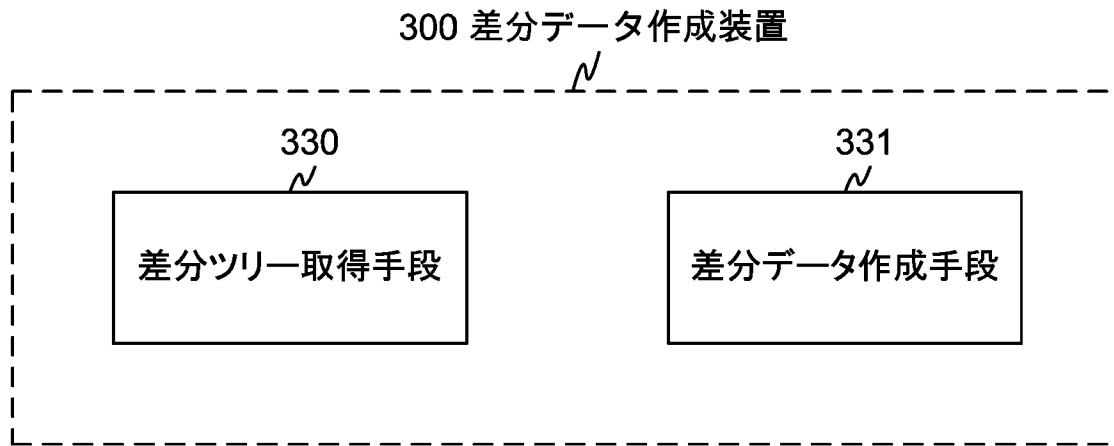
(2)



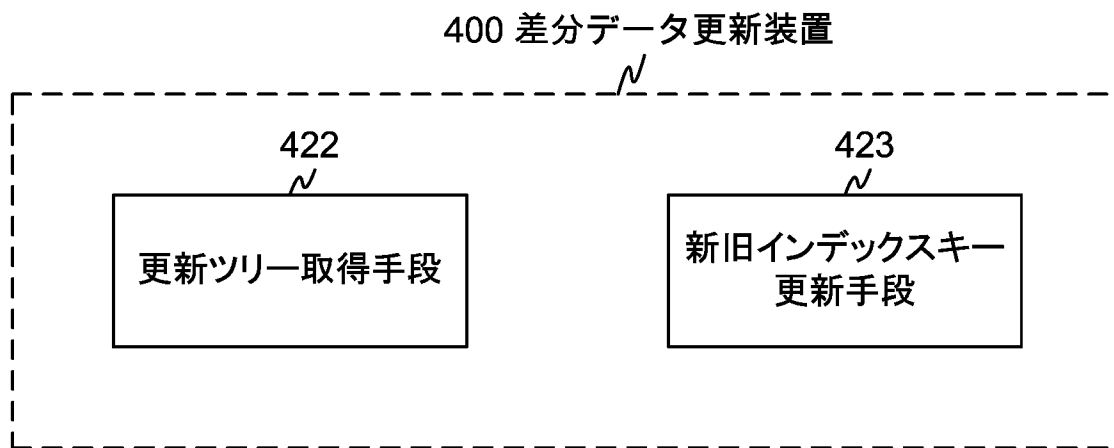
[図3B]



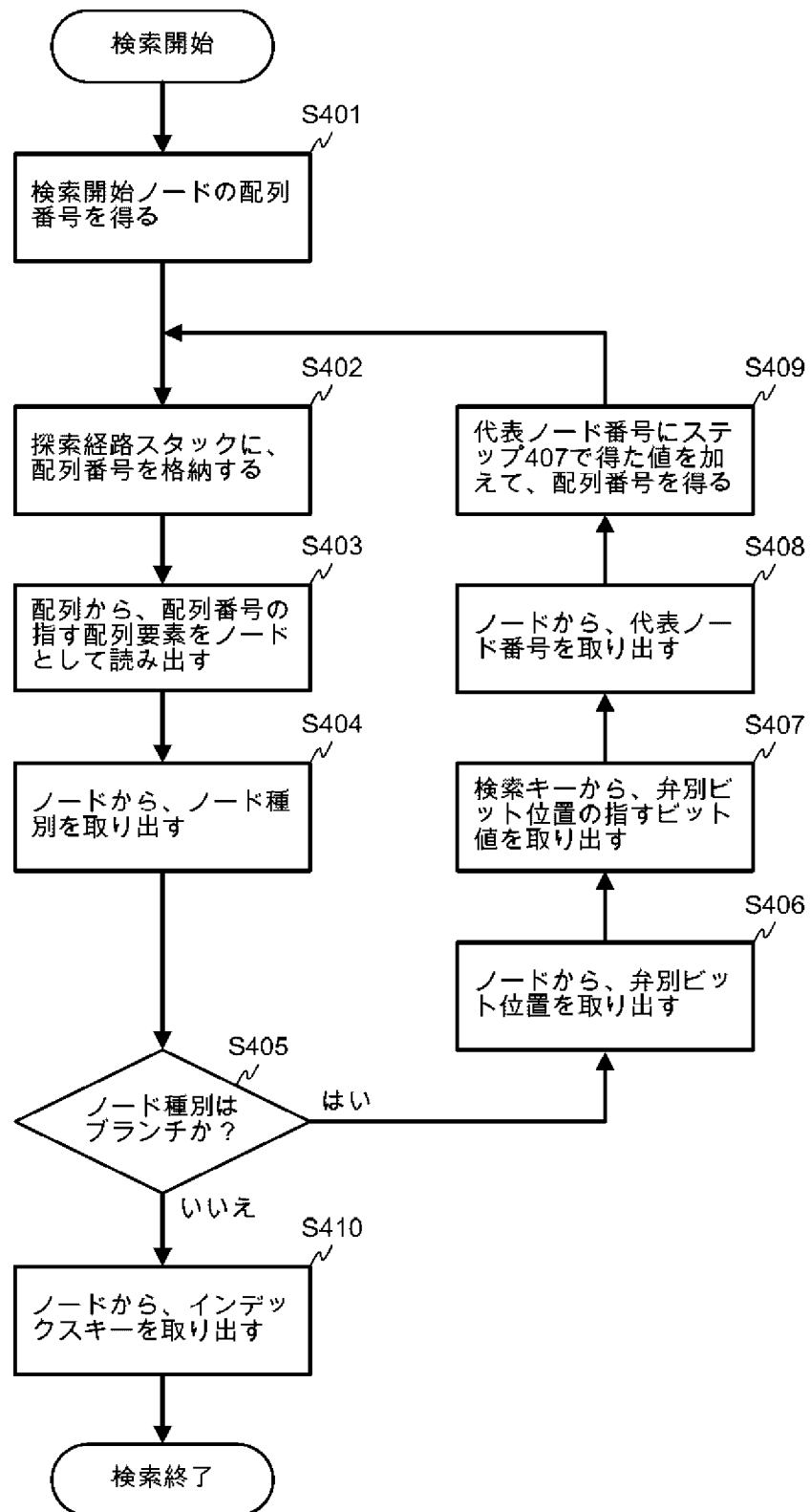
[図3C]



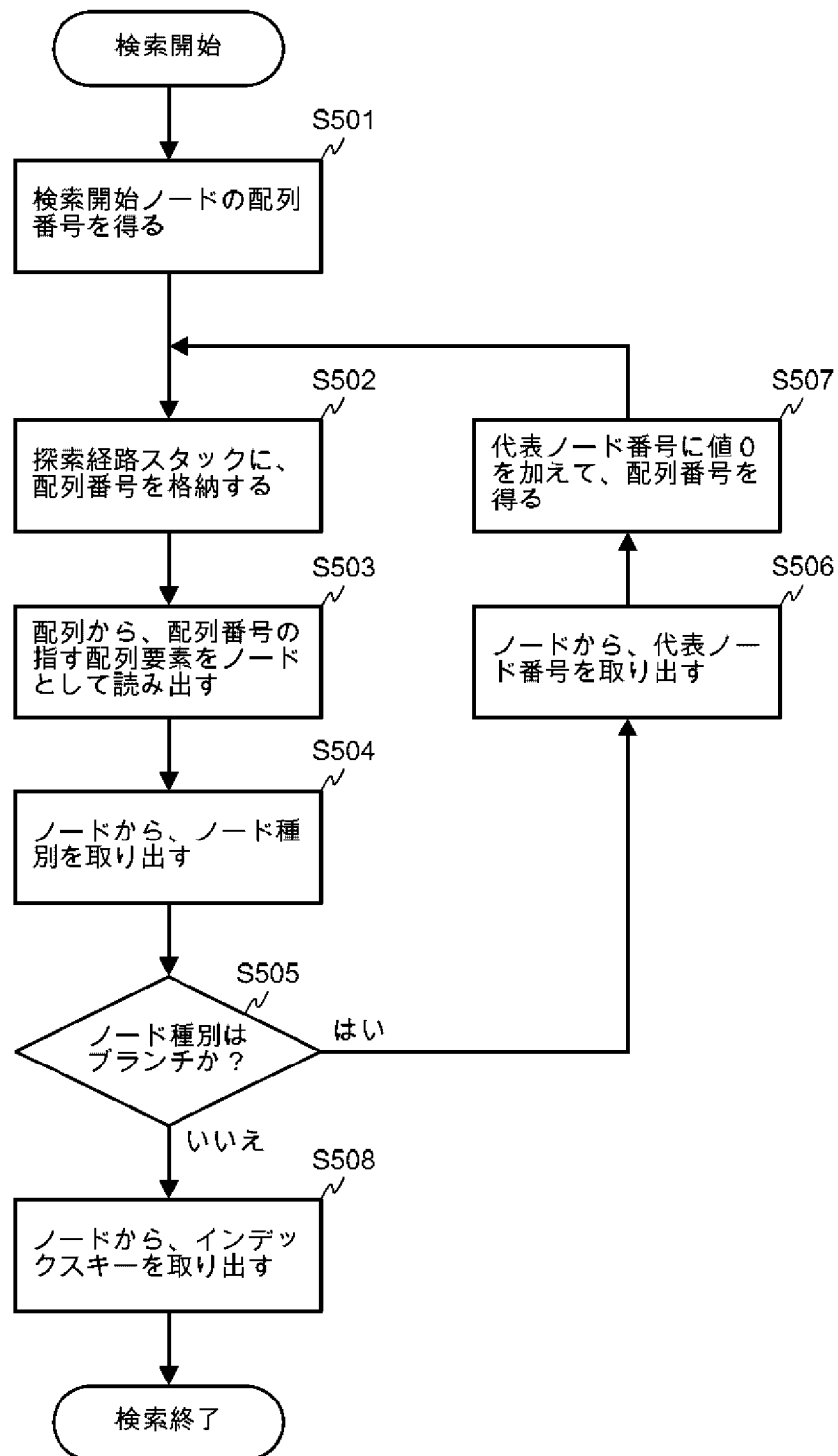
[図3D]



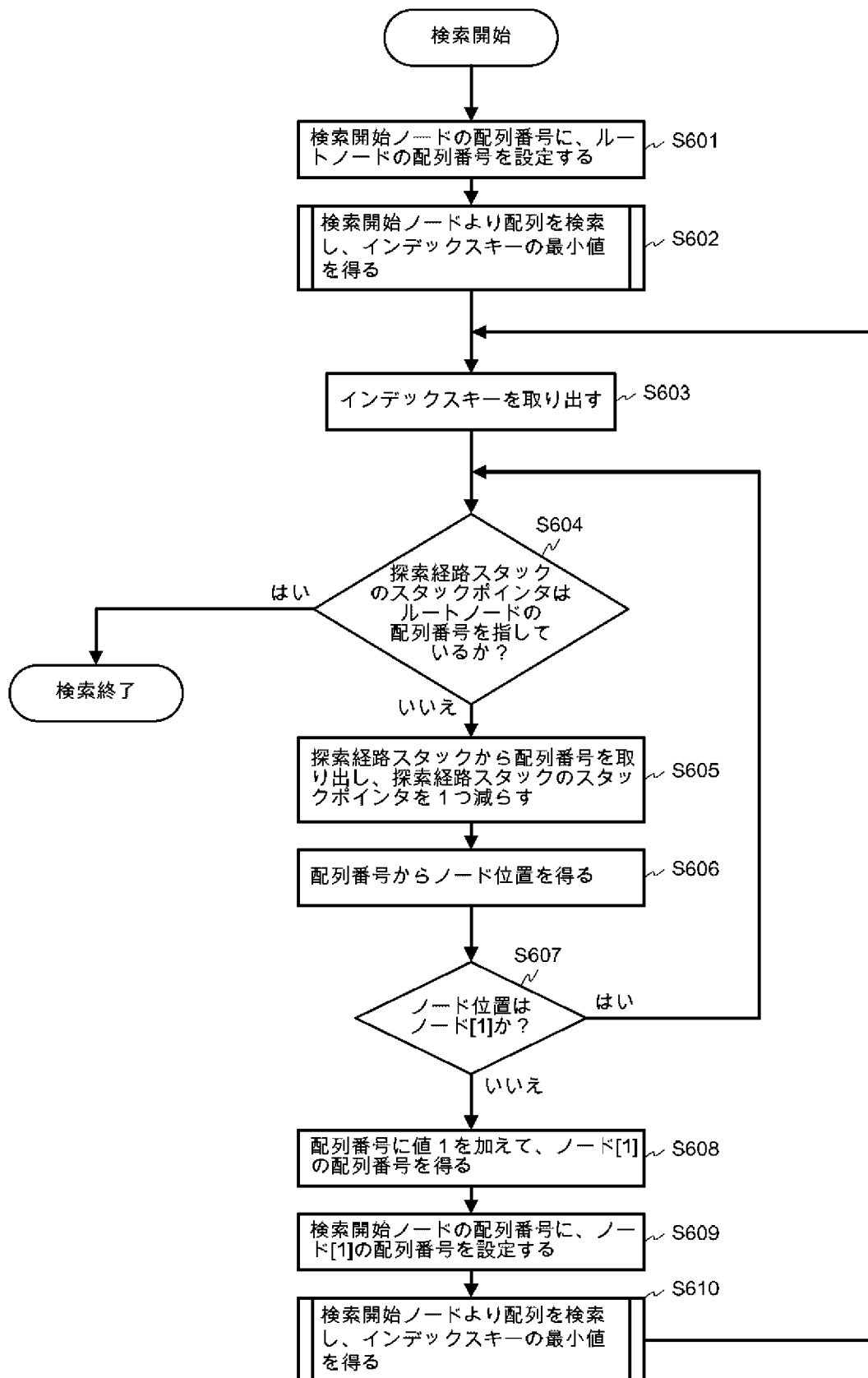
[図4]



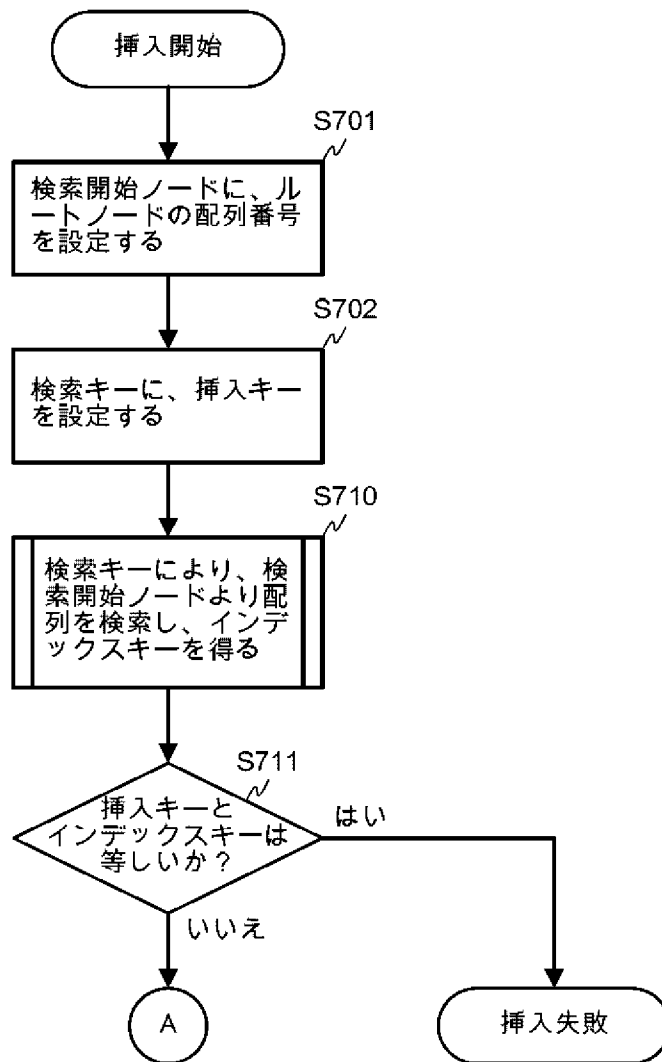
[図5]



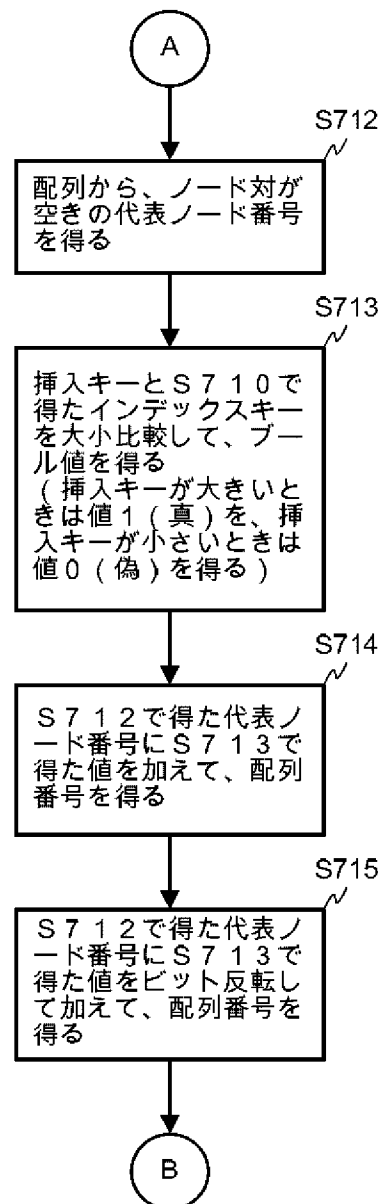
[図6]



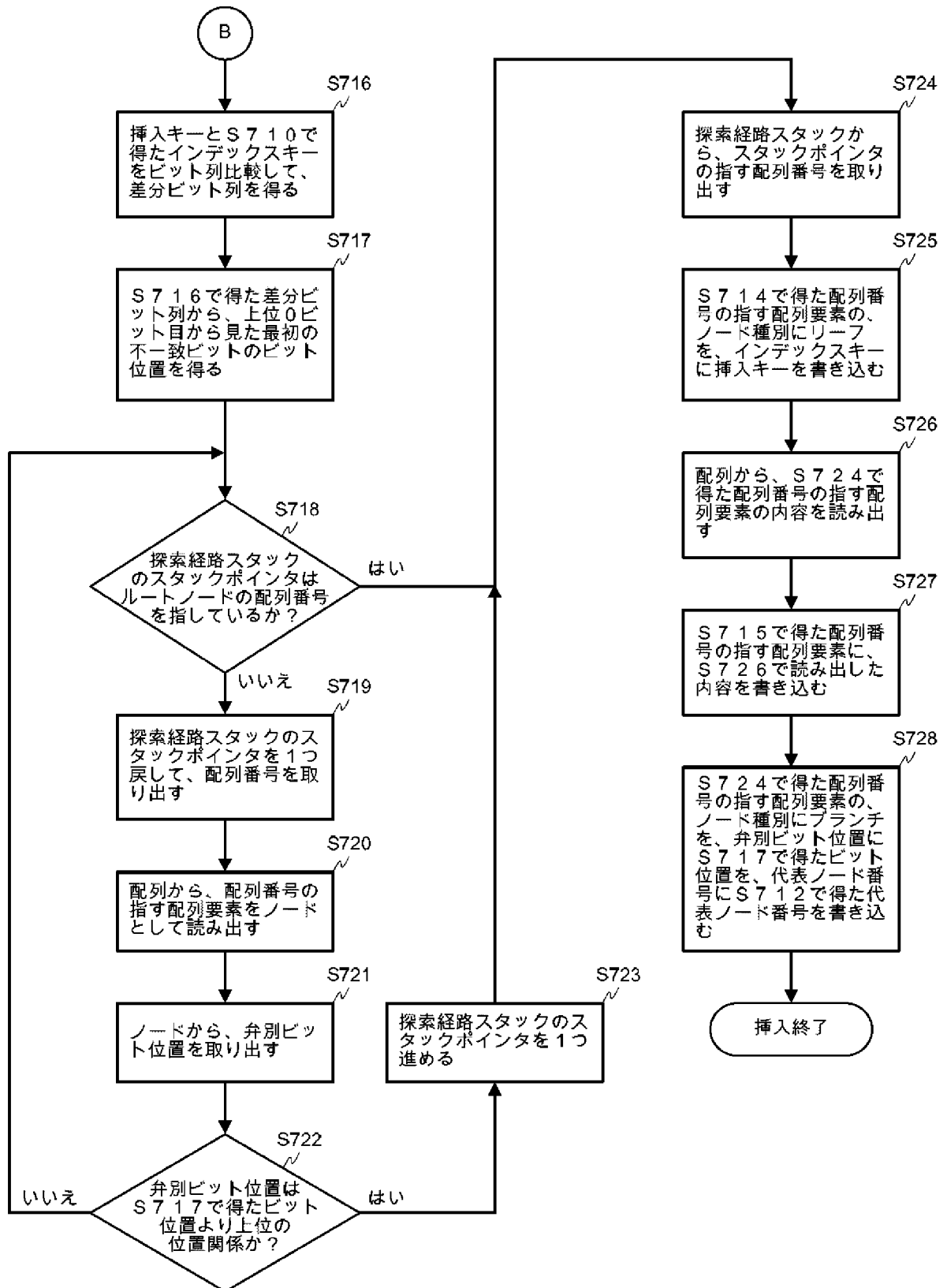
[図7A]



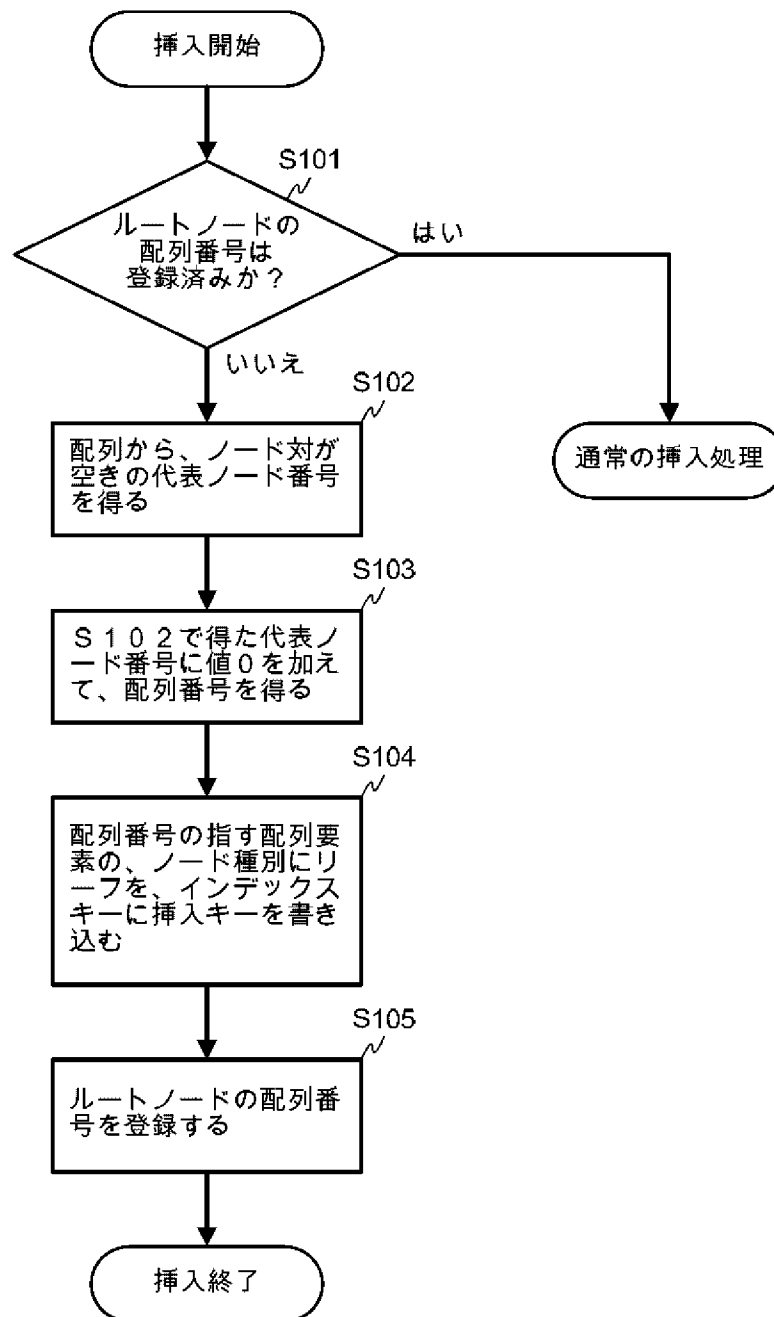
[図7B]



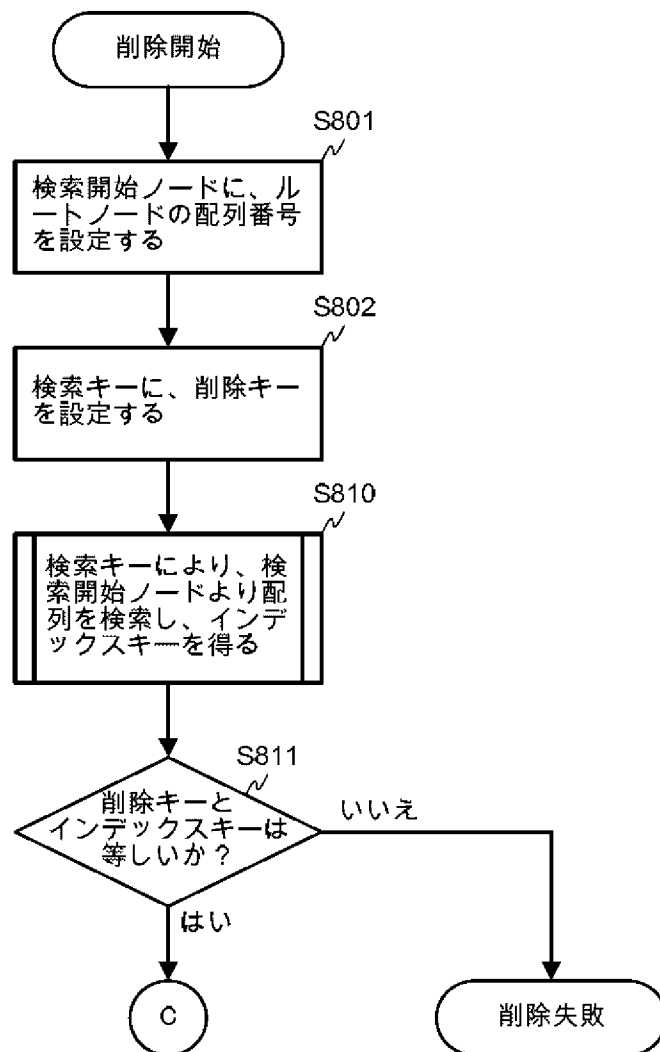
[図7C]



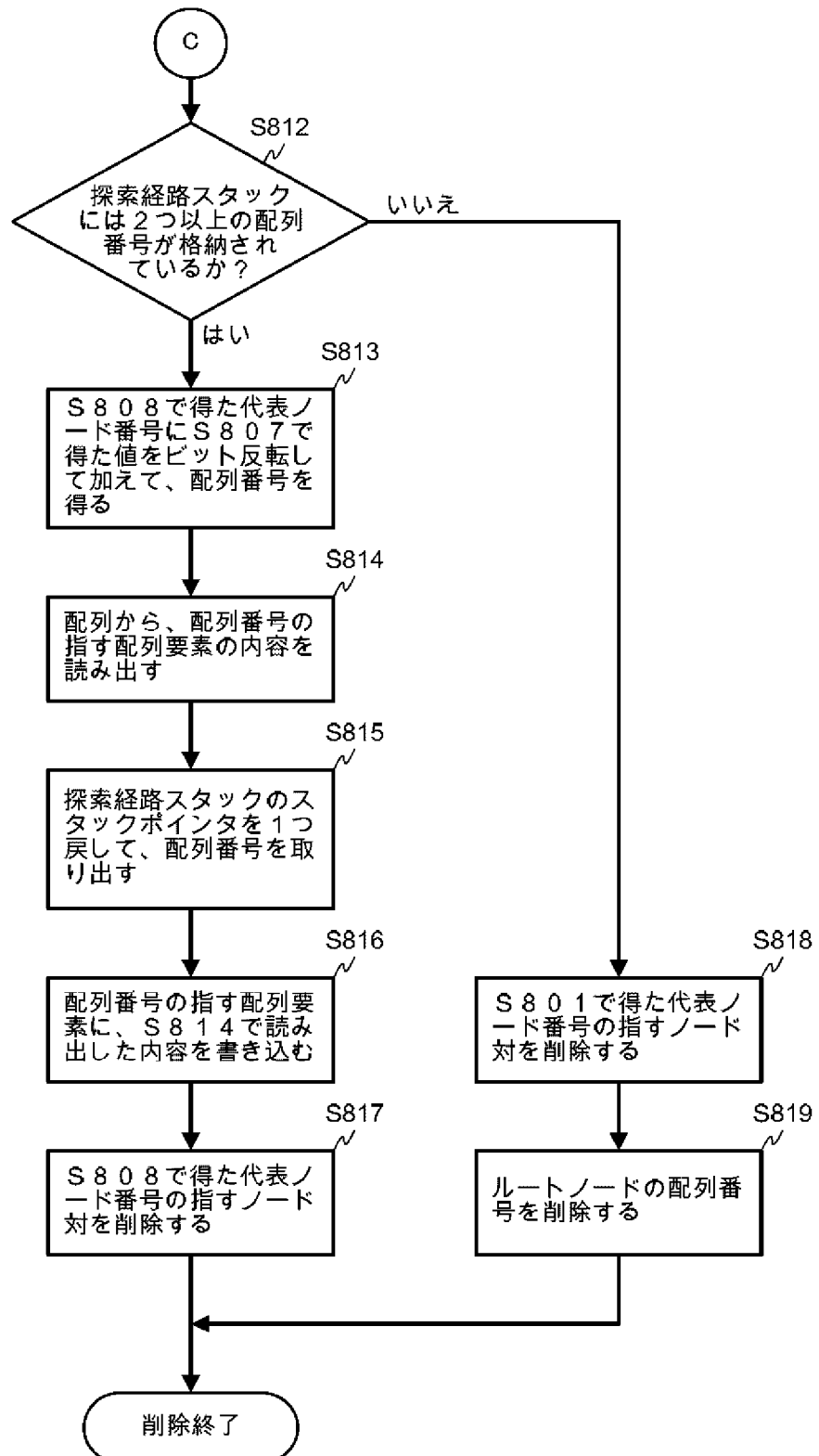
[図7D]



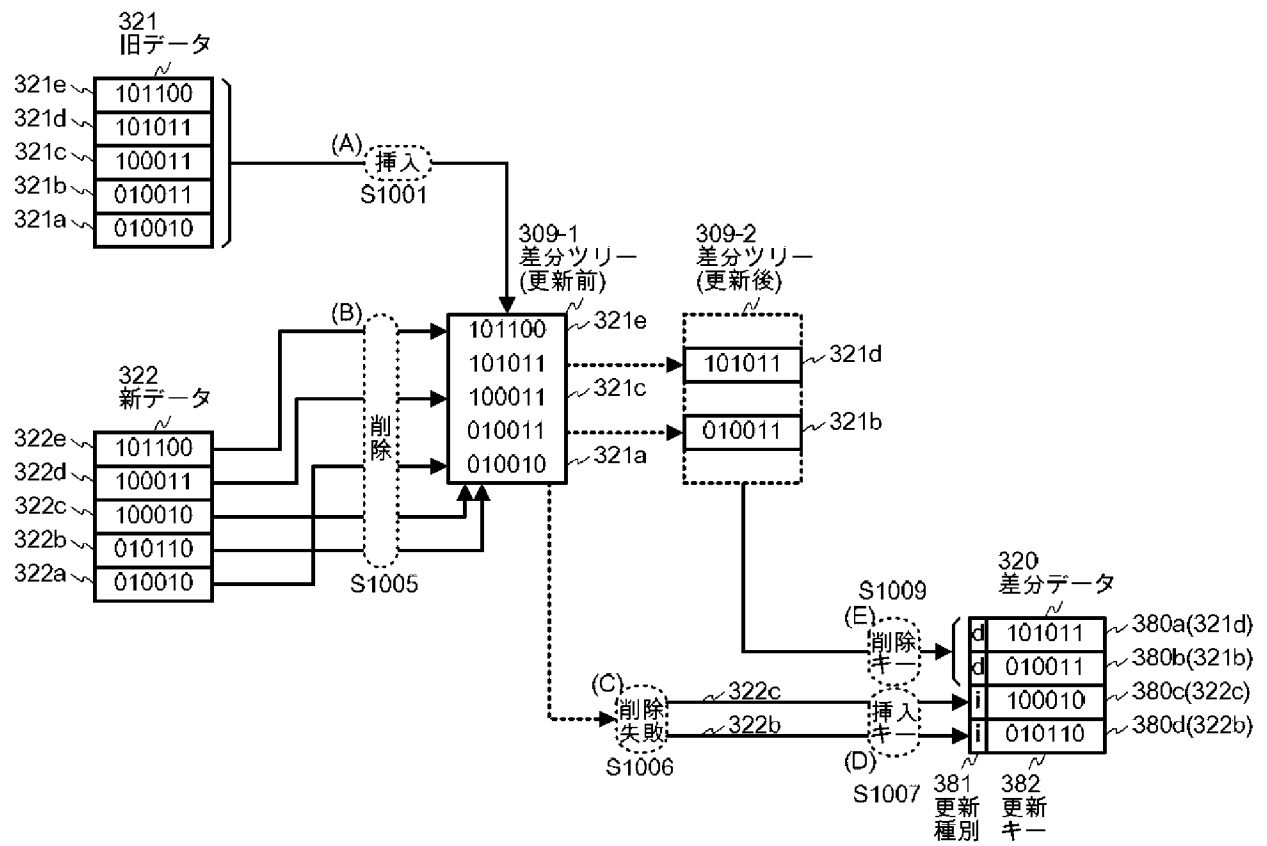
[図8A]



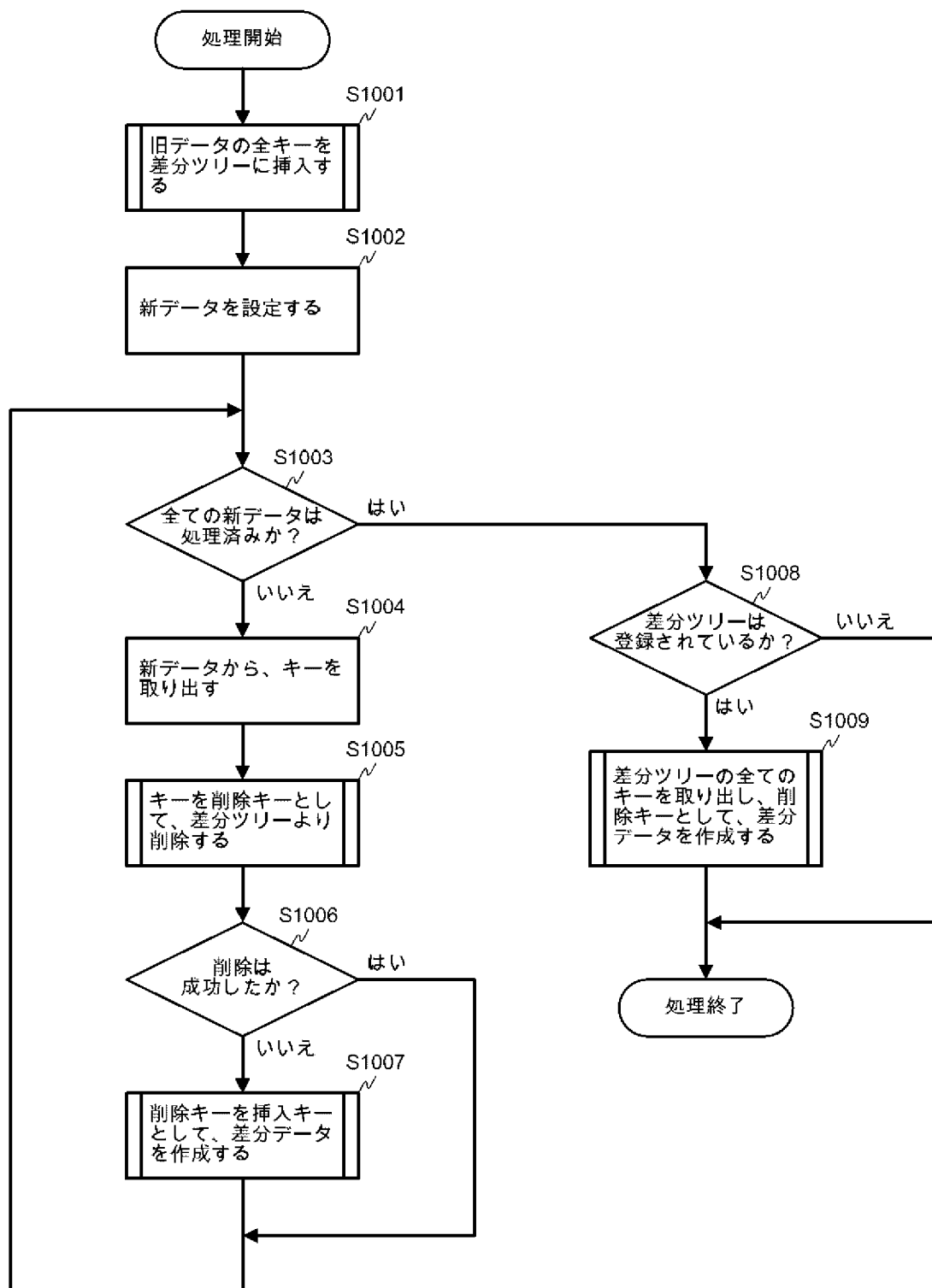
[図8B]



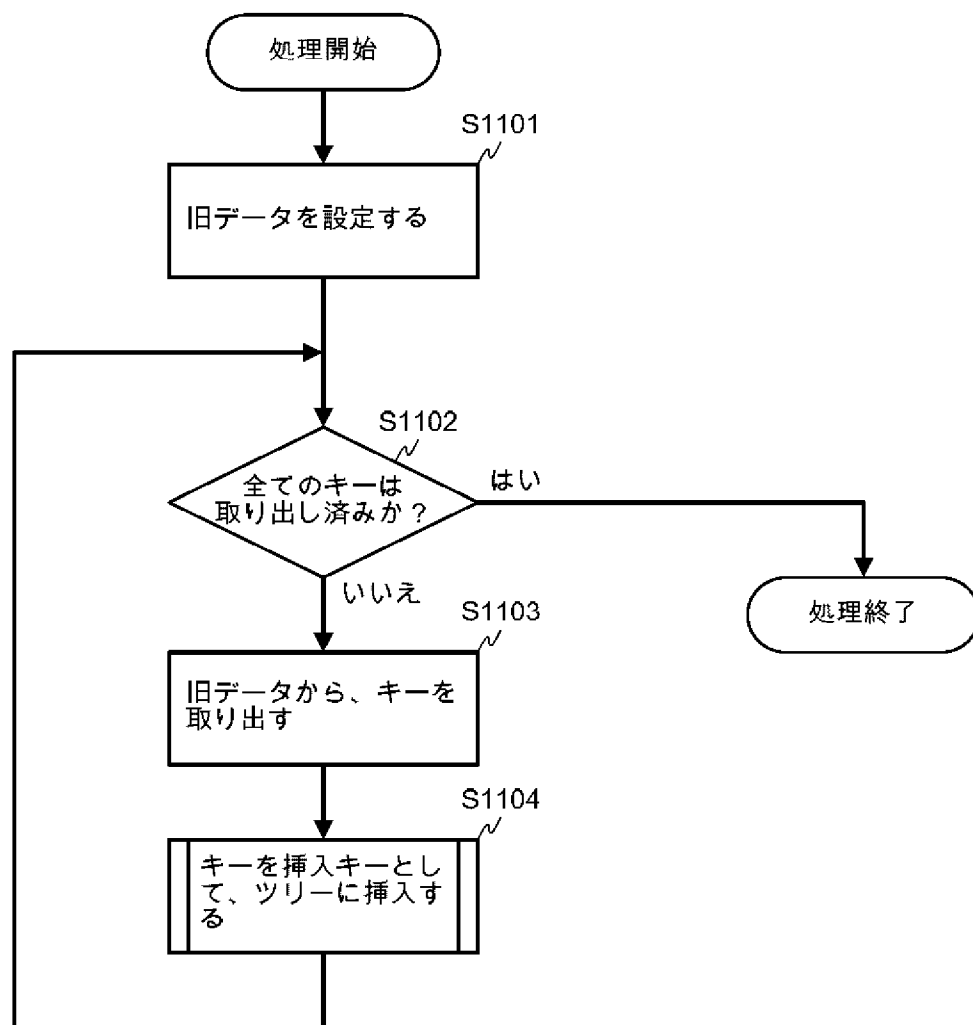
[図9]



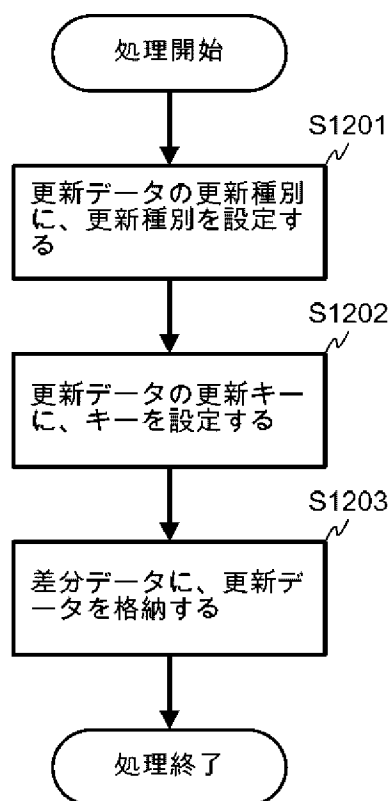
[図10]



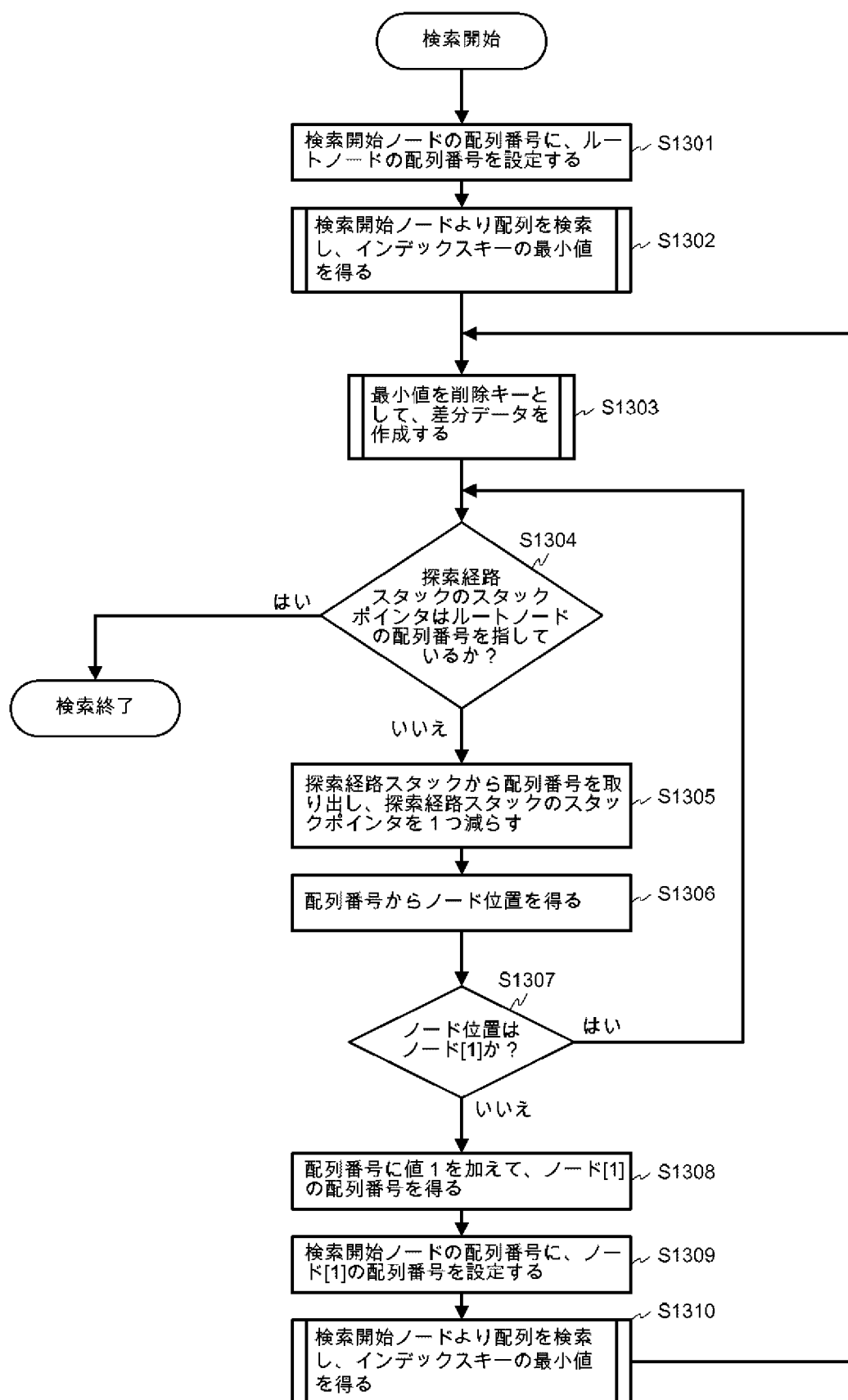
[図11]



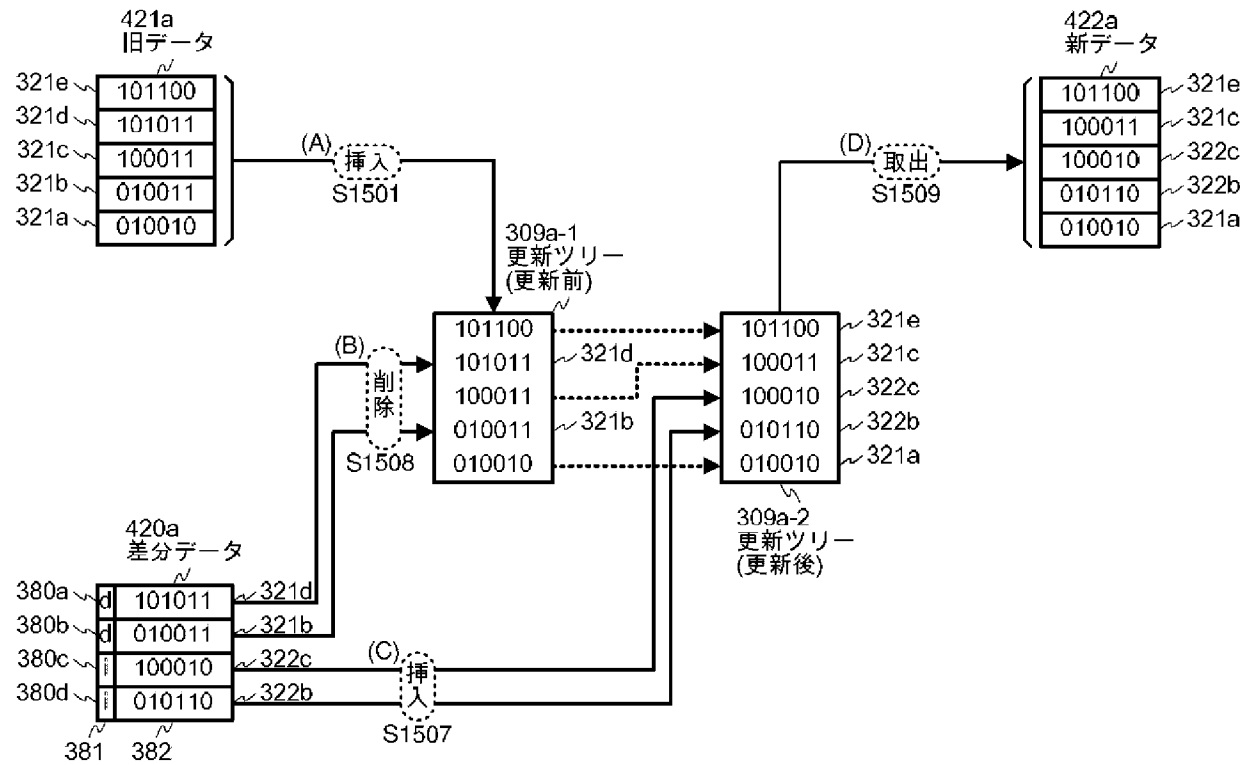
[図12]



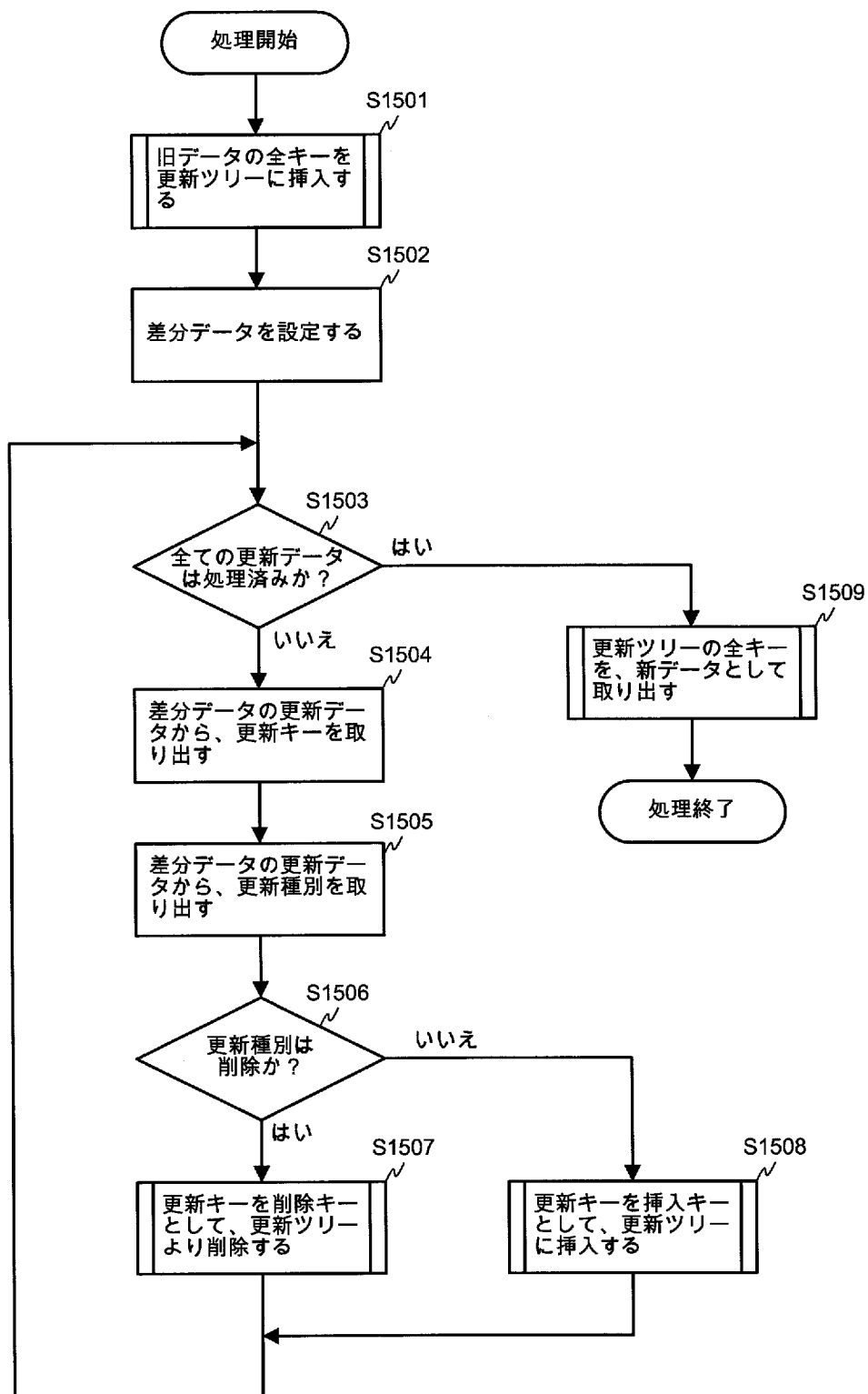
[図13]



[図14]



[図15]



INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP2008/003123

A. CLASSIFICATION OF SUBJECT MATTER

G06F17/30(2006.01) i, G06F12/00(2006.01) i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F17/30, G06F12/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Jitsuyo Shinan Koho 1922-1996 Jitsuyo Shinan Toroku Koho 1996-2008

Kokai Jitsuyo Shinan Koho 1971-2008 Toroku Jitsuyo Shinan Koho 1994-2008

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP 2001-357070 A (NEC Corp.), 26 December, 2001 (26.12.01), Full text; all drawings (Family: none)	1-14
A	Shigeaki YAZAKI, "Zu de Wakaru! Programming no 10 Dai Kiso Chishiki", Nikkei Software, 24 January, 2003 (24.01.03), Vol.6, No.2, pages 44 to 45	3, 4, 6-8
A	Daichi GOTO, "Sawatte Manabu Data Kozo, Tsukatte Minitsuku Algorithm Kochira Java API Kenkyusho! Dai 6 Kai Nibunki to Nibun Tansakuki", JAVA PRESS, 15 February, 2004 (15.02.04), Vol.34, pages 202 to 210	1-14

☒ Further documents are listed in the continuation of Box C.☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search
21 November, 2008 (21.11.08)Date of mailing of the international search report
02 December, 2008 (02.12.08)Name and mailing address of the ISA/
Japanese Patent Office

Authorized officer

Facsimile No.

Telephone No.

INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP2008/003123

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
P, Y	WO 2008/004335 A1 (S. Grants Co., Ltd.), 10 January, 2008 (10.01.08), Full text; all drawings & JP 2008-15872 A	1-14
P, Y	WO 2008/053583 A1 (S. Grants Co., Ltd.), 08 May, 2008 (08.05.08), Full text; all drawings & JP 2008-112240 A	1-14
P, Y	WO 2008/065735 A1 (S. Grants Co., Ltd.), 05 June, 2008 (05.06.08), Full text; all drawings & JP 2008-159025 A	1-14
P, Y	WO 2008/090588 A1 (S. Grants Co., Ltd.), 31 July, 2008 (31.07.08), Full text; all drawings & JP 2008-181260 A	1-14
E, Y	WO 2008/132806 A1 (S. Grants Co., Ltd.), 06 November, 2008 (06.11.08), Full text; all drawings & JP 2008-269197 A & JP 2008-269503 A	1-14

A. 発明の属する分野の分類（国際特許分類（I P C））
Int.Cl. G06F17/30(2006.01)i, G06F12/00(2006.01)i

B. 調査を行った分野

調査を行った最小限資料（国際特許分類（I P C））

Int.Cl. G06F17/30, G06F12/00

最小限資料以外の資料で調査を行った分野に含まれるもの

日本国実用新案公報	1 9 2 2 - 1 9 9 6 年
日本国公開実用新案公報	1 9 7 1 - 2 0 0 8 年
日本国実用新案登録公報	1 9 9 6 - 2 0 0 8 年
日本国登録実用新案公報	1 9 9 4 - 2 0 0 8 年

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

C. 関連すると認められる文献

引用文献の カテゴリー*	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する 請求の範囲の番号
A	JP 2001-357070 A（日本電気株式会社）2001. 12. 26, 全文・全図（ファミリーなし）	1-14
A	矢崎茂明, 図でわかる！プログラミングの10大基礎知識, 日経ソフトウェア, 2003. 01. 24, 第6巻 第2号, pp. 44~45	3, 4, 6-8
A	後藤大地, さわって学ぶデータ構造、つかって身につくアルゴリズム こちら J a v a A P I 研究所！ 第6回 二分木と二分探索木, J A V A P R E S S, 2004. 02. 15, 第34巻, pp. 202~210	1-14

☒ C欄の続きにも文献が列挙されている。

☐ パテントファミリーに関する別紙を参照。

* 引用文献のカテゴリー

「A」特に関連のある文献ではなく、一般的技術水準を示すもの
「E」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの
「L」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）
「O」口頭による開示、使用、展示等に言及する文献
「P」国際出願日前で、かつ優先権の主張の基礎となる出願

の日の後に公表された文献

「T」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの
「X」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
「Y」特に関連のある文献であって、当該文献と他の1以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
「&」同一パテントファミリー文献

国際調査を完了した日

2 1 . 1 1 . 2 0 0 8

国際調査報告の発送日

0 2 . 1 2 . 2 0 0 8

国際調査機関の名称及びあて先

日本国特許庁（I S A / J P）

郵便番号100-8915

東京都千代田区霞が関三丁目4番3号

特許庁審査官（権限のある職員）

池田 聡史

電話番号 03-3581-1101 内線 3599

5M

9475

C (続き) . 関連すると認められる文献		
引用文献の カテゴリー*	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する 請求の範囲の番号
P, Y	WO 2008/004335 A1 (株式会社エスグランツ) 2008.01.10, 全文・全図 & JP 2008-15872 A	1-14
P, Y	WO 2008/053583 A1 (株式会社エスグランツ) 2008.05.08, 全文・全図 & JP 2008-112240 A	1-14
P, Y	WO 2008/065735 A1 (株式会社エスグランツ) 2008.06.05, 全文・全図 & JP 2008-159025 A	1-14
P, Y	WO 2008/090588 A1 (株式会社エスグランツ) 2008.07.31, 全文・全図 & JP 2008-181260 A	1-14
E, Y	WO 2008/132806 A1 (株式会社エスグランツ) 2008.11.06, 全文・全図 & JP 2008-269197 A & JP 2008-269503 A	1-14