(12) 特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局

(43) 国際公開日
2015 年 9 月 3 日(03.09.2015)

WIPO | PCT

(10) 国際公開番号
WO 2015/129109 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/JP2014/080851

(22) 国際出願日: 2014 年 11 月 21 日(21.11.2014)

(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 2014-036341 2014 年 2 月 27 日(27.02.2014) JP

(71) 出願人: ウイングアーク１ｓｔ株式会社 (WINGARC1ST INC.) [JP/JP]; 〒1500031 東京都渋谷区桜丘町２０番１号 Tokyo (JP).

(72) 発明者: 佐々木　盛朗 (SASAKI, Shigero); 〒1500031 東京都渋谷区桜丘町２０番１号　ウイングアーク１ｓｔ株式会社内 Tokyo (JP).

(74) 代理人: 橘　和之 (TACHIBANA, Kazuyuki); 〒1020083 東京都千代田区麹町１丁目４番地　半蔵門ファーストビル３階 Tokyo (JP).

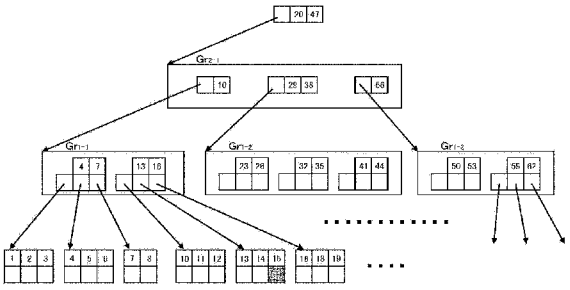
(81) 指定国 (表示のない限り、全ての種類の国内保護が可能): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK,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

(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, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), ユーラシア (AM, AZ, BY, KG, KZ, RU, TJ, TM), ヨーロッパ (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

添付公開書類:
— 国際調査報告（条約第 21 条(3)）

(54) **Title**: INDEX MANAGEMENT DEVICE

(54) 発明の名称 : インデックス管理装置



(57) **Abstract**: The present invention manages the searching and refreshing of an index tree using a first type of node for storing a set of a prescribed number of keys and a prescribed number of pointers indicating the position of a child node or a prescribed number of values, for lower levels at or below an nth level (n is an arbitrary value satisfying "1≤n<number of all levels-1"), by using the quality of the access rate being higher and the refresh rate being lower in higher level nodes and the access rate being lower and the refresh rate being higher in lower level nodes. Meanwhile, the present invention makes it possible to quickly perform data search processing and refresh processing, by managing the searching and refreshing of the index tree using a second type of node for storing a prescribed number of keys and one group pointer indicating the head position of a lower level node group, for higher levels above the nth level.

(57) 要約:

[続葉有]

WO 2015/129109 A1

上位階層のノードほどアクセス比率が高い代わりに更新比率が低く、下位階層のノードほどアクセス比率が低い代わりに更新比率が高いという性質を利用して、第ｎ階層以下（ｎは「１≦ｎ＜全階層数－１」を満たす任意の値）の下位階層において、所定数のキーと子ノードの位置を表す所定数のポインタまたは所定数のバリューとの組を格納した第１の種類のノードによってインデックスツリーの探索および更新を管理する。一方、第ｎ階層よりも上の上位階層においては、所定数のキーと下階層のノードグループの先頭位置を表す１つのグループポインタとを格納した第２の種類のノードによってインデックスツリーの探索および更新を管理することにより、データの検索処理および更新処理を共に高速化できるようにする。

明　細　書

発明の名称 ： インデックス管理装置

技術分野

[0001] 本発明は、インデックス管理装置に関し、特に、データの検索を高速化するために用いるインデックスツリーを管理するインデックス管理装置に用いて好適なものである。

背景技術

[0002] 従来、データの検索を高速化する技術として、インデックスツリーと呼ばれる手法が広く知られている。例えば、特定のキーに対応するデータを検索する場合、データベース内の全てのレコードを先頭から１つずつ調べていくと膨大な時間がかかってしまう。そこで、特定のキーに対する検索を高速化するために、インデックスツリーを付与することが多い（例えば、特許文献１，２参照）。

[0003] 記録するデータの組をレコードと呼び、その中で検索に用いられるデータを特にキーと呼ぶ。その他のデータはバリューと呼ぶ。レコードをキーで検索するには、キー順にレコードがソートされているのが望ましい。しかし、レコードをキー順にソートして記録するのは時間がかかる処理である。そこで、レコードは到着順に記録し、キーと対応するレコードへのポインタをツリー構造でソートして別途記録するのが一般的である。これがインデックスツリーである。ソートした状態をツリー構造で維持するのは、レコードの追加と削除に伴うインデックスツリーのキーの追加と削除を、一部に限定することで処理時間を短縮するためである。

[0004] 図８は、インデックスツリーの概念を説明するための図である。インデックスツリーは、ツリー状の構造を持ち、最下層のノードをリーフノード、その他のノードを内部ノードと呼ぶ。また、一番上のノードをルートノードと呼び、ルートでもリーフでもないノードをブランチノードと呼ぶ。図８ではブランチノードが１階層となっているが、複数階層にすることも可能である

。各ノードには所定数のキー１０１とポインタ１０２との組が格納されているが、内部ノードについては左端のキーが省略されている。

[0005] 各ノードのエントリであるキーとポインタとの組は、キーの値の昇順もしくは降順で並んでいる。これらのエントリはそれぞれ、そのノードの子に相当するノードと１対１に対応し、子ノードの左端のキー（子ノードが内部ノードの場合は左端の省略されているキー）の値と、子ノードの位置を指すポインタとを格納する。ノードの最終階層であるリーフノードのエントリには、各レコードのキーの値とレコードの位置とを格納する。

[0006] 図８の例において、ルートノードには２つのキー“１０”、“１９”と３つのポインタとの組が格納されている。３つのポインタのうち、１つ目（左端）のポインタは、値が“１”以上で“１０”より小さいキーをエントリとして持つ子ノードの格納位置を表す位置情報である。２つ目のポインタは、値が“１０”以上で“１９”より小さいキーをエントリとして持つ子ノードの格納位置を表す位置情報である。３つ目のポインタは、値が“１９”以上のキーをエントリとして持つ子ノードの格納位置を表す位置情報である。

[0007] また、左端のブランチノードには２つのキー“４”、“７”と３つのポインタとの組が格納されている。３つのポインタのうち、１つ目のポインタは、値が“１”以上で“４”より小さいキーをエントリとして持つ子ノードの格納位置を表す位置情報である。２つ目のポインタは、値が“４”以上で“７”より小さいキーをエントリとして持つ子ノードの格納位置を表す位置情報である。３つ目のポインタは、値が“７”以上のキーをエントリとして持つ子ノードの格納位置を表す位置情報である。他のブランチノードも同様に、２つのキーと３つのポインタとの組が格納されている。

[0008] さらに、左端のリーフノードには３つのキー“１”、“２”、“３”と３つのポインタとの組が格納されている。３つのポインタのうち、１つ目のポインタは、キー“１”に対応するデータが格納されているレコードの位置を表す位置情報である。２つ目のポインタは、キー“２”に対応するデータが格納されているレコードの位置を表す位置情報である。３つ目のポインタは

、キー“３”に対応するデータが格納されているレコードの位置を表す位置情報である。他のリーフノードも同様に、３つのキーと３つのポインタとの組が格納されている。

[0009] 図８のように構成されたインデックスツリーを用いて、例えばキー“１１”に対応するデータを検索する場合、ルートノードにおける２番目のポインタ、２番目のブランチノードにおける左端のポインタおよび４番目のリーフノードを辿って、キー“１１”に対応するデータを効率的に検索することができる。

[0010] ところで、レコード内のあるフィールドに対してインデックスを作成した場合、レコードの追加や削除といった更新系の処理をしたときに、レコード自体だけでなくインデックスの内容も更新する必要がある。レコードを追加する場合は、先の検索の場合と同じようにルートノードから順に辿ってエントリを追加するリーフノードを探し出す。ノードに空きがあるなら、昇順または降順の順序を守ってエントリを追加するだけでインデックスの追加は終了する。

[0011] 一方、ノードに空きがないときは、新たにノードを追加して空きエントリを作る必要がある。例えば、図９（ａ）に示すようにインデックスツリーが構成されたフィールドにおいて、キー“８”のレコードを追加する場合、エントリを追加すべきノードとしてルートノードから順に辿って探索した３番目のリーフノードには、既に３つのエントリが格納されていて、空きがない。

[0012] この場合は、図９（ｂ）に示すように、現在の３番目のリーフノードにある３つのキーのうち、分割キー“８”以上のキー“９”の位置で３番目のリーフノードを分割して新たにリーフノードを生成し、分割キー“８”よりも前側のエントリを１つ目の分割ノードに移動し、分割キー“８”よりも後側のエントリを２つ目の分割ノードに移動することにより、キー“８”のレコードを追加可能な空きエントリを生成する。

[0013] ところが、図９（ｂ）のようにツリーを下層側に成長させると、ルートノ

ードからリーフノードまでの階層数が一部のみ変化し、全体としてバランスしない状態となる。このようにバランスが崩れたインデックスツリーでは、検索効率が低下してしまう。そこで、インデックスツリーの一方式である「Ｂツリー」（例えば、非特許文献１参照）では、図１０に示すように、ノードを分割する際にツリーを上層側に成長させるようにしている。このようにすれば、インデックスツリーの階層数はどのリーフノードに対しても同じとなり、全体としてバランスする。

[0014] データベースがハードディスクに格納される場合、ディスクＩ／Ｏの数が、ディスク環境でのインデックスツリーの性能を決定する。すなわち、ディスクのレイテンシは１０ｍｓ程度、メモリのレイテンシは１００ｎｓ程度、キャッシュのレイテンシは１ｎｓ程度であるから、検索効率を上げるには、ディスクＩ／Ｏの数をできるだけ少なくすることが必要となる。

[0015] 一方、インデックスツリーはその構造上、上位階層のノードほどアクセス比率が高くなり、下位階層のノードほどアクセス比率は低くなる（例えば、非特許文献２参照）。したがって、最上位のルートノードをキャッシュに格納し、ブランチノードをメモリに格納し、リーフノードをディスクに格納すれば、ディスクＩ／Ｏの数を減らすことが可能である。特にＢツリーは、ディスクのアクセスに最適化された方式である。

[0016] すなわち、Ｂツリーのノードサイズは、典型的にはディスクブロック（ディスク上のデータのＩ／Ｏの単位で、通例では４Ｋバイト）の大きさに等しい。ノードに格納されるのが４バイトのキーと４バイトのポインタの場合、ノードサイズを４Ｋバイトにすれば、ファンアウト（子ノードの数）は約５００となる。したがって、図１１のように３階層から成るＢツリーの場合、少量（２Ｍバイト程度）のメモリがあれば、多量（１Ｇバイト程度）のデータベースから、１回のディスクＩ／Ｏでデータを取り出すことが可能である。

[0017] これに対して、全てのデータがメモリに格納されたインメモリ環境では、ディスクＩ／Ｏがなくなるので、Ｂツリーの性能は、アクセスするキャッシ

ュラインの数に強く依存する。キャッシュラインとは、ＣＰＵがメモリからキャッシュへとデータを移送する単位である。近年のＣＰＵでは、６４バイトのデータでキャッシュラインを構成することが多い。

[0018] このライン数を削減してインメモリ環境に最適化したインデックスツリーとして、「ＣＳＢ＋ツリー」と呼ばれる方式が知られている（例えば、非特許文献３参照）。ＣＳＢ＋ツリーは、ノードのエントリからポインタを削除することによって記憶容量を削減し、そのぶん１つのノードに格納可能なキーの数を増やすことによってライン数を削減できるようにしたものである。また、ＣＳＢ＋ツリーには、ポインタを記録したキャッシュラインへのアクセスを省略できるという利点もある。

[0019] 図１２に示すように、ＣＳＢ＋ツリーでは、複数のノードをまとめてノードグループを生成する。内部ノードのエントリは、各キーに個別に対応するポインタを持たず、下階層のノードグループの先頭位置を表すポインタのみを持つ。グループ内で各ノードのエントリはメモリの連続領域に格納されており、子ノードの先頭位置からのオフセット量に基づいて該当するキーの位置が特定される。

[0020] 例えば、キー“１３”に対応するデータを検索する場合、ルートノードからの探索によって、キー“１３”はリーフノードにおける２番目のノードグループ内にあることが分かる。ここで、２番目のノードグループの先頭アドレスが“０ｘＡ０００”であったとする。また、ノードサイズが１２バイトであったとすると、キー“１３”に対応するアドレスは“０ｘＡ００Ｃ”（＝０ｘＡ０００＋０ｘ０００Ｃ×１）と計算される。

[0021] ＣＳＢ＋ツリーのようにポインタを削除すると、検索速度は速くなる。しかし、レコードの追加に伴ってインデックスに新たなキーを挿入する場合、その挿入処理はＢツリーに比べて遅くなってしまう。図１３（ａ）のように、Ｂツリーの場合は各キーに対応するポインタがあるので、ノード分割によって新たに生成した子ノードを自由に配置することができる。これに対して、ＣＳＢ＋ツリーの場合は、図１３（ｂ）に示すように、ノードグループ内

でキーの値が昇順または降順となるように子ノードを並び替える必要があるため、そのぶん処理速度が遅くなってしまうという問題があった。

[0022] 特許文献1：特開平５－３３４１５３号公報

特許文献2：特開２００３－１１４８１６号公報

[0023] 非特許文献1：D. Comer. The ubiquitous b-tree. ACM Computing Surveys, 11(2):121-137, 1979

非特許文献2：S. Sasaki. Modularizing B+-trees: Three-Level B+-trees Work Fine. ADMS@VLDB2013: 46-57

非特許文献3：J. Rao and K. A. Ross. Making B+-trees cache conscious in main memory. In SIGMOD, pages 475-486, 2000

発明の開示

[0024] 以上のように、インメモリ環境に最適化されたＣＳＢ＋ツリーを採用してポインタ削減により検索処理を高速化しようとすると、インデックスにおけるキーの挿入処理が低速化してしまうため、レコードの更新が多いワークロードに対して、全体としての処理性能が高くならないという問題があった。

[0025] 本発明は、このような問題を解決するために成されたものであり、インメモリ環境においてデータの検索処理を高速化できるようにするとともに、更新処理の低速化を抑制できるようにすることを目的とする。

[0026] 上記した課題を解決するために、本発明では、所定数のキーと所定数のバリューとの組を格納したリーフノードを第０階層として、第ｎ階層以下（ｎは「１≦ｎ＜全階層数－１」を満たす任意の値）の下位階層において、所定数のキーと子ノードの位置を表す所定数のポインタまたは所定数のバリューとの組を格納した第１の種類のノードによって、インデックスツリーの探索および更新を管理する。一方、第ｎ階層よりも上の上位階層においては、所定数のキーと下階層のノードグループの先頭位置を表す１つのグループポインタとを格納した第２の種類のノードによって、インデックスツリーの探索および更新を管理するようにしている。

[0027] 本発明の他の態様では、第１階層以上で第ｎ階層以下の下位階層において

、所定数のキーと下階層のノードグループの先頭位置を表す１つのグループポインタとを格納するとともに、ノードグループ内の各ノードの位置を表すのに十分なサイズの縮小ポインタを格納した第３の種類のノードによって、インデックスツリーの探索および更新を管理するようにしている。一方、第ｎ階層よりも上の上位階層においては、第２の種類のノードによってインデックスツリーの探索および更新を管理するようにしている。

[0028] 上記のように構成した本発明によれば、一般的にインデックスツリーでは上位階層のノードほどアクセス比率が高い代わりに更新比率が低く、下位階層のノードほどアクセス比率が低い代わりに更新比率が高いという性質を利用して、上位階層においては、検索処理を高速に行うことが可能な第２の種類のノードによってインデックスツリーの探索および更新が管理される。逆に、下位階層においては、キーの挿入処理を高速に行うことが可能な第１の種類のノードによってインデックスツリーの探索および更新が管理される。これにより、インメモリ環境において、データの検索処理を高速化するとともに、更新処理の低速化を抑制することができる。

[0029] 本発明の他の特徴によれば、上位階層においては、検索処理を高速に行うことが可能な第２の種類のノードによってインデックスツリーの探索および更新が管理される。一方、下位階層においては、ノードグループ内のオフセット量が縮小ポインタに基づいて求められるので、ノードグループ内で各ノードを自由に配置することができ、ノード分割等が必要となるデータの更新処理でも高速に行うことができる。また、縮小ポインタに必要な記憶容量は小さくて済むので、１つのノードに格納可能なキーの数を増やすことができ、インデックスツリーの階層数を減らして検索処理も高速化することができる。

図面の簡単な説明

[0030] [図1]第１の実施形態によるインデックス管理装置の機能構成例を示すブロック図である。

[図2]第１の実施形態におけるインデックスツリーの具体例を示す図である。

[図3]第１の実施形態においてキーが挿入された後のインデックスツリーの構成例を示す図である。

[図4]第２の実施形態によるインデックス管理装置の機能構成例を示すブロック図である。

[図5]第２の実施形態におけるインデックスツリーの具体例を示す図である。

[図6]２の実施形態において用いる縮小ポインタの特徴を示す図である。

[図7]第２の実施形態によるインデックス管理装置の他の機能構成例を示すブロック図である。

[図8]インデックスツリーの概念を説明するための図である。

[図9]ノード分割の際にツリーを下層側に成長させる例を説明するための図である。

[図10]ノード分割の際にツリーを上層側に成長させるＢツリーの例を説明するための図である。

[図11]３階層から成るＢツリーのファンアウトの一例を示す図である。

[図12]ＣＳＢ＋ツリーの構成例を示す図である。

[図13]ＢツリーおよびＣＳＢ＋ツリーのノード分割を説明するための図である。

発明を実施するための最良の形態

[0031] （第１の実施形態）

以下、本発明の第１の実施形態を図面に基づいて説明する。図１は、第１の実施形態によるインデックス管理装置の機能構成例を示すブロック図である。第１の実施形態によるインデックス管理装置は、最下位階層であるリーフノードと、最上位階層であるルートノードと、リーフノードとルートノードとの間にある１以上のブランチノードとからなるインデックスツリーを管理するものであって、その機能構成として、検索処理部１、挿入処理部２、下位階層管理部３および上位階層管理部４を備えて構成されている。

[0032] 上記各機能ブロック１～４は、ハードウェア、ＤＳＰ（Digital Signal Processor）、ソフトウェアの何れによっても構成することが可能である。例え

ばソフトウェアによって構成する場合、上記各機能ブロック１～４は、実際にはコンピュータのＣＰＵ、ＲＡＭ、ＲＯＭなどを備えて構成され、ＲＡＭやＲＯＭ、ハードディスクまたは半導体メモリ等の記録媒体に記憶されたプログラムが動作することによって実現される。

[0033] 本実施形態では、所定数のキーと所定数のバリューとの組を格納したリーフノードがある階層を第０階層として、第ｎ階層以下（ｎは「１≦ｎ＜全階層数－１」を満たす任意の値）を下位階層とし、第ｎ階層よりも上の階層を上位階層とする。以下では、ｎ＝１の場合について説明する。つまり、第０階層およびその上の第１階層を下位階層とする。また、第１階層よりも上の第２階層以上を上位階層とする。図１の例では、インデックスツリーは第０階層から第３階層までの４階層で構成されている。このうち、第０階層および第１階層が下位階層、第２階層および第３階層が上位階層である。また、１つのリーフノードは、最大３個のキーと、当該キーと同数のバリューとの組により構成されている。

[0034] 検索処理部１は、インデックスツリーを利用してインメモリ環境のデータベース（メモリ）から所望のデータ（バリュー）を検索するものである。具体的には、検索処理部１は、検索キーを上位階層管理部４に供給し、上位階層管理部４および下位階層管理部３の処理により、検索キーに対応するバリューを検索する。そして、その検索されたバリューを下位階層管理部３から受け取る。

[0035] 挿入処理部２は、インデックスツリーに所望のデータ（キーとバリューとの組）を挿入するものである。具体的には、挿入処理部２は、挿入するキーとバリューとを上位階層管理部４に供給し、上位階層管理部４および下位階層管理部３の処理により、挿入キーの値から挿入すべきリーフノードを決定し、決定したリーフノードの適切な位置にキーとバリューとの組を追加する。そして、下位階層管理部３または上位階層管理部４から挿入完了の通知を受け取る。

[0036] 下位階層管理部３は、インデックスツリーの下位階層において、所定数の

キーと、子ノードの位置を表す所定数のポインタまたは所定数のバリューとの組を格納した第１の種類のノードによって、インデックスツリーの探索および更新を管理する。この第１の種類のノードは、例えばＢツリーで使用されるノードと同じである。

[0037] また、上位階層管理部４は、インデックスツリーの上位階層において、所定数のキーと、下階層のノードグループの先頭位置を表す１つのグループポインタとを格納した第２の種類のノードによって、インデックスツリーの探索および更新を管理する。この第２の種類のノードは、例えばＣＳＢ＋ツリーで使用されるノードと同じである。

[0038] 図２は、下位階層管理部３および上位階層管理部４により探索および更新されるインデックスツリーの具体例を示す図である。図２に示すように、下位階層管理部３により管理される下位階層である第１階層は、最大２個のキーと、当該キーより１つ多いポインタとの組を格納した第１の種類のノードで構成されている。個々のポインタは、１つ下の第０階層にあるリーフノードの左端の位置を表している。もう１つの下位階層であるリーフノードは、最大３個のキーと、当該キーと同数のバリューとの組により構成されている。

[0039] また、上位階層管理部４により管理される上位階層である第２階層および第３階層は、最大２個のキーと、１つ下の階層にあるノードグループの先頭位置を表す１つのグループポインタとを格納した第２の種類のノードで構成されている。第３階層の１つの下の第２階層には１つのノードグループGr_{2-1}が設定され、第２階層の１つの下の第１階層には３つのノードグループGr_{1-1}，Gr_{1-2}，Gr_{1-3}が設定されている。

[0040] ここで、図２のように構成されたインデックスツリーを用いて、検索処理部１による検索処理を行う場合の動作を説明する。まず、検索処理部１は、検索キーを上位階層管理部４に渡す。なお、以下では、検索キーの値が“１５”であるものとして説明する。

[0041] 上位階層管理部４は、最上位の第３階層と、そこでルートノードになって

いるノードとを特定する。そして、上位階層管理部４は、当該ルートノードに格納されているキーの中から、検索キー以下で最大の値を持つキーを探索する。さらに、上位階層管理部４は、探索したキーと共に同じノード内に格納されているグループポインタが示すノードグループの先頭位置からのオフセット量をノードサイズに基づいて計算することによって特定される位置を辿って下位層に遷移する。

[0042] 図２の例の場合、ルートノードに格納されているキーのうち、検索キー“１５”以下で最大の値を持つキーは存在しない（省略されている）。このように、省略された左端のキーが探索された場合、オフセット量はゼロである。この場合、上位階層管理部４は、省略されたキーと共に同じノード内に格納されているグループポインタに従って、１つ下の第２階層にあるノードグループＧｒ$_{2-1}$の先頭位置のノードに遷移する。

[0043] 遷移した第２階層も上位階層であるから、上位階層管理部４によって上述と同様の処理を行う。すなわち、上位階層管理部４は、ルートノードから遷移してきたノードグループＧｒ$_{2-1}$の先頭位置のノードに格納されているキーのうち、検索キー“１５”以下で最大の値を持つキーを探索する。この場合に探索されるキーは“１０”である。このように、省略されたキーも含めてノード内の左端から２番目のキーが探索された場合、オフセット量はノードサイズ×１となる。この場合、上位階層管理部４は、探索したキー“１０”と共に同じノード内に格納されているグループポインタとオフセット量に従って、１つ下の第１階層にあるノードグループＧｒ$_{1-1}$の先頭位置から２番目のノードに遷移する。

[0044] このとき遷移した第１階層は下位階層であるから、下位階層管理部３によって処理を行う。下位階層管理部３は、特定されたノードに格納されているキーの中から、検索キー以下で最大の値を持つキーを探索する。さらに、下位階層管理部３は、探索したキーと共に同じノード内に格納されているポインタが示す位置を辿って下位層に遷移する。

[0045] 図２の例の場合、ノードグループＧｒ$_{1-1}$の先頭位置から２番目のノードに格

納されているキーのうち、検索キー“１５”以下で最大の値を持つキーは“１３”である。この場合、下位階層管理部３は、探索したキー“１３”との組として格納されているポインタに従って、１つ下の第０階層にある左端から５番目のリーフノードにダイレクトに遷移する。検索キー“１５”はこのノード内にあるので、下位階層管理部３は当該検索キーの位置に対応するバリューを取得し、検索処理部１に渡す。これにより、検索処理部１による検索処理が終了する。

[0046] 次に、図２のように構成されたインデックスツリーを用いて、挿入処理部２による挿入処理を行う場合の動作を説明する。まず、挿入処理部２は、挿入キーとバリューとの組を上位階層管理部４に渡す。なお、以下では、挿入キーの値が“９”であるものとして説明する。上位階層管理部４および下位階層管理部３は、上述した検索処理と同様の手順に従って、挿入キー“９”を挿入すべきリーフノードの探索を行う。これにより、左端から３番目のリーフノードに遷移する。

[0047] ここで、下位階層管理部３は、検索されたリーフノードに空きスペースがあるか否かを判定する。そして、空きスペースがある場合は、そのリーフノードに挿入キー“９”とバリューとの組を挿入する。図２の例では、左端から３番目のリーフノードに１つ空きスペースがあるので、そのノード内に挿入キー“９”とバリューとの組を挿入することが可能である。

[0048] 一方、検索されたリーフノードに空きスペースがない場合、下位階層管理部３は、そのリーフノードを分割して挿入キーとポインタとの組を挿入する。例えば、挿入処理部２から上位階層管理部４に渡された挿入キーの値が“１７”であったとする。この場合、上位階層管理部４および下位階層管理部３が挿入キー“１７”を挿入すべきリーフノードの探索を行うことにより、左端から６番目のリーフノードに遷移する。

[0049] しかし、この６番目のノードには既に３つのキーとバリューの組が格納されていて、空きスペースがない。そこで、下位階層管理部３は、この６番目のリーフノードを分割して空きスペースを確保し、挿入キー“１７”とバリ

ューとの組を挿入する。

[0050] 具体的には、下位階層管理部３は、まず、新たな空のノードを取得する。次に、下位階層管理部３は、６番目のリーフノードに含まれる３つのキーのうち、所定の分割キー以上のキーとそれに対応するバリューとの組を新たなノードに移す。ここで、分割キーの値は、例えば、３つのキーの値の中央値とする。その後、下位階層管理部３は、挿入キー“１７”が分割キー以上の場合は新たなノードに、そうでなければ元のノードに挿入キー“１７”とバリューとの組を挿入する。

[0051] このようにノード分割を行った場合、新たに生成したリーフノードを指すポインタを、上位階層のノードのエントリに追加しなければならない。すなわち、下位階層管理部３は、分割キーと新ノードへのポインタを、リーフノードのある第０階層よりも１つ上の第１階層の、探索パス上にあるノード（左端から２番目のノード）に追加する。このとき、そのノードに新たなキーとポインタの組を追加するスペースがなければ、ノードグループ内で新たなノードを確保する。

[0052] 図２の例では、左から２番目のノードに空きスペースがないので、ノードグループ内の３番目の位置のノードを初期化し、このノードに２番目のエントリを移す。このノード分割によってできたスペースに、挿入キー“１７”とバリューとの組を挿入する。この例では、２番目のノードが分割されたためノードのコピーは発生しなかったが、仮に１番目のノードが分割されたとすると、２番目のノードを３番目のノードとしてコピーし、２番目のノードを初期化にする（空にする）ことによって、１番目のノードを分割する。

[0053] 上に述べたとおり、第１階層のようにノードグループが設定されている場合、空きスペースの確保は以下のようにして行う。下位階層管理部３は、まず、第０階層においてノード分割をして挿入キー“１７”を挿入したことに伴い、新たに第１階層でキーを追加しようとするノードが属するノードグループGr_{1-1}にノードを追加するスペースがあるか否かを判定する。スペースがあれば、当該ノードグループGr_{1-1}内の分割されるノードの直後の位置以降の

ノードを１つ右の位置にコピーする。そして、分割するノードの直後のノードを初期化して、新たな空のノードを確保する。これによってノード分割が可能になる。

[0054] 一方、ノードグループＧｒ$_{1-1}$に空のノードがない場合、下位階層管理部３は新たなノードグループを取得する。そして、下位階層管理部３は、一部（例えば、ノードグループＧｒ$_{1-1}$内の後ろ半分）のノードを新たなノードグループに移動させる。これにより、ノードグループＧｒ$_{1-1}$内にノードを追加するスペースができるので、ノード分割によってエントリを追加するスペースを作成できる。

[0055] 第１階層でノードの移動や新たなノードグループの取得などを行った場合、下位階層管理部３は上位階層管理部４に依頼して、第１階層よりも１つ上の第２階層の、探索パス上にあるノードグループ内のノード（左端のノード）に、必要なキーを追加する。このノードに新たなキーを追加するスペースがなければ、第２階層でも第１の階層と同様にノードの移動またはノードグループの取得などを行って挿入スペースを確保する。この場合、ルートノードに新たなキーを追加する必要があるが、空きスペースがなくてルートノードが分割される場合には、未使用かつ最下位の階層のノードを新たなルートノードにする。

[0056] 図３は、図２に示したインデックスツリーに対して挿入キー“１７”とバリューとの組を挿入した後のインデックスツリーの構成例を示す図である。図３に示す例では、挿入キー“１７”とバリューとの組を挿入するために第０階層においてノード分割が行われ、それに伴って第１階層および第２階層においてもエントリの追加が行われている。

[0057] 以上詳しく説明したように、第１の実施形態では、第０階層および第１階層から成る下位階層では、所定数のキーと所定数のポインタまたは所定数のバリューとの組を格納した第１の種類のノード（Ｂツリーのノード）によってインデックスツリーの探索および更新を管理する。一方、第１階層よりも上の第２階層以上の上位階層では、所定数のキーと１つのグループポインタ

とを格納した第２の種類のノード（ＣＳＢ＋ツリーのノード）によってインデックスツリーの探索および更新を管理するようにしている。

[0058] インデックスツリーは、上位階層のノードほどアクセス比率が高い代わりに更新比率が低く、下位階層のノードほどアクセス比率が低い代わりに更新比率が高いという性質を持つ。第１の実施形態ではこの性質を利用して、上位階層においては、検索処理を高速に行うことが可能な第２の種類のノードによってインデックスツリーの探索および更新が管理される。逆に、下位階層においては、キーの挿入処理を高速に行うことが可能な第１の種類のノードによってインデックスツリーの探索および更新が管理される。これにより、インメモリ環境において、データの検索処理を高速化するとともに、更新処理の速度低下を抑制することができる。

[0059]（第２の実施形態）

次に、本発明の第２の実施形態を図面に基づいて説明する。図４は、第２の実施形態によるインデックス管理装置の機能構成例を示すブロック図である。図４に示すように、第２の実施形態によるインデックス管理装置は、その機能構成として、検索処理部１、挿入処理部２、下位階層管理部３、上位階層管理部４および中位階層管理部５を備えて構成されている。なお、この図４において、図１に示した符号と同一の符号を付したものは同一の機能を有するものであるので、ここでは重複する説明を省略する。

[0060] 本実施形態では、リーフノードがある階層を第０階層として、当該第０階層およびそれより１つ上の第１階層を下位階層とする。また、第１階層より１つ上の第２階層を中位階層とし、第３階層以上を上位階層とする。図４の例では、インデックスツリーは第０階層から第３階層までの４階層で構成されている。このうち、第０階層および第１階層が下位階層、第２階層が中位階層、第３階層が上位階層である。

[0061] 中位階層管理部５は、インデックスツリーの中位階層において、所定数のキーと１つ下の階層のノードグループの先頭位置を表す１つのグループポインタとを格納するとともに、当該ノードグループ内の各ノードの位置を表す

ポインタであって第１の種類のノードに格納されるポインタよりもサイズの小さい縮小ポインタを格納した第３の種類のノードによって、インデックスツリーの探索および更新を管理する。

[0062] 図５は、下位階層管理部３、上位階層管理部４および中位階層管理部５により探索および更新されるインデックスツリーの具体例を示す図である。この図５は、図２に示したインデックスツリーと略同じであるが、中位階層として定義した第２階層のノードが図２のインデックスツリーと異なっている。

[0063] 図５に示すように、中位階層である第２階層は、最大２個のキーと、１つ下の階層にあるノードグループの先頭位置を表す１つのグループポインタと、キーと同数の縮小ポインタとを格納した第３の種類のノードで構成されている。例えば、第１階層における通常のポインタのサイズが４バイトであるの対し、縮小ポインタのサイズは２バイトである。これは例えば、ノードが４Ｇバイトのアドレス空間のどこかにある場合にはポインタの大きさは４バイト以上でなければならないが、ノードが６４Ｋバイトのノードグループのどこかにある場合には２バイト以上のポインタでノードを特定できるからである。このように、ノードグループの先頭へのポインタを使えば、ポインタを縮小しても子ノードを特定できる。

[0064] ここで、図５のように構成されたインデックスツリーを用いて、検索処理部１による検索処理を行う場合の動作を説明する。まず、検索処理部１は、検索キーを上位階層管理部４に渡す。なお、以下では、検索キーの値が“１５”であるものとして説明する。

[0065] 上位階層である第３階層における上位階層管理部４の処理は、第１の実施形態と同様である。第３階層から遷移した第２階層は中位階層であるから、中位階層管理部５によって処理を行う。すなわち、中位階層管理部５は、ルートノードから遷移してきたノードグループGr_{2-1}の先頭位置のノードに格納されているキーのうち、検索キー“１５”以下で最大の値を持つキーを探索する。この場合に探索されるキーは“１０”である。

[0066] さらに、中位階層管理部５は、以上のように探索したキー“１０”と共に同じノード内に格納されているグループポインタが示すノードグループの先頭位置からのオフセット量を、当該キー“１０” との組として格納されている縮小ポインタに基づいて計算する。そして、このグループポインタとオフセット量とによって特定される位置を辿ることにより、１つ下の第１階層にあるノードグループＧｒ$_{1-1}$の先頭位置から２番目のノードに遷移する。

[0067] このとき遷移した第１階層は下位階層であるから、下位階層管理部３によって第１の実施形態と同様の処理を行う。これにより、キー“１３”との組として格納されているポインタに従って、１つ下の第０階層にある左端から５番目のリーフノードにダイレクトに遷移し、そのリーフノード内から検索キー“１５”の位置に対応するバリューを取得し、検索処理部１に渡す。これにより、検索処理部１による検索処理が終了する。

[0068] なお、挿入処理部２による挿入処理では、まず、挿入キーを挿入すべきリーフノードの検索を上述した検索処理と同様の手順に従って行う。その後、検索したリーフノードに挿入キーとバリューとの組を挿入する処理は、上述した第１の実施形態と同様なので、ここでは説明を省略する。

[0069] 以上詳しく説明したように、第２の実施形態では、ノードグループ内のオフセット量が縮小ポインタに基づいて求められるので、ノードグループ内で各ノードのエントリはメモリの連続領域に格納されていることは必須でなく、図６のようにノードグループ内で各ノードを自由に配置することができる。そのため、ノード分割等が必要となるデータの更新処理でも高速に行うことができる。

[0070] そして、このようにデータの更新処理を高速にするために各キーに対応させて縮小ポインタを設けつつ、当該縮小ポインタに必要な記憶容量を削減することにより、１つのノードに格納可能なキーの数を増やすことができる。これにより、インデックスツリーの階層数（ライン数）を減らし、検索処理も高速化することができる。

[0071] なお、上記第２の実施形態では、インデックスツリーを上位階層、中位階

層、下位階層に分けて管理し、中位階層において縮小ポインタを用いる例について説明したが、本発明はこれに限定されない。例えば、インデックスツリーを上位階層と下位階層に分けて管理し、下位階層において縮小ポインタを用いるようにしてもよい。

[0072] すなわち、図７に示すように、検索処理部１、挿入処理部２、下位階層管理部３’および上位階層管理部４を備えてインデックス管理装置を構成する。下位階層管理部３’は、リーフノードのある階層を第０階層として、第１階層以上で第ｎ階層以下（ｎは「１≦ｎ＜全階層数－１」を満たす任意の値）の下位階層において、所定数のキーと下階層のノードグループの先頭位置を表す１つのグループポインタとを格納するとともに、ノードグループ内の各ノードの位置を表す縮小ポインタを格納した第３の種類のノードによってインデックスツリーの探索および更新を管理する。なお、第０階層については第１の種類のノードによってインデックスツリーの探索および更新を管理する。

[0073] また、上記実施形態では、内部ノードについては左端のキーが省略されている例について説明したが、左端のキーが省略されていない内部ノードにより構成されるインデックスツリーに対して本発明を適用することも可能である。

[0074] 上記第１および第２の実施形態によるインデックス管理装置は、リレーショナルデータベースのインデックス、多くのプログラムに組み込まれるマップ処理、ファイルシステム、キーバリューストア、ＯＬＡＰ（online analytical processing）システムなど、更新されるデータに対して検索をかけることのあるシステムに対しては広く利用することが可能である。

[0075] その他、上記第１および第２の実施形態は、何れも本発明を実施するにあたっての具体化の一例を示したものに過ぎず、これらによって本発明の技術的範囲が限定的に解釈されてはならないものである。すなわち、本発明はその要旨、またはその主要な特徴から逸脱することなく、様々な形で実施することができる。

符号の説明

[0076] １　検索処理部

２　挿入処理部

３，３’　下位階層管理部

４　上位階層管理部

５　中位階層管理部

請求の範囲

[請求項1] 最下位階層であるリーフノードおよびその他の内部ノードからなるインデックスツリーを管理するインデックス管理装置であって、

所定数のキーと所定数のバリューとの組を格納した上記リーフノードのある階層を第０階層として、第ｎ階層以下（ｎは「１≦ｎ＜全階層数－１」を満たす任意の値）の下位階層において、所定数のキーと子ノードの位置を表す所定数のポインタまたは所定数のバリューとの組を格納した第１の種類のノードによって、上記インデックスツリーの探索および更新を管理する下位階層管理部と、

上記第ｎ階層よりも上の上位階層において、所定数のキーと下階層のノードグループの先頭位置を表す１つのグループポインタとを格納した第２の種類のノードによって、上記インデックスツリーの探索および更新を管理する上位階層管理部とを備えたことを特徴とするインデックス管理装置。

[請求項2] 上記下位階層管理部は、上記第１の種類のノードに格納されているキーの中から検索キー以下で最大の値を持つキーを探索し、探索したキーと共に同じノード内に格納されているポインタが示す位置を辿るようになされ、

上記上位階層管理部は、上記第２の種類のノードに格納されているキーの中から上記検索キー以下で最大の値を持つキーを探索し、探索したキーと共に同じノード内に格納されているグループポインタが示すノードグループの先頭位置からのオフセット量をノードサイズに基づいて計算することによって特定される位置を辿るようになされていることを特徴とする請求項１に記載のインデックス管理装置。

[請求項3] 上記下位階層管理部は、上記第１の種類のノードに格納されているキーの中から挿入キー以下で最大の値を持つキーを探索し、探索したキーと共に同じノード内に格納されているポインタが示す位置を辿ることによって上記挿入キーを挿入すべきリーフノードの探索を行い、

探索されたリーフノードに空きスペースがある場合はそのリーフノードに上記挿入キーとバリューとの組を挿入する一方、探索されたリーフノードに空きスペースがない場合はそのリーフノードを分割して上記挿入キーとバリューとの組を挿入するようになされ、

上記上位階層管理部は、上記第２の種類のノードに格納されているキーの中から上記挿入キー以下で最大の値を持つキーを探索し、探索したキーと共に同じノード内に格納されているグループポインタが示すノードグループの先頭位置からのオフセット量をノードサイズに基づいて計算することによって特定される位置を辿るようになされていることを特徴とする請求項１に記載のインデックス管理装置。

[請求項4] 上記第１階層よりも１つ上の上記第２階層を中位階層、当該第２階層よりも上の第３階層以上を上位階層とし、

上記中位階層において、所定数のキーと下階層のノードグループの先頭位置を表す１つのグループポインタとを格納するとともに、上記ノードグループ内の各ノードの位置を表すポインタであって上記第１の種類のノードに格納されるポインタよりもサイズの小さい縮小ポインタを格納した第３の種類のノードによって、上記インデックスツリーの探索および更新を管理する中位階層管理部を更に備えたことを特徴とする請求項１に記載のインデックス管理装置。

[請求項5] 上記中位階層管理部は、上記第３の種類のノードに格納されているキーの中から上記検索キー以下で最大の値を持つキーを探索し、探索したキーと共に同じノード内に格納されているグループポインタが示すノードグループの先頭位置からのオフセット量を上記縮小ポインタに基づいて計算することによって特定される位置を辿るようになされていることを特徴とする請求項４に記載のインデックス管理装置。

[請求項6] 最下位階層であるリーフノードおよびその他の内部ノードからなるインデックスツリーを管理するインデックス管理装置であって、

所定数のキーと所定数のバリューとの組を格納した上記リーフノー

ドのある階層を第０階層として、第１階層以上で第ｎ階層以下（ｎは「１≦ｎ＜全階層数－１」を満たす任意の値）の下位階層において、所定数のキーと下階層のノードグループの先頭位置を表す１つのグループポインタとを格納するとともに、上記ノードグループ内の各ノードの位置を表すのに十分なサイズの縮小ポインタを格納した第３の種類のノードによって、上記インデックスツリーの探索および更新を管理する下位階層管理部と、

上記第ｎ階層よりも上の上位階層において、所定数のキーと下階層のノードグループの先頭位置を表す１つのグループポインタとを格納した第２の種類のノードによって、上記インデックスツリーの探索および更新を管理する上位階層管理部とを備えたことを特徴とするインデックス管理装置。

[請求項7] 上記下位階層管理部は、上記第３の種類のノードに格納されているキーの中から上記検索キー以下で最大の値を持つキーを探索し、探索したキーと共に同じノード内に格納されているグループポインタが示すノードグループの先頭位置からのオフセット量を上記縮小ポインタに基づいて計算することによって特定される位置を辿るようになされ、

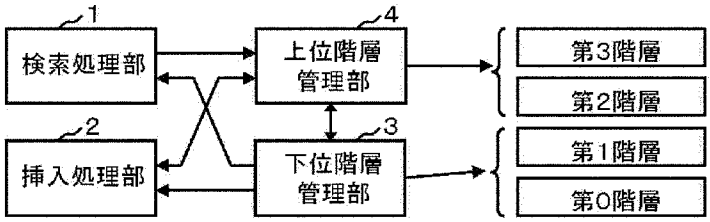
上記上位階層管理部は、上記第２の種類のノードに格納されているキーの中から上記検索キー以下で最大の値を持つキーを探索し、探索したキーと共に同じノード内に格納されているグループポインタが示すノードグループの先頭位置からのオフセット量をノードサイズに基づいて計算することによって特定される位置を辿るようになされていることを特徴とする請求項６に記載のインデックス管理装置。

[請求項8] 上記下位階層管理部は、上記第３の種類のノードに格納されているキーの中から上記挿入キー以下で最大の値を持つキーを探索し、探索したキーと共に同じノード内に格納されているグループポインタが示すノードグループの先頭位置からのオフセット量を上記縮小ポインタ

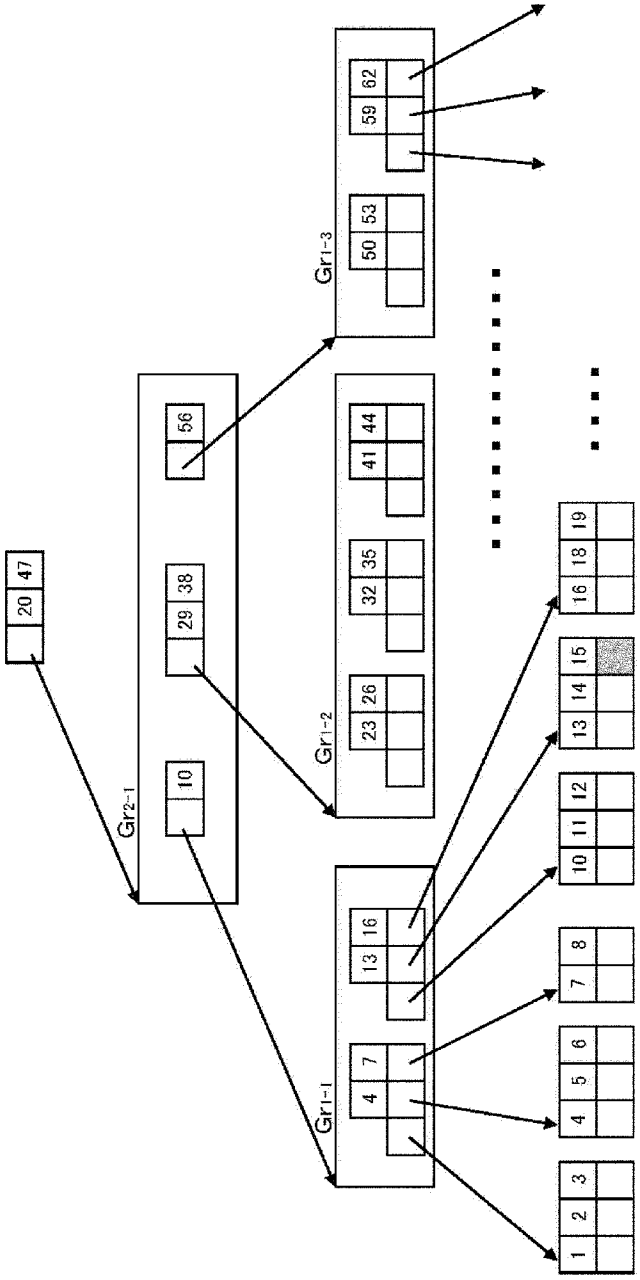
に基づいて計算することによって特定される位置を辿ることによって上記挿入キーを挿入すべきリーフノードの探索を行い、探索されたリーフノードに空きスペースがある場合はそのリーフノードに上記挿入キーとバリューとの組を挿入する一方、探索されたリーフノードに空きスペースがない場合はそのリーフノードを分割して上記挿入キーとポインタとの組を挿入するようになされ、

上記上位階層管理部は、上記第２の種類のノードに格納されているキーの中から上記挿入キー以下で最大の値を持つキーを探索し、探索したキーと共に同じノード内に格納されているグループポインタが示すノードグループの先頭位置からのオフセット量をノードサイズに基づいて計算することによって特定される位置を辿るようになされていることを特徴とする請求項６に記載のインデックス管理装置。

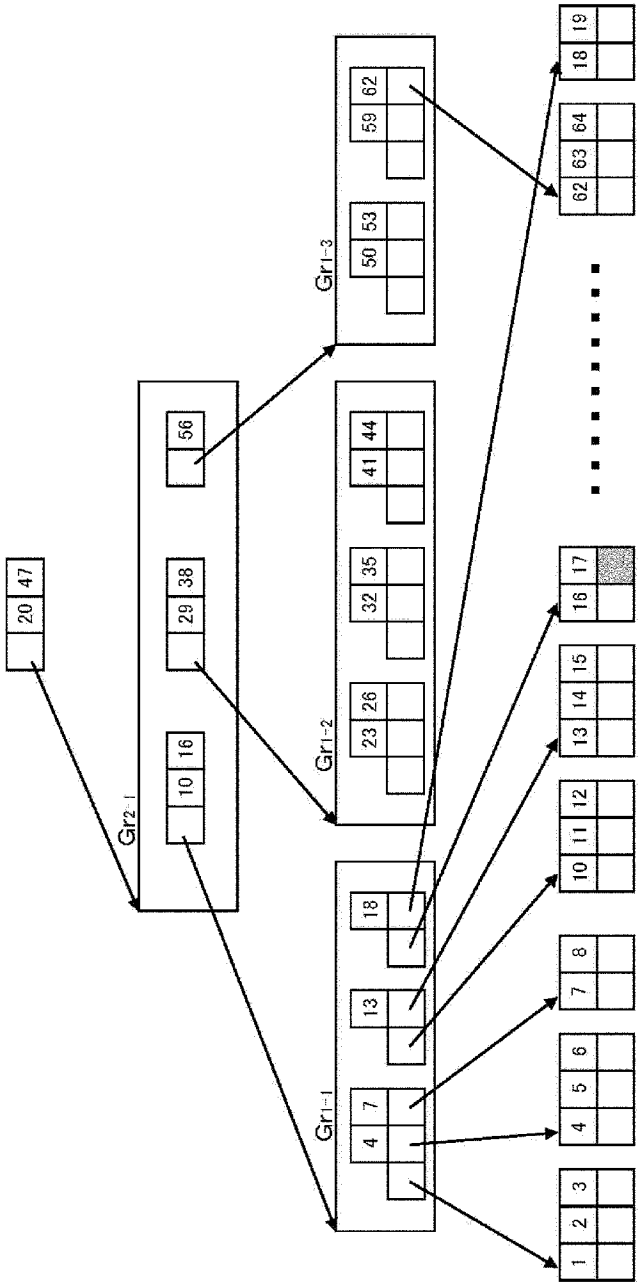
[図1]



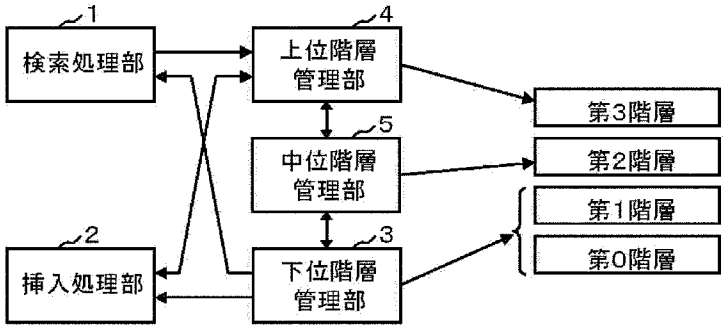
[図2]


Gr2-1
Gr1-3
Gr1-2
Gr1-1
20 47
56
29 38
10
50 53
59 62
41 44
32 35
23 26
13 16
4 7
16 18 19
13 14 15
10 11 12
7 8
4 5 6
1 2 3

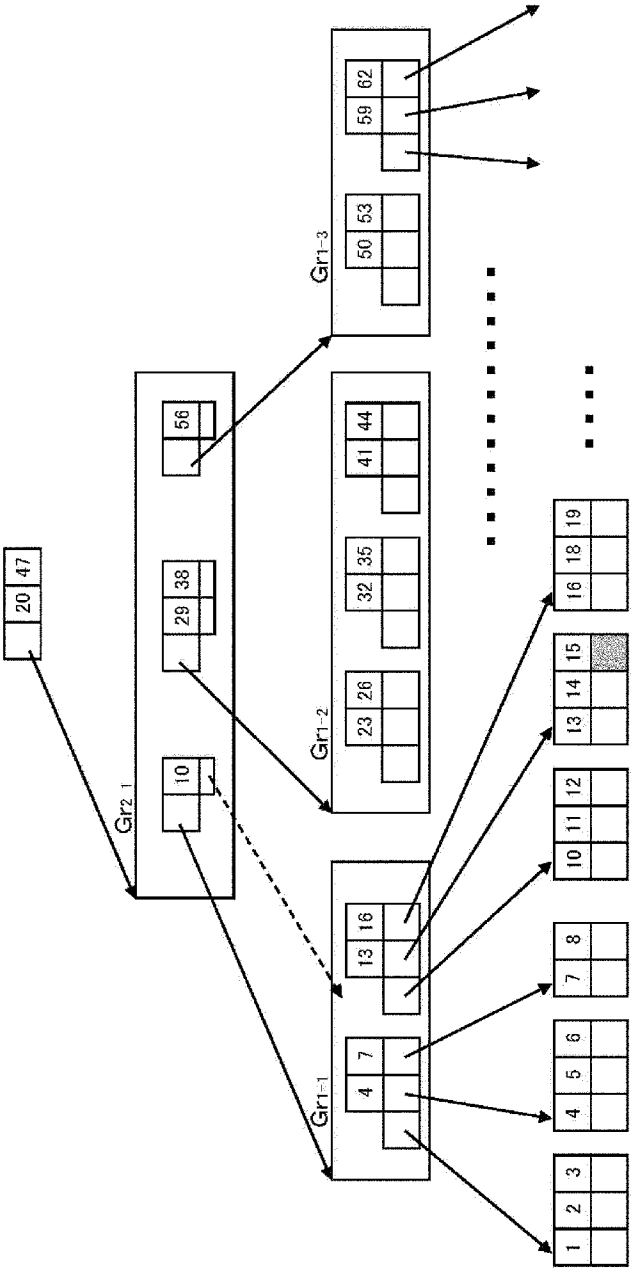
[図3]


Gr2-1
Gr1-3
Gr1-2
Gr1-1
20 47
56
29 38
10 16
50 53
59 62
41 44
32 35
23 26
18
13
4 7
1 2 3
4 5 6
7 8
10 11 12
13 14 15
16 17
62 63 64
18 19

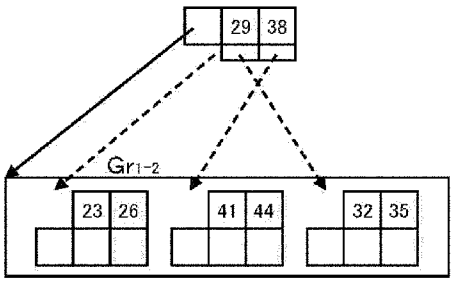
[図4]


1
検索処理部
4
上位階層
管理部
第3階層
5
中位階層
管理部
第2階層
第1階層
第0階層
2
挿入処理部
3
下位階層
管理部

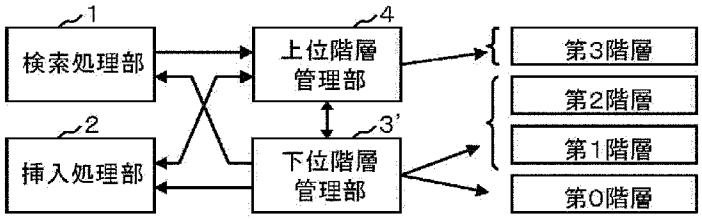
[図5]


Gr2_1
Gr1-3
Gr1-2
Gr1-1
20 47
10 29 38 56
4 7 13 16
23 26 32 35 41 44
50 53 59 62
1 2 3
4 5 6
7 8
10 11 12
13 14 15
16 18 19

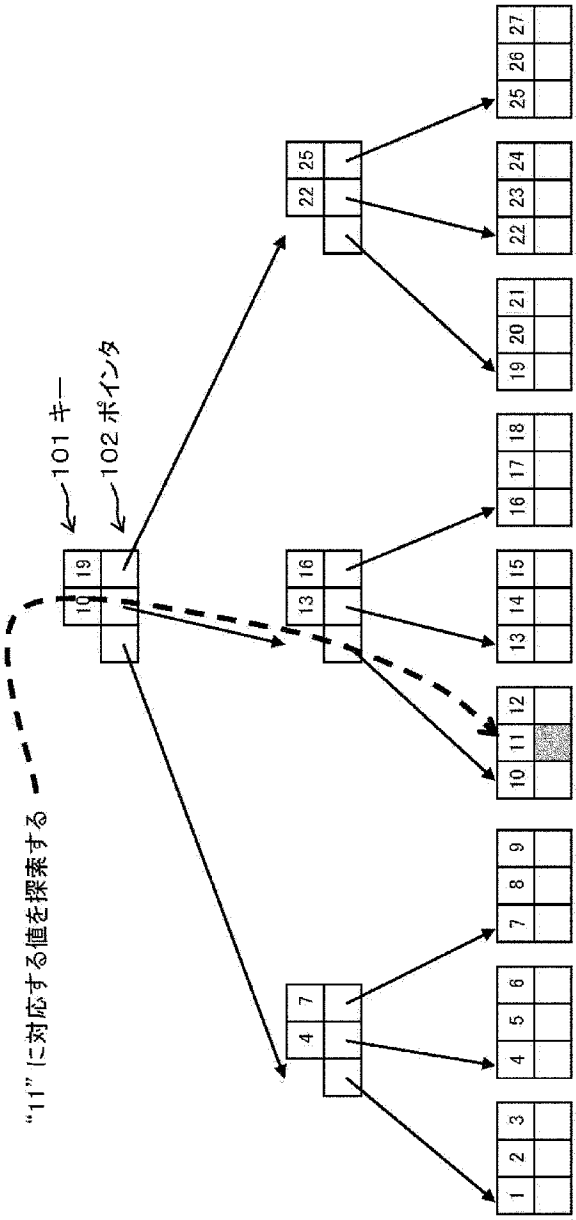
[図6]


29
38
Gr1-2
23
26
41
44
32
35

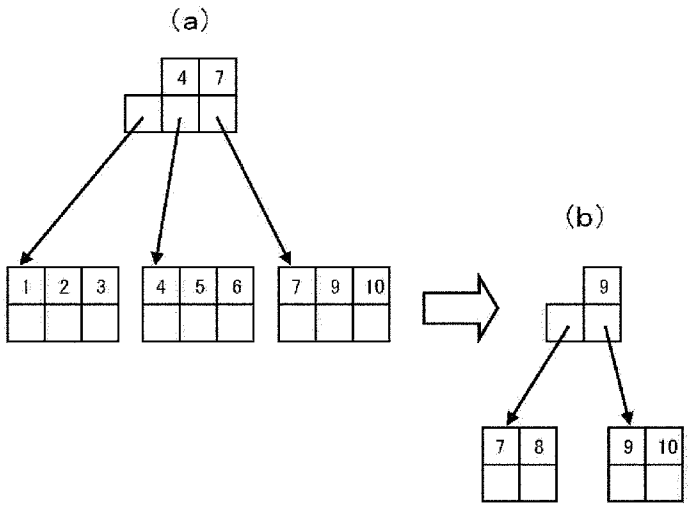
[図7]


1
検索処理部
2
挿入処理部
4
上位階層
管理部
3'
下位階層
管理部
第3階層
第2階層
第1階層
第0階層

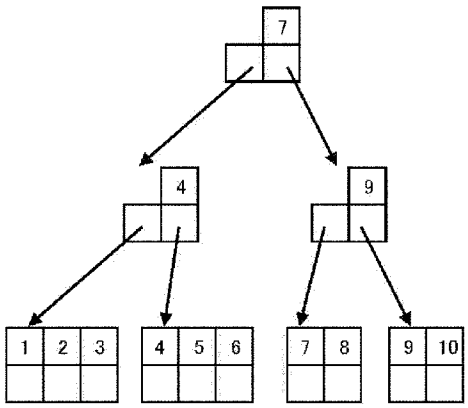
[図8]


101 キー
102 ポインタ
"11" に対応する値を探索する
1 2 3
4 5 6
7 8 9
10 11 12
13 14 15
16 17 18
19 20 21
22 23 24
25 26 27
4 7
13 16
19
22 25

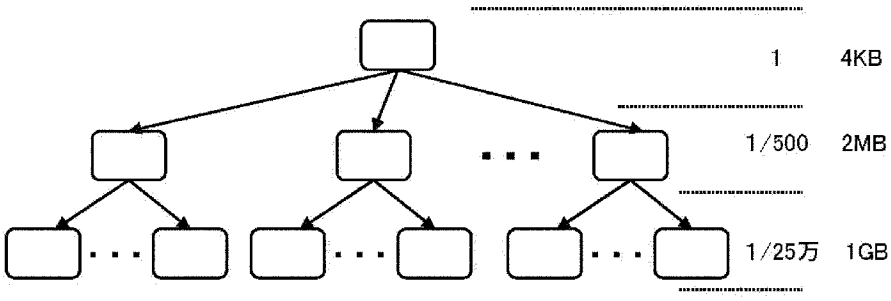
[図9]



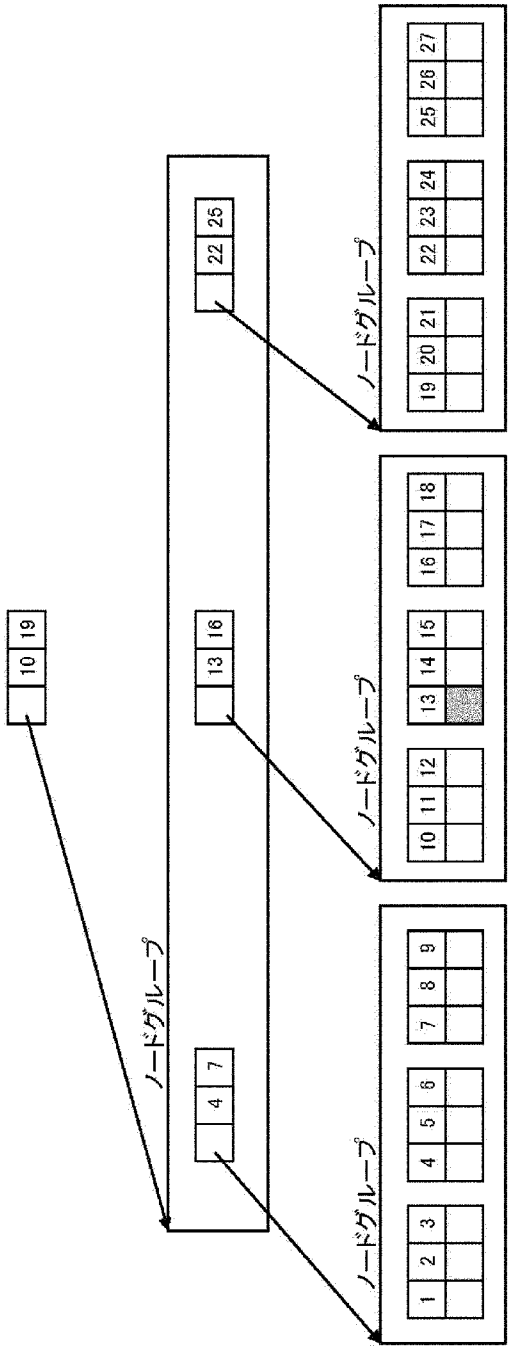
[図10]



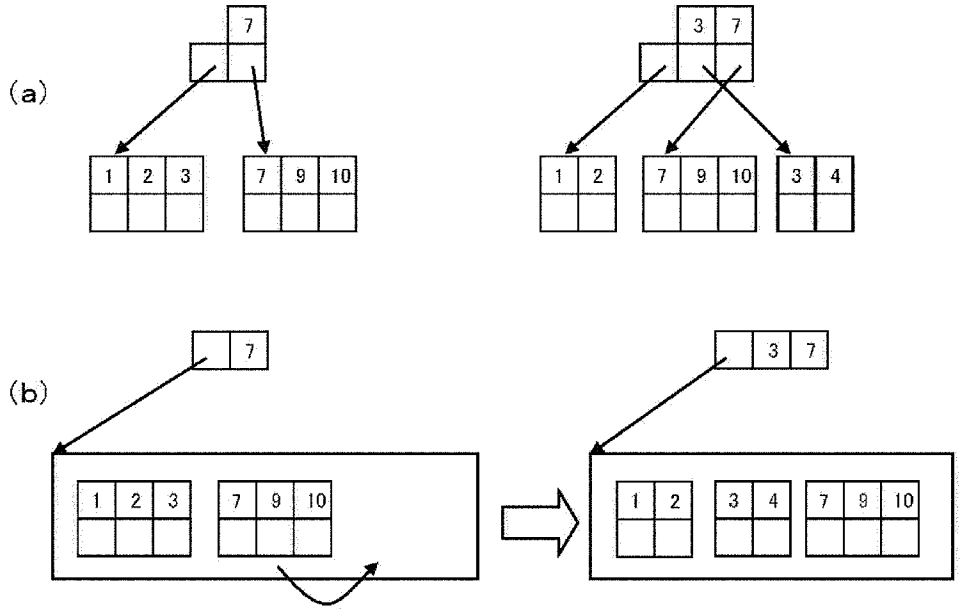
[図11]



[図12]


10 19
13 16
4 7
22 25
1 2 3
4 5 6
7 8 9
10 11 12
13 14 15
16 17 18
19 20 21
22 23 24
25 26 27
ノードグループ
ノードグループ
ノードグループ
ノードグループ

[図13]



INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2014/080851

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	1996-2015
Kokai Jitsuyo Shinan Koho	1971-2015	Toroku Jitsuyo Shinan Koho	1994-2015

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.
Y A	WO 2013/035287 A1 (NEC Corp.), 14 March 2013 (14.03.2013), paragraphs [0002] to [0092]; fig. 1 to 11 (Family: none)	1-3 4-8
Y A	JP 7-191891 A (Microsoft Corp.), 28 July 1995 (28.07.1995), paragraph [0002]; fig. 1 & US 5752243 A & EP 650131 A1 & CA 2117846 A1	1-3 4-8
Y A	Jun Rao and Kenneth A. Ross, Making B+-Trees Cache Conscious in Main Memory, SIGMOD '00 Proceedings of the 2000 ACM SIGMOD international conference on Management of data, ACM, 2000.05.15, p.475-486	2,3 4-8

☒ 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	Date of mailing of the international search report
14 January 2015 (14.01.15)	27 January 2015 (27.01.15)
Name and mailing address of the ISA/ Japan Patent Office Facsimile No.	Authorized officer Telephone No.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2014/080851

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP 2001-202277 A (Hewlett-Packard Co.), 27 July 2001 (27.07.2001), paragraphs [0010] to [0104]; fig. 1 to 13 & EP 1107126 A2	1-8
A	JP 2000-324172 A (NEC Corp.), 24 November 2000 (24.11.2000), paragraphs [0037] to [0096] & EP 1063827 A2 & DE 60032674 T2	1-8

国際調査報告

国際出願番号　ＰＣＴ／ＪＰ２０１４／０８０８５１

Ａ．　発明の属する分野の分類（国際特許分類（ＩＰＣ））

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（ＩＰＣ））

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	１９２２－１９９６年
日本国公開実用新案公報	１９７１－２０１５年
日本国実用新案登録公報	１９９６－２０１５年
日本国登録実用新案公報	１９９４－２０１５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	関連する請求項の番号
Y A	WO 2013/035287 A1（日本電気株式会社）2013.03.14, 段落［０００２］－［００９２］及び図１－１１（ファミリーなし）	１－３ ４－８
Y A	JP 7-191891 A（マイクロソフト　コーポレイション）1995.07.28, 段落【０００２】及び図１ & US 5752243 A & EP 650131 A1 & CA 2117846 A1	１－３ ４－８

☑　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	国際調査報告の発送日
１４．０１．２０１５	２７．０１．２０１５

国際調査機関の名称及びあて先
日本国特許庁（ＩＳＡ／ＪＰ）
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）　５Ｍ　５２８４
野崎　大進
電話番号　０３－３５８１－１１０１　内線　３５９９

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	関連する請求項の番号
Ｙ Ａ	Ｊｕｎ　Ｒａｏ　ａｎｄ　Ｋｅｎｎｅｔｈ　Ａ．　Ｒｏｓｓ，Ｍａｋｉｎｇ　Ｂ＋-Ｔｒｅｅｓ　Ｃａｃｈｅ　Ｃｏｎｓｃｉｏｕｓ　ｉｎ　Ｍａｉｎ　Ｍｅｍｏｒｙ，SIGMOD ’00 Proceedings of the 2000 ACM SIGMOD international conference on Management of data, ＡＣＭ，2000.05.15，ｐ．４７５－４８６	２，３ ４－８
Ａ	JP 2001-202277 A（ヒューレット・パッカード・カンパニー）2001.07.27，段落【００１０】－【０１０４】及び図１－１３ & EP 1107126 A2	１－８
Ａ	JP 2000-324172 A（日本電気株式会社）2000.11.24， 段落【００３７】－【００９６】 & EP 1063827 A2 & DE 60032674 T2	１－８