(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第5460486号
(P5460486)

(45)発行日　平成26年4月2日(2014.4.2)　(24)登録日　平成26年1月24日(2014.1.24)

(51)Int.Cl.　　F I
G06F 12/00 (2006.01)　　G06F 12/00 513D
G06F 17/30 (2006.01)　　G06F 17/30 417
　　G06F 17/30 320C

請求項の数 10　(全 20 頁)

(21)出願番号　特願2010-142973 (P2010-142973)
(22)出願日　平成22年6月23日 (2010.6.23)
(65)公開番号　特開2012-8725 (P2012-8725A)
(43)公開日　平成24年1月12日 (2012.1.12)
審査請求日　平成25年3月1日 (2013.3.1)

(73)特許権者　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
INTERNATIONAL BUSINESS MACHINES CORPORATION
アメリカ合衆国10504　ニューヨーク州　アーモンク　ニュー　オーチャード　ロード
(74)代理人　100108501
弁理士　上野　剛史
(74)代理人　100112690
弁理士　太佐　種一
(74)代理人　100091568
弁理士　市位　嘉宏

最終頁に続く

(54)【発明の名称】データをソートする装置及び方法

(57)【特許請求の範囲】
【請求項１】
複数のキーを有するデータであって、当該複数のキーを用いて複数のデータベース表にアクセスする処理に対して入力されるデータである入力データをソートする装置であって、
前記複数のキーのそれぞれについて、キーが取り得るキー値の各々が属する区分を、各区分に属するキー値に対応する前記データベース表のデータがバッファに読み込み可能なサイズとなるように決定する決定部と、
前記複数のキーがそれぞれ同一の区分に属するキー値を取る前記入力データのブロックを、隣接するブロック間で当該複数のキーの１つのキー以外のキーが同一の区分に属するキー値を取るようにソートし、当該ブロック内のデータを、当該１つのキーでソートするソート部と
を含む、装置。
【請求項２】
前記決定部は、前記区分に区分値を割り当て、
前記ソート部は、前記複数のキーが取る複数のキー値が属する複数の区分にそれぞれ割り当てられた複数の区分値に基づいて、隣接するブロック間で前記１つのキー以外のキーが同一の区分に属するキー値を取ることとなるような前記ブロックの順序を示す順序情報を生成し、当該順序情報を用いて前記ブロックをソートする、請求項１の装置。
【請求項３】

前記区分値は、０から（ｓ－１）までの整数であり、
前記ソート部は、前記複数の区分値を並べることにより前記ブロックの識別情報を生成し、以下の式を用いて、前記順序情報を生成するために並べられる複数の順序値を算出することにより、当該識別情報を当該順序情報に変換する、請求項２の装置（但し、ｓは自然数であり、区分値ｉは識別情報のｉ番目の区分値であり、順序値ｉは順序情報のｉ番目の順序値であり、順序値はｓを法とする剰余類で表す）。
【数１】

$$(\text{順序値}1 \quad \text{順序値}2 \quad \ldots \quad \text{順序値}\ n) = (\text{区分値}1 \quad \text{区分値}2 \quad \ldots \quad \text{区分値}\ n)\begin{pmatrix} 1 & 1 & \ldots & 1 \\ 0 & 1 & \ldots & 1 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 \end{pmatrix}$$

【請求項４】
前記ソート部は、以下の式を用いて、前記順序情報における前記１つのキーに対応する順序値の位置ｐを決定する、請求項３の装置（但し、順序値ｉは順序情報のｉ番目の順序値を表す）。
【数２】

$$p = \{k \mid (\text{順序値}\ k > 0) \wedge (\forall m > k, \text{順序値}\ m = 0)\}$$

【請求項５】
前記入力データは、複数のレコードを含み、
前記ソート部は、前記複数のレコードの各レコードに対し、当該各レコードにおいて前記複数のキーが取る複数のキー値に基づいて得られた前記順序情報と、当該各レコードにおいて前記１つのキーが取るキー値とを含むソートキーを付加し、当該複数のレコードを、当該各レコードに付加された当該ソートキーでソートする、請求項２乃至４の何れかの装置。
【請求項６】
前記入力データが有するＭ個のキーから、前記データベース表のデータをシーケンシャルに読み込み可能と判断するための条件を満たすＮ個のキーを、前記複数のキーとして選択する選択部を更に含む、請求項１乃至５の何れかの装置（Ｍ，ＮはＭ＞Ｎを満たす整数）。
【請求項７】
複数のレコードを含むデータであって、当該複数のレコードの各レコードについて、当該各レコードが有するＭ個のキーを用いて複数のデータベース表にアクセスする処理を、当該複数のレコードについて一括して行う一括処理の対象となるデータである入力データをソートする装置であって、
前記Ｍ個のキーのそれぞれについて、キーが取り得るキー値の各々が属する区分を、各区分に属するキー値に対応する前記データベース表のデータがバッファに読み込み可能なサイズとなるように決定する決定部と、
前記Ｍ個のキーから、前記データベース表のデータをシーケンシャルに読み込み可能と判断するための条件を満たすＮ個のキーを選択する選択部と、
前記複数のレコードの各レコードに対し、当該各レコードにおいて前記Ｎ個のキーが取るＮ個のキー値が属するＮ個の区分にそれぞれ割り当てられたＮ個の区分値に基づいて生成された情報であって、隣接するブロック間で当該Ｎ個のキーの１つのキー以外のキーが同一の区分に属するキー値を取ることとなるような前記入力データのブロックの順序を示す情報である順序情報と、当該各レコードにおいて当該１つのキーが取るキー値とを含むソートキーを付加する付加部と、
前記複数のレコードを、当該複数のレコードの各々に付加された前記ソートキーでソートするソート部と

を含む、装置（Ｍ，ＮはＭ＞Ｎを満たす整数）。
【請求項８】
複数のキーを有する入力データの入力に応じて、当該複数のキーを用いて複数のデータベース表にアクセスする処理を行うシステムであって、
前記複数のキーのそれぞれについて、キーが取り得るキー値の各々が属する区分を、各区分に属するキー値に対応する前記データベース表のデータがバッファに読み込み可能なサイズとなるように決定する決定部と、
前記複数のキーがそれぞれ同一の区分に属するキー値を取る前記入力データのブロックを、隣接するブロック間で当該複数のキーの１つのキー以外のキーが同一の区分に属するキー値を取るようにソートし、当該ブロック内のデータを、当該１つのキーでソートすることにより、ソート済み入力データを生成する生成部と、
前記生成部により生成された前記ソート済み入力データの入力に応じて、前記処理を行う処理部と
を含む、システム。
【請求項９】
複数のキーを有するデータであって、当該複数のキーを用いて複数のデータベース表にアクセスする処理に対して入力されるデータである入力データをソートする方法であって、
前記複数のキーのそれぞれについて、キーが取り得るキー値の各々が属する区分を、各区分に属するキー値に対応する前記データベース表のデータがバッファに読み込み可能なサイズとなるように決定するステップと、
前記複数のキーがそれぞれ同一の区分に属するキー値を取る前記入力データのブロックを、隣接するブロック間で当該複数のキーの１つのキー以外のキーが同一の区分に属するキー値を取るようにソートし、当該ブロック内のデータを、当該１つのキーでソートするステップと
を含む、方法。
【請求項１０】
複数のキーを有するデータであって、当該複数のキーを用いて複数のデータベース表にアクセスする処理に対して入力されるデータである入力データをソートする装置として、コンピュータを機能させるプログラムであって、
前記コンピュータを、
前記複数のキーのそれぞれについて、キーが取り得るキー値の各々が属する区分を、各区分に属するキー値に対応する前記データベース表のデータがバッファに読み込み可能なサイズとなるように決定する決定部と、
前記複数のキーがそれぞれ同一の区分に属するキー値を取る前記入力データのブロックを、隣接するブロック間で当該複数のキーの１つのキー以外のキーが同一の区分に属するキー値を取るようにソートし、当該ブロック内のデータを、当該１つのキーでソートするソート部と
して機能させる、プログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、データをソートする装置及び方法に関する。特に、本発明は、複数のキーを有するデータであって、その複数のキーを用いて複数のデータベース表にアクセスする処理に対して入力されるデータである入力データをソートする装置及び方法に関する。
【背景技術】
【０００２】
金融機関等のコンピュータシステムでは、通常、オンライン端末からデータを入力することで、データに対する処理が行われる。ところが、例えば、大量の顧客の情報を登録する場合のように、オンライン端末からデータを入力するのが現実的でない場合もある。こ

のような場合は、入力するデータを含むファイルを作成し、このファイルを入力することで、データに対する一括処理（「センタカット処理」ともいう）を行うことがある。
ここで、金融機関等のコンピュータシステムでは、数百万件にも及ぶ大量のレコードを決められた時間内に一括処理する必要があり、一括処理の処理性能が課題になっている。一括処理では、オンライン端末からデータを入力する場合と同様の処理が行われるため、入力されたデータに応じて様々なデータベースの表にランダムにアクセスする必要がある。このようにデータベース処理でランダムなディスク読み込みが発生した場合、１回の読み込み当たりの時間が長くなり、これが処理性能に大きな影響を及ぼすことになる。

【０００３】

そこで、従来から、データベースに対する処理の高速化を図る技術が提案されている（例えば、特許文献１、２参照）。
特許文献１では、問い合わせ要求が発行された段階でその問い合わせを処理するトランザクションが該当ブロックに対してレコードの順序を意識した処理を行う場合、順序識別子を判定し、自トランザクションが再利用可能なソート状態であるか否かを判断し、自トランザクションが処理不可能または処理コストが大きいと判断した場合、そのブロックを占有し、ブロック内のレコードをソート、ソートした時の順序識別子を書き換え、ブロックのソート状態を変更後、ブロックを解放している。

【０００４】

特許文献２では、問い合わせのあったクエリーの最適化を行って関係代数で構成された処理ツリーを作成し、この処理ツリーからトポロジカル・ソートの順番で直接的に処理できるタスクを見出し、これらのタスクをそれぞれのタスクがアクセスするデータベースのリレーションをもとにグループ分けを行い、共通のリレーションに対して処理を行うタスクを集めたグループをグループ列（キュー）に挿入し、利用できるオペレーティングシステムのプロセスの数に応じてグループ列からグループを取り出して、それぞれのプロセスにマルチオペレーション法を用いたグループの処理を行わせている。

【０００５】

また、メモリやデータベースにおける処理を効率よく実行するための技術も提案されている（例えば、特許文献３、４参照）。
特許文献３では、第１のバッファ領域と第２のバッファ領域とを含むメモリのそれぞれのアドレスに、並び替えるべきデータが書き込まれるデータ領域と次の書込み先アドレスを示すアドレス値があらかじめ書き込まれたアドレス領域とを形成し、データの書込みの際にそのアドレスのアドレス領域からのアドレス値に従って順次それぞれのアドレスをアクセスしてデータを所定の順序にて書き込み、第１のバッファ領域にデータが書き込まれている際には第２のバッファ領域を順次シーケンシャルにアクセスして、第２のバッファ領域にデータが書き込まれている際には第１のバッファ領域をシーケンシャルにアクセスして入力されたデータ配列とは異なる配列にてデータを読み出している。

【０００６】

特許文献４では、データ位置管理サーバが、記憶装置からボリューム物理記憶位置管理情報と物理記憶装置稼動情報を収集して記憶し、ＤＢホストのＤＢＭＳ内のスキーマ情報から必要な情報を収集して記憶し、ＤＢホストにおけるマッピング情報と仮想ボリュームスイッチ中の仮想ボリューム情報を収集して記憶し、ＤＢホストから実行履歴情報を取得して記憶し、それらの情報をもとにより良好な性能特性を持つデータ再配置案を決定し、それを実現するためのデータ移動指示を発行している。

【先行技術文献】

【特許文献】

【０００７】

【特許文献１】特開平１１－３２６０号公報
【特許文献２】特開２００８－１６５６２２号公報
【特許文献３】特開平１１－８８１９９号公報
【特許文献４】特開２００３－１５０４１４号公報

【発明の概要】
【発明が解決しようとする課題】
【０００８】
ところで、データベースからのデータ読み込みに関し、既存のＲＤＢ（Relational DataBase）製品は、動的なシーケンシャルアクセス機能を有している。これは、ランダムアクセス中にレコードの読み込みパターンを認識し、読み込み対象のレコードが物理的に近い場所に格納されていると判断した場合は、ランダムアクセスを動的にシーケンシャルアクセスに変更する機能である。このシーケンシャルアクセス機能では、１回のＩ／Ｏで、連続した大量のデータをバッファプールに読み込むことができるので、データ読み込みに要する時間を短縮することができる。つまり、一括処理の入力データの並び順と各データベース表のデータの並び順とがほぼ同じであれば、ランダム読み込みをシーケンシャル読み込みに変えることができ、処理性能を大幅に向上させることができる。
しかしながら、上記特許文献の何れにも、入力データの並び順に着目したデータベース表へのアクセスの高速化手法は記載されていない。
【０００９】
本発明の目的は、入力データの並び順に着目してデータベース表へのアクセスを高速化することにある。
【課題を解決するための手段】
【００１０】
かかる目的のもと、本発明は、複数のキーを有するデータであって、複数のキーを用いて複数のデータベース表にアクセスする処理に対して入力されるデータである入力データをソートする装置であって、複数のキーのそれぞれについて、キーが取り得るキー値の各々が属する区分を、各区分に属するキー値に対応するデータベース表のデータがバッファに読み込み可能なサイズとなるように決定する決定部と、複数のキーがそれぞれ同一の区分に属するキー値を取る入力データのブロックを、隣接するブロック間で複数のキーの１つのキー以外のキーが同一の区分に属するキー値を取るようにソートし、ブロック内のデータを、１つのキーでソートするソート部とを含む、装置を提供する。
【００１１】
ここで、決定部は、区分に区分値を割り当て、ソート部は、複数のキーが取る複数のキー値が属する複数の区分にそれぞれ割り当てられた複数の区分値に基づいて、隣接するブロック間で１つのキー以外のキーが同一の区分に属するキー値を取ることとなるようなブロックの順序を示す順序情報を生成し、順序情報を用いてブロックをソートする、ものであってよい。
区分値は、０から（ｓ－１）までの整数であり、ソート部は、複数の区分値を並べることによりブロックの識別情報を生成し、以下の式を用いて、順序情報を生成するために並べられる複数の順序値を算出することにより、識別情報を順序情報に変換する、ものであってよい（但し、ｓは自然数であり、区分値ｉは識別情報のｉ番目の区分値であり、順序値ｉは順序情報のｉ番目の順序値であり、順序値はｓを法とする剰余類で表す）。
【数１】

$$\left(\text{順序値}1 \quad \text{順序値}2 \quad \ldots \quad \text{順序値}\ n\right)=\left(\text{区分値}1 \quad \text{区分値}2 \quad \ldots \quad \text{区分値}\ n\right)\begin{pmatrix}1 & 1 & \ldots & 1\\ 0 & 1 & \ldots & 1\\ \ldots & \ldots & \ldots & \ldots\\ 0 & 0 & \ldots & 1\end{pmatrix}$$

ソート部は、以下の式を用いて、順序情報における１つのキーに対応する順序値の位置ｐを決定する、ものであってよい（但し、順序値ｉは順序情報のｉ番目の順序値を表す）。
【数２】

$$p=\{k\mid(\text{順序値}k>0)\wedge(\forall m>k,\text{順序値}m=0)\}$$

入力データは、複数のレコードを含み、ソート部は、複数のレコードの各レコードに対し、各レコードにおいて複数のキーが取る複数のキー値に基づいて得られた順序情報と、各レコードにおいて１つのキーが取るキー値とを含むソートキーを付加し、複数のレコードを、各レコードに付加されたソートキーでソートする、ものであってよい。

この装置は、入力データが有するＭ個のキーから、データベース表のデータをシーケンシャルに読み込み可能と判断するための条件を満たすＮ個のキーを、複数のキーとして選択する選択部を更に含む、ものであってよい（Ｍ，ＮはＭ＞Ｎを満たす整数）。

【００１２】

また、本発明は、複数のレコードを含むデータであって、複数のレコードの各レコードについて、各レコードが有するＭ個のキーを用いて複数のデータベース表にアクセスする処理を、複数のレコードについて一括して行う一括処理の対象となるデータである入力データをソートする装置であって、Ｍ個のキーのそれぞれについて、キーが取り得るキー値の各々が属する区分を、各区分に属するキー値に対応するデータベース表のデータがバッファに読み込み可能なサイズとなるように決定する決定部と、Ｍ個のキーから、データベース表のデータをシーケンシャルに読み込み可能と判断するための条件を満たすＮ個のキーを選択する選択部と、複数のレコードの各レコードに対し、各レコードにおいてＮ個のキーが取るＮ個のキー値が属するＮ個の区分にそれぞれ割り当てられたＮ個の区分値に基づいて生成された情報であって、隣接するブロック間でＮ個のキーの１つのキー以外のキーが同一の区分に属するキー値を取ることとなるような入力データのブロックの順序を示す情報である順序情報と、各レコードにおいて１つのキーが取るキー値とを含むソートキーを付加する付加部と、複数のレコードを、複数のレコードの各々に付加されたソートキーでソートするソート部とを含む、装置も提供する（Ｍ，ＮはＭ＞Ｎを満たす整数）。

【００１３】

また、本発明は、複数のキーを有する入力データの入力に応じて、複数のキーを用いて複数のデータベース表にアクセスする処理を行うシステムであって、複数のキーのそれぞれについて、キーが取り得るキー値の各々が属する区分を、各区分に属するキー値に対応するデータベース表のデータがバッファに読み込み可能なサイズとなるように決定する決定部と、複数のキーがそれぞれ同一の区分に属するキー値を取る入力データのブロックを、隣接するブロック間で複数のキーの１つのキー以外のキーが同一の区分に属するキー値を取るようにソートし、ブロック内のデータを、１つのキーでソートすることにより、ソート済み入力データを生成する生成部と、生成部により生成されたソート済み入力データの入力に応じて、処理を行う処理部とを含む、システムも提供する。

【００１４】

更に、本発明は、複数のキーを有するデータであって、複数のキーを用いて複数のデータベース表にアクセスする処理に対して入力されるデータである入力データをソートする方法であって、複数のキーのそれぞれについて、キーが取り得るキー値の各々が属する区分を、各区分に属するキー値に対応するデータベース表のデータがバッファに読み込み可能なサイズとなるように決定するステップと、複数のキーがそれぞれ同一の区分に属するキー値を取る入力データのブロックを、隣接するブロック間で複数のキーの１つのキー以外のキーが同一の区分に属するキー値を取るようにソートし、ブロック内のデータを、１つのキーでソートするステップとを含む、方法も提供する。

【００１５】

更にまた、本発明は、複数のキーを有するデータであって、複数のキーを用いて複数のデータベース表にアクセスする処理に対して入力されるデータである入力データをソートする装置として、コンピュータを機能させるプログラムであって、コンピュータを、複数のキーのそれぞれについて、キーが取り得るキー値の各々が属する区分を、各区分に属するキー値に対応するデータベース表のデータがバッファに読み込み可能なサイズとなるように決定する決定部と、複数のキーがそれぞれ同一の区分に属するキー値を取る入力データのブロックを、隣接するブロック間で複数のキーの１つのキー以外のキーが同一の区分に属するキー値を取るようにソートし、ブロック内のデータを、１つのキーでソートする

ソート部として機能させる、プログラムも提供する。

【発明の効果】

【００１６】

本発明によれば、入力データの並び順に着目してデータベース表へのアクセスを高速化することができる。

【図面の簡単な説明】

【００１７】

【図１】本発明の実施の形態の概要について説明するための図である。

【図２】本発明の実施の形態における一括処理システムの構成例を示したブロック図である。

【図３】本発明の実施の形態の一括処理システムにおける前処理装置の区分分割部の動作例を示したフローチャートである。

【図４】本発明の実施の形態の一括処理システムにおける前処理装置のキー選択部の動作例を示したフローチャートである。

【図５】本発明の実施の形態の一括処理システムにおける前処理装置のソートキー付加部の動作例を示したフローチャートである。

【図６】本発明の実施の形態の一括処理システムにおける前処理装置で出力されるソートキー付き入力ファイルの内容の一例を示した図である。

【図７】本発明の実施の形態の一括処理システムにおける一括処理装置の動作について説明するための図である。

【図８】本発明の実施の形態の一括処理システムにおける一括処理装置の動作について説明するための図である。

【図９】本発明の実施の形態を適用可能なコンピュータのハードウェア構成を示した図である。

【発明を実施するための形態】

【００１８】

以下、添付図面を参照して、本発明の実施の形態について詳細に説明する。

まず、本発明の実施の形態の概要を説明する。

本発明の実施の形態は、一括処理において複数のデータベース表（以下、「テーブル」という）にシーケンシャルにアクセスできるよう、一括処理の入力データに対して、複数のテーブルのキーを用いた特殊なロジックによる事前ソートを行うものである。以下、この特殊ロジックによる事前ソートを「特殊ソート」と呼ぶことにする。

【００１９】

特殊ソートの考え方は、次の通りである。

まず、一括処理の入力データ内のデータベースアクセスのためのキー値に対し、キー値のレンジに応じて、区分値を割り当てる。

図１（ａ）に、この区分値の割り当てについて示す。

ここでは、あるテーブルのキー値が「０００００」から「９９９９」までの値をとるものとしている。そして、図示するように、「０００００」から「０９９９」までのキー値に対して区分値「０」を、「１０００」から「１９９９」までのキー値に対して区分値「１」を、「９０００」から「９９９９」までのキー値に対して区分値「９」を、それぞれ割り当てている。

【００２０】

また、一括処理でアクセスする複数のテーブルのキーについて、同様に区分値を割り当てる。そして、区分値を結合したものをブロックの識別情報の一例であるブロック番号とし、一括処理の入力データをブロック番号ごとに分類する。例えば、３つのキーがあれば、区分値も３つになり、「０００」、「００１」、「００２」、・・・、「９９９」のようなブロック番号で識別される複数のブロックができる。この場合、１つのブロックは、同じ区分値に対応する複数件のデータからなり、キー値が近いデータがまとめられたものとなる。

【００２１】
次に、複数のブロックを、隣り合う２つのブロックのブロック番号に含まれる１つの区分値だけが異なるように並べる。
図１（ｂ）に、このときのブロック番号の並べ方について示す。尚、本明細書では、ブロック番号を構成する複数の区分値を、左から何番目であるかによって区別するものとする。
図示するように、「０００」から「００９」までは、３番目の区分値だけが変化するように並べている。「００９」の次は、「０１０」とすると２つの区分値が異なることになるので、それを避けるため、２つ目の区分値だけが変化するように「０１９」としている。「０１０」から「０１８」までは、再び３番目の区分値だけが変化するように並べている。「０１８」の次は、２番目の区分値だけが変化するように「０２８」としている。「０２９」から「０２７」までは、再び３番目の区分値だけが変化するように並べている。
【００２２】
次いで、ブロック内のデータを、そのブロックのブロック番号と１つ前のブロックのブロック番号とで区分値が変わったキーを用いてソートする。例えば、ブロック番号が「００８」から「００９」に変わった場合は３番目のキーで、ブロック番号が「００９」から「０１９」に変わった場合は２番目のキーで、ブロック内のデータをソートする。
【００２３】
このように、一括処理の入力データを複数キーに対応する区分値より区分してブロックにまとめ、隣接するブロック間で１つのキーに対応する区分値のみ異なるようにする。また、ブロック内では、１つ前のブロックから変化した区分値に対応するキーでソートする。こうすると、ブロック内では、ソートに用いたキーに対応するテーブルについて、シーケンシャルアクセスが可能となり、高速に読み込みが行えるようになる。一方、ブロック間で区分値が変わらなかったキーに対応するテーブルについてはランダムアクセスになるが、キーのレンジは１つ前のブロックの場合と同じであり、前ブロックの処理中にデータがバッファプールに読み込まれているので、バッファヒットし高速に読み込みが行える。
【００２４】
図２は、本実施の形態における一括処理システムの構成を示したブロック図である。
図示するように、一括処理システムは、前処理装置１０と、一括処理装置２０とを備える。また、入力装置３０と、データベース（ＤＢ）４０と、記憶装置５０と、入力ファイル６０と、ソートキー付き入力ファイル７０と、ソート済み入力ファイル８０とを含む。
【００２５】
前処理装置１０は、一括処理装置２０のアプリケーションロジックを変更することなく高速に一括処理を行うために、入力ファイル６０内のデータを並べ替えて、ソート済み入力ファイル８０を出力する装置である。この前処理装置１０は、例えば大型汎用コンピュータやＰＣ（Personal Computer）により実現される。
一括処理装置２０は、前処理装置１０で出力されたソート済み入力ファイル８０内のデータをソート後の順序で読み込み、一括処理を行う装置である。この一括処理装置２０は、例えば大型汎用コンピュータにより実現される。本実施の形態では、処理部の一例として、一括処理装置２０を設けている。
【００２６】
入力装置３０は、前処理装置１０に対して、各種パラメータを入力する装置である。この入力装置３０は、例えばキーボード／マウス９０ｉ（図９参照）により実現される。
ＤＢ４０は、一括処理装置２０における一括処理の対象となるデータの集合であり、ＲＤＢの複数のテーブルを含む。また、テーブルの管理やテーブルへのアクセス要求の処理を行うソフトウェアであるＤＢＭＳ（DataBase Management System）をＤＢ４０に含めてもよい。このＤＢ４０は、例えば磁気ディスク装置９０ｇ（図９参照）により、また、ＤＢＭＳを含める場合は、これに加えて、例えば磁気ディスク装置９０ｇ（図９参照）に格納されたＤＢＭＳのプログラムをＣＰＵ９０ａ（図９参照）がメインメモリ９０ｃ（図９参照）に読み込んで実行することにより、実現される。

記憶装置５０は、前処理装置１０で生成され、使用される情報を記憶する装置である。この記憶装置５０は、例えば磁気ディスク装置９０ｇ（図９参照）により実現される。

【００２７】

入力ファイル６０は、一括処理装置２０に入力すべき複数件のデータを格納したファイルである。この入力ファイル６０は、例えば磁気テープや磁気ディスク装置９０ｇ（図９参照）に記憶しておくとよい。本実施の形態では、入力データの一例として、入力ファイル６０を用いている。

ソートキー付き入力ファイル７０は、入力ファイル６０に格納された複数件のデータに対して、前処理装置１０がソートキーを付加することにより生成されたファイルである。このソートキー付き入力ファイル７０も、例えば磁気テープや磁気ディスク装置９０ｇ（図９参照）に記憶しておくとよい。

ソート済み入力ファイル８０は、ソートキー付き入力ファイル７０に格納された複数件のデータをソートキーでソートすることにより生成されたファイルである。このソート済み入力ファイル８０も、例えば磁気テープや磁気ディスク装置９０ｇ（図９参照）に記憶しておくとよい。本実施の形態では、ソート済み入力データの一例として、ソート済み入力ファイル８０を用いている。

【００２８】

次に、前処理装置１０の機能構成について、更に詳しく説明する。

図２に示すように、前処理装置１０は、区分分割部１１と、キー選択部１２と、ソートキー付加部１３と、ソート処理部１４とを備える。

【００２９】

区分分割部１１は、一括処理で参照されるテーブルのキー名、バッファプールサイズ、レコード数、レコード長、キー値等の情報を取得し、テーブルごとに区分分割方法を決定する。また、区分分割方法を決定する際に得られた情報を記憶装置５０に記憶する。本実施の形態では、区分を決定する決定部の一例として、区分分割部１１を設けている。

キー選択部１２は、入力ファイル６０内のレコード数と、記憶装置５０に記憶された情報とに基づいて、特殊ソートに用いる複数のキーを選択する。本実施の形態では、Ｎ個のキーを選択する選択部の一例として、キー選択部１２を設けている。

【００３０】

ソートキー付加部１３は、入力ファイル６０内のデータを読み込み、キー選択部１２が選択した複数のキーに基づいて、レコードごとに、高速に処理するための並び順を示すソートキーを生成する。そして、ソートキーをレコードに付加することにより、ソートキー付き入力ファイル７０を生成する。本実施の形態では、ソート部のソートキーを付加する機能、ソートキーを付加する付加部、ソート済み入力データを生成する生成部の一例として、ソートキー付加部１３を設けている。

ソート処理部１４は、ソートキー付加部１３で生成されたソートキー付き入力ファイル７０内のデータ全体をソートする。この結果、高速処理可能な順番にソートされたデータが得られる。尚、このソート処理部１４では、ソートキーの昇順にソートすればよいので、既存のソートユーティリティを用いてソートすればよい。本実施の形態では、ソート部の一例として、ソート処理部１４を設けている。

【００３１】

次に、本実施の形態における一括処理システムの動作について説明する。

第一に、前処理装置１０の区分分割部１１の動作について説明する。この区分分割部１１は、特殊ソートに用いる候補となるテーブルに関する情報を取得し、テーブルごとに特殊ソートを実施する場合の適切な区分数と区分分割方法とを決定する。

【００３２】

図３は、区分分割部１１の動作例を示したフローチャートである。

まず、入力ファイル６０に含まれ、一括処理で使用される複数のテーブルのそれぞれについて、そのテーブルのテーブル名、そのテーブルのキー名、そのテーブルに対するＤＢＭＳにおけるバッファプールのサイズを、パラメータとして入力装置３０から与える。す

ると、区分分割部１１は、これらの情報を取得する（ステップ１０１）。複数のキーは、特殊ソートに使うキーの候補となるが、実際に使われるキーは、キー選択部１２で選択される。

【００３３】

また、区分分割部１１は、一括処理で使用されるテーブルごとに、そのテーブルのレコード数、そのテーブルのレコード長、ステップ１０１でキー名を取得したキーが取るキー値を、ＤＢＭＳから取得する（ステップ１０２）。

【００３４】

これにより、区分分割部１１は、テーブルごとに、バッファプールに格納可能なレコード数（以下、「格納可能レコード数」という）を算出する（ステップ１０２）。この場合、テーブルごとに専用のバッファプールがあれば、格納可能レコード数は次の式で算出する。

格納可能レコード数＝バッファプールサイズ／レコード長

尚、上記式において、バッファプールサイズはステップ１０１で取得したものであり、レコード長はステップ１０２で取得したものである。

或いは、複数のテーブルが１つのバッファプールを共有する場合は、次の式で算出してもよい。

格納可能レコード数＝バッファプールサイズ／レコード長／バッファプールを共有するテーブルの数／余裕率

【００３５】

また、区分分割部１１は、テーブルごとに、特殊ソートに適した区分数を次の式で算出する（ステップ１０４）。

区分数＝レコード数／格納可能レコード数

但し、小数点以下は切り上げるものとする。

尚、上記式において、レコード数はステップ１０２で取得したものであり、格納可能レコード長はステップ１０３で算出したものである。

【００３６】

更に、区分分割部１１は、各区分がほぼ同数のレコードを含むこととなるように、各区分に含めるキー値のレンジを決定し、レンジごとに「０」から始まる区分値を割り当てる（ステップ１０５）。この場合、キー値の分布が一様であれば、次の式のように、最大キー値と最小キー値の間を区分数で等しく分割することによって区分値を割り当てる。

区分値＝（キー値－最小キー値）×区分数／（最大キー値－最小キー値＋１）

但し、小数点以下は切り捨てるものとする。尚、ここでは、キー値の刻み幅が「１」であるものとして分母に「１」を加えたが、一般的には、キー値の刻み幅を加えるとよい。

尚、上記式において、各キー値はステップ１０２で取得したものであり、区分数はステップ１０４で算出したものである。

或いは、キーの分布が一様でない場合は、各区分がほぼ同数のレコードを含むこととなるように、キーの情報を読み込んで、次の式でキーのレンジを決定してもよい。

区分値＝｛ｋ｜レンジ区切り（ｋ）≦キー値＜レンジ区切り（ｋ＋１）｝

例えば、あるテーブルのキー値が「０００００」から「９９９９」までの値をとり、一様な分布をしており、区分数が「１０」であれば、キー値のレンジに対して、図１（ａ）のように区分値を割り当てることになる。

【００３７】

最後に、区分分割部１１は、これらの処理の結果を、キー選択部１２及びソートキー付加部１３で用いることができるように、記憶装置５０に保存する（ステップ１０６）。

尚、この区分分割部１１の処理は、特殊ソートの特性を決めるためのものなので、テーブルのレコード数やキーの分布が大きく変わらない限り、一度行っておけば、再度行う必要はない。つまり、区分分割部１１の処理は、前処理装置１０における処理を再実行する際にスキップしてもよい。

【００３８】

第二に、前処理装置１０のキー選択部１２の動作について説明する。このキー選択部１２は、入力ファイル６０内のデータの件数と、記憶装置５０に記憶された情報とから、特殊ソートに用いる複数のキーを選択する。ところで、複数のキーでデータをブロック化したとき、ブロック内のデータは、ＲＤＢ製品のシーケンシャルアクセス機能が働き得る程度の密度を持っている必要がある。例えば、入力データが１０００万件で各キーの区分数が「１０」の場合、４つのキーでブロック化しても、１ブロック内のデータは１０００件程度となり、シーケンシャルアクセス機能が働く可能性が高い。一方、入力データが１０万件で各キーの区分数が「１０」の場合、4つのキーでブロック化すると、１ブロック内のデータは１０件になってしまい、シーケンシャルアクセス機能が働く可能性は低くなる。そこで、キー選択部１２は、入力データの件数に応じた適切な特殊ソートが行われるように動作する。

【００３９】

図４は、キー選択部１２の動作例を示したフローチャートである。

まず、キー選択部１２は、入力ファイル６０から、その中に含まれるレコードの数を取得する（ステップ１２１）。

また、キー選択部１２は、区分分割部１１が記憶装置５０に記憶したテーブルごとのキー名、区分数、レコード数を取得する（ステップ１２２）。

【００４０】

次に、ステップ１２２で情報を取得したテーブルのうち、区分数が「１」以外のテーブルを、区分数が小さい順に並べる（ステップ１２３）。つまり、区分数が「１」のテーブルは、常にバッファヒットし高速にアクセスできるため、特殊ソートに用いるキーから除外する。また、区分数が少ないテーブルを選択することが、後述するステップ１２５での条件を満たす上で有利なので、区分数が少ない順にテーブルを並べる。

【００４１】

そして、キー選択部１２は、１つのテーブルの情報を読み込む（ステップ１２４）。

ここで、複数のキーでデータをブロック化し、シーケンシャルアクセス機能が働くようにするためには、次の条件が満たされる必要がある。

入力ファイルのレコード数／（区分数１×区分数２×・・・×区分数ｎ）＞ｍａｘ（テーブルｉのレコード数／区分数ｉ）／係数

但し、「テーブルｉ」はｉ番目のテーブルを表し、「区分数ｉ」はｉ番目のテーブルの区分数を表し、「ｍａｘ（Ｘ）」はｉが１からｎまでの値をとるときのＸの最大値を表している。また、「係数」は、今回読んだページがページシーケンシャルとして扱われるときの直前に読んだページからのページ数である。ここで、ページとは、磁気ディスク上でのデータの格納単位であり、この単位で読み書きが行われる。ある製品の場合、係数は「１６」である。つまり、今回読んだページは、直前に読んだページから１６ページ以内にあれば、ページシーケンシャルとして扱われる。そして、連続して読み込んだ８ページ中、５ページ以上がページシーケンシャルであれば、シーケンシャルアクセスが動的に起動されるようになっている。尚、「係数」は製品によって異なるので、使用するＤＢＭＳ製品に対する適切な値を使用する。

そこで、キー選択部１２は、ステップ１２４で情報を読み込んだテーブルについて、上記条件が満たされているかを判定する（ステップ１２５）。尚、上記条件において、入力ファイルのレコード数はステップ１２１で取得したものであり、区分数ｉ、テーブルｉのレコード数はステップ１２２で取得したものである。

【００４２】

ステップ１２５での判定の結果、ステップ１２４で情報を読み込んだテーブルについて上記条件が満たされていれば、キー選択部１２は、そのテーブルのキーを選択する（ステップ１２６）。

その後、キー選択部１２は、未処理のテーブルがあるかどうかを判定する（ステップ１２７）。未処理のテーブルがあれば、ステップ１２４へ戻り、次のテーブルについて、ステップ１２５での条件判定を行う。そして、これを、ステップ１２５で条件が満たされな

くなるまで、又は、ステップ１２７で未処理のテーブルがなくなるまで、繰り返す。これにより、複数のキーから、条件を満たすできるだけ多くのキーの組が選択されることになる。

【００４３】

第三に、前処理装置１０のソートキー付加部１３の動作について説明する。このソートキー付加部１３は、望ましいレコード順を表すソートキーを生成し、ソートキーをレコードに付加する。即ち、入力ファイル６０内のデータを、キー選択部１２で選択された複数のキーによってブロック化した後、ブロックを効率よく処理できるように並べるとともに、ブロック内は特定のキーによってソートしたい。そこで、ソートキー付加部１３は、ブロックの順序を示す値（以下、「ブロック順序値」という）とブロック内のソートに用いるキーのキー値（以下、「ブロック内キー値」という）とを連結することによりソートキーを作成し、ソートキーをレコードに付加する。ここで、ブロック順序値は、ブロックの順序を示す順序情報の一例である。

【００４４】

図５は、ソートキー付加部１３の動作例を示したフローチャートである。

まず、ソートキー付加部１３は、キー選択部１２から複数のキーを取得する（ステップ１４１）。

また、ソートキー付加部１３は、区分分割部１１が記憶装置５０に記憶した情報のうち、キーごとのキー値のレンジと区分値との対応を取得する（ステップ１４２）。

【００４５】

次に、ソートキー付加部１３は、入力ファイル６０から１つのレコードを読み込む（ステップ１４３）。

また、ソートキー付加部１３は、そのレコードにおける複数のキーに対応する区分値をステップ１４２で取得した情報を参照して求め、これらの区分値を結合することにより、ブロック番号を生成する（ステップ１４４）。例えば、キーが３つあれば、ブロック番号は「０００」、「００１」、「００２」、・・・のようになる。

【００４６】

ところで、ブロック番号の順に単純にブロックを並べると、例えば、ブロック番号「０９９」のブロックの次がブロック番号「１００」のブロックとなってしまう。即ち、隣り合うブロック間で複数の区分値が変わってしまうことがあり、これでは、本実施の形態が目指す動作が行えない。従って、本実施の形態では、ブロックの並びを工夫し、この工夫した並びにおけるブロックの順序を示すブロック順序値をブロック番号から生成し、ソートキーとして用いる。

【００４７】

この場合、ブロックは、隣り合うブロック間でブロック番号の１つの区分値だけが異なるように並べる。例えば、３つのキーがあり、各キーの区分数が「１０」である場合を考える。この場合は、図１（ｂ）に示したように、３番目の区分値を１つずつ増やしていき、一巡したら、２番目の区分値を増やすようにブロックを並べればよい。尚、左側の区分値を増やすときに右側の区分値は変えないようにする。

ソートキー付加部１３は、このような並びを実現するために、ブロック順序値をブロック番号に変換する写像の逆写像を用いて、ブロック番号からブロック順序値を算出する（ステップ１４５）。具体的には、次の式でブロック順序値を算出する。但し、次の式は、区分数を法とする剰余類として計算するものとする。また、ブロック順序値の左からｉ番目のキーに対応する値を「順序値ｉ」と表記するものとする。

【００４８】

【数３】

$$(\text{順序値}1 \quad \text{順序値}2 \quad \ldots \quad \text{順序値}n)=(\text{区分値}1 \quad \text{区分値}2 \quad \ldots \quad \text{区分値}n)\begin{pmatrix}1 & 1 & \ldots & 1\\ 0 & 1 & \ldots & 1\\ \ldots & \ldots & \ldots & \ldots\\ 0 & 0 & \ldots & 1\end{pmatrix}$$

【００４９】

これにより、例えば、ブロック番号「０１９」はブロック順序値「０１０」に、ブロック番号「０１０」はブロック順序値「０１１」に変換される。この変換後の値を、ブロック順序を表すソートキーに用いて、ソート処理部１４で入力ファイル６０内のデータをソートすることにより、望ましいブロックの並びを実現できる。

【００５０】

また、前述の通り、ブロックは、隣り合うブロック間でブロック番号の１つの区分値だけが異なっている。一括処理において、区分値が変化しなかったキーに対応するテーブルのデータの読み込みはバッファヒットするが、区分値が変化したキーに対応するテーブルのデータの読み込みはディスクから行われる。そこで、変化した区分値に対応するキーを使って、ブロック内のデータをソートすることにより、一括処理におけるディスクからのデータ読み込みをシーケンシャルに行えるようにする。

例えば、あるブロックのブロック番号が「００９」で、続くブロックのブロック番号が「０１９」の場合、後者のブロックでは２つ目のキーの区分値に対応するデータがＤＢ４０のテーブルから読み込まれるので、この読み込みがシーケンシャルになるように２つ目のキーでソートしたい。このために、ブロック番号「０１９」のブロックでは、ブロック内キー値に、データ中の２つ目のキーのキー値を使用する。

ここで、ブロック内ソートに用いるキー（以下、「ブロック内キー」という）に対応する順序値のブロック順序値の中での位置であるブロック内キー位置は、ブロック番号を変換して得られたブロック順序値から次の式を用いて決定できる。尚、この式においても、上記同様、ブロック順序値の左からｉ番目のキーに対応する値を「順序値ｉ」と表記するものとする。

【００５１】

【数４】

$$\text{ブロック内キー位置}=\{k\mid(\text{順序値}k>0)\wedge(\forall m>k,\text{順序値}m=0)\}$$

【００５２】

即ち、ソートキー付加部１３は、ブロック内キーとして用いるキーを決定し、そのキーのキー値をコピーすることにより、ブロック内キー値を生成する（ステップ１４６）。

そして、ソートキー付加部１３は、ステップ１４５で生成したブロック順序値とステップ１４６で生成したブロック内キー値とを結合することによりソートキーを生成し、これをステップ１４３で読み込んだレコードに付加する（ステップ１４７）。

【００５３】

その後、ソートキー付加部１３は、未処理のレコードがあるかどうかを判定する（ステップ１４８）。未処理のレコードがあれば、ステップ１４３へ戻り、次のレコードについて、ソートキーの付加を行う。そして、これを、ステップ１４８で未処理のレコードがなくなるまで、繰り返す。これにより、全てのレコードに対して、ブロック順序値とブロック内キー値とからなるソートキーが付加され、ソートキー付き入力ファイル７０として出力される。

【００５４】

図６に、ソートキー付き入力ファイル７０内のデータの一例を示す。

図示するように、ソートキー付き入力ファイル７０内のデータは、入力ファイル６０に元々含まれていたレコードデータに対して、ソートキーが付加されたものである。そして、ソートキーは、ブロック順序値と、ブロック内キー値とを含み、レコードデータは、複

数のキー値を含んでいる。
ここで、キー値のレンジと区分値との対応は、何れのキーについても、図１（ａ）に示したようなものであるとする。
すると、例えば１行目のブロックのブロック番号は「３４６」となる。これをステップ１４５で変換することで、ブロック順序値は「３７３」となっている。そして、ステップ１４６でブロック内キー位置は「３」になるので、ブロック内キー値は「６８６０」となっている。
また、例えば３行目のブロックのブロック番号は「３１６」となる。これをステップ１４５で変換することで、ブロック順序値は「３４０」となっている。そして、ステップ１４６でブロック内キー位置は「２」になるので、ブロック内キー値は「１２１１」となっている。
更に、例えば９行目のブロックのブロック番号は「４６０」となる。これをステップ１４５で変換することで、ブロック順序値は「４００」となっている。そして、ステップ１４６でブロック内キー位置は「１」になるので、ブロック内キー値は「４８５８」となっている。
【００５５】
第四に、前処理装置１０のソート処理部１４の動作について説明する。このソート処理部１４は、ソートキー付加部１３によってレコードに付加されたソートキーを用いて、入力ファイル６０内のデータ全体をソートする。即ち、ブロック順序値とブロック内キー値とからなるソートキーがレコードに付加されているので、ソート処理部１４は、このソートキーを用いて昇順にデータをソートする。
このソートは、データの特定のキーを用いてそのデータを昇順にソートするという意味で、一般に行われているソートと同じである。従って、このソートは、既知のソートアルゴリズムを用いてプログラムを開発し、これを実行することにより行ってもよいし、既存のソートユーティリティーを用いて行ってもよい。
そして、ソートの結果、ソート処理部１４は、一括処理を効率的に実行できる順番に並べられたデータを含むソート済み入力ファイル８０を出力する。
尚、ソート済み入力ファイル８０内のレコードにはソートキーが付加されているが、このソートキーは、ソートが終了すれば、一括処理のためには不要である。従って、ソートキーは、一括処理装置２０がソート済み入力ファイル８０内のデータを読み込む際に読み飛ばしてもよいし、ソート中又はソート終了後に削除してソート済み入力ファイル８０に含まれないようにしてもよい。
【００５６】
第五に、一括処理装置２０の動作について説明する。
前処理装置１０における特殊ソートにより、ソート済み入力ファイル８０において、データは次のように並んでいる。
・ブロック内では、キーの区分値が同一である。
・隣のブロックとは、１つのキーについてだけ、キーの区分値が異なる。
・ブロック内では、１つ前のブロックと区分値が異なるキーでソートされている。例えば、ブロック番号「０９９」、ブロック番号「０９０」、ブロック番号「１９０」の順でブロックが並んだ場合、ブロック番号「０９０」のブロックのデータは３つ目のキーで、ブロック番号「１９０」のブロックのデータは１つ目のキーでソートされている。
【００５７】
一括処理装置２０は、ソート済み入力ファイル８０内のデータを順次読み込み、一括処理を実行していく。すると、区分値が変化したキーに対応するテーブルのデータは、シーケンシャルに読み込まれ、区分値が変化しなかったキーに対応するテーブルのデータは、前のブロックの処理時にバッファプールに読み込まれているためバッファヒットとなり、いずれも高速に処理される。
【００５８】
図７に、このときの処理について示す。（ａ）は、ブロック番号「０９９」のブロック

を処理した後にブロック番号「０９０」のブロックを処理するときの例であり、（ｂ）は、その後にブロック番号「１９０」のブロックを処理するときの例である。尚、ブロック番号を構成する区分値のうち、左から１つ目の区分値がテーブル１のキーの区分値であり、左から２つ目の区分値がテーブル２のキーの区分値であり、左から３つ目の区分値がテーブル３のキーの区分値であるものとする。

【００５９】

（ａ）では、まず、一括処理装置２０が、ソート済み入力ファイル８０からブロック番号「０９０」のブロックのデータを読み込んでいる。

次に、一括処理装置２０は、テーブル１にアクセスしている。この場合、１つ前のブロックから、テーブル１のキーの区分値は変化していない。従って、テーブル１のデータはバッファプールに格納されており、高速に処理可能である。

また、一括処理装置２０は、テーブル２にアクセスしている。この場合、１つ前のブロックから、テーブル２のキーの区分値は変化していない。従って、テーブル２のデータはバッファプールに格納されており、高速に処理可能である。

更に、一括処理装置２０は、テーブル３にアクセスしている。この場合、１つ前のブロックから、テーブル３のキーの区分値は変化しているので、ブロックのデータはこのキーでソートされている。従って、テーブル３のデータはＤＢ４０からシーケンシャルに読み込まれ、高速に処理可能である。

【００６０】

（ｂ）では、まず、一括処理装置２０が、ソート済み入力ファイル８０からブロック番号「１９０」のブロックのデータを読み込んでいる。

次に、一括処理装置２０は、テーブル１にアクセスしている。この場合、１つ前のブロックから、テーブル１のキーの区分値は変化しているので、ブロックのデータはこのキーでソートされている。従って、テーブル１のデータはＤＢ４０からシーケンシャルに読み込まれ、高速に処理可能である。

また、一括処理装置２０は、テーブル２にアクセスしている。この場合、１つ前のブロックから、テーブル２のキーの区分値は変化していない。従って、テーブル２のデータはバッファプールに格納されており、高速に処理可能である。

更に、一括処理装置２０は、テーブル３にアクセスしている。この場合、１つ前のブロックから、テーブル３のキーの区分値は変化していない。従って、テーブル３のデータはバッファプールに格納されており、高速に処理可能である。

【００６１】

ここで、図８に、本実施の形態で入力ファイル６０内のデータをソートした場合の一括処理中のテーブルへのアクセス方法の遷移の例を示す。

最初のブロックだけは、キー１とキー２がソート対象になっておらず、キー１を用いたテーブル１へのアクセスとキー２を用いたテーブル２へのアクセスとがランダムアクセスになるため処理時間がかかるが、２番目のブロック以降については全て、シーケンシャル読み込み又はバッファヒットとなるので、高速に処理できる。

【００６２】

以上、本実施の形態について説明した。

このように、本実施の形態では、一括処理に対するデータの入力順を変えるようにした。これにより、一括処理における複数のテーブルへのアクセスが高速化し、処理性能が向上することとなった。

また、本実施の形態では、一括処理の入力データに対して特殊ソートを事前に行うようにした。これにより、一括処理のアプリケーションロジックを変更しなくても、一括処理の高速化が可能となった。

【００６３】

更に、特殊ソートに関しては、次のような効果が挙げられる。

第一に、入力データを複数のキーの区分値の組によりブロック化し、ブロックごとにソートに用いるキーを変えるようにした。これにより、ソート対象となったキーを用いたテ

ーブルからのデータ読み込みをシーケンシャルに行えるようになった。
第二に、キー値をテーブルのバッファーサイズに応じた区分に分割するようにした。これにより、隣接するブロック間でキーの区分値が同じ場合にバッファヒットするようになった。
第三に、ブロックを、複数の区分値のうち１つのキーに対応する区分値だけが変わるように並べることとした。これにより、テーブルからのデータ読み込みが、シーケンシャルアクセス及びバッファヒットの何れかで行えるようになった。
第四に、テーブルに関する情報と入力データ件数とから、ページシーケンシャル動作可能な最大数のキーの組を自動的に選択するようにした。これにより、入力データ件数に応じた性能向上が得られるようになった。

【００６４】
尚、本実施の形態では、入力ファイル６０内の各レコードにソートキーを付加するようにしたが、ソートキーを各レコードに付加しない構成を採用してもよい。例えば、ソートキーを、各レコードと関連付けた状態で、入力ファイル６０とは別のメモリに保持しておき、このメモリに保持したソートキーに基づいて、入力ファイル６０内のレコードをソートするようにしてもよい。

【００６５】
最後に、本実施の形態を適用するのに好適なコンピュータのハードウェア構成について説明する。図９は、このようなコンピュータのハードウェア構成の一例を示した図である。図示するように、コンピュータは、演算手段であるＣＰＵ（Central Processing Unit）９０ａと、Ｍ／Ｂ（マザーボード）チップセット９０ｂを介してＣＰＵ９０ａに接続されたメインメモリ９０ｃと、同じくＭ／Ｂチップセット９０ｂを介してＣＰＵ９０ａに接続された表示機構９０ｄとを備える。また、Ｍ／Ｂチップセット９０ｂには、ブリッジ回路９０ｅを介して、ネットワークインターフェイス９０ｆと、磁気ディスク装置（ＨＤＤ）９０ｇと、音声機構９０ｈと、キーボード／マウス９０ｉと、フレキシブルディスクドライブ９０ｊとが接続されている。

【００６６】
尚、図９において、各構成要素は、バスを介して接続される。例えば、ＣＰＵ９０ａとＭ／Ｂチップセット９０ｂの間や、Ｍ／Ｂチップセット９０ｂとメインメモリ９０ｃの間は、ＣＰＵバスを介して接続される。また、Ｍ／Ｂチップセット９０ｂと表示機構９０ｄとの間は、ＡＧＰ（Accelerated Graphics Port）を介して接続されてもよいが、表示機構９０ｄがＰＣＩ Ｅｘｐｒｅｓｓ対応のビデオカードを含む場合、Ｍ／Ｂチップセット９０ｂとこのビデオカードの間は、ＰＣＩ Ｅｘｐｒｅｓｓ（ＰＣＩｅ）バスを介して接続される。また、ブリッジ回路９０ｅと接続する場合、ネットワークインターフェイス９０ｆについては、例えば、ＰＣＩ Ｅｘｐｒｅｓｓを用いることができる。また、磁気ディスク装置９０ｇについては、例えば、シリアルＡＴＡ（AT Attachment）、パラレル転送のＡＴＡ、ＰＣＩ（Peripheral Components Interconnect）を用いることができる。更に、キーボード／マウス９０ｉ、及び、フレキシブルディスクドライブ９０ｊについては、ＵＳＢ（Universal Serial Bus）を用いることができる。

【００６７】
ここで、本発明は、全てハードウェアで実現してもよいし、全てソフトウェアで実現してもよい。また、ハードウェア及びソフトウェアの両方により実現することも可能である。また、本発明は、コンピュータ、データ処理システム、コンピュータプログラムとして実現することができる。このコンピュータプログラムは、コンピュータにより読取り可能な媒体に記憶され、提供され得る。ここで、媒体としては、電子的、磁気的、光学的、電磁的、赤外線又は半導体システム（装置又は機器）、或いは、伝搬媒体が考えられる。また、コンピュータにより読取り可能な媒体としては、半導体、ソリッドステート記憶装置、磁気テープ、取り外し可能なコンピュータディスケット、ランダムアクセスメモリ（ＲＡＭ）、リードオンリーメモリ（ＲＯＭ）、リジッド磁気ディスク、及び光ディスクが例示される。現時点における光ディスクの例には、コンパクトディスク－リードオンリーメ

モリ（ＣＤ－ＲＯＭ）、コンパクトディスクーリード／ライト（ＣＤ－Ｒ／Ｗ）及びＤＶＤが含まれる。

【００６８】

以上、本発明を実施の形態を用いて説明したが、本発明の技術的範囲は上記実施の形態には限定されない。本発明の精神及び範囲から逸脱することなく様々に変更したり代替態様を採用したりすることが可能なことは、当業者に明らかである。

【符号の説明】

【００６９】

１０…前処理装置、１１…区分分割部、１２…キー選択部、１３…ソートキー付加部、１４…ソート処理部、２０…一括処理装置、３０…入力装置、４０…ＤＢ、５０…記憶装置、６０…入力ファイル、７０…ソートキー付き入力ファイル、８０…ソート済み入力ファイル

【図１】

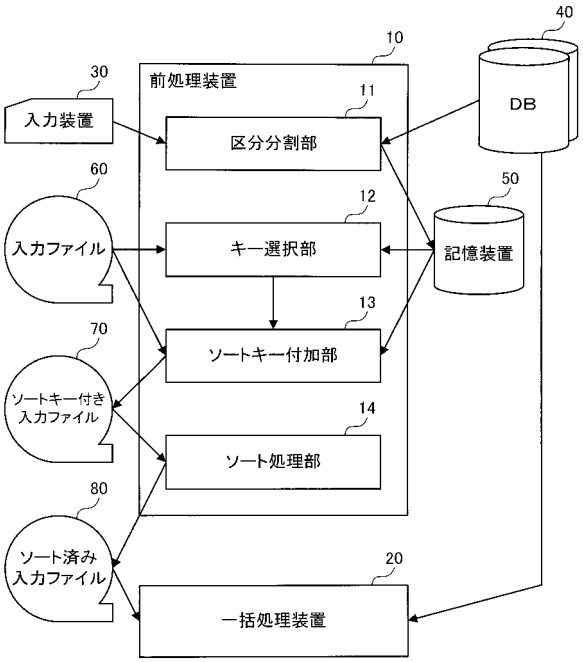
（ａ）

キー値のレンジ	区分値
0000~0999	0
1000~1999	1
……	……
9000~9999	9

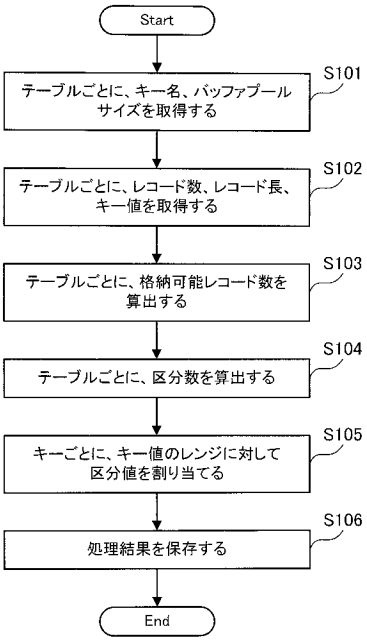
（ｂ）

ブロック番号		
000	019	028
001	010	029
002	011	020
003	012	021
004	013	022
005	014	023
006	015	024
007	016	025
008	017	026
009	018	027

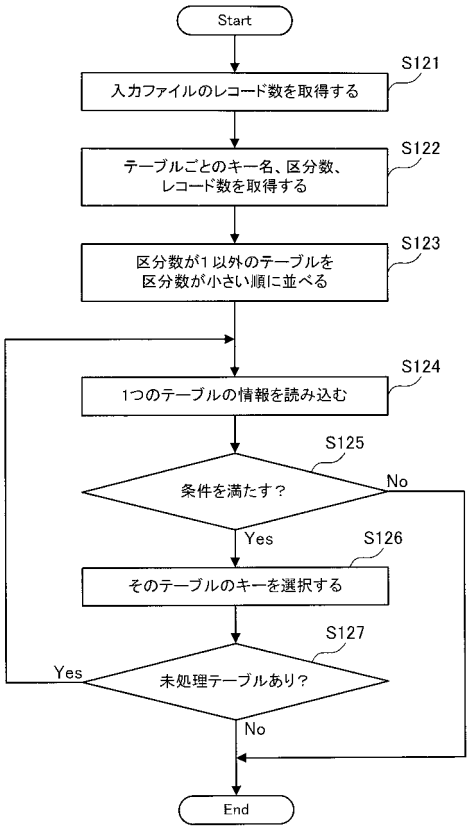
【図２】



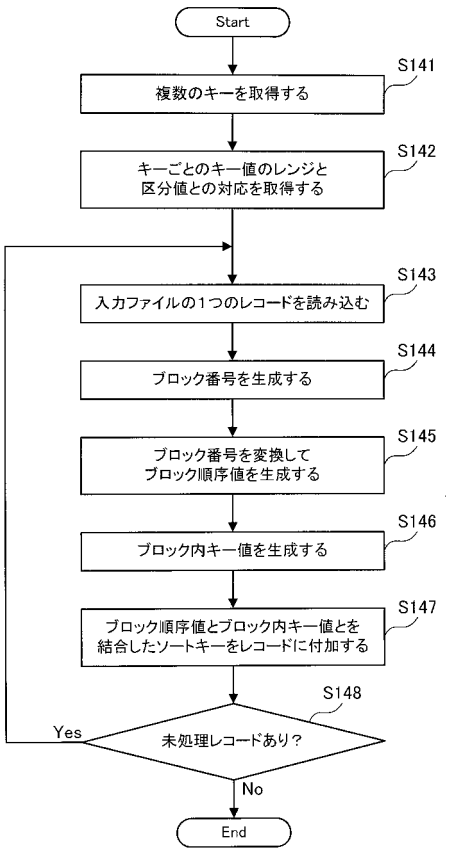
【図３】



【図４】



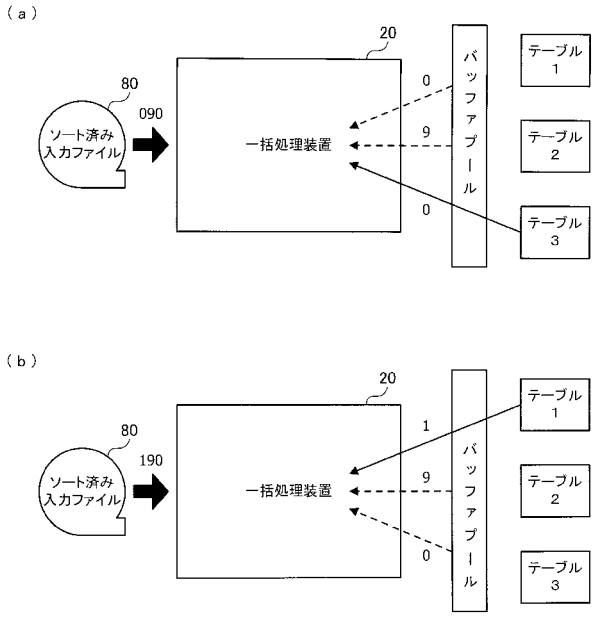
【図５】



【図６】

ソートキー		レコードデータ			
ブロック順序値	ブロック内キー値	キー1	キー2	キー3	……
373	6860	3918	4085	6860	……
298	9804	2618	7322	9804	……
340	1211	3929	1211	6480	……
464	8033	4665	2065	8033	……
125	3143	1155	1862	3143	……
518	7785	5951	6974	7785	……
231	8465	2882	1149	8465	……
404	4608	4030	6816	4608	……
400	4858	4858	6557	669	……
011	852	122	1777	852	……
……	……	……	……	……	……

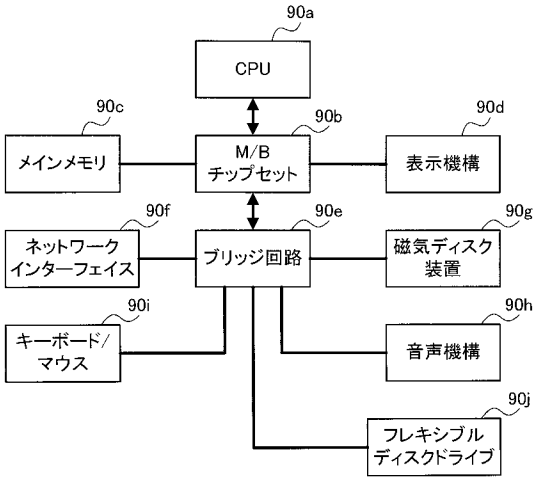
【図７】



【図８】

ブロック番号	キー1	キー2	キー3	ソートキー	テーブル1へのアクセス	テーブル2へのアクセス	テーブル3へのアクセス
000	0000~0999	0000~0999	0000~0999	キー3	ランダム（低速）	ランダム（低速）	シーケンシャル（高速）
001	0000~0999	0000~0999	1000~1999	キー3	バッファヒット（高速）	バッファヒット（高速）	シーケンシャル（高速）
002	0000~0999	0000~0999	2000~2999	キー3	バッファヒット（高速）	バッファヒット（高速）	シーケンシャル（高速）
…	…	…	…	…	…	…	…
009	0000~0999	0000~0999	9000~9999	キー3	バッファヒット（高速）	バッファヒット（高速）	シーケンシャル（高速）
019	0000~0999	1000~1999	9000~9999	キー2	バッファヒット（高速）	シーケンシャル（高速）	バッファヒット（高速）
010	0000~0999	1000~1999	0000~0999	キー3	バッファヒット（高速）	バッファヒット（高速）	シーケンシャル（高速）
011	0000~0999	1000~1999	1000~1999	キー3	バッファヒット（高速）	バッファヒット（高速）	シーケンシャル（高速）
…	…	…	…	…	…	…	…
099	0000~0999	9000~9999	9000~9999	キー3	バッファヒット（高速）	バッファヒット（高速）	シーケンシャル（高速）
090	0000~0999	9000~9999	0000~0999	キー3	バッファヒット（高速）	バッファヒット（高速）	シーケンシャル（高速）
190	1000~1999	9000~9999	0000~0999	キー1	シーケンシャル（高速）	バッファヒット（高速）	バッファヒット（高速）
191	1000~1999	9000~9999	1000~1999	キー3	バッファヒット（高速）	バッファヒット（高速）	シーケンシャル（高速）
…	…	…	…	…	…	…	…

【図９】



フロントページの続き

(72)発明者　河野　紀昭
東京都中央区日本橋箱崎町１９番地２１　日本アイ・ビー・エム株式会社内
(72)発明者　房　律子
東京都中央区日本橋箱崎町１９番地２１　日本アイ・ビー・エム株式会社内

審査官　田川　泰宏

(56)参考文献　特開２００６－０３１６３１（ＪＰ，Ａ）
特開平１１－００３２６０（ＪＰ，Ａ）
特開２００８－１６５６２２（ＪＰ，Ａ）
特開２００３－０８８１９９（ＪＰ，Ａ）
特開２００３－１５０４１４（ＪＰ，Ａ）
米国特許出願公開第２００８／０１０４１０２（ＵＳ，Ａ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１２／００
Ｇ０６Ｆ　１７／３０