

(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
2024年8月2日(02.08.2024)



(10) 国際公開番号
WO 2024/157776 A1

(51) 国際特許分類:
G06F 16/22 (2019.01)

(21) 国際出願番号: PCT/JP2024/000399

(22) 国際出願日: 2024年1月11日(11.01.2024)

(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 2023-010716 2023年1月27日(27.01.2023) JP
特願 2023-010717 2023年1月27日(27.01.2023) JP

(72) 発明者; および

(71) 出願人: 古庄 晋二(**FURUSHO, Shinji**) [JP/JP];
〒2210005 神奈川県横浜市神奈川区松見町
4-1101-7 Kanagawa (JP).

(74) 代理人: 伊東 忠重, 外(**ITOH, Tadashige et al.**);
〒1000005 東京都千代田区丸の内二丁目1

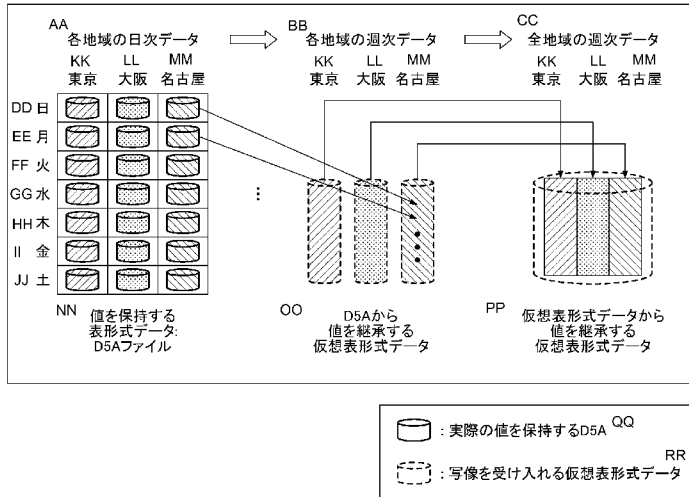
番1号丸の内 M Y P L A Z A (明治安
田生命ビル) 16階 Tokyo (JP).

(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, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, MG, MK, MN, MU, 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, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, 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, CV, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SC, SD, SL, ST, SZ, TZ, UG,

(54) **Title:** DATA PROCESSING DEVICE, DATA PROCESSING METHOD, AND PROGRAM

(54) 発明の名称: データ処理装置、データ処理方法、及びプログラム



AA Daily data for each region
BB Weekly data for each region
CC Weekly data for all regions
DD Sunday
EE Monday
FF Tuesday
GG Wednesday
HH Thursday
II Friday
JJ Saturday
KK Tokyo
LL Osaka
MM Nagoya
NN Tabular data that holds values; D5A file
OO Virtual tabular data that inherits values from D5A
PP Virtual tabular data that inherits values from virtual tabular data
QQ D5A holding actual value
RR Virtual tabular data that receives mapping

(57) **Abstract:** A data processing device according to one aspect of the present disclosure comprises a data operation unit that performs a sort operation, a search operation, or an aggregation operation on virtual tabular data composed of virtual columns having a second data structure obtained by converting a first data structure of one or more columns included in one or more pieces of tabular data by using predetermined mapping, by using the second data structure as an index.

(57) 要約: 本開示の一態様によるデータ処理装置は、1以上の表形式データに含まれる1以上のカラムが持つ第1のデータ構造を所定の写像により変換した第2のデータ構造を持つ仮想カラムで構成される仮想表形式データに対して、前記第2のデータ構造をインデックスとして利用したソート操作、検索操作又は集計操作を行うデータ操作部、を有する。

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, ME, 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：

一 国際調査報告（条約第21条(3)）

明 細 書

発明の名称：データ処理装置、データ処理方法、及びプログラム
技術分野

[0001] 本開示は、データ処理装置、データ処理方法、及びプログラムに関する。

背景技術

[0002] 近年、各種センサ機器や観測機器等の発達により、センシング結果や観測結果等を表す大量のデータ（いわゆる、ビッグデータ）が格納された表形式データが得られるようになった。このため、1つ以上の表形式データから複数のカラムを選択して、自身の利用目的に沿った仮想的な表形式データ（以下、仮想表形式データという。）を作成したいというニーズが増えている。このようなニーズを達成するための1つの方法は、仮想データベース又はデータ仮想化と呼ばれる技術を利用することである（例えば、非特許文献1参照）。これは、利用者からクエリを受け付けると、その背後に分散配置されているデータベースに対してサブクエリを実行するという技術である。

先行技術文献

非特許文献

[0003] 非特許文献1：「データハブ vs データレイク vs データの仮想化 - MarkLogic」，インターネット<URL：<https://jp.marklogic.com/product/comparisons/data-hub-vs-data-lake/>>

発明の概要

発明が解決しようとする課題

[0004] しかしながら、仮想データベース又はデータ仮想化と呼ばれる技術では、実データベースのソート、検索、集計を実現するインデックスを仮想データベースに継承させることができない。

[0005] 本開示は、上記の点に鑑みてなされたもので、実表形式データのソート、検索、集計を実現するインデックスを仮想表形式データに継承できる技術を提供する。

課題を解決するための手段

[0006] 本開示の一態様によるデータ処理装置は、1以上の表形式データに含まれる1以上のカラムを持つ第1のデータ構造を所定の写像により変換した第2のデータ構造を持つ仮想カラムで構成される仮想表形式データに対して、前記第2のデータ構造をインデックスとして利用したソート操作、検索操作又は集計操作を行うデータ操作部、を有する。

発明の効果

[0007] 実表形式データのソート、検索、集計を実現するインデックスを仮想表形式データに継承できる技術が提供される。

図面の簡単な説明

[0008] [図1]値を保持するD 5 Aを起点として階層的に構築される仮想表形式データの一例を説明するための図である。

[図2] A写像の一例（その1）を説明するための図である。

[図3] A写像の一例（その2）を説明するための図である。

[図4] D 5 Aの一例を説明するための図である。

[図5] 割り付けの一例を説明するための図である。

[図6] 列挙型写像による割り付け方法の一例を説明するための図である。

[図7] 1次関数型写像による割り付け方法の一例を説明するための図である。

[図8] ソースカラムと仮想カラムの一例を示す図である。

[図9] ソースカラムと転置構造体と仮想転置構造体の一例を示す図である。

[図10] 2階層の仮想表形式データの割り付け写像の一例を説明するための図である。

[図11] 2階層の仮想表形式データの仮想転置構造体の一例を説明するための図である。

[図12] 本実施形態に係るデータ処理装置を含むシステムの全体構成の一例を示す図である。

[図13] 本実施形態に係るデータ処理装置のハードウェア構成の一例を示す図である。

[図14]仮想表形式データの作成処理の流れの一例を示すフローチャートである。

[図15]仮想表形式データにおけるソート処理の流れの一例を示すフローチャートである。

[図16]仮想表形式データにおける検索処理の流れの一例を示すフローチャートである。

[図17]仮想表形式データにおける集計処理の流れの一例を示すフローチャートである。

発明を実施するための形態

[0009] 以下、本発明の一実施形態について説明する。以下の実施形態では、まず、必要な説明及び定義を行った後、表形式データを表現するD 5 Aと呼ぶデータを定義する。次に、D 5 A又は他の仮想表形式データから新たな仮想表形式データを作成する方法と、（実）表形式データのインデックスを継承して仮想表形式データに対してソート、検索、集計を行う方法とを説明する。また、このとき、仮想表形式データが階層的に構築された場合でもそのインデックスを継承できることについても説明する。そして、最後に、このような仮想表形式データの作成とソート、検索、集計を行うことができるデータ処理装置10について説明する。

[0010] <はじめに>

I o T (Internet of Things) データ、様々な観測データ、ログデータ等のアーカイブデータの蓄積が日々進行している。これらの多くでは表形式データが日次や月次のように一定期間毎に1つにまとめられてアーカイブに追加されていくことが多い。アーカイブに追加された表形式データはR e a d O n l yと考えてよい。このようなアーカイブの表形式データは巨大になることがあるし、しばしばL A N (Local Area Network) 上及びインターネット上に分散している。

[0011] 上記のアーカイブの表形式データを活用しようとする、一般に、以下の2つのステップを必要とし、いずれのステップでも長い時間が掛かり、また

大きい記憶領域を消費することが多い、という問題がある。

- [0012] 第1ステップは、新たな表形式データを生成するステップである。新たな表形式データは複数の表形式データをUNIONしたりJOINしたり必要なカラムのみを抽出したりして作成される。もし元になる表形式データが巨大であったりインターネットのような広域ネットワーク上に分散していたりすると長い時間が掛かる。また、新たな表形式データが大きいとその収容に大きい記憶領域が必要になる。
- [0013] 第2ステップは、ソートしたり検索したり集計したりするステップである。新たに作成された表形式データにはインデックスが付いておらず、ソート、検索、集計に時間が掛かる。それに加えて、新たな表形式データが大きい場合のソート結果、ヒット件数が大きい場合の検索結果、大きい集計結果はいずれも大きい記憶領域を必要とする。
- [0014] 上記の2つのステップの問題は今日進行するアーカイブデータの巨大化と分散したアーカイブデータの利用ニーズの高まりにつれて日々深刻化している。
- [0015] そこで、上記の2つのステップの操作に要する時間を対話型での操作が成り立つ時間に短縮し、かつ、必要な記憶領域がわずかで済む技術を提案する。その操作対象の表形式データは、例えば、1兆レコードであってもよいし、10万カラムであってもよい。また、その操作対象の表形式データのUNIONやJOIN等の組み合わせを階層的に積み上げてよい。更に、その操作対象の表形式データはLAN上に分散していてもよいし、インターネット上の複数のHTTP (Hypertext Transfer Protocol) サーバ上に分散していてもよい。
- [0016] 上記が可能になるのは、D5Aというすべてのカラムにソート・検索・集計を高速化するインデックスが付いた表形式データのファイルフォーマットと、D5Aファイルから直接又は間接的に値を継承する仮想表形式データと、D5Aファイルのインデックスが使うデータ構造を直接又は間接的に継承して自動的に成立するデータ構造を使って成り立つ仮想表形式データ上の仮

想インデックスとを用いてアーカイブデータの写像のネットワークシステムが構築できるからである。仮想表形式データと仮想インデックスは直接又は間接的にD 5 Aと接続するだけで直ちに利用可能になり、それらはわずかな記憶領域しか消費しない。それに加えて、仮想インデックスを使って得られたソート結果、検索結果、集計結果はそれらがどんなに大きいものであってもわずかな記録領域しか必要としない。

[0017] このようなD 5 Aを用いたアーカイブデータの写像のネットワークは、「利用者が利用目的毎にそれに沿った仮想表形式データを作ってネットワーク上に分散したアーカイブデータを対話型で使う」という新しいアーカイブデータの利用法を可能にする。例えば、各組織においては組織内の各部門に分散するアーカイブデータを様々な目的で利用することを可能にし、インターネット上においては世界各地のIoTデータ等のアーカイブデータを組み合わせたり抽出したりして様々な関係して利用することを可能にする。

[0018] ここで、本明細書中で使用する用語を整理する。まず、D 5 Aとは、すべてのカラムにソート・検索・集計を高速化するインデックスが付いた表形式データのファイルフォーマットのことである。次に、仮想表形式データとは、D 5 Aファイル（D 5 Aフォーマットのファイル）又は他の仮想表形式データから値を継承する表形式データのことである。なお、その継承元の表形式データをソース表形式データと呼ぶ。また、D 5 A上のカラムをD 5 Aカラム、D 5 Aカラムのソート・検索・集計を高速化するインデックスをD 5 Aインデックスと呼ぶ。仮想表形式データでも同様に、仮想カラム、仮想インデックスと呼ぶ。ソース表形式データでも同様に、ソースカラム、ソースインデックスと呼ぶ。また、D 5 Aインデックスが使用するデータ構造を転置構造体、仮想インデックスが使用するデータ構造を仮想転置構造体、ソースインデックスが使用するデータ構造をソース転置構造体と呼ぶ。なお、仮想表形式データは「写像表形式データ」等と呼んでもよく、同様に仮想カラムは「写像カラム」等と呼んでもよい。

[0019] <問題の解決方法>

上述したように、アーカイブの表形式データを活用しようとする、一般に、第1ステップと第2ステップの2つのステップを必要とし、いずれのステップでも長い時間が掛かり、また大きい記憶領域を消費することが多い、という問題がある。なお、第1ステップはUNIONしたりJOINしたり必要なカラムのみを抽出したりして新たな表形式データを生成するステップ、第2ステップはソートしたり検索したり集計したりするステップである。

[0020] 第1ステップに時間が掛かるのは、値を読み出し比較し格納する時間が掛かるためである。そこで、その代わりに、ソース表形式データとなる表形式データが存在し、そのどのセルが、新たに作成される表形式データのどのセルに写されるかという対応表とルールで定義される写像を用いて、その写像で新たな表形式データを作成するというアプローチが考えられる。アーカイブデータの場合、その写像の定義は可能かつコンパクトにまとまるケースが少なくない。その場合、新たな表形式データは値を持つ必要が無く、特に新たな表形式データが巨大なときはこのアドバンテージは大きい。新たな表形式データは写像の定義をロードするわずかな時間だけで表示可能になり、必要とする記憶領域も写像の定義を保持するためのわずかなもので済むためである。この新たな表形式データは値を保持せず、元となる表形式データから値を継承することから、これは仮想表形式データである。第1ステップの問題は、このような仮想表形式データで解決できる。

[0021] 第2ステップでは、新たに作成された表形式データのカラムにインデックスが付いていないことと、しばしば巨大になるソート結果・検索結果・集計結果を書き出すのに時間が掛かること、の2つで時間が掛かる。しかし、上記の仮想表形式データのすべてのカラムにはそのカラムのソート・検索・集計を高速化する仮想インデックスが自動的に存在することと、その仮想インデックスが巨大なソート結果・検索結果・集計結果でもわずかな記憶領域を使うだけで保持できるため書き出し時間を短縮できること、の2つにより時間が掛かる問題は解決される。また、仮想インデックスがそれらの結果の収容にわずかな記憶領域しか使わないことで記憶領域の問題も解決される。

このようにして第2ステップの問題は仮想インデックスで解決できる。

[0022] このような仮想表形式データの元になる値と、仮想インデックスが用いるデータ構造の元になるデータ構造とを保持するのがD 5 Aという表形式データの格納フォーマットで表現されたD 5 Aファイルである。

[0023] ≪D 5 A：値とインデックスの供給源となる表形式データの格納フォーマット≫

仮想表形式データは、1つ以上のソース表形式データから値を継承する。ソース表形式データは、別の仮想表形式データ、又は、D 5 Aファイルと呼ばれる表形式データ、のいずれかである。前者の場合、その仮想表形式データは更に別のソース表形式データから値を継承し最終的にD 5 Aファイルに辿り着く。したがって、仮想表形式データは直接的又は間接的にD 5 Aファイルから値を継承して階層的に成り立っているといえる。

[0024] 仮想インデックスは仮想転置構造体を使ってソート・検索・集計を高速化する。その仮想転置構造体は、1つ以上のソース転置構造体を継承して自動的に成立する。ソース転置構造体とは、別の仮想表形式データ上の仮想転置構造体、又は、D 5 Aファイル上の転置構造体、のいずれかである。前者の場合、その仮想転置構造体は更に別のソース転置構造体を継承し最終的にD 5 Aファイル上の転置構造体に辿り着く。したがって、仮想転置構造体は直接的又は間接的にD 5 Aファイル上の転置構造体を継承して階層的に成り立っているといえる。

[0025] D 5 Aは、仮想表形式データへ値を供給し、仮想転置構造体へ転置構造体を供給するための表形式データの格納フォーマットであり、値を保持しそのすべてのカラムに転置構造体を使うD 5 Aインデックスを備える。

[0026] ≪アーカイブデータの写像のネットワークシステムの実現≫

アーカイブデータの写像のネットワークシステムの簡単な想定例を示し、ここまでに説明したD 5 A、仮想表形式データ、仮想インデックスの使用方法について、図1を参照しながら説明する。図1は、東京、大阪、名古屋の各地域の日曜日から土曜日の気象観測データを1つにまとめる処理を示して

いる。この処理は２段階で行われ、まず地域毎に７つの日次データをまとめて週次データにし、次にそれらの週次データを並置して１つの表形式データにまとめている。ここで、図１の左側にある３地域１週間分の２１個の日次データは５５Ａファイルに收容されている。図１の中央にある地域毎の３つの表形式データは仮想表形式データである。これらの仮想表形式データはそれぞれ図１の左側にある１週間分の７つのＤ５Ａファイルをソース表形式データとして、それらから必要なカラムを抽出した上でＵＮＩＯＮしている。図１の右側にある１つの仮想表形式データは、図１の中央にある地域毎の３つの仮想表形式データをソース表形式データとして、それらを並置している。

[0027] 図１に示すように、アーカイブデータの写像のネットワークは、Ｄ５Ａファイルを起点として、その上に仮想表形式データを階層的に構築したものである。既に述べたように、仮想表形式データには仮想インデックスが自動的に備わり、任意のカラムを短時間でソート・検索・集計でき、ソート結果・検索結果・集計結果を記憶するための記憶領域もわずかで済む。このような仮想表形式データは利用者の手元で作成し利用することができ、アーカイブデータの広汎な利用を促進することが期待できる。

[0028] <Ａ写像>

Ｄ５Ａ、仮想表形式データや仮想インデックスはすべて０から始まる自然数の連続区間から何等かの値（これも多くの場合は自然数）への写像の組み合わせで記述される。この写像は添字（インデックス）が０から始まる１次元配列（Ａｒｒａｙ）で表せるので、これをＡ写像と呼ぶ。Ａ写像を用いると対応関係を全体から眺めることが可能になり、その結果、集合や群としての性質を利用したアルゴリズムを導くことが可能になる。以下にＡ写像の利点を説明する。

[0029] Ａ写像の第１の利点は、Ａ写像を様々に結合して新たなＡ写像を作り出せることである。簡単な例を挙げると検索の結果列を表すＡ写像Ｓを、カラムＡを表すＡ写像 C_A と結合すればカラムＡの検索結果を表すＡ写像が得られ、

カラム B を表す A 写像 C_B と結合すればカラム B の検索結果を表す A 写像を得ることができる。このとき、S は結果列を表す A 写像であれば何でもよく、 C_A 及び C_B もカラムを表す A 写像であれば何でもよい。そして、それらはローカルな記憶装置上に存在しても、ネットワーク上に存在してもよい。このような A 写像の結合は代数的な結合であるといえる。

[0030] そして、A 写像と A 写像の結合結果は必ずしも記憶領域に書き出される必要は無く、仮想的な A 写像でもよい。仮想的な A 写像とは 1 以上の A 写像を結合元にして、全体のサイズが分かり、任意の i 番目の部分だけを結合元から取り出すことができる A 写像である。仮想的な A 写像も A 写像であるため、仮想的な A 写像を結合して更なる仮想的な A 写像を階層的に作ることができる。仮想表形式データのカラムは仮想的な A 写像の一種であり、階層的に作ることができる。インデックスは、インデックスのためのデータ構造とそのデータ構造を使うアルゴリズムで実現される仕組みである。仮想インデックスは 1 つ以上の仮想的な A 写像をインデックスのためのデータ構造として使うインデックスであり、階層的に作ることができる。

[0031] A 写像の第 2 の利点は、A 写像を分解して新たな A 写像を作り出せることである。A 写像を分解することにより、分解前の A 写像には無い機能を持った新たな A 写像を作り出して使うことが可能になる。特に有効な分解の一つは、表形式データ間でセルとセルの対応（マッピング）を定める A 写像 M の LP 分解（A 写像 L と A 写像 P への分解）である。M を L と P に分解すると、L でバイセクションサーチによる効率の良い要素の検索が可能になり、P で逆写像を求めることが可能になる。このとき、M はマッピングを定める A 写像であれば何でもよく、L は自動的にバイセクションサーチが可能な A 写像になり、P は自動的に逆元を持つ A 写像になる。A 写像の分解は新たな A 写像を生み出すため、代数的な分解であるといえる。

[0032] 一方で、A 写像が 1 次元の配列であるため、大きい A 写像への要素の挿入と要素の削除には時間が掛かる。しかし、アーカイブデータの場合は更新がほとんど無いため、このデメリットは問題にはならない。

[0033] このようなA写像を用いると、高速にソート・検索・集計ができるインデックスをどのカラムにも等しく備える表形式データの格納フォーマットであるD5Aを設計することができる。そして、D5AとA写像を使って仮想表形式データを定義することができる。その仮想表形式データには1つ以上のA写像をインデックスのためのデータ構造として使う仮想インデックスが自動的に備わる。すると、ネットワーク上に分散するアーカイブデータを様々な組み合わせで使える、アーカイブデータの写像のネットワークシステムを実現できる。

[0034] そこで、以下では、まずA写像の定義を行い、その表記法を定める。次に、A写像を結合するための演算子であるインデックス演算子を導入する。次に、A写像の分解について述べ、特に重要なSN分解、LP分解、スペクトル分解を説明する。最後に、A写像を4つの側面で分類する。

[0035] ≪A写像の定義≫

インデックスが0から始まり、サイズNの1次元配列を考える。その1次元配列は定義域が $0 \sim N-1$ の自然数の連続区間、値域が最大N種類の離散区間である写像と見ることができる。これをA写像と呼ぶ。レコードに0から始まる連続するレコード番号が振られていれば、表形式データのカラムもレコード番号を定義域としたA写像とみなすことができる。

[0036] 例えば、図2に示すように、0番目の要素として「4」、1番目の要素として「0」、2番目の要素として「6」、3番目の要素として「3」が格納されている1次元配列を考える。この1次元配列は0を「4」、1を「0」、2を「6」、3を「3」にそれぞれ対応付けたものであるため、定義域が $\{0, 1, 2, 3\}$ 、値域が $\{0, 3, 4, 6\}$ のA写像とみなすことができる。

[0037] また、例えば、図3に示すように、0番目の要素として「Bob」、1番目の要素として「Alice」、2番目の要素として「Cathy」、3番目の要素として「Bob」が格納されている1次元配列を考える。この1次元配列は0を「Bob」、1を「Alice」、2を「Cathy」、3を

「Bob」にそれぞれ対応付けたものであるため、定義域が $\{0, 1, 2, 3\}$ 、値域が $\{Alice, Bob, Cathy\}$ のA写像とみなすことができる。

[0038] なお、A写像は1次元配列で表せるが、その写像としての側面を議論するときは写像といい、その操作を議論するときは配列という。それに伴い、写像として扱うときは定義域、値域といい、カラムや表形式データを含む配列のときはレコード番号、値ということにする。ただし、A写像の表記法は可能な限り配列としての表記法を用いる。

[0039] ≪A写像の表記法≫

以下に一般的な配列の表記法を取り入れたA写像の表記を記載する。

[0040] 表記法1. A写像を要素の列挙で定義する場合は $(a_0, a_1, \dots, a_{n-1})$ と記載する。

[0041] 表記法2. A写像Aの定義域nを明示する場合は $A_{(n)}$ と記載する。この表記法に従い、カラムは、例えば、Rを全レコード数として $C_{(R)}$ 等と記載する。

[0042] 表記法3. A写像Aの値域が自然数上にあり、その値域の最大値が $n-1$ を超えないことがわかっているとき、値域を明示するために $A^{(n)}$ と記載する。

[0043] 表記法4. A写像 $A_{(n)}$ のi番目の要素を $A_{(n)}[i]$ と記載する。したがって、 $A_{(n)} \equiv (A_{(n)}[0], A_{(n)}[1], \dots, A_{(n)}[n-1])$ である。

[0044] 表記法5. $(i_0, i_1, \dots), (j_0, j_1, \dots), \dots$ の連結を $(i_0, i_1, \dots) + (j_0, j_1, \dots) + \dots$ と記載する。

[0045] 表記法6. A写像の値域上の λ が定義域上の $i_0, i_1, \dots$ からの写像を受け入れていることを $\lambda : (i_0, i_1, \dots)$ と記載する。例えば、図3に示す例は、 $Alice : (1), Bob : (0, 3), Cathy : (2)$ と記載できる。

[0046] 表記法7. $\lambda_0 : (i_0, i_1, \dots), \lambda_1 : (j_0, j_1, \dots), \dots$

・ ・ を連結してひとまとめにするときは記号+を使用して、 $\lambda_0 : (i_0, i_1, \dots) + \lambda_1 : (j_0, j_1, \dots) + \dots$ と表記する。ただし、このとき、 $\lambda_0 < \lambda_1 < \dots$ の順に並べる。例えば、図3に示す例は、 $Alice : (1) + Bob : (0, 3) + Cathy : (2)$ と記載できる。

[0047] 《A写像を結合するインデックス演算子》

A写像の結合を行って新たなA写像を作り出すインデックス演算子 (Index Operator) と呼ぶ演算子・を以下で定義する。

$$[0048] \quad A_{(n)} \cdot B_{(m)}^{(n)} = (A_{(n)} [B_{(m)}^{(n)} [0]]), \dots, A_{(n)} [B_{(m)}^{(n)} [m-1]]) \quad \text{式 (1)}$$

上記のインデックス演算子には、以下の性質1～性質3に示す性質がある。

[0049] 性質1：結合則が成り立つ。すなわち、 $(A_{(n)} \cdot B_{(m)}) \cdot C_{(k)} = A_{(n)} \cdot (B_{(m)} \cdot C_{(k)})$ が成り立つ。

[0050] 性質2：結合結果を表す配列のサイズは右側の配列のサイズである。

[0051] 性質3：結合結果を表す配列のデータ型は左側の配列のデータ型である。

[0052] ここで、例を使って結合則を確かめる。 $A_{(3)} = (Alice, Bob, Cathy)$ 、 $B_{(4)} = (1, 0, 2, 1)$ 、 $C_{(3)} = (0, 2, 1)$ とする。このとき、 $(A_{(3)} \cdot B_{(4)}) \cdot C_{(3)} = (Bob, Alice, Cathy, Bob) \cdot (0, 2, 1) = (Bob, Cathy, Alice)$ となる。一方で、 $A_{(3)} \cdot (B_{(4)} \cdot C_{(3)}) = (Alice, Bob, Cathy) \cdot (1, 2, 0) = (Bob, Cathy, Alice)$ となり、両者は一致する。

[0053] 上記の性質1及び2により、インデックス演算子によるA写像の結合の結果として得られたA写像に出現し得る値の種類数は、インデックス演算子が結合したA写像中の最小のA写像のサイズ以下になる。これは行列におけるrankに似た指標となる。

[0054] 《A写像の分解》

インデックス演算子を用いてA写像を結合すると、ただ1つのA写像が得

られる。一方で、1つのA写像は様々なA写像に分解できる。例えば、 $(Bob, Alice, Cathy, Bob)$ は、 $(Bob, Alice, Cathy, Bob) = (Alice, Bob, Cathy) \cdot (1, 0, 2, 1)$ と分解することもできるし、 $(Bob, Alice, Cathy, Bob) = (Bob, Alice, Cathy) \cdot (0, 1, 2, 0)$ と分解することもできる。

[0055] このような中で一意の分解方法は特別な価値があり、ソートや検索を高速化したり、逆写像を求めたりすることが可能になる。そこで、以下では、その一意の分解方法であるSN分解、LP分解、スペクトル分解を説明する。

[0056] ・SN分解

1つのA写像を2つに分解する方法で、得られる2つのA写像のうちの左側のA写像が左辺のA写像に出現する値だけを保持し、それらの値をユニークかつ昇順に保持する場合は1通りしか存在しない。この分解をSN分解と呼ぶ。

[0057] 例えば、 $(Bob, Alice, Cathy, Bob) = (Alice, Bob, Cathy) \cdot (1, 0, 2, 1)$ はSN分解の例である。

[0058] 右辺のユニークかつ昇順である第1項をSVL (Sorted Value List) と呼ぶ。上記の例では $SVL = (Alice, Bob, Cathy)$ である。右辺の第2項は左辺の各要素をSVL中の格納位置で置き換えたものであると考えることができる。これをNNC (Natural Numbered Column) と呼ぶ。上記の例では $NNC = (1, 0, 2, 1)$ である。

[0059] このようなSN分解を一般的に定義すると以下になる。

[0060] $C_{(R)} = SVL_{(K)} \cdot NNC_{(R)}^{(K)}$;
 $i, j \in 0, \dots, K-1$ かつ $i < j \Rightarrow SVL_{(K)}[i] < SVL_{(K)}[j]$

式(2)

式(2)では上段でカラム $C_{(R)}$ が $SVL_{(K)}$ と $NNC_{(R)}^{(K)}$ に分解され

ることを表しており、下段で $SVL_{(K)}$ の要素は昇順に並ぶことを表している。なお、 $NNC_{(R)}^{(K)}$ については、その表記から $0, \dots, K-1$ の自然数を要素とするサイズ R の A 写像であることがわかる。

[0061] SN 分解を使うと、自然数を要素とするとは限らない A 写像から自然数を要素とする A 写像 NNC を取り出すことができる。取り出した NNC を使うと、カウンティングソート等の自然数の性質を使った効率の良いアルゴリズムが利用可能になる。

[0062] (SN 分解のアルゴリズム)

ここで、 $(Bob, Alice, Cathy, Bob)$ を例に SN 分解のアルゴリズムの一例を述べる。 SN 分解のアルゴリズムでは、以下の手順1-1～手順1-5が実行される。

[0063] 手順1-1: $(Bob, Alice, Cathy, Bob)$ の各値に A 写像内の位置を添えて、 $((Bob, 0), (Alice, 1), (Cathy, 2), (Bob, 3))$ を作成する。

[0064] 手順1-2: 値と位置の組で大小関係を評価しながらソートして、 $((Alice, 1), (Bob, 0), (Bob, 3), (Cathy, 2))$ を得る。

[0065] 手順1-3: 各値に対して、その値の小さい順に0から順番に値番号を付与して、ソート結果 $((Alice=0, 1), (Bob=1, 0), (Bob=1, 3), (Cathy=2, 2))$ を得る。

[0066] なお、上記の例では同じ値 Bob が2回出現しているが、いずれの Bob にも値番号1が付与されることに留意されたい。また、上記のソート結果は、 $((値=値番号, A写像内の位置), \dots)$ という構造である。例えば、上記の例のソート結果における第1要素 $(Alice=0, 1)$ では、値は $Alice$ 、値番号は0、 A 写像内の位置は1である。

[0067] 手順1-4: 上記のソート結果を使って SVL の格納領域(領域のサイズは3)を確保し、「 $SVL[値番号]=値$ 」として書き込む。すなわち、 $SVL[0]=Alice$ 、 $SVL[1]=Bob$ 、 $SVL[2]=Cathy$

yとして書き込む。これにより、 $SVL = (Alice, Bob, Cathy)$ が完成する。

[0068] 手順1－5：上記のソート結果を使ってNNCの格納領域（領域のサイズは4）を確保し、「NNC[A写像内の位置]＝値番号」として書き出す。すなわち、 $NNC[1] = 0$ 、 $NNC[0] = 1$ 、 $NNC[3] = 1$ 、 $NNC[2] = 2$ として書き込む。これにより、 $NNC = (1, 0, 2, 1)$ が完成する。

[0069] 以上により、 $(Bob, Alice, Cathy, Bob) = (Alice, Bob, Cathy) \cdot (1, 0, 2, 1)$ が得られる。

[0070] ・LP分解

LP分解は、重複のない自然数を要素とするA写像M (Mapping projection) を2つに分解するSN分解の特殊なケースである。例えば、 $(5, 2, 7, 3) = (2, 3, 5, 7) \cdot (2, 0, 3, 1)$ はLP分解の例である。

[0071] LP分解で得られる右辺の第1項は自然数を要素とし、ユニークかつ昇順になる。これをL (seLection) と呼ぶ。上記の例では $L = (2, 3, 5, 7)$ である。一方で、右辺の第2項は左辺の各要素をL中の格納位置で置き換えたものであると考えることができる。これをP (Permutation) と呼ぶ。上記の例では $P = (2, 0, 3, 1)$ である。LP分解ではM、L、Pは同じサイズになる。

[0072] LP分解を行うと、Lを用いてM中の要素の有無及び要素の順位を求め、Pを用いて逆元を求めることが可能になる。なお、LP分解はSN分解と同じアルゴリズムで実施できる。

[0073] ・スペクトル分解

上記の表記法7は各値にその値が出現するA写像上の位置（インデックス）を新たなA写像として表記するものである。これをA写像のスペクトル分解と呼ぶ。スペクトル分解により、A写像を値域から定義域に向かって見ることが可能になり、様々なアルゴリズムを可能にする。以下、A写像であるカラムのスペクトル分解によりカラムのソート・検索・集計が簡単にできる

ようになる例を示す。

[0074] (ソートの実行)

図3に示す例で、ソートは、カラムのスペクトル分解 $Alice : (1) + Bob : (0, 3) + Cathy : (2)$ から値を外して $(1) + (0, 3) + (2)$ とし、表記法5で定めた連結を行ってソート結果列 $(1, 0, 3, 2)$ を得ることができる。なお、ソート結果はカラム $C_{(R)} = (Bob, Alice, Cathy, Bob)$ とソート結果列とを以下のように結合して得ることができる。

[0075] ソート結果 $= C_{(R)} \cdot \text{ソート結果列} = (Bob, Alice, Cathy, Bob) \cdot (1, 0, 3, 2) = (Alice, Bob, Bob, Cathy)$

[0076] (検索の実行)

図3に示す例で、Bobの検索は、カラムのスペクトル分解 $Alice : (1) + Bob : (0, 3) + Cathy : (2)$ からバイセクションサーチでBobのパートを特定し、それを取り出して $Bob : (0, 3)$ を得て、検索結果列 $(0, 3)$ を得ることができる。なお、検索結果は、カラム $C_{(R)} = (Bob, Alice, Cathy, Bob)$ と検索結果列とを以下のように結合して得ることができる。

[0077] 検索結果 $= C_{(R)} \cdot \text{検索結果列} = (Bob, Alice, Cathy, Bob) \cdot (0, 3) = (Bob, Bob)$

[0078] (集計の実行)

図3に示す例で、集計は、カラムのスペクトル分解 $Alice : (1) + Bob : (0, 3) + Cathy : (2)$ から値とその出現回数を取り出し、集計結果 $= Alice : 1 \text{回}, Bob : 2 \text{回}, Cathy : 1 \text{回}$ を得ることができる。

[0079] (スペクトル分解の一般式)

ここで、スペクトル分解を一般的に書き直すと以下となる。

[0080]

[数1]

$$\begin{aligned} & \lambda_0 < \lambda_1 < \dots < \lambda_{K-1}; \sum_{i=0}^{K-1} w_i = R; \\ & \lambda_0: X_{0(w_0)}^{(R)} + \lambda_1: X_{1(w_1)}^{(R)} + \dots + \lambda_{K-1}: X_{K-1(w_{K-1})}^{(R)} \end{aligned} \quad \text{式(3)}$$

式(3)で、Rは全レコード数、 λ はキー値、wはキー値 λ に属する転置レコード番号の数、Xは転置レコード番号の配列である。

[0081] 例えば、図3に示す1次元配列のソート結果を式(3)の形で書く場合、以下となる。

[0082] $R = 4$, $\lambda_0 = \text{A l i c e}$, $\lambda_1 = \text{B o b}$, $\lambda_2 = \text{C a t h y}$

$w_0 = 1$, $w_1 = 2$, $w_2 = 1$

$X_{0(1)}^{(4)} = (1)$, $X_{1(2)}^{(4)} = (0, 3)$, $X_{2(1)}^{(4)} = (2)$

(スペクトル分解のアルゴリズム)

ここで、(B o b, A l i c e, C a t h y, B o b)を例にA写像をスペクトル分解するアルゴリズムの一例を示しておく。そのアルゴリズムは、以下の手順2-1～手順2-3からなる。

[0083] 手順2-1: (B o b, A l i c e, C a t h y, B o b)の各値にA写像内の位置を添えて、((B o b, 0), (A l i c e, 1), (C a t h y, 2), (B o b, 3))を作成する。

[0084] 手順2-2: 値と位置の組で大小関係进行评估しながらソートして、((A l i c e, 1), (B o b, 0), (B o b, 3), (C a t h y, 2))を得る。

[0085] 手順2-3: 同じ値毎にA写像内の位置をまとめるとスペクトル分解A l i c e: (1) + B o b: (0, 3) + C a t h y: (2)が完成する。

[0086] ≪A写像の分類≫

A写像は、結合性、検索性、集合を表しているかどうか、簡単に逆元が求まるか、という4つの指標で分類することができる。これらの分類を踏まえておくことは、これ以降の説明の理解に役立つため、ここで述べておく。

[0087] ・結合性

自然数を要素とする A 写像はインデックス演算子の両側におくことができる。そうでない A 写像はインデックス演算子の左側にしかおくことができない。

[0088] ・検索性

要素が昇順に並ぶ A 写像はバイセクションサーチを用いて特定の要素があるかどうか、ある場合はどこにあるかを効率的に検索することができる。このような A 写像を昇順配列と呼ぶ。要素がユニークかつ昇順である A 写像はある値で検索してマッチするものが存在する場合、それがただ一つになる。このような A 写像をユニーク昇順配列と呼ぶ。

[0089] ・集合を表しているかどうか

A 写像の要素に重複がなければそれは集合を表していると思えることができる。ただし、本明細書では要素が自然数であることも前提に加える。本明細書では、集合は結果列であり自然数（レコード番号）を要素とする A 写像であるためである。

[0090] ・簡単に逆元が求まるか

自然数を要素とする A 写像でそのサイズを N とするとき、その A 写像に $0 \sim N-1$ の値を持つ要素がすべて含まれる場合、インデックス演算子に関する逆元を簡単に求めることができる。これを対称配列と呼ぶ。対称配列 P の定義を以下に示す。

$$[0091] \quad i, j \in 0, 1, \dots, N-1; i \neq j \Rightarrow P_{(N)}^{(N)}[i] \neq P_{(N)}^{(N)}[j]$$

対称配列はインデックス演算子に関して群を作る。したがって、単位元と逆元が存在する。P の逆元は仮想表形式データのセルからソース表形式データのセルを特定する際に使用される。

$$[0092] \quad \text{単位元: } E = (0, 1, \dots, N-1)$$

$$\text{逆元: } P[i] = j \Rightarrow P^{-1}[j] = i$$

<D 5 A>

Rは表形式データのレコード数、Kはカラムの中に含まれる値の種類を指すものとする。このとき、表形式データとは、上から順に $0 \sim R - 1$ の番号で識別されるR個のレコードと、左から順に番号又は名前で識別される $1 \sim$ 複数のカラムとを持ち、1つのレコードと1つのカラムとで1つの値が定まり、カラム毎にそのカラムで保持されるデータのデータ型（例えば、整数、浮動小数点、文字列等）が1つに定まっているものをいう。また、インデックスが成立しているとは、対象カラムのソート・検索・集計結果のサイズWがわかり、 $i \in 0, 1, \dots, W - 1$ を指定するとおよそ $O(\log(R))$ 以下の時間でソート・検索・集計結果のi行目を読み出せる状態になっていることをいう。

[0093] 以下では、まずD5Aの内部構造について説明し、次に値を取り出す方法、次にD5Aインデックスについて関連事項を交えながら説明する。

[0094] ≪D5Aの内部構造≫

図4を参照しながらD5Aの内部構造を説明する。図4は、カラム $C_{(4)} = (Bob, Alice, Cathy, Bob)$ が含まれるCSV (Comma Separated Values) 形式の表形式データとそのD5Aフォーマットを示している。

[0095] D5Aは、同じ構造をしたD5AのカラムであるD5Aカラムが集まった構造をしている。各D5Aカラムは、以下に述べる2つの構造を並置したものである。

[0096] 第1の構造はカラムの値の取得のための構造であり、図4中の破線で囲まれた部分である。この構造は、(a) A写像であるカラムをそのまま格納するか、(b) カラムをSN分解して得たSVL、NNCを格納するか、(c) その両方を格納するか、のいずれかで構成される。そのまま格納した(a)の場合は値の読み出しのパフォーマンスが向上するが、NNCを使うアルゴリズムが使えなくなる。SVL、NNCを格納する(b)の場合はNNCを使うアルゴリズムが使えるが、値の読み出しのパフォーマンスが低下する。その両方を格納する(c)の場合はD5Aファイルのサイズが大きくなる。

。本実施形態では（b）のケースで説明を行う。

[0097] 第2の構造はD5Aインデックスのための構造であり、図4中の一点鎖線で囲まれた部分である。この構造はSVL、ACM、INVから成り、転置構造体と呼ばれる。転置構造体は、上述したD5Aカラムのスペクトル分解をA写像の形に書き換えたものである。なお、転置構造体の詳細については後述する。

[0098] 転置構造体（つまり、第2の構造）が得られれば上記の第1の構造と合わせてD5Aカラムとなり、D5Aカラムを集めるとD5Aとなる。

[0099] ≪カラムの値を取得するための構造≫

カラムの値を取得するための構造は、カラム $C_{(R)}$ をSN分解して作成することができる。例えば、図4に示す例の場合、 $C_{(4)} = (Bob, Alice, Cathy, Bob) = (Alice, Bob, Cathy) \cdot (1, 0, 2, 1)$ となる。したがって、 $SVL_{(3)} = (Alice, Bob, Cathy)$ 、 $NNC_{(4)}^{(3)} = (1, 0, 2, 1)$ となる。

[0100] 一方で、SVLとNNCを使ってカラムは以下の式で得られる。

[0101] $C_{(R)} = SVL_{(K)} \cdot NNC_{(R)}^{(K)}$ 式(4)

したがって、カラムのi行目は以下の式で得られる。

[0102] $value = C_{(R)}[i] = SVL_{(K)} \cdot NNC_{(R)}^{(K)}[i]$ 式(5)

例えば、図4に示す例の場合、カラムの1行目は $(Alice, Bob, Cathy) \cdot (1, 0, 2, 1)[1] = Alice$ となる。

[0103] また、SVLからも以下の情報1～情報4が得られる。

[0104] 情報1：カラム内の値の種類数Kが $SVL_{(K)}$ からわかる。例えば、図4に示す例の場合、Kは3となる。

[0105] 情報2：小さい方からi番目の値を $SVL[i]$ として読み出せる。

[0106] 情報3：ある値vがカラム内に存在するかどうかはSVLのバイセクションサーチでわかる。また、その値vが存在するとき、vが何番目の値であるかもわかる。

[0107] 情報4：SVLが集計の次元となる。なお、集計の次元を作るためには従来はソートを行う必要があった。

[0108] ≪転置構造体≫

次に、DSAインデックスのための構造である転置構造体の作成方法について説明する。カラムをスペクトル分解するとソート・検索・集計を高速に行えることを述べたが、スペクトル分解には2つの扱いにくい点がある。1つ目は、A写像でできていないため、インデックス演算子を用いてアクセスすることができない点（第1の問題点）である。2つ目は、ソート結果列のi番目を取得しようとする、 $w_0, w_1, \dots$ を加算しながらi番目を探す必要がある点（第2の問題点）である。スペクトル分解を3つのA写像に変換したものが転置構造体であり、この2点を解決する。以下、その定義と作成方法を説明する。

[0109] ・転置構造体の定義とスペクトル分解からの作成方法

転置構造体の定義を以下に示す。

[0110] [数2]

$$\begin{aligned}
 & \langle SVL_{(K)} : ACM_{(K)}^{(R+1)}, INV_{(R)}^{(R)} \rangle \\
 & = \langle (\lambda_0, \lambda_1, \dots, \lambda_{K-1}) : \left(w_0, w_0 + w_1, \dots, \sum_{i=0}^{K-1} w_i \right), \\
 & \quad (x_{0,0}, x_{0,1}, \dots, x_{0,n_0-1}) + \dots + (x_{K-1,0}, x_{K-1,1}, \dots, x_{K-1,n_{K-1}-1}) \rangle; \\
 & \quad \lambda_0 < \lambda_1 < \dots < \lambda_{K-1}; \sum_{i=0}^{K-1} w_i = R
 \end{aligned} \tag{6}$$

ここで、 $SVL_{(K)}$ 、 $ACM_{(K)}^{(R+1)}$ 、 $INV_{(R)}^{(R)}$ はそれぞれ以下の手順3-1～手順3-3で得られる

手順3-1： $SVL_{(K)}$ は、式(3)のスペクトル分解の値部分である $\lambda_0, \lambda_1, \dots, \lambda_{K-1}$ をそのまま並べてA写像にしたものである。また、式(4)で得られる $SVL_{(K)}$ と同じである。

[0111] 手順3-2: $ACM_{(K)}^{(R+1)}$ は、

[0112] [数3]

$$ACM_{(K)}^{(R+1)} = \left(w_0, w_0 + w_1, \dots, \sum_{i=0}^{K-1} w_i \right)$$

であり、式(3)の値 λ_i の出現回数 w_i を累計数化してA写像にしたものである。このステップには $O(K)$ の時間が掛かる。

[0113] 手順3-3: $INV_{(R)}^{(R)}$ は、

[0114] [数4]

$$INV_{(R)}^{(R)} = (x_{0,0}, x_{0,1}, \dots, x_{0,n_0-1}) + \dots + (x_{K-1,0}, x_{K-1,1}, \dots, x_{K-1,n_{K-1}-1})$$

であり、式(3)の

[0115] [数5]

$$X_{0(w_0)}^{(R)}, X_{1(w_1)}^{(R)}, \dots, X_{K-1(w_{K-1})}^{(R)}$$

を連結してA写像にしたものである。ここで、 $x_{i,j}$ は λ_i が出現するレコード番号のうち上からj番目のレコード番号である。

[0116] [数6]

$$X_{0(w_0)}^{(R)}, X_{1(w_1)}^{(R)}, \dots, X_{K-1(w_{K-1})}^{(R)}$$

が $\lambda_0 < \lambda_1 < \dots < \lambda_{K-1}$ に対応するレコード番号であることから INV は転置レコード番号の配列になる。

[0117] 例えば、図4に示す例の場合、転置構造体は $\langle (A l i c e, B o b, C a t h y) : (1, 3, 4), (1, 0, 3, 2) \rangle$ となる。

[0118] スペクトル分解は、A写像でなく、ソート結果列のi番目を取得しようとするときw_iの加算操作が必要であった。スペクトル分解から導出された転置構造体は3つのA写像からなる構造体で第1の問題点を解決でき、INV[i]でソート結果列のi番目を取得できるため第2の問題点も解決できる。

[0119] ・転置構造体からスペクトル分解を得る方法

以下の方法でD5A内の転置構造体からスペクトル分解を得ることができる。

[0120] まず、式(3)のスペクトル分解のうち、 λ_i は $\lambda_i = S V L_{(K)}[i]$ で得られる。また、予め $ACM[-1] \equiv 0$ と決めておくと、

[0121] [数7]

$$X_{i(w_i)}^{(R)}$$

は以下で得られる。

[0122] [数8]

$$X_{i(w_i)}^{(R)} = (INV_{(R)}^{(R)}[ACM_{(K)}[i-1]], \dots, INV_{(R)}^{(R)}[ACM_{(K)}[i]-1])$$

上記の2つからスペクトル分解の第i項は以下で求まることになる。

[0123] [数9]

$$\lambda_i: X_{i(w_i)}^{(R)} = S V L_{(K)}[i]: (INV_{(R)}^{(R)}[ACM_{(K)}[i-1]], \dots, INV_{(R)}^{(R)}[ACM_{(K)}[i]-1])$$

式(7)

例えば、図4に示す例の場合、スペクトル分解の第1項は以下で求まる。

[0124] [数10]

$$\begin{aligned}\lambda_1: X_{1(w_1)}^{(R)} &= SVL_{(K)}[1]: \left(INV_{(R)}^{(R)}[ACM_{(K)}[1-1]], \dots, INV_{(R)}^{(R)}[ACM_{(K)}[1]-1] \right) \\ &= Bob: \left(INV_{(R)}^{(R)}[ACM[0]], \dots, INV_{(R)}^{(R)}[ACM[1]-1] \right) \\ &= Bob: \left(INV_{(R)}^{(R)}[1], \dots, INV_{(R)}^{(R)}[2] \right) = Bob: (0,3)\end{aligned}$$

・転置構造体から得られる情報

転置構造体を使うと、SVLから得られる情報（上記の情報1～情報4）に加えて、以下の情報5～情報8も得られる。

[0125] 情報5：小さい方からi番目の値の出現回数が以下でわかる。

[0126] $count = ACM_{(K)}[i] - ACM_{(K)}[i-1]$ 式(8)

例えば、図4に示す例の場合、小さい方から1番目の値の出現回数は、 $ACM_{(K)}[1] - ACM_{(K)}[0] = 3 - 1 = 2$ となる。

[0127] 情報6：ソート結果のi番目のレコード番号は以下で読み出せる。

[0128] ソート結果のi番目のレコード番号 $= INV[i]$ 式(9)

例えば、図4に示す例の場合、ソート結果の1番目のレコード番号は、 $INV[1] = 0$ (Bob)として読み出せる。

[0129] 情報7：小さい方からi番目の値を持つレコード番号を以下の配列として読み出すことができる。

[0130] $(INV_{(R)}^{(R)}[ACM_{(K)}[i-1]], \dots, INV_{(R)}^{(R)}[ACM_{(K)}[i]-1])$

例えば、図4に示す例の場合、小さい方から1番目の値 (Bob) を持つレコード番号は、 $(INV[1], \dots, INV[2]) = (0, 3)$ として読み出せる。

[0131] 情報8：値 $v_0 \sim v_1$ を持つレコード番号は以下で読み出せる。

[0132] i_0 を $v_0 \leq SVL[i_0]$ を満たす最小の i_0 、 i_1 を $SVL[i_1] \leq i_1$ を満たす最大の i_1 として、 $(INV_{(R)}^{(R)}[ACM_{(K)}[i_0-1]], \dots,$

$\dots, \text{INV}_{(R)}^{(R)}[\text{ACM}_{(K)}[i_1] - 1])$ 式(10)

例えば、図4に示す例の場合、値Bob~Cathyを持つレコード番号は $i_0 = 1$ 、 $i_1 = 2$ であるため、 $(\text{INV}[\text{ACM}[1 - 1]], \dots, \text{INV}[\text{ACM}[2] - 1]) = (\text{INV}[1], \dots, \text{INV}[4 - 1]) = (0, 3, 2)$ として読み出せる。

[0133] 転置構造体でインデックスを作成できること

上述したように、インデックスの成立要件は、ソート・検索・集計結果のサイズWがわかり、 $i \in 0, 1, \dots, W - 1$ を指定するとおよそ $O(\log(R))$ 以下でソート・検索・集計結果の i 行目を読み出せることである。以下、転置構造体でインデックスを作成できることを確かめる。

[0134] (ソート)

ソート結果のサイズはRであることがわかる。ソート結果の i 番目の要素は式(9)を使い $O(1)$ で読み出せる。したがって、転置構造体はソートについてインデックスを作成するための要件を満たす。

[0135] (検索)

式(10)の場合について説明する。検索結果のサイズは $\text{ACM}_{(K)}[i_1] - \text{ACM}_{(K)}[i_0 - 1]$ であるとわかる。例えば、図4に示す例の場合、値Bob~Cathyを持つレコード番号は $i_0 = 1$ 、 $i_1 = 2$ であるため、 $\text{ACM}_{(K)}[2] - \text{ACM}_{(K)}[1 - 1] = 4 - 1 = 3$ であるとわかる。

[0136] 検索結果の i 番目の要素は、 $(\text{INV}_{(R)}^{(R)}[\text{ACM}_{(K)}[i_0 - 1]], \dots, \text{INV}_{(R)}^{(R)}[\text{ACM}_{(K)}[i_1] - 1])[i]$ により $O(1)$ で読み出せる。例えば、図4に示す例の場合、値Bob~Cathyで検索するとき $i_0 = 1$ 、 $i_1 = 2$ であるため、 $(\text{INV}_{(R)}^{(R)}[\text{ACM}_{(K)}[1 - 1]], \dots, \text{INV}_{(R)}^{(R)}[\text{ACM}_{(K)}[2] - 1])[i] = (0, 3, 2)[i]$ で読み出せる。

[0137] したがって、転置構造体は検索についてインデックスを作成するための要件を満たす。

[0138] (集計)

集計結果のサイズは上記の情報 1 から K であることがわかる。集計結果の i 番目の値は上記の情報 2 で与えられ、 $O(1)$ で得られる。集計結果の i 番目の値の出現回数は式 (8) で与えられ、 $O(1)$ で得られる。したがって、転置構造体は集計についてインデックスを作成するための要件を満たす。

[0139] 例えば、図 4 に示す例の場合で $i = 1$ のとき、集計結果のサイズは 3、集計結果の 1 番目の値は $B o b$ 、集計結果の 1 番目の値の出現回数は $ACM[1] - ACM[1 - 1] = 3 - 1 = 2$ であるとわかる。

[0140] 上記により、転置構造体を使ってソート・検索・集計を高速化する D 5 A インデックスが作成できるといえる。

[0141] ・ D 5 A インデックスによるソート・検索・集計の結果の保持がわずかな記憶領域済むこと

(ソート結果)

ソート結果は既に INV に保持されており、新たな記憶領域を必要としない。

[0142] (検索結果)

式 (10) の場合について説明する。検索結果は配列 ($INV_{(R)}^{(R)}[ACM_{(K)}[i_0 - 1]]$, $\dots$, $INV_{(R)}^{(R)}[ACM_{(K)}[i_1] - 1]$) で与えられている。そのうち、 INV 、 ACM は既に作成されており、検索結果を保持するために必要とする記憶領域はヒット件数とは無関係に i_0 , i_1 の格納領域だけである。

[0143] (集計結果)

集計結果の i 番目の値は上記の情報 2 で与えられ、新たな記憶領域を必要としない。集計結果の i 番目の値の出現回数は式 (8) で与えられ、新たな記憶領域を必要としない。

[0144] 以上のように、D 5 A は仮想表形式データに値を、仮想インデックスにインデックスを供給する役目を持つ。カラムを SN 分解して得られる SVL と NNC を使うと、値を供給するだけでなく、上記の情報 1 ~ 情報 4 の 4 種類

の情報を得ることができる。また、転置構造体を使ってD 5 Aインデックスができる（つまり、ソート・検索・集計で転置構造体をインデックスのためのデータ構造として利用することができる）ことを示した。D 5 Aインデックスはソート・検索・集計を高速化するだけでなく、ソート・検索・集計の結果をわずかな記憶領域だけで保持することができる。

[0145] <仮想表形式データ>

例えば、RDB (Relational Database) でJOINやUNIONを行うとソース表形式データを読み出し、それらを比較し、書き出すデータを作り、その格納位置を決め、新たに確保した記憶領域上に書き出す。すると、時間と記憶領域とを必要とする。それがアーカイブの表形式データを活用しようとする際の第1ステップの問題である。この問題は、新たに生成される表形式データが巨大であればあるほど深刻になる。その生成に長い時間が掛かり、大きい記憶領域を必要とするためである。しかも、往々にして使用されるのは生成された表形式データのわずかな部分である。

[0146] 仮想表形式データは必要な部分を必要になったときに作り出す仕組みで、上記の問題を解決する。つまり、仮想表形式データではあるセルを表示する必要が生じた場合にその時点でそのセルに対応するソース表形式データのセルを参照してそれを表示する。しかし、そのためには仮想表形式データのセルと1つ以上のソース表形式データのセルとの間の対応を定める写像が予め定義されている必要がある。その写像は仮想表形式データから複数のソース表形式データへの写像とすることもできるし、その逆に複数のソース表形式データから仮想表形式データへの写像とすることもでき、どちらも本質的には同じである。しかし、前者は写像の宛先がどのソース表形式データのどのセルという2つの情報の組になるのに対して、後者は仮想表形式データのどのセルという1つの情報だけで済み定義しやすい。そこで、以下では、後者で説明を行う。なお、その両者とも逆写像を必要とする。前者は仮想インデックスが逆写像を必要とし、後者は仮想表形式データがソース表形式データを参照して値を取得する際に必要となる。

[0147] 後者の各ソース表形式データのどのセルを仮想表形式データ上のどのセルに対応させるか、という写像を割り付け写像と呼ぶ。ただし、実際の割り付け写像はソース表形式データ上のカラムである各ソースカラムから仮想表形式データ上のカラムである仮想カラムへの写像を定義するものとする。その各ソースカラムは1つ以上の区間に分割され、区間毎に対応表又はルールによりその区間上のセルが仮想カラム上のセルに対応付けられる。その区間毎の対応付けを区間写像と呼ぶ。割り付け写像はすべての区間写像を合わせたものである。

[0148] <<割り付け写像の作成手順>>

図5を参照しながら割り付け写像の作成手順を説明する。なお、以下、 r_s はソース表形式データ上のレコード番号、 r_v は仮想表形式データ上のレコード番号を指すものとする。

[0149] 仮想表形式データ上のカラムである仮想カラムは1つ以上のソースカラムからの割り付け写像で構成される。図5に示す例では、仮想カラムはソースカラム0とソースカラム1から割り付けられている。

[0150] 図5の左下のソースカラム1を持つ空白D5Aは、仮想カラム生成時にプログラムの自動で生成されるD5Aである。空白D5Aは仮想カラム上の割り付けを受けていないセルに空白値を供給し、仮想カラムが後述する全単射の条件を満たすようにするためのものである。全単射の条件を満たしていると、仮想表形式データには仮想インデックスが自動的に備わる。一方、仮想表形式データをソート・検索・集計する必要が無い場合もある。例えば、UNIONやJOINした結果全体をそのまま利用したい場合である。その場合はもちろん全単射の条件を考慮せず仮想表形式データを定義して使うことができる。

[0151] 各ソースカラムは1つ以上のソース区間に分割される。図5に示す例では、ソースカラム0は区間1及び区間2に分割されている。その各ソース区間から仮想カラムへの写像（つまり、区間写像）は2種類ある。1つ目は、要素が自然数でかつ重複がないA写像Mで定義される列挙型写像である。2つ

目は、1 次関数で定義される 1 次関数型写像である。前者の列挙型写像はあらゆる割り付け写像を定義できる一方で多くの記憶領域を消費する。後者の 1 次関数型写像は列挙型写像の M を 1 次関数で置き換えた写像であり、記憶領域をわずかに消費せず巨大な区間写像でも定義できるが規則性のある割り付けにしか対応できない。すべてのソース区間で区間写像が定義されると、割り付け写像が完成する。

[0152] 上記いずれの区間写像もそれが定義されると、その逆写像及び逆写像の存在条件も確定する。仮想表形式データを表示する際は、まず仮想カラムのレコード番号から上記の逆写像が存在するソース区間を特定し、次にその区間写像の逆写像を使ってソースカラム上のレコード番号を特定し、そしてそのレコード番号から値を取得して表示することになる。

[0153] 以下では、列挙型写像による区間写像と、1 次関数型写像による区間写像とを説明した後、再び図 5 を参照して仮想表形式データの表示方法を説明する。

[0154] 《列挙型写像による区間写像及びその逆写像》

図 6 は列挙型写像による割り付けの例を示している。図 6 において、 R_s はソースカラムのサイズ、 r_s はソースカラムのレコード番号、 R_v は仮想カラムのサイズ、 r_v は仮想カラムのレコード番号、 Q はソースカラム上の区間、 q は区間 Q の開始位置、 u は区間 Q の長さ、 V は区間 Q に対応する仮想カラム上の区間、 v は区間 V の開始位置、 w は区間 V の長さを表す。

[0155] このとき、列挙型写像による区間写像 $F : Q \rightarrow V$ は以下で表される。

$$[0156] \quad r_v = M_{(u)}^{(R_v)} [r_s - q] \quad \text{式 (11)}$$

また、その逆写像 $F^{-1} : V \rightarrow Q$ は以下で表される。

$$[0157] \quad M_{(u)}^{(R_v)} = L_{(u)}^{(R_v)} \cdot P_{(u)}^{(u)} ;$$

$$L_{(u)}^{(R_v)} [j] = r_v ;$$

$$r_s = P_{(u)}^{-1} [j] + q \quad \text{式 (12)}$$

・列挙型写像

まず、列挙型写像による区間写像を定義する式 (11) を説明する。図 6

において、ソースカラム及び仮想カラムの横に記載されている番号はレコード番号である。Mはソース区間の先頭から順に対応する仮想カラム上のレコード番号を書き出すことで定義することができる。図6に示す例では(7, 5, 9, 1)となる。ここで、ソース区間Qの開始位置qは2である。

[0158] ソース区間(区間Q)は $r_s = 2$ から始まり $r_s = 5$ で終わる。式(11)に各 r_s を適用すると以下となる。

[0159] $r_s = 2$ のとき、 $r_v = (7, 5, 9, 1) [2 - 2] = 7$
 $r_s = 3$ のとき、 $r_v = (7, 5, 9, 1) [3 - 2] = 5$
 $r_s = 4$ のとき、 $r_v = (7, 5, 9, 1) [4 - 2] = 9$
 $r_s = 5$ のとき、 $r_v = (7, 5, 9, 1) [5 - 2] = 1$

・ 列挙型写像の逆写像

次に、列挙型写像の逆写像を定義する式(12)を説明する。

[0160] Mは重複のない自然数を要素とするA写像であり、LP分解を行うことができる。式(12)の最上段はそのLP分解を示している。LP分解によって $L = (1, 5, 7, 9)$ 、 $P = (2, 1, 3, 0)$ を得る。Lはユニーク昇順配列であり、要素の有無、要素がある場合はその出現位置をバイセクションサーチで求めることができる。Pは対称配列であり、逆元 $P^{-1} = (3, 1, 0, 2)$ を持つ。

[0161] 式(12)の中段は r_v がL中の何番目に出現するかを求める方法を示しており、jはその出現位置である。なお、jが求まらない場合、 r_v はソース区間Qからの写像を受け入れていないことを意味し、逆写像は存在しない。

[0162] 式(12)の下段は $P^{-1} = (3, 1, 0, 2)$ と $q = 2$ と上記のjを用いて r_s が求まることを意味している。

[0163] 以上を踏まえ、 $r_v = 1, 5, 7, 9$ の場合について、 r_s を計算してみる。

[0164] $r_v = 1$ のとき、 $(1, 5, 7, 9) [j] = 1$ から $j = 0$; $r_s = (3, 1, 0, 2) [j] + 2 = 5$

$r_v = 5$ のとき、 $(1, 5, 7, 9) [j] = 5$ から $j = 1$; $r_s = (3, 1, 0, 2) [j] + 2 = 3$

$r_v = 7$ のとき、 $(1, 5, 7, 9) [j] = 7$ から $j = 2$; $r_s = (3, 1, 0, 2) [j] + 2 = 2$

$r_v = 9$ のとき、 $(1, 5, 7, 9) [j] = 9$ から $j = 3$; $r_s = (3, 1, 0, 2) [j] + 2 = 4$

《 1 次関数型写像による区間写像及びその逆写像 》

図 7 は 1 次関数型写像による割り付けの例を示している。図 7 において、 R_s はソースカラムのサイズ、 r_s はソースカラムのレコード番号、 R_v は仮想カラムのサイズ、 r_v は仮想カラムのレコード番号、 Q はソースカラム上の区間、 q は区間 Q の開始位置、 u は区間 Q の長さ、 V は区間 Q に対応する仮想カラム上の区間、 v は区間 V の開始位置、 w は区間 V の長さを表す。

[0165] このとき、1 次関数型写像による区間写像 $F : Q \rightarrow V$ は以下で表される。

[0166] $r_v = a \times r_s + b$; a, b は整数、 $a \neq 0$ 式 (13)

また、その逆写像 $F^{-1} : V \rightarrow Q$ は以下で表される。

[0167] $r_s = (r_v - b) / a$ 式 (14)

ただし、 $v \leq r_v < v + w$ でないとき、 r_s が整数にならないとき、 r_v は F の値域ではない。

[0168] ・ 1 次関数型写像

まず、1 次関数型写像を定義する式 (13) を説明する。図 7 において、ソースカラム及び仮想カラムの横に記載されている番号はレコード番号である。 a は仮想カラム上のインターバルであり、図 7 に示す例では $a = 3$ 、 $b = -5$ となる。

[0169] ソース区間 (区間 Q) は $r_s = 2$ から始まり $r_s = 5$ で終わる。式 (13) に各 r_s を適用すると以下となる。

[0170] $r_s = 2$ のとき、 $r_v = 3 \times r_s - 5 = 1$

$r_s = 3$ のとき、 $r_v = 3 \times r_s - 5 = 4$

$r_s = 4$ のとき、 $r_v = 3 \times r_s - 5 = 7$

$r s = 5$ のとき、 $r v = 3 \times r s - 5 = 10$

・ 1 次関数型写像の逆写像

次に、1 次関数型写像の逆写像を定義する式 (14) を説明する。上述したように、 $a = 3$ 、 $b = -5$ である。

[0171] まず、 $v \leq r v < v + w$ であることを確かめる必要がある。 $v = 1$ 、 $w = 10$ であるため、 $1 \leq r v < 1 + 10$ でなければならない。その条件を満たす場合、式 (14) の上段を使って $r s$ を求める。このとき、式 (14) の下段にあるように、 $r s$ が整数でない場合、 $r v$ はソース区間 Q からの写像を受け入れていないことを意味し、逆写像は存在しない。

[0172] $r v = 1$ のとき、 $r s = (1 - (-5)) / 3 = 2$

$r v = 4$ のとき、 $r s = (4 - (-5)) / 3 = 3$

$r v = 7$ のとき、 $r s = (7 - (-5)) / 3 = 4$

$r v = 10$ のとき、 $r s = (10 - (-5)) / 3 = 5$

《仮想表形式データの表示》

再び図 5 を参照して仮想表形式データの表示方法を説明する。ソースカラム 0 のソース区間 1 は 1 次関数型写像による区間写像、ソースカラム 0 のソース区間 2 は列挙型写像による区間写像、ソースカラム 1 のソース区間 1 は 1 次関数型写像による区間写像としている。それぞれの区間写像の定義は以下になる。

[0173] ソースカラム 0 のソース区間 1 における区間写像は、式 (13) により、 $F : r v = r s \times 2 + 1$ と定義される。

[0174] ソースカラム 0 のソース区間 2 における区間写像は、式 (11) により、 $F : r v = (2, 7, 0) [r s - 3]$ と定義される。以下のように $r s = 3, 4, 5$ を代入して正しいことが確認できる。

[0175] $r s = 3$ のとき $r v = (2, 7, 0) [3 - 3] = 2$

$r s = 4$ のとき $r v = (2, 7, 0) [4 - 3] = 7$

$r s = 5$ のとき $r v = (2, 7, 0) [5 - 3] = 0$

[0176] ブランク D 5 A のソースカラム 1 のソース区間 1 における区間写像は、式

(13) により、 $F:rv = rs \times 2 + 4$ と定義される。

[0177] 上記の各区間写像 F が定義されるとその逆写像 F^{-1} は以下のように自動的に定まる。

[0178] ソースカラム 0 のソース区間 1 における逆写像は、式 (14) により、 $F^{-1}:rs = (rv - 1) / 2$ となる。

[0179] ソースカラム 0 のソース区間 2 における逆写像は、式 (12) により、 $F^{-1}:rs = (2, 0, 1)[j] + 3; rv = (0, 2, 7)[j]$ となる。
まず、 $rv = 0$ のとき $(0, 2, 7)$ をバイセクションサーチして rv の格納位置 $j = 0$ がわかり、 $rs = (2, 0, 1)[0] + 3 = 5$ となる。同様に、 $rv = 2$ のとき $(0, 2, 7)$ をバイセクションサーチして rv の格納位置 $j = 1$ がわかり、 $rs = (2, 0, 1)[1] + 3 = 3$ となる。同様に、 $rv = 7$ のとき $(0, 2, 7)$ をバイセクションサーチして rv の格納位置 $j = 2$ がわかり、 $rs = (2, 0, 1)[2] + 3 = 4$ となる。

[0180] ブランク D5A のソースカラム 1 のソース区間 1 における逆写像は、式 (13) により、 $F^{-1}:rs = (rv - 4) / 2$ となる。

[0181] 上記をまとめると、 $rv = 0, 1, 2, 3, 5, 7$ のときソースカラム 0 で $rs = 5, 0, 3, 1, 2, 4$ 、 $rv = 4, 6$ のときソースカラム 1 で $rs = 0, 1$ を得る。

[0182] ≪仮想表形式データの構成法≫

上記で説明した機能を実現するため、仮想表形式データは以下の情報で構成される。

- [0183]
1. レコード数
 2. 仮想カラムの名前及びデータ型
 3. 1つ以上のソース表形式データの URL 又はパス
 4. 上記のソース表形式データ中のソースカラムとソース区間の定義
 5. ソース区間毎の区間写像とその逆写像の定義

上記のうち 5 以外は大きな記憶領域を必要としない。また、上記の 5 に大きな記憶領域を必要とするのは、ソース区間が大きくかつ列挙型写像のとき

に限られる。このため、仮想表形式データは多くの場合、コンパクトに構成できる。

[0184] 《異なる区間からの割り付けの衝突検出》

ここで、割り付け写像による割り付け先が衝突していないかを調べる方法について説明する。以下では、2つの割り付け写像を $F_0: Q_0 \rightarrow V_0$ 、 $F_1: Q_1 \rightarrow V_1$ とする。

[0185] まず、自明なケースとして以下の（条件1-1）～（条件1-3）のいずれかを満たす場合、2つの割り付け写像 F_0 、 F_1 は衝突しない。

[0186] （条件1-1） V_0 と V_1 が同一の仮想カラム上にない場合

 （条件1-2） $v_0 + w_0 - 1 < v_1$ である場合

 （条件1-3） $v_1 + w_1 - 1 < v_0$ である場合

 なお、 v_0 は V_0 の開始位置、 w_0 は V_0 の長さ、 v_1 は V_1 の開始位置、 w_1 は V_1 の長さである。

[0187] 上記の自明なケースに該当しない場合、以下の（ケース1）～（ケース3）により2つの割り付け写像 F_0 、 F_1 の衝突有無を調べることができる。

[0188] （ケース1） F_0 、 F_1 がいずれも列挙型写像である場合

$L_0 = (v_0, \dots, v_0 + w_0 - 1)$ 、 $L_1 = (v_1, \dots, v_1 + w_1 - 1)$ として、 L_0 と L_1 の共通範囲を新たに L_0 、 L_1 とする。なお、 w_0 は V_0 の長さ、 w_1 は V_1 の長さである。

[0189] そして、 $i = 0$ 、 $j = 0$ に対して、以下の手順4-1～手順4-3を実行する。

[0190] 手順4-1： $L_0[i] = L_1[j]$ であれば「衝突あり」として終了する。

[0191] 手順4-2： $L_0[i] < L_1[j]$ であれば、 $i \leftarrow i + 1$ として手順4-1に戻る。ただし、 $i \leftarrow i + 1$ とした結果、 i が L_0 の要素を表すインデックスの最大値を超える場合は「衝突なし」として終了する。

[0192] 手順4-3： $L_0[i] > L_1[j]$ であれば、 $j \leftarrow j + 1$ として手順4-1に戻る。ただし、 $j \leftarrow j + 1$ とした結果、 j が L_1 の要素を表すインデック

スの最大値を超える場合は「衝突なし」として終了する。

[0193] (ケース2) F_0 、 F_1 がいずれも1次関数型写像である場合

V_0 と V_1 の共通部分を選び出し、 F_0 、 F_1 を、当該共通部分を値域とする以下の式でそれぞれ記述し直す。その際、 a_0 、 a_1 が正になるように調整する。

[0194] $y_0 = a_0 \times x_0 + b_0$; $x_0 = 0, 1, 2, \dots, N_0 - 1$

$y_1 = a_1 \times x_1 + b_1$; $x_1 = 0, 1, 2, \dots, N_1 - 1$

ここで、 a_0 、 a_1 は1以上の整数、 b_0 、 b_1 は整数である。

[0195] そして、 $x_0 = 0$ 、 $x_1 = 0$ として、以下の手順5-1～手順5-3を実行する。

[0196] 手順5-1: $y_0 = y_1$ であれば「衝突あり」として終了する。

[0197] 手順5-2: $y_0 < y_1$ であれば、 $x_0 \leftarrow x_0 + \text{MaxInt}(1, (y_1 - y_0) \text{div } a_0)$ として手順5-1に戻る。ただし、 $x_0 \leftarrow x_0 + \text{MaxInt}(1, (y_1 - y_0) \text{div } a_0)$ とした結果、 $x_0 > \text{MinInt}(N_0 - 1, \text{LCM}(a_0, a_1) \text{div } a_0)$ である場合は「衝突なし」として終了する。

[0198] 手順5-3: $y_0 > y_1$ であれば、 $x_1 \leftarrow x_1 + \text{MaxInt}(1, (y_0 - y_1) \text{div } a_1)$ として手順5-1に戻る。ただし、 $x_1 \leftarrow x_1 + \text{MaxInt}(1, (y_0 - y_1) \text{div } a_1)$ とした結果、 $x_1 > \text{MinInt}(N_1 - 1, \text{LCM}(a_0, a_1) \text{div } a_1)$ である場合は「衝突なし」として終了する。

[0199] ここで、 MaxInt は最大の整数を選ぶ関数、 MinInt は最小の整数を選ぶ関数、 div は整数の割り算（余りは切り捨て）、 LCM は最小公倍数を返す関数を表す。

[0200] (ケース3) F_0 、 F_1 のいずれか一方が列挙型写像、他方が1次関数型写像である場合

以下、 F_0 を1次関数型写像、 F_1 を列挙型写像であるものとする。 V_0 と V_1 の共通部分を選び出し、 F_0 、 F_1 を、当該共通部分を値域とする以下の式で

それぞれ記述し直す。その際、 a_0 が正になるように調整する。

[0201] $y_0(n) = a_0 \times n + b_0$; $x_0 = 0, 1, 2, \dots, N-1$

$y_1(m) = (v_1, \dots, v_1 + w_1 - 1)$ [m]

ここで、 $a_0 > 0$ である。また、 $(v_1, \dots, v_1 + w_1 - 1)$ はサイズMの昇順配列である。

[0202] そして、 $n = 0$ 、 $m = 0$ として、以下の手順6-1～手順6-3を実行する。

[0203] 手順6-1: $y_0(n) = y_1(m)$ であれば「衝突あり」として終了する。

[0204] 手順6-2: $y_0(n) > y_1(m)$ であれば、 $m \leftarrow m + 1$ として手順6-1に戻る。ただし、 $m \leftarrow m + 1$ とした結果、 $m = M$ である場合は「衝突なし」として終了する。

[0205] 手順6-3: $y_0(n) < y_1(m)$ であれば、 $n \leftarrow (y_1(m) - y_0(n)) \div a_0$ として手順6-1に戻る。ただし、 $n \leftarrow (y_1(m) - y_0(n)) \div a_0$ とした結果、 $n \geq N$ である場合は「衝突なし」として終了する。

[0206] <仮想インデックス>

これまでにD5Aから直接又は間接的に値を継承した仮想表形式データを作成する方法を説明した。以下では、D5A上の転置構造体を直接又は間接的に継承して仮想表形式データ上に仮想転置構造体が自動的に成立する理由を説明する。仮想転置構造体が成立すると、それをインデックスのためのデータ構造として使う仮想インデックスも自動的に成立する。仮想インデックスは、転置構造体の代わりに仮想転置構造体を使うだけで、アルゴリズムはD5Aインデックスと同じものを使うのでD5Aインデックスと同じ機能と特性を持つ。つまり、カラムのソート・検索・集計を高速化でき、かつ、巨大なソート結果・検索結果・集計結果でもわずかな記憶領域だけで保持できる。

[0207] 仮想転置構造体は転置構造体と同様に、仮想SVL、仮想ACM、仮想I

NVを持つ構造体である。ここで、仮想SVL、仮想ACM、仮想INVは仮想配列である。仮想配列とは、仮想表形式データがそうであるように値を保持せず、サイズWが予めわかり、 $i \in 0, 1, \dots, W-1$ を指定するとおよそ $O(\log(R))$ 以下でi番目の要素が取り出せる、配列と同等の機能を持った仕組みのことである。仮想SVL、仮想ACM、仮想INVは、仮想表形式データを定義した割り付け写像が以下の全単射の条件を満たしていれば自動的に成り立つ。

[0208] ・全単射の条件

1. (写像が成り立つこと) ソースカラムのすべてのセルが仮想カラム上のセルに写されていること。

[0209] 2. (単射であること) どの仮想カラム上のセルも2つ以上のソースカラムのセルから写されていないこと。

[0210] 3. (全射であること) すべての仮想カラム上のセルがソースカラムのセルから写されていること。

[0211] 4. (ソースカラムが全単射で作られていること) ソースカラムがD5Aの場合はソースカラムが全単射で作られているといえる。ソースカラムが仮想カラムの場合はソースカラムが上記の1～3を満たして作られていると全単射で作られているといえる。

[0212] 以下では、まず複数の転置構造体から仮想転置構造体が構成できることを説明し、その仮想転置構造体を使う仮想インデックスでソート・検索・集計が高速に実行でき、ソート結果・検索結果・集計結果が新たにわずかな記憶領域を確保するだけで保持できることを確かめる。次に仮想転置構造体と転置構造体でも更なる仮想転置構造体が階層的に作れ、それを使う仮想インデックスでもソート・検索・集計が高速に実行でき、ソート結果・検索結果・集計結果が新たにわずかな記憶領域を確保するだけで保持できることを確かめる。最後にスペクトル分解を用いると階層的に作られた仮想転置構造体の動作でも容易に理解できることを確認する。

[0213] 《仮想転置構造体の構成方法》

一例として、ソースカラム#0 : $C_{0(4)}$ とソースカラム#1 : $C_{1(4)}$ から定義された仮想カラム $C_v(8)$ を図8に示す。 $C_{0(4)}$ と $C_{1(4)}$ の要素は区別のため、 A_0 、 B_0 、 C_0 、 B_1 、 C_1 、 D_1 とソースカラムの番号を添え字として表示しているが、値としては $B_0 = B_1$ 、 $C_0 = C_1$ である。このとき、ソースカラム#0からの割り付け先は仮想カラムの5、7、2、3行目であるため、割り付け写像 $F_0 = (5, 7, 2, 3)$ である。同様に、ソースカラム#1からの割り付け先は仮想カラムの4、0、6、1行目であるため、割り付け写像 $F_1 = (4, 0, 6, 1)$ である。

[0214] ソースカラム#0の転置構造体とソースカラム#1の転置構造体と仮想転置構造体とを図9に示す。ここで、ソースカラム#0の転置構造体中の SVL 、 ACM 、 INV をそれぞれ SVL_0 、 ACM_0 、 INV_0 と表記し、ソースカラム#1の転置構造体中の SVL 、 ACM 、 INV をそれぞれ SVL_1 、 ACM_1 、 INV_1 と表記する。また、仮想カラムの仮想転置構造体中の SVL 、 ACM 、 INV をそれぞれ SVL_v 、 ACM_v 、 INV_v と表記する。

[0215] INV_0 及び INV_1 は共に転置レコード番号を格納した配列であり、ソースカラム上のレコード番号は割り付け写像で仮想カラム上のレコード番号に写されるため、 INV_0 及び INV_1 を読み出す際にはそれぞれ INV_0' 及び INV_1' に読み替える必要がある。 INV_0' 、 INV_1' は以下の式(15)で算出できる。

$$[0216] \quad INV_0' = F_0 \cdot INV_0 = (5, 7, 2, 3) \cdot (3, 0, 2, 1) = (3, 5, 2, 7) \quad \text{式(15) - 1}$$

$$INV_1' = F_1 \cdot INV_1 = (4, 0, 6, 1) \cdot (1, 2, 0, 3) = (0, 6, 4, 1) \quad \text{式(15) - 2}$$

以下、図9を例として、仮想転置構造体の構成方法を仮想配列 SVL_v 、 ACM_v 、 INV_v の順に説明する。

[0217] ・仮想配列 SVL_v の構成方法

まず、 SVL_v のサイズを求める方法を説明する。予め SVL_0 及び SVL_1 に共通して出現する値 B 、 C はソースカラムの順位を加味して大小関係を決

めるものと取り決める。つまり、 SVL_0 の B_0 は SVL_1 の B_1 よりも小さいものと取り決める。すると、各 SVL 間で大小関係のつかない値は存在しなくなり、 SVL_v のサイズは、 SVL_0 のサイズ $=3$ と SVL_1 のサイズ 3 の和である 6 となり、一意に確定する。

[0218] 次に、 SVL_v の i 番目の要素 $SVL_v[i]$ を求める方法を説明する。 SVL_0 又は SVL_1 の中から適当な値 v を選択する。ただし、この v はソースカラムの順位を加味して大小関係を決める上記の取り決めに従うものとする。次に、 SVL_0 及び SVL_1 の中で v より小さい値の個数の和 j を求める。もし $j < i$ ならば v よりも大きい v' を選択し、上記を再び行う。 $j > i$ のときも同様に行う（ただし、 v よりも小さい v' を選択する。）。 $j = i$ となったらそのときの v が $SVL_v[i]$ である。

[0219] 上記の $SVL_v[i]$ を求める方法は、およそ $O(\log(K))$ 程度で行える。すると、 SVL_v はサイズがわかり、その i 番目の要素を $O(\log(R))$ 以下で取り出せるので、仮想配列 SVL_v が存在すると考えてよいことになる。

[0220] $i = 0$ のときの例を示す。最初に v として SVL_1 の B_1 を選択してみる。 SVL_0 中に v より小さい値は 2 個あり、 SVL_1 中には 0 個ある。したがって、 $j = 2 + 0 = 2$ となり、 $i = 0$ より大きい。そこで、次に、 v として SVL_0 の A_0 を選択してみる。すると、 $j = 0$ となり、 $j = i$ となる。このため、 $SVL_v[0] = A_0$ となる。

[0221] $i = 4$ のときの例を示す。最初に v として SVL_0 の B_0 を選択してみる。 SVL_0 中に v より小さい値は 1 個あり、 SVL_1 中には 0 個ある。したがって、 $j = 1 + 0 = 1$ となり、 $i = 4$ より小さい。そこで、次に、 v として SVL_1 の C_1 を選択してみる。すると、 $j = 3 + 1 = 4$ となり、 $j = i$ となる。このため、 $SVL_v[4] = C_1$ となる。

[0222] 上記により、仮想配列 $SVL_v = (A_0, B_0, B_1, C_0, C_1, D_1)$ を作ることができる。

[0223] ・仮想配列 ACM_v の構成方法

まず、 ACM_v のサイズは SVL_v のサイズとして求まる。

[0224] 次に、 ACM_v の i 番目の要素 $ACM_v[i]$ を求める方法を説明する。まず、「仮想配列 SVL_v の構成方法」で説明した方法により $v = SVL_v[i]$ である v を求める。次に、 $SVL_0[i_0] \leq v$ を満たす最大の i_0 を求める。 i_0 が見つからない場合、 $i_0 = -1$ とみなす。なお、 $ACM[-1] \equiv 0$ であることに留意されたい。

[0225] 同様に、 $SVL_1[i_1] \leq v$ を満たす最大の i_1 を求める。そして、 $ACM_v[i]$ は、 $ACM_v[i] = ACM_0[i_0] + ACM_1[i_1]$ として求まる。

[0226] 上記の $ACM_v[i]$ を求める方法は、およそ $O(\log(K))$ 程度で行える。すると、 ACM_v はサイズがわかり、その i 番目の要素を $O(\log(R))$ 以下で取り出せるので、仮想配列 ACM_v が存在すると考えてよいことになる。

[0227] $i = 3$ のときの例を示す。最初に v として SVL_1 の B_1 を選択してみる。 SVL_0 中に v より小さい値は2個あり、 SVL_1 中には0個ある。したがって、 $j = 2$ となり、 $i = 3$ より小さい。そこで、次に、 v として SVL_0 の C_0 を選択してみる。すると、 $j = 2 + 1 = 3$ となり、 $SVL_v[3] = C_0$ であることになる。

[0228] 次に、 $SVL_0[i_0] \leq v$ を満たす最大の i_0 を求めると、 $i_0 = 2$ が求まる。次に、 $SVL_1[i_1] \leq v$ を満たす最大の i_1 を求めると、 $i_1 = 0$ が求まる。したがって、 $ACM_v[3]$ は、 $ACM_v[3] = ACM_0[2] + ACM_1[0] = 4 + 2 = 6$ として求まる。

[0229] 上記により、仮想配列 $ACM_v = (1, 3, 5, 6, 7, 8)$ を作ることができる。

[0230] ・仮想配列 INV_v の構成方法

まず、 INV_v のサイズは R とわかる。ここで、 R はソースカラムのサイズの総和である。

[0231] 次に、 $INV_v[i]$ を求めるために、 $ACM_v[j-1] \leq i \leq (ACM_v$

$[j] - 1$) を満たす j を求める方法を説明する。 j は ACM_V が定める INV_V 上の区間の番号である。

[0232] ACM_V のサイズを K_V とする。 K_V は ACM_0 及び ACM_1 のサイズの和である。まず、 $0, 1, \dots, K_V - 1$ の中から適当な j を選択し、「仮想配列 ACM_V の構成方法」で説明した方法により区間 $ACM_V[j - 1] \sim ACM_V[j] - 1$ を求める。

[0233] もし $i < ACM_V[j - 1]$ であれば、 j をもっと小さい値にして区間を選び直す。一方で、もし $i > ACM_V[j] - 1$ であれば、 j をもっと大きい値にして区間を選び直す。これを繰り返して j を確定し、 j が確定したら $offset = i - ACM_V[j - 1]$ を求めておく。

[0234] 次に、 $v = SVL_V[j]$ を求める。すると、値 v とそのソースカラムが求まる。その v を用いてそのソースカラム上の SVL で $SVL[k] = v$ となる k を求める。これにより、 $INV_V[i] = INV'[ACM[k - 1] + offset]$ として $INV_V[i]$ が求まる。ここで、 INV' 、 ACM は当該ソースカラム上の INV' 、 ACM である。

[0235] 上記の $INV_V[i]$ を求める方法は、およそ $O(\log(K))$ 程度で行える。すると、 INV_V はサイズがわかり、その i 番目の要素を効率良く取り出せるので、仮想配列 INV_V が存在すると考えてよいことになる。

[0236] $i = 4$ のときの例を示す。最初に $j = 3$ を選択してみる。区間の下限は $ACM_V[j - 1] = ACM_V[2] = 5$ 、区間の上限は $ACM_V[j] - 1 = ACM_V[3] - 1 = 5$ となり、選んだ区間が大きすぎるのがわかる。そこで、 $j = 2$ と選択し直してみる。区間の下限は $ACM_V[j - 1] = ACM_V[1] = 3$ 、区間の上限は $ACM_V[j] - 1 = ACM_V[2] - 1 = 4$ となる。このため、 $i = 4$ は $j = 2$ の区間に属することがわかる。 $offset = i - ACM_V[j - 1] = 4 - ACM_V[1] = 1$ である。次に、 $v = SVL_V[j] = SVL_V[2] = B_1$ であることがわかる。次に、ソースカラム # 1 上の $SVL[k] = B_1$ となる k は 0 であることがわかる。これにより、 $INV_V[4] = INV'[ACM[k - 1] + offset] = INV'[1] =$

6を得る。

[0237] 上記により、仮想配列 $INV_v = (3, 5, 2, 0, 6, 7, 4, 1)$ を作ることができる。

[0238] ・仮想インデックスでのソート

仮想転置構造体を用いた仮想インデックスでソートを行ってみる。図8、図9にはNNCが記載されていないためその代わりに図8の $C_{v(8)}$ を用いる。ソート結果は以下となる。

[0239] $C_{v(8)} \cdot INV_v = (B_1, D_1, B_0, A_0, C_1, B_0, B_1, C_0) \cdot (3, 5, 2, 0, 6, 7, 4, 1) = (A_0, B_0, B_0, B_1, B_1, C_0, C_1, D_1)$

上記の結果より確かにソートされていることが分かる。加えて、ソート結果には同じ「B」や「C」であっても $C_{0(4)}$ の「B」や「C」が $C_{1(4)}$ のそれらよりも先に来ることがわかる。これは同じ値のときはカラム番号が小さい方がその値が小さいとの取り決めが反映されたためである。また、ソート結果を保持するために新たな記憶領域を必要とすることがないのはD5Aインデックスと同じである。

[0240] ・仮想インデックスでの検索

仮想転置構造体を用いた仮想インデックスで検索を行ってみる。図8、図9にはNNCが記載されていないためその代わりに図8の $C_{v(8)}$ を用いる。また、検索は「B」～「C」の条件で行うものとする。

[0241] 式(10)で $i_0 = 1$ 、 $i_1 = 4$ となるため、 $(INV_{(R)}^{(R)}[ACM_{(K)}[i_0 - 1]], \dots, INV_{(R)}^{(R)}[ACM_{(K)}[i_1] - 1]) = (5, 2, 0, 6, 7, 4)$ となる。したがって、検索結果は以下となる。

[0242] $C_{v(8)} \cdot (5, 2, 0, 6, 7, 4) = (B_1, D_1, B_0, A_0, C_1, B_0, B_1, C_0) \cdot (5, 2, 0, 6, 7, 4) = (B_0, B_0, B_1, B_1, C_0, C_1)$

また、検索結果を保持するためには i_0 、 i_1 を保持するだけでよい。ためわずかな記憶域しか必要としないことはD5Aインデックスと同じである。

[0243] ・仮想インデックスを用いた集計

仮想転置構造体を用いた仮想インデックスで集計を行ってみる。集計結果の i 番目の値 $value$ は上記の情報 2 で与えられ、集計結果の i 番目の値の出現回数 $count$ は式 (8) で与えられる。それに従って集計を行うと、以下の表 1 のようになる。

[0244] [表1]

i	value	count	備考
0	A_0	1	ソースカラム#0より
1	B_0	2	ソースカラム#0より
2	B_1	2	ソースカラム#1より
3	C_0	1	ソースカラム#0より
4	C_1	1	ソースカラム#1より
5	D_1	1	ソースカラム#1より

ここで、同じ値でもその値を提供した $D5A$ が異なると分かれて集計結果が得られる。このため、その方が望ましいユーザはそれをそのまま、そうでないユーザは同じ値に対する出現回数を合算する必要がある。合算は最大で $D5A$ の数を超えないのでそのために要する処理時間は大きくない。また、集計結果を保持するために新たな記憶域を必要とすることがないのは $D5A$ インデックスと同じである。

[0245] 以上、仮想 SVL 、仮想 ACM 、仮想 INV が存在することがわかった。したがって、仮想転置構造体が存在するといえる。転置構造体は $D5A$ インデックスのためのデータ構造であり、同様の理由により仮想転置構造体は仮想インデックスのためのデータ構造であると言える。

[0246] また、仮想転置構造体はソースカラムの転置構造体への参照で構成されているが、ソースカラムが仮想カラムである場合にはその転置構造体は仮想転置構造体であるので、その場合、仮想転置構造体が階層的に構築されていることになる。

[0247] ≪階層的に作成された仮想表形式データの仮想インデックス≫

仮想転置構造体が階層的に作れることを述べたが、以下、実例を用いてそれを確かめる。図10は、D5Aファイルである $C_{0(3)}$ と $C_{1(2)}$ で仮想表形式データ $C_{V0(5)}$ を作成し、その仮想表形式データ $C_{V0(5)}$ とD5Aファイルである $C_{2(3)}$ で更なる仮想表形式データ $C_{V1(8)}$ を作成する例を示している。以下に割り付け写像の定義を示す。

$$\begin{aligned}
 [0248] \quad F(C_{0(3)} \rightarrow C_{V0(5)}) &= (2, 3, 0) \\
 F(C_{1(2)} \rightarrow C_{V0(5)}) &= (4, 1) \\
 F(C_{V0(5)} \rightarrow C_{V1(8)}) &= (2, 3, 7, 1, 5) \\
 F(C_{2(3)} \rightarrow C_{V1(8)}) &= (0, 6, 4)
 \end{aligned}$$

このとき、図10の転置構造体がどうなるのかを示したが図11である。 $C_{0(3)}$ 、 $C_{1(2)}$ 、 $C_{V0(5)}$ 、 $C_{2(3)}$ の転置構造体又は仮想転置構造体は同図に示したとおりになる。それぞれのINVは式(15)と同様に、以下のようにしてINV'に読み替えられる。

$$\begin{aligned}
 [0249] \quad INV_0' &= F(C_{0(3)} \rightarrow C_{V0(5)}) \cdot INV_0 = (2, 3, 0) \cdot (1, 0, 2) = (3, 2, 0) \\
 INV_1' &= F(C_{1(2)} \rightarrow C_{V0(5)}) \cdot INV_1 = (4, 1) \cdot (0, 1) = (4, 1) \\
 INV_{V0}' &= F(C_{V0(5)} \rightarrow C_{V1(8)}) \cdot INV_{V0} = (2, 3, 7, 1, 5) \cdot (3, 4, 2, 0, 1) = (1, 5, 7, 2, 3) \\
 INV_2' &= F(C_{2(3)} \rightarrow C_{V1(8)}) \cdot INV_2 = (0, 6, 4) \cdot (2, 1, 0) = (4, 6, 0)
 \end{aligned}$$

こうしてできたSVL、ACM、INV'を使って既に説明した手順により $C_{V0(5)}$ 、 $C_{V1(8)}$ の仮想転置構造体が構成できる。

[0250] ここで、階層的に作成された仮想表形式データを、仮想インデックスを使ってソート・検索・集計を行ってみる。

[0251] ・階層的にできた仮想表形式データの仮想インデックスを使ったソート
階層的にできた仮想転置構造体を用いた仮想インデックスでソートを行ってみる。図 10、図 11 には NNC が記載されていないためその代わりに図 10 の $C_{V1(8)}$ を用いる。ソート結果は以下となる。

[0252] $C_{V1(8)} \cdot INV_{V1} = (D_2, A_0, B_0, C_1, A_2, A_1, B_2, B_0) \cdot (1, 5, 4, 7, 2, 6, 3, 0) = (A_0, A_1, A_2, B_0, B_0, B_2, C_1, D_2)$

上記の結果より確かにソートされていることが分かる。加えて、ソート結果には同じ「A」や「B」や「C」であっても $C_{0(3)}$ の「A」や「B」が $C_{1(2)}$ 、 $C_{2(3)}$ のそれらよりも先に来ることがわかる。これは同じ値のときはカラム番号が小さい方がその値が小さいとの取り決めが反映されたためである。また、ソート結果を保持するために新たな記憶領域を必要とすることがないのは D5A インデックスと同じである。

[0253] ・階層的にできた仮想表形式データの仮想インデックスを使った検索
階層的にできた仮想転置構造体を用いた仮想インデックスで検索を行ってみる。図 10、図 11 には NNC が記載されていないためその代わりに図 10 の $C_{V1(8)}$ を用いる。また、検索は「B」～「C」の条件で行うものとする。

[0254] 式 (8) で $i_0 = 3$ 、 $i_1 = 5$ となるため、 $(INV_{(R)}^{(R)}[ACM_{(K)}[i_0 - 1]], \dots, INV_{(R)}^{(R)}[ACM_{(K)}[i_1] - 1]) = (7, 2, 6, 3)$ となる。したがって、検索結果は以下となる。

[0255] $C_{V1(8)} \cdot (7, 2, 6, 3) = (D_2, A_0, B_0, C_1, A_2, A_1, B_2, B_0) \cdot (7, 2, 6, 3) = (B_0, B_0, B_2, C_1)$

また、検索結果を保持するためには i_0 、 i_1 を保持するだけでよいためわずかな記憶域しか必要としないことは D5A インデックスと同じである。

[0256] ・階層的にできた仮想表形式データの仮想インデックスを用いた集計

階層的にできた仮想転置構造体を用いた仮想インデックスで集計を行ってみる。計結果の i 番目の値 $value$ は上記の情報 2 で与えられ、集計結果の i 番目の値の出現回数 $count$ は式 (8) で与えられる。それに従って集計を行うと、以下の表 2 のようになる。

[0257] [表2]

i	value	count	備考
0	A_0	1	$C_{0(3)}$ より
1	A_1	1	$C_{1(2)}$ より
2	A_2	1	$C_{2(3)}$ より
3	B_0	2	$C_{0(3)}$ より
4	B_2	1	$C_{2(3)}$ より
5	C_1	1	$C_{1(2)}$ より
6	D_2	1	$C_{2(3)}$ より

ここで、同じ値でもその値を提供した $D5A$ が異なると分かれて集計結果が得られる。このため、その方が望ましいユーザはそれをそのまま、そうでないユーザは同じ値に対する出現回数を合算する必要がある。その場合でも最大で $D5A$ の数を超えないのでそのために要する処理時間は大きくない。また、集計結果を保持するために新たな記憶域を必要とすることがないのは $D5A$ インデックスと同じである。

[0258] ≪スペクトル分解を通じた仮想インデックスの理解≫

図 11 の $C_{0(3)}$, $C_{1(2)} \rightarrow C_{V0(5)}$ と $C_{V0(5)}$, $C_{2(3)} \rightarrow C_{V1(8)}$ をスペクトル分解の立場から解釈すると分かりやすい。分かりやすさのために、以下の式はすべて割り付け写像を使って INV を INV' に読み替えた後の形で

記載する。

[0259] $C_{0(3)}$ と $C_{1(2)}$ をスペクトル分解で表すと以下となる。

[0260] $C_{0(3)} = A_0 : (3) + B_0 : (2, 0)$

$C_{1(2)} = A_1 : (4) + C_1(1)$

このため、 $C_{V0(5)}$ は、以下で表すことができる。

[0261] $C_{V0(5)} = C_{0(3)} + C_{1(2)} = A_0 : (3) + A_1 : (4) + B_0 : (2, 0) + C_1(1)$

また、 $C_{V0(5)}$ と $C_{2(3)}$ をスペクトル分解で表すと以下となる。

[0262] $C_{V0(5)} = A_0 : (1) + A_1 : (5) + B_0 : (7, 2) + C_1(3)$

$C_{2(3)} = A_2 : (4) + B_2 : (6) + D_2 : (0)$

このため、 $C_{V1(8)}$ は、以下で表すことができる。

[0263] $C_{V1(8)} = C_{V0(5)} + C_{2(3)} = A_0 : (1) + A_1 : (5) + A_2 : (4) + B_0 : (7, 2) + B_2 : (6) + C_1(3) + D_2 : (0)$

したがって、ソート結果列は (1, 5, 4, 7, 2, 6, 3, 0) となる。
また、「B」

～「C」での検索結果列は (7, 2, 6, 3) となる。集計も上記の表 3 と同様にできる。このため、手計算の場合はスペクトル分解による計算は分かりやすく優れていることが分かる。

[0264] <データ処理装置 10 を含むシステムの全体構成例>

本実施形態に係るデータ処理装置 10 を含むシステムの全体構成例を図 12 に示す。図 12 に示すように、本実施形態に係るデータ処理装置 10 は、インターネット等のネットワーク 20 上に分散して配置されているデータベースサーバと通信可能に接続される。これらのデータベースサーバには、D5A 又は仮想表形式データの少なくとも一方が 1 つ以上格納されている。

[0265] ここで、本実施形態に係るデータ処理装置 10 は、仮想表形式データ作成部 101 と、ソート部 102 と、検索部 103 と、集計部 104 と、記憶部 105 とを有する。なお、仮想表形式データ作成部 101、ソート部 102、検索部 103 及び集計部 104 は、例えば、データ処理装置 10 にインス

トールされた1以上のプログラムが、CPU (Central Processing Unit) 等のプロセッサに実行させる処理により実現される。また、記憶部105は、例えば、HDD (Hard Disk Drive)、SSD (Solid State Drive)、フラッシュメモリ等により実現される。

[0266] 仮想表形式データ作成部101は、上記の「仮想表形式データの構成法」で説明した1～5の情報を持つ仮想表形式データを作成する。また、仮想表形式データ作成部101は、上記の「仮想表形式データの構成法」で説明した5の区間写像を定義する際に、「異なる区間からの割り付けの衝突検出」で説明した方法により、区間写像（割り付け写像）による割り付け先が衝突しているか否かを判定する。

[0267] ソート部102は、上述した仮想表形式データの仮想転置構造体を用いた仮想インデックスでのソートを実行する。例えば、ソート部102は、仮想表形式データのソート対象の仮想カラムと、その仮想カラムの仮想転置構造体に含まれる仮想INVとを用いて、それらのインデックス演算によりソート結果を計算する。

[0268] 検索部103は、検索条件を受け付けると、上述した仮想表形式データの仮想転置構造体を用いた仮想インデックスで検索を実行する。例えば、検索部103は、仮想表形式データの検索対象の仮想カラムと、その仮想カラムに対する検索条件と、その仮想カラムの仮想転置構造体に含まれる仮想INV及び仮想ACMとを用いて、式(8)及び式(10)を計算した後、当該仮想カラムと式(8)及び式(10)の計算結果とのインデックス演算により検索結果を計算する。

[0269] 集計部104は、上述した仮想表形式データの仮想転置構造体を用いた仮想インデックスで集計を行う。例えば、集計部104は、仮想表形式データの集計対象の仮想カラムの仮想転置構造体を用いて、上記の情報2と式(8)により集計を行う。

[0270] 記憶部105は、各種データ（例えば、仮想表形式データ、DSA、ソート結果、検索結果、集計結果等）を記憶する。

[0271] <データ処理装置 10 のハードウェア構成例>

本実施形態に係るデータ処理装置 10 のハードウェア構成例を図 13 に示す。図 13 に示すように、本実施形態に係るデータ処理装置 10 は、入力装置 201 と、表示装置 202 と、外部 I/F 203 と、通信 I/F 204 と、RAM (Random Access Memory) 205 と、ROM (Read Only Memory) 206 と、補助記憶装置 207 と、プロセッサ 208 とを有する。これらの各ハードウェアは、それぞれがバス 209 を介して通信可能に接続される。

[0272] 入力装置 201 は、例えば、キーボード、マウス、タッチパネル、物理ボタン等である。表示装置 202 は、例えば、ディスプレイ、表示パネル等である。なお、データ処理装置 10 は、例えば、入力装置 201 及び表示装置 202 の少なくとも一方を有していなくてもよい。

[0273] 外部 I/F 203 は、記録媒体 203a 等の外部装置とのインタフェースである。記録媒体 203a としては、例えば、CD (Compact Disc)、DVD (Digital Versatile Disk)、SD メモリカード (Secure Digital memory card)、USB (Universal Serial Bus) メモリカード等が挙げられる。

[0274] 通信 I/F 204 は、データ処理装置 10 をネットワーク 20 に接続するためのインタフェースである。RAM 205 は、プログラムやデータを一時保持する揮発性の半導体メモリ (記憶装置) である。ROM 206 は、電源を切ってもプログラムやデータを保持することができる不揮発性の半導体メモリ (記憶装置) である。補助記憶装置 207 は、例えば、HDD、SSD、フラッシュメモリ等の不揮発性の記憶装置である。プロセッサ 208 は、例えば、CPU 等の各種演算装置である。

[0275] なお、図 13 に示すハードウェア構成は一例であって、データ処理装置 10 は、他のハードウェア構成を有していてもよい。例えば、データ処理装置 10 は、複数の補助記憶装置 207 や複数のプロセッサ 208 を有していてもよいし、図示したハードウェアの一部を有していなくてもよいし、図示したハードウェア以外の種々のハードウェアを有していてもよい。

[0276] <仮想表形式データの作成処理の流れ>

以下では、仮想表形式データの作成処理の流れについて、図 14 を参照しながら説明する。

- [0277] ステップ S 101 : まず、仮想表形式データ作成部 101 は、仮想表形式データのレコード数と仮想カラムの名前及びデータ型の指定を受け付ける（上記の「仮想表形式データの構成法」で説明した 1～2 の情報）。なお、仮想表形式データのレコード数と仮想カラムの名前及びデータ型は、例えば、ユーザによって指定される。
- [0278] ステップ S 102 : 次に、仮想表形式データ作成部 101 は、1 つ以上のソース表形式データの URL 又はパスの指定を受け付ける（上記の「仮想表形式データの構成法」で説明した 3 の情報）。なお、1 つ以上のソース表形式データの URL 又はパスは、例えば、ユーザによって指定される。また、1 つ以上のソース表形式データに含まれる 1 以上のカラムの URL 又はパスが指定されてもよい。
- [0279] ステップ S 103 : 次に、仮想表形式データ作成部 101 は、各ソース表形式データ中のソースカラムとソース区間の定義を受け付ける（上記の「仮想表形式データの構成法」で説明した 4 の情報）。なお、各ソース表形式データ中のソースカラムとソース区間の定義は、例えば、ユーザによって指定される。
- [0280] ステップ S 104 : 次に、仮想表形式データ作成部 101 は、ソース区間毎の区間写像の定義を受け付け（上記の「仮想表形式データの構成法」で説明した 5 の情報）、逆写像の定義を算出する。なお、ソース区間毎の区間写像とその逆写像の定義は、例えば、ユーザによって指定される。また、区間写像が定義されるとその逆写像は式 (12) 及び式 (14) で自動的に定まり、記憶部 105 に格納される。
- [0281] ステップ S 105 : 最後に、仮想表形式データ作成部 101 は、上記のステップ S 101 ～ステップ S 104 で受け付けた情報を仮想表形式データとして記憶部 105 に保存する。
- [0282] <仮想表形式データにおけるソート処理の流れ>

以下では、仮想表形式データにおけるソート処理の流れについて、図 15 を参照しながら説明する。

[0283] ステップ S 2 0 1 : ソート部 1 0 2 は、仮想表形式データの仮想カラムのうち、ソート対象の仮想カラムの仮想転置構造体を用いた仮想インデックスでのソートを実行する。

[0284] ステップ S 2 0 2 : そして、ソート部 1 0 2 は、上記のステップ S 2 0 1 におけるソート結果を記憶部 1 0 5 に保存する。ただし、ソート部 1 0 2 は、ソート結果を保存しなくてもよい。

[0285] <仮想表形式データにおける検索処理の流れ>

以下では、仮想表形式データにおける検索処理の流れについて、図 16 を参照しながら説明する。

[0286] ステップ S 3 0 1 : まず、検索部 1 0 3 は、検索条件を受け付ける。

[0287] ステップ S 3 0 2 : 次に、検索部 1 0 3 は、仮想表形式データの仮想カラムのうち、検索対象の仮想カラムの仮想転置構造体を用いた仮想インデックスでの検索を実行する。

[0288] ステップ S 3 0 3 : そして、検索部 1 0 3 は、上記のステップ S 3 0 2 における検索結果を記憶部 1 0 5 に保存する。

[0289] <仮想表形式データにおける集計処理の流れ>

以下では、仮想表形式データにおける集計処理の流れについて、図 17 を参照しながら説明する。

[0290] ステップ S 4 0 1 : 集計部 1 0 4 は、仮想表形式データの仮想カラムのうち、集計対象の仮想カラムの仮想転置構造体を用いた仮想インデックスで集計を実行する。

[0291] ステップ S 4 0 2 : そして、集計部 1 0 4 は、上記のステップ S 3 0 3 における集計結果を記憶部 1 0 5 に保存する。ただし、集計部 1 0 4 は、集計結果を保存しなくてもよい。

[0292] <まとめ>

以上のように、本実施形態に係るデータ処理装置 10 では、D 5 A 又は既

に定義済の仮想表形式データをソース表形式データとして、割り付け写像により新たな仮想表形式データを定義することができる。しかも、これらの仮想表形式データは仮想インデックスによりソート、検索、集計を高速に実現することが可能となる。

[0293] このため、例えば、元となる表形式データのD 5 Aを起点として様々な仮想表形式データを階層的に作成することが可能となり、これらの仮想表形式データのソート、検索、集計も高速に実行することが可能となる。また、利用者は、自身の利用目的に応じて、既に公開されているD 5 Aや仮想表形式データから新たな仮想表形式データを作成し、公開することも可能となる。

[0294] <補足>

以下、補足として、既存技術ではインデックスの継承ができない一方で、本実施形態ではインデックスの継承が可能な理由について説明する。

[0295] インデックスはインデックスのためのデータ構造とそのデータ構造を使って処理を高速化するアルゴリズムのセットで実現される。転置構造体はソート、検索、集計を高速化するインデックスのためのデータ構造と見做すことができる。転置構造体からカラムの要素を並べ替えたカラムに対する仮想転置構造体を作成することができる。また、複数の転置構造体から仮想転置構造体を作成することができる。更に、複数の仮想転置構造体を階層化して更なる仮想転置構造体を作成することもできる。このように、仮想転置構造体は、階層的にも或いはカラムの要素が並べ替えられても作成することができる。転置構造体を使うインデックスのアルゴリズムは仮想転置構造体にも適用でき、それが仮想インデックスとなる。既存のインデックスは転置構造体のように複数を組み合わせても階層的に組み合わせても成り立つインデックスのためのデータ構造を持たないため、組み合わせたり階層化したりしたときに継承ができないのである。

[0296] 本発明は、具体的に開示された上記の実施形態に限定されるものではなく、請求の範囲の記載から逸脱することなく、種々の変形や変更、既知の技術との組み合わせ等が可能である。

[0297] 本願は、日本国に2023年1月27日に出願された基礎出願2023-010716号及び基礎出願2023-010717号に基づくものであり、その全内容はここに参照をもって援用される。

符号の説明

[0298]	10	データ処理装置
	20	ネットワーク
	101	仮想表形式データ作成部
	102	ソート部
	103	検索部
	104	集計部
	105	記憶部
	201	入力装置
	202	表示装置
	203	外部I/F
	203a	記録媒体
	204	通信I/F
	205	RAM
	206	ROM
	207	補助記憶装置
	208	プロセッサ
	209	バス

請求の範囲

- [請求項1] 1以上の表形式データに含まれる1以上のコラムを持つ第1のデータ構造を所定の写像により変換した第2のデータ構造を持つ仮想コラムで構成される仮想表形式データに対して、前記第2のデータ構造をインデックスとして利用したソート操作、検索操作又は集計操作を行うデータ操作部、
- を有するデータ処理装置。
- [請求項2] 前記1以上の表形式データには、前記第2のデータ構造を持つ仮想コラムで構成される仮想表形式データが含まれる、請求項1に記載のデータ処理装置。
- [請求項3] 前記1以上の表形式データには、前記コラムの値を昇順に保持する配列であるSVLと、前記コラムの値の転置レコード番号を保持する配列であるINVと、前記SVLと前記INVとの対応関係を保持する配列であるACMとで構成されるD5A、及び、前記仮想表形式データ、が含まれる、請求項1に記載のデータ処理装置。
- [請求項4] 前記1以上の表形式データには、前記コラムが更に含まれる、請求項3に記載のデータ処理装置。
- [請求項5] 前記1以上の表形式データには、前記コラムの値を前記SVL上の位置に置き換えた自然数を保持する配列であるNNCが更に含まれる、請求項3に記載のデータ処理装置。
- [請求項6] 前記データ操作部は、
- 前記仮想表形式データのソート対象のコラムに関する前記第2のデータ構造に含まれるINVにより、前記ソート対象のコラムの値を順に取得することで、前記ソート操作を行う、請求項3乃至5の何れか一項に記載のデータ処理装置。
- [請求項7] 前記データ操作部は、
- 前記仮想表形式データの検索対象のコラムに関する前記第2のデータ構造に含まれるINV及びACMと、検索条件として与えられた値

v_0 及び v_1 とを用いて、 $v_0 \leq SVL[i_0]$ を満たす最小の i_0 と $SVL[i_1] \leq i_1$ を満たす最大の i_1 から ($INV[ACM[i_0 - 1]]$, $\dots$, $INV[ACM[i_1] - 1]$) を検索結果として求めることで、前記検索操作を行う、請求項 3 乃至 5 の何れか一項に記載のデータ処理装置。

[請求項 8] 前記データ操作部は、
前記仮想表形式データの集計対象のカラムに関する前記第 2 のデータ構造に含まれる SVL 及び ACM を用いて、 $ACM[i] - ACM[i - 1]$ により、前記仮想表形式データの SVL の i 番目の要素の集計結果を算出することで、前記集計操作を行う、請求項 3 乃至 5 の何れか一項に記載のデータ処理装置。

[請求項 9] 1 以上の表形式データに含まれる 1 以上のカラムを持つ第 1 のデータ構造を所定の写像により変換した第 2 のデータ構造を持つ仮想カラムで構成される仮想表形式データに対して、前記第 2 のデータ構造をインデックスとして利用したソート操作、検索操作又は集計操作を行うデータ操作手順、
をコンピュータが実行するデータ処理方法。

[請求項 10] 1 以上の表形式データに含まれる 1 以上のカラムを持つ第 1 のデータ構造を所定の写像により変換した第 2 のデータ構造を持つ仮想カラムで構成される仮想表形式データに対して、前記第 2 のデータ構造をインデックスとして利用したソート操作、検索操作又は集計操作を行うデータ操作手順、
をコンピュータに実行させるプログラム。

[請求項 11] 作成対象の仮想表形式データのレコード数と、前記仮想表形式データのカラムを表す仮想カラムと、1 以上の表形式データと、前記 1 以上の表形式データに含まれる 1 以上のカラムとの指定を受け付ける受付部と、

前記 1 以上の表形式データに含まれる前記 1 以上のカラムを持つ第

1 のデータ構造を所定の写像により変換した第2 のデータ構造を前記仮想カラムに割り付けることで、前記仮想表形式データを作成する割り付け写像定義部と、

を有するデータ処理装置。

[請求項12] 前記1 以上のカラムの値を、前記仮想表形式データの前記仮想カラム上に表示する表示部、を有する請求項1 1 に記載のデータ処理装置。

[請求項13] 前記写像には、前記第1 のデータ構造と前記第2 のデータ構造との対応関係が1 次関数で表現される1 次関数型写像と、前記第1 のデータ構造と前記第2 のデータ構造との対応関係が前記対応関係の列挙で表現される列挙型写像とが含まれる、請求項1 1 又は1 2 に記載のデータ処理装置。

[請求項14] 異なる第1 のデータ構造に関して、異なる写像によりそれぞれ変換した第2 のデータ構造が衝突するか否かを判定する衝突判定部、を有する請求項1 3 に記載のデータ処理装置。

[請求項15] 前記衝突判定部は、

前記異なる写像のうちの任意の2 つの写像を F_0 、 F_1 として、

前記写像 F_0 の値域と前記写像 F_1 の値域とが同一の仮想カラム上にない場合、前記写像 F_0 の変換先の第2 のデータ構造と前記写像 F_1 の変換先の第2 のデータ構造とは衝突しないと判定し、

前記写像 F_0 の値域を表す区間の開始位置を表すインデックスを v_0 、前記写像 F_0 の定義域を表す区間の長さを w_0 、前記写像 F_1 の値域を表す区間の開始位置を表すインデックスを v_1 としたとき、 $v_0 + w_0 - 1 < v_1$ である場合、前記写像 F_0 の変換先の第2 のデータ構造と前記写像 F_1 の変換先の第2 のデータ構造とは衝突しないと判定し、

前記写像 F_1 の定義域を表す区間の長さを w_1 としたとき、 $v_1 + w_1 - 1 < v_0$ である場合、前記写像 F_0 の変換先の第2 のデータ構造と

前記写像 F_1 の変換先の第2のデータ構造とは衝突しないと判定する、請求項14に記載のデータ処理装置。

[請求項16]

前記衝突判定部は、

前記写像 F_0 と前記写像 F_1 とがいずれも列挙型写像である場合、前記写像 F_0 の値域を表す区間の長さを w_0 、前記写像 F_1 の値域を表す区間の長さを w_1 、 $L_0 = (v_0, \dots, v_0 + w_0 - 1)$ 、 $L_1 = (v_1, \dots, v_1 + w_1 - 1)$ として、 L_0 と L_1 の共通範囲 L_0' 、 L_1' を求め、各 i 、 j に対して $L_0'[i] = L_1'[j]$ とならないとき、前記写像 F_0 の変換先の第2のデータ構造と前記写像 F_1 の変換先の第2のデータ構造とは衝突しないと判定し、

前記写像 F_0 と前記写像 F_1 とがいずれも1次関数型写像である場合、前記写像 F_0 の値域と前記写像 F_1 の値域との共通部分を値域とする $y_0 = a_0 \times x_0 + b_0$ ； $x_0 = 0, 1, 2, \dots, N_0 - 1$ で前記写像 F_0 を記述し直すと共に、前記共通部分を値域とする $y_1 = a_1 \times x_1 + b_1$ ； $x_1 = 0, 1, 2, \dots, N_1 - 1$ で前記写像 F_1 を記述し直し、 $x_0 \leq \text{MinInt}(N_0 - 1, \text{LCM}(a_0, a_1) \text{div } a_0)$ と $x_1 \leq \text{MinInt}(N_1 - 1, \text{LCM}(a_0, a_1) \text{div } a_1)$ の x_0 、 x_1 に対して $y_0 = y_1$ とならないとき、前記写像 F_0 の変換先の第2のデータ構造と前記写像 F_1 の変換先の第2のデータ構造とは衝突しないと判定し、

前記写像 F_0 が1次関数型写像、前記写像 F_1 が列挙型写像である場合、前記写像 F_0 の値域と前記写像 F_1 の値域との共通部分を値域とする $y_0(n) = a_0 \times n + b_0$ ； $x_0 = 0, 1, 2, \dots, N - 1$ で前記写像 F_0 を記述し直すと共に、前記共通部分を値域とする $y_1(m) = (v_1, \dots, v_1 + w_1 - 1)[m]$ で前記写像 F_1 を記述し直し、各 n 、 m に対して $y_0(n) = y_1(m)$ とならないとき、前記写像 F_0 の変換先の第2のデータ構造と前記写像 F_1 の変換先の第2のデータ構造とは衝突しないと判定する、請求項15に記載の

データ処理装置。

[請求項17] 前記第1のデータ構造及び前記第2のデータ構造は、転置レコード番号を要素とする配列を表す1N Vが少なくとも含まれるデータ構造である、請求項16に記載のデータ処理装置。

[請求項18] 作成対象の仮想表形式データのレコード数と、前記仮想表形式データの列を表す仮想列と、1以上の表形式データと、前記1以上の表形式データに含まれる1以上の列との指定を受け付ける受付手順と、

前記1以上の表形式データに含まれる前記1以上の列が持つ第1のデータ構造を所定の写像により変換した第2のデータ構造を前記仮想列に割り付けることで、前記仮想表形式データを作成する割り付け写像定義手順と、

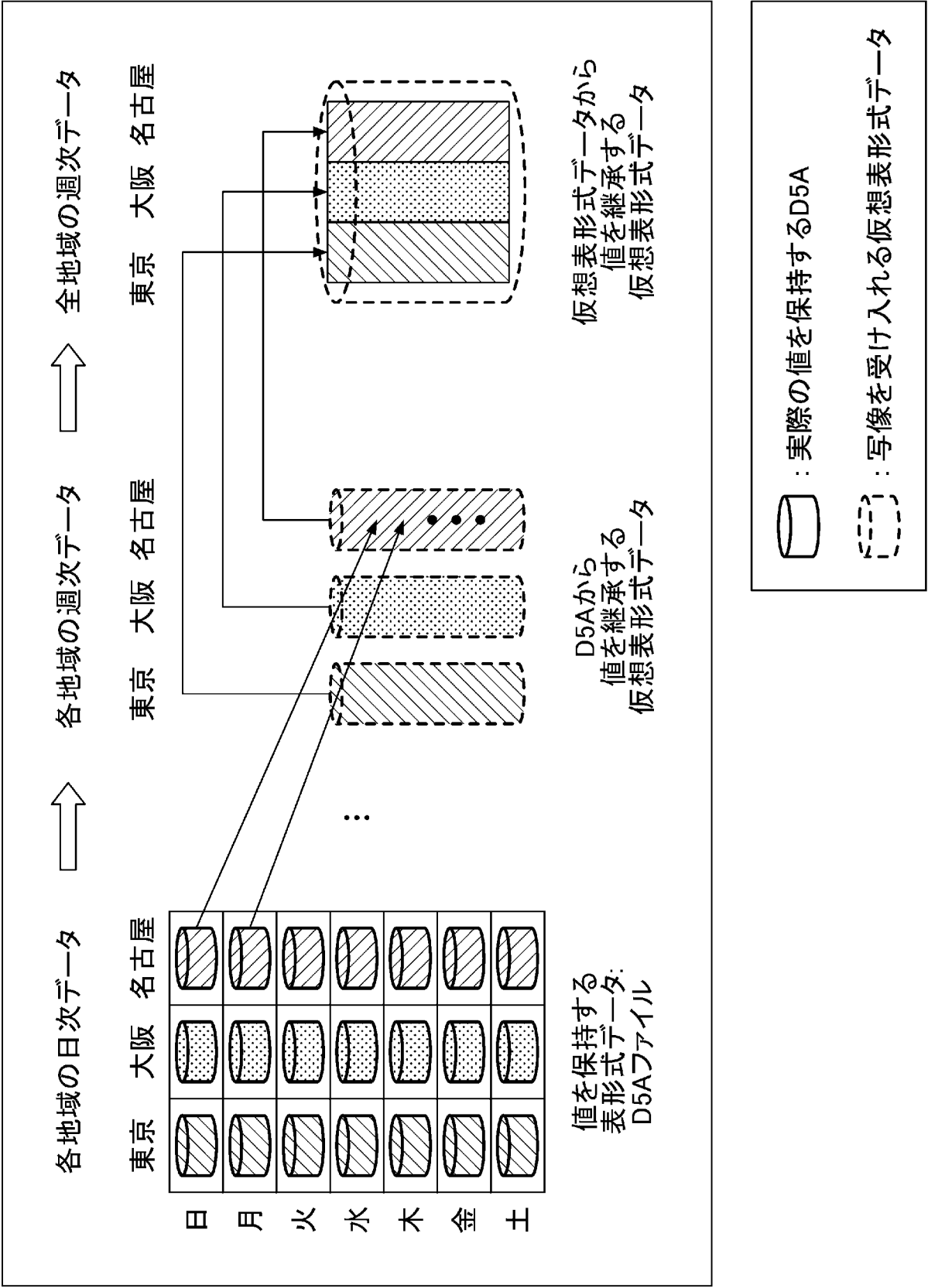
をコンピュータが実行するデータ処理方法。

[請求項19] 作成対象の仮想表形式データのレコード数と、前記仮想表形式データの列を表す仮想列と、1以上の表形式データと、前記1以上の表形式データに含まれる1以上の列との指定を受け付ける受付手順と、

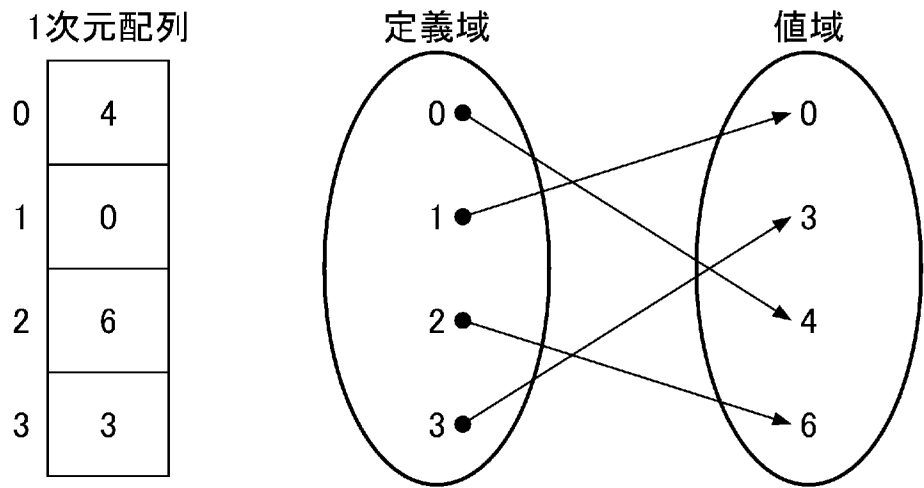
前記1以上の表形式データに含まれる前記1以上の列が持つ第1のデータ構造を所定の写像により変換した第2のデータ構造を前記仮想列に割り付けることで、前記仮想表形式データを作成する割り付け写像定義手順と、

をコンピュータに実行させるプログラム。

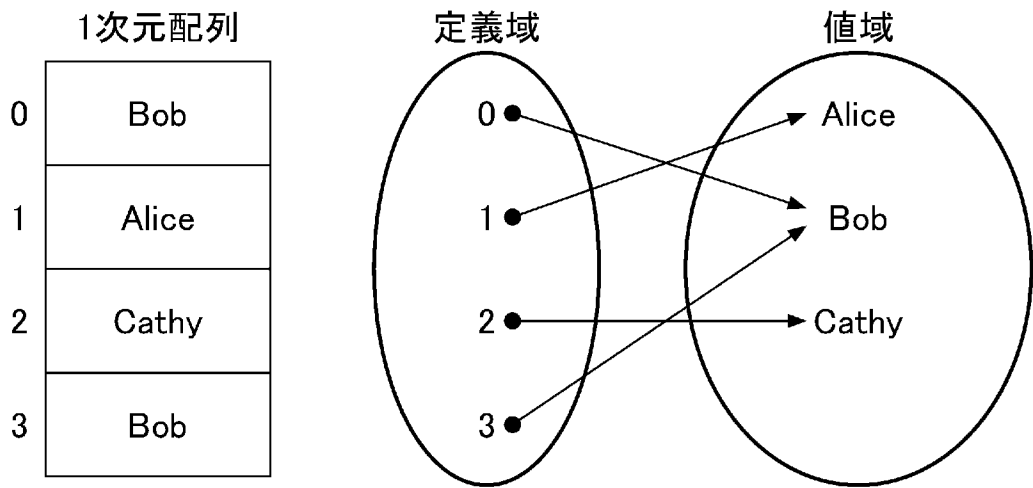
[図1]



[図2]



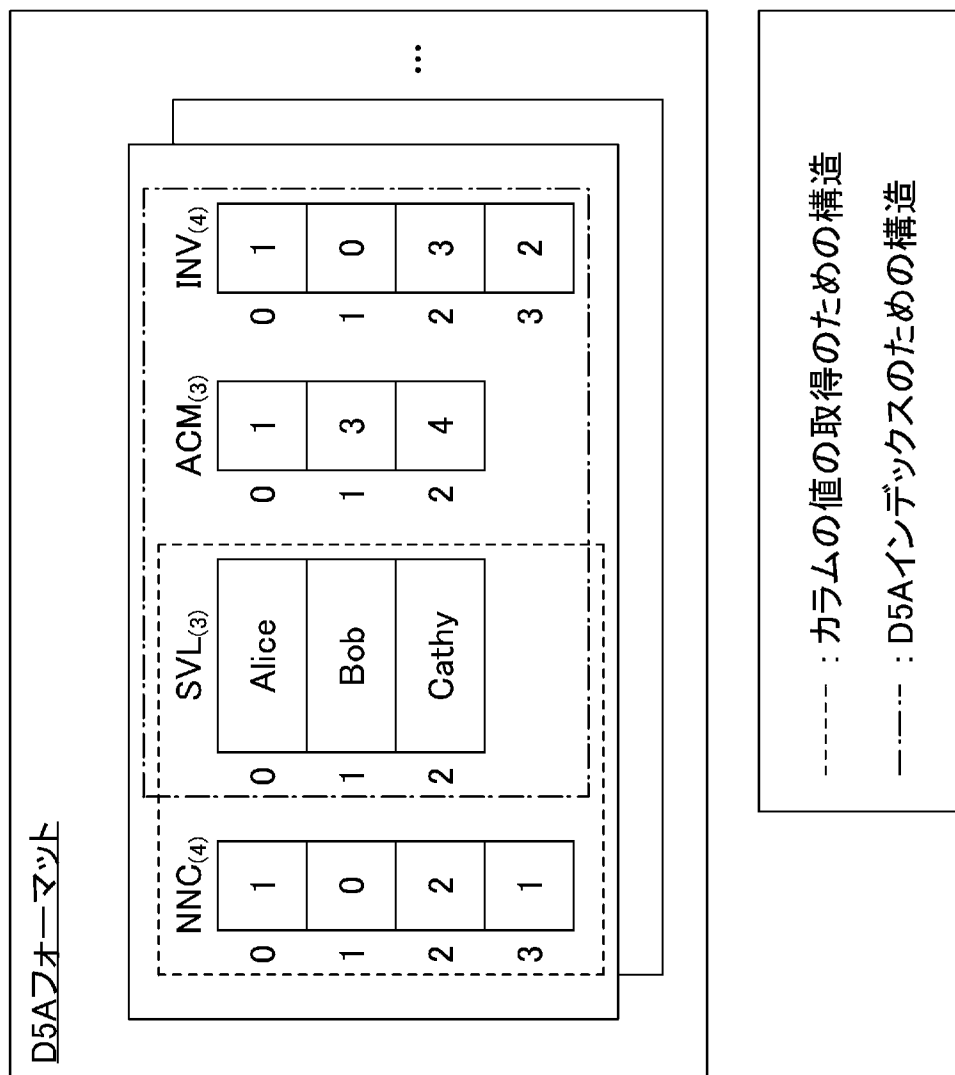
[図3]



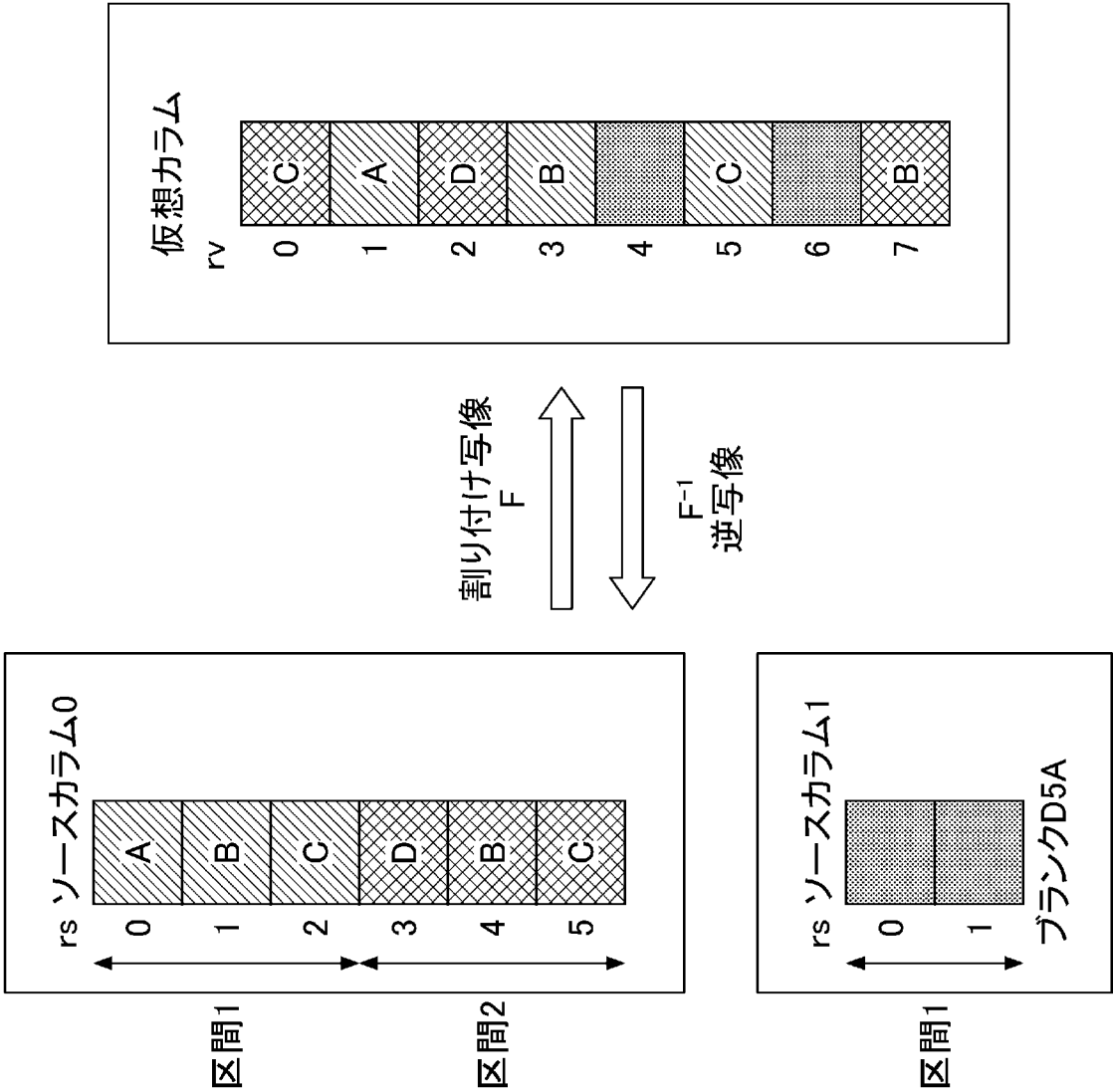
CSV形式

0	Bob		
1	Alice		
2	Cathy		
3	Bob		

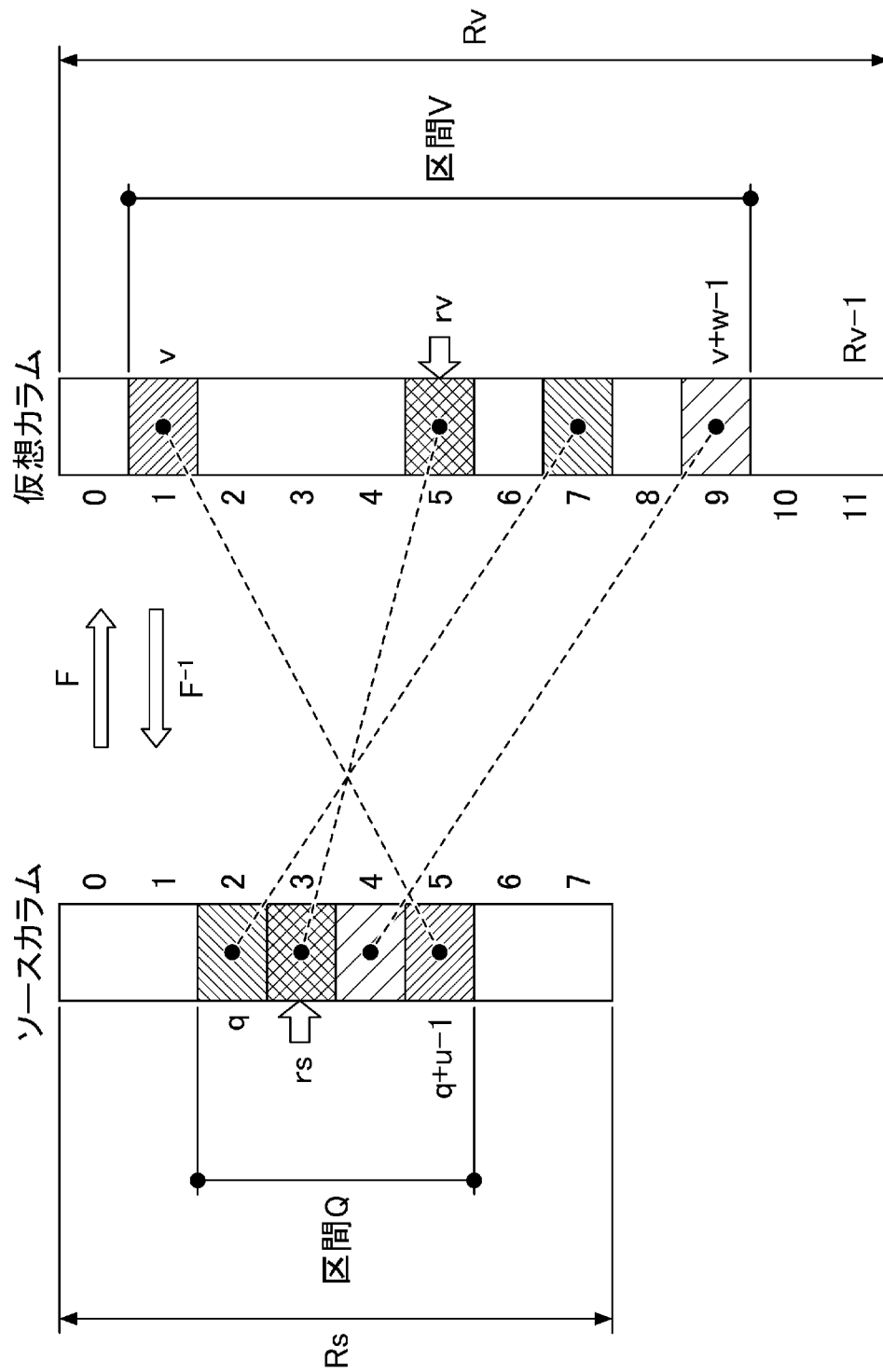
...



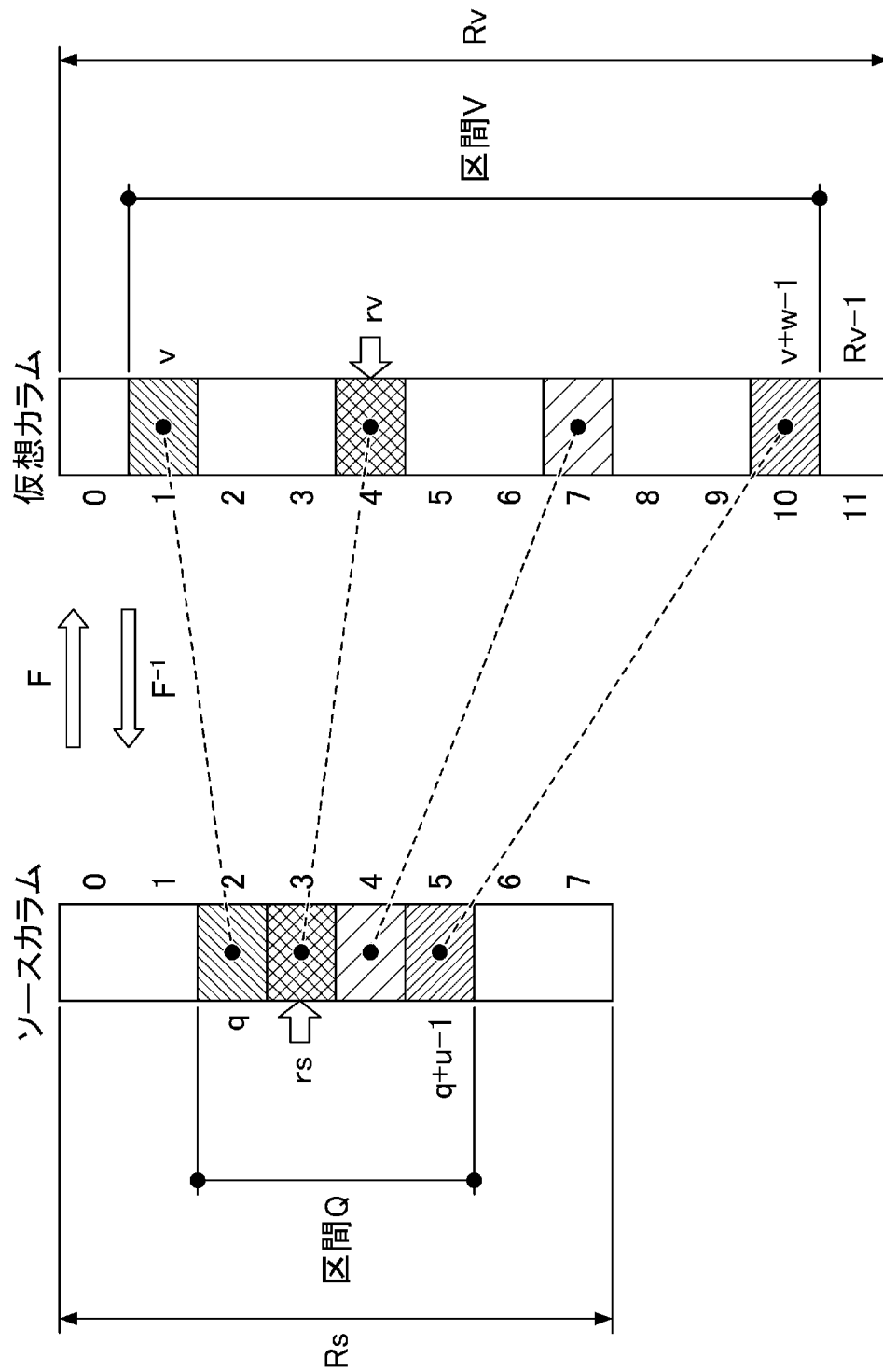
[図5]



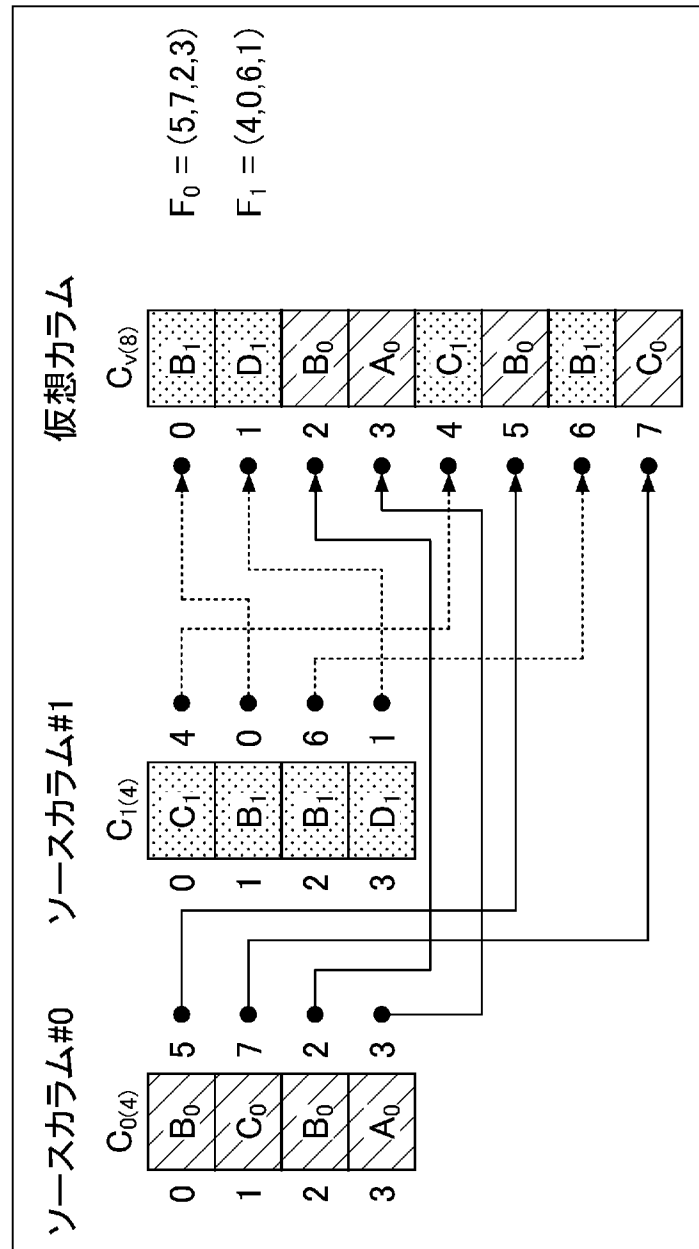
[図6]



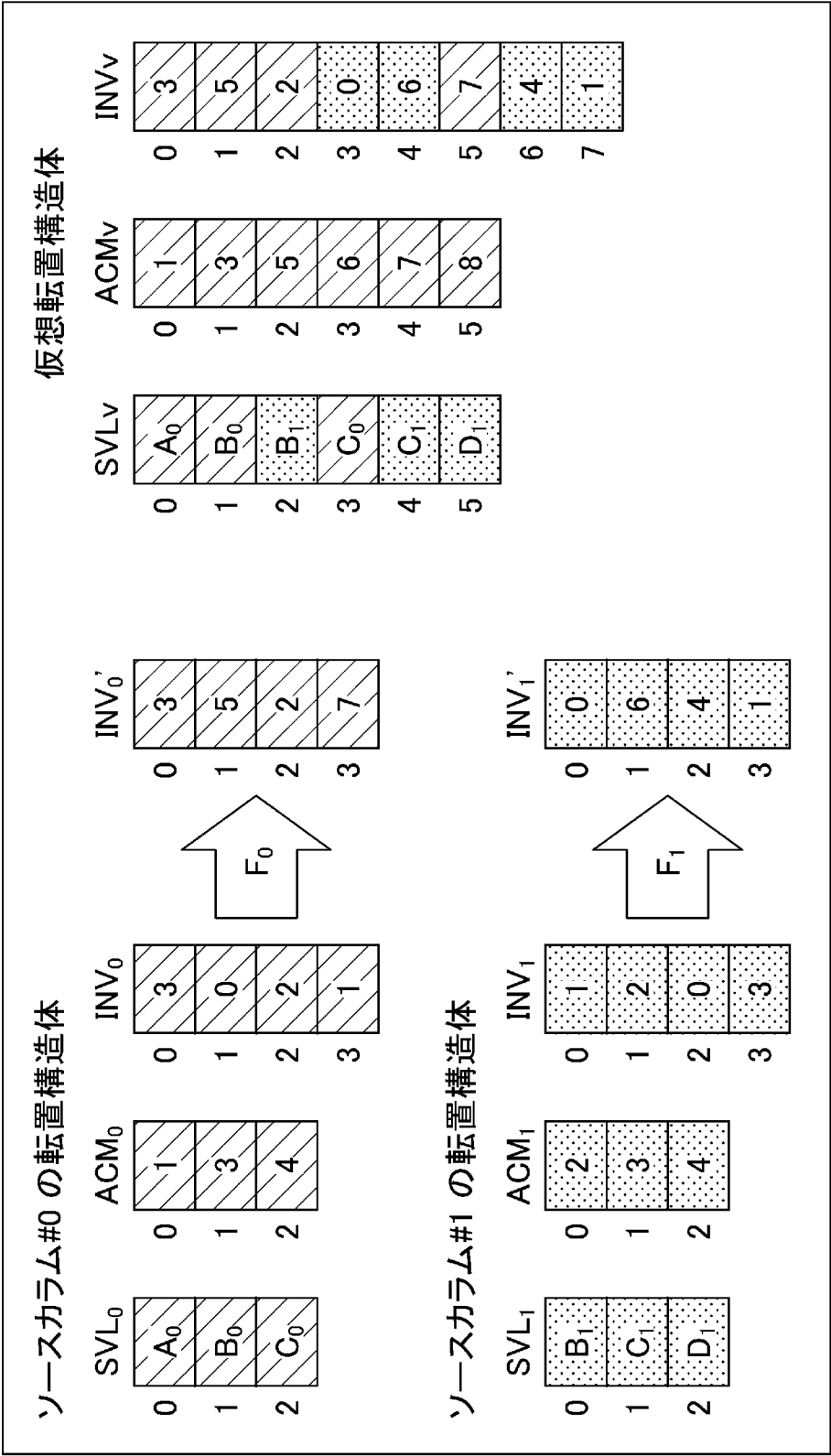
[図7]



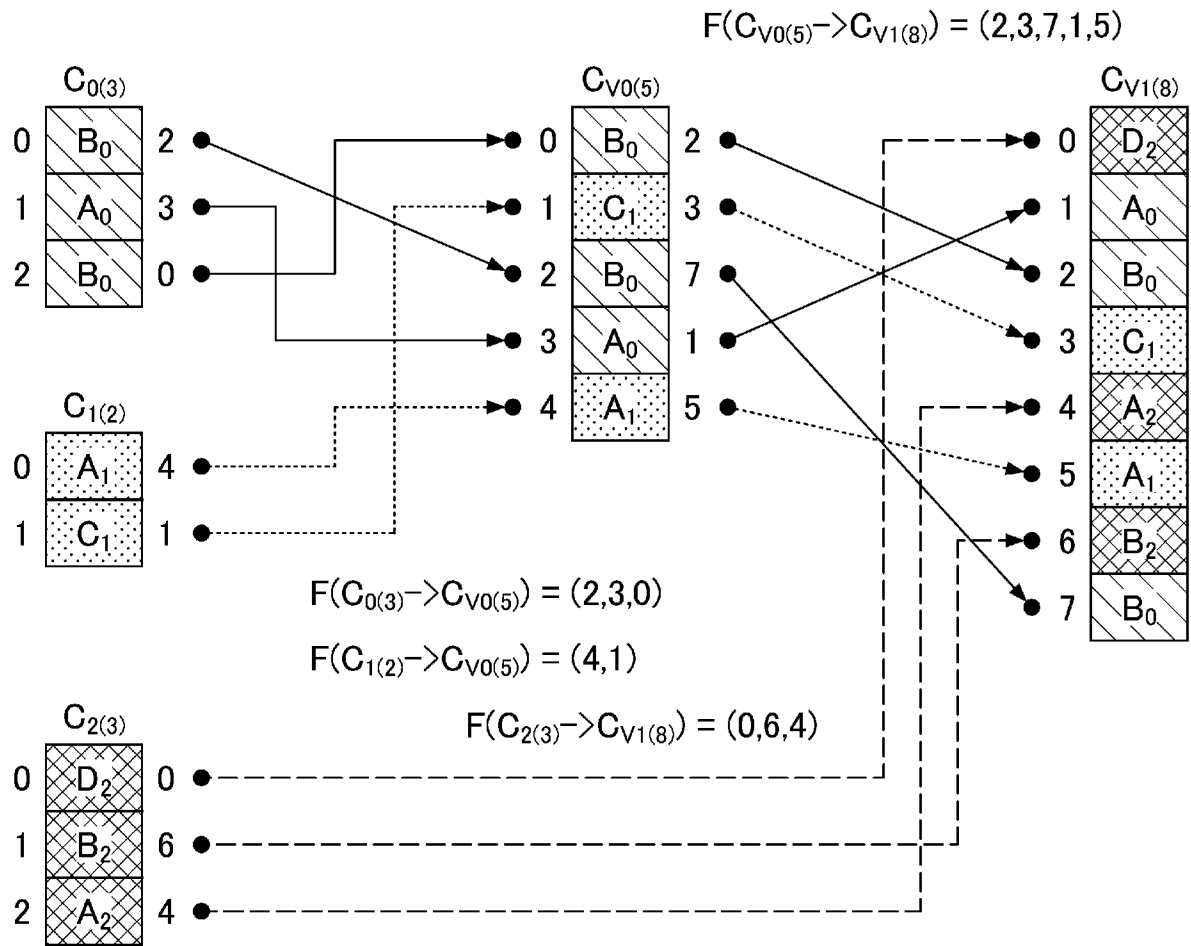
[図8]



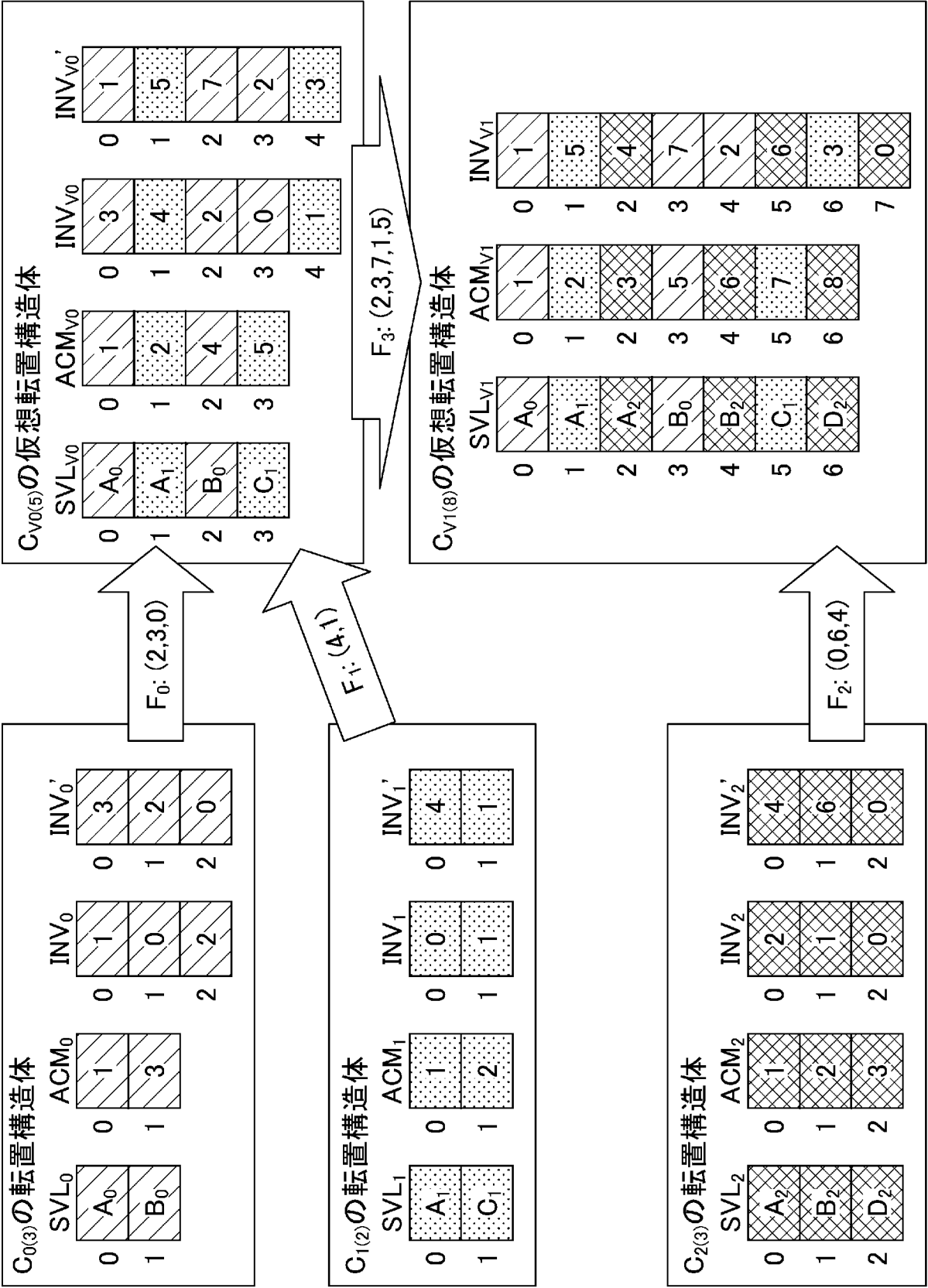
[図9]



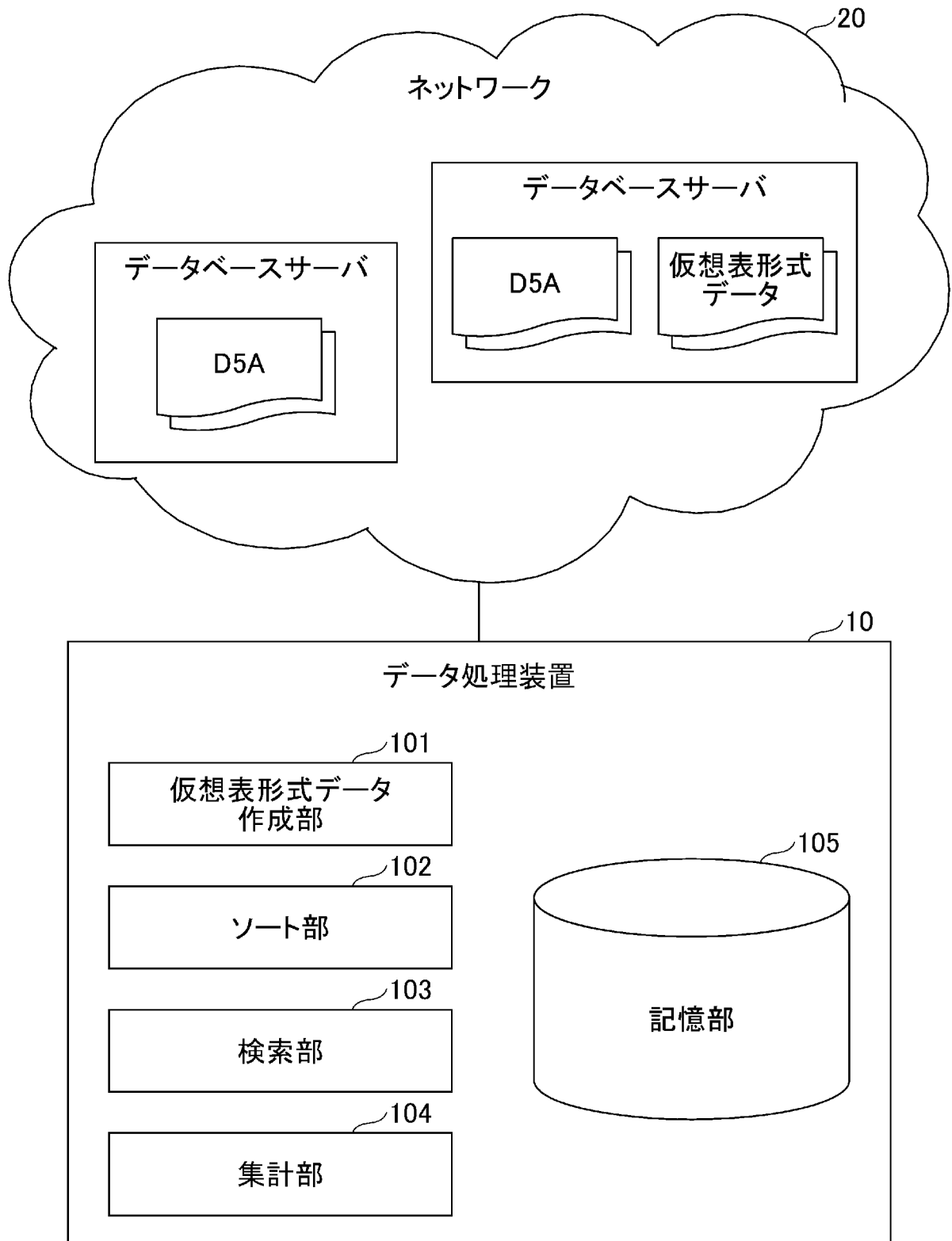
[図10]



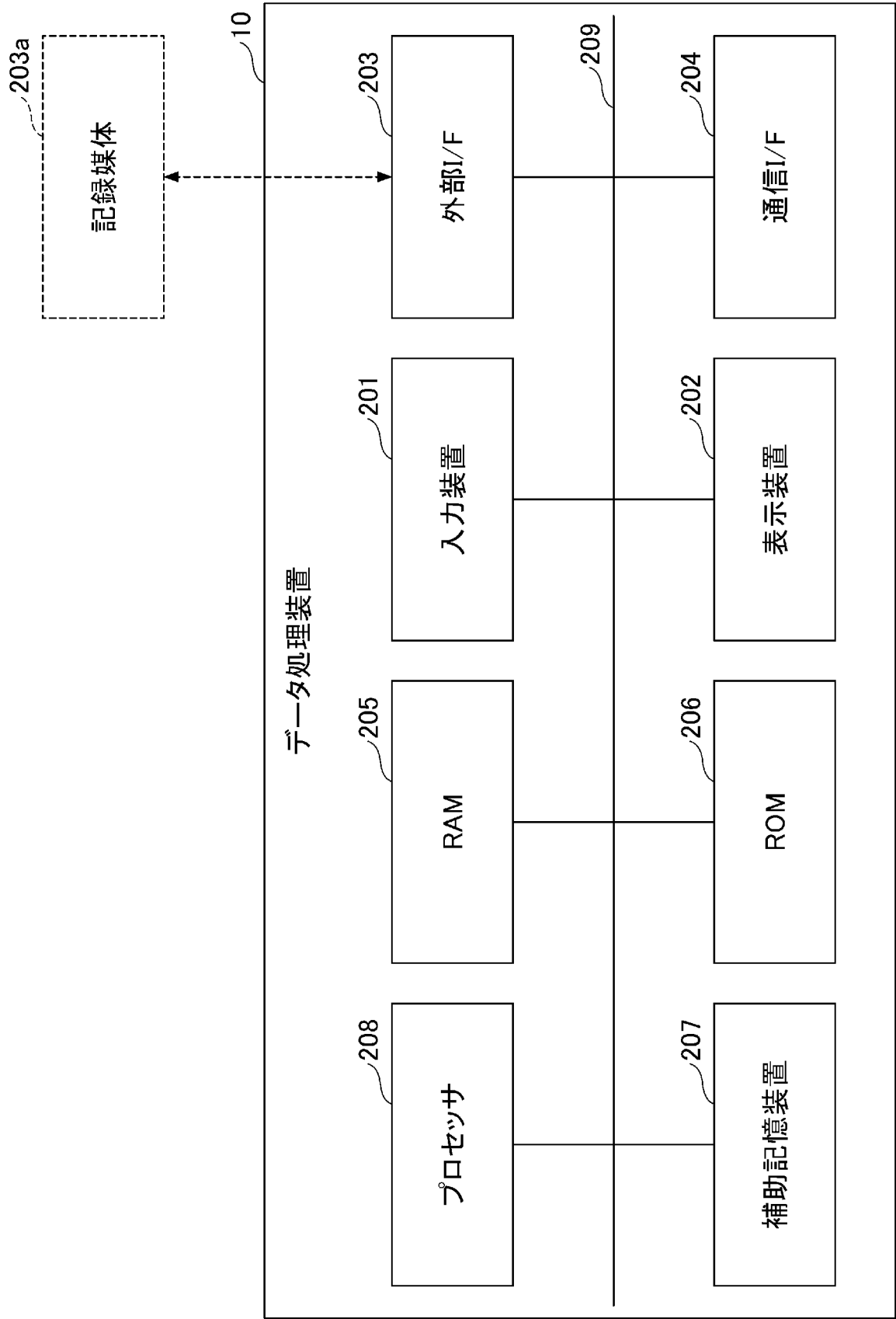
[図11]



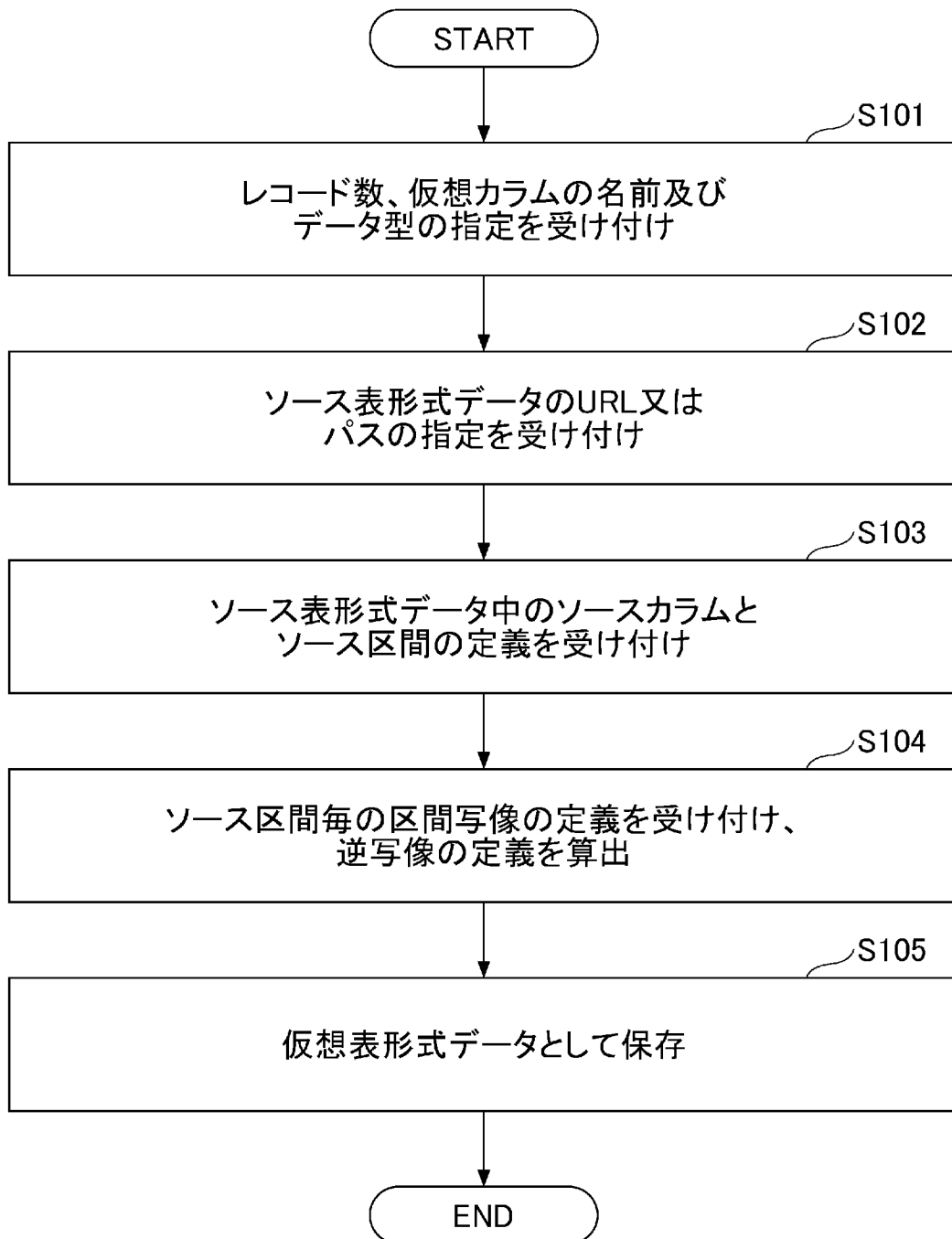
[図12]



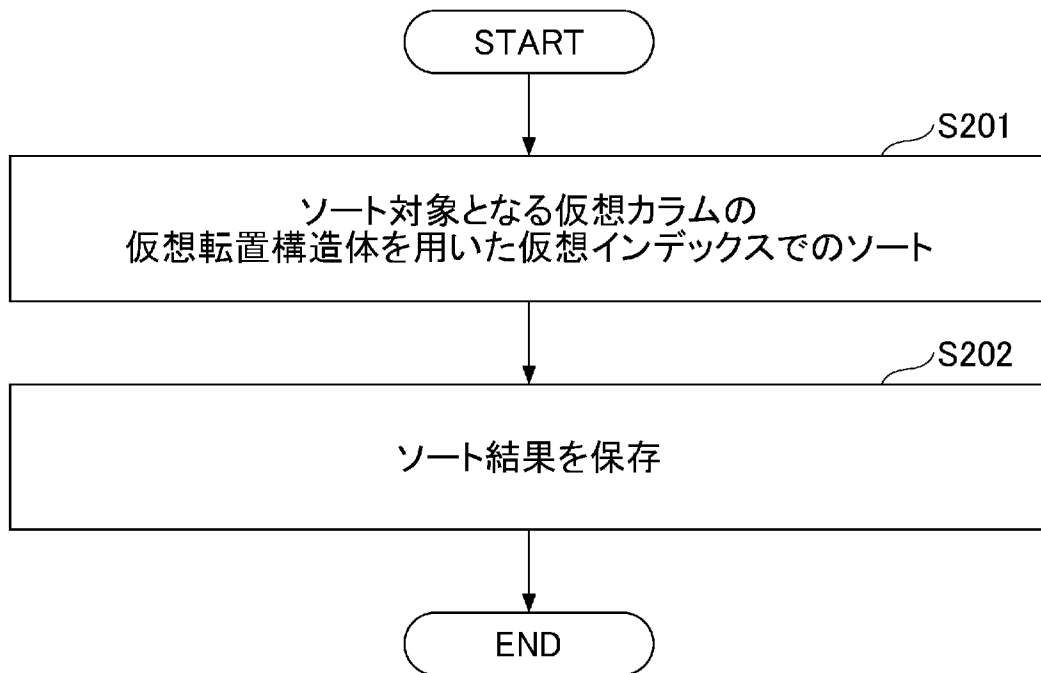
[図13]



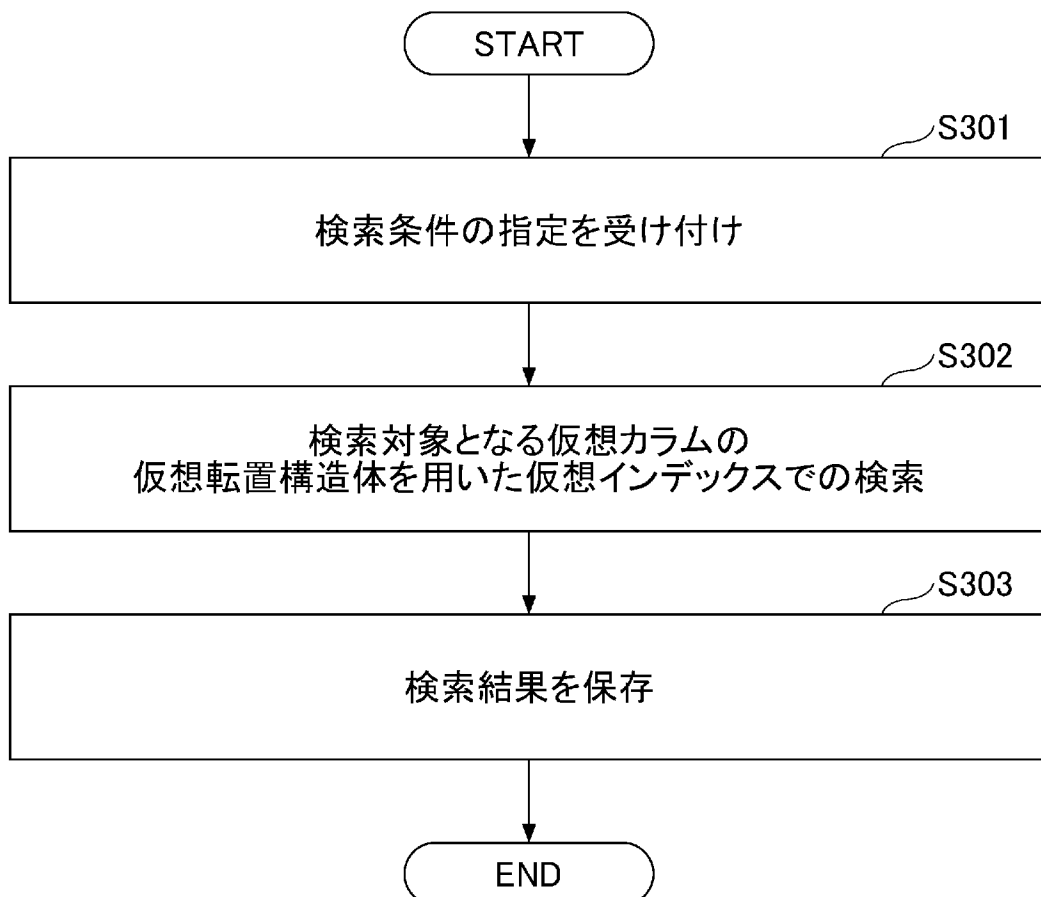
[図14]



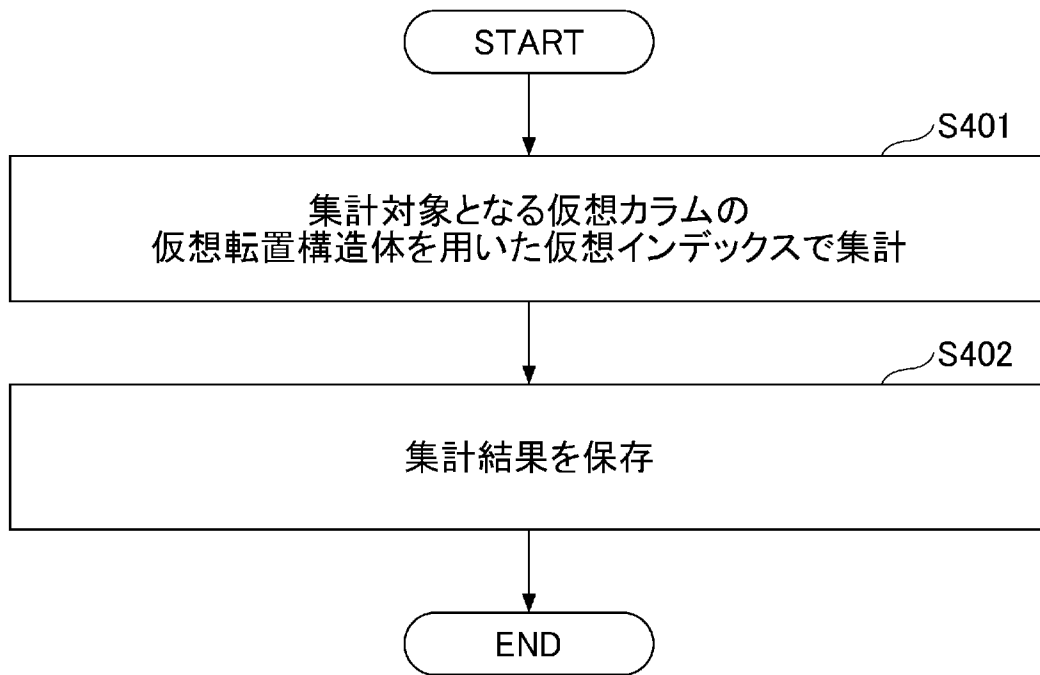
[図15]



[図16]



[図17]



INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP2024/000399

A. CLASSIFICATION OF SUBJECT MATTER G06F 16/22 (2019.01)i FI: G06F16/22 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) G06F16/22 Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched Published examined utility model applications of Japan 1922-1996 Published unexamined utility model applications of Japan 1971-2024 Registered utility model specifications of Japan 1996-2024 Published registered utility model applications of Japan 1994-2024 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.
X	JP 2013-117873 A (HITACHI SYSTEMS LTD.) 13 June 2013 (2013-06-13) paragraph [0018]	1-2, 9-12, 18-19
Y	paragraph [0018]	3-8
A	paragraph [0018]	13-17
Y	WO 2023/276162 A1 (FURUSHO, Shinji) 05 January 2023 (2023-01-05) paragraphs [0011], [0012], [0075], [0076]	3-8
A	JP 2015-207026 A (TURBO DATA LAB KK) 19 November 2015 (2015-11-19) entire text, all drawings	1-19
☐ 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 “D” document cited by the applicant in the international application “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 22 March 2024		Date of mailing of the international search report 02 April 2024
Name and mailing address of the ISA/JP Japan Patent Office (ISA/JP) 3-4-3 Kasumigaseki, Chiyoda-ku, Tokyo 100-8915 Japan		Authorized officer Telephone No.

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/JP2024/000399

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
JP	2013-117873	A	13 June 2013	(Family: none)	
WO	2023/276162	A1	05 January 2023	(Family: none)	
JP	2015-207026	A	19 November 2015	WO	2014/034383 A1

A. 発明の属する分野の分類（国際特許分類（IPC））

G06F 16/22(2019.01)i

FI: G06F16/22

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））

G06F16/22

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 - 2024年

日本国実用新案登録公報

1996 - 2024年

日本国登録実用新案公報

1994 - 2024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
X	JP 2013-117873 A（株式会社日立システムズ）13.06.2013（2013 - 06 - 13） [0018]	1-2, 9-12, 18-19
Y	[0018]	3-8
A	[0018]	13-17
Y	WO 2023/276162 A1（古庄 晋二）05.01.2023（2023 - 01 - 05） [0011]-[0012], [0075]-[0076]	3-8
A	JP 2015-207026 A（株式会社ターボデーターラボラトリー）19.11.2015（2015 - 11 - 19） 全文全文	1-19

☐ 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

“D” 国際出願で出願人が先行技術文献として記載した文献

“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抵触するものではなく、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

22. 03. 2024

国際調査報告の発送日

02. 04. 2024

名称及びあて先

日本国特許庁(ISA/JP)

〒100-8915

日本国

東京都千代田区霞が関三丁目4番3号

権限のある職員（特許庁審査官）

原 秀人 5N 9644

電話番号 03-3581-1101 内線 3586

国際調査報告
パテントファミリーに関する情報

国際出願番号
PCT/JP2024/000399

引用文献			公表日	パテントファミリー文献			公表日
JP	2013-117873	A	13.06.2013	(ファミリーなし)			
WO	2023/276162	A1	05.01.2023	(ファミリーなし)			
JP	2015-207026	A	19.11.2015	WO	2014/034383	A1	