(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6427592号
(P6427592)
(45)発行日　平成30年11月21日(2018.11.21)　(24)登録日　平成30年11月2日(2018.11.2)

(51)Int.Cl.　F I
G06F 17/30 (2006.01)　G06F 17/30 180D
G06F 17/30 414Z

請求項の数 21　(全 22 頁)

(21)出願番号　特願2016-553899(P2016-553899)
(86)(22)出願日　平成27年2月19日(2015.2.19)
(65)公表番号　特表2017-515183(P2017-515183A)
(43)公表日　平成29年6月8日(2017.6.8)
(86)国際出願番号　PCT/US2015/016517
(87)国際公開番号　WO2015/134193
(87)国際公開日　平成27年9月11日(2015.9.11)
審査請求日　平成29年12月20日(2017.12.20)
(31)優先権主張番号　61/949,477
(32)優先日　平成26年3月7日(2014.3.7)
(33)優先権主張国　米国(US)

(73)特許権者　509123208
アビニシオ　テクノロジー　エルエルシー
アメリカ合衆国　０２４２１　マサチューセッツ州　レキシントン　スプリング　ストリート　２０１
(74)代理人　100107984
弁理士　廣田　雅紀
(74)代理人　100102255
弁理士　小澤　誠次
(74)代理人　100096482
弁理士　東海　裕作
(74)代理人　100188352
弁理士　松田　一弘
(74)代理人　100131093
弁理士　堀内　真

最終頁に続く

(54)【発明の名称】データ型に関連するデータプロファイリング操作の管理

(57)【特許請求の範囲】
【請求項１】
コンピューティングシステムでデータを処理するための方法であって、
複数のフィールドのそれぞれのフィールドに関する１又は２以上の値をそれぞれが有する複数のレコードを前記コンピューティングシステムの入力デバイス又はポートを介して受信するステップと、
１又は２以上のデータ型のそれぞれを少なくとも１つの識別子と関連付けるデータ型情報を前記コンピューティングシステムのストレージ媒体に記憶するステップと、
前記レコードからの複数のデータ値を前記コンピューティングシステムの少なくとも１つのプロセッサを用いて処理するステップであって、
前記レコードから複数のデータユニットを生じさせることであって、各データユニットが、前記フィールドのうちの１つを一意に特定するフィールド識別子及び前記レコードのうちの１つからのバイナリ値を含み、前記バイナリ値が、前記フィールド識別子によって特定される前記レコードの前記フィールドから抽出される、生じさせること、
複数の前記データユニットからのバイナリ値についての情報を集約すること、
前記フィールドのうちの１又は２以上のそれぞれに関するエントリのリストを生じさせることであって、前記エントリのうちの少なくとも一部が、それぞれ、前記バイナリ値のうちの１つ、及び複数の前記データユニットから集約された前記バイナリ値についての情報を含む、生じさせること、
前記データ型情報から第１の識別子に関連するデータ型を取り出し、前記取り出され

たデータ型を前記リストのうちの１つのエントリに含まれる少なくとも１つのバイナリ値と関連付けること、並びに
　複数の前記データユニットからのバイナリ値についての情報を集約することの後、前記フィールドに現れる特定のバイナリ値の取り出されたデータ型に少なくとも部分的に基づいて前記フィールドのうちの少なくとも１つに関するプロファイル情報を生じさせること
を含む、処理するステップと
を含む、前記方法。
【請求項２】
フィールド識別子によって特定されるレコードのフィールドから抽出されたバイナリ値が、ビットの型なしシーケンスとして抽出され、前記フィールドに現れる特定のバイナリ値の取り出されたデータ型に少なくとも部分的に基づいてフィールドのうちの少なくとも１つに関するプロファイル情報を生じさせることが、前記ビットの型なしシーケンスを前記取り出されたデータ型を有する型付きデータ値として再解釈することを含む、請求項１に記載の方法。
【請求項３】
プロファイル情報が、レコードからの複数のデータ値の元のデータ型に応じて決まる型に依存するプロファイリングの結果を含む、請求項２に記載の方法。
【請求項４】
複数のデータユニットからのバイナリ値についての情報を集約することが、複数のデータユニットからのバイナリ値をエントリのリストのバイナリ値と比較して、前記バイナリ値の間に一致があるかどうかを判定することを含む、請求項１に記載の方法。
【請求項５】
複数のデータユニットから集約されたバイナリ値についての情報が、バイナリ値を比較するときに一致が判定される度にインクリメントされる一致した前記バイナリ値の総カウントを含む、請求項４に記載の方法。
【請求項６】
第１のバイナリ値と第２のバイナリ値との間の一致が、前記第１のバイナリ値を含むビットのシーケンスが前記第２のバイナリ値を含むビットのシーケンスと同一であることに対応する、請求項４に記載の方法。
【請求項７】
データ型情報が、１又は２以上のデータ型のそれぞれをフィールド識別子のうちの少なくとも１つと関連付ける、請求項１に記載の方法。
【請求項８】
データ型情報から第１の識別子に関連するデータ型を取り出すことが、第１のフィールド識別子に関連するデータ型を取り出すことを含む、請求項７に記載の方法。
【請求項９】
各データユニットが、フィールド識別子のうちの１つ、レコードのうちの１つからのバイナリ値、及びデータ型のうちの１つを一意に特定するデータ型識別子を含む、請求項１に記載の方法。
【請求項１０】
データ型情報が、１又は２以上のデータ型のそれぞれをデータ型識別子のうちの少なくとも１つと関連付ける、請求項９に記載の方法。
【請求項１１】
データ型情報から第１の識別子に関連するデータ型を取り出すことが、第１のデータ型識別子に関連するデータ型を取り出すことを含む、請求項１０に記載の方法。
【請求項１２】
取り出されたデータ型をリストのうちの１つのエントリに含まれる少なくとも１つのバイナリ値と関連付けることが、前記取り出されたデータ型を有するローカル変数をインスタンス化することと、前記インスタンス化された変数を前記エントリに含まれる前記バイ

ナリ値に基づく値に初期化することとを含む、請求項１に記載の方法。
【請求項１３】
取り出されたデータ型をリストのうちの１つのエントリに含まれる少なくとも１つのバイナリ値と関連付けることが、前記取り出されたデータ型に関連するポインタを前記エントリに含まれる前記バイナリ値が記憶されるメモリ位置に設定することを含む、請求項１に記載の方法。
【請求項１４】
各データユニットが、フィールド識別子のうちの１つ、レコードのうちの１つからのバイナリ値、及び前記バイナリ値の長さのインジケータを含む、請求項１に記載の方法。
【請求項１５】
長さインジケータが、バイナリ値へのプレフィックスとして記憶される、請求項１４に記載の方法。
【請求項１６】
複数のレコードに関連するレコードフォーマット情報をコンピューティングシステムの入力デバイス又はポートを介して受信するステップをさらに含む、請求項１に記載の方法。
【請求項１７】
データ型情報が、受信されたレコードフォーマット情報に少なくとも部分的に基づいて生じさせられる、請求項１６に記載の方法。
【請求項１８】
処理するステップが、第１のフィールドに関するリストのうちの第１のリストのそれぞれの異なるエントリからのバイナリ値を前記バイナリ値に関連する取り出されたデータ型から目標のデータ型に変換することと、第２のフィールドに関するリストのうちの第２のリストのそれぞれの異なるエントリからのバイナリ値を前記バイナリ値に関連する取り出されたデータ型から同じ目標のデータ型に変換することとをさらに含む、請求項１に記載の方法。
【請求項１９】
コンピュータ可読媒体に非一時的形態で記憶されたソフトウェアであって、コンピューティングシステムに、
複数のフィールドのそれぞれのフィールドに関する１又は２以上の値をそれぞれが有する複数のレコードを前記コンピューティングシステムの入力デバイス又はポートを介して受信することと、
１又は２以上のデータ型のそれぞれを少なくとも１つの識別子と関連付けるデータ型情報を前記コンピューティングシステムのストレージ媒体に記憶することと、
前記レコードからの複数のデータ値を前記コンピューティングシステムの少なくとも１つのプロセッサを用いて処理することであって、
前記レコードから複数のデータユニットを生じさせることであって、各データユニットが、前記フィールドのうちの１つを一意に特定するフィールド識別子及び前記レコードのうちの１つからのバイナリ値を含み、前記バイナリ値が、前記フィールド識別子によって特定される前記レコードの前記フィールドから抽出される、生じさせること、
複数の前記データユニットからのバイナリ値についての情報を集約すること、
前記フィールドのうちの１又は２以上のそれぞれに関するエントリのリストを生じさせることであって、前記エントリのうちの少なくとも一部が、それぞれ、前記バイナリ値のうちの１つ、及び複数の前記データユニットから集約された前記バイナリ値についての情報を含む、生じさせること、
前記データ型情報から第１の識別子に関連するデータ型を取り出し、前記取り出されたデータ型を前記リストのうちの１つのエントリに含まれる少なくとも１つのバイナリ値と関連付けること、並びに
複数の前記データユニットからのバイナリ値についての情報を集約することの後、前記フィールドに現れる特定のバイナリ値の取り出されたデータ型に少なくとも部分的に基

づいて前記フィールドのうちの少なくとも１つに関するプロファイル情報を生じさせること
を含む、処理することと
を行わせるための命令を含む、前記ソフトウェア。
【請求項２０】
複数のフィールドのそれぞれのフィールドに関する１又は２以上の値をそれぞれが有する複数のレコードを受信するように構成されたコンピューティングシステムの入力デバイス又はポートと、
１又は２以上のデータ型のそれぞれを少なくとも１つの識別子と関連付けるデータ型情報を記憶するように構成された前記コンピューティングシステムのストレージ媒体と、
前記レコードからの複数のデータ値を処理するように構成された前記コンピューティングシステムの少なくとも１つのプロセッサであって、前記処理が、
前記レコードから複数のデータユニットを生じさせることであって、各データユニットが、前記フィールドのうちの１つを一意に特定するフィールド識別子及び前記レコードのうちの１つからのバイナリ値を含み、前記バイナリ値が、前記フィールド識別子によって特定される前記レコードの前記フィールドから抽出される、生じさせること、
複数の前記データユニットからのバイナリ値についての情報を集約すること、
前記フィールドのうちの１又は２以上のそれぞれに関するエントリのリストを生じさせることであって、前記エントリのうちの少なくとも一部が、それぞれ、前記バイナリ値のうちの１つ、及び複数の前記データユニットから集約された前記バイナリ値についての情報を含む、生じさせること、
前記データ型情報から第１の識別子に関連するデータ型を取り出し、前記取り出されたデータ型を前記リストのうちの１つのエントリに含まれる少なくとも１つのバイナリ値と関連付けること、並びに
複数の前記データユニットからのバイナリ値についての情報を集約することの後、前記フィールドに現れる特定のバイナリ値の取り出されたデータ型に少なくとも部分的に基づいて前記フィールドのうちの少なくとも１つに関するプロファイル情報を生じさせること
を含む、少なくとも１つのプロセッサと
を含む、前記コンピューティングシステム。
【請求項２１】
複数のフィールドのそれぞれのフィールドに関する１又は２以上の値をそれぞれが有する複数のレコードを受信するための手段と、
１又は２以上のデータ型のそれぞれを少なくとも１つの識別子と関連付けるデータ型情報を記憶するための手段と、
前記レコードからの複数のデータ値を処理するための手段であって、前記処理が、
前記レコードから複数のデータユニットを生じさせることであって、各データユニットが、前記フィールドのうちの１つを一意に特定するフィールド識別子及び前記レコードのうちの１つからのバイナリ値を含み、前記バイナリ値が、前記フィールド識別子によって特定される前記レコードの前記フィールドから抽出される、生じさせること、
複数の前記データユニットからのバイナリ値についての情報を集約すること、
前記フィールドのうちの１又は２以上のそれぞれに関するエントリのリストを生じさせることであって、前記エントリのうちの少なくとも一部が、それぞれ、前記バイナリ値のうちの１つ、及び複数の前記データユニットから集約された前記バイナリ値についての情報を含む、生じさせること、
前記データ型情報から第１の識別子に関連するデータ型を取り出し、前記取り出されたデータ型を前記リストのうちの１つのエントリに含まれる少なくとも１つのバイナリ値と関連付けること、並びに
複数の前記データユニットからのバイナリ値についての情報を集約することの後、前記フィールドに現れる特定のバイナリ値の取り出されたデータ型に少なくとも部分的に基

づいて前記フィールドのうちの少なくとも１つに関するプロファイル情報を生じさせること
を含む、手段と
を含む、コンピューティングシステム。
【発明の詳細な説明】
【技術分野】
【０００１】
関連出願の相互参照
本出願は、２０１４年３月７日に出願した米国特許出願第６１／９４９，４７７号明細書の優先権を主張するものである。
【０００２】
この説明は、データ型に関連するデータプロファイリング操作を管理することに関する。
【背景技術】
【０００３】
データベース又はその他の情報管理システムは、多くの場合、さまざまな特徴が知られ得ないデータセットを含む。例えば、データセットに関する値の範囲若しくは典型的な値、データセット内の異なるフィールドの間の関係、又は異なるフィールドの値の間の関数従属性が、未知である可能性がある。データプロファイリングは、そのような特徴を判定するためにデータセットを調べることをともなう可能性がある。データプロファイリングのための一部の技術は、データプロファイリングジョブについての情報を受信することと、データプロファイリングジョブを実行することと、そして、データプロファイリングに関わるさまざまな処理ステップを実行することがどれくらい時間がかかるかに基づく遅延の後に結果を返すこととを含む。多くの処理時間をともなう可能性があるステップのうちの１つは、さらなる処理を容易にするために、データセットのレコード内に現れる値のデータ型を所定の又は「正規の」データ型に変更することをともなう「正規化」である。例えば、正規化は、値を人が読める文字列表現に変換することを含み得る。
【発明の概要】
【課題を解決するための手段】
【０００４】
一態様においては、概して、コンピューティングシステムでデータを処理するための方法が、複数のフィールドのそれぞれのフィールドに関する１又は２以上の値をそれぞれが有する複数のレコードをコンピューティングシステムの入力デバイス又はポートを介して受信するステップを含む。前記方法は、１又は２以上のデータ型のそれぞれを少なくとも１つの識別子と関連付けるデータ型情報をコンピューティングシステムのストレージ媒体に記憶するステップを含む。前記方法は、レコードからの複数のデータ値をコンピューティングシステムの少なくとも１つのプロセッサを用いて処理するステップを含む。前記処理するステップは、レコードから複数のデータユニットを生じさせることであって、各データユニットが、フィールドのうちの１つを一意に特定するフィールド識別子及びレコードのうちの１つからのバイナリ値を含み、バイナリ値が、フィールド識別子によって特定されるそのレコードのフィールドから抽出される、生じさせることと、複数のデータユニットからのバイナリ値についての情報を集約することと、フィールドのうちの１又は２以上のそれぞれに関するエントリのリストを生じさせることであって、エントリのうちの少なくとも一部が、それぞれ、バイナリ値のうちの１つ、及び複数のデータユニットから集約されたそのバイナリ値についての情報を含む、生じさせることと、データ型情報から第１の識別子に関連するデータ型を取り出し、取り出されたデータ型をリストのうちの１つのエントリに含まれる少なくとも１つのバイナリ値と関連付けることと、複数のデータユニットからのバイナリ値についての情報を集約することの後、フィールドに現れる特定のバイナリ値の取り出されたデータ型に少なくとも部分的に基づいてフィールドのうちの少なくとも１つに関するプロファイル情報を生じさせることとを含む。

【０００５】
態様は、以下の特徴のうちの１又は２以上を含み得る。
【０００６】
フィールド識別子によって特定されるそのレコードのフィールドから抽出されたバイナリ値は、ビットの型なし（un-typed）シーケンスとして抽出される。フィールドに現れる特定のバイナリ値の取り出されたデータ型に少なくとも部分的に基づいてフィールドのうちの少なくとも１つに関するプロファイル情報を生じさせることは、ビットの型なしシーケンスを取り出されたデータ型を有する型付き（typed）データ値として再解釈することを含む。
【０００７】
プロファイル情報は、レコードからの複数のデータ値の元のデータ型に応じて決まる型に依存するプロファイリングの結果を含む。
【０００８】
複数のデータユニットからのバイナリ値についての情報を集約することは、複数のデータユニットからのバイナリ値をエントリのリストのバイナリ値と比較して、バイナリ値の間に一致があるかどうかを判定することを含む。
【０００９】
複数のデータユニットから集約されたそのバイナリ値についての情報は、バイナリ値を比較するときに一致が判定される度にインクリメントされる一致したバイナリ値の総カウントを含む。
【００１０】
第１のバイナリ値と第２のバイナリ値との間の一致は、第１のバイナリ値を含むビットのシーケンスが第２のバイナリ値を含むビットのシーケンスと同一であることに対応する。
【００１１】
データ型情報は、１又は２以上のデータ型のそれぞれをフィールド識別子のうちの少なくとも１つと関連付ける。
【００１２】
データ型情報から第１の識別子に関連するデータ型を取り出すことは、第１のフィールド識別子に関連するデータ型を取り出すことを含む。
【００１３】
各データユニットは、フィールド識別子のうちの１つ、レコードのうちの１つからのバイナリ値、及びデータ型のうちの１つを一意に特定するデータ型識別子を含む。
【００１４】
データ型情報は、１又は２以上のデータ型のそれぞれをデータ型識別子のうちの少なくとも１つと関連付ける。
【００１５】
データ型情報から第１の識別子に関連するデータ型を取り出すことは、第１のデータ型識別子に関連するデータ型を取り出すことを含む。
【００１６】
取り出されたデータ型をリストのうちの１つのエントリに含まれる少なくとも１つのバイナリ値と関連付けることは、取り出されたデータ型を有するローカル変数をインスタンス化することと、インスタンス化された変数をエントリに含まれるバイナリ値に基づく値に初期化することとを含む。
【００１７】
取り出されたデータ型をリストのうちの１つのエントリに含まれる少なくとも１つのバイナリ値と関連付けることは、取り出されたデータ型に関連するポインタをエントリに含まれるバイナリ値が記憶されるメモリ位置に設定することを含む。
【００１８】
各データユニットは、フィールド識別子のうちの１つ、レコードのうちの１つからのバ

イナリ値、及びバイナリ値の長さのインジケータを含む。
【００１９】
長さインジケータは、バイナリ値へのプレフィックスとして記憶される。
【００２０】
方法は、複数のレコードに関連するレコードフォーマット情報をコンピューティングシステムの入力デバイス又はポートを介して受信するステップをさらに含む。
【００２１】
データ型情報は、受信されたレコードフォーマット情報に少なくとも部分的に基づいて生じさせられる。
【００２２】
処理するステップは、第１のフィールドに関するリストのうちの第１のリストのそれぞれの異なるエントリからのバイナリ値をバイナリ値に関連する取り出されたデータ型から目標のデータ型に変換することと、第２のフィールドに関するリストのうちの第２のリストのそれぞれの異なるエントリからのバイナリ値をバイナリ値に関連する取り出されたデータ型から同じ目標のデータ型に変換することとをさらに含む。
【００２３】
別の態様においては、概して、コンピュータ可読媒体に記憶されたソフトウェアが、コンピューティングシステムに、複数のフィールドのそれぞれのフィールドに関する１又は２以上の値をそれぞれが有する複数のレコードをコンピューティングシステムの入力デバイス又はポートを介して受信することと、１又は２以上のデータ型のそれぞれを少なくとも１つの識別子と関連付けるデータ型情報をコンピューティングシステムのストレージ媒体に記憶することと、レコードからの複数のデータ値をコンピューティングシステムの少なくとも１つのプロセッサを用いて処理することとを行わせるための命令を含む。処理は、レコードから複数のデータユニットを生じさせることであって、各データユニットが、フィールドのうちの１つを一意に特定するフィールド識別子及びレコードのうちの１つからのバイナリ値を含み、バイナリ値が、フィールド識別子によって特定されるそのレコードのフィールドから抽出される、生じさせることと、複数のデータユニットからのバイナリ値についての情報を集約することと、フィールドのうちの１又は２以上のそれぞれに関するエントリのリストを生じさせることであって、エントリのうちの少なくとも一部が、それぞれ、バイナリ値のうちの１つ、及び複数のデータユニットから集約されたそのバイナリ値についての情報を含む、生じさせることと、データ型情報から第１の識別子に関連するデータ型を取り出し、取り出されたデータ型をリストのうちの１つのエントリに含まれる少なくとも１つのバイナリ値と関連付けることと、複数のデータユニットからのバイナリ値についての情報を集約することの後、フィールドに現れる特定のバイナリ値の取り出されたデータ型に少なくとも部分的に基づいてフィールドのうちの少なくとも１つに関するプロファイル情報を生じさせることとを含む。
【００２４】
別の態様においては、概して、コンピューティングシステムが、複数のフィールドのそれぞれのフィールドに関する１又は２以上の値をそれぞれが有する複数のレコードを受信するように構成されたコンピューティングシステムの入力デバイス又はポートと、１又は２以上のデータ型のそれぞれを少なくとも１つの識別子と関連付けるデータ型情報を記憶するように構成されたコンピューティングシステムのストレージ媒体と、レコードからの複数のデータ値を処理するように構成されたコンピューティングシステムの少なくとも１つのプロセッサとを含む。処理は、レコードから複数のデータユニットを生じさせることであって、各データユニットが、フィールドのうちの１つを一意に特定するフィールド識別子及びレコードのうちの１つからのバイナリ値を含み、バイナリ値が、フィールド識別子によって特定されるそのレコードのフィールドから抽出される、生じさせることと、複数のデータユニットからのバイナリ値についての情報を集約することと、フィールドのうちの１又は２以上のそれぞれに関するエントリのリストを生じさせることであって、エントリのうちの少なくとも一部が、それぞれ、バイナリ値のうちの１つ、及び複数のデータ

ユニットから集約されたそのバイナリ値についての情報を含む、生じさせることと、データ型情報から第１の識別子に関連するデータ型を取り出し、取り出されたデータ型をリストのうちの１つのエントリに含まれる少なくとも１つのバイナリ値と関連付けることと、複数のデータユニットからのバイナリ値についての情報を集約することの後、フィールドに現れる特定のバイナリ値の取り出されたデータ型に少なくとも部分的に基づいてフィールドのうちの少なくとも１つに関するプロファイル情報を生じさせることとを含む。

【００２５】

別の態様においては、概して、コンピューティングシステムが、複数のフィールドのそれぞれのフィールドに関する１又は２以上の値をそれぞれが有する複数のレコードを受信するための手段と、１又は２以上のデータ型のそれぞれを少なくとも１つの識別子と関連付けるデータ型情報を記憶するための手段と、レコードからの複数のデータ値を処理するための手段とを含む。処理は、レコードから複数のデータユニットを生じさせることであって、各データユニットが、フィールドのうちの１つを一意に特定するフィールド識別子及びレコードのうちの１つからのバイナリ値を含み、バイナリ値が、フィールド識別子によって特定されるそのレコードのフィールドから抽出される、生じさせることと、複数のデータユニットからのバイナリ値についての情報を集約することと、フィールドのうちの１又は２以上のそれぞれに関するエントリのリストを生じさせることであって、エントリのうちの少なくとも一部が、それぞれ、バイナリ値のうちの１つ、及び複数のデータユニットから集約されたそのバイナリ値についての情報を含む、生じさせることと、データ型情報から第１の識別子に関連するデータ型を取り出し、取り出されたデータ型をリストのうちの１つのエントリに含まれる少なくとも１つのバイナリ値と関連付けることと、複数のデータユニットからのバイナリ値についての情報を集約することの後、フィールドに現れる特定のバイナリ値の取り出されたデータ型に少なくとも部分的に基づいてフィールドのうちの少なくとも１つに関するプロファイル情報を生じさせることとを含む。

【００２６】

態様は、以下の利点のうちの１又は２以上を有する可能性がある。

【００２７】

データプロファイリングは、データプロファイリング手順を管理する目的に専用であるユーザインターフェースを提供するプログラムによって実行されることがある。そのような状況で、ユーザは、大量のデータ（例えば、大きなデータセット及び／又は多数のデータセット）をプロファイリングするとき、比較的長い遅延を予測している可能性がある。場合によっては、データプロファイリング機能を（例えば、データフローグラフとして表現される）データ処理プログラムを開発するためのユーザインターフェースなどの別のユーザインターフェース内に組み込むことが有用である可能性がある。しかし、ユーザがデータを処理するためのプログラムを開発するプロセス中である場合、たとえその開発ユーザインターフェースの中から特定のデータセットに関するデータプロファイリングの結果を要求することが有用である可能性があるとしても、ユーザにそれらの結果を得る前に長い遅延を我慢させることは適切でない可能性がある。

【００２８】

本明細書において説明される技術を用いて、特定のデータプロファイリング手順（例えば、特に、フィールドレベルのデータプロファイリング）に関する一部の遅延を短縮することが可能である。例えば、データプロファイリングの結果を３分待たなければならない代わりに、ユーザは、３０秒待ちさえすればよい可能性がある。速度向上の少なくとも一部をもたらすために使用される技術のうちの１つは、正規化及びその他の型に依存する処理が、正規化されるべきデータ値を集約するその他のデータプロファイリング手順の後に遅らされ、ずっと効率的に実行され得るという認識に基づく。処理は、フィールドのすべての区別可能な値に対して実行されるが、それぞれの区別可能な値の必ずしもすべての出現に対して実行されるわけではなく、このことが、概して、（フィールドが一意の値を相当投入されているのでない限り）ずっと少ない操作につながるのでより効率的である。この技術は、レコードの大きな集合に関して、潜在的に大量の時間がかかる可能性がある冗

長な処理を避けることができる。この技術の帰結のうちの１つは、以下でより詳細に説明されるように、正規化及び型に依存する妥当性の確認のために後で使用されるデータ型情報を適切に管理するニーズである。

【００２９】

そのように処理の遅延が削減されることにより、そのとき、データプロファイリングが開発ユーザインターフェース内から実行されることを可能にすることが適切である可能性がある。ユーザインターフェース要素（例えば、コンテキストメニュー）が、例えば、ユーザがデータセットを表すアイコン又はデータフローを表すリンクと（例えば、右クリックアクションを用いて）インタラクションするときに表示されるために開発ユーザインターフェースに組み込まれる可能性がある。ユーザインターフェース要素は、関連するデータセット又はデータフローに対して実行されるべき１又は２以上のデータプロファイリング手順を開始するためのオプションをユーザに対して提示する可能性がある。比較的短い遅延の後、結果が、ユーザインターフェースのウィンドウ内に表示される可能性がある。遅延の間、待ち時間が比較的短いことをユーザに示すためにプログレスバーが表示される可能性がある。１つの例示的な筋書きでは、ユーザがデータセットの特定のフィールドが値の決まった組のみを有することを期待している場合、データプロファイルの結果は、いずれかの予期しない値がデータセット内に存在するかどうかをユーザに示す可能性がある。別の例示的な筋書きでは、ユーザが、すべて埋められることを期待されるフィールドがいずれかの空白又はヌル値を有するかどうかを知るためのチェックを行うことができる。そのとき、ユーザは、任意の観測された予期しない場合を適切に扱うためにプログラムに「防御論理（defensive logic）」を組み込む可能性がある。別の例示的な筋書きでは、ユーザが、データフローグラフの最後の実行において特定のデータフローを介して流れたすべてのデータの要約を見たいと望む可能性がある。

【００３０】

本発明のその他の特徴及び利点は、以下の説明及び請求項から明らかになるであろう。

【図面の簡単な説明】

【００３１】

【図１】データ処理システムのブロック図である。

【図２】データプロファイリング手順の概略図である。

【図３】フィールド－値ペア及びデータ型情報の抽出の概略図である。

【図４】データプロファイリング手順の流れ図である。

【発明を実施するための形態】

【００３２】

図１は、効率的なデータプロファイリングのためにデータ型情報を管理するための技術が使用され得るデータ処理システム１００の例を示す。システム１００は、ストレージデバイス、又はオンラインデータストリームへの接続などのデータの１又は２以上のソースを含み得るデータソース１０２を含み、それらの１又は２以上のソースのそれぞれは、さまざまなフォーマット（例えば、データベーステーブル、スプレッドシートファイル、フラットテキストファイル、又はメインフレームによって使用されるネイティブフォーマット）のいずれかでデータを記憶又は提供し得る。実行環境１０４は、プロファイリングモジュール１０６及び実行モジュール１１２を含む。プロファイリングモジュール１０６は、データソース１０２からのデータに対して、又は実行モジュール１１２によって実行されるデータ処理プログラムによって生じさせられた中間データ若しくは出力データに対してデータプロファイリング手順を実行する。データソース１０２を提供するストレージデバイスは、実行環境１０４のローカルにあり、例えば、実行環境１０４をホストするコンピュータに接続されたストレージ媒体（例えば、ハードドライブ１０８）に記憶される可能性があり、又は実行環境１０４のリモートにあり、例えば、（例えば、クラウドコンピューティングインフラストラクチャによって提供される）リモート接続を介して実行環境１０４をホストするコンピュータと通信するリモートシステム（例えば、メインフレーム１１０）でホストされる可能性がある。

【００３３】
実行環境１０４は、例えば、ＵＮＩＸオペレーティングシステムのバージョンなどの好適なオペレーティングシステムの制御下の１又は２以上の多目的コンピュータでホストされる可能性がある。例えば、実行環境１０４は、ローカルの（例えば、対称型マルチプロセッシング（ＳＭＰ，symmetric multi-processing）コンピュータなどのマルチプロセッサシステム）又はローカルに分散された（例えば、クラスタ若しくは超並列処理（ＭＰＰ，massively parallel processing）システムとして接続された複数のプロセッサか、或いは遠隔の又は遠隔に分散された（例えば、ローカルエリアネットワーク（ＬＡＮ，local area network）及び／又は広域ネットワーク（ＷＡＮ，wide-area network）を介して接続された複数のプロセッサ）か、或いはこれらの任意の組合せかのいずれかの複数の中央演算処理装置（ＣＰＵ，central processing unit）又はプロセッサコアを用いるコンピュータシステムの構成を含むマルチノード並列コンピューティング環境を含む可能性がある。
【００３４】
プロファイリングモジュール１０６は、データソース１０２からデータを読み、プロファイリングモジュール１０６によって実行されたデータプロファイリング手順によって生じさせられたプロファイル情報１１４を記憶する。プロファイリングモジュール１０６は、実行環境１０４内の実行モジュール１１２と同じ（１又は２以上の）ホスト上で実行される可能性があり、又は実行環境１０４と通信する専用データプロファイリングサーバなどのさらなるリソースを使用する可能性がある。プロファイル情報１１４は、データプロファイリング手順の結果と、以降でより詳細に説明されるセンサス（census）データなどの、結果を生じさせるプロセスでまとめられる中間データとを含む。プロファイル情報１１４は、実行環境１０４がアクセスし得るデータソース１０２若しくはデータストレージシステム１１６に戻して記憶されるか、又はその他の方法で使用される可能性がある。
【００３５】
実行環境１０４は、開発者１２０がデータ処理プログラムの開発とデータプロファイリング手順の開始との両方を行うことができる開発ユーザインターフェース１１８も提供する。一部の実施形態において、開発ユーザインターフェース１１８は、頂点間の（作業要素（work element）、すなわちデータのフローを表す）有向リンクによって接続された（データ処理構成要素又はデータセットを表す）頂点を含むデータフローグラフとしてデータ処理プログラムの開発を容易にする。例えば、そのようなユーザインターフェースは、参照により本明細書に組み込まれる「Managing Parameters for Graph-Based Applications」と題した米国特許出願公開第２００７／００１１６６８号明細書により詳細に説明されている。そのようなグラフに基づく計算を実行するためのシステムは、参照により本明細書に組み込まれる「EXECUTING COMPUTATIONS EXPRESSED AS GRAPHS」と題した米国特許第５，９６６，０７２号明細書に説明されている。このシステムによって作成されるデータフローグラフは、プロセス間で情報を移動するため及びプロセスに関する実行の順序を定義するためにグラフの構成要素によって表される個々のプロセスに情報を出し入れするための方法を提供する。このシステムは、任意の利用可能な方法からプロセス間通信の方法を選択するアルゴリズムを含む（例えば、グラフのリンクに従った通信経路は、ＴＣＰ／ＩＰ若しくはＵＮＩＸドメインソケットを使用するか、又はプロセス間でデータを渡すために共有メモリを使用する可能性がある）。データプロファイリング手順が開発ユーザインターフェース１１８の中から開始されることに加えて、データプロファイリング手順は、入力データセットにデータフローリンクによって接続された入力ポートと、データプロファイリングの結果を使用するタスクを実行するように構成された下流構成要素にデータフローリンクによって接続された出力ポートとを有するデータフローグラフのプロファイラ構成要素によって実行される可能性もある。
【００３６】
プロファイリングモジュール１０６は、異なる形態のデータベースシステムを含む、データソース１０２を具現化し得るさまざまな種類のシステムからデータを受信する可能性

がある。データは、（「属性」又は「列」とも呼ばれる）おそらくはヌル値を含む、それぞれのフィールドに関する値を有するレコードの集合を表すデータセットとして編成される可能性がある。データソースから初めにデータを読むとき、プロファイリングモジュール１０６は、概して、そのデータソース内のレコードについての何らかの最初のフォーマット情報から始める。場合によっては、データソースのレコード構造は、最初は知られていない可能性があり、その代わりに、データソース又はデータの分析後に決定される可能性がある。レコードについての最初の情報は、例えば、個々の値を記憶するために使用されるビット数、レコード内のフィールドの順序、及び特定のフィールド内に現れる値のデータ型（例えば、文字列、符号付き／符号なし整数）を含む可能性がある。

【００３７】

概して、特定のデータセットのレコードは、すべて、同じレコードフォーマットを有し、特定のフィールド内のすべての値は、同じデータ型を有する（すなわち、レコードフォーマットは「静的」である）。レコードのそれぞれのサブセットが異なるフォーマットを有する可能性があり、及び／又は１若しくは２以上のフィールドが異なるデータ型の値を有する可能性がある「動的な」レコードフォーマットのデータセットが存在し得る。本明細書において説明される一部の例は、静的なレコードフォーマットを仮定するが、動的なレコードフォーマットをサポートするためにさまざまな修正がなされ得る。例えば、レコードフォーマットの変更があるデータセット内のレコードの各サブセットの初めに、処理が再初期化される可能性がある。代替的に、同じレコードフォーマットを有するレコードの各サブセットは、静的なレコードフォーマットを有する異なる仮想的なデータセットとして扱われる可能性があり、それらの仮想的なレコードフォーマットに関する結果は、必要に応じて後で合併される可能性がある。

【００３８】

データプロファイリングを実行するとき、プロファイリングモジュール１０６は、データソース１０２からデータを読み、プロファイル情報１１４を記憶し、プロファイル情報１１４は、異なるデータセット及び異なるデータセット内の異なるフィールドを特徴付けるためにさまざまな種類の分析を実行するために使用され得る。一部の実施形態において、プロファイル情報１１４は、特定のフィールド（例えば、選択されたデータセットの選択されたフィールド又はすべてのデータセットのすべてのフィールド）内に現れる値のセンサスを含む。センサスは、フィールド内の区別可能な値のすべてをリスト化し、それぞれの区別可能な値が現れる回数を定量化する。一部の実施形態において、センサスデータは、単一のデータ構造に記憶され、フィールドによってインデックス付けされていてもよく、その他の実施形態において、センサスデータは、複数のデータ構造に、例えば、各フィールドにつき１つずつ記憶される。

【００３９】

プロファイリングされる特定のフィールドに関するセンサスデータは、各エントリがフィールドに関する識別子、フィールド内に現れる値、及びその値がデータセット内のそのフィールドに現れる回数のカウントを含むエントリのリストとして編成される可能性がある。一部のデータセットに関して、カウントは、値がそのフィールドに現れるレコードの数に等しい。その他のデータセット（例えば、入れ子になったベクトルを一部のフィールドに関する値として含む階層的なデータセット）に関して、カウントは、レコード数とは異なる可能性がある。一部の実施形態において、センサスエントリ（census entry）は、値がヌルであるか否かも示す可能性がある。それぞれの区別可能な値に関してエントリが存在し、したがって、エントリの各値はその他のエントリの値とは異なり、エントリの数はフィールド内に現れる区別可能な値の数に等しい。フィールドに関する識別子は、プロファイリングされるフィールドを一意に特定する任意の値である可能性がある。例えば、プロファイリングされるフィールドは、１からプロファイリングされるフィールドの数までの範囲内の整数インデックスを各フィールドに割り振ることによって列挙される可能性がある。そのようなインデックスは、センサスデータ構造内にコンパクトに記憶され得る。たとえ異なるフィールドに関するセンサスデータが別々のデータ構造に記憶されるとし

ても、（例えば、異なるデータ構造からのエントリを区別するために）そのフィールドに関する特定のフィールド識別子をデータ構造の各エントリ内に含めることがやはり有用である可能性がある。代替的に、一部の実施形態においては、異なるフィールドに関するセンサスデータが別々のデータ構造に記憶される場合、フィールドは、そのデータ構造に関して１回記憶されるだけでよく、各エントリは暗黙的にそのフィールドに関連付けられ、値及びカウントを含むだけである。

【００４０】

図２は、プロファイリングモジュール１０６によって実行されるセンサスに基づくデータプロファイリング手順の例を示す。抽出モジュール２００は、テーブル２０１などのプロファイリングされるデータセットから抽出されたフィールド－値ペアのストリーム２０３を生じさせるための抽出手順を実行する。この例において、テーブル２０１は、FIELD1、FIELD2、及びFIELD3と名付けられた３つのフィールドを有し、テーブル２０１の初めの数個のデータレコード（すなわち、初めの３つの行）が、３つのフィールドのそれぞれに関するそれぞれの値と共に示される。センサス生成モジュール２０２は、それぞれのフィールドに関する１又は２以上のセンサスファイル２０５を生じさせるためのフィールド－値ペアのストリーム２０３を処理する。型依存処理モジュール２０４は、型に依存するプロファイリングの結果がプロファイル情報１１４に含まれることを可能にするために、保存されたデータ型情報２０８からのさまざまなデータ型をそれぞれのフィールドに関連付ける。型非依存処理モジュール２０６は、型依存処理モジュール２０４によって復元された型付きデータ値を正規化し、これは、元のデータ型に依存しないさらなる処理を容易にするために所定の型のデータ値を提供する。型に依存する処理及び正規化をセンサスの生成の後まで遅らせることによって、同じデータ値の複数のインスタンスが別々にではなく一緒に（すなわち、その共通のデータ値に関して１回）処理され得るので、潜在的に大きな速度向上が実現される。

【００４１】

抽出モジュール２００は、特定のデータレコードをそれぞれがフィールドインデックス及びバイナリデータ値を含む一連のフィールド－値ペアにばらすことによってフィールド－値ペアを生じさせる。フィールドインデックスは、特定のフィールドを一意に（及び効率的に）特定するためにそのフィールドに割り振られたインデックス値であり（例えば、1=FIELD1、2=FIELD2、3=FIELD3）、バイナリデータ値は、そのフィールドに関してデータレコードに含まれる対応するデータ値を表すビットの型なし（又は「生（raw）」）シーケンスである。この例において、テーブル２０１の第１のデータレコードは、次のフィールド－値（すなわち、フィールドインデックス，バイナリデータ値）ペア、すなわち、(1, bin(A))、(2, bin(M))、(3, bin(X))（ここでは、例示を目的として、この例における「bin(A)」がデータ値「Ａ」を表すバイナリデータ値（すなわち、ビットのシーケンス）を表すことが理解される）を生じる。

【００４２】

センサス生成モジュール２０２は、ストリーム２０３内のフィールド－値ペアからのバイナリデータ値を集約してセンサスファイル２０５を生成する。センサスの生成の一部である集約を実行するためには、特定のデータ値が別のデータ値と同じであるかどうかを知るのに十分な情報を持っていれば十分である。このマッチングは、生バイナリデータ値を用いて実行される可能性があり、したがって、貴重な処理時間がセンサス生成モジュール２０２にデータ型を提供するのに費やされる必要がない。（図２において、センサスファイル２０５のエントリに示される値は、テーブル２０１の初めの３つのデータレコードに対応し、テーブル２０１のさらなるデータレコードからのフィールド－値ペアがセンサス生成モジュール２０２によって処理されたときに更新された。）

【００４３】

特定のデータセットに関して、フィールド－値ペアが、任意の順序でストリーム２０３に挿入される可能性がある。この例において、ストリーム２０３は、データレコードがテーブル２０１に現れるときの次のデータレコードに関するフィールド－値ペアのすべてが

後に続く特定のデータレコードに関するフィールド－値ペアのすべてを含む。代替的に、テーブル２０１は、フィールドがテーブル２０１に現れるときの次のフィールドに関するフィールド－値ペアのすべてが後に続く特定のフィールドに関するフィールド－値ペアのすべてをストリームが含むようにフィールド毎に処理される可能性がある。より高次元のデータセットも、例えば、データセットを読むために又は結果として得られるストリーム２０３からセンサスファイルを生じさせるために最も効率の良い順序に基づいてフィールド－値ペアがストリーム２０３に追加されるようにしてこのように処理され得る。フィールド－値ペアのストリーム２０３は、すべてのフィールド－値ペアが生じさせられた後に、下流のセンサス生成モジュール２０２によって処理されるファイルに書き込まれる可能性があり、又はフィールド－値ペアのストリーム２０３は、（例えば、結果として得られるパイプライン並列処理を利用するために）それらが生成されているときに下流のセンサス生成モジュール２０２に提供される可能性がある。

【００４４】

センサス生成モジュール２０２は、（例えば、ストリームがデータレコードの有限のバッチに対応する場合のストリームの終わり（end-of-stream）レコード、又はストリームがデータレコードの連続的なストリームに対応する場合の作業単位（unit of work）を区切るマーカによって示されるように）ストリーム２０３の終わりが到達されるまでフィールド－値ペアを処理する。モジュール２０２は、フィールド－値ペアのバイナリデータ値が前に処理されたフィールド－値ペアからの前のバイナリデータ値に一致するかどうかを判定するために、「センサスマッチング操作」と呼ばれるフィールド－値ペアに対するデータ操作を実行する。モジュール２０２は、ストリーム２０３内の各フィールド－値ペアに関して少なくとも１回センサスマッチング操作を実行する。モジュール２０２は、メモリデバイス内の作業メモリ空間に記憶されるデータ構造にセンサスマッチング操作の結果を記憶する。センサスマッチング操作が前のデータ値との一致を発見した場合、そのデータ値に関連する記憶されたカウントがインクリメントされる。そうではなく、センサスマッチング操作が前のデータ値との一致を発見しなかった場合、新しいエントリがデータ構造に記憶される。

【００４５】

例えば、データ構造は、配列内の関連付けられた値を探し出すために使用される一意キーを有するキー－値ペアを記憶することができる連想配列である可能性がある。この例において、キーは、フィールド－値ペアからのバイナリデータ値であり、値は、センサスデータに関する総カウントまでインクリメントされるカウントである。カウントは、キー－値ペアが特定のバイナリデータ値を連想配列内に既に存在するいかなるキーとも一致しないそのキーとしてフィールド－値ペアに関して生成されるときに１で始まり、別のフィールド－値ペアが既存のキーに一致するバイナリデータ値を有する度に１ずつインクリメントされる。モジュール２０２は、プロファイリングされるフィールドのそれぞれのために１つの連想配列が割り当てられた状態で、異なる連想配列内の（各フィールド－値ペア内のフィールドインデックスによって決定される）異なるフィールドに関するフィールド－値ペアのバイナリデータ値を探し出す。一部の実施形態においては、プロファイリングされるフィールドの数が、事前に知られており、（最小限の量の記憶空間のみ使用する）空の連想配列が、プロファイリング手順の初めに各フィールドのために割り当てられる。

【００４６】

連想配列は、例えば、キーの効率的な探索及び関連する値の修正を提供するハッシュテーブル又はその他のデータ構造を使用して実装され得る。キー－値ペアのキーとして使用されるバイナリデータ値は、バイナリデータ値自体のコピー、又は作業メモリの異なる位置に記憶される（例えば、フィールド－値ペアのコピーに記憶される）バイナリデータ値へのポインタを記憶する可能性がある。そして、連想配列は、フィールド－値ペアからのバイナリデータ値の記憶されたコピー、又はさらにはフィールド－値ペア自体の全体と一緒に、集合的に、センサスマッチングの結果を記憶するデータ構造と考えられる可能性がある。フィールド－値ペアのバイナリデータ値へのポインタが連想配列に記憶される実施

形態においては、特定のキーを含む最初のフィールド－値ペアのみが、作業メモリに記憶される必要があり、その特定のキーを含む後続のフィールド－値ペアは、センサスマッチング操作の後、作業メモリから削除され得る。

【００４７】

下の例において、プロファイリングされるフィールドに関するこれらの連想配列は、「センサス配列（census array）」と呼ばれ、キー－値ペアは、センサス配列内の「センサスエントリ」と呼ばれる。センサスの生成の終わりに、センサス生成モジュール２０２によって生じさせられたセンサス配列は、テーブル２０１内に現れるすべての区別可能なバイナリデータ値を別々のセンサスエントリ内に記憶し、プロファイリングされるデータレコードを表すテーブル２０１の行内にそのバイナリデータ値が現れる回数の総カウントを記憶する。おそらくは型に依存する処理の一部として、センサス配列は、センサスエントリがテーブル２０１内に現れるすべての区別可能な型付きデータ値を記憶するように、生バイナリデータ値だけでなく、それらのバイナリデータ値に関連する型も記憶するように更新される可能性があってもよい。フィールドのすべてのデータ値が同じデータ型を有する静的なレコードフォーマットに関して、型指定される前に区別可能であるデータ値は、型指定された後も区別可能である。

【００４８】

型依存処理モジュール２０４は、データプロファイリング手順が、特定のデータ値がそのデータ値の元のデータ型のために有効であるかどうかを判定することを可能にする。例えば、特定のフィールド内に現れる値の元のデータ型が「date」データ型を有するものとして（レコードフォーマットで）定義される場合、そのフィールドに関する有効なデータ値は、特定の文字列フォーマットと、その文字列の異なる部分に関して許容される値の範囲とを有する可能性がある。例えば、文字列フォーマットは、ＹＹＹＹ－ＭＭ－ＤＤと指定される可能性があり、ここで、ＹＹＹＹは、年を表す任意の４桁の整数であり、ＭＭは、月を表す１～１２の間の任意の２桁の整数であり、ＤＤは、日を表す１～３１の間の任意の２桁の整数である。「date」型の有効な値に対するさらなる制約は、例えば、特定の月が１～３０の間の日しか許容しないことをさらに指定する可能性がある。別の例において、データ型「ＵＴＦ－８」を有するフィールドのデータ値が有効であることを確認することは、有効なＵＴＦ－８文字で許容されないビットの特定のシーケンスに関してデータ値の各ＵＴＦ－８文字に現れるバイトを調べることを含む可能性がある。別の例において、特定のメインフレームフォーマットのデータ型を有するフィールドのデータ値が有効であることを確認することは、そのメインフレームフォーマットによって定義される特定の特徴に関して調べることを含む可能性がある。型に依存する妥当性のチェックは、データ値の元のデータ型に依存するユーザ定義の妥当性確認規則の適用も含む可能性がある。データ値がそのデータ型のために有効でないことが分かる場合、型依存処理モジュール２０４は、そのデータ値に有効でないものとしてフラグを立てることができ、そのことは、型非依存処理モジュール２０６によって実行されたであろうそのデータ値に対するさらなる処理を実行する必要を回避し得る。そのような妥当性のチェックを実行する前に、型依存処理モジュール２０４は、以下でより詳細に説明されるように、保存されたデータ型情報２０８からデータ型を取り出し、取り出されたデータ型を、所与のフィールドに関するセンサスに含まれるそれぞれのバイナリ値と関連付ける。

【００４９】

バイナリデータ値を抽出し、後で使用するためにデータ型情報２０８を別に保存するための技術の一例が、図３に示される。テーブル２０１は、現れる値のデータ型をテーブル２０１の３つのフィールドと共に記述するレコードフォーマット３００に関連付けられる。この例において、レコードフォーマット３００は、ここに示されるようにフィールドの宣言のリストを用いて定義される。

T_1 L_1 FIELD1
T_2 L_2 FIELD2
T_3 L_3 FIELD3

レコードフォーマット３００は、テーブル２０１の３つのフィールド、すなわち、FIELD1、FIELD2、FIELD3のそれぞれに関するフィールドの宣言を含む。FIELDiに関して、フィールドの宣言は、フィールドのデータ値のデータ型の識別子T_i、フィールドのデータ値の長さの識別子L_i、及びフィールド名を含む。データ型及び長さの識別子（T_i及びL_i）は、（i = 1～フィールドの数に関して）この例においては記号的に表されるが、実際のレコードフォーマットは、データ型及び長さを特定するためのさまざまなキーワード、句読点、及びその他の構文規則の要素のいずれかを使用する可能性がある。

【００５０】

データ型及び長さの識別子が特定の種類の構文規則（すなわち、データ操作言語（ＤＭＬ，Data Manipulation Language）の構文規則）を用いてどのようにして指定される可能性があるかを示す例が、ここで示され、フィールドの宣言は、用語「record」及び「end;」によって区切られ、それぞれがセミコロンで終わる。

```
record
  string(7) FIELD1;
  int(4) FIELD2;
  decimal(10) FIELD3;
end;
```

この例においては、３つの異なるデータ型、すなわち、キーワード「string」によって特定される文字列データ型と、キーワード「int」によって特定される整数データ型と、キーワード「decimal」によって特定される浮動小数点１０進数データ型とが存在する。また、この例は、データ型のキーワードの後に括弧付きで（バイトを単位とする）長さの識別子を含む。

【００５１】

その他の種類のレコードフォーマットも、使用され得る。例えば、一部のレコードフォーマットにおいて、データ値の長さは、（例えば、可変長のデータ値又は区切られたデータ値に関して）フィールドの宣言において明示的に指定されない。一部のレコードフォーマットは、例えば、条件付きレコードフォーマット又は特定の種類のストレージシステムのためのレコードフォーマット（例えば、ＣＯＢＯＬのコピーブック）を指定するために使用され得る潜在的に複雑な構造（例えば、階層的な又は入れ子にされた構造）を有する。この複雑な構造は、解析される（又は「ウォークされる（walked）」）可能性があり、各フィールドは、一意のフィールド識別子を割り振られる可能性がある。

【００５２】

抽出モジュール２００は、型依存処理モジュール２０４が型に依存する処理のために元のデータ型を復元することができるように、レコードフォーマット３００によって定義されたフィールドの元のデータ型についての十分な情報を含むデータ型情報２０８を記憶する。例えば、データ型の識別子T_i及び長さの識別子L_iは、図３に示されるように、データ型の識別子及び長さの識別子をそれらのそれぞれのフィールドに関する対応するフィールド識別子と関連付ける連想配列に記憶される可能性がある。また、フィールドの名前などのさらなる情報が、データ型情報２０８に記憶される可能性があってもよい。データ型の識別子T_iは、それらの元の形態で、又は元のデータ型を復元するために十分な情報を保存する異なる形態で記憶される可能性がある。データ型情報２０８のこの生成は、データセット毎に１回実行されさえすればよく、センサスの生成のためにフィールド－値ペアを生じさせるときにデータ値と一緒にデータ型を抽出するシステムによって行われる潜在的に大量の作業を避ける。たとえデータ型情報２０８の生成が複数回繰り返される（例えば、抽出モジュール２００によって１回行われ、型依存処理モジュール２０４によって再び行われる）としても、潜在的に大量の作業は、このデータ型の抽出をデータセットのレコード数程度の回数ではなく一定の回数に制限することによってやはり避けられ得る。

【００５３】

データ型情報２０８は、抽出モジュール２００によって１回生じさせられる場合、型依存処理モジュール２０４による取り出し（又は別の方法での型依存処理モジュール２０４

への伝達）のために記憶される可能性がある。その代わりに、データ型情報２０８が抽出モジュール２００によって生じさせられ、（例えば、そのデータ型情報２０８が必要とされる直前に）型依存処理モジュール２０４によって別途生じさせられる場合、レコードフォーマットによって定義されたフィールドにインデックス値を割り振るための処理がそれら両方に関して同じである限り、同じデータ型情報２０８が取得される。どのモジュールがデータ型情報２０８を生じさせるとしても、そのモジュールは、フィールド－値ペアにフィールドインデックス値を挿入するために抽出モジュール２００によって使用されるのと同じ、特定のフィールドにマッピングされたフィールドインデックス値３０２をやはり使用する。

【００５４】

抽出モジュール２００の出力は、フィールドインデックス及びバイナリデータ値からなるペアに加えてその他の情報を含む可能性があってもよい要素のストリームである。図３を引き続き参照すると、抽出モジュール２００によってテーブル２０１から抽出されるフィールド－値ペアの代替的なストリーム２０３’が、レコードインデックス及び各バイナリデータ値に関する長さをやはり含む要素で構成される。ストリーム２０３’内の１つの要素３０６は、（FIELD2に対応する）フィールドインデックス値「２」、長さ値len(M)、バイナリデータ値bin(M)、及び（テーブル２０１の最初のレコードに対応する）レコードインデックス値「１」を含む。フィールドインデックス及びバイナリデータ値は、上述のようにセンサスエントリをまとめるために使用される。レコードインデックスは、以下で説明されるように位置情報をまとめるために使用され得る。長さ値len(M)は、例えば、後に続くバイナリデータ値bin(M)に関する長さ４バイトを表す（例えば、固定長の整数として符号化された）値４である可能性がある。長さのない図２のストリーム２０３は、長さ（L_i）がレコードフォーマットで指定され、ストリーム２０３の要素を読むときにセンサス生成モジュール２０２によってアクセスされ得るデータ値の固定長フィールドのために十分である可能性がある。しかし、レコードフォーマットが固定長を指定せず、その代わりに、データ値の長さが可変であることを許容する場合、ストリーム２０３’と同様に、長さプレフィックス（length prefix）が各要素に含まれる可能性がある。空のデータ値に関しては、長さプレフィックス０が使用される可能性があり、対応するバイナリデータ値が後に続かない。

【００５５】

センサスの生成は、概して、プロファイル情報がフィールドに現れる値を特徴付ける「フィールドレベル」プロファイリングを可能にするために実行される。一部の実施形態において、センサス生成モジュール２０２は、「レコードレベル」プロファイリングのために有用である、それぞれの区別可能なデータ値に関してそのデータ値が現れるデータセットのあらゆるレコードを特定する位置情報を各センサスエントリにやはり追加する。この例において、位置情報は、テーブル２０１内の特定のレコードにマッピングされたレコードインデックス値３０４に基づいてまとめられ得る。例えば、特定のフィールドの特定のデータ値に関するセンサスエントリの生成中、ビットベクトルは、そのフィールドにそのデータ値を有するあらゆるレコードのレコードインデックスの整数値に対応するそのビット位置の組を有する。ビットベクトルは、必要とされる記憶空間を削減するために圧縮され得る。レコードインデックス値３０４は、例えば、各レコードに連続する整数のシーケンスを割り振ることによって生じさせられる可能性がある。そして、位置情報は、レコードレベルの統計をまとめ、データプロファイリングによって発見された特定の特性を有するレコード（例えば、特定のフィールドに現れると予測されなかった特定のデータ値を有するすべてのレコード）の位置を特定する（又はそれらのレコードに「ドリルダウンする」）ためにモジュール２０４及び２０６又はその他のデータプロファイリング手順によって使用される可能性がある。位置情報をまとめることは、さらなる処理時間がかかる可能性があるが、センサスの生成の前にあらゆるレコードを直接処理することによってレコードレベルの統計をまとめることほどはコストがかからない可能性がある。

【００５６】

さまざまな技術が、元のデータ型を型に依存する処理及び結果として起こる正規化のためのセンサス配列内の対応するバイナリデータ値と関連付けるために型依存処理モジュール２０４によって使用され得る。一部の実施形態において、センサスエントリからのバイナリデータ値は、そのセンサスエントリからのフィールドインデックスを用いてデータ型情報２０８から取り出されたデータ型を有するローカル変数をインスタンス化することによって復元されるそのバイナリデータ値のデータ型を有する。ローカル変数は、（例えば、ローカル変数のデータ型に応じてバイナリデータ値を解析することによって）センサスエントリに含まれるバイナリデータ値に対応する値に初期化される。例えば、ローカル変数は、型依存処理モジュール２０４が実装されるプログラミング言語の変数（例えば、Ｃ又はＣ＋＋の変数）である可能性がある。そして、そのインスタンス化され、初期化されたローカル変数は、型依存処理モジュール２０４によって実行される処理のために使用され得る。代替的に、一部の実施形態においては、新しいローカル変数をインスタンス化する代わりに、取り出されたデータ型に関連するポインタが、（例えば、センサスエントリで）バイナリデータ値が記憶されるメモリ位置に設定される可能性がある。元のデータ型の復元も、バイナリデータ値を型付きデータ値として再解釈するためのコードの行を生じさせるための処理モジュール（例えば、ＤＭＬコードを解釈するための専用エンジン）を呼び出す関数呼び出しを用いて実行される可能性がある。例えば、関数呼び出しは、次のように表され得る。

reinterpret_as(<data type identifier>,<binary data value>)

関数「reinterpret_as」は、第２の「binary data value」引数を第１の「data type identifier」引数に対応するデータ型を有する型付きデータ値として再解釈するために必要な処理を呼び出す。

【００５７】

上述のように、実行される型に依存する処理の一部は、データ値がそのデータ値の型のために有効であるかどうかを判定するためのチェックを行うことを含む可能性がある。データ値がそのデータ値の型のために有効であるのか有効でないのかを判定する前に、データ値がヌルであるのか又は見つからないのか（フィールドが少なくとも１つのレコードに関して空であったことを示す）を判定するためのチェックが存在する可能性がある。レコードフォーマットで定義され得る所定のヌル値（又は任意の数の「空白」文字などの値）が存在する可能性がある。見つからない値は、例えば、バイナリデータ値の長さゼロを示す長さプレフィックスによって示され得る。場合によっては、見つからない値は、ヌル値とは異なるように扱われる可能性がある。どのデータ値がヌル値と考えられるかは、データ型に応じて決まる可能性がある。

【００５８】

正規化は、（モジュール２０６による）型に依存しない処理の初め又は（モジュール２０４による）型に依存する処理の終わりに実行される可能性がある。正規化は、例えば、すべてのデータ値を目標のデータ型「string」を有するデータ値に変換することを含み得る。データ値が（復元された）データ型「string」を既に有する場合、正規化手順は、そのデータ値に対するいかなる操作も実行しない可能性がある。一部の実施形態において、正規化手順は、たとえデータ値が目標のデータ型を有するとしてもいくつかの操作（例えば、先頭の又は末尾の「空白」文字を削除すること）をやはり実行する可能性がある。正規化が２つの異なるデータ値を同じ正規化されたデータ値にマッピングすることがあり得る。例えば、（第１のフィールドからの）データ型「string」を有するデータ値「3.14」は、末尾の「空白」文字を削除させて「string」の値「3.14」を生じ、（第２のフィールドからの）データ型「decimal」を有するデータ値「3.14」は、同じ「string」の値「3.14」に変換される可能性がある。同じフィールドからの（したがって、同じ元のデータ型を有する）２つの異なるデータ値が同じ正規化されたデータ値に変換される場合、一部の実施形態において、型非依存処理モジュール２０６は、新しい値に関するカウントが古い値の個々のカウントの合計であるようにそれら２つのデータ値に関するセンサスエントリを集約するために適切なセンサス配列を更新する可能性があってもよい。

【００５９】
図４は、遅らされた型に依存する処理及び正規化のためのデータ型管理技術を用いる手順データプロファイリングの例の流れ図４００を示す。この流れ図は、データプロファイリングのためのあり得るアルゴリズムを表すが、どの順序で特定のステップが実行され得るかを限定するように意図されていない（例えば、異なる形態の並列処理を可能にする）。外側のループにおいて、システム１００は、プロファイリングされることになるデータセットを受信し（４０２）、データ型をプロファイリングされる各フィールドに関する対応するフィールド識別子と関連付ける対応するデータ型情報を記憶する（４０４）。内側のループにおいて、システム１００は、フィールドのうちの１つを一意に特定するフィールド識別子と、レコードのうちの１つからのバイナリ値とをそれぞれが含むデータユニット（すなわち、フィールド－値ペア）を生じさせる（４０６）。バイナリ値は、フィールド識別子によって特定されるレコードのフィールドから抽出される。システム１００は、内側のループを終了する条件として、処理すべきいずれかのさらなるレコードが存在するかどうかを判定するためのチェックを行い（４０８）、外側のループを終了する条件として、処理すべきいずれかのさらなるデータセットが存在するかどうかを判定するためのチェックを行う（４１０）。

【００６０】
潜在的に、（例えば、図２のモジュールのパイプライン並列処理を用いて）内側及び外側のループと並列に、システム１００は、データユニット（例えば、フィールド識別子に基づく特定のフィールドに関するデータユニット）のグループからバイナリ値についての情報を集約する（４１２）。一部の実施形態において、この集約は、プロファイリングされる各フィールドに関してエントリのリストが生じさせられる（４１４）センサス手順の形態である。各センサスエントリは、バイナリ値のうちの区別可能な１つと、複数のデータユニットから集約されたそのバイナリ値についての情報（例えば、総カウント）とを含む。型に依存する処理のフェーズで、システム１００は、データ型情報からそれぞれのフィールド識別子に関連するデータ型を取り出し（４１６）、それぞれの取り出されたデータ型を（フィールド識別子に基づいて）リストのうちの適切な１つのエントリに含まれるバイナリ値と関連付ける。これは、システム１００が、複数のデータユニットからのバイナリ値についての情報を集約した後、フィールドに現れる特定のバイナリ値の取り出されたデータ型に少なくとも部分的に基づいてフィールドのうちの１又は２以上に関するプロファイル情報を効率的に生じさせる（４１８）ことを可能にする。

【００６１】
型に依存する処理及び正規化をセンサスに基づく集約の後まで遅らせることと共に、その他の技術が、データプロファイリングの効率をさらに向上させる（潜在的な遅延を減らす）ために組み合わせて使用され得る。例えば、技術は、オーバーフロー記憶空間へと作業メモリ空間を効率的にスピルさせるために使用され得る。一部の実施形態において、データプロファイリング手順を実行するプログラム、又はプログラムの一部（例えば、センサス生成モジュール２０２）は、プログラムが使用することを許容されるメモリデバイス内の作業メモリ空間の最大量を設定するメモリ制限を与えられる可能性がある。プログラムは、許容される作業メモリ空間の最大量のほとんどを必要とする可能性があるセンサス配列を記憶するため、及びセンサス配列よりもかなり少ない空間を必要とする可能性があるその他の一時的な値を記憶するために作業メモリ空間を使用し得る。作業メモリ空間に対するオーバーフロー条件は、モジュール２０２が、センサス配列にさらなるエントリを追加するのに十分な利用可能な作業メモリ空間が存在しなさそうであると判定するか、又は（例えば、追加された最後のエントリが原因で）さらなるエントリを追加するためのいかなる利用可能な作業メモリ空間ももはや存在しないと判定するときに満たされる。モジュール２０２は、（センサス配列内のすべてのデータ値若しくはポインタによって参照されるフィールド－値ペアを含む）センサス配列の合計サイズを測定し、このサイズをメモリ制限（若しくはその他の閾値）と比較することによって、又はセンサス配列の合計サイズを直接測定することなく、残された利用可能な作業メモリ空間の量（例えば、メモリア

ドレスの割り当てられたブロックから残されるメモリアドレスの範囲）を判定することによってこの判定を行うことができる。

【００６２】

一部の実施形態において、プログラムは、センサス配列の合計サイズがメモリ制限に近いときを検出するためのオーバーフロー閾値を設定する。センサス配列の合計サイズは、例えば、個々のセンサス配列のサイズの合計を計算することによって直接測定される可能性があり、個々のセンサス配列のサイズは、そのセンサス配列によって占有される作業メモリ空間内のビット数として測定される。代替的に、センサス配列の合計サイズは、例えば、作業メモリ空間内の残された利用可能な空間の量を計算することによって間接的に測定される可能性がある。一部の実施形態において、プログラムは、その他の値のためのいくらかの空間を残しておくためにメモリ制限のすぐ下にオーバーフロー閾値を設定する。一部の実施形態において、オーバーフロー閾値は、例えば、その他の値のために必要とされる空間が無視できる場合、及び／又はプロファイリングモジュール１０６が厳密なメモリ制限を課さず、メモリ制限が比較的短い期間少しの量だけ超えられることを許容する場合、メモリ制限に等しい可能性がある。

【００６３】

オーバーフロー条件がトリガされた後、プログラムは、ストレージデバイス（例えば、データストレージシステム１１６）内に完成されたセンサス配列オーバーフロー記憶空間を生じさせるために必要とされる何らかのデータを記憶するためのオーバーフロー処理手順を使用する。厳密に何がオーバーフロー記憶空間に記憶されるかは、使用されるオーバーフロー処理手順の種類に応じて決まる。参照により本明細書に組み込まれる、「MANAGING MEMORY AND STORAGE SPACE FOR A DATA OPERATION」と題した米国特許出願公開第２０１４／０３４４５０８号明細書は、プログラムが、オーバーフロー条件がトリガされた後、処理される各フィールド－値ペアに関するセンサスマッチング操作を実行し続け、データ操作の結果に関連する情報（すなわち、センサスエントリのインクリメントされたカウント又は新しいセンサスエントリ）を作業メモリ内のセンサス配列の同じ組か又は作業メモリ内のセンサス配列の新しい組かのどちらかに記憶するオーバーフロー処理手順の例を記載する。オーバーフロー条件がストリーム２０３のフィールド－値ペアの処理中のある時点でトリガされた場合、一部のデータは、作業メモリ空間に記憶され、一部のデータは、オーバーフロー記憶空間に記憶される。場合によっては、両方の位置のデータが、完成されたセンサス配列を生じさせるために何らかの方法で組み合わされる。各センサス配列は、型依存処理モジュール２０４による処理のために独自のセンサスファイル２０５内に出力される。各バイナリデータ値が抽出され、そのバイナリデータ値のデータ型を示すそのバイナリデータ値の関連するメタデータなしにセンサス配列に記憶され得るので、センサスデータの記憶サイズが少なく維持され得、そのことは、さらに、オーバーフローが起こる可能性を小さくする。

【００６４】

上述の技術は、例えば、好適なソフトウェア命令を実行するプログラミング可能なコンピューティングシステムを用いて実装される可能性があり、又はフィールドプログラマブルゲートアレイ（ＦＰＧＡ，field-programmable gate array）などの好適なハードウェアで、若しくは何らかの混成の形態で実装される可能性がある。例えば、プログラミングされる手法において、ソフトウェアは、それぞれが少なくとも１つのプロセッサ、（揮発性及び／又は不揮発性のメモリ及び／又はストレージ要素を含む）少なくとも１つのデータストレージシステム、（少なくとも１つの入力デバイス又はポートを用いて入力を受け取るため、及び少なくとも１つの出力デバイス又はポートを用いて出力を与えるための）少なくとも１つのユーザインターフェースを含む、（分散、クライアント／サーバ、又はグリッドなどのさまざまなアーキテクチャである可能性がある）１又は２以上のプログラミングされた又はプログラミング可能なコンピュータシステムで実行される１又は２以上のコンピュータプログラムの手順を含み得る。ソフトウェアは、例えば、データフローグラフの設計、構成、及び実行に関連するサービスを提供するより大きなプログラムの１又

は２以上のモジュールを含む可能性がある。プログラムのモジュール（例えば、データフローグラフの要素）は、データリポジトリに記憶されたデータモデルに準拠するデータ構造又はその他の編成されたデータとして実装され得る。

【００６５】

ソフトウェアは、ＣＤ－ＲＯＭ又は（例えば、多目的若しくは専用のコンピューティングシステム若しくはデバイスによって読み取り可能な）その他のコンピュータ可読媒体などの有形の非一時的媒体で提供されるか、或いはそのソフトウェアが実行されるコンピューティングシステムの有形の非一時的媒体にネットワークの通信媒体を介して配信される（例えば、伝搬信号に符号化される）可能性がある。処理の一部又はすべては、専用のコンピュータで、又はコプロセッサ若しくはフィールドプログラマブルゲートアレイ（ＦＰＧＡ）若しくは専用の特定用途向け集積回路（ＡＳＩＣ，application-specific integrated circuit）などの専用のハードウェアを用いて実行される可能性がある。処理は、ソフトウェアによって指定された計算の異なる部分が異なる計算要素によって実行される分散された方法で実装される可能性がある。それぞれのそのようなコンピュータプログラムは、本明細書において説明された処理を実行するためにストレージデバイスの媒体がコンピュータによって読み取られるときにコンピュータを構成し、動作させるために、多目的又は専用のプログラミング可能なコンピュータによってアクセス可能なストレージデバイスのコンピュータ可読ストレージ媒体（例えば、ソリッドステートメモリ若しくは媒体、又は磁気式若しくは光学式媒体）に記憶されるか又はダウンロードされることが好ましい。本発明のシステムは、コンピュータプログラムで構成された有形の非一時的媒体として実装されるものとして考えられる可能性もあり、そのように構成された媒体は、本明細書において説明された処理ステップのうちの１又は２以上を実行するために特定の予め定義された方法でコンピュータを動作させる。

【００６６】

本発明のいくつかの実施形態が、説明された。しかしながら、上述の説明は、添付の請求の範囲によって画定される本発明の範囲を例示するように意図されており、限定するように意図されていないことを理解されたい。したがって、その他の実施形態も、添付の請求の範囲内にある。例えば、本発明の範囲を逸脱することなくさまざまな修正がなされ得る。さらに、上述のステップの一部は、順序に依存しない可能性があり、したがって、説明された順序とは異なる順序で実行される可能性がある。

【図１】

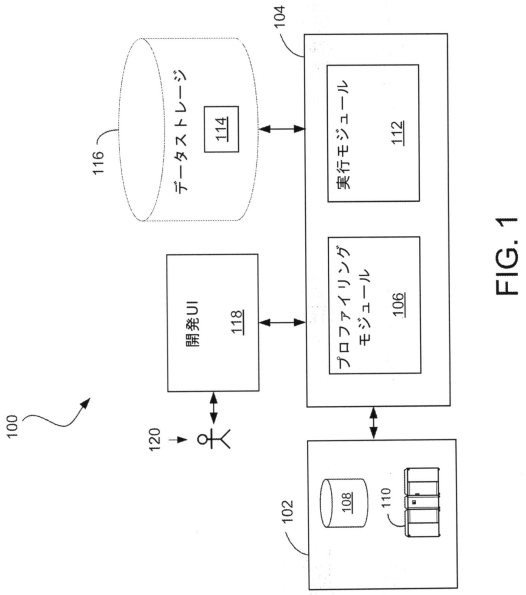

FIG. 1

【図２】

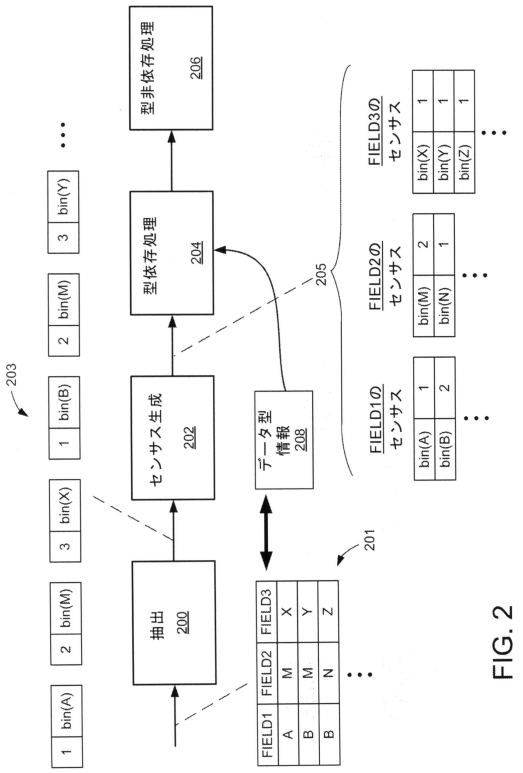

FIG. 2

【図３】

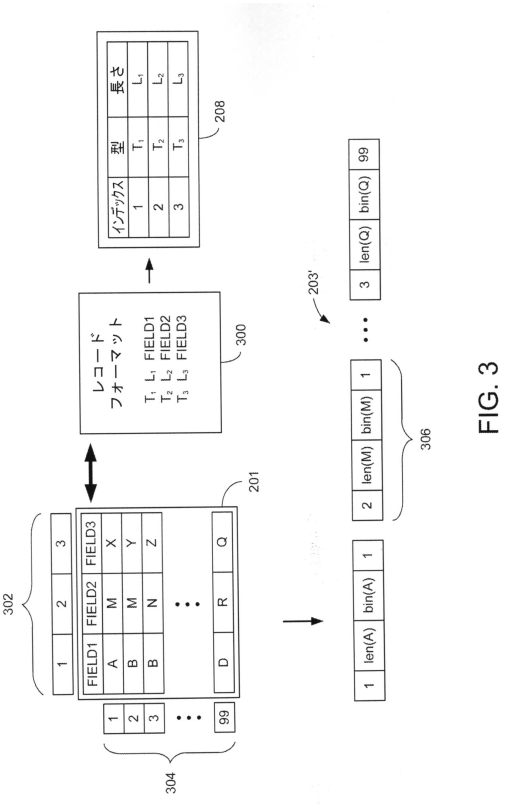

FIG. 3

【図４】

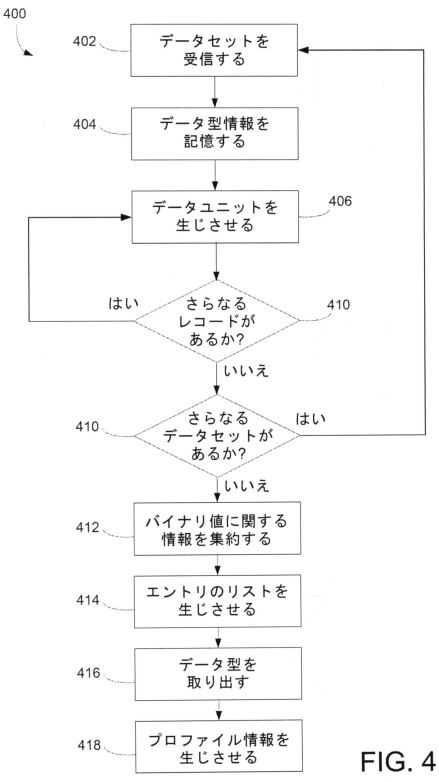

FIG. 4

フロントページの続き

(74)代理人 100150902
弁理士 山内 正子
(74)代理人 100141391
弁理士 園元 修一
(74)代理人 100198074
弁理士 山村 昭裕
(74)代理人 100145920
弁理士 森川 聡
(74)代理人 100096013
弁理士 富田 博行
(72)発明者 カーン ムハンマド アルシャド
アメリカ国 ０２０６７ マサチューセッツ シャロン クラークストリート３１

審査官 山本 俊介

(56)参考文献 特表２００７－５０６１９１（ＪＰ，Ａ）
特開２０００－２１５２１６（ＪＰ，Ａ）
米国特許出願公開第２００５／００９１６４８（ＵＳ，Ａ１）
米国特許出願公開第２０１１／００２９４７８（ＵＳ，Ａ１）

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