

(19) 日本国特許庁(JP)

(12) 特 許 公 報(B2)

(11) 特許番号

特許第6893209号  
(P6893209)

(45) 発行日 令和3年6月23日(2021.6.23)

(24) 登録日 令和3年6月2日(2021.6.2)

(51) Int.Cl. F 1  
G 0 6 F 16/21 (2019.01) G 0 6 F 16/21

請求項の数 26 (全 23 頁)

|                    |                               |           |                             |
|--------------------|-------------------------------|-----------|-----------------------------|
| (21) 出願番号          | 特願2018-522637 (P2018-522637)  | (73) 特許権者 | 515091706<br>アクシオム コーポレーション |
| (86) (22) 出願日      | 平成28年10月28日 (2016.10.28)      |           | アメリカ合衆国 7 2 0 3 3 アーカンソ     |
| (65) 公表番号          | 特表2019-502979 (P2019-502979A) |           | ー、コンウェイ、イースト デイブ ウォ         |
| (43) 公表日           | 平成31年1月31日 (2019.1.31)        |           | ード ドライブ 3 0 1               |
| (86) 国際出願番号        | PCT/US2016/059378             | (74) 代理人  | 110000855<br>特許業務法人浅村特許事務所  |
| (87) 国際公開番号        | W02017/075392                 | (72) 発明者  | ボトナー、マーク                    |
| (87) 国際公開日         | 平成29年5月4日 (2017.5.4)          |           | アメリカ合衆国、アーカンソー、リトル          |
| 審査請求日              | 令和1年8月6日 (2019.8.6)           |           | ロック、リヴァークレスト ドライブ 1         |
| (31) 優先権主張番号       | 62/248,619                    |           | 2 7 0 1                     |
| (32) 優先日           | 平成27年10月30日 (2015.10.30)      | (72) 発明者  | コリンズ、ダブリュー.、ドウェイン           |
| (33) 優先権主張国・地域又は機関 | 米国 (US)                       |           | アメリカ合衆国、アーカンソー、コンウェ         |
|                    |                               |           | イ、ロバート オット サークル 3 0 3       |
|                    |                               |           | 5                           |

最終頁に続く

(54) 【発明の名称】 構造化されたマルチフィールドファイルのレイアウトの自動解釈

(57) 【特許請求の範囲】

【請求項 1】

複数のフィールドをそれぞれ含む複数のレコードを含むデータファイルからファイルレイアウトを推測する方法であって、

a. 推測エンジンにおいて前記データファイルを受信するステップと、

b. 前記データファイルの予備分析を実行するステップであって、前記予備分析が前記データファイルがフィールド区切りファイルであるか固定幅フィールドファイルであるかを判断するステップを含む、実行するステップと、

c. 前記データファイルがフィールド区切りファイルであると判定された場合、前記データファイルに対して区切られた分析を実行するステップと、

d. 前記データファイルが固定幅ファイルであると判定された場合、前記データファイルに対して固定幅分析を実行するステップと、

e. 前記データファイルに列タイプ識別を適用するステップであって、当該データファイルに列タイプ識別を適用する前記ステップは、少なくとも1つの基本オラクル、少なくとも1つの標準オラクル、および少なくとも1つのメタオラクルを前記データファイルに適用するステップを含み、前記基本オラクルは、単一の固定フィールドにおける基本ベシクタイプに対して真理値で応答するように構成されたソフトウェアプログラムを含み、前記標準オラクルは、前記単一の固定フィールドにおける標準フィールドに対して真理値で応答するように構成されたソフトウェアプログラムを含み、前記メタオラクルは、単一の混在するフィールドと1つ以上の隣接フィールドとの両方にわたるメタフィールドタ

10

20

イプに対して真理値で応答するように構成されたソフトウェアプログラムを含む、適用するステップと、

f. 前記データファイルの最終的な列タイプ情報を出力するステップとを含む、方法。

【請求項2】

前記少なくとも1つの基本オラクルが、アルファベットオラクル、英数字オラクル、空白オラクル、数字オラクル、または数値オラクルの1つ以上を含む、請求項1に記載の方法。

【請求項3】

前記少なくとも1つの基本オラクルが、アルファベットオラクル、英数字オラクル、空白オラクル、数字オラクル、および数値オラクルのそれぞれを含む、請求項2に記載の方法。

10

【請求項4】

前記少なくとも1つの標準オラクルが、住所リンクオラクル、消費者リンクオラクル、文書識別子オラクル、商号オラクル、都市オラクル、国オラクル、国オラクル、日付オラクル、ドメインオラクル、電子メールオラクル、ファーストネームオラクル、または性別オラクルの1つ以上を含む、請求項2に記載の方法。

【請求項5】

前記少なくとも1つの標準オラクルが、住所リンクオラクル、消費者リンクオラクル、文書識別子オラクル、商号オラクル、都市オラクル、国オラクル、国オラクル、日付オラクル、ドメインオラクル、電子メールオラクル、ファーストネームオラクル、および性別オラクルのそれぞれを含む、請求項4に記載の方法。

20

【請求項6】

前記少なくとも1つのメタオラクルが、完全な住所オラクルまたはフルネームオラクルの1つ以上を含む、請求項2に記載の方法。

【請求項7】

前記少なくとも1つのメタオラクルが、完全な住所オラクルおよびフルネームオラクルのそれぞれを含む、請求項6に記載の方法。

【請求項8】

区切られた分析を前記ファイル上で実行する前記ステップが、  
a. 非数字、非アルファベット文字の初期頻度表を計算するステップと、  
b. 可能な区切り文字のセットから試行区切り文字を使用して、前記データファイル内の各行について列数をカウントするステップと、  
c. フィールドおよび行の数で前記データファイル内の各行について前記列数を要約するステップと、  
d. 少ない要約カウントをフィルタリングして除くステップと、  
e. 前記基本オラクルまたは前記標準オラクルの1つ以上を使用してフィールドカウントをランク付けするステップと、  
f. 最終的な区切られた決定を出力するステップとを含む、請求項1に記載の方法。

30

40

【請求項9】

前記ファイルに対して固定幅分析を実行する前記ステップが、  
a. 前記データファイル上に空間ヒストグラムを作成するステップと、  
b. 前記データファイル上に文字マップを作成するステップと、  
c. 前記空間ヒストグラムおよび文字マップの1つ以上を使用して前記データファイル上に列抽出をマッピングするステップと、  
d. 最終的な固定幅決定を出力するステップとを含む、請求項1に記載の方法。

【請求項10】

前記データファイルに列タイプの識別を適用する前記ステップが、

50

- a. 少なくとも1つの基本オラクルを使用して前記データファイル内の有効値をカウントするステップと、
  - b. 少なくとも1つの標準オラクルを使用して前記データファイル内の有効値をカウントするステップと、
  - c. 前記少なくとも1つの基本オラクルおよび少なくとも1つの標準オラクルを使用して、前記データファイル内の有効値をカウントする前記ステップに続いて、少なくとも1つのメタオラクルを使用して最初の列タイプを計算するステップと、
  - d. 未知の列タイプが残っている場合に、1つ以上の通常オラクルまたは基本オラクル情報を適用するステップと、
  - e. 最終的なタイプ決定を出力するステップと
- を含む、請求項1に記載の方法。

10

**【請求項11】**

前記データファイルに列タイプの識別を適用する前記ステップが、前記データファイル内の少なくとも1万個のレコードの各フィールドの内容の分析を含む、請求項1に記載の方法。

**【請求項12】**

前記データファイルに列タイプの識別を適用する前記ステップが、前記データファイル内の少なくとも10万個のレコードの各フィールドの内容の分析を含む、請求項1に記載の方法。

**【請求項13】**

前記データファイルに列タイプの識別を適用する前記ステップが、前記データファイル内の少なくとも100万個のレコードの各フィールドの内容の分析を含む、請求項1に記載の方法。

20

**【請求項14】**

少なくとも1つの基本オラクル、少なくとも1つの標準オラクル、および少なくとも1つのメタオラクルを前記データファイルに適用する前記ステップが、異なるオラクルを使用して前記データファイルの各フィールドの予想されるデータタイプに関する複数の潜在的に不確実な決定を行うステップを含み、前記データファイルの各フィールドの前記予想されるデータタイプに関する最良の選択を行うために、前記複数の潜在的に不確実な決定の結果を組み合わせるステップをさらに含む、請求項1に記載の方法。

30

**【請求項15】**

複数のフィールドをそれぞれ含む複数のレコードを含むデータファイルからファイルレイアウトを推測するシステムであって、

a. 複数の基本オラクルであって、各基本オラクルが、前記複数のフィールドの少なくとも1つ中の特定のタイプの文字の存在を決定するように構成されたソフトウェアを含む、複数の基本オラクルと、

b. 複数の標準オラクルであって、各標準オラクルが、前記複数のフィールドのうちの少なくとも1つ中に多くとも少数の共通表現を有する共通かつ頻繁に現れるフィールドタイプを識別するように構成されたソフトウェアを含む、複数の標準オラクルと、

c. 複数のメタオラクルであって、各メタオラクルが、複数の隣接フィールドを使用して、前記複数のフィールドのうちの少なくとも2つにわたる複合データタイプを識別するように構成されたソフトウェアを含む、複数のメタオラクルと、

40

d. 前記データファイルのファイルレイアウトを決定するために、前記データファイル内の前記複数のレコードの少なくともサブセットに前記基本オラクル、標準オラクル、およびメタオラクルを適用するように構成されたソフトウェアを含むオラクル分析サブシステムと

を含む、システム。

**【請求項16】**

前記複数の基本オラクルの少なくとも1つが、アルファベットオラクル、数字オラクル、英数字オラクル、数字オラクル、または空白オラクルを含む、請求項15に記載のシス

50

テム。

【請求項 17】

前記複数の標準オラクルの少なくとも1つが、住所リンクオラクル、消費者リンクオラクル、文書識別子オラクル、商号オラクル、都市オラクル、国オラクル、日付オラクル、ドメインオラクル、電子メールオラクル、ファーストネームオラクル、ラストネームオラクル、または性別オラクルを含む、請求項16に記載のシステム。

【請求項 18】

前記複数のメタオラクルの少なくとも1つが、完全な住所オラクルまたはフルネームオラクルを含む、請求項17に記載のシステム。

【請求項 19】

前記オラクル分析サブシステムが、前記基本オラクル、標準オラクル、およびメタオラクルから複数の潜在的に不確実な重複決定を受け、かつ前記基本オラクル、標準オラクル、およびメタオラクルからの前記重複決定に基づいて、複数の可能な解釈からフィールドタイプおよびフィールド位置の少なくとも1つに対して最良の選択肢を選択するようにさらに動作可能である、請求項18に記載のシステム。

【請求項 20】

前記複数のフィールドの少なくとも1つについて、基本オラクルが真の所見を返すように動作可能であり、標準オラクルが同じフィールドについて真の所見を返すように動作可能であり、前記オラクル分析サブシステムが、前記基本オラクル所見よりも前記標準オラクル所見を選択して前記フィールドタイプを決定するように動作可能である、請求項19に記載のシステム。

【請求項 21】

複数のフィールドをそれぞれ含む複数のレコードを含むデータファイルからファイルレイアウトを推測する方法であって、

a. 前記データファイル内の前記レコードの少なくともサブセットを使用して、前記複数のフィールドの少なくとも1つ中の特定のタイプの文字の存在を決定するステップと、

b. 前記データファイル内の前記レコードの少なくともサブセットを使用して、前記複数のフィールドの1つ中に共通のフィールドタイプを識別するステップと、

c. 前記データファイル内の前記レコードの少なくともサブセットを使用して、2つ以上の隣接フィールドにわたる複合データタイプを識別するステップと、

d. ステップ(a)～(c)の結果を適用して前記データファイルのファイルレイアウトを決定するステップと

を含む、方法。

【請求項 22】

前記データファイルがフィールド区切りファイルであるか固定幅フィールドファイルであるかを決定するステップをさらに含む、請求項21に記載の方法。

【請求項 23】

前記データファイルがフィールド区切りファイルであると決定された場合に、前記データファイルに対して区切り分析を実行するステップ、あるいは、前記データファイルが固定幅フィールドファイルであると決定された場合に、前記データファイルに対して固定幅分析を実行するステップをさらに含む、請求項22に記載の方法。

【請求項 24】

前記データファイルに対する前記区切り分析が、

a. 非数字、非アルファベット文字の初期頻度表を計算するステップと、

b. 可能な区切り文字のセットから試行区切り文字を使用して、前記データファイル内の各行について列数をカウントするステップと、

c. フィールドおよび行の数で前記列数を要約するステップと、

d. 少ない要約カウントをフィルタリングして除くステップと、

e. フィールドカウントをランク付けするステップと

を含む、請求項23に記載の方法。

**【請求項 2 5】**

前記ファイルに対して固定幅分析を実行する前記ステップが、

a. 前記データファイル上に空間ヒストグラムまたは文字マップの一方または両方を作成するステップと、

b. 前記空間ヒストグラムまたは前記文字マップの一方または両方を使用して前記データファイル上に列抽出をマッピングするステップと

を含む、請求項 2 4 に記載の方法。

**【請求項 2 6】**

請求項 2 4 に記載のステップ (a) ~ (c) の前記結果を適用して前記データファイルのファイルレイアウトを決定するステップが、

a. 前記データファイルの各フィールドの予想されるデータタイプに関する複数の潜在的に不確実な決定を行うステップと、

b. 前記データファイルの各フィールドの前記予想されるデータタイプに関する最良の選択を行うために、前記複数の潜在的に不確実な決定の前記結果を組み合わせるステップと

を含む、請求項 2 5 に記載の方法。

**【発明の詳細な説明】****【技術分野】****【0001】**

本発明は、データファイル内の各フィールド（すなわち、レイアウト）の、表で表される特定のデータタイプを識別するための自動化された方法であって、表では、ビジネスデータサービスやその他のビジネス目的で一般的に使用されているもののように、各行は単一のレコードを表し、各列は特定の属性／名前フィールドを表す。

**【背景技術】****【0002】**

この背景技術の項で言及する参照は、本発明に関する先行技術であるとは認められない。

**【0003】**

今日のビジネスは、既存のクライアントデータ、在庫データ、新しい顧客や製品の予測データ、重要なビジネス上の意思決定に必要なその他の関連ビジネスデータなどの、膨大な量のデータを消費する必要がある。このデータは数多くのファイルで表現されており、その中には数百のフィールドを含む数百万のレコードを含むものもある。多くの場合、そのようなファイルはクライアントまたはデータ集約会社から提供され、レイアウトまたはフィールドフォーマット情報を含まない。固定幅ファイル（すなわち、各レコードの各フィールドが同じ数の文字位置／バイトを有するファイル）は、レイアウトなしが最も一般的である。これらのデータファイルには、レイアウトの非常に正確な説明を提供するいくつかの前処理ステップが必要である。この手順を手動で実行すると、エラーが発生しやすくなり、費用がかさむ。

**【0004】**

ビジネスデータサービスに使用されるデータなどのデータを含むファイルは、一連のレコードとして編成され、各レコードにはいくつかのフィールドが含まれている。各フィールドは、特定の属性に関連付けられている。例えば、消費者データを含むレコードを含むファイルにおいて、各レコードは単一の消費者に関するものであり、各レコードに含まれるフィールドには、例えば、ファーストネーム、ラストネーム、街路住所、都市、州、郵便番号、電話番号、人口統計データ（年齢、性別、所得など）、以前の購入活動などが含まれる。しばしば、データは表として表され、表の各行はレコードを表し、各列はフィールドを表す。この種のファイルが、追加データによるファイルの拡張や、データの「衛生」（重複排除および標準化）、データ分析または他のビジネス活動を実行したりする目的などで取り込まれる（すなわち処理される）場合、ファイルの各レコードの各フィールドのデータタイプを正確に認識する必要がある。これは、業界全体でそのようなファイルレ

10

20

30

40

50

コードの標準化されたフォーマットがないという事実の結果である。従来、各レコードフィールドにおけるデータタイプ（すなわち、各列のデータタイプ）を識別するこのステップは手動で実行されている。このタイプのデータを処理する人は、単にコンピュータの画面に表示されるデータの各列を定期的に見て、見ているものに基づいて列（フィールド）にラベルを割り当てる。この手法は、人間の正確さに依存し、エラーが発生しやすく、非常に時間がかかり、費用がかさむ。これらのエラーは、識別すべき多数のフィールドが手動で順序付けられること（上記のように、レコードは数百のフィールドを含むことがある）や、多くの場合に各フィールドの識別が1つのレコードまたはごく少数のレコードにのみ基づいていることが原因で起きる。数百万の個々のレコード（行）を含んでいるかもしれないファイル内のすべてのレコードを人間が調査するのは、全く実用的でない。

10

#### 【0005】

さらに、「レイアウト」（例えば、ヘッダ行）にファイルが提供された場合、人間の調査担当者は、ファイル内のデータそのものを検証することなく、その情報に依存する傾向が強い。多くの場合、提供されたレイアウトが不正確または不完全なことがある。これは、例えば、データレイアウトがファイルの以前のバージョンからのものである場合、または誤った情報が含まれている場合に起きる可能性がある。正しいレイアウトが存在する場合であっても、各フィールドデータタイプのための標準化された命名規則がないため、レイアウト記述自体を分析してその意味を判断する必要がある。

#### 【0006】

レイアウトが提供されていない場合、一部のフィールドは、その特定のフィールドの情報を見るだけでは正確に特定できない。例えば、「y」および「n」の文字を含むフィールドは、質問に対する「はい」または「いいえ」の回答を表す可能性があるが、追加のコンテキストがなければその質問が何であるかを判断できないので、答えが何を意味するかも判断できない。

20

#### 【0007】

企業がより多くの表形式のデータを消費し続け、各レコード内のレコード数とフィールド数が増加するにつれて、重要な各フィールドに対する手動による識別およびデータタイプの検証の全体的な負荷が増加し続けている。さまざまな特定の問題により、このプロセスが非効率的になる可能性がある。そのようなデータファイルはしばしば区切られ、隣接するフィールド値は、例えば共通の区切り文字によって分離される。次の表に、カンマ区切り文字を使用するファイルの簡単な例を示す。

30

#### 【0008】

##### 【表1】

- |   |                                                        |
|---|--------------------------------------------------------|
| 1 | ジョン、ドウ、123メイン通り、リトルロック、アーカンソー州、72207                   |
| 2 | メアリ、スミス、456 1番通り、フェニックス、アリゾナ州、85001                    |
| 3 | フレッド、ジョーンズ、2006ラバーズ路地、ダラス、テキサス州、75201                  |
| 4 | マーティン、ヴァンビューレン、1902クリーブランド通り、キンダーホック、<br>ニューヨーク州、12106 |

表1

40

#### 【0009】

ただし、多くのファイルでは固定幅のレイアウトが使用されており、ここでは各フィールドの文字幅は固定されており、1つの固定文字で埋められる。表2は、空白文字をパディングとして使用するこのような固定幅ファイルの例である。

#### 【0010】

## 【表 2】

|   |       |          |         |           |         |         |           |
|---|-------|----------|---------|-----------|---------|---------|-----------|
| 1 | ジョン   | ドウ       | 1 2 3   | メイン通り     | リトルロック  | アーカンソー州 | 7 2 2 0 7 |
| 2 | メアリ   | スミス      | 4 5 6   | 1 番通り     | フェニックス  | アリゾナ州   | 8 5 0 0 1 |
| 3 | フレッド  | ジョーンズ    | 2 0 0 6 | ラバーズ路地    | ダラス     | テキサス州   | 7 5 2 0 1 |
| 4 | マーティン | ヴァンビューレン | 1 9 0 2 | クリーブランド通り | キンダーホック | ニューヨーク州 | 1 2 1 0 6 |

表 2

## 【0 0 1 1】

区切り文字ファイルの場合、従来の区切り文字は実際には予想外の位置頻度で正当なフィールド値内に現れることがあるので、区切り文字の正確な識別は困難である。また、ファイルレイアウトが提供されていても、不正確な場合があったり、一般的に使用されていない、または理解されていないビジネス特有のフィールド名が使用されたりする。これらの場合、フィールド区切りおよびレイアウト識別は、実際のデータ内容によって決定する必要がある。

10

## 【0 0 1 2】

固定幅ファイルについては、レイアウトが提供されている場合、与えられるレイアウト情報はいくつかの異なる表現で表示されることがある。すべての表現は、各フィールドの名前、レコードにおける幅および位置を示すためのものである。この各フィールドの位置情報は、一連のフィールド幅として表示されることができ、各フィールドの開始位置と終了位置は、これらの値から、または開始位置として計算されなければならない。後者の場合、先頭位置が数字 1 で表されるインデックス表現と、先頭位置が数字 0 によって表されるオフセット表現の 2 つの変形がある。表現のタイプは、レイアウトから推測しなければならない。レイアウトがない場合、フィールド位置の識別も実際のデータによって決定されなければならない。各レコードのデータに大きな差異がある大きなファイルの場合、そのような決定は困難で時間がかかることがある。

20

## 【0 0 1 3】

フィールド位置が識別されたら、後続の適切な情報を抽出して一般的に使用される従来の分析技法で検証できるように、各フィールドのデータタイプを識別する必要がある。ただし、これらの自動分析技法がデータを正しく抽出するためには、各フィールドの正確な位置およびデータタイプの識別が不可欠である。

30

## 【0 0 1 4】

正確なフィールド位置およびデータタイプ識別のための技術の現状は、上記のように、純粋に手作業プロセスを使用するもの、または場合によっては手作業を補足する自動システムを用いるものである。これらの自動化されたシステムは、名前、住所、および金額、電話番号、日付などの周知の識別文字列といった小さなセットのフィールドデータタイプを識別する。しかし、このような自動化されたシステムは、これらのデータタイプのそれぞれについて、厳格で非常に限られた予想フォーマットを組み込んでいる。データファイルはさまざまなソースやコンテキストから生成および作成されるため、これらのデータタイプの多くはさまざまな形で表示される。例えば、2016 年 10 月 5 日のデータファイルに含まれる有効なフォーマットの一部には、「10/05/2016」、「20161005」、「100516」、および「2016 OCT 05」が含まれる。また、このような日付は、単一のフィールド内で頻繁に現れるが、別々のフィールドに分割することもできる。例えば、年を月/日の情報から切り離すことができ、あるいは 3 つの構成要素すべてを複数の可能な順序で異なるフィールド内に表示することもできる。同様に、名前および住所はそれぞれ単一のフィールドまたは複数のフィールドに表示することができ、その構成要素に関してこれらのフィールドの順序は、データファイル内で様ではない（つまり、ファーストネーム/ラストネームおよびラストネーム/ファーストネームの連続フィールドは両方とも一般的である）。また、複数のデータタイプを表現できる多くの単語および文字列がある。「ワシントン」という言葉は、人の名前の構成要素、商号名の構成

40

50

要素、街路名、州、または都市を容易に表すことができる。最後に、フィールドの同じデータ表現がファイルのすべてのレコードで使用されるという保証はない。

#### 【0015】

現在の技術に存在する半自動化された方法では、単一のレコードのフィールドタイプを一度に決定しようとするそれぞれの試みや最終的な決定は、各アルゴリズムが適用される精度および／または確率の所定のランク付けされたシステムに基づいている。しかし、文字列（名前）の使用や英数字文字列（日付、住所表現など）の異なる表現の両方にばらつきがあるので、これらの手法は不確実であることに注意することが重要である。したがって、これらの手法は、上記のレコード間の不一致に起因して曖昧になりがちである。それゆえ、このような自動化されたシステムの性能および精度を劇的に向上させ、ファイルレイアウトを決定するシステムのスループットを改善し、計算サイクルを短縮するために、本発明者らは、この限定された解釈範囲が、認知的に深いフレームワークから正しい決定を下すことができる、より豊かでよりコンテキストに富んだものに拡張されるべきだと認識した。

10

#### 【発明の概要】

#### 【発明が解決しようとする課題】

#### 【0016】

本発明は、手動または半自動のファイルレイアウトを識別する方法に起因するエラーおよび非効率性を低減する必要性に対処するもので、大量のビジネスデータフィールドの処理において特に価値がある。本発明は、これらの複数の視点にわたって各フィールドのデータタイプを非常に高い精度で識別し検証する、計算処理上豊かなコンテキストおよびカスタムプロセスを利用する。これは、人間による入力がほとんどまたは全くなく、提供されたファイルレイアウトを使用しない自動化によって実現される。本発明は、データベース管理、販売、経理、ダイレクトマーケティング、および他の保守活動などであるがこれらに限らない目的でのデータファイルの取り込みを改善する。

20

#### 【課題を解決するための手段】

#### 【0017】

本発明は、各レコードの他のフィールドからのコンテキストを使用して特定のフィールド内のデータの意味を決定する手法を利用する。特定の実施例では、本発明は、特定のタイプのフィールドについて優勢な文字パターンを識別し、そのようなフィールドが「はい」または「いいえ」を表す「y」または「n」などの少数の異なる値（すなわち、列挙）を含むかどうかを判定する。レイアウトの決定は、ファイルのデータ自体を使用することのみによって達成され、データは、レイアウトの決定においてコンテキストの使用を最大限にするために、複数の視点を使用して反復的に解釈される。これらの視点には、個々のレコード内だけでなく大量のレコードセット内における、実際のデータからのデータフィールドタイプの順序付けの解釈が含まれる。また、各フィールドの潜在的なデータタイプの識別と、潜在的なデータタイプの、隣接および近接するデータフィールドの潜在的なデータタイプに対する関係とを使用して、複数の解釈が可能なフィールドの正確なデータタイプを解釈する。分析するレコードの数は、必要な計算時間と精度との所望の釣り合いに基づいてカスタマイズしてもよい。

30

40

#### 【0018】

特定の実施例では、本発明は、複数の視点からデータファイルの内容のコンテキストを見る3つの高度に相関するサブシステムを組み合わせることで、豊かなコンテキストフレームワークを作成する。第1のサブシステムは、区切られたファイルおよび固定幅のファイルの両方のフィールド位置の識別と、ファイル内で使用される文字セットおよび異なる種類の区切り文字の種類を識別するための前処理ステップとに焦点を合わせる。第2のサブシステムは、フィールドのセットの考えられる解釈を決定するために相互に作用する「オラクル（oracles）」（本明細書でさらに説明する）の複数のコンテキスト層から成る。これらの異なるオラクルの層内の相互作用およびその後の解釈は、非常にきめ細かいレベル（一度に1つのフィールド）からコンテキスト上豊かなレベル（連続また

50

は局所的に位置する潜在的に相関するフィールドのセット)までの情報を組み入れる。第3のサブシステムは、入力ファイル全体の重要ではない部分を消費し、前の2つのサブシステムの結果を使用して、データファイルの最終解釈を計算して出力し、この最終解釈は、ビジネスユースケースに特有のアプリケーションの実際のデータ自体を読み込んで解釈するために使用される。

#### 【0019】

本発明の一実施例では、最終レイアウト情報を決定するために使用するファイルの量をユーザが予め設定することができる。上記のように、一度に1つのレコードを使用すると、しばしば貧弱で一貫性のない結果が生成される。レイアウトの決定に使用する行を増やせば結果はより正確になるが、この精度の向上のためにはサブシステムの実行時間がかさむ。予想されるように、20フィールド1万行ファイルを使用するのと100フィールド1000万行ファイルを使用するのでは顕著な時間差がある。しかし、いずれの場合でも、本明細書で述べるような完全に自動化されたシステムを使用することによって節約される時間は、複数の人時から人日を要する可能性のある現状の技術水準よりはるかに多い。さらに、手作業による確認を使用する手法では、フィールドタイプを決定する際に数千、数十万、さらには数百万ものフィールドを手動で確認することは不可能であるため、フィールドタイプを決定するために確認できるフィールドの数は本質的に制限されることがわかる。本発明の1つの実施例では、例えば、1万行といった、フィールド分析で使用される行のデフォルトの固定数が指定される。ただし、ユーザは、ファイルの特定のコンテキストやサイズ、プロセスの所望実行時間、および望ましい結果の精度に基づいて、その値を簡単に変更できる。

#### 【0020】

オラクルサブシステムは、データの所定のフィールドがデータタイプの一貫した形式を有するかどうかを示すオラクルの多層集合に基づいている。本明細書で使用される「オラクル」という用語は、コンピュータシステム上で動作するソフトウェアプログラムであり、ユーザには高次の意思決定のための「ブラックボックス」と見える。単一のオラクルには、一連の入力データに反応する特定のコンテキストフレームワークがある。本発明では、それぞれの潜在的なフィールドデータタイプには、関連付けられたオラクルがある。例えば、データに「ファーストネーム」データが含まれている場合、ファーストネームオラクルが存在する。オラクルは、評価中のすべてのレコードの固定フィールドから値のセットを取り込み、続いて反応する。反応には、入力値が割り当てられたデータタイプのものであるかどうかの真理値と、関連するコンテキスト情報の小さなセットとの両方が含まれる。この情報をその後のフィールドデータタイプの決定に続けて使用することは、実際のオラクルの内部構成に依存せず、その出力のみに依存する。したがって、オラクルの反応が常に正しいという隠された仮定はない。例えば、1万レコードセットのテストでは、多数のレコードが既知のファーストネームのインスタンスと思われる場合、「ファーストネーム」オラクルはそのフィールドに正の反応を示す。したがって、これらのオラクルは、データの単一の行ではなく、ファイルの適切なレコードセット上の単一のフィールドまたは一連の連続フィールドに作用する。このコンテキストでは、所与のフィールドのすべてのレコード値を指定されたデータタイプとして解釈する必要はないが、かなりの割合の値がそのように認識される。したがって、オラクルの反応の正確さは、指定されたフィールド内の各レコードの値のタイプを識別する能力のみに依存するものではない。同様に、上記のように、人名、事業名、街路名、都市、州などに使用される多数の「名前」がある。したがって、所与のフィールドは複数のオラクルから正の反応を受けることができる。複数のタイプのオラクルからという曖昧度は、実際には、フィールドデータタイプの正確な識別という本発明の能力の主な強みである。特定の実施例では、本発明は、実際には、ファイル内のさまざまなレコードのフィールドにある必然的な曖昧さおよび不確かさの可能性を利用して、すべての列についての合理的な解釈をすべて考慮し、最も認知的に一貫性があり完全である1組の解釈を選択することによって、フィールドタイプを防御的に識別する。

10

20

30

40

50

## 【0021】

本発明のオラクルサブシステムが、ファイル内のフィールドのセットについて正確な全体論的決定を行えるようにするために、いくつかの実装例では、データタイプの観点から異なるレベルの細かさのフィールドタイプの解釈を提供する3つのレベルのオラクルがある。オラクルの第1のレベルは、空白、数字列、アルファベット文字列および英数字列などの基本データパターンを識別できるものから成る。これらは、固定幅ファイルのフィールド位置の識別だけでなく、在庫部品番号や他の独自の識別子など、データタイプが高度に特殊化されたフィールドの文字構造の定義に有用である。本発明のオラクルの第2のレベルは、ほとんどのデータファイルに共通するフィールドタイプを識別するものから成る。これらには、ファーストネーム、ラストネーム、名前のプレフィックス、街路名、単位指定子（アパート、レベル、階）、都市名といった名前および住所の構成要素の他、電話番号、日付、郵便番号、社会保障番号といった一般的に使用される数値フィールドが含まれる。ビジネスデータファイルには、個人向けの自律キーも含まれることが多く、MD5、SHA1、SHA256などの異なるハッシュ文字列に対するオラクルがあってもよい。最後に、電子メールアドレス、緯度および経度、ならびにIPアドレスなどの個人または位置についての他の識別子は、対応するオラクルを有していてもよい。

10

## 【0022】

特定の実施例では、本発明のオラクルの第3のレベルは、一連の連続するフィールドの全体的な意味を解釈するために使用される「メタ」オラクルから成る。これらのオラクルは、個人のフルネームや完全な郵便宛先など、複数のフィールドをまたぐ基本的なデータタイプの配置に有用である。上記のように、これらのデータは単一のフィールドに表示することも、隣接する一連のフィールドとして表示することもできる。これらのメタオラクルは、使用されるフィールドフォーマットに関係なく、そのようなケースを特定する役割を担っている。特に、これらのオラクルへの入力、識別されたフィールドに対するオラクルの最初の2つのレベルの出力である。上記のように、各フィールドは複数のオラクルから正の反応を持つことができる。したがって、上記のように、これらのメタオラクルは、マルチフィールドデータタイプと一致する連続フィールドのデータタイプパターンを検索する。

20

## 【0023】

特定の実施例において、これらのオラクルはまた、それらが見つけた異なるデータ値の頻度を記録する能力を有する。これは、性別などの列挙されたフィールドを識別するのに便利であり、ここでは少数の値またはコードのセットのみがデータを表現するために使用される。同様に、これらのオラクルは、関連するデータタイプであるという明確な証拠を有するフィールド内のレコード数もカウントする。これらのカウントは、同じタイプの情報を含むと思われるフィールドの曖昧さを解消するのに非常に有用である。異なるコンテキストレベルのオラクルの相互作用およびフィールドに特有の情報は、一連のフィールドの内容の非常に豊かかつ明確なビューを示しており、これは多種多様なデータファイルに対して単一のオラクルレベルのビューでは提供できないものである。

30

## 【0024】

特定の実施例では、フィールド位置サブシステムは、区切られたファイルおよび固定幅ファイルの両方を処理する。区切られたファイルの場合、システムは、大部分の句読文字および特殊なスペース文字が含まれる最も頻繁に使用される区切り文字のセットから潜在的な区切り文字を探す。各区切り文字の頻度および位置がレコードのセットについて発見され、セット内のすべてのレコードにわたって行ごとの可能な区切り文字のカウントの整合性および差異が測定される。すべてのレコードで一致するように思われる単一の区切り文字がある場合、オラクルサブシステムを呼び出して、結果のフィールドのサブセットを判別して、コンテキスト上意味があるように思われる（例えば、ファーストネームフィールドとラストネームフィールドとの間に郵便番号がないと思われる）ことを確認できる。潜在的な区切り文字が複数ある場合は、オラクルサブシステムを使用して、各区切り文字から構成されたフィールドの第1のパスの解釈に基づいて、最も可能性のある区切り文字

40

50

を区別できる。一部のフィールドには、そのデータタイプの有効な文字として可能性のある区切り文字を含めることができるので、すべての行で一致する必要はない。しかし、これらの場合、オラクルサブシステムは、解釈に必要な一貫性のためにフィールドタイプを検査することがある。例えば、いくつかのレコードに余分なカンマがあると、残りのレコードが一貫した構造であるので、分析されるフィールドが正しく分析される。

#### 【0025】

特定の実施例では、固定幅ファイル内のフィールドの識別には、さまざまな異なる視点が必要である。特に、ファイル内の実際のデータから生成されたいくつかのタイプの画像に対して画像処理エッジ検出技術を用いることができる。使用される1つの画像は、文字が非空白文字である場合、ファイルの各（行、列）文字を1（白）にマッピングし、それ以外の場合は0（黒）を画像内の（行、列）位置にマッピングすることによって、データファイルの行および列を画像の行および列にマッピングするバイナリのものである。少なくとも一部のフィールドのエッジは、隣接する列単位で白い部分が最大である列によって識別することができる。この技法は、大部分のデータ値にブランクのパディングが含まれている列に対してうまく機能する。第2の画像は同様の方法で作成されるが、第1のレベルのオラクルを使用して、異なる色の使用により異なるタイプの非空白文字を区別する。本発明の一実施例では、空白文字、アルファベット文字、数字、および句読点のそれぞれに異なる色が割り当てられ、これらの色の分布は、上記の方法では見つからない列や特定のフィールド内の異なる文字パターンを識別するために使用される。この最後の観察では、他のタイプのフィールドと比較して、完全な住所フィールドを明確に識別することができる。最後に、オラクルサブシステムは、既存の識別されたフィールドで使用され、いくつかのフィールドが実際には上記の方法では検出されなかった識別可能なデータタイプを備える隣接フィールドの連結であるかどうかを判断する。

#### 【0026】

特定の実施例では、第3のサブシステムは、ファイルのレイアウトの識別と確認のための実際のプロセスで構成され、その処理において前の2つのサブシステムを複数の方法で使用する。このサブシステムは、ファイルのレイアウトの決定を導く役割を担っている。このサブシステムは、分析に使用されるレコードのサブセットを取り込み、ASCIIやUTF8などのファイルの作成に使用される文字エンコーディングとレコード区切り文字とを判別することから始まる。ファイルの適切なメタデータが見つかり、サブシステムは、ファイルが区切られたものか固定幅のものかを判断する。上記のように、ファイル内に潜在的な区切り文字がある場合、その結果のフィールドデータタイプのサブセットの識別は、選択された区切り文字の判別で行われる可能性がある。このプロセスは、（そのような区切り文字が存在する場合）2つの以前のサブシステムを織り交ぜて最終的に区切り文字を識別する。区切り文字が識別されない場合、このサブシステムは2つの文字タイプ画像を作成し、後に続く画像処理アルゴリズムを実行して、すでに識別された固定幅ファイル上のフィールド位置の第1のパス判定を見つける。ここでもまた、このサブシステムは、オラクルサブシステムを呼び出して、フィールド位置の最終的な決定に役立てることができる。

#### 【0027】

特定の実施例では、フィールドの場所が識別されると、オラクルサブシステムが呼び出されて、個々のフィールドおよび連続サブセットフィールド両方の、可能性のある異なる解釈のビューが作成される。これらの解釈は評価され、フィルタリングされて、最終的にデータファイルの最終レイアウトが識別される。このレイアウトは、その後、一実施形態では全ファイルのデータを消費する後続の下流プロセスによって消費されるようにエクスポートされる。

#### 【0028】

完全に正確なシステムはないので、結果のレイアウトにエラーが含まれている場合には、当然訂正を行うことができる。本発明者らが行ったテストでは、区切り文字ファイルおよび固定幅ファイル両方について異なるビジネスファイルレイアウトを大量にサンプリン

グしたところ、処理時間を数秒に短縮して、97%を超える精度を有する本発明の特定の実施例が示された。この精度の値は、手動または混合システムで報告されている値よりも大きい値である。識別されたエラーは、既存のデータ値のパターンの不一致のために人間が識別することが極めて困難または不可能な、データが非常にまばらな領域で主に発生しており、この問題に対する先行技術の手法ではこれよりよい結果は期待できないだろう。

#### 【0029】

本発明のこれらおよび他の特徴、目的および利点は、以下に記載する図面と合わせて、以下の好ましい実施形態および添付の特許請求の範囲の詳細な説明を考慮することにより、よりよく理解されるであろう。

#### 【図面の簡単な説明】

10

#### 【0030】

【図1】本発明の実施例によるファイルレイアウト推測システムの高レベルアーキテクチャ図である。

【図2A】本発明の実施例による基本オラクルの動作を示す図である。

【図2B】本発明の実施例による標準オラクルの動作を示す図である。

【図2C】本発明の実施例によるファイルレイアウト推測システムのオラクルサブシステム内の3つのレベルのオラクル間の相関の高レベルな概要である。

【図2D】本発明の実施例によるファイルレイアウト推測システムの3つのタイプのオラクルの例を提供する。

【図3A】本発明の実施例によるファイルレイアウト推測システムのフィールド位置サブシステムの区切られた列分析部分の詳細なアーキテクチャフローである。

20

【図3B】本発明の実施例によるファイルレイアウト推測システムのフィールド位置サブシステムの固定長列分析部分の詳細なアーキテクチャフローである。

【図3C】本発明の実施例による文字位置の例示的なヒストグラムである。

【図3D】本発明の実施例による例示的な文字マップである。

【図4】本発明の実施例によるファイルレイアウト推測システムの最終フィールドデータタイプ識別サブシステムのプロセスの詳細なアーキテクチャフローである。

【図5】本発明の実施例によるハードウェアシステムの概略図である。

#### 【発明を実施するための形態】

#### 【0031】

30

本発明をさらに詳細に説明する前に、本発明は記載された特定の実施形態および実施例に限定されず、特定の実施形態および実施例の説明に使用される用語は、それらの特定の実施形態および実施例を説明するためだけのものであり、本発明の範囲は特許請求の範囲によってのみ限定されるため、用語は限定することを意図するものではないことを理解されたい。

#### 【0032】

図1を参照して、本発明の実施例によるファイルレイアウト推測システムの全体的なアーキテクチャ設計を説明する。入力ファイルのレコードの適切なサンプルを10とすると、補足情報処理12は、ファイルの符号化に使用される文字セットの決定と、フィールドレイアウトを記述する先頭行がファイルに含まれるように思われるかどうかを決定することを含む。

40

#### 【0033】

予備分析14は、主にレコード区切り文字の決定に焦点を当てている。これは、本発明の特定の実施例で 사용할 ことができる1つ以上のプログラミング言語に特有の1つ以上の「readline」関数を使用することによって行うことができる。このような利用可能な「readline」関数のそれぞれについて、関数は少数回呼び出され、結果の文字の行が比較される。各行が同じ1文字または2文字で終わり、行の長さに劇的な違いがない場合、区切られたレコードはこの共通のサフィックスに設定される。一方、そのような方法が利用できない場合、または上記の基準を満たさない場合、共通の「end of line」文字区切り文字のセットを調べて、何がファイルを行に分割しているかを

50

検査することができる。これらの文字には、「ラインフィード」(LF: Line Feed) および「キャリッジリターン」(CR: Carriage Return) が含まれる。これらの文字や LF+CR または CR+LF のような 2 つの文字の組み合わせは、ファイルを行セグメントに分割するために使用される。各分割は、ファイルを合理的な行のセットに分割するという観点から再度評価される。最良の結果を示す文字分割がレコード区切り文字として選択される。

#### 【0034】

レコード区切り文字が決定されると、各レコードの文字長といった追加の一般的なレコード属性が決定される。これらの属性は、ファイルが可変幅フィールドの区切りファイルか固定幅ファイルのどちらであるかを示す強力な初期ヒントを与える。レコードの文字長さの大部分が等しい場合、ファイルが固定幅フォーマットを有する可能性が非常に高くなる。その場合、処理の効率を高め、タスクを完了するために必要な処理サイクルを短縮するために、次のフィールドタイプ分析の順序を切り換えることができる（すなわち、固定幅分析が最初に行われる）。

#### 【0035】

この予備分析 14 が完了すると、フィールド位置サブシステムが展開される。このサブシステムの区切られた分析部分 20 は、数および結果として得られるフィールドタイプの両方が一貫している可能性のあるフィールド区切り文字があるかどうかを決定するが、これについては後に詳細に説明する。そのような区切り文字がある場合、それは識別され、ファイルはフィールドタイプの識別の準備が整っている。識別された区切り文字がない場合、固定幅の分析 22 が実行される。この分析は、上記の画像処理技術を介してフィールド位置を識別し、この場合、ファイルは再び次のフィールドタイプ識別処理の準備が整っている。

#### 【0036】

次に、フィールドタイプ識別 18 は、識別されたフィールド位置ごとに各非メタオラクルの結果を収集する。次にこれらの結果は、次のセクションで説明するように、フィールドタイプの数、および各フィールドおよびその隣接フィールドの潜在的なフィールドタイプのコンテキストの観点から解釈される。メタオラクルの使用および異なるタイプのフィールドの予想される連続処理が、最終的な解釈を決定するために使用される（すなわち、郵便番号は街路名の直前には予想されない）。最後に、オラクルサブシステム 16 は、フィールドタイプ分析のための確かな証拠、およびフィールドタイプを識別する際の最終決定の基礎を含む。上記のように、このオラクルサブシステム構造により、すべてのレコードのフィールドタイプを明確に解釈せずに、最終的な決定を高い精度で行うことができる。

#### 【0037】

図 2 A は、各基本オラクルの全体的なアーキテクチャ設計およびフローを示し、図 2 B は、オラクルサブシステム内の各標準オラクルの全体的なアーキテクチャ設計およびフローを示す。潜在的に識別される各データタイプには、対応するオラクルがあり、そのジョブは「指定されたフィールドはもしかするとあなたのデータタイプか？」という質問に答えることである。これは、指定されたフィールドのサンプルレコードの実際の値を単一のデータセットとしてオラクルに渡すことで実現する。

#### 【0038】

各オラクルは、オラクルの割り当てられたタイプの値を識別するために使用されるフレームワークを設定することによって構築される。図 2 A に描かれているような基本オラクルは、空白、アルファベット、数字、および英数字（基本データタイプ）といったレコードセット 90 からのフィールド値における特定のタイプの文字の存在を探すだけである。これは、実装コンピュータプログラミング言語特有の文字識別関数（「isNumeric」、「isAlpha」など）を直接使用するか、実装コンピュータプログラミング言語の組み込み正規表現ライブラリでサポートされている非常に単純な正規表現によって実現される。この処理はステップ 92 で実行される。データファイルの一部のフィールドに

は、ピリオド、アポストロフィ、ハイフンなどの予期しない文字は非常に少数しか含まれていないため、オラクルは、そのような予期しない文字がいくつあったかを検査できる。そのようなケースがごくわずかである場合、文字が削除され、値が再度検査されて、変更された値の文字列が期待されたパターンを有するかどうかを確認される。有していれば、値は一致すると見なされるが、そのような小さな変更の後にのみ一致する。例えば、「数値」オラクルの場合、入力文字列「1 2 3 4. 5 3 6」は数値として識別されない。しかし、文字列にピリオドが1つしかなく、それを削除した後にできる文字列「1 2 3 4 5 3 6」は非常に軽微な変更が加えられた「数値」として識別される。

#### 【0039】

オラクルのデータタイプパターンについて各値が検査されると、追加情報が計算される。特に、オラクルのデータタイプを有すると識別された個別の値の分布が計算され、オラクルに渡されたフィールド内の非空白値の総数と、オラクルのデータタイプを有するとして識別される前に軽微な変更が必要であった値の数とが計算される。値は、ステップ96で集約され、フィールドがオラクルのデータタイプであるかどうかの最終決定に使用される。

10

#### 【0040】

識別された個別の値の分布は、後でフィールドが実際に列挙されているかどうかを判断するために使用される。例えば、「1文字」の基本オラクルが正の反応を返し、個別の値の分布が5000個の「M」、3500個の「F」、および4000個の空白である場合、すべてのフィールドタイプの最終決定の労力の結果、この列が列挙であることがわかり、この特定のフィールドの近くにあるフィールドタイプに応じて、そのフィールドが性別値を表すと判断することができる。基本タイプフィールドが列挙であるかどうかを区別するこの機能は、後で説明するように、ファイルのフィールドタイプの最終決定の精度においてしばしば重要である。

20

#### 【0041】

このデータがステップ94で計算され、ステップ96で集約されると、オラクルはステップ98でその決定を行う。非空白値の合計の大部分が変更の必要がないデータタイプを有すると識別された場合、オラクルはフィールドをオラクルのデータタイプを有するとして識別し、100で出力する。変更が必要な場合、オラクルは、適切なごく一部の値のみが識別された場合に限り、軽微な変更の後にのみ、同じ結論に達するかどうか検査する。それ以外の場合、オラクルは正のデータタイプ識別を主張しない。最後に、オラクルがそのフィールドのデータタイプであると識別しない場合、100で負の反応が返される。一方で、オラクルがフィールドをそのように識別した場合、正の反応および収集されたデータの報告の両方が100で返される。

30

#### 【0042】

図2Bは、オラクルサブシステム内の各標準オラクルの全体的なアーキテクチャ設計およびフローを示している。標準オラクルは、多くても少数の一般的な表現を有する共通かつ頻繁に現れるフィールドタイプを識別する。そのようなフィールドタイプには、電話番号、日付、名前、標準ハッシュ、個人識別子、および住所／位置情報が含まれる。しかし、基本オラクルとは異なり、これらの標準オラクルは、特定のデータタイプを識別するために、正規表現や言語特有のタイプの関数以外の手法を使用しなければならない。例えば、ファーストネーム、ラストネーム、街路名、商号、都市、州などの名前データタイプを識別するオラクルは、それぞれのタイプに対して有効なインスタンスの辞書を使用する。上記のように、オラクルは各レコードのフィールドでタイプを識別する必要はないので、これらの辞書は予想される名前を包括的に含む必要はなく、むしろ強力な統計的な範囲を含んでいればよい。また、このコンテキストでは、一部の名前フィールドは、特定のタイプを示す共通の文字パターンを表す文字列を探す正規表現拡張をサポートすることができる。例えば、「ville」、「ham」、「ford」または「ton」で終わっていたり、「new」、「little」またはコンパス方向から始まっていたり、「falls」などの風景実体を含んでいたたりする文字列は、都市である可能性が高い。

40

50

## 【0043】

同様の方法で、有効な電話番号、日付、郵便番号などの主として数値のデータは、指定されたパターンに従う必要がある。これらのデータタイプの標準オラクルは、文字タイプのパターンに加えて、これらのパターンを検査する必要がある。例えば、最近まで、社会保障番号は、地理的および数字の両方のパターンに基づいた特定のルールに従うことが求められていた。これらのルールは2011年に廃止されたが、大部分の人々はこれらのパターンに従った社会保障番号を持っている。このようなフィールドのデータタイプを正しく識別するために、オラクルは、十分な数のこれらのタイプの値を識別するだけでよいので、社会保障のオラクルは、2011年以前のルールが有効で正確であるかどうかを検査するだけでよい。したがって、ステップ102において、各値は構成要素に分割され、構成要素はオラクルデータタイプパターンについて検査され、次に、基本オラクルと同様に、ステップ94の、各値について収集されたフィールド間の位置情報に処理が進む。

10

## 【0044】

そのような情報を含むデータファイルに対する一般的な予想は、そのような各フィールドが単一の表現を使用することである。しかし、これは必ずしも当てはまるとは限らず、したがって、これらのオラクルは、すべてのレコードのフィールド値ごとに、指定されたタイプの多種多様な可能な表現を考慮する。例えば、日付オラクルは、2016年1月7日の日付、01/07/2016、01/07/16、07JAN2016、01072016、010716、1716や他の同様のバリエーションなどの可能性のある表現に敏感でなければならない。これら2つの妥当性検査は、それぞれが他方に直接依存するので、ステップ102において同時に行わなければならない。したがって、潜在的な表現の1つが無効な日付になる場合、オラクルは別の潜在的な日付形式の識別を試みる必要がある。処理は、そのオラクルに一致するデータタイプパターンで各値について集められた情報の集約に進む。有効な日付解釈をもたらす表現が見つからない場合、日付オラクルは、ステップ106で、そのデータタイプを持たない値を考慮する。いずれの場合も、基本オラクルと同様に、真理値と収集された情報とがステップ100で返される。

20

## 【0045】

個人のファーストネームやラストネームが同じフィールドにある場合、または住所番号や街路名の両方が同じフィールドに表示されている場合、標準オラクルは、たとえそのフィールド内に他のデータがあっても、フィールドがそのタイプのデータを含むように思われるかどうかを識別しなければならない。これらの場合、オラクルはそのようなフィールドのデータタイプインスタンスの位置を記録できる。例えば、フィールドに「ジョン・スミス」などの複数の名前構成要素のインスタンスが多数含まれている場合、「ファーストネーム」オラクルは、「ジョン」および「スミス」のように完全なデータ文字列を空白で分割して取得した各名前構成要素を分析する必要がある。このオラクルは、第1の名前構成要素は「ファーストネーム」であるが、第2の構成要素は「ファーストネーム」ではないことを記録する。同様に、「ラストネーム」構成要素は適切な反対の情報を報告する。値の文字列表現の各構成要素を分析するこの能力は、フィールドのデータタイプの最終的な意思決定において重要である。したがって、標準オラクルの第1のステップは、ステップ102において各値を空白で区切られた構成要素に分割することである。次にオラクルは、以前の基本オラクルの場合と同様に機能して、各構成要素がオラクルのタイプであるかどうかを判断する。フィールド値に構成要素が1つしかない場合は、上記の基本オラクルと同じ方法で処理される。一方、複数の構成要素がある場合、オラクルがステップ100で収集した情報を渡すために、構成要素の少なくとも1つをオラクルのデータタイプとして識別しなければならない。フィールドの値の少なくとも1つの構成要素がオラクルのデータタイプとして識別される場合、基本オラクル中にあるのと同じタイプの情報が収集される。しかし、値の文字列内の情報のタイプを正しく解釈するためには、追加のコンテキスト情報を追加する必要がある。この情報は、ステップ104で決定された、値の文字列内にある構成要素の数、および構成要素のうちのどれがオラクルのデータタイプであると識別されたかから成る。したがって、入力住所文字列「123ワシントン通り」の場合

30

40

50

、「街路番号」オラクルは、その文字列を、その街路名が3つの構成要素の最初に表示されるといふ追加情報とともに報告する。対応する情報は、「街路名」オラクルと「街路サフィックス」オラクルとによって報告される。これらの文字列には多少の曖昧さが含まれている可能性があるため、「ラストネーム」オラクルは「街路名」オラクルと同じ情報を報告することに注意することが重要である。この場合、実際のオラクル自体（それ自身のデータタイプ）は、後続の処理が行われるときの解釈を区別する。

#### 【0046】

オラクルの処理の次のフェーズは、基本オラクルの関連ステップ、すなわち識別された文字列（識別された少なくとも1つの構成要素）に類似しており、収集された情報はステップ104で意思決定のために集約される。意思決定のステップは基本オラクルの情報に非常に似ているが、1つの小さいが非常に重要な違いがある。特に、十分な値が正で識別されている必要があるだけでなく、どの実際の構成要素が実際に正で識別されているかについての強い支配的パターンも存在しなければならない。例えば、住所に関する上記のケースを検討する。すべての値に2つまたは3つの構成要素がある場合、「街路サフィックス」オラクルは、識別されたケースの大部分が最後の構成要素に対して正の反応を持つ場合にのみ、フィールドにそのようなデータタイプが含まれていると判断する。同様に、最後から二番目の構成要素（街路番号のない街路住所の第1の構成要素であり得る）が主に識別された構成要素である場合、「街路名」（および場合によっては「ラストネーム」）オラクルは、正の識別を返す。また、基本オラクルについては、標準オラクルはその決定と関連情報とを返す。予想されるように、正で識別されたフィールドについてこれらのオラクルが報告する付加的な情報は、ステップ106における全ファイルのデータタイプレイアウトの最終決定において重要である。

#### 【0047】

基本オラクル30および標準オラクル32は、フィールドの異なる解釈に関連して曖昧さがある場合、相互に作用する。例えば、数字の基本オラクル30および日付の標準オラクル32が、同じフィールドに正と反応することがある。これは、この潜在的な日付フィールドの表現が、句読点表記を使用するもの（01/31/2016）ではなく、数字のみのもの（01312016）であることを示している。複数の主要な数字タイプの列を見つけるのは珍しくない。基本オラクル30が表現の主要な特徴を識別したという事実は、そのようなフィールドの他のフィールドタイプへの位置付けが数字フィールドの解釈に役立つので、より良い意思決定を可能にする。そのような場合、フィールドにはしばしば有効な日付であるように思われるビジネス特有の情報が含まれるため、正しい解釈はただの数字のフィールドとなる。

#### 【0048】

図2Cに示すように、第3の（またはメタ）レベルのオラクル34は、以前のレベルの結果の出力の取り込みを試み、隣接フィールドがフルネームまたは完全な住所などのより複雑なデータタイプを構成しているかどうかを決定する。これらのより大きなデータタイプを形成する潜在的な解釈を有するフィールドの連続したシーケンスの結果を考慮する。これは、表3に示すデータの例で説明する。

#### 【0049】

【表3】

| オラクル     | ジョージ | ワシントン | カーバー |
|----------|------|-------|------|
| ファーストネーム | ✓    | ✓     |      |
| ミドルネーム   | ✓    | ✓     |      |
| ラストネーム   |      | ✓     | ✓    |
| 都市       |      | ✓     |      |

表3

#### 【0050】

例えば、上記のように連続したフィールドに表示される「ジョージ・ワシントン・カーパー」という名前を検討する。第1のフィールドは、ファーストネームオラクルおよびミドルネームオラクルの両方から正の反応を受け、次の第2のフィールドは、ファーストネーム、ミドルネーム、およびラストネーム、および都市名オラクルから正の反応を受け、第3のフィールドは、ラストネームオラクルから正の反応を受ける。この場合、3つのフィールドの1つの連続的な解釈は、ファーストネーム、ミドルネーム、ラストネームである。名前メタオラクル34は、フルネーム表現（この場合はファーストネーム、ミドルネーム、ラストネームの組み合わせ）と一致するパターンを見つけるために、この一連の連続フィールドについて識別されたデータタイプのすべての組み合わせを調べる。したがって、この場合、フルネームメタオラクルは正の識別情報を返す。

10

#### 【0051】

図2Dは、表40内の基本オラクル30、標準オラクル32、およびメタオラクル34のセットの単純化された例を提供する。提示される基本オラクル30は、特定のビジネスデータファイルにおいて最も頻繁に現れるものである。特定の句読文字を含むパターンを含めるために、これらのオラクルを拡張する必要があることもある。標準オラクル32は、名前および住所の構成要素、いくつかの特定の識別子、標準匿名ハッシュ値、IPアドレス、地理的位置、社会的な肩書き、電話番号、および性別の識別を可能にする。さらにこのリストは、車両識別番号、一般的な在庫コード、およびいくつかのバイナリ列挙（はい／いいえ、真／偽、有効／無効）のような他の値によって増強することができる。識別されたフルネームおよび完全な住所の2つのメタオラクル34は、ほとんどのビジネスデータファイルに見られる最も一般的なものである。

20

#### 【0052】

図3Aは、図1の区切られた列の分析20をより詳細に示す。このプロセスは、まず、入力データファイル10が（可変長の）区切りフィールドを有するかどうかを決定し、区切りフィールドがある場合、そのファイル特有のフィールド区切り文字を決定する。第1のステップは、ステップ50で、ファイルのサンプルレコードに現れる非数字および非アルファベット文字の頻度を計算することである。多くの場合、このような文字はファイルの区切り文字だが、その保証はない。この初期頻度分布が計算されると、ステップ52で、それぞれの適切な区切り文字候補を使用すると判明したフィールドの数がカウントされ、ステップ54で、これらの列カウントがフィールド数およびレコード数で要約される。比較的少ない要約カウントを有する区切り文字は、ステップ56でフィルタリングされて除かれる。この時点で、残っている区切り文字のみがサンプルレコード全体で一貫している。ステップ58で、最終的な区切り文字を決定するために、図2Cのオラクルサブシステムから基本オラクル30／標準オラクル32がこれらの区切り文字の関連フィールドセットに対して呼び出され、結果が収集される。区切り文字は、オラクルサブシステムによって認識されるフィールドの数および総レコード範囲の観点からランク付けされ、ステップ60で、最終的な区切り文字が選択される。

30

#### 【0053】

図3Bは、図1の固定幅列の分析22をより詳細に示す。この図で表される固定幅列分析は、ファイルのフィールド区切り文字が識別されていない場合に、フィールドの固定幅位置の計算を試みる。第1のステップは、ステップ70で、列に対するパディング空白のために明らかであるフィールド境界を識別することである。図3Bでも説明したが、図3Cに示すように、このステップで、サンプルレコードの空白ヒストグラムが計算される。これは、各サンプルレコード内の各文字位置について、空白以外の文字がその位置にある回数をカウントすることによって行われる。図のヒストグラムは棒グラフで、各位置の灰色の棒の高さがこのカウントを示している。このタイプのファイルは一貫して各フィールドを同じ方法でパディングするので、大きなカウントから小さいカウントへ、あるいは逆の劇的な変化を示す位置はフィールド境界を示す。図3Cの空白のヒストグラムは、横軸上の円でこの種の境界を示す。

40

#### 【0054】

50

これらの境界が識別されると、ステップ72で、サンプリングされたレコードの文字マップが計算される。このマップは、基本文字タイプを示す色または陰ピクセルによって各文字を符号化する。一例を図3Dに示す。この図の所与の文字タイプでは、白は空白を表し、黒はアルファベット文字を表し、濃いグレーは数字を表し、明るいグレーは句読文字を表す。このイメージは、異なるデータタイプのフィールドを色の規則性およびパターンによって明確に描写するので、非常に表現力がある。図示の例では、第1のフィールドはファーストネームであり、第2のフィールドはミドルネームのイニシャルであり、第3のフィールドはラストネームであることが明らかである。次のフィールドは、住所番号および街路名と、二次的な単位指定子（アパート番号）との両方を含む完全な街路住所フィールドである。住所「123メイン通り」は、このパターンに適合する住所の例である。次のフィールドは都市と思われ、続く2つの小さな文字フィールドは州の略語である。次にZIPコードが表示されるが、他のフィールドの解釈には追加のコンテキストが必要である。右側の列の1つに示される別のパターンは電話番号で、7つの数字で構成されていることから決定される。列境界の識別は数値型の列が隣接していると特に困難であり、空白ヒストグラムに基づいてすべての列境界を識別できるわけではないため、各文字のタイプのカウントは、文字が空白、アルファベット、数字、または句読タイプの文字であるかどうかに基づいて、各文字位置に対して行ってもよい。

#### 【0055】

文字マップの表現力は、フィールドの位置を識別する中心的な役割を果たすだけでなく、多くのフィールドの共通の使用順序に基づいて多くのデータタイプを視覚的に示すためにも使用できる。ステップ74で、これら2つのマッピングでフィールド（列）位置が識別されると、ステップ76で、（基本オラクル30および標準オラクル32を使用して）オラクルサブシステムが呼び出され、列の位置の確認に使用される。ただし、フィールドの中には、サンプル内のレコードからのエントリがほとんどないものがある。したがって、分析結果が不明確である場合は、より大きなサンプルを使用してファイルのフィールドの位置およびタイプを再評価することができる。

#### 【0056】

図1を参照して説明したように、一度フィールドの位置が識別されると、ファイルのフィールドおよびデータタイプの最終的な識別を担当する第3のサブシステムは、列タイプ識別18として、ファイル内の各フィールドのデータタイプを識別するために進む。第1のステップとして、オラクルサブシステム16（図1）が呼び出され、基本オラクル30および標準オラクル32のそれぞれは、上記の識別技術によって、識別された各フィールドについて分析し、反応を返すが、これらはそれぞれステップ80および82で行われる。上記のように、異なるオラクルが同じフィールドに対して正の反応を返すことがあり、しかも頻繁に起こるが、これは個人、商号、街路、および都市名で使用される共通語の大きなセットがあるためである。その後の分析の準備のために、各フィールドについて、反応および補助データ（上記の値頻度分布、同一フィールド内のインスタンス数、データ値がオラクルの識別基準を満たすレコード数、および位置情報）が、これらのオラクルから収集される。各フィールドは、今後の分析のためにこれらの2つの結果セットを記録する。

#### 【0057】

上記のデータが収集されると、ステップ84で、メタオラクル34が使用されて、フルネームまたは完全な住所のように単一の大きなデータタイプを構成する隣接するフィールドがあるかどうかが判定される。これらの大きなデータタイプのそれぞれは基本的な構造パターンを持っている。例えば、フルネームのデータタイプは、いくつかの隣接するアルファベットの「名前」データタイプから形成される。フルネームが置かれたフィールドは1つだけでよく、一般的にフルネームを保持するフィールドは最大5つ連続していてもよい（名前タイトル／プレフィックス、ファーストネーム、ミドルネーム、ラストネーム、名前世代サフィックス）。完全な住所フィールドも、アルファベット、数字、および英数字のフィールドタイプが混在する1つ以上の連続したフィールドから形成することができ

る。したがって、各メタオラクル 3 4 は、可能性のあるすべてのフィールドシーケンスを考慮する必要はなく、シーケンスがその大きなデータタイプ内で一貫している複数の値を含む単一のフィールドを探すことから始まる。次に、オラクルは、データタイプの組み合わせがオラクルの大きなデータタイプのサブパターンを形成する連続したフィールドの対（例えば、数字フィールドおよび街路名フィールド）を識別することができる。そのような候補が見つかり、メタオラクル 3 4 は隣接するフィールドを検索して、候補について報告されたデータタイプと新しいフィールドとの組み合わせがオラクルのデータタイプと一致するかどうかを調べ、一致すればこれらの新しいフィールドは候補セットに追加される。このプロセスは、一貫性のある方法で新しいフィールドを追加することができなくなるか、データタイプを構成するフィールドの完全なセットが見つかるまで続く。このプロセスは、候補セットに含まれていないフィールドのセットの全体にわたって続けられる。2つの候補間で潜在的なフィールドが重複する場合、重複するフィールドを以前のグループのフィールドから削除することができ、以前のグループがオラクルのデータタイプの完全なインスタンスでない場合は現在の見込みフィールドに追加することができる。先に例を挙げたように、フィールドには曖昧さを生じる可能性のある複数の正の反応があることがよくある。したがって、メタオラクル 3 4 は、フィールドの潜在的なシーケンスを調べ、これらの複数の反応をまとめて有効なメタタイプを形成できるかどうかを判断しなければならない。すべての候補セットが識別された後、関連付けられたデータタイプの完全なインスタンスを形成するものが、メタオラクルによって識別される。

#### 【0058】

このようなメタタイプが識別されると、ステップ 1 8 で、残りのフィールドタイプが決定される。この決定は、連続解釈が有効である最大数のフィールドのデータタイプを選択することに基づく。選択されたフィールドタイプのシーケンスの有効性は、データタイプの一貫性および共通の関係パターンに大いに基づいている。使用される関係パターンには、一連の個人の名前フィールドにいかなる形式の数字データタイプのフィールドも含まれないという前提がある。州の略語や郵便番号などの住所構成要素は、都市名や街路名などの他の標準住所データタイプに隣接しないフィールドにはないというルールもある。別の前提には、ファイルの大部分に日付フィールドは通常は 2 つしかないということがあり、そのようなフィールドが 2 つある場合、それらは通常隣接しているか、互いを隔てるフィールドは非常に少ない。説明のために、表 4 にデータ例を示す。

#### 【0059】

【表 4】

|          |   |   |   |   |
|----------|---|---|---|---|
| ファーストネーム | ✓ |   | ✓ |   |
| ラストネーム   |   |   | ✓ | ✓ |
| アルファベット  | ✓ |   | ✓ | ✓ |
| 数字       |   | ✓ |   |   |
| 電話       |   | ✓ |   |   |
| 商号       | ✓ |   |   | ✓ |

表 4

#### 【0060】

表 4 の例では、フルネームメタオラクル 3 4 は、第 2 の数字のフィールドが原因で、4 つのフィールドをフルネームにグループ化しないであろう。最後の 2 つのフィールドは、（フルネームメタオラクル 3 4 がフルネームとして識別するであろう）フルネームデータタイプのバリエーションを形成する。そのようなデータファイル内でファーストネームが離されるのはまれなので、第 1 のフィールドは、ファーストネームフィールドとして識別されない。第 2 のフィールドは潜在的な電話番号であり、第 1 のフィールドも商号と予想されるため、これらの 2 つのフィールドをそれぞれのタイプで識別することは、ビジネスデータファイルのレイアウトの一般的な習慣に従う。一方、第 1 のフィールドが潜在的な商号として識別されず、別のフィールドがすでに電話番号として識別されている場合、第

2のフィールドは数字フィールドとして識別される。1つのタイプのアルファベットまたは数字フィールドが、いくつかの非常に異なるフィールドタイプ（すべての数字日付および識別コードは共通インスタンスである）に対して強いパターンを有することは珍しいことではない。したがって、適切な特定のまたは一般的なデータタイプが正しく選択されるように、基本および標準オラクルの両方にこのようなフィールドを解釈させることが重要である。

#### 【0061】

すべてのフィールドがこのように処理されて、基本タイプとされたフィールドが多すぎるように見える場合、連続パターンの一貫性を維持しながらそのような基本タイプのフィールドの数を減らすために、軽い調整を行う意図でフィールドをまたがる第2のパスができることがある。これは、強力な証拠を有する複数の潜在的な候補データタイプを有するフィールドを再検討することによって行われる。これらのいずれかが基本データタイプが割り当てられたフィールドに隣接する場合、そのような強力な候補のそれぞれを置換して、結果としてできるフィールドの解釈の一貫性を維持しながら基本データタイプフィールドを通常データタイプに合わせられるかどうかを確認する。

10

#### 【0062】

図5は、本技術分野で現在行われている手動システムと本発明の特定の実施例とのデータファイルのレイアウトを決定する時間の比較を表している。示されているこれらの時間は、サンプルレコードの初期分析から最終的な列データタイプの最終決定および出力までである。本発明のこのインスタンスは、Red Hat Enterprise Server 6.7を実行中の4つのCPUおよび16GBのRAMを搭載した単一のLinux x86\_64プロセッサマシンに実装された。すべてのサブシステムを実装するためのコードは、Python 3.4.2で書かれている。Webサービス用のFlaskマイクロフレームワークのインスタンスがインストールされ（BSDライセンス—<http://flask.pocoo.org/>）、結果として得られるREST APIにより、任意の場所の任意のユーザがシステムにアクセスできる。各ファイルのデータは、POSTコマンドによってユーザからシステムに渡される。最終結果はAPIを通じてJSON構造を介してユーザに返される。

20

#### 【0063】

基本および標準オラクルが各列についての所見を報告すると、列データタイプの最終分析中の時間計算量が最大となる。列の数が増えると、潜在的な列解釈の組み合わせの数が指数関数的に増加する。5つ以下のような少数の合計列については、レイアウトを識別する時間は2秒未満である。100～200列のファイルの場合、1～2分の間にレイアウトが決定される。

30

#### 【0064】

このインスタンスの1つの特定のバッチ実行は、区切られたレイアウトおよび固定幅のレイアウトならびにさまざまな数の列を含む63の異なるデータファイルで構成されている。合計1000列以上あった。データをサービスへダウンロードする時間を含む、これらのファイルを完全に処理するための合計時間は25分であった。

#### 【0065】

他に記載がない限り、本明細書で使用されるすべての技術用語および科学用語は、本発明が属する技術分野の当業者によって一般的に理解されるのと同じ意味を有する。本明細書に記載された方法および材料と類似するまたは等価な任意の方法および材料も、本発明の実施または試験において使用することができるが、本明細書には限定された数の例示的な方法および材料が記載される。本発明の概念から逸脱することなく、より多くの変更が可能であることは、当業者には明らかであろう。

40

#### 【0066】

本明細書で使用されるすべての用語は、コンテキストと一致する最も広い可能な方法で解釈されるべきである。ここでグループ化が使用される場合、そのグループのすべての個々のメンバーおよびそのグループの可能なすべての組み合わせおよび部分的な組み合わせ

50

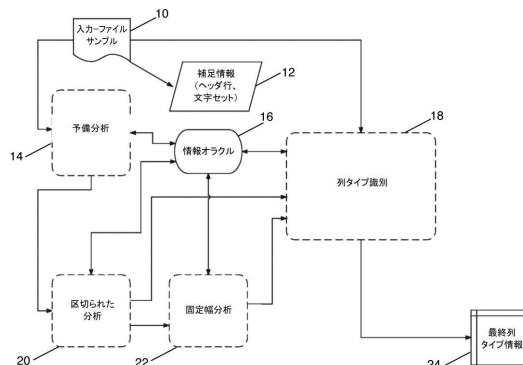
が個別に含まれることが意図される。本明細書に範囲が記載される場合、その範囲は、範囲内のすべての部分範囲および個々の点を含むことが意図される。本明細書に引用されるすべての参照は、本明細書の開示と矛盾しない範囲で、参照により本明細書に組み込まれる。

【0067】

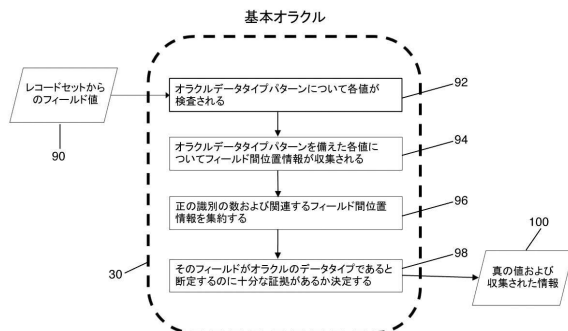
本発明は、特定の好ましいおよび代替の実施形態を参照して説明されており、添付の特許請求の範囲に記載されているように、単なる例示であって本発明の全範囲を限定するものではない。

【図 1】

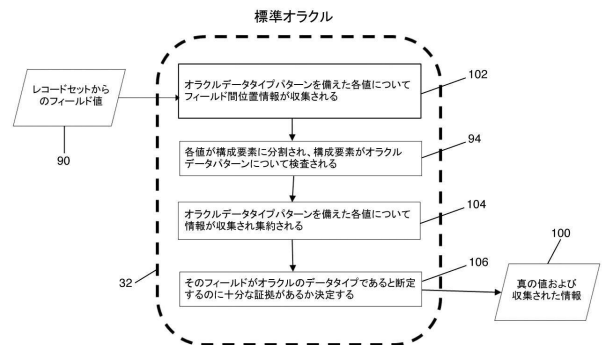
ファイルレイアウト推測エンジンフルシステム設計



【図 2 A】

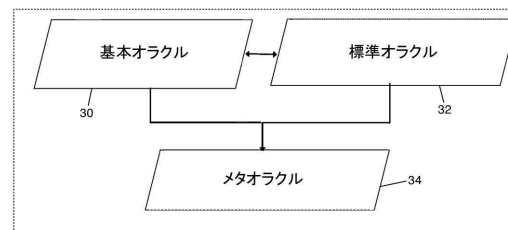


【図 2 B】



【図 2 C】

オラクルサブシステムのフロー



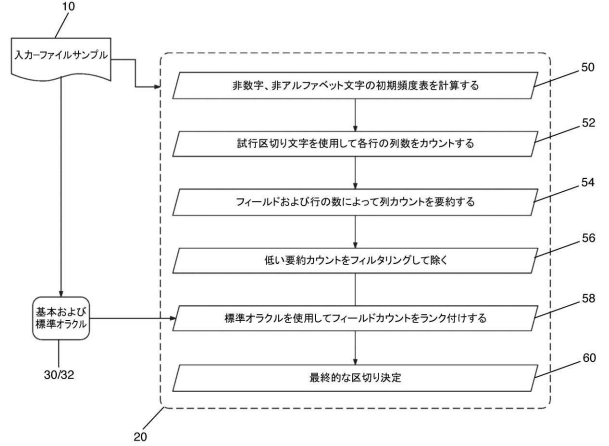
【図 2 D】

フィールドタイプオラクル

| 基本   | 標準          |           |           | メタ    |
|------|-------------|-----------|-----------|-------|
| アルファ | Axiom住所リンク  | IPv4アドレス  | SHA1      | 完全な住所 |
| 英数字  | Axiom消費者リンク | IPv6アドレス  | SHA256    | フルネーム |
| 空白   | Axiom文書ID   | ラストネーム    | SSN       |       |
| 数字   | 商号          | 緯度        | 州         |       |
| 数値   | 都市          | 経度        | 街路方向      |       |
|      | 国           | MD5       | 街路名       |       |
|      | 日付          | ミドルネーム    | 街路サフィックス  |       |
|      | ドメイン        | 名前プレフィックス | 単位指定子     |       |
|      | 電子メール       | 名前サフィックス  | 郵便番号      |       |
|      | ファーストネーム    | 電話番号      | 郵便番号の追加4桁 |       |
|      | 性別          | 私書箱       | 9桁の郵便番号   |       |

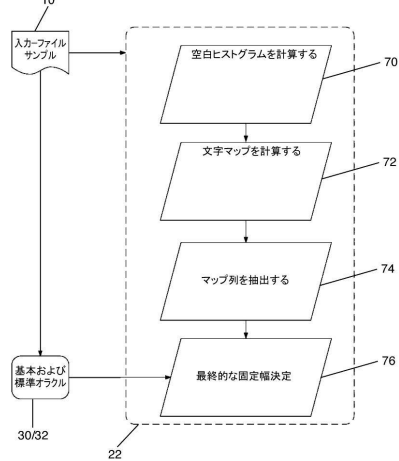
【図 3 A】

ファイルレイアウト推測エンジン-区切り列分析

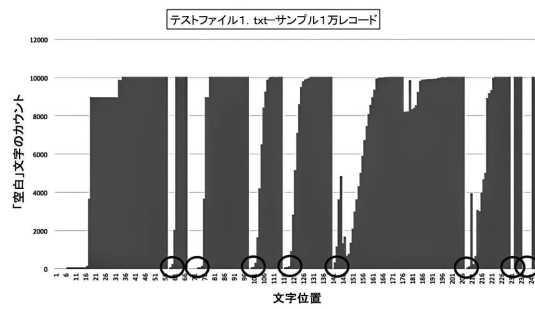


【図 3 B】

ファイルレイアウト推測エンジン-固定幅分析



【図 3 C】



【図 3 D】

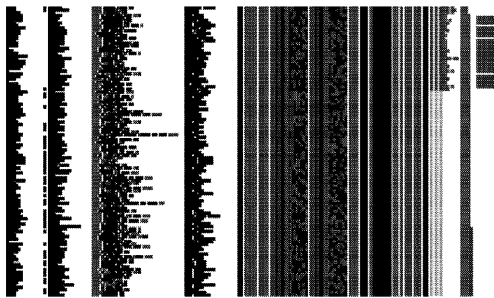
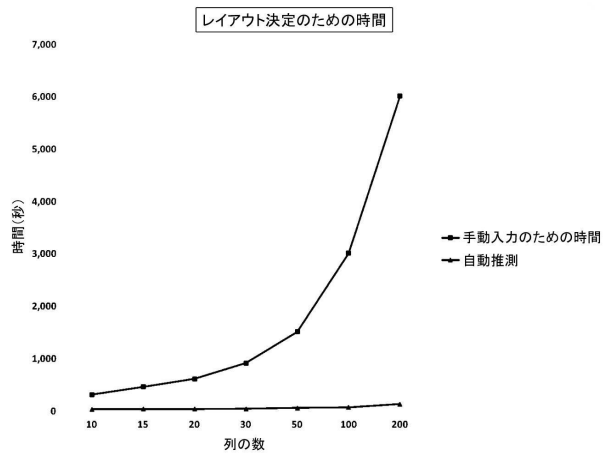


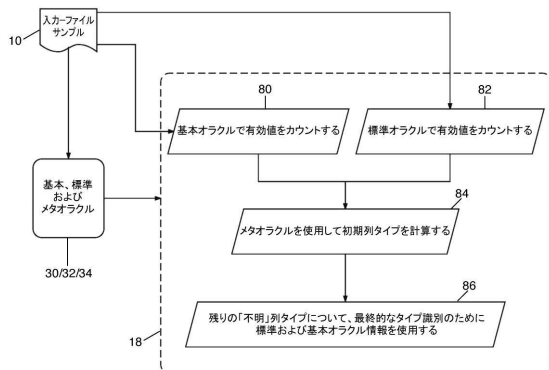
Fig. 3D

【図 5】



【図 4】

ファイルレイアウト推測エンジン-列タイプ識別



---

フロントページの続き

審査官 甲斐 哲雄

(56)参考文献 特表2007-506191 (JP, A)  
特表2013-511097 (JP, A)  
特開2013-191062 (JP, A)

(58)調査した分野(Int.Cl., DB名)  
G06F 16/00-16/958