

(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7505014号
(P7505014)

(45)発行日 令和6年6月24日(2024.6.24)

(24)登録日 令和6年6月14日(2024.6.14)

(51)国際特許分類

F I

G 0 6 F 8/34 (2018.01)

G 0 6 F 8/35 (2018.01)

G 0 6 F 16/28 (2019.01)

G 0 6 F 8/34

G 0 6 F 8/35

G 0 6 F 16/28

請求項の数 21 (全50頁)

(21)出願番号	特願2022-553152(P2022-553152)	(73)特許権者	509123208
(86)(22)出願日	令和3年3月4日(2021.3.4)		アビニシオ テクノロジー エルエルシー
(65)公表番号	特表2023-517310(P2023-517310 A)		アメリカ合衆国 0 2 4 2 1 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 2 0 1
(43)公表日	令和5年4月25日(2023.4.25)	(74)代理人	100108453
(86)国際出願番号	PCT/US2021/020871		弁理士 村山 靖彦
(87)国際公開番号	WO2021/178665	(74)代理人	100110364
(87)国際公開日	令和3年9月10日(2021.9.10)		弁理士 実広 信哉
審査請求日	令和6年3月4日(2024.3.4)	(74)代理人	100133400
(31)優先権主張番号	62/986,374		弁理士 阿部 達彦
(32)優先日	令和2年3月6日(2020.3.6)	(72)発明者	ヨナ・エゲノルフ
(33)優先権主張国・地域又は機関	米国(US)		アメリカ合衆国・マサチューセッツ・0 1 8 9 0・ウィンチェスター・アーバー・レーン・1 5
(31)優先権主張番号	17/025,751		
(32)優先日	令和2年9月18日(2020.9.18)		
	最終頁に続く		最終頁に続く

(54)【発明の名称】 スキーマからの最適化された論理の生成

(57)【特許請求の範囲】

【請求項1】

1つ又は複数のインターフェースを提供するためのデータ処理システムと、1つ又は複数の属性を有するデータセットを記憶する記憶装置とによって実装される方法であって、前記1つ又は複数のインターフェースが前記データセットの前記又は複数の属性へのアクセスを提供し、前記方法が、

スキーマにアクセスするステップであって、前記スキーマが、前記スキーマにおいて表されるデータセット間の関係、前記データセットのうちの1つ又は複数に対する1つ又は複数の計算、又は前記データセットのうちの1つ又は複数の1つ又は複数の変換を指定する、アクセスするステップと、

前記データセットの中からデータセットを選択することと、

前記選択されたデータセットに関連する1つ又は複数の他のデータセットを前記スキーマから識別することと

によって、記憶装置における複数の前記データセットを前記データセットの間から識別するステップと、

前記複数の前記データセットの属性を識別するステップと、

前記複数の前記データセットの識別された属性を表し、前記属性間の1つ又は複数の関係を更に表す論理データを生成するステップと、

1つ又は複数のインターフェースを通じて、前記複数の前記データセットの識別された前記属性を表す前記論理データのうちの1つ又は複数の部分へのアクセスを提供するステッ

プと、

前記1つ又は複数のインターフェースから、動作を実行するときに、識別された前記属性のうちの少なくとも1つを指定する仕様を受信するステップと、

前記仕様と、前記論理データによって表される識別された前記属性間の前記1つ又は複数の関係とに基づいて、複数からの少なくとも1つのデータセットに記憶装置からアクセスすることによって前記動作を実行するように構成されているコンピュータプログラムを生成するステップであって、アクセスされた前記少なくとも1つのデータセットが、前記仕様において指定された前記属性のうちの少なくとも1つを有する、生成するステップと、を含む、方法。

【請求項2】

前記1つ又は複数のインターフェースが、前記複数のデータセットに記憶装置からアクセスすることなく、前記論理データのうちの前記1つ又は複数の部分へのアクセスを提供する、請求項1に記載の方法。

【請求項3】

前記仕様において指定された前記属性のうちの前記少なくとも1つを含む前記複数のデータセットからのデータセットを識別するステップと、

識別された前記データセットに記憶装置からアクセスするステップと、を含む、請求項1に記載の方法。

【請求項4】

記憶装置からアクセスされた前記少なくとも1つのデータセットを使用して前記コンピュータプログラムを実行するステップを含む、請求項1に記載の方法。

【請求項5】

前記仕様において指定された前記属性のうちの前記少なくとも1つを有する前記複数のデータセット内のデータセットのみに記憶装置からアクセスすることによって前記動作を実行するように構成されている最適化されたコンピュータプログラムを作成するように、前記コンピュータプログラムを最適化するステップを含む、請求項1に記載の方法。

【請求項6】

前記1つ又は複数の属性が、前記複数の前記データセットのフィールド名を含む、請求項1に記載の方法。

【請求項7】

前記1つ又は複数の属性が、記憶装置内の前記複数の前記データセットにアクセスするための情報を含む、請求項1に記載の方法。

【請求項8】

前記スキーマから、選択された前記データセットと前記1つ又は複数の他のデータセットとを結合するための1つ又は複数のパラメータを識別するステップを含む、請求項1に記載の方法。

【請求項9】

前記1つ又は複数のパラメータが、選択された前記データセットと前記1つ又は複数の他のデータセットのうちの少なくとも1つとを結合するためのキーを含む、請求項8に記載の方法。

【請求項10】

クライアントデバイスから、選択された前記データセットを指定する選択データを受信するステップを含む、請求項1に記載の方法。

【請求項11】

選択された前記データセットが、前記論理データのルートノードを含み、前記1つ又は複数の他のデータセットのうちの少なくとも1つが、選択された前記データセットに結合される、請求項1に記載の方法。

【請求項12】

前記データセットのうちの1つ又は複数に対する前記1つ又は複数の計算、又は前記データセットのうちの1つ又は複数の1つ又は複数の変換が、前記複数の前記データセットのう

10

20

30

40

50

ちの少なくとも1つの仮想フィールドを定義する、請求項1に記載の方法。

【請求項13】

前記仕様と、前記論理データによって表される識別された前記属性間の前記1つ又は複数の関係とに基づいて前記動作を実行するように構成されている実行可能なデータフローグラフを生成するステップを含み、前記実行可能なデータフローグラフが、前記1つ又は複数の属性のうちの少なくとも1つを入力として含む、請求項1に記載の方法。

【請求項14】

前記仕様において指定された前記属性のうちの前記少なくとも1つを含まない前記複数の前記データセット内の少なくとも1つのデータセットに記憶装置からアクセスする動作を前記コンピュータプログラムから排除するステップを含む、請求項1に記載の方法。

【請求項15】

前記コンピュータプログラムが、選択ステートメントによる複数からの少なくともいくつかのデータに記憶装置からアクセスするように構成され、前記選択ステートメントが、前記仕様において指定された前記属性のうちの前記少なくとも1つのみを選択するように最小化される、請求項1に記載の方法。

【請求項16】

前記論理データはデータソースとして読み取られる、請求項1に記載の方法。

【請求項17】

前記複数の前記データセットの識別された前記属性を表す前記論理データの前記1つ又は複数の部分を表示するための、また、さらに、前記動作を実行する際に、識別された前記属性の少なくとも1つを指定する前記仕様を定義するための1つ又は複数のグラフィカルユーザインターフェースを含む開発環境を提供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18】

1つ又は複数のインターフェースと、1つ又は複数の属性を有するデータセットを記憶する記憶装置とを提供するためのシステムであって、前記1つ又は複数のインターフェースが前記データセットの前記1つ又は複数の属性へのアクセスを提供し、前記システムが、

1つ又は複数のプロセッサと、前記1つ又は複数のプロセッサによって実行されると、前記1つ又は複数のプロセッサに動作を実行させるように動作可能な命令を記憶する1つ又は複数の記憶デバイスであって、前記動作が、

スキーマにアクセスすることであって、前記スキーマが、前記スキーマにおいて表されるデータセット間の関係、前記データセットのうちの1つ又は複数に対する1つ又は複数の計算、又は前記データセットのうちの1つ又は複数の1つ又は複数の変換を指定する、アクセスすることと、

前記データセットの中からデータセットを選択することと、

前記選択されたデータセットに関連する1つ又は複数の他のデータセットを前記スキーマから識別することと

によって、記憶装置における複数の前記データセットを前記データセットの間から識別することと、

前記複数の前記データセットの属性を識別することと、

前記複数の前記データセットの識別された属性を表し、前記属性間の1つ又は複数の関係を更に表す論理データを生成することと、

前記1つ又は複数のインターフェースを通じて、前記複数の前記データセットの識別された前記属性を表す前記論理データのうちの1つ又は複数の部分へのアクセスを提供することと、

前記1つ又は複数のインターフェースから、動作を実行するときに、識別された前記属性のうちの少なくとも1つを指定する仕様を受信することと、

前記仕様と、前記論理データによって表される識別された前記属性間の前記1つ又は複数の関係とに基づいて、複数からの少なくとも1つのデータセットに記憶装置からアクセスすることによって前記動作を実行するように構成されているコンピュータプログラムを生

10

20

30

40

50

成することであって、アクセスされた前記少なくとも1つのデータセットが、前記仕様において指定された前記属性のうちの少なくとも1つを有する、生成することと、を含む、システム。

【請求項19】

前記コンピュータプログラムが、前記仕様において指定された前記属性のうちの前記少なくとも1つを有するデータセットのみに記憶装置からアクセスするように構成されている、請求項18に記載のシステム。

【請求項20】

命令を記憶する非一時的コンピュータ可読記憶媒体であって、前記命令が、コンピューティングシステムに、

スキーマにアクセスすることであって、前記スキーマが、前記スキーマにおいて表されるデータセット間の関係、前記データセットのうちの1つ又は複数に対する1つ又は複数の計算、又は前記データセットのうちの1つ又は複数の1つ又は複数の変換を指定する、アクセスすることと、

前記データセットの中からデータセットを選択することと、

前記選択されたデータセットに関連する1つ又は複数の他のデータセットを前記スキーマから識別することと

によって、記憶装置における複数の前記データセットを前記データセットの間から識別することと、

前記複数の前記データセットの属性を識別することと、

前記複数の前記データセットの識別された属性を表し、前記属性間の1つ又は複数の関係を更に表す論理データを生成することと、

前記1つ又は複数のインターフェースを通じて、前記複数の前記データセットの識別された前記属性を表す前記論理データのうちの1つ又は複数の部分へのアクセスを提供することと、

前記1つ又は複数のインターフェースから、動作を実行するときに、識別された前記属性のうちの少なくとも1つを指定する仕様を受信することと、

前記仕様と、前記論理データによって表される識別された前記属性間の前記1つ又は複数の関係とに基づいて、複数からの少なくとも1つのデータセットに記憶装置からアクセスすることによって前記動作を実行するように構成されているコンピュータプログラムを生成するステップであって、アクセスされた前記少なくとも1つのデータセットが、前記仕様において指定された前記属性のうちの少なくとも1つを有する、生成することと、を行わせる、非一時的コンピュータ可読記憶媒体。

【請求項21】

前記コンピュータプログラムが、前記仕様において指定された前記属性のうちの前記少なくとも1つを有するデータセットのみに記憶装置からアクセスするように構成されている、請求項20に記載の非一時的コンピュータ可読記憶媒体。

【発明の詳細な説明】

【技術分野】

【0001】

(優先権)

本出願は、2020年9月18日に出願された米国特許出願第17/025,751号及び2020年3月6日に出願された米国特許仮出願第62/986,374号の優先権及び利益を主張するものであり、それらの各々の全容は参照により本明細書に組み込まれる。

【0002】

(発明の分野)

本開示は、データベーススキーマなどのスキーマから論理を生成することに関する。

【背景技術】

【0003】

複雑な計算は、多くの場合、有向グラフを通るデータフローとして表すことができ、計算のコンポーネントは、グラフの頂点と関連付けられ、コンポーネント間のデータフローはグラフのリンク（弧、辺）に対応する。そのようなグラフベースの計算を実施するシステムは、「Executing Computations Expressed as Graphs」と題された米国特許第5,966,072号に記載されており、その全容は参照により本明細書に組み込まれる。場合によっては、頂点と関連付けられた計算は、「ビジネスルール」と称される人間が読み取れる形式で記述される。

【0004】

データフローグラフを生成するための1つの技術は、ビジネスルールエディタを使用する。ビジネスルールエディタの例は、「Editing and Compiling Business Rules」と題された米国特許第8,069,129号に開示されており、その全容は参照により本明細書に組み込まれる。

【発明の概要】

【0005】

概して、第1の態様では、開発環境を提供するためのデータ処理システムと、1つ以上の属性を有するデータセットを記憶する記憶装置とによって実装される方法であって、開発環境は、データセットの1つ以上の属性へのアクセスを提供し、方法は、スキーマにアクセスすることであって、スキーマは、スキーマにおいて表されるデータセット間の関係、データセットのうちの1つ以上に対する1つ以上の計算、又はデータセットのうちの1つ以上の1つ以上の変換を指定する、アクセスすることと、データセットの中から、記憶装置の中の複数のデータセットを識別することであって、識別することが、データセットの中からデータセットを選択することと、スキーマから、選択されたデータセットに関連する1つ以上の他のデータセットを識別することと、によって行われる、識別することと、複数のデータセットの属性を識別することと、複数のデータセットの識別された属性を表し、属性間の1つ以上の関係を更に表す論理データを生成することと、開発環境に、論理データを提供することと、開発環境によって、複数のデータセットの識別された属性を表す論理データのうちの1つ以上の部分へのアクセスを提供することと、開発環境から、動作を実行するときに、識別された属性のうちの少なくとも1つを指定する仕様を受信することと、指定と、論理データによって表される識別された属性間の1つ以上の関係とに基づいて、複数からの少なくとも1つのデータセットに記憶装置からアクセスすることによって動作を実行するように構成されているコンピュータプログラムを生成することとであって、アクセスされた少なくとも1つのデータセットが、仕様において指定された属性のうちの少なくとも1つを有する、生成することと、を含む。

【0006】

概して、第2の態様では、第1の態様と組み合わせ可能であり、開発環境は、複数のデータセットに記憶装置からアクセスすることなく、論理データのうちの1つ以上の部分へのアクセスを提供する。

【0007】

概して、第3の態様では、第1の態様又は第2の態様と組み合わせ可能であり、本方法は、仕様において指定された属性のうちの少なくとも1つを含む複数のデータセットから1つのデータセットを識別することと、識別されたデータセットに記憶装置からアクセスすることと、を含む。

【0008】

概して、第4の態様では、第1～第3の態様のうちのいずれかと組み合わせ可能であり、本方法は、記憶装置からアクセスされた少なくとも1つのデータセットを使用してコンピュータプログラムを実行することを含む。

【0009】

概して、第5の態様では、第1～第4の態様のうちのいずれかと組み合わせ可能であり、本方法は、仕様において指定された属性のうちの少なくとも1つを有する複数のデータセット内のデータセットのみに記憶装置からアクセスすることによって動作を実行するよ

10

20

30

40

50

うに構成されている最適化されたコンピュータプログラムを作り出すように、コンピュータプログラムを最適化することを含む。

【0010】

概して、第6の態様では、第1～第5の態様のうちのいずれかと組み合わせ可能であり、1つ以上の属性は、複数のデータセットのフィールド名を含む。

【0011】

概して、第7の態様では、第1～第6の態様のうちのいずれかと組み合わせ可能であり、1つ以上の属性は、記憶装置内の複数のデータセットにアクセスするための情報を含む。

【0012】

概して、第8の態様では、第1～第7の態様のうちのいずれかと組み合わせ可能であり、本方法は、スキーマから、選択されたデータセットと1つ以上の他のデータセットとを結合するための1つ以上のパラメータを識別することを含む。

10

【0013】

概して、第9の態様では、第1～8つの態様のうちのいずれかと組み合わせ可能であり、1つ以上のパラメータは、選択されたデータセットと1つ以上の他のデータセットのうちの少なくとも1つとを結合するためのキーを含む。

【0014】

概して、第10の態様では、第1～第9の態様のうちのいずれかと組み合わせ可能であり、本方法は、クライアントデバイスから、選択されたデータセットを指定する選択データを受信することを含む。

20

【0015】

概して、第11の態様では、第1～第10の態様のうちのいずれかと組み合わせ可能であり、選択されたデータセットは、論理データのルートノードを含み、1つ以上の他のデータセットのうちの少なくとも1つは、選択されたデータセットに結合される。

【0016】

概して、第12の態様では、第1～第11の態様のうちのいずれかと組み合わせ可能であり、データセットのうちの1つ以上に対する1つ以上の計算、又はデータセットのうちの1つ以上の1つ以上の変換は、複数のデータセットのうちの少なくとも1つの仮想フィールドを定義する。

【0017】

概して、第13の態様では、第1～第12の態様のうちのいずれかと組み合わせ可能であり、本方法は、仕様と、論理データによって表される識別された属性間の1つ以上の関係とに基づいて動作を実行するように構成されている実行可能なデータフローグラフを生成することを含み、実行可能なデータフローグラフは、1つ以上の属性のうちの少なくとも1つを入力として含む。

30

【0018】

概して、第14の態様では、第1～第13の態様のうちのいずれかと組み合わせ可能であり、本方法は、仕様において指定された属性のうちの少なくとも1つを含まない複数のデータセット内の少なくとも1つのデータセットに記憶装置からアクセスする動作をコンピュータプログラムから排除することを含む。

40

【0019】

概して、第15の態様では、第1～第14の態様のうちのいずれかと組み合わせ可能であり、コンピュータプログラムは、複数の選択ステートメントによって少なくともいくつかのデータに記憶装置からアクセスするように構成され、選択ステートメントは、仕様において指定された属性のうちの少なくとも1つのみを選択するように最小化される。

【0020】

概して、第16の態様では、第1～第15の態様のうちのいずれかと組み合わせ可能であり、開発環境は、論理データをデータソースとして読み取る。

【0021】

概して、第17の態様では、第1～第16の態様のうちのいずれかと組み合わせ可能で

50

あり、コンピュータプログラムは、仕様において指定された属性のうちの少なくとも1つを有するデータセットのみに記憶装置からアクセスするように構成される。

【0022】

概して、第18の態様では、第1～第17の態様のうちのいずれかと組み合わせ可能であり、開発環境と、1つ以上の属性を有するデータセットを記憶する記憶装置とを提供するためのシステムであって、開発環境は、データセットの1つ以上の属性へのアクセスを提供し、システムは、1つ以上のプロセッサと、1つ以上のプロセッサによって実行されると、1つ以上のプロセッサに動作を実行させるように動作可能な命令を記憶する1つ以上の記憶デバイスであって、動作は、スキーマにアクセスすることであって、スキーマは、スキーマにおいて表されるデータセット間の関係、データセットのうちの1つ以上に対する1つ以上の計算、又はデータセットのうちの1つ以上の1つ以上の変換を指定する、アクセスすることと、データセットの中から、複数のデータセットを識別することであって、識別することが、データセットの中からデータセットを選択することと、スキーマから、選択されたデータセットに関連する1つ以上の他のデータセットを識別することと、によって行われる、識別することと、複数のデータセットの属性を識別することと、複数のデータセットの識別された属性を表し、属性間の1つ以上の関係を更に表す論理データを生成することと、開発環境に、論理データを提供することと、開発環境によって、複数のデータセットの識別された属性を表す論理データのうちの1つ以上の部分へのアクセスを提供することと、開発環境から、動作を実行するときに、識別された属性のうちの少なくとも1つを指定する仕様を受信することと、仕様と、論理データによって表される識別された属性間の1つ以上の関係とに基づいて、複数からの少なくとも1つのデータセットに記憶装置からアクセスすることによって動作を実行するように構成されているコンピュータプログラムを生成することであって、アクセスされた少なくとも1つのデータセットが、仕様において指定された属性のうちの少なくとも1つを有する、生成することと、を含む1つ以上の記憶デバイスと、を備える。

【0023】

概して、第19の態様では、第1～第18の態様のうちのいずれかと組み合わせ可能であり、非一時的コンピュータ可読記憶媒体は、命令を記憶しており、命令は、コンピューティングシステムに、スキーマにアクセスすることであって、スキーマは、スキーマにおいて表されるデータセット間の関係、データセットのうちの1つ以上に対する1つ以上の計算、又はデータセットのうちの1つ以上の1つ以上の変換を指定する、アクセスすることと、データセットの中から、記憶装置内の複数のデータセットを識別することであって、識別することが、データセットの中からデータセットを選択することと、スキーマから、選択されたデータセットに関連する1つ以上の他のデータセットを識別することと、によって行われる、識別することと、複数のデータセットの属性を識別することと、複数のデータセットの識別された属性を表し、属性間の1つ以上の関係を更に表す論理データを生成することと、開発環境に、論理データを提供することと、開発環境によって、複数のデータセットの識別された属性を表す論理データのうちの1つ以上の部分へのアクセスを提供することと、開発環境から、動作を実行するときに、識別された属性のうちの少なくとも1つを指定する仕様を受信することと、仕様と、論理データによって表される識別された属性間の1つ以上の関係とに基づいて、複数からの少なくとも1つのデータセットに記憶装置からアクセスすることによって動作を実行するように構成されているコンピュータプログラムを生成することであって、アクセスされた少なくとも1つのデータセットが、仕様において指定された属性のうちの少なくとも1つを有する、生成することと、を行わせる。

【0024】

上記の実装形態のうちの1つ以上は、以下の利点のうちの1つ以上を提供し得る。本明細書に記載の技術は、データセットの属性に関する情報を含む論理データを生成するために、データセット及びデータセット間の関係に関する情報を使用する。開発環境において論理データをデータソースとして提供することによって、論理データは、物理データセッ

10

20

30

40

50

ト自体にアクセスする負担を伴わずに、データセットへの論理アクセスを提供することができる。このようにして、物理データセットにデータベース記憶装置からアクセスすることと関連する計算リソースの消費を低減することができる。それとは別に、計算論理は、実際のデータセットにアクセスする必要なくデータセットの属性を使用して開発環境を通して指定することができ、これにより、データセットを不要なアクセスから保護することが可能になる。すなわち、データセットは、計算論理を指定し、それからアプリケーションをコンパイルすることを依然として可能にしながら、データセットを含め、セキュアに保たれる。加えて、本明細書に記載の技術は、ユーザが、物理データセットにアクセスし処理するときに必要なオーバーヘッド及び処理時間を伴わず、論理データ内の属性を使用して計算論理を柔軟に開発する権限を与えられるため、開発環境のユーザの生産性を改善することができる。

10

【0025】

計算論理が開発されると、本明細書に記載の技術により、論理を高度に最適化された様式で処理することが可能になる。例えば、ユーザは、計算論理を開発するときに、原則として、不要であることが判明した多数のデータセットを考慮するか又はアクセスし得る。論理データによって提供される情報を使用して、データフローグラフなどのアプリケーションを生成することができ、アプリケーションは、所望の出力を作成するために処理において必要とされるデータのサブセットのみを最小限にロードして結合することができる。そうすることで、本明細書に記載の技術は、論理を処理するために必要な計算リソースを低減しながら、計算論理の生成及び実行の速度を増加させる。

20

【0026】

1つ以上の実装形態の詳細が、添付の図面及び以下の記載において明らかにされる。本明細書に記載のその他の特徴、目的、及び利点は、記載及び図面から、並びに特許請求の範囲から明らかになるであろう。

【図面の簡単な説明】

【0027】

【図1】スキーマのブロック図である。

【図2A】論理データを作り出し、論理データを使用してコンピュータプログラムを生成するためのシステムのブロック図である。

【図2B】論理データを作り出すためのシステムのブロック図である。

30

【図2C】開発環境のブロック図である。

【図2D】開発環境のブロック図である。

【図2E】コンピュータプログラムを実行するためのシステムのブロック図である。

【図3】論理データを作り出し、論理データを使用してコンピュータプログラムを生成するためのプロセスの図である。

【図4A】論理データを作り出すためのシステムのブロック図である。

【図4B】論理データを作り出すためのシステムのブロック図である。

【図4C】論理データを作り出すためのシステムのブロック図である。

【図5A-1】論理データを作り出し、論理データを使用してコンピュータプログラムを生成するためのシステムのブロック図である。

40

【図5A-2】論理データを作り出し、論理データを使用してコンピュータプログラムを生成するためのシステムのブロック図である。

【図5A-3】論理データを作り出し、論理データを使用してコンピュータプログラムを生成するためのシステムのブロック図である。

【図5B】論理データを作り出すためのシステムのブロック図である。

【図5C】論理データを作り出すためのシステムのブロック図である。

【図5D】コンピュータプログラムを生成するためのシステムのブロック図である。

【図5E】コンピュータプログラムを生成するためのシステムのブロック図である。

【図5F】コンピュータプログラムを生成するためのシステムのブロック図である。

【図5G】コンピュータプログラムを生成するためのシステムのブロック図である。

50

【図 5 H】 コンピュータプログラムを生成するためのシステムのブロック図である。
【図 5 I】 コンピュータプログラムを生成するためのシステムのブロック図である。
【図 5 J】 コンピュータプログラムをテストするためのシステムのブロック図である。
【図 5 K】 コンピュータプログラムをテストするためのシステムのブロック図である。
【図 5 L】 コンピュータプログラムをテストするためのシステムのブロック図である。
【図 5 M】 コンピュータプログラムをテストするためのシステムのブロック図である。
【図 5 N】 コンピュータプログラムをテストするためのシステムのブロック図である。
【図 5 O】 コンピュータプログラムをテストするためのシステムのブロック図である。
【図 5 P】 コンピュータプログラムをテストするためのシステムのブロック図である。
【図 5 Q】 コンピュータプログラムをテストするためのシステムのブロック図である。
【図 6 A】 開発環境のブロック図である。
【図 6 B】 コンピュータプログラムを生成するためのシステムのブロック図である。
【図 6 C】 コンピュータプログラムを実行するためのシステムのブロック図である。
【図 7 A】 開発環境のブロック図である。
【図 7 B】 コンピュータプログラムを生成するためのシステムのブロック図である。
【図 7 C】 コンピュータプログラムを生成するためのシステムのブロック図である。
【図 7 D】 コンピュータプログラムを生成するためのシステムのブロック図である。
【図 7 E】 コンピュータプログラムを生成するためのシステムのブロック図である。
【図 8】 論理データを作り出し、論理データを使用してコンピュータプログラムを生成するためのプロセスのフローチャートである。
【発明を実施するための形態】

【 0 0 2 8 】

本明細書には、記憶システム又はメモリに記憶された物理データセットを表す論理データを生成するためのシステムが記載される。論理データは、これらの物理データセットを、とりわけ、物理データセットの属性を含めることによって、これらの物理データセットの記憶場所のアドレスを指定するポインタを含めることによって、又は物理データセットにどのようにアクセスするかを表す他の情報を指定することによって、又はそれらの組み合わせによって表す。この例では、論理データ又は論理データの一部は、どのデータセット（又はデータセットの属性）が使用され、アクセスされるかを指定する仕様の開発を可能にするために、開発環境においてアクセス可能である。概して、仕様は、データセット又はデータセットの属性に対して実行される動作（例えば、計算論理）を指定する。仕様は、コンピューティングシステム上で実行することができるコンピュータプログラム（例えば、実行可能なデータフローグラフ）にコンパイルされるか、さもなければそれを作成するために使用される。いくつかの例では、コンピュータプログラムは、実行可能なマシンコードを含む。論理データが、データセット又はそれらの属性に物理的にアクセスする必要なく開発環境においてアクセス可能であるため、論理データは、物理的負担を伴わずに論理アクセスを提供する。

【 0 0 2 9 】

便宜上、限定するものではないが、本明細書に記載の特徴のうちのいくつかの視覚表現は、特徴自体と称され得る。例えば、データフローグラフの視覚表現は、データフローグラフと称され得る。論理データの視覚表現は、論理データと称され得る。データベーススキーマの視覚表現は、データベーススキーマと称され得る。コンポーネントの視覚表現は、コンポーネントと称され得、以下同様である。

【 0 0 3 0 】

図 1 を参照すると、スキーマ 2 が示されており、スキーマ 2 は、記憶システムに記憶されたデータセット 6 a、6 b、6 c、6 d の間の階層的関係などの関係 4 a、4 b、4 c を指定する。いくつかの例では、スキーマ 2 は、データベース管理システム（Database Management System、DBMS）によってサポートされる形式言語でデータベースの構造を記述するデータベーススキーマである。スキーマ 2 は、記憶システムに記憶されたデータセット 6 a、6 b、6 c、6 d、及びそれらのデータセットの関係 4 a、4 b、4 c

に関する情報に基づいて生成することができる。いくつかの例では、記憶されたデータセットの各々に関する情報は、他の情報の中でもとりわけ、データセットの名前、データセットのアクセスパラメータ（例えば、ファイル名、場所）、データセットのレコードフォーマット、データセットに含まれるデータタイプ、又はそれらの組み合わせを含む。いくつかの例では、データセット間の関係に関する情報は、データセット間の関係のタイプ（例えば、関係 4 b 及び 4 c などの 1 対 1、1 対多、関係 4 a などの多対多）若しくはデータセット内のデータを結合するためのキー（例えば、主キー、外部キー）、又はその両方についての情報など、データセットをどのように結合することができるかに関する情報を含む。

【0031】

スキーマ 2 を生成するために使用される情報は、ユーザ（例えば、技術ユーザ）によって指定されるか、（例えば、記憶システムに結合された 1 つ以上のコンピューティングシステムによって）記憶システムから自動的に取得されるか、又はその両方が可能である。例えば、いくつかの例では、記憶システムに通信可能に結合された 1 つ以上のコンピューティングシステムは、データ検出技術、セマンティック検出技術、又は他の機械学習技術を使用してスキーマ 2 を生成するために、データセット 6 a、6 b、6 c、6 d に関するメタデータ又は他の情報をインポートすることができる。いくつかの例では、情報を処理すること、データセット 6 a、6 b、6 c、6 d のうちの 1 つ以上に対する計算又はデータセット 6 a、6 b、6 c、6 d のうちの 1 つ以上の変換などが、（例えば、技術ユーザによって）指定され、スキーマ 2 に含まれる。例えば、スキーマ 2 は、計算又は動作を実行するための命令（又は動作を実行するために、実行可能なデータフローグラフなどのコンピュータプログラムを呼び出すための命令）を含む。これらの計算又は変換は、データセット 6 a、6 b、6 c、6 d 内の既存のフィールドを修正し、データセット内に新しいフィールド（仮想フィールド又は計算フィールドと称されることもある）を作成し、又は完全に新しいデータセットを作成することができる。いくつかの例では、修正された又は新たに作成されたフィールド又はデータセットの値は、以下に記載されるように、ランタイム（例えば、フィールド又はデータセットを使用するコンピュータプログラムによって実行されるとき）までデータ入力されない。

【0032】

図 2 A は、記憶システム 1 2 及びクライアントデバイス 1 4 を有する環境 1 0 を示している。この例では、環境 1 0 はまた、論理データジェネレータ 1 6 を含んでいる。論理データジェネレータ 1 6 は、例えば、実際の物理データセット（又は物理データセットに基づく論理データセット）の属性に関する情報を含む論理データを生成するように構成されている。この例では、論理データは、記憶システム 1 2 からのデータセットへの物理アクセスを必要とせずに、例えば、記憶システム 1 2 に記憶され得る物理データセットへの論理アクセスを提供する。この例では、論理アクセスとは、それ自体が記憶システム 1 2 に記憶される物理データセットの属性のリスト又は他の仕様を指す。別の例では、論理データは、論理データで表される物理データセットが、記憶システム 1 2 からアクセスされ得るアドレス又は場所を識別するポインタ若しくは他の情報、又は物理データセットにアクセスするための命令若しくはパラメータ、又はその両方を含み得る。

【0033】

この例では、記憶システム 1 2 は、論理データジェネレータ 1 6 と通信するように構成され、論理データジェネレータに、他の情報の中でもとりわけ、物理データセットの位置を指定する情報、物理データセットの属性を指定する情報、物理データセットの間の関係を指定する情報、又は物理データセット自体、又はそれらの組み合わせなどの、論理データの生成に使用する情報を提供する。クライアントデバイス 1 4 はまた、論理データジェネレータ 1 6 と通信するように構成され、それにより、クライアントデバイス 1 4 は、論理データジェネレータ 1 6 に、他の情報の中でもとりわけ、論理データからどの物理的データセット又は物理データセットのどの属性を含む（又は省略する）のかを指定する情報、論理データのルートノードを指定する情報、又はそれらの組み合わせなど、論理データ

10

20

30

40

50

を生成するための情報を送信し得る。

【0034】

環境10はまた、開発環境18を含み、開発環境18は、ユーザ（例えば、開発環境18に通信可能に結合され得るクライアントデバイス14のユーザ）が、データフローグラフなどのコンピュータプログラムを生成する際に、論理データにおいて表されるどのデータセット（又はデータセットの属性）にユーザがアクセス又は使用したいかを指定するためのグラフィカルユーザインターフェース又は他のユーザインターフェースをユーザに提供する。開発環境18は、グラフジェネレータ22と結合され、グラフジェネレータ22は、開発環境18から受信した情報からデータフローグラフを生成するように構成されている。開発環境18から受信した情報は、この情報がコンピュータプログラム（例えば、実行可能なデータフローグラフ）の機能を指定し、実行又はアプリケーション自体への仕様のコンパイル中にどのデータセット（又は属性）がアクセスされるべきかを指定するため、仕様と称されることが多い。

10

【0035】

環境10はまた、コンパイラ24を含み、コンパイラ24は、仕様及び／又はデータフローグラフを、データ処理システム26によって（例えば、マシンコードで）実行可能なコンピュータプログラムにコンパイルするように構成されている。この例では、開発環境18は、仕様をデータフローグラフを生成するグラフジェネレータ22に送信し、グラフジェネレータ22がデータフローグラフを生成する。次に、グラフジェネレータ22は、データフローグラフをコンパイラ24に送信し、コンパイラ24は、データフローグラフをコンピュータプログラム（例えば、実行可能なデータフローグラフ）にコンパイルする。コンパイラ24は、コンピュータプログラムを実行及び／又は記憶するために、コンピュータプログラムをデータ処理システム26に送信する。この例では、コンピュータプログラムは、記憶システム12から、属性が論理データに含まれた、又は仕様において指定された、又はその両方であった複数のデータセットのうちの少なくとも1つにアクセスするように構成されている。

20

【0036】

図2Bを参照すると、環境20は、環境10の更なる詳細を示している。この例では、記憶システム12は、スキーマ21を論理データジェネレータ16に送信する。データベーススキーマ21は、階層関係など、記憶システム12に記憶されたデータセット21a、21b、21c、21dの間の関係を指定する。例では、スキーマ21は、データベーススキーマである。スキーマ21は、図1を参照して記載されたスキーマ2と同じ又は類似であってもよい。

30

【0037】

クライアントデバイス14はまた、選択データ23を論理データジェネレータ16に送信する。選択データ23は、ルートノード、例えば、論理データを定義する際に親ノードとなるデータセットを指定する。この例では、ルートノードは、論理データ内のルートノードである初期データセットを定義するパースペクティブである。この例では、パースペクティブは情報の抽出であり、スキーマ内に選択された開始点を指定する。パースペクティブは、スキーマ内に選択された開始点を含み、対象のルート論理エンティティを表す。選択データ23を生成するために、クライアントデバイス14は、グラフィカルユーザインターフェース27を表示する。グラフィカルユーザインターフェース27は、データセット部分28と、データセット21dが論理データのルートノードとして選択されることを表すアイコン29aで更新される選択されたパースペクティブ部分29とを含む。データセット部分28は、それぞれ、データセット21a、21b、21c、21dの視覚表現28a、28b、28c、28dを表示する。選択されたパースペクティブ部分29は、ボタンであり得る選択可能な部分29bを含む。選択可能な部分29bが選択されると、ユーザは、視覚表現28a、28b、28c、28dのうちの1つを論理データのルートノードとして選択することができる。この例では、ユーザは視覚表現28dを選択して、データセット21dが論理データのルートノードであることを指定する。視覚表現28

40

50

dが選択され、選択可能な部分292bと対話されると、選択されたパースペクティブ部分29は、データセット21dが論理データのルートノードであることを指定するアイコン28aを表示するように更新される。選択データ23は、ルートノードがデータセット21dであることを指定する。

【0038】

対象のルート論理エンティティが論理データに対して指定されると、論理データは、対象のそのルート論理エンティティに関連する他のデータセットの情報を含むように拡張される。本明細書に記載されるように、その、他の情報は、属性、フィールド、ソース、命令、パラメータ又は対象のルート論理エンティティ及び関連データセットへのポイントなどを含み得る。この例では、論理データは、対象のルート論理エンティティのフィールド及び対象のルート論理エンティティに関連する他のデータセットのフィールドのエントリを有するワイドレコードに具体化することができる。概して、ワイドレコードは、同じ構造内に保持される関連データのグループを含む。論理データはまた、対象の論理エンティティ及び他の関連データセットのメモリ内の物理的場所へのポイントなど、他の属性のワイドレコードに具体化することができる。

【0039】

論理データジェネレータ16は、スキーマ21及び選択データ23を使用して、論理データ25を生成する。例えば、論理データジェネレータ16は、データセット21dがルートノードであることを特定する選択データ23を受信し、データセット21dの属性又は利用可能なフィールドに関する情報を論理データ25に含める。いくつかの例では、情報は、データセット21dの利用可能な属性又はフィールドのベクトルを含む。論理データジェネレータ16は、スキーマ21を使用して、データセット21dに関連する他のデータセットを識別する。例えば、この例では、論理データジェネレータ16は、データセット21a、21b、21cがデータセット21dに関連していると判定し、したがって、データセット21a、21b、及び21cの属性又は利用可能なフィールドに関する情報を論理データ25に含める。この例では、論理データ25は、データセット21a、21b、及び21cの各々の属性又は利用可能なフィールドを指定するベクトルを含む。属性又は利用可能なフィールドのこれらのベクトルは、フィールド自体内のフィールド又はデータに実際にアクセスすることなく、属性又はフィールド名を指定し、属性又はフィールド名にどのようにアクセスするかを命令し、さもなければ属性又はフィールド名を表す。このため、論理データ25は、記憶システム12からこれらのデータセットに実際にアクセスする物理的負担を伴わずに、データセット21a、21b、21c、及び21dへの論理アクセスを提供する。

【0040】

図2Cを参照すると、環境30は、論理データ25を受信する開発環境18を示している。例えば、開発環境18は、論理データジェネレータ16から、又は記憶装置（例えば、記憶システム12）から論理データ25を受信する。この例では、ビジネスルールエディタインターフェース32が、ビジネスルール及び他の論理ルールを定義するために開発環境18によって生成される。概して、エディタインターフェース32は、式を含むセルをグラフィカルに識別し得る。これは、ユーザが、それ自体で真又は偽に評価されることになる式と、列変数に対して比較される値を返す式との間の差を理解するのに役立つことになる。ユーザがタイプ入力しているとき、ユーザは、例えば、式の始めにアスタリスクをタイプ入力することによって、特定のセルが式セルであることを示すことができる。この例では、エディタインターフェース32は、入力部分33及びルール定義部分34を含む。入力部分33は、それらの属性（例えば、フィールド）及び論理データ25において表されるデータセット、並びに他のデータソース（論理データに対応してもしなくてもよい）の視覚表現を提供する。例えば、入力部分33は、データセット21a（図2Bに示される）を表す視覚表現35を含む。入力部分33はまた、データセット21aの「フィールドA」を表す視覚表現35aを含む。この例では、視覚表現35aは、視覚表現35からインデントされることによって、データセット21a内のフィールドであるよう

10

20

30

40

50

に視覚的に描かれている。入力部分 3 3 はまた、それぞれ、データセット 2 1 b 及びデータセット 2 1 b 内の「フィールド B」を表す視覚表現 3 6 及び 3 6 a を含む。入力部分 3 3 はまた、それぞれ、データセット 2 1 c 及びデータセット 2 1 c 内の「フィールド C」を表す視覚表現 3 7 及び 3 7 a を含む。入力部分 3 3 はまた、それぞれ、データセット 2 1 d 及びデータセット 2 1 d 内の「フィールド D」を表す視覚表現 3 8 及び 3 8 a を含む。この例では、入力部分 3 3 内の視覚表現は、ビジネスルールを定義する際にユーザに利用可能なデータセット及びフィールドを表す。入力部分 3 3 に表される利用可能なデータセット及びフィールドは、論理データ 2 5 から識別され、それによって、物理的メモリからデータセット（又はフィールド）にアクセスする必要なくデータセット及びフィールドへのアクセスをユーザに提供する。

【0041】

ルール定義部分 3 4 は、一連のルールケースを含む。この例では、ルール定義部分 3 4 は、スプレッドシートフォーマットを含む。スプレッドシート内のトリガ列は、利用可能なデータ値に対応し、行は、ルールケース、例えば、利用可能なデータ値を関連付ける基準のセットに対応する。ルールケースは、所与のレコードのデータ値が、ルールケースが基準を有する各トリガ列のトリガ基準を満たす場合、そのレコードに適用される。ルールケースが適用される場合、出力列に基づいて出力が生成される。その入力関係の全てが満たされているルールケースは、「トリガされた」と称され得る。出力列は、潜在的な出力変数に対応し、該当する行の対応するセル内の値は、もしあれば、その変数の出力を決定する。セルは、変数に割り当てられた値を含むことができ、又は以下で考察されるように、出力値を生成するために評価されなければならない式を含むことができる。図 2 C には 1 つのみが示されているが、2 つ以上の出力列があってもよい。

【0042】

ルール定義部分 3 4 内のセルの入力を指定することによってルールの定義が完了すると、開発環境 1 8 は、ルールケースを指定し、ルールを実装するためにどのフィールドがアクセスする必要があるかを指定するルール仕様 3 9 a を生成する。しかしながら、ルールを定義するこの段階では、論理データ 2 5 は、ユーザに、物理アクセスなしにそれらのフィールドへの論理アクセスを提供する。例えば、ユーザは、入力部分 3 3 内の記憶システム 1 2 に記憶された様々なデータセットから利用可能なフィールドを閲覧することができることによって、論理アクセスが提供された。開発環境 1 8 は、ルール仕様 3 9 a をグラフィジェネレータ 2 2 に送信する。開発環境 1 8 はまた、論理データ 2 5 をグラフィジェネレータ 2 2 に送信する。

【0043】

図 2 D を参照すると、環境 4 0 は、開発環境 1 8 の別の例を示している。この例では、開発環境 1 8 は、コンポーネント部分 4 2、入力部分 4 3、及びキャンバス区画 4 4 を有するグラフィカルユーザインターフェース 4 1 を描画している。コンポーネント部分 4 2 は、計算論理を定義するために利用可能な様々な動作を表す視覚表現 4 2 a ~ 4 2 f を含む。入力部分 4 3 は、論理データ 2 5 において表されるデータセット及び属性（例えば、フィールド）の視覚表現 4 5、4 5 a、4 6、4 6 a、4 7、4 7 a、4 8 b、4 8 a を表示している。入力部分 4 3 はまた、他のデータソース（例えば、論理データ 2 5 以外のデータソース）において表されるデータセット及びフィールドの視覚表現 4 9 及び 4 9 a を表示している。すなわち、入力部分 4 3 内の視覚表現は、計算論理を定義するために利用可能なデータセット及びフィールドを表す。

【0044】

キャンバス部分 4 4 は、データフローグラフの形態で計算論理を定義するために使用され、視覚化 4 4 a として視覚的に描かれる（以下、便宜上、限定することなく、「データフローグラフ 4 4 a」と称される）。視覚化 4 4 a によって表されるデータフローグラフは、ノードを有するデータ構造を含む。ノードの各々は、少なくとも 1 つの動作プレースホルダフィールド及び少なくとも 1 つのデータプレースホルダフィールドを含み、これらは、キャンバス部分 4 4 においてユーザによって指定された動作及びデータ（例えば、論

理データ、「データセットV」などの他のデータソース）が入力される。この例では、データフローグラフ44aは、視覚表現42a～42fのうちの1つ以上を、コンポーネント部分42からキャンバス部分44上にドラッグ及びドロップすることによって生成される。視覚表現42a～42fの各々は、データ構造によって又はデータ構造に対して実行される動作を表す。視覚表現がキャンバス部分44上に配置されると、それらは、キャンバス部分44上のアイコンになる。開発環境18は、データフローグラフ44aによって視覚的に表される計算論理を使用して、仕様39bを生成する。仕様39bは、キャンバス部分44において視覚的に描かれた計算論理を指定する。開発環境18は、仕様39b及び論理データ25をグラフジェネレータ22に送信する。グラフジェネレータ22は、仕様39b及び論理データ25を使用して、以下に記載のように、データフローグラフ44aの各ノードの動作フィールド及びデータプレースホルダフィールドにデータ入力することができる。

【0045】

図2Eを参照すると、環境50は、環境10の更なる詳細を示している。この例では、グラフジェネレータ22は、開発環境18（例えば、仕様及び論理データ）から受信した情報からデータフローグラフ52を生成する。コンパイラ24は、データフローグラフ52を受信し、それを実行可能プログラム54（例えば、実行可能なデータフローグラフなどのコンピュータプログラム）にコンパイルする。コンパイラ24は、実行可能プログラム54を、コンピュータプログラムの実行及び／又は記憶のためにデータ処理システム26に送信する。この例では、コンピュータプログラムは、記憶システム12から、属性が論理データに含まれた、又は仕様において指定された、又はその両方であった複数のデータセットのうちの少なくとも1つにアクセスするように構成されている。

【0046】

図3を参照すると、スイムレーン図300は、論理データを生成し、その論理データを使用して、最適化されたデータフローグラフを生成するためのプロセスを例示している。動作中、記憶システム12は、スキーマを論理データジェネレータ16に送信する（302）。論理データジェネレータ16は、スキーマを受信する（304）。論理データジェネレータ16は、スキーマを表すデータを提示するためのグラフィカルユーザインターフェース（Graphical User Interface、GUI）データを生成する（306）。論理データジェネレータ16は、GUIデータをクライアントデバイス14に送信する（308）。クライアントデバイス14は、ユーザに表示されるようにGUIデータを描画する（310）。クライアントデバイス14は、（例えば、クライアントデバイス上に描画されたGUIと対話するユーザから）ルートノード選択データを受信する（312）。ルートノード選択データは、論理データのルートノードに選択されるデータセットを指定するデータを含む。クライアントデバイス14は、ルートノード選択データを論理データジェネレータ16に送信する（314）。論理データジェネレータ16は、ルートノード選択データを受信する（316）。次いで、論理データジェネレータ16は、受信したルートノード選択データ及びスキーマを使用して論理データを生成する（318）。論理データジェネレータ16は、論理データを開発環境18及びグラフジェネレータ22に送信する（320）。いくつかの例では、論理データジェネレータ16は、論理データを開発環境18に送信し、次いで、開発環境18が、論理データをグラフジェネレータ22に渡す。

【0047】

開発環境18及びグラフジェネレータ22の各々は、論理データを受信する（322、324）。開発環境18は、論理データのフィールド又は他の属性を表示するためのGUIデータを生成する（326）。開発環境18は、GUIデータをクライアントデバイス14に送信する（328）。GUIデータは、フィールド属性などの属性、又は論理データに含まれる他の属性を表し、それにより、物理的負担を伴わずに論理アクセスを提供する。クライアントデバイス14は、受信されたGUIデータを描画し（330）、選択されたフィールド、データセット、又は他の属性を指定する選択データを受信する（332）。明確にするために、本明細書に記載の選択されたフィールド又はデータセットは、論

10

20

30

40

50

理データ自体から選択された情報を指す。いくつかの例では、選択データはまた、選択されたフィールドに対して実行される動作又は論理を指定する。クライアントデバイス 14 は、選択されたファイルを指定する選択データを開発環境 18 に送信する (334)。開発環境 18 は、選択されたフィールドを指定する選択データを受信し (336)、選択されたフィールド (及び選択されたフィールドに対して実行される動作) で仕様を生成する (338)。開発環境 18 は、仕様をグラフジェネレータ 22 に送信する (340)。

【0048】

グラフジェネレータ 22 は、仕様 (102) を受信する (342)。グラフジェネレータ 22 は、仕様及び論理データを使用してデータフローグラフを生成する (344)。概して、データフローグラフ (又は永続的コンピュータプログラム) は、以下のような仕様から生成される。仕様は、構造化データ項目 (例えば、データレコード) 内の 1 つ以上のフィールドの 1 つ以上の値を処理するためにコンピュータプログラムによって実装される複数のモジュールを指定する。これらの複数のモジュールは、ルール、命令、データフローグラフのコンポーネントなどを含み得る。本明細書に記載のシステムは、仕様を、複数のモジュールを実装するコンピュータプログラムに変換し、変換することは、複数のモジュールのうちの 1 つ以上の第 1 のモジュールの各々について、第 1 のモジュールの出力に少なくとも部分的に基づく入力を各々が受信する複数のモジュールのうちの 1 つ以上の第 2 のモジュールを識別することと、各々が (i) 第 1 のモジュールにアクセス可能であり、(ii) 第 1 のモジュールの出力に少なくとも部分的に基づいて 1 つ以上の第 2 のモジュールのうちの少なくとも 1 つへの入力として指定される構造化データ項目の 1 つ以上のフィールドの 1 つ以上の値のみを第 1 のモジュールが出力するように、第 1 のモジュールの出力データフォーマットをフォーマットすることと、「Transforming a Specification into a Persistent Computer Program」と題された米国特許出願公開第 2019/0130048 (A1) 号に記載されているように、1 つ以上の第 1 のモジュールの各々についてフォーマットされた出力データフォーマットを指定する保存されたコンピュータプログラムを用いて、コンピュータプログラムを永続メモリに保存することとを含み、その全容は参照により本明細書に組み込まれる。システムはまた、各モジュールのコンテンツがコンピュータプログラムに含まれ、かつ／又はコンピュータプログラムの適切なフォーマットにある命令に変換されることを指定する様々なルールを含む。この例では、グラフジェネレータ 22 は、最初に、論理データにおいて表されるデータソースでデータフローグラフを生成する。グラフジェネレータ 22 はまた、データフローグラフがデータシンクを必要とするため、データフローグラフにデータシンクを追加する。次いで、グラフジェネレータ 22 は、データフローグラフに、ソートコンポーネントなどのデータフローグラフの計算効率を増加させるためにグラフジェネレータ 22 が自動的に追加するように構成された様々なコンポーネントを追加する。グラフジェネレータ 22 はまた、様々なデータソースからのデータを適切に結合するために、結合コンポーネントを追加するように構成されている。データソースにアクセス又は結合するための命令、パラメータ、又は他の情報を論理データに含めることができる。最後に、グラフジェネレータ 22 は、仕様において指定された計算論理を含む変換コンポーネントを追加する。変換コンポーネント自体は、仕様が上記のようにデータフローグラフに変換されるとき、別のデータフローグラフを表す様々なコンポーネント又はサブコンポーネントを含み得る。

【0049】

例では、グラフジェネレータ 22 は、最適化されたデータフローグラフを生成するためにデータフローグラフを最適化する (346)。概して、グラフジェネレータ 22 は、仕様を分析して、どのフィールド及び関連付けられたデータソース、仕様がアクセスされているかを識別しているかを識別することによって、データフローグラフを最適化するオブティマイザを実行する。次に、オブティマイザは、フィールドが仕様において参照されていないデータソースを識別し、オブティマイザは、データフローグラフから、フィールドが仕様において参照されていないデータソースを除去する。いくつかの例では、オブティ

10

20

30

40

50

マイザは、仕様において参照されるデータセット及びフィールドのみが取り出されるように、選択ステートメント（例えば、データベースの言語で発行されたデータベース選択ステートメント）を最小化する。いくつかの例では、オプティマイザは、「Systems and Methods for Dataflow Graph Optimization」と題された米国特許出願第2019/0370407（A1）号に記載されるように、一連の最適化ルールを適用することによってこれを行い、その全容は参照により本明細書に組み込まれる。そうすることで、オプティマイザは、所望の出力を作成するために論理データが論理アクセスを提供するデータのサブセットのみを最小限にロードし結合するデータフローグラフを作成することができる。オプティマイザはまた、計算効率を改善するために、データフローグラフ内のコンポーネントの順序を再配置するなど、他の最適化を実行し得る。例えば、フィルタコンポーネントが結合コンポーネントの前にくることが計算上より効率的であり得、その結果、結合コンポーネントは最終的にフィルタ除去されるデータを結合しない。こうして、オプティマイザは、フィルタコンポーネントを結合コンポーネントの前にくるように移動させ得る。

【0050】

図4Aを参照すると、環境60は、論理データ23を使用してデータセット21dを論理データのルートノードとして識別する論理データジェネレータ16を例示している。これは、図4Aにおいて星印及び輪郭付けされているデータセット21dによって示されている。論理データジェネレータ16はまた、データセット21dに関連する他のデータセットを識別するためにスキーマ21を使用する。他の関連データセットは、データセット21a、21b、及び21cを含む。データセット21dをルートノードとして使用して、論理データジェネレータ16は、論理データ25を生成する。前述のように、論理データ25は、データセット21dが論理データ25のパースペクティブ又はルートノードであることを指定するエントリ25aを含む。エントリ25aは、データセット21dのフィールド及び／又は属性を含む。データセット（例えば、データセット21d）の属性は、データセット内のフィールドの名前、又はデータセット内のフィールドを表す他の情報を含むことができる。他の情報の中でもデータセット21d内のフィールドの名前を含めることによって、論理データ25は、記憶装置内のデータセット21dに物理的にアクセスする必要なく、データセット21d内のフィールドへのアクセスを提供する。論理データ25はまた、それぞれ、データセット21c、21b、及び21aに対応するエントリ25b、25c、及び25dを含む。この例では、エントリ25b、25c、25dは、データセット21dとの関係に従って順序付けられる。この例では、データセット21dはルートノードであり、データセット21cは子ノードである。こうして、データセット21cを表すエントリ25bは、論理データ25においてエントリ25aの直下に順序付けられる。加えて、データセット21a、21bは、データセット21cの子である。こうして、エントリ25c、25dは、データセット21a、21b、及び21cの間の関係を表すために、エントリ25bの下に順序付けられている。エントリ25b、25c、及び25dの各々は、それぞれのデータセットの属性及び／又はフィールドを含む。前述のように、これらの属性及び／又はフィールドは、フィールドの名前又は他の識別情報であり得、これにより、論理データ25が、データセットデータセット21a、21b、21c、及び21dに記憶装置から実際にアクセスする負担を伴わずに、それらのデータセットへの論理アクセスを提供することが可能になる。論理データ25は、データセット21a、21b、21c、及び21dの属性又はフィールドを識別するために使用することができ、かつ／又は必要に応じてそれらのデータセットにアクセスするために使用することができる情報を含むため、論理アクセスを提供することができる。

【0051】

図4Bを参照すると、環境70は、環境60（図4A）の変動を示し、環境70において、データセット21bが、破線及び星印の輪郭によって示されるように、ルートノードとして選択されている。データセット21bは、例えば、ユーザが図2Bの視覚表現28bを選択するときに、ルートノードとして選択される。データセット21bがルートノード

ドとして選択されると、論理データジェネレータ16は、論理データ72を生成し、論理データ72において、図4Aに示されるように、データセット21bがルートノードとして指定され、論理データにおける他のデータセットの順序が、論理データ25におけるデータセットの順序に対して変更されている。この例では、論理データ72は、データセット21bを表すエントリ72aを含む。この例では、データセット21bは、データセット21cの子であり、エントリ72bは、データセット21cを表す論理データ72に含まれる。データセット21aは、データセット21cの子であり、エントリ72cは、データセット21aを表す論理データ72に含まれる。データセット21cは、データセット21dの子であり、エントリ72dは、データセット21dを表す論理データ72に含まれる。図4Aを参照して上述したように、エントリ72a、72b、72c、及び72dの各々は、それぞれのデータセットの各々についての属性又はフィールドに関する情報、及び／又はデータセットの特性若しくはデータセットにどのようにアクセスするかを指定する他の情報を含む。

10

【0052】

図4Cを参照すると、環境80は、データベーススキーマ84に対する論理データ82の生成を例示している。この例では、論理データジェネレータ16は、データベーススキーマ84を受信し、また、(破線及び星印の輪郭によって示されるように)スキーマ84においてデータセット84dがルートノードであることを指定する選択データ23を受信する。この例では、スキーマ84は、データセット84a、84b、84c、84d、及び84eを含む。例では、スキーマ84は、データセット84d(例えば、データセット84dのフィールドのフィールド又は値)に対して計算を実行すること、さもなければデータセット84dを変換してデータセット84eを作り出すことを行うための命令を含む。例えば、スキーマ84は、1つ以上の動作を実行するための命令、又は入力としてデータセット84d(又はその一部)を含み、出力としてデータセット84e(又はその一部)を作り出す実行可能プログラム(例えば、データフローグラフ)を呼び出す命令を含むことができ。いくつかの例では、これらの計算、変換、又は他の動作は、スキーマ84において動作を指定する命令を含むことなどによって、スキーマ84において直接定義される。いくつかの例では、スキーマ84は、動作を実行する命令にアクセスするためのリンク、ポインタ、又は他の情報を含むことができる。いくつかの例では、これらの動作は以前に実行され、動作によって作り出されたデータセット84eは、記憶システムに記憶された物理データセットである。いくつかの例では、データセット84eは、1つ以上の計算された属性若しくは仮想属性、仮想フィールド、又は実行中にデータ入力される他の仮要素(例えば、データセット84eがデータフローグラフなどのコンピュータプログラムで使用されるとき)など、仮想データを含む。

20

30

【0053】

データセット84dがルートノードであるため、論理データジェネレータ16は、エントリ82a～82eを有する論理データ82を生成する。エントリ82aは、ルートノードであるデータセット84dを表す。エントリ82aは、データセット84dの属性を含むことができる。本明細書で前述したように、属性は、フィールドの名前、フィールドへのポインタなどを含む。データセット84c及び84eがデータセット84dの子であるため、論理データ82における次のエントリは、データセット84eを表すエントリ82b及びデータセット84cを表すエントリ82cである。エントリ82b及び82eの各々は、フィールド属性を含む。データセット84a及び84bがデータセット84cの子であるため、論理データ82における次のエントリは、データセット84bを表すエントリ82d及びデータセット84aを表すエントリ82eである。エントリ82d及び82eの各々は、フィールド属性を含む。

40

【0054】

図5Aを参照すると、環境90は、論理データを生成し、論理データを使用して最適化されたデータフローグラフを生成する実際の実施例の概要を示している。この例では、論理ジェネレータ16は、スキーマ91を記憶システムから受信する。論理データジェネレ

50

ータ 16 はまた、クライアントデバイス 14 から選択されたルートノードを示す選択データ 92 を受信する。このスキーマ 91 及び選択データ 92 を使用して、論理データジェネレータ 16 は、本明細書に記載の技術に従って論理データ 94 を生成する。論理データジェネレータ 16 は、論理データ 94 を開発環境 18 に送信する。この論理データ 94 を使用して、開発環境 18 は、記憶の基礎となる物理データセットにアクセスすることなく、属性又はフィールドなど、論理データ 94 に含まれる情報を、（例えば、クライアントデバイス 14 を使用して）開発環境 18 と対話することができるユーザによって閲覧可能又はアクセス可能にするグラフィカルユーザインターフェース又は他のユーザインターフェースを生成する。ユーザは、選択された属性の実行又は使用のために、開発環境 18 を使用して、論理データ 94 内の属性のうちの少なくとも 1 つ並びに 1 つ以上の動作を選択する。この情報に基づいて、開発環境 18 は、論理データ 94 の属性及び／又はフィールドのうちのどれがデータフローグラフを生成する際に含まれるべきかを指定する仕様 96 a を作り出す。グラフジェネレータ 22 は、論理データ 94 及び仕様 96 a を受信し、仕様 96 a において指定された属性と関連付けられた（さもなければ、仕様 96 a で動作を実行するために必要とされる）物理データセットのみにアクセスするように最適化されたデータフローグラフ 98 a を作り出す。

【0055】

同じ又は異なるユーザが、開発環境 18 を使用して、論理データ 94 の 1 つ以上の異なる属性、選択された属性に対して実行する 1 つ以上の異なる動作、又はその両方を選択し得る。例えば、ユーザは、データフローグラフ 98 a の処理において識別されたエラーに
20 応答して、仕様 96 a において指定された選択された属性又は動作を変更してもよく、又は異なる属性及び動作を選択して、完全に新しいデータフローグラフを作り出してもよい。この情報を使用して、開発環境は、仕様 96 a とは異なる仕様 96 b を作り出す。グラフジェネレータ 22 は、論理データ 94 及び仕様 96 b を受信し、仕様 96 b において指定された属性と関連付けられた物理データセットのみにアクセスするために、データフローグラフ 98 a とは異なるように最適化されたデータフローグラフ 98 b を作り出す。このようにして、論理データ 94 は、そこに含まれるデータセット及び属性の全てへの論理
30 アクセスを可能にするが、それを行う物理的負担を伴わない。これは、エンドユーザ（例えば、開発環境 18 のユーザ）に多大な柔軟性を提供し、エンドユーザは、論理データ 94 に含まれる全ての物理データを（そのようなデータに物理的にアクセスする負担を伴わずに）閲覧及び選択することができ、それらの仕様を遂行するために必要な物理データのみにアクセスするように個別対応された高度に最適化されたデータフローグラフを取得することができる。

【0056】

図 5 B を参照すると、環境 100 は、環境 90 の更なる詳細を示している。この例では、記憶システム 12 は、スキーマ 91 をデータセット 101、102、103、104 と共に記憶する。「オファー状況」データセット 101 は、「キー」フィールド 101 a 及び「オファー受入」フィールド 101 b を含む。フィールド 101 a は、例えば、主キー、外部キー、又はその両方（別個のフィールドで定義され得る）を含むことができる。「
40 分数」データセット 102 は、フィールド 102 a、102 b、102 c、及び 102 d を含む。「顧客」データセット 103 は、フィールド 103 a 及び 103 b を含む。「リロード日」データセット 104 は、フィールド 104 a 及び 104 b を含む。この例では、「残りの分数」フィールド 102 d は、例えば、上記のようにスキーマ 91 において定義された仮想フィールド又は計算フィールドである。例えば、スキーマ 91 は、1 つ以上の動作又は、データセット 102 若しくは別のデータセット内の 1 つ以上の他のフィールドからフィールド 102 d を生成する、他の命令を指定し得る。特に、スキーマ 91 は、フィールド 102 b と 102 c との間の差としてフィールド 102 d を定義し得る。この例では、正方形の括弧を使用して、フィールド 102 d が仮想フィールド又は計算フィールドであることを示している。この例では、データセット 101、102、103、104 は、それらのキーの値を介して互いに関連している。すなわち、データセット 101、
50

102、103、104の各々は、互いに一致するキーの値を有し、1つのデータセットからのデータを別のデータセットと結合するために使用することができる。

【0057】

論理データジェネレータ16は、スキーマ91を記憶システム12から受信する。クライアントデバイス14は、（例えば、論理データジェネレータ16（図示せず）から受信されたスキーマ91にどのデータセットが含まれるかを指定するGUIデータに基づいて）グラフィカルユーザインターフェース105を表示する。GUI105は、データセット部分106及び選択されたパースペクティブ部分107を含む。データセット部分106は、それぞれ、データセット101、102、103、104の視覚表現106a、106b、106c、106dを含む。選択されたパースペクティブ部分107は、ボタン107aを含み、その選択により、閲覧者が視覚表現106a～106dのうちの1つを選択することが可能になる。この例では、ユーザは、データセット103を表す視覚表現106cを選択している。これが選択されると、選択されたパースペクティブ部分107は、論理データジェネレータ16によって生成される論理データのルートノードとしてデータセット107が選択されていることを指定しているアイコン107bで更新される。クライアントデバイス14は、データセット103がルートノードとして選択されることを指定する選択データ92を生成する。クライアントデバイス14は、選択データ92を論理データジェネレータ16に送信する。論理データジェネレータ16は、スキーマ91及び選択データ92を使用して、論理データ94を作り出す。

【0058】

図5Cを参照すると、環境110は、スキーマ91及び選択データ92から論理データ94を生成する例示を示している。この例では、論理データ94は、図5Bに示されるように、データセット101、102、103、104及びそれぞれのフィールドにどのようにアクセスするかを指定する一連の命令、パラメータ、又は他の情報を含む。いくつかの例では、論理データ94は、命令、パラメータ、又はフィールド102dなどの仮想フィールド若しくは計算フィールドをどのように生成するか、さもなければどのようにアクセスするかを指定する他の情報を含む。いくつかの例では、論理データは、基礎となるデータセットの属性、フィールド、又は他の特徴を含むワイドレコードに具体化される。論理データジェネレータ16は、論理データ94を開発環境18に送信する。

【0059】

図5Dを参照すると、環境120は、ビジネスルール及び他の論理ルールを定義するために開発環境18によって生成されたビジネスルールエディタインターフェース121の例を示している。概して、エディタインターフェース121は、式を含むセルをグラフィカルに識別し得る。これは、ユーザが、それ自体で真又は偽に評価されることになる式と、列変数に対して比較される値を返す式との間の差を理解するのに役立つことになる。ユーザがタイプ入力しているとき、ユーザは、例えば、式の始めにアスタリスクをタイプ入力することによって、特定のセルが式セルであることを示すことができる。この例では、エディタインターフェース121は、入力部分122及びルール定義部分123を含む。入力部分122は、論理データ94で表されるフィールド及びデータセット（下向きの矢印によって示されるような拡張ビューに示されている）、並びに他のデータソースの視覚表現（右向きの矢印によって示されるような折り畳みビューに示されている）を提供する。例えば、入力部分122は、（図5Bに示される）データセット101を表す視覚表現124を含む。入力部分122はまた、データセット101において「オファー受入」フィールド101bを表す視覚表現124aを含む。この例では、視覚表現124aは、視覚表現124からインデントされることによって、データセット101においてフィールドであるとして視覚的に描かれている。入力部分122はまた、それぞれ、データセット102及びフィールド102b、102c、及び102dを表す視覚表現125及び125a、125b、及び125cを含む。入力部分122はまた、それぞれ、データセット103及びフィールド86bを表す視覚表現126及び126aを含む。入力部分122はまた、それぞれ、データセット104及びフィールド104bを表す視覚表現127及

び 1 2 7 a を含む。この例では、入力部分 1 2 2 内の視覚表現は、ビジネスルールを定義する際にユーザに利用可能なデータセット及びフィールドを表す。入力部分 1 2 2 において表される利用可能なデータセット及びフィールドは、論理データ 9 4 から識別され、それによって、データセット及びフィールドへのアクセスをユーザに提供するが、それらのデータセット（又はフィールド）に物理的メモリから実際にアクセスする必要はない。

【 0 0 6 0 】

ルール定義部分 1 2 3 は、一連のルールケースを含む。この例では、ルール定義部分 1 0 6 は、スプレッドシートフォーマットを含む。スプレッドシート内のトリガ列 1 2 8 a、1 2 8 b、及び 1 2 8 c は、利用可能なデータ値に対応し、行 1 2 9 c ~ 1 2 9 g は、ルールケース、例えば、利用可能なデータ値を関連付ける基準のセットに対応している。ルールケースは、所与のレコードのデータ値が、ルールケースが基準を有する各トリガ列のトリガ基準を満たす場合、そのレコードに適用される。ルールケースが適用される場合、出力列 1 2 9 a に基づいて出力が生成される。その入力関係の全てが満たされているルールケースは、「トリガされた」と称され得る。出力列 1 2 9 a は、潜在的な出力変数に対応し、該当する行の対応するセル内の値は、その変数について、存在する場合、出力を決定する。セルは、変数に割り当てられた値を含むことができ、又は以下で考察されるように、出力値を生成するために評価されなければならない式を含むことができる。図 5 D には 1 つのみが示されているが、2 つ以上の出力列があってもよい。

【 0 0 6 1 】

特に、行 1 2 9 a は、ルールの相対的な入力及び出力を指定する。行 1 2 9 b は、ルールを定義する際に使用されるフィールド及び出力が何であるかを指定する。この例では、行 1 2 9 b は、セル 1 2 8 a、1 2 8 b、及び 1 2 8 c を含む。セル 1 2 8 a は、入力部分 1 2 2 において視覚表現 1 2 6 a の周りの点線によって視覚的に描かれるように、視覚表現 1 2 6 a のユーザ選択時にルール定義部分 1 2 3 に追加される。この選択の結果として、セル 1 2 8 a は、「名前」フィールド 1 0 3 b e が（図 5 B に示される）、ルール定義部分 1 2 3 において指定されたルールを定義する際の入力として使用されることを指定する。セル 1 2 8 b は、「残りの分数」フィールド 1 0 2 d（図 5 B に示される）はまた、ルール定義 1 2 3 において示されるルールを定義する際の入力として使用されることを指定する。この例では、視覚表現 1 2 5 c が選択されると、セル 1 2 8 b は、「残りの分数」フィールド 1 0 2 d がルールへの入力として使用されることを表すように更新される。同様に、セル 1 2 8 c は、「使用された分数」フィールド 1 0 2 c（図 5 B に示される）を表す視覚表現 1 2 5 b が選択された後に、「使用された分数」1 0 2 c もまた、ルール定義 1 2 3 において示されるルールを定義する際の入力として使用されることを指定する。セル 1 2 8 a、1 2 8 b、及び 1 2 8 c は、ユーザが、記憶システム 1 2 に記憶されたデータセットからフィールドの属性（フィールドの名前など）にアクセスすることができることを例示するが、それらのデータセット（又はフィールド）自体に物理的にアクセスする必要はない。ルール定義部分 1 2 3 はまた、ルールケースの様々な基準が満たされたときに様々なルールケース及び出力を指定する行 1 2 9 c、1 2 9 d、1 2 9 e、1 2 9 f、及び 1 2 9 g を含む。

【 0 0 6 2 】

ルール定義部分 1 2 3 においてセルの入力を指定することによってルールの定義を完了すると、開発環境 1 8 は、ルールケース、及びルールを実装するためにどのフィールドにアクセスすることが必要になるかを指定するルール仕様 9 6 a を生成する。この例では、ルール仕様 9 6 a は、「名前」フィールド 1 0 3 b、「残りの分数」フィールド 1 0 2 d、及び「使用された分数」フィールド 1 0 2 c（各々、図 5 B に示される）をルールの入力として使用されることを指定する。すなわち、これらのフィールドの値は、ルールの入力として使用される。こうして、ルール自体を実行すると、それらのフィールドは、ルールを実行するときに物理的にアクセスすることが必要となる。しかしながら、ルールを定義するこの段階では、論理データ 9 4 は、ユーザに、物理アクセスなしにそれらのフィールドへの論理アクセスを提供する。例えば、ユーザは、入力部分 1 2 2 において記憶シス

10

20

30

40

50

テム 12 に記憶された様々なデータセットから利用可能なフィールドを閲覧することができることによって、論理アクセスが提供された。開発環境 18 は、ルール仕様 96 a をグラフジェネレータ 22 に送信する。開発環境 18 はまた、論理データ 94 をグラフジェネレータ 22 に送信する。

【0063】

図 5 E を参照すると、環境 130 は、ルール仕様 96 a 及び論理データ 94 からデータフローグラフを生成及び最適化する例を示している。グラフジェネレータ 22 は、ルール仕様 96 a 及び論理データ 94 を受信する。グラフジェネレータ 22 は、最適化されたデータフローグラフ 98 a を生成する際に、オブティマイザ 132 をルール仕様 96 a 及び論理データ 94 の両方に適用する。この例では、グラフジェネレータ 22 は、データフローグラフ 134 を生成するためにルール仕様 96 a 及び論理データ 94 を使用する。この例では、データフローグラフ 134 は、コンポーネント 134 a ~ 134 m を含む。次いで、グラフジェネレータ 22 は、オブティマイザ 132 をデータフローグラフ 134 に適用する。概して、オブティマイザ 132 は、データフローグラフ（例えば、データフローグラフ 134）における冗長性を低減し、データフローグラフによって使用されていないデータソースを排除する。すなわち、ルールが特定のデータソースからフィールド（例えば、データセット）にアクセスすることをルール仕様 96 a が指定しない場合、オブティマイザ 132 は、そのデータソースをデータフローグラフから除去することになる。いくつかの例では、オブティマイザ 132 は、（例えば、ソースデータがリレーショナルデータベースに記憶されているときに）選択ステートメントを最小化することによってこれを行い、それにより、ルール仕様 96 a において指定され、論理データ 94 に含まれているデータセット及びフィールドのみがアクセスされる。

【0064】

初期に、グラフジェネレータ 22 は、例えば、論理データ 94 において指定されたデータセットにアクセスするための命令、パラメータ、又は他の情報に基づいて、データソースとして論理データ 94 に含まれるデータセット及びフィールドでデータフローグラフ 134 を生成する。この例では、データフローグラフ 134 内のコンポーネント 134 a ~ 134 m は、論理データ 94 において表されるデータソース（例えば、データセット）に基づいている。いくつかの例では、グラフジェネレータ 22 はまた、仕様 96 a 又は論理データ 94、又はその両方に含まれる情報をデータフローグラフ 134 にどのように変換するかを指定する組み込み機能に依存し得る。例えば、組み込み機能は、例えば、仕様 96 a 若しくは論理データ 94、又はその両方からの情報に基づいて、とりわけ、ソート、パーティション、又は結合動作などの様々な動作をデータフローグラフに挿入する機能を含むことができる。

【0065】

データフローグラフ 134 はまた、1つ以上の変換コンポーネントを含むことができる。概して、変換コンポーネントは、1つ以上のデータソース、例えば、入力データセットから入力レコードを受信し、計算論理に基づいて出力レコードを作り出す。変換コンポーネントを作り出すために、グラフジェネレータ 22 は、入力に適用されるように、論理の仕様（例えば、仕様 96 a からのルールセット、又は論理データ 94 からの命令、パラメータ、又は他の情報）を受信することができる。次いで、グラフジェネレータ 22 は、データフローを表すリンク要素によって接続されたデータ処理コンポーネントを有するグラフベースの計算として変換を生成し実装することができる。この例では、データフローグラフ 134 は、ルール仕様 96 a において指定されたルールを実行する論理を含む変換コンポーネント 134 l を含む。この例では、データフローグラフ 134 はまた、計算フィールド 102 d を生成するための論理を含む変換コンポーネント 134 i を含む。この例では、生成された変換は、データフローグラフ 134 におけるコンポーネント（例えば、コンポーネント 134 l）である。グラフジェネレータ 22 はまた、例えば、ルールセットが編集されたときに変換を更新し得る。例えば、ルールセットが編集されたとき、エディタ（例えば、開発環境 18）は、ルールセット全体をグラフジェネレータ 22 に提供す

るか、又は新しい若しくは修正されたルール若しくはルールケースのみを提供してもよい。グラフジェネレータ 22 は、変換を使用するシステムの能力及びニーズに応じて、元の変換を置き換えるために完全に新しい変換を生成してもよく、又はグラフジェネレータ 22 は、変換を含む新しいコンポーネントを提供してもよい。

【0066】

グラフジェネレータ 22 は、オブティマイザ 132 をデータフローグラフ 134 に適用して、データフローグラフ 136 を生成する。オブティマイザ 132 は、データフローグラフ 136 のクロスアウト部分によって示されるように、データフローグラフ 134 からコンポーネント 134 a、134 c、134 f、134 g、134 j を除去する。オブティマイザ 132 は、これらのコンポーネントがルール仕様 96 a によって参照又は使用されないデータセットに関連するため、これらのコンポーネントを除去することを決定する。すなわち、ルール仕様 96 a は、除去されるデータセットに含まれるいずれのフィールドへの参照も含まない。いくつかの例では、ルートノード（例えば、この例におけるデータセット 103 又はコンポーネント 134 b）としての役割を果たすデータセットは、ルール仕様 96 a によって使用されるかどうかにかかわらず、最適化されない場合があることに留意されたい。最適化の最終結果は、ルール仕様 96 a によって指定されたルールを実行するために必要とされないデータセットの全て、並びにそれらのデータセットにアクセスするためにインスタンス化された他のコンポーネント（例えば、ソート、結合など）を除去するように最適化されたデータフローグラフ 98 a である。

【0067】

図 5 F を参照すると、環境 140 は、ビジネスルール及び他の論理ルールを定義するために開発環境 18 によって生成されたビジネスルールエディタインターフェース 121 の別の例を示している。環境 140 では、ルール定義 123 は、図 5 D に示される環境 120 におけるルール定義に対して変更されている。具体的には、トリガセル 128 b 及び 128 c が削除されており、セル 128 a は、「使用された分数」フィールド 102 c（図 5 B に示される）が、「使用された分数」フィールド 102 c を表す視覚表現 125 b を選択した後にルール定義 123 に示されるルールを定義する際の唯一の入力であることを指定するように修正されている。列 129 d、129 e、129 f、及び 129 g のルールケースも更新されている。そのため、開発環境 18 は、図 5 D に示されるルール仕様 96 a の修正バージョンであるルール仕様 142 を生成する。この例では、ルール仕様 142 は、「使用された分数」フィールド 102 c がルールの唯一の入力として使用されることを指定する。開発環境 18 は、ルール仕様 142 をグラフジェネレータ 22 に送信する。開発環境 18 はまた、論理データ 94 をグラフジェネレータ 22 に送信する。

【0068】

図 5 G を参照すると、環境 150 は、修正された仕様 142 及び論理データ 94 からデータフローグラフを生成及び最適化する例を示している。初期に、グラフジェネレータ 22 は、修正された仕様 142 において指定されたルールを実行する論理を含む変換コンポーネント 152 l を除いて、図 5 E に示されるデータフローグラフ 134 に類似したデータフローグラフ 152 を生成する。この例では、修正された仕様 142 において指定され、コンポーネント 152 l によって実装されるルールが、仕様 96 a で指定され、コンポーネント 134 l によって実装されるルールとは異なるため、変換コンポーネント 152 l は、変換コンポーネント 134 l（図 5 E に示される）とは異なる。グラフジェネレータ 22 は、オブティマイザ 132 をデータフローグラフ 152 に適用して、データフローグラフ 154 を生成する。そうすることで、オブティマイザ 132 は、データフローグラフ 154 のクロスアウト部分によって示されるように、データフローグラフ 154 からコンポーネント 134 a、134 c、134 f、134 g、134 j、及び 134 i を除去する。オブティマイザ 132 は、これらのコンポーネントがルール仕様 142 によって参照又は使用されないデータセットに関連するため、これらのコンポーネントを除去することを決定する。ルートノード（例えば、この例におけるデータセット 103 又はコンポーネント 134 b）として役割を果たすデータセットが仕様 140 において参照されてい

いが、最適化されていないことに留意されたい。最適化の最終結果は、ルール仕様 1 4 2 によって指定されたルールを実行するために必要とされないデータセットの全て、並びにそれらのデータセットにアクセスするためにインスタンス化された他のコンポーネント（例えば、ソート、結合など）を除去するように最適化されたデータフローグラフ 1 5 6 である。データフローグラフ 1 5 6 は、それぞれのグラフの仕様 9 6 a、1 4 2 において依存される異なる属性に起因して、同じ論理データ 9 4 ソースを使用するにもかかわらず、データフローグラフ 9 8 a とは異なる。

【0069】

図 5 H を参照すると、環境 1 6 0 は、ビジネスルール及び他の論理ルールを定義するために開発環境 1 8 によって生成されたビジネスルールエディタインターフェース 1 2 1 の更に別の例を示している。環境 1 4 0 において、ルール定義 1 2 3 は、それぞれ、図 5 D 及び図 5 F に示される環境 1 2 0 及び 1 4 0 におけるルール定義に対して変更されている。ここで、セル 1 2 8 a は、「最終リロード」フィールド 1 0 3 b（図 5 B に示す）を表す視覚表現 1 2 7 a を選択した後に、「最終リロード」フィールド 1 0 3 b が、ルール定義 1 2 3 に示されるルールを定義する際の唯一の入力であることを指定する。列 1 2 9 c、1 2 9 d、1 2 9 e、1 2 9 f、及び 1 2 9 g のルールケースも変更されている。そのため、開発環境 1 8 は、ルール仕様 9 6 a（図 5 A に最初に示されている）、1 4 2 の各々とは異なるルール仕様 9 6 b を生成する。この例では、ルール仕様 9 6 b は、「最終リロード」フィールド 1 0 3 b がルールの唯一の入力として使用されることを指定する。開発環境 1 8 は、ルール仕様 9 6 b をグラフジェネレータ 2 2 に送信する。開発環境 1 8 はまた、論理データ 9 4 をグラフジェネレータ 2 2 に送信する。

【0070】

図 5 I を参照すると、環境 1 7 0 は、仕様 9 6 b 及び論理データ 9 4 からデータフローグラフを生成及び最適化する例を示している。初期に、グラフジェネレータ 2 2 は、修正された仕様 9 6 b（この例では、変換コンポーネント 1 3 4 l 及び 1 5 2 l の各々とは異なる）において指定されたルールを実行する論理を含む変換コンポーネント 1 7 2 l を除いて、それぞれ、図 5 E 及び図 5 G に示されるデータフローグラフ 1 3 4 及び 1 5 2 に類似するデータフローグラフ 1 7 2 を生成する。グラフジェネレータ 2 2 は、オブティマイザ 1 3 2 をデータフローグラフ 1 7 2 に適用して、データフローグラフ 1 7 4 を生成する。そうすることで、オブティマイザ 1 3 2 は、データフローグラフ 1 7 4 のクロスアウト部分によって示されるように、データフローグラフ 1 5 4 からコンポーネント 1 3 4 a、1 3 4 c、1 3 4 f、1 3 4 e、1 3 4 h、及び 1 3 4 i を除去する。オブティマイザ 1 3 2 は、これらのコンポーネントがルール仕様 9 6 b によって参照又は使用されないデータセットに関連するため、これらのコンポーネントを除去することを決定する。ルートノード（例えば、この例におけるデータセット 1 0 3 又はコンポーネント 1 3 4 b）として役割を果たすデータセットが仕様 1 4 0 において参照されていないが、最適化されていないことに留意されたい。最適化の最終結果は、ルール仕様 9 6 b によって指定されたルールを実行するために必要とされないデータセットの全て、並びにそれらのデータセットにアクセスするためにインスタンス化された他のコンポーネント（例えば、ソート、結合など）を除去するために最適化されたデータフローグラフ 9 8 b（図 5 A に最初に示されている）である。データフローグラフ 9 8 b は、それぞれのグラフの仕様において依存される異なる属性に起因して、同じ論理データ 9 4 ソースを使用するにもかかわらず、データフローグラフ 9 8 a 及び 1 5 6 とは異なる。

【0071】

図 5 J を参照すると、環境 1 8 0 は、データフローグラフ 9 6 a の実行の結果を示している。グラフ生成システム 1 8 は、データフローグラフ 9 6 a をコンパイラ 2 4 に送信する。コンパイラ 2 4 は、以下のように、データフローグラフ 9 6 a を実行可能プログラム 1 8 2 にコンパイルする。

【0072】

データフローグラフは、計算プロセスを表す複数の頂点であって、各頂点が、関連付け

10

20

30

40

50

られたアクセス方法を有する頂点として、及び、複数のリンクであって、リンクの各々が、互いに少なくとも2つの頂点を接続し、接続された頂点間のデータの流れを表す、複数のリンクとして、計算を表す。データフローグラフは、(1) データフローグラフをユーザ入力としてコンピューティングシステムに受け取ることと、(2) 各頂点の実行可能状態になり、各リンクが、リンクによって接続された頂点のアクセス方法と互換性のある少なくとも1つの通信方法と関連付けられるまでに、グラフ変換ステップをコンピューティングシステム上で実行することによって、実行のためのデータフローグラフを準備することと、(3) コンピューティングシステムによって、リンクの通信方法に応じて、通信チャネル及び／又はデータストアの組み合わせを作成することによって、各リンクを起動することと、(4) プロセスの実行を呼び出すことによってコンピューティングシステム上で各プロセスを起動することと、によって実行される。

10

【0073】

概して、実行のためのデータフローグラフは、以下のように準備される。

【0074】

ドライバプログラム（又は、単に、略して「ドライバ」）は、ユーザインターフェースを通して受信されたユーザからの入力に基づいて、データフローグラフを描くための手段を提供する。データフローグラフの視覚表現を表す1つ以上のデータフローグラフデータ構造が、ドライバによって生成される。ドライバは、初期にユーザによって描かれたデータフローグラフにアクセスし、グラフ変換を適用することによって実行のためのデータフローグラフを準備する。これらの変換を実行するときに、初期データフローグラフを定義するデータフローグラフデータ構造は、各頂点及び任意の関連付けられたリンクをフェッチするために、既知の様式でトラバースされる。いくつかの例では、以下に記載されるように、実行のためのデータフローグラフを準備するために、フェッチされたデータ構造に対して5つのデータフローグラフ変換が使用される。

20

【0075】

データフローグラフは依然として実行可能な形態ではないが、以下に記載される5つのデータフローグラフ変換が、実行可能なデータフローグラフが取得されるまで、任意の順序で、しばしば必要に応じて（全くないことを含む）、選択及び適用され得る。5つのデータフローグラフ変換は、(1) ファイルアダプタを挿入することと、(2) 通信アダプタを挿入することと、(3) ファイル頂点の状態を `Complete` に設定することと、(4) プロセス頂点の状態を `Runnable` 又は `Unrunnable` に設定することと、(5) データリンクの通信方法を設定することと、を含む。これらの変換の各々及び各々が実行され得る条件をここに記載する。

30

【0076】

ファイルアダプタを挿入すること

この変換では、ドライバは、リンクをファイルアダプタに（すなわち、リンク、ファイル頂点、及び別のリンクに）置き換える。すなわち、データフローグラフデータ構造のトラバース中にリンクを表す各データフローグラフデータ構造がフェッチ又はアクセスされるとき、元のデータ構造を修正、拡張、又は置換する新しいデータ構造が作成され得る。

【0077】

ソース（宛先）ファイルアダプタの場合、ファイル頂点のホストは、ソース（宛先）頂点のホストと同じであり、ファイル頂点のファイルは、ソース（宛先）頂点の作業ディレクトリ内に位置する新しいファイルである。この変換は、

40

(1) ソースが、ファイル頂点又は `Done` 状態にないプロセス頂点のいずれかである場合、及び

(2) 宛先が、`Incomplete` 状態のファイル頂点又は `Done` 状態にないプロセス頂点のいずれかである場合、に実施され得る。

【0078】

通信アダプタを挿入すること

この変換では、ドライバは、リンクを通信アダプタに（すなわち、リンク、プロセス頂

50

点、及び別のリンクに) 置換する。プロセス頂点は、コピープログラムを実行し、コピープログラムはデータをその入力からその出力へコピーするものであり、基礎となるサブストレイトによって支持される通信チャネル又はデータストアのうちのいずれかから読み取り／そこに書き込みすることができる。ソース(宛先)通信アダプタの場合、プロセス頂点のホストは、ソース(宛先)頂点のホストと同じであり、作業ディレクトリは、ソース(宛先)頂点の作業ディレクトリと同じである。プロセス頂点は、`Enabled`状態で作成される。この変換は、

(1) ソースが、`Done`以外の状態のプロセス頂点、又はファイル頂点のいずれかである場合、及び

(2) 宛先が、`Done`以外の状態のプロセス頂点、又は`Incomplete`状態のファイル頂点のいずれかである場合、に作成され得る。

10

【0079】

ファイル頂点の状態を`Complete`に設定すること

この変換では、ファイル頂点の状態が`Complete`に設定される。この変換は、ファイル頂点の状態が不完全であり、ファイル頂点への全ての入力が`Done`状態のプロセス頂点である場合に実行され得る。

【0080】

プロセス頂点の状態を`Runnable`又は`Unrunnable`に設定すること

この変換において、プロセス頂点の状態は、`Runnable`又は`Unrunnable`のいずれかに設定される。この変換は、プロセス頂点の状態が`Enabled`である場合に実行され得る。

20

【0081】

データリンクの通信方法を設定すること

この変換では、通信方法がデータリンクに設定される。この変換は、データリンクの通信方法が`Unbound`である場合に実行され得る。

【0082】

以下の3つのプロパティを有するデータフローグラフが実行される。

(1) 全てのプロセス頂点が、次の状態、すなわち、`Done`、`Runnable`、`Unrunnable`、又は`Disabled`のうちの1つにある。

(2) 全てのデータリンクが、以下の基準の全てを満たす。

30

1) データリンクのソース又は宛先が`Runnable`プロセス頂点である場合、データリンクの通信方法は、特定の通信方法にバインドされなければならない。

2) データリンクの通信方法が`File`以外のものである場合、そのソース及び宛先の両方がプロセス頂点でなければならない、1つのプロセス頂点が`Runnable`である場合、両方のプロセス頂点が`Runnable`でなければならない。

3) データリンクの通信方法が`File`である場合、そのソース又は宛先がファイル頂点でなければならない。宛先が`Runnable`プロセス頂点である場合、ソースは`Complete`ファイル頂点でなければならない。ソースが`Runnable`ファイル頂点である場合、宛先が`Incomplete`ファイル頂点でなければならない。

(3) 通信方法にバインドされた全てのリンクは、通信方法に固有の制約を満たす。

40

1) 通信方法が、そのソース及び宛先ポートのアクセス方法と互換性がなければならない(これは、プログラムテンプレートを参照することによって決定され得る)。前述の拡張サブストレイトの場合、全ての通信方法が、`SOC`アクセスと互換性があり、`Shared Memory`以外の全てが、`File Descriptor`アクセスと互換性があり、`Named Pipe`及び`File`が、`Named Pipe`アクセスと互換性があり、ファイルのみが`File`アクセスと互換性がある。

2) いくつかの通信方法は、ソース及び宛先の頂点のノードが同一であることを必要とする。前述の拡張サブストレイトの場合、これは、`TCP/IP`以外の全ての通信方法について当てはまる。

【0083】

50

データフローグラフ変換は、実行可能グラフが取得されるまで、任意の順序で適用され得る（例えば、データフローグラフデータ構造は、全ての変換が完了するまで繰り返しトラバースされ得る）。いくつかの例では、データフローグラフ変換は、以下の順序、すなわち、（１）ファイルアダプタを挿入すること、（２）ファイル間リンクを置換すること、（３）Completeファイル頂点を識別すること、（４）Unrunnableプロセス頂点を識別すること、（５）Runnableプロセス頂点を識別すること、（６）残りのEnabled頂点をUnrunnableに設定すること、（７）条件が満たされる場合、更にファイルアダプタを挿入すること、（８）通信方法を選択すること、及び（９）通信アダプタを挿入すること、で適用される。ここで、この例のステップをより詳細に記載する。

10

【0084】

（１）ファイルアダプタを挿入すること

ファイルアダプタを挿入するために、データフローグラフ内の全てのリンクについて、以下のステップが実行される。リンクのソースポートがファイルの使用を必要とするデータアクセス方法を有し、宛先が同じノード上のファイルではない場合、ソースファイルアダプタを挿入する。リンクの宛先ポートがファイルの使用を必要とするデータアクセス方法を有し、ソースが同じノード上のファイルではない場合、宛先ファイルアダプタを挿入する。リンクの宛先がDisabled状態のプロセス頂点であり、ソースがEnabled状態のプロセス頂点である場合、宛先ファイルアダプタを挿入する。

【0085】

（２）ファイル間リンクを置換すること

ファイル間リンクを置換するために、データフローグラフの全てのリンクについて、以下のステップが実行される。ソース及び宛先が両方ともファイル頂点である場合、ソース通信アダプタを挿入する。（加えて、ソース及び宛先が異なるノード上にある場合、宛先通信アダプタも挿入する、図示せず）。

20

【0086】

（３）Completeファイル頂点を識別すること

Completeファイル頂点を識別するために、データフローグラフの全てのファイル頂点について、以下のステップが実行される。全てのアップストリーム頂点がDone状態にあるプロセス頂点である場合、その状態をCompleteに設定する。

30

【0087】

（４）Unrunnableプロセス頂点を識別すること

Unrunnableプロセス頂点を識別するために、データフローグラフの全てのリンクについて、以下のステップが実行される。「Unrunnability」テストは、次のように実行される、すなわち、リンクのソースがIncompleteファイル頂点であり、その宛先がEnabled状態のプロセス頂点である場合、プロセス頂点の状態をUnrunnableに設定し、ソースがEnabled以外の任意の状態のプロセス頂点であり、宛先がEnabled状態のプロセス頂点である場合、宛先プロセス頂点をUnrunnableとしてマーク付けする。Unrunnableとしてマーク付けされ得る頂点がなくなるまで、このテストを繰り返す。

40

【0088】

（５）Runnableプロセス頂点を識別すること

Runnableプロセス頂点を識別するために、データフローグラフ内の全てのプロセス頂点について、以下のステップが実行される。「Runnability」テストは、以下のように実行される、すなわち、頂点がEnabled状態にあり、全てのアップストリーム頂点がCompleteファイル頂点又はRunnableプロセス頂点のいずれかである場合、頂点の状態をRunnableに設定する。Runnableとしてマーク付けされ得る頂点がなくなるまで、このテストを繰り返す。

【0089】

（６）残りのEnabled頂点をUnrunnableに設定すること

50

残りの `Enabled` 頂点を `Unrunnable` に設定するために、グラフ内の全てのプロセス頂点について、以下のステップが実行される。頂点が `Enabled` 状態にある場合、その状態を `Unrunnable` に設定する。

【0090】

(7) 更にファイルアダプタを挿入すること

更にファイルアダプタを挿入するために、データフローグラフ内の全てのリンクについて、以下のステップが実行される。リンクのソースが `Runnable` プロセス頂点であり、宛先が `Unrunnable` プロセス頂点である場合、ソースファイルアダプタを挿入する。

【0091】

(8) 通信方法を選択すること

通信方法を選択するために、データフローグラフ内の全てのリンクについて、以下のステップが実行される。このステップは、いずれかの終了時に、実行可能なプロセスに取り付けられ、かつ通信方法にバインドされていないリンクにのみ適用される。リンクのソース（宛先）がファイル頂点であり、その宛先（ソース）が同じノード上のプロセス頂点である場合、リンクの通信方法を `File` に設定する。そうでない場合、利用可能な通信方法のうちの1つを選択し、それにより、その方法の制約の全てが満たされる。速度について、通信方法は、`Shared Memory`、`Named Pipe`、及び `TCP/IP` の順序で考慮され得る。いくつかの例では、上術の制約を満たす第1の方法が選択される。基準サブストレイトでは、以下のルールが使用され得る。まず、リンクが `SOC` 接続を受け入れるポートに取り付けられている場合、リンクは、ソース及び宛先が同じノード上にある場合は `Shared Memory` を使用し、それらが異なるノードにある場合は `TCP/IP` を使用することになる。そうでない場合、ソース及び宛先が同じノード上にある場合、`Named Pipe` 方法が使用されることになる。全ての他の場合において、単一の通信方法では十分ではなく、システムは通信アダプタ（以下）に戻すことになる。

【0092】

(9) 通信アダプタを挿入すること

通信方法を選択する前のステップにおいて単一の通信方法が選択されておらず、全てが試みられた場合、ソース通信アダプタを挿入し、アダプタの2つのリンクの通信方法を選択しようと試みることによって継続する。これが失敗した場合、新たに挿入されたソース通信アダプタを宛先通信アダプタに置換することを試みる。これが失敗した場合、ソース通信アダプタ及び宛先通信アダプタの両方を挿入し、結果として生じる二重アダプタ内の3つのリンクの通信方法を選択する。基準サブストレイトにおいて、通信アダプタが、ソース及び宛先が異なるノード上にあり、リンクが、`SOC` 接続方法を受け入れないファイル頂点又はプロセス頂点のいずれかに接続される場合にのみ必要とされる。この場合、アダプタは以下のように選択され得る。

【0093】

ソースがファイル頂点である場合、ソース通信アダプタを挿入する。ソース通信アダプタ内の2つのリンクは、次に、`File` 及び `TCP/IP` 通信方法を使用することになる。

【0094】

ソースが `SOC` 通信方法を受け入れないポートである場合、ソース通信アダプタを挿入する。ソース通信アダプタ内の2つのリンクは、次に、`TCP/IP` 及び `File` 通信方法を使用する。

【0095】

宛先がファイル頂点である場合、宛先通信アダプタを挿入する。

【0096】

アダプタ内の2つのリンクは、次に、`TCP/IP` 及び `File` 通信方法を使用する。

【0097】

宛先が `SOC` 通信方法を受け入れないポートである場合、宛先通信アダプタを挿入する。アダプタ内の2つのリンクは、次に、`TCP/IP` 及び `Named Pipe` 通信方法を

10

20

30

40

50

使用する。

【0098】

フェーズC：データリンクを起動すること

データリンクは、Unlaunched状態で作成され、起動されなければならない。リンクを起動するために、リンクをスキャンして、Unlaunchedであり、通信方法にバインドされ、Runnableソース又は宛先を有するリンクを見つける。全てのそのようなリンクについて、様々な通信方法によって使用され得る識別子が生成される。上記の拡張サブストレイトの場合、識別子は以下のように作成される。全てのリンクは、2つの識別子、すなわち、ストリームオブジェクト識別子及び通信チャンネル／ファイル識別子を有する。ストリームオブジェクト識別子は、SOCメカニズムによって使用され、リンクの名前と同一である。チャンネル／ファイル識別子は、リンクによって用いられるFile、NamedPipe、SharedMemory区域、又はTCP／IP接続を識別するために使用される。加えて、プロセス頂点がNamedPipe又はFile通信方法を必要とする場合、チャンネル／ファイル識別子は、プロセス頂点が、起動されると、UNIXファイルシステムを使用してチャンネル／ファイルに取り付けることができるように、利用可能にされることになる。

【0099】

識別子が生成された後、チャンネル又はストリームオブジェクトを作成するために、サブストレイトが呼び出される。通信方法がNamedPipeである場合、NamedPipeを作成するために、サブストレイトがまたも呼び出される。

【0100】

実行可能プログラム182が生成されると、コンパイラ24は実行可能プログラム182をデータ処理システム26に送信する。データ処理システム26は、記憶システム12からレコードを受信し、実行可能プログラム136をバッチモードで実行して、例えば、バッチ結果184を作り出す。バッチ結果184は、特定のルールケース「始動」（例えば、処理されたデータレコードによってルールが何回トリガされたか）の回数を示す。この例では、「ゴールド」オファーは、他のルールと比較して、途方もない時間量をトリガした。そのため、ユーザは、例えば、ゴールドオファーがトリガされる時間数を減少させるために変更を行うことができるかどうかを決定するために、自分が作成したルールをテストすることを望む場合がある。

【0101】

図5Kを参照すると、環境190は、ビジネスルール及び他の論理ルールを定義及びテストするために開発環境18によって生成されたビジネスルールエディタ及びテストインターフェース191の例を示している。この例では、インターフェース191は、レコード192aによるテスト、式192bによってテスト、エラー192cによってテスト、ベースラインデルタ192dによってテストし、ルールケース192eによるテストを含む、様々なテストカテゴリ192を可能にする。この例では、開発環境18のユーザは、ルールケース192eによってテストするように選択され、ケース2（ゴルフオファーに対応する）を指定している。ここから、ユーザは、図5Lに示されるように、ボタン193aと対話することによって、指定されたルールケース（すなわち、ルールケース2）をトリガしたレコード193をステップスルーすることができる。この例では、インターフェース191のルール定義部分123において影付き塗りつぶしの太い輪郭によって示されるように、レコード4が、ルールケース2をトリガしている。データフロログラフが実行された（これにより、物理データがアクセスされた）ため、入力部分122において示されるフィールドは、現在のレコード（図5Lのレコード4）のデータ値194が入力される。これらの値から分かるように、レコード4は、トリガケース2の定義されたルール内に十分に含まれる。図5Mに示されるように、ボタン193aと対話することにより、ケース2をトリガした次のレコードに進む。データ値194から、レコード24がルールケース2の「使用された分数」閾値に有意に近いことが分かる。

【0102】

そのため、ゴールドオファァの数を減少させるために、ユーザは、図 5 N に示されるように「使用された分数」閾値を増加させ得る。この例では、ユーザは、太枠の網掛けセル 195 に示されるように、ルールケース 2 における「使用された分数」のトリガ値を「> 400」に変更している。変更の全体的な結果を見るために別のバッチテストを実行する前に、ユーザは、ルールが予想通りに機能していることを確実にするために、個々のレコード又は少数のレコードに対するルール変更を望む場合がある。そうするために、ユーザは、テストされるレコードをウィンドウ 193 に入力し、「テストレコード」ボタン 196 と対話することができる。この例では、ユーザは、レコード 24 をテストするように選択している。「テストレコード」ボタン 196 を選択したことに応答して、開発環境 18 は、修正された仕様 197 を生成し、仕様 197 をグラフジェネレータ 22 に送信する。開発環境 18 は、仕様 197 全体又は修正された部分のみを伝送することができる。その修正された仕様 197 及び論理データ（図示せず）を使用して、グラフジェネレータ 22 は、修正されたデータフローグラフ 198 を作り出した。データフローグラフ 198 は、コンパイル及びその後の実行のためにコンパイラに送信される。開発環境 18 はまた、テストされるレコードを指定するデータ 199 を、実行のためにデータ処理システム 26 に送信する。

【0103】

図 5 O を参照すると、環境 200 は、指定されたテストレコード 204 に対するデータフローグラフ 198 の実行を示している。この例では、コンパイラ 24 は、データフローグラフ 198 を受信し、それをコンパイルして実行可能プログラム 202 を作り出し、実行可能プログラム 202 は、データ処理システム 26 に送信される。データ処理システム 26 は、テストされるレコードを指定するデータ 199 を受信し、記憶システム 12 から指定されたテストレコード 204（例えば、この例ではレコード 24）を取り出す。次いで、データ処理システム 26 は、テストレコード 204 を使用して実行可能プログラム 202 を実行して、更新された結果 206、すなわち、指定されたレコードを更新された実行可能データフローグラフで処理した結果を作り出す。これらの結果は、図 5 P に示されるように、インターフェース 191 においてテストを実行したユーザに提示される。図 5 P に見られるように、レコード 24 は、修正されたルールの下でケース 2（ゴールドオファァを表す）ではなく、ケース 3（シルバーオファァを表す）をトリガする。

【0104】

修正されたルールケース及びデータフローグラフが意図されるように機能していることを確認すると、図 5 Q に示されるように、バッチテストを実行することができる。そうするために、データ処理システム 26 は、記憶システム 12 からレコードを受信し、バッチモードで実行可能プログラム 202 を実行してバッチ結果 208 を作り出す。バッチ結果 208 は、ゴールドオファァの数が、図 5 J に示される修正前のバッチ結果 184 と比較して著しく減少していることを示している。

【0105】

図 6 A を参照すると、環境 210 は、ビジネスルール及び他の論理ルールを定義するために開発環境 18 によって生成されたビジネスルールエディタインターフェース 121 の別の例を示している。この例では、論理データ 211 は、「取引先データ」データセットをルートノードとして含み、「取引先データ」データセットは、「取引」データセット、「支払データ」データセット、及び「出金データ」データセットを含む様々な他のデータセットに関連している。これらのデータセットの各々及びそれらのそれぞれのフィールドは、インターフェース 121 の入力部分 122 において視覚化される。具体的には、入力部分 122 は、「取引先データ」及びそのフィールドの視覚表現 212、212 a、212 b、及び 212 c、「取引」及びそのフィールドの視覚表現 213、213 a、及び 213 b、「支払データ」及びそのフィールドの視覚表現 214 及び 214 a、並びに「出金データ」及びそのフィールドの視覚表現 215 及び 215 a を含む。

【0106】

ルール定義部分 123 は、一連の入力及びルールケースを含む。この例では、それぞれ

、「価格」フィールド及び「場所」フィールドは、セル 1 2 8 a 及び 1 2 8 b に示されるように、ルールを定義する際の入力として使用される。「取引先場所」及び「取引先口座残高」フィールドは、ルール定義部分 1 2 0 において指定されたルールケースを定義する際の式の一部として使用される。ルールケースが適用される場合、出力列 1 2 9 a に基づいて出力が生成される。この列に示されるように、ルールケース 1 2 9 c、1 2 9 d、1 2 9 e の各々の出力は、指定されたトリガ基準に基づいて、レビューのための特定の取引の承認、否認、又はフラグ付けに関連する。ルール定義部分 1 2 3 においてセルの入力を指定することによってルールの定義を完了すると、開発環境 1 8 は、ルールケースと、ルールを実装するためにどのフィールドにアクセスする必要があるかを指定するルール仕様 2 1 6 を生成する。開発環境 1 8 は、ルール仕様 2 1 6 をグラフジェネレータ 2 2 に送信する。開発環境 1 8 はまた、論理データ 2 1 1 をグラフジェネレータ 2 2 に送信する。

【0 1 0 7】

図 6 B を参照すると、環境 2 2 0 は、ルール仕様 2 1 6 及び論理データ 2 1 1 から連続動作のために構成されたデータフローグラフを生成及び最適化する例を示している。グラフジェネレータ 2 2 は、ルール仕様 2 1 6 及び論理データ 2 1 1 を受信する。バッチ又は非連続設定と同様に、グラフジェネレータ 2 2 は、初期に、例えば、論理データ 2 1 1 において指定されたデータセットにアクセスするための命令、パラメータ、及び他の情報に基づいて、データソースとして論理データ 2 1 1 に含まれるデータセット及びフィールドにアクセスするように構成されたデータフローグラフ 2 2 2 を生成する。しかしながら、データフローグラフ 2 2 2 のコンポーネント及びデータがアクセスされ、処理される様式は、連続設定では異なる。この例では、サブスクライブコンポーネント 2 2 2 a を使用して、ルートノードである「取引先データ」からのデータのフローをサブスクライブしている。次いで、ルートノードからの各着信フローユニット（又はその一部分）は、例えば、ルックアップコンポーネント 2 2 2 c を使用して、論理データ 2 1 1 において定義された関連レコードの後続のルックアップに使用するために、複製コンポーネント 2 2 2 b を通して複製される。

【0 1 0 8】

初期データフローグラフ 2 2 2 を生成した後、グラフジェネレータ 2 2 は、データフローグラフ 2 2 4 を生成するために、オブティマイザ 1 3 2 をデータフローグラフ 2 2 2 に適用する。オブティマイザ 1 3 2 は、データフローグラフ 2 2 4 のクロスアウト部分によって示されるように、データフローグラフ 2 2 2 からコンポーネント 2 2 2 d、2 2 2 f、2 2 2 g、2 2 2 h、及び 2 2 2 i を除去する。オブティマイザ 1 3 2 は、これらのコンポーネントがルール仕様 2 1 6 によって参照又は使用されないデータセットに関連するため、これらのコンポーネントを除去することを決定する。すなわち、ルール仕様 2 1 6 は、除去されたデータセットに含まれるいずれのフィールドへの参照も含まない。最適化の最終結果は、ルール仕様 9 6 a によって指定されたルールを実行するために必要とされないデータセットの全て、並びにそれらのデータセットにアクセスするためにインスタンス化された他のコンポーネント（例えば、ソート、結合など）を除去するように最適化されたデータフローグラフ 2 2 6 である。これにより、本明細書に記載の論理データは、入力データが連続的、半連続的、又は非連続的であるかどうかにかかわらず、物理的負担を伴わずに論理アクセスを提供し、最適化を容易にするのに有効である。

【0 1 0 9】

図 6 C を参照すると、環境 2 3 0 は、連続データフローグラフ 2 2 6 の実行の結果を示している。グラフ生成システム 1 8 は、データフローグラフ 2 2 6 をコンパイラ 2 4 に送信し、コンパイラ 2 4 は、データフローグラフ 9 6 a を実行可能プログラム 2 3 2（例えば、実行可能なデータフローグラフ）にコンパイルする。コンパイラ 2 3 は、実行可能プログラム 2 3 2 をデータ処理システム 2 6 に送信する。データ処理システム 2 6 は、データストリーム 1 2（例えば、連続データ）を受信し、実行可能プログラム 2 3 2 を実行して、データストリームを処理し、リアルタイム又はほぼリアルタイムの結果 2 3 4 を作り出す。

【0110】

図7Aを参照すると、環境240は、仕様252を生成する開発環境18の別の実世界の例を示している。この例では、開発環境18は、コンポーネント部分242、入力部分243、及びキャンバス部分244を有するグラフィカルユーザインターフェース241を描画している。コンポーネント部分242は、計算論理を定義するために利用可能な様々な動作を表す視覚表現242a～242fを含む。入力部分243は、論理データ94において表されるデータセット及びフィールドの視覚表現245、245a、246、246a、246b、246c、247、247a、248、248aを表示している。入力部分243はまた、他のデータソースにおいて表されるデータセット及びフィールドの視覚表現249及び249aを表示している。すなわち、入力部分243における視覚表現は、計算論理を定義するために利用可能なデータセット及びフィールドを表す。

10

【0111】

キャンバス部分244は、データフローグラフの形態で計算論理を定義するために使用され、視覚化250として視覚的に描かれる（以下、便宜上、及び限定することなく、「データフローグラフ250」と称される）。視覚化250によって表されるデータフローグラフは、ノードを有するデータ構造を含む。ノードの各々は、少なくとも1つの動作ブレースホルダフィールド及び少なくとも1つのデータブレースホルダフィールドを含み、これらは、キャンバス部分244においてユーザによって指定された動作及びデータが入力される。この例では、データフローグラフ250は、視覚表現242a～242fのうちの1つ以上を、コンポーネント部分242からキャンバス部分244上にドラッグ及びドロップすることによって生成される。視覚表現242a～242fの各々は、データ構造によって又はデータ構造に対して実行される動作を表す。視覚表現がキャンバス部分244上に配置されると、それらは、キャンバス部分244上のアイコンになる。アイコン251aなどのこれらのアイコンのうちのいくつかは、特定のデータセット又はフィールドに関して実行する動作（例えば、フィルタ動作）を指定する。この例では、アイコン251aは、入力部分243において視覚表現246aによって表される「追加された分数」に対してフィルタ動作が実行されることを指定する。アイコン251bは、論理データ部分243において視覚表現246cによって表される「残りの分数」フィールドに対してフィルタ動作が実行されることを指定する。開発環境18は、データフローグラフ250によって視覚的に表される計算論理を使用して、仕様252を生成する。仕様252は、キャンバス部分244において視覚的に描かれた計算論理を指定する。開発環境18は、仕様252及び論理データ94をグラフジェネレータ22に送信する。グラフジェネレータ22は、仕様252及び論理データ94を使用して、データフローグラフ250の各ノードの動作フィールド及びデータブレースホルダフィールドにデータ入力することができる。

20

30

【0112】

図7Bを参照すると、環境260は、最適化されたデータフローグラフを生成するグラフジェネレータ22の例を例示しており、その視覚化は、視覚化268（本明細書では、便宜上、及び限定することなく、「データフローグラフ268」と称される）によって示される。グラフジェネレータ22は、仕様252及び論理データ94を受信する。その仕様252及び論理データ94を使用して、グラフジェネレータ22は、図7Cに示されるように、コンポーネント262a～262rを含むデータフローグラフ262を生成する。特に、グラフジェネレータ22は、データフローグラフ252の各ノードについて動作フィールド及びデータブレースホルダフィールドにデータ入力し、かつ前述の技術を使用することによって、仕様252及び論理データ94からデータフローグラフ262を生成する。例えば、指定された計算論理が変換コンポーネント134lによって実装されるデータフローグラフ98aとは異なり、データフローグラフ262は、仕様252において指定された計算論理に基づいて別個のコンポーネント262o、262p、262qを含む。データフローグラフ262は、論理データ94において表され、別個の「オフアー」データセット249及びその「月次」フィールド249aと結合されるデータセットを表

40

50

し、データフローグラフ 262 はまた、データフローグラフ（例えば、ソート、パーティションなど）を生成するために必要な追加の組み込み機能を表す。

【0113】

この例では、グラフジェネレータ 22 は、最適化されたデータフローグラフ 268 を作り出すために、オブティマイザ 132 を図 7C に示されるデータフローグラフ 262 に適用する。最適化の様々な中間段階が図 7D 及び図 7E に示されている。オブティマイザ 132 は、仕様 252 若しくは論理データ 94、又はその両方を分析して、仕様 252 において使用されるフィールドを識別し、次に、それらのフィールドを含むデータセットを識別する。オブティマイザ 132 は、データフローグラフ 262 から、仕様 252 によって使用又は参照されないデータセットを除去する。オブティマイザ 132 はまた、必要に応じてグラフにパーティションコンポーネントを追加することを担うことができる。いくつかの例では、オブティマイザ 132 は、ルール仕様 252 において指定され、論理データ 94 に含まれるデータセット及びフィールドのみがアクセスされるように、選択ステートメントを最小限に抑えることによってこれを行う。図 7D に示されるように、オブティマイザ 132 は、データフローグラフ 262 からコンポーネント 262a、262s、262c、262f、262i、262v、及び 262h を除去する（それにより、時間 T2 においてデータフローグラフ 264 を作り出す）。これは、コンポーネント 262a がデータセット「オファー状況」を表し、そのフィールド「オファー受入」が、仕様 252 によって参照又は使用されないためである。同様に、コンポーネント 262c は、データセット「リロード日」を表し、そのフィールド「最終リロード」は、仕様によって参照又は使用されない。これらの入力ソース（すなわち、コンポーネント 262a 及び 262i によって表されるもの）を除去することにより、残りのコンポーネントを不要なもの（「デッドコンポーネント」と称されることもある）として描写し、したがって、これらのコンポーネント（すなわち、262s、262c、262v、262h）も除去することができる。

【0114】

オブティマイザ 132 はまた、コンポーネント 262k によって指定された結合動作の前にフィルタコンポーネント 262o 及び 262p を移動させる更なる最適化を実行し、それによって、図 7E に示されるように、時間 T3 においてデータフローグラフ 266 を作り出す。そうすることによって、フィルタ動作が結合動作の前に実行されて、結合される必要のあるデータの量を低減するため、オブティマイザ 122 は、より高速で、より効率的で、より少ない計算リソースを使用するデータフローグラフを作り出す。フィルタ動作が結合動作の後に実行される場合、システムが最終的にフィルタ除去されるデータを結合しなければならないため、より多くの組成リソースが使用される。最適化の結果は、データフローグラフ 268 である。

【0115】

概して、オブティマイザ 132 は、データフローグラフにおいて指定された動作のうちの 1 つ以上に従ってデータを処理するために必要とされ得る最適化若しくは他の変換を実行するか、又は最適化若しくは変換、又はその両方を伴わずにデータを処理することと比較して、データフローグラフにおいて指定された動作のうちの 1 つ以上に従ってデータを処理することを改善する。例えば、オブティマイザは、データフローグラフ 262 の所望の機能を有する変換されたデータフローグラフ 268 を作り出すために、とりわけ、1 つ以上のソート動作、データタイプ動作、データフローグラフにおいて指定されたキーに基づく結合動作を含む結合動作、パーティション動作、自動並列処理動作、又はメタデータを指定する動作を追加する。いくつかの実装形態では、変換されたデータフローグラフ 268s は、最適化を適用する前の変換されたデータフローグラフの計算効率と比較して、変換されたデータフローグラフの計算効率を改善するために、1 つ以上のデータフローグラフ最適化ルールを変換されたデータフローグラフに適用することによって最適化されたデータフローグラフである（又は最適化されたデータフローグラフに変換される）。データフローグラフ最適化ルールは、例えば、「Editor for Generating

10

20

30

40

50

Computational Graphs」と題された米国特許出願第62／966，768号に記載されるように、とりわけ、デッドコンポーネント又は冗長コンポーネントの削除、早期フィルタリング、又はレコードの絞り込みを含むことができ、その全容は参照により本明細書に組み込まれる。

【0116】

本明細書に記載の技術は、開発環境を使用してユーザ（例えば、ビジネスユーザ）の生産性を改善し、最適化されたデータ処理を可能にするために、データセット間の関係についての情報を使用する。ユーザ（例えば、技術ユーザ）は、初期に、開発環境に公開する論理データを（例えば、ルートノードとして使用するためにデータセットを選択するか、又は仮想フィールドを定義することによって）定義する必要があるが、ビジネスユーザは、公開された論理データから自身の計算論理を柔軟に開発する権限を与えられ、その論理に基づいて、最適化された様式で論理を実行するために、多種多様なデータフローグラフを生成することができる。

【0117】

本明細書に記載される技術は、ユーザが、記憶システムに記憶されたデータセットの複雑なセットから、迅速かつ強力に、論理データを開発環境に公開する権能を与えるものである。いくつかの例では、技術ユーザは、作業に関心がある一組のデータセットを選択し、これらのデータセットの全ての中からスキーマ定義が発見されるか、さもなければ取得される。例えば、スキーマは、データベース内にあるこれらのデータセットからエクスポートされ、データ発見、セマンティック発見、若しくは他の機械学習を使用して、又は技術ユーザからの追加の入力を受信することによって、又はそれらの組み合わせによって発見することができる。いくつかの例では、技術ユーザは、他のデータ要素の中からのアグリゲーションなど、追加の計算フィールド又は仮想フィールドをスキーマ内に生成することができる。いくつかの例では、技術ユーザは、論理データのルートノード又はパースペクティブを選択することができる。

【0118】

次いで、開発環境において動作するビジネスユーザは、論理データ（実際の物理データ要素又は技術ユーザが定義した論理データ要素に対応し得る）に含まれる属性のいずれかを使用して、自分のビジネスニーズに適用可能な計算論理を開発することができる。いくつかの例では、ビジネスユーザは、出力を視認し、開発環境で書いた論理（例えば、ルール）をテストすることができる。

【0119】

ビジネスユーザが、自分が開発した（及び任意選択的にテストされた）計算論理に満足すると、最適化されたデータフローグラフは、そのデータフローグラフに必要とされるデータセットのみを処理するグラフジェネレータによって生成することができる。例えば、ビジネスユーザは、計算論理を開発するときに、不要であることが判明した多数のデータセットにアクセスし得る。グラフジェネレータ及びオプティマイザが論理データからのデータセットについて詳細な情報を有するため、データセットが生成するデータフローグラフは、劇的に最適化することができる。

【0120】

最適化されたデータフローグラフが生成されると、最適化されたデータフローグラフは、例えば、データ処理システムによって実行することができる。いくつかの例では、データフローグラフは、2つの異なるモード、すなわち、バッチ又はリアルタイムで実行することができる。いくつかの例では、ビジネスユーザが異なるセットのデータに依存する異なるセットのルールに関心があった場合、ビジネスユーザは、所望のデータフローグラフを生成することができ、そのデータフローグラフも、技術ユーザが関与する必要なく最適化することができる。

【0121】

図8は、論理データを作り出し、論理データを使用してコンピュータプログラムを生成するための例示的なプロセス800のフローチャートを例示している。プロセス800は

、図1～図7を参照して記載された技術を実装するように構成されたコンピューティングシステムのうちの1つ以上を含む、本明細書に記載のシステム及びコンポーネントのうちの1つ以上によって実装することができる。

【0122】

プロセス800の動作は、スキーマにおいて表されるデータセット間の関係を指定するスキーマ、1つ以上のデータセットに対する1つ以上の計算、又は1つ以上のデータセットに対する1つ以上の変換にアクセスすること(802)を含む。例では、スキーマは、データベーススキーマである。例では、データセットのうちの1つ以上に対する1つ以上の計算又はデータセットの1つ以上の1つ以上の変換は、複数のデータセットのうちの少なくとも1つの論理フィールド、仮想フィールド、又は計算フィールドを定義する。

10

【0123】

記憶装置内のデータセットの中からの複数のデータセットは、データセットの中からデータセットを選択することと、スキーマから、選択されたデータセットに関連する1つ以上の他のデータセットを識別することと、によって識別される(802)。例では、選択されたデータセットは、論理データのルートノードであり、1つ以上の他のデータセットのうちの少なくとも1つは、選択されたデータセットに結合される。例では、選択されたデータセットを指定する選択データは、クライアントデバイスから受信される。例では、選択されたデータセットと1つ以上の他のデータセットとを結合するための、1つ以上のキーなどの1つ以上のパラメータが、スキーマから識別される。

20

【0124】

複数のデータセットの属性が識別される(806)。例では、1つ以上の属性は、複数のデータセットのフィールド名を含む。例では、1つ以上の属性は、複数のデータセットにアクセスするための情報を含む。複数のデータセットの識別された属性を表し、属性間の1つ以上の関係を更に表す論理データが生成される(808)。

【0125】

論理データは、開発環境に提供される(810)。開発環境は、複数のデータセットの識別された属性を表す論理データのうちの1つ以上の部分へのアクセスを提供する(812)。例では、開発環境は、複数のデータセットに記憶装置からアクセスすることなく、論理データのうちの1つ以上の部分へのアクセスを提供する。例では、開発環境は、論理データをデータソースとして読み取る。

30

【0126】

動作を実行するときに、識別された属性のうちの少なくとも1つを指定する仕様が、開発環境から受信される(814)。仕様と、論理データによって表される識別された属性間の1つ以上の関係に基づいて、複数からの少なくとも1つのデータセットに記憶装置からアクセスすることによって動作を実行するように構成されているコンピュータプログラムが生成され(816)、アクセスされた少なくとも1つのデータセットが、仕様において指定された属性のうちの少なくとも1つを有する。例では、コンピュータプログラムは、記憶装置からアクセスされた少なくとも1つのデータセットを使用して実行される。例では、動作は、仕様において指定された属性のうちの少なくとも1つを含む複数のデータセットから1つのデータセットを識別することと、識別されたデータセットに記憶装置からアクセスすることと、を含む。

40

【0127】

例では、コンピュータプログラムは、仕様において指定された属性のうちの少なくとも1つを有する複数のデータセット内のデータセットのみに記憶装置からアクセスすることによって動作を実行するように構成されている最適化されたコンピュータプログラムを作り出すように最適化される。例では、仕様において指定された属性のうちの少なくとも1つを含まない複数のデータセット内の少なくとも1つのデータセットに記憶装置からアクセスする動作は、コンピュータプログラムから除去される。例では、コンピュータプログラムは、選択ステートメントによって複数からの少なくともいくつかのデータに記憶装置からアクセスするように構成され、選択ステートメントは、仕様において指定された属性

50

のうちの少なくとも1つのみを選択するように最小化される。例では、動作は、仕様と、論理データによって表される識別された属性間の1つ以上の関係に基づいて、動作を実行するように構成されている実行可能なデータフローグラフを生成することを含み、実行可能なデータフローグラフは、1つ以上の属性のうちの少なくとも1つを入力として含む。

【0128】

本明細書に記載の主題及び動作の実装形態は、本明細書に開示された構造及びそれらの構造的等価物を含むデジタル電子回路、又はコンピュータソフトウェア、ファームウェア、若しくはハードウェア、又はそれらのうちの1つ以上の組み合わせで実装することができる。本明細書に記載の主題の実装形態は、1つ以上のコンピュータプログラム（データ処理プログラムとも称される）（すなわち、データ処理装置によって実行するために、又はデータ処理装置の動作を制御するためにコンピュータ記憶媒体上に符号化されたコンピュータプログラム命令の1つ以上のモジュール）として実装することができる。コンピュータ記憶媒体は、コンピュータ可読記憶デバイス、コンピュータ可読記憶基板、ランダムアクセス若しくはシリアルアクセスメモリアレイ若しくはデバイス、又はそれらのうちの1つ以上の組み合わせであってもよく、又はそれらに含まれてもよい。コンピュータ記憶媒体はまた、1つ以上の別個の物理的コンポーネント又は媒体（例えば、複数のCD、ディスク、又は他の記憶デバイス）であってもよく、又はそれらに含まれてもよい。本主題は、非一時的コンピュータ記憶媒体に記憶されたコンピュータプログラム命令に実装され得る。

【0129】

本明細書に記載の動作は、1つ以上のコンピュータ可読記憶デバイス上に記憶された、又は他のソースから受信されたデータに対してデータ処理装置によって実行される動作として実装することができる。

【0130】

「データ処理装置」という用語は、データを処理するためのあらゆる種類の装置、デバイス、及びマシンを包含し、例として、プログラマブルプロセッサ、コンピュータ、システムオンチップ、又は前述のものの複数のもの、又は組み合わせを含む。装置は、専用論理回路（例えばFPGA（Field Programmable Gate array、フィールドプログラマブルゲートアレイ）又はASIC（Application Specific Integrated Circuit、特定用途向け集積回路）を含むことができる。装置はまた、ハードウェアに加えて、問題のコンピュータプログラムの実行環境を提供するコード（例えば、プロセッサファームウェア、プロトコルスタック、データベース管理システム、オペレーティングシステム、クロスプラットフォームランタイム環境、仮想マシン、又はそれらのうちの1つ以上の組み合わせを構成するコード）を含むことができる。装置及び実行環境は、ウェブサービス、分散コンピューティング、及びグリッドコンピューティングインフラストラクチャなどの様々な異なるコンピューティングモデルインフラストラクチャを実現することができる。

【0131】

コンピュータプログラム（プログラム、ソフトウェア、ソフトウェアアプリケーション、スクリプト、又はコードとしても知られている）は、コンパイル言語若しくはインタープリタ言語、宣言型言語若しくは手続き型言語を含む任意の形式のプログラミング言語で書くことができ、スタンドアロンプログラムとして、又はモジュール、コンポーネント、サブルーチン、オブジェクト、若しくはコンピューティング環境での使用に好適な他のユニットとして含む、任意の形態で展開することができる。コンピュータプログラムは、ファイルシステム内のファイルに対応してもよいが、対応する必要がない場合がある。プログラムは、他のプログラム又はデータ（例えば、マークアップ言語文書に記憶された1つ以上のスクリプト）を保持するファイルの一部分、問題のプログラム専用の単一のファイル、又は複数の協調ファイル（例えば、1つ以上のモジュール、サブプログラム、又はコードの一部分を記憶するファイル）に記憶することができる。コンピュータプログラムは、1つのコンピュータ上で、又は1つのサイトに位置するか、又は複数のサイトに分散さ

れ、通信ネットワークによって相互接続された複数のコンピュータ上で実行されるように展開することができる。

【0132】

本明細書に記載のプロセス及び論理フローは、入力データに対して動作し、出力を生成することによってアクションを実行するために、1つ以上のコンピュータプログラムを実行する1つ以上のプログラマブルプロセッサによって実行することができる。プロセス及び論理フローはまた、専用論理回路（例えば、FPGA（フィールドプログラマブルゲートアレイ）又はASIC（特定用途向け集積回路）によって実行することができ、装置は専用論理回路として実装することができる。

【0133】

コンピュータプログラムの実行に好適なプロセッサは、例として、汎用マイクロプロセッサ及び専用マイクロプロセッサの両方、並びに任意の種類のデジタルコンピュータの任意の1つ以上のプロセッサを含む。概して、プロセッサは、リードオンリーメモリ若しくはランダムアクセスメモリ又はその両方から命令及びデータを受信するであろう。コンピュータの必須要素は、命令に従ってアクションを実行するためのプロセッサと、命令及びデータを記憶するための1つ以上のメモリデバイスとを含む。一般に、コンピュータはまた、データを記憶するための1つ以上の大容量記憶デバイス（例えば、磁気ディスク、光磁気ディスク、又は光ディスク）を含むか、それらからデータを受信するか、それらにデータを転送するか、又はその両方を行うように動作可能に結合することになるが、コンピュータは、そのようなデバイスを有する必要はない。更に、コンピュータは、別のデバイス（例えば、携帯電話、携帯情報端末（Personal Digital Assistant、PDA）、モバイルオーディオ若しくはビデオプレーヤ、ゲームコンソール、全地球測位システム（Global Positioning System、GPS）受信機、又は携帯型記憶デバイス（例えば、ユニバーサルシリアルバス（Universal Serial Bus、USB）フラッシュドライブ））に組み込むことができる。コンピュータプログラム命令及びデータを記憶するのに好適なデバイスは、例として、半導体メモリデバイス（例えば、EPROM、EEPROM、及びフラッシュメモリデバイス）、磁気ディスク（例えば、内蔵ハードディスク又はリムーバブルディスク）、光磁気ディスク、並びにCD-ROM及びDVD-ROMディスクを含む、あらゆる形態の不揮発性メモリ、媒体及びメモリデバイスを含む。プロセッサ及びメモリは、専用論理回路によって補完され得るか、又は専用論理回路に組み込まれ得る。

【0134】

本明細書に記載の主題の実装形態は、バックエンドコンポーネント（例えば、データサーバとして）を含むか、又はミドルウェアコンポーネント（例えば、アプリケーションサーバ）を含むか、又はフロントエンドコンポーネント（例えば、ユーザが本明細書に記載の主題の実装形態と対話することができるグラフィカルユーザインターフェース又はWebブラウザを有するユーザコンピュータ）、又は1つ以上のそのようなバックエンドコンポーネント、ミドルウェアコンポーネント、若しくはフロントエンドコンポーネントの任意の組み合わせを含む、コンピューティングシステムに実装することができる。システムのコンポーネントは、デジタルデータ通信（例えば、通信ネットワーク）の任意の形態又は媒体によって相互接続することができる。通信ネットワークの例には、ローカルエリアネットワーク（Local Area Network、LAN）及びワイドエリアネットワーク（Wide Area Network、WAN）、インターネットワーク（例えば、インターネット）、及びピアツーピアネットワーク（例えば、アドホックピアツーピアネットワーク）が含まれる。

【0135】

コンピューティングシステムは、ユーザ及びサーバを含むことができる。ユーザ及びサーバは、一般に、互いに遠隔であり、典型的には、通信ネットワークを通して対話する。クライアントとサーバとの関係は、それぞれのコンピュータ上で実行され、互いにクライアント-サーバ関係を有するコンピュータプログラムによって生じ得る。いくつかの実装形態では、サーバは、データ（例えば、HTMLページ）をユーザデバイスに送信する（例えば、ユーザデバイスと対話するユーザにデータを表示し、ユーザデバイスと対話する

ユーザからユーザ入力を受信する目的で)。ユーザデバイスにおいて生成されたデータ（例えば、ユーザ対話の結果）は、サーバにおいてユーザデバイスから受信することができる。

【0136】

本明細書は多くの特定の実装形態の詳細を含むが、これらは、任意の実装形態又は特許請求され得るものの範囲に対する限定として解釈されるべきではなく、特定の实装形態に特有の特徴の記載として解釈されるべきである。別個の実施態様の文脈において本明細書に記載されるある特定の特征はまた、単一の実装形態において組み合わせて実装することができる。逆に、単一の実施態様の文脈において記載される様々な特徴はまた、複数の実装形態において別々に、又は任意の好適な二次組み合わせにおいて実装することができる。更に、特徴は、ある特定の組み合わせにおいて作用するものとして上述され、初めにそのように特許請求され得るが、特許請求される組み合わせからの1つ以上の特徴は、場合によっては、組み合わせから切除することができ、特許請求される組み合わせは、二次組み合わせ又は二次組み合わせの変形を対象とし得る。

10

【0137】

同様に、動作は、特定の順序で図面に描かれているが、望ましい結果を達成するために、そのような動作が示された特定の順序で又は連続した順序で実行されるか、又は全ての例示された動作が実行されることを必要とするものとして理解されるべきではない。ある特定の状況では、マルチタスク処理及び並列処理が有利であり得る。更に、上記の実装態様における様々なシステムコンポーネントの分離は、全ての実装形態においてそのような分離を必要とするものとして理解されるべきではなく、記載されたプログラムコンポーネント及びシステムは、一般に単一のソフトウェア製品に共に統合されるか、又は複数のソフトウェア製品にパッケージ化され得ることを理解されたい。

20

【0138】

他の実装形態は、以下の特許請求の範囲の範囲内である。

【符号の説明】

【0139】

2 スキーマ

10 環境

12 記憶システム

14 クライアントデバイス

16 論理データジェネレータ

18 開発環境

20 環境

22 グラフジェネレータ

24 コンパイラ

26 データ処理システム

27 グラフィカルユーザインターフェース

30 環境

32 エディタインターフェース

33 入力部分

34 ルール定義部分

40 環境

42 コンポーネント部分

43 入力部分

44 キャンバス部分

50 環境

52 データフローグラフ

54 実行可能プログラム

60 環境

30

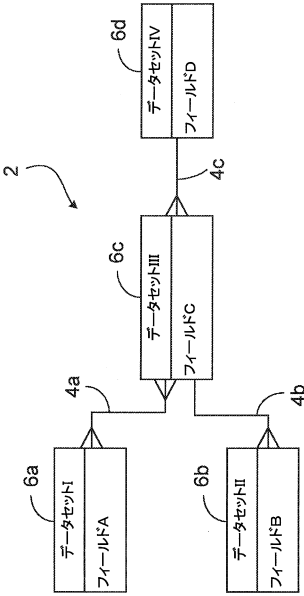
40

50

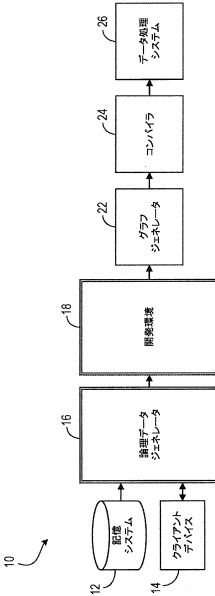
7 0	環境	
7 2	論理データ	
8 0	環境	
8 2	論理データ	
8 4	スキーマ	
9 0	環境	
9 1	スキーマ	
9 2	選択データ	
9 4	論理データ	
9 6 a	仕様	10
9 6 b	仕様	
9 8 a	データフローグラフ	
9 8 b	データフローグラフ	
1 0 0	環境	
1 0 1	データセット	
1 0 2	データセット	
1 0 3	データセット	
1 0 4	データセット	
1 0 5	グラフィカルユーザインターフェース	
1 0 6	データセット部分	20
1 0 7	パースペクティブ部分	
1 1 0	環境	
1 2 0	環境	
1 2 1	インターフェース	
1 2 2	入力部分	
1 2 3	ルール定義	
1 2 4	視覚表現	
1 2 5	視覚表現	
1 2 6	視覚表現	
1 2 7	視覚表現	30
1 3 0	環境	
1 3 2	オブティマイザ	
1 3 4	データフローグラフ	
1 3 6	データフローグラフ	
1 4 0	環境	
1 4 2	仕様	
1 5 0	環境	
1 5 2	データフローグラフ	
1 5 4	データフローグラフ	
1 5 6	データフローグラフ	40
1 6 0	環境	
1 7 0	環境	
1 7 2	データフローグラフ	
1 7 4	データフローグラフ	
1 8 0	環境	
1 8 2	実行可能プログラム	
1 8 4	バッチ結果	
1 9 0	環境	
1 9 1	インターフェース	
1 9 2	テストカテゴリ	50

1 9 3	ウィンドウ	
1 9 4	データ値	
1 9 5	太枠の網掛けセル	
1 9 6	ボタン	
1 9 7	仕様	
1 9 8	データフローグラフ	
1 9 9	データ	
2 0 0	環境	
2 0 2	実行可能プログラム	
2 0 4	テストレコード	10
2 0 6	結果	
2 0 8	バッチ結果	
2 1 0	環境	
2 1 1	論理データ	
2 1 3	視覚表現	
2 1 4	及びそのフィールドの視覚表現	
2 1 5	及びそのフィールドの視覚表現	
2 1 6	ルール仕様	
2 2 0	環境	
2 2 2	データフローグラフ	20
2 2 4	データフローグラフ	
2 2 6	データフローグラフ	
2 3 0	環境	
2 3 2	実行可能プログラム	
2 3 4	ほぼリアルタイムの結果	
2 4 0	環境	
2 4 1	グラフィカルユーザインターフェース	
2 4 2	コンポーネント部分	
2 4 3	論理データ部分	
2 4 3	入力部分	30
2 4 4	キャンバス部分	
2 4 5	視覚表現	
2 4 6	視覚表現	
2 4 7	視覚表現	
2 4 8	視覚表現	
2 4 9	視覚表現	
2 5 2	仕様	
2 6 0	環境	
2 6 2	データフローグラフ	
2 6 4	データフローグラフ	40
2 6 6	データフローグラフ	
2 6 8	データフローグラフ	
3 0 0	スイムレーン図	
8 0 0	プロセス	

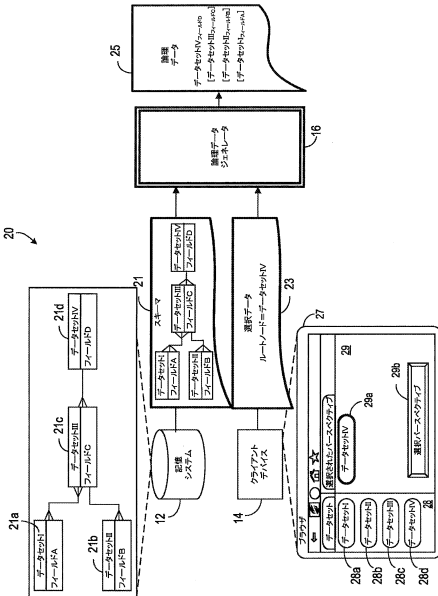
【図面】
【図 1】



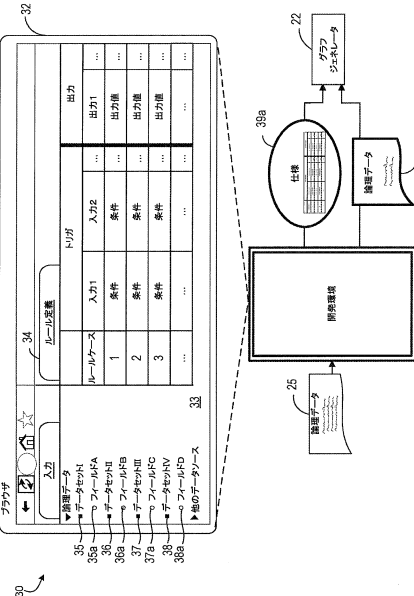
【図 2 A】



【図 2 B】



【図 2 C】



10

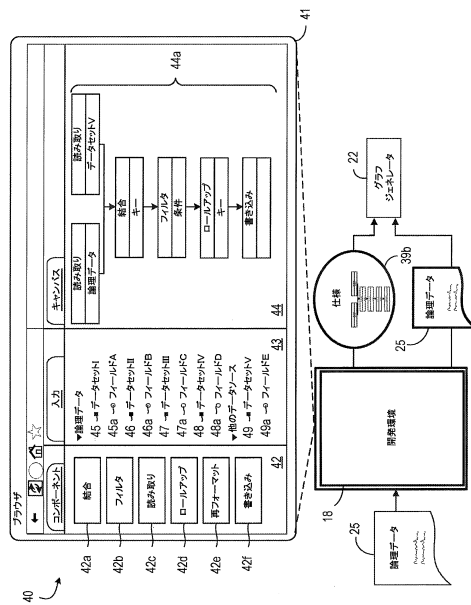
20

30

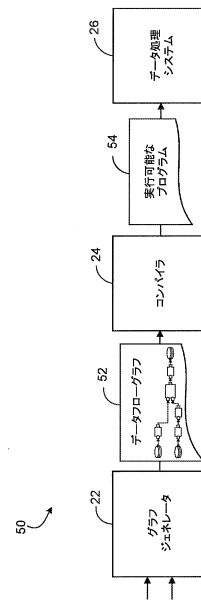
40

50

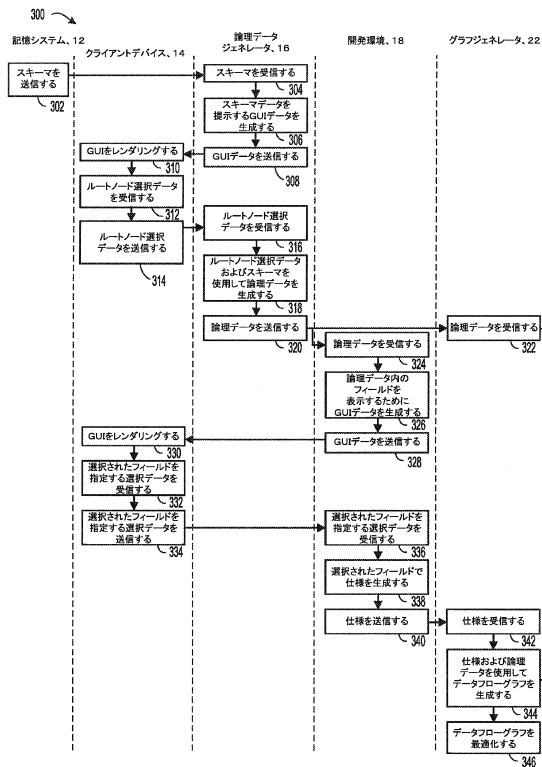
【図 2 D】



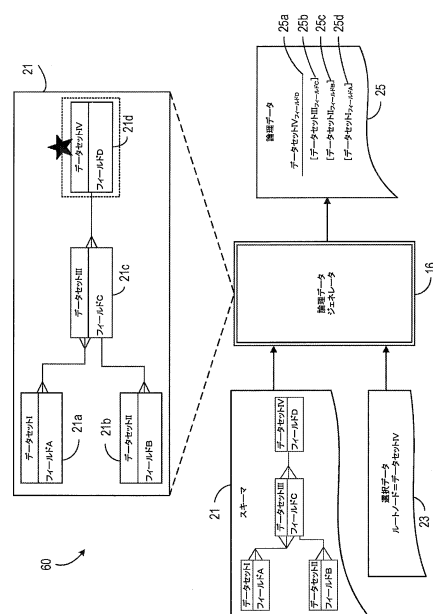
【図 2 E】



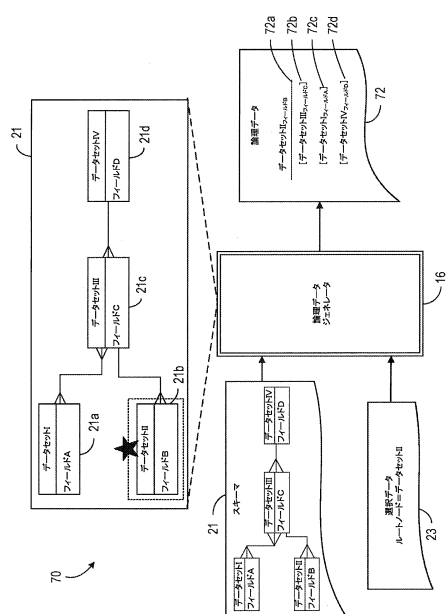
【図 3】



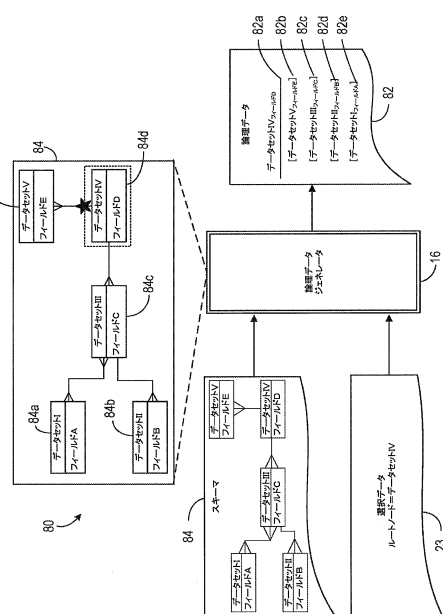
【図 4 A】



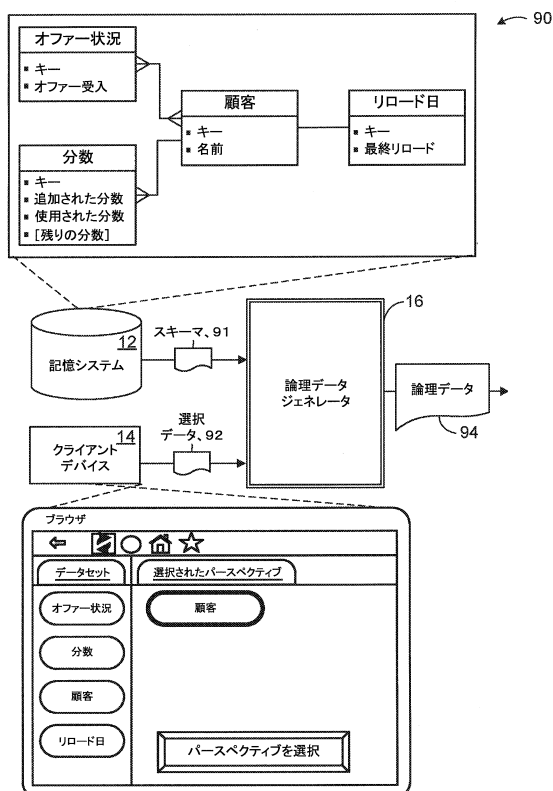
【図 4 B】



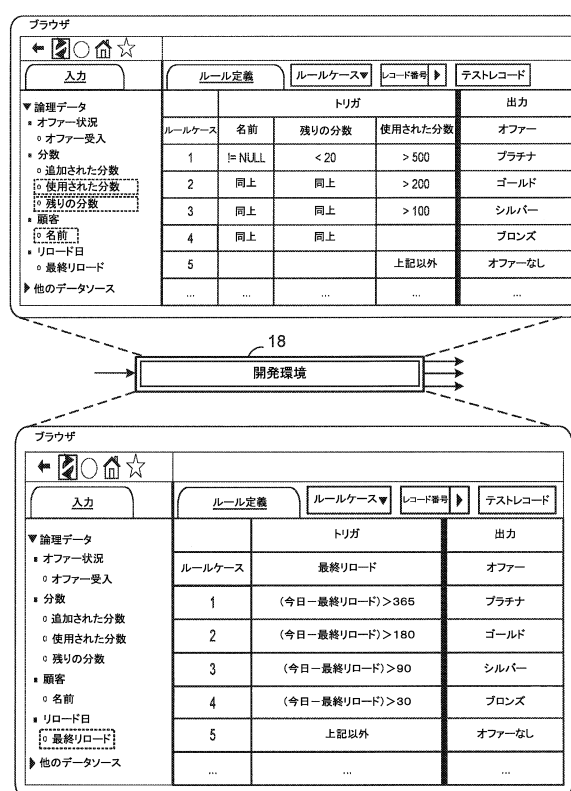
【図 4 C】



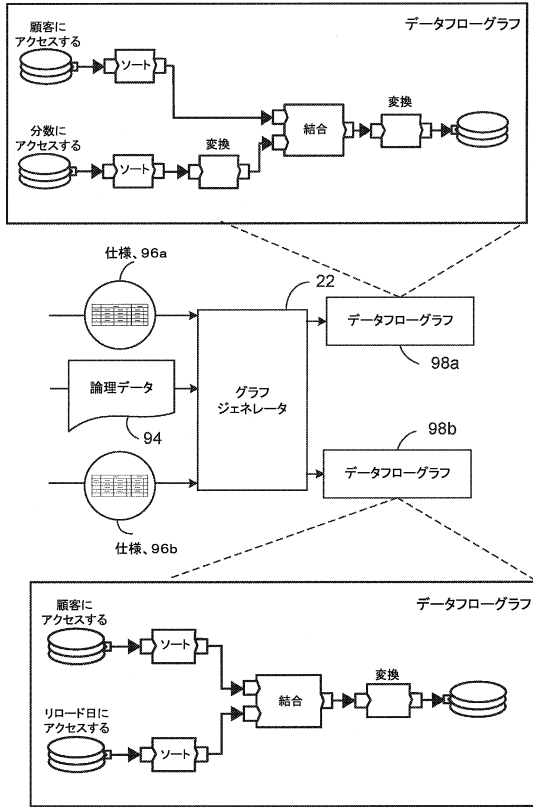
【図 5 A - 1】



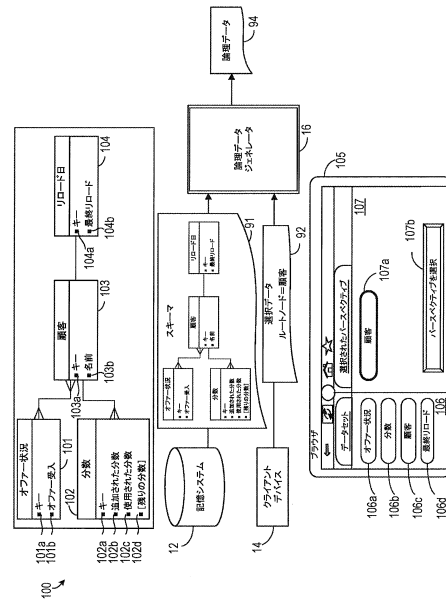
【図 5 A - 2】



【図 5 A - 3】



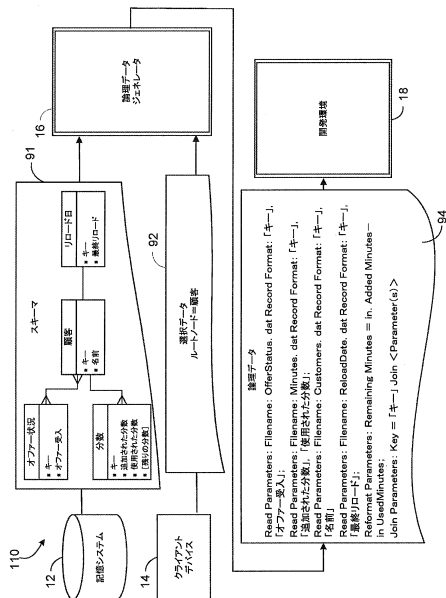
【図 5 B】



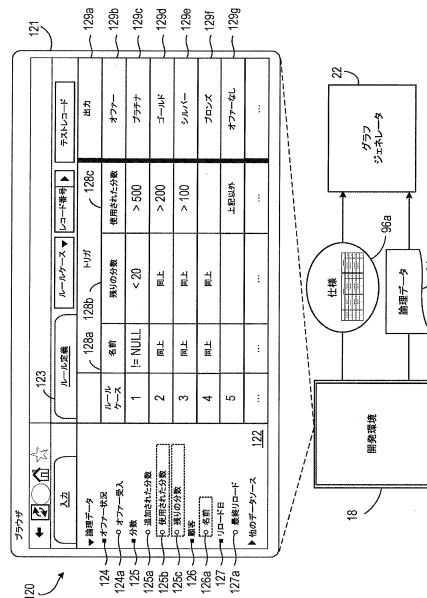
10

20

【図 5 C】



【図 5 D】

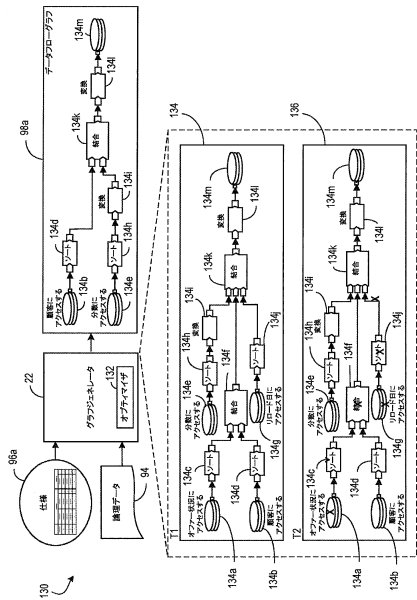


30

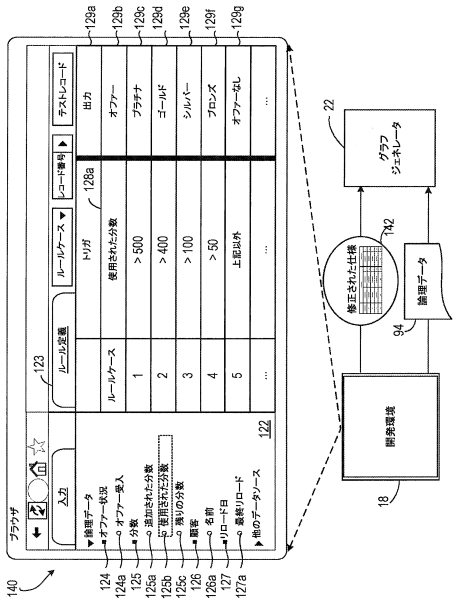
40

50

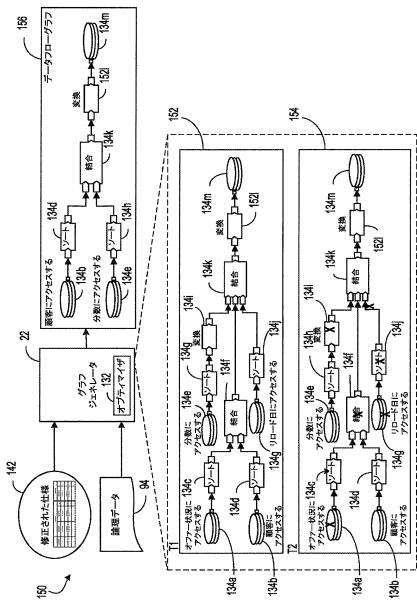
【図 5 E】



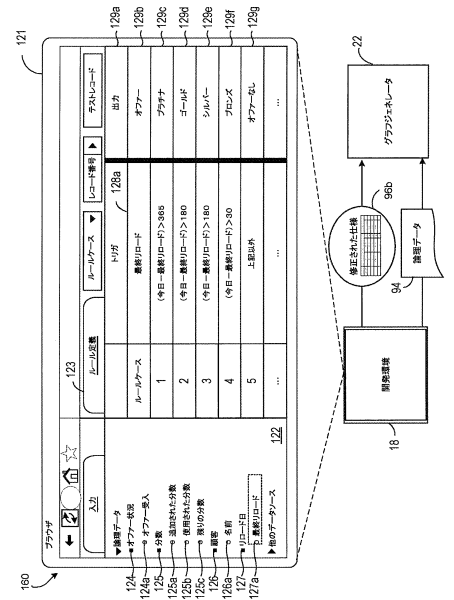
【図 5 F】



【図 5 G】



【図 5 H】



10

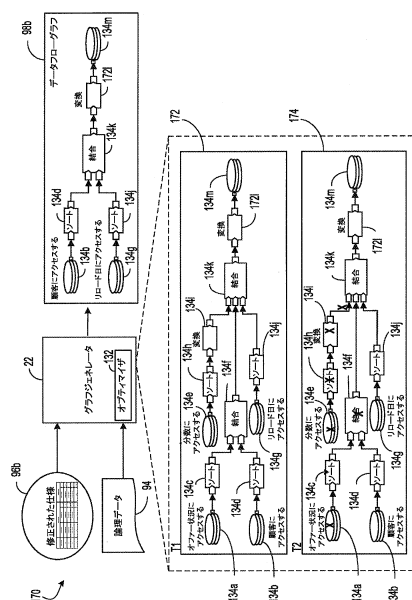
20

30

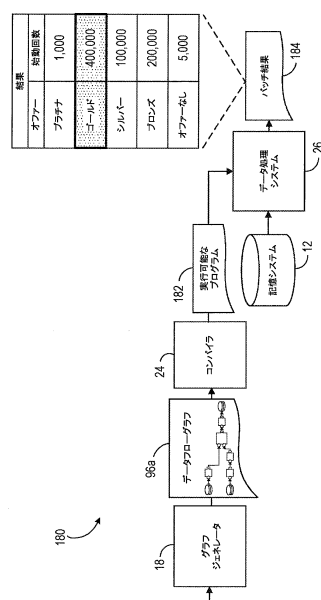
40

50

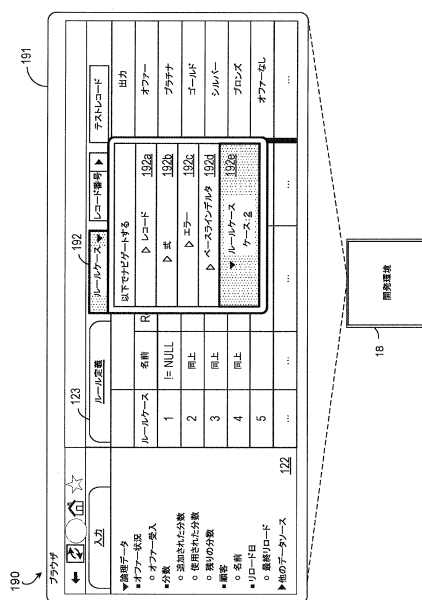
【図 5 I】



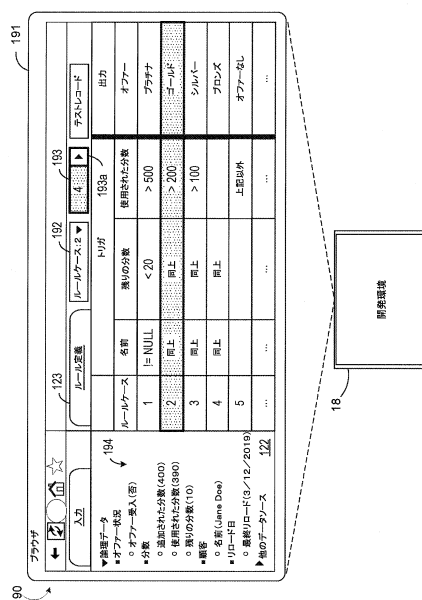
【図 5 J】



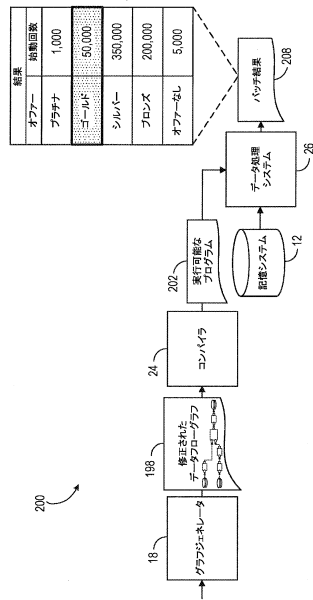
【図 5 K】



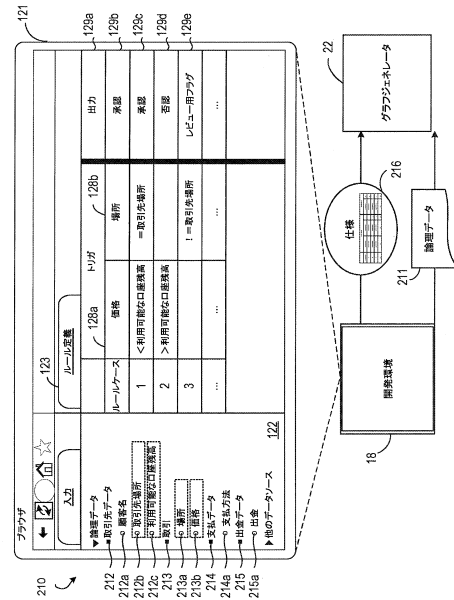
【図 5 L】



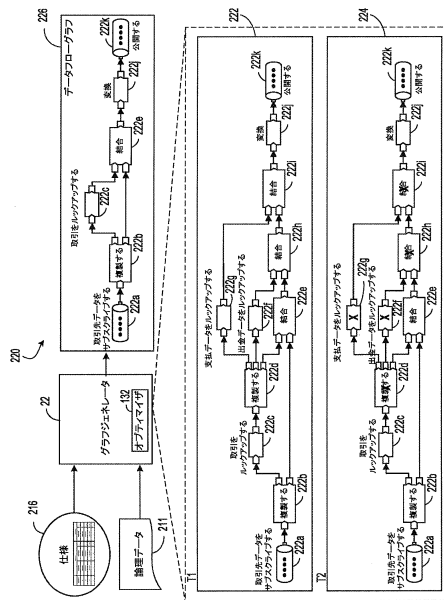
【図 5 Q】



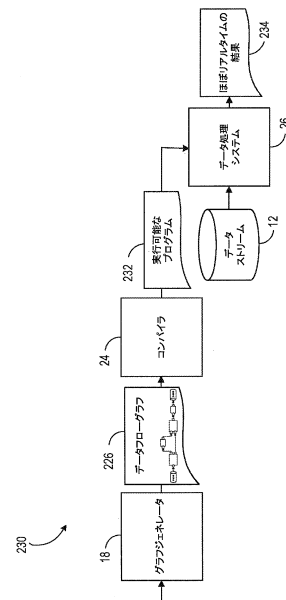
【図 6 A】



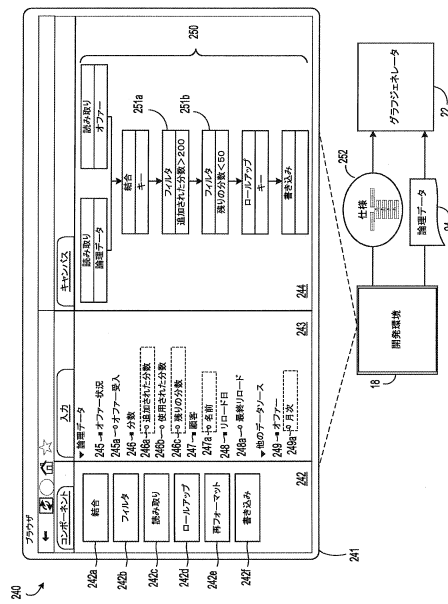
【図 6 B】



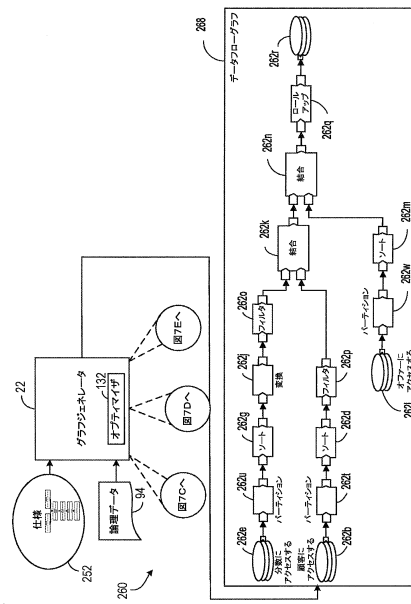
【図 6 C】



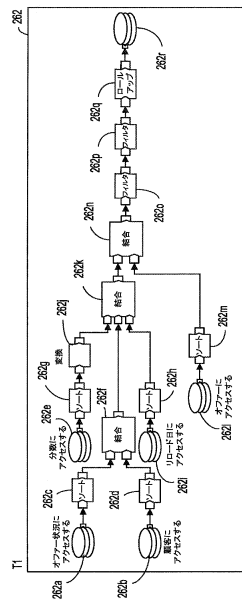
【図 7 A】



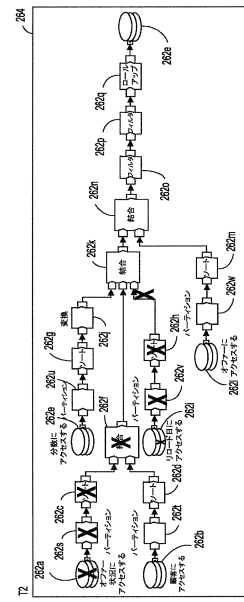
【図 7 B】



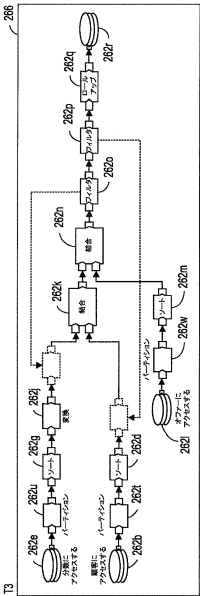
【図 7 C】



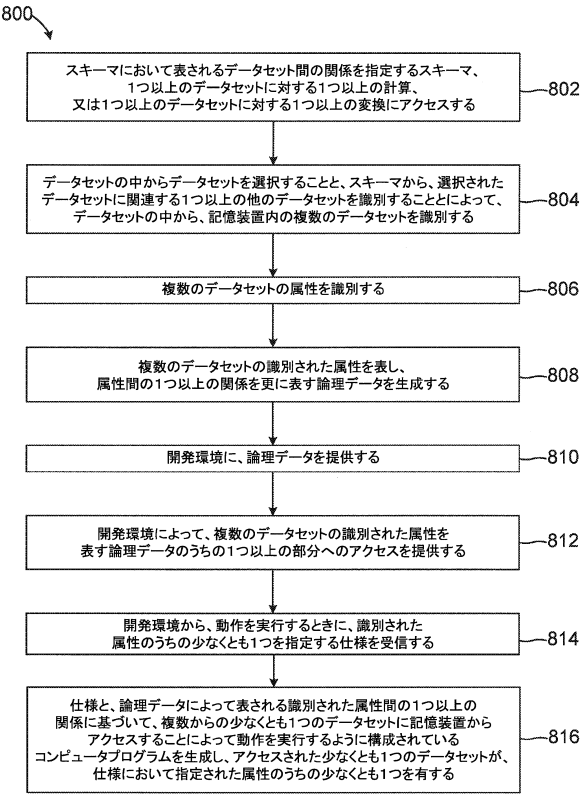
【図 7 D】



【図 7 E】



【図 8】



10

20

30

40

50

フロントページの続き

(33)優先権主張国・地域又は機関
米国(US)

早期審査対象出願

(72)発明者 マーシャル・エー・イスマン
アメリカ合衆国・マサチューセッツ・02458・ニュートン・バレー・スプリング・ロード・11

(72)発明者 イアン・シェクター
アメリカ合衆国・マサチューセッツ・02067・シャロン・ブルック・ロード・94

審査官 多賀 実

(56)参考文献 米国特許出願公開第2019/370407 (US, A1)
特表2013-541784 (JP, A)
米国特許出願公開第2010/198844 (US, A1)

(58)調査した分野 (Int.Cl., DB名)
G06F 8/00-8/77
G06F 16/00-16/958