

(19) 日本国特許庁 (JP)

(12) 特 許 公 報 (B2)

(11) 特許番号

特許第6754696号
(P6754696)

(45) 発行日 令和2年9月16日 (2020.9.16)

(24) 登録日 令和2年8月26日 (2020.8.26)

(51) Int. Cl.

F I

G 0 6 F 16/17 (2019.01)

G 0 6 F 16/17 1 0 0

G 0 6 F 9/46 (2006.01)

G 0 6 F 9/46 4 3 0

G 0 6 F 16/23 (2019.01)

G 0 6 F 16/23

請求項の数 9 (全 16 頁)

(21) 出願番号 特願2016-558637 (P2016-558637)
 (86) (22) 出願日 平成27年3月26日 (2015.3.26)
 (65) 公表番号 特表2017-509997 (P2017-509997A)
 (43) 公表日 平成29年4月6日 (2017.4.6)
 (86) 国際出願番号 PCT/US2015/022829
 (87) 国際公開番号 W02015/148849
 (87) 国際公開日 平成27年10月1日 (2015.10.1)
 審査請求日 平成30年1月19日 (2018.1.19)
 (31) 優先権主張番号 61/971, 273
 (32) 優先日 平成26年3月27日 (2014.3.27)
 (33) 優先権主張国・地域又は機関
 米国 (US)
 (31) 優先権主張番号 14/639, 266
 (32) 優先日 平成27年3月5日 (2015.3.5)
 (33) 優先権主張国・地域又は機関
 米国 (US)

(73) 特許権者 502303739
 オラクル・インターナショナル・コーポレ
 イション
 アメリカ合衆国カリフォルニア州9406
 5レッドウッド・シティー, オラクル・パ
 ークウェイ500
 (74) 代理人 110001195
 特許業務法人深見特許事務所
 (72) 発明者 カイ, ジミン
 中華人民共和国、100193 ペイジン
 、ハイディアン・ディストリクト、チョン
 グアンツン・ソフトウェア・パーク、ビル
 ディング・ナンバー・24、オラクル・ビ
 ルディング

最終頁に続く

(54) 【発明の名称】 ヘテロジニアスコンピューティング環境におけるデータタイプ変換をサポートするためのシステムおよび方法

(57) 【特許請求の範囲】

【請求項 1】

ヘテロジニアスコンピューティング環境におけるトランザクションシステムをサポートするためのシステムであって、

1つ以上のマイクロプロセッサと、

前記トランザクションシステムにおける1つ以上のレコードバッファとを備え、前記1つ以上のレコードバッファは、前記ヘテロジニアスコンピューティング環境におけるメインフレームコンピューティング環境における1つ以上のデータ構造を表現するように動作し、前記システムはさらに、

前記1つ以上のマイクロプロセッサ上で動作するトランザクションアダプタを備え、前記トランザクションアダプタは、前記メインフレームコンピューティング環境における前記1つ以上のデータ構造と、前記トランザクションシステムにおける前記1つ以上のレコードバッファとの間で、1つ以上のデータ要素を変換するように動作し、

前記トランザクションシステムは、1つ以上のトランザクションデータタイプに基づく1つ以上のトランザクションバッファを用いるとともに、前記1つ以上のレコードバッファと前記1つ以上のトランザクションバッファとの間で前記1つ以上のデータ要素を変換するように動作し、

前記トランザクションアダプタは、

ツリー構造を用いて前記メインフレームコンピューティング環境における前記1つ以上のデータ構造を表現するように動作し、

10

20

シリアル化されたツリーを用いてオブジェクトファイルまたはメモリ内に前記ツリー構造を維持するように動作し、

前記ツリー構造に含まれる複数のノードを事前に順序付けするように動作する、システム。

【請求項 2】

前記 1 つ以上のデータ構造は、前記メインフレームコンピューティング環境におけるコピブック内に定義されている、請求項 1 に記載のシステム。

【請求項 3】

前記トランザクションアダプタは、深さ優先方式に基づいて、前記複数のノードを事前に順序付けするように動作する、請求項 1 または 2 に記載のシステム。

10

【請求項 4】

前記 1 つ以上のレコードバッファ内に記憶された前記 1 つ以上のデータ要素を維持するように動作するレコードキャッシュをさらに備える、請求項 1 ~ 3 のいずれか 1 項に記載のシステム。

【請求項 5】

前記トランザクションシステムは、1 つ以上のレコード機能を用いて、前記 1 つ以上のレコードバッファ内に記憶された前記 1 つ以上のデータ要素を処理するように動作する、請求項 1 ~ 4 のいずれか 1 項に記載のシステム。

【請求項 6】

前記トランザクションアダプタは、前記メインフレームコンピューティング環境における前記 1 つ以上のデータ構造に基づいて記述ファイルを作成するように動作する、請求項 1 ~ 5 のいずれか 1 項に記載のシステム。

20

【請求項 7】

前記トランザクションアダプタは、バイナリフォーマットのオブジェクトファイルを前記記述ファイルから生成するように動作する、請求項 6 に記載のシステム。

【請求項 8】

ヘテロジニアスコンピューティング環境におけるトランザクションシステムをサポートするための方法であって、

前記トランザクションシステムにおいて 1 つ以上のレコードバッファを提供するステップと、

30

前記ヘテロジニアスコンピューティング環境において、前記 1 つ以上のレコードバッファを用いて 1 つ以上のデータ構造を表現するステップと、

トランザクションアダプタを介して、メインフレームコンピューティング環境における前記 1 つ以上のデータ構造と、前記トランザクションシステムにおける前記 1 つ以上のレコードバッファとの間で、1 つ以上のデータ要素を変換するステップとを備え、

前記トランザクションシステムは、1 つ以上のトランザクションデータタイプに基づく 1 つ以上のトランザクションバッファを用いるとともに、前記 1 つ以上のレコードバッファと前記 1 つ以上のトランザクションバッファとの間で前記 1 つ以上のデータ要素を変換し、

前記トランザクションアダプタは、

40

ツリー構造を用いて前記メインフレームコンピューティング環境における前記 1 つ以上のデータ構造を表現し、

前記トランザクションアダプタは、シリアル化されたツリーを用いてオブジェクトファイルまたはメモリ内に前記ツリー構造を維持し、

前記ツリー構造に含まれる複数のノードを事前に順序付ける、方法。

【請求項 9】

実行されると、以下のステップをシステムに行なわせる命令を含むコンピュータプログラムであって、前記以下のステップは、

トランザクションシステムにおいて 1 つ以上のレコードバッファを提供するステップと

50

ヘテロジニアスコンピューティング環境において、前記１つ以上のレコードバッファを用いて１つ以上のデータ構造を表現するステップと、

トランザクションアダプタを介して、メインフレームコンピューティング環境における前記１つ以上のデータ構造と、前記トランザクションシステムにおける前記１つ以上のレコードバッファとの間で、１つ以上のデータ要素を変換するステップとを備え、

前記トランザクションシステムは、１つ以上のトランザクションデータタイプに基づく１つ以上のトランザクションバッファを用いるとともに、前記１つ以上のレコードバッファと前記１つ以上のトランザクションバッファとの間で前記１つ以上のデータ要素を変換し、

前記トランザクションアダプタは、

ツリー構造を用いて前記メインフレームコンピューティング環境における前記１つ以上のデータ構造を表現し、

前記トランザクションアダプタは、シリアル化されたツリーを用いてオブジェクトファイルまたはメモリ内に前記ツリー構造を維持し、

前記ツリー構造に含まれる複数のノードを事前に順序付けする、コンピュータプログラム。

【発明の詳細な説明】

【技術分野】

【０００１】

著作権表示：

この特許文献の開示の一部は、著作権保護の対象となる題材を含んでいる。著作権の所有者は、特許商標庁の包袋または記録に掲載されるように特許文献または特許情報開示を誰でも複製できることに対して異議はないが、その他の点ではすべての如何なる著作権をも保有する。

【０００２】

発明の分野：

本発明は概してコンピュータシステムおよびソフトウェアに関し、特にヘテロジニアスコンピューティング環境（heterogeneous computing environment）におけるトランザクションシステムをサポートすることに関する。

【背景技術】

【０００３】

背景：

トランザクションシステムは、組織内のさまざまなトランザクションを処理し得るエンタープライズアプリケーションサーバを含む。高性能ネットワークおよびマルチプロセッサコンピュータなどの新しい技術の開発に伴って、トランザクションシステムの性能をさらに向上させる必要がある。これらは、概して、本発明の実施形態が取り組むことを意図している分野である。

【発明の概要】

【課題を解決するための手段】

【０００４】

概要：

ヘテロジニアスコンピューティング環境におけるトランザクションシステムをサポートし得るシステムおよび方法がここに記載される。トランザクションシステムは１つ以上のバッファを提供し得る。上記１つ以上のバッファは、ヘテロジニアスコンピューティング環境におけるメインフレームコンピューティング環境における１つ以上のデータ構造を表現するように動作する。また、トランザクションシステムはトランザクションアダプタを提供し得る。このトランザクションアダプタは、メインフレームコンピューティング環境における上記１つ以上のデータ構造と、トランザクションシステムにおける上記１つ以上のバッファとの間で、１つ以上のデータ要素を変換するように動作する。

【０００５】

10

20

30

40

50

図面の簡単な説明：

【図面の簡単な説明】

【0006】

【図1】本発明の一実施形態に係る、ヘテロジニアスコンピューティング環境におけるさまざまなアプリケーション間の遣り取りをサポートすることの説明図である。

【図2】本発明の一実施形態に係る、ヘテロジニアスコンピューティング環境におけるデータタイプ変換をサポートすることの説明図である。

【図3】本発明の一実施形態に係る、コピーブックレコードオブジェクト記述の説明図である。

【図4】本発明の一実施形態に係る、レコードオブジェクト構造をサポートするためのシリアライズされたツリーの使用の説明図である。

10

【図5】本発明の一実施形態に係る、レコードキャッシュをサポートすることの説明図である。

【図6】本発明の一実施形態に係る、ヘテロジニアスコンピューティング環境におけるさまざまなトランザクションプロセスをサポートすることの説明図である。

【図7】本発明の一実施形態に係る、ヘテロジニアスコンピューティング環境におけるデータタイプ変換をサポートするための例示的フローチャートである。

【発明を実施するための形態】

【0007】

詳細な説明：

20

本発明は、添付の図面において、限定ではなく例示のために示されており、図中同様の参照符号は同様の要素を指している。この開示において「或る」、「1つの」または「いくつかの」実施形態と言及する場合、これは必ずしも同じ実施形態である必要はなく、このように言及する場合、少なくとも1つを意味していることに留意すべきである。

【0008】

以下の本発明の説明において、トランザクションシステムの一例としてOracle Tuxedoシステム（以下、単に「Tuxedoシステム」または「Tuxedo」と称する）が用いられている。他のタイプのトランザクションシステムが限定されることなく用いられ得ることは、当業者には明らかであろう。

【0009】

30

ヘテロジニアスコンピューティング環境におけるトランザクションシステムをサポートし得るシステムおよび方法がここに記載される。

【0010】

ヘテロジニアスコンピューティング環境におけるデータタイプ変換

ヘテロジニアスコンピューティング環境におけるさまざまなアプリケーションは、ネットワークを通じてさまざまなプロトコルを用いて互いに通信し得る。

【0011】

たとえば、メインフレームコンピューティング環境において主に用いられるプログラミング言語であるCOBOLは、コピーブックを用いてデータ構造を定義する。COBOLでは、コピーブックファイルを用いて、メインフレームプログラムによって参照され得るデータ要素を定義し得る。COBOLレコードの記述フォーマットはフレキシブルであり、グループ、可変サイズテーブル、REDEFINESの関係などのさまざまな機能性をサポートする。

40

【0012】

コピーブックは、1つ以上のコピーブックレコード基本項目と、1つ以上のコピーブックレコードグループ項目とを含み得る。コピーブックレコード基本項目は、従属データ項目を有さないレコードのデータ項目である（すなわち、基本項目はさらに細分されない）。コピーブックレコードグループ項目は1つ以上の従属データ項目を含む。データ項目の1つの階層内では、基本項目は2つ以上のグループ項目には従属し得ない。また、グループは1つ以上のサブグループを含み得る。

50

【 0 0 1 3 】

さらに、ヘテロジニアスコンピューティング環境は、さまざまな特徴をサポートするためのさまざまなバッファを用い得るトランザクションシステム（たとえばOracle Tuxedoシステムなど）を含み得る。たとえば、Oracle Tuxedo ATMI（Application to Transaction Manager Interface）クライアントは、さまざまなカテゴリ（またはタイプ）のメモリ領域である型付きバッファを用いて、ATMIサーバにメッセージを送信する。Tuxedoシステムにおいて、型付きバッファは1つ以上のサブカテゴリ（またはサブタイプ）に任意に関連付けられてもよい。

【 0 0 1 4 】

図1は、本発明の一実施形態に係る、ヘテロジニアスコンピューティング環境におけるトランザクションアプリケーション間の遣り取りをサポートすることの説明図である。図1に示すように、ヘテロジニアスコンピューティング環境100におけるトランザクションシステム101は、メインフレームコンピューティング環境102において展開されたさまざまなソフトウェアプログラム104と遣り取りし得る。

10

【 0 0 1 5 】

たとえば、メインフレームソフトウェアプログラム104は、コピーブック106を用いて、メインフレームソフトウェアプログラム104によって用いられる1つ以上のデータ構造116を定義し得る。

【 0 0 1 6 】

本発明の一実施形態によれば、トランザクションシステム101は、レコードバッファ105を用いてメインフレームコンピューティング環境におけるさまざまなデータ構造116（たとえばコピーブック106におけるレコードなど）を表現し得る。

20

【 0 0 1 7 】

図1に示すように、レコードバッファ105は、（メモリ内の）ツリー構造を用いてCOBOLレコード定義をコピーブック106内にマップし得る。また、レコードバッファ105とコピーブック106との間でデータを転送し得る。

【 0 0 1 8 】

本発明の一実施形態によれば、ソースデータ（source data）をデスティネーションデータ（destination data）のタイプに自動的に変換し得る。たとえば、トランザクションシステム101は、Tuxedoメインフレームアダプタ（TMA：Tuxedo mainframe adapter）などのアダプタ103を用いて、メインフレームコンピューティング環境102とトランザクションシステム101との間で1つ以上のデータ要素110を自動的に変換し得る。

30

【 0 0 1 9 】

加えて、トランザクションアプリケーション107は、レコードバッファ105におけるデータに直接アクセスし、そのデータを処理し得る。

【 0 0 2 0 】

図2は、本発明の一実施形態に係る、ヘテロジニアスコンピューティング環境におけるデータタイプ変換をサポートすることの説明図である。図2に示すように、ヘテロジニアスコンピューティング環境200は、メインフレームコンピューティング環境におけるコピーブック206と、トランザクションシステムにおけるレコードバッファ201との間のデータタイプ変換をサポートし得る。

40

【 0 0 2 1 】

本発明の一実施形態によれば、トランザクションシステムは、ツール205を用いてCOBOLコピーブック206をレコードオブジェクトファイル203に変換し得る。このレコードオブジェクトファイル203は、実行時にトランザクションアプリケーションによって用いられ得る。

【 0 0 2 2 】

図2に示すように、変換ツール205は、（データ構造216のための）COBOLコピーブック206からレコード記述ファイル204を生成し得る。さらに、変換ツール2

50

05は、レコード記述ファイル204に基づいてレコードオブジェクトファイル203を生成し得る。実行時に、レコード記述ファイル204が用いられて、レコードバッファ201構造にアクセスしてこれらの構造を処理し得る。これによって、COBOLコピーブック定義が表現通りにレコードオブジェクトファイル203内にマップされる。

【0023】

たとえば、Tuxedoは、cpy2recordツールを用いてソースレコード記述ファイル204をコンパイルし、バイナリフォーマットのレコードオブジェクトファイル203を作成し得る。実行時に、Tuxedoは、RECORDFILESおよびRECORDDIRの環境変数における構成に基づいてレコードオブジェクトファイル203を用い得る。RECORDFILES変数は、アプリケーションのための、オブジェクトレコードファイルのカンマ区切りリストを含む。RECORDDIR変数は、関連するファイル名を有するRECORDオブジェクトファイルを見つけるために用いられ得る、ディレクトリのコロン区切りリストを指定し得る。このため、Cプログラミング言語を用いるTuxedoアプリケーションは、コピーブック206のレコード記述ファイル204をCOBOLプログラミング言語で利用することができる。

【0024】

本発明の一実施形態によれば、変換ツール205は、さまざまな構文解析ルールに基づいてコピーブックレコードタイプを構文解析し、複数のノードを有する構文ツリーとすることができる。

【0025】

たとえば、構文ツリー中のノードは、さまざまなTuxedo VIEWおよびRECORDの情報を含み得る。また、Tuxedoは、DigitParseRuleを用いてCOBOLにおける外部10進形式（たとえばS9999）を処理し得る。この外部10進形式は、Tuxedo VIEW整数およびRECORD RTYPE__DIGIT形式に変換され得る。加えて、Tuxedoは、Comp3ParseRuleを用いてCOBOLにおける内部10進形式（すなわちPACKED-DECIMAL、たとえばS9999COMP-3）を処理し得る。この内部10進形式は、Tuxedo dec__t形式に変換され得る。さらに、TuxedoはRecordfileGeneratorを用いてさまざまなレコード項目を作成し得る。RecordfileGeneratorは、さまざまなレコード項目の構造（たとえばオフセット/タイプなど）が正しく計算および構成されることを確実にすることができる。

【0026】

図2に示すように、システムは、メモリ210におけるレコードバッファ201を直接定義し、当該レコードバッファ201にアクセスし、かつこれを処理するために1組のレコード機能202を提供し得る。たとえば、Tuxedoレコード機能202は、割当てられたRECORDバッファを初期化するための機能（たとえばRinit（）機能）を含み得る。また、レコード機能202は、RECORDバッファから項目またはレコード全体を検索するための機能（たとえばRget（）機能）を含み得る。Rget（）機能を用いると、RECORDにおけるソースデータが自動的にデスティネーションデータのタイプに変換される。加えて、レコード機能202は、項目またはレコード全体をRECORDバッファに設定するための機能（たとえばRset（）機能）を含み得る。

【0027】

本発明の一実施形態によれば、レコード機能202は、1つ以上のエラー値を戻すことによって、エラー処理をサポートし得る。たとえば、エラーでなければあり得ない値（たとえば値-1）を戻すことによってエラー条件を表わし得る。Tuxedoにおいて、Ferrorで定義されたエラー番号とともにエラーメッセージが表示され得る。加えて、Tuxedoは、標準エラー出力時にメッセージを作成することができるF__error（）機能を提供し得る。また、Tuxedoは、メッセージカタログからエラーメッセージのテキストを検索することができるFstrerror（）機能を提供し得る。

【0028】

図 3 は、本発明の一実施形態に係るコピーブックレコードオブジェクト記述の説明図である。図 3 に示すように、ツリー 300 はノード A ~ J を含んでおり、このツリー 300 を用いて、以下のリスト 1 に示されるコピーブックのためのレコードオブジェクト記述を表現することができる。

【0029】

【表 1】

```

01 A.
05 B.
    10 E PIC S9(9) COMP-5.
    10 F.
        15 K PIC X(4).
        15 L REDEFINES K PIC S9(8) COMP.
05    C PIC S9(4).
05    D OCCURS 10 TIMES DEPENDING ON E.
10    G PIC S9999 SIGN LEADING.
10    H PIC S9(5)V9(2) PACKED-DECIMAL.
10    I USAGE COMP-1.
        10 J PIC X(10).

```

10

リスト 1

【0030】

20

図 3 に示すように、ツリー 300 の階層構造は、上記リスト 1 のコピーブックにおける関係を表現する。さらに、ツリー 300 におけるノード A ~ J の各々は、コピーブックにおけるフィールドを表現する。たとえば、ツリー 300 におけるノード E、F、および K はノード B の下にあり、このことは、コピーブックにおいてフィールド E、F、および K がグループ B の下にあることを表わしている。

【0031】

さらに、ツリー 300 は、ノード D がノード E に従属することを示すリンク 310 を含む。リンク 310 を用いて可変長テーブルを表現することができ、この可変長テーブルは、フィールド D の `od o` リンクフィールドにおいて `OCCURS DEPENDING ON (ODO)` 節 (`D OCCURS 10 TIMES DEPENDING ON E`) を用いて定義される。

30

【0032】

加えて、ツリー構造 300 はリンク 320 を含んでおり、このリンク 320 は、フィールド K を指し示すフィールド L の `redefine link` フィールドを用いて定義される。リンク 320 を用いて、上記リスト 1 における `REDEFINES` 節である「`L REDEFINES K`」をサポートすることができ、このことは、異なるデータ記述（すなわちフィールド L および K）が同じコンピュータ記憶領域を記述し得ることを表わす。

【0033】

図 4 は、本発明の一実施形態に係る、レコードオブジェクト構造をサポートするためのシリアライズされたツリー (serialized tree) の使用の説明図である。図 4 に示すように、シリアライズされたツリー 410 は (図 3 に示された) ツリー 300 に対応しており、複数のノード (たとえばノード A ~ J) を含み得る。

40

【0034】

本発明の一実施形態によれば、レコードオブジェクト構造 400 は、レコードオブジェクトファイル内および / またはメモリ内のシリアライズされたツリー 410 を用いて実現され、レコードオブジェクト構造 400 を用いてコピーブックレコードを表現することができる。

【0035】

加えて、ノード A ~ J はレコードオブジェクト構造 410 において事前に順序付けされ (pre-ordered) 得る。図 4 に示すように、ノード A ~ J は深さ優先方式に基づくことが

50

できる。

【 0 0 3 6 】

T u x e d o は、ツリーデータ構造を用いてメモリ内の R E C O R D オブジェクト記述を表現し得る。シリアル化されたツリー 4 1 0 を用いると、T u x e d o における R E C O R D のメモリレイアウトを連続的にすることができ、これによってコピーブックレイアウトをミラーリングし得る。このため、T u x e d o アプリケーションは、提供された R E C O R D 記述に基づいて、容易に R E C O R D バッファを処理することができる。

【 0 0 3 7 】

本発明の一実施形態によれば、シリアル化されたツリー 4 1 0 におけるノード A ~ J の各々はレコード項目を表現する。たとえば、T u x e d o R E C O R D 構造は複数の R E C O R D I T E M 要素に関連付けることができ、R E C O R D I T E M 要素の各々は、以下のリスト 2 に示される C 構造を用いて定義することができる。

【 0 0 3 8 】

【表 2】

```

struct RECORDITEM {
    unsigned int rname; /* offset to data item name in string pool */
    unsigned short rhash; /* hash value of the data item name */
    unsigned int vname; /* offset to corresponding view name in string pool */
    unsigned short type; /* data type number */
    unsigned int flag; /* flag options in bit pattern (the meaning type dependent) */
    unsigned int count; /* # of occurrence of the data item */
    unsigned int mincount; /* minimum occurrence of the data item */
    unsigned int maxcount; /* maximum occurrence of the data item */
    unsigned int odolink; /* link to OCCURS DEPENDING ON item */
    unsigned int rerulelink; /* link to redefine rule */
    unsigned int redfinelink; /* link to REDEFINES item */
    unsigned int size; /* size of the item */
    unsigned int offset; /* offset of the item to beginning of the buffer */
    unsigned int plink; /* link to parent */
    unsigned int clink; /* link to child */
    unsigned int fslink; /* link to forward sibling */
    unsigned int bslink; /* link to backward sibling */
    unsigned int rec_flds[4]; /* Unused. Reserve for future extension */
};

```

リスト 2

【 0 0 3 9 】

加えて、さまざまなレコード項目 A ~ J は、さまざまなコピーブックデータタイプに対応するさまざまなタイプに関連付けられ得る。以下のリスト 3 はさまざまなタイプの T u x e d o R E C O R D I T E M 要素を示す。

【 0 0 4 0 】

【表 3】

#define RTYPE_RECORD	0	/* RECORD or group */	
#define RTYPE_DIGIT	1	/* external decimal */	
#define RTYPE_UDIGIT	2	/* unsigned external decimal */	
#define RTYPE_FLOAT	3	/* single-precision float */	
#define RTYPE_DOUBLE	4	/* double-precision float */	
#define RTYPE_LONG	5	/* long - 8 bytes */	10
#define RTYPE_ULONG	6	/* unsigned long */	
#define RTYPE_INT	7	/* int - 4 bytes */	
#define RTYPE_UINT	8	/* unsigned int */	
#define RTYPE_SHORT	9	/* short - 2 bytes */	
#define RTYPE_USHORT	10	/* unsigned short */	
#define RTYPE_DEC	11	/* decimal, dec_t */	
#define RTYPE_STRING	12	/* string - null terminated */	
#define RTYPE_CARRAY	13	/* character array */	20

リスト 3

【0041】

図 5 は、本発明の一実施形態に係るレコードキャッシュの説明図である。図 5 に示すように、トランザクションシステム 500 は、レコードキャッシュ 510 を用いてさまざまなレコード 501 ~ 505 を維持し得る。Tuxedo では、実行時に、レコードキャッシュ 510 が作成され、さまざまな RECORD のためのサマリー情報と各 RECORD にアクセスするためのポインタとを記憶し得る。

【0042】

本発明の一実施形態によれば、レコードキャッシュ 510 はダブルリンクリストフォーマットを用いて実現可能である。たとえば、Tuxedo RECORD キャッシュのサイズは、設定変数である FRECORDCACHE SIZE によって制御され得る（デフォルトサイズは 128 ）。

【0043】

以下のリスト 4 は Tuxedo RECORD キャッシュの構造を定義する。

【0044】

30

【表 4】

```

#define FRECORDNAMESIZE 32

struct RECORDTBL {

    char rname[FRECORDNAMESIZE]; /* name of record */

    unsigned int tosize; /* total size of RECORD */

    unsigned int nitems; /* # of items in the RECORD */

    unsigned short last; /* link to last RECORD */

    unsigned short next; /* link to next RECORD */

    char *recordptr; /* ptr to RECORD */

};

```

10

リスト4

【0045】

加えて、レコードキャッシュ510がフルであるとき、レコードキャッシュ510は、LRU (least recent used: 最後に使われてから最も長い時間が経過した) アルゴリズムを用いて、どのレコードを廃棄し得るかを決定し得る。

20

【0046】

図6は、本発明の一実施形態に係る、ヘテロジニアスコンピューティング環境におけるさまざまなトランザクションプロセスをサポートすることの説明図である。図6に示すように、ヘテロジニアスコンピューティング環境600は、1つ以上のメインフレームアプリケーション607と遣り取りし得るさまざまなトランザクションプロセス（またはプロセッサ）、たとえばTuxedoクライアント601およびTuxedoサーバ602などを含み得る。

【0047】

たとえば、メインフレームアプリケーション607は、顧客情報制御システム/拡張システムアーキテクチャ(CICS TS: customer information control system/extended system architecture) および情報管理システム(IMS: information management system) インターフェイス、または他のアプリケーションプログラム間通信(APPC: application program-to-program communication) もしくはCICS TSインターフェイスに基づくことができる。

30

【0048】

図6に示すように、Oracle Tuxedoメインフレームアダプタ(TMA: Oracle Tuxedo Mainframe Adapter) 605などのアダプタを用いて、メインフレームアプリケーション607とTuxedoプロセス（またはプロセッサ）601および602との間の通信をサポートし得る。TMA 605は、通信リソースマネージャ(CRM: communications resource manager) を用いて通信リソースを管理し得る。CRMは、ATMプラットフォーム上で動作しているアプリケーションとメインフレーム上で動作しているアプリケーションとの間のデータフローを調整し得る。

40

【0049】

本発明の一実施形態によれば、ヘテロジニアスコンピューティング環境600は、メインフレームアプリケーション607によって用いられるコピーブック606と、トランザクションプロセス（またはプロセッサ）601～602によって用いられるレコードバッファ603（たとえばRECORD型付きバッファ）との間のデータタイプ変換をサポートし得る。

【0050】

50

図6に示すように、ドメインベースのTMA605によって、レコードバッファ603とコピーブック606との間でデータを自動的に変換することが可能となる。このため、TMA605は、さまざまなTuxedoプロセス601~602とメインフレームアプリケーション607との間に双方向接続を提供し得る。

【0051】

たとえば、Tuxedoはレコードバッファ603を用いてコピーブックレコード606を表現し得る。レコードバッファ603タイプは、個々のレコード構造を表現するサブタイプを有し得る。このため、Tuxedoは、レコードバッファ603に変換された後のCOBOLコピーブック606を利用することができる。なぜなら、レコードバッファ603がCOBOLコピーブック606へのアクセスを提供するからである。また、Tuxedoクライアント601およびTuxedoサーバ602は、RECORD記述ファイルに基づいてレコードバッファ603内に記述されたデータ構造を用い得る。

【0052】

さらに、Tuxedoによって、レコードバッファ603構造と他のバッファ構造（たとえば、C言語構造に基づくTuxedo FML/VIEWバッファ604など）との間のデータ転送が可能になる。たとえば、Tuxedoは、レコードバッファ603をFML/VIEWバッファ604に変換するためのFvrtos（）機能を用い得る。また、Tuxedoは、FML/VIEWバッファ604をレコードバッファ603に変換するためのFvstor（）機能を用い得る。

【0053】

本発明の一実施形態によれば、ソースデータとデスティネーションデータとの間の変換は自動的に実行可能である。たとえば、レコードバッファ603内のデータは、COBOL内部表現を維持するフォーマットを用いる。Rget（）機能が呼出されると、データがC言語構造のデータタイプに自動的に変換される。Rset（）機能が呼出されると、データがC言語構造からCOBOL内部表現に自動的に変換される。

【0054】

代替的には、TMA605は、コピーブック606を、Tuxedo VIEW/VIEW32バッファ604などの他のバッファ構造に直接変換し得る。VIEW/VIEW32は、コピーブックのすべての情報を正しく表現できるわけではない可能性がある。また、ユーザはこれらのバッファ構造を直接処理することができない可能性がある。加えて、コピーブック606における各グループがVIEWに翻訳された場合、システムに多数のVIEWが生じる可能性がある。

【0055】

加えて、Tuxedoは、CARRAY/FML/FML32/STRING/VIEW/VIEW32バッファなど、さまざまなタイプのバッファを利用することができる。各バッファは、特定のタイプとして指定可能である。加えて、バッファのタイプが異なっていると、メッセージの初期化、送信、および受信を行うのにさまざまなルーチンが必要となる可能性がある。また、バッファのタイプが異なっていると、データをエンコードおよびデコードするのにさまざまなルーチンが必要となる可能性がある。Tuxedoは、プログラミング介入無しで、特定タイプのための適切なルーチンを自動的に呼出し得る。

【0056】

図7は、本発明の一実施形態に係る、ヘテロジニアスコンピューティング環境におけるデータタイプ変換をサポートするための例示的フローチャートである。図7に示すように、ステップ701において、システムは、ヘテロジニアスコンピューティング環境におけるトランザクションシステムにおいて1つ以上のレコードバッファを提供し得る。さらに、ステップ702において、システムは、ヘテロジニアスコンピューティング環境において、上記1つ以上のレコードバッファを用いて1つ以上のデータ構造を表現し得る。次いで、ステップ703において、トランザクションアダプタは、メインフレームコンピューティング環境における上記1つ以上のデータ構造と、トランザクションシステムにおける上記1つ以上のレコードバッファとの間で、1つ以上のデータ要素を変換し得る。

【 0 0 5 7 】

本発明は、本開示の教示内容に従ってプログラムされた1つ以上のプロセッサ、メモリ、および/またはコンピュータ読取可能記憶媒体を含む1つ以上の従来の汎用もしくは専用デジタルコンピュータ、演算装置、マシン、またはマイクロプロセッサを用いて好都合に実現され得る。本開示の教示内容に基づいて、熟練したプログラマによって適切なソフトウェアコーディングが容易に作成され得ることは、ソフトウェア技術の当業者には明らかであろう。

【 0 0 5 8 】

本発明の一実施形態によれば、ヘテロジニアスコンピューティング環境におけるトランザクションシステムが提供される。本実施形態のトランザクションシステムは、たとえば
10 トランザクションシステム101、500など、前述の実施形態で説明した如何なるトランザクションシステムであってもよい。さらに、トランザクションシステムおよびその構成要素は、前述の実施形態で説明した本発明の原理に基づく如何なる適切な動作をも実行し得る。

【 0 0 5 9 】

本実施形態の例では、トランザクションシステムは、1つ以上のレコードバッファとトランザクションアダプタとを備え得る。1つ以上のレコードバッファは、ヘテロジニアスコンピューティング環境におけるメインフレームコンピューティング環境における1つ以上のデータ構造を表現するように動作し得る。トランザクションアダプタは、メインフレームコンピューティング環境における1つ以上のデータ構造と、1つ以上のレコードバッ
20 ファとの間で、1つ以上のデータ要素を変換するように動作し得る。

【 0 0 6 0 】

本実施形態の例では、1つ以上のデータ構造は、メインフレームコンピューティング環境におけるコピーブック内に定義され得る。

【 0 0 6 1 】

本実施形態の例では、トランザクションアダプタは、ツリー構造を用いてメインフレームコンピューティング環境における1つ以上のデータ構造を表現するように動作し得る。

【 0 0 6 2 】

本実施形態の例では、トランザクションアダプタは、シリアライズされたツリーを用いてオブジェクトファイルまたはメモリ内にツリー構造を維持するように動作し得る。
30

【 0 0 6 3 】

本実施形態の例では、トランザクションシステムは、1つ以上のレコードバッファ内に記憶された1つ以上のデータ要素を維持するように動作するレコードキャッシュをさらに備え得る。

【 0 0 6 4 】

本実施形態の例では、トランザクションシステムは、1つ以上のレコード機能を用いて、1つ以上のレコードバッファ内に記憶された1つ以上のデータ要素を処理するように動作し得る。

【 0 0 6 5 】

本実施形態の例では、トランザクションアダプタは、メインフレームコンピューティング環境における1つ以上のデータ構造に基づいて記述ファイルを作成するように動作し得る。
40

【 0 0 6 6 】

本実施形態の例では、トランザクションアダプタは、バイナリフォーマットのオブジェクトファイルを記述ファイルから生成するように動作し得る。

【 0 0 6 7 】

本実施形態の例では、トランザクションシステムは、1つ以上のトランザクションデータタイプに基づく1つ以上のトランザクションバッファを用いるように動作し得る。

【 0 0 6 8 】

本実施形態の例では、トランザクションシステムは、1つ以上のレコードバッファと1
50

つ以上のトランザクションバッファとの間で1つ以上のデータ要素を変換するように動作し得る。

【0069】

いくつかの実施形態において、本発明は、本発明の如何なるプロセスも実行するようにコンピュータをプログラムするために用いることができる命令を格納した記憶媒体またはコンピュータ読取可能媒体であるコンピュータプログラム製品を含む。記憶媒体は、フロッピー（登録商標）ディスク、光ディスク、DVD、CD-ROM、マイクロドライブ、および光磁気ディスクを含む任意のタイプのディスク、ROM、RAM、EPROM、EEPROM、DRAM、VRAM、フラッシュメモリ装置、磁気カードまたは光カード、ナノシステム（分子メモリICを含む）、または、命令および/もしくはデータを格納するの

10

【0070】

本発明の上述の記載は例示および説明のために提供されたものである。それは、網羅的なものとして意図されたものではなく、または、開示された厳密な形態に本発明を限定するよう意図されたものではない。多くの変更例および変形例が当業者に明らかになるだろう。変更例および変形例は、開示された特徴の如何なる関連する組合せをも含む。実施形態は、本発明の原理およびその実用的応用を最良に説明するために選択され説明されたものであり、それにより、考えられる特定の使用に適したさまざまな実施形態についての、およびさまざまな修正を有する本発明を、当業者が理解できるようにする。本発明の範囲は、以下の請求項およびそれらの均等物によって定義されるよう意図されている。

20

【図1】

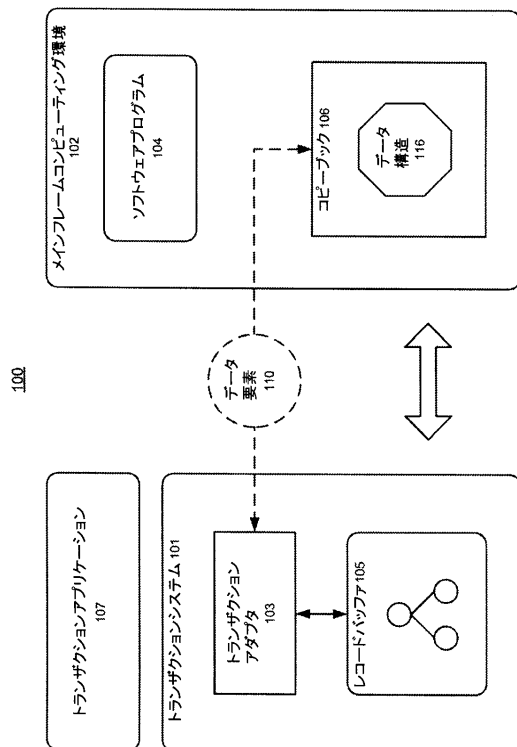


FIGURE 1

【図2】

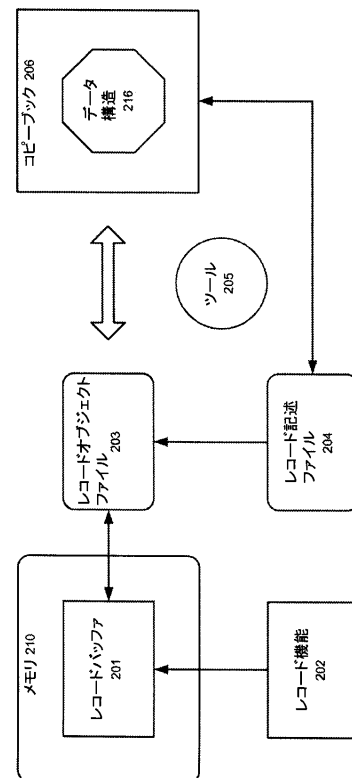
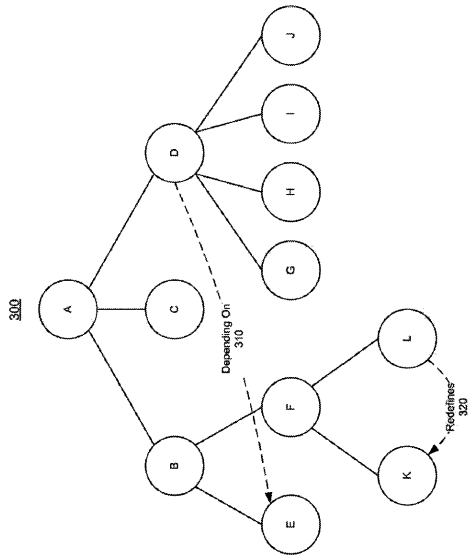
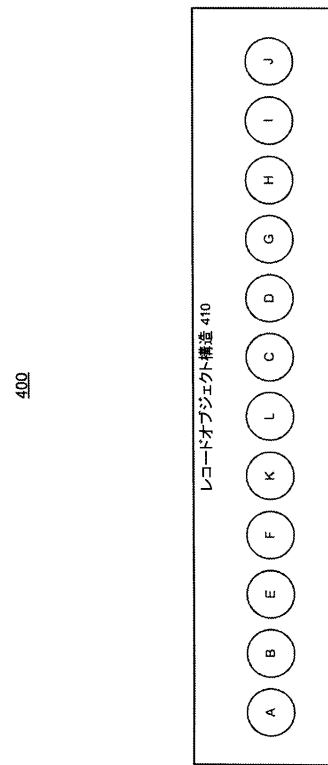


FIGURE 2

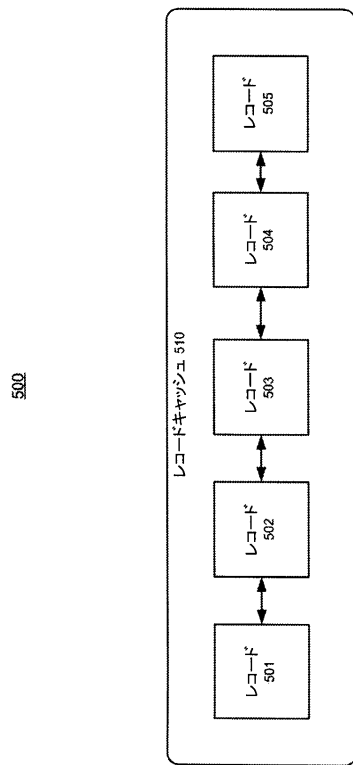
【図 3】



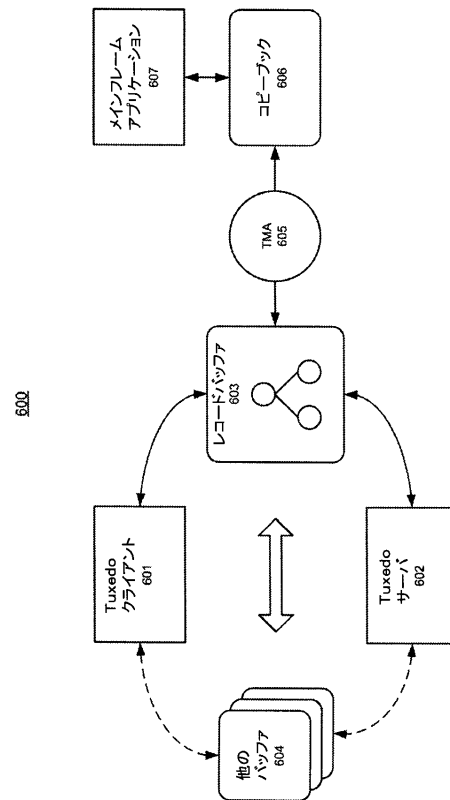
【図 4】



【図 5】



【図 6】



【図 7】

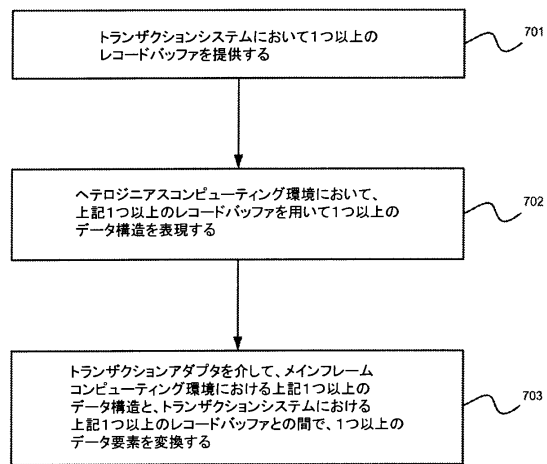


FIGURE 7

フロントページの続き

- (72)発明者 リー, シアンドン
中華人民共和国 ベイジン、シュエンウー・ディストリクト、チャンチュンジェシーリー、9 - 5
- 1
- (72)発明者 ワン, リーボン
中華人民共和国、100193 ベイジン、ハイディアン・ディストリクト、チョングアンツン・
ソフトウェア・パーク、ビルディング・ナンバー・24、オラクル・ビルディング
- (72)発明者 リトル, トッド・ジェイ
アメリカ合衆国、60067 - 6675 イリノイ州、パラタイン、イリノイ・アベニュー、115
5

審査官 漆原 孝治

- (56)参考文献 特開平04 - 227562 (JP, A)
GRANPOWER5000シリーズ, FM - Server, FMVシリーズ BEA TUX
EDO メインフレーム接続 V06 使用手引書 B1FN-4001-01-00, 日本, 富士通株式会社,
1998年 5月31日, 第1版, pp.1-13, 74-78

- (58)調査した分野(Int.Cl., DB名)
- | | |
|------|---------|
| G06F | 16 / 17 |
| G06F | 9 / 46 |
| G06F | 16 / 23 |