(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2005-513592
(P2005-513592A)
(43)公表日　平成17年5月12日(2005.5.12)

(51)Int.Cl.[7]	F I			テーマコード（参考）
G06F 15/16	G06F	15/16	620F	5B045
G06F 9/46	G06F	9/46	360F	5B098

審査請求 未請求　予備審査請求 有　（全 30 頁）

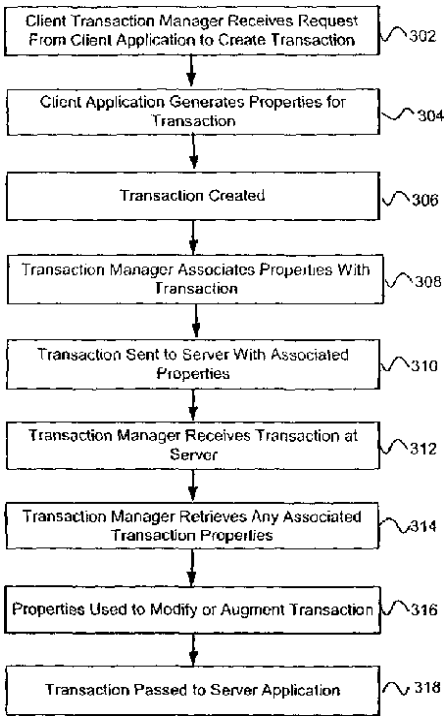
(21)出願番号	特願2003-514404 (P2003-514404)
(86)(22)出願日	平成14年7月16日 (2002.7.16)
(85)翻訳文提出日	平成16年3月10日 (2004.3.10)
(86)国際出願番号	PCT/US2002/022633
(87)国際公開番号	WO2003/009127
(87)国際公開日	平成15年1月30日 (2003.1.30)
(31)優先権主張番号	60/306,102
(32)優先日	平成13年7月17日 (2001.7.17)
(33)優先権主張国	米国(US)
(31)優先権主張番号	10/196,687
(32)優先日	平成14年7月15日 (2002.7.15)
(33)優先権主張国	米国(US)

(71)出願人　500105160
ビーイーエイ　システムズ，　インコーポレイテッド
ＢＥＡ　Ｓｙｓｔｅｍｓ，　Ｉｎｃ．
アメリカ合衆国　カリフォルニア　９５１３１，　サン　ノゼ，　ノース　ファースト　ストリート　２３１５
２３１５　Ｎｏｒｔｈ　Ｆｉｒｓｔ　Ｓｔｒｅｅｔ，　Ｓａｎ　Ｊｏｓｅ，　ＣＡＬＩＦＯＲＮＩＡ　９５１３１　Ｕ．Ｓ．Ａ．
(74)代理人　100082005
弁理士　熊倉　禎男
(74)代理人　100067013
弁理士　大塚　文昭

最終頁に続く

(54)【発明の名称】トランザクション・プロパティー機能を持つトランザクション処理のためのシステムおよび方法

(57)【要約】
本発明は、いくつかの対応付けられたトランザクション・プロパティ(308)を含ませるための、トランザクションを可能とするトランザクション処理システムを使用するためのメカニズムを提供する。トランザクション・プロパティは次に、名称付けされたオブジェクトが、トランザクションと対応付けられることを可能とする。トランザクション・プロパティは、トランザクションがアクティブな場所なら、どこでも利用可能である。それらは、トランザクションが、分散されたシステム(310)内の種々のサーバに流れる、または、一つのシステム内の異なったソフトウェア・モジュールの間を流れるにつれて、追加物または帯域外の、通信メカニズムのフォームとして使用され得る。



【特許請求の範囲】
【請求項１】
第１のアプリケーション・プロセスから第２のアプリケーションプロセスへのトランザクションの通信中にトランザクション・プロパティをトランザクションと対応付けるシステムであって、
前記第１のアプリケーション・プロセスが、特定のトランザクションと対応付けられるべきトランザクション・プロパティを指定することを可能とするファンクションを含む第１のトランザクション・インターフェース、
前記トランザクション・プロパティを、前記特定のトランザクションと対応付ける第１のトランザクション・マネジャ、
前記対応付けられたトランザクション・プロパティと一緒に前記特定のトランザクションを、前記第１のトランザクション・マネジャから、第２のトランザクション・マネジャに、通信する通信メカニズム、
前記特定のトランザクションと対応付けられた前記トランザクション・プロパティを検索する第２のトランザクション・マネジャ、および、
前記第２のアプリケーション・プロセスが、前記特定のトランザクションと対応付けられた前記トランザクション・プロパティを読む(read)ことを可能とするファンクションを含む第２のトランザクション・インターフェース、
を備えるシステム。
【請求項２】
前記トランザクション・インターフェースが、前記第１のアプリケーションが、特定のトランザクションと対応付けられるべきトランザクション・プロパティを指定することを可能とする複数のファンクションを含み、
更に、複数のトランザクション・プロパティが、前記特定のトランザクションと対応付けられ得る、
請求項１に記載のシステム。
【請求項３】
前記第１のトランザクション・インターフェースが、前記第１のアプリケーション・プロセスによる、前記トランザクション・コンテキストとともに余分の情報を伝搬させるための使用のためのsetPropertyファンクションを含む、請求項１に記載のシステム。
【請求項４】
前記トランザクション・プロパティが、キー値の組であり、
更に、前記キー値の組のキー要素が、ストリングであり、そして、
前記キー値の組の前記値の要素が、直列可能な(serializable)エンティティーである、
請求項１に記載のシステム。
【請求項５】
前記トランザクション・インターフェースが、Javaトランザクション・インターフェースを拡張する、請求項１に記載のシステム。
【請求項６】
前記ファンクションが、トランザクション名を、トランザクションと対応付けるためのsetNameファンクションである、
請求項１に記載のシステム。
【請求項７】
前記setNameファンクションが、それらのトランザクション・タイプによってトランザクションを名称付けするために用いられる、請求項６に記載のシステム。
【請求項８】
前記トランザクション・インターフェースが、プロパティ・タイプを、前記特定のトランザクションに対応付けるための、addPropertiesファンクションを含む、
請求項１に記載のシステム。
【請求項９】

前記インターフェースが、更に、
以前に、前記特定のトランザクションと対応付けられたプロパティ・タイプの値を設定するためのsetPropertyファンクション、
を含む、請求項８に記載のシステム。
【請求項１０】
前記第１のアプリケーション・プロセスが、トランザクションを開始する時に、特有の(unique)トランザクション識別子が、前記トランザクション・マネジャによって生成される請求項１に記載のシステム。
【請求項１１】
前記第２のトランザクション・マネジャが、前記トランザクション・プロパティの検索に起因して、トランザクション応答(transaction reply)を、前記第１のトランザクション・マネジャに通信する、請求項１に記載のシステム。
【請求項１２】
第１のアプリケーション・プロセスから第２のアプリケーション・プロセスへのトランザクションの通信中に、トランザクション・プロパティを、トランザクションと対応付けるための方法であって、
第１のアプリケーション・プロセスにおいて、第１のトランザクション・インターフェースへのコールによって、特定のトランザクションと対応付けられるべきトランザクション・プロパティを指定し、
第１のトランザクション・マネジャにおいて、前記トランザクション・プロパティを、前記特定のトランザクションと対応付け、
前記対応付けられたトランザクション・プロパティを持つ前記特定のトランザクションを、第２のアプリケーション・プロセスに通信し、
第２のトランザクション・マネジャにおいて、前記特定のトランザクションと対応付けられた前記トランザクション・プロパティを検索し、そして、
第２のアプリケーション・プロセスにおいて、第２のトランザクション・インターフェースへのコールによって、前記特定のトランザクションと対応付けられた前記トランザクション・プロパティを読取る、
ステップを含む方法。
【請求項１３】
前記特定のトランザクションと対応付けられた前記トランザクション・プロパティを検索した後に、前記第１のトランザクション・マネジャに応答するステップを更に含む、
請求項１２に記載の方法。
【請求項１４】
前記トランザクション・インターフェースが、前記第１のアプリケーション・プロセスが、特定のトランザクションと対応付けられるべきトランザクション・プロパティを指定する(specify)ことを可能とするための複数のファンクションを含み、そして、更に、複数のトランザクション・プロパティが、前記特定のトランザクションと対応付けられ得る、請求項１２に記載の方法。
【請求項１５】
前記第１のトランザクション・インターフェースが、前記第１のアプリケーション・プロセスによる、前記トランザクション・コンテキストとともに余分の(extra)情報を伝搬させるための使用のためのsetPropertyファンクションを含む、
請求項１２に記載の方法。
【請求項１６】
前記トランザクション・プロパティが、キー値の組であり、
更に、前記キー値の組の前記キー要素が、ストリングであり、
前記キー値の組の前記値の要素が、直列可能なエンティティである、
請求項１２に記載の方法。
【請求項１７】

前記トランザクション・インターフェースが、Javaトランザクション・インターフェースを拡張する、
請求項１２に記載の方法。
【請求項１８】
前記ファンクションが、トランザクション名を、トランザクションと対応付けるための、セット名(set name)ファンクションである、
請求項１２に記載の方法。
【請求項１９】
前記セット名ファンクションが、それらのトランザクション・タイプによって、トランザクションを名称付けするために用いられる、請求項１８に記載の方法。
【請求項２０】
前記トランザクション・インターフェースが、プロパティ・タイプを、前記特定のトランザクションと対応付けるための、addPropertiesファンクションを含む、
請求項１２に記載の方法。
【請求項２１】
前記インターフェースが、更に、以前に、前記特定のトランザクションと対応付けられたプロパティ・タイプの値を設定するための、setPropertyファンクションを含む、
請求項２０に記載の方法。
【請求項２２】
前記第１のアプリケーション・プロセスが、トランザクションを開始する時に、
特有の(unique)トランザクション識別子が、前記トランザクション・マネジャによって生成される、請求項１２に記載の方法。
【請求項２３】
第１のアプリケーション・プロセスから、第２のアプリケーション・プロセスへのトランザクションの通信中に、トランザクション・プロパティを、トランザクションと対応付けるためのシステムであって、
前記第１のアプリケーション・プロセスから、特定のトランザクションのためのリクエストを受信する第１のトランザクション・マネジャ、
前記第１のトランザクション・マネジャから、第２のトランザクション・マネジャへ、前記トランザクション・リクエストを通信する通信メカニズム、
前記第２のアプリケーション・プロセスが、前記特定のトランザクションと対応付けられるべきトランザクション・プロパティを指定することを可能とするファンクションを含む第１のトランザクション・インターフェース、
前記トランザクション・リクエストに応答する前に、前記トランザクション・プロパティを、前記特定のトランザクションと対応付ける、前記第２のトランザクション・マネジャ、および、
前記第１のアプリケーション・プロセスが、前記特定のトランザクションと対応付けられた前記トランザクション・プロパティを読み取ることを可能とするファンクションを含む第２のトランザクション・インターフェース、
を備えるシステム。
【請求項２４】
前記トランザクション・インターフェースが、前記第１のアプリケーション・プロセスが、特定のトランザクションと対応付けられるべきトランザクション・プロパティを指定することを可能とするための複数のファンクションを含み、更に、
複数のトランザクション・プロパティが、前記特定のトランザクションと対応付けられ得る、
請求項２３に記載のシステム。
【請求項２５】
前記第１のトランザクション・インターフェースが、前記第１のアプリケーション・プロセスによって、前記トランザクション・コンテキストとともに余分な(extra)情報を伝

搬させるために使用されるためのsetPropertyファンクションを含む、請求項２３に記載のシステム。
【請求項２６】
前記トランザクション・プロパティが、キー値の組であり、
更に、
前記キー値の組の前記キー要素が、ストリングであり、
前記キー値の組の前記値の要素が、直列可能なエンティティである、
請求項２３に記載のシステム。
【請求項２７】
前記トランザクション・インターフェースが、Javaトランザクション・インターフェースを拡張する、請求項２３に記載のシステム。
【請求項２８】
前記ファンクションが、トランザクション名を、トランザクションと対応付けるためのsetNameファンクションである、請求項２３に記載のシステム。
【請求項２９】
前記setNameファンクションが、それらのトランザクション・タイプによって、トランザクションを名称付け(name)するために使用される、請求項２８に記載のシステム。
【請求項３０】
前記トランザクション・インターフェースが、プロパティ・タイプを、前記特定のトランザクションと対応付けるためのaddPropertiesファンクションを含む、請求項２３に記載のシステム。
【請求項３１】
前記インターフェースが、更に、以前に、前記特定のトランザクションと対応付けられたプロパティ・タイプの値を設定するためのsetPropertyファンクションを含む、
請求項３０に記載のシステム。
【請求項３２】
前記第１のアプリケーション・プロセスが、トランザクションを開始する時に、
特有の(unique)トランザクション識別子が、前記トランザクション・マネジャによって生成される、
請求項２３に記載のシステム。
【請求項３３】
第１のアプリケーション・プロセスから、第２のアプリケーション・プロセスへのトランザクションの通信中に、トランザクション・プロパティを、トランザクションと対応付けるための方法であって、
第１のトランザクション・マネージャにおいて、特定のトランザクションのための、第１のアプリケーション・プロセスからのリクエストを受信し、
前記トランザクション・リクエストを、第２のトランザクション・マネジャに通信し、
第２のアプリケーション・プロセスにおいて、第１のトランザクション・インターフェースへのコールによって、前記特定のトランザクションと対応付けられるべきトランザクション・プロパティを指定し、
前記第２のトランザクション・マネジャにおいて、前記トランザクション・プロパティを、前記特定のトランザクションと対応付け、
前記トランザクション・リクエストに応答し、そして、
前記第１のアプリケーション・プロセスにおいて、第２のトランザクション・インターフェースへのコールによって、前記特定のトランザクションと対応付けられた前記トランザクション・プロパティを読み出す、
ステップを含む方法。
【請求項３４】
前記トランザクション・インターフェースが、
前記第１のアプリケーション・プロセスが、特定のトランザクションと対応付けられる

べきトランザクション・プロパティを指定することを可能とするための複数のファンクションを含み、そして、
更に、複数のトランザクション・プロパティが、前記特定のトランザクションと対応付けられ得る、
請求項３３に記載の方法。
【請求項３５】
前記第１のトランザクション・インターフェースが、前記第１のアプリケーション・プロセスによる、前記トランザクション・コンテキストとともに余分の(extra)情報を伝搬させるための使用のための、setPropertyファンクションを含む、
請求項３３に記載の方法。
【請求項３６】
前記トランザクション・プロパティが、キー値の組であり、更に、
前記キー値の組の前記キー要素が、ストリングであり、そして、
前記キー値の組の前記値の要素が、直列可能な(serializable)エンティティである、
請求項３３に記載の方法。
【請求項３７】
前記トランザクション・インターフェースが、Javaトランザクション・インターフェースを拡張する、請求項３３に記載の方法。
【請求項３８】
前記ファンクションが、トランザクション名を、トランザクションと対応付けるためのsetnameファンクションである、請求項３３に記載の方法。
【請求項３９】
前記setnameファンクションが、それらのトランザクション・タイプによって、トランザクションを名称付けする(name)ために使用される、請求項３８に記載の方法。
【請求項４０】
前記トランザクション・インターフェースが、プロパティ・タイプを、前記特定のトランザクションと対応付けるためのaddPropertiesファンクションを含む、請求項３３に記載の方法。
【請求項４１】
前記インターフェースが更に、以前に前記特定のトランザクションと対応付けられた、プロパティ・タイプの値を設定するためのsetPropertyファンクションを含む、
請求項４０に記載の方法。
【請求項４２】
前記第１のアプリケーション・プロセスが、トランザクションを開始する時に、
特有の(unique)トランザクション識別子が、前記トランザクション・マネジャによって生成される、請求項３３に記載の方法。
【請求項４３】
第１のアプリケーション・プロセスから、第２のアプリケーション・プロセスへの、トランザクションの通信中に、トランザクション・プロパティを、トランザクションと対応付けるシステムであって、
前記第１のアプリケーション・プロセスが、特定のトランザクションと対応付けられるべきトランザクション・プロパティを指定ことを可能とするファンクションを含むトランザクション・インターフェース、
前記トランザクション・プロパティを、前記特定のトランザクションと対応付ける第１のトランザクション・マネジャ、および、
前記対応付けられたトランザクション・プロパティと一緒に前記特定のトランザクションを、前記第２のアプリケーション・プロセスによる使用のために、前記第１のトランザクション・マネジャから、第２のトランザクション・マネジャに、通信する通信メカニズム、
を備えるシステム。

【請求項４４】
第１のアプリケーション・プロセスから、第２のアプリケーション・プロセスへのトランザクションの通信中に、トランザクション・プロパティを、トランザクションと対応付けるためのシステムであって、
トランザクション・リクエストを受信するために構成された通信メカニズム、および、
前記第１のアプリケーション・プロセスから、特定のトランザクションのためのリクエストを受信する第１のトランザクション・マネジャ、および、
前記トランザクション・リクエストに応答する前に、前記第２のアプリケーション・プロセスが、前記特定のトランザクションと対応付けられるべきトランザクション・プロパティを指定することを可能とするファンクションを含む、トランザクション・インターフェース、
を備えるシステム。
【請求項４５】
getおよびsetプロパティが、第１のおよび第２のフェーズで対応付けられる、
請求項１２に記載の方法。
【請求項４６】
getおよびsetプロパティが、第１のおよび第２のフェーズで対応付けられる、
請求項３３に記載の方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は一般的に、アプリケーション・サーバおよびトランザクション・サーバに関連し、特に、トランザクションまたはメッセージにトランザクション・プロパティーを含ませるためのシステムに関連する。
【背景技術】
【０００２】
Java２（Javaは登録商標である。以下同様。）プラットフォーム、エンタープライズ版（Ｊ２ＥＥ）仕様は、多層エンタープライズ・アプリケーションを開発するための現在のスタンダードの一つを規定する。Ｊ２ＥＥは、設計、開発、アセンブリ、および、エンタープライズ・アプリケーションの開発、（これら双方が、コストを削減し、より迅速な設計および実行を可能とする）への、コンポーネント・ベースのアプローチを提供する。Ｊ２ＥＥプラットフォームは、開発者に、多層（multi-tiered）の分配されたアプリケーション・モデル、コンポーネントを削減する能力、統一されたセキュリティー・モデル、および、柔軟なトランザクション制御を与える。それらは、革新的な顧客ソリューションをマーケットに、以前より迅速に提供するだけでなく、結果としての、プラットフォームから独立のＪ２ＥＥコンポーネント・ベースのソリューションは、いかなるベンダーの製品およびアプリケーション・プログラム・インターフェース（ＡＰＩｓ）にも紐付き（tied to）でない。
【０００３】
Ｊ２ＥＥ仕様は、以下の種類のコンポーネントを規定する。即ち、アプリケーション・クライアント・コンポーネント；エンタープライズJavaBeans(EJB)；サーブレットおよびJavaサーバページ（ＪＳＰ）（Webコンポーネントとも呼ばれる）；および、アプレットである。多層の分配アプリケーション・モデルは、アプリケーション・ロジックが、ファンクションに従ってコンポーネントに分割されること、および、異なったアプリケーション・コンポーネントが、同じまたは異なったサーバの上でＪ２ＥＥアプリケーションを作り得ることを示唆する。どこにアプリケーション・コンポーネントが実際にインストールされるかは、多層Ｊ２ＥＥ環境内のどのティアー（tier）に、アプリケーション・コンポーネントが属するかに依存する。多層は、図１に説明される。そこに示されるように、ＥＪＢコンテイナ、及び／又は、アプリケーション・サーバ層（tier）４は、サーブレット、ＪＳＰ、およびhtmlページ14のような、プレゼンテーション・コンテイナを開発するた

めに用いられる。これらは次に、クライアント層2（ここで、クライアント8およびクライアント・アプリケーションが開発される）と、エンタープライズ・リソース・プラニング（ＥＲＰ）システムのような、エンタープライズまたはレガシー・アプリケーションのホスティングのために用いられるバックエンド層6の間のインターフェースとして用いられる。

【０００４】

クライアント層－これらは、ファイヤーウォールの内外の双方の、クライアント層内で走る、ブラウザ、Javaベースのプログラム、または、他のウェブ・エネーブルされたプログラミング環境であり得る。

【０００５】

アプリケーション・サーバ層－通常、この層は、プレゼンテーション・ロジックおよびビジネス・ロジックの組合せをホストして、クライアント・リクエストをサポートする。プレゼンテーション・ロジックは、ＨＴＭＬページをディスプレイするＪＳＰページおよびサーブレットを介してサポートされる一方、ビジネス・ロジックは、リモート・メソッド・インボケーション（ＲＭＩ）オブジェクトおよびＥＪＢ12を介してサポートされる。ＥＪＢは、トランザクション、ライフサイクルおよびステート・マネジメント、リソース・プーリング、セキュリティ、等（これらが、一緒になって、ラン・タイム環境内でビーンズが実行される、当該ラン・タイム環境を作る）のためのコンテイナ環境に依存する。

【０００６】

バックエンド層－これは一般的に、存在するアプリケーションとデータ・ストアの組合せである。これは、エンタープライズ・リソース・プラニング（ＥＲＰ）、メイン・フレーム・トランザクション・プロセッシング、データベース・システム、および他のレガシー情報システムのようなシステムを含み得るので、これは、エンタープライズ情報システム（ＥＩＳ）層とも呼ばれる。

【０００７】

Ｊ２ＥＥアプリケーションのコンポーネントは、別個に走るので、そして、しばしば、異なった装置（device）上で走るので、クライアントおよびアプリケーション・サーバ層コードが、ルック・アップして、他のコードおよびリソースを参照する方法が存在すべき必要がある。クライアントおよびアプリケーション・コードは、例えば、Javaネーミングおよびディレクトリ・インターフェース（ＪＮＤＩ）16を用いて、エンタープライズ・ビーンズのような、ユーザによって規定されたオブジェクト、および、Javaデータベース・コネクタ（ＪＤＢＣ）DataSourceオブジェクトの位置のような環境エンティティー、（これらは、次に、バックエンド層でのリソース、およびメッセージ・コネクションをルック・アップするために用いられる）をルック・アップ（look up）することが出来る。

【０００８】

セキュリティおよびトランザクション・メネジメントのようなアプリケーション・ビヘイビアは、Ｗｅｂおよびエンタープライズ・ビーン・コンポーネントの上で開発時間で構成され得る。この開発時間機能は、アセンブリによって変動し得るコンフィギュレーション設定からアプリケーション・ロジックを切離す。Ｊ２ＥＥセキュリティ・モデルは、システム・リソースが、オーソライズされたユーザによってのみアクセスされるように、開発者に、Ｗｅｂまたはエンタープライズ・ビーン・コンポーネントを構成させる。例えば、Ｗｅｂコンポーネントは、ユーザ名およびパスワードの入力を促すように構成され（configured）得る。エンタープライズ・ビーン・コンポーネントは、特別のグループの人だけが、一定の種類のそのメソッドを呼び出せるように構成されうる。或いは、サーブレット・コンポーネントは、誰にでもアクセス可能なそのメソッドのいくつかを持ち、組織内の一定の特権的な人だけがアクセス可能な少しだけのメソッドを持つように、構成され得る。同じサーブレット・コンポーネントは、誰にでも利用可能な全てのメソッドを持つために、或いは、選択された少数人だけに利用可能な全てのメソッドを持つために、他の環境のために構成され得る。

【０００９】

ＢＥＡシステムズ社（カリフォルニア州、San Jose）の製品たるWebLogicサーバのようないくつかのアプリケーション・サーバは、サーバの上で走るコンポーネントの使用のきめ細かな制御を可能とする、アクセス・コントロール・リスト（ＡＣＬ）メカニズムを用いる。ＡＣＬを用いて、開発者は、Javaメソッド・レベルにおいて、どのユーザまたはユーザのグループによって、何が実行出来るか、出来ないか、を定義出来る。このこのＡＣＬメカニズムは、ＥＪＢｓ（これは、ＥＪＢ仕様で規定された、これら自身のアクセス制御メカニズムを持つ）を除いた、アプリケーション・サーバの上で走る全てををカバーする。セキュリティ領域（realms）は、アドミニストレータが、存在するオーソリゼーション・システムまたは認証システムから、ＡＣＬに情報をインポートすることを可能とする。

【００１０】

＜Javaサーブレット＞

サーブレットは、Ｗｅｂサーバの機能を拡張するプログラムである。サーブレットは、クライアントからリクエストを受け取り、レスポンス（多分、データベースに、リクエストを履行するように問合せする）をダイナミックに生成し、次に、ＨＴＭＬまたはＸＭＬ文書を含むレスポンスを、クライアントに送る。サーブレットは、ＣＧＩに類似するが、サーブレットは、Javaクラスおよびストリームを用いるので、一般的により容易に書くことができる。サーブレットは、Javaバイトコードにコンパイルされ、実行時にサーブレット・インスタンスがメモリに保存され、各クライアント・リクエストが、別個のスレッドで走るので、これらは、より迅速に実行される。サーブレットは、データから、ＨＴＴＰレスポンス・ストリームを、ダイナミックなやり方で生成することを容易にする。各クライアント・リクエストは、新規の接続として実行されるので、フロー制御は、自然にリクエストの間に来ない。このセッション・マネジメントを可能とするために、リクエストの間の特定のクライアントの状態を維持する。いくつかのアプリケーション・サーバにおいて、サーブレットは、ＨＴＴＰセッション・オブジェクトが、メソッド・リクエストの間で、それらの状態をセーブすることを利用する。このオブジェクトは、フェイルオーバの目的のために、クラスター化された環境で複製され得る。

【００１１】

＜Javaサーバ・ページ＞

ＪＳＰページは、サーブレットを開発するための、テキスト・ベースで、プレゼンテーション中心の方法である。ＪＳＰページは、サーブレットの全ての利点を提供し、JavaBeansクラスと結合された時に、コンテントとディスプレイ・ロジックとを別個に維持するための容易な方法を提供する。コモン・ゲートウェイ・インターフェース（ＣＧＩ）に較べて、ＪＳＰページとサーブレットの双方が、より望まれる。何故なら、それらは、プラットフォームから独立で、より少ないオーバーヘッドしか使用しないからである。類似のルック・アンド・フィールを持つページからなるＷｅｂサイトを構築するためのＷｅｂテンプレートを定義するために、ＪＳＰページは、JavaBeansクラスとともに使用され得る。JavaBeansクラスは、データ・レンダリングを実行するので、テンプレートは、Javaコードを持たない。これは、それらが、ＨＴＭＬエディタによって維持され得ることを意味する。ＪＳＰページを用いる、単純なＷｅｂベースのアプリケーションが、JavaBeansクラスの替わりに、カスタム・タグまたはスクリプトを用いて、コンテントをアプリケーション・ロジックにバインド（bind）するために使用され得る。カスタム・タグは、ＪＳＰページにインポートされるタグ・ライブラリにバンドルされる（bundled）。スクリプトレットは、直接ＪＳＰページに埋めこまれた、小さなJavaコード・セグメントである。

【００１２】

＜データベース・アクセス・サービス（ＪＤＢＣ）＞

ＪＤＢＣは、リレーショナル・データベースへの橋として振舞い、ＯＤＢＣ（Open Database Connectivity：オープン・データベース接続性）仕様についてモデル化されている。これは、データベースを、ドライバの使用を通じて、プログラム・コードから切離す。いくつかのＪＤＢＣの実行によって、進んだデータ・タイプのためのサポートが提供され

、スクロール可能な結果の組、および、バッチ・アップデートの機能性（functionality）をもサポートする。

【００１３】

＜Javaメセージング・サービス（ＪＭＳ）＞

ＪＭＳは、Ｊａｖａプログラム間のメッセージの交換をサポートするために用いられるＪ２ＥＥメカニズムである。これは、送り手および受け手が、互いを知る必要がなく、彼らが独立にオペレートできる場合に、いかにしてＪａｖａが、非同期式の通信をサポートするか、である。ＪＭＳは、２つのメッセージング・モデルをサポートする。

（１）ポイント・ツー・ポイントーこれは、メッセージ・キュー（message queues）に基づく。このモデルにおいて、メッセージ・プロデューサは、メッセージをキューに送る。メセージ・コンシューマは、それ自身をキューに添付して、メッセージを聞く（listen）ことが出来る。メッセージが、キューに到達したときに、コンシューマはそれをキューから取り去り、それに応答する。メッセージは、一つのキューにのみ送られ得、一つのコンシューマだけによって使用される。コンシューマは、彼ら（they）が欲する正確なメッセージ・タイプを特定（specify）するためにメッセージをフィルタリングするための選択肢を持つ。

（２）パブリッシュおよびサブスクライブーこれは、プロデューサがメッセージを、トピックに送ること、および、そのトピックについての全ての再登録されたコンシューマに対して、それらのメッセージを検索することを可能とする。この場合には、多くのコンシューマが、同じメッセージを検索出来る。

【００１４】

＜Ｊａｖａインターフェース定義原語（ＩＤＬ：JAVA Interface Definition language）＞

ＣＯＲＢＡオブジェクトは、ＩＤＬを用いて、コントラクト（即ち、いかにして、それらが、他のオブジェクトと相互作用するか）を特定する。JavaＩＤＬを用いると、コントラクトは、Ｊａｖａ世界とＣＯＲＢＡ世界の間で定義され得る。ＳｕｎのＪＤＫ1.2から始めて、ＯＲＢが含まれる。ＯＲＢは、Ｊａｖａアプリケーションが、インターネットＯＲＢ（ＩＩＯＰ）プロトコルを介して、リモートＣＯＲＢＡオブジェクトを呼び出すことを可能とする。

＜エンタープライズＪａｖａビーンズ（ＥＪＢ）＞

ＥＪＢコンポーネントは、ビジネス・ロジックをカプセル化することによって、開発者が、データベース・アクセス、トランザクション・サポート、セキュリティ、キャッシング、および同時実行のような典型的なタスクに対するプログラミング・コードに関わる必要が無いようにするために設計されている。ＥＪＢ仕様において、これらのタスクが、ＥＪＢコンテイナの責任である。エンタープライズ・ビーンは、インターフェースおよびクラスからなる。クライアントは、エンタープライズ・ビーンノホームおよびリモート・インターフェースを通じて、エンタープライズ・ビーン・メソッドにアクセスする。ホーム・インターフェースは、エンタープライズ・ビーンの、生成、除去、および位置特定（locating）のためのメソッドを提供し、リモート・インターフェースは、ビジネス・メソッドを提供する。展開（deployment）時間において、コンテイナは、これらのインターフェースからクラスを生成する。次に、コンテイナは、これらのインターフェースを用いて、エンタープライズ・ビーン上でのビジネス・メソッドの生成、除去、位置特定、およびコール、を求めているクライアントへのアクセスを提供する。エンタープライズ・ビーン・クラスは、ビジネス・メソッドのための実行、メソッドの生成、および、よき細かいメソッド、を提供する。そしてもし、ビーンが、それ自身の持続性（persistence）を取り扱うならば、そのライフサイクル・メソッドのための実行を提供する。

【００１５】

２つのタイプのエンタープライズ・ビーンズが存在する。それらは、エンティティー・ビーンズおよびセッション・ビーンズである。セッション・ビーンは、クライアントとの過渡的な対話を表し、データベースからの読出しおよびそこへの書き込みを実行し得る。

セッション・ビーンは、ＪＤＢＣコール自身を呼び出せるか、または、セッション・ビーンは、エンティティー・ビーンを使用して、コールを為し得る（この場合には、セッション・ビーンは、エンティティー・ビーンに対するクライアントである）。セッション・ビーンのフィールドは、対話の状態を含み、過渡的（transient）である。もし、サーバまたはクライアントが、クラッシュしたならば、セッション・ビーンは無くなる（gone）。

【００１６】

エンティティー・ビーンは、データベース内のデータ、および、そのデータ上で動作するためのメソッドを表す。従業員情報のテーブルのためのリレーショナル・データベース・コンテキストにおいて、テーブル内の各行に対する一つのビーンが存在し得る。エンティティー・ビーンは、トランザクショナルであり、長寿命である。データがデータベース内に残っている限り、エンティティー・ビーンは存続する。このモデルは、リレーショナル・データベースのために容易に用いられ得、オブジェクト・データベースに限定されない。

【００１７】

セッション・ビーンズは、ステートフル（stateful）または、ステートレス（stateless）であり得る。ステートフルなセッション・ビーンは、クライアントのための対話ステートを含む。対話ステートは、セッション・ビーンのインスタンス・フィールド値＋セッション・ビーンのフィールドから到達可能な全てのオブジェクトである。ステートフル・セッション・ビーンズは、永続データ・ストアのデータを表さないが、それらは、アクセス可能で、クライアントのためにアップデート出来る。ステートレス・セッション・ビーンズは、特定のクライアントのための、いかなるステート情報をも持たない。それらは一般的に、いかなる特定のステートをも維持しない、サーバ・サイドのビヘイビアを提供する。ステートレス・セッション・ビーンズは、より少ない、システム・リソースしか要求しない。一般的なサービスを提供する、または、ストアされたデータの共有のビューを表す、ビジネス・オブジェクトは、ステートレス・セッション・ビーンのための良い候補である。

【００１８】

コンテイナ管理された永続を用いるエンタープライズ・ビーンが、リレーショナル・データベースをアクセスする場合には、開発者が、データベース・アクセスに対するいかなるＪＤＢＣ2.0ＡＰＩｓをも使用することも必要とされない。何故なら、コンテイナがこれをハンドルするからである。しかし、もしビーン管理された永続性（bean-managed persistence）が用いられたならば、または、もし、リレーショナル・データベース以外のエンタープライズ情報システムにアクセスする必要があるならば、それを行うための適切なコードが、提供されねばならない。

【００１９】

データベースにアクセスするためにビーン管理された永続性を用いるエンタープライズ・ビーンの場合には、ＪＤＢＣ2.0ＡＰＩコードを持つビーンの寿命メソッドは、データのローディングおよびストアリング、および、ラン・タイムと永続データベース・ストレージ（persistent database storage）との間の一貫性の維持をハンドルするために実行されねばならない。Ｗｅｂティアー（tier）が、ティアーの間でデータを移転するためにＨＴＴＰまたはＨＴＴＰＳを用いる間に、ＥＪＢティアーは、ＲＭＩ－ＩＩＯＰを用いる。ＲＭＩ－ＩＩＯＰは、エンタープライズ・ビーンにアクセスする、いかなるクライアントまたはWebティアープログラムにも、ＥＪＢティアー内のサービスへの直接のアクセスを与える、フルスケールの分配された計算プロトコルである。これらのサービスには、エンタープライズ・ビーンズのリフェレンシング（referencing）のためのＪＮＤＩ、非同期性のメッセージの発信および受信のためのJavaメッセージング・サービス（ＪＭＳ）、および、リレーショナル・データベース・アクセスのためのＪＤＢＣを含む。

【００２０】

＜トランザクション管理＞

WebLogicサーバ・システムのような、何らかのアプリケーション・サーバの最も基本的

な機能の一つは、トランザクション管理である。トランザクションは、そのデータベース・トランザクションが正確に完了したこと、および、それらが、
－最小単位性（atomicity）：トランザクションがデータベースに為す全ての変化が、永続的に為されるか、さもなければ、全ての変化がロールバックである、
－一貫性（consistency）：旨くいくトランザクションが、データベースを、以前の有効な状態から、新規の状態に変換する、
－分離（isolation）：トランザクションがデータベースに為す変化が、トランザクションがその作業を終えるまで、他のオペレータに対して可視的でない、
－耐久性（durability）：トランザクションがデータベースに為す変化が、将来のシステムにおいても、または媒体故障があっても生存する、
を含む、高性能トランザクションの全ての「ＡＣＩＤ」プロパティを持つ（take on）ことを保証するための手段である。

【００２１】

Ｊ２ＥＥトランザクション・モデルは、アプリケーション開発者に、開発時間において、単一のトランザクションを備えるメソッドの間の関係を特定（specify）させることによって、一つのトランザクション内の全てのメソッドが、単一のユニットとして扱われるようにさせる。トランザクションは、全てが完了せねばならない、または、もしそれらが全て完了しないならば全てが逆転される、一続きのステップなので、このことは望ましい。例えば、開発者は、第１のアカウント（account）を借方に記入し、第２のアカウントを貸方に記入することによって、お金を、一つの銀行口座から他に動かすエンタープライズ・ビーンに一続きのメソッドを持つかもしれない。この例において、彼等は、全体の移転オペレーションが、一つの単位として扱われることによって、もし借方記入（debit）の後で、貸方記入（credit）の前に、に失敗が存在するならば、借方記入がロールバックされることを望むであろう。

【００２２】

トランザクション属性は、アセンブリ中に、アプリケーション要素（component）の上で特定されることによって、開発者が、アプリケーション要素に亘って、メソッドをトランザクションにグループ化することを可能とする。このやり方で、アプリケーション要素は、Ｊ２ＥＥアプリケーションで変更され得、トランザクション属性は、コード変更無しに再割当てされ得る。Javaトランザクション・サービス（ＪＴＳ）およびJavaトランザクションＡＰＩ（ＪＴＡ）は、Ｊ２ＥＥでの、そしてより詳細には、ＥＪＢおよびＪＤＢＣ2.0に対するトランザクション・サポートの基礎を形成する。ＪＴＳ仕様は、Javaをオブジェクト管理グループ(ＯＭＧ)オブジェクト・トランザクション・サービスにマッピングする、トランザクション管理のための、低レベルアプリケーション・プログラム・インターフェース（ＡＰＩ）である。ＪＴＡ仕様は、トランザクション処理およびデータベース・システム分野での、先導的な産業界のパートナーとの協力のもと、サン・マイクロシステムによって開発され、トランザクション・マネジャー、リソース・マネジャー、アプリケーション・サーバ、および、トランザクション・アプリケーションの間の標準Ｊａｖａインターフェースを規定する。特に、ＪＴＡは、２つの部分からなる高レベルＡＰＩである。

【００２３】

トランザクション・インターフェース：これは、グローバル・トランザクションによって束縛される(bound by)分散された要素(component)によって為される作業を可能とする。また、これは、トランザクションを構成するオペレーションのグループをマーキングする、または、それらを識別する手段である。

【００２４】

ＸＡリソース・インターフェース：分散されたトランザクションのハンドリングを可能とする、Ｘ／オープン、または、ＸＡインターフェースに基礎を置くインターフェースである。これらは、データベースまたはキューの中、またはその間のような、一つより多いリソースに亘ったトランザクションの調整を伴う。

【００２５】
開発者は殆どの時間、ＪＴＡとのプログラミング明示トランザクション(programming explicit transactions)に関わる必要が無い。何故なら、その作業は、コンテイナによってハンドルされ、アプリケーション配置記述子(application deployment descriptor)によって構成される、ＪＤＢＣおよびＥＪＢ　ＡＰＩを通じて実行されるからである。開発者は、トランザクションの実行に焦点を当てる替わりに、トランザクションの設計に焦点を絞れる。
【００２６】
WebLogicサーバは、エンタープライズ・アプリケーションに対する、分散されたトランザクションと、２相コミット・プロトコルとの双方をサポートするアプリケーション・サーバの例である。分散されたトランザクションは、調整されたやり方で、(データベースのような)複数のリソースメネジャをアップデートするトランザクションである。対照的に、ローカルのトランザクションは、一つのリソース・マネジャをアップデート(更新)する。２相のコミット・プロトコルは、２つまたはそれ以上のリソース・メネジャに亘った一つのトランザクションの調整のメソッドである。それは、トランザクションの更新(updates)が、全ての参加データベース(participating database)でコミットされること、または、全てのデータベースの中から完全にロールバックされることによって、トランザクションの開始の前の状態に戻ること、を確かにすることによって、データの統合性を保証する。言いかえれば、全ての参加データベースが更新されるか、それらの一つも、更新されないかのいずれか、である。分散されたトランザクションは、以下の参加者(participants)を含む。
【００２７】
トランザクション・オリジネータ：これは、トランザクションを開始する。トランザクション・オリジネータは、ユーザ・アプリケーション、エンタープライズJavaBean、またはＪＭＳクライアントであり得る。
【００２８】
トランザクション・マネジャー：これは、アプリケーション・プログラムのためにトランザクションを管理する。トランザクション・マネージャは、アプリケーション・プログラムからのコマンドを調整して、これらのトランザクションに参加している全てのリソース・マネジャーと通信することによって、トランザクションを開始し、完了する。リソース・マネジャが、トランザクション中に失敗した時に、トランザクション・マネージャは、リソース・マネージャが、ペンディングのトランザクションを、コミットするか、ロールバックするかを決定するのを助ける。
【００２９】
回復可能なリソース：これは、データのための永続性の記憶を提供する。リソースは、殆どの場合、データベースである。
【００３０】
リソース・マネージャ：これは、情報およびプロセスの収集へのアクセスを提供する。
トランザクション認知(transaction-aware)ＪＤＢＣドライバは、共通のリソース・マネジャである。リソース・メネージャは、アクションのトランザクション能力および永続性を提供する。これらは、分散されたトランザクション内で、アクセスされ、制御されるエンティティーである。リソース・マネージャと特定のリソースとの間の通信は、トランザクション・ブランチと呼ばれる。
【００３１】
２相コミット・プロトコルの第１のフェーズは、準備フェーズと呼ばれる。要求される更新は、トランザクション・ログ・ファイルに記録され、リソースは、リソース・マネージャを通じて、それが、変更を為すことについて準備完了であることを示さねばならない。リソースは、更新をコミットするためにボート(vote)するか、以前の状態にロールバックするか、のいずれかを行える。第２のフェーズで起こることは、いかにして、リソースがボート(vote)するかに依存する。もし全てのリソースが、コミットするためにボートす

るならば、トランザクションに参加する全てのリソースが更新される。もし一つあるいはそれ以上のリソースが、ロールバックにボートするならば、トランザクションに参加する全てのリソースが、それらの以前の状態にロールバックする。

【００３２】

＜ビジネス・トランザクションへのサポート＞

トランザクションは、以下に例示の状況において適切である(但し、これらの状況は、単に説明目的であって、排他的ではない)。

第１の例として、クライアント・アプリケーションは、複数のオブジェクトへの呼出しを為す必要がある。これには、一つあるいはそれ以上のデータベースへの書き込み動作、の呼出しが含まれ得る。もし一つの呼出しが不成功ならば、(メモリ、更に一般的にはデータベースのいずれかに)書きこまれたいかなる状況が、ロールバックされねばならない。例えば、旅行エージェントのアプリケーションを考慮して欲しい。クライアント・アプリケーションは、例えば、フランスのストラスブルグから、オーストラリアのアリス・スプリングスまでの、遠い場所への旅をアレンジする必要がある。そのような旅は、不可避的に、複数の個々のフライト予約を要求する。クライアント・アプリケーションは、旅の各個々のセグメントを、シーケンシャルな順で(例えば、ストラスブルグからパリ、パリからニューヨーク、ニューヨークからロサンゼルス)予約することによって作動する。しかし、もし何らかの個々のフライト予約が不可能ならば、クライアント・アプリケーションは、そのポイントまでの全てのフライト予約をキャンセルする方法が必要がある。クライアント・アプリケーションは、サーバ・アプリケーションによって管理されるオブジェクトとの対話が必要がある。そして、クライアント・アプリケーションは、特定のオブジェクト・インスタンスの上での複数の呼出しを為す必要がある。対話は、一つあるいはそれ以上の以下の事項によって特徴付けられ得る。

【００３３】

各連続呼出し中に、またはその後に、データは、メモリでキャッシュされるか、データベースに書きこまれる。データは、対話の終わりにおいてデータベースに書きこまれる。クライアント・アプリケーションは、各呼出しの間でメモリ内コンテキスト(context)を維持するためにオブジェクトを必要とする。即ち、各連続する呼出しは、対話に亘ってメモリに維持されているデータを用いる。対話の終了において、クライアント・アプリケーションは、対話中に、または、その最後に、発生したかもしれない全てのデータベースの書き込みオペレーションをキャンセルする能力を必要とする。

【００３４】

代替例として、インターネット・ベースのオンライン・ショッピング・カート・アプリケーションを考慮する。クライアント・アプリケーションのユーザは、オンライン・カタログ中をブラウズし、複数の購入選択を為す。ユーザが、彼らが購入したい全てのアイテムの選択を終わった時に、彼等は、チェック・アウトに進み、購入を行うために、彼らのクレジット・カード情報に入る。もしクレジット・カードの精算が失敗するならば、ショッピング・アプリケーションは、ショッピング・カートでの全てのペンディングの購入選択をキャンセルするか、または、対話中に為された全ての購入トランザクションをロールバックする、メカニズムを必要とする。オブジェクトへの一つのクライアント呼出しの視野内で、オブジェクトは、データベース内のデータに対して複数の編集(edit)を実行する。もし、編集の一つが失敗するならば、オブジェクトは、全ての編集をロールバックするメカニズムを必要とする。(この場合には、個々のデータベース編集は、必ずしもＥＪＢまたはＲＭＩ呼出しである必要はない。アプレットのようなクライアントは、ＪＮＤＩを用いて、Transaction and TransactionManagerオブジェクトへの参照（reference）を獲得し、トランザクションを開始できる。）

【００３５】

他の例として、バンキング・アプリケーションを考慮する。クライアントは、テラー・オブジェクトでトランスファー・オペレーションを呼び出す。トランスファー・オペレーションは、テラー・オブジェクトに、バンク・データベース上で以下の呼出しを行うよう

に要求する。一つのアカウント上でデビット(debit)・メソッドを呼出し、他のアカウント上でクレジット・メソッドを呼び出す。もし、バンク・データベース上のクレジット呼出しが失敗するならば、バンキング・アプリケーションは、以前のデビット呼出しをロールバックするメカニズムを必要とする。

【００３６】

従前の、トランザクション管理のメソッドに伴う上述の問題は、それらが、記述的または機能的プロパティの、特定のトランザクションへの付加、または、それとの対応付け、を可能としないことである。例えば、アプリケーションまたはトランザクション・サーバは、指定のトランザクションに対する文脈上の(contextual)情報を識別(identify)できない。これは、例えば頻繁に飛行機を利用する客であるというような、特定の顧客(および、トランザクション)の識別が不可能で、異なったやり方(多分、より高い優先度)でハンドルされる、これらの特定のトランザクションを持つ、旅行エージェントによって使用されるシステムと相関され得る。呼び出されたオブジェクトは、トランザクション・コンテキストの知識を持たず、そのようなコンテキスト(context)に基づいた知的な決定が出来ない。イー・コマース・アプリケーションの増大する複雑さ、および、イー・コマースの顧客の要求は、これらの要求に応えるために、追加の柔軟さが、トランザクション・メッセージングまたはアプリケーション・システムに取り込まれねばならぬことを決定した。

【発明の開示】

【課題を解決するための手段】

【００３７】

本発明は、複数の対応付けられたトランザクション・プロパティ(properties)を含むトランザクションを可能とするトランザクション処理システムを伴う使用のためのメカニズムを提供する（ここでのトランザクション・プロパティが次に、名称付けられたオブジェクトが、特定のトランザクションと対応付けられることを可能とする）。トランザクション・プロパティは、トランザクションがアクティブならどこでも利用可能であり、トランザクションが、分散されたシステムでの種々のサービスに流れるにつれて、「追加物」または帯域外通信メカニズムとして使用され得る。

【００３８】

ＪＴＡは一般的に、ビジネス・トランザクションに対するサポートを提供する。本発明の実施例によって提供されるＪＴＡの実行は、ビジネス・トランザクションに対する以下のサポートを提供する：

【００３９】

クライアント・アプリケーションが、トランザクションを開始する時に、特有のトランザクション識別子を生成し、トランザクションが表すビジネス・プロセスを記述するオプションのトランザクション名、をサポートする；

【００４０】

トランザクションのためにリソース・マネジャーがアクセスされた時に、リソース・マネジャーに通知する。リソース・マネジャーは次に、アクセスされたレコードを、トランザクションの終了までロックする；そして、

【００４１】

トランザクションが完了した時に、２相コミットを編成する(orchestrates)。これによって、トランザクションでの全ての参加者(participants)が、彼らの更新(updates)を自動的にコミットすること(即ち、全ての更新が実行されるか、更新のいずれもが実行されないか、のいずれか)が保証される。それは、Open GroupのＸＡプロトコルを用いて更新されている、いかなるデータベースとのコミットをも調整し、トランザクションが停止されるべき時にロールバック手順を実行し、失敗が起こった時に回復手順を実行する。それは、どのトランザクションが、クラッシュの時刻時に、マシン内でアクティブであったかを決定し、次に、トランザクションがロールバックされるべきか、コミットされるべきかを決定する。

【００４２】

本発明によって提供されるＪＴＡは、トランザクション・タイムアウト(timeouts)をも管理する：もしビジネス・オペレーションが余りに多くの時間を消費するならばまたは、失敗によって部分的にだけ完了するならば、システムは自動的に、トランザクションに対するタイムアウトを発行するアクションを取り、データベース・ロックのようにリソースを開放する。

【００４３】

一般的に説明すると、本発明の一つの実施例は、第１のアプリケーション・プロセスから、第２のアプリケーション・プロセスへのトランザクションの通信中に、トランザクション・プロパティを、トランザクションと対応付けるためのシステムを備え、当該システムは、第１のトランザクション・インターフェースであって、当該第１のアプリケーション・プロセスが、特定のトランザクションと対応付けられるべきトランザクション・プロパティを特定することを可能とするためのファンクションを含む当該第１のトランザクション・インターフェース、第１のトランザクション・マネジャであって、当該トランザクション・プロパティを当該特定のトランザクションと対応付ける当該第１のトランザクション・マネジャ、および、当該対応付けられたプロパティを持つ当該特定のトランザクションを、第２のアプリケーション・プロセスに通信するための通信メカニズム、を備える。

【００４４】

本発明の他の実施例は、第１のアプリケーション・プロセスから、第２のアプリケーション・プロセスへのトランザクションの通信中に、トランザクション・プロパティを、トランザクションと対応付けるための方法であって、当該方法は、第１のアプリケーション・プロセスにおいて、第１のトランザクション・インターフェースへのコールによって、特定のトランザクションと対応付けられるべきトランザクション・プロパティを特定し、第１のトランザクション・マネジャにおいて、当該トランザクション・プロパティを、当該特定のトランザクションと対応付け、そして、当該対応付けられたトランザクション・プロパティを持つ当該特定のトランザクションを、第２のアプリケーション・プロセスに通信する、ステップを含む。

【発明を実施するための最良の形態】

【００４５】

＜著作権についての注意＞

本特許文書の開示の一部分は、著作権保護に服する材料を含む。著作権者は、特許庁の特許ファイルまたは記録で明らかになる本特許文書または特許開示の、誰によるファクシミリ再製に対しても異論を持たないが、それ以外の場合に対しては、全てのいかなる著作権をも留保する。

【００４６】

＜優先権の主張＞

本出願は、米国特許出願”SYSTEM AND METHOD FOR TRANSACTION PROCESSING WITH TRANSACTION PROPERTY FEATURE”（出願番号　、2002年7月15日出願、）、および、仮出願”SYSTEM FOR TRANSACTION PROCESSING WITH TRANSACTION PROPERTY FEATURE”（出願番号60/306、102、2001年7月17日出願：これは、ここに参照として取りこまれる）、からの優先権を主張する。

【００４７】

＜相互参照＞

本出願は、本出願の基礎出願が同時に米国に出願中の、”SYSTEM FOR TRANSACTION PROCESSING WITH DELIGATED COMMIT FEATURE”（出願番号　、発明者：Edward P. Felt, Priscilla Fung, Alex Somogyi, および Sriram Srinivasan、2002年7月15日出願）および”SYSTEM FOR TRANSACTION PROCESSING WITH SYNCHRONIZED CALLBACK PROCESSING FEATURE”（出願番号　, 発明者：Edward P. Felt, Priscilla Fung, Alex Somogyi, および Sriram Srinivasan, 2002年7月15日出願）、（これらの双方全体が、ここに参照として取りこまれる）に関連する。

【００４８】
広く説明すると、本発明は、一つあるいはそれ以上の対応付けられたトランザクション・プロパティを含ませるためのトランザクションを可能とする、トランザクション処理またはメッセージング・システムでの使用のためのシステムおよび方法を提供する。トランザクション・プロパティは、名称付けられたオブジェクトが、特定のトランザクションと対応付けられることを可能とするために使用され得る。このやり方で、トランザクション・プロパティ(properties)は、分散されたシステムでの異なったサーバ間のトランザクション・フローとして、サービスからの、および、サービス間の、情報を提供する帯域外(out-of-band)通信メカニズムとして使用され得る。ここに記載された種々の実施が、本発明を、Javaに関して、特にＥＪＢ実行について説明する一方、当業者にとって、ここに記載されたトランザクション・プロパティ・メカニズムが、他のシステムおよび他の実施についても使用され得ることは明白である。

【００４９】
図２は、本発明の一つの実施例によるトランザクション・プロパティのオペレーションの概略的説明を示す。図２に示されるように、クライアント・アプリケーション104を含むクライアント102は、トランザクションによっていくつかのサーバと通信する。図２に示されるように、この複数のサーバは、サーバＡ106、サーバＢ108、およびサーバＣ110を含む。通常の環境において、サーバＡ，ＢおよびＣの間を流れねばならないデータに関連するいかなるトランザクションも、帯域内(in-band)通信を用いて通過されねばならない。つまり、情報は、一つのサーバから、良く規定されたインターフェースの使用を通じて、他のサーバに明示的に(explicitly)通過させられる。本発明はしかし、トランザクション関連データの帯域外通信を可能とする。即ち、情報は、実際の情報のフローで、明示的に通過させられ得、インターフェースを介して明示的に送られる必要がない。図２に示されるように、例えばクライアント・アプリケーション104は、サーバＡ106上のリソースを使用するために、トランザクション112のためにリクエストを為す。トランザクション・コンテキストの一部分として、トランザクション・プロパティ114は、トランザクション112と対応付けられ得る。いかなる特定のトランザクションも、一つ以上のサーバを伴い得る。図２の例において、トランザクションは、サーバＡ106だけでなく、サーバＢ108、およびサーバＣ110をも含む。トランザクションが、サーバＡからサーバＢに流れ、次に、サーバＣに流れるにつれて、そのトランザクションと対応付けられたプロパティは、いずれかのサーバによって修正(modified)及び／又は読み取られ得る。例えば、図２に示されるように、トランザクション・プロパティ114を含むトランザクション112が、サーバＡからサーバＢに通過させられた時に、追加のトランザクション・プロパティ116が付加され得る。結合されたトランザクション・プロパティ120は現在、トランザクション・プロセスの一部分としてサーバＣに通過させられ得る。同時に、更新されたトランザクション・プロパティ120は、通過させられ、または、サーバＡに返送され122得る。これらの更新されたトランザクション・プロパティは、オリジナルのコール(サーバＡからサーバＢへの)がサーバＡに戻る時にも、自動的にサーバＡに返送される。このやり方で、本システムは、クライアント・アプリケーションがトランザクション・プロパティ(これは、分散されたシステムにおける、異なったサーバまたはモジュールに亘って伝搬される)を設定するための能力を提供する。この効果は、その特定のトランザクションによって影響を受けるソフトウェア要素(components)のための帯域外通信メカニズムを提供することである。このメカニズムを記述する追加の実施は、以下に更に詳細に説明される。

【００５０】
従って本発明は、いくつかの対応付けられたトランザクション・プロパティを含ませるためのトランザクションを可能とする、何らかのトランザクション処理システムでの使用のためのメカニズムを提供する。トランザクション・プロパティは次に、名称付けされたオブジェクトが、トランザクションと対応付けられることを可能とする。トランザクション・プロパティは、トランザクションがアクティブな時にはいつでも利用可能である。トランザクション・プロパティは、トランザクションが流れるに際して、またはそれが、分

散されたシステム内の種々のサーバの間で通信されるに際しての、付加物(attachment)または帯域外通信メカニズムとして使用され得る。

【００５１】

本発明の一つの実施例は、スタンダードＪＴＡ仕様が、詳細な実施をカバーしない場合に、または、追加の能力が要求される場合に、スタンダードJavaトランザクションＡＰＩへの拡張を提供する。本発明は、クライアントによって開始されたトランザクション(client-initiated transactions)を可能とする能力をも提供する。標準のＪＴＡトランザクション・メネジャ・インターフェース(javax.transaction.TransactionManager)は、Java Named Directory Interface (J}N:Java名称付けディレクトリ・インターフェース)を通じて、クライアント、およびビーン・プロバイダに利用可能である。このインターフェースは、ビーン管理されたトランザクション(bean-managed transactions)を使用しているクライアントおよびＥＪＢｓに、トランザクションを停止して再開することを可能とする。トランザクション－トランザクション(transaction-transaction)の視野は、クラスタ内と、クラスタの間、の双方で作動可能である。

【００５２】

図３は、本発明を利用できる一般的なシステムの概略を示す。共通にクライアント・アプリケーションと呼ばれる、リモート装置(device)またはアプリケーション202は、アプリケーション・サーバ204と、或いはより一般的には、アプリケーション・サーバでホストされたアプリケーションと通信する。クライアント・エンド(client end)において、トランザクション・メネジャー206は、クライアント・アプリケーションによって使用されて、トランザクション208、212を、インターフェース207を介して、サーバに送り、そこから受け取る(或いは多くのサーバへ送り、および、そこから受け取る)。同様に、サーバ・エンドにおいて、トランザクション・マネジャ210およびインターフェース211が用いられて、クライアントとのトランザクションを管理(manage)する。一つの実施例において、いかなるサーバも、トランザクションを開始するための、トランザクション・コーディネータ214, トランザクション状態(status)を監視(monitor)するためのサブ・コーディネータ216, または、トランザクションをロギングするためのトランザクション・ロッガー(logger)218, として動作出来る。サーバのグループによってではなく、その替わりに、これらのタスクの各々が、同じ(即ち一つの)サーバによってもハンドルされ得ることは明白である。トランザクションが発生するにつれて、それらは、後の監査(audit)、参照(reference)、またはロールバック目的のためにトランザクション・データベース220に記録され得る。本発明は、プロパティ(properties)が各トランザクションと対応付けられることを可能とするトランザクション・マネジャを提供する。クライアントまたはサーバ・アプリケーションは次に、トランザクションが通信される222に際して、これらのプロパティを解析でき、それに対応して反応できる。

【００５３】

この機能(function)を達成するために、本発明の一つの実施例は、スタンダードＪＴＡへの拡張(extensions)として、クラスおよびインターフェースの組を提供する(新規のクラスをスタンダードＪ２ＥＥ／ＪＴＡクラスから区別するために、説明目的のために、以下の例のいくつかで名称「Weblogic」が使用されるが、本発明は、特にWeblogicシステムに限定されず、以下に示される例示のクラスまたはインターフェースに限定されないことに留意してほしい)。

【００５４】

＜トランザクション・プロパティ＞

図４は、本発明の一つの実施例によるトランザクション工程(process)のフローチャートを示す。図４に示されるように、クライアントのトランザクション・マネジャは、クライアント・アプリケーションからリクエストを受け取り、トランザクションを生成または開始(initiate)する(ステップ302)。いくつかの場合には、実際には異なった参加クライアント(participating client)アプリケーションであり得る、クライアント・アプリケーションもまた、ステップ304において、そのトランザクションと対応付けられるべきプロ

パティを生成する。
【００５５】
ステップ306において、クライアント・アプリケーションまたはいくつかの外部アプリケーションが、トランザクションのためのプロパティを生成する。クライアント・トランザクション・マネジャは、クライアント・アプリケーションからリクエストを受け取り、トランザクションを生成し、何らかの所望のプロパティを、トランザクションと対応付ける(ステップ308)。ステップ310において、トランザクションは、対応付けられたプロパティと一緒にサーバに送られる。トランザクション・マネジャは、サーバにおいてトランザクションを受け取る(ステップ312)。この時点でトランザクション・マネジャは、何らかの対応付けられたトランザクション・プロパティを受け取る(314)。トランザクションと対応付けられるべき発見された何らかのプロパティは、トランザクションを修正(modify)または増加(augment)させるために用いられ(ステップ316)、トランザクションは、サーバ・アプリケーションに渡される(ステップ318)。
【００５６】
一つの実施例のトランザクション・プロパティは、キー(key)／値(value)の組が、対応付けられた値(value)とともに一つあるいはそれ以上のキー(key)を含むという意味での、キー(key)／値(value)の組である。一旦、トランザクション・プロパティが設定されると、その全寿命(lifetime)の間中それは、トランザクソｈンと対応付けされ続ける。トランザクションが、システムに亘って移動するにつれて、プロパティは、マシン(コンピュータ)の間を伝搬される。プロパティは、トランザクション・ログにもセーブされ、クラッシュ回復処理中に復元(restore)され得る。本発明の一つの実施例において、もしトランザクション・プロパティが一回以上設定されるならば、直近の過去の値が維持される。しかし、いくつかのスレッドされた(threaded)及び／又はマルチ・サーバ状態において、このビヘイビヤは。望ましくない、或いは、予想可能かもしれない。
【００５７】
回復(recovery)および監査追跡(audit-trail)手順(procedures)は一般的に、トランザクション・タイプに関する情報，トランザクション識別子，ロールバックについての特定の理由，および何らかのヒューリスティック・エラー・メッセージ，のような、ＪＴＡトランザクション・インターフェースが装備するものに較べて、トランザクションについてのより多くの情報を必要とする。トランザクションによって費やされる時間、および、残された時間の総計は、一定のオペレーションについてのリミットとして使用されうる。
【００５８】
本発明の一つの実施例によって、トランザクション・プロパティを設定するために、getProperty/setPropertyインターフェースが、トランザクション・サービス(または、クライアント・アプリケーション)のユーザによって使用されて、トランザクション・コンテキストとともに余分の情報が伝搬する。キーは、ストリングス(strings)であり得る一方、値は、いかなる、直列可能な(serializable)エンティティーでも有り得る。例えばワード「weblogic」で始まるもののような、いくつかのキーは、リザーブされ得るか、または、特別の意味を持ち得る。他の物は、トランザクションがシステムを通じて流れるにつれて、サービスによって解釈されずに、トランザクションとともに伝搬され、トランザクション媒介者(intermediaries)および受け取り人(recipients)に利用可能となる。
【００５９】
＜トランザクション・インターフェース＞
トランザクション・インターフェースは、トランザクション・プロパティ機能(transaction property features)の機能化(functioning)を制御するために用いられる。以下のインターフェースは、それがWeblogicサーバシステムに適用され得るような形での、本発明の特定の実行を示す。本発明の精神および視野を維持しつつ、以下に記載されるインターフェースおよび技術が、他のシステムでの使用のために修正され得ることは当業者に明白である。
【００６０】

＜weblogic.transaction.TransactionManager＞
このTransaction Managerインターフェースは、Ｊ２ＥＥスタンダードjavax.transaction.Transaction Managerインターフェースを拡張し、それは、ＪＴＡトランザクション・マネジャ・オブジェクトによってサポートされる。それは、スタート。アップ(startup)に際して、ＸＡリソースが、トランザクション・マネジャで、それら自身をレジスタし、アン・レジスタすることを可能とする。TransactionManagerは、何らかのアプリケーション・サーバ特有のクラスをインポートせねばならぬことを避けるために、ＪＮＤＩを通じて参照(referenced)され得る。

```
Javax.naming.Context ctx = getInitialContext();
TransactionManager tm = (TransactionManager)
ctx.lookup (”javax.transaction.TransactionManager”);
tm.begin();// begin a transactlon.
Transaction tx = tm.getTransaction();
tm.commit();
```

【００６１】
トランザクション・マネジャは、キャッシュされ、全てのアプリケーション・スレッドによって使用され得る。このやり方によって、ＪＮＤＩルックアップは、プロセスの寿命(lifetime)中に一回だけ為される必要がある。UserTransactionの獲得および使用は、上と同じパターンを辿る。それは、ＪＮＤＩから得られることができ、引き続き、キャッシュされ、再使用される。

```
javax.naming.Context ctx = getInitialContext();
UserTransaction ut = (UserTransaction)
ctx.lookup (”javax.transaction.UserTransaction”);
ut.begin(); // begin a transaction.
ut.commit ();
```

【００６２】
＜weblogic.transaction.Transaction＞
上述の、weblogic.transaction.Transactionインターフェースは、スタンダードjavax.transaction.Transactionを拡張するトランザクション・インターフェースであり、ユーザまたはアプリケーションが、トランザクション・プロパティを獲得(get)し、及び／又は、設定(set)することを可能とする。一つの実施例において、トランザクション・インターフェースは、以下のメソッドを含む：
【００６３】
＜setName(java.lang.String n)＞
setName()メソッドまたはファンクションは、名称(name)を、トランザクションと対応付ける。トランザクションの名称は、トランザクションの、アプリケーション特有の意味的な意義(semantic meaning)(「タイプ」)への参照(reference)とされることが意図され、トランザクションの全ての個々のインスタンスに対して特有であることは意図されない。このように、トランザクション名は、ビジネス・トランザクションのクラスを表すと考えられ得る。トランザクションへの名称を想定することは、より良いエラー・レポート(error reporting)を与える。内部的に生成されたエラー・レポートは、もし名称が存在するならば、名称を取り込む。例えば、トランザクションに名称”funds transfer”を与えるために、金融サービスオペレーションで、ファンクションtx.setName("funds transfer")が使用され得る。
【００６４】
＜getName()＞
getName()ファンクションは、以前に、トランザクションと対応付けられた名称を検索する(即ち、それは、トランザクション名を戻す)。

【００６５】
＜addProperties(java.util.Map props)＞
addProperties()ファンクションは、プロパティ(element)の各要素(properties)についてsetPropertyをコールすることによって、プロパティの組をトランザクションに加える。それは、パラメータとして、複数のプロパティ(即ち、加えるべきキー／値の組のセット)を受け入れる。
【００６６】
＜setProperty(java.lang.String key, java.io.Serializable value)＞
setProperty（）ファンクションは、トランザクションと対応付けられるべきプロパティ値を設定する。このファンクションは、パラメータとして、キー－(ストリング・キー)、および値－(プロパティ値、ここで、ヌル(null)の値は、プロパティを削除することを意味する)を受け入れる。
【００６７】
＜getProperty(java.lang.String key)＞
getProperty() ファンクションは、特定のプロパティの値を獲得(get)し、パラメータとして、キー－(プロパティ名)を受け入れる。ファンクションは、プロパティの値を戻すか、或いは、もし設定されていない(unset)ならば、ヌル(null)を戻す。
【００６８】
＜getProperties（）＞
getProperties（）ファンクションは、現在、トランザクションと対応付けられている全てのプロパティを得る(gets)。本ファンクションは、全てのキー／値の組を包含するオブジェクトを戻す。
【００６９】
＜setRollbackOnly(java.lang.String msg, java.lang.Throwable reason)＞
setRollbackOnly() ファンクションは、ロールバック・ストリング・メッセージ、および、NestedExceptionインスタンス内への除外(exception)をラップ(wrap)し、この新規の除外でsetRollbackOnlyをコールする。
【００７０】
＜setRollbackOnly(java.lang.Throwable reason)＞
このfunction()は、行われる何らかのロールバックのための理由(reason)を提供する。トランザクション・サービス内部でトリガされた全てのロールバックが、理由(タイムアウト、ＸＡエラー、beforeCompletionでのチェックされていない除外、コーディネータ(coordinator)にコンタクトする能力欠除、等)を設定する。このファンクションは、パラメータとして、理由(ロールバックの背後の理由)のための値を受け取る。
【００７１】
＜getRollbackReason＞
getRollbackReason（）ファンクションは、(もし存在するなら)、トランザクションと対応付けられたロールバックの理由を検索し、何らかのコーリング・アプリケーションにロールバック・リーズン(rollback reason)を戻す。
【００７２】
＜getHeuristicErrorMessage()＞
getHeuristicErrorMessage()ファンクションは、(もし存在するなら)、トランザクションと対応付けられたヒューリスティックなエラー・メッセージを戻す。
【００７３】
＜getXID()＞
getXID()ファンクションは、トランザクション識別子を得る。いくつかの場合において、トランザクション・プロセスは、それがリソースをコールする時に、いくらかの余分な情報を、Ｘｉｄの修飾子(qualifier)に加え得る。従って、これに引き続く、(最初に)リソースをコールする前の、tx.getXID().getGlobalTransactionId()へのコールおよび、次に、リソースをコールした後のtx.getXID().getGlobalTransactionId()へのコールは、実

際に戻されたものにおいて、異なり得る。
金融サービス・シナリオの例において、トランザクション識別子(ｘｉｄ)は、トランザクションを開始するブランチを示すブランチ修飾子を含み得る。トランザクションが、他のブランチに進められる時に、トランザクション自身のコンテキストへの修正無しに、この修飾子は修正され得る。いくつかの実施例において、システムによって、いくつかの保証が提供される。
1.xidequals()xid影響を受けない。tx.getXID()から得られたxidおよび、後の時間での他に得られたxidは、(同じトランザクションに対して)等しいと考えられる。この保証は、xidの成分(constituents)、即ち修飾子には拡張されない。
2.全ての、そのＸＡResourceメソッドにおいて、ＸＡリソースは常に、所定のトランザクションに対して、(同じ修飾子を持つ)同じxidを供給される。
【００７４】
＜getStatusAsString＞
getStatusAsString（）ファンクションは、トランザクションの状態を、ストリングとして戻す。このストリングは、(「ロギング」，「事前準備」、等のような)マイナーな状態で更に修飾されたjavax.transaction.Statusで指定された主要な状態を含む。
【００７５】
＜getMillisSinceBegin()＞
getMillisSinceBegin() ファンクションは、ミリセカンドで、トランザクションが存在した時間の総量を戻す。
【００７６】
＜getTimeToLiveMillis()＞
TimeToLiveMillis() ファンクションは、ミリセカンドでの、トランザクションがこれから生存を許された時間の総量を戻す。トランザクションがアクティブで、それがコミットまたはロールバックを再トライしていない時にのみ、この時間は、大きい意味を持つ。
【００７７】
＜weblogic.transaction.TxHelper＞
TxHelperクラスは、ユーザまたはアプリケーションが、現在のトランザクション・マネジャおよびトランザクションを得ることを可能とする。

```
import weblogic.transaction.TxHelper;
TransactionManager tm = TxHelper.getTransactionManager();
Transaction tx = TxHelper.getTransaction();
```

トランザクション・オブジェクトは、他のコンテキスト的なデータの輸送のために、および、詳細な状態情報を得るために、Transactionインターフェースの拡張に放り出され(cast)得る。

```
import weblogic.transaction.Transaction;
Transaction tx = (Transaction) TxHelper.getTransaction();
```

【００７８】
＜weblogic.transaction.XAResource＞
このインターフェースは、ＸＡリソースのためのデリストメント(delistment)能力を提供する。ＸＡリソース・インターフェースは、エンリストメント(enlistment)、デリストメント、準備(prepere)、コミット、ロールバック、および、リカバリ(recovery)、を含む２相コミット・プロトコル(Two Phase Commit protocol)の参加(participation)でのリソース・コンタクトをも表す。ＪＴＡ仕様は、いかにして、トランザクションのリソースが初期化されるか、或いは、いかにしてユーザまたはアプリケーションが、TransactionManagerまたはUserTransactionインターフェースを実行するオブジェクトを得ることが期待されるか、のような一定の詳細を標準化しない。以下のセクションでは、そのような詳

細を開示し、更に、例によって、本発明によって提供され得るTransactionおよびTransactionManagerインターフェースへのいくつかの拡張を説明する。

図５は、本発明の実施例はによるトランザクション・プロパティを含むトランザクション402の表現の説明を示す。ここに示されるトランザクション・フォーマットは、単に、金融サービス環境で使用され得る一つの例であるに過ぎず、また、フォーマットＩＤ404，グローバルＩＤ406，コーディネータ410、タイムスタンプ412、カウンタ414、ブランチ・クオリファイヤ416、およびリソース名418を含む。このトランザクション・フォーマットは、プロパティまたはクオリファイヤを、恐らくそのトランザクションを特定の顧客のクラス(即ち、例えば「ゴールド」または「プラチナ」レベルの顧客)に属するものとして区別するために、または、起点の(originating)または中間の銀行支店(bank branch)を識別するために、特定のトランザクションに追加するために使用され得る。当業者にとって、ここに示される例示のフォーマットが、単に、一つの実施例の説明目的であること、および、本発明の精神および視野内に維持しながら、他のフォーマットが、使用され得ることは明白となろう。

【００７９】

図６は、本発明の実施例による、種々のトランザクション・プロパティ・フィールドの説明を示す。プロパティには、次のデータのフォーマット・バージョン・ナンバー502、トランザクション識別子504、生存時間(Time to Live)506、システムによって規定されたプロパティ・フィールド(System_Defined Properties field)508、他のプロパティ510、リソース・マネジャ名512、登録された同期コールバックを持たないサブ・コーディネータ名514、および、登録された同期コールバックを持つサブ・コーディネータ名516、が含まれる。これらのフィールドのいくつかは必要となり得るか、或いは、オプションのフィールドと考えられ得る。

【００８０】

＜TransactionManagerおよびUserTransactionへの参照＞

ＪＴＡ仕様がこれらのインターフェースを規定する一方、それは、現在のトランザクションにおいて得るための方法を命令しない。ＥＪＢ仕様だけが、ビーンを用いてそれを行うための方法を提供する。本発明を用いると、これらのインターフェースへの参照を得るために、いかなる特定のクラスをもインポートする必要は無い。

【００８１】

＜クライアントによって開始された(Client-Initiated)トランザクションのためのサポート＞

ＪＴＡ仕様は、アプレットのようなクライアントが、TransactionManagerへの参照を獲得し、トランザクションを開始できるか否かを明確に規定しない。本発明を用いるとこれが可能となる。Ｊ２ＥＥスタンダードは、Ｊ２ＥＥクライアントが、ＪＮＤＩを用いてUserTransactionオブジェクトへの参照をポータブルに(portably)得ることを可能にする。本発明を用いると、この機能(facility)は、TransactionおよびTransactionManagerオブジェクトの双方に対しても利用可能である。

【００８２】

＜クラスタと、クラスタされていないサーバをつなぐ(span)能力＞

本発明は、トランザクションが、リモート・メソッド・コールが可能であればどこにおいても、転送される(forwarded)ことを可能とする。特定のトランザクションの視野内でタッチされた(touched)全てのサーバは、セキュリティ設定または好み(preferences)によって課せられた何らかの制約内で、コーディネート可能である。

【００８３】

＜ユーザによって定義されたデータ・セクション(User-defined Data section)＞

トランザクション・コンテキストは明確に制御のスレッドに従うので、そして、トランザクション境界は一般的に、原子ビジネス・タスク(atomic business task)を表すので、本発明は、トランザクションを、それらのタイプ(例えば、「ファンズ・トランスファー(funds transfer)」)によって名称付けする方法、および、コンテキストとともにユーザに

よって定義されたデータ構造を運ぶ方法を提供する。この情報は次に、エラー・リポーティングおよび統計収集理由(statistics-gathering reasons)のために使用され得る。

【００８４】

＜エラー報告(error reporting)＞

そのエラー報告を改善するために、本発明の実施例によって、ＪＴＡサービスは、トランザクション・ロールバックのための、または、setRollbackOnly()またはrollback()をコーリングする前に理由を与えるための、根底にある理由を獲得方法を持つアプリケーションを提供できる。スタンダードのJava仕様ＪＴＡサービスは、この要求に応えない。本発明は、トランザクション識別子および詳細な状態レポートを得て、全てのメセージが、ビジネスレベルで意味を持つための十分な詳細を含むことを確実にすることをも可能とする。殆どの伝統的トランザクションのシステムは、単に、トランザクション識別子およびそれらの状態の組をリストするだけであろう(これは、オペレータに対して余り意味を持たないか、或いは全く意味を持たない)。

【００８５】

＜beforeCompletion中のビーンズ呼出しのためのサポート＞

スタンダードＥＪＢも、スタンダードＪＴＡ仕様も、いかにして、ビーンのbeforeCompletionまたはejbStoreメソッドが、他のビーン(このbeforeCompletion及び／又はejbStoreメソッドが既にコールされている)を呼び出す場合に、これらのケースをハンドルする(handle)かを定義しない。本発明は、複数のサーバに亘って、そのようなシナリオをサポートする。beforeCompletion（）コールバック・メソッドで走るアプリケーション・コードの結果として接触された(contacted)新規のサーバまたはリソースが、追跡される(tracked)。システムは次に、これらのサーバおよびリソースが、トランザクション・コミット処理に完全に参加する(必要な際に、追加のbeforeCompletion（）コールバックを作成することを含む)ことを保証する。

【００８６】

＜ＥＪＢインタラクション＞

管理されたトランザクションを生成するためにＥＪＢを用いる時に、ＥＪＢがトランザクションを開始するときにはいつでも、ＥＪＢコンテイナは自動的に、トランザクション名を設定する。もしディプロイヤ(deployer)が、ディプロイ記述子(deployment descriptor)内の名称を提供していならば、ＥＪＢサービスは、トランザクションのデフォールト名としてのclass.method名を提供する。ＥＪＢコンテイナが、トランザクションを開始するときにはいつでも、ＥＪＢコンテイナは自動的に、トランザクション・タイムアウトを設定できる。もしディプロイヤが、ディプロイ記述子にタイムアウト値を提供しなかったならば、ＥＪＢコンテイナは、タイムアウトをゼロに設定する。この場合には、構成パラメータ(configuration parameter)に基づくデフォールトのタイムアウトが使用される。ＥＪＢディプロイメント記述子は、ＥＪＢコンテイナによって開始された何らかのトランザクションのために設定されたプロパティを指定できる。設定され得る一つの有用なプロパティは、トランザクション追跡プロパティである。追跡プロパティは、追跡(tracing)がサーバで開始され、次に、トランザクションを伴ってネットワークに亘って、第２のサーバに流れることを可能とする。追跡は、一つのトランザクションと対応付けられた制御のスレッドにとってのみ特有である。このことは、ビジーなプロダクション・システムにおける利点である。追跡出力は、各サーバのエラー・ログ・ファイルで印刷され、それは、アプリケーション・コード、システム・コード、リソース、または、構成設定(configuration setting)のいずれかにおける問題の診断のために有用である。

【００８７】

＜ＪＭＳインタラクション(interaction)＞

ＪＭＳは、トランザクション名、タイムアウト、およびトランザクション・プロパティ、を制御するために、構成パラメータを受け取る。

【００８８】

＜実行(implementation)の例＞

図７は、本発明を用いて使用するための、または、それを取り込む、システムの例を示す。ここに示されるように、クライアントまたはクライアント・アプリケーション602は、トランザクション607を生成する。他のクライアント・アプリケーション604、またはサーバ・プロセスまたはファンクション606は、トランザクションの修正、または、増加(augment)(多分、非常に重要な顧客に属するものとして、または、頻繁にフライトする顧客として、トランザクションを識別するプロパティ608の追加)を希望するかもしれない。クライアントとの通信状態で、クライアントで走っているプロセスか中間サーバ上のプロセスかのいずれかとして作動する、トランザクション・マネージャ・プロセス610が、例えば、setPropertyコマンドまたはセット・プロパティ・インターフェースを用いることによって、プロパティを、トランザクションと対応付けるために用いられる。対応付けられたプロパティ612とともに、トランザクション607は、次に、レシピエント(recipient)・サーバ(または複数のサーバ)614に渡される。ここで、他のトランザクション・マネージャ616が、例えば、getPropertyインターフェースを用いることによって、対応付けられたトランザクション・プロパティを検索する。対応付けられたプロパティに依存して、受信レシピエント・サーバまたはプロセスは、それらのニーズに適合させるために、複数の異なった方法(618、620、622)でのトランザクションを起因として作動し得る。

【００８９】

図８は、本発明の実施例によるトランザクション・プロパティの組の設定のためのサンプル・コードのリスト702を示す。再度述べるが、この例もまた、金融サービス環境の中でトランザクションが、恐らくアカウントのクレジット(account credits)、デビット(debit)、株売却、等、に影響を与えるために、クライアント・アプリケーションとサーバ・アプリケーションの間を、行ったり来たりする、当該金融サービス環境に関連する。図８に示される例において、クライアント・アプリケーションは、setPropertyファンクションを用いて、貴重な顧客のプロパティ(ＶＩＰ：valued customer property)(「評価」＝「ＶＩＰ」)を、特定のトランザクションに対応付ける。この「ＶＩＰ」プロパティは、全システムを通じて、トランザクションがクライアントからサーバに流れるにつれて、或いは多分一つのサーバから他のサーバに流れるにつれて、トランザクション・プロパティとしてトランザクションとともに移動する。

【００９０】

図９は、本発明の一つの実施例による、トランザクション・プロパティの組を検索するための、サンプル・コードのリスト802を示す。この例において、以前に、トランザクションと対応付けられた「ＶＩＰ」トランザクション・プロパティは、例えば、getPropertyへのコールを用いて、サーバ・アプリケーションによって検索される。ここで、サーバ・アプリケーションは、トランザクションのオリジネータ(originator)またはオーナー(owner)のプロパティを識別または増加する「評価(rating)」のための値を検索する。図９に示される例において、「評価」の値が、「ＶＩＰ」と等しいことは、トランザクションが、特別の扱いを受けるべきことを示す。サーバ・アプリケーションは、このトランザクション・プロパティに対して、恐らく、トランザクション・リクエストを、より高い優先度で処理することによって、或いは、トランザクションを、「ＶＩＰ」トランザクションだけをハンドルするために供された特別のサーバに渡すことによって、適切に応答できる。

【００９１】

本発明の以上の説明は、説明と記述の目的のために提供されてきた。これが排他的であること、または、本発明を開示された正確な形式に限定することは意図されない。明らかに、多くの修正および変更が、当業者にとって明白であろう。実施例は、本発明の原理およびその実際の応用を最も良く説明することによって、当業者が、種々の実施例について本発明を理解し、考えられる特定の使用方法に適した種々の修正を持つ本発明を理解することを可能とするために、選ばれ、記載された。本発明の視野は、添付の請求項およびその均等物によって規定されることが理解される。

【図面の簡単な説明】

【００９２】
【図１】Ｊ２ＥＥアーキテクチャの説明を示す。
【図２】本発明の実施例によるトランザクション・プロパティ機能の説明を示す。
【図３】本発明を用いた使用のための一般的なシステムの説明を示す。
【図４】本発明の実施例によるトランザクション・プロパティプロセスのフローチャートを示す。
【図５】本発明の実施例によるトランザクション・プロパティを含むトランザクションの表現の説明を示す。
【図６】本発明の実施例による種々のトランザクション・プロパティフィールドの説明を示す。
【図７】本発明の実施例の使用のための、または、それを取り込んだシステムの例を示す。
【図８】本発明の実施例によってトランザクション・プロパティの組を設定するサンプル・コードのリストを示す。
【図９】本発明の実施例によるトランザクション・プロパティの組の検索のためのサンプル・コードのリストを示す。

【図１】

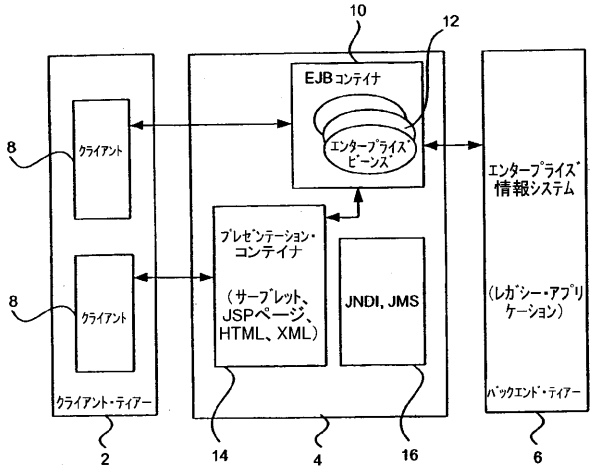


FIGURE 1

【図２】

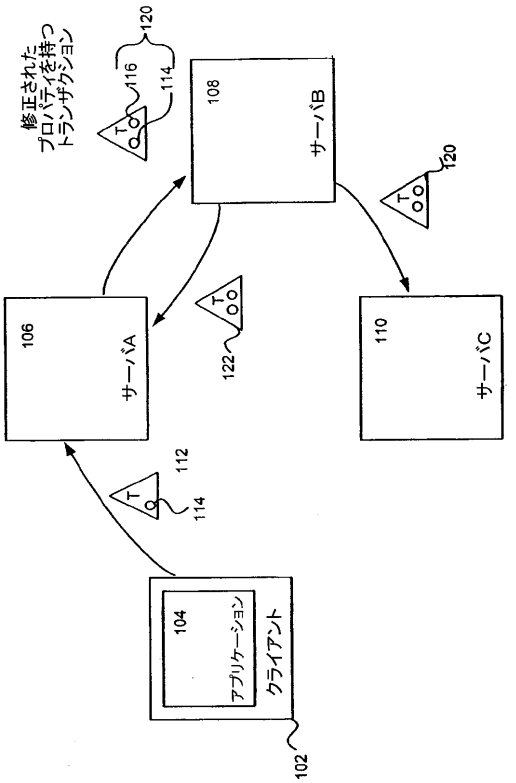


FIGURE 2

【図３】

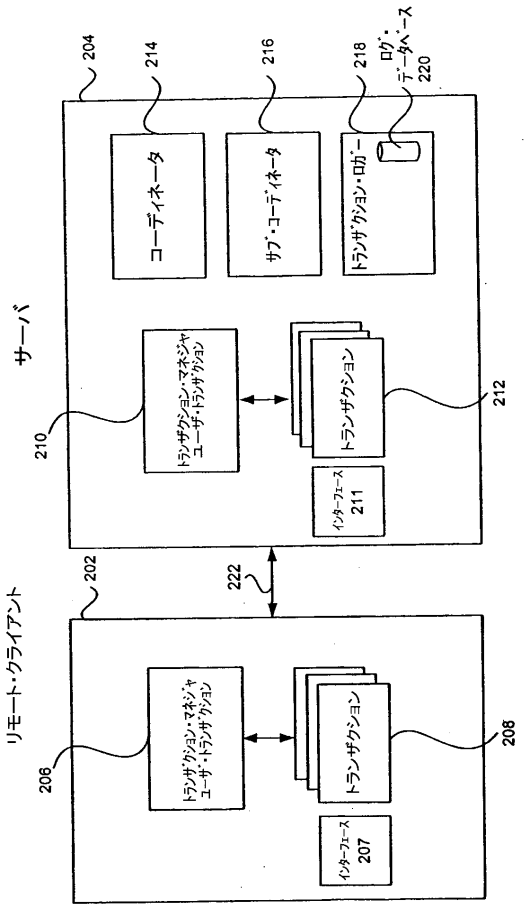


FIGURE 3

【図４】

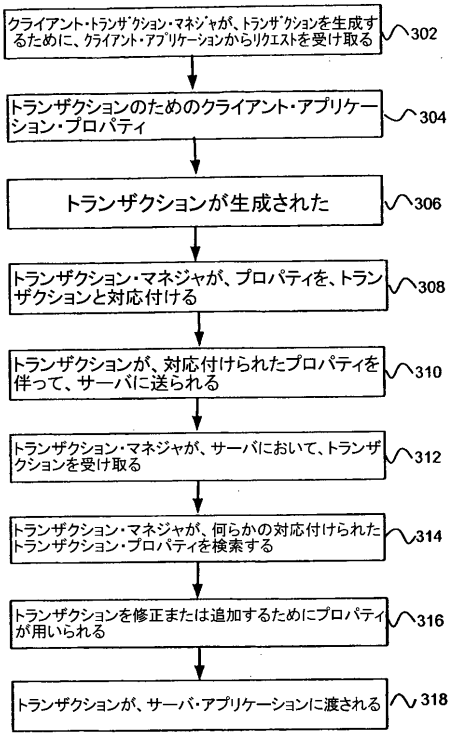


FIGURE 4

【図５】

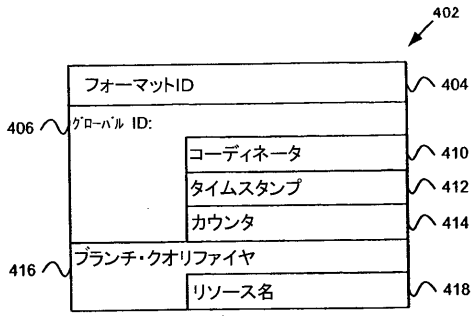


FIGURE 5

【図６】

後続のデータに対するフォーマット・バージョン・ナンバー	502
トランザクション識別子	504
余命	506
システムによって規定されたプロパティ	508
他のプロパティ	510
リソース・マネジャ名	512
登録された同時コールバックを持たないサブ・コーディネータ名	514
登録された同時コールバックを持つサブ・コーディネータ名	516

FIGURE 6

【図７】

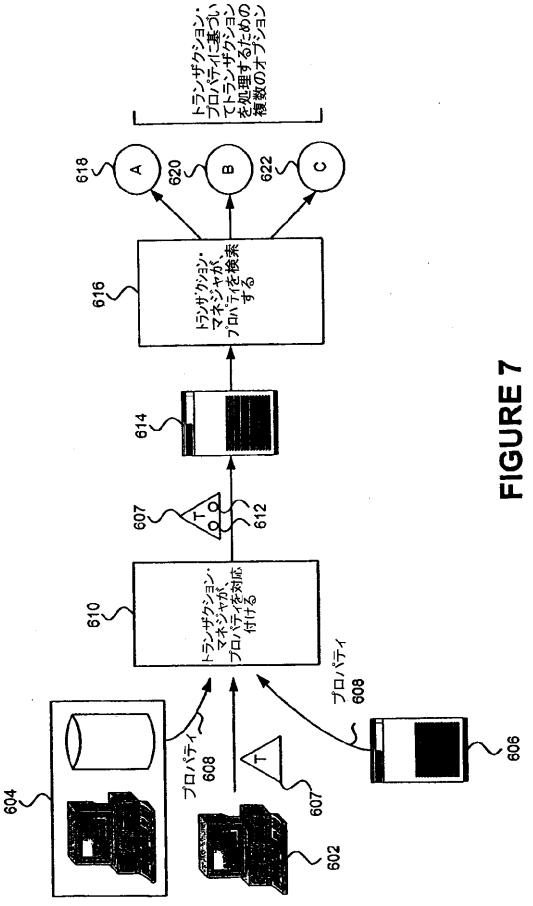


FIGURE 7

【図８】

```
import server.transaction.Transaction;
import server.transaction.TransactionManager;
import server.transaction.TxHelper;
...
TransactionManager tm = TxHelper.getTransactionManager();
Transaction tx;
tm.begin();
tx = (server.transaction.Transaction)tm.getTransaction();
tx.setProperty("rating", "VIP");
```

702

FIGURE 8

【図９】

```
import server.transaction.Transaction;
import server.transaction.TransactionManager;
import server.transaction.TxHelper;
...
int balance;
...
TransactionManager tm = TxHelper.getTransactionManager();
Transaction tx;
tx = (server.transaction.Transaction)tm.getTransaction();
String rating = (String)tx.getProperty("rating");

if (balance < 0 && rating.equals("VIP") {
  // Overdrawn VIP customer gets special treatment
}
```

802

FIGURE 9

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US02/22633

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) : G06F 7/00, 9/00, 17/30
US CL : 707/10; 709/101
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 707/10; 709/101

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X --- Y	US 5,857,100 A (PHILLIPS et al) 05 January 1999 (05.01.1999), column 6, lines 40-61; column 7, line 58-column 10 line 34; column 10, line 38-column 23, line 11; column 23, line 15-column 24, line 16; column 60, lines 16-34	1-7, 10-19, 22-29, 32-39, 42-44 --- 8-9, 20-21, 30-31, 40-41
Y	US 5,958,004 A (HELLAND et al) 28 September 1999 (28.09.1999), column 3, lines 44-50; column 11, line 54-column 12, line 28; column 16 lines 27-59; column 17, line 19-column 18, line 52	8-9, 20-21, 30-31, 40-41
Y	US 5,586,312 A (JOHNSON et al) 17 December 1996 (17.12.1996), column 1, line 8-column 5, line 2; column 6, line 64-column 8, line 4	1, 12, 23, 33, 33, 43, 44

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
09 October 2002 (09.10.2002)	10 DEC 2002
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703)305-3230	Authorized officer Ayaz R Sheikh Telephone No. 703-305-3900

Form PCT/ISA/210 (second sheet) (July 1998)

フロントページの続き

(81)指定国 AP(GH,GM,KE,LS,MW,MZ,SD,SL,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,MD,RU,TJ,TM),EP(AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,IE,IT,LU,MC,NL,PT,SE,SK,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AT,AU,AZ,BA,BB,BG,BR,BY,BZ,CA,CH,CN,CO,CR,CU,CZ,DE,DK,DM,DZ,EC,EE,ES,FI,GB,GD,GE,GH,GM,HR,HU,ID,IL,IN,IS,JP,KE,KG,KP,KR,KZ,LC,LK,LR,LS,LT,LU,LV,MA,MD,MG,MK,MN,MW,MX,MZ,NO,NZ,OM,PH,PL,PT,RO,RU,SD,SE,SG,SI,SK,SL,TJ,TM,TN,TR,TT,TZ,UA,UG,UZ,VN,YU,ZA,ZM,ZW

(74)代理人 100074228
弁理士 今城 俊夫
(74)代理人 100086771
弁理士 西島 孝喜
(72)発明者 フェルト エドワード ピー
アメリカ合衆国 ニュージャージー州 ０７７４７ メタワン クレセント プレイス ３１
(72)発明者 ファン プリシーラ
アメリカ合衆国 カリフォルニア州 ９４５８７ ユニオン シティー ランフォード テラス ３４８６９
(72)発明者 ソモジ アレクサンダー ジェイ
アメリカ合衆国 ニュージャージー州 ０７９２１ ベッドミンスター フィールドストーン ロード １７
(72)発明者 スリニヴァンサン スリラム
アメリカ合衆国 カリフォルニア州 ９４７０５ バークリー フローレンス ストリート ２９４０

Ｆターム(参考) 5B045 BB28 BB42 GG01
5B098 AA10 GA04 GC16