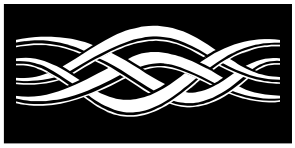
Microsoft

Microsoft® Transact Se



技　　術　　資　　料

1997 年 3 月

Microsoft Transaction Server の概要

Microsoft® Transact Se

Microsoft
Transaction
Server

Microsoft Transaction Server は、スケーラブルで信頼性の高い高性能の分散アプリケーションをより簡単に開発し、導入できるようにする、新しいカテゴリーのプロダクトです。これは、コンポーネント ベースの開発および導入環境と、トランザクション処理モニタの信頼性とスケーラビリティを組み合わせることによって実現されます。

Microsoft Corporation • One Microsoft Way • Redmond, WA 98052-6399 • USA

0197 Part No. 098-68638

目次

Microsoft Transaction Server(MTS)は、スケーラブルで信頼性の高い高性能なイントラネットおよびインターネット アプリケーションを、より簡単に構築できるように設計されています。このようなアプリケーションは、すでに何年も前から構築が可能になっていましたが、そのためにはほとんどの企業にとって多大なる投資と高度な技術的スキルが必要でした。

Microsoft Transaction Server は実証済みのトランザクション処理手法をベースにしていますが、その重要性はトランザクション処理モニタの領域を超えたところにあります。Microsoft Transaction Server はコンポーネント ベースの分散サーバー アプリケーションのための単純なプログラミング モデルと実行環境を定義しています。

アプリケーションは、ビジネス アプリケーション機能を提供する ActiveX™コンポーネントの集まりから構成されています。これらのコンポーネントはシングル ユーザー用のコンポーネントと同じように開発されます。その後、これらのコンポーネントをインストールして Transaction Server 環境で実行すると、サーバー アプリケーションは自動的に、高い性能と信頼性で複数の同時クライアントをサポートできるように拡張されます。

Transaction Server は、特にサーバー アプリケーションが非常に広い範囲でのスケーラビリティを持てるように設計されています。これには、小さなシングル ユーザー システムから、大規模なインターネット サーバーまでが含まれます。Transaction Server は、これまではハイエンドのトランザクション処理システムのものとされていた堅牢性と保全性を提供します。

を使用する理由

このセクションでは、優れたアプリケーション サーバーを開発することにまつわる複雑さを簡単に説明します。ここでは 3 つの異なる観点からさまざまな問題を扱います。まず、ユーザーが満足できるサービス レベルを提供するためにネットワーク サーバーが何をしなければならないのかを取り上げます。次に、コンポーネント ベースのアプリケーションを構築するときに生じる問題について論じます。最後に、障害が起こったときに、アプリケーションの整合性を保つことがどれだけ重要かを説明します。

Microsoft Transaction Server は、アプリケーション開発者をこれらの複雑な要素から開放するアプリケーション プログラミング モデルを提供するので、開発者はアプリケーションの機能に集中することができ、イントラネット／インターネット向きアプリケーションの構築に必要なコストと時間が軽減されます。

0 サーバー インフラストラクチャ

サーバーは高度なインフラストラクチャを必要とします。ネットワーク アプリケーション サーバーをゼロから構築するのは簡単な仕事ではありません。オンライン ブックストアのオーダーを処理するといった実際のビジネス機能のインプリメントは、作業のほんの一部でしかありません。一般に、サーバー システムは許容可能なレベルのパフォーマンスとスケールを実現するために、高度なインフラストラクチャを必要とします。

通常、アプリケーション サーバーの開発者は、インフラストラクチャの多くの部分を自分で開発しなければなりません。たとえば、豊富なサービスを提供する RPC システムでも、開発者は以下の作業を行う必要があります。

- ビジネス機能中心のサーバー プロセスの実行可能ファイルを開発する。
- サーバーをディレクトリ システムに登録する。
- サーバー プロセス プールとスレッド プールを管理する。最終的に、サーバーは 1 つのクライアントのために動作するスレッドを実行するのではなく、複数のクライアント要求にサービスを提供するスレッド プールを管理する必要があります。
- クライアントからの同時要求の間で、共有データとリソースへのアクセスを同期させる。このためには、デッドロック、競合条件、スタベーション、およびその他のパフォーマンス上の問題に対処する高度なロッキング プロトコルが必要です。
- クライアントの状況を管理する。これには、データベース接続や、ユーザーごとのデータ構造(またはオブジェクト)が含まれます。
- 低速なネットワーク上での待ち時間を改善するために、クライアント上で状態のキャッシングを行う。
- ビジネス機能とオブジェクトが、その権限を持つユーザーからしか使用できないように、セキュリティをインプリメントする。
- サーバーのリモート インストレーションと管理を可能にするために、管理および構成ツールをインプリメントする。

Transaction Server は、これらの要件を満たすアプリケーション サーバー インフラストラクチャを提供します。

1

2　コンポーネント ベースのアプリケーションの構築

コンポーネントを使ったアプリケーションの構築は、大きな魅力を持っており初期のオブジェクト指向コンピューティングの約束の 1 つでもありました。ビジネス機能を自然な形でカプセル化することができるので、特にサーバー アプリケーションにとって魅力がありました。しかし、コンポーネントをベースにしたアプリケーションのエンジニアリングは、最初思われていたよりも難しい作業でした。初期のオブジェクト システムの根本的な弱点は、さまざまな当事者が作成したオブジェクトを、同じプロセスの中で、あるいは複数のプロセスの間で開発者が統合できるような共通のフレームワークが存在しなかったという点にあります。Component Object Model(COM)はこの問題に対処しています。

しかし、共通のコンポーネント オブジェクト モデルだけでは、コンポーネントからのサーバー アプリケーションの構築はできません。コンポーネントは共通のサーバー フレームワークを持つ必要があるのです。独自のサーバー フレームワークを構築している開発者は、他の人々が開発したコンポーネントを使用する機会が制限されます。

Transaction Server のアプリケーション アーキテクチャとプログラミング インターフェイスは、コンポーネント ベースのサーバー アプリケーションを構築するための共通のフレームワークを提供します。

3　アプリケーションの保全性の維持

ビジネス システムにとって、ビジネスの状態を正確に保持できることは非常に重要です。たとえば、オンライン ブックストアはオーダーを信頼性の高い形で追跡できなくてはなりません。これに失敗すると、大きな損失が生じる可能性があります。現在のオーダーが失われたり、オーダーの処理に遅れが生じるかもしれません。これに不満を抱いた顧客は、他のシステムに注文するようになるかもしれません。

ビジネス システムの保全性は、特に障害が起こったときには、決して簡単に実現できることではありませんでした。皮肉なことに、コンピュータの信頼性が高まっても、全体としてのシステムの信頼性は低下しつつあります。数十、数百、さらには数十万のサーバー マシンにイントラネットやインターネットを経由して接続された数百、数千、あるいは数百万のデスクトップ マシンから構成されているシステムでは、障害は日常茶飯事として起こっています。

この問題は、分散アプリケーションの需要の高まりによって、いっそう複雑になっています。本のオーダーといったビジネス トランザクションには、複数のサーバーが関与するようになってきています。クレジット カードの確認、本の出荷、在庫の管理、顧客に対する請求などの作業が必要です。このため、複数のサーバー上の複数のデータベースで更新処理を行わなければなりません。分散アプリケーションの開発者は、アプリケーションの一部が障害を起こしたときも、他の部分は実行を続けているというような状況を想定する必要があります。このような障害シナリオは、全体として障害を起こすモノリシックなアプリケーションよりもはるかに複雑になります。

ビジネス アプリケーションは、複数の作業を 1 つのビジネス トランザクションとして調整しなければならないことがよくあります。たとえば、オンライン ブックストアでは、請求処理を行わずに本の出荷のスケジュールを立てるべきではありませんし、出荷のスケジュールを立てずに顧客に請求するのも困ります。複数の作業のすべてが実行されるか、さもなければ 1 つも実行されないようにするという調整は、システムからの特殊なサポートがないと非常に困難です。

障害が起こった場合でもアトミックな更新を保証するのは簡単なことではありません。アプリケーションが複数のデータベースやシステムに分散している場合には特に困難になります。このような状況で、複数のコンポーネントを使用すると、コンポーネントはその設計上、内部のインプリメンテーションを隠すものですから、問題はさらに複雑になります。

また、アプリケーションは複数のクライアントがコンポーネントにアクセスしているときも、一貫性のある動作をしなければなりません。同じ本に対して同時にオーダーがあったときも、2人の顧客に1冊の本を送るというような結果になってはなりません。アプリケーションが適切に作成されていないと、いつかは競合条件のために矛盾が発生します。これらの問題の解決は困難で高価であり、ボリュームと並列性が高まるにつれて発生の頻度も高まります。ここでもコンポーネントを使用することで、問題がさらに複雑になります。

Transaction Serverは、トランザクションをコンポーネント ベースのプログラミングと統合することにより、堅牢なコンポーネント ベースの分散アプリケーションの開発を支援します。

Mi

crosoft Transaction Server

のアーキテクチャ

このセクションでは、Transaction Server の主要なアーキテクチャ上の要素について簡単に紹介します。以下の要素があります。

- ActiveX コンポーネント。アプリケーション機能をインプリメントします。
- Transaction Server エグゼクティブ。アプリケーション コンポーネントが使用するランタイム サービスを提供します。
- サーバー プロセス。アプリケーション コンポーネントを中心とする疑似プロセス環境を提供します。
- リソース マネージャ(RM)。アプリケーションの持続性のある状態を管理します。例としては、リレーショナル データベース システムやトランザクショナル メッセージ キューがあります。
- リソース ディスペンサ。プロセス内のコンポーネントの非持続的な共有状態(データ)を管理します。例としては、データベース接続のプーリングなどがあります。
- Microsoft 分散トランザクション コーディネータ。複数のリソース マネージャ、リソース ディスペンサ、およびアプリケーション コンポーネントの間でトランザクションの調整を行います。
- Transaction Server Explorer。Transaction Server システムを管理するためのグラフィカル インターフェイス。

4　Transaction Server のコンポーネント

アプリケーション コンポーネントはビジネスのアクティビティをモデル化します。これらのコンポーネントは、アプリケーションの状態の表示と変換を提供することで、ビジネス ルールをインプリメントします。オンライン ブックストアを例に取って考えてみましょう。ビジネスの持続的な状態、たとえば未処理のオーダー、手持ち在庫、受取勘定などは、1 つまたは複数のデータベース システムの中のレコードによって表現されます。アプリケーション コンポーネントは、新しいオーダーや在庫の出荷などの変化を反映するために、状態を更新します。

Transaction Server アプリケーション コンポーネントは ActiveX のプロセス内サーバー(DLL)です。これらのコンポーネントは、Visual Basic®、Visual C++®、Visual J++®、または任意の ActiveX 互換の開発ツールで作成することができます。Component Object Model(COM)をベースにした ActiveX は、以下の特徴を持っています。

- クライアントがオブジェクトからサービスを要求するための手段であるインターフェイスという概念。
- 複数のプロセス間で、またマシンの境界を越えて透過的にオブジェクトと通信を行う能力。
- コンポーネントを識別し、動的にロードと実行を行うためのメカニズム。
- オブジェクトが複数のインターフェイスをサポートできるようなアーキテクチャ。このアーキテクチャにより、クライアントはオブジェクトに対して、特定のインターフェイスをサポートしているかどうかを問い合わせることができます。これにより、コンポーネントはさまざまなレベルの機能を提供することができますし、新しいバージョンの導入も円滑に行えます。

Transaction Server は COM を拡張して一般的なサーバー アプリケーション フレームワークを提供しています。上記の COM 自身の機能に加えて、Transaction Server は以下の特徴を持っています。

- サーバー登録、プロセスおよびスレッド管理、コンテキスト管理、共有リソースの管理と同期、およびコンポーネント ベースのセキュリティを処理します。
- プログラミング モデルに、アトミックな更新を実現し、複数のコンポーネント、データベース システム、およびネットワーク境界を超えた一貫性を提供するためのメカニズムとしてのトランザクションを導入します。各コンポーネントは、そのコンポーネントのトランザクション上のセマンティクスを示すトランザクション プロパティを持っています。これにより、Microsoft Transaction Server はトランザクショナル コンテキストを自動的に管理することができます。

Transaction Server は、開発者を複雑なサーバー関連の問題から開放するので、開発者はビジネス機能のインプリメントに集中することができます。Transaction Server の下で動作するコンポーネントはトランザクション処理ができるので、アプリケーションを開発する際には、そのアプリケーションがまったく独立して動作するものと考えることができます。並列性、リソース プーリング、コンテキスト管理、およびその他のシステム レベルの複雑な処理は、Transaction Server が処理します。データベース サーバーやその他のタイプのリソース マネージャと協調して動作するトランザクション システムにより、同時トランザクションがアトミックであり、一貫性を持ち、適切な独立性を持ち、いったんコミットされた変更に永続性があることが保証されます。

アプリケーションはパッケージと呼ばれる ActiveX コンポーネントの集まりとして導入されます。パッケージは障害の独立性と責任範囲を定義します。

5 Transaction Server エグゼクティブ

Transaction Server エグゼクティブは、Transaction Server コンポーネントのためのランタイム サービスを提供する DLL です。これらのサービスにはスレッド管理とコンテキスト管理が含まれます。この DLL は、アプリケーション コンポーネントをホストするプロセスにロードされ、バックグラウンドで透過的に実行されます。

6 サーバー プロセス

サーバー プロセスは、アプリケーション コンポーネントの実行を中心とするシステム プロセスです。各サーバー プロセスは 1 つのコンポーネント パッケージを中心に、十、百、さらには数千のクライアントに対してサービスを提供します。1 台のコンピュータ上で複数のサーバー プロセスが実行されるように構成することができます。各サーバー プロセスは、線引きをした責任範囲と障害からの独立をもたらします。

他のプロセス環境も、アプリケーション コンポーネントを中心とすることができます。このため、さまざまな分散性、パフォーマンス、および障害分離の要件に応じた形で、アプリケーションの導入を行うことができます。たとえば、Transaction Server コンポーネントを Microsoft SQL Server™や Microsoft Internet Information Server(IIS)に直接ロードするように構成することができます。また、クライアント プロセスに直接ロードするように構成することもできます。

7

8　　リソース マネージャ

リソース マネージャは、持続性のあるデータを管理するシステム サービスです。サーバー アプリケーションは、手持ち在庫、未処理のオーダー、および受取勘定のレコードといった、アプリケーションの持続性のある状態を保持するために、リソース マネージャを使用します。リソース マネージャはトランザクション マネージャと協力して、アプリケーションに対してアトミック性と独立性を保証します。リソース マネージャの例としては、Microsoft SQL Server、持続的なメッセージ キュー、トランザクショナル ファイル システムなどがあります。

アトミック性は、特定のトランザクションの下で完了したすべての更新が、コミットされる(持続性のある状態になる)か、中断されて以前の状態にロールバックされるかのどちらかになることを保証します。

一貫性とは、トランザクションがシステムの状態の正しい変遷であり、もしくは状態の不変遷を保持することを意味します。

独立性とは、同時に実行されるトランザクションが、互いの部分的な未コミットの結果を見ることがないという意味です。このようなことが起こると、アプリケーションの状態に矛盾が生じる可能性があります。リソース マネージャは、アクティブ トランザクションの未コミットの作業を分離するために、トランザクション ベースの同期プロトコルを使用します。

永続性とは、管理対象のリソース(データベース レコードなど)に対するコミットされた更新が、通信の障害、プロセスの障害、サーバー システムの障害などを含めた障害によって損傷を受けないという意味です。トランザクショナル ロギングを利用することで、ディスク媒体の障害が起こった後でも、持続的な状態を回復することができます。

アトミック性と独立性が組み合わされることで、トランザクションは瞬時に実行されるように見えます。トランザクションの中間状態はトランザクションの外からは見えず、すべての作業が実行されるか、何も実行されないかのどちらかになります。このため、各トランザクションがシーケンシャルに、また並列性とはまったく関係なく実行されると仮定してアプリケーション コンポーネントを作成することができます。これはアプリケーション開発者にとっては大幅な単純化です。

Transaction Server は、OLE トランザクション プロトコルか X/Open XA プロトコルをインプリメントするリソース マネージャをサポートします。リソース マネージャを開発するためのツールキットが用意されています。

9　　リソース ディスペンサ

リソース ディスペンサは、プロセス内のアプリケーション コンポーネントのために、非持続的な共有状態を管理するサービスです。リソース ディスペンサはリソース マネージャに似ていますが、持続性の保証は行いません。Transaction Server は 2 つのリソース ディスペンサを用意しています。

- the ODBC Resource Dispenser
- the Shared Property Manager

リソース ディスペンサを開発するためのツールキットが用意されています。

10

11 ODBC Resource Dispenser

ODBC Resource Dispenser は、標準の ODBC インターフェイスを使用する Transaction Server コンポーネントのために、データベース接続のプールを管理しています。リソース ディスペンサはデータベース接続のプールを管理し、オブジェクトに対して素早く効率的に接続を割り当てます。接続は自動的にオブジェクトのトランザクションに登録されます。リソース ディスペンサは接続を自動的にリクレームし、再利用することができます。ODBC リソース ディスペンサは、この機能を透過的に提供する DLL で、Transaction Server に組み込まれています。

12 Shared Property Manager

Shared Property Manager は、アプリケーション定義のプロセス全体のプロパティ(変数)への同期化されたアクセスを提供します。これを使用して、Web ページのヒット カウンタの保守、不変データのキャッシング、データベース ホットスポットを避けるためのインテリジェントなキャッシングの提供(一意の受信番号を生成するなど)を行うことができます。

13 Microsoft 分散トランザクション コーディネータ

Microsoft 分散トランザクション コーディネータは、複数のリソース マネージャにまたがるトランザクションを調整するシステム サービスです。別々のマシンに置かれた複数のリソース マネージャにまたがる作業であっても、アトミックなトランザクションとしてコミットすることができます。

Microsoft 分散トランザクション コーディネータは、もともと Microsoft SQL Server 6.5 の一部としてリリースされたもので、Transaction Server プロダクトにも含まれています。これは、トランザクションに関与しているすべてのリソース マネージャの間で、トランザクションの結果(コミットまたはアボート)が一貫性のある状態になることを保証する 2 フェーズ コミット プロトコルをインプリメントしています。Microsoft 分散トランザクション コーディネータは、障害(ノード クラッシュ、ネットワーク クラッシュ、リソース マネージャまたはアプリケーションの不正な動作など)、競合条件(トランザクションがコミットを開始したときに、リソース マネージャがアボートを開始した、など)、またはアベイラビリティ(リソース マネージャがトランザクションを準備したが、制御が返ってこない、など)にかかわらず、アトミック性を保証します。

Microsoft 分散トランザクション コーディネータは、OLE トランザクションまたは X/Open XA プロトコルをインプリメントするリソース マネージャをサポートします。

結論

Microsoft Transaction Server は、これまでのビジネス アプリケーションの開発方法を変化させます。コンポーネント ベースのオブジェクト指向テクノロジーと、分散型のオンライン トランザクション処理の実証済みのテクニックの組み合わせにより、市販のコンポーネントとカスタム コンポーネントから構成されたアプリケーションを簡単に導入できるようになります。この経済的な利点によりビジネス コンポーネントの新しい市場が創出されます。その結果、これまではコスト的に不可能だったビジネス ソリューションが実現されることになるでしょう。

Microsoft Transaction Server のロールアウトは 2 つのフェーズに分けて行われました。最初に、分散トランザクション コーディネータが 1996 年 4 月に Microsoft SQL Server Version 6.5 の一部として出荷されました。このテクノロジーは、異種のデータストアの間で分散型の 2 フェーズ コミットを実現します。

1996 年 12 月になって、Microsoft Transaction Server が出荷されました。これは、ActiveX コンポーネントを信頼性の高い、スケーラブルな分散環境で実行するためのプログラミング モデルとランタイム実行環境を提供します。

14 関連情報

Microsoft Transaction Server の最新情報については、http://www.microsoft.com/transaction/を参照してください。

また、「Transaction Processing: Concepts and Techniques"、Jim Gray and Andreas Reuter 著。Morgan Kaufmann Publishers, 1993 も参照してください。