Oracle8*i*

Integration Server 概要

リリース 8.1

2000 年 11 月

部品番号：J02323-01

ORACLE®

Oracle8*i* Integration Server 概要 , リリース 8.1

部品番号：J02323-01

原本名：Oracle8*i* Integration Server Overview, Release 3 (8.1.7)

原本部品番号：A83729-01

原本著者：Thomas Kurian, Chitra Sharma

原本協力者：Anna Sears, Jon WIlkinson

Copyright © 1996, 2000, Oracle Corporation. All rights reserved.

Printed in Japan.

制限付権利の説明

プログラム（ソフトウェアおよびドキュメントを含む）の使用、複製または開示は、オラクル社との契約に記された制約条件に従うものとします。著作権、特許権およびその他の知的財産権に関する法律により保護されています。

当プログラムのリバース・エンジニアリング等は禁止されております。

このドキュメントの情報は、予告なしに変更されることがあります。オラクル社は本ドキュメントの無謬性を保証しません。

* オラクル社とは、Oracle Corporation（米国オラクル）または日本オラクル株式会社（日本オラクル）を指します。

危険な用途への使用について

オラクル社製品は、原子力、航空産業、大量輸送、医療あるいはその他の危険が伴うアプリケーションを用途として開発されておりません。オラクル社製品を上述のようなアプリケーションに使用することについての安全確保は、顧客各位の責任と費用により行ってください。万一かかる用途での使用によりクレームや損害が発生いたしましても、日本オラクル株式会社と開発元である Oracle Corporation（米国オラクル）およびその関連会社は一切責任を負いかねます。当プログラムを米国国防総省の米国政府機関に提供する際には、『Restricted Rights』と共に提供してください。この場合次の Notice が適用されます。

Restricted Rights Notice

Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

このドキュメントに記載されているその他の会社名および製品名は、あくまでその製品および会社を識別する目的にのみ使用されており、それぞれの所有者の商標または登録商標です。

目次

2 方法論とソリューション

3 独自の Oracle Integration Server の概要

4 基本的な統合の概念

第 II 部　製品

5　同期アプリケーションの統合

6　データ・レプリケーションとゲートウェイ

7　Oracle アドバンスト・キューイングと JMS

8 Oracle Message Broker と JMS

9 ディレクトリ・サービス（LDAP）

索引

はじめに

このマニュアルでは、Oracle Integration Server とその統合ソリューションへの応用を説明します。

次の項目が含まれています。

Oracle Integration Server の目的

企業 または E-Business の統合が目指すものは、複数のアプリケーション間での相互通信を可能にして、ビジネス・プロセスを簡素化することです。企業の統合は、異なるシステム間におけるデータの同期化、ビジネス・プロセスの自動化、または相互のパッケージ・アプリケーションの分離に対する必要性によって推進されます。E-Business、特に B2B 取引は、従来の企業統合に対する需要を増進し、まったく新しい種類の統合に対する需要を創造しています。

Oracle Integration Server（OIS）は、E-Business で使用される様々なテクノロジの統合を容易にします。Oracle Integration Server は、Oracle 製品内部とサード・パーティ製品の両方で動作します。

OIS は、次の内容を可能にします。

- 高度なレプリケーションとゲートウェイを使用してシステム間におけるデータの同期化を行う。
- 次の方法で、アプリケーションが相互に同期および非同期で通信できるようにする。
  - メッセージの格納と管理
  - メッセージのルーティングと伝播
  - データとメッセージの変換サービス
  - ビジネス・プロセスとワークフロー・サービス

範囲

このマニュアルでは、企業統合に関する技術的な要件の概要とその要件を満たす Oracle Integrations Server 内の様々なコンポーネントを説明します。

第 III 部では、OIS と様々なサード・パーティ統合ソリューションとの間の相互運用性などの統合に関する高度なトピックをいくつか説明します。

このマニュアルの目的は、統合に関する概念的な要素をいくつか紹介し、Oracle Integration Server でこれらの要件を満たす方法を説明することです。特定の統合シナリオで利用できる OIS のコンポーネントについても説明します。個々のコンポーネントについての詳細な説明はありませんが、その製品の使用方法、その製品を使用したアプリケーションの開発方法、およびその製品を配布し管理する方法に関するドキュメントへのリンクが用意されています。

対象読者

このマニュアルは、OIS を使用してメッセージ・ベースの統合ソリューションを定義、開発および実装するシステム設計者を対象にして作成されていますが、OIS のインストールを管理するデータベース管理者や統合ソリューションの事前に定義されたコンポーネントを作成する開発者にも役立ちます。

このマニュアルの読者は、アプリケーション統合の複雑さや問題点について基本的に理解してることが前提です。トランザクション処理および Oracle8*i* の概念を理解しておくことをお薦めします。

このマニュアルの構成

このマニュアルは、3 つの部で構成されています。それぞれの部では、E-Business 統合と Oracle 統合製品に関する具体的な側面を説明します。

第 I 部では、E-Business 統合の概要を説明し、統合をビジネス的に推進する要素、取り組む必要がある具体的な技術上の問題点および Oracle Integration Server の主要な統合概念を説明します。

- 第 1 章「E-Business 統合の概要」
- 第 2 章「方法論とソリューション」
- 第 3 章「独自の Oracle Integration Server の概要」
- 第 4 章「基本的な統合の概念」

第 II 部では、OIS の各製品コンポーネントに関する概要を説明し、該当製品のドキュメントを紹介するためのリンクを提供します。

- 第 5 章「同期アプリケーションの統合」
- 第 6 章「データ・レプリケーションとゲートウェイ」
- 第 7 章「Oracle アドバンスト・キューイングと JMS」
- 第 8 章「Oracle Message Broker と JMS」
- 第 9 章「ディレクトリ・サービス（LDAP）」
- 第 10 章「Oracle Workflow」

第 III 部では、OIS と様々なサード・パーティの統合ソリューションとの相互運用性など、統合に関する高度なトピックを説明します。

関連ドキュメント

OIS 内のコンポーネントの詳細は、次のドキュメントを参照してください。

Oracle8*i* Server 関連ドキュメント

- 『Oracle8*i* アプリケーション開発者ガイド アドバンスト・キューイング』
- 『Oracle8*i* CORBA 開発者ガイド』
- 『Oracle8*i* レプリケーション・ガイド』
- 『Oracle8*i* 分散システム』
- 『Oracle Internet Directory アプリケーション開発者ガイド』
- 『Oracle Enterprise Manager 管理者ガイド』

コンポーネント製品関連ドキュメント

- 『Oracle8*i* Workflow ガイド』
- 『Oracle Message Broker Administration Guide Release 2.0.1.0 for SPARC Solaris and Windows NT』
- 『Oracle8*i* PL/SQL パッケージ・プロシージャ リファレンス』
- 『Oracle8*i* Java パッケージ・プロシージャ リファレンス』
- Oracle Objects for OLE のオンライン・ヘルプ
- 『Oracle8*i* XML リファレンス・ガイド』
- 『Oracle Applications InterConnect User's Guide』

表記規則

この項では、Oracle8*i* ドキュメント・セットの本文およびコード例の中で使用する表記規則を説明します。項目は次のとおりです。

本文の表記規則

本文では、特別な用語をすぐに識別できるように、様々な表記規則を使用しています。次の表に、表記規則とその使用例を示します。

表記規則	意味	例
イタリック体	イタリック体は、構文の句またはプレースホルダを示します。	*parallel_clause* を指定でき ... U*old_release*.SQL を実行します。*old_release* は、アップグレード前のリリースを示します。
大文字の固定幅フォント	大文字の固定幅フォントは、システムが指定する要素を示します。この要素には、システム指定の列名、データベース・オブジェクトと構造、ユーザー名およびロールの他に、実行可能ファイル、パラメータ、権限、データ型、SQL のキーワード、SQL*Plus やユーティリティのコマンド、パッケージおよびメソッドが含まれます。	NUMBER 列に対してのみこの句を指定できます。 この値は、ALTER TABLE 文で変更できます。 ... DEPTNO 列によってグループ化され ... ROLLBACK_SEGMENTS パラメータを指定し ... ... DBMS_STATS.GENERATE_STATS プロシージャ ...
小文字の固定幅フォント	小文字の固定幅フォントは、ユーザーが指定する要素のサンプルを示します。これらの要素には、ユーザー指定のデータベース・オブジェクトと構造、列名、パッケージとクラス、ユーザー名とロール、プログラム・ユニットおよびパラメータの値の他に、コンピュータやデータベースの名前、ネット・サービスの名前、接続識別子が含まれます。	deptno、dname および loc の各列が scott.dept 表にあります。 QUERY_REWRITE_ENABLED 初期化パラメータを true に設定します。 sales@sf.acme.com データベースに接続します。 oe ユーザーで接続します。

コード例の表記規則

コード例は、SQL、PL/SQL、SQL*Plus またはその他のコマンドラインの文を示します。これらのコード例は、次のように固定幅フォントで表示され、通常のテキストと区別されています。

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

次の表に、コード例の表記規則とその使用例を示します。

表記規則	意味	例
[ ]	大カッコで囲む対象は、1 つまたは複数のオプション項目です。大カッコは入力しません。	`DECIMAL (digits [ , precision ])`
{ }	中カッコで囲む対象は複数の項目で、その中から 1 つを選択する必要があります。中カッコは入力しません。	`{ENABLE | DISABLE}`
|	縦線を使用して、大カッコまたは中カッコで囲まれた複数のオプションから選択できる項目を表しています。オプションから 1 つの項目を入力します。縦線は入力しません。	`{ENABLE | DISABLE}` `[COMPRESS | NOCOMPRESS]`
...	水平方向の省略記号は、次のいずれかを示します。 ■ その例に直接関係のないコードの一部が省略されていることを示します。 ■ コードの一部が繰り返して続くことを示します。	`CREATE TABLE ... AS subquery;` `SELECT col1, col2, ... , col`*n*` FROM emp;`
. . .	垂直方向の省略記号は、その例に直接関係ない一部のコード行が省略されていることを示します。	
その他の句読点	大カッコ、中カッコ、縦線および省略記号以外の句読点は、表記どおりに入力する必要があります。	
イタリック体	イタリック体のテキストは、特定の値の指定が必要な変数を示します。	`STARTUP PFILE=init`*sid*`.ora` この例では、文字列 `init`*sid*`.ora` 全体がパラメータ・ファイルのプレースホルダとなり、特定のインスタンス ID または SID をユーザーが指定する必要があります。

表記規則	意味	例

大文字	大文字は、システムが指定する要素を示します。これらの語句は、ユーザー定義の語句と区別するために大文字で表記されています。大カッコで囲まれている語句以外は、表記どおりに同じ順序で入力します。ただし、これらの語句は大 / 小文字を区別しないため、小文字で入力することができます。	`SELECT ename, empno FROM emp;`
		`SQLPLUS username/password`
		`INTO TABLENAME 'table'`

小文字	小文字は、ユーザーが指定するプログラムの要素を示します。たとえば、表、列またはファイルの名前を示します。	`SELECT ename, empno FROM emp;` `SQLPLUS scott/tiger`

第I部

E-Business と統合の概要

第Ⅰ部では、E-Business を推進し、統合ソリューションを見いだすためのビジネス概念を紹介し、取り組む必要がある技術上の具体的な問題点を説明します。また、Oracle Integration Server の主要概念を概説します。

1

E-Business 統合の概要

この章では、E-Business 統合ソリューションの開発の概要を説明します。次の項目が含まれています。

- E-Business 統合の概要
- E-Business 統合の理由
- E-Business 統合テクノロジとアプローチ

E-Business 統合の概要

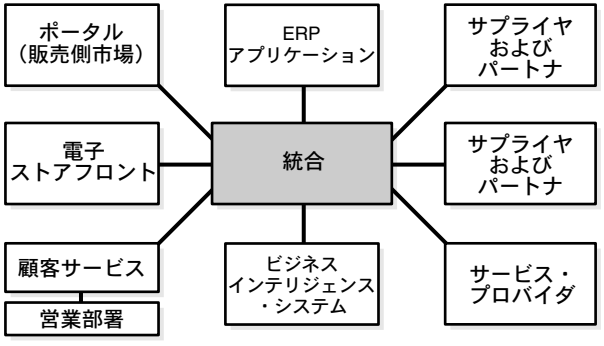
インターネット革命は、すべての企業が E-Business を避けられない段階にまで進んできました。E-Business は緊急課題であり、選択の余地はありません。したがって、企業は、いつ、どのような方法で E-Business を推進するかを判断する必要があります。

E-Business とは、組織のビジネス処理方法に対する根本的な変更です。E-Business ではインターネット・テクノロジを次の目的に使用します。

- 自社の製品とサービスを購入する顧客を引きつけ、満足させて確保します。
- サプライ・チェーン、製造および購買の各システムを簡素化し、適切な製品とサービスを顧客に効率的に提供します。
- 企業のビジネス・プロセスを自動化して、セルフサービスによるコスト削減と効率性の向上を実現します。
- 顧客と企業運営に関するビジネス・インテリジェンスを獲得して分析し、共有します。これによって経営陣は、ビジネスの意志決定をより的確に行い、継続してビジネス方針を改善できます。

E-Business に必要なのは、各種のインターネット対応アプリケーションです。その中には、E-Commerce Web サイト、ポータル、サプライ・チェーン管理、購買管理、オンライン・マーケットプレイス、カスタマ・リレーションシップ・マネジメント（CRM）、ERP などのアプリケーションが含まれます。企業の E-Business を実現するには、これらすべてのアプリケーションを相互に統合する必要があります。

図 1-1 統合、E-Business 統合に対する E-Business ドライバの基本



「待機時間ゼロの組織」を目指す企業の必要性が、企業規模の情報システムとアプリケーションの統合を推進します。たとえば、E-Business を円滑に運営するには、次のような要件が必要です。

- 電子ストアフロントで受信した注文は、顧客の注文状態に関する問合せに応答するために顧客サポート担当が自動的に表示できること。
- その注文が自動的にサプライ・チェーン・アプリケーションに伝播され、計画と実行の操作が開始されること。
- 注文情報が、サービス業務と搬送に関わっているサプライヤやパートナとインターネット上で交換されること。

次の発展が E-Business 統合の必要性を推進させます。

- 合併と買収
- パッケージ・アプリケーション
- ビジネス・プロセス・リエンジニアリング
- バーチャルで動的なサプライ・チェーン
- カスタマ・リレーションシップ・マネジメント（CRM）
- 企業セルフサービス
- 企業間取引
- アプリケーション・サービス・プロバイダとホスティング

合併と買収

2 つ以上の企業が合併したり、ある企業が別の企業を買収した場合、これらの企業は、新規企業でのビジネス・プロセスを企業規模で自動化する必要があります。自動化のためには、互いの企業の情報システムを接続し、情報を同期化して共有します。

たとえば、合併と買収によって世界規模の整理統合が進んでいる電気通信業界では、各企業の連結された顧客ベースの共通ビューが必要です。その結果、各企業のフロントエンド・データベース間、課金システム間、他のフロントエンド・アプリケーション間で顧客情報を同期化して共有する必要があります。

パッケージ・アプリケーション

企業は、ビジネスの一部を簡素化するためにパッケージ・アプリケーションを購入するため、このアプリケーションを企業内の別のパッケージ・アプリケーションやレガシー・システムと統合する必要があります。エンタープライズ統合テクノロジを使用すると、各アプリケーションを相互に交信させてビジネス・プロセスを自動化できます。

ビジネス・プロセス・リエンジニアリング

ビジネス・プロセス・リエンジニアリングによって、企業統合の必要性が出てきます。組織がビジネス・プロセスを再設計、簡素化する場合、そのビジネス・プロセスをサポートしている基礎となるアプリケーションのインフラストラクチャは、異なるシステムと新しい方法で通信する必要があります。ビジネス・プロセス・リエンジニアリングを推進する企業は、統合ミドルウェアを配布して異なるシステム間を接続します。

これまでに述べた 3 つ項目の発展が、企業内における企業統合に対する主要な誘因ですが、E-Business への移行を推進する企業は、さらに他の統合ニーズにも直面しています。E-Business では、顧客とサプライヤは組織の内部ビジネス・プロセスに透過的に直接アクセスします。その結果、E-Business そのものが企業内と企業間での企業統合の需要を創出します。需要創出の要因として、次に内容を示します。

- バーチャルで動的なサプライ・チェーン
- カスタマ・リレーションシップ・マネジメント（CRM）
- 企業セルフサービス
- 企業間取引
- アプリケーション・サービス・プロバイダとホスティング

バーチャルで動的なサプライ・チェーン

多くの E-Business では、サプライ・チェーン実行プロセスの重要な部分をパートナにアウトソーシングしています。パートナには、特化されたコンポーネントを作成する製造スペシャリスト、ロジスティックスと注文の実行を提供する実行スペシャリスト、倉庫内のサプライヤの在庫を管理する倉庫管理スペシャリストなどがいます。

たとえば、多くのパーソナル・コンピュータ・メーカーでは、コンピュータの組立ては自社で行いますが、PC ボードの製造はアウトソーシングし、サード・パーティのロジスティックス・プロバイダを使用して PC を直接顧客に出荷します。このような場合は、企業のサプライ・チェーン・アプリケーションとビジネス・プロセスが、サプライヤのサプライ・チェーン・システムと緊密にリンクしている必要があります。このリンクによって、確実に在庫レベルと需要パターンがすべての関係者にグローバルに表示されます。企業間統合ソフトウェアは、これらのシステムをリンクします。

カスタマ・リレーションシップ・マネジメント（CRM）

E-Business では、顧客は様々な機能を使用して企業にアクセスします。それらの機能を使用して、顧客は次の内容を実行できます。

- 企業の Web サイトを通してその企業の製品情報にアクセスできます。
- その企業の Web ストアを通して製品を注文できます。

- 企業の顧客サービス・オペレーションに電話して、注文の状態をチェックできます。
- 企業のパートナにフォローアップを行い、購入した製品に対するサービスを受けることができます。

顧客が利用するアクセス・チャネルに関係なく、顧客に企業の統一ビューを提供するには、企業のすべてのフロントエンド・アプリケーションが統合されている必要があります。理想としては、すべてのフロントエンド・アプリケーションが統合できるように設計されていることです。相互に統合されないアプリケーションの場合、企業は統合ミドルウェアを使用し、システムを統合する必要があります。

企業セルフサービス

企業が E-Business へ移行するのに従って、顧客、サプライヤおよび従業員に対応するビジネス・プロセスはセルフサービスへと変わります。顧客は、企業の Web ストアにアクセスして製品の注文を入力します。サプライヤは発注書を入力し、自身の安全な Web サイトを介して購買依頼処理を完了します。従業員は経費精算書を提出し、発注書を作成してオフィス用品を購入し、さらに、自分の出張用チケットをすべてオンラインで購入します。

セルフサービスを容易にするには、ビジネス・プロセスを簡素化して首尾一貫した処理を実行する必要があります。たとえば、従業員が経費精算書を提出すると、ワークフロー・プロセスによってその従業員の上司に通知され、適切な金額が企業の財務システムの借方に記入され、一方では、企業の給与システムで従業員の貸方に記入されます。E-Business 統合ミドルウェアは、これらすべての個別システムを統合し、ビジネス・プロセスを容易にします。

企業間取引

オンライン・マーケットプレイスまたはオンライン取引所を通して企業間取引を行う企業の増加に従って、これらのオンライン取引所に接続しているサプライヤと顧客は、やり取りを自動化して次の要件を満たす必要があります。

- 製品カタログと価格をそのオンライン取引所に一致させます。
- 顧客の要求と入札に応答します。
- オークションおよび逆オークションに参加します。

サプライヤや顧客は、E-Business 統合ミドルウェアを使用して、各自の ERP アプリケーションをオンライン・マーケットプレイスに結び付け、企業間取引の簡素化を開始しています。

たとえば、オラクル社は、企業が構築中のオンライン取引所にサプライヤをリンクする E-Business 統合ミドルウェアの独自のバージョンを提供しています。これが、Oracle Integration Server と呼ばれるものです。

アプリケーション・サービス・プロバイダとホスティング

企業の総力を独自の中核能力に結集し、エンタープライズ・アプリケーションをアプリケーション・サービス・プロバイダ（ASP）やホスティング企業にアウトソーシングする企業が増えています。これによって、企業独自のレガシー情報システムを ASP のシステムに接続したり、1 つの ASP のアプリケーションを別の ASP のアプリケーションに接続する必要が生じます。この結果、企業のバックオフィス・システムを企業のデータ・センターから ASP のデータ・センターに移行することになり、これらの異なるシステムを結合するための E-Business 統合ミドルウェアの需要が生じます。

E-Business 統合の理由

E-Business 統合をビジネス的に推進する要素は、統合インフラストラクチャ内の具体的な要件に変換されます。必要な統合ミドルウェアを理解するために、E-Business 統合に関する次の 3 つの基本的な理由を説明します。

- 情報システム間のデータの同期化
- アプリケーションとビジネスの切離し
- マルチステップ・ビジネス・プロセスの簡素化

情報システム間のデータの同期化

第 1 の理由は、情報システム間のデータの同期化の必要性です。E-Business では、ビジネス・オブジェクトまたは情報の一貫した、グローバルな企業レベルのビューが必要です。たとえば、次のようなビューが必要です。

- 多様なビジネス全体にわたる顧客ごとの単一の統合ビュー
- サプライヤとパートナを含めた各サプライ・チェーンの在庫と需要パターンの一貫性のあるビュー
- 本質的に異なるすべての財務追跡管理システムにわたる財務状態の一貫性のあるビュー

これら 3 つすべてにおける基本的な統合の必要性は、異なるシステム全体にわたる一貫性のあるグローバルな情報のビューにあります。このビューが異なる情報システム間のデータを同期化します。

システム間におけるデータの同期化は、定期的なバッチ転送か、または連続的な反復トランザクションのいずれかによって実行できます。

バッチスタイルのデータの同期化

たとえば、オンライン・トランザクション処理システムとデータ・ウェアハウス間でのデータの同期化は、通常同期バッチ移送によって行われます。バッチ・ジョブは、トランザクション・システムから新しい情報を抽出し、その情報を適切なフォーマットに変換して、データ・ウェアハウスにロードまたは移入します。

トランザクション・データの同期化

しかし、他の例ではトランザクション・データの同期化が必要です。たとえば、銀行の顧客が口座振替をセルフサービスの Web サイト上で行う場合、サイトは変更内容を即時にその顧客の口座に反映させる必要があります。銀行のバックオフィス・システムに格納されている口座情報は、Web サイトをバックアップしているデータベースとトランザクションを通して同期化する必要があります。場合によっては、銀行は顧客の口座情報のコピーを 2 つの場所（Web サイトをバックアップしているデータベースと銀行のバックオフィス・システム）で保持していないこともあります。むしろ、銀行は顧客情報を 1 つのシステムで保持し、複数のアプリケーションにそのシステムへのアクセスを提供します。この方法には、パフォーマンス、拡張性およびセキュリティに関する利点とトレードオフがあります。それについては次の項で説明します。

アプリケーションとビジネスの切離し

企業内統合または企業間統合が必要な第 2 の理由は、1 つの企業のビジネス・プロセスとアプリケーションを取引先のものと分離することにあります。

アプリケーション相互の切離し

ソフトウェアはますます複雑になり、それに対応して新規バージョンのソフトウェアへのアップグレードも難しいため、アプリケーションの切離しが必要になります。ソフトウェアが複雑になるにつれて、開発者は大きな問題を複数の小さい問題に分割して個別に解決する傾向が増えています。

開発者は、複数のモジュールを作成し、相互間のインタフェースを適切に定義し、それらのモジュールを結合して完全なアプリケーションを開発します。開発者は各モジュール内の小さい問題を解決することに集中します。適切に定義された標準化インタフェースによるアプリケーション・モジュール間の通信の制限と、モジュール間でデータを共有しないことによって、開発者は 1 つのアプリケーション・モジュールの変更またはアップグレードを他のアプリケーション・モジュールに影響を与えずに実行できます。プログラム間通信はアプリケーションの置換処理を簡素化し、アプリケーションの変更、データ・センターの再配置またはアプリケーションの完全アウトソーシング化などの意思決定の影響を最小限に抑えます。

ビジネス相互の切離し

さらに、ソフトウェアの複雑さは、企業間取引における企業相互間の通信方法に影響を与えます。各企業は、その取引先を、適切に定義された標準インタフェースを備えたサービスの提供者として期待しています。開発者は、モジュールごとに標準インタフェースを定義するアプリケーション・コンポーネント・モデルとモジュール間の通信を行う標準化プロトコルを使用して、プログラム間通信を標準化します。

同じ方法で、企業間の通信も現在標準化が進んでいます。つまり、通信に使用する標準インタフェースの定義（XML ベースのビジネス・オブジェクト・ドキュメントとして獲得）および RosettaNet や OAGIS などの標準ネットワーク・プロトコルを使用した標準化が進んで

います。ある企業の内部ビジネス・プロセスを別の企業の内部ビジネス・プロセスと切り離すことによって、各企業はその取引先に影響を与えずに内部プロセスを変更できます。

たとえば、企業は自社の内部発注書承認プロセスを変更できます。それは、サプライヤが発注書を標準インターネット・プロトコル上の標準フォーマットで企業に送信するため、その発注書がその企業の発注書承認プロセスから切り離されているためです。このような分離は、企業間取引の基本要件です。

マルチステップ・ビジネス・プロセスの簡素化

ビジネス・プロセスの自動化とは、業界ではよくストレート・スルー・プロセッシングと呼ばれますが、マルチステップ・ビジネス・プロセスを簡素化するプロセスを定義することです。自動化によって、アプリケーション相互間での直接通信が可能となり、人間の介入がなくなります。

たとえば、オンライン取引を行っている大部分の小規模な電子ストアフロントや企業は、各自のフロントエンド Web ストア・データベースが受け入れた新しい注文をバックオフィス財務システムやサプライ・チェーン・システムに移送するために、ファイル転送プロトコル（FTP）などの手動プロセスを使用しています。このような手動による介入によって次の 3 つの問題が発生します。

- 人為ミスの可能性が生じます。
- 注文処理のスピードと効率が低下します。
- 注文量が増えるにつれてストアフロントの運営コストが上昇します。

大規模な Web サイトでは、多数の企業統合テクノロジを使用して、このマルチステップ・ビジネス・プロセスを自動化し、簡素化しています。簡素化によって、ストアフロント・データベースは、最初に注文内容を財務システムに自動的に伝播し、顧客のクレジット履歴を検証してから、サプライ・チェーン・システムに伝播して製造と搬送のプロセスを開始します。ビジネス・プロセスの自動化は、企業のコスト削減、顧客満足の向上、ビジネス・プロセスのスピードアップおよび競合他社に対する迅速な対応に役立ちます。

統合という観点から、マルチステップ・ビジネス・プロセスは、次の 2 つの方法で簡素化できます。

同期通信：要求と応答

マルチステップ・ビジネス・プロセスの一部を構成する特定のアプリケーションは、要求と応答の構造を使用してリンクできます。たとえば、Web ストアフロントを設立した企業は、顧客の信用評価や自社製品の購買履歴をチェックしてから、チェックアウトに進む許可をその顧客に与えることができます。この場合は、ビジネス・プロセスの 2 つのステップ、つまり、Web ストアフロント・イベントと信用評価アプリケーションが要求と応答の構造でリンクされています。ストアフロントは要求を信用評価アプリケーションに送信し、続行する前に応答を要求します。ビジネス・プロセスのこの 2 つのステップは、論理的には大型複合アプリケーションの一部となっているため、相互に同期的に通信する必要があります。

対話型複合アプリケーションは、最も緊密に結合された統合プロセスを意味します。このアプリケーションは常にリアルタイムで動作します。複合アプリケーションが急速に広まっている理由は、企業がフロントエンド・マーケティングとサービスの提供に力を入れているためです。複合アプリケーションは、通常エンド・ユーザーには単一の Web アプリケーションのように見えますが、背後では、1 つ以上のメインフレーム・トランザクションを起動したり、パッケージ・アプリケーションや Windows NT 上のアプリケーションへのコールを行っています。

非同期通信

マルチステップ・ビジネス・プロセスの一部を構成する大部分のアプリケーションは、要求と応答の構造ではなく、非同期でリンクされています。

マルチステップ・プロセスは、非常に多くのビジネス・ニーズに対応している、最も一般的な統合形式です。マルチステップ・プロセスでは、各ステップがそのプロセス内の前段階における別のシステムの動作結果であるため、アプリケーションは論理的に独立しています。たとえば、Web ストアで受け入れた注文は、追跡管理可能な財務アプリケーションに送信されてから、サプライ・チェーン・アプリケーションに送信され、サプライ・チェーンの計画プロセスが開始されます。Web ストアが認識する必要があるのは、そのメッセージがバックオフィス・アプリケーションに確実に配信され、購入品が購買キューの受注順序に従って 1 回限りで搬送されることのみです。

様々なビジネス・プロセスのステップで構成されている異なるアプリケーション間の通信は、メッセージ・ミドルウェアを使用して、非同期で実行されます。非同期通信には次のような利点があります。2 つのアプリケーションを結合します。アプリケーションをネットワークやシステムの障害から切り離し、各アプリケーションを別のアプリケーションのソフトウェア障害から守ります。

非同期通信の使用は、企業内のビジネス・プロセスを簡素化するために増加しています。非同期通信は、企業内で有用なだけでなく、企業間取引に必要なビジネス・プロセスの企業間リンクの基盤となります。

まとめ : エンタープライズ・アプリケーションの全体図は、個別に設計されたシステムの寄せ集めを拡大したものに近くなっています。一体として設計されたシステム・セットで企業全体のすべてのビジネス機能を実装することは現実的ではありません。ユーザーの要件は、本質的に非常に複雑で動的なため、1 つの設計チームがすべてのソリューションを提供することは不可能です。一般的な企業では、数年前に社内で作成したアプリケーション、その後購入した独立ソフトウェア・ベンダーの複数アプリケーション・パッケージ、エンド・ユーザーのクライアント / サーバー・アプリケーションおよび増えつつあるインターネットとイントラネット・アプリケーションなどを併用して対処する必要があります。ただしこの先、統合化されていないアプリケーションを受け入れることはできません。これは、企業の管理者は、自社システム間における統合のレベルアップを要求しているためです。これらの異機種間システムを統合するタスクが、今日の IT 組織の中心的職務です。その結果、企業では、データの整合性、アプリケーションの切離しおよびマルチステップ・ビジネス・プロセスの自動化などの技法を組み合せて、自社の統合ニーズに対応する必要があります。

E-Business 統合テクノロジとアプローチ

この項では、統合に使用できる各種のテクノロジ代替案とその代替案を使用可能にする統合アプローチを検証します。E-Business 統合は、データの整合性と同期化、アプリケーションの切離しおよびマルチステップ・ビジネス・プロセスの自動化という 3 種類の基本的関係によって構成されることはすでに説明しました。ここで、E-Business 統合に関する次の 3 つの主要テクノロジの代替案について考えてみます。

- データ・レプリケーションとデータベース・ゲートウェイを使用したデータの整合性と同期化のテクノロジ
- コンポーネント指向の開発機能、オブジェクト・リクエスト・ブローカおよび同期統合方法論
- メッセージ指向ミドルウェアと非同期統合機能

これらの統合テクノロジは、それぞれが特定の種類の統合問題に適しています。すべての統合問題に最適な、単一の統合方法論はありません。

最初に、3 つの異なる種類の統合テクノロジを検証し、次に特定の統合問題の解決に最適なテクノロジをそれぞれ説明します。この項には次の内容が含まれています。

データの整合性と同期化のテクノロジ

データの整合性パターンの目的は、複数のシステムに格納されている冗長なデータから事実を取得することにあります。システム間におけるデータの同期化には、多数の異なるメカニズムが使用されていますが、それらのメカニズムは、次の 2 つのカテゴリに大きく分類できます。

データ移動テクノロジ

異なるシステム間でデータを同期化する方法の 1 つは、当該システム自体の間でデータを移動することです。事実、バッチ・データ転送は、データ同期化のためのデフォルトのアプローチです。従来から、多くの企業は、FTP などの手動プロセスを使用してシステム間でデータを移動しています。データベース・レプリケーション・テクノロジもデータベース内に格納されているデータを同期化します。バッチ転送または事前に計画されたトランザクションでは、手動プロセスでもデータは効果的に同期化されます。

しかし、E-Business によって、情報の一貫性を維持する必要性が増加し、リアルタイムの介入が頻繁に必要になります。その結果、E-Business には、データベース・レプリケーションなどの自動化されたデータ移動テクノロジが必要になります。頻度の低いデータ・バッチの実行に依存せずに、個々の更新を認識した場合は即時にその更新をほぼリアルタイムに転送して、データを同期化する必要があります。

異機種間データ・アクセス・テクノロジ

システム間データ移動に対する 1 つの代替案は、データを 1 箇所で保持し、複数の異種アプリケーションからそのデータにアクセスする方法です。この方法では、常時同期化しておくためにシステム間でデータを移動する必要がなくなります。システム間でのデータ移動は、システムの増加につれてますます複雑になります。簡単にいえば、異なるシステムがアクセスする対象の全データを、そのデータのサブセットが必要なすべてのアプリケーションがアクセスできる 1 つの中心的な位置に格納します。

この方法によって、継続的なデータ移動は不要になります。ただし、単一のデータベースのみでデータ同期化の問題を解決できるかどうかを判断するには、次の 3 つの基本的な問題を考慮する必要があります。

- データ・アクセスに関する総合的なパフォーマンスと拡張性の要件 : データ・アクセスが必要なアプリケーションの数と頻度はどの程度か
- アプリケーションの配布方法は集中か分散か

  たとえば、アプリケーションが低帯域幅のワイド・エリア・ネットワーク（WAN）全体に分散されている場合、WAN 全体でのデータの移動では円滑性や効率性が失われます。したがって、この場合は、バッチまたは手動転送によってデータをシステム間で移動する必要があります。異機種間データ・アクセス・テクノロジは、主としてメインフレーム・データベース、階層データ・ストア、フラット・ファイル・ストアなどの異機種間データ・ストアへのアクセスを可能にするためのゲートウェイです。
- 実行するトランザクションにマルチステップ・ビジネス・プロセスを含めるかどうか

  たとえば、1 つの注文によって、一定日数の間に受注入力、売上管理、製造、サプライ・チェーン計画と実行のシステムなどの論理的に関連している一連のトランザクションが作成されます。場合によっては、バッチ・データ転送テクノロジがマルチステップ・ビジネス・プロセスの自動化に使用されます。

重要なビジネス・プロセスの場合は、ワークフロー・システムなどのワークロード管理ツールを使用してワークフローを自動化します。重要度の低いプロセスや適切に管理されていない重要なプロセスの場合は、人間の介入を含んだバッチ・データ転送を使用して制御することがよくあります。高速のエンドツーエンド・プロセスのニーズが高まるにつれて、バッチ転送がボトルネックになる可能性があります。エンドツーエンド・プロセスの自動化などの計画には通常、個別のイベントを即時に他のシステムに送信するためのメッセージ機能が必要です。データ移動とデータ・アクセスのテクノロジは、異なるシステム間で情報を同期化し、一貫性のあるグローバルな情報ビューを提供するためには最適なテクノロジです。

コンポーネント指向の開発テクノロジ

アプリケーションを相互に切り離すには、アプリケーションの変更時にも安定性が確保できるように、適切に定義されたインタフェースの小さなセットを使用してアプリケーション間の通信を制限します。モジュール開発の代表的な例では、コンポーネントは、適切に定義された標準インタフェースを使用し、プログラム間通信を介して対話します。アプリケーションはデータを共有しません。プログラム間の対話を切り離してパブリック・インタフェースの小さなセットにすると、コンポーネントはビジネス・ロジックをカプセル化できます。モジュール性の持つ重要な付加的な利点は、コンポーネントを再利用できることにあります。たとえば、コンポーネントが正しく設計されている場合は、これらのコンポーネントをプラグ・アンド・プレイ方式で組み立ててアプリケーションを構築できます。

現在業界で使用されている 3 つのプライマリ・コンポーネントは、次のとおりです。

- 最も広く使用されているのは、Java2 Enterprise Edition（J2EE）または Enterprise JavaBeans（EJB）のコンポーネント・モデルで、Sun Microsystems 社、オラクル社および IBM 社などの大手ソフトウェア・ベンダーがサポートしています。
- Object Management Group（OMG）内の Common Object Request Broker Architecture（CORBA）標準化委員会も CORBA コンポーネント・モデルを促進しています。
- Microsoft 社には独自の COM+ オブジェクト・モデルがあります。

コンポーネント・ベースの技法を利用したアプリケーションの開発が増えるにつれて、コンポーネント間相互の通信方法が、基本的な統合の問題になります。通信方法には次の 2 通りがあります。

同期通信

3 つの共通コンポーネント・モデルがそれぞれ独自のホスティング環境をアプリケーション・コンポーネント（コンテナと呼ばれます）に対して提供します。コンテナはコンポーネントを操作可能にするサービスのセットを提供します。サービスには、トランザクション・サービス、ネーミング・サービスとディレクトリ・サービス、ブローカ・サービスと取引サービスがあります。EJB の場合、これらのサービスは、Enterprise JavaBean Transaction Server と呼ばれるコンテナによって提供されます。また、CORBA の場合は、オブジェクト・リクエスト・ブローカまたは ORB と呼ばれるコンテナによって提供されます。COM+ の場合、これらのサービスは、Windows NT オペレーティング・システム自体によって提供されます。

これらのコンテナは、同期リモート・プロシージャ・コールのメカニズムを使用してコンポーネント間の通信を管理します。同期通信は、切り離す必要のあるアプリケーションが要求と応答の構造で相互に関連している場合、理想的な通信方法です。ORB のようなコンポーネント・ミドルウェアは、プログラム間インタフェースの定義をドキュメント化し通信を管理するため、このタイプの統合には最適です。

非同期、メッセージ指向ミドルウェア

非常にまれですが、異種アプリケーションで一方向の非同期イベント通知を従来の ORB コールを使用して実装することも技術的には可能です。標準 ORB は、双方向の要求応答式の

対話を中心にして設計されています。つまり、疎結合アプリケーションに必要な高度なメッセージ機能が欠けています。特にマルチステップ・ビジネス・プロセスを容易にするための機能がありません。このような統合の実現に適しているのが、メッセージ指向のミドルウェア・ソリューション、特に新世代の統合ブローカです。

メッセージ指向ミドルウェア・テクノロジ

企業内および企業間取引内でのビジネス・プロセスの自動化が進むにつれて、新世代のミドルウェア・テクノロジが登場してきました。このデクノロジは、アプリケーションとビジネスの疎結合を行う非同期通信がベースになっています。メッセージ機能の基本原理は、情報の提供者を情報の利用者と切り離すことです。その結果、アプリケーションの追加、削除または変更を別のシステムに影響を与えずに実行できます。

メッセージ指向のミドルウェアによって、アプリケーションとビジネス・プロセスはアプリケーション相互間でメッセージを送信して通信することができます。アプリケーションはミッション・クリティカルであるため、ミドルウェアが、1 回限りのメッセージ配信の保証や格納キューイング、転送キューイングなどの機能を提供します。さらに、このメッセージ機能プラットフォームには、高度なメッセージ・ルーティング機能と配布機能が追加されています。追加された機能は、次のとおりです。

- コンテンツに基づいてメッセージを異なるロケーションに送信する、コンテンツ・ベースのルーティング。
- サブジェクトに基づいてメッセージを異なるロケーションに送信する、トピック・ベースのルーティングまたはサブジェクト・ベースのルーティング。
- 送信者がメッセージのサブスクライブに関心のあるサブスクライバのキューにメッセージを公開するだけの、パブリッシュ・サブスクライブ・ルーティング。パブリッシャにはメッセージの受信者がわかりません。

Enterprise JavaBeans などの大部分のコンポーネント・モデルには、次々と非同期通信インタフェースが追加されているため、それを使用すると疎結合方式で相互通信が可能になります。基本的なメッセージ指向ミドルウェアを分散アプリケーション内で使用し、1 つのアプリケーションを別のアプリケーションに統合できます。これは、アプリケーションが本来接続不能なためです。非同期のメッセージ機能は、マルチステップ・ビジネス・プロセスの 1 部分を構成するアプリケーションを疎結合する必要がある場合（要求応答方式で関連していない場合）に最適な機能です。

メッセージ・ミドルウェアは、サーバーとアプリケーション間のダイレクト・ゲートウェイ接続とは根本的に異なる方法で異種アプリケーションを接続します。ダイレクト・ゲートウェイは特に、1 つか 2 つのバックエンド・アプリケーションを Web フロントエンドを使用して拡張する場合など、計算された要求応答式の対話には最適です。

メッセージ・ベースのソリューションは、データの整合性またはマルチステップ・プロセス・アプリケーションのいずれかを必要とする非同期アプリケーションと複数の異機種間参加者を持つシステム化された複合アプリケーションに最適です。メッセージ機能によって通信セマンティックと管理の層が増加します。一部のプロジェクトには、このような複雑さは不要です。しかし、メッセージ機能は、1 つのソリューション内で整合性、マルチステップ

および複合パターンを処理する、機能が豊富で包括的なインフラストラクチャを提供します。コンポーネントは、アプリケーション内のアーキテクチャを統制しているため、メッセージ・ベースの統合インフラストラクチャへの接続にしばしば使用されます。

最終的には、メッセージ中心の統合は、総合的な企業レベルのハブを介してすべてのアプリケーションを相互に接続します。すべてのアプリケーションは、この統合ハブに情報を公開します。この場合、送信先、受信相手または受信者が望むフォーマットなどを知る必要はありません。アプリケーション全体のポートフォリオの柔軟性は、接続論理と配信指示がアプリケーション自体ではなくインフラストラクチャに常駐しているため、そのまま維持されます。

メッセージ機能によって、プログラムはメッセージをキューに置いたまま作業を続行し、生成側の問合せサーバーとして動作できます。キューイング・システムは適切な受信者にメッセージを確実に配信します。コンシューマとしての受信者は、キューから要求を取り出し、それに基づいて行動します。サービスの要求とサービスの提供を切り離すと、メッセージ機能による効率が上がり、複合タスクをスケジュールするインフラストラクチャが提供されます。メッセージ機能では、プログラムは相互間で直接通信しません。プログラム相互間の接続は切断されているため、異なるアプリケーション・プログラム間の通信ハブとしての役割を持つメッセージ・システムを介して通信します。したがって、メッセージ機能は、多数のプログラム相互間の通信に役立つ代表的な例を提供します。

3 つの特定アプリケーションの設計上の問題が、多くの場合、アプリケーション間通信にメッセージ・サービスを使用する動機となります。

要求応答なしの要件

メッセージ機能は、プログラムがメッセージを別のプログラムに送信した後、独自の作業を続行できるアプリケーションに最適です。つまり、最初のプログラムは 2 番目のプログラムからの応答を待機しないで先に進みます。さらに、メッセージの取出し時まで作業を続行できるアプリケーションに最適です。

最初のプログラムが先に進む前に応答が必要な場合、メッセージ機能は適切な通信メカニズムではありません。この場合は、Net8 や CORBA RPC などの同期通信メカニズムが適切です。同期メカニズムでは、最初のプログラムは要求を 2 番目のプログラムに送信した後、2 番目のプログラムが応答を送信するまで待機します。最初のプログラムは、この応答で次のプロセスを実行します。

対照的に、メッセージ機能はこのような関係が不要なアプリケーションに最適です。つまり、最初のプログラムはメッセージをキューに置くだけで、2 番目のプログラムからの応答を待機せずに作業を続行します。

これらの 2 つのモデル、同期通信とメッセージ機能は同じアプリケーション内で一緒に使用できます。たとえば、出荷アプリケーションの受注入力プログラムは、別の顧客管理プログラムと通信して、その注文の受け入れ前に顧客の妥当性をチェックします。この場合は、同期通信が必要です。これは、受注入力プログラムはその注文の処理を続行する前に顧客管理プログラムからの応答が必要なためです。

ただし、受注入力アプリケーションは、注文の完了時に、注文を顧客に送付する必要があることを出荷プログラムに通知します。このような通信は、メッセージ機能を使用して最適に

処理されます。受注入力アプリケーションは、出荷プログラムからの応答を待たずにその注文の処理を進めます。

独立した処理

プログラム内通信にメッセージ・システムを使用する動機となる 2 番目の要因は、アプリケーションの配布アーキテクチャです。同期通信を有効にするには、すべてのプログラムが同時に実行中で使用可能であることが必要です。これらのプログラムを接続するネットワークが使用可能であり、プログラムを実行するシステムが起動していて、すべてのプログラムが同時に使用できることが必要です。配布環境のノードに 1 つでも信頼できないものがあると、メッセージ機能はさらに堅固なソリューションを提供します。メッセージ機能は、プログラム間のタイム依存関係を解消します。その結果、アプリケーションはプログラム障害の影響を受けにくくなります。ネットワーク、システムおよびアプリケーションで障害が発生した場合に、遅延実行を適切に動作させるためには、サービス要求を表すメッセージを永続的に格納し、正確に 1 回のみ処理する必要があります。メッセージを保持できる機能は、次の 4 つの理由で企業におけるメッセージ交換インフラストラクチャの基本です。

- **着信時のメッセージの処理不能 :** アプリケーションは、クライアントから同時に着信した未処理のメッセージを処理する必要があります。アプリケーションにすべての要求を即時に処理するリソースがない場合があります。メッセージ・システムには、永続キューにメッセージを格納し、後で受信者が処理できる時点でそのメッセージを配信する機能が必要です。
- **メッセージ機能のスケジューリング :** さらに、メッセージ・システムには、優先順位に従って処理できるメッセージの永続性が必要です。つまり、後から着信するメッセージが、前に着信したメッセージより優先順位が高かったり、前に着信したメッセージが、アクションの実行前に後から着信するメッセージを待機する必要がある場合があります。このようなメッセージの優先順位は、一定期間変更することもできます。たとえば、特定キューのメッセージを、特定の時間枠の間、他のキューのメッセージより重要視することができます。メッセージの永続性によって、優先順位の高いキューのメッセージを最初に処理できます。一方、優先順位の低いメッセージは格納して後で処理できるため、優先順位の高いメッセージの妨げになることはありません。
- **メッセージの監査 :** また、メッセージの永続性は、メッセージの制御コンポーネントがペイロード情報自体と同じくらい重要になる場合があるため、非常に重要です。たとえば、メッセージの受信と送信タイムが、メッセージの重要部分になる場合があります。E-Commerce 環境では、メッセージの永続性によって、多様な顧客からの注文情報が格納されるため、その情報に問い合せると、需要がピークに達する期間を識別したり、注文状態を判断できます。したがって、メッセージは実行後も依然として重要な場合があります。永続的なメッセージの格納は、情報の格納、問合せまたは監査を確実に実行するためには非常に重要です。
- **障害 :** メッセージ機能クライアント間の通信リンクは、常時使用可能であるとは限りません。他の用途に予約されている場合もあります。処理リソースの不足または障害のいずれかの理由でメッセージを即時に処理できない場合、メッセージ・システムはメッセージを永続的に格納し、リソースが使用可能になった時点で配信できることが必要で

す。エンタープライズ・アプリケーションの統合時には、各メッセージが受信者ごとに正確に 1 回配信されるという、メッセージの配信保証が非常に重要です。

2

方法論とソリューション

この章では、E-Business 統合の方法論を説明し、Oracle Integration Server による統合の課題への取り組み方の例を示します。この章には次の項目が含まれています。

適切な E-Business 統合方法論の選択

データ同期化テクノロジ、コンポーネント指向の開発テクノロジおよびメッセージ指向のミドルウェアの 3 つの主要な統合テクノロジについてはすでに検証しました。ここでは、これらのテクノロジを、3 つの基本的な統合問題、つまり、システム間におけるデータの同期化、アプリケーション相互間の切離し、企業内と企業間のマルチステップ・ビジネス・プロセスの自動化に最適にマップする方法を考えます。

方法論の選択には通常、統合の具体的な実装環境に関する詳しい調査が必要ですが、広範囲をカバーする特定のアーキテクチャ原理を使用して、統合の意思決定の指針とすることができます。

この項には次の内容が含まれています。

システム間におけるデータの同期化

システム間でデータを同期化する方法を決める場合は、異なるシステム間でのデータ移動が必要か、セントラル・ロケーションから各種システムへのデータ・アクセスが必要かどうかによって、基本アーキテクチャの選択肢が異なります。データ・アクセスを提供する必要がある場合、統合テクノロジの理想的な選択肢は、ゲートウェイを使用して、データが常駐しているデータベースやレガシー・システムにアクセスする方法です。

システム間でのデータ移動が必要な場合は、データベース・レプリケーションと非同期メッセージ機能という 2 つの異なるメカニズムを使用できます。テクノロジの選択に影響を与える要因は、次の 3 つです。

- 第 1 に、データの同期化が、1 つか 2 つのバックエンド・アプリケーションを Web フロントエンドに拡張するような簡単な方法で実行できる場合は、情報のデータベース・レプリケーションのみで十分です。同様に、情報をセントラル・データベースから複数の小規模ワークグループ・データベースに分散する場合は、データベース・レプリケーションで十分です。この 2 つの場合、メッセージ・ベースのソリューションを選択すると、複雑さが増大します（たとえば、専用ミドルウェアの配布が必要になります）。このソリューションはもっと複雑なアプリケーションに適しています。
- 第 2 に、レプリケーションが使用できるのはほとんどの場合、情報の送信者と受信者が同種のデータベース、たとえば、両者とも Oracle データベースであるときです。データベースには同じスキーマが必要です。データがデータ移動の一部として複雑な変換を経由する必要がある場合は、非同期メッセージ・テクノロジのほうが適切です。
- 第 3 に、データベース・レプリケーションを使用するためには、受信側アプリケーションは送信側アプリケーションにそのスキーマへのダイレクト・アクセスを提供する必要があります。スキーマにそのようなダイレクト・アクセスを提供する場合、次の 2 つの問題が発生します。つまり、2 つのアプリケーションは明白に定義されたインタフェー

ス・セットを介して通信しないため、ダイレクト・スキーマ・アクセスはカプセル化とコンポーネント・ベースの開発に違反します。さらに、ダイレクト・スキーマ・アクセスは、アプリケーション・レベルのセキュリティ・ポリシーを回避する可能性があります。

このような問題はありますが、2 つのデータ・ソース間の単純なデータ同期化が唯一の統合要件である場合は、データベース・レプリケーションの使用によって、アプリケーションの作成と配布方法が簡素化されるため、おそらくこれが最適な選択です。

アプリケーションとビジネスの相互の切離し

統合シナリオの焦点が、主としてアプリケーションとビジネスを相互に切り離すことにある場合、データ・レプリケーションやデータベース・ゲートウェイなどのデータ同期化テクノロジは適切ではありません。アプリケーションを相互に切り離すには、アプリケーション間の通信を適切に定義されたパブリック・インタフェースの標準セットに限定する必要があります。定義上、データ同期化テクノロジはカプセル化に違反しています。したがって、アプリケーションの切離しには適していません。

アプリケーションを相互に切り離す一方で、相互間の通信を容易にする必要がある場合、主な選択肢は、同期通信機能と非同期通信機能の中間に位置する機能です。次に、どの通信メカニズムを使用するかを決定する 3 つの要因を示します。

- 第 1 に、2 つのアプリケーション・コンポーネントが大規模な複合アプリケーションの一部で、ともに要求と応答の構造で作動している場合は、オブジェクト・リクエスト・ブローカ（ORB）などの非同期のコンポーネント指向ミドルウェアの機能や CORBA IIOP などの通信プロトコルが最適の選択肢です。この場合、両アプリケーションは、CORBA Interface Definition Language（IDL）で適切に定義されたパブリック・インタフェースを備えた CORBA サービスとしてラップされ、ORB と呼ばれるミドルウェア・ハブを介して通信する必要があります。ORB は、2 つのアプリケーションに対して次のサービスを提供します。
  - 両アプリケーションをサービスとして登録します。
  - 各アプリケーションのパブリック・インタフェースを他のアプリケーションにアクセス可能にします。
  - 各プロトコルの要求をアプリケーション相互間でルーティングします。
  - ネットワーク上のオブジェクトのロケーションを検索して、そのオブジェクトを検出します。
  - 要求の受信時にオブジェクトを起動します。
- 第 2 に、2 つのアプリケーション・コンポーネントが要求と応答の構造で関連しているのではなく、マルチステップ・ビジネス・プロセス内のステップを構成している場合は、非同期のメッセージ指向ミドルウェアが最適の選択肢です。このような状況での選択肢は主に、配布の必要があるメッセージ機能環境の複雑さやメッセージ・アーキテク

チャによって異なります。これらの選択肢は、マルチステップ・ビジネス・プロセスの自動化の場合の選択肢と似ています。これについては、次の項で説明します。

- 第 3 に、統合要件が、2 つの企業のビジネス・プロセスをインターネットで接続することである場合、同期的統合は実現不可能です。疎結合のメッセージ・ベース統合が唯一のソリューションです。

マルチステップ・ビジネス・プロセスの自動化

マルチステップ・ビジネス・プロセスを自動化する場合、アプリケーションは疎結合構造で相互に通信する必要があります。この場合、統合テクノロジの唯一の適切な選択肢は、非同期のメッセージ指向ミドルウェアです。ミドルウェア・ソリューションの配布を選択する場合、主なアーキテクチャ上の意思決定は統合問題の複雑さによって異なります。次の 4 つの重要な問題を考慮する必要があります。

- 統合トポロジ
- メッセージ・アーキテクチャ
- データ変換
- ビジネス・プロセス管理とワークフロー

統合トポロジ

メッセージ・ミドルウェアを配布する場合は、アプリケーションのリンクに Point-to-Point インタフェースを使用するか、またはハブ・アンド・スポーク・アーキテクチャを使用するかを判断する必要があります。2 つのアプリケーションを接続する場合、Point-to-Point 接続は単純ですが、2 つ以上のアプリケーションの接続が必要な場合は、すぐに管理不能となります。

このような場合は、ハブ・アンド・スポーク・アーキテクチャを使用します。このアーキテクチャでは、アプリケーションは直接相互に接続されません。各アプリケーションは、すべてのアプリケーション間の接続を行うハブに接続されます。アプリケーションの 1 つを変更またはアップグレードする場合は、ハブとの関係のみを変更するので、それ以外の統合対象アプリケーションにはまったく影響を与えません。

メッセージ・アーキテクチャ

次に、使用するメッセージ・アーキテクチャを次の 3 つの問いに基づいて判断する必要があります。

メッセージを永続的に格納する必要があるか？ ビジネス上重要なメッセージで、たとえば、企業間のトラブル解決または情報の流れの追跡などの目的で監査や追跡の対象となる場合、そのメッセージのヘッダーとコンテンツはデータベースに永続的に格納する必要があります。このようなメッセージは、標準的な意思決定支援ツールを使用して格納および分析することができます。ビジネス上重要でないメッセージに対しては、メッセージ・ブローカ機能によって揮発性キューイング機能を提供できます。

メッセージを保証付き配信で伝播する必要があるか？ メッセージがビジネス上重要な場合、またはメッセージ・システムが 2 つのミッション・クリティカルなアプリケーションを接続している場合は、メッセージを保証付きの 1 回限り、順序付き配信で伝播する必要があります。それ以外の場合は、メッセージを非永続的に伝播するのみでかまいません。

メッセージのルート方法は？ メッセージは、そのサブジェクトやトピック、コンテンツやペイロード情報に基づくか、またはパブリッシュ・サブスクライブ・メソッドを使用して、アプリケーションとビジネス・プロセス間でルートできます。

大部分の E-Business 統合シナリオでは、メッセージはそのコンテンツに基づいてルートされます。つまり、多くの場合、適切な宛先に送信する前に、特定のワークフローを起動して特定の方法で処理されます。

データ変換

ビジネス・プロセスの一部を構成するアプリケーションは通常、異なるデータ・フォーマットでデータを格納および管理します。たとえば、Oracle E-Business Suite のパッケージ・アプリケーションは、SQL フォーマットでデータを格納および管理します。SAP アプリケーションは、ASCII から派生した iDOCs フォーマットでデータを操作します。

アプリケーションを相互に接続する場合は、次の 4 つのデータ変換に関する問題に対処する必要があります。

- **データ型変換**とは、データを元のアプリケーションのフォーマットからターゲット・アプリケーションのフォーマットに変換することです。たとえば、Oracle の CRM アプリケーションを SAP アプリケーションに接続する場合、Oracle データ型を SQL フォーマットから、iDOCs に最適な ASCII フォーマットに変換する必要があります。
- **セマンティック変換**とは、顧客名を Oracle フォーマットから SAP フォーマットに変換することです。たとえば、Oracle では顧客名を、名、姓、ミドル・イニシャルの 3 つのフィールドで表現します。SAP では顧客名を単一のフィールドで、姓、カンマ、名の順に表現します。セマンティック変換を適用して姓の次に名を連結する必要があります。
- **機能変換**とは、送信側アプリケーションのビジネス・イベントを受信側アプリケーションのビジネス・イベントとインタフェースにマップすることです。たとえば、発注書が Oracle アプリケーションから SAP アプリケーションに送信されると、特定のビジネス・イベントのセットが開始され、特定のパブリック・インタフェースのセットが起動される必要があります。メッセージをアプリケーション相互間で伝播するには、このような機能変換情報を獲得する必要があります。

  データ変換に通常使用されるアプローチは次の 2 つです。最初のアプローチは、データを元のアプリケーションのフォーマットからターゲット・アプリケーションのフォーマットに直接変換します。2 番目のアプローチは、最初に標準的な中間フォーマットに変換してから、ターゲット・アプリケーションのフォーマットに変換します。直接変換の方が高速ですが、中間フォーマットへの変換によって、1 つのアプリケーションを他のアプリケーションで変更する必要がなくなります。たとえば、Oracle CRM アプリケーションを Baan、SAP または PeopleSoft などのアプリケーションに接続する場合は、中間表現に情報をマップすることによって、Oracle CRM アプリケーションを Baan ア

プリケーションにアップグレードする必要がなくなります。必要な唯一の変更は、中間表現と Baan アプリケーションとの間のマッピングを変更することです。

- **ロケーション変換**とは、データ変換を実行するロケーションを決定することです。場合によっては、データ変換はスポークにあるアプリケーションの近くで実行されます。ハブは、変換済みメッセージを 1 つのアプリケーションから別のアプリケーションに送信するのみです。ただし、多くの場合、データ変換は次の理由でハブで行う必要があります。
  - 監査対象のメッセージは、事前に変換済みのフォーマットで格納されています。
  - 多数の異なるターゲット・アプリケーションに送信されるメッセージは、元のアプリケーション内でハブに送信される中間フォーマットに変換されます。次にハブは、各種の変換機能を適用して、その情報を各ターゲット・アプリケーションのフォーマットに変換します。
  - コンテンツに基づいて異なるロケーションにルートされるメッセージは、ハブで変換された後、そのハブによってルーティングされます。ただし、これはハブ自体の機能や、接続の必要なアプリケーションにハブが提供しているサービスの種類によって異なります。

ビジネス・プロセス管理とワークフロー

最後に、メッセージをそのまま 1 つのアプリケーションから別のアプリケーションに送信するか、または伝播前の処理が必要なのかを決定する必要があります。たとえば、ある企業が発注書を自社のサプライ・チェーン・アプリケーションから他社の調達システムに送信する場合、その発注書に対する送信側企業の購買マネージャの承認が必要な場合があります。承認が必要な場合、メッセージ・ミドルウェアはメッセージを取引先に転送する前にワークフロー・アプリケーションを起動する必要があります。この場合、メッセージ機能プラットフォームにはワークフローのビジネス・プロセスを管理する機能が必要です。

アプリケーション統合 : ソリューションの範囲

すべての統合シナリオには、3 つの基本的な統合問題、つまり、システム間におけるデータの同期化、アプリケーションとビジネスの相互の切離し、マルチステップ・ビジネス・プロセスの自動化が含まれることは前に述べました。統合には一連の異なるテクノロジが必要です。それぞれのテクノロジが特定タイプの統合問題に適合しています。

オラクル社は、統合というものが、単一のテクノロジで解決できる単一のごく限定された問題ではないことを認識しています。この複雑な問題を完全に解決するには、広範囲のテクノロジが必要です。そして、すべてのテクノロジが透過的に統合される必要があります。この項では、代表的な統合シナリオを示しながら、複数の選択肢を解明します。これらのシナリオは、包括的なソリューションを構成する様々なタイプの統合テクノロジを理解する上で役に立つはずです。

この項には次の内容が含まれています。

データの統合

シナリオ : 企業には多数のアプリケーションがあり、それぞれに独自のデータベースがあります。これらのアプリケーションは、相互に関連するデータベース内の情報を頻繁にしかも個別に変更しています。

問題 : アプリケーションは、別のアプリケーションのデータベース内の最新情報をどのようにして取得するのか？ たとえば、受注入力アプリケーションが顧客のアドレスを使用するには、売掛金アプリケーションと同じ顧客データにアクセスする必要があります。

ソリューション : Data Access Gateway とレプリケーション : ゲートウェイによって、アプリケーションは、他のデータベースに直接アクセスして必要な情報を簡単に取得できます。レプリケーションは、複数のデータベースの情報を自動的に同期化するため、各データベースには最新情報が蓄積されています。どのデータベースの変更内容も即時に他のデータベースに反映されます。

アプリケーションの統合

複数のアプリケーションを含んでいるビジネス・プロセスには、アプリケーション・ロジックとアプリケーション機能の統合が必要です。この統合を容易にするためには、アプリケーションは重要なビジネス情報を交換するために通信する必要があります。同期通信と非同期通信という 2 つの異なる通信モデルがあります。

機能インタフェース付き同期通信

シナリオ : 注文を受け入れるフロントエンドの E-Commerce Web アプリケーションを作成します。アプリケーションは、注文を受け入れる前に、クレジット・カードの認可、顧客の信用評価、在庫レベル、配送スケジュール、価格設定などの情報が必要です。これらのサービスの 1 つでも使用不能な場合、受注アプリケーションはその取引を完了できません。これは、後で再注文を顧客に依頼することを意味します。

問題 : 受注入力に必要なサービスは、他の専用アプリケーションによって提供されています。では、E-Commerce アプリケーションはどのようにしてこれらのサービスにアクセスす

るのか？ どのようにすればこれらのアプリケーションを統合してビジネス・プロセスを実装し、自動化できるのか？

ソリューション : 機能インタフェースをベースにした同期的な要求応答プロトコル : 各アプリケーションは、パブリック・インタフェースとして特定の機能セットを定義してサービスを提供します。これらのサービスを要求するために、アプリケーションは対応するインタフェースを起動します。機能インタフェースをベースにした要求応答プロトコルの例には、Remote Procedure Calls（RPC）、Common Object Request Broker（CORBA）、COM、Java Remote Method Invocation （RMI）などがあります。意味的に豊富なインタフェース、特に分散企業取引アプリケーションに向けた、Enterprise JavaBeans や COM+ などのコンポーネント・モデルを備えた規格が登場しつつあります。

メッセージ・ベース・インタフェース付きの非同期通信

シナリオ : 受注入力や製造、在庫管理、配分、課金など多数のアプリケーションを含んだエンドツーエンド注文実行プロセスを実装します。注文実行プロセスは、顧客の発注で使用する受注入力アプリケーションから始まります。その後、ビジネス・オブジェクトとビジネス・イベントはアプリケーションの間を流れていく必要があります。

問題 : これらのアプリケーションは、ワイド・エリア・ネットワーク（WAN）全域に分散され、異なる組織が所有しており、異なる内部アーキテクチャが存在しています。さらに、これらのアプリケーションは、単一機能のスタンドアロン・アプリケーションとして設計されているため、統合に関する計画は考慮されていません。アプリケーションを接続するネットワークは、信頼できない場合があります。アプリケーションを所有する組織が、他のアプリケーションを所有する組織に通知せずにアプリケーションを変更したり、再配布したり、置換することがあります。さらに、他のアプリケーションからの応答を必要としないで先に進むアプリケーションがあります。したがって、同期通信による統合は実行可能なオプションではありません。

ソリューション : メッセージ・キューイング付きの非同期通信 : アプリケーションは、キューを経由した情報をメッセージとして交換して相互に通信します。各アプリケーションは、入力として受け入れるメッセージのセットと出力として公開するメッセージのセットを他のアプリケーションに対して定義します。このメッセージは、顧客レコードなどのビジネス・オブジェクトや新しい出荷要求などのビジネス・イベントを表します。メッセージ・キューイング付きの非同期通信を使用すると、疎結合が使用可能になり、個々のアプリケーションはアプリケーション、ネットワークおよびシステムの障害から完全に切り離されます。

業界標準の Java Messaging Service（JMS）は、メッセージ・キューイング・インタフェースの一例です。

ビジネス・プロセスのモデル化と実行

シナリオ : 前の項で説明した注文実行のシナリオを考えてみます。アプリケーションは統合され、ビジネス・プロセス全体は自動化されていると仮定します。

問題 : ビジネス・マネージャがビジネス・プロセスを変更する必要があると決定します。たとえば、注文実行プロセスを作業員指向のシステムから、異なる承認プロセスが必要な Web ベースのセルフサービスに移行します。プロセスの流れを変更し、新しいアプリケーションを追加する必要があります。このビジネスの意思決定を実現するにはどのくらい時間がかかるのか？ また、意思決定から実装までの時間をどのようにして最短化するのか？

ソリューション : ビジネス・プロセスのモデル化と実行 : 最初に、ビジネス・アナリストがグラフィカルなモデリング・ツールを使用してビジネス・プロセス全体をモデル化します。次に、技術アナリストが詳細を書き込み、そのモデルを基礎となる統合インフラストラクチャにマップします。次に、そのモデルを検証し、リポジトリに生成します。最後に、Oracle Business Process Coordinator がそれを実行します。プロセスを変更する場合に必要なのは、上位レベルのモデルに対する変更のみです。変更の識別から新モデルの実行までのサイクル全体を迅速に実装できます。

ビジネス・プロセス・インテリジェンス

シナリオ : 前の項と同じです。

問題 : エンドツーエンド・ビジネス・プロセスが、個々のアプリケーションが適切にチューニングされているのに、期待どおりに実行されません。ビジネス・プロセスの全体ビューが使用できません。標準システム管理と監視ツールでは、個別のコンポーネントのパフォーマンス・メトリックのみが提供され、プロセス全体のパフォーマンス・メトリックが提供されません。パフォーマンスが市場の変動に伴って毎日および季節ごとに変化し、問題の正確な原因を特定することができないことが、問題をさらに複雑にしています。このような状況で、リソースを効率よく配布するには、どのように非能率とボトルネックを識別するのか？

ソリューション : このソリューションには、方針としてビジネス・プロセス・インテリジェンスを使用することです。全ビジネス・プロセスの全体ビューを取得する唯一の方法は、各トランザクションをエンドツーエンドで追跡することです。これは、トランザクションを後で再構築するためにすべての情報（データ、メッセージおよびビジネス・イベント）を追跡することを意味します。さらに、期間を拡張して、これらの情報を収集、格納、分析し、そのビジネス・プロセスに関するインテリジェンスを収集します。蓄積された情報を詳しく調べるとパターンが検出され、リソースの最適な配布方法を推察することができます。

企業間統合

前の項では、完全なアプリケーション統合ソリューションに必要な一連のテクノロジを説明しました。これまで、企業の大部分の統合プロジェクトの焦点は、企業内のアプリケーションの統合に向けられていました。企業の枠を越えてパートナやサプライヤとの統合を含むプロジェクトは少数で、適用範囲も限られていたため、FTP や電子メールなどの単純なメカニズムが使用されてきました。真にミッション・クリティカルな企業内部のトランザクションは、EDI、HL-7、SWIFT などの独自のプロトコルを使用したプライベートな付加価値通信網（VAN）上で行われています。

E-Business 統合には、顧客、サプライヤおよびパートナとの幅広く柔軟性のある動的な協力と協調動作が必要です。このタイプの統合を必要とする新しいビジネス・モデルの例は、次のとおりです。

- 企業の Web ベース・セルフサービス・アプリケーション。ビジネス・プロセスに関する統一された最新ビューをユーザーに提供する必要があります。
- バーチャルな拡張サプライ・チェーン
- オンライン・マーケットプレイスまたは企業間取引などのオンライン取引所を介しての調達
- 動的な注文実行
- アプリケーション・サービス・プロバイダによるアプリケーションのホスティング

統合は、低コストで広く普及しているインターネット上で行われる必要があります。それによって、企業はパートナ、サプライヤおよびベンダーを世界規模で選択できます。この必要性によって、ソリューションに対する追加要件が発生します。追加要件には、次の内容が含まれます。

- **エンドツーエンド・セキュリティ :** ビジネス・パートナには、実行するトランザクションはライフ・サイクル全体を通して完全に保護されているという確信が必要です。情報は、データベースに常駐している間も、ネットワーク上で送信している間も、アプリケーションで処理している間も保護されている必要があります。
- **監査と追跡 :** 従来、プライベート・ネットワーク・ベンダーは、パートナ間の全ビジネス・トランザクションの監査と追跡などの付加価値サービスを提供してきました。インターネットはプライベート・ネットワークの安価で広く普及している代替案となります。ただし、この安価なトランスポートを活用する統合ソリューションには、監査と追跡というサービスの提供が必要です。
- **高可用性 :** ビジネスをオンラインでグローバルに展開している企業は、停止時間なしで、常時オープンしている必要があります。したがって、これらの企業とそのアプリケーションにリンクしている統合ソフトウェアは、ミッション・クリティカルで、信頼性と拡張性が高く、常時使用可能であることが必要です。統合ソフトウェアは、少なくとも、リンクしているアプリケーションと同じくらいミッション・クリティカルです。
- **複合ビジネス・プロセス :** E-Business 統合は、継続時間、関係する組織の数、アプリケーションの数の面でも複合ビジネス・プロセスに対処する必要があります。これらの

アプリケーションが持つ高度な独立、異種および分散という性質によって、統合ソリューションのあらゆる側面に要件が追加されます。

- **インターネット規格 :** 本来、企業間トランザクションは、合意に基づいた規格に従って行う必要があります。1 企業または 1 統合ベンダーの固有なプロトコルが、それぞれに独自の統合ソリューションを持つ広範囲のパートナの集まり全体を基準化するとは限りません。汎用ソフトウェアは、次のような標準インターネット・プロトコルをすべてサポートしている必要があります。
  - 回線またはトランスポート・プロトコルとしての HTTP
  - Open Applications Group が提唱しているような、共通ビジネス・オブジェクトに対する標準定義付きメッセージ・フォーマット用の XML
  - Open Buying on the Internet（OBI）や RosettaNet などのビジネス・プロセス・プロトコル

3

独自の Oracle Integration Server の概要

この章では、Oracle Integration Server（OIS）の概要を説明します。OIS は、異なるコンポーネントの統合という E-Business の要件に対応するソフトウェアの完全なセットです。多彩な機能、堅牢さおよび要求の高い複雑な統合シナリオに対処するためのツールを備えています。

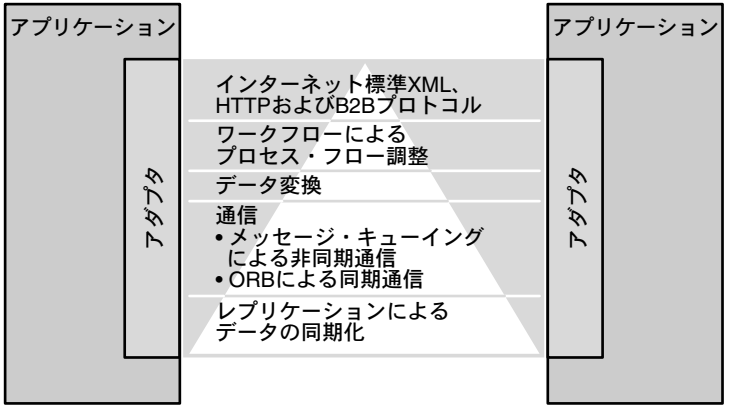
この章には次の項目が含まれています。

Oracle Integration Server の概要

Oracle Integration Server は、企業内の単純なフロントオフィスとバックオフィスの統合、企業内の IT 統合インフラストラクチャ計画の作成、最終的には企業間統合までを含む広範囲の統合問題に対応しています。OIS が提供する機能は、次のとおりです。

- データ統合
- レプリケーション
- アプリケーション統合
- ビジネス・プロセス・インテリジェンス
- データ変換
- アプリケーション・アダプタ
- ビジネス・プロセスのモデル化と実行
- システム管理

図 3-1 Oracle Integration Server の機能



データ統合

Oracle Data Access Gateway を使用すると、アプリケーションは異種データ・ソースからデータにアクセスし、管理できます。

- Oracle Procedural Gateway によって、CICS、IMS/TM、IDMS-DC などの Oracle 以外のトランザクション・システムへのプロシージャ型アクセスが行えます。
- Oracle Transparent Gateway によって、Sybase、Informix、SQL Server および DB2 を含む 30 を超える Oracle 以外のデータベースへの SQL アクセスが行えます。

レプリケーション

Oracle Integration Server には、複数の分散データベース全体のデータを同期化する高性能で柔軟性のあるレプリケーション機能を提供するアドバンスト・レプリケーションが含まれています。主な機能は、次のとおりです。

- 各アプリケーションの表と関連オブジェクトの単一グループとしてのレプリケーション
- 全表とサブセット表のレプリケーション
- ユーザー選択の競合解消ルールによる自動競合検出と競合解消
- レプリケーション・グループ・レベルでのユーザー定義可能なレプリケーション間隔による同期および非同期レプリケーション
- パフォーマンス改善のための自動パラレル・データ伝播

アドバンスト・レプリケーションは、Oracle Replication Manager によって集中的に管理、構成およびメンテナンスされます。

アプリケーション統合

同期通信

Oracle Integration Server は、組み込まれている Java ベースの CORBA 2.0 準拠のオブジェクト・リクエスト・ブローカ（ORB）と Enterprise JavaBeans（EJB）サーバーを使用して、同期通信ベースの統合をサポートします。

Oracle Integration Server の追加 CORBA 機能は、次のとおりです。

- **トランザクション**: トランザクション CORBA アプリケーションの開発用に Java Transaction Service（JTS）および CORBA Object Transaction Service（OTS）を提供します。
- **ディレクトリ・ネーミング:** 標準的な Java Naming Directory Interface（JNDI）および COSNaming インタフェースを業界標準のディレクトリ・サービスである Lightweight Directory Access Protocol（LDAP）に提供します。

- **オブジェクト・アダプタ :** 2 つの重要な機能を提供する永続 CORBA オブジェクトに対してオブジェクト・アダプタを実装します。第 1 に、CORBA オブジェクトのディレクトリの役割をします。第 2 に、CORBA クライアントによる初期起動時に、CORBA オブジェクトの位置の識別とロードを支援します。
- **Java と CORBA のサービス :** Java プログラマによる CORBA サービスの開発を容易にする多数の機能を提供します。Caffeine は、IDL を不要にする Java と IIOP 間のダイレクト・マッピング・ツールです。`java2iiop`、`idl2java` および `java2idl` などの他のツールは、アプリケーション開発を簡素化します。
- **セキュリティ** : 機能には、IIOP 上の Secure Sockets Layer（SSL）を使用した暗号化、ユーザー名とパスワードを使用した認証、ロールと権限を使用したアクセス制御などが含まれます。

Oracle JDeveloper は、EJB コンポーネントの開発で使用する開発ツールです。Caffeine は、Java から IDL へのコンパイラです。Oracle Integration Server は、サード・パーティのブリッジを介して Microsoft Com+ コンポーネント・モデルと相互運用できます。

非同期通信

Oracle Integration Server は、そのアドバンスト・キューイング機能を使用してアプリケーションの非同期統合をサポートします。アドバンスト・キューイングは、フルサービスのメッセージ・キューイング・システムです。主な機能は、次のとおりです。

- **保証された、正確な 1 回限りの配信 :** アドバンスト・キューイングは、ネットワーク、システムおよびアプリケーションの障害にかかわらず、各メッセージがその宛先に正確に 1 回、指定した時間間隔内で配信されることを保証します。
- **サブジェクト・ベースとコンテンツ・ベースのパブリッシュ・サブスクライブ :** パブリッシュ・サブスクライブはアプリケーションの疎結合を有効化する通信モデルです。よく知られているパブリッシュ・サブスクライブ・モデルには、サブジェクト・ベースとコンテンツ・ベースの 2 つがあります。アドバンスト・キューイングは両方ともサポートします。
- **伝播 :** アプリケーションは、ローカル・キューにメッセージをエンキューし、そのメッセージに対してリモート・キューの宛先を指定します。アドバンスト・キューイング・プロパゲータはメッセージを、ローカル・キューとリモート・キューの間を透過的に移動させ、分散アプリケーション間の通信を可能にします。
- **Java Messaging Service:** アドバンスト・キューイングは、業界標準の Java Messaging Service（JMS）を実装した最初のメッセージ・キューイング・システムです。アドバンスト・キューイング機能には、JMS に加えて、PL/SQL、C、C++ および Visual Basic を使用してアクセスできます。
- **メッセージの管理 :** これは、メッセージの保存という基本的な機能で提供されるアドバンスト・キューイング固有の機能で、次の項のビジネス・プロセス・インテリジェンスの中で説明します。

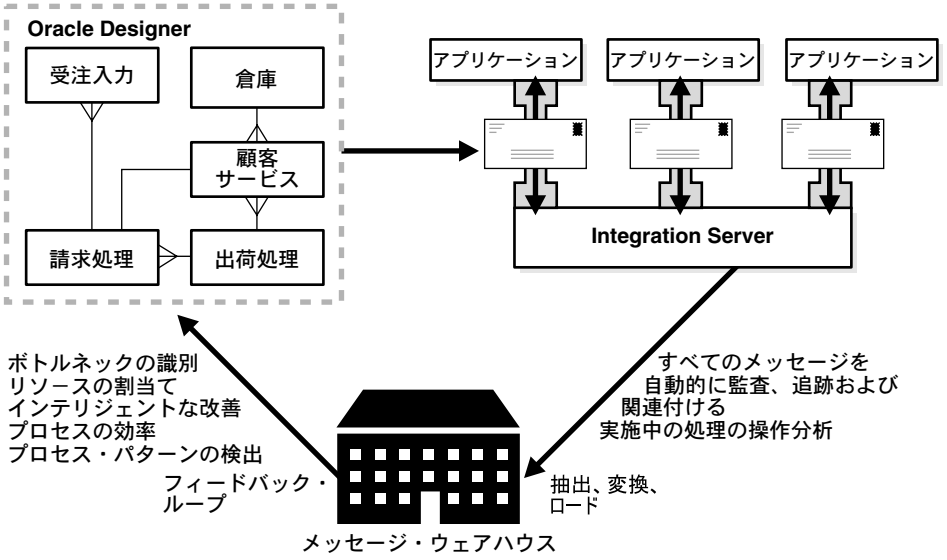
ビジネス・プロセス・インテリジェンス

Oracle Integration Server は、関連ツールと連携して、ビジネス・プロセスに関するビジネス・インテリジェンスを収集し分析する機能を企業に提供します。

Oracle Integration Server のアドバンスト・キューイング機能は、そのキューを流れるすべてのメッセージを自動的に保存し、追跡します。また、エンキュー回数、デキュー回数、配送された宛先、メッセージを処理したトランザクションの識別およびメッセージと例外条件との間の関係など、メッセージに関するすべての関連詳細情報を追跡します。さらに、この情報を標準的な問合せとレポート作成のツールでアクセスできる、問い合せ可能キューに保存します。この情報を活用して、次の 2 つの方法でビジネス・プロセスを改善できます。

- **実際のトランザクションまたはインプロセス・トランザクションの分析 :** キューに保存されている情報からすべてのトランザクションの完全な履歴と現状が判断できます。たとえば、着信したトランザクションの特定の状態を知り、次の処理ステップを判断します。例外的な状況や処理が停滞した場合、管理者は問題の原因を正確に把握し、調整するヒントを得てその処理を再開できます。この情報をアドバンスト・キューイングが提供する統計と組み合せて、ビジネス・プロセスのパフォーマンスをリアルタイムに改善できます。
- **メッセージのウェアハウスと分析 :** 追跡情報を保存し、一定の期間にわたって蓄積すると、メッセージ・ウェアハウスを作成でき、そこから標準 SQL を使用してメッセージを問い合せることができます。情報を抽出 / 変換し、分析とマイニング用に Oracle データ・ウェアハウスにロードできます。すべてのビジネス・プロセスを完全に再構築するために必要な情報はすべて使用可能なため、常に変化する多様な条件の下でシステムの実行方法を正確に判断できます。

図 3-2 プロセスの効率を改善するビジネス・プロセス・インテリジェンス



ウェアハウスは、次のような質問に応答することができます。

- 顧客が Web サイトを介して注文してから製品が顧客に配送されるまでどのくらいの時間が経過したか？
- この経過時間は最近 12 か月の間にどのように変化したか？
- この経過時間は 1 日の間にどのように変化したか？
- 最近 6 か月の間に、平均して、プロセス内のどのステップの実行に一番時間がかかったか？
- 追加リソースを配布する最適な場所はどこか？
- 応答時間で評価すると、最も優秀なサプライヤはどこか？
- このサプライヤをコンポーネント単位を基準にして評価する方法は？

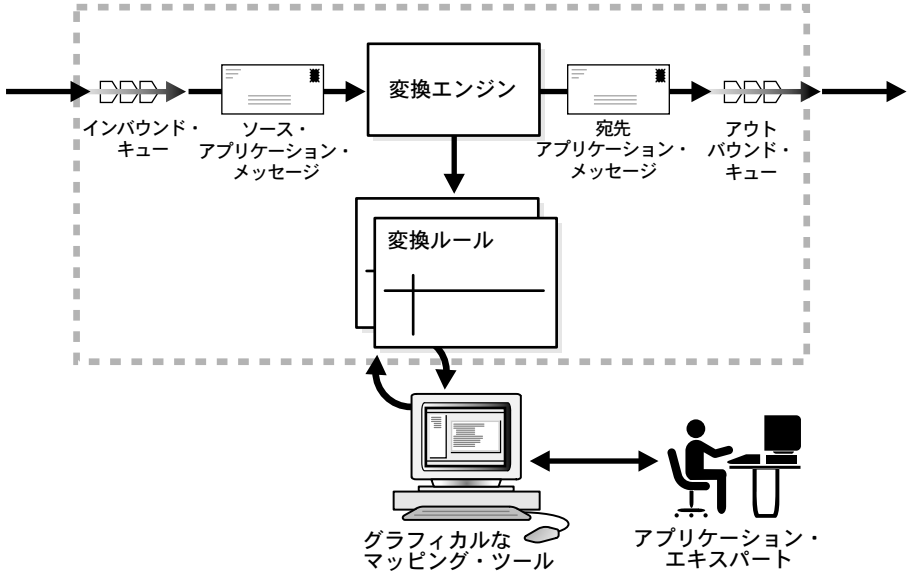
Oracle は、きわめて大規模なメッセージ・ウェアハウスを作成し、それに対して複雑な分析のための問合せを短い応答時間で実行できます。情報分析を支援する Oracle Tools には、Oracle Report、Oracle Express および Oracle Discoverer があります。ビジネス・プロセス分析によって、企業のマネージャは、必要な情報を取得してそれぞれのビジネスについてイン

テリジェントな意思決定をし、プロセスを簡素化してより効果的にリソースを配布し、全体的な効率を向上させることができます。

データ変換

Oracle Integration Server は、複数の異種アプリケーションを接続し、アプリケーション間のデータ、メッセージ、ビジネス・オブジェクトおよびビジネス・イベントの流れを円滑にします。各アプリケーションには、ビジネス・オブジェクト、スキーマおよびメッセージのフォーマットに関する独自の定義があります。Oracle Integration Server は、アプリケーションが相互に通信できるように、1 つのアプリケーションの出力フォーマットを別のアプリケーションの入力フォーマットに変換する変換サービスを提供しています。

図 3-3 ソースから宛先までのメッセージのカプセル化



OIS は、データ型の定義と異なるデータ型間での変換用に設計時に使用するビジュアル・ツールを提供しています。このビジュアル・ツールを使用して定義できる変換は、文字列操作、数学的操作、日付フォーマットの変換および形式変更です。コールアウト・メカニズムを使用して独自の変換ルーチンを作成できます。このツールを使用すると、Oracle と SAP のアプリケーションのリポジトリから型定義をインポートできます。さらに、XML メッセージ変換用の XML 文書型定義（DTD）もインポートできます。将来リリースされる Oracle Integration Server では、XML メッセージの XSLT ベースの変換がサポートされます。

データ型と型間の変換は設計時のアクティビティです。結果はサーバーのリポジトリに格納されます。Oracle Integration Server のランタイム変換エンジンは、この情報を使用してデータがアプリケーション間を流れるように変換します。Oracle Integration Server が定義するオープン API を使用すると、他の互換性がある変換を使用できます。

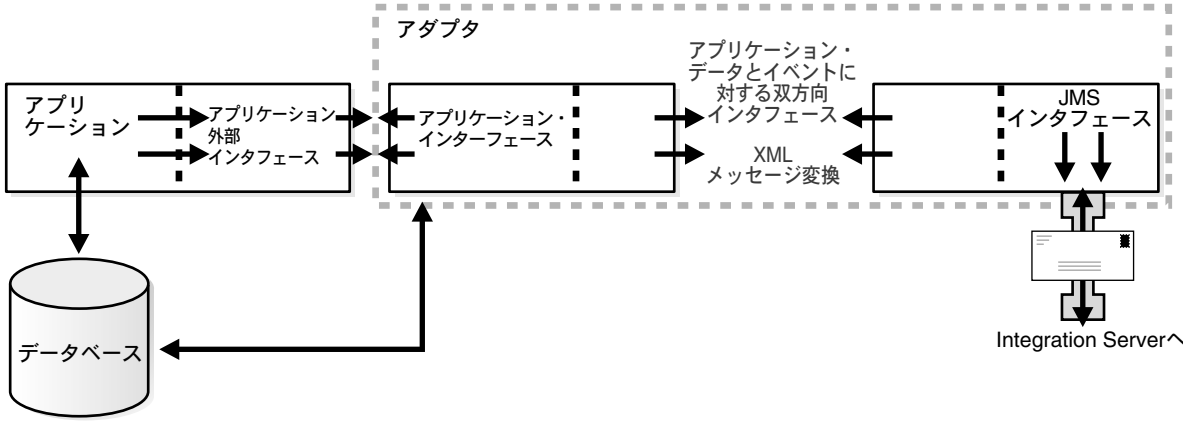
アプリケーション・アダプタ

Oracle Integration Server は、既存アプリケーションの変更が不要な統合ソリューションです。これは、既存のアプリケーションのリエンジニアリングが不要、つまり、最小限度のリエンジニアリングのみが必要であることを意味します。大部分のアプリケーションは統合できるように設計されていないため、単純な方法では Oracle Integration Server と通信できません。したがって、アプリケーションと Oracle Integration Server 間をブリッジするアダプタが必要です。

機能：アダプタはアプリケーションと対話して次の操作を実行します。

- アプリケーションのネイティブ・インタフェースを使用して、適切に定義されたビジネス・オブジェクトをアプリケーションに送信またはアプリケーションから抽出します。
- アプリケーションで発生するビジネス・イベント（新しい発注書の作成など）を検出し、この情報を他のアプリケーションに公開します。

図 3-4 アダプタのアーキテクチャの概要



- 必要に応じて、ビジネス・オブジェクトとビジネス・イベントを XML フォーマットのメッセージに変換します。
- 必要に応じて、妥当性チェックとエラー・チェックを実行します。
- JMS などの標準インタフェースを使用して、情報を Oracle Integration Server に転送します。

配布： アダプタのアーキテクチャは、インタフェース先のアプリケーションによって異なります。アダプタは通常、アプリケーションに近いコンピュータ上で実行します。場合によっては、Oracle アプリケーション、データベース・サーバーなどのアプリケーション自体の実行環境で実行されます。アダプタはデータベースの実行環境に配布することもできます。特に、Java または PL/SQL に実装される場合がその例です。オラクル社の目標は、あらゆる場合に、アダプタのホスティングと実行に関して集中的に管理する環境を提供することです。

アダプタ SDK: Oracle は、多数のベンダーと提携して、パッケージ・アプリケーションと業界固有のアプリケーションにアダプタを提供しています。さらに、アダプタ SDK を提供し、カスタム・アプリケーションとレガシー・アプリケーションのためのアダプタの開発を簡素化しています。アダプタ SDK は、トリガーの配布、XML メッセージの解析および JMS の使用に必要なライブラリを提供します。

ビジネス・プロセスのモデル化と実行

Oracle Integration Server には、グラフィカルなビジュアル・プロセス・モデリング・ツールが含まれています。このツールは次のようなシナリオで使用できます。

- ビジネス・アナリストが業界標準の Unified Modeling Language（UML）の作業図を使用し、ビジネス・プロセスを図で説明します。この段階で、プロセスに含まれるアプリケーションが識別され、プロセスの流れの詳細が述べられます。このモデルはサーバーのリポジトリに永続的に生成されます。
- 技術アナリストがその高水準モデルを、基礎となる統合インフラストラクチャにマップします。アプリケーション・インタフェース、ビジネス・イベント、変換およびメッセージの詳細情報が書き込まれます。完成したモデルは再度リポジトリに格納されます。
- このモデルはランタイム実行エンジンによって検証され、実行の準備を整えます。
- このツールを使用してランタイム・プロセスを変更できるのは、ビジネス・アナリストか技術アナリストのいずれかです。Oracle Business Process Coordinator は複数のモデルを同時にサポートします。
- ツールが表示するビジネス・プロセスの全体ビューによって、ユーザーは全機能を視覚化して理解することができます。

図 3-5 モデル化の手順

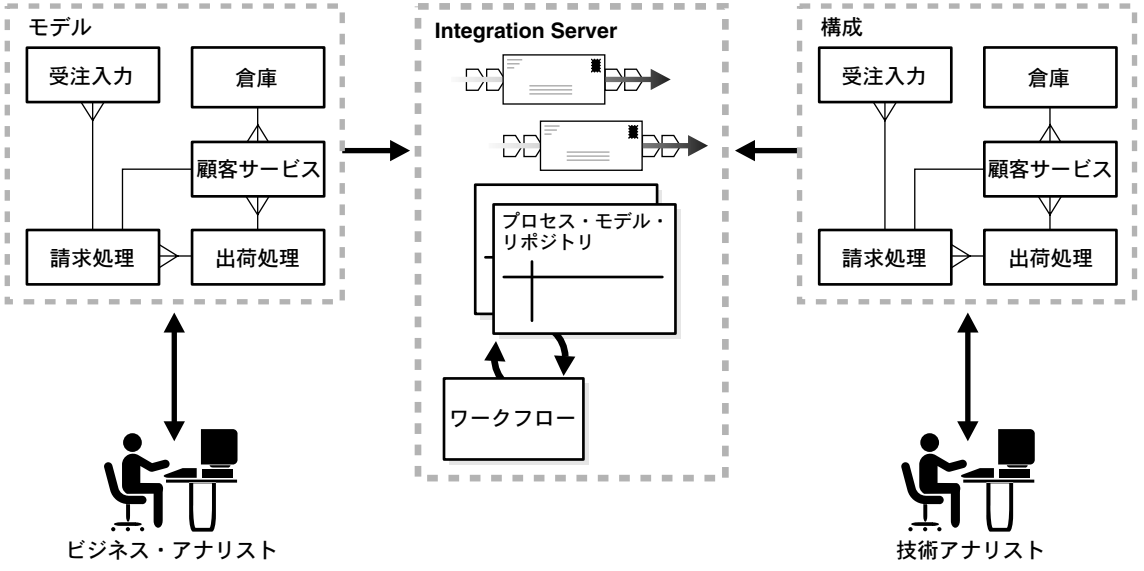
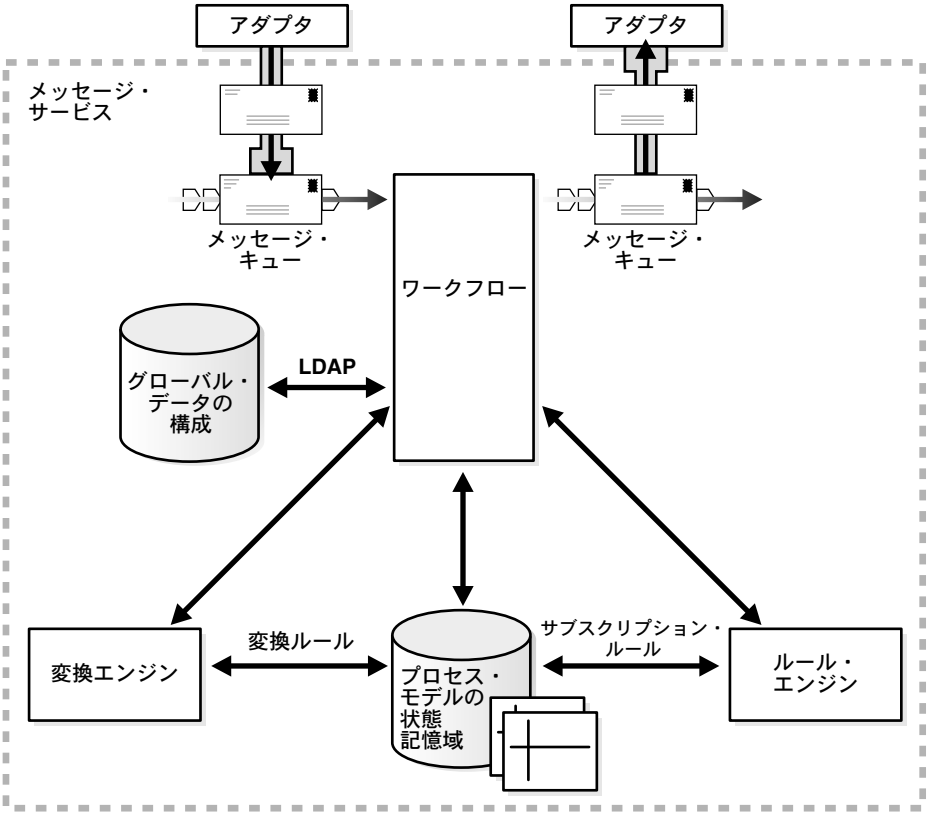

モデル
受注入力
倉庫
顧客サービス
請求処理
出荷処理
Integration Server
プロセス・モデル・
リポジトリ
ワークフロー
構成
受注入力
倉庫
顧客サービス
請求処理
出荷処理
ビジネス・アナリスト
技術アナリスト

図 3-6 ワークフローおよびワークフローと他のコンポーネントとの対話



実行エンジン： 検証モデルは、Oracle Business Process Coordinator と呼ばれるランタイム・エンジンによって実行されます。Oracle Business Process Coordinator は、実行中の全プロセスの推移を示すステータス情報を保持しています。このエンジンは、他のすべての機能コンポーネントを使用してタスクを完了します。Oracle Business Process Coordinator は、人間の介入が不要なプロセス（システム間プロセス）のみでなく、介入が必要なプロセス（従来のワークフロー）も自動化できるという点が特徴です。

システム管理

Oracle Integration Server は、Oracle Enterprise Manager（OEM）によって管理および監視されています。OEM は、Oracle アプリケーションと Oracle データベース、Application Server のインスタンスも監視できるシステム管理ツールです。このツールを使用すると、管理者は 1 つのセントラル・コンソールから分散環境を管理できます。機能が拡張され、統合環境を移動するすべてのピースを管理できるようになりました。Oracle Enterprise Manager は、次の 3 つの詳細レベルでシステム・オブジェクトを管理します。

- **キュー、メッセージ、キュー伝播 :** 開始、停止、スケジュール伝播、ビュー統計
- **個別のビジネス・プロセス :** 開始、停止、再開、問合せ
- **システム・プロセス :** Oracle Integration Server、Oracle Message Broker、アダプタ、アプリケーションおよびデータベース

Oracle Integration Server の設計目的

前の項では、Oracle Integration Server の高水準な機能の概要を説明しました。この項では、Oracle Integration Server をより有効に活用するために、その設計目的を説明します。内容は次のとおりです。

短期の一時的なソリューションではない、計画的なインフラストラクチャ

市販されている大部分の統合製品は、たとえば、特定のフロントオフィス・アプリケーションを特定のバックオフィス ERP アプリケーションに接続するなど、特定の統合問題の解決を目的にしています . これらの製品の価値は、身近なビジネス上の問題を解決するために 2 つのアプリケーションを早急に統合することにあります。製品の中には、2 つのアプリケーション間の特定のトランザクションを統合するのみというさらに限定された製品もあります。

Oracle Integration Server は、特定の問題を解決できますが、その設計目的は、企業が統合を一時的なソリューションとしてではなく、IT インフラストラクチャの一計画コンポーネントとして捉えるように促すことにあります。

オラクル社は、企業は特定のアプリケーションから独立して設計された統合アーキテクチャを最初から使用することが望ましいと考えています。企業は、現在と将来のニーズの両方を満たす製品を選択する必要があります。Oracle Integration Server は、企業全体の統合の基幹になるように設計されています。Oracle Integration Server は、エンタープライズ・ソフトウェアの特性として要求される堅牢さ、拡張性、オープン・アーキテクチャ、業界標準への準拠および管理と開発のためのツールなど、すべての特性を備えています。

必要に応じた選択と使用

Oracle Integration Server は、多様な統合問題の解決に役立つ多くのテクノロジを提供する包括的ソリューションです。さらに、このテクノロジのそれぞれに多彩な機能が備わっています。

オラクル社は、特定のテクノロジの使用について制限を設けていません。つまり、各自の問題を確実に解決するテクノロジを選択することができます。すべてのテクノロジは透過的に統合されています。ただし、Oracle のテクノロジは相互に独立して使用できるため、特定のテクノロジを選択したときにも無用な複雑さを避けることができます。たとえば、ORB ベースの同期通信のインフラストラクチャやアドバンスト・レプリケーションに関する知識がなくても、アドバンスト・キューイングを非同期通信に使用できます。

これが可能なのは、Oracle Integration Server が最初から幅広いテクノロジを提供するように設計されているためです。市販されている大部分の統合ソリューションは、メッセージ指向のミドルウェア、パブリッシュ・サブスクライブ・エンジン、データ変換、ワークフローなどの狭い範囲のテクノロジに基づいています。これらの製品は、それぞれ中核となる機能以外では微力となる傾向があります。これらの製品は、パートナ製品と疎結合によってその弱点を補っています。結果として、同じテクノロジを使用してあらゆるタイプの問題に対処することを余儀なくされる製品が生まれます。

ミッション・クリティカルな企業規模の統合

Oracle Integration Server は、ミッション・クリティカルな企業規模のビジネスの実装を目的に設計されています。この目的を達成するために、次の 2 つの主要な機能を提供しています。

- 高い信頼性、可用性および拡張性（RAS)
- 企業規模の管理

RAS 機能

Oracle Integration Server は、Oracle8*i* プラットフォームの実績のあるプロセス・アーキテクチャとランタイム環境を基にし、この環境からすべての RAS 特性を透過的に継承しています。Oracle8*i* は、何千ものコンカレント・ユーザーときわめて大量のトランザクション量をサポートする大規模な Online Transaction Processing（OLTP）アプリケーションで実証済みの製品です。そして、対称型マルチ・プロセッサ（SMP）アーキテクチャとクラスタ・アーキテクチャの両方で直線的に拡張していきます。Oracle8*i* には、多数の信頼性の高いソリューションが含まれています。その中には、ホット・スタンバイ・サーバー、透過的なアプリケーション・フェイルオーバー、障害の迅速なリカバリなどがあります。

管理

オラクル社は、分散ハブ・アンド・スポーク適用モデルによる統合管理の簡素化をお薦めします。また、データベース・サーバーの場合と同じように、すべての統合論理を多数の小規模サーバーに分散しないで、少数の大規模サーバーに整理統合することが望ましいと考えて

います。Oracle Integration Server は、前の項で説明したような高度な RAS 機能を持つアーキテクチャをサポートしています。

多く製品が、分散バス・アーキテクチャを推奨することによって、不十分な RAS 機能を補おうとしています。このアーキテクチャを使用すると管理が極端に複雑になることは、すでに何回か証明されています。したがって、ミッション・クリティカルな環境には適しません。ただし、Oracle Integration Server のアーキテクチャでは、分散**バス**・トポロジでの適用も可能です。

投資の有効活用

多くの企業は、技術や専門知識の開発のため、Oracle のデータベース・テクノロジに投資しています。Oracle Integration Server は、このような企業の技術への投資を守り、有効な活用方法を供与します。Oracle Integration Server では、Oracle Enterprise Manager、Oracle Designer、Oracle JDeveloper、Oracle Report および Oracle Discoverer などの使い慣れたツールを使用します。Oracle Integration Server の機能へは、Java、C、C++ および PL/SQL などの標準言語を使用してアクセスできます。さらに、既存のデータベース管理技術を使用して Oracle Integration Server を管理できます。

Oracle Integration Server の機能

Oracle Integration Server（OIS）は、併用してアプリケーションの通信を可能にする多数の製品で構成されています。OIS は、複雑な統合シナリオを処理するように設計されています。

Oracle Integration Server は、次の機能を提供します。

- アプリケーション間の非同期通信を可能にするメッセージ・キューイング
- 複数のアプリケーション間を移動するビジネス・オブジェクトを変換するツール
- 複合ビジネス・プロセスのモデル化、生成および自動実行
- Data Access Gateway を介してのデータ統合
- レプリケーションによるデータの同期化
- アプリケーション間の同期通信を可能にする標準的な要求応答プロトコル
- ビジネス・プロセスの調整、改善および実装を行うビジネス・インテリジェンス・ツール
- 単一のコンソールから環境全体を管理および監視することができるシステム管理

Oracle Integration Server は、次の目的で Oracle8*i* データベースを使用します。

- アドバンスト・キューイングによるデータとメッセージの格納
- 同期コミット・モデルによるトランザクションの整合性
- ビジネスの処理と変換ルールのためのリポジトリ

Oracle Integration Server には、次の製品が組み込まれています。

- Oracle Workflow －ビジネス・プロセスを制御します。
- Oracle Message Broker（OMB）－他のメッセージ・テクノロジとの相互運用に使用します。
- Oracle Workflow Builder －自動化されたプロセス・フローの定義をするグラフィック・ツールを備えています。

これらの製品は、Java Messaging Services（JMS）や拡張可能マークアップ言語（eXtensible Markup Language: XML）のようなメッセージ処理規格と E-Business 規格をサポートしています。

Oracle Integration Server の主な目的

この項の内容は次のとおりです。

セキュリティ

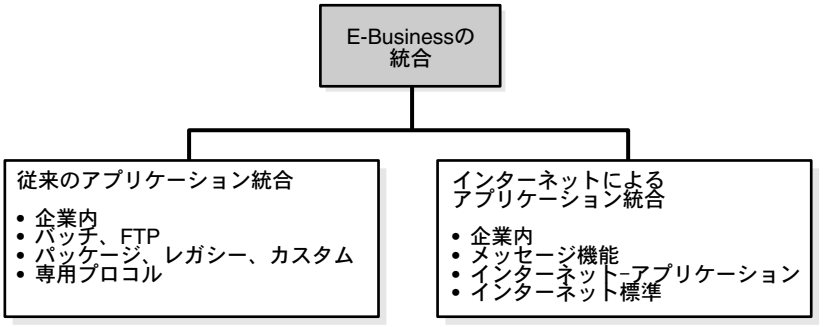
Oracle Integration Server（OIS）は、作成ポイントから受信ポイント（複数ポイント可）までの安全な通信推移を責任を持って完全に実行します。また、ポイント間の通信の整合性も保証します。

OIS は、通信の安全なコピーを永続的メディアに格納し、システム障害の発生による損害から守り、通信の監査証跡を確保します。OIS では、通信はトランザクションの一部としてのみ存在するため、移行中に紛失したり、間違って記録されることはありません。また、通信の正確な受信記録を保持しています。

製品開発のライフ・サイクル

OIS は、インターネット・アプリケーション、トランスポート層、プロトコルおよびレガシー・アプリケーションの言語などの統合をサポートします。また、開発環境と本番の環境の両方で、構成、開発および管理の実行に必要なバージョニング機能とインタフェースを提供します。

図 3-7 製品開発のライフ・サイクル



拡張性

OIS は、元のコンポーネントを大幅に変更しないで、徐々に開発および拡張できます。この機能は、拡張性と呼ばれ、次の操作が容易になります。

- 新しいアプリケーションのソリューションへの追加
- 既存アプリケーションのインタフェースの拡張
- 新しい統合コンポーネントの追加
- 統合コンポーネントのアップグレードの管理
- 市場に対する新しいチャネルの迅速な開発
- プロトコルとメッセージに関する新しい規格のサポート
- 新しい法的な規制措置に対するサポート

カプセル化

カプセル化は、オブジェクトとの通信を定義済みインタフェースに限定することによって、そのオブジェクトを自己完結型にします。OIS を構成しているすべての製品はカプセル化できます。

カプセル化を使用すると、通信対象のコンポーネントに影響を与えずに、コンポーネントの置換ができます。新しいコンポーネントは、そのコンポーネントが同じ定義済みのインタフェースを介して通信している場合、または同等かそれ以上の統合サービスを提供している場合は、正常にカプセル化できます。

カプセル化は、コンポーネント置換による影響の分析を簡素化し、コンポーネントのインタフェースの再設計を最小化し、他のコンポーネントのコードとインタフェースへの影響を最小化します。

コンポーネント・ベースのアーキテクチャ

コンポーネント・ベースのアーキテクチャは、論理的には拡張性とカプセル化の概念に従っています。その目的は、アーキテクチャを、コンポーネントの集合として設計し、それぞれのコンポーネントに論理的に分類されたサービスと機能のセットを与えることです。個々のコンポーネントを削除または置換しても機能にはほとんど影響を与えません。基本のアーキテクチャは有効で適切なまま残ります。

特定のコンポーネントに対する要件が大幅に変更された場合でも、他のコンポーネントへの影響は最小化されます。このためには、コンポーネントを相互に疎結合し、標準的なプロトコルと通信トランスポートで通信を行う必要があります。

ソフトウェアとハードウェアを疎結合する場合、ソフトウェアは簡単に移動ができ、異なるハードウェアのプラットフォーム間で通信できることが必要です。

新しいメッセージ・テクノロジ

オラクル社は、企業規模のアプリケーション統合に必要な非同期メッセージ交換インフラストラクチャの開発にとって重要な、新しい機能を組み込むメッセージ・テクノロジを開発しました。最も重要なテクノロジは、次のとおりです。

- 監査と追跡
- ビジネス・プロセス・コーディネーション
- ビジネス・インテリジェンス

監査と追跡

E-Business には、従来のビジネスで要求されている内容と同じ厳格な会計慣習を取込む必要があります。ビジネス・イベント・ドリブンの電子的な対話の監査証跡を獲得し、複製できることが必要です。Oracle アドバンスト・キューイングは、データベースにメッセージの持続記憶域を組み込み、処理後のメッセージも保存できる機能によって、強力な監査証跡を提供することができます。

持続記憶域を使用すると、進行中のビジネス・イベントの追跡が可能となり、ビジネス・トランザクションの現状を迅速に表示して評価できます。

ビジネス・プロセス・コーディネーション

ビジネス・プロセス・コーディネーションは、統合環境の管理を効果的に行うために必要不可欠です。ビジネス・プロセス・コーディネーションのソフトウェアは、ワークフロー・エンジンを管理する概念を論理的に拡張したものです。ワークフロー・エンジンは、ユーザーが実行する手動ステップも含めて、ビジネス・プロセス内のドキュメントの流れを管理します。

その原理は、ビジネス・プロセス・コーディネーションの原理と同じです。ただし、ビジネス・プロセス・コーディネーションは、人によるビジネス・プロセスではなく、自動化されたビジネス・プロセスを管理する点が異なります。ビジネス・プロセス・コーディネーショ

ンは自動化されたプロセス間のメッセージの流れを管理し、メッセージ・フローの状態も管理します。この機能を使用すると、トランザクションの管理に使用するテクノロジとは別に、時間を要するビジネス・トランザクションの状態を管理できます。

オラクル社は、ビジネス・プロセス・コーディネーションを提供するために、Oracle Workflow 製品を拡張してアドバンスト・キューイングと接続させました。

ビジネス・インテリジェンス

Oracle アドバンスト・キューイングのテクノロジは、データベースとメッセージ機能との間の密結合方式を作成します。この密結合方式を使用すると、Oracle Business Intelligence ソフトウェアを使用してビジネス・イベントを分析し、傾向とパターンを識別できます。さらに、Oracle Integration Server 内からコールしたプロセス・ステップが提供するサービス・レベルと応答時間を監視できます。

4

基本的な統合の概念

ここまで、Oracle Integration Server が多様な E-Business 統合アーキテクチャをサポートする方法を見てきました。そして、E-Business 統合の主要ドライバとその要件を満たす Oracle Integration Server の特定のコンポーネントについての知識を習得しました。次に、企業統合に必要な非同期メッセージ機能の使用について詳しく説明します。この章には次の項目が含まれています。

非同期メッセージ・ベースの統合

非同期メッセージ・ベースの統合は、企業統合に必要な主要テクノロジとして急速に発展しています。次の項目は、このテクノロジを最も効果的に使用する方法を理解するための手助けとなります。

- エンドツーエンドの企業統合のシナリオを検証し、統合を促進するメッセージ機能の使用方法を説明します。
- 統合に使用できる 2 つの異なる概念を持つメッセージ・ベースのアーキテクチャである Point-to-Point 統合とハブ・アンド・スポーク統合を比較し、この 2 つのアーキテクチャに関する利点とトレードオフを説明します。
- 企業統合に対してメッセージ機能を使用するときに関与する主要テクノロジのコンポーネントを要約し、メッセージの格納と管理、メッセージのルーティングと伝播およびメッセージの変換に加え、メッセージの相互運用性と標準の使用について説明します。

E-Business 統合に対してメッセージ機能を最も効果的に使用する方法を理解するために、サプライヤのサプライ・チェーン・アプリケーションと企業間取引市場またはオンライン取引所との接続に関連する技術的アーキテクチャを詳しく考えてみます。この項の内容は次のとおりです。

B2B 統合に対するメッセージ機能の使用例

たとえば、サプライヤが、次の項目を自動的に実行するためには、B2B 取引と対話する方法を自動化する必要があります。

- 最新の価格情報と在庫情報のオンライン取引所への転送
- オンライン取引所からの情報要求（RFI）とオークションに対する迅速な応答

ここで、サプライヤとオンライン取引所の両方の観点から統合作業を考えてみます。統合という観点から見た場合、この通信に対応するには、3 つの主要テクノロジ要件が必要です。テクノロジ要件は次のとおりです。

サプライヤとオンライン取引所との間の通信

サプライ・チェーン・システムからオンライン取引所にメッセージを送信するには、サプライヤはメッセージ・ペイロードを受信し、それをオンライン取引所に伝播するソリューションを使用する必要があります。メッセージ伝播機能は、次のような多数のサービスを備えている必要があります。

- 非同期通信 : 次の 3 つの理由で、サプライヤのシステムとオンライン取引所を非同期メッセージ機能を使用して接続する必要があります。
  - サプライヤとオンライン取引所は、要求応答モデルでは緊密に結合されていません。これは、両方とも応答を必要とせずに、ビジネス・アクティビティを続行できるためです。
  - サプライヤとオンライン取引所間の通信には、アプリケーション、システムおよびネットワークの停止に対する耐性が必要です。
  - サプライヤもオンライン取引所も他の取引先に対して、そのシステム間で同期的対話に必要な 2 フェーズ・コミット操作の実行を有効化することはできません。
- 保証付き配信 : サプライヤとオンライン取引所間で交換されるメッセージはビジネス上重要であるため、メッセージ機能はメッセージの正確に 1 回限りの順序付き配信を保証する必要があります。
- メッセージの格納と管理 : サプライヤとその取引先間で発生する可能性のある問題を解決するために、メッセージ機能は、オンライン取引所から発生した送受信メッセージを格納し、監査と追跡ができるようにする必要があります。

メッセージとデータの変換

サプライヤは、メッセージをサプライ・チェーン・システムからオンライン取引所に伝播するとき、メッセージまたはデータの変換という 2 つの問題に対処する必要があります。

- メッセージ伝播プロトコル : サプライヤとオンライン取引所および他の取引先との間の通信は、インターネット上で発生するため、そのメッセージは標準 HTTP-S プロトコル上で伝播される必要があります。セキュリティ上の理由から、HTTP より SSL をお薦めします。
- メッセージ伝播フォーマット : メッセージは HTTP-S 上で伝播されますが、メッセージ・ペイロードはオンライン取引所自体とその他の取引先の両方を理解できるフォーマットで送信する必要があります。大半の企業は、メッセージ・ペイロードを、すべての取引先およびオンライン取引所が合意する XML フォーマットで送信します。
- メッセージとデータの変換 : サプライヤのサプライ・チェーン・アプリケーションは、データを専用のフォーマットで格納する場合があるため、サプライ・チェーン・アプリケーションからデータを受け取り、それを適切な XML フォーマットに変換するメッセージ変換機能が必要です。

ビジネス・プロセス管理とワークフロー

メッセージをオンライン取引所に送信する前に、サプライヤは、担当役員から価格表の承認を得る必要がある場合があります。同様に、オンライン取引所からの購買依頼に応答するときに、サプライヤはそのメッセージを企業の財務アプリケーションに送信して、サプライ・チェーン・アプリケーションの在庫状態を更新する必要がある場合もあります。このマルチステップ・ビジネス・プロセスを管理するために、サプライヤは、ビジネス・プロセス管理

またはワークフロー機能をメンテナンスして、異なるアプリケーションとオンライン取引所間のメッセージを次のように調整する必要があります。

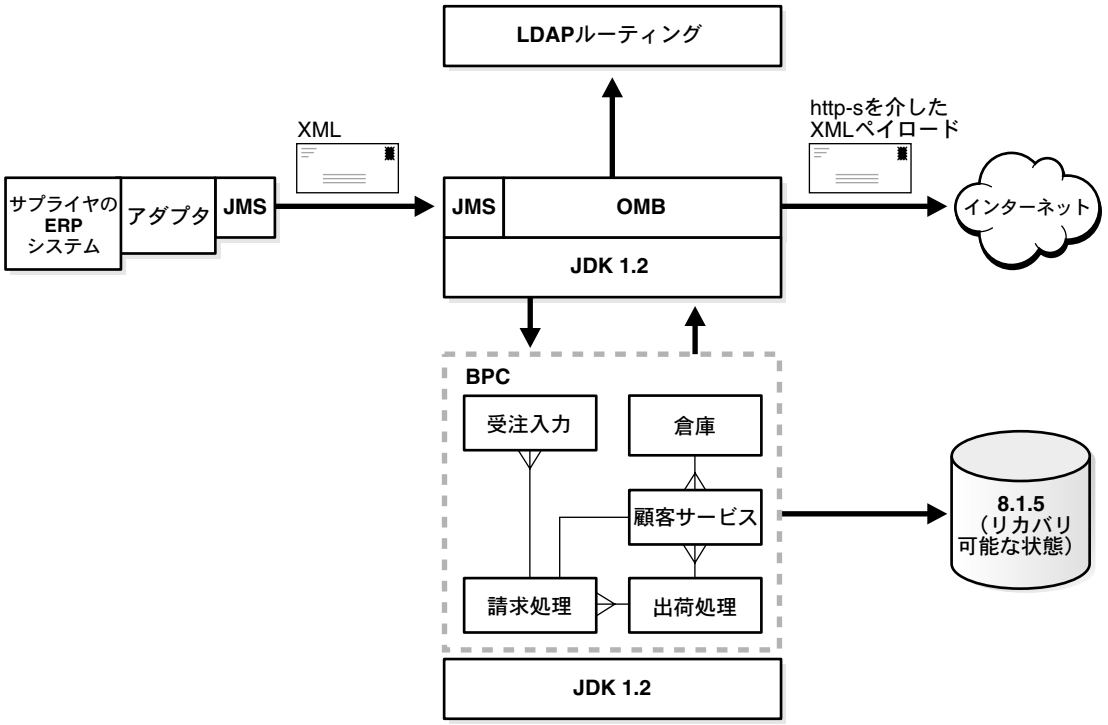
メッセージをオンライン取引所に送信する前に、ローカル・ワークフロー・プロセスを起動し、新しい価格表の各品目に関連する在庫が十分どうかを判断する必要があります。この場合、ビジネス・プロセス管理機能が必要なのは、サプライヤ側です。ビジネス・プロセス管理またはワークフロー機能は、新しい価格表をバックオフィス・アプリケーションから企業の在庫アプリケーションにルートし、価格表にある品目の在庫状態と比較する必要があります。サプライヤは、このマルチステップ・ビジネス・プロセスを管理するために、ビジネス・プロセス管理またはワークフロー機能をメンテナンスして、異なるアプリケーションとオンライン取引所間のメッセージを調整する必要があります。さらに、単一のビジネス・プロセスが、企業のエンタープライズ・アプリケーション間のビジネス・イベントとオンライン取引所または外部取引先の間の調整に関与することもあるため、単一のビジネス・プロセス管理またはワークフロー機能は、企業内および企業間の対話の両方で使用する必要があります。

ここまでは、この統合シナリオの主要な要件です。次に、サプライヤとオンライン取引所両方の観点から、統合の詳細を技術的に詳しく説明します。

オンライン取引所への統合のシナリオ : サプライヤの観点

多様なステップと統合コンポーネントが、サプライヤのサプライ・チェーン・アプリケーションと企業間取引をリンクします。このシナリオでは、Oracle Exchange がその役割を果します。この接続の確立に関わるステップとコンポーネントを図 1 に示します。

図 4-1 B2B 取引の統合 : サプライヤの観点



このステップとコンポーネントには次の内容が含まれています。

- アダプタ・テクノロジ : アプリケーションをオンライン取引所に接続するために、サプライ・チェーン・アプリケーションは最初に、アダプタを使用してメッセージをオンライン取引所の統合機能に転送します。このアダプタは、特定のメッセージ・キューに転送する新しい在庫ステータス情報を待機しています。また、アダプタは、アプリケーション固有のデータ・フォーマットでメッセージを受信すると、そのデータを XML に変換します。
- メッセージ伝播インフラストラクチャ : アプリケーションのアダプタは最初に、メッセージを標準メッセージ・インタフェースを介してメッセージ伝播インフラストラクチャにエンキューします。メッセージ交換インフラストラクチャは、次の 3 つの追加サービスを提供できます。
  - メッセージを静的にまたはサブジェクトやコンテンツに基づいてオンライン取引所にルートできます。

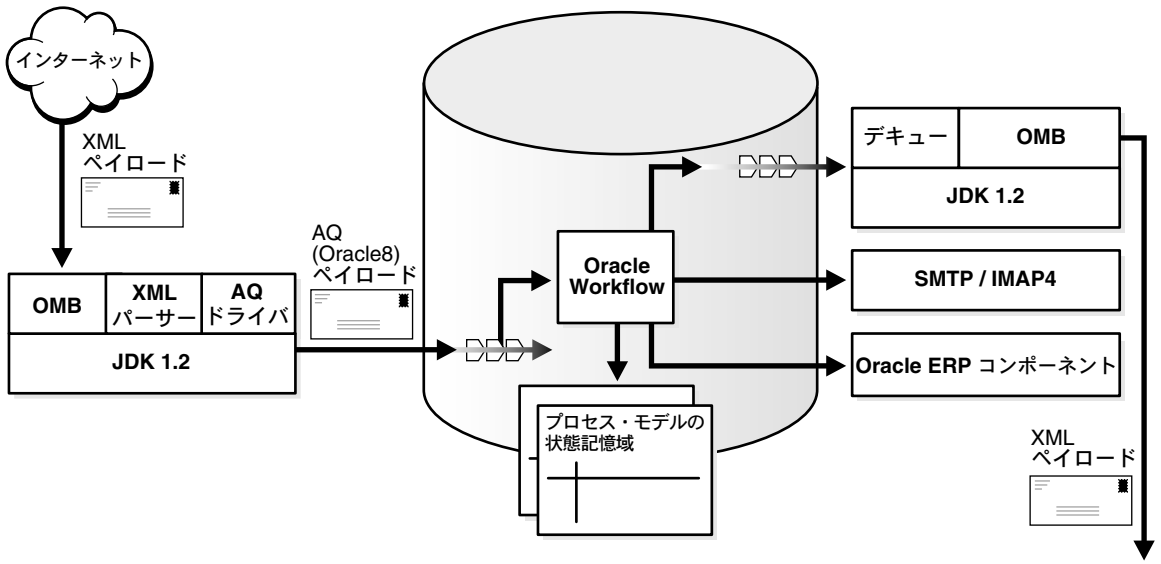
- LDAP ディレクトリ・サービスに照会し、メッセージの送信先を判断できます。
- 必要に応じて、メッセージの監査または追跡ができるように、メッセージを永続的に格納できます。

- ローカル・ワークフロー・プロセス：ローカル・プロセスが必要な場合、ローカル・ビジネス・プロセス管理機能は、メッセージをオンライン取引所に送信するために、メッセージをメッセージ伝播インフラストラクチャからデキューし、それをローカルで処理して、処理が完了したメッセージをメッセージ伝播インフラストラクチャに戻すことができます。

オンライン取引所への統合のシナリオ : オンライン取引所側の観点

Oracle Exchange の統合アーキテクチャは、前の項で説明したアーキテクチャと似ています。

図 4-2 B2B 取引の統合 : オンライン取引所側の観点



XML ペイロードは、HTTP-S プロトコルの回線を経由して受信され、次のステージを通過します。

- メッセージの受信と伝播 : メッセージ管理機能は、HTTP-S 上で XML ペイロードとしてメッセージを受信します。
- データ変換とサービス解析 : メッセージを受信した機能は、多くの場合、そのメッセージを、問題解決の目的で監査および追跡できるように、永続的に格納します。メッセー

ジ・ヘッダーとメッセージ・ペイロードは、次の 2 つの理由で、XML パーサーを使用して頻繁に解析する必要があります。

- メッセージのサブジェクトまたはトピック、あるいはペイロードのコンテンツのいずれかに基づいて情報のルート先を判断するため。たとえば、サプライヤがそのオンライン取引所のしきい値以下の在庫レベルを転送しようとする場合、そのメッセージは特定の承認サイクルを通して送信される必要があります。
- メッセージを構造化フォーマットに解析してビジネス・プロセス管理機能またはワークフロー機能に配置し、その処理を有効化するため。

- ビジネス・プロセス管理 : ワークフロー・システムは、メッセージをメッセージ伝播インフラストラクチャからデキューし、全ワークフロー・プロセスを実行して、そのオンライン取引所のコンポーネントを構成している多様なアプリケーションを更新します。さらに、ワークフロー機能は次のことを実行できます。
  - たとえば、小規模サプライヤに通知する電子メール・メッセージを、そのサプライヤ独自の SMTP/IMAP4 インタフェースを直接コールして送信します。
  - XML メッセージを外部に送信します。ワークフロー・システムは、XML メッセージを外部のサプライヤに HTTP-S 上で送信する必要がある場合、そのメッセージをターゲットに適した XML ペイロード・フォーマットにシリアル化して LDAP ディレクトリに照会し、メッセージのルート方法を判断してから、そのメッセージをメッセージ伝播インフラストラクチャにエンキューできます。

メッセージ・テクノロジとメッセージ・アーキテクチャ

非同期メッセージ交換インフラストラクチャを使用したアプリケーションとビジネス・プロセスの統合の概念についてはすでに説明しました。この項では、E-Business 統合に必要な非同期メッセージ機能の使用に関する特定の技術的側面を詳しく説明します。この項の内容は次のとおりです。

メッセージ・テクノロジ ― 概要

通常、密結合プロセスと呼ばれる相互依存性の高いプロセスは、相互に独立して操作するプロセスに比べて、メンテナンスが難しく、障害がおきやすく、また、リカバリが困難です。このため、できる限りプロセス相互を独立させたモジュール化アプリケーションを設計する開発者が増えています。

2 つのプロセスの相互依存性を制限するために、開発者は次の機能を使用します。

- メッセージ機能 : プロセス相互間の通信を形式化します。
- 分離機能 : プロセス相互間の対話の数と複雑さを最小化する技法です。
- ミドルウェア : 2 つのプロセス間の中間装置です。

ミドルウェアは、キューと呼ばれるメッセージの格納メディアまたはメッセージ・ブローカと呼ばれる汎用メッセージ転送プロセス、あるいはキューとメッセージ・ブローカの両方で構成されています。このミドルウェアを介して通信すると、各プロセスは他のプロセスの可用性、パフォーマンス、物理ロケーションまたは実装スタイルから独立して操作を行うことができます。

E-Business 統合に必要なメッセージ・システムの側面は、次のとおりです。

- 同期および非同期処理
- セッション・ベースの通信およびセッションレス通信
- ステートレス通信およびステートフル通信
- 双方向通信および一方向通信

同期通信および非同期通信

2 つのプロセスは、メッセージを介して同期的または非同期的のいずれかで通信できます。

同期処理では、送信側プロセスは受信側プロセスに制御を渡し、受信側プロセスから応答を受信するまで処理を一時停止して、受信側プロセスがメッセージを受信して処理したことを確認します。

非同期処理では、ミドルウェアのキューが送信側プロセスと受信側プロセスを切り離します。送信側プロセスは、ミドルウェアがメッセージを受信すると同時に処理を続行します。マルチスレッド・プログラムは、同期的に操作を実行します。スレッドは処理するメッセージを待機しますが、このプログラムは、スレッドが応答を待機している間に他のスレッドの処理を続行します。

セッション・ベースの通信およびセッションレス通信

メッセージは、カプセル化またはセッションのコンテキストのいずれかで作成または処理できます。セッションとは、ユーザーまたはプロセスが、承認された事前定義済みの接続を別のプロセスと確立するまでの期間を指します。メッセージは、永続的か非永続的かのいずれかです。永続的なメッセージは、現在実行中のセッションとは独立して送信できます。非永

続的なメッセージは、独立して送信することはできません。セッション・ベースの通信が必要です。

プロセス間のセッション・ベース通信は、セッションのコンテキスト内で実行されます。セッションを管理するルールは、メッセージの作成方法と処理方法に影響を与えます。セッションのルールは、ミドルウェアによるメッセージの管理方法、つまり、ルート、格納および保存の方法、応答を送信者に送信するかどうかなどにも影響を与えます。

セッションレス通信は、実行中のセッションのコンテキスト外で実行されます。永続的なメッセージの処理は、すでに確立済みのデータベース、ミドルウェアまたはアプリケーションとの接続を介して行われます。セッションレス・メッセージは、実行中のセッションが提供する追加情報の要件と関係なく管理および処理できます。

ステートレス（「状態なし」）通信およびステートフル（「状態付き」）通信

通信のトランザクションの状態は、「状態付き」または「状態なし」のいずれかです。トランザクションの状態とは、通信の状態を指します。メッセージ自体のライフ・サイクルのステージ、つまり、作成済み、処理準備完了、処理済みなどのステージとは関係ありません。

「状態付き」通信は、メッセージの作成時、管理時および単一トランザクションのコンテキスト内での処理時に発生します。トランザクション内のメッセージの位置は、メッセージの処理方法に影響を与えます。メッセージを作成するプロセス、ミドルウェアおよびメッセージ上で動作するプロセスはすべて独立して「状態付き」条件を保持します。その結果、メッセージはトランザクションのコンテキスト内で処理されます。

「状態なし」通信は、メッセージがその位置を参照せずに単一トランザクション内で処理されるとき、またはメッセージがトランザクションの境界で交差するときに発生します。

双方向通信および一方向通信

同期通信は双方向通信です。メッセージの送信者はメッセージが正常に処理されたという応答を受信するまで待機します。一般的に、「状態付き」通信またはセッション・ベースの通信は双方向通信です。セッションまたはトランザクションのコンテキストで送信されるメッセージは、ほとんどの場合、受信側プロセスが送信する応答に関連しています。

メッセージに含まれているデータを、メッセージのペイロードといいます。1 つのディスクリート・プロセスまたはアプリケーションから別のディスクリート・プロセスまたはアプリケーションに、セッションまたはトランザクションから独立して送信されたメッセージには、処理を進めるために必要な情報が含まれています。メッセージの処理に送信者との追加通信が不要なため、自律型のペイロードを持つメッセージと言えます。

自律型のペイロードを持つ非同期、「状態なし」、セッションレス・メッセージは、一方向通信です。このメッセージの管理と処理は、送信者のプロセスに影響を与えません。送信者はメッセージの送信と同時にプロセスを続行できます。一方向通信では、メッセージの送信者と受信者間の分離はさらに広がります。

メッセージ・ベースの統合アーキテクチャ

メッセージ機能の多彩な技術的側面がわかったので、次は、E-Business 統合にメッセージ機能を使用する場合の 2 つの主要な代替アーキテクチャを検証します。ソリューションの編成には、データ処理とメッセージの要件に応じて、次の 2 つのアーキテクチャのいずれかを使用できます。

2 つのメッセージ・ベースの代替統合アーキテクチャとそれに関連する利点とトレードオフは、次のとおりです。

- Point-to-Point 統合
- ハブ・アンド・スポーク統合

Point-to-Point 統合

Point-to-Point 統合アーキテクチャは、各アプリケーションをそれぞれの通信先アプリケーションと直接統合します。アプリケーションは相互に 1 対 1 ベースで直接通信します。このアーキテクチャは、従来から使用されていて、アプリケーション間のファイル・ベースの統合を提供します。

アーキテクチャ・インフラストラクチャは、比較的必要ありません。フェーズ実装では、早期フェーズの実装が迅速化されます。ただし、このアーキテクチャは、すでに限界が見えています。特にメンテナンス・コストが上昇（ハブ・アンド・スポーク・アーキテクチャに比べて指数関数的に増える可能性がある）し、柔軟性と管理性に欠けます。

ハブ・アンド・スポーク統合

Point-to-Point 統合とは異なり、ハブ・アンド・スポーク・アーキテクチャは、最も新しい統合シナリオとして重要な役割を果しています。ハブ・アンド・スポーク・アーキテクチャは、相互に通信するためにアプリケーションが通過する中心的な中間装置（ハブと呼ばれる）を提供します。このアーキテクチャでは、各アプリケーション（スポークに当たる）は中間装置（ハブ）と通信し、次に、ハブが他のアプリケーションとの通信を管理します。アプリケーションは直接相互間で通信しません。

ハブ・アンド・スポーク統合アーキテクチャのコンテキストで、ハブは次のような様々なサービスに対する中心点になっています。

- メッセージの伝播と通信 : ハブは、異なるアプリケーション間におけるメッセージのルーティングと伝播を管理する中心的な機能を提供します。メッセージをコンテンツまたはトピックに基づいて、複数のアプリケーションにルートする必要がある場合、ハブ・アンド・スポーク・アーキテクチャは、Point-to-Point アーキテクチャと比較してはるかに高度な柔軟性を提供します。
- メッセージの管理、追跡および監査 : ハブは、異なるアプリケーション間を流れるメッセージを格納、監査および追跡するため、アプリケーション間のすべての対話を管理する中心的な機能を備えています。Point-to-Point 統合アーキテクチャでは、監査に必要な情報は各アプリケーション内に格納されているため、情報の流れの分析が複雑になります。

- メッセージとデータの変換 : ハブは、メッセージとデータの変換を行い、データを送信側アプリケーションのフォーマットから受信側アプリケーションのフォーマットに変換します。ハブ・アンド・スポーク・アーキテクチャでは、各アプリケーションは、受信側アプリケーションのフォーマットに変換できる共通ビューにデータを変換するだけです。送信側アプリケーションがメッセージを新しいアプリケーションに送信する場合、ハブに必要な操作は、新しい変換を共通ビューから新しいアプリケーションのフォーマットに適用するだけです。この操作によって新しい Point-to-Point マップの要件は不要になります。
- ビジネス・プロセス管理 : ハブはビジネス・プロセス管理またはワークフロー・サービスを提供して、アプリケーション間のマルチステップ対話を管理します。たとえば、あるアプリケーションが別のアプリケーションにメッセージを送信し、そのアプリケーションから応答を受信した後で、その情報を 3 番目のアプリケーションに送信するとします。ハブ・アンド・スポーク・アーキテクチャでは、ハブにビジネス・プロセス管理を集中化して、異なるアプリケーション間のワークフロー管理を簡素化します。

統合アーキテクチャは、多くの異なるレベルで定義されているため、ハブ・アンド・スポーク構造がすべてのレベルに存在するとは限りません。たとえば、ハブとスポークがすべて同じハードウェアに配布されている場合もあれば、異なるシステムに配布されている可能性もあります。通常、各スポークはハブのみと通信するため、ハブはアプリケーション・インタフェースと統合要件に集中します。これによって、統合サービスに対するリソース要件は切り離されます。

ハブ・アンド・スポーク・アプローチは、アプリケーションに対する統合ポイントの数を減らすことによって、アプリケーションへのインタフェースを簡素化していると一部の人は思っています。実際は、API は Point-to-Point アーキテクチャと同じくらい複雑です。しかし、ハブ・アンド・スポーク・アプローチでは、アプリケーションの統合ポイントが 1 箇所に集中されているため、アプリケーションの設計、開発および管理を簡単に行うことができます。

利点とトレードオフ

ハブ・アンド・スポーク・アーキテクチャと Point-to-Point アーキテクチャの 2 つの主要な相違点を考えてみます。

- アプリケーションの複雑さ : Point-to-Point アーキテクチャは、比較的単純な統合シナリオに適しています。単純な統合では、統合するアプリケーション数が少なく、アプリケーションの置換、アップグレードまたは大幅な変更がほとんどなく、アプリケーションの多くが同じテクノロジを使用しています。ハブ・アンド・スポーク・アーキテクチャは、多数の異なるアプリケーションとシステムの統合に適しています。ただし、統合ハブの複雑さが発生します。このアーキテクチャは、Point-to-Point アーキテクチャと比較すると、新しいアプリケーションをハブのみに統合し、他のすべてのアプリケーションと統合する必要がないため、拡張性がはるかに高くなります。

- 統合要件の複雑さ : Point-to-Point 統合アーキテクチャは、次のような比較的単純な統合要件に適しています。
  - データはアプリケーション間のみで変換されます。
  - メッセージは単純なルーティング・スキーマを使用して少数の宛先のみにルートされます。
  - 集中化されたビジネス・プロセス・コーディネーション機能は不要です。
  - 全ビジネス・イベントの共通ビューは不要です。

  ハブ・アンド・スポーク・アーキテクチャでは、ハブがこれらのサービスをすべてのアプリケーションに提供し、統合方法を簡素化します。
- ビジネス上重要な性質の統合 : ハブ・アンド・スポーク・アーキテクチャは、次のようなビジネス上重要な統合シナリオにも適しています。
  - 機密性の高い情報がアプリケーションとビジネス・プロセス間で通信されます。
  - 統合の停止や通信の損失による犠牲は膨大です。
  - ビジネス・イベント情報の保存、監査および追跡という基本的ニーズが存在しています。
- 統合環境の管理性 : ハブ・アンド・スポーク・アーキテクチャを使用すると、調整ハブの複雑さが追加されますが、統合固有の機能を結集させた中心点が提供され、そこで集中管理、メンテナンス、バックアップとリカバリ、追跡、ルーティング、変換などが行われます。ハブ・アンド・スポーク・アーキテクチャは、各アプリケーションの統合とインタフェース要件を定義、設計およびメンテナンスするための軽量スポークを提供します。これによって、メンテナンス、影響分析、アップグレード・サイクルおよびレガシー・システムの廃棄が簡素化されます。

メッセージ・テクノロジ

ここまでで E-Business 統合に非同期メッセージ機能を使用する方法が理解できました。次に、この項で具体的に説明するメッセージ・テクノロジの主な要素のいくつかを検証します。

- メッセージの伝播と通信
- メッセージ通知モデル
- メッセージ変換とデータ変換の要件
- メッセージとデータの変換に関する問題

メッセージの格納と管理

E-Business 統合のプラットフォームを提供する非同期メッセージ交換インフラストラクチャの基本要件は、そのインフラストラクチャがメッセージの持続記憶域と管理を提供することです。前述のとおり、メッセージを格納および管理する機能は、次の 4 つの理由から非常に重要です。

- メッセージの着信時に処理ができない場合 : アプリケーションは、すべての要求を即時に処理できるリソースがない場合、同時に着信したクライアントからの多数の未処理メッセージを扱う必要があります。メッセージ・システムは、永続キューにメッセージを格納し、後で受信者が処理できる時点でそのメッセージを配信する必要があります。
- メッセージのスケジューリング : さらに、メッセージ・システムには、メッセージを優先順位に従って処理するために、メッセージの永続性が必要です。
  - 後から着信するメッセージが前に着信したメッセージより優先順位が高くなる場合があります。
  - 前に着信したメッセージは、処理の実行前に後から着信するメッセージを待機する必要がある場合もあります。

  このようなメッセージの優先順位を、一定期間変更することもできます。たとえば、特定キューのメッセージを、特定の時間枠の間、他のキューのメッセージより重視することができます。メッセージの永続性によって、優先順位が高いキューのメッセージを最初に処理できます。優先順位の低いメッセージは格納して後で処理できるため、優先順位の高いメッセージの邪魔になることはありません。
- メッセージの監査 : メッセージの制御コンポーネントがペイロード情報そのものと同じくらい重要になる場合があるため、メッセージの永続性は非常に重要です。たとえば、メッセージの受信タイムと送信タイムが、メッセージの最も重要な部分になる場合があります。E-Commerce 環境では、メッセージの永続性によって、多様な顧客からの注文情報が格納されるため、その情報に問い合せると、需要がピークに達する期間を理解したり、注文状態を判断することができます。したがって、メッセージは実行後も依然として重要な場合があります。永続的なメッセージの格納は非常に重要であるため、このような情報は格納し、問合せまたは監査の対象にすることができます。
- 通信障害 : メッセージ・クライアント間の通信リンクは常時使用できるとは限りません。他の用途に予約されている場合もあります。システムがメッセージを即時に処理できない場合、メッセージ・システムはメッセージを永続的に格納し、処理リソースが使用可能になった時点で配信する必要があります。エンタープライズ・アプリケーションの統合には、各メッセージがそれぞれの受信者に正確に 1 回配信されるというメッセージの保証付き配信が非常に重要です。

メッセージの永続性に対するニーズは、企業間取引のシナリオで増大しています。これは、企業間取引では、単一の企業内取引に比べ、通信障害の可能性と監査および追跡の必要性がはるかに大きいためです。統合シナリオの大部分のメッセージは永続的に格納される必要があります。ただし、場合によっては、揮発性メッセージで十分です。たとえば、アプリケーションに対してメッセージの送受信が完了したことを通知する電子メール・メッセージは監査のために格納する必要はありません。

大部分のメッセージ・システムには、自動化されたメッセージの永続性機能が備わっていません。メッセージは、RAW フォーマットでファイル・システムに格納されるだけです。このため、メッセージ・ヘッダーとメッセージ・ペイロードに対する問合せは実行できません。永続的に格納する必要があるメッセージの場合、メッセージ・システムでは、2 フェーズ・コミット操作をそのシステム自体とデータベース・システム間で実行する必要があります。

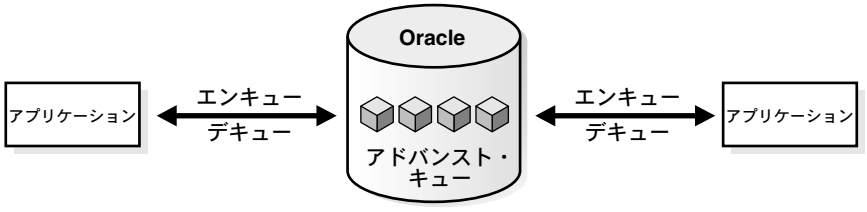
このような複雑な操作とは対照的に、オラクル社は、メッセージ交換インフラストラクチャを企業のデータベースに直接統合します。つまり、キューをデータベース表に関連付けて、このキューに配置されたメッセージが自動的にデータベースに永続することを保証します。次に、そのメッセージをアーカイブして、標準 SQL を使用してメッセージ・ヘッダーとメッセージ・ペイロードの両方に対して問合せを実行します。

メッセージの伝播と通信

メッセージは 1 つのアプリケーションまたはビジネス・プロセスから別のアプリケーションまたはビジネス・プロセスに伝播される必要があるため、メッセージ交換インフラストラクチャは、メッセージを送信者から受信者にルートするメカニズムを提供します。メッセージの伝播とルーティングは、次の 2 つの方法のいずれかで実行できます。

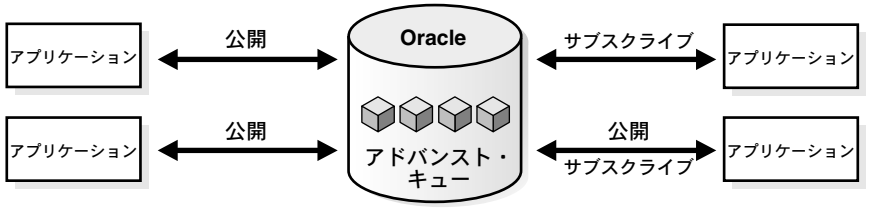
- Point-to-Point ルーティングでは、送信側アプリケーションは、メッセージを送信する宛先（受信側アプリケーション）を認識しています。Point-to-Point ルーティングでは、送信側アプリケーションまたはビジネス・プロセスと受信側アプリケーションまたはビジネス・プロセスはある程度の密結合となります。

図 4-3 Point-to-Point モデル



- パブリッシュ・サブスクライブ・ルーティングでは、送信側アプリケーションはメッセージを送信する宛先を認識していません。かわりに、メッセージの受信者を管理する責任をミドルウェアに割り当てます。ミドルウェアはメッセージ作成者が公開するメッセージのトピックのリストを定義します。

図 4-4 パブリッシュ・サブスクライブ・モデル



プロセスは、自分がサブスクライバであることをミドルウェアに示し、受信するメッセージのトピックを定義します。ミドルウェアは、サブスクライバがそのトピックに関するメッセージは不要であるか、またはサブスクライブを中止するかを通知してこない限り、そのトピックに関するメッセージを将来も送信します。この設計技法によって、プロセス間には分離レベルが追加されます。送信者はメッセージを、そのメッセージの受信者、配信ロケーションまたはメッセージの処理方法に関係なく公開します。ミドルウェアは次の操作を実行します。

- サブスクライバの登録と条件付きサブスクリプション・ルールの施行
- 個別メッセージの受信者の判断
- パブリッシャによるメッセージ作成とサブスクライバによるメッセージのコレクションの両方の追跡

サブスクライバは、特定のトピックに関するメッセージを受信することを登録し、メッセージの送信者またはパブリッシャと作成場所に関係なくサブスクライブします。

Point-to-Point とパブリッシュ・サブスクライブ・ベースのメッセージ・ルーティングは、大きく 2 つのメッセージ・ルーティング・カテゴリに分割されます。実際には、これらのスキームはそれぞれ、さらに複雑な場合があります。たとえば、Point-to-Point とパブリッシュ・サブスクライブ環境の両方で、メッセージは、サブジェクト・ベース、トピック・ベースおよびルール・ベースの 3 つの方法でルートできます。ここでは、その 1 つ 1 つを取り上げて、それぞれのパブリッシュ・サブスクライブ・ルーティングへの適用方法を、例を使用して説明します。

- サブジェクト・ベースまたはトピック・ベース : これは、最も単純な構成のパブリッシュ・サブスクライブです。プロセスは、適切に名前指定されたトピックまたはサブジェクトに関連するメッセージを公開します。たとえば、サブジェクトが PLACE AN ORDER であるとします。このトピックに関するメッセージの受信を希望することを次のように記述して、ミドルウェアに登録したサブスクライバは、そのトピックのサブスクライブを中止するまで、そのトピックに関するすべてのメッセージを受信します。たとえば「トピック PLACE AN ORDER に関するすべてのメッセージを送信する。」という指定が可能です。

- コンテンツ・ベース : これは、トピック・ベースのパブリッシュ・サブスクライブの拡張バージョンです。サブスクライバは、メッセージ内に含まれている値に応じて、特定トピックに関する受信メッセージのサブセットのみの送信を依頼します。たとえば、「トピック PLACE AN ORDER に関するメッセージの中で、メッセージ・データの国別コード・フィールドに UK が含まれているすべてのメッセージを送信してください」と記述します。
- ルール・ベース : ルール・ベースのパブリッシュ・サブスクライブは、コンテンツ・ベースのメカニズムを拡張したものです。サブスクライバは、メッセージ・データの値に関連するか、またはしないかという条件を管理するルールを指定します。たとえば、「トピック PLACE AN ORDER に関するメッセージの中で、メッセージの国別コード・フィールドに UK が含まれていて、カレント・タイムが午前 9 時から午後 5 時までのすべてのメッセージを送信してください」と記述します。

メッセージ通知モデル

企業は、通常 2 つのメッセージ・モデルを使用して内部通信を容易にしています。完全な通信ソリューションには通常、両モデルの使用が必要です。

イベント通知

イベント通知モデルは、企業全体にビジネス・イベントを通信するニーズに対応しています。ビジネス・イベントは、ビジネス・シナリオの論理的な状態変化を指します。顧客の発注のような単純なイベントもあれば、発注時にクレジット限度額を超えるという複雑なイベントもあります。一方のシステム・イベントは、プログラムの実行時における物理ポイントです。このポイントで、たとえば、ファイルへの書込みなどの識別可能な計算タスクが発生します。単一ビジネス・イベントは通常、多数のシステム・イベントに関連付けられています。

イベント通知モデルは、アプリケーション間の通信ポイントと通信のコンテンツを定義します。非同期メッセージ機能とパブリッシュ・サブスクライブ・ルーティングを使用してビジネス・イベントのインスタンスを表すメッセージを配信します。各ビジネス・イベントは、個別のパブリッシュ・サブスクライブのトピックとしてミドルウェアに定義されます。ビジネス・イベントを処理するアプリケーション（複数可）も、ビジネス・イベントのインスタンスが発生するたびにメッセージを公開します。そのビジネス・イベントに関する知識が必要なアプリケーションは、そのイベントに関するトピックをサブスクライブします。ミドルウェアは、サブスクライバへのメッセージ配信とサブスクライバによるメッセージ・コレクションを管理します。

通常、ビジネス・イベントを発表するメッセージは自律型で、メッセージの配信以外はアプリケーション間の通信を必要としません。一般にサブスクライバは、パブリッシャに応答する必要はありません。サブスクライバは、メッセージの受信情報を処理してから、別のビジネス・イベントのインスタンスを呼び出し、このビジネス・イベントに関するメッセージを別のトピックとして公開できます。

サービス要求

最も一般的に使用されているメッセージ・モデルの 2 つ目は、サービス要求モデルで、イベント通知に代替モデル化スタイルを提供します。これは、1 つのアプリケーションは別のアプリケーションのサービスを利用できるという前提に基づいています。たとえば、アプリケーション A が、メッセージの形式でサービス要求を作成し、アプリケーション B のサービスを要求します。

サービス要求モデルには、イベント通知モデルに比較するとはるかに多くのバリエーションがあります。選択するオプションは、個々のソリューションの要件によって異なります。たとえば、サービス配信の確認が不要な場合、サービス要求モデルは非同期メッセージ機能を使用して実装されます。このオプションでは、パブリッシュ・サブスクライブ・ルーティングを使用できます。確認が必要な場合、肯定否定に関係なく、サービス要求モデルは、非同期メッセージ機能を使用するか、または呼出応答形式のメッセージ・テクノロジを使用して同期的に実装されます。

サービス要求モデルでは、自ら提供できないサービスを必要とするアプリケーションは、そのサービスを提供するアプリケーションの名前を指定してサービス要求メッセージを作成します。ミドルウェアは、そのメッセージを名前指定したアプリケーションにルートします。そのサービスを提供する側のアプリケーションには、キューおよびコール可能な API のサービスを起動する手段が定義されています。サービス要求モデルが使用可能なバリエーションの中から 3 つを次に示します。

- 非パブリッシュ・サブスクライブ・サービス要求モデル : この構成では、サービスが必要なアプリケーションは、サービスの要求を名前指定したアプリケーションにミドルウェアを介して送信します。
- パブリッシュ・サブスクライブ・サービス要求モデル : この構成では、サービスを提供するアプリケーションは、サービス・プロバイダとしてミドルウェアに登録されています。サービスが必要なアプリケーションは、その要求をミドルウェアに公開します。ミドルウェアはこのメッセージを登録されているサービス・プロバイダにルートします。
- 配信確認付きパブリッシュ・サブスクライブ・サービス要求モデル : この構成では、サービスを提供するアプリケーションは、サービスを正常に配信した後、サービスの配信を確認するメッセージを返します。ミドルウェアはその確認を要求発信元のアプリケーションにルートします。

メッセージ変換とデータ変換の要件

非同期メッセージを使用してアプリケーションまたはビジネス・プロセスを統合する場合の基本要件は、発信元アプリケーションが送信したメッセージ・ペイロードを、受信側アプリケーションが受け入れることができる異なるフォーマットに変換することです。変換プロセスを理解するために、Oracle Customer Relationship Management（CRM）アプリケーションを考えてみます。このアプリケーションは、Web ストア・モジュールで受信した発注書を SAP 財務アプリケーションに送信します。特定の変換要件には次の 2 つあります。

データ型の変換

Oracle CRM アプリケーションは、すべてのデータを SQL フォーマットで格納します。SAP アプリケーションは、そのデータをフラット・ファイル形式の iDOCS で格納してアクセスします。SQL データ型は、バイト表現レベルで iDOCS データ型に変換する必要があります。これは、データ型レベル変換またはデータ・レベル変換と呼ばれています。データ変換そのものを考えてみても、多数の異なる種類の変換が適用できます。

- 値 : データ値を 123 から ABC に変更します。
- 名前 : データ名を `CUSTNUM` から `customer_id` に変更します。
- 型 : データ型の数値をアルファベットに変更します。
- ペイロード定義のマッピング : `BranchId`、`CustomerNo`、`AccountType` および `CheckDigit` を連結して `AccountNumber` にします。
- ペイロード・フォーマット : メッセージ・ペイロードの実際の構造を変更します。たとえば、HTML を XML に、XML を固定ファイル形式に、XML をカンマで区切られた値（CSV）に、名前値のペアを XML にというように変更します。構造変換をプログラム変換と混同しないでください。
- プログラム : プログラム言語がサポートするフォーマットとメッセージ・テクノロジがサポートするフォーマット間でメッセージ・ペイロードのフォーマットを変更します。たとえば、C 構造体を XML 文字列に、Java クラスを文字列オブジェクトに、COBOL を ADT（ Pro*COBOL と呼ばれる）に変更します。
- メッセージ API : 標準メッセージ・サービスの中には、異なるテクノロジに異なる方法で実装されているものがあります。メッセージの変換は、たとえば、OMB JMS から OJMS に、または OJMS から MQSeries JMS にのように、基礎となる構造のフォーマット間で実行する必要があります。
- ヘッダー : 1 つのメッセージ・テクノロジから別のメッセージ・テクノロジに移動、たとえば、AQ から MQSeries に、または AQ から TIBrv に移動する場合は、ヘッダーに記録されたメッセージ・プロパティのマッピングが必要になる場合があります。

セマンティック変換

Oracle CRM アプリケーションでは、顧客名を Mr./Mrs./Miss、姓、名、ミドル・イニシャルの 4 つのフィールドに格納します。これとは対照的に、SAP アプリケーションでは顧客名を、最初のフィールドで、Mr./Mrs./Miss を指定し、2 番目のフィールドで姓と名の間に空白を入れて連結して格納しています。この場合は、顧客名を SAP アプリケーションに送信する前に、Oracle アプリケーションの顧客名のフィールドを連結する必要があります。このような変換はセマンティック変換と呼ばれています。

メッセージとデータの変換に関する問題

2 つの異なる種類の変換要件の理解に加えて、特定の統合シナリオに必要なメッセージの変換方法について考えると、さらに、次の 4 つの問題を検討する必要があります。

変換ロケーション

Point-to-Point 統合アーキテクチャでは、送信側アプリケーションがメッセージを受信側アプリケーションのフォーマットに変換した後、その変換済みメッセージをメッセージ伝播インフラストラクチャ上に配置します。次に、そのインフラストラクチャがメッセージを受信側アプリケーションに渡します。

ハブ・アンド・スポーク・アーキテクチャでは、送信側アプリケーションから受信側アプリケーションにデータを変換する場合、メッセージをスポークで変換するかハブで変換するかを決定する必要があります。たとえば、送信側アプリケーションが Point-to-Point 環境の場合と同じように動作するときは、単にハブをメッセージ伝播とルーティング・インフラストラクチャとして使用します。これに対して、単一アプリケーションが情報を複数の受信側アプリケーションに送信する場合は、未変換のペイロードをハブに送信するのみです。ハブはメッセージを受信するターゲット・アプリケーションに適した変換を行います。

変換メカニズム

さらに、ハブ・アンド・スポーク・アーキテクチャでは、ハブが各送信側アプリケーションのペイロードを受信側アプリケーションのフォーマットに直接変換できます。他の選択肢として、各アプリケーションはアプリケーション固有ビューのデータを共通ビューのデータに変換してから、そのデータをハブに送信できます。ハブではそのデータを共通ビューからターゲット・アプリケーション・ビューに変換します。変換に共通ビューを使用する主な利点は、新しいアプリケーションが追加されたとき、または既存のアプリケーションがアップグレードされたとき、統合者は共通ビューからアプリケーション固有ビューへのマッピングを変更するのみで済むことにあります。ただし、このアプリケーションと通信するすべてのアプリケーションのマッピングを変更する必要はありません。したがって、変換に共通ビューを使用するほうが、直接変換するよりはるかに拡張性が高くなります。

データ変換に共通ビューを使用すると、データ自体の変換方法も影響を受けます。通常のシナリオでは、データの共通ビューを XML（Java クラスの場合もある）を使用して表現する傾向が増えています。送信側アプリケーションに関連付けられたアダプタは、アプリケーション固有のフォーマットによるデータを共通ビュー表現に変換し、そのデータを XML ドキュメントに配置してから、ハブのメッセージ交換インフラストラクチャに送信します。次に、ハブは、必要なセマンティック情報に基づいて、XSLT 変換を XML ドキュメントに適用し、そのドキュメントを発信 XML 表現に変換して、メッセージ伝播インフラストラクチャに配置します。次に、XML ペイロードを受信した受信側アプリケーションのアダプタは、XML ペイロードを解析し、それを受信側アプリケーションのデータ・フォーマットに変換します。

データを XML に変換する重要な 2 つのメリットを次に示します。

- XML メッセージ・ペイロードは HTTP-S 上で送信できます。その結果、共通インフラストラクチャは、メッセージ伝播にインターネットを使用して企業内と企業間のアプリケーションを統合できます。
- XML は、完全に拡張可能なデータ記述言語です。したがって、あらゆる種類のデータを表現できるため、変換目的で各種のデータを記述する場合には最適で理想的な言語です。

ただし、XML の使用には、次の 2 つのデメリットがあります。

- XML は、文字列指向またはテキスト・ベースの言語であり、変換にはデータを解析する必要があるため、オーバーヘッドが生じます。
- 市販されている大部分の XML パーサーでは、サイズが 10MB を超えるメッセージ・ペイロードは、その DOM API ではハンドルできません。つまり、ドキュメントの解析時に作成される DOM 解析ツリーが大きすぎるためです。SAX API を使用することもできますが、これも多くの場合効率がよくありません。

変換イベントの頻度

最後に、変換の頻度と変換が必要なビジネス・イベントについて考えてみます。変換が必要なイベントには次の 3 種類あります。

- 固有 : 変換は、特定のアプリケーション間を往復する特定のビジネス・イベントに固有です。
- 定義済み : 変換は標準メッセージ定義間のマップ（XML から SWIFT など）のセットとして定義され、ある型のメッセージを通信フォーマットの異なるアプリケーションまたはビジネス・プロセスに送信するたびに必ず適用する必要があります。
- 汎用 : たとえば、MQSeries メッセージから AQ メッセージに変更した場合、変更内容は特定のアダプタのすべてのメッセージに適用されます。

関連項目：『Oracle Applications InterConnect User's Guide』

メッセージ・システムの相互運用性

E-Business 統合のシナリオを考えるとき、理解する必要のあるメッセージ機能の最後の側面は、異なるメッセージ・キューイング・システムと異なる種類の統合ミドルウェア間の相互運用性です。

- メッセージ・システムの相互運用性 : 最初に、どのようにして 2 つのメッセージ・システムをプログラム・インターフェース・レベルで相互に通信させ、メッセージを相互に送信させるか？ 関連する質問として、次のようなものがあります。どのようにしてアプリケーションと複数のメッセージ・システムを単一の標準プログラム・インタフェースを介して通信させるか？

- メッセージ・ペイロードの相互運用性 : 第 2 に、どのようにして異なるメッセージ・システムを共通メッセージ・ペイロード・フォーマットを介して通信させるか、または他のシステムのメッセージ・ペイロードを解析させるか？

ペイロードとシステムの両レベルでメッセージ・システム間の相互運用性を保証するために、メッセージ・ミドルウェア・ベンダーは、次の 2 つの規格を採用しています。

- Java Messaging Service（JMS）API。メッセージ・システムの相互運用性を保証します。
- XML（OAG、XML、OBI などの多様な派生製品を含む）。メッセージ・ペイロードの相互運用性を保証します。

次の項では、この 2 つの規格の概要を説明します。

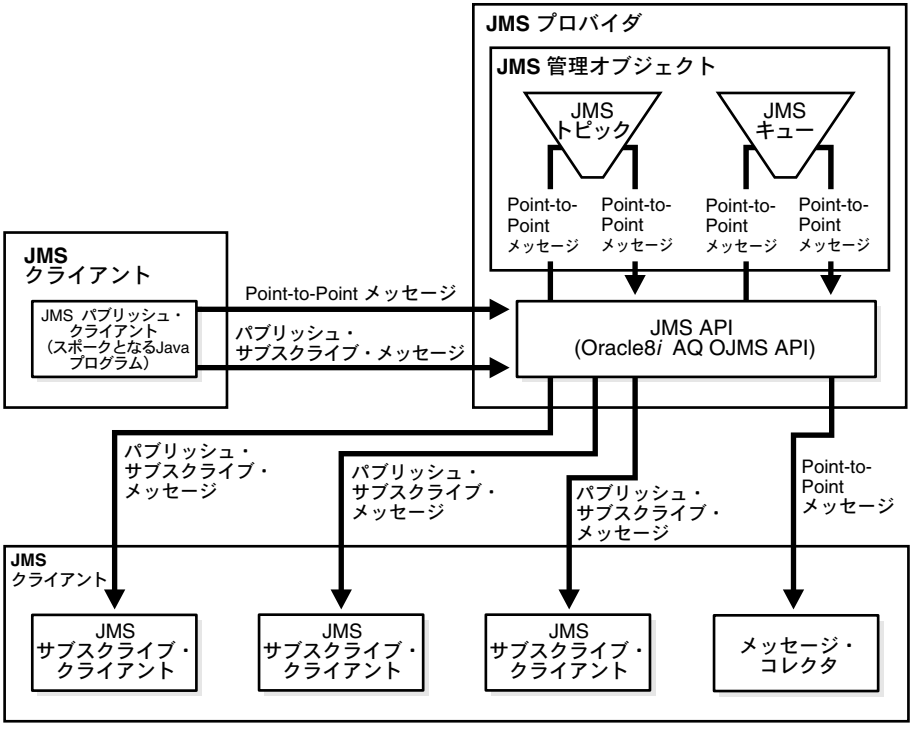
Java Messaging Service

Java Messaging Services（JMS）は、非同期メッセージ機能に対する Java 業界標準です。Sun Microsystems 社およびオラクル社を含むその他のエンタープライズ・メッセージ・ベンダーが共同で開発しました。これは、Java Remote Method Invocation（RMI）が指定する同期通信モデルを補完します。

JMS についての最も一般的なイメージは、JMS はメッセージ・サービスを提供するアプリケーションまたはコンポーネントであるというものです。JMS そのものは、単なる標準の定義です。メッセージ・テクノロジのベンダー（オラクル社など）は、JMS 標準に準拠し、JMS 標準準拠の Java API を介してアクセスできるメッセージ・サービスを提供するソフトウェア（Oracle アドバンスト・キューイングなど）を開発し販売しています。

これらの JMS 準拠サービスは、プログラム・インタフェース（API）に対する JMS 標準に準拠している公開インタフェースを介して使用できます。API が提供するフレームワークを使用すると、移植可能なメッセージ・ベースのアプリケーションを Java プログラミング言語で開発できます。

図 4-5 Java Messaging Service



JMS 標準は、次の 4 つの重要なコンポーネントを定義しています。

- クライアント : このプログラムまたはプロセスは、メッセージを作成して使用します（たとえば Java アプリケーション・プログラム）。
- プロバイダ : このプログラムまたはプロセスは、JMS サービス（AQ など）を提供します。
- 管理オブジェクト : このオブジェクトは、プロバイダがメッセージ（たとえば、JMS トピックおよび JMS キューなど）を管理し、ルートするために使用します。
- メッセージ : メッセージそのものです。

JMS 標準は、次の 2 種類のメッセージ・ルーティングを定義しています。

- JMS トピック。パブリッシュ・サブスクライブ ・ルーティングの定義を提供します。
- JMS キュー。あるポイントから別のポイントに移動するメッセージの Point-to-Point ルーティングの定義を提供します。

JMS 標準は、メッセージそのものに関連するプロパティ（メッセージ・ヘッダーに格納）およびメッセージ・コンテンツのペイロードが持つ多彩なスタイルなど、メッセージについての様々な特性を定義しています。さらに、メッセージ・ヘッダーとそのプロパティに、フィルタ条件を使用したコンテンツ・ベースのパブリッシュ・サブスクライブの実装方法も定義しています。次の 5 つのスタイルがあります。

- `TextMessage`
- `BytesMessage`
- `MapMessage`
- `StreamMessage`
- `ObjectMessage`

Oracle による JMS の実装

JMS の実装を提供する各ベンダーは、標準に準拠したサービス・セットを開発しています。ベンダーは自由に実装を拡張し、JMS 標準で定義されているサービスを超えたサービスを提供しています。

Oracle8*i* アドバンスト・キューイングは、Oracle Java Messaging Services（OJMS）と呼ぶ拡張 JMS インタフェースを提供しています。また、Oracle は、アドバンスト・キューイングのシングル・コンシューマ・キュー機能を使用した JMS キューを提供しています。各 JMS キューは 1 対 1 でシングル・コンシューマ・キューにマップされます。JMS トピックは、アドバンスト・キューイングのマルチ・コンシューマ・キューによってサポートされます。すべての JMS トピックは、マルチ・コンシューマ・キューにマップされます。すべての JMS メッセージ型は Oracle オブジェクト型にマップされます。JMS 接続は JDBC 接続をカプセル化します。このカプセル化によって、クライアントは、JMS メッセージ機能と他の JDBC 操作を同じデータベース・トランザクションで組み合せることができます。データベース JVM の外で実行する JMS クライアントは、Thin JDBC ドライバまたは OCI JDBC ドライバを使用して JMS API にアクセスできます。Oracle8*i* JVM の内部で実行する JMS クライアントは、Oracle Server Driver を使用して JMS API にアクセスできます。標準 JMS インタフェースは、`javax.jms` パッケージで使用できます。

JMS 実装に対する Oracle の拡張機能は、`oracle.jms` パッケージで確認できます。

> **関連項目：**『Oracle8*i* アプリケーション開発者ガイド アドバンスト・キューイング』

XML

XML（eXtensible Markup Language）は構造化情報を含んだドキュメント用のマークアップ言語です。メッセージまたはドキュメントなどの情報セットに構造を適用するためのメカニズムを提供します。これによって、セット内のデータはラベルが付けられ、属性、特性またはデフォルト値が割り当てられ、情報セットの確認または検証を実行できます。XML は、E-Business 用の標準メッセージ・フォーマットになりつつあり、大部分の大手 IT ベンダーにとって最優先の標準です。

XML の主な利点は、その柔軟性と単純さにあります。すべてのメッセージの中には、メッセージの構造を理解する上で必要な情報が含まれています。このために、自己記述型メッセージと呼ばれています。したがって、XML のルールを認識しているプログラムは、XML メッセージとしてフォーマット化されたメッセージはすべて理解できます。各メッセージの構造についてメタデータを格納したりオーバーレイする必要はありません。

XML は標準汎用マークアップ言語（Standard Generalized Markup Language: SGML）のサブセットとして導出されました。SGML は過去 10 年以上にわたって、構造化ドキュメントのリポジトリをメンテナンスするための標準的なベンダーに依存しない方法でした。SGML は、E-Commerce に使用するにはあまりに高性能で複雑です。そこで、XML が SGML の簡素化バージョンとして開発されました。

メッセージまたはドキュメントのコンテンツは、情報の論理セットと考えることができます。その情報は頻繁に構造化する必要があります。その結果、情報のピースはメッセージまたはドキュメントのサブセットとして識別、アクセスおよび操作できます。これらのサブセットは要素と呼ばれています。各要素は、任意の数の他の要素で作成できます。また、要素は 1 つのグループ内で繰返し使用できます。

XML が持つ構文セマンティックによって、要素の開始点と終了点を定義するタグと呼ばれるラベルを定義できます。また、全体としての要素の特性を記述する属性を定義できます。この属性は、すべてのサブ要素と要素内に含まれるデータに適用されます。

さらに、XML によって、文書型定義、つまり、DTD と呼ばれる検証テンプレートを定義できます。DTD を使用すると次の 2 つのレベルの検証が実行できます。

- 適切な構成 : メッセージまたはドキュメントが正しく構成されていることをチェックします。合格すれば、そのメッセージまたはドキュメントは適切な構成と記述されます。
- 有効 : メッセージまたはドキュメントが DTD で定義したルール・セットに照合して有効であることをチェックします。検証は、XML パーサーと呼ばれるサービスが実行します。Oracle を含めた大部分のベンダーのソフトウェアは、このサービスを提供しています。

関連項目：『Oracle8*i* XML リファレンス・ガイド』

第II部

製品

第 II 部では、統合との関連から OIS の各製品コンポーネントを説明します。製品別の詳細は、製品固有のドキュメントを参照してください。製品固有の適切なドキュメントへのリンクがあります。

5

同期アプリケーションの統合

Oracle8*i* Java 仮想マシン（JVM）は、サーバー指向の Java アプリケーションを実行するための、拡張性と可用性に優れた環境を提供します。

この章には次の項目が含まれています。

- Oracle8i JavaVM が提供する機能
- Oracle データベースを使用した Java アプリケーションの開発
- Oracle8i JavaVM での Java サポート : アーキテクチャの概要

Oracle8*i* JavaVM が提供する機能

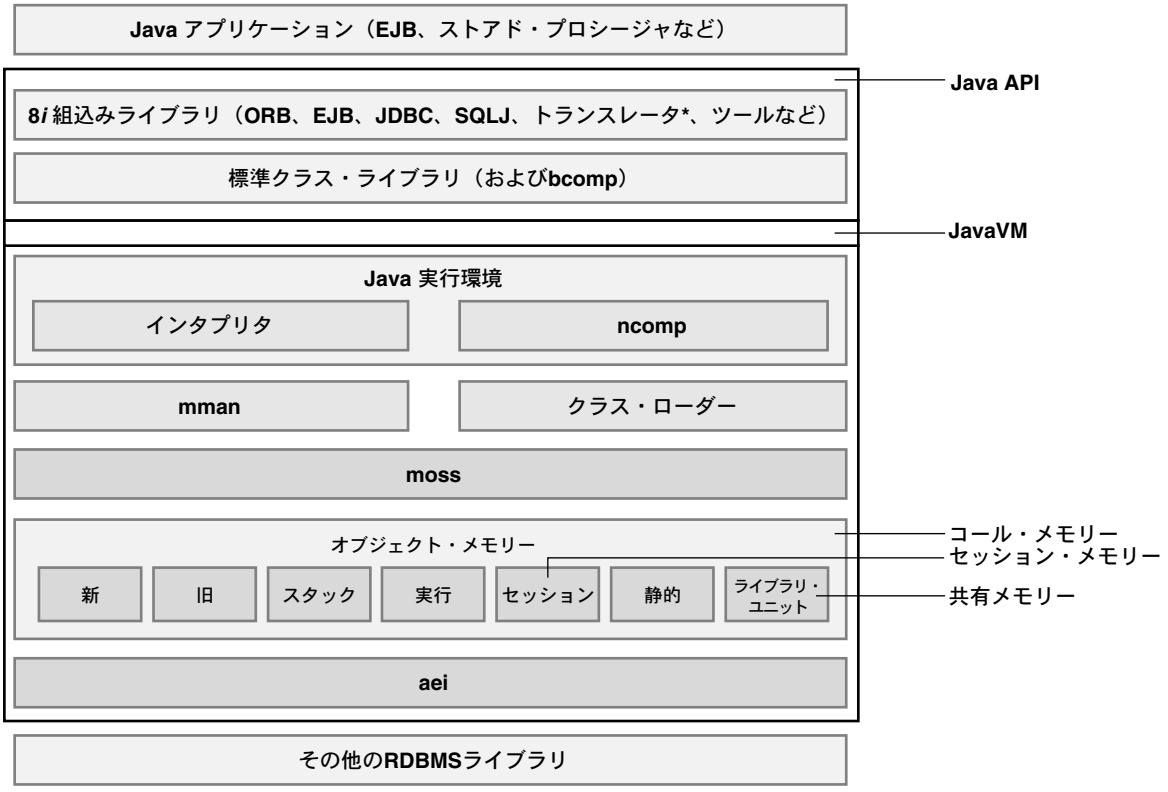
Oracle8*i* JVM には、キーとなるアーキテクチャ・コンポーネントがいくつかあります。この項では、次の項目を説明します。

- JavaVM の中心的な機能
- JavaVM の中心的なランタイム機能
- JavaVM とデータベースとの統合

JavaVM は 100% 完全な Java 実行環境であり、Java 言語仕様と JavaSoft 社の JDK 1.1.6 規格に指定されている JavaVM に準拠しています。また、データベースと緊密に統合され、バイトコード・コンパイラ、ガベージ・コレクタ、統合された Java クラス・ローダーおよび Java-through-C コンパイラなど、多数のコンポーネントを備えています。これらすべてのコンポーネントは、データベース環境において最適なパフォーマンスと拡張性が得られるように設計されています。

JavaVM は、データベース・カーネルと同一のプロセス空間とアドレス空間で実行され、ヒープ・メモリーを共有し、パフォーマンスを最適化するためにデータベースのバッファ・キャッシュに直接アクセスして、メモリーのコピーを回避します。JavaVM は、Java オブジェクトに対してランタイム環境を提供します。Java データ構造すべての表記が含まれており、Java メソッド・ディスパッチをサポートしています。さらに、リカバリ不可能な Java の例外モデルや Java 言語レベルのスレッドも含めた標準 Java バイトコード操作のすべてをサポートしています。VM のディスパッチと例外モデルは、ハイブリッドのインタプリタまたはコンパイル済みシステムをサポートするように設計されているため、ユーザーは Java-through-C 変換にアクセスし、パフォーマンスを向上させることができます。

図 5-1 Oracle JavaVM のアーキテクチャ



JavaVM の中心的な機能

JavaVM の中心的な機能は、次のとおりです。

- **オブジェクト・メモリー管理 :** Oracle8*i* JavaVM は、**オブジェクト・メモリー**と呼ばれる標準チャンクに対してメモリーの割当てと解放を行います。このメモリーは用途に従って、様々な方法で特化できます。オブジェクト・メモリーには、次の 4 つの重要な利点があります。
  - オブジェクト・メモリーの位置は透過的であるため、オブジェクトのシリアライズ化とデシリアライズ化に付随する手間をかけず、しかも Java の均一参照セマンティックを保持しながら、スレッド、プロセス、ホストまたは一時的な境界間で効率よく再配置できます。これは、高度なロード・バランシングとフェイルオーバー

が Oracle VM によって提供され、低レベルでエラーになりやすい Java スレッドをユーザーが拡張のために使用しなくてすむことを意味します。

- オブジェクト・メモリーは、ガベージ・コレクションの方法で最適な内容に特化できます。この場合、オブジェクト・メモリーは、事前に初期化された状態かどうか、またはトランザクション用かどうかにかかわらず、また、ロールバックおよびアトミック・コミット・セマンティックが必要かどうかに関係なく特化できます。
- 永続および共有オブジェクト・メモリーは、ディスクに効率的にアーカイブできます。
- オブジェクト・メモリーは、他の JavaVM および Java アプリケーションの実装方法から影響を受けません。

- **メモリー・マネージャとガベージ・コレクタ :** JavaVM には、優れたパフォーマンスと拡張性を Oracle データベース環境に提供するための最適化されたガベージ・コレクタがあります。ガベージ・コレクタは、VM の Java メモリー・ヒープを自動的に管理し、オブジェクト・メモリーの効果的な割当てと収集を行います。
- **Java Library Manager:** Java Library Manager は、データベース内の Java プログラムをロード、格納および管理するための機能を提供する JavaVM のコンポーネントです。ユーザーは、ソース、バイナリとリソース、およびアーカイブの 3 種類のフォームで Java を交換（インポートとエクスポートの両方）できます。Java プログラムはデータベース・ライブラリ・ユニット（OS ファイルと等価）として格納されます。
- **Java クラス・ローダー :** クラス・ローダーは、実行時に JavaVM から要求を受け入れ、ローカル DBMS に格納されている Java クラスの位置を（Java バイナリまたはシステム固有のコンパイル済みフォームで）識別してロードし、初期化します。

JavaVM の中心的なランタイム機能

JavaVM の中心的なランタイム機能は、次のとおりです。

- **バイトコード・コンパイラ :** JavaVM には、標準 Java ソース・プログラムを標準 `Java.class` バイナリ表記に変換する JavaSoft 社の標準 Java バイトコード・コンパイラが組み込まれています。
- **インタプリタ・ランタイム :** Oracle8*i* JavaVM は、バイトコード・インタプリタと標準 Java バイナリを実行する関連 Java ランタイムを提供しています。このインタプリタは、Java スレッドや例外などの高度な機能も含めた標準 Java（現在はリリース 1.1.6）を 100% 実装しています。さらに、このランタイムは、ホスト環境からのコールインとコールアウト、システム固有のメソッド、およびシステム固有にコンパイルされた Java モジュールもサポートしています。
- **標準ライブラリ :** Oracle8*i* JavaVM は、GUI 関連コンポーネント、Abstract Windowing Toolkit（AWT）を除くすべての標準 JDK 1.1.6 ライブラリをサポートしています。

- **ネイティブ・コンパイル :** Oracle8*i* JavaVM には、Java プログラムのパフォーマンスをコンパイル済み C 言語のレベルにまで向上させるネイティブ・コード・コンパイラが組み込まれています。ネイティブ・コンパイラは、標準 Java バイナリを C 言語の実行可能ファイルに変換します。このファイルは、ライブラリ・ユニットとしてデータベースに永続的に格納され、JavaVM では、DLL または .so としてロードできます。
- **Java ネイティブ・インタフェース :** JavaVM は、システム固有のメソッドに対するアクセスを提供する Java ネイティブ・インタフェースをサポートしています。

> **注意 :** オラクル社は、データベースのアドレス空間に危険を伴う C 言語ライブラリがリンクされないように、JNI を外部の汎用アプリケーション開発者に提供していません。

ただし、JavaVM では、データベースから C および C++ プログラムにコールアウトする機能は提供しています。

JavaVM とデータベースとの統合

JavaVM は、他の Oracle8*i* データベースとのインタフェースを 4 つの異なるメカニズムを介して行います。

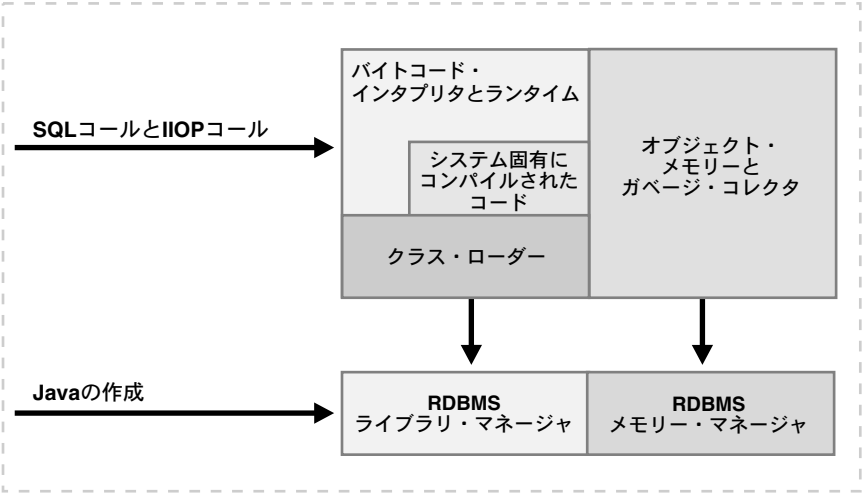
- **JDBC ドライバと Java トランスレータ内の埋込み SQL :** 永続データは、リレーショナル表として、または SQL を使用して定義されたオブジェクト・リレーショナル表として、Oracle データベースに格納されています。データベース内の Java プログラムは、SQL または JDBC を使用してデータベース内の SQL と PL/SQL にアクセスします。静的な SQL 問合せについては、JDBC よりも簡単な SQLJ アプリケーションで記述できます。動的な SQL を記述するには、JDBC をコールします。
- **JDBC ドライバ :** VM には、Oracle JDBC ドライバの特化された実装内容が組み込まれています。JDBC ドライバは、JDBC 1.2.2 仕様に対応しており、オブジェクト・リレーショナルなタイプ、コレクションと LOB に対するサポートを含め、すべての JDBC 2.0 仕様に実質的に対応しています。また、Oracle クライアント側の JDBC または Oracle コール・インタフェースと同じプログラミング API、および Thin JDBC ドライバを移植します。これにより、アプリケーション開発者はアプリケーション・コードを変更せずにデータベースに Java アプリケーションを移植できます。クライアント側の JDBC ドライバとサーバー側のドライバとの唯一の違いは、サーバーの Java オブジェクトは、コールされたデータベース接続のコンテキスト内で常に実行されるため、結果的に、このドライバでは接続を明示的にオープンできない点にあります。
- **Java トランスレータ内の埋込み SQL :** サーバーの JavaVM は、Java トランスレータ内に埋込み SQL を提供しています。ストアド・プロシージャとファンクションを、さらにレベルが高く、生産性の高い SQLJ インタフェースに書き込むことができます。Java トランスレータ内の埋込み SQL は、サーバーの JavaVM により透過的に起動され、埋込み SQL 文を使用するアプリケーション・コードを、埋込み JDBC コールを使用する標準

Java コードに変換します。変換されたコードは、JavaVM の埋込みバイトコード・コンパイラでコンパイルされ、そのランタイム機能で実行できます。JDBC へのコールは、実行時に、データベース内に埋め込まれた SQLJ ランタイムにルート変更されます。

- **相互言語メソッド・サービス（Inter-Language Method Services: ILMS）:** ILMS はデータベース内の共通コンポーネントで、SQL、PL/SQL、Java および C による外部プロシージャ間における言語間でのメソッドの呼出しをすべて管理します。SQL と PL/SQL は、ILMS を使用してデータベースの Java ストアド・プログラムをコールします。これは、SQL がデータベース内の PL/SQL をコールするために使用するメカニズムと同じメカニズムです。
- **CORBA Object Request Broker:** Oracle8*i* データベースは、Java ベースの CORBA 2.0 準拠の Object Request Broker （ORB）、Visibroker（Visigenics Java ORB）を統合しているため、CORBA IIOP プロトコルを使用しているデータベースをコールインおよびコールアウトできます。Oracle8*i* 内の Java ストアド・プログラムと Enterprise JavaBeans は、IIOP で起動できます。この場合、データベースは CORBA サーバーとして動作します。同様に、Oracle8*i* からの IIOP コールアウトによって、データベースは標準 CORBA クライアントとして動作可能になります。ORB は Oracle8*i* 内で効果的に使用されます。Visigenics ORB コンポーネントは、IIOP プロトコルの解釈とオブジェクト活性化のみに使用され、データベースのマルチスレッド・サーバー・カーネルは、拡張性について使用されます。データベースは、Netscape 4.0 や中間層 ORB などのブラウザに対する直接サポートも含め、各種の CORBA クライアントをサポートしています。さらに、データベースの CORBA 機能は、自動化された Caffeine Java-to-IDL コンパイラなど、多数の自動化ツールによって Java 環境に適合しています。
- **開発ツールのサポート :** Oracle8*i* JavaVM は、データベースの Java 機能を簡単に使用するためのコマンドライン・ツールを多数提供しています。次のようなツールがあります。
  - データベースへの Java のロード、およびデータベースへの Java ストアド・プログラムのドロップ
  - データベースへの CORBA サーバーの配置と登録の自動化
  - Oracle8*i* での Enterprise JavaBeans のパッケージ化と配置の自動化

  これらのツールは、Oracle JDeveloper 2.0 ツールに統合されています。

図 5-2 Oracle JavaVM のランタイム・コンポーネント



Oracle8*i* JavaVM は、3 種類の方法でプログラムできます。

- Java ストアド・プロシージャ
- Java で定義された CORBA サービス
- Enterprise JavaBeans

Oracle データベースを使用した Java アプリケーションの開発

Oracle8*i* の JavaVM は、3 つのタイプの Java アプリケーションの開発に使用できます。

- **データベース・ストアド・プロシージャ、トリガーおよびメソッド :** これらは、従来のデータベース・プログラマと SQL 指向のデータベース・クライアントをサポートします。
- **Enterprise JavaBeans:** JavaVM は、Enterprise JavaBeans と呼ばれる分散 Java コンポーネントに対してトランザクション・サーバー・プラットフォームを提供します。
- **Java の CORBA サーバー :** Oracle8*i* によって、分散システムの開発者は、データベースの JavaVM 上に Java の CORBA サーバーを実装することもできます。

Oracle8*i* JavaVM が提供する CORBA に対するサポートにより、同期をとりながら要求に対して応答を行う方法で通信を行うアプリケーションにとって理想的なプラットフォームが作成されます。Oracle8*i* JVM は、次のような標準 CORBA サービスを提供しています。

- CORBA 2.0 完全準拠の ORB
- Java で CORBA アプリケーションを開発するための IDL コンパイラも含めた標準 CORBA ツール

この項では、次の項目を説明します。

Oracle8*i* の CORBA 機能

前述の IIOP 機能と ORB 機能に加え、データベースには様々な追加の CORBA 機能が統合されています。

- **トランザクション** : Oracle8*i* JavaVM は、埋込み JDBC ドライバを使用した CORBA Object Transaction Service（OTS）の API を提供しています。この API は、OTS 参照可能トランザクションをサポートするために拡張されています。この OTS は、JavaVM 上に実装されている CORBA サーバーにトランザクション・プロパティを提供します。
- **ディレクトリ / ネーミング :** JavaVM は、業界標準のディレクトリ・サービスである LDAP（Lightweight Directory Access Protocol）に対して標準的な COSNaming インタフェースを提供しています。したがって、この業界標準に準拠したディレクトリ・サービスを使用して、データベースの CORBA ORB に登録されたサーバーの EJB コンポーネントにアクセスすることができます。また、CosNaming/LDAP 準拠のネーム・サービスを統合し、分散コンポーネントの登録と検索を簡素化します。
- **オブジェクト・アダプタ :** Oracle8*i* は、永続 CORBA オブジェクトに対してオブジェクト・アダプタを提供しています。このアダプタには 2 つの重要な機能があります。第 1 に、このアダプタはデータベース内に公開されている CORBA オブジェクトのディレクトリとして動作します。第 2 に、CORBA クライアントによる初期活性化の際、CORBA オブジェクトの検索とロードに役立ちます。オブジェクト・アダプタは、公開されている CORBA オブジェクト名、パブリッシャのスキーマ名、および CORBA オブジェクトを実装している Java クラス名の 3 種類の情報を持つ標準データベース表として実装されています。オブジェクト・アダプタはデータベース表であるため、拡張性と高速アクセスのためにハッシュ索引付けされています。オブジェクト・アダプタは、膨大な数の CORBA オブジェクトと Enterprise JavaBeans をサポートするために、複数のデータベース・パーティションに分割することもできます。
- **Java CORBA サービス :** Oracle8*i* データベースは、Java プログラマが CORBA サービスを簡単に開発できるように、多数の機能を提供しています。第 1 に、IDL を必要としない Caffeine（Java から IIOP への直接マッピング）をサポートしています。第 2 に、値と拡張可能な構造体で標準オブジェクトをサポートしているため、Java プログラマにとって非常に便利な機能を提供します。最後に、アプリケーション開発を簡素化するための `Java2IIOP`、`IDL2Java` および `Java2IDL` も含めた多数のツールを提供しています。

- **セキュリティ**: データベース内の CORBA サービスは、Enterprise JavaBeans 実行環境と同様に、次の 3 つの厳重なセキュリティ・レイヤーによる高度なセキュリティ環境で実行されます。つまり、SSL から IIOP までの暗号化、従来のデータベース・ユーザー名とパスワードを使用した認証、およびロールと権限を使用したアクセス管理です。

  関連項目：『Oracle8*i* CORBA 開発者ガイド』

Oracle8*i* JavaVM は、Enterprise JavaBeans コンポーネント・モデルおよび多数の標準 EJB サーバーもサポートしています。

Enterprise JavaBeans の概要

Enterprise JavaBeans（EJB）は、明確に指定されたインタフェースを使用したコンポーネントとして定義された比較的荒削りな一連の Java アプリケーション・ロジックであり、EJB トランザクション・サーバーが提供するホスト・システム・インフラストラクチャで効果的に実行できます。EJB は、特異な Java クラスではありませんが、ホスティング・サーバーが指定する一連の制約に従う必要があります。EJB コンポーネント・モデルは、サーバー側のビジネス論理に関する便利で生産性の高いコンポーネント・モデルを Java アプリケーション・プログラマに提供し、次の重要な方法によってアプリケーション・コードの再利用と複数層アプリケーションの開発を容易にします。

外部インタフェース定義言語（Interface Definition Language: IDL）不要 : EJB では、高レベルのインフラストラクチャとプログラミング・インタフェースが定義されているため、CORBA や D/COM などの外部 IDL の知識がなくても、アプリケーション開発者はコンポーネント・モデルを Java で完全に指定できます。

システム・レベルのプログラミング不要 : EJB はトランザクション・サーバーの概念に基づいて設計されています。つまり、拡張性、ロード・バランシング、接続プーリング、フェイルオーバーおよびトランザクションなどのシステム・レベルのインフラストラクチャとサービスが提供されます。これにより、アプリケーション・プログラマは低レベルのシステム・ソフトウェアとの関わりから解放されます。トランザクション・サーバーがトランザクションと状態の管理を引き受けるため、結果として、プログラマは EJB コンポーネントのトランザクション動作を宣言的に指定できます。

実行環境の多様性 : EJB の仕様は、EJB コンポーネントを多様なホスト環境に配置して実行できるように設計されているため、TP モニター、アプリケーション・サーバーおよびデータベースを含めた EJB トランザクション・サーバーとして動作します。Oracle は、Oracle8*i* と Oracle Application Server を統合した JavaVM 上の EJB コンポーネントに対して一貫したモデルを提供し、2 つの環境間における共通サービス・セットの共有を提供します。

通信インフラストラクチャの多様性 : EJB は、クライアント、中間層およびデータベース間で使用される特定の通信インフラストラクチャから Java 開発者を解放するために設計されています。特に、EJB によって、Java 開発者はアプリケーションを EJB コンポーネントとして設計および実装できます。この EJB コンポーネントは、CORBA/IIOP、DCOM および Java 自体のリモート・メソッド起動（Remote Method Invocation: RMI）プロトコルを含めた多様な通信インフラストラクチャに移植できます。EJB アプリケーションは基礎となるイ

ンフラストラクチャと 1 方向で互換性があります。Oracle は、通信インフラストラクチャとして CORBA/IIOP に対するサポートを選択しました。

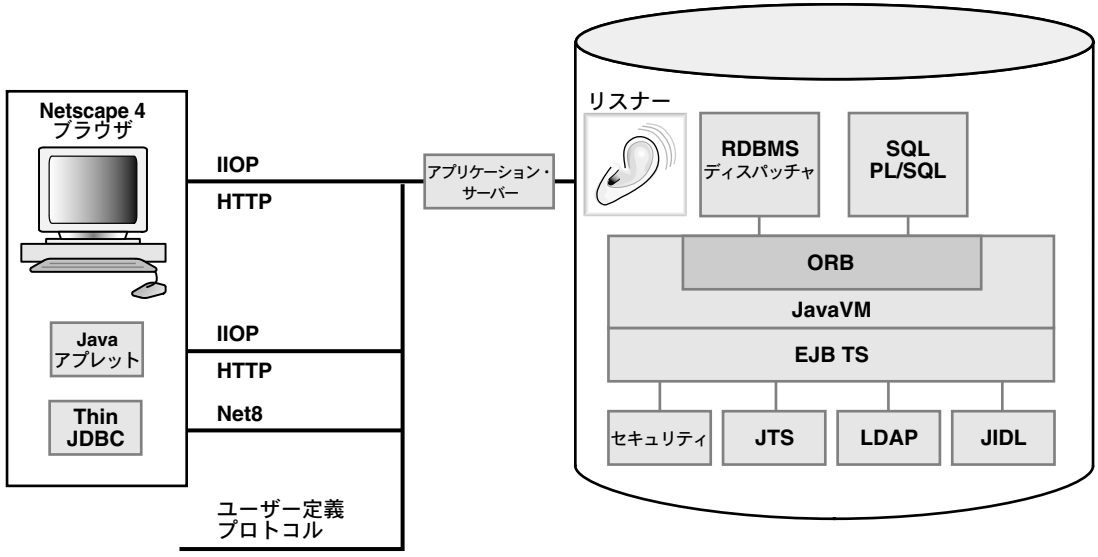
生産性 : EJB は、サーバー側 Java コンポーネントのトランザクションとセキュリティのプロパティをプログラム的でなく宣言的に指定する方法を提供しています。Java ツールは、JDeveloper も含めて、Java 開発者が EJB 指定のインタフェースを直接サポートできるように設計されており、ビジネス論理を獲得するサーバー側 EJB コンポーネントを Thin クライアントと組み合せるための生産性の高い環境をユーザーに提供しています。

代替のオープン・システム : Microsoft トランザクション・サーバー内で実行される Visual Basic コンポーネントとは対照的に、Oracle8*i* の Enterprise JavaBeans 機能は、2 つの重要な利点を提供します。つまり、Java ベースのビジネス・オブジェクト・ロジックをサポートする非パラレル・データ管理機能と、顧客を専有的なベンダー固有のソリューションに拘束しないオープンで移植可能なアプリケーション・ソリューションの両方を提供する統合サーバーです。

Oracle8*i* JavaVM での Java サポート : アーキテクチャの概要

Oracle8*i* データベースは、データベースを生産性と拡張性の高い EJB トランザクション・サーバーにするための多数の機能を提供しています。アーキテクチャ的には、図のように異なるコンポーネントを統合しています

図 5-3 Oracle8*i* 分散オブジェクトのアーキテクチャ



この項では、次の項目を説明します。

セッション管理機能

サーバーのセッション管理機能は、IIOP 起動の低レベルでの登録をサポートし、セッションに対する IIOP 要求のデータ依存ルーティングを提供し、関連リソースを解放します。これらの機能は、次のコンポーネントを含めたデータベースのマルチスレッド・サーバー・プラットフォームの異なるコンポーネントによって提供されます。

- **IIOP リスナー :** このリスナーは、Oracle8*i* データベースの 1 コンポーネントで、データベースが回線から受信したメッセージにアクセスして変換します。リスナーは、デコードされたメッセージに基づき、必要に応じてデータベースのディスパッチャを起動します。Oracle8*i* リスナーは、標準 CORBA/IIOP バインド・プロトコルをサポートするために拡張されており、ユーザーは IIOP を使用してデータベースのコールインとコールアウトを行うことができます。
- **IIOP ディスパッチャ :** このディスパッチャは、データベース内のタスクとセッションをスケジュールするためのデータベース・コンポーネントです。Oracle8*i* では、IIOP ベースのメソッド起動をサポートするためにディスパッチャが拡張されています。処理とディスパッチを行うために、ディスパッチャはこれらのメッセージを順番に CORBA 2.0 ORB に送信します。
- **CORBA 2.0 ORB 統合 :** ディスパッチャが ORB を起動すると、その ORB はコールされた IIOP メソッドを整理し、起動が実行される適切な EJB コンポーネント（ORB に登録されている）を識別してから、メッセージを JavaVM にディスパッチします。データベースの EJB コンポーネントは、ORB インフラストラクチャを使用して、他の分散コンポーネント、つまり、他の EJB サーバーの CORBA コンポーネントまたは EJB のいずれかをコールアウトすることもできます。
- **マルチスレッド・サーバー・プラットフォーム :** Oracle8*i* JavaVM は、マルチスレッド・サーバーが提供する独自の機能セットを活用します。このセットには、拡張性のある EJB サーバーにするための、ロード・バランシング、フェイルオーバー、データ依存ルーティング、接続プーリングおよび共有メモリー管理が含まれています。
- **JavaVM 実行プラットフォーム :** Oracle8*i* JavaVM には、EJB コンポーネントに必要な実行環境が用意されています。これらのコンポーネントは、標準 Java アプリケーション・プログラムであるため、この章で前述した VM のパフォーマンスと拡張性に関する最適化の内容をすべて活用できます。
- **永続的な状態へのアクセス :** データ・サーバー内の EJB コンポーネントは、そのコンポーネントの永続的な状態にアクセスします。この状態とは、JDBC ドライバまたはサーバー内に統合されている SQLJ トランスレータを通じ、トランザクションの境界を越えて存続している状態を指します。永続的な状態は、リレーショナル表またはオブジェクト・リレーショナル表として格納できます。

Enterprise JavaBeans サービス

複数層の環境全体にわたる透過的な相互運用性を実現するために、Enterprise JavaBeans は、ネットワーク・コンピューティング・サービスの共通セットを活用します。重要なサービスは、次のとおりです。

- **トランザクション :** Oracle8*i* JavaVM は、埋込み JDBC ドライバを使用する Java Transaction Service（JTS）API を提供しています。このドライバは、JTS 参照可能トランザクションをサポートするように拡張されています。JTS は、JavaVM 上の EJB コンポーネントにトランザクション・プロパティを提供します。
- **ディレクトリ・ネーミング :** Oracle8*i* JavaVM は、業界標準の LDAP に対応するディレクトリ・サービスに JNDI（Java Naming and Directory Interface）インタフェースを提供しています。サーバーの EJB コンポーネントはディレクトリ・サービスに配置され、そこから JNDI を使用してアクセスすることができます。
- **セキュリティ :** Oracle8*i* データベースは、セキュア IIOP と Secure Sockets Layer（SSL）機密機能を含めた各種のセキュリティ・メカニズムを採用しています。このメカニズムは、Netscape 4.0 ブラウザで現在使用され、イントラネットとインターネットにおけるデフォルト規格です。
- **Net8 接続マネージャ :** 接続マネージャは、Net8 接続に加え、多重 IIOP 接続まで拡張されています。

関連項目 : 『Oracle8*i* CORBA 開発者ガイド』

6

データ・レプリケーションとゲートウェイ

分散データベース・システムにおける Oracle Replication とは、個別の Oracle データベースにある同じデータの複数コピーを保持するプロセスです。レプリケーションは、あるロケーションのデータに対して行われた変更を、各リモート・ロケーションに転送して適用する前に、取得して格納します。各ロケーションのユーザーは、一貫して同じデータを参照します。標準的な分散データベース・システムのように、データに遠隔的にアクセスする必要はありません。この章には次の項目が含まれています。

Oracle Replication の概要

レプリケーションは、分散データベース・システムを構成している複数のデータベースに、表などのデータベース・オブジェクトをコピーして保持します。Oracle Replication は、Oracle データベース・サーバーに完全に統合された機能であり、別個のサーバーではありません。

レプリケーションでは、分散データベース・テクノロジを使用して複数のサイト間でデータを共有しますが、レプリケート・データベースと分散データベースは同じものではありません。分散データベースでは、データを多数の場所で使用できますが、特定の表は 1 箇所にのみ常駐しています。

たとえば、DB1、DB2 および DB3 データベースを含んだ分散データベース・システムで、emp 表は DB1 データベースにのみ常駐しています。レプリケーションの目的は、同一のデータを複数の場所で使用可能にすることにあります。たとえば、emp 表を DB1、DB2 および DB3 で使用可能にすることができます。

この項では、次の項目を説明します。

レプリケーションの利点

一般的に、レプリケーションは次のような目的で使用されます。

- **可用性 :** レプリケーションにより、データ・アクセスの代替手段が提供されるため、アプリケーションの可用性が改善されます。あるサイトが使用不能になった場合、ユーザーは引き続き残りのロケーションを問い合せたり、更新することができます。つまり、レプリケーションは優れたフェイルオーバー保護機能を提供します。
- **パフォーマンス :** レプリケーションにより、複数のサイト間でアクティビティが均衡化されるため、共有データへの高速なローカル・アクセスが可能になります。ユーザーが特定のサーバーにアクセスしている間に、他のユーザーは他のサーバーにアクセスできるため、すべてのサーバーの負荷が削減されます。また、ユーザーはアクセス・コストが最も低いレプリケーション・サイト（通常、地理的に最も近距離にあるサイト）からデータにアクセスできます
- **モバイル・コンピューティング :** スナップショットは、ある時点における、ターゲット・マスター表の完全なまたは部分的なコピー（レプリカ）です。スナップショットによって、ユーザーは中央のデータベース・サーバーに接続していない間、そのデータベースのサブセットで作業できます。後で接続が確立されたとき、必要に応じてスナップショットを同期化（リフレッシュ）できます。ユーザーがスナップショットを**リフレッシュ**すると、そのユーザーが行ったすべての変更内容と切断中に発生した変更内容に基づいて中央のデータベースが更新されます。

- **ネットワーク負荷の削減 :** レプリケーションを使用すると、データを複数の地域に分散できます。アプリケーションは単一の中央サーバーにアクセスしないで、各地域のサーバーにアクセスします。この構成によって、ネットワーク負荷が大幅に削減されます。
- **大量配布 :** 組織では、データの使用と操作が必要なアプリケーションを大量に配布する必要性がますます高まっています。Oracle Replication では、配置テンプレートによって複数のスナップショット環境を迅速に作成できます。変数を使用すると、各スナップショット環境を個々のニーズにあわせてカスタマイズできます。たとえば、営業活動の自動化に配置テンプレートを使用できます。この場合、テンプレートには、各営業地域と営業担当に関する変数が含まれます。

レプリケーションは、異なる場所にあるシステム間でデータを同期化するには理想的で、データに遠隔的にアクセスすることなく、各データの表示の一貫性を保持できます。

レプリケーションの使用

レプリケーションでは、論理的に関連しているオブジェクトを**レプリケーション・グループ**に編成し、表の他に、ビュー、データベース・トリガー、パッケージ、索引およびシノニムなど、その他のサポート・オブジェクトをレプリケートできます。レプリケーション・オブジェクトは、分散データベース・システムにおける複数サーバー上に存在するデータベース・オブジェクトです。レプリケーション環境では、あるサイトでレプリケーション・オブジェクトが更新されると、その更新内容は他すべてのサイトにあるコピーに適用されます。レプリケーションは、次の場合に使用します。

- データとスキーマに対する変更を 1 つのマスター・グループから多数のスナップショット・グループにレプリケートする場合
- データとスキーマに対する変更を 2 つ以上のマスター・グループにレプリケートする場合
- データに対する変更を 1 つのスナップショット・グループからマスター・グループにレプリケートして戻す場合
- データの読取り専用マテリアライズド・ビューを別のデータベース上に作成する場合
- マスター・グループ間での更新内容の差異を事前に作成した方法で自動的に解決する場合
- データセットのコピーを更新するために、レプリカ・プロシージャの実行をレプリケートして同期化する場合

特定のレプリケーション機能を選択し、レプリケーションが同時に発生するように設定することによって、トランザクションの整合性を確認できます。（非同期レプリケーションが標準です。）レプリケーションには、**レプリケーション・カタログ**と呼ばれるレプリケーション自体のデータ・ディクショナリ表とビューのセットがあります。レプリケーション管理 API は、複数の PL/SQL パッケージで構成されており、レプリケーション・オブジェクトを管理するために使用することができます。

レプリケーションのタイプ

Oracle データベースは、3 つのタイプのレプリケーションをサポートしています。

- マルチマスター
- スナップショット
- ハイブリッド構成

マルチマスター・レプリケーション

マルチマスター・レプリケーション（Peer-to-Peer または N-Way レプリケーションとも呼ばれます）では、同等のピアとして機能する複数のサイトが、レプリケートされたデータベース・オブジェクトのグループを管理できます。マルチマスター・レプリケーション環境では、各サイトが**マスター・サイト**です。

アプリケーションは、マルチマスター構成内の任意のサイトにあるレプリケート表を更新できます。マルチマスター環境でマスター・サイトとして稼働している Oracle データベース・サーバーは、表レプリカのすべてのデータを自動的に集めることによって、グローバルなトランザクションの一貫性とデータの整合性を保証します。

スナップショット・レプリケーション

スナップショットには、ある時点における、ターゲット・マスター表の完全なまたは部分的なコピーが含まれています。スナップショットは、読取り専用または更新可能の場合があります。すべてのスナップショットには、次の利点があります。

- ローカル・アクセスを可能にし、応答時間と可用性を改善します。
- ローカル・スナップショットを問合せできるため、マスター・サイトからの問合せ負荷を軽減します。
- ターゲット・マスター表のデータ・セットのうち、選択したサブセットのみをレプリケートできるため、データ・セキュリティが向上します。

ハイブリッド構成

マルチマスター・レプリケーションとスナップショットをハイブリッド（複合）構成で組み合せると、様々なアプリケーションの要件を満たすことができます。**複合構成**では、任意の数のマスター・サイト、および各マスターに複数のスナップショット・サイトを指定できます。

たとえば、2 つのマスター間にあるマルチマスター（または N-Way）レプリケーションは、2つの地域をサポートするデータベース間にある全表レプリケーションをサポートできます。

マスターに対してスナップショットを定義して、表全体または表のサブセットを各地域内のサイトにレプリケートすることができます。

関連項目：『Oracle8*i* 概要』

Data Access Gateway

オラクル社は、Oracle 以外の製品およびテクノロジとの通信を容易にするために多くの製品を提供しています。これらの製品は、Data Access Gateway の下でグループ化され、Oracle Transparent Gateway、Oracle Procedural Gateway および Oracle Access Manager の 3 つの大きなカテゴリに分類されます。Oracle Transparent Gateway と Oracle Procedural Gateway では、Oracle 環境から Oracle 以外のテクノロジにアクセスできます。Oracle Access Manager では、Oracle 以外の環境から Oracle テクノロジにアクセスできます。

この項では、次の項目を説明します。

- Oracle Transparent Gateway
- Oracle Procedural Gateway
- Oracle Procedural Gateway for APPC
- Oracle Access Manager
- 相互運用性

Oracle Transparent Gateway

Oracle Transparent Gateway は、トランザクション SQL を使用して、DB2 や IMS、DB2/400 などの Oracle 以外のデータ・ストアに対するアクセスをアプリケーションに提供します。

Oracle Transparent Gateway には、次の機能があります。

- データが単一のローカル・リレーショナル・データベースに存在しているかのようなデータへのアクセス
- データへの動的な SQL アクセス
- 単一の SQL 文で、異なるソースの表を結合する機能

OTG は、DRDA と EDA/SQL 用の 2 つの主要な IBM 指向の Transparent Gateway、ODBC 用の Transparent Gateway、Sybase、Infomix、Ingres、Microsoft SQL Server、HP IMAGESQL、Digital RMS および Rdb 用の Transparent Gateway を提供しています。

Oracle Transparent Gateway for IBM DRDA は、IBM DRDA アーキテクチャの Oracle 実装版です。これにより、DB2 OS/390、SQL/DS、DB2/400 および DB2/UDB を含めた DRDA

サーバー・データベースへの読込みと書込みのアクセスが Oracle Applications に提供されます。

OS/390 上で実行される Oracle Transparent Gateway for EDA/SQL によって、VSAM、IMS、ISAM、Teradata、ADABAS および DB2 を含めた Oracle 以外の大半のデータベースとファイル・システムに対するアクセスが可能となります。

Oracle Procedural Gateway

Oracle Procedural Gateway は、Oracle トランザクションのコンテキスト内から Oracle 以外のトランザクションへのプログラム的なアクセスを提供します。これらのゲートウェイは IBM 指向で、Procedural Gateway for APPC と Procedural Gateway for MQSeries の 2 つのタイプに分類されます。

Procedural Gateway for MQSeries はメッセージ指向のため、第 7 章「アドバンスト・キューイング」で説明します。

Oracle Procedural Gateway for APPC

Oracle Procedural Gateway for APPC は、ネイティブな IBM APPC / LU6.2 メカニズムを使用し、IBM CICS や IBM IMS/TM、CA-IDMS/DC などのトランザクション・モニターを介してメインフレーム・トランザクションを実行します。

これらのトランザクションは、次のような各種データベースとファイル・システムにアクセスします。

- IMS
- DB2
- VSAM
- CA-IDMS/DB
- ADABAS
- Model 204

Oracle Procedural Gateway for APPC には、次の機能があります。

- Oracle と Oracle 以外のアプリケーション間で、トランザクションを共有できます。
- 単一の UNIX ベース・テクノロジで、複数のトランザクション・モニターと複数のオペレーティング・システムをサポートできます。
  - MVS: CICS、IMS/TM、CA-IDMS/DC および APPC/MVS
  - CICS/VM
  - DOS/VSE: CICS
  - AS/400: CICS/400

Oracle Access Manager

Oracle Transparent Gateway と Oracle Access Manager を組み合せて使用すると、メインフレーム・アプリケーションから任意のデータ・ストアに実質的にアクセスできます。Oracle Access Manager では、既存の OS/390 と AS/400 アプリケーションを使用します。これは、Oracle Server のデータへのアクセスを SQL を介してこれらのアプリケーションに提供することによって行われます。Access Manager for AS/400 によって、RPG、COBOL および C で記述された AS/400 アプリケーションは、Oracle に格納されているデータにアクセスできます。

Oracle Access Manager for CICS および Oracle Access Manager for IMS/TM は、Oracle for MVS Client Solution の一部で、TSO、バッチ、CICS および IMS/TM プログラムから Oracle のデータにアクセスできます。

- TSO セッションを開始して、エンタープライズの各所にある Oracle データベース・サーバーに直接対話形式でアクセスできます。
- バッチ・プロセスがリモートの Oracle Server を直接更新するため、複雑なファイル転送メカニズムの設定は不要です。
- CICS と IMS/TM トランザクションは、Oracle データベースとその他のリカバリ可能なリソースを更新します。トランザクションは、SYNCPOINT 処理を使用した 2 フェーズ・コミットで保護されています。

Oracle Replication と Oracle Gateway の使用

初期のアプリケーション統合方法は、**データ・レベルでの統合**によって、異なるアプリケーションにあるデータの複数コピーを保持し、それらのコピーを定期的に同期化する方法でした。統合に関する要件が基本的な場合、データ・レベルでの統合は、単純でコスト的にも有効で、管理が容易なソリューションを提供します。

データ・レベルでの統合を効率的に採用しているソリューションのすべてまたは大半には、次のような特徴があります。

- すべてのアプリケーションとデータは、単一の組織によって所有され管理されています。
- アプリケーション・インスタンスの数は、比較的小規模です（20 未満）。
- アプリケーションは、ほとんどリレーショナル・テクノロジに基づいています。
- アプリケーションは、ほとんど Oracle データベースに基づいています。
- データ構造（顧客 / アカウント / トランザクション表）は、すべてのアプリケーションで広範囲にわたって類似しています。
- 各データ項目は、1 つのアプリケーションのみで作成され、変更されます。
- データ変更アクセスに対する読込みアクセス率は、相対的に高い割合です。

- 拡張リアルタイムまたはリアルタイムに近いデータの同期化を必要としません。
- アーキテクチャと統合されるアプリケーションは、頻繁に変更されません。

この統合スタイルを使用する典型的なソリューションは、次のとおりです。

- 低い頻度でバックエンド・システムと同期可能なモバイル・アプリケーション
- 中間層にキャッシュされる読取り専用データが必要な Web ベースのアプリケーション
- OLTP アプリケーションへの分散が必要な、中央管理の値リスト参照データ

この統合レベルを提供するための最適な Oracle 製品は、Oracle Replication と Oracle Data Access Gateway です。

Oracle Replication では、**ジャーナル・ログ**を使用し、分散 Oracle データベース全体にわたってデータの同期を非同期化できます。これは、表構造のマスター・コピーに対する変更と低レベルでのデータ変更の両方を取得するために行われます。ログは、表のスナップショット・コピーに対する変更に適用されます。

Oracle Replication は、基本的な機能を次のように拡張できる高度な機能を数多く含んだ完成した製品です。

- 複数のマスター・コピーの管理と同期化
- データ変更の競合の解決
- トランザクションの一部としての、データ変更のリアルタイム・レプリケーションの提供
- レプリケート・ストアド・プロシージャの実行の同期化

Oracle Replication は、データ・レベルの統合ソリューションにおける、Oracle データベースに基づいたアプリケーション間の統合で中心的な存在です。

相互運用性

オラクル社は、Oracle が制御するトランザクションで、他のデータベースとの透過的な統合を可能にする透過的なゲートウェイを数多く提供しています。Oracle により管理されていないトランザクションの場合は、Oracle Access Manager を使用して、Oracle データベース内のデータが Oracle の制御外トランザクションによってアクセスされるようにします。

7

Oracle アドバンスト・キューイングと JMS

この章には次の項目が含まれています。

各製品の簡単な説明

この項では、Oracle Integration Server に関係する各種 Oracle 製品を説明します。次の項目が含まれています。

アドバンスト・キューイング

アドバンスト・キューイング（アドバンスト・キューイング : AQ）は、Oracle8*i* Enterprise Edition データベース管理システムのデータベース統合メッセージ・キューイング・コンポーネントで、アプリケーション内またはアプリケーション間でメッセージを送信するタスクを簡素化するためのインフラストラクチャを提供します。

機能的な特長は、次のとおりです。

- 蓄積転送機能
- 単純なメッセージ管理
- 障害時点までのリカバリ
- トランザクションの整合性
- データベース間における保証されたメッセージ送信
- メッセージ・マイニング
- メッセージの追跡
- メッセージをキューに配置したり、キューから削除するプログラミング・インタフェース（API）の選択

アドバンスト・キューイングのコンポーネント

アドバンスト・キューイングには、各種のキー・コンポーネントがあります。

メッセージ

メッセージは、キューに挿入したり、キューから取り出す情報の最小単位です。メッセージの構成は、次のとおりです。

- 制御情報（メタデータ）
- ペイロード（データ）

制御情報とは、AQ がメッセージの管理に使用するメッセージ・プロパティを示します。ペイロード・データとは、キューに格納されている情報で、Oracle AQ に対しては透過的な情報です。1 つのメッセージは、1 つのキューにのみ常駐できます。メッセージは**エンキュー・コール**で作成され、**デキュー・コール**で使用されます。

キュー

キューは、メッセージに対するリポジトリです。キューには 2 種類のタイプがあります。通常のキューとしても知られている**ユーザー・キュー**と**例外キュー**です。ユーザー・キューは通常のメッセージ処理用です。取り出して処理することが不可能なメッセージは、例外キューに転送されます。キューの作成、変更、開始、停止および削除には、Oracle AQ 管理インタフェースを使用します。

キュー表

キューはキュー表に格納されています。各**キュー表**はデータベース表で、1 つ以上のキューが含まれています。各キュー表には、デフォルトの例外キューが含まれています。

エージェント

エージェントはキューの使用者です。エージェントは、エンド・ユーザーまたはアプリケーションである可能性があり、2 種類のタイプがあります。

- キューにメッセージを配置する生成側の問合せサーバー（エンキュー）
- メッセージを取り出すコンシューマ（デキュー）

所定の時間にキューにアクセスできる生成側の問合せサーバーとコンシューマに数の制限はありません。エージェントは、Oracle AQ 操作インタフェースを使用して、キューにメッセージを挿入し、キューからメッセージを取り出します。

エージェントは、名前、アドレスおよびプロトコルによって識別されます。

- エージェントの名前は、アプリケーションの名前またはアプリケーションによって割り当てられた名前である場合があります。キュー自体がエージェントとなり、別のキューからエンキューまたはデキューする場合があります。

- アドレスフィールドは、1024 バイトまでの文字フィールドで、プロトコルのコンテキスト内で解析されます。たとえば、プロトコルのデフォルト値である 0（ゼロ）は、データベース・リンクのアドレスを示します。この場合、このプロトコルに対するアドレスは次の形式です。

```
queue_name@dblink
```

queue_name 部分は [schema.]queue のフォームで、dblink 部分には完全修飾されたデータベース名またはドメイン名なしのデータベース・リンク名のいずれかを指定できます。

受信者

メッセージの受信者は、その名前のみで指定できます。この場合、その受信者はメッセージがエンキューされたキューから、メッセージをデキューする必要があります。受信者は、名前とプロトコル値 0（ゼロ）のアドレスで指定できます。このアドレスは、同一のデータベース内、または（データベース・リンクが識別する）別の Oracle8*i* データベース内の別のキューの名前にする必要があります。別のデータベースの場合は、メッセージが指定のキューに伝播され、指定された名前でコンシューマがデキューできます。受信者の名前が NULL の場合、そのメッセージはアドレスで指定したキューに伝播され、アドレスで指定したキューのサブスクライバがデキューできます。プロトコル・フィールドが 0（ゼロ）以外の場合、名前とアドレスフィールドは、システムで解釈されず、メッセージは特殊なコンシューマがデキューできます。

受信者とサブスクリプション・リスト

1 つのメッセージを複数のコンシューマが使用するように設計できます。これには、次の 2 つの方法があります。

- エンキュー元は、メッセージを取り出せるコンシューマを、そのメッセージの受信者として明示的に指定できます。受信者とは、名前、アドレスおよびプロトコルで識別されるエージェントです。
- キュー管理者は、キューからメッセージを取り出せる受信者のデフォルト・リストを指定できます。デフォルト・リストに指定されている受信者は、**サブスクライバ**ともいわれます。受信者の指定がないままメッセージがエンキューされると、メッセージはそのキューのすべてのサブスクライバに暗黙的に送信されます。

  各キューは異なるサブスクライバを保有でき、同じ受信者を複数のキューに対するサブスクライバにすることができます。さらに、キュー内の特定メッセージをそのキューに対するサブスクライバでない可能性のある特定の受信者に送信できます。ここでは、サブスクライバ・リストの上書きが行われます。

ルール

ルールは、1 つまたは複数のサブスクライバがメッセージをサブスクライブする際の対象を定義するものです。この規格に合ったメッセージがサブスクライバに送られます。つまり、ルールは、キューのメッセージから、サブスクライバが必要とするトピックのメッセージをフィルタします。

ルールは、SQL 問合せの WHERE 句に類似した構文を使用し、ブール式（TRUE または FALSE のいずれかに評価する式）として指定されます。このブール式には、次の条件を含めることができます。

- メッセージ・プロパティ（現在は、優先順位と相関識別子）
- ユーザー・データのプロパティ（オブジェクト・ペイロードのみ）
- ファンクション（SQL 問合せの WHERE 句で指定）

ルールベースのサブスクライバ

ルールベースのサブスクライバには、デフォルトの受信者リストにそのサブスクライバに関連付けられたルールがあります。ルールベース・サブスクライバには、そのメッセージに対応するルールが TRUE と評価された場合、受信者が明示的に指定されていないメッセージが送信されます。

キュー・モニター

キュー・モニター（QMNn）は、キュー内のメッセージを監視するバックグラウンド・プロセスです。キュー・モニターは、メッセージの遅延、期限切れおよび再試行の遅延を管理するメカニズムを提供します。また、QMNn は、キュー表とその索引および索引構成表のガベージ・コレクションも行います。

最大 10 個のキュー・モニターを同時に開始できます。目的の数のキュー・モニターを開始するには、動的な init.ora パラメータ aq_tm_processes を設定します。たとえば、メッセージが期限切れまたは処理準備完了にマークされている場合、キュー・モニターは 1 分ごと、または処理する作業があると開始されます。

アドバンスト・キューイングの一般的な機能

AQ は、Oracle8*i* データベース管理システム（DBMS）の強みを利用して、次の一般的な機能を提供します。

SQL によるアクセス

メッセージはデータベース表の通常の行に配置されているため、標準 SQL を使用して問い合せることができます。このことは、SQL を使用することによって、メッセージのプロパティ、メッセージ履歴およびペイロードにアクセスできることを意味します。索引などの使用可能なすべての SQL テクノロジは、メッセージへのアクセスを最適化するために利用できます。

統合されたデータベース・レベルの操作サポート

リカバリ、再起動および Enterprise Manager などの標準データベース機能がサポートされています。Oracle AQ のキューはデータベース表内に実装されているため、高可用性、拡張性および信頼性などのすべての操作上の利点をキュー・データに適用することができます。さらに、データベースの開発と管理ツールをキューで使用できます。たとえば、キュー表をインポートおよびエクスポートできます。

構造化ペイロード

ユーザーは、オブジェクト型を使用してメッセージ・ペイロードを構成し、管理できます。一般的に RDBMS には、メッセージ・システムに比較してはるかに豊富な型指定システムがあります。Oracle8*i* はオブジェクト・リレーショナル DBMS であるため、従来のリレーショナル型とユーザー定義型の両方をサポートしています。外部の型指定システムによって定義された形式のコンテンツなど、これまでに厳密に型指定された結果、多数の強力な機能が使用可能になりました。次の機能があります。

- **コンテンツ・ベースのルーティング :** 外部エージェントは、コンテンツを検査し、そのコンテンツに基づいてメッセージを別のキューにルート変更できます。
- **コンテンツ・ベースのサブスクリプション :** パブリッシュ・サブスクライブ・システムは、メッセージ・システムの上部に形成され、サブスクリプションに基づいて内容を提供します。
- **問合せ :** メッセージ・コンテンツの問合せを実行する機能で、メッセージ・ウェアハウスを使用可能にします。

保存とメッセージの履歴

AQ のユーザーは、使用後にメッセージを保存するように指定できます。システム管理者は、メッセージの保存期間を指定できます。Oracle AQ は、各メッセージの履歴に関する情報を保存し、ローカルまたはリモートの受信者宛てメッセージに関する遅延、失効および保存についてのキュー・プロパティとメッセージ・プロパティを保存します。この情報には、`ENQUEUE/DEQUEUE` の時間と各要求を実行したトランザクションの内容が記録されています。これにより、ユーザーは関連するメッセージの履歴を保持できます。履歴は、追跡、データ・ウェアハウスおよびデータのマイニング操作に使用できます。

追跡とイベント・ジャーナル

保存されているメッセージは、相互に関連付けることができます。たとえば、メッセージ `m1` の使用結果としてメッセージ `m2` が生成された場合、`m1` は `m2` に関連付けられます。これによって、ユーザーは関連付けられた一連のメッセージを追跡できます。これらの一連のメッセージが、アプリケーションによって組み立てられることの多いイベント・ジャーナルとなります。Oracle AQ は、アプリケーションによってイベント・ジャーナルが自動的に作成されるように設計されています。

統合化トランザクション

制御情報とコンテンツ（データ・ペイロード）の統合は、アプリケーションの開発と管理を簡素化します。

キュー・レベルのアクセス制御

Oracle8*i* では、8.1 形式のキューの所有者が、キューに対するキュー・レベルの権限を付与したり、取り消すことができます。DBA は、新しい AQ システム・レベルの権限を任意のデータベース・ユーザーに対して付与したり、取り消すことができます。また、DBA は任意のデータベース・ユーザーを AQ 管理者にすることもできます。

非永続キュー

AQ は、非永続メッセージをサブスクライバに非同期で送信できます。これらのメッセージはイベント・ドリブン方式で、システム（またはインスタンス）障害の発生後は存続しません。AQ は、共通 API を使用して、永続メッセージと非永続メッセージをサポートします。

パブリッシュ・サブスクライブのサポート

機能の組合せによって、アプリケーション間でのパブリッシュ・サブスクライブ・スタイルのメッセージ処理が可能になります。これらの機能には、ルールベース・サブスクライバ、メッセージ伝播、リスニング機能および通知機能があります。

キュー操作開発のための 2 つのコンテキスト

Oracle AQ は、2 つの開発コンテキストを提供しています。

- 3 種類の異なるプログラム環境からアクセスできるネイティブ AQ
  - `DBMS_AQ/AQADM` PL/SQL パッケージを使用した PL/SQL からのアクセス
  - Oracle Objects for OLE（OO4O）を使用した Visual Basic からのアクセス
  - `oracle.AQ` Java パッケージを使用した Java からのアクセス
- Java メッセージ・サービス（JMS）

  `oracle.jms` Java パッケージを使用します。この公開規格の実装によって、定義済みの W3C インタフェースが拡張されるため、JMS コンテキストで操作を行う開発者は、ネイティブ AQ 内部で作業する場合と同じ方法を使用できます。

包括的なグラフィカル・インタフェースは、データベース・リリース 8.1.7 にある DBA 管理パックの一部である Enterprise Manager DBA Studio（スキーマ管理）を使用して、アドバンスト・キューイング・オブジェクトの管理をサポートします。

関連項目：

- アドバンスト・キューイング機能の詳細とその使用方法の例は、『Oracle8*i* アプリケーション開発者ガイド アドバンスト・キューイング』を参照してください。
- 『Oracle8*i* PL/SQL パッケージ・プロシージャ リファレンス』
- 『Oracle8*i* Java パッケージ・プロシージャ リファレンス』
- Oracle Objects for OLE のオンライン・ヘルプ

Oracle Java Messaging Service（OJMS）

オラクル社は、標準 JMS API ではアクセスできない Oracle アドバンスト・キューイングの機能を活用するために、JMS API を拡張しています。また、この拡張部分を JMS 標準に組み込むように、標準化機構に働きかけています。

標準 JMS API から拡張した部分は、次のとおりです。

エージェント

標準 JMS の定義では、各メッセージが単一の受信者に送信されることを前提としています。AQ は、複数の受信者へのメッセージ送信をメッセージのコピーなしに、記録、管理および追跡できるため、OJMS API では**複数エージェント**の使用をサポートしています。

エージェントは、メッセージの受信者ごとにメッセージに関連付けられています。このエージェントは、名前、トランスポート・プロトコルおよび転送プロトコルに固有のアドレスを持つオブジェクトです。

エージェント・モデルは、AQ のパブリッシュ・サブスクライブ機能および AQ とその他のメッセージ処理テクノロジとの相互運用では特に重要です。

追加されたメッセージ管理プロパティ

標準 JMS メッセージ・プロパティ定義ではサポートされていない標準的な AQ 機能を使用するために、OJMS のメッセージ管理プロパティ・セットが拡張され、次の機能が組み込まれました。

- **遅延間隔**：メッセージの作成者は、指定した時間（遅延）が経過するまで、メッセージを使用不可に指定できます。この機能を標準 JMS の期限切れ機能とともに使用すると、メッセージの**使用期間**（全受信者が正常に収集できる期間）を定義できます。この機能は、記録が必要なイベントには特に役立ちます。
- **例外処理**：AQ は、期限切れまたは指定した試行回数内で使用できないメッセージを自動的に例外キューに移動します。この例外処理プロパティによって例外キューを指定できます。この機能は標準 JMS の期限切れ機能の拡張です。

追加されたメッセージ型

AQ は、厳密な型指定のメッセージ・ペイロードをサポートするメッセージ処理テクノロジの中でも独自なもので、メッセージ・ペイロードがキューにある場合にも標準 SQL を使用してアクセスでき、コンテンツ・ベースのルーティングをサポートします。これを実現するために、AQ では Oracle オブジェクト型（ADT としても知られています）を使用します。この機能をサポートするために、OJMS には ADTMessage と呼ばれるメッセージ型が追加されています。

トランザクション的なセッション

AQ メッセージは、Oracle データベースの `COMMIT` トランザクションのコンテキスト内で作成できます。同じトランザクションに、表の挿入や `SAVEPOINT` などの SQL 操作を含めることができます。このために、2 フェーズ・コミットまたは XA 準拠のコミットは不要です。

JMS のトランザクション化されたセッション機能は、この Oracle 独自の機能をサポートするために拡張され、Oracle ベースのアプリケーションと対話するアダプタの開発に役立っています。

管理

JMS 標準には**管理オブジェクト**の概念があります。第 4 章「基本的な統合の概念」には、Oracle JMS の実装が詳述されています。Oracle JMS の実装には、これらのオブジェクトを作成および管理するための API があります。

セッション・インタフェースは、キュー表およびキューとトピックの作成、およびそのプロパティの変更など、管理機能をサポートするようになりました。

接続先インタフェースは、キューとトピックおよび接続先との間のメッセージの AQ 伝播の管理をサポートしています。これは、ハードウェア構成、セキュリティ・モデルまたは永続スキーマに使用されるハブ・アンド・スポーク・アーキテクチャにとって重要です。

制限事項

OJMS インタフェースの最初のリリース（データベース・リリース 8.1.6）には、標準外の方法でサポートされている、またはされていない JMS の機能が少数あります。

- 一時キュー
- 非永続サブスクライバ
- JMS キューに対するメッセージ・フィルタ処理（AQ シングル・コンシューマ・キュー）。ただし、OJMS の永続サブスクライバのフィルタ処理に対するサポートは、標準 JMS の定義よりも実質的に豊富です。
- JNDI を使用した管理オブジェクトへのアクセス。管理インタフェースに対する OJMS の拡張版では、この機能を提供しています。

Oracle Procedural Gateway for IBM MQSeries

Oracle Procedural Gateway for IBM MQSeries によって、Oracle Applications で MQSeries メッセージを直接作成したり、AQ メッセージを MQSeries メッセージに（またはその逆に）変換することができます。また、MQSeries 構造体に Oracle オブジェクト型のマッピングを定義したり、Oracle オブジェクト型および COBOL Copybook 定義をインポートするために使用できる、ビジュアル・ワークベンチと呼ばれるグラフィカル・インタフェースがあります。

SQL*Plus、PL/SQL パッケージ、Pro*Series 言語などからアクセス可能な MQSeries キューに対して透過的なトランザクション API を提供する PL/SQL パッケージを生成して、構成を完成します。

Oracle 用の TIB アダプタ

TIBCO 社は、AQ メッセージを TIB/Rendezvous メッセージに（またはその反対に）変換するためのアダプタを販売しています。この製品は、Oracle と共同で開発され、Oracle アドバンスト・キューイングと TIB/Rendezvous（TIBrv）との間のコード不要な変換ソリューションを提供します。

このソフトウェアは、メッセージを AQ から TIBrv に変換してから、TIB バスを介して移送し、AQ メッセージ・プロパティの損失なしに、再び AQ に変換し直すように設計されています。メッセージは、TIB/Rendezvous Certified Messaging と Oracle の同期コミット・モデルのバージョンを使用してトランザクションごとに変換されます。

統合ソリューションへの製品の適用

この項では、次の製品を統合ソリューションにインタフェースする方法を検討します。製品は、次のとおりです。

- アドバンスト・キューイング
- OJMS
- Procedural Gateway for MQSeries
- 相互運用性

アドバンスト・キューイング

ビジネス・イベントの統合

ビジネス・イベントに基づく統合ソリューションにおいて、AQ のメッセージ・キューイング機能はソリューションに対するバックボーンを提供し、以前のメッセージ処理テクノロジでは不可能な、次のビジネス・イベント要件をサポートします。

- 履歴の保持による受信者のメッセージ使用の追跡
- システム障害の際にリカバリ可能なメッセージとその履歴の永続的な記録
- 証明監査証跡を形成するための使用後メッセージと履歴の保存
- メッセージ・マイニング用メッセージへの単純問合せアクセス権の提供
- メッセージが保存される場合にも永続的に実行するアーキテクチャの提供
- 高度なルーティング・ルールの定義と適用

アドバンスト・キューイングは、Oracle8*i* データベースを使用してビジネス・イベントをサポートし、メッセージ、トランザクションのサポートやリカバリ用のリポジトリを提供します。Oracle Integration Server の大半の製品は、提供されている AQ インタフェースを使用して直接アクセスできます。これにより、ソリューションのバックボーンとしての AQ の位置付けが強化されます。

AQ は、メッセージの作成時に、メッセージの受信者を決定および記録して履歴を保持します。受信者がメッセージを収集しようとするたびに、その試みが自動的に記録されます。これにより、各受信者によってメッセージが 1 回のみ収集されること、およびメッセージの収集時点を常に把握することが保証されます。

AQ メッセージには、ヘッダー、ペイロードおよび履歴の 3 つの主要なコンポーネントがあります。3 つのコンポーネントはすべてデータベース内の一連の一般的なデータベース・オブジェクトで構成されるキュー表に格納されています。データベースが適切なバックアップとリカバリ・ルーチンで保護されている場合、メッセージはシステム障害の際に矛盾のない状態にまでリカバリできます。

AQ メッセージは、メッセージをアクセスしているユーザーによってメッセージ・ヘッダー、ペイロードまたは履歴が直接変更されない API を使用してアクセスされます。この設計機能と使用済メッセージの保存機能が組み合せると、監査証跡を作成できます。

AQ はこの強力な機能を実装するために、高度なデータベース・オブジェクトのセットを使用しますが、このセットには各キュー表に対するビューのセットも含まれているため、メッセージのヘッダー、ペイロードおよび履歴を SQL を使用して簡単に問い合せることができます。

ペイロードの構造が単純なオブジェクト型で定義されている場合は、SQL で属性に直接アクセスできます。ただし、CLOB を含む汎用的なオブジェクト型を使用してペイロードを XML フォーマットなどで格納する場合は、問い合せる前に、文字列をより厳密な型指定の構造に変換する必要があります。

前述のように、AQ は一連の一般的なデータベース・オブジェクトを使用して、キュー表の論理的な構造体を実装します。このオブジェクト・セットは、保存されている使用済みメッセージが他のメッセージの作成またはコレクションに悪影響を与えないように開発されました。

事前定義ルールに従ってメッセージの受信者を判断する機能は、アドバンスト・キューイングで包括的にサポートされています。

関連項目： 第 4 章「基本的な統合の概念」

マルチ・コンシューマ・キューは、永続サブスクライバにパブリッシュ・サブスクライブ・ルーティングを提供します。

- サブジェクト・ベースのルーティング（第 4 章「基本的な統合の概念」）は、各サブジェクトに対するキューと、そのキューに対する受信者のサブスクリプション・リストを定義することによって提供されます。
- コンテンツ・ベースのルーティング（第 4 章「基本的な統合の概念」）は、サブスクリプション・リストのサブスクリプション・ルールの使用によって、オブジェクト型ペイロードでキューに提供されます。サブスクリプション・ルールには、定数値 `tab.user_data` を使用して属性名に接頭辞を付け、ペイロードへの参照を含めることができます。
- ルールベースのルーティング（第 4 章「基本的な統合の概念」）は、ルールに副問合せを指定して実現できます。

伝播を使用して、次のシナリオのようにパブリッシュ・サブスクライブ・コンテキスト内の受信者のグループに、メッセージをルート変更することもできます。

1. 受信者は、キュー `RedCars` にサブスクライブして、赤い車に関するメッセージを受信します。
2. キュー `RedCars` は、`tab.user_data.color = RED` のルールでキュー `SportsCars` にサブスクライブします。
3. パブリッシャは、キュー `SportsCars` にメッセージを作成します。
4. 色を `RED` に設定する属性が車にある場合、キュー `RedCars` はメッセージの受信者として記録されます。
5. AQ は、キュー `RedCars` にメッセージを伝播し、サブスクリプション・リストの全受信者を伝播するメッセージの受信者として記録します。

サブスクライバに永続性であっても、変更が頻繁（毎日またはそれ以上に頻繁）なときは、アドバンスト・キューイングはルーティングに対する最も適切なメカニズムです。

サブスクリプションが静的で、ルーティングのルールが複雑な場合は、ワークフロー（第 10 章「Oracle Workflow」）がソリューションの開発にとって効率のよい選択肢を提供します。

データの統合

Oracle Replication に必要な機能がない場合は、アドバンスト・キューイングのテクノロジを使用して、事前定義済みレプリケーション環境を開発します。

2 種類の方法で実行できます。

- データベース表の行、または接続先データベースにメッセージを転送するための伝播を変更するときに、データベース・トリガーを定義して AQ メッセージを作成します。接

続先データベースのキューにメッセージがあるときは、そのメッセージを削除して、表のコピーに適用できます。

- 非永続キューを使用して、DML と DDL 変更をデータベース・イベントとして獲得します。この場合、メッセージは OCI を使用して処理する必要があります。

AQ を使用したユーザー独自のレプリケーション・ソフトウェアの開発には、レプリケーションの問題に関する特別な理解、および事前定義済みソフトウェアのメンテナンスとアップグレードに対するコミットメントが必要です。

OJMS

ソリューションのプログラム可能な部分に開発言語として Java を使用する場合、およびメッセージ処理テクノロジを使用して通信するための標準として JMS を使用する場合、OJMS は、アドバンスト・キューイングに対する効果的な API を提供します。

B2B 指向のソリューションでは、オブジェクト型ペイロード、トランザクション化されたセッションおよびエージェント・モデルなどの JMS 標準に含まれている機能に加えて、AQ 機能を使用できます。JMS 標準に対する OJMS 拡張機能によって、これらの特別な AQ の機能が使用できるようになります。これらの機能を使用すると、OJMS は Oracle Message Broker の JMS 実装に対する優れた代替を提供します。

Procedural Gateway for MQSeries

Procedural Gateway for MQSeries は、Oracle が提供しているその他のメッセージ処理テクノロジへの唯一のゲートウェイです。

これは MQSeries とアドバンスト・キューイングの間でメッセージを送信するために使用します。現在のフォームでは、MQSeries デキュー・パッケージまたは AQ エンキュー API をコールするために、簡単な PL/SQL ブリッジ・プロシージャを開発する必要があります。このゲートウェイは、単純なペイロードの変換のみをサポートし、コレクションまたは LOB はサポートしません。

単一のキュー・ブリッジ全体のメッセージ・スループットは、システムの仕様とメッセージ・サイズによって変化します。毎秒数百件のメッセージが予想される場合は、Procedural Gateway for MQSeries を選択しないでください。

相互運用性

Oracle Integration Server を構成する製品は、アドバンスト・キューイングと円滑に対話し、多くのサード・パーティのアプリケーション統合ツールが提供するサービスに匹敵する非同期サービスを提供します。

ただし、一部のアプリケーションでは、MQSeries や TIBCO 社の Rendezvous などの他のメッセージ処理テクノロジとの相互運用が必要です。要件の分離、アプリケーション・プラットフォーム、既存のインタフェースまたは以前に選択された戦略的テクノロジが必要な理由です。

アプリケーションに、AQ 以外のメッセージ処理テクノロジにメッセージを抽出するインタフェースがある場合は、AQ、OMB またはワークフローにメッセージ処理テクノロジを直接統合するために、プログラミング言語を使用するか、または専用ソフトウェア使用するかを決定する必要があります。

アドバンスト・キューイングに統合する代替方法は、次のとおりです。

プログラム 1 : 両方のメッセージ・キューイング製品に特定言語に対する API がある場合は、一方の製品からメッセージを収集するプログラムを記述して、そのメッセージ・プロパティを転送し、もう一方の製品でメッセージを作成します。

プログラム 2 : 相互に通信でき、メッセージ・キューイング製品への API がある 2 種類の言語を選択します。

ソフトウェア : 2 つのメッセージ処理製品間で変換可能な専用の製品を選択します。

MQSeries の例

ここに示す例は、処理内容を示すためのもので、ソリューションの 1 つとして推奨するものではありません。

シナリオ

ハブ・アンド・スポーク型統合ソリューションに組み込まれているアプリケーションの 1 つには、既成のメッセージ・ベースのインタフェースがあります。このインタフェースは、MQSeries キューのセットにメッセージを配置し、MQSeries メッセージのフォームでビジネス・イベントを予定しています。アプリケーション・インタフェースは C 言語で記述され、システム固有の MQSeries API を使用してメッセージを作成および使用します。

目的は、MQSeries から AQ に、および AQ から MQSeries にメッセージを転送することにあります。

ソリューションは、アーキテクチャ原理、言語、統合ソリューションで使用されている言語、モジュール化へのアプローチ、使用している規格などによって異なります。

第 2 のメッセージ・キューイング製品が、MQSeries ではなく TIBCO 社の TIB/Rendezvous の場合は、Oracle ではなく TIBCO 社がソフトウェア・プロバイダで、ソフトウェアが TIB Adapter for Oracle と呼ばれることを除き、シナリオは同じです。

ビジネス・インテリジェンスとメッセージ・ウェアハウス

この章で前に説明しように、アドバンスト・キューイングは、永続キューと非永続（揮発性）キューの 2 種類のキューを管理できます。

この項では、次の項目を説明します。

- 永続キュー
- 揮発性キュー
- メッセージ格納の基本原理
- Oracle Reports
- Oracle Discoverer
- Oracle Express

永続キュー

永続キューは、アプリケーション間（エンタープライズでアプリケーションを統合する場合）または 2 つのビジネス・プロセス間（B2B 取引用にビジネスを統合する場合）で受け渡しされるメッセージの管理に最適です。次の 2 つの理由から、これらのメッセージは永続的に格納しておく必要があります。

- **監査と追跡 :** あるアプリケーションまたはビジネスから別のアプリケーションまたはビジネスに送信されるメッセージは、論争解決のための監査と追跡ができるように保存しておく必要があります。一般的に、メッセージの監査には 3 種類のキューが実行されます。
  - どの情報をどこに送信するかを決定するためのメッセージ宛先に対するキュー
  - 送信する情報のタイプを決定するためのメッセージ・ヘッダー（サブジェクトまたはトピックとも呼ばれます）に対するキュー
  - 実際に送信する情報を決定するためのメッセージ・ペイロードまたはコンテンツに対するキュー

  特に B2B 取引の場合、メッセージの監査と追跡は重要な要件です。
- **順序に従った 1 回限りのメッセージ送信の保証 :** B2B と EAI メッセージ機能に関する重要な要件の 1 つは、順序に従った 1 回限りのメッセージ送信を保証するメッセージ・インフラストラクチャです。

AQ のメッセージ・ペイロードは、SAP TDOC や XML ベースのビジネス・オブジェクト・ドキュメントなどのフラット・ファイル指向のデータに対する RAW または BLOB などの非構造化データ型にすることができます。メッセージ・ペイロードのウェアハウス処理時には、3 つの問題を考慮します。

- **メッセージ・ヘッダーの格納 :** リレーショナル表またはオブジェクト・リレーショナル表の AQ ペイロードからメッセージ・ヘッダーを格納できます。メッセージを XML

フォーマットで受信する場合は、メッセージを解析して、そのヘッダー情報を SQL フォーマットに変換し、表に格納することもできます。解析と格納を行うかどうか、その手順をどのように行うかを決定する場合は、2 つのトレードオフを考慮してください。

- メッセージ・ヘッダーの SQL フォーマットへの変換では、分析のためのメッセージ問合せが頻繁に発生します。
- 未変換のヘッダーとペイロードの格納では、頻繁に監査が発生します。

- メッセージ・ペイロードの格納 : 構造化されたメッセージ・ペイロードはリレーショナル表またはオブジェクト・リレーショナル表として格納し、標準 SQL を使用して問合せできます。構造化されていないペイロードは、RAW または BLOB フォーマットのいずれかで格納できます。B2B メッセージを XML ペイロードとして受信する場合は、そのペイロードを完全に解析する方法を決定する必要があります。分析操作を頻繁に実行する必要がある場合は、ペイロードの解析が必要です。ペイロードが大量の場合は DOM 解析ツリーを作成し、小量の場合は SAX API を使用します。（通常、100MB を超えるペイロードは大量と考えられます。）XML ドキュメント内の小さなフィールドのセットに対してのみ分析が必要な場合は、解析ツリーからそのフィールドのみを抽出し、SQL フォーマットで表に格納します。他のペイロードは RAW または BLOB フォーマットで保存します。

  Oracle XML 機能の詳細は、『Oracle8*i* XML リファレンス・ガイド』を参照してください。

- **パーティション・ウィンドウとローリング・ウィンドウの利用 :** メッセージ・ウェアハウスはデータ・ウェアハウスと類似しているため、データ・ウェアハウスに適用可能な機能は、メッセージ・ウェアハウスにも適用できます。B2B 取引環境において最も重要な点は、毎日受信するメッセージを個別のデータベース・パーティションに格納および管理できることです。パーティション化には、次の利点があります。
  - 索引に加え、集計や移動平均などの分析計算をパーティション・レベルで作成および実行できます。たとえば、B2B メッセージの合計量を毎日分析するには、その日のパーティション内のデータに対して、単に SORT と SUM 操作を実行します。
  - 日次ベースでメッセージをパーティション化すると、個々のメッセージをオンラインとオフラインで移動できます。オフラインで新規パーティションを追加したり、既存のパーティションを取得するローリング・ウィンドウを保持します。

メッセージ交換インフラストラクチャは、メッセージの永続的な格納を保持して、メッセージ伝播またはトランスポートの障害時に送信メッセージを保証する必要があります。

揮発性キュー

永続的なメッセージ処理環境とは反対に、揮発性キューはメッセージ送信に最も効果的です。たとえば、B2B メッセージの受信を確認するために送信したメッセージは、永続的に保存する必要はありません。

メッセージ格納の基本原理

前述のように、AQ を使用して渡された各メッセージはキューに格納されます。単一または複数のキューのコンテンツはキュー表に格納できます。ビジネス分析とビジネス・インテリジェンスに関するメッセージは、4 つの基本原理に従って格納されます。

- メッセージ・ペイロード構造
- メッセージ・ロギング機能
- メッセージ・ウェアハウス・アーキテクチャ
- 問合せ、分析およびレポートの各機能

ビジネス・インテリジェンス・ツール

Oracle Business Intelligence Tools は、Oracle データベース内のビジネス情報に対する簡単で直感的なアクセスを提供する 3 つの Oracle 製品の集合です。これらの製品を使用して、メッセージ・ウェアハウスに格納されているビジネス・インテリジェンスを分析します。

Oracle Reports、Oracle Discoverer および Oracle Express には、標準的なインターネット・ブラウザを使用して、どこからでもアクセスできるグラフィカル・インタフェースがあります。そして、意思決定支援情報をビジネス・ユーザーに（通常はデータ・ウェアハウスまたはデータ・マートから）提供します。最近、非同期メッセージ処理統合ソリューションで、スループット分析とサービス・レベル監視を行う操作監視ができるように拡張されました。

これらのツールは、動的に作成されたレポートの生成と表示、仮想と具体化されたデータ構造に対する動的な問合せの実行、および傾向の分析と予測に使用できます。

Oracle Reports

Oracle Reports は、高度で高品質なレポート開発者のためのグラフィカルなツールです。豊富な機能によって、SQL の全機能を使用した、複雑な階層構造やマトリックスを持つレポートを素早く簡単に定義できます。

ユーザーは、このような動的に作成されたレポートを標準的な Web ブラウザで表示できます。

Oracle Discoverer

Oracle Discoverer は、主にビジネス・ユーザー向けの使いやすいブラウザ・ベースのツールで、動的な問合せを実行してレポートとチャートを提供し、Web 公開を可能にします。

データベース・オブジェクト全体の抽象化レイヤーであるエンド・ユーザー・レイヤーを採用して、問合せを簡素化し、問合せの事前定義を可能にしています。また、ドラッグ・アンド・ドロップ機能によって、ユーザーが SQL の構文を理解するという手間を省きます。

統合された強力なチャート機能は、傾向と例外を明示し、チャートを使用して特定データを詳細表示するドリル・ダウンを可能にしています。

さらに、次のような高度な機能があります。

- **高度な問合せ予測 :** 発行する問合せの概略の実行時間が表示されます。
- **インテリジェント・サマリー・リダイレクション :** このツールは、実行時間を削減するために、事前作成サマリー表を使用して問合せを自動的に最適化します。
- **Oracle Designer の生成 :** エンド・ユーザー・レイヤーは、Oracle Designer を使用して保持およびバージョン管理できます。

Oracle Express

Oracle Express は、オンライン分析処理（OLAP）エンジンを使用した意思決定支援ツールです。行と列をディメンションと呼ばれるデータの複数のカテゴリに拡張するマルチディメンション・データ・モデルを採用しています。

このタイプのデータ・モデルは、意思決定を支援するために最適化されています。また、配列算術を使用して、重複型の問合せでデータに迅速にアクセスします。さらに、分析、予測、モデル化および what-if シナリオ展開のための組込みファンクションを提供しています。

標準的な Web ブラウザでアクセスできる直感的なツールを使用して、データ・モデルを統合できます。OLAP エンジンでは、問合せパフォーマンスを支援するために、ディメンション・データの格納にデータ・キャッシュを使用できます。

8

Oracle Message Broker と JMS

この章では、Oracle Message Broker を説明します。次の項目が含まれています。

- 概要
- OJMS と OMB の使用
- 使用可能なツール

概要

Oracle Message Broker（OMB）は、Java ベースのメッセージ管理サブシステムで、AQ、IBM MQSeries および TIBCO 社の Rendezvous などの主要なメッセージ・キューイング・システムに対してメッセージ・ブローカの機能を提供します。Oracle Message Broker のドライバは、一貫したオープンな JMS 準拠の API をこれらのメッセージ・キューイング・システムに提供します。

OMB は、それ自体の揮発性キューイングと伝播を提供しています。OMB の管理メタデータはローカルまたはリモートの LDAP サーバーに保存できるため、OMB は Oracle データベース・インスタンスから独立して動作します。OMB は、JMS の標準パブリッシュ・サブスクライブやトピック・ベース・ルーティングをサポートし、永続と非永続サブスクライバの両方をサポートしています。メッセージ管理プロパティにフィルタを適用することもできます。OMB は、JMS のトランザクション化されたセッションをサポートしています。

Oracle Message Broker は、次のコンポーネントで構成されています。

- Oracle Message Broker Core
- ドライバ
- 管理コンポーネントと LDAP ディレクトリ
- クライアント・プログラミング・インタフェース
- アダプタ開発者用ツールキット

Oracle Message Broker Core

Oracle Message Broker Core は、**JMS プロバイダ**として動作します。JMS 標準の定義は、第 4 章「基本的な統合の概念」を参照してください。

異なる場所にある複数の Oracle Message Broker は互いに通信し、ブローカ間のメッセージ送信を調整します。

Oracle Message Broker Core は、次の機能を提供します。

- **JMS クライアント**へのメッセージ送信のサポート。第 4 章「基本的な統合の概念」を参照してください。
- メッセージ通知をサポートしていないメッセージ・キューイング・システムのポーリング。
- コアベースの管理タスク。
- 基礎となるメッセージ・キューイング・システムへの JMS クライアント接続の多重化。
- メッセージ・キューイング・システム API 全体に対する JMS ベースの抽象化レイヤーを提供する標準ドライバ。

ドライバ

Oracle Message Broker には、基礎となるメッセージ・キューイング・システムに JMS API を提供する 5 種類のドライバがあります。Oracle Message Broker メッセージ・キューイング・システムを使用して、メッセージの永続性とメッセージの管理を提供します。

ドライバは、メッセージをメッセージ・キューイング・システムがメッセージを格納するために必要なシステム固有の格納形式に変換する際の調整も行います。

ドライバには、次の種類があります。

- **Oracle Advanced Queuing ドライバ**は、OJMS API に代わる、Oracle8*i* アドバンスト・キューイングのドライバです。
- **Oracle Volatile ドライバ**は、軽量なメモリー内の通信機能を使用して JMS メッセージを高速で送信します。この Volatile ドライバは、メッセージ・システムがメッセージを永続的に格納する必要がない場合、メッセージのスループット向上に貢献します。
- **MQSeries ドライバ**は、商用メッセージ・システムで幅広く使用されている機能の大半をサポートしています。
- **Oracle Multicast ドライバ**は、軽量なマルチキャスト通信機能を使用して JMS メッセージを高速で送信します。Oracle Multicast ドライバは、Oracle Application Server のマルチキャスト通信ライブラリを使用します。
- **TIBCO ドライバ**は、軽量なマルチキャスト通信機能に基づいて、一時メッセージを高速で送信します。TIB/Rendezvous（TIBCO 社）ドライバは、TIB/Rendezvous リリース 5.x または TIB/Rendezvous Pro リリース 5.x で動作するように記述されています。

管理コンポーネントと LDAP ディレクトリ

Oracle Message Broker は LDAP ディレクトリを使用して、管理情報を格納およびアクセスします。Oracle Message Broker には、管理ユーティリティや監視ユーティリティを提供するコマンドライン・ツールおよびグラフィック・ツールが含まれています。また、Oracle Object Request Broker のダイナミック・モニタリング・サービス（Dynamic Monitoring Service: DMS）に基づくパフォーマンス監視ツールも提供しています。

クライアント・プログラミング・インタフェース

クライアント・プログラミング・インタフェースは、Oracle Message Broker が Java クライアント・プログラムに提供する一連のサービスで、JMS API とも呼ばれています。一連の C++ クラス・ライブラリは、類似したプログラム・インタフェースを C++ プログラマに提供しています。

アダプタ開発者用ツールキット

オラクル社は、アダプタの開発に使用できる Java ベースの開発者用ツールキットを Oracle Message Broker バージョン 2 の一部として提供しています。このキットは、標準的な構成のプログラムを開発でき、アプリケーションにアクセスする次の 3 種類の方法をサポートするフレームワークを提供します。

- アプリケーション API を使用してアクセスします。
- SQL を使用してアクセスします。
- インタフェース・ファイルからのコレクションによってアクセスします。

このツールキットによって、JMS のキュー / トピックを介して、アプリケーションやキューイング・テクノロジをリンクすることができ、トランザクション管理と基本的な例外管理が提供されます。開発者は、必要なアプリケーションへのアクセス方法に応じて、アプリケーション API 固有のコール、ファイル・ベースのアクセスまたは JDBC 接続を追加して、フレームワークを拡張します。

SQL アクセスのフレームワークは、JMS のキュー / トピックからデキューされた XML フォーマットのメッセージを、Java のビジネス・コンポーネント（BC4J）に変換します。BC4J オブジェクト内に定義されている SQL は、変更内容をアプリケーションに JDBC 接続経由で適用します。

OJMS と OMB の使用

オラクル社は、Oracle Java Messaging Services と Oracle Message Broker の 2 つの製品を介して Java メッセージ・サービス（JMS）を提供しています。特定の要件との関係で、製品を最適に使用するには、2 つの製品の相違を理解する必要があります。

この項では、次の項目を説明します。

OJMS は、AQ の拡張機能を使用して JMS の AQ 部分のみの実装を提供しています。したがって、データベースと緊密に連動しています。

OMB は、AQ および OMB 固有のキューイング・メカニズムを含めた主要なメッセージ処理テクノロジ全体に、未拡張の Java Messaging Service を提供します。OMB は、最小の領域を使用し、そのメタデータには LDAP サービスを使用するように設計されています。OMB 固有のキューイング・メカニズム（揮発性キューと呼ばれます）によって、OMB は非永続サブスクリプションを JMS トピックに提供できます。

OMB が AQ をメッセージ・ストアとして使用できるようにする AQ ドライバには、JMS 標準にはない AQ 機能を使用するための拡張機能は含まれていません。ただし、OMB は、JMS

トピックを使用したパブリッシュ・サブスクライブのルーティングを提供しており、メッセージのコレクション時点でメッセージにフィルタを適用します。OJMS ドライバは、メッセージのフィルタ機能を提供していません。

次の要因が、機能上またはアーキテクチャ上重要である場合は、Oracle Message Broker の使用を考慮してください。

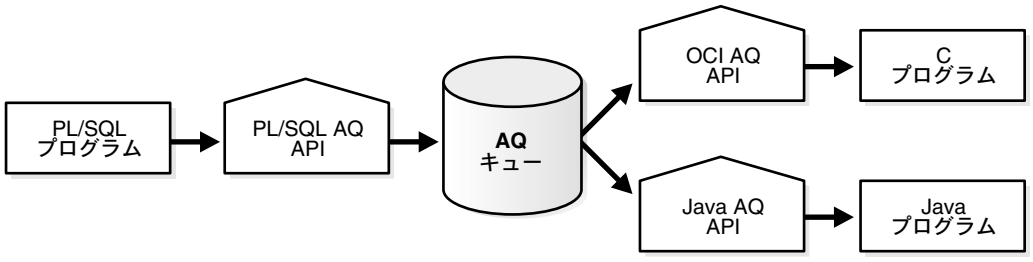
- JMS サービスを、Oracle8*i* Enterprise Edition リリース 8.1.6 データベースがインストールされていないコンピュータに配布する場合
- 非永続サブスクライバが JMS トピックをサブスクライブする場合
- 特定のプラットフォーム上に基礎となるメッセージを格納するときは、MQSeries などの別のメッセージ処理テクノロジを使用する場合
- JMS に AQ の拡張機能が不要な場合
- アダプタ・ツールキットが役に立つ場合

AQ API の互換性

統合ソリューションで OMB または OJMS（あるいはその両方）を使用する場合は、次の互換性に関する考慮事項を慎重に検討してください。

- 複合プログラミング環境（たとえば、PL/SQL、C および Java）で言語間の交換に AQ を使用する場合は、JMS API を使用するよりもネイティブ AQ API を使用する方が、適切な選択肢です。
- ネイティブ AQ API の互換性とは、あるネイティブ AQ API でメッセージを挿入することができ、別のネイティブ AQ API で変換を必要とせずにそのメッセージを削除できることを意味します。たとえば、次の図のように、PL/SQL AQ API を使用してメッセージをキュー 1 にエンキューし、そのメッセージを、ある受信者に対しては OCI AQ API を使用して、別の受信者に対しては Java AQ API を使用してキュー 1 からデキューできます。

図 8-1 AQ API の互換性



あいにく、異なる JMS 実装間には、相互間の互換性もネイティブ API との互換性もありません。これは、これらの実装では、JMS メッセージのプロパティとペイロードの記録に AQ リポジトリで異なる属性が使用されるためです。したがって、ある JMS API（OMB JMS AQ ドライバなど）を使用して作成されたメッセージは、別の JMS API（OJMS API など）による収集前に変換する必要があります。ネイティブ AQ API の 1 つを使用してメッセージをデキューし、メッセージ・プロパティと JMS プロパティを操作し、場合によっては、メッセージを新しいフォーマットで互換性のあるキューにエンキューする前に、JMS プロパティをペイロードから切り離す必要があります。

その他のメッセージ処理テクノロジとの相互運用性

OMB には、Oracle AQ、TIBCO 社の Rendezvous、IBM MQSeries が、Java プログラミング環境で JMS メッセージ・サーバーとして動作できるためのドライバが多数用意されています。これらのメッセージ・サーバーを使用すると、AQ、MQSeries、Rendezvous で OMB 固有の JMS フォーマットのメッセージを作成できます。また、MQSeries と Rendezvous では、通常のメッセージも作成できます。

これらの異なるメッセージ・フォーマットとメッセージ・サーバーを使用している複数のキューの間で伝播を定義することによって、OMB をキュー間におけるメッセージ変換サービスとして使用し、たとえば、ネイティブ MQSeries から OMB 固有の JMS フォーマットの AQ メッセージへの変換などのメッセージ変換を行うことができます。

サポートされるメッセージのタイプには、いくつかの制約事項があります。詳細は、『Oracle Message Broker Administration Guide Release 2.0.1.0 for SPARC Solaris and Windows NT』および 8-3 ページの「ドライバ」を参照してください。

OMB では、クライアント側のコールアウトを Java、C および C++ で定義できます。このコールアウトは、Mercator Enterprise Broker への接続に使用できます。

Oracle Workflow や Java API には、OMB から直接、または PL/SQL コールを使用して接続できます。OMB から PL/SQL API への接続は、Java で事前定義されているプログラミングを使用します。

MQSeries の例

AQ の章で使用した例と同じ MQSeries の例を使用して、OMB への統合シナリオの一部を説明します。

次に示す例は、単に処理を表現するために使用したもので、ソリューションの 1 つとして推奨するものではありません。

シナリオ

ハブ・アンド・スポーク型統合ソリューションに組み込まれているアプリケーションの 1 つに、既成のメッセージ・ベースのインタフェースがあり、このインタフェースは、一連の MQSeries キューにメッセージを配置し、MQSeries メッセージのフォームでのビジネス・イベントを予定しています。アプリケーション・インタフェースは C 言語で記述され、システ

ム固有の MQSeries API を使用してメッセージが作成および使用されます。その目的は、MQSeries から AQ に、および AQ から MQSeries にメッセージを転送することです。

この目的は単に、MQSeries のメッセージを OMB に取得し、OMB から MQSeries のメッセージを取得することです。

このソリューションは、MQSeries との間で送受信するメッセージのタイプ、アーキテクチャ原理、言語または統合ソリューションに使用される言語、モジュール化に対する考え方、使用される規格などによって異なります。

図 8-2 MQ キュー - MQ ドライバ - OMB

関連項目：

- ソリューションは簡単ですが、第 7 章「Oracle アドバンスト・キューイングと JMS」にある制約と制限事項の対象です。
- Rendezvous への通信の場合も同様で、ソリューションは簡単です。ただし、第 7 章「Oracle アドバンスト・キューイングと JMS」を参照してください。

プログラム的な図

図 8-3 MQSeries - C プログラム - OMB

クライアント側の標準的な OMB API を使用して OMB と通信する事前定義済みの Java、C または C++ プログラムを開発することは、OMB キューをその他のメッセージ処理テクノロジに間接的に接続する代替手段です。

ワークフローの例

あるアプリケーション統合のシナリオでは、アプリケーションが MQSeries や TIBCO 社の Rendezvous などの他のメッセージ処理テクノロジと対話することが望ましい場合があります。これは、要件の分離、アプリケーション・プラットフォーム、既存のインタフェース、または以前に選択した戦略的テクノロジが理由です。これらのメッセージ処理テクノロジが、いかに効率よく Oracle Integration Server の製品と対話できるかを理解することが重要です。

AQ 以外のメッセージ処理テクノロジにメッセージを抽出するアプリケーションへのインタフェースがある場合、最初のタスクは、目的を識別してそのメッセージ処理テクノロジから目的にたどり着くまでのルートを確立することです。この時点で最もありがちな誤りは、A から B への最も直接的なルートが、最短であり、単純で簡単、または最も効率的であると思い込むことです。

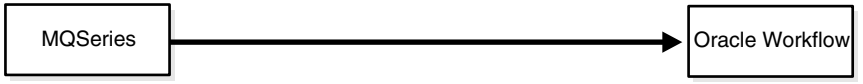
プロセスを例で示すと確かに簡単です。次のシナリオは処理を単に表現するために使用したもので、ソリューションの 1 つとして推奨するものではありません。

シナリオ

ハブ・アンド・スポーク型統合ソリューションのアプリケーションの 1 つに、既成のメッセージ・ベースのインタフェースがあり、このインタフェースは、一連の MQSeries キューにメッセージを配置し、MQSeries メッセージの形式でのビジネス・イベントを予定しています。アプリケーション・インタフェースは C 言語で記述され、システム固有の MQSeries API を使用してメッセージが作成および使用されます。目的の 1 つは、Oracle Workflow のビジネス・プロセスをインスタンス化するためのビジネス・イベントを示す MQSeries のメッセージを取得することです。

ソリューションは、アーキテクチャ原理、統合ソリューションに使用される言語、モジュール化に対する考え方、使用される規格などによって異なります。

図 8-4 MQSeries - Oracle Workflow



代替シナリオやソリューションの考えられる全パターンを網羅することは不可能ですが、シナリオのバリエーションをもとにポイントをいくつか示します。

バリエーション 1

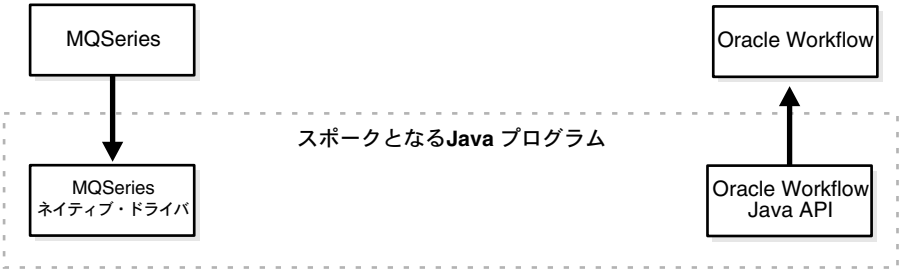
JMS 標準を適用せず、Java プログラミング言語を使用して、コード・ベースのアダプタを開発するアプローチを採用します。デザイナは、荒削りなアプローチ（つまり、大きなプログラム単位で、単位数は少なく）によるモジュール化を希望しています。

次の要件は不要です。

- 監査証跡へのメッセージの保存
- SQL を使用した未使用メッセージの表示
- MQSeries 未対応パブリッシャからのメッセージの受信

このバリエーションでは、MQSeries のネイティブ Java API を経由してメッセージを取得し、ビジネス・プロセスをインスタンス化する Java ワークフロー・メソッドを起動する前に、メッセージ・ペイロードを必要なパラメータ・フォーマットに変換する簡単な事前定義済み Java プログラムを記述します。

図 8-5 バリエーション 1



バリエーション 2

Java プログラミング言語を使用して、コード・ベースのアダプタを開発するアプローチを採用します。すべてのメッセージ・インタフェースに対して JMS 標準を適用します。デザイナは、荒削りなアプローチ（つまり、大きなプログラム単位で、単位数は少なく）によるモジュール化を希望しています。

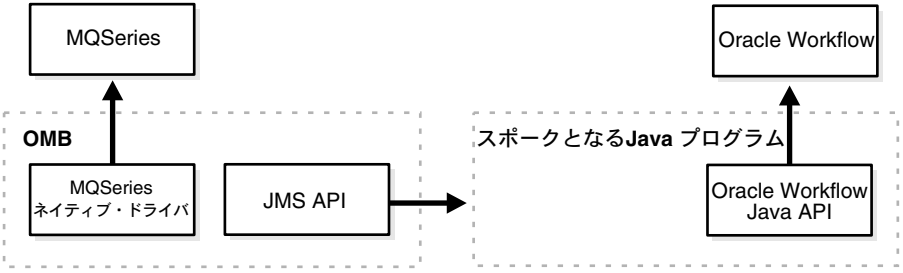
MQSeries のキューと Oracle Workflow のインスタンスは異なるコンピュータ上に常駐し、Oracle データベースはアプリケーション・コンピュータにインストールされていません。将来は、TIBCO 社の Rendezvous メッセージと AQ メッセージを使用して、パブリッシャから同じビジネス・イベントを受信する必要があります。

次の要件は不要です。

- 監査証跡へのメッセージの保存
- SQL を使用した未使用メッセージの表示

このバリエーションでは、Oracle Message Broker を経由してメッセージを取得し、ビジネス・プロセスをインスタンス化する Java ワークフロー・メソッドを起動する前に、メッセージ・ペイロードを必要なパラメータ・フォーマットに変換する簡単な事前定義済み Java プログラムを記述します。

図 8-6 バリエーション 2



バリエーション 3

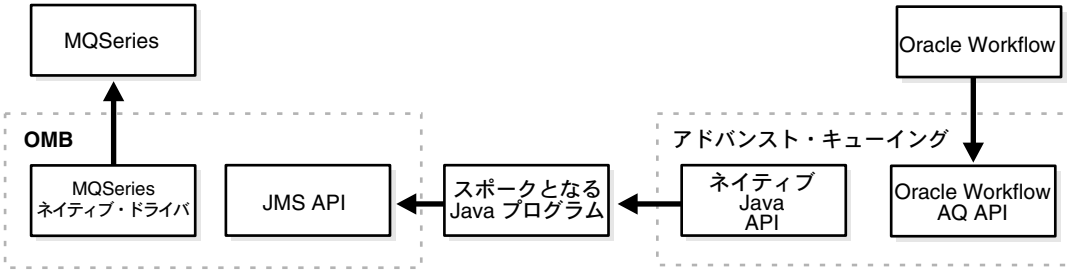
状況はバリエーション 2 と同じですが、このケースでは、Oracle Workflow のビジネス・プロセスのステップまたはアクティビティの最中にビジネス・イベントが発生したことを MQSeries 対応のアプリケーションに非同期で通知します。

次の要件は不要です。

- 監査証跡へのメッセージの保存
- SQL を使用した未使用メッセージの表示

このバリエーションでは、ワークフロー AQ API を使用して非同期メッセージを記録するアダプタを開発します。ネイティブ Java AQ API を使用してメッセージを取得して関連する JMS プロパティを設定し、Oracle Message Broker にメッセージを渡し、メッセージが MQSeries キューに記録されるように、簡単な事前定義済み Java プログラムを開発します。

図 8-7 バリエーション 3



バリエーション 4

アダプタとコネクタの開発にグラフィカルなアプローチを採用し、コーディング作業を最小化します。特定の言語は指定されていません。また、外部インタフェースには XML が有力です。MQSeries のキューと Oracle Workflow のインスタンスは異なるコンピュータ上に常駐し、アプリケーション・システムには Oracle データベースがインストールされていません。

メッセージは監査証跡として保存が必要です。将来は、TIBCO 社の Rendezvous メッセージと AQ メッセージを使用して、パブリッシャから同じビジネス・イベントを受信する必要があります。

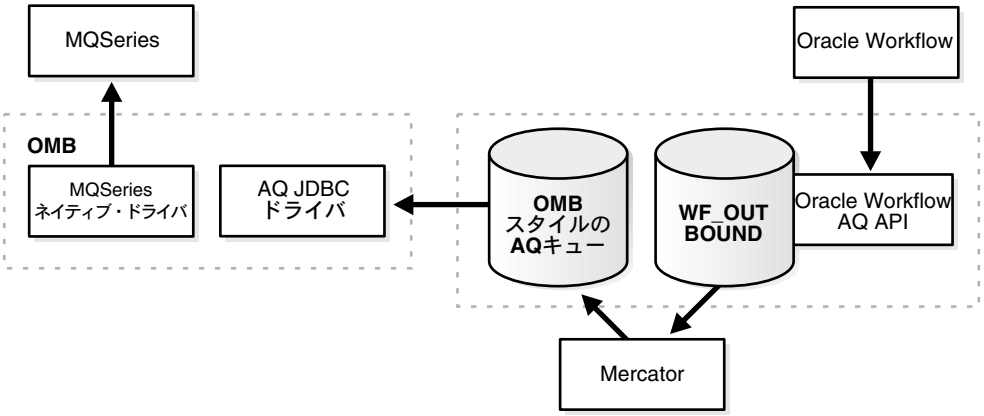
ビジネス・イベントはバリエーション 3 と同じで、Oracle Workflow のビジネス・プロセスのステップまたはアクティビティの最中にビジネス・イベントが発生したことを MQSeries 対応のアプリケーションに非同期で通知します。

このケースの場合、コーティング作業は削減されますが、多数のコンポーネントが関与します。`WF_OUTBOUND` キューに書き込むことによって、Oracle Workflow のステップが非同期コールアウトを実行します。

Mercator は、`WF_OUTBOUND` をサブスクライブしてメッセージを OMB スタイルの JMS フォーマットに変換し、そのメッセージを OMB スタイルの AQ キューに配置します。OMB は、AQ キューと MQSeries キューにメッセージを伝播します。

変換ステップの数が増加すると、個々のメッセージのエンドツーエンド送信時間がわずかながら増加する可能性がありますが、メッセージのスループットには影響を及ぼしません。

図 8-8 バリエーション 4



バリエーション 5

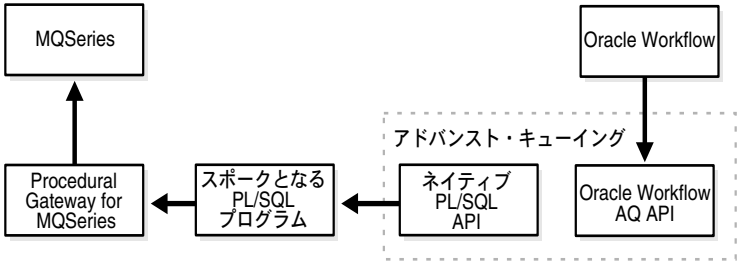
このバリエーションの場合は、コード・ベースのアダプタ開発アプローチを採用します。ただし、特定の言語は指定されていません。アプリケーションは、ほぼ Oracle ベースのみで、Oracle データベースが実行されていない数少ないコンピュータの 1 つでは MQSeries が使用されています。

メッセージは監査証跡として保存が必要です。将来は、TIBCO 社の Rendezvous メッセージと AQ メッセージを使用して、パブリッシャから同じビジネス・イベントを受信する必要があります。

ビジネス・イベントはバリエーション 3 および 4 と同じで、Oracle Workflow のビジネス・プロセスのステップまたはアクティビティの最中にビジネス・イベントが発生したことを MQSeries 対応のアプリケーションに非同期で通知します。

MQSeries メッセージは単純で、予想されるメッセージ量は毎分約 200 メッセージです。

図 8-9 バリエーション 5



WF_OUTBOUND キューに書き込むことにより、Oracle Workflow のステップが非同期コールアウトを実行します。

事前定義済み PL/SQL プログラムが WF_OUTBOUND キューをサブスクライブしてメッセージ・データをアプリケーションが必要とするフォーマットに変換し、そのメッセージを MQSeries に配置する Oracle Procedural Gateway をコールします。次に、MQSeries を使用して、メッセージを収集できるアプリケーションがあるコンピュータにそのメッセージを伝播します。

使用可能なツール

この項では、次の各種ツールを説明します。

- プログラミング言語
- 変換エンジン
- メッセージ変換ツール

プログラミング言語

事前定義済みプログラムを記述して、メッセージ処理テクノロジを Oracle Integration Server の任意の製品に接続することができます。ただし、これはメッセージ処理テクノロジと Oracle Integration Server 製品の両方に、そのプログラミング言語に対する API がある場合に限ります。

必要なプログラミングの量は、2 つの製品間のメッセージ・フォーマットの相違の程度、API がベースとしている標準、および API が標準をサポートしている程度によって異なります。事前定義済みプログラムは、単純性を保つために、1 つのサービス・コールまたはトピックには 1 つのトピックのみを接続するように設計されていることに留意してください。コードを保持することも必要です。

メッセージ処理製品のキューと別の製品の API 部分との間の汎用的な通信を支援するプログラムは、開発に時間がかかり、完全なテストが困難で、不安定になりがちで、直線的に基準化することが困難です。

効果的なプログラムの多くは通常、複数の製品ベンダーによる共同作業で、大規模な研究開発費を投じ、複雑なマルチスレッド・プログラミングを使用して許容可能な拡張レベルに対応するように開発されます。

変換エンジン

Mercator Enterprise Broker などの変換ツールは、あるメッセージ処理テクノロジと別のメッセージ処理テクノロジとの間でメッセージを変換できます。メッセージ処理テクノロジ間でメッセージを移行するという要件のみでは、このような特化した製品の購買と実装を正当化し保証するためには不十分です。そのような製品は、大量の記憶領域を使用し、トランザクション的に弱点があったり、テクノロジ間を移動するときに同期コミット・モデルを提供していない場合もあります。

メッセージ変換ツール

オラクル社や他のベンダーが提供する製品の多くは、あるテクノロジから別のテクノロジにメッセージを変換するために必要なプログラミング作業を簡略化します。

3 種類の重要なメッセージ変換ツールがあります。

- **Procedural Gateway for MQSeries :** MQSeries への PL/SQL 接続と AQ 接続に対応する Oracle 製品です。
- **Oracle Message Broker :** Oracle AQ、TIBCO 社の Rendezvous および IBM MQSeries が Java プログラミング環境で JMS メッセージ・サーバーとして動作できるドライバを数多く提供する Oracle 製品です。いくつかの制限事項があります。詳細は『Oracle Message Broker Administration Guide Release 2.0.1.0 for SPARC Solaris and Windows NT』を参照してください。
- **Oracle 用 TIB アダプタ :** この TIBCO 社の製品は、オラクル社と共同で開発され、Oracle アドバンスト・キューイングと TIB/Rendezvous（TIBrv）間の変換ソリューションをコーディング不要で提供します。AQ から TIBrv にメッセージを変換し、TIB バスを介してメッセージを移送し、AQ メッセージ・プロパティを失わずに TIBrv から AQ に再び戻す変換するように設計されています。メッセージは Oracle の同期コミット・モデルを使用して変換されます。

9

ディレクトリ・サービス（LDAP）

Oracle Message Broker（OMB）は、Lightweight Data Access Protocol（LDAP）ディレクトリ内の接続先アドレスを検索して、Point-to-Point メッセージ機能に関するメッセージ・ルーティング情報を判断します。OMB は、標準の Java Naming and Directory Interface（JNDI）ドライバを使用して LDAP ディレクトリにアクセスします。

OMB は、LDAP にメッセージ・ルーティング情報を格納することによって、2 つの重要な利点を提供します。

- メッセージ・ルーティング情報の管理を集中化します。
- LDAP ディレクトリ内のエントリを変更することによって、メッセージ・ルーティングへの動的な変更を簡単に行うことができます。

この章では、Oracle LDAP ディレクトリ、Oracle Internet Directory（OID）が提供するディレクトリ・サービスについて説明します。この章には次のトピックが含まれています。

Java とディレクトリ・サービスの統合

この項では、次の項目を説明します。

ディレクトリ・サービス ― 概要

ディレクトリ・サービスは、集中化されたネットワーク・ベースのリポジトリです。このリポジトリは、アプリケーション間で共有されるか、または高度に分散される情報を格納し、それらの情報に対するアクセスを提供します。ディレクトリ・サービスは、ネットワーク全体にわたるユーザー、システム、ネットワーク、サービスおよびアプリケーションに関する情報の共有を支援し、イントラネットおよびインターネット・アプリケーションの開発できわめて重要な役割を果たします。

ディレクトリ・サービスは、長い間、様々な形で使用されてきました。たとえば、電子メール・システムには、ユーザー情報を格納するディレクトリや、送信者と受信者間でメッセージをルートするディレクトリが含まれています。大企業では、名前、部門、マネージャ、社会保障番号およびその他の組織情報などを集中的に格納する従業員情報のリポジトリとして、様々なフォームのディレクトリを使用することがあります。

多くの企業におけるディレクトリ・サービスの進展は、これまで漸進的でしたが、その価値はインターネット・コンピューティングの成長に伴って急激に高まってきました。その本来の性質から、インターネット・アプリケーションは高度に分散されるため、ネットワーク全体で情報を共有するための効果的なメカニズムが必要です。

問題点

たとえば、クライアント / サーバー・アプリケーションからインターネット・アプリケーションに移行する経費報告アプリケーションを想定します。数人の人事管理専門職がインターネット・セルフサービス・アプリケーションに会社全体の経費報告を入力し、各従業員も自分の経費報告を入力します。セルフサービス・インターネット・アプリケーションにアクセスするユーザーの数は、クライアント / サーバー・システムにアクセスする数をはるかに上回ります。その結果、インターネット・ユーザーを管理するコストは、アプリケーションを管理する合計コストの大きな部分を占めることになります。

同様に、最近のインターネット・アプリケーションは、モジュール化された Enterprise JavaBeans または CORBA コンポーネントとして開発され、各コンポーネントは、その位置とパブリック・インタフェースを示す特定の名前でアクセスされます。コンポーネントがネットワーク内のあるノードから別のノードに移動する都度、そのコンポーネントをすべてのクライアントが検索できる新しい名前を指定する必要があります。イントラネットに多数の異なるコンポーネントが配置されると、これらのコンポーネントを管理するコストは、システムを所有するための合計コストの大きな部分を占めることになります。

最終的には、アプリケーションの開発にインターネット・アーキテクチャを使用して、アプリケーションの配布方法を簡素化します。つまり、アプリケーションを多数のクライアン

ト・コンピュータに配布するのではなく、集中化され、専門的に管理される少数のサーバー上に配布します。ただし、インターネット・アプリケーションには多くの新しいインフラストラクチャ・コンポーネントが必要です。一部の名前を挙げると、ORB、Web サーバー、Java 仮想マシン、データベースおよびアプリケーション・サーバーなどがあります。このようなインフラストラクチャ・コンポーネントを管理するためのコストも管理する必要があります。

ソリューション

すべてのインターネット・アプリケーションには、共通の問題があります。つまり、ユーザーやアプリケーションが効果的にアクセスおよび共有できる単一のリポジトリに、いかにして情報を集中化するかの問題です。ソリューションはディレクトリ・サービスです。ディレクトリ・サービスを使用すると、異なる種類の様々な情報を格納および共有できます。

たとえば、ディレクトリ・サービスは、OLTP アプリケーションおよびデータベースにアクセスするインターネット、イントラネットまたはエクストラネットのユーザーを管理できます。ユーザーは、ディレクトリに集中的に格納できるセキュリティ証明とアクセス制御権限で識別されます。したがって、管理者は、ユーザー情報や権限をユーザーがアクセスする個々のアプリケーションやデータベースで変更しないで、単一の場所にあるユーザー情報と権限を変更します。さらに、ディレクトリ・サービスは中央リポジトリとして動作して、イントラネット環境における Net8 や IIOP リスナー、ディスパッチャ、接続マネージャなどの分散ネットワーク・リソースを管理します。

さらに、ディレクトリ・サービスを使用すると、Net8 の名前を `TNSNames` ファイルに保存する場合と同様の方法で、CORBA および Enterprise JavaBeans アプリケーション・コンポーネントの名前を格納できます。

このように、ディレクトリ・サービスは、インターネット・アプリケーションの合計所有コストを軽減するための管理インフラストラクチャの重要なコンポーネントとして認識されています。ディレクトリ・サービスは、2 種類の方法でこれを実現します。

- 分散情報の管理を単一の場所に集中化する。
- 企業イントラネットおよびエクストラネット内のユーザーとアプリケーションが、ディレクトリに格納されている情報を簡単に共有できるようにする。

ディレクトリ・サービスと LDAP の技術的概要

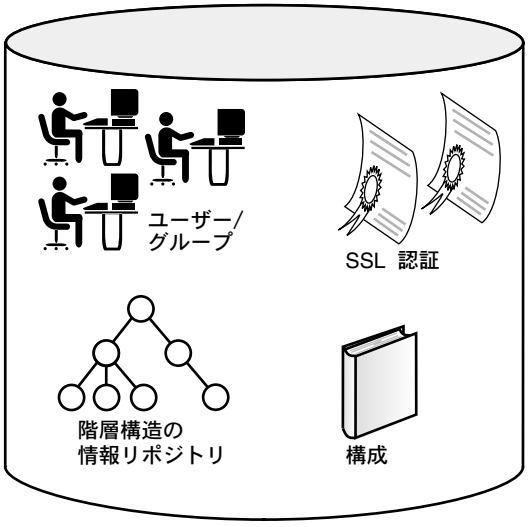
ディレクトリ・サービスには、情報を格納するための堅固で高速な、しかも拡張性の高いデータ・ストアが必要です。そのため、このサービスは通常、エンタープライズ・データベース上に構築されます。したがって、ディレクトリ・サービスの本質は、簡単に拡張して異なる種類の様々な情報を格納できる階層スキーマを持つ特化されたデータベースです。

大半のディレクトリ・サービスは、Lightweight Directory Access Protocol （LDAP）を経由してアクセスされます。この LDAP では、オープンで、ベンダーに依存しない中立的な業界標準ネットワーク・プロトコルおよびディレクトリに対する一連のアクセス方法が定義されています。LDAP は、IP アドレス参照に使用される Domain Name System（DNS）と同様に、イントラネットとインターネット上の大半のシステムで、ディレクトリ情報に関する標

準的なアクセス方法となりました。LDAP は、大半のネットワーク・オペレーティング・システム、グループウェアおよびシュリンクラップのインターネット・アプリケーションでサポートされています。

LDAP は、X.500 OSI ディレクトリ・サービスの規格から発展したものです。X.500 は、有用なアイデアを数多く採り入れましたが、インターネット・アプリケーションに役立てるにはあまりにも重く複雑でした。LDAP は、完全な X.500 仕様の 90% の機能をその 10% のコストで提供するように設計されました。LDAP では、メッセージが回線を経由して移送される際のフォーマットを徹底的に簡素化し、データ要素を簡単な文字列で表現し、X.500 であまり使用されなかった冗長な操作の大半が削除されました。回線上での効率とインターネット・アプリケーションとの適合性によって、LDAP はインターネット・ディレクトリ・サービスの最前線に進出しました。

図 9-1 LDAP ディレクトリ・サービス



LDAP では、4 つの基本的な情報モデルが定義されます。これらのモデルによって、LDAP ディレクトリに格納できる情報の種類、その情報を使用する操作が完全に記述されます。4 つのモデルは、次のとおりです。

- **情報モデル :** LDAP 情報モデルは、LDAP ディレクトリに格納できる情報の種類を定義します。
- **ネーミング・モデル :** LDAP ネーミング・モデルは、LDAP ディレクトリ内情報の編成方法と参照方法を定義します。

- **機能モデル :** 機能モデルは、LDAP ディレクトリ内の情報を使用して行うことができる内容およびその情報へのアクセス方法と更新方法を定義します。
- **セキュリティ・モデル :** セキュリティ・モデルは、LDAP ディレクトリ内情報の保護方法、およびユーザーとアプリケーションがディレクトリにアクセスするために必要な権限の種類を定義します。

次に、インターネット・アプリケーションの開発にあたってこれらのモデルの使用方法を理解するために、それぞれのモデルの詳細を説明します。

LDAP 情報モデル

LDAP 情報モデルは、**エントリ**の概念がその中心となっています。エントリはディレクトリに作成され、実社会におけるオブジェクトまたは概念（個人、組織、プリンタなど）に関する情報を保持します。エントリは、オブジェクトについて記録された情報を含んだ複数の属性で構成されています。各属性には、1 つの型と 1 つ以上の値があります。

属性の型には関連付けられた構文があります。この構文は、属性の値に格納できる情報の種類、その値の検索や他のディレクトリ操作時における動作を定義します。たとえば、属性 cn（common name の短縮名）には、次の 3 つの特性を含んだ caseIgnoreString と呼ばれる構文があります。

- 属性は辞書編成順にソートされます。
- 比較時に大 / 小文字の区別は無視されます。
- 値は文字列であることが必要です。

属性の型に様々な制約事項を関連付けて、属性に格納できる値の個数を 1 つまたは多数に制限したり、値のサイズを制限することもできます。たとえば、個人の識別番号を保持する属性は、単一の値にできます。写真を保持する属性は、過度に記憶領域を使用しないように 10KB 以下のサイズに制限できます。エントリに必要で許容される属性は、サーバー・ベースごとに定義されているコンテンツ・ルールまたは各エントリ内の objectclass と呼ばれる特別な属性で管理されます。この objectclass 属性の値は、エントリの種類（個人、組織など）を識別し、必須の属性とオプションの属性を判断します。

たとえば、objectclass 個人には、sn （surname）と cn（common name）属性が必要です。これは、スキーマに関連付けられている LDAP のルールと等価です。

LDAP リリース 3 では、スキーマ規定を完全に廃止して、選択した属性をエントリに追加する機能が追加されています。これは、スキーマ・エントリを新たに追加して全クライアントとサーバーに新しい要素を認識させるオーバーヘッドが高い環境では有効です。これを容易にするために、この LDAP リリース 3 プロトコルには、extensibleObject と呼ばれる特別な objectclass が追加されています。エントリの objectclass 属性に値 extensibleObject が含まれている場合、他の属性は、実施されているスキーマ・ルールに関係なくエントリで許可されます。

LDAP ネーミング・モデル

プロトコルの要件ではありませんが、通常、LDAP ディレクトリ・エントリは、たとえば、地理的や組織的な分散に続く階層ツリー構造に配置されています。エントリには階層内の位置に従って識別名（DN）が付いています。DN の各コンポーネントは、相対識別名（RDN）と呼ばれ、エントリからの 1 つ以上の属性で構成されています。UNIX や Windows のファイル・システムに詳しい方には、この概念はパス名とファイル名に相当します。RDN はファイルの名前に相当し、DN はファイルに対する絶対パス名に相当します。ファイル・システムと同様に、兄弟関係のエントリ（同じ親を持つエントリ）には、異なる RDN が必要です。ただし、LDAP には、ファイル・システムと異なる重要な相違点が 2 つあります。

- **並び順 :** ファイル名とは異なり、LDAP コンポーネントは、最下位（エントリ自体に名前を付ける）コンポーネントで開始し、最上位（ルート直下の）コンポーネントに続きます。これとは対照的に、ファイル名は通常、ルートから開始してファイルまたはディレクトリに続きます。これは、構文構成上の違いのみで、意味上の違いはありません。
- **名前の区切り :** LDAP エントリは、識別名と呼ばれるフォーマットを使用して名前が付けられています。**識別名**は、カンマまたはセミコロンで区切られた一連の RDN コンポーネントです。LDAP 名コンポーネントはカンマで区切られ、ファイル名のようにスラッシュまたはバックスラッシュでは区切られていないことに注意してください。たとえば、ファイル名は、次の構造になります。

  ```
  /usr/local/ldap/include/ldap.h
  ```

  一方、LDAP エントリは、次のような構造になります。

  ```
  cn = Steve Harris, o=Software Industry, c=India
  ```

この例のように、別のエントリを指し示す別名エントリによって、ツリー状の階層を避けることができます。また、階層はサポートされていますが、LDAP では不要です。情報を階層的に編成して検索する機能は、多くのアプリケーションで有用ですが、一方で不便な場合もあります。LDAP モデルでは、双方のケースを処理できます。これは、非階層のケースでは 1 レベルの階層を平面的なネームスペースとして使用できるためです。

LDAP 機能モデル

LDAP ディレクトリに情報を配置すると、LDAP はクライアントを認証する 9 種類の異なる操作を定義します。これによって、クライアントは、ディレクトリにアクセスして検索し、情報を取り出して更新できます。

検索操作によって、LDAP ツリーの定義済み領域から選択基準に従って情報が選択されます。一致する各エントリに対して、要求された一連の属性（値付きまたは値なし）が戻されます。検索されるエントリは、単一のエントリ、エントリのすぐ下位のエントリ、またはエントリのサブツリー全体に及ぶことができます。検索では、別名エントリを自動的に後に続けることができ、クライアントは検索のサイズと時間の制限を指定できます。LDAP リリース 3 のディレクトリでは、問合せの結果をスクロール可能な結果セットのフォームに取り出すこともできます。つまり、検索結果をページごとに取り出し、クライアントは必要な結果を上下にスクロールしながら参照できます。

LDAP のまとめ

LDAP には、ディレクトリ・サービスにアクセスする理想的な方法を提供するための 3 つの重要な機能があります。

分割可能なネーミング・コンテキスト：第 1 に、LDAP は、多数の異なるネーミング・コンテキストに分割できるネームスペースを提供しています。ディレクトリ内の情報はこのネーミング・コンテキストに編成できます。各ネーミング・コンテキストは、単一の論理ディレクトリに関する認識をクライアントに与えるために、物理的な個別のサーバー・ノード上でホストになることができます。これにより、ディレクトリ・サービス管理者は、頻繁にアクセスされる情報をパフォーマンスの高いサーバー・コンピュータに配置して、パフォーマンスを上げることができます。たとえば、多数の異なるサプライヤがアクセスする E-Commerce アプリケーションのセットを管理する場合、優先順位に基づいてサプライヤを一連のネームスペースに分割できます。たとえば、最優先順位のサプライヤを 1 つのネームスペースに、次の層のサプライヤを別のネームスペースに格納することができます。

また、LDAP ネームスペースは物理的に分割可能なため、管理者は、ディレクトリ情報をミラー化した CPU 構成に格納して、単一場所での障害発生によるリスクを回避することができます。

階層情報：第 2 に、LDAP ベースのディレクトリ・サービスは、情報を階層パターンに編成します。これにより、ディレクトリ・サービスは、サプライチェーン・アプリケーション用の在庫リスト、E-Commerce アプリケーション用の価格表やカタログなどの階層情報をしめす自然な方法になります。たとえば、在庫の例では、各製品は、子エントリを持つディレクトリの単一の製品エントリとして示されます。子エントリは、製品自体の各サブコンポーネントに対する在庫レベルを示します。

ディレクトリ・サービスの情報は、ネーミング・コンテキストのルート・ノードを検索するために、その情報自体を動的に問い合せることもできます。また、ディレクトリの階層を順番にナビゲートすると、子エントリにアクセスできます。

セキュリティ規定：最後に、LDAP ディレクトリ・サービスは、格納されている情報を保護するために、厳格なセキュリティ・メカニズムを数多く提供しています。たとえば、ディレクトリ・ユーザーは、ユーザー名とパスワード、または SSL/X.509 リリース 3 の認証（バインド操作を経由）のいずれかを使用して、ディレクトリに対する認証を受ける必要があります。ユーザーが認証されても、アクセスできる情報は、Access Control Lists によってなお制限されています。

Access Control Lists は、ディレクトリ内の情報を読み込むことができるユーザーを指定します。ディレクトリのエントリは階層的に格納されているため、エントリのサブツリーに加え、個々のエントリとエントリ属性の両方に対してもアクセス制御を簡単に適用できます。さらに、製品エントリに適用されているアクセス制御は、すべての子に自動的に継承されます。その結果、ディレクトリ・サービスによって、詳細にわたる方法でディレクトリの情報へのアクセスを制御できます。そして、ディレクトリ・サービスは情報を共有するための理想的なメカニズムとなります。

オラクル社は、イントラネット、エクストラネットまたはインターネットを介して配布される情報または機能を取り扱う集中管理機能として、ディレクトリ・サービスの使用をお薦めします。これには、ユーザー、アプリケーション・コンポーネント、セキュリティおよびビジネス・プロセスまたはワークフローが含まれます。企業イントラネットおよびエクストラネット内のアプリケーション間で共有される情報を管理する場合にも、ディレクトリの使用をお薦めします。

Oracle Internet Directory

Oracle Internet Directory は、ディレクトリ・サービスを実装する Oracle8*i* データベース・サーバー上で実行されるアプリケーションです。Oracle Internet Directory は、LDAP リリース 3 に準拠しています。

Oracle Internet Directory は、Oracle8*i* サーバーをベースとしており、次のようなデータベースの標準的な機能を活用して、サイズ、パフォーマンス、信頼性、可用性および管理の点で卓越した拡張性を提供しています。

- データベースのバックアップとリカバリの機能
- マルチスレッド・サーバーのマルチスレッド・モデル
- 接続プール機能
- 地域的障害を持つ LDAP サーバーの実装に対応する高度な対称型レプリケーション

LDAP サーバーへのアクセスは、Secure Socket Layers（SSL）リリース 3 に対する OID のサポートで確実に提供されます。

> **関連項目：** Oracle Internet Directory とその機能の詳細は、『Oracle Internet Directory 管理者ガイド』を参照してください。

10

Oracle Workflow

この章では Oracle Workflow を説明します。次の項目が含まれています。

- 概要
- ワークフローの主要コンポーネント
- ワークフローの主要機能

概要

Oracle Workflow は当初、Oracle Applications のユーザーに焦点を絞った複雑なワークフローを管理するためのツールとして開発されました。オラクル社は、Oracle Workflow をユーザー・ベースの複雑なビジネス・プロセスを管理する強力なツールと認識し、後続のリリースでは、スタンドアロン製品として使用できるようしました。

最新のリリースには、拡張された製品機能が含まれており、アドバンスト・キューイングを使用して同期および非同期のコールアウトに対応して、完全に自動化されたビジネス・プロセスを提供しています。

Oracle Workflow は、次のコンポーネントで構成されています。

- Oracle Workflow Builder
- ワークフロー・エンジン
- ワークフロー定義ローダー
- ワークフロー・モニター

ワークフローの主要コンポーネント

Oracle Workflow Builder

Oracle Workflow Builder では、簡単なドラッグ・アンド・ドロップ操作を使用して、ビジネス・プロセスを作成、表示または変更できます。Oracle Workflow Builder を使用すると、アクティビティ、項目の型およびメッセージを含めてすべてのワークフロー・オブジェクトを作成および変更できます。

ワークフロー・アクティビティは、いつでも追加、削除または変更でき、また、複数アクティビティ間に前提条件となる関係を新たに設定できます。ワークフローの概要レベル・モデルで簡単に作業を行い、必要に応じて、アクティビティをワークフロー内で詳細レベルに拡張することができます。さらに、Oracle Workflow Builder は、デスクトップ PC または切断したラップトップ PC から操作できます。

ワークフロー・エンジン

Oracle8*i* データベース・サーバーに埋め込まれているワークフロー・エンジンは、ワークフローの状態を監視し、プロセスのアクティビティのルーティングを調整します。ワークフロー・アクティビティの完了など、ワークフローにおける状態の変更は、PL/SQL API や Java API を介してエンジンに連絡されます。柔軟に定義されているワークフロー・ルールに基づいて、ワークフロー・エンジンは、実行に適切なアクティビティを判断して実行します。ワークフロー・エンジンは、ループ、ブランチ、パラレル・フロー、サブフローを含めた高度なワークフロー・ルールをサポートしています。

ワークフロー定義ローダー

ワークフロー定義ローダーは、データベースと対応するフラット・ファイル表記との間でワークフロー定義を移動させるユーティリティ・プログラムです。このユーティリティを使用すると、ワークフロー定義を開発用データベースから本番データベースに移動したり、既存の定義にアップグレードを適用することができます。ワークフロー定義ローダーは、スタンドアロン・サーバー・プログラムである他に、Oracle Workflow Builder に統合され、データベースとファイルの両方でワークフロー定義をオープンおよび保存できます。

ワークフロー・モニター

Oracle Workflow のキュー API は、アプリケーション・プログラムまたはランタイム・フェーズのワークフロー・ファンクションによってコールされ、ワークフローのアドバンスト・キューイング処理を操作することができます。Oracle Workflow では、アウトバウンドおよびインバウンドのキューが確立されます。キューにあるデータ・パッケージは、**イベント**または**メッセージ**と呼ばれます。ここでのメッセージとは、通知アクティビティに関連付けられているメッセージとは異なります。

イベントは、エージェントが使用して処理するためにアウトバウンド・キューにエンキューされます。これらのエージェントは、データベースの外部にあるアプリケーションとなる可能性があります。同様に、エージェントは、ワークフロー・エンジンが使用して処理するために、メッセージをインバウンド・キューにエンキューする場合もあります。アウトバウンドとインバウンドのキューは、外部アクティビティのワークフロー・プロセス内への統合を支援します。

注意： バックグラウンド・エンジンは、個別キューまたは遅延キューを使用します。

ワークフローの主要機能

この項では、次の項目を説明します。

- 完全なプログラム拡張性
- 電子通知
- 電子メールの統合
- インターネット対応のワークフロー
- 監視と管理
- ビジネス・イベント・システム
- 使用方法
- AQ API
- PL/SQL コールアウト機能
- PL/SQL と Java を使用したビジネス・プロセス・インスタンスのインスタンス化
- 相互運用性

完全なプログラム拡張性

Oracle Workflow では、ユーザー自身の PL/SQL プロシージャまたは外部ファンクションをワークフローのアクティビティとして含めることができます。プログラムの前提条件を満たしていることをワークフロー・エンジンが検出した場合は、アプリケーション・コードを変更せずに、ユーザー自身のプログラムを実行させることができます。

電子通知

Oracle Workflow では、購買依頼または受注の承認などの自動化できないアクティビティを処理するために、ワークフロー内にユーザーを組み込むことができます。電子通知は、個々のユーザーまたはユーザーのグループであるロールにルートされます。そのロールに関連付けられているユーザーは誰でも、その通知に対処できます。

各通知には、ユーザーが意思決定を行う上で必要なすべての情報が記載されたメッセージが含まれています。情報は、メッセージ本体に埋め込まれているか、または独立したドキュメントとして添付されている場合があります。Oracle Workflow では、次のワークフロー・アクティビティへの移動を判断するために、各通知アクティビティに対する回答を解析します。

電子メールの統合

電子メールのユーザーは、選択している電子メール・アプリケーションを使用して、未処理の作業項目の通知を受信したり、その通知に対して応答することができます。電子メール通知には、その通知の応答に別の手段を提供する添付ファイルを含めることができます。

インターネット対応のワークフロー

標準 Web ブラウザへのアクセス手段があるユーザーは、ワークフローに組み込むことができます。Web ユーザーは、「Notification Web Page」にアクセスして未処理の作業項目を参照してから、別のページに移動して詳細を参照したり応答することができます。

監視と管理

ワークフロー管理者およびユーザーは、Java をサポートしている標準 Web ブラウザを使用してワークフロー・モニターに接続し、ワークフロー・プロセス内作業項目の進捗状況を参照できます。ワークフロー・モニターは、ワークフロー・プロセスの特定のインスタンスに関するプロセス図を注釈付きで表示し、それによってユーザーは作業項目のステータスをグラフィックに参照できます。また、ワークフロー・モニターは、作業項目、プロセスおよびそのプロセス内の各アクティビティについて、個別のステータス・サマリーも表示します。

ビジネス・イベント・システム

ビジネス・イベント・システムは、Oracle アドバンスト・キューイング（AQ）のインフラストラクチャを使用してシステム間でビジネス・イベントを通信するアプリケーション・サービスです。ビジネス・イベント・システムはイベント・マネージャとイベント・アクティビティで構成されています。イベント・マネージャでは、システムにとって重要なイベントに関するサブスクリプションを登録でき、イベント・アクティビティでは、ワークフロー・プロセス内のビジネス・イベントをモデル化できます。

イベントが発生すると、イベントを発生させたコードと同じトランザクションでサブスクライバ・コードが実行されます。サブスクリプション処理には、他のキューまたはシステムへのイベント情報の送信、イベント情報でのカスタム・コードの実行、およびワークフロー・プロセスへのイベント情報の送信を含めることができます。

ワークフロー・モニター

ワークフロー・モニターは、実行中のワークフローの特定インスタンスの現状をグラフィックで表示します。このワークフロー・モニターは、長時間にわたるメッセージ・フローが統合ソリューションの中を次々と移動するのを、手動で追跡する場合に非常に便利です。このモニターは、Java ベースのアプレットで、ブラウザを使用してアクセスできます。

使用方法

リリース 2.5.2 またはそれ以降の Oracle Workflow Standalone Edition を統合フロー・マネージャとして使用し、ビジネス・イベントのフローをインスタンス化し、制御、同期化してアプリケーション間のビジネス・イベント通信を調整することができます。

Oracle Workflow Builder のグラフィカル・インタフェースを使用して、ビジネス・プロセスのステップを実行する順序と条件を定義します。さらに、PL/SQL API および AQ API を使用して自動化されたビジネス・プロセスのステップを実装し、ワークフロー・エンジンによる統合フローのインスタンス化を要求します。このエンジンは、外部で定義されたプロセス・ステップ（ペイロード変換の実施など）の実行を要求します。

Oracle Workflow は、ユーザーの手動による介入を強力にサポートします。フローを定義して、異常な、疑わしい、または問題のあるビジネス・イベントのインスタンスを検出し、これらのイベントをユーザーによる承認、訂正または拒否のために、電子メール、Lotus Notes、MS Exchange、UNIX SendMail、Worklist User Interface およびその他の製品にルートすることができます。

注意： オラクル社は、統合ソリューションに採用する Oracle Workflow のインスタンスは、統合フローの管理のみに活用することをお薦めします。

この限定的なアプローチによって、統合ソリューションの中でアプリケーションのワークフローを使用可能にする試みを回避できます。また、これにより、統合するアプリケーションから独立して統合ソリューションをアップグレードできることが保証されます。

Oracle Workflow とアドバンスト・キューイングの間の非同期通信は、ワークフロー AQ API、PL/SQL コールアウト機能のどちらでも実行可能です。

AQ API

Oracle Workflow Standalone Edition リリース 2.5.2 で提供されるワークフロー AQ API は、簡単で、次の条件すべてを満たす 3 つのキューが含まれています。

- 同じオブジェクト型のペイロード（`wf_payload_t`）を保持しています。
- メッセージの保存は許可していません。
- シングル・コンシューマ・キューを採用しています。

ペイロードのオブジェクト型には、キューされたメッセージを特定のフローに関連付けるために Oracle Workflow が必要とする多数の属性があります。オブジェクト型には、Oracle Workflow に固有の名前 / 値一対形式のパラメータを含めるようにフォーマットできるテキスト属性もあります。

キューはシングル・コンシューマであるため、ルーティングは、AQ エージェントを使用してアウトバウンド・メッセージの受信者を決定することはできません。itemtype と相互関係の組合せを使用して、ビジネス・プロセスまたはその範囲を識別します。

3 種類のキューには、次の目的があります。

- WF_OUTBOUND: ワークフロー・エンジンは、フローが外部コールアウトとして定義したステップに到達したときに、このキューにメッセージを配置します。外部プロセスは、WF_Queue.DequeueOutbound または WF_Queue.DequeueEventDetail と呼ばれる PL/SQL パッケージを使用して、キューからメッセージを収集します。
- WF_INBOUND: アウトバウンド・キューにメッセージを配置した後、ワークフロー・エンジンはインバウンド・キューからの応答を待機します。外部プロセスがメッセージを処理したときに、PL/SQL パッケージ WF_Queue.EnqueueInbound を使用してインバウンド・キューに結果を配置します。すると、ワークフロー・エンジンは、その結果を適切なフローに関連付けます。
- WF_DEFERRED: このキューは、バックグラウンドのワークフロー・エンジンまでフローの処理を遅延させるために、フォアグラウンドのワークフロー・エンジンが使用します。

Oracle Workflow のすべてのキュー API は、WF_QUEUE と呼ばれる PL/SQL パッケージに定義されています。API はアカウントに依存しているため、これらのキュー API はすべて同一の Oracle Workflow アカウントから実行する必要があります。

注意： これらの API の使用については、Oracle8*i* アドバンスト・キューイングの概念とテクノロジに関する知識があることを想定しています。

関連項目： AQ API の詳細は、『Oracle8*i* アプリケーション開発者ガイド アドバンスト・キューイング』を参照してください。

キュー API

- EnqueueInbound
- DequeueOutbound
- DequeueEventDetail
- PurgeEvent
- PurgeItemtype
- ProcessInboundQueue

- GetMessageHandle
- Deferred_queue
- Inbound_queue
- Outbound_queue

インバウンド・キューのための開発者 API

次の API を使用すると、開発者は、WF_QUEUE.EnqueueInbound() を使用せずに、メッセージを内部スタックに作成して、インバウンド・キューに書き込むことができます。内部スタックは純粋に記憶領域であるため、スタック上に作成する各メッセージは結局はインバウンド・キューに書き込む必要があります。

効率的なパフォーマンスのために、スタックが大きくなりすぎないようにインバウンド・キューには定期的に書込みを行ってください。

- ClearMsgStack
- CreateMsg
- WriteMsg
- SetMsgAttr
- SetMsgResult

ペイロードの構造

Oracle Workflow のすべてのキューは、データ型 system.wf_payload_t を使用して、所定のメッセージに対するペイロードを定義します。ペイロードには、そのイベントに関する必要な情報すべてが含まれています。system.wf_payload_t の説明は、次のとおりです。

表 10-1

列名	型	説明
ITEMTYPE	Varchar2(8)	イベントの項目の型。
ITEMKEY	Varchar2(240)	イベントの項目キー。
ACTID	Number	ファンクション・アクティビティのインスタンス ID。
FUNCTION_NAME	Varchar2(200)	実行するファンクションの名前。

表 10-1

列名	型	説明
PARAM_LIST	Varchar2(4000)	`value_name=value` 一対のリスト。インバウンドで使用する場合は、項目属性と項目属性値の対で渡されます。アウトバウンドで使用する場合は、ファンクションのすべての属性と属性値の対で渡されます（アクティビティ属性）。
RESULT	Varchar2(30)	オプションのアクティビティ完了結果。考えられる値は、ファンクション・アクティビティの結果の型によって決まるか、またはエンジン標準の結果です。

関連項目：『Oracle8*i* アプリケーション開発者ガイド 基礎編』および『Standard API for an Oracle Workflow 2.5.2』

PL/SQL コールアウト機能

プロセス・ステップは、PL/SQL コールアウトとして定義することができます。プロセス・ステップに到達すると、そのステップに指名されている PL/SQL プロシージャがコールされ、そのステップに指定されているパラメータがコールされたプロシージャに渡されます。プロシージャが完了すると、成功または失敗を示す結果が戻されます。

PL/SQL プロシージャを使用すると、メッセージを AQ キューに作成したり、AQ キューからメッセージを収集することができます。この場合は、次の点を考慮します。

- プロシージャは、Oracle Workflow Builder のヘルプに定義されている規格に準拠している必要があります。（「Standard API for PL/SQL Procedures Called by Function Activities」）
- トランザクションの影響。Oracle Workflow は、フロー・ステップを分割するためのセーブポイントを使用します。AQ メッセージは、セーブポイントトランザクションの一部として作成または収集できます。ただし、作成内容または収集内容はコミットが発行されるまでは表示されません。
- PL/SQL AQ API を使用して AQ キューから JMS メッセージをデキューし、PL/SQL で JMS プロパティとペイロードを変換するには、OMB/OJMS がそのメッセージを AQ にフォーマットする方法を理解する必要があります。（『Oracle Message Broker Administration Guide Release 2.0.1.0 for SPARC Solaris and Windows NT』）

PL/SQL と Java を使用したビジネス・プロセス・インスタンスのインスタンス化

WF_ENGINE と呼ばれる PL/SQL パッケージには、ビジネス・プロセスの作成（wf_engine.CreateProcess）と、そのプロセスを開始（wf_engine.StartProcess）するための 2 つのプロシージャが含まれています。

プロセスが確実に非同期で実行されるためには、フローの最初のステップに関連付けられているコストをワークフロー・エンジンのしきい値よりも大きくする必要があります。

Oracle Workflow の Java インタフェースは、Oracle Workflow と通信するために Java プログラムでコールできる WF_ENGINE および WF_NOTIFICATION パッケージを Java メソッドとして提供しています。Java メソッドは、WF_ENGINE および WF_NOTIFICATION PL/SQL パッケージ・プロシージャとビューを直接参照し、JDBC を経由して Oracle Workflow データベースと通信します。

相互運用性

以前に、Oralce 以外のメッセージ処理テクノロジを使用しているアプリケーションを統合サーバー製品に統合するために使用できるソリューションのタイプを示すために、MQSeries による統合シナリオを使用しました。

そのシナリオはここで効果的に再検証できます。

アプリケーションへのインタフェースが、AQ 以外のメッセージ処理テクノロジに対するメッセージを抽出する場合は、そのメッセージ処理テクノロジをプログラミング言語を使用して Oracle Workflow に直接統合するか、または専用ソフトウェアを使用して間接的に統合するかを決定する必要があります。

Oracle Workflow への統合に関する代替方法は、次のとおりです。

- **プログラム 1:** メッセージ・キューイング製品と Oracle Workflow の両方に特定言語に対する API がある場合は、メッセージ処理テクノロジからメッセージを収集するプログラムを記述して Oracle Workflow にパラメータを渡すことができます。または、新規ワークフローをインスタンス化するプログラムを記述できます。あるいは、Oracle Workflow からコールアウトしてメッセージ処理テクノロジでメッセージを作成できます。
- **プログラム 2:** 相互に通信できる 2 つの言語（一方にはメッセージ処理テクノロジへの API があり、もう一方には Oracle Workflow への API がある）を選択します。
- **プログラムとソフトウェアの複合 :** Oracle Workflow にも対応する API がある言語で API を提供する Procedural Gateway for MQSeries などのソフトウェアを使用します。

第 III 部
リファレンス

第 III 部では、OIS と各種サード・パーティ製品との間の相互運用性に関するソリューションをいくつか説明します。

次の章が含まれます。

- 付録 A「Mercator Enterprise Broker と OIS」
- 付録 B「フロントエンド統合とバックエンド統合」
- 付録 C「自律型ペイロードとポインタ・ペイロード」
- 付録 D「ビジネス・イベントとシステム・イベント」

A

Mercator Enterprise Broker と OIS

Mercator Software 社は、Enterprise Broker を含む一連のエンタープライズ・アプリケーション統合ツールのベンダーです。Enterprise Broker は、Oracle Integration Server とともに使用できる豊富な機能のデータ変換サービスを提供しています。特にレガシー・システム環境における XML 以外のデータ変換に有効です。この付録では、Oracle Integration Server とともに Mercator Enterprise Broker を使用する方法の概要を説明します。次の項目が含まれています。

概要

アプリケーション間の統合を実現する Enterprise Broker は、データとメッセージの変換に特に有用です。ソフトウェアは、1 つ以上の変換ステップによってオブジェクトがある形式または状態から別の形式または状態に変換されるような変換の流れを中心にして設計されています。

各変換ステップには、次のプロパティがあります。

- トリガーまたはドリガー・イベント
- 1 つの入力オブジェクトまたは複数オブジェクト
- 変換ルール
- 1 つの出力オブジェクトまたは複数オブジェクト

Enterprise Broker 自体は、4 つのグラフィック・ベースのエディタで構成されています。

- システム・エディタ
- タイプ・ツリー・エディタ
- データベース・エディタ
- マッピング・エディタ

システム・エディタ

システム・エディタは、変換の流れを定義します。システム・エディタは、ある形式または状態から別の形式または状態にメッセージを変換する複数のステップをモデル化する多数の変換ステップにリンクしています。

システム・エディタを使用すると、変換を促進するトリガーまたはトリガー・システムのイベントを定義できます。一般的なトリガーには、次のものがあります。

- キューへのメッセージの着信
- ディレクトリへのファイルの作成
- 表への行の挿入
- ファイルの最終変更タイムスタンプの変更

タイプ・ツリー・エディタ

Mercator Software 社の製品は、タイプ・ツリーと呼ばれる構造体を使用して、外部オブジェクトのデータ構造を内部的に表現します。タイプ・ツリーの形式は、次の 3 つのいずれか 1 つになります。

- **事前定義 :** Mercator は、一般的な標準メッセージ・フォーマット（EDI や HL7 など）および一般的な ERP パッケージ（SAP/R3 など）のパブリック・インタフェース・オブジェクトを示す一連の事前定義のタイプ・ツリーを提供しています。
- **インポート :** データベース・エディタを使用して、大半のリレーショナル・データベースのデータベース・オブジェクトを示すタイプ・ツリーを作成できます。
- **定義可能 :** タイプ・ツリー・エディタを使用して、タイプ・ツリーを手動で定義できます。

タイプ・ツリーは、変換への入力を提供する各ソースごと、および変換からの出力を受け取る各ターゲットごとに定義する必要があります。

データベース・エディタ

データベース・エディタを使用して、リレーショナル・データベースの所定のデータベース・オブジェクトに対するタイプ・ツリーを作成します。このエディタは、データベース・ユーザーとしてデータベースに接続する必要があります。この結果、エディタは、そのユーザーに対して定義されているオブジェクトの権限、許可およびビューを持ちます。

Oracle では、次のオブジェクトに対してタイプ・ツリーを作成できます。

- **表 :** タイプ・ツリーでは、列はフィールドになります。
- **プロシージャ :** `IN`、`OUT` および `IN/OUT` パラメータはフィールドになります。
- **キュー :** キューのペイロードがオブジェクト型として定義されている場合、各属性はタイプ・ツリーでフィールドになります。ペイロードが `RAW` 型の場合、タイプ・ツリーは、単一のバイナリ文字列（`BLOB`）で示されます。

マッピング・エディタ

マッピング・エディタを使用して入力カード、出力カードおよびマップを定義し、個々の変換ステップを構成します。

変換ステップは、製造プロセスの 1 つのフェーズに似ています。

- **入力カード**には、サプライヤと原材料の仕様を定義します。
- **マップ**には、製造プロセスを定義します。
- **出力カード**には、製品仕様と顧客の搬送先所在地を定義します。

入力カード

入力カードには、次の内容を定義します。

- ソースへの接続のタイプ、トランザクション動作、フェッチ当りの行数およびリトライ・ルールなどのコレクションに関するルール
- タイプ・ツリーのすべてまたは一部の仕様による着信メッセージのフォーマット

単一のステップに複数の入力カードを定義でき、入力カードは複数回反復することができます。たとえば、受注の入力カード、および受注明細の入力カードを定義します。各受注ごとに、受注明細の反復が複数存在します。

出力カード

出力カードには、次の内容を定義します。

- 宛先への接続のタイプ、トランザクション動作、プッシュ当りの行数およびリトライ・ルールなどの送信に関するルール
- タイプ・ツリーのすべてまたは一部の仕様による送信メッセージのデータ・フォーマット

単一のステップ内に複数の出力カードを複数回反復して定義できます。

マップ

マップは、入力カードと出力カードを相互に結び付けるメカニズムです。マップには、あるフォームまたは状態から、別のフォームまたは状態にデータを変換するルールを定義します。定義済みデータのサブセット上で、特定のファンクションを実行するために定義されたマップは、**機能マップ**と呼ばれ、他のマップ内にネストできます。

マップの基本的な目的は、入力カードから出力カードにデータを変換することです。マップは、多様な方法でこれを実現します。

- 入力カードからフィールドをドラッグして、出力カードのフィールドにドロップする。
- 出力カードのフィールドに静的な値を定義する。
- 出力カードのフィールドを NULL で定義する。
- 出力カードのフィールドを日付時刻タイムスタンプなどのシステム値に設定する。
- 入力カード・フィールドをプロシージャ・ロジック、ファンクションまたは SQL 参照に指定し、結果を出力カードに適用する。

Enterprise Broker エンジン

エディタを使用して変換プロセスを構成した後、エンジンを使用して変換ステップを実行します。これらのステップは、コマンドラインから、システム・エディタから、または C または Java で記述されたプログラム内から（Mercator C と Java API を介して）インスタンス化できます。

エンジンは、各変換ステップに関する情報を取得するロギング機能と追跡機能を提供しています。この機能は、エラーのデバッグに役立ちますが、パフォーマンス・オーバーヘッドを招きます。

このエンジンには、拡張性を高める多数のパフォーマンス改善機能もあります。

- エンジンは、大きなメッセージをデータ・ストリームとして処理します。これにより、エンジンは、メッセージ全体の到着を待たずに、メッセージの処理を開始できます。
- エンジンは、マップとデータベース接続を再利用します。
- エンジンは、ロギングと追跡をオフに切り替えます。

Oracle Integration Server における Enterprise Broker の使用

オラクル社は、Mercator Enterprise Broker を特に複雑な変換に対する変換エンジンとして使用することをお薦めします。XML を別のフォーマットに、または別のフォーマットを XML に変換する必要がある場合は特に有効です。

入力カードと出力カードが同じ接続文字列を使用している場合は、AQ アダプタの実装によって、同じデータベース内での AQ キュー間の変換が簡単になります。この実装によって、Oracle 同期コミット・モデルはデキュー − 変換 − エンキューのプロセスを単一のトランザクションとして送信できるため、メッセージの保存が保証されます。

データ変換をプロセス内の個別のステップとして定義することは、オラクル社がお薦めするコンポーネント・ベースのアプローチと整合性があります。

ヒント

- Mercator Enterprise Broker を使用すると、AQ と MQSeries の間でメッセージを変換できます。ただし、メッセージが失われないようにトランザクションを追跡する必要があります。
- AQ アダプタは、CLOB および正規の Oracle データ型をサポートしていますが、メッセージの構成前にバイナリ文字列の長さを決定する必要があるため、BLOB/RAW AQ キューへのマッピングは慎重に行う必要があります。

注意： Enterprise Broker リリース 2.1 には、JMS メッセージの解析を支援する機能は含まれていません。メッセージの解析前に、OJMS と OMB AQ ドライバの相互関連を理解しておく必要があり、JMS メッセージのタイプ・ツリーは慎重に定義する必要があります。

B

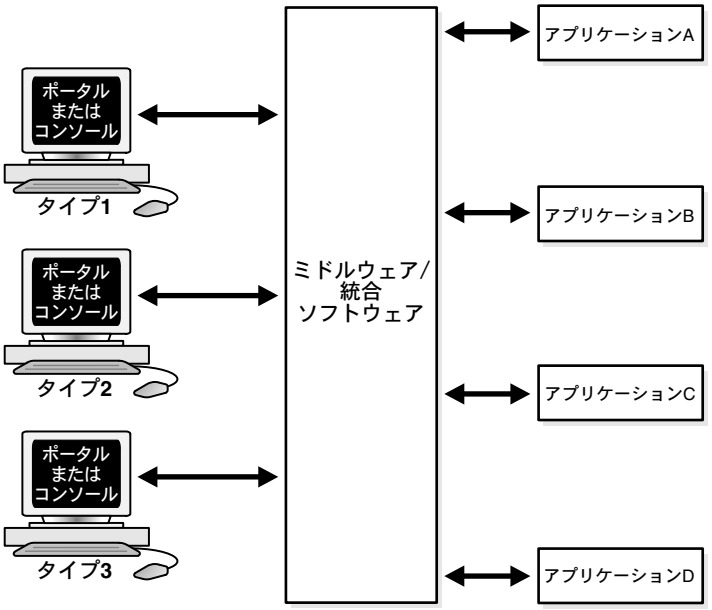
フロントエンド統合とバックエンド統合

フロントオフィス統合およびバックオフィス統合に対する要件は、根本的に異なっています。したがって、基本的に異なる統合アプローチが必要です。次の項目が含まれています。

フロントエンド統合

この実装では、ミドルウェアまたは同種の統合ソリューションがアプリケーション間の対話をインターセプトします。ミドルウェアは、アプリケーション間における要求と応答を透過的に整理、管理および指示するための中間層アプリケーション・サーバーとして動作します。一般的に、要求側のアプリケーションは、ミドルウェアが別のアプリケーションのサービスを要求している間は待機しています。

図 10-1 フロントエンド統合



フロントエンド統合ソリューションには一般的に、次の一部またはすべての特徴があります。

- 統合レイヤーとアプリケーション間の対話は、次のようになります。
  - 同期、または同期処理をシミュレートします。
  - サービス要求モデルに基づく双方向通信です。
- トランザクション処理の所要時間は短時間です。
- アプリケーション（ユーザー）は、本質的に異なる複数のアプリケーション・サービスを使用していることを認識しません。

- ミドルウェアは、中間層アプリケーション・サーバーのように、要求と応答を整理、管理および指示し、トランザクションとリカバリの問題を管理します。

フロントエンド統合には長所と短所があります。

長所

- 異なるユーザー、顧客およびサプライ・チャネルに対して、多数のプレゼンテーション・レイヤーを開発できます。
- すべてのプレゼンテーション・レイヤーが同じサービスを共有することにより、チャネルに関係なく一貫した対話を確保できます。
- 必要な機能すべてにアクセスするために、各プレゼンテーション・レイヤーに必要なサインオンは 1 回のみです。

短所

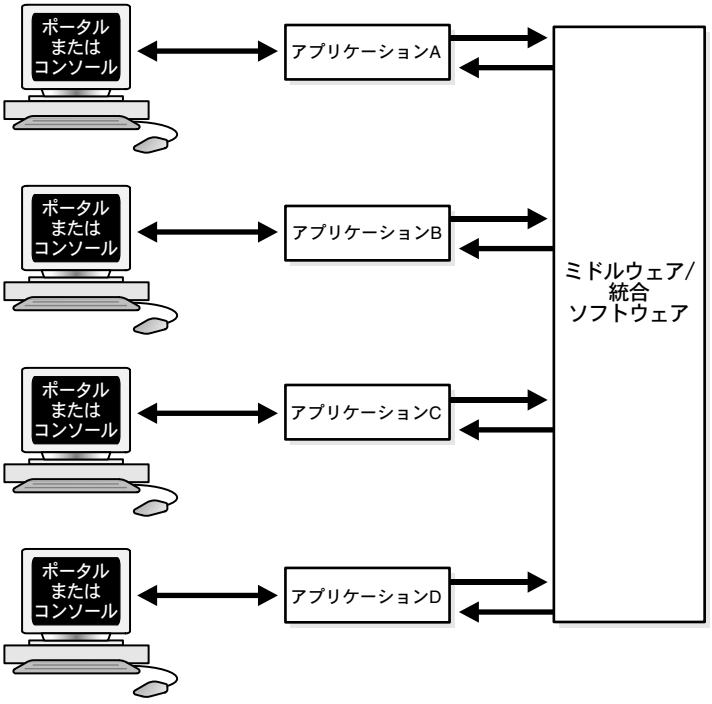
アプリケーションが対話に加わるためには、次の基準に適合している必要があります。

- すべての機能は、サービスベースの API を使用して提供されることが必要です。
- すべてのビジネスおよびデータ・ロジックは、プレゼンテーション・レイヤーから独立している必要があります。
- トランザクション / 同期コミット・モデルは、選択された複数フェーズのコミット処理では、リソース・マネージャとして動作する必要があります。
- レガシーおよび現時点で使用可能な ERP/CRM パッケージで、これらの基準に適合するものは少数です。したがって、カスタム・アプリケーションの開発が必要になる場合があります。
- パッケージ・アプリケーションのプレゼンテーション・レイヤーは使用できない場合があります。したがって、サポートの提供が困難であり、実装期間が長期に及ぶようになるなどの可能性があります。
- 複雑なアーキテクチャの必要性、適切で強力なツールの不足、およびプログラム指向の送信への依存のため、開発環境とテスト環境の管理は困難です。

バックエンド統合

この実装では、ユーザーは、自分のユーザー・ロールで決定した方法で 1 度に 1 つのアプリケーションのみと対話します。アプリケーションは、必要に応じて、ユーザー対話の重要な局面を他のアプリケーションに通知します。アプリケーション間の対話は、さらに遂次的な通知方法に基づくことができます。

図 10-2 バックエンド統合



バックエンド統合ソリューションには、次の一部またはすべての特徴があります。

- 統合レイヤーとアプリケーション間の対話は、次のようになります。
  - 主に非同期です。
  - 一方向通信です。
- サービス要求とイベント通知モデルの両方が使用されます。

- ビジネス・プロセス定義はビジネスを厳密に反映するため、トランザクションに時間がかかります。
- ユーザーは単一のアプリケーションまたは少数のアプリケーションと対話します。
- この統合ソリューションは、分散して同期をとる高度なエージェントとして透過的に動作します。

長所

- アプリケーションへのプレゼンテーション・レイヤー・インタフェースは、そのアプリケーションと同じアーキテクチャを採用する必要はありません。
- パッケージ・アプリケーションと同梱の組込みプレゼンテーション・レイヤーを使用できます。
- アプリケーション内のビジネスとデータ・ロジックは、配布されているプレゼンテーション・レイヤーから分離する必要はありません。
- 非同期のブレーク・ポイントが適切なテスト・ポイントを提供するため、開発環境とテスト環境の管理が容易です。
- 開発を容易にするグラフィカルなツールが準備されています。

短所

- 各アプリケーションには、それぞれ異なる操作性とサインオンがあるため、アプリケーションは統合が不十分のように見えます。
- ユーザー・インタフェースが狭い範囲に制限されているため、ユーザーは複数の異なるロールで動作するように限定されます。

C

自律型ペイロードとポインタ・ペイロード

メッセージのペイロードの情報量は、メッセージ設計の観点から非常に重要です。この付録では、ペイロード・サイズの操作に関する 2 つの異なる方法、およびその折衷案の実装を説明します。

折衷案による実装も 1 つの方法とした場合、次の 3 つの方法からペイロードを選択できます。

- ポインタ・ペイロード
- 自律型ペイロード
- ハイブリッド・ペイロード

ポインタ・ペイロード

この方法に基づくペイロードには、他のアプリケーションがイベントに関する詳細情報をメッセージのパブリッシャから要求するのに必要な情報のみが含まれています。大量のデータを送信することは現実的ではないため、メッセージ・サイズを制限することが重要な場合に、この方法のペイロードを使用します。

このペイロード・スタイルが適切である環境には、次のような特徴があります。

- 永続的に保護されている他のフォームにコピーがあるため、ペイロードを即時に記録する必要がない環境。
- 必要に応じて顧客に情報を送信できる環境。
- コンシューマがデータを処理した後、そのデータの永続的なコピーが不要な環境。
- ルーティング、追跡またはビジネス・インテリジェンスのための情報が不要か、または他に使用可能な情報がある環境。

例 1: ビデオ・フィルム

ビデオ・オン・デマンドでは、データは純粋なバイナリのデジタル・データ・ストリームのため、全データをメモリーまたはディスクに記録する機会がほとんどありません。最終宛先に到達する前のデータ解析で得られものはありません。

また、顧客がデータ・ストリームの最初の部分のみを実際に要求する可能性が強い場合は、全データの送信は不要です。

例 2: 名前とアドレスのデータベースへの変更

ネットワークの通信量が多く、ディスク I/O が激しい場合、名前とアドレスの変更内容は通常、操作したシステムにその日中に記録されます。

変更が発生したことを操作した以外のシステム（マーケティング・データベースなど）に通知することによって、通知を受けたデータベースでは、影響のあるデータをリアルタイムで廃止としてマークできます。後で、ネットワークとディスクの使用率が低くなったときに、変更の詳細によってデータを更新できます。

自律型ペイロード

このペイロードの方法には、メッセージの作成者に照会せずにメッセージを処理できる十分な情報が含まれています。自律型ペイロードは、アプリケーションの相互依存性を低下させます。パブリッシュ・サブスクライブ・ルーティングの配置が必要な場合、この方法は必須です。

次のようなビジネス・イベント機能に関係した適切な情報すべてが含まれています。

- コンテンツ・ベースのルーティング
- 完全な監査証跡
- ビジネス・インテリジェンスの使用
- メッセージの追跡

非同期で一方向のメッセージ処理環境で、ペイロードが比較的小さく（2MB 未満）、メッセージがほぼリアルタイムで処理されている場合は、自律型ペイロード方法の使用が自明の選択です。

例 1: 取引の共有

顧客の株式ポートフォリオを管理している銀行では、売買要求の詳細を監査証跡に確実に獲得することが必要です。取引に関する情報が比較的小量の場合は、その情報を数秒内に取引市場に配置する必要があり、ペイロードに含まれている情報に基づいて、別の取引市場にルートすることが必要です。

例 2: 在庫管理

倉庫の在庫管理アプリケーションは、日中は受注アプリケーションをサブスクライブし、夜間は獲得したイベントを使用して再発注レベルを決定します。メッセージは比較的少量です。在庫管理アプリケーションは、受注アプリケーションとは切り離した操作が必要で、通常の営業時間中にのみ操作できます。

ハイブリッド・ペイロード

この方法は、ペイロードには、他のアプリケーションがイベントに関する詳細情報をメッセージのパブリッシャから要求するのに十分な情報が必要であるという点では、ポインタ・ペイロードと似ています。メッセージには、イベントで処理が必要かどうかをサブスクライバが判断するための十分な情報が含まれていることも必要です。

ハイブリッド・ペイロードは、メッセージが大きく、サブスクライバが受信後にメッセージのフィルタ処理を行う場合に有効です。メッセージ・ペイロードの一部は、統合機能に対応しています。たとえば、コンテンツ・ベースのルーティングが必要かどうかが示されます。

例 : マーケティング

電子メール・ブロードキャスト配布には、提供するビデオ・クリップ、デジタル画像および販売製品の完全な詳細を含んだ製品パックをダウンロードしたいと顧客に思わせるのに十分な情報（販売価格、販売日付）が含まれている必要があります。

メール送信のヒット率を低く予想している場合は、初期のブロードキャストで完全パックを送信することは、非効率です。ただし、Web サイトへのポインタのみを顧客に送信しても、製品に対する顧客の関心を刺激するには不十分です。

D

ビジネス・イベントとシステム・イベント

ビジネス・イベント

ビジネス・イベントとは、ビジネス・シナリオ内に定義可能な状態変化です。状態変化には、顧客による発注などの一般的な高レベルの状態変化もあります。また、顧客が発注する際のクレジット制限の超過など、さらに特化したイベントの場合もあります。イベントの中には、特定な株の株価が指定した値を超えるなど、値の変更によって実行されるイベントもあります。

ビジネス・イベントでは、ビジネスの様々な部署がそれぞれのアプリケーションを使用して登録および実行する必要がある、重要な出来事が定義されます。Oracle Integration Server は、ビジネス・イベントを登録および実行するすべてのアプリケーションを統合しています。

次の項目が含まれています。

システム・イベント

システム・イベントとは、プログラム実行中のある時点で、識別可能な計算タスクを開始する時を指します。システム・イベントの例には、ファイルへの書込み、データベース表へのデータの挿入、サブルーチンのコール、プログラム・スレッドのインスタンス化などがあります。

特定のビジネス・プロセスを実装するプログラムには通常、多数のシステム・イベントが含まれており、これら複数のシステム・イベントは、特定のビジネス・イベントを獲得するための適切なトリガーを提供することができます。

例 : 受注の発生

多くの統合アプリケーションには、ビジネス・イベント、`Raise an Order` が含まれています。アプリケーションのいくつかは、発注が行われたことを知る必要があるため、このようなイベントをサブスクライブする必要があります。発注を獲得する別のアプリケーション・プログラムには、次のシステム・イベントを実行するためのプログラム・ステップがあります。

- 記帳日の取得
- 顧客住所の取得
- 受注合計の計算
- 受注ヘッダーへの挿入

これらの各システム・イベントは、受注ごとに発生します。どのシステム・イベントも 'Raise an Order' ビジネス・イベントを実行するトリガーとなる適切な時点を提供できます。

ビジネス・イベントとシステム・イベントの相違

ビジネス・イベントとシステム・イベントの間には、3 つの重要な相違点があります。

- ビジネス・イベントは、ビジネス・シナリオ内の論理的な状態変化です。システム・イベントは、ビジネス・イベントの発生を認識するために使用するメカニズムです。
- ビジネス・イベントは、アプリケーション内での実装方法に影響されません。システム・イベントは、その実装の一部です。
- ビジネス・イベントを示すメッセージは、そのビジネス・イベントに関する情報を提供するように構成されている必要があります。この情報は、そのビジネス・イベントのトリガーとなるシステム・イベントに関する情報ではありません。

統合ソリューションでビジネス・イベントを強調するには、メッセージ・ベースの方法を使用します。システム・イベントを強調するには、メッセージ処理テクノロジを使用してデータ・レプリケーションを実装する方法を使用します。

例 : システム・イベントの強調

受注処理を統合するには、PlaceanOrder と呼ばれるビジネス・イベントを識別します。CallCreateOrderHeader という名のシステム・イベントがある場合は、コールの直前または直後にメッセージの作成をインスタンス化できます。

作成したメッセージには、受注明細、顧客の詳細（必要な場合）、受注合計、および CreateOrderHeader サブプログラムの一部として作成された受注ヘッダーの一部またはすべての情報を含んだ受注に関するすべての情報が含まれています。

次に、このメッセージは、Place an Order サブジェクトまたはトピックのインスタンスとして、ミドルウェアに公開されます。ミドルウェアは、PlaceanOrder ビジネス・イベントをサブスクライブしているアプリケーションにこのメッセージをルーティングする責任を負います。

例 : ビジネス・イベントの強調

RegistrationofaNew Customer と呼ばれるイベントを含んだ、顧客との対話に関連している一連のビジネス・イベントを識別します。

新規顧客の登録時に、必ず発生するシステム・イベントを識別します。このシステム・イベントは、カスタマ・リレーションシップ・マネジメント（CRM）・アプリケーションの customer 表への挿入行です。

このシステム・イベントは、customer 表に行が挿入されたときに開始するデータベース・トリガーを作成して、ビジネス・イベントを発生させることができます。データベース・トリガーは、新規顧客の登録を示すメッセージを組み立てる PL/SQL プロシージャまたは Java ファンクションをコールします。このコールは、address 表などの他のリレーショナル表から情報を収集し、信用評価の計算などの他のファンクションをコールして行われます。

索引

L

M

N

O

P

Q

R

S

T

と

な

に

ね

は

ひ

ふ

へ

ほ

ま

め

も

ゆ

よ

り

る

れ

ろ

わ