

# Caché 入門

Version 5.1  
2006-03-14

## Caché 入門

Caché Version 5.1 2006-03-14

Copyright © 2006 InterSystems Corporation.

All rights reserved.

このドキュメントは、Sun Microsystems、RenderX Inc.、アドビ システムズ および ワールドワイド・ウェブ・コンソーシアム (www.w3c.org) のツールと情報を使用して、Adobe Portable Document Format (PDF) で作成およびフォーマットされました。主要ドキュメント開発ツールは、InterSystemsが構築したCaché と Javaを使用した特別目的のXML処理アプリケーションです。



Caché 製品とロゴは InterSystems Corporation の登録商標です。



Ensemble 製品とロゴは InterSystems Corporation の登録商標です。



InterSystems という名前とロゴは InterSystems Corporation の登録商標です

このドキュメントは、インターシステムズ社(住所: One Memorial Drive, Cambridge, MA 02142)あるいはその子会社が所有する企業秘密および秘密情報を含んでおり、インターシステムズ社の製品を稼動および維持するためにのみ提供される。この発行物のいかなる部分も他の目的のために使用してはならない。また、インターシステムズ社の書面による事前の同意がない限り、本発行物を、いかなる形式、いかなる手段で、その全てまたは一部を、再発行、複製、開示、送付、検索可能なシステムへの保存、あるいは人またはコンピュータ言語への翻訳はしてはならない。

かかるプログラムと関連ドキュメントについて書かれているインターシステムズ社の標準ライセンス契約に記載されている範囲を除き、ここに記載された本ドキュメントとソフトウェアプログラムの複製、使用、廃棄は禁じられている。インターシステムズ社は、ソフトウェアライセンス契約に記載されている事項以外にかかるソフトウェアプログラムに関する説明と保証をするものではない。さらに、かかるソフトウェアに関する、あるいはかかるソフトウェアの使用から起こるいかなる損失、損害に対するインターシステムズ社の責任は、ソフトウェアライセンス契約にある事項に制限される。

前述は、そのコンピュータソフトウェアの使用およびそれによって起こるインターシステムズ社の責任の範囲、制限に関する一般的な概略である。完全な参照情報は、インターシステムズ社の標準ライセンス契約に記載され、そのコピーは要望によって入手することができる。

インターシステムズ社は、本ドキュメントにある誤りに対する責任を放棄する。また、インターシステムズ社は、独自の裁量にて事前通知なしに、本ドキュメントに記載された製品および実行に対する代替と変更を行う権利を有する。

Caché および InterSystems Caché、Caché SQL、Caché ObjectScript および Caché Object は、インターシステムズ社の商標です。

ここで使われている他の全てのブランドまたは製品名は、各社および各組織の商標または登録商標です。

インターシステムズ社の製品に関するサポートやご質問は、以下にお問い合わせください:

InterSystems ワールドワイド カスタマサポート

Tel: +1 617 621-0700

Fax: +1 617 374-9391

Email: support@InterSystems.com

# 目次

1 Caché とは? .....	1
1.1 特有のアーキテクチャ .....	2
1.2 ポスト・リレーショナルとは? .....	2
1.3 Caché の事例 .....	3
1.4 インターシステムズとの連絡先 .....	4
2 Caché データベース・エンジン .....	5
2.1 トランザクション多次元ストレージ .....	5
2.1.1 オブジェクトと多次元ストレージ .....	6
2.1.2 柔軟性 .....	6
2.2 プロセス管理 .....	7
2.3 分散データ管理 .....	8
2.4 ジャーナル管理 .....	9
2.5 ロック管理 .....	9
2.6 デバイス管理 .....	10
2.7 可搬性 .....	10
2.8 アプリケーション稼動オプション .....	10
2.8.1 基本的なクライアント/サーバ構成 .....	10
2.8.2 シャドウ・サーバ構成 .....	11
2.8.3 多層構成 .....	12
3 オブジェクト、SQL、統一データ・アーキテクチャ .....	13
3.1 統一データ・ディクショナリ .....	13
3.1.1 柔軟なストレージ .....	15
3.2 オブジェクト .....	15
3.2.1 クラス定義 .....	16
3.3 SQL .....	17
3.3.1 オブジェクト/リレーショナル接続 .....	18
3.3.2 継承と SQL .....	19
3.3.3 SQL へのオブジェクト拡張 .....	20
4 開発ツールとデータベース・ユーティリティ .....	21
5 接続性 .....	23

# 図一覧

永続オブジェクトのストレージ .....	6
ストレージ・インデックス .....	6
プロセス管理 .....	7
ECP (Enterprise Cache Protocol) .....	8
クライアント/サーバ構成 .....	11
シャドウ・サーバ構成 .....	11
多層構成 .....	12
統一データ・アーキテクチャ .....	14

# テーブル一覧

オブジェクト機能のリレーショナル表 .....	18
Person クラスの SQL ビュー : SELECT * FROM Person .....	19
Employee クラスの SQL ビュー : SELECT * FROM Employee .....	19
Person クラスの改訂 SQL ビュー : SELECT * FROM Person .....	20



# 1

## Caché とは?



Caché ポストリレーショナル・データベースへようこそ。

ここでは、Caché を構成している主要コンポーネントとテクノロジーの概要を説明しています。コンポーネントは、以下の通りです。

- ・ 分散データベース生成機能が組み込まれた、強力で多次元のトランザクション・エンジン
- ・ 強力なオブジェクトと高性能な SQL を結合した統合データ・アーキテクチャ
- ・ 高速なデータベースと Web アプリケーション開発を促進するテクノロジーとツール
- ・ オブジェクト指向に基づく、ネイティブな XML、Web サービスのサポート Caché XML 機能の詳細は、“Cachéでの XML の使用法”を参照してください。
- ・ EJB、JDBC、ActiveX、.NET、C++、ODBC、XML、SOAP などとの自動的な相互運用性

Caché についてさらに学習するには、残りの Caché マニュアルをお読みください。また、Caché テクノロジー・ガイド には、Caché の機能とアーキテクチャの概要を説明しています。特定のトピックを詳細に学習するには、ホーム・ページのオンライン・ドキュメントで閲覧可能な他のマニュアルを参照し

てください。また、Caché には、さまざまな開発やシステム管理に関するオンライン・チュートリアルが多数あります。

# 1.1 特有のアーキテクチャ

Caché のパワーの多くは、その独自のアーキテクチャに由来しています。その中心となる Caché データベース・エンジンは、複雑なデータベース管理システムの構築に必要なサービス一式を提供します。例えば、データ・ストレージ、同時実行管理、トランザクション、プロセス管理などです。Caché エンジンを強力なデータベース・ツールキットとして考えることができます。これにより、Caché は、完全なオブジェクトとリレーショナル・データベースの管理システムを実装します。

- ・ オブジェクトとリレーショナル・データベース・システムは、データベース・エンジンと直接会話するため、非常に効率的な処理を実行します。オブジェクト・リレーショナル・ミドルウェアや、SQL とオブジェクト間のブリッジ・テクノロジーは不要です。
- ・ 物理的な実装からデータベースが論理的に分離されているため、アプリケーション・ロジックを変更せずに、アプリケーションの設定を素早く再構成できます。
- ・ データベース・エンジンのインタフェースはオープンであるため、必要に応じて機能を直接使用できます。この機能とは、独自にカスタマイズしたデータベース管理システムの構築から、パフォーマンスに重点を置いたアプリケーションの最適化の追加まで広範囲に渡ります。
- ・ 将来を見通したプラットフォーム： Caché アーキテクチャでは、既存のアプリケーションに影響を与えずに、データベース・エンジンの今後の拡張に対応します。たとえば、拡張性とパフォーマンスが大幅に向上した Caché バージョン 4.1 を新規の物理データ構造に導入した場合、既存のアプリケーションを変更したり、Caché オブジェクトやリレーショナル・システムを変更したりする必要はありません。また、Caché は、XML など新規のテクノロジーをサポートします。ここでも、既存のアプリケーションに影響せずに、固有の高性能コンポーネントとして導入できます。

# 1.2 ポスト・リレーショナルとは？

Caché は、リレーショナル・モデルの限界を超えるように設計されており、現存する何千ものリレーショナル・データベース・アプリケーションへの最新のアップグレード・パスを提供したり、市場にある SQL ベースのリポート・ツールをサポートします。

“ポスト・リレーショナル” の “リレーショナル” とは、Caché がフル機能のリレーショナル・データベースであることを示しています。Caché データベースのすべてのデータを実際のリレーショナル・テーブルで使用でき、ODBC、JDBC、オブジェクト・メソッド経由の標準 SQL を使用して、問い合わせや修正ができます。Caché の基礎となる強力なデータベース・エンジンにより、Caché は、今日、最速で最も拡張性のあるリレーショナル・データベースです。

また、“ポスト・リレーショナル”の“ポスト”とは、Caché が、リレーショナル・データベースの限界を超える機能を持ち、さらに標準のデータ・リレーショナル・ビューをサポートしていることを示しています。その機能は、以下の通りです。

- ・ オブジェクトとしてデータをモデリングする機能 (各オブジェクトは、自動的に生成、同期化されたネイティブのリレーショナル表現を持ちます)。これは、データベースとオブジェクト指向アプリケーション環境のインピーダンス不整合を取り除き、リレーショナル・モデリングの複雑さを軽減します。
- ・ より簡単なオブジェクト単位の同時実行モデル
- ・ ユーザ定義のデータ型
- ・ データベース・エンジンで、多態を含むメソッドと継承を活用する機能
- ・ オブジェクト識別とリレーションシップを操作するための SQL へのオブジェクト拡張
- ・ 単一アプリケーションで、SQL とオブジェクト・ベース・アクセスを組み合わせ、適宜使用する機能
- ・ アプリケーションの性能を最大にするように、データの格納に使用する物理的階層とクラスタリングを管理

オブジェクトとリレーショナルの両方へアクセスできる大半のデータベースは、他方の上位にその一方へのアクセス形式を提供しますが、Caché の SQL とオブジェクトは、双方向へ直接データを渡します。この二方向のダイレクト・アクセスは、Caché のポスト・リレーショナルの利点です。

## 1.3 Caché の事例

Caché は、シングルユーザ向けのシステムから何万の同時ユーザを持つ企業規模のマルチユーザ・システムまで、世界中で使用されているアプリケーションです。

以下は、Caché で構築されたシステムの事例です。

- ・ 何百人もの患者を処理するアプリケーションが稼動している大規模な健康医療ネットワーク向けのアプリケーション・プラットフォームとして。ネットワークには、データ、アプリケーション・サーバ、3 万以上のクライアント・マシンが動作する Caché システムが含まれます。
- ・ 大手金融機関向けの Java ベースの企業メッセージ・システムのデータ・サーバとして。Caché は、パフォーマンスと、従来のリレーショナル・データベースでは不可能であったカスタマイズされたタスクを実行できる機能の両方から選択されました。
- ・ 1400 以上の同時ユーザを持つ大規模な政府組織向け SQL ベースの OLTP (オンライン・トランザクション処理) システムとして。Caché は、他のリレーショナル製品では不可能なことを、(アプリケーションを変更せずに) 実行するために導入されました。

- ・ 一流の工科大学で使用されるオンライン教育システム向けのオブジェクト・データベースと Web アプリケーション・フレームワークとして。Caché は、迅速な開発 (アプリケーション構築に 3ヶ月)、オブジェクトの性能、アプリケーションの再作業を行わずに拡張できる機能から選択されました。
- ・ 世界選手権で、プロ選手のリアルタイムの順位や走行タイムを追跡するために使用するオブジェクト・データベースとして。Caché は、(代表的なオブジェクトとリレーショナル・データベースに比較して) そのパフォーマンスとネイティブ C++ インタフェースから選択されました。
- ・ 数百万のユーザがアクセスする人気のある Web サイト用の分散 SQL データ・エンジンとして。このサイトは、費用対効果のある Linux ベース・サーバと Caché の分散データ管理を使用して、スケーラブルでかつパーソナライズされたサイトを提供します。ミドルウェアや Web キャッシュ・インフラストラクチャはありません。このシステム (市販の 4 つの Linux マシン) のハードウェア費用は、“インターネット・アプリケーションで代表的なデータベース” と比較すると、10% 以下で済みます。

## 1.4 インターシステムズの連絡先

インターシステムズの製品のサポートに関する質問については、販売代理店もしくはインターシステムズのサポート窓口までお問い合わせください。

- ・ 電話番号 : 0120-17-1972
- ・ 電子メール : [jpnsup@intersystems.com](mailto:jpnsup@intersystems.com)
- ・ Web : <http://www.intersystems.co.jp>

# 2

## Caché データベース・エンジン

Caché データベース・エンジンは、Caché の心臓部分です。データベース・エンジンは、パフォーマンス、同時実行、スケーラビリティ、信頼性を最適化します。サポートされるプラットフォームによって最大限のパフォーマンスを獲得するために、プラットフォーム特有の高度な最適化が行われています。

Caché は、フル機能のデータベース・システムです。基幹アプリケーション (ジャーナリング、バックアップ、リカバリ、管理ツールなど) を実行するために必要なすべての機能が含まれています。処理負荷を軽減するために、Caché は、他のデータベース製品よりデータベース管理が簡単になるように設計されています。したがって、導入された大半の Caché システムでは、データベース管理者は必要ありません。

データベース・エンジンの主機能は、以下のセクションで説明しています。

### 2.1 トランザクション多次元ストレージ

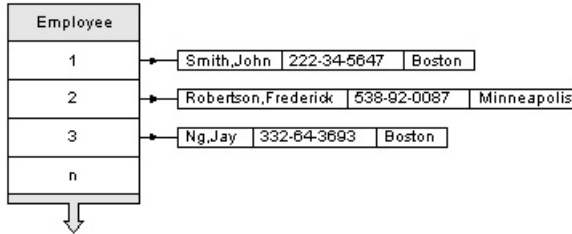
Caché のすべてのデータは、まばらな多次元配列に格納されます。標準の OLAP (オンライン解析処理) 製品で使用される多次元配列と異なり、Caché は、多次元構造で、トランザクション処理操作 (挿入、更新、ロック、トランザクション) をサポートします。また、大半の OLAP エンジンと異なり、多次元構造は、使用可能なメモリ・サイズに制限されません。代わりに、Caché には、高度で効率的なデータ・キャッシュがあります。

Caché は完全可変長データをまばらな配列の形で格納するため、一般的なリレーショナル・データベースの半分程度の格納領域しか必要としません。ディスク容量が少なくなくて済むばかりでなく、コンパクトなデータ保存によって、一度の I/O 処理でより多くのデータの読み書きが可能となり、データがより効率良くキャッシュされるため、パフォーマンスが向上します。

## 2.1.1 オブジェクトと多次元ストレージ

Caché オブジェクトは、そのオブジェクト永続テクノロジーの基盤として多次元ストレージを使用します。例として、名前、ID 番号、勤務地の従業員データを持つ **Employee** クラスがあります。Employee オブジェクトのインスタンスは、ここで図示されているように、多次元配列で格納されます。

永続オブジェクトのストレージ



この場合、配列はオブジェクト識別子の値で添え字を付け、各インスタンスのデータは、配列ノードに簡潔に格納されます。Caché オブジェクトは、永続クラスに最適なストレージ構造を自動的に生成します。

**Location (勤務地)** プロパティのようにクロス・インデックスが必要な場合、永続エンジンは異なる多次元配列を使用し、以下の図のように、対応するオブジェクト識別子に勤務地の値を関連付けます (通常、二次元構造で実行されます)。

ストレージ・インデックス



これらのインデックスは、データベースが変更されると自動的に維持されます。Caché SQL エンジンには、このようなインデックスを自動的に使用して、勤務地で Employees を検索するクエリを実行します。

## 2.1.2 柔軟性

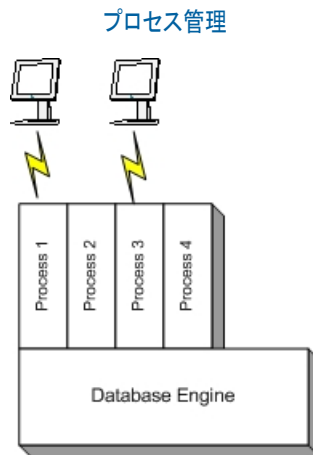
多次元配列により、データを格納する際のアプリケーションの柔軟性が大幅に向上しました。たとえば、密接に関連するオブジェクト・セット、**Invoice** オブジェクトとそれに対応する **LineItem** オブジェクトの構造、**LineItem** オブジェクトは **Invoice** オブジェクトに物理的にクラスタされるように簡単に構成できるため、かなり効果的にアクセスできます。

“添え字マッピング”という独自の機能を使用すると、1 つ以上の配列内にあるデータを、どのようにして物理的なデータベース・ファイルにマップするかを指定できます。このようなマッピングは、データベース管理タスクです。クラスやテーブル定義、あるいはアプリケーション・ロジックを変更する必要はありません。また、マッピングは特定のまばらな配列内で行うこともできます。ある値の範囲を別の物理ロケーションにマップしたり、他のファイル、ディスク・ドライブ、あるいは他のデータベース・サーバにもマップしたりできます。これにより、(スケーリングなど) Caché アプリケーションの再構成が簡単に実行できます。

トランザクション多次元ストレージの柔軟性により、Caché は従来のリレーショナル・データベースを使用した二次元構造以上の利点があります。この柔軟性のために、たとえば、Caché は高性能の SQL、オブジェクト、XML データベースとして機能します。つまり、Caché アプリケーションは、テクノロジーの将来的な進歩により良く備えることができます。

## 2.2 プロセス管理

Caché には、Caché プロセスでデータベース操作やあらゆる規模のビジネス・ロジックを実行する機能があります。



プロセスとは、Caché サーバで稼動する Caché 仮想マシンのインスタンスです。通常の Caché サーバは、ハードウェアとオペレーティング・システムの性能によって、何千もの同時プロセスを実行できます。各プロセスは、多次元ストレージ・システムに直接アクセスします。

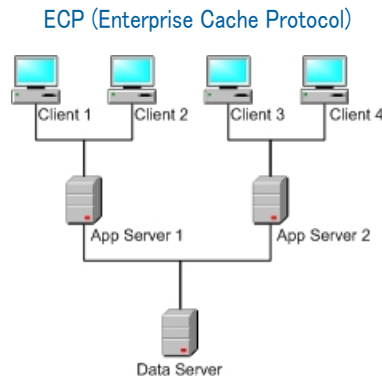
Caché 仮想マシンは、“P-code” と呼ばれる命令を実行し、トランザクション処理とデータウェア・ハウジング・アプリケーションに必要なデータベースの入出力とロジック処理の性能を最適化します。以下は、仮想マシン・コードの生成方法です。

- ・ SQL – Caché に発行される SQL クエリは、Caché SQL オプティマイザ で処理された後、効率的に実行可能 P-code に変換します (既存のインデックスを使用します)。

- ・ オブジェクトの動作 – Caché オブジェクト・テクノロジーは、メソッド・ジェネレータ (事前に定義されたルールに沿ってコードを生成するオブジェクト・メソッド) を使用して、実行可能な P-code を自動的に生成することで、(オブジェクトの永続性など) サーバ側のオブジェクトの動作を提供します。
- ・ Caché ObjectScript – アプリケーションでは、Caché ObjectScript 言語を使用してあらゆるロジックを組み込み、Caché データあるいはアプリケーション・サーバで実行できます。このようなコードは、オブジェクト・メソッドの形式を取るか (オブジェクト指向のフル機能を使用できます。リレーショナル界に格納されたプロシージャに類似していますがそれ以上に強力です)、あるいは Caché サーバで実行する小規模プログラム、完全な “ルーチン” を備えることができます。
- ・ Caché Basic – オブジェクト・メソッドは、Basic プログラミング言語を使用しても実装できます。Caché には、世界中のソフトウェア開発者が一般的に使用している強力なオブジェクト・ベースの Basic プログラミング言語が含まれています。Caché Basic は、それぞれにサポートされたプラットフォームで実行し、Caché ObjectScript と完全に相互運用できます。

## 2.3 分散データ管理

Caché で最も強力な機能の 1 つは、分散データ・ネットワークを構築するためにサーバを 1 つにリンクする機能です。このようなネットワークでは、主にデータを提供するマシンをデータ・サーバと言います。一方、処理がメインで、データをほとんど持たないマシンをアプリケーション・サーバと言います。



サーバは、Caché の ECP (Enterprise Cache Protocol) を使用して、データ (とロック) を共有できます。ECP は、データをパッケージ化して転送することで転送効率を向上させています。ネットワーク経由でデータを要求された場合、必要なデータとそれに関連するデータをパッケージ化して返送します。オブジェクトと多次元データ・モデルを継承する特有のデータ・リレーションシップにより、要求された元のデータに関連する情報を識別し、取り込むことができます。“関連する” 情報は、クライアントあるいはアプリケーション・サーバのどちらかのローカル・サーバでキャッシュされます。通常、データのアクセスは近くのキャッシング情報から先に検索を行い、ネットワーク上で不必要なデータ

要求を行わないようにしています。クライアントがデータを更新した場合、変更されたデータのみをデータベース・サーバに転送します。

ECPにより、アプリケーションは、多階層とピアツーピア通信を含む広範囲のランタイム構成をサポートできます。

## 2.4 ジャーナル管理

整合性と信頼性を備えたデータベースにするため、Cache には、データベースの更新状況を物理的あるいは論理的に追跡するジャーナリング・サブシステムがあります。ジャーナル管理テクノロジーは、トランザクションのサポート（ジャーナルは、トランザクションのロールバック処理を実行）と、データベースのシャドウイング（ジャーナルは、シャドウ・サーバとプライマリ・データ・サーバの同期を取ります）にも使用します。他のサブシステムと同様に、Cache は、ジャーナリング・システムをアプリケーションの要件に合うように構成します。

## 2.5 ロック管理

同時データベース・アクセスをサポートするために、Cache には、強力なロック管理システムがあります。

数千のユーザをサポートするシステムでは、競合するプロセス間の衝突の削減は、パフォーマンスの向上に不可欠です。もっとも大きな競合は、同じデータをアクセスするトランザクション間にあります。Cache のロック管理には、このような衝突を削減する以下の機能があります。

- ・ アトミック処理 – 典型的なパフォーマンス・ホット・スポットを削除するため、Cache は、アプリケーション・レベルのロックが不要な複数のアトミック処理をサポートします。たとえば、一意の値をオブジェクト識別あるいは行識別の最小単位で割り当てる機能です（リレーショナル・アプリケーションで共通のボトルネックです）。
- ・ 論理ロック – Cache は、更新中にデータのページ全体をロックしません。大半のトランザクションは、少量のデータへの頻繁なアクセスや変更であるため、Cache は、各オブジェクト（行）単位で取得する最小単位の論理ロックをサポートします。
- ・ 分散ロック – 分散データベース構成では、Cache は、自動的に分散ロックをサポートします。

## 2.6 デバイス管理

Cache は、無数のデバイス(ファイル、TCP/IP、プリンタなど)をプラットフォームに依存せずにサポートします。このため、Cache アプリケーションは、他のテクノロジーのホストと相互運用できます。Cache で可能な相互接続オプション (CSP、ODBC、SOAP、Java など) は、この基本的な機能をサポートした上で構築されます。

## 2.7 可搬性

Cache は、Windows (98、NT、200、XP など)、OpenVMS、UNIX の各主要バージョンを含むさまざまなハードウェア・プラットフォームとオペレーティング・システムで稼動し、最適化されます。

Cache で開発したアプリケーションやデータをあるプラットフォームから別のプラットフォームへ簡単に移植できます。これは、Cache を新規のプラットフォームにインストールし、データベース・ファイルを新規システムに移植するのと同様に簡単です。システム間を移動する場合、(あるエンディアン表現を別の表現に変換するために) 組み込みのデータ変換ユーティリティを実行する必要があるかもしれません。

## 2.8 アプリケーション稼動オプション

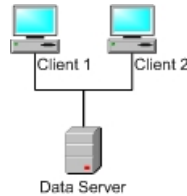
Cache は、アプリケーションを実稼動する際に、最大限の柔軟性を提供するさまざまなランタイム構成をサポートします。Cache のシステム設定を変更して、さまざまな稼動オプションを交互に使用できます。通常、アプリケーション・ロジックを変更する必要はありません。

基本的な稼動オプションは以下の通りです。

### 2.8.1 基本的なクライアント/サーバ構成

単純なクライアント/サーバ構成では、単一の Cache データ・サーバは、多くのクライアント (アプリケーションとプラットフォームによって 1 から数千まで) を持ちます。

### クライアント/サーバ構成



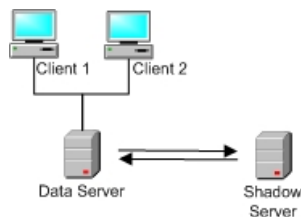
クライアント・システムは、以下のいずれかです。

- ・ スタンドアロンのデスクトップ・システム。クライアント/サーバ・プロトコル (ODBC、ActiveX、JDBC、Java など) 経由で接続するクライアント・アプリケーションを実行します。
- ・ CSP (Caché Server Pages)、SOAP、他の接続オプション (ODBC、JDBC など) を経由して Caché と会話する Web サーバ・プロセス。各 Web サーバ・プロセスは、多数のブラウザ・ベースあるいはマシン・ツー・マシンのセッションを提供します。
- ・ ODBC、JDBC などを経由して Caché に接続するミドルウェアのプロセス (Enterprise Java Bean アプリケーション・サーバなど)。
- ・ サポートされるプロトコル (TELNET、TCP/IP など) の 1 つを使用して Caché に接続するターミナルやハードウェアなどのデバイス。
- ・ 上記の方法の中からの、いくつかの方法の組み合わせ。

## 2.8.2 シャドウ・サーバ構成

シャドウ・サーバ構成とは、基本的なクライアント/サーバの設定に、1 つ以上のシャドウ・サーバを追加して構築されます。各シャドウ・サーバは、トランザクション・ジャーナルに接続し、監視することで、サーバ自身と、メイン・データ・サーバ内にあるデータとの同期を取ります。

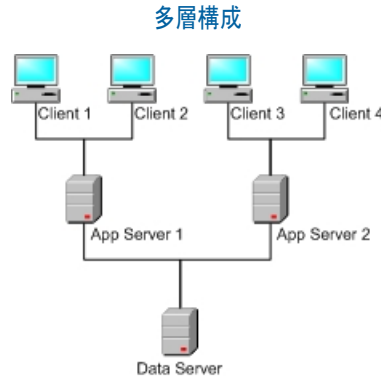
### シャドウ・サーバ構成



シャドウ・サーバは典型的には、一時的クエリ、大量のレポート、バッチ処理を行うために使用され、メイン・トランザクション・システムへの影響を制限します。また、フェイルオーバー・システムとしても使用できます。

### 2.8.3 多層構成

多層構成は、Caché の分散データベース・テクノロジー (ECP) を使用して、大量のクライアントをシステムに接続できます。



最も簡単な多層設定は、アプリケーション・サーバとして動作する1つ以上の Caché システムを、中央データ・サーバとさまざまなクライアント・システム間に配置します。この場合、アプリケーション・サーバはデータを格納せず、代わりに、データ・サーバの CPU をアンロードするクライアント特有の動作を実行するプロセスを収容します。この構成は、“参照の局所性”を示すアプリケーションに最適です。つまり、大半のトランザクションは関連するデータがある程度含んでいるため、アプリケーション・サーバ間のロックを小さくすることができます。(大半の Web アプリケーションのように) 適切な読み取りアクセス数を持つこのようなアプリケーションは、上記モデルの実行に最適です。

また、複数のデータ・サーバや、アプリケーション・サーバ・マシンに格納されたデータを持つといったさらに複雑な構成も可能です。

通常、アプリケーションは、(ホット・スタンドバイ・システムとして動作するアプリケーション・サーバで) 拡張性や高可用性のために多層構造を使用します。

# 3

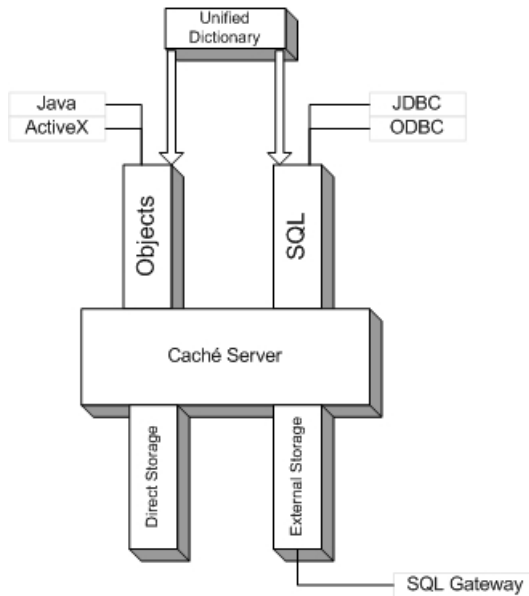
## オブジェクト、SQL、統一データ・アーキテクチャ

Caché の強力な機能に、固有の統一データ・アーキテクチャがあります。これは、高性能のオブジェクトとリレーショナル・アクセスを Caché に格納したデータに同時に提供します。

### 3.1 統一データ・ディクショナリ

Caché では、ユーザのアプリケーション・コンポーネントをオブジェクトとして定義できます。オブジェクトは、オブジェクトのデータ (プロパティ) と動作 (メソッド) を定義するクラスで構築されます。

## 統一データ・アーキテクチャ



各クラスのメタ情報あるいは定義は、Cache クラス・ディクショナリと呼ばれる共通のリポジトリに格納されます。クラス・ディクショナリ自身は、Cache に格納されるオブジェクト・データベースで、オブジェクトを使用してコンテンツにアクセスします。クラス・コンパイラを使用し、クラス・ディクショナリは、永続オブジェクトに必要なストレージ構造を定義し、クラス定義を実行可能コードの平行・セットに変換します。このコード・セットは、このストレージ構造に対してオブジェクトとリレーショナル・アクセスの両方を提供します。このアーキテクチャにより、オブジェクトとリレーショナル・コード・パスは、効果的また自動的に互いに同期化されます。

クラス定義は、いくつかの方法で、クラス・ディクショナリに追加できます。

- ・ Cache スタジオ 開発環境のインタラクティブな使用。
- ・ リレーショナル DDL の使用。Cache は、標準 SQL DDL 文を受け入れ、対応するクラスとテーブル定義を自動的に生成します。
- ・ テキストの XML を使用。Cache は、外部 XML で記述されたクラス定義をサポートします。通常、これは、ソース・コード管理、運用、自動コード生成、他のツールとの相互運用性のために使用します。
- ・ プログラム的にオブジェクトを使用。Cache のクラス定義オブジェクトのセットを利用し、クラス・ディクショナリと直接通信するプログラムを生成し、アプリケーション実行時に新規クラスを生成します。
- ・ Cache スタジオ に組み込まれている XML スキーマ・ウィザードの使用。大半の XML スキーマ・ファイルからクラス定義を生成できます。

### 3.1.1 柔軟なストレージ

Caché のオブジェクト・モデルは、プロパティやメソッド以外にもプログラミング言語で使用されるモデルと異なり、インデックス、制約条件、ストレージ構造などストレージに関連する動作を指定できます。

永続オブジェクトによって使用されるストレージ構造は、クラスの論理定義から独立しているため、非常に柔軟性があります。開発者は、クラス・コンパイラから提供される既定の構造を使用することも、特定のケースに合わせて構造を調整することもできます。Caché SQL ゲートウェイ を使用して、外部リレーショナル・データベースにクラスを格納できます。

## 3.2 オブジェクト

Caché のフル機能、次世代オブジェクト・データベースは、複雑なトランザクション向きのアプリケーション要件に合うように設計されています。Caché オブジェクト・モデルには、以下の機能があります。

- ・ クラス – ユーザのアプリケーション・コンポーネントの状態（データ）と動作（コード）を決定するクラスを定義できます。クラスは、ランタイム・コンポーネントとして、あるいはデータベースに格納される要素として、オブジェクトのインスタンスを生成するために使用します。
- ・ プロパティークラスには、各オブジェクトのインスタンスに関連するデータを指定するプロパティがあります。プロパティは、単純なリテラル（文字列、数値など）、ユーザ定義型（データ型クラスで定義される）、複雑なオブジェクト（あるいは埋め込みオブジェクト）、コレクション、他のオブジェクト参照です。
- ・ リレーションシップ – クラスは、オブジェクトのインスタンスが他のインスタンスにどのように関連しているかを定義できます。システムは、データベース内に、リレーションシップの操作メソッドと参照一貫性を自動的に提供します。
- ・ メソッド – クラスは、メソッドを使用し動作を定義します。メソッドとは、オブジェクトに関連付けられた実行可能なコードです。オブジェクト・メソッドは、Caché サーバ・プロセスで実行します（リモート・クライアントから実行できます）。オブジェクト・メソッドは、Caché ObjectScript、SQL を使用して記述するか、または、メソッド・ジェネレータを使用して生成できます。メソッド・ジェネレータとは、ユーザが定義した規則に従ってカスタマイズされたメソッドを自動的に生成するコードです。
- ・ オブジェクトの永続性 – 永続オブジェクトには、自分自身をデータベースに自動的に格納したり取り出したりする機能があります。永続性のサポートには、自動トランザクション管理、同時アクセス・コントロール、インデックス・メンテナンス、データの妥当性検証を含む完全なデータベース機能があります。永続オブジェクトは、SQL クエリから自動的に見ることができます。
- ・ 継承 – 新規クラスを既存のクラスから派生することで、以前記述したコードを再利用したり、特有用なクラスを生成したりできます。

- ・ポリモフィズム (多態) – Caché は、完全なオブジェクトのポリモフィズムをサポートします。つまり、アプリケーションは、適切なインタフェース (スーパークラスからのメソッドやプロパティのセット) を使用し、またシステムは、各オブジェクト・タイプを元に、正確なインタフェースの実装を自動的に呼び出します。これにより、柔軟なデータベース・アプリケーションの開発が可能になります。
- ・スウィズリング (“遅延ロード”) – Caché は、オブジェクトが別のオブジェクトから参照される場合、関連する永続オブジェクトを自動的にスウィズル (ディスクからメモリに格納) します。これにより、複雑なデータ・モデルでの作業を簡素化します。

### 3.2.1 クラス定義

Caché のクラス定義で、単純で最も一般的な方法は、Caché スタジオ 開発環境を使用することです。スタジオでクラスを定義するには、構文エディタ内の単純なテキスト形式、あるいはグラフィカルなポイント・アンド・クリックのインタフェースのいずれかを使用します。これら 2 つのビューは、交互に使用でき、自動的に同期を取ります。

ここでは、極めて単純な永続オブジェクト、**Component** を定義します。Caché スタジオでは以下のように表示されます。

```
Class MyApp.Component Extends %Persistent [ClassType = persistent]
{
Property TheName As %String;
Property TheValue As %Integer;
}
```

このクラスは、永続クラスとして定義されます (つまり、データベース内に自分自身を格納できます)。この場合、Caché が提供する **%Persistent** クラス (アプリケーション・クラスと区別するため、システム・クラス名には先頭に “%” 文字があります) は、すべて必要な永続コードを継承します。クラスは、パッケージ、“MyApp” に属します。パッケージは、関連するクラスを一まとめにし、大規模アプリケーションの開発を大幅に簡素化します。クラスは 2 つのプロパティを定義します。文字列値を持つ **TheName** と整数値を持つ **TheValue** です。

メソッド内などの Caché ObjectScript コード内から、このオブジェクト構文を使用して、**Component** オブジェクトのインスタンスを処理できます。

Basic を使用する場合、**Component** オブジェクトのインスタンスを操作し、メソッドを定義できます。

```
' Create a new component
component = New Component()
component.TheName = "Widget"
component.TheValue = 22

' Save the new Component to the database
component.%Save()
```

ここで、**Component** の新規インスタンスは、システムが割り当てた一意のオブジェクト識別子を持ち、データベースに格納されます。(このオブジェクト識別子を使用して) これをオープンすると、オブジェクトを取り出すことができます。

```
' Open an instance and double its value:
component = OpenId Component(id)

component.TheValue = component.TheValue * 2
component.%Save()
```

Caché のさまざまなクライアント結合を使用して、ネイティブの Java、C++、ActiveX、または .NET を使用した場合と同一の処理を実行できます。クラス・コンパイラは、外部からのオブジェクトへのアクセスに必要な追加コードを生成し、同期を取ります。たとえば、Java で Caché を使用している場合、クラス・コンパイラは、永続データベース・クラスにリモート・アクセスする Java プロキシ・クラスを自動的に生成、保持します。Java プログラムでは、元々このオブジェクトを使用できます。

```
// Get an instance of Component from the database
component = (MyApp.Component)MyApp.Component._open(database, new Id(id));

// Inspect some properties of this object
System.out.println("Name: " + component.getName());
System.out.println("Value: " + component.getValue());
```

## 3.3 SQL

Caché SQL は、フル機能のリレーショナル・データベース・エンジンで、Caché のオブジェクト・テクノロジーと完全に統合されています。標準 SQL-92 機能の他に、Caché SQL には以下の機能があります。

- ・ ストリーム (SQL では BLOB) のサポート
- ・ (オブジェクト・メソッドとして実装された) ストアド・プロシージャのサポート
- ・ オブジェクト単位の拡張セット
- ・ ユーザ定義が可能なデータ型
- ・ トランザクション・ビットマップ・インデックスのサポート

ビットマップ・インデックスは、通常、大規模なデータの保管や OLAP システムで使用し、複雑に組み合わせられた条件による高速検索が実行できます。このようなビットマップ・インデックスでは、リアルタイム更新はできませんが、通常はバッチ処理として更新されます。Caché SQL はビットマップ・インデックスをサポートし、挿入/更新の処理動作を低下させずに高性能な検索機能を提供します。これにより、トランザクション処理アプリケーションは、データ・ウェアハウス形式の問い合わせを実行でき、データ・ウェアハウス・アプリケーションは、リアルタイム更新が可能になります。詳細は、Caché SQL ドキュメントを参照してください。

### 3.3.1 オブジェクト/リレーショナル接続

Caché ディクショナリのすべてのコンポーネントは、クラスとして定義されます。Caché クラス・コンパイルは、リレーショナル・テーブルとして永続クラスを自動的に投影します。各オブジェクト機能には、以下の表のように対応するリレーショナル・オブジェクトがあります。

オブジェクト機能のリレーショナル表

オブジェクト機能	リレーショナル・オブジェクト
パッケージ	スキーマ
クラス	テーブル
オブジェクト・インスタンス	テーブル行
プロパティ	列
リレーションシップ	外部キー
埋め込みオブジェクト	複数の列
メソッド	ストアド・プロシージャ
インデックス	インデックス

Caché が SQL DDL (データ定義言語) 文をロードする場合、この逆投影を使用して、リレーショナル・テーブルに対応するクラスを生成します。

以下は、リレーショナル投影へオブジェクトを示す簡単な例です。以下は、2つのプロパティ、**Name** と **Home** を持つ永続 **Person** クラス (“MyApp” と呼ばれるパッケージの一部) の定義です。

```
Class MyApp.Person Extends %Persistent [ClassType = persistent]
{
Property Name As %String(MAXLEN=100);
Property Home As Address;
}
```

**Person** クラスは、Caché が持つ **%Persistent** スーパークラスから永続的な動作を継承します。**Name** プロパティは、100 文字までの簡単な文字列で定義します。

**Home** プロパティは、複雑なユーザ定義のデータ型を表しています。この場合、**Address** クラスは以下のように定義されます。

```
Class MyApp.Address Extends %SerialObject [ClassType = serial]
{
Property City As %String;
Property State As %String;
}
```

**Address** クラスは、**%SerialObject** スーパークラスから派生します。このクラスには、それ自身のクラスを直列化し（自身を単一の文字列表現に変換する）、別のクラス内に（**Person** クラスのように）埋め込む機能があります。

SQL 表示される **Person** クラスの構造は、以下のようになります。

**Person** クラスの SQL ビュー : `SELECT * FROM Person`

ID	Name	Home_City	Home_State
1	Smith,John	Cambridge	MA
2	Doe,Jane	Dallas	TX

オブジェクト識別子は、列として表示されていることに注意してください。また、埋め込みのオブジェクトの **Address** フィールドは、分割フィールドとして投影されます。フィールドには、**Home\_City** と **Home\_State** という複合的なフィールド名があり、2つの独立したフィールドが定義されているかのように動作します。

### 3.3.2 継承と SQL

継承は、オブジェクト・ベースのシステムで重要な機能です。この機能は、リレーショナル・データベースにはありません。Cache SQL は、標準リレーショナル構造を使用して、この強力な継承機能を使用できます。たとえば、前例で使用した **Person** クラスから新規 **Employee** クラスを派生できます。

```
Class MyApp.Employee Extends Person [ClassType = persistent]
{
Property Salary As %Integer(MINVAL=0,MAXVAL=100000);
}
```

新規クラスは、**Salary** プロパティを追加して、**Person** クラスを拡張します。

SQL で表示される **Employee** クラスの構造は、以下のようになります。

**Employee** クラスの SQL ビュー : `SELECT * FROM Employee`

ID	Name	Home_City	Home_State	Salary
3	Divad, Gino	Irvine	CA	22000

継承されたすべてのプロパティは、列として使用可能であることに注意してください。また、**Employee** の実インスタンスである行のみが含まれていることにも注意が必要です。以下は、すべての **Person** インスタンスを再要求します。

Person クラスの改訂 SQL ビュー : `SELECT * FROM Person`

ID	Name	Home_City	Home_State
1	Smith,John	Cambridge	MA
2	Doe,Jane	Dallas	TX
3	Divad, Gino	Irvine	CA

この場合、**Employee** ごとに返されるすべての行は **Person** のインスタンスとして定義されています。しかし、**Person** で定義されたプロパティのみが表示されます。

### 3.3.3 SQL へのオブジェクト拡張

オブジェクト・アプリケーションで SQL を使用しやすくするために、Cacheé には SQL へのオブジェクト拡張が多数あります。

最も特殊な拡張の 1 つは、リファレンス演算子 (“->”) を使用して、オブジェクト参照を実行する機能です。例として、2 つのクラス、**Contact** と **Region** を参照する **Vendor** クラスがあるとします。リファレンス演算子を使用すると、関連するクラスのプロパティを参照できます。

```
SELECT ID,Name,ContactInfo->Name
FROM Vendor
WHERE Vendor->Region->Name = 'Antarctica'
```

また、SQL JOIN 構文を使用しても、同様のクエリ式を記述できます。リファレンス演算子構文の利点は、簡潔で理解しやすい点です。

# 4

## 開発ツールとデータベース・ユーティリティ

Cachéには、多くのアプリケーション開発ツールとデータベース管理ユーティリティがあります。Windows システムでは、タスク・バーの Caché キューブ・アイコン  からアクセスできます。(アイコンを右クリックします。このアイコンが表示されていない場合、Windows の [スタート] メニューを開き、[プログラム] から [Caché] フォルダを見つけ、[Caché] の項目をクリックします)。さまざまなグラフィカル・ユーティリティは、クライアント/サーバ・アプリケーションです。クライアントは Windows システムで稼働しますが、リモートの Windows、Linux、UNIX、OpenVMS サーバ・システムと共に使用できます。

Caché 開発ツールとユーティリティは以下の通りです。

### Caché スタジオ

Caché スタジオ は、Caché の主要な開発環境です。スタジオでは、フル機能のエディタを使用して、クラス定義、CSP (Caché Server Pages)、ルーチンを生成できます。スタジオには、高度な構文チェック機能とカラー表示機能、グラフィカル・デバッグ機能、ポイント・アンド・クリックのクラス・インスペクタがあります。Caché スタジオは、Caché データベース (Caché アプリケーション) に直接作動し、複数の開発者をサポートすると同時に、システムの整合性を取るため、ソースを制御します。スタジオ・テンプレートを使用して、スタジオをカスタマイズできます。

### Caché システム管理ポータル

システム管理ポータルは、Caché サイトを管理するための Web ベースのインタフェースです。この管理ポータルには、システム管理者、セキュリティ・マネージャ、データベース・オペレータ、および Caché にアクセスする必要があるその他のユーザ用のツールが含まれています。この管理機能では、Caché システムを設定し、構成パラメータを表示および変更し、システム設定を調整し、データベースおよびネームスペースを作成および編集できます。データベース参照機能には、Caché データベースの内容を参照する機能、データ・エ

エンジンのデータ (グローバル) を検証する機能、クラスとルーチンを参照する機能だけでなく、データベースのアクティビティの監視およびバックアップ処理の実行機能もあります。セキュリティ関連の機能によって、ユーザ、ロール、リソース、および権限の追加、変更、および削除が可能で、他のセキュリティ管理タスクも実行できます。SQL 関連機能として、グラフィカルな Caché データベースの SQL ベースのビューがあります。それを使用すると、SQL ロールおよびアクセス許可の管理、テーブルおよびビュー定義の参照、任意 SQL クエリの実行、クエリ・キャッシュの管理、およびデータのインポートとエクスポートが可能です。

### Caché ターミナル

Caché ターミナルには、Caché とインタラクティブなコマンド行のインタフェースがあります。ターミナルを使用して、Caché データベース・エンジンと直接やりとりできます。ターミナルは、アプリケーションのテストとトラブルシューティングを実行する重要なツールです。

### Caché DocBook

Caché DocBook は、Web ベースで、全文検索可能なデータベース・アプリケーションとして、Caché のドキュメンテーションを提供します。

# 5

## 接続性

Caché には、さまざまなテクノロジーがあり、他の大半のソフトウェア・アーキテクチャと効率的かつ簡単に接続、相互運用できます。

### Caché Server Pages

Caché Server Pages (CSP) には多様なテクノロジーがあり、高性能で拡張可能なデータベース・アプリケーションを素早く構築できます。多くのプラットフォームの主要な Web サーバで動作します。詳細は、“Caché Server Pages (CSP) の使用方法”を参照してください。

### Caché ODBC

Caché には、高性能なネイティブ ODBC ドライバがあり、リレーショナル・アプリケーションとツールにアクセスできます。Caché ODBC ドライバは、多くの UNIX プラットフォームと Windows で使用できます。

### Caché JDBC

Caché には、高性能なネイティブ JDBC ドライバがあり、リレーショナル Java アプリケーションとツールにアクセスできます。

### Caché XML と Web サービス

Caché には、XML と SOAP を相互運用する多くの方法があります。詳細は、“Caché での XML の使用方法”と “Caché での SOAP と Web サービスの使用法”を参照してください。

### Java と EJB 用 Caché オブジェクト・サーバ

Caché Java バインディングによって、Caché 永続クラスを Java と EJB (Enterprise Java Bean) アプリケーションで使用できます。詳細は、“Caché での Java の使用方法”を参照してください。

### ActiveX 用 Caché オブジェクト・サーバ

ActiveX 用 Caché オブジェクト・サーバでは、Caché 永続クラスを Visual Basic や .NET といった ActiveX 環境で使用できます。詳細は、“Caché での ActiveX の使用法”を参照してください。

### C++ 用 Caché オブジェクト・サーバ

Caché C++ バインディングによって、Caché 永続クラスを C++ アプリケーションで使用できます。詳細は、“Caché での C++ の使用法”を参照してください。

### Caché SQL ゲートウェイ

Caché SQL ゲートウェイは、サード・パーティのリレーショナル・データベースに対し、Caché アプリケーション・オブジェクトへのアクセスを提供します。詳細は、“Caché SQL ゲートウェイの使用法”を参照してください。

### Caché ActiveX ゲートウェイ

Caché ActiveX ゲートウェイは、ActiveX/COM/.NET コンポーネントに対し、Caché アプリケーションの直接アクセスを提供します。詳細は、“Caché ActiveX ゲートウェイの使用法”を参照してください。