

リレーショナル・データベースの失敗、オブジェクト・テクノロジーの出現、ハイブリッド・データベースの必要性

革命的パラダイム

1960年代後半に初めて紹介されたオブジェクト・テクノロジーは、極めて革命的なテクノロジーでした。1980年代後半までに、さまざまな理由から主流となったオブジェクト・テクノロジーは、インタフェース開発を簡単にしただけでなく、柔軟で機能的にデータ処理を行うことができたため、アプリケーションの構築方法を根本から変えました。リレーショナル・データベースのように固定のテーブルでデータを表現する代わりに、オブジェクト・テクノロジーは、クラスという観点でデータを表現します。特定のオーク・ツリーが「オーク・ツリー」クラス・インスタンスであるのと同様に、オブジェクトは、クラスのインスタンスです。

オブジェクト・テクノロジーが継承の概念を確立したため、クラスを階層的に配置できるようになりました。例えば、「オーク・ツリー」クラスは、さらに一般的な「ツリー」クラスからデータ構造と振る舞いを継承できます。

オブジェクト・テクノロジーは、その世界を実世界に合わせて適切に表現し、オブジェクト指向言語は、大半のプログラミング領域で、多岐に渡り使用できます。これにより、プログラミング言語はより自然な言葉に近づき、ソフトウェア開発の考え方に大きく影響を与えました。オブジェクト指向は「新しい枠組」と呼ばれ、その影響は広範囲に及びました。

オブジェクト指向の機能は既存の言語にすぐに追加され、C++などの言語が開発されました。また、Visual Basic、Visual C++、PowerBuilder、Delphi、Cachéなど新しいオブジェクト指向の開発環境も出現しました。オブジェクト・テクノロジーは、先進の開発環境を採用しながらも、正式に認められるまでには時間がかかりました。しかし、真のオブジェクトベースの世界を考え構築するには、さらに時間がかかるでしょう。今のところ、それは未知の世界です。

WWWにより広まったオブジェクト・テクノロジー

WWWはあらゆる情報交換の方法を変えてしまったため、オブジェクト指向のプログラミング言語、JavaがWeb開発者の重要な言語となりました。C++を基にJavaは、ブラウザから使用できる小規模なアプリケーション（Javaアプレット）を作成できます。

Sunは、Javaへの理解を深めてもらうため、無料でJavaの環境を提供しました。数年間で多くのコピーがダウンロードされ、Javaはあちこちに広まりました。JavaScript、C#、Jscriptのような多くのオブジェクト指向言語が開発されました。また、インターネットの発展により、PerlやPHPなど他の新しいオブジェクト指向言語も出現しました。現在の開発者は、オブジェクト指向の機能を当たり前のように使用しています。

オブジェクトの出現

オブジェクト・テクノロジーは、ソフトウェア開発のすべての側面を特徴付けています。また、オブジェクト指向モデリングは、先進の標準UML方法論を使用したアプリケーション・モデリング市場の中心です。

1990年代は、オブジェクト指向のミドルウェア製品が出現し、オブジェクト指向アプリケーション間の安全な通信サービスを提供しました。ミドルウェア市場は、1998年に発表された JMS (Java Messaging Service) により飛躍的に進歩しました。JMS は、包括的なメッセージング機能を備えた API を定義し、J2E を実装する際、JMS サーバを組み込むよう義務付けました。これにより標準化が進み、ミドルウェアのコストを下げ、企業規模のオブジェクトベース・アプリケーションを記述するプラットフォームを提供しました。

XML と Web サービス

1998年、Web ページを記述するためのマークアップ言語、HTML を拡張し、XML (eXtended Mark-up Language) を標準化しました。XML の構文は、独自のデータ形式を作成して文書を定義します。これは、データベース内のデータ定義に類似しています。XML では、プログラムでデータに定義を与え、データとそのデータの意味をそれぞれ決定します。これにより、請求書や注文書など独自に標準化されたデータ・オブジェクトを、企業内や企業間で簡単にデータ交換できるよう定義することができます。XML は Web サービスにも展開されており、カスタマイズしなくても、他のプログラムと瞬時に情報をやり取りすることができます。現在、正式な Web サービスの動作環境には、J2EE と .NET の 2 つがあります。今日 XML は、データに強力な影響を与えています。開発者は、SQL と同様に、データを取得するための標準として XML を使用できますが、オブジェクト・レベルでデータを定義する標準言語としても使用できます。XML の人気は、オブジェクト指向と同様に急上昇してきています。その結果、XML を基盤にしたデータ・オブジェクトの新しい標準と新しい開発製品が出現し続けています。

オブジェクト・データベース – 市場参入の遅れ

オブジェクト・テクノロジーが、ソフトウェア開発において全面的に急速に受け入れられたのとは著しく対照的に、オブジェクト・データベースの導入は立ち遅れ、最近になりようやく限定的に採用されるようになりました。オブジェクト・データベースに対する理解の遅れには、多くの理由があります。

初期のオブジェクト指向言語は、データ・ストレージという考えはありませんでした。プログラムはメモリでデータを操作し、すべてのデータ・イメージは、プログラムが次に実行された時に読み込めるようファイルに格納されました。この方法では、アプリケーションはデータを共有できず、また、データの回復性、管理の容易性、スケラビリティもありません。

当然のことながら、市場にはさまざまなオブジェクト・データベース製品があります。例えば、Versant、Objectivity、ObjectStore、GemStone などは、オブジェクト指向の開発環境に適切なデータを格納することができます。これらのデータベースは、市場で重要な製品となり、市場を独占さえできるだろうと、当初かなりの意気込みと大きな期待を受けていました。

しかし、オブジェクト・データベースが出現する頃には、リレーショナル・データベース・ベンダがすでに人気を得て、市場に浸透していました。標準の SQL インタフェースを使用して、オブジェクト指向ルーチンを記述し、リレーショナル・データベースに簡単にアクセスできました。一方、初期のオブジェクト・データベースは、SQL 機能にはまったく対応しておらず、クエリ・アプリケーションには不向きでした。その結果、オブジェクト・データベースは、ビジネス・システムで強力な地盤を築くことができませんでした。しかし、CAD/CAM、電気通信、マルチメディア、人工知能、金融商品のモデリング、患者の治療追跡システム、科学アプリケーションなど複雑なオブジェクトを管理、格納するアプリケーション分野のすき間市場で地位を確立しました。

データベース市場は、オブジェクト・データベースにそれほど注目してきませんでした。しかし、データ定義言語として XML が出現し、そこで定義したデータ管理にオブジェクト・データベースが無理なくマッチしたため、オブジェクト・データベースが見直されるようになりました。XML を使用し、複雑なデータを格納する必要性が高まったことが、オブジェクト・データベースの復活を導いたようです。

2003年9月の InfoWorld で発表されたディベロッパ調査では、驚くべき結果が出ています。回答者の 89.2% がリレーショナル・データベースを使用していましたが、52% が、オブジェクト指向データベースあるいは XML データベースも使用していると回答しています。格納したデータの種類に関する質問には、40.2% が永続オブジェクト、58.9% が XML データ、89% がリレーショナル・データと答えています。オブジェクト・データベースは、一般的に考えられている以上に広く使用されている、と Baroudi Bloor は確信しており、実際に、需要に合わせて市場に徐々に浸透してきています。

また、InfoWorld の調査では、新規の開発の場合、オブジェクト指向言語の選択が有力であると明言しています。このような統計は、現在開発者が直面する苦悩を表しているのでしょう。開発者は、使用しているオブジェクト言語が適切に動作するデータベースが必要ですが、リレーショナル・データベースに備わっているクエリ機能も必要です。

リレーショナル・データベース – 歴史と状況

プログラムがあると、必ずデータがありました。IT を代表するビジネス価値として初期に重要であったものの一つに、データ管理があります。データ管理の自動化によりビジネスは拡大し、これまでになかったやり方での競争が可能になります。したがって、卓越したビジネス技術者は、当然早くからデータ管理市場に焦点を当てていました。オブジェクト・データベースが発案される 10 年以上も前に、Dr. E. F. Codd が提唱したデータのリレーショナル理論から、リレーショナル・データベース製品が生まれました。80 年代の半ばには、データが持つ論理上の問題はすべて解決し、実際の問題もすぐに解決できるであろうという盲信が IT 産業の中に広がりましたが、当然、そのようなことは起こりませんでした。

リレーショナル・データベースは、単純な二次元テーブルでデータを表現します。これにより、プログラマが理解しやすいように多くのデータを効果的に表現できます。SQL と共に使用することで、リレーショナル・データベースは、標準的なデータ・アクセス言語になりました。これらは、論理的な物理構造を持ち、アプリケーション中立で、多くのビジネス・アプリケーション用として適切に動作しました。

しかし、リレーショナル理論の基礎の一つに、使用するデータとプログラムは、相互に独立できる (すべき) という考え方がありました。ここが、オブジェクト・テクノロジーの全般的な考え方と昔も今も異なる点です。オブジェクト・テクノロジーは、設計者が、データをテーブルではなくオブジェクトとして考える必要があります。オブジェクトを使用するオブジェクトとメソッドは、お互い独立していません。

例えば、複雑なオブジェクトとして自動車を考えてみましょう。自動車に乗る場合、自動車を一つの物体、つまりオブジェクトとして使用します。自動車は、多くの動作 (オブジェクト指向の言葉ではメソッド) を伴います。例えば、車を操縦し、ギアを変え、合図を出し、ヘッドライトをつけるなどを行います。車がオブジェクトだとすると、このような動作はオブジェクトのメソッドであり、基本的なことです。したがって、このような動作が車から独立していると考えerことは不適切です。また、車を車庫に駐車する場合、その機能を含め一つの物体として駐車します。車両本体は車庫に入れ、ステアリング、変速装置、方向指示、ライトの点灯といった機能を別の場所に置くことはありません。したがって、データとデータに関連するプロセスは分割できる (すべき) ものではなく、オブジェクト・データベース内に存在するものなのです。

実際、リレーショナル・データベースは、利点と限界を兼ね備えています。プロセスの中には、データから独立しているものもあります。大量のデータ・セットにアクセスするクエリの場合には特にそうです。クエリは、条件をもとにデータを単純に選択するため、データを素早く取得できる限り、データの性質や構成など重要ではありません。クエリはデータから独立していますが、オブジェクトのメソッドは独立していません。

リレーショナル・データベースの限界

リレーショナル・データベースの機能には、予想以上に多くの制約があります。極めて一般的なデータ構造でも、格納、表現することが非常に難しいものもあります。例えば、バス停を単純に並べたバスルートのリストを考えてみます。リレーショナル・データベースは、不規則に並んだリス

トとしてテーブルを持ち、特別に構築したインデックスを追加した場合にのみ、整列したリストを取得することができます。オブジェクト・データベースは、整列したリストを問題なく使用でき、インデックスも必要ありません。インデックスは、リレーショナル・データの構造上の限界により、人工的に作成されます。

別の一般的な例として、製造システムの製品とその構成部品に関する部品表を考えてみます。構成部品自体も、さまざまな部品から構成されていることがあります。すべての部品のリレーショナル・データベース・テーブルが、部品を構成する部品など部品同士のリレーションシップを表現することはありません。このようなリレーションシップは重要なデータを表しています。一つの製品とそれを構成するすべての部品のデータベースをクエリすることは容易でしょう。しかし、リレーショナル・データベースの構造では、この単純なクエリを実行する開発者のジョブを、不必要に複雑にかつ困難にします。このような例は、一つの地図と複数の道路、川、建物や、一つの Web サイトと複数のページ、リンク、画像がある場合など、多く溢れています。実際、集まった情報が複雑になればなるほど、階層やクロス・リレーションシップが増えるため、リレーショナル・データベースが持つ単純なテーブル構造では、そのデータを表現することが困難になってきます。オブジェクト・データベースはこのような制限がなく、この種の問題に対処するように設計されています。

リレーショナル・データベース製品は成熟し、過去 10 年間でコンピュータの機能も飛躍的に伸びているにも関わらず、使用しているリレーショナル・データベースのパフォーマンスが不十分であるためにプロジェクトが失敗した、ということをいまだに耳にします。一般的に、これはリレーショナル・データベースが物理的にデータを格納する方法に問題があるからです。必要なデータをアセンブルするために、開発者は複数のテーブル間に多くの JOIN を作成しなければならない場合があります。データを取得するためには、データベースで最適化ルーチンを実行し、データを集める最適な方法を決定後、データを取得します。このようなプロセスは時間がかかり、パフォーマンスに悪影響を及ぼすことがあります。リレーショナル・データベース・オブティマイザは、時間をかけて改良されてきましたが、オブジェクト・データベースより大幅なパフォーマンスのオーバーヘッドが現在も生じています。

リレーショナル・データベースとインピーダンスミスマッチ

リレーショナル・データベースが抱える問題は、使用する基本的なデータ構造が二次元テーブルであるということです。リレーショナル理論では、データは正規化されたテーブルで組織されると考えられます。つまり、データを一方向でのみ取得できるように組織するため、開発者は冗長性を排除し、データの修正を一貫して行うことができます。この設計技術は、リレーショナル・テーブルが、キーにのみ関連付けられた独立したデータ・セットを確実に持つよう導入されました。リレーショナル理論は、数学の集合論 (Set Theory) に由来しますが、集合論では、データが持つすべてのリレーションシップと構造を表現できないという点が問題です。

データを正規化して格納すると、プログラマは、データベースにオブジェクトを格納する前に分解し、オブジェクトを使用するために SQL 要求 (複数の JOIN) で再構築しなければならない場合があります。これは、まるで駐車場に車を駐車するときに、ドアやシート、車輪などを取り外して駐車するのと同じことで、時間がかかるだけの無駄な動作です。

このような問題は、オブジェクト指向言語が支配的になると表面化してきました。オブジェクト指向言語あるいはリレーショナル・データベースで使用するデータへのアプローチ方法の違いを、通常はオブジェクト・リレーショナル・インピーダンスミスマッチと言い、プログラマが問題に対処するしかありませんでした。現実には、大半のリレーショナル・データベースは、実装時に完全には正規化されません。しかし、その場合も、インピーダンスミスマッチの問題が生じ、開発を困難にします。実際、リレーショナル・データベースを使用するオブジェクト指向開発者は、25% ~ 40% の時間をコードの記述に費やし、オブジェクトをリレーショナル・テーブルにマップさせようとしています。

この基本的な問題が、オブジェクト・データベースへの要望を強めたのでしょう。しかし、大半のオブジェクト・データベースにも大きな問題があり、SQL をほとんどサポートしませんでした。多くのソフトウェア・ツール、特にビジネス・インテリジェンス・アプリケーションでは、SQL インタフェースが

必要です。SQL インタフェースを持つオブジェクト・データベースでさえ、ビジネス・インテリジェンス・アプリケーションが生成するクエリ・トラフィックを管理することはありませんでした。

オブジェクト・リレーショナル・データベース

リレーショナル・データベース・ベンダは、オブジェクトの出現に注目しました。複雑なデータ項目の正規化が無意味なのは、極めて明らかでした。極端な例として、ビットマップ・イメージ（ピクセルのリスト）を正規化すると、行にピクセルとその属性を置き、系列を示すプライマリ・キーを持つテーブルとなります。したがって、オブジェクトとしてデータを格納する方が、明らかに適しています。

そこで、「オブジェクト・リレーショナル」データベースという考えが提案されました。オーバーライドするリレーショナル・データベース構造を保存するためですが、これにより、リレーショナル・テーブルの列に複雑なオブジェクトを置くことができます。これらのオブジェクトは、プロセス（一種のストアド・プロシージャ）を使用して、その複雑なデータを分解することができます。その後 SQL の機能を強化し、「オブジェクト・メソッド」に相当するリレーショナルの呼び出しができるようになりました。

この方法は、データのリレーショナル理論を模倣しています。実際、リレーショナルは全く無視していますが、複雑なオブジェクト（マップ、ベクタ・グラフィック、画像、テーブル全体）をリレーショナル構造内で定義し、項目として保持できます。したがって、これらの機能が実装され、製品化されたものもあります。Informix は、埋め込みプロセスを DataBlades と呼び、Oracle は Cartridges と呼びました。

これは、オブジェクトを格納するオブジェクト・データベースに代わるものですが、基本的な問題は未解決のままです。オブジェクト・リレーショナル・データベースは、今でもインピーダンスミスマッチの問題を抱えています。

オブジェクト・データベースとリレーショナル・データベース

実際、オブジェクト・データベースは、リレーショナル・データベースを越える大きな利点があります。以下は主な利点です。

- トランザクショナル・アプリケーションで高速に動作
- 複雑なオブジェクトを効率よく処理
- 優れた生産性を提供
- 管理が簡単

特定の状況で、パフォーマンスを向上させるために、リレーショナル・データベースをオブジェクト・データベースに替えることもありました。例えば、リレーショナル・データベースの分野と考えられていた、複雑なオブジェクトを格納しない大規模なビジネス・アプリケーションでさえもです。

オブジェクト・データベースのパフォーマンスの主な利点は、リレーショナル・データベースと異なり、通常は、使用前にデータをアSEMBLする必要がないことです。最も頻繁に使用される形式でデータを格納するため、パフォーマンスの向上に役立ちます。オブジェクト・データベースは、データが要求されたときにメモリに置くキャッシュ戦略を実装できます。したがって、データを取得するために最適化の必要はありません。

新しいシステムが開発されるにつれ、ドキュメント、洗練されたグラフ、Web ページ、マルチメディアなど複雑なデータを操作する必要性が増してきています。オブジェクト・データベースは、このような要求に適切に対応することができます。

オブジェクト指向の現在

これまで、ソフトウェア開発全般で、オブジェクト・テクノロジー導入の長期的な成長を見てきました。オブジェクト・データベースを使用するすべての SE がリレーショナル・データベースをやめたわけではありませんが、InfoWorld のレポートによると、最後の聖域とも言えるデータベースにおいてさえ、52% の開発者が、オブジェクト・データベースであるオブジェクト指向データベースあるいは XML データベースを使用しています。また、オブジェクト指向の構成を使い易くするハイブリッド形式のデータベースを選択するものもいます。Web インタフェースが新規のアプリケーション開発で有力な選択肢となり、かつ Web サービスがアプリケーション対話に使用する対象のメカニズムであるため、オブジェクト指向の世界の構築が現実的になってきたようです。

2003 年 9 月の InfoWorld の調査では、プログラマは、至る所でオブジェクト指向言語を使用していると示しています。実際、永遠に C を使用すると主張するものもありますが、オブジェクト指向言語は、現在のプログラマの 90% が選択肢として考えているようです。またその調査で、オブジェクト指向言語は、Web ベース・アプリケーションでプログラマが優先的に選択し、使用が簡単なオブジェクト・プログラミングであり、スクリプト言語であると示しています。今後、オブジェクト指向をきちんと勉強したソフトウェア・エンジニアが益々市場に参入してくるため、新規の開発では、オブジェクト指向テクノロジーが、唯一高い評価を受けた基準のテクノロジーとなるでしょう。

まとめ

リレーショナル・データベースは、すき間市場に参入しているオブジェクト・データベースと共に、データベース市場を支配し続けるかもしれません。あるいは、現在使用されている複雑なオブジェクトを、より適切に処理できるオブジェクト・データベースが、市場シェアを拡大するかもしれません。しかし、別の可能性もあります。データベース・テクノロジーが真のハイブリッド製品を発明し、リレーショナル・インタフェースとオブジェクト・インタフェースの両方の利点を提供することです。これは可能です。実際、InterSystems による Caché は、すでにこの条件を満たす製品のひとつです (Caché データベースは偶然にも、自身をリレーショナル・データベースともオブジェクト・データベースとも表現せず、ポスト・リレーショナル・データベースと位置付けています)。使用中の製品が、リレーショナルあるいはオブジェクトのいずれのカテゴリに分類されようが、データベース・ベンダは、ハイブリッドの方向にシフトしていくでしょう。

このためには、開発者がデータベースにアクセスできるように、マッピング層を持つデータベースが必要です。マッピング層はオープン・スタンダードを基準にし、インピーダンスミスマッチの問題を解決します。その後データベースの呼び出しは、SQL、オブジェクト・クラスへの直接要求、あるいはクラス・コレクションのいずれかで行います。マッピング層は、これらの呼び出しをデータベースへの物理的なデータ要求に変換し、データを取得します。これにより、インピーダンスミスマッチを未然に防ぐことができます。

いずれの場合でも、機能を変更するためにデータベースを変更することは簡単ではありません。オブジェクト・データベースは、大量のデータ・セットから選択されたクエリを取得するような高速なインデックス機能が必要です。クエリの取得に最適なリレーショナル・データベースは、ビットマップ・インデックスを使用しますが、データの更新時にオーバーヘッドを生じることがあります。このためオブジェクト・データベースは、ほとんどこの機能を持ちません。この点においてリレーショナル・データベースは、柔軟な物理データ構造を備える必要があります。リレーショナル・データベースの発展により、物理レベルでテーブルを実装するようになってきました。リレーショナル・データベースは、柔軟性のない制約を排除し、変則的なデータ構造を格納できなければなりません。これにより、データベースを使用する利点は大幅に広がるでしょう。オブジェクト・データベースとリレーショナル・データベースのそれぞれの利点を組み合わせると、以下のようになります。

- 最適なトランザクショナル・パフォーマンス
- 複雑なデータの管理
- 管理のし易さ
- 迅速な開発

- 柔軟なクエリ機能
- 標準データ・アクセス・インタフェース
- ビジネス・インテリジェンス・アプリケーションへの適合性

このように利点を組み合わせることで、すべてのアプリケーションに適合する 1 つのデータ・セット定義を持つデータベース・エンジン開発への可能性が広がります。したがって Baroudi Bloor は、産業にハイブリッド・データベースが必要だと確信しています。ベンダは、柔軟な態度でハイブリッド・データベースへの移行を受け入れる必要があります。そうしなければ、COBOL やパンチカードと共に衰退の一途をたどることになるでしょう。

Copyright 2003, 2004 Baroudi Bloor International, Inc.

この文書は、IT 業界の研究、分析、戦略顧問会社である Baroudi Bloor の Robin Bloor の記事を翻訳したものです。

Robin Bloor は、Baroudi Bloor International Inc の調査担当かつ Bloor Research の社長です。彼は、世界でも優秀な IT アナリストの一人であり、世界中の IT ユーザやベンダ組織に、研究と分析結果を配信するコンサルタント組織です。 のメール・アドレス: robin@baroudi.com