ORACLE® *Oracle Press*™

ORACLE
24×7システム構築ガイド 上

ORACLE

24×7 Tips & Techniques

Venkat S. Devraj／著
SE編集部／訳
日本オラクル株式会社／監修

SE
SHOEISHA

本書内容に関するお問い合わせについて

このたびは翔泳社の書籍をお買い上げいただき、まことにありがとうございます。弊社では、読者の皆様からのお問い合わせに適切に対応させていただくため、以下のガイドラインへのご協力をお願いしております。下記項目をお読みいただき、手順に従ってお問い合わせください。

●ご質問される前に

弊社Webサイトの「Q&Aコーナー」(http://www.shoeisha.com/info/help.asp) をご参照ください。これまで受けたご質問への回答 (FAQ) や、的確なご質問方法に関する情報を掲示しています。

●ご質問方法

弊社Webサイトの質問専用フォーム (http://www.shoeisha.com/book/qa/) をご利用ください。記載漏れや独自の用紙等によるご質問、お電話や電子メールによるお問い合わせ、本書にはさみ込まれたアンケートはがきに記入されたご質問等は、お受けしておりません。

※質問専用シートのお取り寄せについて

Webサイトにアクセスする手段をお持ちでない方は、ご氏名、ご送付先 (ご住所／郵便番号／電話番号またはFAX番号／電子メールアドレス) および「質問専用シート送付希望」と明記のうえ、電子メール (qaform@shoeisha.com)、FAX、郵便 (80円切手をご同封願います) のいずれかにて“ 編集部読者サポート係 ”までお申し込みください。申し込まれた手段によって、折り返し質問シートをお送りいたします。シートに必要事項を漏れなく記入し、“ 編集部読者サポート係 ”までFAXまたは郵便にてご返送ください。

●回答について

回答は、ご質問いただいた手段によってご返事申し上げます。ご質問の内容によっては、回答に数日ないしはそれ以上の期間を要する場合があります。

●ご質問に際してのご注意

本書の対象を越えるもの、記述箇所を特定されていないもの、また読者固有の環境に起因するご質問等にはお答えできませんので、予めご了承ください。

●郵便物送付先およびFAX番号

送付先住所　：〒160-0006 東京都新宿区舟町5
FAX番号　：03-5362-3818
宛先　：(株) 翔泳社出版局 編集部読者サポート係

CONTENTS

まえがき

企業は、基幹業務用の複雑なアプリケーションを配備し、業務要件を満たしています。しかし、組織全体のユーザーが現在享受しているこのようなアプリケーションの可用性は、見直されつつあります。Oracleデータベースは、このようなアプリケーションの心臓部にあります。

可用性の方程式には、コンピューティングプラットフォーム、周辺装置、ネットワーク、データベースアプリケーションなど、それぞれが共生の関係で機能しなければならないさまざまな要素を含める必要があります。システムベンダーの助けを借りている顧客は今まで、この方程式の中のハードウェア部分を重視してきました。「システムのパフォーマンスに信頼性がないと、アプリケーションの可用性を実現できない」というのが、彼らの主張です。それと同様に、ほとんどの組織のITの実装でネットワークのインフラが果たす役目が重要になるにつれて、ルーターやハブなどネットワークに固有な要素の可用性もますます重視されるようになっています。しかし、企業における実装が進化し、アプリケーションソフトウェアがビジネスプロセス自体と統合されるケースが非常に増えているため、現在ではデータベースの可用性が非常に重視されるようになっています。

可用性に関する要件の実現が非常に切迫しているため、さまざまな組織では、複数レベルの冗長性(障害の発生を回避するための一連の機構)を備えたシステム設計に十分な注意を払うべきであると気付き始めています。ネットワークやシステムのアップタイム自体が長くても、アプリ

ケーションの可用性が最大になるとは限りません。可用性の方程式では、データベースシステムがますます重要なコンポーネントになりつつあります。高可用性を目指すアーキテクトは、信頼性を確保するために、ハードウェアやネットワークを超えた部分に目を向け、データベースの可用性に固有なテクニックや最適な実践手法を探し始めています。

アプリケーションの可用性とパフォーマンスを向上させるという強い要求の背後には、次のような市場での推進要素があります。

■ **インターネット**　インターネットほど強力な推進要素はおそらくないでしょう。あらゆる規模、あらゆる業種の組織が、インターネットによって、24×7、つまり24時間365日、ITに依存した世界に参入せざるをえなくなっています。今日、電子商取引による革命は、ビジネスモデルを文字通り一夜にして変えてしまいます。顧客は、Webベースのサーバー、ネットワークデバイス、アプリケーションがいつでも利用可能な状態にあるものと考えています。株式取引用のWebサイト、インターネット・サービス・プロバイダ（ISP）、あるいは消費者オンラインサービスで障害が発生すると、ダウンタイムが発生します。そのような障害が発生すると、今では国中に広がるニュースのネタとなり、当の企業の名声はたやすく地に落ちることになります。世界中のユーザーからアクセスされるオンライン受注エントリシステムのダウンタイムは、非常に目立つ場合もあります。今では、組織の境界や地域的な境界を越えて展開されているアプリケーション同士の統合および相互の依存関係が、より緊密になっています。そのため、アプリケーションの可用性がさらに重要性を増しています。

■ **ビジネスプロセスとITの統合の増加**　可用性というニーズに対するインターネットの影響に匹敵する推進要素として、ビジネスプロセスとITの統合があります。そのような統合は、ビジネスプロセス自体が、それを可能にするアプリケーションから実質的に区別できないほどのレベルにまで到達しています。たとえば、顧客情報の保存や取り出しを組織で実行する手段として、従来はフォルダやファイルキャビネットを使っていました。しかし、現在では、それらの領域をデータウェアハウスやデータマイニング用のアプリケーションがカバーしており、アクティビティ全体が以前とは大きく変わっています。このような変化に伴い、アプリケーションの重要性がますます高くなっています。そして、システムが貧弱であると特定のマーケットで商機を失う場合がある、というレベルまでその風潮は進んでいます。

■ **市場やビジネスのグローバル化**　可用性に対する強い推進要素として、ビジネスや組織の急速なグローバル化という要素もあります。今日では、たとえばシドニーにあるイントラネットサーバー上のデータベースが利用不能になると、ロンドン、ニューヨーク、東京での業務運営に支障を来たす場合もあります。これは、ほんの数年前に比べて、ダウンタイムの影響がはるかに大きいことを示しています。当時は、多国籍性の非常に強い企業でさえ、一連のアプリケーションをローカルなサイトや地域限定のサイトにしか実装していませんでした。グローバルな通信を実現するために配備するシステムは、24時間利用可能な状態にある必要があります。アクセス性や信頼性に対する需要が増大しているため、保守作業用に通常の時間枠を設けるというやり方は急速に減っています。

グローバルなインターネット時代での競争力をユーザーに提供するため、Oracleではチューニングを柔軟に行え、可用性の高いデータベースエンジンを提供しています。Oracleの高可用性（HA：High Availability）機能を活用し、ダウンタイムのリスクを最小限に抑えるには、広範囲にわたる計画を実施し、最適な実践方法を使用する必要があります。Oracle ConsultingのSystems Performance Groupでは、HAアーキテクチャの設計、システム管理プロシージャ（System Management Procedures）、HA妥当性チェックの分野でサービスを提供し、目標のアップタイムを顧客が実現できるようにサポートしています。これらのソリューションは、基幹業務環境でのダウンタイム（予定済みあるいは突発的）のリスクを最小限に抑えるためのものです。

現在、さまざまな企業では可用性の重要性に気付き始めています。そのような絶好の時期に、本書は登場しました。著者は、予定済みあるいは突発的なダウンタイムの管理に対して、非常に包括的にアプローチしています。本書は、読みやすく、リファレンス形式になっているため、Oracleデータベースの配備を検討している人にとっては貴重な参考書になります。このようなテーマのもとに、最適な実践方法とサンプルが広範囲にまとめられたケースは初めてです。本書は、すべてのOracle DBAだけでなく、通常の環境用のアプリケーションを展開する任務を負ったシステムアーキテクトやアプリケーション設計者にとっても貴重な情報源になります。高可用性を実現するための詳細で専門的な方法やテクニックは、Oracleのあらゆる環境の設計および操作を実施するうえで指針となります。

本書の著者は、「可用性を確保するには、広範囲にわたる能動的な計画と実績が必要である」ということを実証するうえで、非常に優れた仕事を成し遂げました。高可用性を実現するためのソリューションの展開を検討しているDBAには、本書に掲載されている実例とヒントが非常に役立つはずです。掲載されている情報のほとんどが、サンプルを使ってうまく記述されています。高可用性を実現するためのソリューションを設計する際、データの消失は許されません。また、作業を進めていく過程で問題が生じても、増分的な改善を行うこともできません。最初から的確に処置を施しておくと、ダウンタイムのコストを大幅に減らすことができます。本書は、予定済みおよび突発的な機能休止を避けるための貴重な参考書になります。機能休止の検出、通知、回避、リカバリのためのテクニックが満載されており、さまざまなレベルの可用性を実現しようとしているすべてのOracleユーザーに役立ちます。

99.999%の可用性を実現するうえで、本書は必携の書といえるでしょう。

Oracle Services
Systems Performance Group
Director
Ravi Balwada

謝辞

不朽の作は、誰かの援助なしには実現できません。私からすると、本書は控え目に言っても不朽の作です。本書の執筆には、約1年かかりました。本書には、データベース業界での10年にわたる経験が盛り込まれています。経験は通常、1週間の労働時間を40時間とした勤務年数で判断します。しかし、私は何年間にもわたって、優にその2倍を超える時間に全力を投じてきました。したがって、本書にはデータベース畑の中核で働いた10年をはるかに超える経験とノウハウが詰まっていると、自信を持って断言することができます。

私はこれまで、非常に才能豊かな人々と働くことができ、非常に感謝しています。先ほど述べたような非常に長い期間、そのような仲間といっしょに働くことができたため、彼らの経験と先見性に浴し、通常であれば犯していたはずのミスの多くを未然に防ぐことができました。本書の執筆中も、私は現場で働き、可用性とパフォーマンスに影響を与える問題に毎日取り組み、それらの問題を皆で克服しました。このように執筆に専念しないというアプローチをとったため、本書が遅れたのは必定の結果です。しかし、そのおかげで本書が技術的により豊かな書籍になったと、私は確信しています。Oracleサーバーとの深夜にまで及ぶ闘いについて、本書の随所に記述してありますが、それは、Oracleサーバーを次のレベル、つまり「本当の意味での」24×7モードに昇格させるための闘いでした。

まず、本書が日の目を見ることができるように援助してくれたJeremy Judson、Carolyn Welch、

Monika Faltiss、Scott Rogers、Robert Campbell、Lee HealyなどのOsborne/McGraw-Hillのチームに感謝いたします。JeremyとScottの情熱は、本書の執筆に対する私の情熱と完全に一致していました。Monikaの手際の良さと堅実なアドバイスのおかげで、仕事がはかどりました。Carolynのガイダンスのおかげで、私は原稿を完成させることができました。チームの皆さん、ほんとうにありがとう。単なる願望から生まれた本書が、夢から現実のものとなりました。

また、Oracle CorporationのPeter UtzigとRavi Balwadaにも感謝の意を表します。Peterは本書の技術的レビューアを務めてくれ、彼の洞察力に満ちたコメントと技術的な貢献には非常に感謝しています。Chapter 9の図9-5は、彼が描いた図の1つがベースになっています。Ravi Balwadaは、本書のために、示唆に富んだ「まえがき」を書いてくれました。また、Robert Deszellにも、非常に感謝しています。彼は、本書の一部を技術面でレビューしてくれました。さらに、Oracle Corporationの多数の技術スタッフ、特にOracle Development、Oracleサポートサービス、Oracle Servicesの技術スタッフにも感謝しています。彼らは、Oracleサーバーに従事し、それを現在のすばらしい状態にまで拡張しただけでなく、その処置に関してさまざまな論文や書籍を記述しています。また、Oracle Corporationの内外を問わず、世界中のOracleカンファレンスでプレゼンテーションをした人も何人かいます。何年もの間、彼らの論文や書籍を熱心に読んでOracleソフトウェアを操作したため、私はOracleの製品をよく理解することができました。

よき友人であり、仲間でもあるUday Nayakは、Chapter 20（本書下巻）を助けてくれました。彼は、本書での技術的な疑問点の解決を自分の任務とし、その調査に多くの時間を割いてくれました。彼は常に、何でも知っているか、答えをどこからか探してきてくれました。

Raymond James Consultingの友人や仲間、特にMonte Malenke、Charles Livingston、Scott French、Dhiraj Soni、Manoj Gopalkrishnanにも、非常に感謝しています。彼らは、コンサルタントとして私をサポートしてくれました。また、現在あるいはかつて、クライアントサイトと会社でともに勤務してきたすべての友人や仲間、特にSunli Nair、Harbinder Saini、Ham Pasupuleti、Scott Lockhart、George Elango、Badruddin El Sadiq、Khalid Sattar Khan、Suleiman、Dave Davis、Christine Pimblett、Dave Warner、Chris Heroux、Reg Brown、Mike Martell、Peter Wenkerにも非常に感謝しています。彼らからは、非常にたくさんの事柄を学びました。

また、非常に親しい友人Anatasia Zaikina、Chandra Honnavalli、Pankaj Soni、Vivek Amin、Trupti Mehta、Anil Gajra、Deepak Dodaniにも、非常に感謝しています。私は長年、彼らに励まされて成長してきました。

さらに、良き指導者であるRainier Luistroにも感謝しています。数年前にRainierと知り合う機会に恵まれましたが、技術的な事柄であるかどうかを問わず、彼から学ぶことはまだ非常にたくさんあります。説教が非常に少なく実演のほうがはるかに多いRainierは、自分の豊かな知識を人々と共有することをためらったことがありません。実際、本書を執筆したいという思いは、多くの意味で彼との接触の結果生まれたものです。

最後に、私の家族(両親であるS. DevrajanとVatsala Devrajan、兄弟のShiv Devraj、おじのV.Balakrishnan)からもらったゆるぎない愛とサポートに感謝します。一生懸命に働き、若いうちに何かを成そうという志は、おじから感化されたものです。本書のような非常にボリュームがある書籍を執筆するという困難な旅に出発できたのは、家族の強力なサポートがあったからこそです。

はじめに

ここでは、本書の執筆中に考えていた事柄をいくつか紹介します。

本書の背景と必要性

1994年10月、Mosaic Communications（後のNetscape Communications Corporation）が、ブラウザNetscapeのベータ0.9をリリースしました。このたった1つのイベントにより、全世界のユーザーが魔法のようなソフトウェアを無料でダウンロードできるようになったのです。このソフトウェアが世界中のビジネスに多大な好機を実際にもたらすことに、この時点ではほとんど誰も気付いていませんでした。以前は研究者や学者の領分と考えられていたインターネットが、このすばらしいブラウザによって、数百万という一般人に実質的に解放されたのです。Netscapeの使いやすさは、実際に革命を起こしました。つまり、近所の主婦が子供が眠っている間にインターネットを使い、クリスマス用のおもちゃのショッピングをしたり、多忙な販売部の重役がランチタイムにはるかに離れた街中のオフィスで株の取り引きを行ったりできるようになったのです。その結果、多くの既存の企業や新しい企業は非常に巨大な市場、つまり全世界を相手にして、ブームのe-ビジネスを展開できるようになったのです。

この革命により、システムやリソースを24時間オープンでアクセス可能な状態にしておく必要性も出てきました。なぜなら、世界は文字どおり眠らないからです。「インターネットはすべてを変える」は最近のOracle Corporationのスローガンの1つであり、それは現実となっています。電子商取引の急速な成長およびWebをベースとするデータベースの増殖に伴って、世界中の企業はWebという威を借りることに躍起になっています。しかし、単に威を借りるだけでなく、競争力を維持するためには、Webを介して実際に市場に進出する必要もあります。

Web対応のデータベースを配備し、顧客からの注文を収集して処理すれば、このようなことはおおむね可能です。もはや顧客は、地理上の制約を受けることはなく、世界中に存在しています。北米の顧客が眠っている間にオーストラリアやアジアの顧客が注文する場合もありますし、その逆もあります。特定の地域だけ(「北米のみ」など)で操業している商売であっても、「昼につけ夜につけ、顧客は自分たちの都合のいいときにショッピングしている」ことに気付き始めています。このような運営では、データベースを24時間365日利用可能な状態にせざるを得ません。

つまり、企業では、「自社のデータおよび情報を人々が常に利用できる状態にしておく必要がある」と真剣に考えています。何も手を打たない企業は、ダウンタイムが1時間発生するたびに、信頼性や営業権は言うまでもなく、数百万ドルを失うことになります。そのため、IS要員は、「システムの保守・管理を能動的に実施し、全体的な可用性を確保すること」という大きな使命を痛切に感じています。

インターネットとe-ビジネスによる革命が勃発する前でも、特定のビジネスでは高可用性が切迫した要件でした。しかし、そのようなビジネス(航空業界など)にはメインフレームを購入する体力があり、それによって高可用性を確保することができました。e-ビジネスの登場によって、それらよりはるかに小さなビジネス(新規ビジネスなど)でも、経済的に生き残るには高可用性を確保せざるを得なくなっています。比較的大きなビジネスは、このような若き参入者に負けないようにするには、e-ビジネスという激戦に参加せざるを得ないという脅威にさらされています。高可用性に対する要求がこのように非常に高くなっているため、メインフレームを購入できないにしろ、同等の信頼性と可用性を必要とするビジネスが多数存在していることは、容易に推測することができます。

では、そのようなビジネスでは、どうすればよいのでしょうか。機能休止が発生すると、新聞紙上で悪評となり、顧客、売上げ、出来高、利益が減ることになります。メインフレームのような大がかりなものだけが、実際の答えなのでしょうか。いいえ、それは明らかに間違っています。中規模および低規模のマシンはどんどんよくなっており、現在では数年前のメインフレームをしのぐような容量を備えています。

重要なのは、総合的なアプローチをとることです。つまり、必要とされる可用性を正確に判断し、アップタイムに関するニーズを技術的なアーキテクチャ、アプリケーションの分析、設計・実装のサイクルに反映させる必要があります。また、スタンバイやフェイルオーバーに関する堅牢なソリューションも、施行する必要があります。保守と監視は、可用性にできるだけ影響を与えないで実施する必要があります。つまり、概念から物理的なインプリメンテーションにいたるシステムのライフサイクル全体を通して、可用性を検討する必要があります。

このような広範囲にわたるアプローチを別の角度から考えてみると、読者がDBA、アーキテクト、開発者のいずれであっても、高可用性を確保するためにはそれぞれ独自の役割を果たす必要があることになります。組織に高可用性のソリューションをうまく実装できるかどうかは、その役割および実施すべき独自の手順をきちんと把握しているかどうかによって決まります。これこそ、本書でさまざまなヒントや実際のサンプルを使用して説明しようとしている事柄です。それを参考にすると、高可用性の中核となる目標をOracleベースの最終的なソリューションに盛り込むことができます。

本書の対象読者と読み方

「24×7」というアップタイムを完璧に実現するという要件を持っていなくても、本書は十分に活用することができます。Oracleの現在の可用性を向上させたり、現在の可用性の一貫性と信頼性を上げたりするためのテクニック(標準的なドキュメントには記述されていない)を探している場合は、最適な書籍を今手にしていることになります。本書に掲載した数々のヒントでは、どの24×7サイトでも可用性とパフォーマンスが両方とも最優先事項であると仮定しています。実際、そのいずれかだけを最優先事項とすることは、難しいものと思われます。「遅いデータベースは、データベースがないのとほとんど同じくらい悪い状況」というクライアントも存在しています。また、本書では、システムを管理しやすいという点も、重要な要素であると考えています。

本書に掲載したヒントの多くでは、専門的な知識が必要になります。そのため、読者は、管理と開発に関する基本的な分野を熟知している必要があります。ただし、Oracleを知ったのが比較的最近であるという方の場合でも、本書を参考書として読めば、詳細な情報が必要となる特定分野の深い知識が得られます。また、そのようなヒントのほとんどは、技術スタッフ(DBAや開発者など)を対象として記述してありますが、技術職でない方や準技術職の方が、技術性の高い処理や人間を管理しているようなサイトも、私は見かけたことがあります。そのため、一部の章ではそういったマネジャーやスーパバイザーのためのヒントを記述し、ハードウェア、OS、ネットワーク、データベース、そしてそれらを毎日管理している技術スタッフを管理しやすくなるようなヒントも用意してあります。読者の上司が技術職でない場合、あるいは準技術職である場合は、それらのヒントを見せてあげるとよいでしょう。

技術的な役割が異なれば、本書の読み方も違ってきます。もちろん私は、最初から最後まで順番に読むことを推奨します。しかし、そのような読み方が非現実的なものであることも、私はよく知っています。

たとえば、私の場合、技術書を最初から最後まで順番に読むことはめったにありません。ほとんどの場合は、ページをぱらぱらとめくった後、興味があるトピックや必要なトピックを集中的に読みます。本書は、リファレンス形式になっているので、そのような読み方がうってつけです。先に述べましたが、本書はOracle関連の多数のトピックを参照するための常備的な手引書として使用することもできます。ただし、所定の順序に従って読みたいという場合は、役割ごとに次のような順番で読み進めるとよいでしょう。

役割	推奨される章の順序（11章以降は本書下巻に所収）
アーキテクト	1、2、3、4、5、6、15、16、17、18、20。後は興味がある章や役に立つ章
DBA	1、2、3、4、5、6、7、8、9、10、11、12、13、14、15、16、17、18、19、20
開発者	1、2、3、4、5、6、15、16、19、20。後は興味がある章や役に立つ章

最初のChapter 5 〜 6は別として、後は適宜読み飛ばし、好きな順序で読んでください。どの章にも、該当する分野での包括的なヒントが記載されています。また、ほかの章を必要に応じて参照している場合もあります。

本書を執筆した理由と本書の執筆を他人に任せなかった理由

私がなぜ本書を執筆したと思いますか？　本書は、苦悩と苦痛を経て生まれたわけですが、なぜ私はそのような行動に出たのでしょうか。私にはもともと、ほとんど1年もの間、腰を落ち着けて本を執筆するのに必要な自制心が欠如しています。学生時代は、作文が嫌いでした。職場でも、技術的な文書を記述するのが苦手です(でも、やっていますが)。最初の数カ月は実際、自分のしていることが気に入っており、自制心を働かせ、重要な章をいくつか記述しました。しかし、しばらくすると、壁に打ち当たりました。カフェインも効かなくなりました。意気消沈したために休暇が必要でしたが、そのような時間はありませんでした。私はどうにかして、執筆に数カ月間専念するというプロジェクトに自分を放り込むことに成功しました(作文が嫌いなことは、先ほど述べましたよね)。生産性よりイライラ度のほうが高い段階も通過しました。しかし、最後まで仕上げるという気持ちを自分で鼓舞し、そのような段階もなんとか通過することができたのです。

しかし、本当の意味で強い推進力になったのは、インターネットのカルチャーです。ただし、それは、e-ビジネスのブームを指しているとは限りません。金は、(私も含め、多くの人々にとって)最大の推進力です。ただ、このプロジェクトの場合は儲かりません(このプロジェクトの場合は、金の話をすること自体が無意味なのです)。専門性の高い技術書の著者は、「スティーヴン・キングやマイケル・クライトンでないなら、執筆活動で生計を立てようなどと考えてはいけない」ということでしょう。私は、テクニカルコンサルタントとして十分に稼いでいるので、食べていくのに困ってはいません。そして、本書から得られる報酬は、微々たるものです。

しかし、正直なところ、金で買えないものを公益のために提供することも必要な場合があります。常勤の仕事(「常勤」をはるかに超えて仕事をしたことは、しょっちゅうですが)と並行して毎晩遅くまで執筆できたのは、まさしくその精神のおかげなのです。インターネットの世界ではいたるところで、革新的な人々が一生懸命に働き、さまざまなものを無償で提供しています。Marc

AndreessenとNetscape Communicationsの彼のチームのことを読んだことがありますが、彼らは自分たちの「金の壺」、つまり彼らが作ったブラウザを無償で提供しています。それと同様に、Linus Torvaldsは、Linuxのソースコードを無償で提供しています。

私は、Marc AndreessenでもLinus Torvaldsでもありません。しかし、何年もの間、蓄積してきた知識の一部をほとんど無償で提供することくらいはできます。私を信じてください。ほとんど無償なのです。執筆で私が得た報酬ではたぶん、ピザ、コーヒー、ジュースやコーラをすべてまかなうことさえできませんでした。本書に記述した知識はすべて、さまざまな書籍、白書、知識の豊富な仲間から何年もかけて収集したものです。その過程では、Oracleのソフトウェアを何時間も操作したり、本番環境でなりふりかまわず真剣に作業したり、朝2時半にポケットベルに叩き起こされたりしたものです(このような事柄は、まだまだありますが)。

しかし、収集し理解した知識は、返す必要があり、私の周りの人々と共有しなければなりません。インターネットの世界の住民は、そのような精神(個人的な楽しみを公益のために少し犠牲にする)を少なくとも心には抱いています。さらに拍車を駆ける要因として、高度なトピックをカバーした書籍(データベーステクノロジのほとんどの基本的な側面に精通している人々のためのヒントとテクニックを記述した書籍であり、読者を次の知識レベルにまで上げるものと期待される書籍)を執筆する場合は、大きな責任感があります。「本書のおかげでそのような次の段階まで進むことができた!」と読者が言ってくれるのを、私は期待しています(もちろん、本書を読み終えた後のことですが)。本書を書き終えて、私自身も次の段階に進むことができました。

私は常日頃から、現場の人(常に現場の仕事に従事している人々や実際の入力作業を行っている人々)のために、理屈満載でない書籍が必要であると感じていました。さまざまなクライアントサイトで、必要なアップタイムを確保するために技術スタッフが格闘しているのを見てきました。24×7というアップタイムを確保するのに必要な情報が世の中に普及していなかったため、「高可用性に関するOracle担当員のための書籍」という隙間市場が存在していたのです。

Oracleの各コンポーネントについて理屈で詳しく説明してある本は、世の中にいくつも出回っています。しかし、そのような純粋理論を実際に知る必要がある人々や興味がある人々は、いったい何人いるのでしょうか。そのような書籍の多くには、現実世界の話が反映されていません。そのような書籍に理屈が多すぎて落胆したDBAを、私は多数見てきました。あまりに落胆したため、彼らは理屈重視のドキュメントや書籍に目を通すのを控えるようになっています。

私は、「退屈で眠くなる」ような書籍ではなく、現実世界のためのヒント(特定の状況で利用可能なヒント)が書かれた書籍を出版したいと考えていました。知識満載の本書を頑張って読み、読者が何か(たとえば、Oracleのカーネルについて理解を深め、可用性とパフォーマンスを向上させるためのチューニング方法を会得するなど)をつかんでくれるのを、私は期待しています。当たり前のことや、ほとんどの書籍に記載されているようなことは割愛し、専門的な理屈で読者を絶えず悩ませることは、意図的に避けました。その代わり、ささいな事柄を重視しました。つまり、適切に実施すればチューニングの効果が著しいような事柄を重視しました。

もちろん、そのようなささいな事柄は、大きな事柄(マニュアルなど、通常にアクセスできるドキュメントに詳しく記述されている事柄)と併せて実施する必要がありますが。ただし、ヒン

トの理論的根拠は、適度の理屈を使って説明してあります。また、各種の「舞台裏」の処理についても、詳しく説明してあります。ほかの参考書（さまざまな理屈が記述されていて、必要に応じて読む必要がある）も、適宜紹介してあります。

本書の中で、興味がわくような事例が出てきたら、そういった理屈重視の本を入手し、入念に読んでください。本書を読んだことで十分な自信がつき、高可用性を実現するための各種の手段をよく理解できたら、貴社のデータベースのアップタイムを最適化することができます。そのような状況になったら、本書は成功したといえます。総合的な自信は、経験からしか得られません。本書は、そのような経験を得るための大きな推進力になるはずです。

本書を読む前に最後の重要な注意事項を

本書を読む前に、注意事項を紹介しておく必要があります。本書には、Oracleのマニュアルに記載されていない機能（init.oraのパラメータやトレース機能など）が記載してあります。それらを掲載しているのは、Oracleの内部動作の理解を助けるという目的のためだけです。Oracleサポートサービスから明確に指示された場合を除き、ドキュメント化されていない機能は変更しないでください。そのような機能の一部は、Oracleの動作方式に直接影響を与えますので、機能と影響を正確に理解しないまま変更を施すと、データベースが破壊され、使用不能になるおそれがあります。また、Oracleサポートサービスから許可を得ないでそのような機能を変更すると、Oracleサポートサービスとの契約に違反することになり、データベースが保守の対象外になります（24×7の可用性を実現するどころの話ではなくなります）。

では、このことを前提として、Chapter 1から読み始めてください。必要な可用性とパフォーマンスが得られますように!　成功を祈ります!

1999年10月1日
コロラド州デンバー
Venkat S. Devraj

PART I

概要

CHAPTER 1

可用性に関する要件の把握

あなたは、インターネット上で書籍を販売する企業のオペレーションチームの一員であるとします。顧客は、世界中から貴社のデータベースにログオンし、オンライン書籍カタログにアクセスして注文を行っています。真夜中（あなたが最も暇な時間）でも、貴社のデータベースには数百人のユーザーがログオンしています。データベースのダウンタイムは、許されません。パフォーマンス上のボトルネックがあってもいけません。貴社のデータベースは、常時高速に稼動していることが期待されています。このような環境の場合は、データベースの保守をどのような方法で実施すればよいのでしょうか。バックアップは、いつ取得しますか。また、データベースの断片化の解消は、いつ行いますか。データベースに必要なこれらの保守作業はすべて、いつ行えばよいのでしょうか。

24×7の世界へようこそ！「24×7」という表現は、情報テクノロジ（IT）の世界でよく使われる用語であり、リソース（データベースやコンピュータシステムなど）を常時利用可能な状態を表すためにシステムおよびデータベース稼動の分野で特によく使われます。つまり、データベース／システムが1日24時間、週7日間、ユーザーにオープンな状態で利用可能な状態にあると、「24×7モード」で稼動しているといいます。

「24×7」は、多くの企業ですでに馴染みのある言葉になっています。これらの企業では、ダウンタイムを理解しており、データベースの高可用性の重要性を把握しています。しかし、アップタイムを長くするには、コストがかかります。日常的に実施するデータベースの管理タスクでさえ、大変な作業であり、広範囲にわたる計画を事前に綿密に立て、多くの作業をこなして、神によく祈る（!）必要があります。

多くの企業のオペレーションマネージャたちは昔から、24×7という完全なサポートを要求してきました。彼らは最近まで、「データベースを24×7モードで稼動させる必要がある」と言っていました。DBAたちは肩をすぼめてそれに心から同意し、「わかりました！　メインフレームを使いましょう」と言ったものです。それは、リソースを多用するOracleデータベースをミッドレンジのマシンで動作させると問題が多発したからです。

しかし、最近では、話が変わってきています。もちろん、オペレーションマネージャは今でも、データベースを常時利用可能な状態にすることに固執しています。しかし、驚くことに、DBAたちは、肩をすぼめて容易な結論に逃げることがなくなりました。Oracle8およびOracle8*i*がリリースされ、高可用性に焦点が当てられたからです。電子商取引および大規模なグローバルマーケットの急増により、DBAたちはコールドバックアップのためにさえも、データベースを無頓着にダウンさせることができなくなったのです。

完璧な世界（データベースの破壊、クラッシュ、断片化や、環境の災害などがない世界）では、スタンバイシステムがなくても（さらに言えばバックアップシステムさえなくても）、Oracle7やOracle8のデータベースだけで100%のアップタイムを実現することができます。しかし、現実世界の条件下では、高可用性を実現するための十分な措置を講じることなくデータベースシステムだけで24×7の可用性を実現することは非現実的な話です。24×7モードを維持するには、アップタイムに関する要件全体を理解し、ダウンタイムのさまざまな原因を予測して、必要な可用性を低下させることなく、それぞれの事態に備えて周到な準備をしておく必要があります。

Oracle7でもOracle8でも、高可用性を現実的な方法で実現するための機能をサポートしています。堅牢な99.99%の可用性(完全な100%の可用性とはいわないまでも)を実現するには、グローバルにレプリケーションされた非常に高価なスタンバイデータベースがまだ必要かもしれません。しかし、90%という良好な可用性であれば、Oracle7またはOracle8を使用しているほとんどの企業で、例外的な手段を講じることなく実現できる範囲にあります。でも、皆さんなら、90%という可用性だけで満足しますか。90%という可用性は、1日に2.4時間のダウンタイムに相当します。これは、10日に1日のダウンタイム、1年に36.5日のダウンタイムに相当しています。データベースの高可用性を期待している企業にとって、これがあまり魅力的な数値でないことは明白ですね。しかし、90%を超える可用性は、本当に必要なのでしょうか。本書の冒頭となるこの章では、データベースのアップタイムに関する要件に焦点を当てることにします。

この章で紹介するヒントとテクニックは、次のとおりです。

- 24×7のアップタイムが自社に必要かどうかを分析する
- 「データベース」のコンポーネントを理解する
- 予想される障害とその発生確率を分析する
- 24×7システムを実現する際の一般的な目標を理解する
- 堅牢なService Level Agreementによって目標を管理する
- 90%の可用性でよいなら必要コストの90%が不要になる
- 保守作業が可用性に与える影響を理解する
- 24×7のアップタイムを実現するにはオペレーションの厳格な管理が必要
- データベースに対する重要なアクセス時に待機できる「24×7チーム」を作成する
- 危機発生時に通知できる要員のエスカレーションリストを保守する
- カスタマ／エンドユーザーにはダウンタイムに先だって常に連絡する

組織にとっての24×7の意味

ここでは、24×7モードで稼動させる際の一般的な意味の概要について説明します。それが終わったら、ダウンタイムに弱いコンポーネントを見てみます。

24×7のアップタイムが自社に必要かどうかを分析する

アップタイムの要件を24×7に昇格させる前に、可用性に対する組織内でのニーズを評価する必要があります。そこで、2つのクライアントサイトでの実際の事例を紹介しましょう。

3年前、私はシカゴにあるクライアントサイト(仮にA社と呼びます)を訪れました。A社は、旅行・発券業界の大手で、同社が抱える数百という旅行代理店は、北米およびヨーロッパ全土に広

がっており、A社のデータベースにアクセスしていました。A社では、本番データベースのパフォーマンスが悪いという大きな問題を抱えていました。「utlbstat/utlestat」を数時間実行し、データベースのアラートログを調べたら、データベースに対して大きなチューニングを実行する必要があることがわかりました。そこで、私は、データベースをチューニングする際に実施できる「上位10項目」の概要を示した、すばらしく印象的なリストを作成しました。それらの項目は、簡単に実行できると思いました。

オンサイトのDBAは、「上位10項目」のリストを見せたとき、「そんなことはもう知っている」と言いました。では、彼はなぜ調整作業を実行しなかったのでしょうか。私は、そのDBAは相手にしないことに決め、調査結果と推奨事項を詳細に記述したレポートをCIOに提出しました。その推奨事項の一部を紹介すると、次のようになります。

- 「init.ora」の一部のパラメータを変更し、データベースを再起動する。
- データベースがダウンしている間に、UNIXボックスも再起動する（メモリの断片化がひどかったため）。
- DATA表領域およびINDEX表領域を再構築する（これも、断片化がひどかったため）。
- メイン表（1,600万行が格納されている）の4つの索引を、削除した後、作成し直す。これらの索引はすべて「フラット」であるため、大規模なレンジスキャンが実行される原因となる（この理由については、Chapter 6を参照）。
- ホットバックアップを常時取得するのを止めること（本番システムのピーク時であっても、ホットバックアップを1日に14時間連続して取得していた）。

「推奨事項を適用するには、データベースを再起動する必要があるのですか?」とCIOに聞かれました。まだまだ世間知らずだった私が「はい」と答えると、私は荷物をまとめて出て行くように言われました。

何が間違っていたのでしょうか。A社の要件は、卓越したパフォーマンスだけではなかったのです。主たる要件はデータベースの24×7のアップタイムであり、優れたパフォーマンスはそれに「付随する」要件だったのです。A社のデータベースは、切符を予約するために旅行代理店から昼夜を問わずアクセスされます。データベースが1時間でもダウンすると、収益が減るのは明白です。つまり、A社では、24×7のアップタイムが本当に必要だったのです。

そのすぐ後で、ウッドリッジにある別のクライアントサイト（B社と呼ぶことにします）を訪問する機会がありました。B社では、ケーキやお菓子を製造していました。仕事の初日、オペレーションマネージャは堂々と、「週末も含め、24×7の完全な可用性を期待している」と私に言いました。ケーキ製造会社の従業員が、土曜日の夜にOracleデータベースにアクセスしなければならない理由は何だろう、と私は不思議に思いました。

一方、私は、UNIXの「cron」コマンドを使用してデータベースを定期的にポーリングし、使用パターンを記録するような効率のよい（リソースの消費量が最少という意味）スクリプトをいくつか作成しました。それらのスクリプトは、1週間（週末も含む）の間、10分ごとに実行しました。収

集した情報を、**表1-1**に示します。

私はオペレーションマネージャのところにこのレポートを持っていき、使用時間のほとんどが通常の勤務時間帯にあることを説明しました。本当は、24×7のアップタイムは不必要だったのです。業務上のニーズが実際に存在していないにもかかわらず24×7モードの稼動を実現しようとすると、データベースを管理する際の柔軟性が失われることを、私は彼に説明しました。付け加えるまでもありませんが、このコンサルタント業務もあまりうまくいきませんでした。皆さんは、上級マネージャの耳に、彼らが聞きたくない事柄を入れようとしたことはありますか？　気が付いてみると、私は自分のオフィスで次の仕事を待っていました。

これら2つの例を見ると、A社の場合はデータベースの24×7のアップタイムが本当に必要であり、B社の場合は24×7のアップタイムが必要であると間違って信じ込み、それを実現しようとしていたことになります。では、システムを24×7モードで稼動させる場合のコストを見積もってみましょう。

24×7のアップタイムの「コスト」の評価

通常、24×7モードでシステムを稼動させると、次のようなことが悪影響を受けます。

管理上の柔軟性

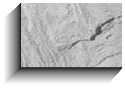
まず、通常は当たり前と考えられているタスクが、ほとんど不可能な状態になります。そのようなタスクとしては、次のようなものがあります。

- データベースまたは基礎となるオペレーティングシステムへのパッチの適用
- コールドバックアップ
- 表領域および表の再編成
- 索引の作成／再構築
- VALIDATE STRUCTUREオプションを使った表／索引の分析

時間帯	使用パターン
午前8時～午後6時	同時ユーザーセッションの平均個数は60個
午後6時～午後8時	同時ユーザーセッションの平均個数は15個
午後8時～午後10時	同時ユーザーセッションの平均個数は6個
午後10時～午前0時	同時ユーザーセッションの平均個数は2個
午前0時～午前4時	ログオンしているユーザーの人数は0（ホットバックアップが実行中）
午前4時～午前7時	ログオンしているユーザーの人数は0（ホットバックアップが完了。アクティビティはなし）
午前7時～午後8時	同時ユーザーセッションの平均個数は10個

表1-1　B社の24時間における使用パターン

実際、重要な表に排他ロックをかける必要があるタスクや、データベースインスタンスまたはマシンの再起動を実行することができません。本番稼動時のピーク時にマシンがクラッシュでもしたらとんでもありません(特に、この7カ月間、マシンを再起動していないのですから)。マシンとデータベースを純粋な気まぐれで誤用しているサイトを、私は多数見てきました。そのようなサイトではパフォーマンスに関する各種の問題を抱えており、たとえばマシンをリブートさえできれば、断片化されたメモリを開放することが可能だったのです。つまり、高可用性という要件によって影響を最初に受けるのは、管理上の柔軟性なのです。



注意

マシンやデータベースの再起動を、頻繁にあるいは気まぐれに実行するよう推奨しているわけではありません。実際私は、マシンをできるだけ長く稼動状態にしておき、パフォーマンスの大きな低下を防ぐよう強く推奨しています。しかし、ハードウェアやソフトウェア(OS、Oracle、またはその他のソフトウェア)のバグが原因で、深刻なメモリの断片化やメモリリーク(メモリが、プログラムによって常に消費され、解放されないような状態。malloc()に対応してfree()がコールされない場合や、ポインタが間違っている場合に発生することがある)が発生する場合もあります。この問題を解決するには、相当な量の分析とトラブルシューティングを実施して原因を調べ、該当するベンダー(ハードウェア、OS、Oracleサポートサービス)といっしょに作業を進める必要があります。また、複数のベンダーと同時に作業を進めなければならない場合もあります。多くの場合は、問題点がすぐには解決せず、ベンダーがパッチを用意するまで待つ必要があります。そのような場合は、事前に決定されている周期(4〜6カ月に一度など)でマシンを再起動することが(浪費されているメモリや断片化されているメモリをすばやく解放するための)必要悪になります。ただし、再起動の決断は、断片化のレベルを注意深く評価してから行うようにしてください。

このような事柄を考えると、Oracleを24×7モードで稼動させることは、非常に厳しいタスクではないでしょうか。これには簡単な解決策はありません。詳細な要件分析を行い、堅牢なアーキテクチャを使って設計し、実装を適切に実施すれば、期待される所定の可用性およびパフォーマンスを満たしてOracleは十分に稼動することができます。しかし、高価なハードウェアを組み立てて、組織の期待をすべて満たそうとしても、それは無益な結果になります。まずパフォーマンスを実現してから、中核の要件である可用性を実現する必要があります。

パフォーマンス

現実世界の条件下では、パフォーマンスとデータベースの可用性が**図1-1**のように反比例している場合がよくあります(常にではありませんが)。RAID 0は、この反比例の関係の古典的な例です。RAID 0の場合は、パフォーマンスを向上させるために、データが複数のディスク／コントローラにわたってストライプ化されます。しかし、RAID 1などのミラー化機構が存在していないと、どのディスクも単一障害点になります。関与するディスクの個数が増えるにつれ、障害の発生確率が高くなります。これをデータベースの場合で考えてみると、データベースのチェックポイント(変更済みのデータブロックバッファをディスク上のデータファイルに書き込むプロセス)の個数を減らすと、物理的なI/O処理が減るので、パフォーマンスが改善されます。しかし、

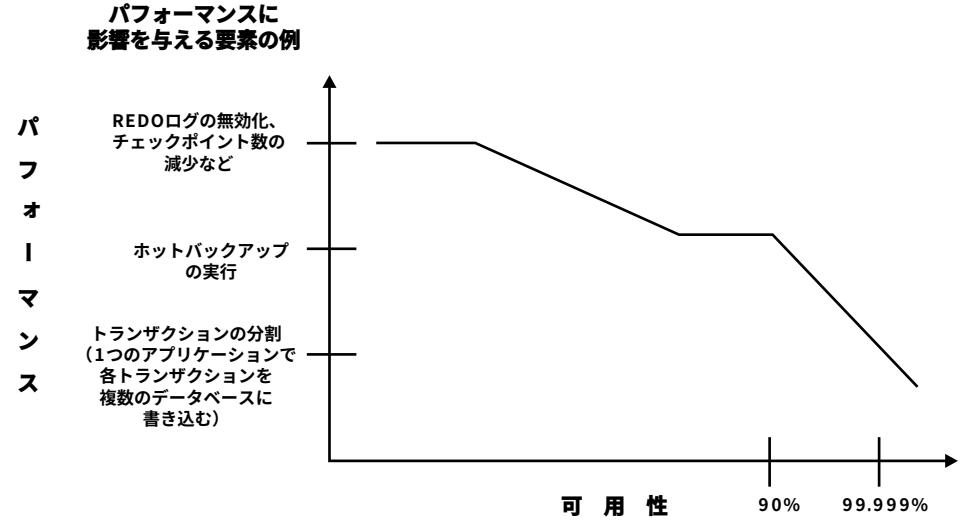
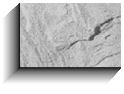


図 1-1　データベースのパフォーマンスと可用性の間の現実世界での反比例の関係

チェックポイントの個数を減らすと、データベースがクラッシュした場合のリカバリにかかる時間が長くなるので、ダウンタイムも長くなります。

データベースの場合の例をもう1つ挙げると、UNRECOVERABLEモードまたはNOLOGGINGモードで索引を作成すると、索引を速く作成することができるので、パフォーマンスが飛躍的に向上します。しかし、索引を作成した直後にデータベースがクラッシュした場合は、これらの句を指定して索引を作成したために、データベースのダウンタイムが長くなります。その場合のリカバリプロセスでは、新しく作成した索引がリカバリされません。そのため、データベースを利用可能な状態にする前に、該当する索引を手作業で作成し直す必要があり、ダウンタイムが長くなります(UNRECOVERABLE/NOLOGGINGを指定した場合にダウンタイムが長くなる理由については、後述の段落で詳しく説明します)。

もちろん、例外はいつでも存在しています。たとえば、索引をすばやく作成することができると、行が数百万個も存在している表の場合は特に、(CREATE INDEX文が設けた排他ロックがリリースされるやいなや)該当する表にすべてのアプリケーションからそれだけ早くアクセスできるようになります。この観点から見た場合は、UNRECOVERABLEモードまたはNOLOGGINGモードで索引を作成すると、可用性が実際に向上します(ただし、先ほど述べたように、索引を作成した直後にデータベースのクラッシュが発生しない場合)。



注意

パフォーマンスと可用性は、定義上では反比例の関係にありません。理論的にいうと、可用性はパフォーマンスのサブセットです(可用性がゼロの場合は、スループットもゼロです)。しかし、実用面で考えると、パフォーマンスと可用性は別々の事象です。パフォーマンスは一般に

「データベースの速度」と関係があり、可用性は「ユーザーがデータベースにアクセスしなければならない回数」と関係があります。この観点から見て、パフォーマンスと可用性の反比例の関係を、例（UNRECOVERABLEを指定した場合とチェックポイント処理の場合）を使って説明します。

では、ここでちょっと遠回りをし、UNRECOVERABLE/NOLOGGINGを指定した場合にダウンタイムが長くなる場合があるのはなぜかについて考えてみましょう。これから紹介する例は、Oracle8*i*より前の環境の例です。索引の作成中は、該当する表がロックされるため、それと同時にDML（insert、update、delete）を実行することができません。そのため、Oracle8*i*より前のバージョンでは、表の索引を作成している期間、その表を利用することができません。その表がアクセス頻度の高い表の場合は、その表にアクセスするすべてのアプリケーションが実質的に動作不能になります。そして、そのようなアプリケーションがないと、エンドユーザーはデータベースにアクセスすることができません。では、**表1-2**と**表1-3**の2つのシナリオを見てみましょう。

シナリオ1

イベントのシーケンス	発生したダウンタイム
UNRECOVERABLE（またはNOLOGGING）モードでの索引の作成	30分
データベースのクラッシュ	
前夜のバックアップを使用したデータベースのリカバリ	30分
データベースをオープンし、データベースが利用可能な状態になる	
新しく作成した索引の消失	
索引の再作成	30分
ダウンタイムの合計	**90分**

表1-2　UNRECOVERABLE/NOLOGGINGモードで索引を作成した場合のイベント

シナリオ2

イベントのシーケンス	発生したダウンタイム
RECOVERABLE（またはLOGGING）モードでの索引の作成	40分
データベースのクラッシュ	
前夜のバックアップを使用したデータベースのリカバリ	30分
データベースをオープンし、データベースが利用可能な状態になる	
新しく作成した索引は存在している	
ダウンタイムの合計	**70分**

表1-3　RECOVERABLE/LOGGINGモードで索引を作成した場合のイベント

これら2つのシナリオを見ると、UNRECOVERABLEを指定した場合は、パフォーマンスが向上するものの、ダウンタイムが長くなる可能性もあります。また、RECOVERABLEモードで索引を作成している2番目のシナリオでは、索引を作成するための実際の時間が長くなりますが、データベースがクラッシュした場合のダウンタイム全体は短くなります。

24×7のアップタイムという要件を満たし、しかも管理上の柔軟性が欠如しているためにパフォーマンスを妥当な範囲で維持するのに必要な仲介操作を行えないため、パフォーマンスは低下する傾向にあります。データベースを24時間365日稼動させなければならないためパフォーマンスが悪いのは当たり前であると中途半端に妥協したり(あるいはユーザー／幹部を無理やり妥協させようとしたり)、良好なパフォーマンスと高い可用性を両立させるのは無理であると考えたりする前に、初期のインストールおよび構成処理の時点から可用性とパフォーマンスを双頭の目標として適切な手順を実行することが重要です。可用性だけを重視してはいけません。パフォーマンスの悪いデータベースでは、スループットも応答時間も向上しません。実際、パフォーマンスが所定のレベルを下回ると、ユーザーは「利用可能」な状態にあるとみなさなくなる場合もあります。私が説明しようとしているのは、その点にあります。

パフォーマンスと可用性は、反比例の関係にある場合がよくあります。しかし、そこにこそ、DBA、システムアナリスト、データモデラー、アーキテクト、または開発者として講じることのできる具体的な手順が存在しています。そのような手順を実行すると、可用性とパフォーマンスを、一方のために他方を損ねることなく、両方とも最大限にすることができます(それらの手順としては、STORAGEパラメータの適切な指定方法、分析、表領域内のセグメントの断片化の低減、頻繁な保守の低減、パフォーマンスの低下防止などがありますが、それらの手順については以降の章で説明します)。それらの手順の多くは、能動的に実行する必要があります。また、貴社に合った24×7をうまく実装するには、広範囲にわたる分野(データベースの内部および外部)をよく理解しておく必要もあります。また、信頼性の高い無人監視(自動スクリプトが多数の監視処理を実行し、問題となる危険性がある場合は管理要員に通知する)も必須です(これは、人為的なミスに対する免疫として、管理要員が多数存在している企業にもお勧めです)。可用性とパフォーマンスのギャップを広げてそれらの関係を必要以上に反比例の関係にすることなく、可用性とパフォーマンスのギャップを実際に埋めることができるのは、このような措置を講じている場合だけです。

金銭面でのコスト

管理上の柔軟性とパフォーマンス以外に、これまでの段落を読むと、コストも大きな要因になっています。高可用性を実装するには、多額のコストがかかる場合もあります。コストと可用性の間には、直接的な関係があります。必要な可用性が高くなるほど、コストも大きくなります(**図1-2**を参照)。たとえば、1週間のうちの5日間、午前9時～午後5時の間だけデータベースが利用可能な状態にあればよく、システムまたはデータベースの問題が原因でデータベースが利用できない場合もあるものとします。それでも、会社の収益や生産性が大きく落ちない場合、データベース運営要員は、データベースを単一のマシン上で動作させるような構成にし(つまり、スタンバイマシンは不要)、通常のバックアップ処理で午前9時～午後5時という可用性を可能な限り

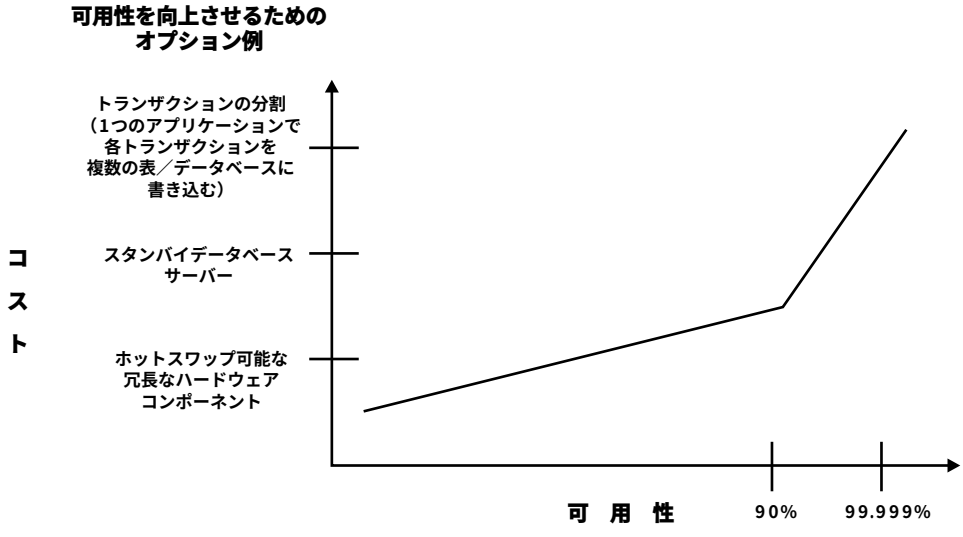


図1-2　コストとデータベースの可用性の間の直接的な関係

満たすことができます。通常の勤務時間中にデータベースの障害が発生することがたまにありますが、それでも会社に被害がない場合もあります。そのような場合は、冗長なハードウェアやフェイルオーバー用のシステム／データベースをITの予算で考慮する必要がないため、コストを節約することができます。

しかし、可用性に関する要件がもっと厳しい場合、たとえば24×7や8×5（1日8時間、週5日）の可用性が必要で、そのような必須稼動時間帯にダウンタイムが発生すると会社が財政的に苦しい場合は、非標準的なシステム構築テクニックを検討する必要があるため、冗長なハードウェア、フェイルオーバー用のシステム、および管理要員の追加をITの予算に組み込むことによって、ITの予算が増えることになります。つまり、可用性に関する要件が高くなると、企業にかかるコストも高くなります。それは、そのような要件を満たすために、ハードウェア／ソフトウェア／ファームウェアを追加する必要があるからです。

可用性に関する要件の評価

24×7の「ペナルティ」（パフォーマンス、コスト、柔軟性の点）に関するこのような背景情報をふまえて、可用性に関する要件の評価方法を考えてみましょう。1日24時間、週に7日間、本当にデータベースにアクセスする必要がありますか。それとも、1日に24時間アクセスする必要はあるものの、1週間に5日だけアクセスできれば十分ですか。あるいは、1日に実際には18時間だけアクセスできれば十分であるものの、週に7日間アクセスできなければなりませんか。このようなことは、どのようにして調べればよいのでしょうか。ユーザーは常に、「いつでもすべてのも

のにアクセスできる」ことを要求します。見るたびに、データベース内で誰かが常に何かを実行しているように見えます。

そのような場合は、ユーザーと直接話すのが最適なアプローチです。ユーザーが非常に協力的な場合もあります。複数のデータベースサーバーをさまざまな場所に配備し、それらのサーバーをすべて高速なネットワークで接続してレプリケーションをほとんど瞬時に作成し、本当の意味での24×7のアクセス性を実現するには数百万ドルかかるとユーザーに打ち明けた場合は、特にそうです。蛇足ですが、「レプリケーション」という言葉を使う場合、OracleのAdvanced Replicationオプションを指していない場合もあります。OracleのAdvanced Replicationオプションを指す場合は、「Advanced Replication」と記述します。「レプリケーション」という言葉を使う場合は、OracleのAdvanced Replicationや、市販されているその他のサードパーティ製のハードウェア／ソフトウェアベースのレプリケーションソリューションも含め、レプリケーション用の製品／ツールを広く指しています。

ユーザーが少し頑固な場合に可用性に関する要件を評価するには、データベースの使用パターンに関する統計情報を収集(該当するスクリプトについては本書下巻Chapter 19を参照)し、その統計情報を添えてユーザーに説明すると、裏づけのある説明をすることができます。

一般に、土曜日の深夜や日曜日の早朝は、保守作業に最適な時間帯です。そのような時間帯には、世界規模で見て、データベースのトラフィックが(あったとしても)最少になります。日曜日の早朝に数名のユーザーがデータベースにアクセスしているからといって、その時間帯に保守作業を実行するようなスケジュールを組めないわけではありません。ユーザーはたぶん、説得すれば、別の時間帯に作業を実行してくれるでしょう。また、そのような時間帯に自動バッチジョブが動作している場合でも、ジョブを保守作業の完了後に実行したり、平日の夜間にだけ実行したりすることも可能な場合があります。しかし、セールス面や経済的な理由からそのような作業のスケジュール変更ができない場合は、24×7のアップタイムをサイトで本当に実現する必要があるのかもしれません。

24×7のアップタイムの実現を阻止することが私の目標ではありません。私の目標は、そのような高い可用性が読者のサイトで本当に必要であることを検証し、それを実現する場合のコストに備える(必要なリソースを購入するために資金を投資するか、データベースの管理上の柔軟性が欠如していることでパフォーマンス上のペナルティを認める)ようにすることです。また、アップタイムに関する要件を分析しておくと、脆弱な時間帯や領域の認識にも役立ちます。

- データベース全体が利用可能な状態になければならない時間帯は、いつですか。データベースにアクセスするアプリケーションのピーク時とそうでない時間帯は、いつですか。データベースに対して実行されるトランザクションの数が通常より増えた場合(年度末処理など)、ピークの週や月は存在しますか。
- ピーク時と非ピーク時にダウンタイムがもたらす損失は、どのくらいですか。
- データベースの一部だけを利用可能な状態とすることは、可能ですか。つまり、所定の時間帯、特定のアプリケーションをダウンさせたり、特定の表領域をオフラインにしたりすることは可能ですか。可能な場合、そのようなアプリケーションと表領域はどれですか。

- 各種のデータベースコンポーネントの間には、どのような機能的および物理的な依存関係がありますか。
- 特定のコンポーネントが利用不能な状態になった場合、各アプリケーションはどのような方法で反応しますか。
- プライマリサーバーが保守期間にあるとき、プライマリサーバーよりパワーの弱いスタンバイサーバー上で、どのアプリケーションも低速で動作を継続できますか。
- このようなパフォーマンスの低下を、貴社で受け入れることができますか。

これらの項目も、アップタイムに関する要件を分析する際に検討しなければならない重要な項目であり、ダウンタイムに対する貴社の耐久力を理解するためのものです。時間をかけて負荷の特性を把握しておくと、スループット／応答時間に関する要件、可用性に対するニーズ、および可用性を確保できない場合の損失（これが一番重要）をよく理解することができます。24×7という可用性（だけに限りませんが）を実現するには、業務方針を確立しておく必要があります。データベースの可用性を向上させるために投資する場合、損失を出すおそれがあるというのは一番重大な懸念事項です。

可用性を確保しない場合の「実際のコスト」の評価

金銭的な損失は、収益における直接的な損失と生産性の損失に照らして測定することができます（投資収益率が最適でない場合）。そして、次のようなさまざまな項目に関連付けることができます。

- ダウンタイムの間に顧客や販売の機会を失った（社会的な信用を失う場合や、報道メディアで酷評される場合もある）。
- ダウンタイムの間、リソースのコストが空費された（「隠れた」コスト）。
- リカバリにかかる時間が増え、リカバリ時に余計なマンパワーが必要になる場合がある（データベースをリカバリするために専門家を呼んだりするなど）。冗長なハードウェア／ソフトウェアに投資することで24×7というアクセス性に対して十分に準備しておけば、特定（プライマリ）システムのダウンタイムに対する耐久性は一般的に高くなるといえる。それは、フェイルオーバー（セカンダリ）システムを利用できるからです。したがって、そのようなシステム（フェイルオーバーシステム）が機能しているため、多くの場合は、社内のスタッフがリカバリを実行しても（スタッフの経験が比較的不足していてそのようなリカバリ処理に長い時間がかかっても）耐久性は高いといえる。そのため、リカバリの実行時にも可用性が確保されることになる。しかし、十分なフェイルオーバー機能を持たない通常のシステムを使用している場合は、ピークの時間帯にシステム障害が発生すると、経験の豊かな専門家（リカバリ処理の専門家であり、リカバリ処理を頻繁に実行している）を至急呼んでリカバリを実行しなければならない場合もある。そのようにすると、ピーク時の損失を緩和することができる。

■　特定のアプリケーション(医療、航空／交通制御など)の場合は、ダウンタイム時に人命が失われるおそれさえある。

ダウンタイムが貴社に与える「実際のコスト」を見積もる場合は、このような項目をすべて検討しておく必要があります。たとえば、すべての項目を検討した後、企業が受ける損失が最悪時に5,000ドル(営業時間帯のピーク時にダウンタイムが発生した場合の1時間当たりの損失)であった場合、データベースを常時利用可能な状態にしておくために百万ドル投資する必要はありますか。また、複数のアプリケーション(受注、部品表、会計など)のデータが単一のデータベースに格納される可能性がある場合は、そのようなすべてのアプリケーションからのアクセスに優先順位を付ける必要があります。データベースのリカバリやフェイルオーバーは、非常に複雑な場合もあります。そのため、すべてのアプリケーションに同一の優先順位を与えることはできません。そこで、「このアプリケーションからのアクセスが遅延すると、業務がストップする」かどうかを、判断する必要があります。

たとえば、受注の場合は、商機を逃すと損をする場合もあります。また、顧客を失うという最悪の事態になる場合もあります。ダウンタイムが頻繁に発生すると、収益が減り、株価が下落する場合もあります。つまり、次のようなことがいえます。強力な業務方針を定め、組織のニーズとダウンタイムが収益／生産性に与える影響をよく理解している場合にだけ、組織に合った高可用性システムを独自に設計することができるのです。

この場合の課題としては、組織および技術に関する広範なコンポーネントを詳しく知っておく必要があります。いずれかの分野の知識が欠如していると、ソリューションを間違えることになります。データベースの成長や使用パターンを把握するにつれて、可用性とパフォーマンスを向上させるための予測も計画も簡単に行えるようになります。24×7のアップタイムを達成する場合、「最初の段階」が非常に困難なのはこのためです。

「データベース」のコンポーネントを理解する

では、「データベース」の構成要素を挙げてください。これは、冗談ではありません。データベースを構成している要素は、何ですか。バージョンですか、あるいは物理ファイルとメモリ構造ですか。私は、DBAと面談するとき、この質問をするのがとても好きなのです。私が聞いたさまざまな答え(それは、まだちょっと紹介しませんが)を聞くと、読者は驚くことでしょう。技術的に正しいと思われる解答は、いくつかあります。しかし、私の個人的な見解は、次のとおりです。

どのデータベースも複数のレイヤーから構成されており、どのレイヤーもその次のレイヤーと同じくらい重要です(**図1-3**)。それらのレイヤーは、DBAには見えないかもしれませんが、互いに絡み合っている場合もあります。それにもかかわらず、それらのレイヤーを明確に示してみると、次のようになります。

■　ハードウェア

■　オペレーティングシステム

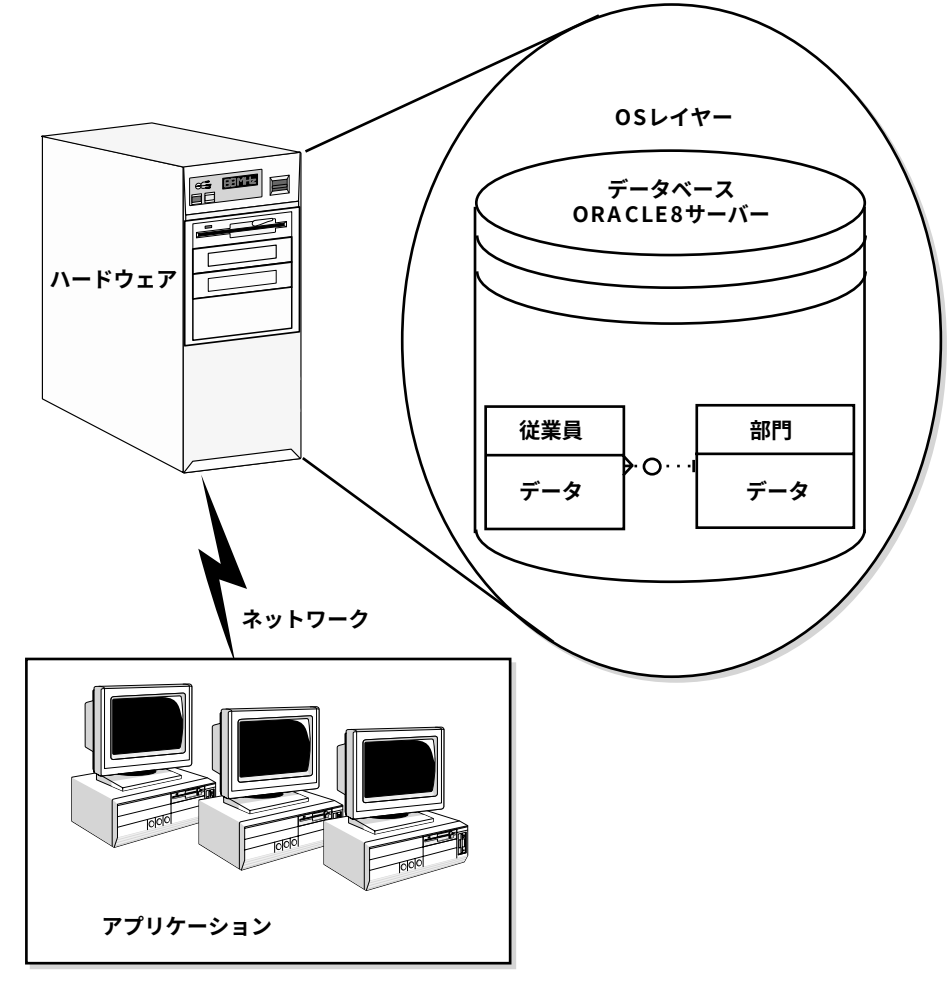


図1-3　データベースの「コンポーネント」

- ネットワーク
- DBMS（Oracle）
- DBMSを使用するアプリケーション群
- DBMSが管理するデータ

これらのいずれか1つが存在しないという状況は、考えられません。Oracle Corporationの「Raw Iron」というイニシアティブでさえ、一種のオペレーティングシステムの雰囲気を備えています（Raw IronまたはDatabase Applianceの詳細については、下巻のChapter 20を参照してく

ださい)。これらのコンポーネントのいずれかで障害が発生すると、データベースが利用不能な状態になります。24×7モードでの稼動については、これくらいにしておきましょう。

この章で1つ覚えておくべきことがあるとしたら、「データベースは多数の要素に依存しており、それらの各要素の構成処理とチューニングを別々に実行しなければならない」ということです。有能なDBAなら誰でも、「Oracleを普通にインストールすると、効率がよくない」とアドバイスしてくれるでしょう。Oracleは複雑なため、ユーザーの環境に最適なインストレーションを行うには事前の計画とカスタマイズを大量に実施する必要があります。Oracleのツール(特にNT)の多くは最近、非常にユーザーフレンドリーなものになっています。そのため、比較的新米のDBAでも、Oracleをそのまま素直にインストールし、動作させることができます。しかし、Oracleを「うまく」動作させることは、それとはまったく別物です。その場合は、これまでに説明してきたことをよく認識しているかどうかが非常に重要になります。各レイヤーを、うまく調整してスムーズに機能させ、ユーザーに意識させないようにする必要があるからです。

どのコンポーネントも、別々に調整し、DBMSと同じくらい慎重に扱う必要があります。その場合は、ちょっとしたチームワークが必要になります。

「完璧な」DBAなら、データベース管理、ネットワーク管理、システム管理、およびアプリケーション開発に関する知識を備え、データ管理についてももちろん知っておく必要があります。そんなにすべてのことを理解している人は、実在しているのでしょうか。私の基準に従えば、そのような人は1人もいないでしょう。そうなると、DBAは、優れたシステム管理者、ネットワーク管理者、開発者、データ管理者とチームを結成することになります。環境によって異なりますが、「24×7チーム」にはWebマスターも入れる必要があるかもしれません(Web上で電子商取引アプリケーションを展開している場合は、特にそうです)。

高い可用性を達成することは、あいまいなアートではなく、厳密な科学です。各コンポーネントをうまく調整して機能させれば、長いアップタイムをきっと実現することができます。では、データベースのアップタイムを各コンポーネントが低下させるケースを実際に見てみましょう。

- **ハードウェア**　Oracleのデータファイルが格納されているディスクが破壊される。その結果、データベースがダウンし、利用不能な状態になる。
- **オペレーティングシステム**　OSのスワップ領域が不足する。OSのスワップ領域が上限値に到達し、追加領域の割り当てが行えない。ページング、スワップ、ページフォールトが何度も発生した後、システムがクラッシュする。その結果、データベースがダウンし、利用不能な状態になる。
- **ネットワーク**　ネズミがネットワークケーブルを噛み切る(これは、データセンターで実際にあった話)。ネットワークがクラッシュする。クライアントは、ネットワーク経由でデータベースに接続することができなくなる。その結果、データベースが利用不能な状態になる。
- **DBMS**　頻繁にアクセスされる表に対してあまりに多数のDMLが実行されているため、索引を再構築する必要がある。索引を再構築するには、該当する表に排他ロックをかける

必要がある。そのため、該当する表を使用しているアプリケーションをすべて、ダウンさせる必要がある。その結果、データベースが利用不能な状態になる。

- **アプリケーション**　深刻なバグが本番システムに紛れこんだため、該当するアプリケーションをダウンさせ、「修復」する必要がある。該当するアプリケーションに、ユーザーは2時間アクセスすることができない。その結果、該当するアプリケーションを使用してデータベースにアクセスしていたユーザーは、データベースにアクセスすることができなくなる。
- **データ**　22個の表から構成されているデータウェアハウスがある。ソースシステムからそのウェアハウスにデータを移入するバッチジョブに間違いがあったため、スター型スキーマ（データウェアハウスで広く使われている特殊な設計のスキーマ）内のプロダクトディメンション（プロダクト詳細が格納されている表）のデータが破壊される。

24×7を実現するには、「データベースの涅槃」に到達するまでDBMSをチューニングするだけでは済みません。今まで紹介したコンポーネントをすべてよく理解し、それらのコンポーネントを能動的にチューニングする必要もあります。つまり、各コンポーネントで発生するおそれがある問題点を予測し、こうした障害に対する対処方法を理解しておく必要があります。各コンポーネント内の単一障害点をすべて認識し、そのような各単一障害点に対処するための「計画B」（実際にすぐに実施できる妥当な計画）を定めておく必要があります。たとえば、障害がすぐにダウンタイムにつながるようなクリティカルなハードウェアコンポーネントの場合は、冗長なコンポーネントを用意しておきます。そうすれば、障害が発生した場合、システムがまだ動作している最中に、処理の引き継ぎを実行したり、スワップを実行したりすることができます（ホットスワップ可能な場合）。

さて、これで講義は終わりです。先へ進みましょう。今まで紹介した各コンポーネントの構成処理およびチューニング方法については、詳しく記述されている書籍がいくつかあります。しかし、万全を期すために、Oracleの立場から見てそれらの各領域に触れておきます。ハードウェア、OS、ネットワーク、アプリケーション、およびデータ管理に関する項目については、以降の章で説明します。

予想される障害とその発生確率を分析する

適切な時期に再起動も実行せず、データベースを数カ月間も稼動しっぱなしにしておくと、どのような問題が発生するでしょうか。答えは簡単です。「多数の問題が発生します」。発生する（通常は真夜中に発生する）おそれがあるさまざまな問題を大まかに分類すると、下記のようになります。ただし、このような分類は、単なる例であり、厳密なものではありません。結局、新しい問題が毎晩発生することになるのです。

操作ミス（ユーザーのミス、管理者のミス）

このような問題は、他の問題に比べて発生する頻度が高いと思われます。操作ミスは通常、

ユーザーのミスと管理者のミスという2つのミスに分類することができます。ユーザーのミスは一般に、ユーザーが所持している権限が多すぎて、そのような権限を知らないうちに誤用していることが原因で発生します。表の削除／切り捨てをユーザーが「間違って」実行したことは、ありますか。そのようなことがあった場合は、私の言っていることがわかりますね。一方、管理者のミスの場合は、働きすぎていたり熱心すぎる（あるいは報酬が足りない）システム管理者、ネットワーク管理者、またはデータベース管理者がコンピュータルームで「アクシデント」として遭遇したことから直接的に発生する傾向があります。ネットワークケーブルにつまづいてサービス障害の原因になりかねなかったと、CIOが新米のシステム管理者に向かって金切り声を張り上げたのを見たことはありますか。

これは、「セキュリティの欠如」にもつながるおそれがあります。つまり、新米のDBAが、怠け者であることが判明したために解雇され、退社する前に間違ってデータベースを一掃するような場合です。

ハードウェア／ソフトウェアの中断と障害

すでに述べましたが、データベースが頼りにしている各コンポーネントは、バグが発生する場合もあるソフトウェア（OS、データベース）か、障害が発生する場合もあるハードウェアです。ソフトウェアのバグとハードウェア障害のいずれの場合でも、システム全体をクラッシュさせるおそれがあります。ソフトウェアのバグを検出した場合は、適切なパッチを施し、マシンを再起動する必要があります。実際、それほど特殊な問題が検出されなくても、ミッドレンジのOS（つまり、UNIXやNT）のほとんどでは、メモリ／ページフォールト、コアダンプ、過度のスワッピング、メモリリークなどによる中断がよく発生します。平均的なシステム管理者やDBAであれば、このような癖を把握し、再起動によって正常な状態をリカバリできるようになります。しかし、高可用性環境では、そのようなアプローチをとることができません。

ハードウェアを購入する場合は、下調べを事前に十分に実施する必要があります。高可用性を目指しているサイトでは、該当する新しいハードウェアに対して、実際のデータとアプリケーションを使用してベンチマークテストを長時間（品質保証期間に基づく）実施することが珍しくありません。オペレーションスタッフは、各製品の技術的な仕様をよく理解しておく必要があります。そして、各製品の理論上および実働上のMTBF（平均故障間隔。MTBFの詳細については26ページを参照）には、特に注意する必要があります。

たとえば、ディスクドライブ（ハードウェア障害の一番の原因の1つ）の理論上のMTBFが50万時間である場合は、障害が発生して交換が必要な時期になるまで、そのディスクドライブは（理論上）57年間連続して24×7モードで動作することができます。また、本番環境を忠実にシミュレートしたテスト用のプラットフォームを設けることも、珍しくありません。どのような変更（OSのパッチ、リリース、新規開発）であっても、このようなテスト用の環境でまずテストします。新しいものはすべて、十分にチェックしてから本番環境に入れます。このようなテスト環境を維持するためのコストは、高可用性を実現するためのコストに含まれる場合もあります。

環境の障害

土曜日に停電があるという旨の通知が、ビルの管理部署から最近ありましたか？　でも、心配には及びません。その仕事は、地域の公益企業が担当します。電力は、6時間後に速やかにリカバリします。電気的なサージや停電を除き、データベースのスムーズな稼動を邪魔するおそれがある環境要素としては、労使紛争やストライキなどもあります。数年前、私は製薬会社にフルタイムの外注DBAとして勤務していました。そのとき、工場の従業員がストライキに入りました。外注者はすべて、騒動が発生しないようにするために、工場内に入るのを禁止されました。その結果、私はデータベースを起動することさえできず、DBAのタスクをすべてリモートで実行する必要がありました。オンサイトで実施すべきオペレーションをすべてオフサイトで検討するなど、思いもよらないことでした。その際は、データベースのバックアップを取得することさえ、問題外でした。テープドライブのテープを交換することができなかったのですから。

自然災害と人為的な災害

地震、洪水、火事、戦争、暴動……もっと挙げますか?

これらの事柄は、いずれ発生するものです。これらの事柄に関しては、「準備はできているか。それが明日発生したらどうするか。それらすべての事柄に対して準備をしておくべきか、あるいはソフトウェア、ハードウェア、ユーザーのミスに対してだけ準備しておけばよいか」と自問してみるとよいでしょう。これらは、私がクライアントに本当に尋ねる質問事項です。通常時は24×7のアクセス性を必要とする人でも、戦争や地震の場合は24×7のアクセス性を必要としない場合もあります。これは、根本的な真実です。もちろん、国際的に活躍している企業は、たくさんあります。電子商取引の分野では、そのような企業が特にたくさんあります。その場合、先ほど述べた非常事態にも、トータルな意味で24×7のアクセス性を必要としていることがあります。それは、彼らにとって正規な業務要件なのです。しかし、貴社が小規模または中規模の企業の場合は、非常事態に対して完全な保護機構が必要であっても、数百万ドルもするようなソリューションを購入できるとは限りません。その場合は、どれが最適なソリューションなのか、十分に時間をかけて自社の目標と要件を理解してから、それに合ったソリューションを画策するようにしてください。ソリューションが組織の目標からかけ離れていなければ、うまく受け入れられ、実装できる可能性があります。

理想的な２４×７システムを目指して

ここでは、24×7システムを実現する際に望ましい一部の指標について説明し、それを実現するための具体的な手順を紹介します。その中には、他の指標より読者の組織に適しているものがあるかもしれません。そこで、ソリューションを設計する場合は、次のような指標を頭に入れて設計してください。

24×7システムを実現する際の一般的な目標を理解する

理想的な24×7システムを設計する場合は、次のような目標をできるだけ頭に置いてください。

■ **堅牢性**　24×7システムでは、システムおよびデータベースのさまざまな障害に対処することができなければなりません。すべての単一障害点を認識し、該当する各コンポーネントの冗長性を十分に確保する必要があります。

■ **透過的なフェイルオーバー**　24×7システムでは、プライマリシステムでクラッシュが発生した場合のフェイルオーバー処理をできるだけ透過的なものとし、主要なサービスの中断を防ぐ必要があります。フェイルオーバー処理をスムーズに実行できるかどうかは、プライマリシステムの障害を検出してそれに対処するのにかかる時間によって左右されます。プライマリシステムの安定性をチェックするには、定期的なポーリングが必要な場合もあります。しかし、ポーリングを頻繁に実行しすぎると、システムのパフォーマンス全体が低下する場合もあります。逆に、ポーリングの実行頻度が低すぎると、ポーリングが実行される前にサービスの中断が検出され、それがエンドユーザーに露見するおそれもあります。また、障害がいったん検出されたら、新しいデータベース接続のリルーティングおよび既存の接続の転用(再接続)を十分に実行できるほど、フェイルオーバー処理が堅牢でなければなりません。既存の接続は、アクティブな場合もありますし、非アクティブな場合もあります。アクティブな接続の転用は、実際に1つの課題になる場合もあります。24×7のソリューションでは、再起動を可能とする機能をアプリケーション(アクティブな接続を起動したアプリケーション)に組み込む必要があります。それが不可能な場合は、必要なときに再起動できるようなラップアラウンド方式のアプリケーション接続をソリューションでサポートし、処理の重複およびフェイルオーバーインスタンスに対するコールの再発行を最少化する必要があります。アクティブな接続が解除されたときは、サービスの瞬断がユーザーに露見する場合もあります。

■ **データの整合性**　24×7システムでは、データの整合性を確保する必要があります。データが消失したり、データの本来の関係に対して違反が発生したりしないようにします。たとえば、書き込まれたデータをプライマリデータベースからセカンダリデータベースにすべて伝播しているときに、データが消失しないようにします。

■ **コスト**　24×7システムを実現するためのコストとそのシステム自体のコストを足した金額は、24×7のアップタイムを実現しない場合に発生する収益／生産性の損失額より小さくなければなりません。短期間で見ると、24×7システムのコストのほうが高くなる場合もあります(帳簿上では、該当する会計期間におけるハードウェアやソフトウェアのコストが相対的に天文学的な数値になる場合もあります)。しかし、そのように高いコストであっても、それらは資本の支出であるため、時間の経過に伴って緩和される場合があります。ダウンタイムによる予想損失と24×7システムのコストを比較する場合は、この事実を考慮する必要があります。収益の損失を計算する場合は、システム障害がピーク時に発生するものと、常に考えてください。そのようにすると、最悪のケースを評価することができます。つまり、「システム障害は、ピーク時に発生するか、ピーク時以外のときに

発生する。しかし、ピーク時以外のときに発生した場合、タイムリーなリカバリを実行できないと、ピーク時の時間帯に流れ込むおそれがある」というケースを想定することができます。

- **リソースの最適な使用**　24×7システムでは、システムリソースを、上限値（システムの飽和点）に到達することなくできるだけ最適な方法で使用する必要があります。ピーク時は、特にそうです。
- **シンプルな実装**　24×7システムへの移行は、ユーザーに対するダウンタイムおよびサービスの中断を妥当な範囲（この「範囲」は、ソリューションを画策する前に、分析の一部で決めておく必要があります）に抑えて、比較的簡単に実行できなければなりません。関連するコンポーネント（アプリケーションやセキュリティの原則など）への変更も、妥当な範囲内とする必要があります。
- **保守性**　既存の社内要員は、実行不可能な手段（頻繁に実施されるオフサイト研修、複雑なツール、定期的に訪れる専門のコンサルティングスタッフ）に逃げることなく、新しいシステムの保守を実行できなければなりません。
- **パフォーマンス**　新しいシステムのパフォーマンスは、毎日計測するものとし、目標を（大きく）下回ってはいけません。所定レベルの影響は、避けられない場合もあります。たとえば、24×7のソリューションで、アプリケーションにおけるすべての書き込みを分割し、複数のデータベース（リモートサイトのデータベースも含む）に対して書き込みを実行しなければならない場合もあります。そのような場合は、各書き込み処理の実行速度が、ネットワークの遅延から影響を受けることもあります。ただし、そのような影響は、妥当な範囲内にある必要があります。

実際のソリューションで、このような目標（特にコスト）をすべて満たすことができない場合もあります。しかし、組織にとって一番重要なものから始めて、できるだけ多くの目標を実現することが肝心です。以降の章で紹介するヒントを参考にすると、現実的で実用的なソリューションを、広くはシステム全体の観点から、狭くはデータベースの観点からうまく選択することができます。特にChapter 2では、これまでに概要を説明したいくつかの状況で必要なソリューションの種類について詳しく説明します。

堅牢なSLAによって目標を管理する

SLA（Service Level Agreement：サービス品質保証）とは、可用性およびパフォーマンスに関する具体的なシステム要件を保証する契約のことです。そのような要件は一般に、エンドユーザーのコミュニティや経営幹部とディスカッションした後で得られるものです。SLAを利用すると、オペレーションスタッフとエンドユーザーのコミュニティが同一のページに基づいて議論を進め、必要なサービス品質に関する前提条件をすべて明記することができます。SLAには、一般的なシステム障害と、そこからリカバリするためのオプションを明確に記述します。また、パフォーマンスと可用性の間でのトレードオフや、高い可用性およびパフォーマンスを実現するた

めのオプションにかかるコストも記述すると、優れたSLAになります。そのような内部的なSLAは、ハードウェアおよびソフトウェアのサードパーティベンダーと自社の間で設けるほかの外部的なSLAの基礎になります。

経験からいうと、有用なSLAを作成することなくデータセンターを運営している企業は、非常にたくさんあります。SLAを作成している企業でさえ、SLAに記述されている条項にあまり注意を払っていません。しかし、データベースの高い可用性を追求するような企業には、SLAが必要不可欠です。自分のニーズを文書化してよく把握していないと、それらのニーズを満たすための方法は満足に探せません。

データベースのアップタイムに関する要件を評価する際に役に立つ一番重要な情報は、エンドユーザーからの期待です。皆さんの組織では、データ入力オペレータや中級・上級マネージャなど、さまざまなニーズや要望を抱えているエンドユーザーをサポートする可能性があるでしょう。データベースに8時間アクセスできれば十分というエンドユーザーもいます。また、VP(副社長)やCEO(最高経営責任者)のようなエンドユーザーの場合は、1日に数分しかデータベースにアクセスしないかもしれません。しかし、その数分の間に、彼らはリソース集中型のグラフィカルレポートなどのパワフルなアプリケーションを実行する場合もあります。

エンドユーザーの要望は、あいまいで気まぐれなものになるおそれがあります。そのため、皆さんは、重要な要件とそうでない要件を見分ける必要があります。また、その結果を公開しないことには、話が進みません。そのような事柄は、あなた方(オペレーション部門)と該当するエンドユーザーの間に設けるSLAに詳しく記述する必要があります。エンドユーザーが地理的に分散しており、電子商取引サイトのように容易に特定できない場合は基本的に、ハイレベルな設計明細項目に基づいて可用性に関する要件の概要をSLAに記述する必要があります。

そのようなハイレベルな設計明細項目は、初期の試行で実施したデータベースの使用パターンの監視や、一般的な目安からも導出することができます。たとえば、通常の金融アプリケーションサイトの開始時間はだいたい決まっているので、該当するデータベースの稼動時間帯は午前8時～午後6時とすることになります。そのため、初期のSLAは、この前提に基づいて作成する場合もあります。たとえば、使用パターンを2カ月監視(対話型およびバッチ型)した後では、もっと確実な推論を実施できる場合もあるので、そのような場合は実際の使用時間帯に合った新しいSLAを作成することもできます。

SLAには、次のような項目を記述することが重要です。

完全性

自社の業務に影響を与える重要な成功要因をすべて記述しておくと、優れたSLAになります。たとえば、業務を遂行するためにデータベースをどんなときでも3時間以上ダウンさせることができない場合は、そのことを明確に文書化しなければなりません。また、データベースでサポートできる同時セッションの最少個数を200とする場合も、その旨を明確に記述しなければなりません。

現実的な数字

重要な各成功要因に関連する要望はすべて、現実的な方法で文書化する必要があります。つまり、SLAには、要望を現実的な数字で記述しなければなりません。たとえば、スタンバイデータベースを街中に設置することがシステムにとって必要な場合は、その設置にXドルかかり、本番用のプライマリデータベースのパフォーマンスに（レプリケーション処理のために）Yドルの予想負荷がかかるということを記述する必要があります。XとYは、実際の金額でもかまいませんし、根拠のある推測に基づいたパーセンテージでもかまいません。そのような具体的な数字を記述しておくと、高可用性を実現する場合のトレードオフの種類をユーザーコミュニティおよび経営幹部に示すことができます。

パフォーマンスに関する何らかのトレードオフが存在しているということを記述するだけでは、同じSLAを使用しても、ユーザーコミュニティや経営幹部との間で誤解が生じるおそれがあります。あいまいな説明のままにしておくのではなく実際の具体的な数字を使用して記述すると、説得力や現実性が増します。もう一度言いますが、SLAは契約書です。そのため、「こうしたら、こうなる」と単に述べるのではなく、「こうしたら、こうなる。そうしたら、そうなる。ただし、後者を選択した場合は、コストが2倍かかる」というように明確に記述する必要があります。

順応性

SLAは、動的に変更し、常に最新の状態でなければなりません。ある組織では、SLAを作成しましたが、数カ月もするとそのSLAのことを都合よくすっかり忘れていました。その大きな理由は、SLAが厳格すぎた点です。厳格すぎたために、運用が現実に即していなかったのです。そのSLAでは、SLAの作成時に適用可能だったさまざまな事柄を考慮してありました。しかし、組織で変更がいくつか実施されたため、それらの事柄自体を静的なままにしておくことができなくなったのです。そのため、それらの変更を反映させるには、SLAを定期的に改定する必要があります。特に今のようなインターネット時代では、使用パターンに対する変更が速やかに実施されるため、SLAはそれらの変更を反映させて拡張できるよう十分に柔軟なものでなければなりません。SLAを2カ月に一度改定しなければならないのであれば、そうしてください。

たとえば、新規電子商取引サイトの場合、そのサイトを訪問する顧客は「当て推量」でしか判断することができません。そのため、初期のSLAは、その当て推量に基づいて作成することになります。しかし、数カ月経つと、実際の利用状況を示すもっと具体的な数字を使用して、SLAを改定することができます。SLAが古くなると、SLAで扱っている事柄と現実とのギャップが大きくなります。「最新のテクノロジを採用した新しいハードウェア／ソフトウェアの購入」という表現が、SLA内の一部の条項を不要なものにする場合もあります。そのような場合は、SLAをすぐに更新しなければなりません。

明白性

SLAは、それに記述されているすべての事柄について、具体的かつ明白でなければなりません。また、同一項目の解釈方法が複数存在しないように、十分な説明を記述する必要もありま

す。たとえば、「高可用性が必要である」と包括的に記述するだけではだめなわけです。予想されるアップタイムを、明確に記述する必要があります。また、保守作業のスケジュールや通知方法の概要も、記述する必要があります。同一のデータベースにアプリケーションを複数個格納する場合は、すべてのアプリケーションをカバーした説明データを用意してもかまいませんし、アプリケーションごとの情報を用意し、それらのすべてのアプリケーションを記述した上位の表を用意してもかまいません。また、可用性に関する要件を1日の中の時間帯に分けて明記しておいても、非常に役に立ちます。**表1-4**は、アップタイムに関するこのような要件を示した例です。

パフォーマンスに関する要件も記述しておく必要があります。たとえば、「所定の小さなレポート処理は、ユーザー数が150未満の場合、10分未満の時間だけ実行することができる。しかし、(ピーク時のように)ユーザー数がもっと多い場合は、30分まで実行することができる」と規定することもできます。SLAには、パフォーマンスに関するピーク時と非ピーク時の尺度、およびクリティカルなアプリケーションとそうでないアプリケーションを記述する必要があります。

要件をこのように具体的に定義しておくと、オペレーション部門ではバックアップや索引の再構築といった通常の保守タスクをいつ実行すればよいかがわかります。また、個々の時間帯に実行するアプリケーションがすべてリストされているので、DBAはエンドユーザーに与えることのできる可用性の種類(特定の時間帯にオフライン状態にすることができる表領域、オンライン状態でなければならない表領域など)を把握することができます。さらに、このような情報があると、DBAはリカバリ処理時に、最初にリストアしなければならないアプリケーションがわかります(データウェアハウスのバッチジョブをすぐに実行する必要がない場合は、データウェアハウスの表を最初にリストアする必要がなく、代わりに優先順位に基づいて販売アプリケーションの表を利用可能な状態にすることができます)。

コストおよび技術面での妥当性

SLAを有効なものにするには、コストおよび技術面での制約を考慮する必要があります。SLAに記述する高可用性ソリューションはすべて、技術的に可能(実装に使うテクノロジがすぐに利用可能)なだけでなく、コスト面でも組織で実現可能なものでなければなりません。このような条項に違反するSLAは、不当なものであり、実装に適していません。たとえば、24×7の可用性を実現するためにレプリケーション処理をリアルタイムに実行する場合、パフォーマンスに与える悪影響を最小限に抑えることが環境によっては技術的に不可能な場合もあります。また、本番デー

時間帯	アップタイムに関する要件	該当する時間帯に動作するアプリケーション
午前8時～午後6時	非常にクリティカル	資材管理、販売、財務、製造計画、製造
午後6時～午後11時	クリティカル	資材管理、製造
午後11時～午前4時	使用せず	
午前4時～午前8時	クリティカル	製造、データウェアハウスバッチジョブ

表 1-4　使用するアプリケーションのアップタイムに関する要件

タベース全体のレプリケーションを海を隔てた別のデータセンター内に作成することが、技術的に可能であっても、コスト面で企業にとって不可能であり、受け入れられない場合もあります。

SLAをうまく作成するには、オペレーション部門、上級幹部、エンドユーザーコミュニティの代表などが集まって議論することが重要です。エンドユーザーは基本的な要件を提示し、オペレーション部門は可能なソリューションを提示します。また、上級幹部は、コスト面での妥当性に注目します。オペレーションチームが技術的に妥当と見なした事柄が、コストの面で上級幹部に受け入れられない場合もあります。高可用性を実現するためのコストが、そのような可用性から得られる利便性を超えている場合は、可用性を向上させる作業が不必要になります（作業全体とはいわないまでも、コストがかかりすぎる部分は少なくとも不必要になります）。

また、ソリューションがエンドユーザーから得られる場合も、よくあります。基幹業務用の特定のバッチジョブを日中に実行するためのコストが非常に高い場合は、エンドユーザーがそのようなジョブを毎日実行しないよう選択したり、そのようなジョブを深夜に実行するよう選択したりする場合もあります。そのような提案は、エンドユーザーから出す必要があります。オペレーション要員は一般に、そのようなバッチジョブを遅延させてよいかどうかをほとんど知らないからです。

妥当な平均時間

SLAを有効なものにするには、データベースの妥当なMTBFおよびMTTRを、SLAに直接記述するか、あるいは補足的なドキュメントに記述する必要があります。「MTBF（平均故障間隔）」は、障害が発生するまでシステムが動作する平均時間です。たとえば、ディスクやネットワークの障害が原因でクラッシュするまでデータベースが600時間稼動する場合、MTBFは600時間になります。データベースのMTBFは通常、基礎となる一連のコンポーネントのMTBFによって決まります。

MTBFには、理論上のMTBFと実働上のMTBFという2種類のものがあります。「理論上」のMTBFは、コントロールされた研究開発環境で実施した膨大なテストに基づいて該当するベンダーが提供する統計情報です。現実世界で発生する突発的な出来事（管理者の操作ミスなど）は、反映されていません。一方、「実働上」のMTBFは、該当するコンポーネントで実際に発生した出来事に基づくMTBFです。

では、例を使ってMTBFを説明しましょう。ABC社のOracle8データベースはSun Enterprise 4000上で動作しており、そのデータはRAID-Sを採用した1台のEMC 3100ディスクアレイ上にストライプ化されています。このディスクアレイは、MTBFが25万時間であり、32台のディスクから構成されているものとします。この場合、各ディスクのMTBFは、7,800時間（250,000時間/32台）になります。つまり、約10カ月に一度（7,800[時間]/24[時間/日]/30[日/月]=10カ月）、ディスクはクラッシュする可能性があります。

したがって、この場合は、データベースのMTBFが10カ月を超えることはありません（データベースはそのような一連のディスクが稼動状態にあることに依存しているため）。また、データベース全体のMTBFを求めるには、ネットワークなどのさまざまなハードウェア周辺装置（CPU、

コントローラなど)といった主要なコンポーネントも考慮する必要があります。データベースのMTBFは一般に、すべてのクリティカルなコンポーネントのMTBFを超えることはありません。

「MTTR(平均リカバリ時間)」とは、障害の発生時にサービスのリストアにかかる平均時間のことです。では、先ほどの例を再度使って、MTTRについて説明しましょう。該当するOracle8データベースでは、データベースのオンラインバックアップを毎晩取得するものとします。そのバックアップ処理は、午前2時に開始され、だいたい午前4時に終了します。また、そのデータベースにはオンラインREDOログファイルが4つあり、ログスイッチが1時間当たり平均2回発生します(そのため、アーカイブログファイルが1時間に2つ作成されます)。データベースのリカバリ処理時に、各アーカイブログファイルをロールフォワードするのに、平均3分かかるものとしましょう。このとき、ディスクコントローラの障害が原因で午前11時というピーク時にデータベースがダウンしたとすると、すべてのデータベースオペレーションのリストアとリカバリを実行するのに約8時間かかります(**表1-5**)。そのため、この場合のMTTRは、8時間になります。

MTBFは、簡単に考えると、データベースが立ち上がって動作している時間の平均値です。それと同様に、MTTRは、データベースがダウンしていて利用できない時間の平均値です。

先ほどの説明からわかるように、SLAには、何らかの重要な調査、評価、およびベンチマークの結果を記述します。また、組織におけるクリティカルなリソースの主な用途も、SLAに記述します。ただし、データベースの可用性に関して真剣な企業は、そのようなリソースに投資してSLAを作成する必要があります。システムチェーン内の弱いリンクはすべて、SLAに記述します。そのような弱いリンクを完全に把握してからでないと、組織ではデータベースのアップタイムに関する要件を満たすための有効なソリューションを準備することができません。

タスク	必要な時間
破損したディスクコントローラを交換する	6時間(この時間は、ディスクベンダーと交わしたSLAによって決まる)
前夜のバックアップを使用して、影響を受けたデータファイルをテープからディスクにリストアする	30分
データベースのリカバリを実行する(14個のアーカイブREDOログファイル[7時間分。アーカイブREDOログファイルが1時間に2個作成されるため、アーカイブREDOログファイルが14個あると7時間分のバックアップに相当する]を使用してロールフォワードを実行する)	42分(アーカイブログ1個当たり3分)
データベースのリカバリを実行する(未コミットのトランザクションをロールフォワードする)	15分
データベースを再度オープンし、すべてのアプリケーションを再起動する	8分
緩衝時間(これまでのタスクの遅延を吸収)	30分
必要な総時間	**8時間(概算)**

表1-5　ディスクコントローラの障害からリカバリするためのタスク

90%の可用性でよいなら必要コストの90%が不要になる

アップタイムに関する要件を検討する場合は、「90%ルール」が役に立つかもしれません。90%しか利用できない場合は、コストを90%削減できます。つまり、24×7モードで稼動させるコストの90%は、最後の10%という可用性の実現に費やされるわけです。そのため、アップタイムに関する要件が現実世界での時間にどのように対応しているのかを理解しておくことが、非常に重要です。

表1-6は、アップタイムのパーセンテージとそれに相当する現実世界でのダウンタイムを示しています。すでに述べましたが、90%という可用性は、1日に2.4時間のダウンタイムに相当します。これは、10日に1日のダウンタイム、1年に36.5日のダウンタイムに相当しています。十分に熟練したオペレーションスタッフが監視および能動的な保守を定期的に実行していても、データベースで障害がときどき発生するのは普通のことです。ハードウェアやソフトウェアの障害がたまに発生すると、新しいコンポーネントを用意したり、パッチを当てたりする必要があります。また、台風や地震といった自然災害が発生する場合もあります。データセンターの場所によっては、このような出来事から大きな影響を受ける場合もあります。

こうした障害がまれにしか発生しない場合は、リスクを覚悟してやっていくこともできるでしょう。そのためには、障害が発生する確率と発生した場合の被害の程度を調べておく必要があります。そして、「該当する障害が毎日は発生しない場合、その対策のために会社で多額の投資をするかどうか」を自問してみてください。一般論でいうと、バックアップに関する通常の原則と堅牢なリカバリ計画さえあれば、データセンターはかなり円滑に運営することができます。つまり、システム構築に関する標準的なテクニックがあれば、通常は90%の可用性を十分に実現することができます。

以降の章で紹介するヒントを参考にすると、高可用性を実現するためのコスト効率に優れた計画を作成して、コンポーネントの障害に対処し、データベースの保守タスクを実行することができます。ただし、そのような計画では一般に、自然災害や人為的な災害をカバーすることができません。そのような出来事が発生した場合にも可用性を確保するには、多額の投資を行い、地理的に離れた場所にデータセンター全体のレプリケーションを設置する必要があります。業務要件、環境、およびデータベースの全コンポーネントを注意深く分析して、「あなた」が正しい結論に到達する必要があります。

アップタイムのパーセンテージ	1年当たりのダウンタイム
90.0	36日と12時間
99.0	3日と15.5時間
99.9	9時間
99.99	50分
99.999	5分

表1-6　アップタイムのパーセンテージとそれに相当する現実世界でのダウンタイム

保守作業が可用性に与える影響を理解する

オペレーションスタッフたちは、「保守」という言葉をよく使っていますね。来週の週末にデータベースをダウンさせたり、次の土曜日の真夜中にシステムの電源を落としたりすることは、なぜ必要なのでしょうか。その答えは、「保守のため」という一言です。オペレーションマネージャや管理者はすべて、システム／データベースの保守の重要性をよく知っています。

どのようなシステムでも、円滑に動作させるには、小さな変更や調整をときどき実施する必要があります。24×7を前提にすると、保守タスクは次の2種類に分類することができます。

- 能動的な保守
- 受動的な保守

名前からわかるように、「能動的な保守」は、データベースの実際の監視や、以前から培われてきた所定の「作法」に基づいて実施します。前者の例としては、「最大エクステントに達しました」エラーを防止するために、成長の速い表のエクステント数をその使用状況に応じて増やすことなどが挙げられます。後者の例としては、OracleのOFA（Optimal Flexible Architecture）のガイドラインに基づいて表セグメントとは別のディスクセットにその索引セグメントを格納し、それら両者の競合を防止することなどが挙げられます。

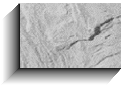
「受動的な保守」は一般に、監視していたデータベースの直前の状態または発生した個々のエラー／問題点に基づいて実施します。前者の例としては、DATA表領域に空き領域が3MBしかなく、成長の速い表の次のエクステントが5MBと大きい場合に、DATA表領域に別のデータファイルを追加することなどが挙げられます。後者の例としては、「ORA-1653：表user tableを拡張できません」というメッセージが全アプリケーションユーザーの画面に表示された後、CIOの厳しい監視下でDATA表領域に別のデータファイルを追加することなどが挙げられます。

可用性に関する要件が厳しくなると、非常に高度な能動的保守（受動的な保守が必要な場合もある）が必要となるため、絶体絶命という状況が発生しやすくなります。そのような保守を実行する場合は、ダウンタイムが余計に長くなることがよくあり、高可用性という目的を果たせなくなります。そのため、可用性に悪影響を与えることなく能動的な保守を定期的に実行するようスケジューリングすることのできる革新的なソリューションの実装が必要になります（このようなソリューションの一部については、Chapter 15〜18を参照）。もう一度言っておきますが、使用パターンを把握していないと、データベースや任意の新規アプリケーションの初期段階で必要な保守作業が増えます。この点は最初に作成するSLAに記述し、注意したほうが賢明です。

必要な保守作業はすべて把握し、SLAに記述しなければなりません。関係する技術スタッフは全員、アップタイムに関する要件を検討する前に、それらの保守作業（作業内容、必要な理由、実行頻度など）を十分に理解しておく必要があります。どのようなオペレーションを実行するとダウンタイムがなぜ発生するのか、およびダウンタイムを発生させることなく等価な目標を達成するための方法があるかどうかを理解しておくことは、非常に重要です。**表1-7**は、保守作業の例を示しています。この表には、貴社に固有な要件に従って作業などを追加してもかまいません。また、この表は、SLAに入れることもできます。

タスク	実行頻度	ダウンタイムは必要か
表の再構築（断片化の解消、行連鎖の修復）	四半期に一度か大容量なロードを実行した後のいずれか近いほう	はい
索引の再構築（断片化の解消）	四半期に一度か大容量なロードを実行した後のいずれか近いほう	はい
コストベースオプティマイザ用のセグメントの分析	毎夜	いいえ
構造の妥当性をチェックするためのセグメントの分析（データベースの破壊のチェック）	四半期に一度（破壊の兆候がある場合はすぐに実施）	はい
バックアップ－ホットバックアップ	毎夜	いいえ
バックアップ－所定の表／スキーマのエクスポート	毎週	いいえ
統計ユーティリティ（utlbstat、utlestat、その他の自社製ユーティリティ）の実行	毎週	いいえ
セグメントの特性（記憶領域パラメータ、FREELISTS、INITRANS、PCTUSED、PCTFREE、並列度など）の変更	必要に応じて	オペレーションによっては必要
データベースの静的な初期化パラメータの変更（「init.ora」のパラメータ）	必要に応じて	はい
新しい表領域の作成、既存の表領域に対する新規データファイルの追加	処理容量の見積もり（および必要）に応じて	いいえ
新しいバージョンへのデータベースのアップグレード、必要なパッチの適用	必要に応じて	はい
他の特殊なデータベースへのデータのコピー（使用中のデータソースからのデータウェアハウスおよびデータマートへのデータのロード）	毎夜	いいえ
オフラインメディアへのアーカイブREDOログの移動	毎夜	いいえ
オンラインREDOログファイルのサイズの変更、REDOログファイルの作成	必要に応じて	いいえ
データベース全体の再構築（データベースのブロックサイズの変更、別のマシンへのデータベースの移動、データベース全体の断片化の解消[データベースがそれほど大きくない場合]）	必要に応じて	はい
スタンバイデータベースサーバーへのアーカイブREDOログのコピー／適用	必要に応じて	いいえ

表 1-7　保守作業、実行頻度、ダウンタイムの必要性の一例



注意

表1-7の「実行頻度」は、該当するタスクを実行する頻度の単なる例であり、基準値ではありません。必要な実行頻度は、さまざまな要因によって決まり、サイトごとに異なる場合もあります。

準技術マネージャおよびスーパバイザーのためのヒント

これから紹介するヒントは、準技術マネージャおよびスーパバイザーのためのものです。

24×7のアップタイムを実現するにはオペレーションの厳格な管理が必要

24×7は、単なる要件ではなく、総合的な理念です。24×7を実現するには、設計者および開発者も含め、チーム全体に高い能動性と責任が必要とされます。高可用性を追求しない場合は、緩慢であってもかまいません。しかし、24×7モードの場合は、緩慢さが災害にすぐにつながる場合もあります。データベースをよく調整して最高のパフォーマンスを常に提供するのか、あるいは調整を実施しない弱いデータベースで絶えずクラッシュを発生させるのかは、アプローチによって決まります。オペレーションは、どんなに簡単なものに思える場合でも、事前に評価する必要があります。つまり、システムやデータベースにどのような影響を与えるか、重要な表に排他ロックをかけるか、データベースを使用不能にするおそれはないか、非常に多くのリソースを占有してデータベースの動作をきわめて遅くするおそれはないか、などを事前に検討しておく必要があります。

データベースに対する重要なアクセス時に待機できる「24×7チーム」を作成する

すでに説明しましたが、発生する可能性がある問題点を1人のDBAでこなすのは無理です(DBAだけで構成されるチームでこなすことさえ無理です)。DBAは、十分なサポート／オペレーションスタッフと組ませる必要があります。大切な時期は、特殊なチームを形成し、それぞれの時間帯を担当させるのが非常に重要です。

24×7チームは、DBA、システム管理者、ネットワーク管理者から構成し、該当する時間帯にオンサイトに設置するのが理想的です。そして、緊急時にすぐ連絡できる開発者チームやデータ管理者に24×7チームをうまくサポートさせる必要があります。では、例を見てみましょう。

ある企業ではWebベースの大きなデータベース(80GB以上)を3つ稼動させており、キー要員とその担当時間帯は**表1-8**のようになっていました。

表1-8の例を見ると、1日を業務に基づいていくつかの時間帯に分割していることがわかります。また、「非常にクリティカル」な時間帯の場合は、DBAとシステム管理者をそれぞれ2名以上オンサイトに置いていることもわかります。もちろん、皆さんの会社の場合は、組織の要件と関連

時間帯	業務に対する時間帯の重要性	担当要員
午前7時～午後3時	非常に重要	主（オンサイト）―― Susan M：DBA Marty G：DBA Joe A：システム管理者 Michelle P：システム管理者 Rob F：システム管理者 Dave S：ネットワーク管理者 Brian G：開発者 Peter H：開発者／データ管理者 副（呼び出し待機）―― Sam J：DBA Bill K：システム管理者
午後3時～午後7時	非常に重要	主（オンサイト）―― Sam J：DBA Lucas S：DBA Thomas A：システム管理者 Vladimir P：システム管理者 Bill T：ネットワーク管理者 Patti N：開発者 副（呼び出し待機）―― Susan M：DBA Rob F：システム管理者
午後7時～午後11時	重要	主（オンサイト）―― Sam J：DBA Lucas S：DBA Thomas A：システム管理者 Vladimir：システム管理者 Patti N：開発者／データ管理者 副（呼び出し待機）―― Marty G：DBA Joe A：システム管理者
午後11時～午前7時	重要でない	主（オンサイト）―― John L：DBA Lisa L：システム管理者 副（呼び出し待機）―― Lucas S：DBA Vladimir P：システム管理者

表1-8　時間帯別のコールリスト

する収益に合わせて、2名より多くすることも少なくすることもできます。

ただし、発生する可能性がある問題に備えて人員を投入しても、問題を妨げるとは限りません。このような24×7チームを配備しても、データベースやシステムのダウンタイムは発生することがあります。しかし、24×7チームを配備しておくと、次の2つの点で直接的な利点があります。

- 問題点を能動的に監視するため、問題点の発生を防止することができる
- 専用のリソースを利用できるので、問題の発生時、すばやく対処することができる

データベースアクティビティのピーク時と、データベースが利用不能になった場合の時間帯ごとの損失も把握しておく必要があります。午後7時～午後9時の時間帯にデータベースが利用不能になったときの損失が甚大な場合は、その時間帯にDBAとシステム管理者をそれぞれ2名以上配備する必要があるかもしれません。しかし、その時間帯の損失が取るに足らない場合は、賃金の比較的安いオペレータを1名配備したり、DBAやシステム管理者の1名にだけシステムを監視させてもよいでしょう。災害が発生した場合は、そのようなオペレータからもっと高給のDBAやシステム管理者にポケットベルまたは電話で通知させてもよいでしょう。

表1-8では暗黙的に示していますが、副(呼び出し待機)にリストしてある人々は、正規(主)のシフトで8時間すでに働いた人々です。そのため、彼らの個人的な時間をなるべく邪魔しないような順番で呼び出すようにすることが非常に重要です。たとえば、Susan Mは、午前7時から午後3時まで働いた後、午後8時(子供といっしょにいる時刻)にポケットベルで再度呼び出されると、あまりうれしくないかもしれません。そのような時間帯には、誰か別の人間を呼び出すものとし、Susanは非常に緊急な場合にだけ呼び出すようにしたほうがよいでしょう。スタッフを維持するには、このような配慮が効果的です(24×7環境では、スタッフが短期間で消耗する傾向があるため、スタッフの維持が非常に重要な課題になっています)。

また、複数名の主要なスタッフに休暇を同時に取らせてはいけません。これは、非常に重要なルールです。オペレーションマネージャおよびCIOは、全員このことを知っていますね?　このルールは、常に守る必要があります。次のような会話は、何度も聞いたことがあります。

> オペレーションマネージャ(OM)「Bob､問題があるんだ。どうもデータベースがクラッシュしたようだ」
> CIO 「そうか。Jeffに見させているんだろう?」
> OM 「いや、Jeffは今、休暇中なんだ」
> CIO 「えっ、じゃあLisaは?」
> OM 「実は、Lisaも今いないんだ。彼女は20日に帰って来る予定だが……」
> CIO 「そうか。何かいい手はないのか。Jeffに何とかして連絡はつかないか?」
> OM 「Jeffは、今どこにいるのかつかめないんだ。30分前にポケットベルで知らせてみたんだが……。システムは今、Joeという若い奴と私で見ている。もう少し詳しいことがわかったら、すぐに知らせるよ」

この話の教訓――自分の部下のことをよく知っている各分野のキーパーソン（1名以上）には、「常に」連絡がつくようにしておくこと。

危機発生時に通知できる要員のエスカレーションリストを保守する

「エスカレーションリスト」には、災害の発生時に通知すべき人々を、災害の重大性に合わせて特定の順番で記述します。エスカレーションリストの例を、**表1-9**に示します。

このようなエスカレーションリストを作成しておくと、ダウンタイムの発生時にうまく対処することができます。たとえば、データベースがダウンした場合や2時間以上ダウンすると思われる場合は、販売部門のVPに知らせたほうがよいでしょう。そうすれば、そのVPは主要な顧客に連絡し、状況を伝えることができます。上級幹部は常に、主要な顧客に連絡し、ダウンしているサイトの状況を伝えたほうがよいでしょう。何度ログオンしようとしてもつながらないという状況から、よくない事態を推測させるようなことは、しないに越したことはありません。

また、「データベースがダウンしていて、自分たちがその対処に当たっていることをCEOが知っている」ということに技術スタッフが気付くと、事態が急速に解決する場合もあります（ただし、スタッフの肩越しにCEOが後ろから監視していない場合）。この場合も、バランスが重要です。エスカレーションがあまりにすばやく実施されると、DBAにプレッシャーがかかりすぎ、DBAの作業がうまく進まなくなります。

カスタマ／エンドユーザーにはダウンタイムに先だって常に連絡する

経験からいうと、エンドユーザーや顧客は、ダウンタイム自体に腹を立てるのではなく、その突発性（通常のビジネス機能が中断される）に腹を立てるのです。余裕をもって事前に通知すれば、エンドユーザーや顧客は多くの場合、ダウンタイムに対処するための計画を立て、影響を最小限に抑えることができるのです。

ダウンタイム（時間）	通知先	通知先の肩書き
直後	「コールリスト」を参照	該当する技術管理者（1名または複数名）[DBA、システム管理者、ネットワーク管理者]
0.5	Jef A	オペレーションマネージャ
0.5	Christine L	顧客サービスマネージャ
1.0	Sheila P	CIO
1.0	Gregs S	VP－顧客サービス
2.0+	Timothy S	VP－販売
2.0+	Leslie G	CEO

表1-9　エスカレーションリストの例

オペレーションスタッフが無能であったり、エンドユーザーに対して不注意であったりすると、突発的なダウンタイムが発生します。特に、電子商取引の場合は、顧客に接することが非常に困難なため親密な関係を維持することがおろそかになり、事前に通知することがきわめて難しくなるのです。しかし、それだからこそ、事前に通知することが余計に重要なのです。

このような状況でできる最善の方策は、Webサイトのホームページにメッセージを掲載し、ダウンタイムを伝えることです。また、可能な場合は、主要な顧客のグループに電子メール(スパムメールでないこと!)を送信し、ダウンタイムに関する情報を伝えることもできます。ダウンタイムの重大性によりますが、少なくとも24〜48時間前に通知するのが理想的です。

まとめ

この章では、高可用性の必要性に焦点を当て、データベースのアップタイムに関する要件を明確にしました。また、24×7に関する基本的な概念について、技術的な観点とそうでない観点から説明しました。さらに、データベースで使用する主要なコンポーネントも紹介しました。最後に、エンドユーザーとオペレーション要員が使用できる重要な武器、つまりService Level Agreementについても説明しました。

次の章では、よく発生するさまざまな緊急事態について説明し、その対処方法を見てみます。