データベース選びのポイント

# **免責事項**

*本書に掲載されている情報は作成された時点での情報であり、その内容は予告なく変更される可能性があります。本書に記載された情報は、本書各項目に関する発行日現在の Microsoft の見解を表明するものです。Microsoftは絶えず変化する市場に対応しなければならないため、ここに記載した情報に対していかなる責務を負うものではなく、情報の信憑性については保証できません。*

*本書は情報提供のみを目的としています。 Microsoft は、明示的または暗示的を問わず、本書にいかなる保証も与えるものではありません。*

*すべての当該著作権法を遵守することはユーザーの責務です。Microsoftの書面による明示的な許可なく、本書の一部または全部について、転載や検索システムへの格納または挿入を行うことは、どのような形式または手段（電子的、機械的、複写、レコーディング、その他）、および目的を問わず、禁じられています。これらは著作権で保護された権利を制限するものではありません。*

*Microsoft は、本書の内容を保護する特許 (申請中のものも含む)、商標、著作権、またはその他の知的所有権を保有している場合があります。Microsoft から書面による明示的な使用許諾契約書が供給される場合を除き、本書の提供はこれらの特許、商標、著作権、またはその他の知的財産へのライセンスを与えるものではありません。*

*特に断りのない限り、本書に例示した会社、組織、製品、ドメイン名、電子メール アドレス、ロゴ、人物、場所、およびイベントは架空のもので、実在の会社、組織、製品、ドメイン名、電子メール アドレス、ロゴ、人物、場所、あるいはイベントとは一切無関係です。*

Copyright © 2011 Microsoft Corporation. All rights reserved.

Microsoft、SQL Server*、およびWindows は米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。*

*その他、記載されている会社名および製品名は、各社の商標または登録商標です。*

目次

[第一章 : 開発環境としてのデータベース 1](#_Toc285639023)

[開発環境としての SQL Server 2000 の優位性 1](#_Toc285639024)

[マルチクライアント環境におけるアプリケーション開発デモ 3](#_Toc285639025)

[まとめ　～データベースはアプリケーション プラットフォームに～ 12](#_Toc285639026)

[第二章 : 止まらないデータベース選び 14](#_Toc285639027)

[フェールオーバー クラスタとその必要性 14](#_Toc285639028)

[SQL Server 2000 によるフェールオーバー クラスタの構築デモ 16](#_Toc285639029)

[まとめ　～データベースに 1 ランク上の信頼性を～ 21](#_Toc285639030)

[第三章 : データベースの管理 22](#_Toc285639031)

[そもそもデータベース管理とは 22](#_Toc285639032)

[Microsoft SQL Server 2000 の豊富な管理機能 24](#_Toc285639033)

[Microsoft Operations Manager による管理ツール 30](#_Toc285639034)

[まとめ ～データベース管理とはシステム全体を管理すること～ 32](#_Toc285639035)

[第四章 : データベース セキュリティ 33](#_Toc285639036)

[データベース セキュリティとは 33](#_Toc285639037)

[SQL Server 2000 によるセキュリティ設定の流れ 37](#_Toc285639038)

[まとめ　～セキュリティ設定は簡単～ 41](#_Toc285639039)

# 第一章 : 開発環境としてのデータベース

データベースを選択するとき、何を指標に選べばよいでしょうか。データベースの評価指標の主役は長い間、安定性やパフォーマンスでした。しかし、現代のデータベースにとって安定性やパフォーマンスが良いのはもはや当たり前となっています。では、安定性やパフォーマンス以外にデータベース選びのポイントとなる指標としてどのような要素が考えられるのでしょうか。ここでは、「最新　データベース選びのポイント」と題し、数章にわたってデータベース選びの新しい指標について紹介します。

第一章は「開発環境としてのデータベース」という切り口で、データベース評価のポイントを紹介します。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 開発環境としてのデータベース 開発環境としての SQL Server 2000 の優位性 *  *安定性、ハイパフォーマンス以外の評価ポイントは ？  データベースを利用しないビジネス アプリケーションはほとんどありません。これは、ビジネスの行為においてデータを保存したり、閲覧したりする必要のない状況が存在しないことを考えると当たり前のことと言えます。それゆえに、データベースの安定性やパフォーマンスは、ビジネスにコンピュータが利用され始めて以来、常に重要視されてきた要素であり、今もそれは変わりません。しかし、データベースの安定性やパフォーマンスといった要素は、今や良くて当たり前のものとなっており、データベースを評価する指標としてはある意味一般的すぎる要素となっています。では、安定性やパフォーマンス以外にデータベースを評価する指標としてどのような要素が考えられるでしょうか。  *開発環境としてのデータベースの重要性  既に触れた通り、ビジネス アプリケーションにおいて、データベースを利用しないものはほとんどありません。このことは、アプリケーションの開発には必ずデータベースの利用が伴うということを意味しています。つまり、ビジネス アプリケーションの開発において、データベースは言語などと同様に開発生産性に大きく影響を及ぼす重要な要素と考えることができます。また、開発生産性はそのままアプリケーションの開発や運用コストに跳ね返るため、TCO という観点からも開発環境としてのデータベースの特性は考慮しなければならない重要な要素と言えます。  *開発ツールとの親和性とクライアントとの接続性に注目  それでは開発環境としてデータベースを評価する場合、どのような評価指標が考えられるでしょうか。何より重要なのは、開発ツールとの親和性です。開発ツールとデータベースは独立したアプリケーションであるため、親和性が低いと、2 つのアプリケーションの操作環境を行き来しながら、開発作業を進めなくてはならず、これは非常に非効率です。開発環境としてのデータベースは開発ツールと高度に統合されることが理想的です。  では、開発環境との親和性以外にはどのような評価指標が考えられるでしょうか。いくつかの指標が考えられますが、ここでは、Windows アプリケーション、Web アプリケーション、モバイル アプリケーション、Web サービス アプリケーション開発など、様々なクライアント環境からのデータベース連携という観点でデータベースを評価してみたいと思います。なぜなら、データベースの機能はどのような環境下においても、データを安全に蓄積し、必要に応じて高速に検索するという共通のものであるため、データベースの付加価値はデータベースそのものの機能ではなく、むしろ接続クライアントとの親和性の高さにあるとも考えることができるからです。  *開発環境としての SQL Server 2000 の優位性  データベースが登場して以来、長い間、データベースへの接続モデルはクライアント / サーバー型であり、クライアント環境としては、当初はコンソール アプリケーション、近年では Windows アプリケーションというスタイルが確立しています。この時代は比較的長く続いたこともあり (今でも重要な開発スタイルであることに変わりはありません)、どのようなデータベースにおいても開発生産性は比較的高いものとなっています。しかし、近年、データベースへの接続モデルは Web 経由、モバイル経由、Web サービス (SOA) など、急速に多様化が進み、クライアント環境に応じてデータベースを使い分けなければならないような事態も発生してきています。当然のことながらデータベースが変われば開発環境や場合によっては使用する言語なども変更する必要があり、開発生産性が著しく低下する場合があります。しかし、SQL Server 2000 はあらゆるクライアントに対してシンプルかつ柔軟な接続性を提供可能であることはもちろん、Visual Studio .NET 2003 と高度に統合されており、驚くほど簡単にマルチクライアント環境におけるアプリケーション開発を行うことができます。  ここでは、実際のビジネスにおいてよくあるニーズを想定したデモシナリオを作成し、ニーズに応えて行くかたちで、SQL Server 2000 と Visual Studio .NET 2003 を利用した、Windows アプリケーション、Web アプリケーション、Web サービ スアプリケーション、モバイル (PDA) アプリケーションの 4 つのアプリケーション開発手法を紹介します。  SQL Server を使用したアプリケーション マルチクライアント環境におけるアプリケーション開発デモ *  ここでは、仮想会社である MS 商事でのニーズに対応していくかたちで、SQL Server 2000 が持つ開発環境としての優位性を紹介します。  MS 商事は全国に支店をもつ、中堅卸売会社です。MS 商事では、長年使用してきたオフコンベースの在庫管理システムのオープン系へのリニューアルを検討してきましたが、様々な観点からの検討の結果SQL Server 2000 をベースに在庫管理システムにリプレースすることにしました。SQL Server 2000 へのリプレースを決定した理由は下記のようなものです。  デモシナリオ   |  |  | | --- | --- | | 1. | 本社の要望  本社では、在庫の閲覧はもちろん、在庫管理 (入力) を行うため、オペレータの操作性を考慮し、タブオーダーや入力規則が細かく設定できるWindowsアプリケーションが望ましい。 | | 2. | 支店の要望  支店では在庫の閲覧が中心なのでクライアント環境に特にこだわりは無いが、システム管理や教育のための人材を十分に確保できないため、ソフトのインストールや教育が必要の無いWebアプリケーションをクライアント環境とするのが望ましい。 | | 3. | 取引先の要望  在庫情報をもらえれば、品切れ時期を見計らって無駄の無い納品が可能であるため、在庫情報を連携したい。ただし、WebEDIなどは便利だが、人が閲覧し、在庫を判断しなければならないため、連携の柔軟性が高いWebサービスが望ましい。 | | 4. | フィールド営業の要望  外出先で在庫を閲覧したい。PDAは既に全員が持っているので、PDAを活用したい。Webアプリケーションでもかまわないが、オフラインでも閲覧できることが望ましい。 |   MS商事の場合、在庫情報自体をオフコンのデータベースからオープン系のデータベースに移行することは簡単であることが分かっていましたが、そのデータベースに対する、様々なクライアント環境からの接続性をいかに確保するか、さらにそのようなアプリケーションをいかに効率よく低コストで開発するかということが課題となっていました。こうした条件を考慮した結果、最適なデータベースとしてSQL Server 2000 を選択しました。  以下、具体的にこれらのアプリケーションを開発する手順を紹介します。  *開発の準備　SQL Server 2000 と Visual Studio .NET 2003 の親和性  では、早速開発を行いたいと思いますが、まず、開発に利用するデータベースおよびテーブルを準備します。ここでは、オフコン環境から既に移行済みのデータベース [MSCompany] 、および商品在庫テーブル [GoodsMaster] が SQL Server 2000 上に存在しているものとして話を進めます。通常、データベースを操作するツールと開発ツールは独立したアプリケーションですが、SQL Server 2000 と Visual Studio .NET 2003 は高度に統合されており、SQL Server 2000 に対するほとんど全ての操作を開発ツールである Visual Studio .NET 2003 から行うことができます。  Visual Studio .NET 2003 から SQL Server 2000 を操作し、必要なデータベースやテーブルを設定します。  [Visual Studio から SQL Server のデータベース、テーブル等の変更が可能](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/vs0.gif)  今回の開発に使用するデータベース テーブルの説明をします。今回使用するデータベースは１つで、テーブルも在庫データテーブルのみです。  データベース名 : MSCompany テーブル名 : GoodsMaster 下表がテーブル [GoodsMaster] のカラム説明です。   | **列名** | **データ型** | **説明** | | --- | --- | --- | | gs\_id | int | 商品 ID | | gs\_code | varchar | 商品コード | | gs\_name | varchar | 商品名 | | gs\_price | int | 商品価格 | | gs\_cost | int | 商品コスト | | gs\_stock | int | 商品在庫数 | | gs\_supplyer | varchar | 仕入れ業者 | | gs\_supplyer\_tel | varchar | 仕入れ業者連絡先 | | gs\_date | datetime | 情報更新日 |   商品名が [gs\_name]、在庫数が [gs\_stock] となります。以下ではこのデータの一覧が出力される各アプリケーションを構築します。   |  |  | | --- | --- | | ※詳細は「アプリケーション開発手順書」をご参照下さい。 |  |   *Windows アプリケーションの開発  本社では、在庫閲覧アプリケーションを Windows アプリケーションとして開発します。SQL Server 2000 と Visual Studio .NET 2003 を利用すれば、閲覧のみのアプリケーションであれば、ほんの数分で開発することが可能です。  開発手順は、以下のようになります   |  |  | | --- | --- | | 1. | Visual Studio .NET 2003 で、Windows アプリケーション プロジェクトを作成。 | | 2. | TextBox コントロールと DataGrid コントロールを配置します。 | | 3. | サーバー エクスプローラから閲覧するデータベースのテーブルをデザイナ画面にドラッグ アンド ドロップ。 | | 4. | DataGrid コントロールにデータを表示するためのコードを 2，3 行追加します。 | | 5. | ビルドして完了です。  ※詳細は「アプリケーション開発手順書」をご参照下さい。 |   **開発画面**  [Windows アプリケーション開発画面](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/vs1.gif)  サーバー エクスプローラから取得したテーブルをドラッグ アンド ドロップするだけで簡単にデータが取得できるように設定されます。  **実行画面**  [Windoes アプリケーション実行画面](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/vs2.gif)  アプリケーション実行画面。簡単なステップを実行するだけで、在庫閲覧データが表示可能となります。  *Web アプリケーションの開発  本社以外の支店からもリアルタイムに在庫が閲覧したいとの要望に応えるため、支店用の在庫閲覧アプリケーションの開発を行うことにしました。アプリケーションのメンテナンス性などを考慮し、Web アプリケーションとして開発することにしました。Windows アプリケーションと同様、SQL Server 2000 と Visual Studio .NET 2003 を組み合わせることで、ほんの数分で開発を行うことが可能です。  また、携帯電話などのモバイル デバイスからの Web の閲覧に関しても ASP.NET モバイル アプリケーション Web アプリケーション を利用することで簡単にモバイル用の Web ページを構築することができます。  開発手順は、以下のようになります   |  |  | | --- | --- | | 1. | Visual Studio .NET 2003 で、ASP.NET Web アプリケーション プロジェクトを作成。 | | 2. | TextBox コントロールと DataGrid コントロールを配置します。 | | 3. | サーバー エクスプローラから閲覧するデータベースのテーブルをデザイナ画面にドラッグ アンド ドロップ。 | | 4. | DataGrid コントロールにデータを表示するためのコードを 2，3 行追加します。 | | 5. | ビルドして完了です。  ※詳細は「アプリケーション開発手順書」をご参照下さい。 |   **開発画面**  [Web アプリケーション開発画面](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/vs3.gif)  Windowsアプリケーションと同様、サーバー エクスプローラからドラッグ アンド ドロップでデータ設定。 ほとんど変わらない操作で構築が可能となります。  **実行画面**  [Web アプリケーション実行画面](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/vs4.gif)  Web アプリケーション実行画面。ビルドすると在庫データがWeb ブラウザに自動的に表示されます。  *Web サービスアプリケーションの開発  取引先から、在庫情報をリアルタイムで参照したいという要望がありました。Web アプリケーションの公開を検討しましたが、取引先の要望は人が確認するのではなく、システム同士を連携させたいという要望です。システム同士を連携させるためには、データベース サーバー接続用にファイヤウォールの再設定が必要です。これは、脆弱性を生む要因になります。また、データベースのプラットフォームが異なる場合、連携自体が不可能であったり、可能な場合にも連携先毎に接続用アプリケーションの開発を行わなければならず、非常に手間がかかります。そこで、在庫参照機能を Web サービスとして開発することにしました。これにより、ファイヤウォールの設定を変更することなく、SSL を利用した安全な通信や、プラットフォームに依存しないデータ連携が可能となります。  また、戻り値としてデータセットを戻すこともできます。  開発手順は、以下のようになります   |  |  | | --- | --- | | 1. | Visual Studio .NET 2003 で、ASP.NET Web サービス プロジェクトを作成。 | | 2. | 在庫データを取得して、結果を戻すメソッドを数行記述します。 | | 3. | ビルドして完了です。  ※詳細は「アプリケーション開発手順書」をご参照下さい。 |   **開発画面**  [Web サービス アプリケーション開発画面](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/vs5.gif)  コードを数行付け加えるだけで、簡単にデータベースに格納されているデータをWebサービスとして提供できます。 DataSetを戻り値とすることもできます。  **実行画面**  [Web サービス アプリケーション実行画面](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/vs6.gif)  実行すると Services1 画面が表示され、作成した Web メソッド一覧が表示されます。それぞれのメソッド名をクリックすると起動画面に移動し、[起動] ボタンをクリックするとメソッドを実行した時の戻り値が XML データとして表示されます。また、WSDL ファイルは実行時に自動生成されます。  *モバイル (PDA) アプリケーションの開発  開発が落ち着いたのもつかの間。今度は、フィールドの営業マンからリアルタイムに在庫情報を参照したいという要望がありました。そこで、PDA 用に参照アプリケーションを開発することにしました。PDA とのデータ連携には既に開発済みの Web サービスを利用します。また、オフライン環境での閲覧も考慮し、Web サービスで取得したデータを SQL Server CE に保存することにします。  開発手順は、以下のようになります   |  |  | | --- | --- | | 1. | Visual Studio .NET 2003 で、スマート デバイス アプリケーション プロジェクトを作成。 | | 2. | 使用するコンポーネントをインポートします。 | | 3. | Web サービスを参照します。 | | 4. | ボタンと DataGrid コントロールを配置します。 | | 5. | データベースを作成するコード、Web サービスで取得したデータを格納するコード、取得したデータを表示するコードをそれぞれ追加します。 | | 6. | ビルドして完了です。  ※詳細は「アプリケーション開発手順書」をご参照下さい。   |  | | --- | |  | |   **開発画面**  [モバイル (PDA) アプリケーション開発画面](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/vs7.gif)  モバイルデバイスでもほとんど変わらないインターフェイスで開発が可能となります。  **実行画面**  [モバイル (PDA) アプリケーション実行画面](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/vs8.gif)  モバイル デバイスがなくても、エミュレータを使用してモバイル デバイスと全く同じ環境でアプリケーションの実行が可能となります。 まとめ　～データベースはアプリケーション プラットフォームに～ *  SQL Server 2000 が優れた開発環境であると同時に、様々なクライアント環境のバックエンジンとして機能するかがご理解いただけたかと思います。ここでは Windows アプリケーション、Web アプリケーション、Web サービス アプリケーション、モバイル アプリケーションという 4 つのクライアント環境を紹介しましたが、今後クライアント環境はさらに多様化することが予想されます。もし、クライアント環境に応じてバックエンジンとなるデータベースを変更しなければならない、あるいは開発ツールを変更しなければならないとすればどうでしょうか。開発生産性の低下はもちろん、導入や運用コストの増大は避けられません。そして何よりそのようなシステムは急速に変化するビジネスに対応する柔軟性を失い、直ぐに陳腐化してしまうでしょう。  システムに柔軟性を持たせる手法としては、近年、Web サービスをキーアーキテクチャとした SOA も注目されていますが、Web サービスの開発のセクションで示した通り SQL Server 2000 は SOA (サービス指向アーキテクチャ) のバックエンド エンジンとしても最適なデータベースとなっています。  データベースはもはやデータを蓄積する入れ物ではなく、アプリケーションの開発や運用の基盤として、すなわちアプリケーション プラットフォームとして評価しなければならない時代となっています。  SQL Server 2000 は開発ツールとの群を抜いた親和性の高さをはじめ、モバイル環境から大規模基幹システム、さらには最新の SOA に対応する優れた機能と操作性を提供する、まさにアプリケーション プラットフォームと呼ぶにふさわしい可能性に満ちたデータベースと言えます。  データベースはアプリケーション プラットフォームに  データベースは情報の入れ物から、企業情報戦略の基盤を担うアプリケーション プラットフォームに変化しています。     |  | | --- | |  | |

# 第二章 : 止まらないデータベース選び

第一章でも触れたように、現代のデータベースにとって安定性やパフォーマンスが良いのはもはや当たり前となっています。しかし、どんなにデータベースの安定性やパフォーマンスが向上しても、残念ながら問題が皆無になるわけではありません。特に安定性の問題は、データベースがソフトウエア単体でなく、ハードウェアを伴うシステムとして稼動している以上、パーツの故障やインフラのトラブルによるダウンは避けられない問題です。こうした問題を解決するため、ミッションクリティカルな運用環境においては、サービスの停止を最小限にとどめるために、同じサービスを提供するサーバーを複数台用意し、必要に応じてサーバーを切り替えるという、フェールオーバー クラスタ構築という手法が利用されることが一般的です。

第二章では、本当にとまらないデータベース選びと題して、データベースの可用性向上のために最も重要かつ高度なフェールオーバー機能という切り口でデータベース評価のポイントを紹介します。

*

|  |
| --- |
|  |



## フェールオーバー クラスタとその必要性

*

***フェールオーバークラスタとは?**

フェールオーバーとは、あるサービスを提供するサーバーに何らかのトラブルがあった場合に、自動的に補助サーバーにリクエスト先を切り替えることで、サービスを継続的に提供する機能を意味します。フェールオーバーを実現するためには、通常複数台のサーバー (クラスタ) が必要になりますが、フェールオーバーを目的として導入されたサーバー群をフェールオーバー クラスタと呼びます。フェールオーバーはあらゆるサービスや機能を提供するサーバーにおいて有益な機能ですが、情報システムの基盤を担うデータベースとっては、最も重要な機能の 1 つと言えます。どんなにソフトウエアとしてのデータベースの安定性が向上しても、ハードウェアやネットワークインフラのトラブルなど、物理的なトラブルは避けることができません。このようなトラブルに対応するためには、サーバーを多重化 (クラスタを構成) する必要があります。

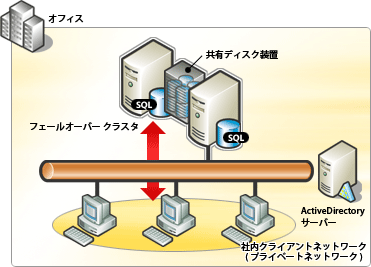
***フェールオーバークラスタの必要性**

どのような状況においてもフェールオーバー クラスタの導入が必須というわけではありません。フェールオーバー クラスタの実現にはスタンドアロン環境を構築する場合の少なくとも 2 倍のコストがかかることは明らかです。提供するサービスに要求される品質とコストのバランスを考慮した上で、慎重に導入を検討することが重要です。 24 時間、365 日のサービス提供が必要であれば、フェールオーバーは間違いなく不可欠な機能です。また、特定時間のみのサービス提供の場合でも、ある期間、ノンストップでサービスを提供しなければならない場合などには不可欠な機能と言えます。

また、フェールオーバーはトラブル対応時のみに有用というわけではありません。ソフトウエアのアップデートや設定更新に伴う再起動など、メンテナンスであっても、サービスを止められないという状況においては不可欠な機能と言えます。さらに、フェールオーバーはバックアップ・リカバリの 1 つの手法としても有用な場合があります。

***フェールオーバー機能の評価のポイント**

フェールオーバー機能という切り口でデータベースを評価する場合、その機能の有無でデータベースを評価することはあまり意味がありません。なぜなら、現在販売されている商用データベースのほとんどはフェールオーバー機能を搭載しているからです。このような状況においては、データベースをフェールオーバー機能の有無ではなく、機能の完成度の高さ、特に導入の容易さという点で評価するべきでしょう。フェールオーバーのような重要かつ複雑になりがちな機能においては、何より導入の容易さが重要です。どんなに優れた機能でも、使いこなすことができなければ全く意味がありません。



## SQL Server 2000 によるフェールオーバー クラスタの構築デモ

*

***GUI で簡単クラスタリング**

SQL Server 2000 Enterprise Edition を利用すれば、 GUI により、簡単にフェールオーバー クラスタを導入、運用することが可能です。 Windows Server および SQL Server に関する基本的な知識があれば、誰でも簡単にフェールオーバー クラスタを導入、運用することができるのです。

***フェールオーバークラスタの構築**

フェールオーバー クラスタの構築には、Windows Server 2003 から [クラスタ アドミニストレータ] を利用して設定しています。[クラスタ アドミニストレータ] で、クラスタの作成とクラスタで使用するノードの追加を行った後に、SQL Server のインストールを行います。

クラスタを構成するには、Active Directory が必ず必要となるため、Active Directory サーバーがない場合は構築する必要があります。 また、マシンを何台も用意することが大変であるため、Virtual Server を利用して設定しています。

※詳細は「クラスタ構築手順書」をご参照下さい。

|  |
| --- |
|  |

主な手順は以下のようになります。

|  |  |
| --- | --- |
| 1. | クラスタの構成に使用するマシンを Virtual Server のバーチャル マシンとして 3 台作成する。 1 台は Active Directory 用 (DC) 、 2 台はクラスタ サーバー用 (MSCS1, MSCS2) とする。  [Virtual Server でのバーチャルマシンの設定](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/fc1.gif)  Virtual Server でのバーチャル マシンの設定 |
| 2. | 作成したバーチャル サーバーのうち 1 台に Active Directory をインストールしドメイン コントローラとして設定する。  [Active Directory のインストール ウィザード](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/fc2.gif)  Active Directory のインストール ウィザード |
| 3. | 設定したドメイン コントローラのドメインに残りの 2 台を参加させる。  [サーバーをドメインへ参加させる](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/fc3.gif)  サーバーをドメインへ参加させる |

|  |  |
| --- | --- |
| 4. | サーバー (MSCS1) にクラスタ アドミニストレータを利用して、クラスタを作成する。  [クラスタ アドミニストレータによるクラスタの作成](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/fc4.gif)  クラスタ アドミニストレータによるクラスタの作成 |
| 5. | 作成したクラスタにもう 1 つのサーバー (MSCS2) をクラスタ ノードとして追加する。  [クラスタ アドミニストレータによるクラスタ ノードの追加](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/fc5.gif)  クラスタ アドミニストレータによるクラスタ ノードの追加 |
| 6. | サーバー (MSCS1) に SQL Server をインストールする。インストーラに従い、インストール先を「仮想サーバー」に設定すると、クラスタの設定としてインストールされる。  [SQL Server のインストール](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/fc6.gif)  SQL Server のインストール  [SQL Server インストールでのクラスタサーバーの設定](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/fc7.gif)  SQL Server インストールでのクラスタ サーバーの設定 |
| 7. | フェールオーバー クラスタの動作確認。Web アプリケーションを実行し、フェールオーバー時のアプリケーションの動作とデータを確認  [フェールオーバーの実行確認 (左 MSCS1 、右が MSCS2 )](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/fc8.gif)  フェールオーバーの実行確認 (左 MSCS1 、右が MSCS2 ) |

※詳細は「クラスタ構築手順書」をご参照下さい。

## まとめ　～データベースに 1 ランク上の信頼性を～

*

第一章でも触れたとおり、データベースにアクセスしないビジネス アプリケーションは存在しないと言っても過言ではありません。これは同時に、データベースがダウンすれば全てのビジネス アプリケーションが機能停止することを意味しています。データベースがハードウェアを伴うシステムとして動作する以上、突発的な事故は避けられません。サーバー自体の故障によるサービス停止を防止する方法は基本的にはフェールオーバー クラスタしかありません。ハードウェアが非常に高価だった時代は、システムの多重化は金融機関など、一部の特殊なシステムだけに許された冗長化手法でした。しかし、ハードウェアの低価格化が進んだ現在においては、フェールオーバー クラスタによる信頼性向上は非常にシンプルかつ有益な手法と言えます。

SQL Server 2000 はフェールオーバー クラスタという高度な機能も GUI で簡単に設定することができるため、エンタープライズレベルはもちろん、部門、クループレベルの情報管理においても 1 ランク上の信頼性を実現することが可能であり、規模の大小を問わず、常時運用が求められる現代のビジネスシーンを支えるデータベースとして、最もふさわしい製品といえます。

# 第三章 : データベースの管理

本章は管理という点でデータベースの選定ポイントをご紹介します。データベース評価のポイントは数多く存在しますが、導入後の視点という意味では、最も重要な評価ポイントと言えるかもしれません。このシリーズは「最新のデータベース選びのポイント」ということなので、データベースの内部管理と外部管理という新しい視点を紹介したいと思います。

|  |
| --- |
|  |



## そもそもデータベース管理とは

*

データベースの管理と聞いて、どのようなことを想像するでしょうか。

一口にデータベースの管理と言っても、その内容はバックアップやリカバリといった基本的なものから、パフォーマンス チェックやクラスタの監視といった専門性の高いものまで広範囲に及びます。そもそもデータベース管理とは、期待している通りにデータベースが動作しているかをチェックすることを基本とし、万が一異常が発見された場合には速やかに正常な状態に復旧するといった作業であり、データベースの管理機能はそれらのオペレーションを支援する機能と言えます。これらの機能は地味ではありますが、非常に重要な機能です。

しかし、重要であるがゆえに、現在市販されているほとんどのデータベースにおいては、少なくともカタログ上の管理機能はどれも同じように充実しているように見えます。では、どのような基準でデータベースの管理機能を評価すればよいのでしょうか。

***最も重要なポイント管理機能の使い勝手**

データベースの管理機能を評価する場合のポイントはいくつかありますが、最も重要なポイントは、管理機能の使い勝手です。管理機能が充実していることはもちろんですが、それ以上に管理機能の使い勝手は重要な評価ポイントとなります。なぜなら、管理機能の使い勝手が悪いと、管理自体がおろそかになる可能性があり、データベースの信頼性自体に深刻な影響を与える場合があるからです。管理の現場でデータベースの基本的な管理を怠ったことに起因するトラブルが、しばしば発生することがあります。せっかく機能も正しく設定されなければ宝の持ち腐れです。

では、使い勝手の良さとは、具体的にどのようなことでしょうか。管理という観点で使い勝手の良さを評価する場合、次の 3 つについて考慮するとよいでしょう。

|  |  |
| --- | --- |
| • | 必要な機能が簡単に操作できるか (GUI が提供されているか) |
| • | 操作を自動化するための機能があるか。 |
| • | 期待通りの性能が得られているかをチェックするための機能があるか。 |

SQL Server 2000 は非常にシンプルかつ強力な使い勝手の良い管理機能を提供します。管理者は日常的に利用するデータベース管理操作のほとんどを SQL Server 2000 の管理クライアントである Enterprise Manager が提供する GUI 環境から利用することが可能です。また、GUI で提供される機能は、個々の機能の操作にとどまらず、複数の操作を組み合わせたり、実行のスケジューリング設定などといった、データベースの管理を省力化のために必要な機能も提供されます。もちろん、他のアプリケーションと組み合わせたり、詳細なコントロールが必要とするユーザーのためにコマンドラインのインターフェイスも用意されています。

必要な性能が得られているかどうかも、パフォーマンス モニタによりビジュアルにモニタリングすることが可能です。

***実運用で重要となる管理機能の管理**

ここまでは、使い勝手という評価のポイントを紹介し、それを具現化する SQL Server 2000 の優位性について説明してきました。しかし、机上の空論ではなく、実運用という視点でデータベース管理を語る場合、ここまでの評価は重要な概念が欠落したものとなっています。それは、データベースの (管理) 機能自体が正常に動作しているかどうかをいかに管理するかという概念です。つまり、管理の管理です。例えば、バックアップ作業 1 つをとっても、毎日の運用の中で、毎日バックアップ操作に四苦八苦するということはほとんどありません。むしろ問題になるのは、バックアップ タスクが正常に動作したかどうかを管理することです。これは、他のタスクについても同じことが言えます。

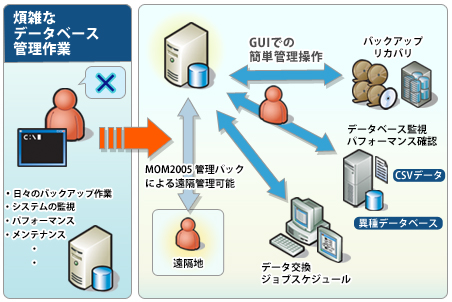
実運用においては、データベース自体が原因となるトラブルはあまり多くはありません。仮にトラブルがあった場合でも、上位アプリケーション側のプログラム ミスに起因する一過性のトラブルであることがほとんどです。慢性的なトラブルの原因としては、ハードディスクやメモリ不足によるタスクのダウンや、ハードウェア トラブルによる事実上のサーバー ダウンなど、データベース以外に根本的な原因があることがほとんどです。

残念ながら、データベース (ソフト) に直接的な原因がないからっと言って、データベース サーバーのダウンの責任から管理者が解放されることはまれです。このような問題を未然に防ぐ、あるいは、問題が発生した場合に迅速に対応できる環境を整えるためには、データベース (の管理機能) が正常に動作しているかをデータベースの外部から管理する必要があります。

つまり、実運用という観点では、データベースの管理には、データベース内部の管理と、データベース外部の管理という 2 つの管理が不可欠ということになります。

マイクロソフトでは、この外部の管理に対応するために Microsoft Operations Manager 2005 を提供しています。Microsof Operations Manager 2005 は SQL Server 2000 にとどまらず、Windows Server System として提供されるサーバー製品の管理機能を提供します。それぞれの製品に特化した管理機能は管理パックとして提供され、SQL Server 2000 に対しても SQL Server 2000 に特化した管理パックが提供されており、それを利用することにより、外部から SQL Server 2000 の動作状況をモニタリングすることが可能になります。

以下では、SQL Server 2000 が提供するデータベース内部の管理機能と Microsoft Operations Manager 2005 が提供するデータベース外部の管理機能について、簡単に紹介します。



## Microsoft SQL Server 2000 の豊富な管理機能

*

では、SQL Server 2000 が提供する管理機能がどれほどシンプルで使い勝手の良いものなのかを紹介します。多くの機能を GUI で操作することができ、複雑な機能においてもウィザード形式で対話的に利用することが可能です。

***データベースバックアップリカバリ機能**

データベースのバックアップはデータベースの管理においても、最も重要かつ使用頻度の高い機能と言えます。では、早速 SQL Server 2000 のバックアップの操作を見てみましょう。

データベースのバックアップには、データベースを停止させて行うオフライン バックアップと稼動状態のままで行うオンライン バックアップの 2 つの手法があります。日常的に多く利用するのはオンライン バックアップです。ここではオンライン バックアップの手法を紹介します。

オンライン時のデータベースのバックアップは SQL Server 2000 の管理ツールである Enterprise Manager を使います。

※詳細は「データベース管理手順書」をご参照下さい。

主な手順は以下のようになります。

|  |  |
| --- | --- |
| 1. | Enterprise Manager を起動させ、バックアップしたいデータベースの全てのタスクメニューからデータベースのバックアップをクリックします。  [データベースバックアップ](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/sqladmin1-1b.gif)  バックアップ タスクの起動 |
| 2. | バックアップ先にデーベースのバックアップ ファイルの保存先を選択し、OK ボタンを押して完了です。保存先に設定した名前のバックアップ ファイルが生成されています。  データベースバックアップ手順1  [SQL Server バックアップ] ダイアログ ボックスでのバックアップ処理  ここでは、手動によるバックアップを紹介しましたが、バックアップ パターンをスケジュールにあわせて自動実行するように設定することもできます。SQL Server 2000 では、高度な知識が無くても簡単にバックアップを行うことができるため、データベース専任のエンジニアがいないシーンにおいても、簡単に運用することが可能です。  ※詳細は「データベース管理手順書」をご参照下さい。 |

***データのインポート/エクスポート、データ変換機能**

企業内でデータベースの運用を行う場合、外部連携無く、単体で動作することは稀であり、必ずと言ってよいほど、外部データソースとの連携が必要になります。

SQL Server 2000 では、異種データベースやファイルからのデータの読み込みや出力をウィザードに従うだけで簡単に行うことができます。

例えばデータ分析のバック エンジンとしてデータベースを利用する場合などは、外部の様々なデータソースから、データをインポートするといった機能が不可欠となります。また、逆に、データをエクスポートする場合もあるでしょう。このような機能は ETL 機能と呼ばれ、現代のデータベースには必須の機能とされています。

SQL Server 2000 では DTS と呼ばれる ETL 機能が搭載されており、他社のデータベースはもちろん、Excel ファイルや CSV ファイルからの入力や出力機能が標準搭載されており、しかも、ビジュアルに操作が可能となっています。

では、DTS の操作を簡単に紹介します。

※詳細は「データベース管理手順書」をご参照下さい。

主な手順は以下のようになります。

|  |  |
| --- | --- |
| 1. | Enterprise Manager を起動させ、インポートを行いたいデータベースの全てのタスク メニューから [データのインポート] をクリックします。[データ変換サービス インポート/エクスポート ウィザード] が立ち上がります。  [DTS インポート/エクスポート ウィザード](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/sqladmin2-1b.gif)  DTS インポート/エクスポート ウィザードの起動 |
| 2. | 後はウィザードに従い、インポートしたいファイルとインポート先のデータテーブルを指定して実行します。これでインポートが完了しました。  [インポート ウィザード成功](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/sqladmin2-2b.gif)  DTS インポート処理の完了  データベースのインポートとエクスポートの設定手順も同様のウィザードを実行します。また、ファイル以外にも異種データベースから取り込んだり、Access のデータから取り込んだりすることも簡単に行うことができます。  ※詳細は「データベース管理手順書」をご参照下さい。 |

***SQL Server の監視**

データベースを実際に使用するとなると、実際にどのくらいの速さでクエリが処理されるか、またはデータベース システムに異常がなく稼働しているかどうかは気になるところです。パフォーマンス モニタを使用すればサービス稼働中の SQL Server 2000 がどのくらいのパフォーマンスで実行されているかを確認することができます。

パフォーマンス モニタ自体は Windows OS の機能ですが、SQL Server 2000 をインストールすることによって、パフォーマンス モニタが SQL Server を確認する専用のパフォーマンス カウンタが追加されます。このパフォーマンス カウンタの様々な値を確認することで、SQL Server の監視を行うことができます。下に簡単な手順を紹介します。

※詳細は「データベース管理手順書」をご参照下さい。

主な手順は以下のようになります。

|  |  |
| --- | --- |
| 1. | 管理ツールのパフォーマンスを起動し、右ペインの新しいカウンタ セットをクリックし、追加ボタンをクリックします。  [パフォーマンス モニタ](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/sqladmin4-1b.gif)  パフォーマンス モニタでの新しいカウンタ セットの実行 |
| 2. | カウンタの追加でパフォーマンス オブジェクトから監視したいオブジェクトとそのカウンタを設定します。  [カウンタの追加](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/sqladmin4-2b.gif)  SQL Server パフォーマンス オブジェクトのカウンタの追加 |
| 3. | 設定が終わると SQL Server のバッファ キャッシュを監視し始め、モニタに反映します。これで監視をいつでも行うことができます。  [追加したパフォーマンス カウンタの確認](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/sqladmin4-3b.gif)  各カウンタのモニタリング  ※詳細は「データベース管理手順書」をご参照下さい。   |  | | --- | |  | |

## Microsoft Operations Manager による管理ツール

*

***Microsoft Operations Manager 2005 SQL Server 2000 管理パックとは**

Microsoft Operations Manager 2005 を利用することにより、SQL Server を遠隔管理したり、サーバー全体を監視したりすることができますが、実際に監視するためには Microsoft Operation Manager 2005 をインストールするだけでは使用できません。個々の製品専用の管理パックをインストールして、各設定を行う必要があります。SQL Server 2000 の監視の場合は、SQL Server 2000 用の管理パックをインストールすることになります。

次にこの管理パックを利用して監視できる内容をご紹介します。

***管理パックでできること**

Microsoft Operations Manager 2005 SQL Server 2000 管理パックで監視できる主な内容は以下のようになっています。管理パックでは、あらかじめマイクロソフトにより設定された理想的な閾値が設定されており、十分な経験がなくても、熟練した管理者と同様の管理を行うことが可能となっています。

|  |  |
| --- | --- |
| • | データベースの接続確認 |
| • | SQL Server サービス、エージェント サービス などの稼働状態の監視 |
| • | データベース、トランザクション ログなどの領域の監視 |
| • | セキュリティの監視 |
| • | 最低限の Service Pack が適用されているかどうかの監視 |
| • | エージェント ジョブが長時間実行されているかの確認 |
| • | バックアップとジョブの監視 |

次に簡単な、Microsoft Operations Manager 2005 での SQL Server 2000 の監視テスト手順を説明します。ここでは、SQL Server エージェント サービスが開始されているかどうかを確認し、サービスが開始されていなければアラートが発生していることを確認します。

※詳細は「データベース管理手順書」をご参照下さい。

主な手順は以下のようになります。

|  |  |
| --- | --- |
| 1. | Microsoft Operations Manger 2005 管理コンソールから SQL Server エージェント サービスの稼働を管理するイベント ルールを有効にします。  [MOM 2005 管理コンソール](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/sqladmin3-1b.gif)  管理コンソールでのイベントルールの設定 |
| 2. | Microsoft Operations Manager Agent がインストールされている監視先サーバーの SQL Server エージェント サービスを停止します。  SQL Server Agent の停止  SQL Server エージェント サービスの停止 |
| 3. | アラートが出ているかどうかを確認するのは、オペレータ コンソールを利用します。オペレータ コンソールのすべてのアラートを確認すると、SQL Server エージェント サービスが停止していることを感知して、アラートを出力していることがわかります。  [MOM 2005 管理コンソールでのエラーメッセージのチェック](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/sqladmin3-3b.gif)  オペレータ コンソールでのアラートの確認  ※詳細は「データベース管理手順書」をご参照下さい。 |

## まとめ ～データベース管理とはシステム全体を管理すること～

*

データベースにとって管理機能は非常に重要な機能です。それゆえに、データベースの管理機能に対する評価は何度となく行われてきました。しかし、それらの評価の多くは、データベースの内部的な管理機能に対してであり、サーバーの管理が複雑化した現代の現場的視点での評価という意味では不十分と言わざるを得ません。なぜなら、データベースのトラブルは、必ずしもデータベースに起因するとは限らないからです。データベースの内部的な状況の管理が重要であることは言うまでもありませんが、完璧な管理を目指すならデータベース サーバー自身の状態を外部から管理するという考えを持つことは非常に重要です。今回は SQL Server 2000 が提供する内部的な管理機能と共に、外部管理という観点から Microsoft Operations Manager 2005 を紹介しました。巨視的な視野を持つことが重要な現代のビジネスシーンにおいて、データベース評価も巨視的な視点で行ってみる価値があるのではないでしょうか。

# 第四章 : データベース セキュリティ

2005 年 4 月より、いわゆる個人情報保護法が施行されるなど、社会全体として、情報セキュリティへの関心が高まっており、あらゆる分野において情報セキュリティへの対応は避けては通れないものになっています。情報のほとんどが電子化され、データベースに蓄積されている現代においては、「情報セキュリティ = データベース セキュリティ」と言っても過言ではないかもしれません。本章では、「データベース セキュリティ」という視点からデータベース選びのポイントについて紹介します。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| データベース セキュリティ データベース セキュリティとは *  そもそもデータベース セキュリティ (対策) とは何でしょうか。  答えは簡単です。データベースに格納されている情報を、不正な人に見せない (利用させない) ようにすることです。逆に、正しい人に、正しく利用してもらうようにデータベースを設定することとも言えるでしょう。つまり、データベース セキュリティという視点でデータベースを評価するときのポイントは、いかに不正な人に情報を見られないようにするための機能が充実しているか、ということになります。もちろん、それらの機能が使い勝手が良く提供されている必要があります。これは、このシリーズで何度も触れてきましたが、どんなに優れた機能も使い勝手が悪いのであれば、結果として使いものにならないからです。  *不正な人に情報を見られる可能性  データベース セキュリティに対して正しく理解するための第一歩は、不正な人に情報を見られてしまう可能性 (原因)、つまり、脆弱性となりうる要素を理解することです。  データベース内の情報が結果として不正な人に見られてしまう可能性としては、技術的なものはもちろん、運用的なものも含め様々ありますが、ここでは、技術的な要素について、その代表的なものを挙げ、それぞれに対してどのような対策が有効かについても紹介します。さらに、それぞれの課題に対して SQL Server や Windows プラットフォームがいかに課題を解決できるかについても紹介します。  [セキュリティ構成](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/sqlsecurity1_b.gif)  データベースを含むシステム全体におけるセキュリティ チェック要素   |  |  | | --- | --- | | 1. | 途中経路での不正閲覧 (INTERNET)  データベースの内容をインターネット経由で利用する場合、途中経路で不正に閲覧されてしまう可能があります。このような問題は、SSL などによって通信内容を暗号化することなどで回避することが可能です。SQL Server 2000 はSQL Server プロトコルの暗号化や SSL による HTTPS などに対応していますので、安全なデータ通信環境を構築することが可能です。 | | 2. | 外部からの不正アクセス  データベース サーバーにグローバル IP アドレスが振られているような場合、不特定多数のユーザーから不正にアクセスされる可能性があります。このような問題は Firewall を設置し、適切な設定を行うことで回避することが可能です。SQL Server 2000 自体には Firewall 機能はありませんが、ISA Server などを組み合わせることにより、決め細やかな Firewall 機能を実現することが可能です。SQL Server では、デフォルトでは 1433 および 1434 ポートを使用し、外部と通信を行っています。Firewall により、これらのポートをブロックすることにより、不正なアクセスを遮断することができます。   |  |  | | --- | --- | |  |  | | | 3. | 途中経路での不正閲覧 (LAN)  暗号化に SSL アクセラレータなどを利用した場合、Firewall や SSL を通過した後の情報は平文となります。このような場合、LAN 内を通過する情報は暗号化されませんので、LAN 内の不正ユーザーに閲覧される可能性があります。このような問題は End - to - End の暗号化を利用することで回避することが可能です。SQL Server 2000 は自身の機能で暗号化通信を行うことができるため、情報経路の端から端までを完全に暗号化することが可能です。 | | 4. | 内部からの不正アクセス  Firewall でインターネットからの不正アクセスを回避することはできますが、LAN 内からの不正アクセスをブロックすることはできません。このような問題は OS 自体が装備している Firewall 機能を利用し、データベース サーバーにアクセスできるネットワーク セグメントや端末数を最小限度に留めることで、リスクを低減することが可能です。また、不正アクセスに限らず、ウイルス感染なども防止する効果があります。  Windows Server 2003 は標準で Firewall 機能を搭載しているため、それらを適切に設定することで情報漏えいやウイルスによるリスクを低減することが可能です。  また、実際に不正アクセスや疑わしいアクセスがあったかどうかの監視は SQL Server 2000 に標準搭載されている SQL プロファイラによりチェックすることができます。SQL プロファイラはその他、不正アクセスだけでなく、SQL Server の起動、停止、実行される SQL まで、様々な監視を行うことが可能です。 | | 5. | 不正ログイン  データベースを利用するためには通常、ID とパスワードによるユーザー認証が必要となります。しかし、パスワードがブランクになっていたり、ID やパスワードが容易に推測可能なものであったりした場合、情報を不正に閲覧される可能性があります。このような問題は、ID やパスワードを複雑にしたり、定期的に変更したりすることで回避することが可能です。しかし、パスワードの管理は考えている以上に複雑なものです。複雑にするのは簡単ですが、ログインの対象となるようなサーバが複数台存在する場合などは、パスワードを覚えきれないため、どうしても簡単なパスワードになりがちです。また、覚え切れないといってメモしておくわけにもいきません。  SQL Server 2000 は Windows と統合可能な認証システムを提供することで、複雑な問題を解決します。ID 、パスワードに関する問題を解決するためには特に Windows Server 2003 が提供する Active Directory による ID 、パスワードの一元管理が有効です。Active Directory を利用すれば、複数台にまたがる ID やパスワードを統合し、一元的に管理することができます。また、Identity Integration Server は異なったプラットフォーム間でのユーザーおよび ID 、パスワードの一元管理機能を提供します。これらの製品を利用すれば、単純な ID、パスワードの統合だけでなく、パスワードの複雑さの設定やロックアウト (数回ログインしてエラーならユーザーをロックする) までの回数なども管理することができます。さらに、Operations Manager を利用すれば、ログインの失敗などのイベントが発生した場合、管理担当者にメールを送信するといった機能も実現可能です。SQL Server 2000 は Windows が提供するこのような認証機能をデータベース認証に統合できるため、柔軟かつ強固な認証を実現可能です。基本的な管理は既に紹介した SQL プロファイラでも十分可能ですが、Windows プラットフォームの他の製品と組み合わせることでより柔軟かつ強固な認証を実現可能です。   |  | | --- | |  | | | 6. | 権限設定ミスによる不正アクセス  正しくログインしたユーザーも残念ながら安心はできません。権限設定にミスがあった場合、本来は閲覧できない情報が閲覧できてしまう可能性があります。このような問題を解決するためには注意深く権限設定を行う必要があります。ミスを起こさないためには、使い勝手の良いビジュアルな管理機能を提供するデータベースを選ぶべきです。SQL Server 2000 は使い勝手の良い GUI ベースの管理ツールを使って、ミス無く各種設定を行うことが可能です。 | | 7. | ファイルへの不正アクセス  データベースは通常、サーバー プロセスとして動作しており、その動きを目でみることはできません。しかし、どのようなソフトウエアでも、その実態は単純なファイルとしてハードディスク内に存在しているため、データベース内でどのような高度な対応を行っても、データベースを構成しているファイル自体をコピーされてしまう可能性があります。このような問題は、通常 OS 自体のユーザー権限を正しく設定することで回避することが可能です。 |   ここでは、配慮すべき代表的な脆弱性の要素を紹介しましたが、これ以外にも、アプリケーションのコーティング ミス (動的に受け取るパラメータのエラー処理ミスなど) により発生する問題や、バックアップ ファイルの持ち出しなどのリスクなど、考慮すべき要素は無数にあります。情報セキュリティ対策にこれで完璧というものはありません。コストなどを考慮しながら、目的に応じて柔軟に対応することが重要です。  *危険性は内と外の両方にある  ここまでは、データベース システムにおいて情報が不正に見られてしまう危険箇所を見てきました。既にお気づきの通り、危険性のいくつかはデータベース単体の機能では対処ができないものもあります。データベース セキュリティ対策に際しては、まず、このことを理解し、データベース内でどのような対応を行うのか、データベースの外でどのような対応を行うのかを明確にし、目的に合った設定を行う必要があります。   |  |  | | --- | --- | | • | **データベース内で対応可能なもの**  1. 途中経路での不正閲覧 (INTERNET)  3. 途中経路での不正閲覧 (LAN)  5. 不正ログイン  6. 権限設定ミスによる不正アクセス | | • | **データベース外で対応しなければいけないもの**  2. 外部からの不正アクセス (ISA Server)  4. 内部からの不正アクセス (ISA Server 、Windows Server 標準 Firewall 機能)  7. ファイルへの不正アクセス (ユーザー権限の適切な設定) |  SQL Server 2000 によるセキュリティ設定の流れ *  では、SQL Server 2000 におけるセキュリティ設定の流れを簡単に見てみましょう。ここでは、データベースの利用者 (ログイン / ユーザー) の設定、権限の設定、パスワード ポリシーの設定、通信の暗号化設定といったセキュリティ対策のための基本的な要素を紹介します。  *利用者 (ログイン / ユーザー) の設定  SQL Server 2000 では、データベースの利用者を効率よく管理するためにログインとユーザーという 2 つの概念を用います。ログイン アカウントはデータベース システム自体に接続するためのアカウントであり、ユーザー アカウントはデータベースごとに設定されたデータへのアクセス権限を持つアカウントです。両者ともにパスしてはじめてデータにアクセスすることができます。どちらの設定も GUI を用いて簡単に行うことが可能です。  [ログイン アカウントの設定](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/sqlsecurity2_b.gif)  ログイン アカウントの設定  [ユーザー アカウントの設定](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/sqlsecurity3_b.gif)  ユーザー アカウントの設定  *権限の設定  SQL Server 2000 には多くの機能があり、それぞれの機能の利用の可否について、きめ細かく設定することが可能です。しかし、それらの設定は GUI でチェックボックスを選択するという操作で簡単に行うことが可能です。洗練された GUI により、設定ミスなどを最低限度に留めることができます。  [ユーザー権限の設定](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/sqlsecurity4_b.gif)  ユーザー権限の設定  [データベース ロールの設定](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/sqlsecurity5_b.gif)  データベース ロールの設定  *パスワード ポリシーの設定 (Windows Server)  SQL Server 2000 で提供される Windows 統合認証を利用することで、利用者の管理を Windows のユーザーアカウント管理と統合することが可能です。利用者の登録、削除はもちろん、パスワードの複雑さや有効期限、ロックアウトまでのトライ数などを設定することが可能です。  [パスワード ポリシーの設定](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/sqlsecurity7_b.gif)  パスワード ポリシーの設定  [Active Directory による SQL Server 委任認証設定](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/sqlsecurity6_b.gif)  Active Directory による SQL Server 委任認証設定  *通信プロトコルの暗号化  通信プロトコルの暗号化というと、複雑なイメージがありますが、SQL Server 2000 ではチェックボックスの選択 1 つで通信内容を暗号化することが可能です。  [SQL Server ネットワーク ユーティリティ](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/sqlsecurity8_b.gif)  SQL Server ネットワーク ユーティリティでの暗号化設定   |  |  | | --- | --- | |  |  | |  |  |   *SQL プロファイラによる SQL 文の監査  SQL プロファイラは GUI で提供される SQL Server 2000 の監視ツールです。ログインの成功、失敗から、実行される SQL まで、SQL Server 2000 を安全に安定して運用するためのヒントとなる様々なステータスを監視することができます。  [SQL プロファイラによる トレースの設定](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/sqlsecurity9-1_b.gif)  SQL プロファイラでログイン失敗イベントをトレース設定  [トレースの実行](http://www.microsoft.com/japan/sqlserver/2005/evaluation/select/images/sqlsecurity9-2_b.gif)  トレースの実行でログイン失敗のログを確認 まとめ　～セキュリティ設定は簡単～ *  データベース セキュリティと聞くと、複雑そうですが、基本は難しくはありません。家のセキュリティの基本は何でしょうか。玄関に鍵をかけることです。  データベースも同じで、まず ID、パスワード管理をしっかり行うことです。しかし、家のセキュリティ同様、すべての窓の鍵、ガス栓の状態、水漏れなど、気にすればきりがありません。だからといって、窓をすべてコンクリートで塞いでしまうわけにもいきません。現実的な最良解は、要所を一元的かつビジュアルに管理でき、問題があった場合には通知してくれるようなシステムがあればいいということになるでしょう。SQL Server 2000 および Windows プラットフォームを利用することで、まさにそのようなシステムを実現することが可能になります。運用的観点から言えば、複雑さも脆弱性を生む要因の 1 つです。SQL Server 2000 および Windows プラットフォームが提供するシンプルかつ強力なセキュリティ機能は複雑化した現在にふさわしいものと言えるでしょう。 |