Sun Java™ System

Directory Server 5.2
配備計画ガイド

2005Q1

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819-2015

目 次

図目次

表目次

はじめに

このマニュアルには、ディレクトリ配備を計画し、データ型、アクセス制御、レプリケーション、およびサイジングといった問題について重要な決定を行うために必要な情報が記載されています。

Sun™ のマニュアルへのアクセス方法、および Sun のマニュアルの使用方法については、次の節を参照してください。

- 表記規則
- 関連マニュアル
- マニュル、サポート、およびトレーニング
- 関連するサードパーティの Web サイトの参照
- ご意見、ご要望の送付先

表記規則

表 1 は、このマニュアルで使用されるフォントの表記規則を示しています。

表 1　フォントの表記規則

フォント	意味	例
`AaBbCc123` (モノスペース)	API および言語の要素、HTML タグ、Web サイトの URL、コマンド名、ファイル名、ディレクトリパス名、画面出力の表示、サンプルコード。	`.login` ファイルを編集します。 `ls -a` を使用してすべてのファイルを表示します。 `% You have mail.`
`AaBbCc123` (太字のモノスペース)	画面出力の表示に対し、ユーザーが入力する文字。	**`% su`** パスワード :

表 1　フォントの表記規則 (続き)

フォント	意味	例
AaBbCc123 (イタリック)	実際の名前や値に置き換えられるコマンド行変数。	ファイルは *ServerRoot* ディレクトリに格納されています。

表 2 は、このマニュアルで使用されるプレースホルダの表記規則を示しています。

表 2　プレースホルダの表記規則

項目	意味	例
install-dir	ソフトウェアバイナリがインストール後に置かれる場所のディレクトリ接頭辞のプレースホルダ。	Solaris システムのデフォルトの *install-dir* 接頭辞は / です。 Red Hat システムのデフォルトの *install-dir* 接頭辞は /opt/sun です。
ServerRoot	サーバーインスタンスとデータが置かれているディレクトリのプレースホルダ。 *ServerRoot* の下の各サーバーは、クライアント側サーバーコンソールからリモートで管理できます。サーバーコンソールは、サーバー側システムで直接実行する必要があるタスクを実行するときは、サーバー側の Administration Server を使用します。	デフォルトの *ServerRoot* ディレクトリは /var/opt/sun/serverroot です。
slapd-*serverID*	*ServerRoot* の下に置かれている特定のサーバーインスタンスと、それに関連するデータがデフォルトで存在するディレクトリのプレースホルダです。	デフォルトの *serverID* はホスト名です。

表 3 は、このマニュアルで使用される記号の表記規則を示しています。

表 3　記号の表記規則

記号	意味	表記法	例
[ ]	省略できるコマンドオプションを含みます。	O[*n*]	-O4, -O

表 3　記号の表記規則 (続き)

記号	意味	表記法	例
{ }	必須コマンドオプションの選択肢を含みます。	d{y|n}	-dy
|	コマンドオプションの選択肢を区切ります。		
+	グラフィカルユーザーインタフェースで使用されるキーボードショートカットで、同時に押すキーを連結します。		Ctrl+A
-	グラフィカルユーザーインタフェースで使用されるキーボードショートカットで、連続して押すキーを連結します。		Esc-S
>	グラフィカルユーザーインタフェースで選択するメニュー項目を示します。		「ファイル」>「新規」 「ファイル」>「新規」>「テンプレート」

表 4 は、このマニュアルで使用されるシェルプロンプトの表記規則を示しています。

表 4　シェルプロンプト

シェル	プロンプト
C シェル	*machine-name*%
C シェルスーパーユーザー	*machine-name*#
Bourne シェルと Korn シェル	$
Bourne シェルおよび Korn シェルのスーパーユーザー	#

Directory Server コマンドの入力および出力は、通常は LDIF (LDAP Data Interchange Format) [RFC 2849] を使って表されます。行の折り返しは、見やすくするためのものです。

関連マニュアル

次のマニュアルは、HTML 形式および PDF 形式で、
http://www.sun.com/documentation/ の Web サイトから入手できます。

Directory Server のマニュアル

『Directory Server リリースノート』

『Directory Server Technical Overview』

『Directory Server 配備計画ガイド』

『Directory Server Installation and Migration Guide』

『Directory Server Performance Tuning Guide』

『Directory Server 管理ガイド』

『Directory Server Administration Reference』

『Directory Server Plug-In Developer's Guide』

『Directory Server Plug-In Developer's Reference』

『Directory Server Man Page Reference』

Administration Server のマニュアル

『Administration Server リリースノート』

『Administration Server Administration Guide』

『Administration Server Man Page Reference』

Directory Proxy Server のマニュアル

『Directory Proxy Server リリースノート』

『Directory Proxy Server 管理ガイド』

関連する Java Enterprise System マニュアル

『Java Enterprise System インストールガイド』

『Java Enterprise System アップグレードと移行』

『Java Enterprise System 用語集』

マニュル、サポート、およびトレーニング

表 5 に、Sun のマニュアル、サポート、およびトレーニング情報へのリンクを示します。

表 5　マニュル、サポート、およびトレーニングへのリンク

フォント	意味	例
マニュアル	http://www.sun.com/documentation/	PDF 形式または HTML 形式のマニュアルをダウロードしたり、印刷されたマニュアルを注文したりできます。
サポートおよびトレーニング	http://www.sun.com/supportraining/	テクニカルサポートを受けたり、パッチをダウンロードしたり、Sun のトレーニングコースについての情報を入手したりできます。

関連するサードパーティの Web サイトの参照

Sun は、このマニュアルに記載されているサードパーティ Web サイトの利用可能性について責任を負いません。Sun は、このようなサイトまたはリソースから得られるあらゆる内容、広告、製品、およびその他素材を保証するものではなく、責任または義務を負いません。Sun は、このようなサイトまたはリソースで得られるあらゆるコンテンツ、製品、またはサービスによって生じる、または生じたと主張される、または使用に関連して生じる、または信頼することによって生じる、いかなる損害または損失についても責任または義務を負いません。

ご意見、ご要望の送付先

Sun では、マニュアルの品質向上のため、お客様のご意見、ご要望をお受けしております。次の Web ベースの書式を利用して、Sun までフィードバックをお寄せください。

http://www.sun.com/hwdocs/feedback/

各フィールドにマニュアルの正式名称と Part No. をご記入ください。Part No. は、マニュアルのタイトルページまたは最上部に記載されている 7 桁または 9 桁の番号です。たとえば、このマニュアルの Part No. は、819-2015 です。

第 1 章

Directory Server の概要

Directory Server は、イントラネット、ネットワーク、およびエクストラネットの情報を集中管理するディレクトリサービスを提供します。Directory Server は、既存のシステムと統合することができ、社員、顧客、サプライヤ、パートナーなどの情報を管理する中央リポジトリとして機能します。また、Directory Server を拡張して、ユーザーのプロファイルや環境設定、エクストラネットのユーザー認証などを管理することもできます。

LDAP とディレクトリの基本的な概念については、『Directory Server Technical Overview』を参照してください。この章では、サーバーアーキテクチャの概要を示し、Directory Server のインストールを計画する際に考慮に入れるべき問題点など、設計と配備のプロセスについて高いレベルで説明します。この章は、次の節で構成されています。

サーバーアーキテクチャの概要

Directory Server の配備には、次のような要素が含まれます。

- Directory Server
- Administration Server
- Sun Java System サーバーコンソール

これらの要素のそれぞれは、配備において別々の役割を果たします。

Directory Server は、サーバーおよびアプリケーションの設定と、企業内のその他のサーバーが使用するユーザー情報を格納します。通常、アプリケーションとサーバーの設定情報は Directory Server の 1 つの「サフィックス」に格納され、ユーザーとグループのエントリは別のサフィックスに格納されます。サフィックスはディレクトリツリー内のエントリの名前で、データはその下に格納されます。

設定ディレクトリまたは Configuration Directory Server (CDS) には、Directory Server 自体の設定方法についての情報が格納されます。通常このディレクトリは最初にインストールされ、後続の各サーバーがこのディレクトリに登録されます。単一の設定ディレクトリを使用することによって、全サーバーを中央管理することができます。

ユーザーディレクトリには、ディレクトリサービスにアクセスするユーザーやグループのエントリが格納されます。通常ユーザーディレクトリはネットワークドメインで一意であり、ほかのサーバーがユーザーやグループの情報を得るためにアクセスします。単一のユーザーディレクトリを使用することによって、ユーザーやグループを集中管理できます。

小規模の配備では、設定、ユーザー、およびその他のディレクトリを同じディレクトリインスタンスにインストールすることが可能です。大規模の配備では、設定とユーザーのディレクトリを別々のサーバーに配置することを検討してください。

サーバーコンソール は、すべての Sun Java System サーバーのためのフロントエンド管理アプリケーションです。これは、設定ディレクトリに登録されたすべてのサーバーとアプリケーションを検索し、それらをグラフィカルインタフェースで表示して、ユーザーがそれらを管理および設定できるようにします。

サーバーコンソール にログインすると、HTTP (Hypertext Transfer Protocol) を使用して Administration Server のインスタンスに接続されます。Administration Server は、1 つのルートフォルダにインストールされているすべての Sun Java System 製品に対する要求を管理します。

このアーキテクチャについては、『Administration Server Administration Guide』の「Remote Server Administration Overview」を参照してください。

ディレクトリ設計の概要

ディレクトリの設計段階では、環境、データソース、ユーザー、ディレクトリを使用するアプリケーションなど、ディレクトリに求められる要件に関するデータを収集します。

Directory Server が備える柔軟性により、配備後も、予期しない状況や要件の変更に対応して設計を見直すことができます。ただし、優れた設計によって修正を避けるべきであることは言うまでもありません。

設計プロセスは、次の段階に分けることができます。

- インストールの計画
- データとデータアクセスの計画
- スキーマの設計
- ディレクトリツリーの設計
- トポロジの設計
- レプリケーションの設計
- 安全なディレクトリの設計
- 監視戦略の策定

インストールの計画

Directory Server をインストールする前に、次の点を必ず考慮しておいてください。

1. ディレクトリサービスの配備により複数のディレクトリにインストールするためのサーバー設定、ユーザー、およびグループの集中管理を可能にするには、設定ディレクトリとユーザーディレクトリの適切な位置を決めます。設定、ユーザーおよびグループのデータを保存する適切な位置については、『Administration Server Administration Guide』を参照してください。
2. ホストシステムへの物理アクセスを制限します。Directory Server に多数のセキュリティ機能が組み込まれているとしても、ホストシステムへの物理アクセスが制御されていない場合ディレクトリのセキュリティは危険にさらされます。
3. ホストシステムがスタティック IP アドレスを使用していることを確認します。
4. Directory Server インスタンス自体がネットワークのネームサービスを提供していない場合、または Directory Server の配備によりリモート管理を可能にするには、ネームサービスと、そのホストのドメイン名が適切に構成されていることを確認します。

5. 設計時に Directory Server の各インスタンスに使用するポート番号を選択します。可能であれば、Directory Server が実際の運用段階に入ったあとはポート番号を変更しないでください。ポート番号をあとでコンソールから変更すると、次のスクリプトに対する必要な変更が行われず、これらのスクリプトを手動で変更しなければならなくなります。`bak2db.pl`、`schema_push.pl`、`db2bak.pl`、`check-slapd`、`db2index.pl`、`db2ldif.pl`、`monitor`、`ldif2db.pl`、`ns-accountstatus.pl`、`ldif2ldap`、`ns-activate.pl`、`ns-inactivate.pl`。

   ここに示すスクリプト名はスタンドアロンのツール名であり、`check-slapd` コマンドは、一般に公開されている API の一部でないため、記載されていないことに注意してください。詳細は、『Directory Server Administration Reference』を参照してください。

データとデータアクセスの計画

ディレクトリには、ユーザー名、電話番号、ユーザーが属するグループの情報などのデータが入ります。組織内のさまざまなデータソースを分析し、それらの相互関係を理解するには、第 2 章「ディレクトリデータの計画とアクセス」を参照してください。この章では、ディレクトリ内の保管に適したデータのタイプ、このデータにアクセスするための方法、ディレクトリの内容を設計する際に行う必要があるその他のタスクについて説明します。

スキーマの設計

Directory Server は、ディレクトリ対応アプリケーションをサポートするように設計されています。これらのアプリケーションには、ディレクトリに格納されるデータの特定の要件があります。スキーマは、格納されたデータの特性を決定します。第 3 章「Directory Server のスキーマ」では、Directory Server に含まれる標準スキーマを紹介し、スキーマをカスタマイズする方法を説明します。スキーマの一貫性を保持するためのヒントも記載されています。

ディレクトリツリーの設計

ディレクトリに格納するデータを決めたら、そのデータを編成して、参照させる必要があります。これが、ディレクトリツリーの目的です。第 4 章「ディレクトリ情報ツリー」では、ディレクトリツリーを紹介し、データ階層の設計手順を示します。また、エントリのグループ化と属性の管理を最適化するために使用するメカニズムを説明し、ディレクトリツリー設計の例を示します。

トポロジの設計

トポロジの設計では、ディレクトリツリーを複数の物理サーバー上に分割する方法や、これらのサーバー間での通信方法を決定します。第 5 章「分散、連鎖、およびリフェラル」では、トポロジ設計の基礎となる一般原則について説明します。また、複数のデータベースの使用方法、分散したデータをリンクするのに使用できるメカニズム、Directory Server で分散したデータを追跡する方法について説明します。

レプリケーションの設計

レプリケーションを使用すると、複数の Directory Server が同じディレクトリデータを保持するので、パフォーマンスが向上し、耐障害性が高まります。第 6 章「レプリケーションについての理解」では、レプリケーションのしくみ、レプリケートできるデータの種類、一般的なレプリケーションの使用例について説明します。また、可用性の高いディレクトリサービスを構築するためのヒントを示します。

安全なディレクトリの設計

ディレクトリ内のデータを保護する方法を計画し、ユーザーとアプリケーションのセキュリティ要件を満たすように、サービスを別の側面から検討することは重要です。第 7 章「アクセス制御、認証、および暗号化」では、一般的なセキュリティ上の危険、セキュリティ手法の概要、必要なセキュリティ要件を分析する際の手順について説明します。また、アクセス制御を設計し、ディレクトリデータの整合性を保持するためのヒントを示します。

監視戦略の策定

適切に設計された監視戦略では、ディレクトリの配備が成功したかどうかを評価したり、毎日のディレクトリアクティビティを行うことができます。第 8 章「Directory Server の監視」では、SNMP、Directory Server コンソール、ログファイル、および Directory Server に用意されているデータベース監視ツールとレプリケーション監視ツールを使ってディレクトリを監視する方法について説明します。

ディレクトリ配備の概要

ディレクトリサービスの設計が完了したら、配備段階に進みます。配備段階は、次の段階から構成されます。

- ディレクトリの試験
- 運用環境へのディレクトリの配備

ディレクトリの試験

配備段階の最初の段階では、サーバーのインスタンスを試験的にインストールし、サービスがユーザーの負荷を処理できるかどうかをテストします。サービスが適切でない場合は、設計を調整してパイロットテストを繰り返します。自信を持って全社的に配備できるような堅牢なサービスが完成するまで、設計を調整します。

試験的なディレクトリ作成と実装の概要については、『Understanding and Deploying LDAP Directory Services』(T. Howes、M. Smith、G. Good 著、Macmillan Technical Publishing 発行、1999 年) を参照してください。

運用環境へのディレクトリの配備

パイロットテストを実施して、サービスを調整したら、ディレクトリサービスを試験的な配備から実際の運用環境に配備する計画を立て、それを実行します。次のような内容の運用計画を作成します。

- 必要なリソースの見積もり
- サーバーをインストールする前に実行しておくべき作業
- 達成すべき事柄と作業日程
- 配備が成功したかどうかを測定するための一連の基準

ディレクトリの管理と保守については、『Directory Server 管理ガイド』を参照してください。

第 2 章

ディレクトリデータの計画とアクセス

ディレクトリに格納されるデータには、ユーザー名、電子メールアドレス、電話番号、ユーザーが所属するグループの情報が含まれ、それ以外の種類の情報が含まれる場合もあります。ディレクトリ内のデータのタイプによって、ディレクトリの構成方法、データへのアクセスが可能なユーザー、およびアクセスの要求方法と応答方法が決まります。Directory Server では、LDAP または DSML 経由でディレクトリデータにアクセスできるので、データと直接やり取りを行えるアプリケーションのタイプが広がります。

この章では、ディレクトリデータの計画とアクセスに関連する問題点と手法について説明します。この章は、次の節で構成されています。

ディレクトリデータの概要

データのタイプには、ディレクトリに適しているものとそうでないものがあります。ディレクトリに入れるのに最適なデータには、次のような特性があります。

- 読み取り回数が書き込み回数より多い。

  ディレクトリは読み取り操作用に調整されているため、書き込み操作はサーバーのパフォーマンスを低下させます。

- 属性値の形式で表すことができる (たとえば、`surname=jensen`)。
- 多くのユーザーにとって重要である。

  たとえば、社員の名前やプリンタが実際に置かれている場所は、多くのユーザーやアプリケーションにとって重要です。

- 複数の物理的位置からアクセスされる。

  たとえば、ある社員のアプリケーションの環境設定は、そのアプリケーションだけがこの情報にアクセスできればよいため、ディレクトリには適しません。ただし、このアプリケーションにディレクトリから環境設定を読み込む機能がある場合は、環境設定情報をディレクトリに入れておくと、別のサイトでそのアプリケーションを使用するときに自分の環境設定をディレクトリから読み込めるので便利です。

ディレクトリに格納できるデータ

次に、ディレクトリに格納できるデータの例を示します。

- 電話番号、住所、電子メールアドレスなどの連絡先情報。
- 社員番号、役職、管理者 ID、業務に関する利害関係などの記述的な情報。
- 電話番号、住所、管理者 ID、業務についての説明などの組織の連絡先情報。
- プリンタの設置場所、プリンタの機種、プリンタの印刷速度などの機器に関する情報。
- 会社の取引先、得意先、顧客などの連絡先情報と請求情報。
- 顧客名、期限、作業内容、価格情報などの契約情報。
- 個人のソフトウェア設定やソフトウェア構成に関する情報。
- Web サーバーへのポインタ、特定のファイルやアプリケーションのファイルシステムなどのリソースの場所。

サーバー管理データとは別に、ディレクトリに次のような情報を格納することもできます。

- 契約や顧客アカウントに関する情報
- 給与データ
- 物理的なデバイス情報
- 自宅連絡先情報
- 企業のさまざまなサイトに関する職場連絡先情報

ディレクトリに含めないデータ

Directory Server は、クライアントアプリケーションが読み取りや書き込み (頻繁ではない) を行う膨大な量のデータを管理するには適していますが、画像やほかのメディアなど、構造化されていない大きなオブジェクトを処理するようには設計されていません。このようなオブジェクトは、ファイルシステムで管理する必要があります。ただし、FTP、HTTP、またはその他の URL タイプを使用して、このようなタイプのアプリケーションへのポインタをディレクトリに格納することはできます。

Directory Server は読み取り操作で効果を発揮するので、頻繁に更新させる情報はディレクトリに入れないようにします。書き込み操作の回数を減らすことにより、検索処理全体のパフォーマンスが向上します。

データ要件の定義

ディレクトリデータを設計するときは、現在必要なデータだけでなく、将来含める可能性のあるデータも考慮に入れてください。設計の段階で将来ディレクトリに必要となる要件を考慮することが、データを拡張したり分散させる際に影響します。

配備の計画時には、次の点を考慮してください。

- 現時点でどのようなデータをディレクトリに入れるか。ディレクトリの配備により解決したい当面の問題は何か。使用するディレクトリ対応アプリケーションですぐに必要となるデータは何か。
- 近い将来どのようなデータをディレクトリに格納するか。たとえば、使用している会計システムが現時点では LDAP に対応していないが、近い将来 LDAP または DSML に対応することが分かっている場合などです。この場合、アプリケーションで使用するデータを判別しておき、その機能が利用可能になったときに、データをディレクトリに移行するようにします。
- 将来どのデータをディレクトリに格納する可能性があるか。たとえば、ホスト環境の場合、新しい顧客が現在の顧客とは異なるデータを要求することも考えられます。新しい顧客が、JPEG 画像の格納にディレクトリを使用する可能性もあります。少なくとも、このような方法で計画を行なっていれば、ほかの方法では思いつかなかったデータソースを特定するのに役立ちます。

サイト調査の実施

サイト調査は、ディレクトリの内容を調べ、その特性を把握するための適切な手法です。ディレクトリアーキテクチャで重要なのはデータです。サイト調査には十分な時間をかけてください。サイト調査では次の作業を行います。ここではそれぞれについて簡単に説明し、詳細については後述します。

- ディレクトリを使用するアプリケーションを特定する。

  配備するディレクトリ対応アプリケーションとそのデータ要件を決定します。

- アプリケーションがどのようにディレクトリにアクセスするかを特定する。

  アプリケーションが LDAP または HTTP/SOAP 経由の DSML のどちらのモードを使用するかを決定します。

- データソースを特定する。

  自社内を調査して、データの取得元 (Windows、NetWare ディレクトリ、PBX システム、人事部データベース、電子メールシステムなど) を確認します。

- ディレクトリに入れる必要があるデータの特性を把握する。

  ディレクトリに入れる必要のあるオブジェクト (ユーザーまたはグループなど) と、管理するオブジェクトの属性 (ユーザー名やパスワードなど) を決定します。

- 提供すべきサービスレベルを決定する。

  クライアントアプリケーションにどの程度までディレクトリデータの利用を許可するかを決定し、それに応じてアーキテクチャを設計します。ディレクトリデータをどの程度まで利用可能にするかによって、データのレプリケート方法や、リモートサーバーに格納されているデータに接続するための連鎖ポリシーの設定が変わってきます。

  アプリケーションについては、第 6 章「レプリケーションについての理解」を参照してください。連鎖については、第 5 章「分散、連鎖、およびリフェラル」を参照してください。

- データマスターを特定する。

  データマスターには、ディレクトリデータのプライマリソースが含まれます。このデータは、ロードバランスと回復の目的のために、ほかのサーバーにコピーされている場合があります。各データについて、そのデータのマスターを特定します。

- データ所有者を決定する。

  各データについて、データを最新の状態に保つ責任のあるユーザーを決定します。

- データアクセスを決定する。

  ほかのソースからデータをインポートする場合は、一括インポートと増分更新について計画を立てます。この手法の一環として、どのデータについてもマスターは 1 か所に配置し、そのデータを変更できるアプリケーションの数を制限します。また、データに書き込みを行えるユーザーの数も制限します。このグループのメンバー数が少ないほど、データの整合性が確保でき、管理に伴うオーバーヘッドを軽減できます。

- サイト調査の結果を文書化する。

  多くの組織がディレクトリによって影響を受けるため、影響のある各組織からの代表者によるディレクトリ配備チームを編成することをお勧めします。このチームでサイト調査を実施します。

  会社は一般に、人事、経理、製造 (1 つまたは複数)、営業 (1 つまたは複数)、開発 (1 つまたは複数) などの部署から形成されています。これらの各組織からの代表者をチームに加えると、調査を迅速に進めることができます。また、影響を受けるすべての組織が直接参加することで、部署ごとのローカルデータ保管から、ディレクトリによる集中化されたデータ管理へ移行しやすくなります。

- サイト調査を繰り返す。

  企業の事務所が複数ある場合は、事務所ごとに調査を繰り返す必要があります。各事務所でサイト調査チームを作り、結果を中央のサイト調査チーム (各地の代表者で構成) に送ります。

クライアントアプリケーションの識別

一般的に、ディレクトリにアクセスするアプリケーションと、それらのアプリケーションで必要となるデータが、ディレクトリの内容を決定する主な原因となります。ディレクトリを使用する一般的なアプリケーションには、次のアプリケーションが含まれます。

- White pages などのディレクトリブラウザアプリケーション。これらのアプリケーションは、通常は電子メールアドレス、電話番号、社員名などの情報にアクセスします。
- メッセージングアプリケーション、特に電子メールサーバー。すべての電子メールサーバーは、電子メールアドレス、ユーザー名、およびルーティング情報を必要とします。サーバーの中には、ユーザーのメールボックスが格納されているディスク上の格納場所、休暇通知情報、およびプロトコル情報 (たとえば、IMAP や POP) など、さらに高度な情報を必要とするものもあります。
- ディレクトリ対応の人事管理アプリケーション。このアプリケーションは、政府指定の ID 番号、自宅の住所、自宅の電話番号、生年月日、給与明細、役職など、個人に関する詳細な情報を必要とします。

- セキュリティ、Web ポータル、個人化用アプリケーション。これらのアプリケーションは、プロファイル情報にアクセスします。

ディレクトリを使用するアプリケーションを調査するときは、各アプリケーションが使用する情報のタイプに注目します。次の表に、アプリケーションと各アプリケーションが使用する情報の例を示します。

表 2-1 アプリケーションが必要とするデータ

アプリケーション	データクラス	データ
電話帳	ユーザー	名前、電子メールアドレス、電話番号、ユーザー ID、パスワード、部署番号、マネージャ、メールの配信停止
Web サーバー	ユーザー、グループ	ユーザー ID、パスワード、グループ名、グループのメンバー、グループの所有者
Calendar Server	ユーザー、会議室	名前、ユーザー ID、広さ、会議室の名前
Web ポータル	ユーザー、グループ	ユーザー名、ユーザー ID、パスワード、グループ名、グループのメンバー

アプリケーションおよび各アプリケーションが使用する情報を特定すると、いくつかのタイプのデータが複数のアプリケーションによって使用されていることがわかってきます。データの計画段階でこのような課題に取り組むことで、ディレクトリ内のデータが重複する問題を避けることができます。

ディレクトリ内で管理されるデータと、その管理が開始される時期は、次のことに影響されます。

- 旧バージョンのアプリケーションで必要とされるデータと、そのアプリケーションを使用するユーザーの数。
- 旧バージョンのアプリケーションが LDAP ディレクトリと通信できるかどうか。

データソースの特定

ディレクトリに含まれているデータを特定するには、既存のデータソースを調査します。調査では、次の作業を行います。

- 情報を提供する組織を特定する。

  企業にとって重要な情報を管理している組織をすべて特定します。通常、この組織には、情報サービス部、人事部、給与計算部、および経理部が含まれます。

- 情報のソースであるツールとプロセスを特定する。

一般的な情報ソースには、ネットワーク用のオペレーティングシステム (Windows、Novell Netware、UNIX NIS)、電子メールシステム、セキュリティシステム、PBX (電話交換) システム、人事管理アプリケーションなどがあります。

- データの集中化が各データに与える影響を判定する。

  集中データ管理では、新しいツールとプロセスが必要になることがあります。集中化によって、ある組織のスタッフを増員してほかの組織のスタッフを減らすことが必要になる場合は、問題が生じる可能性もあります。

調査中に、次の表のような雛形を用意して、企業内の情報ソースをすべて特定しておくことをお勧めします。

表 2-2 情報ソース

データソース	データクラス	データ
人事管理データベース	ユーザー	名前、住所、電話番号、部署番号、マネージャ
電子メールシステム	ユーザー、グループ	名前、電子メールアドレス、ユーザー ID、パスワード、電子メールの環境設定
設備システム	設備	建物の名前、フロアの名前、広さ、アクセスコード

ディレクトリデータの特徴づけ

特定するデータは、次のように特徴づけることができます。

- 形式
- サイズ
- さまざまなアプリケーションで使用される回数
- データ所有者
- ほかのディレクトリデータとの関係

ディレクトリに含める計画のある各データをよく調査して、ほかのデータと共通している特徴を明確にします。これにより、第 3 章「Directory Server のスキーマ」で説明しているスキーマの設計段階で、時間を節約することができます。

たとえば、ディレクトリデータの特徴を記載した次のような表を作成することができます。

表 2-3　ディレクトリデータの特徴

データ	形式	サイズ	所有者	関連するエントリ
社員の名前	テキストの文字列	128 文字	人事	ユーザーエントリ
ファックス番号	電話番号	14 桁の数字	設備	ユーザーエントリ
電子メールアドレス	テキスト	多くの文字	情報システム部	ユーザーエントリ

ディレクトリの可用性要件の特定

提供するサービスの可用性のレベルは、ディレクトリ対応のアプリケーションを利用するユーザーが必要とするサービスによって決まります。アプリケーションで必要なサービスレベルを決定するには、まずそのアプリケーションがいつどのように使用されているかを確認します。

ディレクトリの利用が進むにつれて、さまざまなサービスレベルをサポートする必要が出てきます。ディレクトリの配備後にサービスレベルを上げることは困難なので、将来の要件にも対応できる設計を初期の段階から心がけるようにしてください。

データマスターサーバーの特定

データマスターは、データのプライマリソースになるサーバーです。データセンターの物理的な位置が複数ある場合は、データのマスターを置くサーバーと、そのデータのマスターから更新を受け取るサーバーを決定する必要があります。

レプリケーション時のデータマスターの作成

レプリケーションを使用する場合は、どのサーバーをデータのマスターソースにするかを決定します。Directory Server では、複数のサーバーが同じデータのマスターとなることができるマルチマスター設定がサポートされています。レプリケーションとマルチマスターレプリケーションについては、第 6 章「レプリケーションについての理解」を参照してください。

もっとも単純な構成では、すべてのデータのマスターソースを 2 つの Directory Server に置き、そのデータを 1 台または複数のコンシューマサーバーにレプリケートするようにします。2 つのマスターサーバーがあれば、1 つのサーバーが故障してオフラインになったときにフェイルオーバーが実行されます。より複雑な構成では、データを複数のデータベースに格納し、データの更新または検索を行うアプリケーションの近くにあるサーバーが、エントリのマスターを作成するようにします。

複数アプリケーションにわたるデータマスターの作成

ディレクトリと間接的に通信するアプリケーションがある場合には、データのマスターソースについても考慮する必要があります。このような場合は、データの変更処理と、データ変更を実行する場所とを、できる限り単純に保ちます。単一のサイトでデータのマスターを作成することにした場合は、同じサイトでそこに含まれているほかのすべてのデータのマスターを作成します。このようにマスターの作成先を単一のサイトにしておくと、企業内でデータベースの同期がとれなくなった場合の障害追跡が簡単になります。

次に、データマスターの作成方法を示します。

- ディレクトリおよびそのディレクトリを使用しないすべてのアプリケーションの両方で、データマスターを作成する。

  マルチマスターを管理する場合は、ディレクトリやその他のアプリケーションとの間でデータをやり取りするためのカスタムスクリプトは必要ありません。ただし、1 か所でデータが変更されると、ほかのすべてのサイトでもデータを変更する必要があります。ディレクトリおよびそのディレクトリを使用しないすべてのアプリケーションでマスターデータを管理すると、企業全体のデータが同期しなくなることがあります (このような状態をディレクトリが回避しようとする)。

- ディレクトリでデータマスターを作成し、Sun Java System Meta Directory を使用してほかのアプリケーションとデータの同期をとる。

  さまざまなディレクトリ対応アプリケーションやデータベースアプリケーションを使用している場合は、ほかのアプリケーションと同期をとるデータマスターを管理する方法がもっとも合理的です。Meta Directory については、Sun Java System 製品の販売店にお問い合わせください。

  ディレクトリ以外のアプリケーションでデータマスターを作成してから、そのデータをディレクトリにインポートするスクリプト、プログラム、またはゲートウェイを作成します。

  すでに使用しているアプリケーションの中にデータマスターを作成できるものが 1 つまたは 2 つあり、ディレクトリを検索の目的 (オンラインの企業電話帳など) にのみ使用する場合は、ディレクトリ対応以外のアプリケーションでデータマスターを作成する方法がもっとも合理的です。

データのマスターコピーの管理方法は、個々の要件によって異なります。ただし、どのような管理方法を選択しても、簡単で一貫性のあるものにしてください。たとえば、複数のサイトでデータマスターを作成し、競合するアプリケーション間で自動的にデータを交換するようなことは避けてください。そのような方法では、「最新の変更が優先される」ことになり、管理に伴うオーバーヘッドが増大します。

ある社員の自宅の電話番号を管理する場合を考えてみます。この情報は、LDAP ディレクトリと人事 (HR) データベースの両方に格納されています。HR データベースは LDAP 対応なので、LDAP ディレクトリと HR データベースの間で自動的にデータを転送するアプリケーションを作成することができます。ここで、電話番号への変更を

LDAP ディレクトリと HR データベースの両方でマスタリングすると、最後に変更した電話番号のデータが、もう一方のデータベースの情報を上書きしてしまいます。これは、最後にデータを書き込んだアプリケーションが正しい情報を持っている場合には、適切な処理です。ところが、この情報が古い場合 (たとえば、HR データがバックアップから再読み込みされたものである場合など) は、LDAP ディレクトリ内の正しい電話番号が削除されてしまいます。

データ所有者の決定

「データ所有者」とは、データを最新の状態に維持する責任のある個人または組織のことです。データの設計時に、ディレクトリにデータを書き込めるユーザーを決めておきます。データの所有者を決めるには、一般に次の規則に従います。

- ディレクトリの内容を管理する少人数のマネージャグループを除くすべてのユーザーに対して、ディレクトリへの読み取り専用アクセスを許可する。
- 各ユーザーが自分自身に関する重要な情報を管理できるようにする。

  この情報には、パスワード、そのユーザーに関する情報と組織内での役割、自動車のナンバープレート番号、自宅やオフィスの電話番号などの連絡先の情報が含まれます。
- 人に関する重要な情報 (連絡先情報や役職など) を上司が書き込めるようにする。
- 組織の管理者がその組織に関するエントリを作成および管理できるようにする (これにより、実質的に組織の管理者が、ディレクトリの内容の管理者にもなる)。
- グループ内のユーザーに読み取りアクセス権または書き込みアクセス権を与えるロールを作成する。

  たとえば、人事、財務、経理などのロールを作成します。これらのロールごとに、給与情報や政府指定の ID 番号 (社会保障番号)、自宅の電話番号と住所など、そのグループが必要とするデータへの読み取りアクセス権、書き込みアクセス権、またはその両方を許可します。

  ロールとエントリのグループ化については、第 4 章「ディレクトリ情報ツリー」を参照してください。

データに書き込みを許可するユーザーを決定するときに、複数のユーザーに対して同じデータへの書き込み権限が必要になることがあります。たとえば、社員のパスワードへの書き込み権限を、情報システムまたはディレクトリ管理グループに許可するとします。同時に、社員にも自分のパスワードへの書き込み権限を許可する場合があります。複数のユーザーに同じ情報への書き込み権限を与えなければならない場合がありますが、このような場合はこのグループを少人数に保ち、容易に識別できるようにします。グループを少人数に保つことにより、データの整合性を維持しやすくなります。

ディレクトリのアクセス制御の設定については、第 7 章「アクセス制御、認証、および暗号化」を参照してください。

データアクセスの決定

データ所有者を決定したら、各データを読み取ることができるユーザーを決定します。たとえば、ある社員の自宅の電話番号をディレクトリに保存することにした場合を考えてみます。このデータは、その社員の上司や人事部など、多くの組織にとって有用です。また、その社員自身が確認のためにこの情報を読み取れるようにしたい場合もあります。しかし、自宅の連絡先情報は機密事項とも考えられます。したがって、この種のデータを企業全体で広く利用可能にするかどうかを決定する必要があります。

ディレクトリに格納する各情報について、次の事項を決定します。

- データを匿名で読み取れるようにするか。

  LDAP プロトコルでは匿名アクセスがサポーされているので、オフィスの場所、電子メールアドレス、会社の電話番号などの一般情報を簡単に検索できます。ただし、匿名アクセスの場合は、ディレクトリへのアクセス権を持つユーザーであればだれでも一般情報にアクセスできてしまいます。そのため、匿名アクセスの使用はできるだけ避けてください。

- データを企業全体で広く読み取れるようにするか。

  ディレクトリにログイン (あるいはバインド) しない限り、特定の情報を読み取れないようにアクセス制御を設定することができます。匿名アクセスとは異なり、この形式のアクセス制御では、ディレクトリの情報を閲覧できるユーザーを組織内のメンバーだけに限定できます。また、ログイン情報をディレクトリのアクセスログに取り込めるので、情報にアクセスしたユーザーの記録を残すことができます。

  アクセス制御については、180 ページの「アクセス制御の設計」を参照してください。

- データを読み取る必要があるユーザーグループまたはアプリケーションを特定できるようにするか。

  一般に、データへの書き込み特権を持つユーザーには読み取り権限も必要です (パスワードへの書き込み権限は例外)。また、特定の組織やプロジェクトグループだけが必要とするデータが存在することがあります。これらのアクセス要件を特定しておくと、ディレクトリで必要となるグループ、ロール、およびアクセス制御を決める際に役立ちます。

  グループとロールについては、第 4 章「ディレクトリ情報ツリー」を参照してください。アクセス制御については、第 7 章「アクセス制御、認証、および暗号化」を参照してください。

ディレクトリの各データに対してこれらの事項を決定する際には、ディレクトリのセキュリティポリシーを定義していることが前提となります。これらの決定は、サイトの性質と、すでに利用可能なセキュリティのタイプによって異なります。たとえば、サイトにファイアウォールが使用されている場合や、インターネットに直接アクセスしていない場合は、インターネット上に直接ディレクトリを配置している場合に比べ、匿名アクセスをサポートしやすくなります。

多くの国では、データ保護に関する法律によって個人情報をどのように管理すべきかが規定されており、個人情報にアクセスする人を制限しています。たとえば、法律によって住所や電話番号への匿名アクセスが禁止されていたり、住所や電話番号を表すエントリ内の情報を閲覧、訂正する許可をユーザーに与えなければならない場合があります。社内の法務部に問い合わせて、ディレクトリの配備が、業務拠点としている国々の該当する法律に違反していないことを確認してください。

セキュリティポリシーの作成と実装方法については、第 7 章「アクセス制御、認証、および暗号化」で詳しく説明します。

サイト調査の記録

データの設計は複雑なため、サイト調査の結果は文書に記録しておくことをお勧めします。サイト調査の各段階で、データを把握するための簡単な表を作成することをお勧めしました。決定事項と未解決の問題を概説する簡単な表を作成することを検討してください。

表 2-4 は、基本的なデータ追跡例を示しています。この表には、データ所有者と、サイト調査で特定した各データについてのデータアクセス権限が示されています。

表 2-4 サイト調査を記録するためのデータ追跡表の例

データ名	所有者	マスターサーバーアプリケーション	本人による読み取り / 書き込み	全員による読み取り	人事部 (HR) による書き込み	情報システム部 (IS) による書き込み
社員の名前	HR	People Soft	読み取り専用	可 (匿名)	はい	はい

表 2-4　サイト調査を記録するためのデータ追跡表の例 (続き)

データ名	所有者	マスターサーバー アプリケーション	本人による読み取り / 書き込み	全員による読み取り	人事部 (HR) による書き込み	情報システム部 (IS) による書き込み
ユーザーパスワード	IS	Directory US-1	読み取り / 書き込み	いいえ	いいえ	はい
自宅の電話番号	HR	People Soft	読み取り / 書き込み	いいえ	はい	いいえ
社員の所属場所	IS	Directory US-1	読み取り専用	可 (ログインが必要)	いいえ	はい
会社の電話番号	設備	電話スイッチ	読み取り専用	可 (匿名)	いいえ	いいえ

社員名を表す行には、次の項目が含まれています。

- 所有者

  人事部がこの情報を所有し、この情報の更新と変更の責任を負います。

- マスターサーバー / アプリケーション

  PeopleSoft というアプリケーションで社員名に関する情報を管理します。

- 本人による読み取り / 書き込み

  ユーザーは自分の名前の読み取りはできますが、書き込み (変更) はできません。

- 全員による読み取り

  ディレクトリへのアクセス権を持つすべてのユーザーは、匿名で社員名を読み取ることができます。

- 人事部 (HR) による書き込み

  HR グループのメンバーは、社員名を追加、変更、および削除することができます。

- 情報システム部 (IS) による書き込み

  情報サービスグループのメンバーは、社員名を追加、変更、および削除することができます。

サイト調査の繰り返し

サイト調査は数回実施した方が良い場合があります。特に、複数の都市や国にオフィスを持つ企業の場合は、これが当てはまります。情報に関する要件があまりにも複雑なため、中央のサイトで情報を一元的に管理するよりも、複数の組織がそれぞれの現地オフィスで情報を保管するようにしなければならない場合もあります。このような場合は、情報のマスターコピーを保管する各オフィスで、独自のサイト調査を実施するようにします。この過程の完了後、中央のチーム (各オフィスの代表者で構成される) に各調査結果が戻され、企業全体のデータスキーマモデルとディレクトリツリーの設計に使用します。

HTTP/SOAP 経由の DSML によるディレクトリデータへのアクセス

Directory Server 5.2 では、HTTP/SOAP 経由の DSMLv2 (Directory Service Markup Language version 2) を使用して、ディレクトリデータにアクセスすることもできます。

Directory Server 5.2 より前のバージョンの Directory Server では、LDAP (Lightweight Directory Access Protocol) を使用してディレクトリデータにアクセスできます。

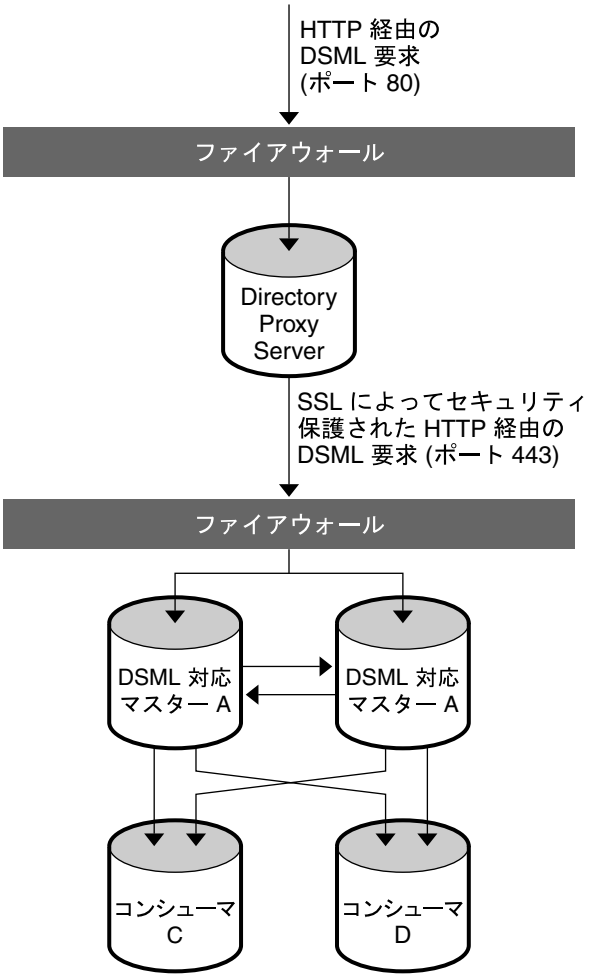
DSMLv2 はマークアップ言語、つまり、ディレクトリサービスによるデータ処理の構造と内容を XML (eXtensible Markup Language) ドキュメントとして記述するためのボキャブラリとスキーマです。DSMLv2 は、XML でディレクトリサービス情報を表す方法を標準化します。Directory Server では、ハイパーテキスト転送プロトコル (HTTP/1.1) で DSMLv2 を使用できます。また、DSML の内容を転送するためのプログラミングプロトコルとして SOAP (Simple Object Access Protocol) バージョン 1.1 が使われます。

DSML フロントエンドの設定と、HTTP/SOAP 経由の DSMLv2 を使用したデータへのアクセスとデータの検索については、『Directory Server 管理ガイド』の「DSML の設定」を参照してください。

HTTP/SOAP 経由の DSMLv2 の配備

DSML 対応の Directory Server と Sun Java System Web Proxy Server を使用した次の配備例では、非 LDAP クライアントがディレクトリデータとやり取りできます。

図 2-1　DSML 対応のディレクトリの配備例



この配備例では、非 LDAP クライアントアプリケーションからの DSML 形式の更新要求は、まず、HTTP ポート 80 でファイアウォールを通過し、非武装地帯 (DMZ) に入ります。ここから、逆プロキシサーバーとして設定された Directory Proxy Server は、セキュリティ保護された HTTP ポート 443 の使用を強制し、要求は第 2 のファイアウォールを越えてイントラネットドメインに入ります。要求は、DSML に対応していないコンシューマ C、D にレプリケートされる前に、2 つのマスターレプリカ Master A、B で処理されます。

このような配備により、非 LDAP アプリケーションでディレクトリの操作を実行することができます。クライアント要求が単なる検索要求の場合、DSML 対応の Directory Server は、データのコピーが読み取り専用であっても、読み書き両方に対応していても、どちらも検索要求を処理できるので問題ありません。しかし、非 LDAP クライアントが変更要求を送信する場合、DSML 対応 Directory Server は読み書きに対応したデータコピーを保持している必要があります。変更要求を受け取ったコンシューマは、デフォルトの設定では、クライアントの変更要求を正しく実行できる可能性のあるマスターの LDAP URL リストのリフェラルを返します。非 LDAP クライアントアプリケーションに対して HTTP 経由で LDAP URL を返しても、クライアントとディレクトリの間のトラフィックで LDAP を使用しないという目的を果たすことができません。このために読み書き両方に対応したコピーが必要なのです。図 2-1 の配備では、DSML 対応の Directory Server のマスター A、B は読み書き両方に対応したデータコピーを保持します。これらのマスターは、変更要求を処理し、DSML に対応していないコンシューマ C、D にデータをレプリケートします。

DSML フロントエンドは、アクセスを制限した HTTP サーバーを構築しています。このサーバーは、DSML HTTP ポスト処理だけを受け付け、SOAP/DSML 仕様に準拠していない要求を拒否します。これにより、その他の種類の HTTP Web サーバーと比較してリスクの範囲は狭められています。配備に DSML 対応の Directory Server を含めるときは、それでもなお次のセキュリティ上の注意点に留意する必要があります。

- ファイアウォールを実装して、DSML 対応の Directory Server を保護する。
- ポート 443 で SSL でセキュリティ保護された HTTP を使用するか、クライアントで SSL 経由の HTTP を使用したくない場合は、Web プロキシサーバーを実装する。

第 3 章

Directory Server のスキーマ

第 2 章で実施したサイト調査により、ディレクトリに格納しようと計画しているデータについての情報を収集できました。次に、このデータの表現方法を決める必要があります。ディレクトリスキーマは、ディレクトリに格納できるデータのタイプを表します。スキーマの設計時には、各データ要素を LDAP 属性に割り当て、関連する要素を集めて LDAP オブジェクトクラスに入れます。スキーマを適切に設計することで、データの整合性を維持できます。

この章では、スキーマの設計方法について次の項目ごとに説明します。

Directory Server のオブジェクトクラスと属性、およびスキーマファイルとディレクトリ設定属性については、『Directory Server Administration Reference』を参照してください。サーバー間でのスキーマのレプリケーション方法については、156 ページの「スキーマのレプリケーション」を参照してください。

Directory Server のスキーマ

ディレクトリスキーマは、データ値のサイズ、範囲、および形式に制限を課すことにより、データの整合性を維持します。設計者は、ディレクトリに含まれるエントリの種類 (ユーザー、デバイス、組織など) と各エントリで使用できる属性を決めます。

Directory Server に事前に定義されているスキーマには、標準の RFC LDAP スキーマ、サーバーの機能をサポートするための追加アプリケーション固有のスキーマ、および Directory Server に固有のスキーマ拡張が含まれます。定義済みのスキーマは大半のディレクトリの要件を満たしますが、ユーザー独自の要件にも対応できるように、このスキーマを拡張して新しいオブジェクトクラスと属性を追加する必要がある場合もあります。スキーマの拡張方法については、48 ページの「スキーマのカスタマイズ」を参照してください。

Directory Server は、LDAP プロトコルバージョン 3 (LDAPv3) のスキーマ形式に準拠しています。このプロトコルでは、ディレクトリサーバーが LDAP 自体を通じてスキーマを公開することによって、ディレクトリクライアントアプリケーションがプログラムを使用してスキーマを検索し、検索したスキーマに基づいて自分の動作を調整できるようにする必要があります。Directory Server のグローバルスキーマセットは、`cn=schema` というエントリに含まれています。

Directory Server のスキーマは、RFC 2256 のコア LDAPv3 スキーマだけでなく、多数の一般的なスキーマもサポートしています。また、Directory Server はスキーマエントリ内で X-ORIGIN という非公開フィールドを使用します。このフィールドは、スキーマエントリの定義元を示します。たとえば、スキーマエントリが標準 LDAPv3 スキーマで定義されている場合、X-ORIGIN フィールドの値は RFC 2252 になります。スキーマエントリが、Directory Server 用に Sun で定義されている場合は、X-ORIGIN フィールドに `Sun ONE Directory Server` という値が入ります。

たとえば、標準の person オブジェクトクラスはスキーマ内で次のように示されます。

```
objectclasses: ( 2.5.6.6 NAME 'person' DESC 'Standard Person
 Object Class' SUP top MUST (objectclass $ sn $ cn) MAY
(description $ seealso $ telephoneNumber $ userPassword) X-ORIGIN
'RFC 2252' )
```

このスキーマエントリは、クラスのオブジェクト識別子 (OID) (`2.5.6.6`)、オブジェクトクラス名 (`person`)、クラスの説明 (`Standard Person Object Class`)、必須の属性 (`objectclass`、`sn`、および `cn`)、および許可された属性 (`description`、`seealso`、`telephoneNumber`、および `userPassword`) を示しています。

すべての Directory Server のスキーマと同様、オブジェクトクラスは Directory Server で直接定義され、格納されています。これは、ディレクトリのスキーマを標準の LDAP 操作で問い合わせたり、変更したりできるということです。

スキーマ設計のプロセス

スキーマの設計時には、Directory Server によって格納されるエントリを表すのに使用するオブジェクトクラスと属性を選択および定義します。スキーマの設計は、次の手順で行います。

- できるかぎり多くの要件を満たす、定義済みのスキーマ要素を選択する。
- Directory Server の標準スキーマを拡張して、残りの要件を満たす新しい要素を定義する。
- スキーマの保守を計画する。

可能であれば、最善の方法は、Directory Server が提供する標準スキーマに定義されている既存のスキーマ要素を使用することです。標準スキーマの要素を選択すれば、ディレクトリを使用するアプリケーションとの互換性を保証できます。また、スキーマは LDAP 標準に基づいているので、多くのディレクトリユーザーによってレビューされ、承認されています。

デフォルトスキーマへのデータの割り当て

30 ページの「サイト調査の実施」で説明したように、サイト調査で識別したデータをデフォルトディレクトリスキーマに割り当てる必要があります。この節では、デフォルトスキーマを表示する方法と、適切なスキーマ要素にデータを割り当てる方法について説明します。

デフォルトスキーマとマッチしない要素が独自のスキーマ内にある場合は、カスタマイズしたオブジェクトクラスと属性を作成する必要があります。詳細は、48 ページの「スキーマのカスタマイズ」を参照してください。

デフォルトのディレクトリスキーマの表示

Directory Server で提供されるスキーマは、次のディレクトリに保存される一連のファイルに記述されています。

ServerRoot/slapd-*serverID*/config/schema

このディレクトリには、Sun Java System 製品のすべての共通スキーマが入っています。LDAPv3 標準のユーザースキーマと組織スキーマは、00core.ldif ファイルに記述されています。旧バージョンのディレクトリで使用された設定スキーマは、50ns-directory.ldif ファイルに記述されています。

注 サーバーの稼動中は、このディレクトリ内のファイルを変更しないでください。

手動で加えた変更は、LDAP または Directory Server コンソールを使用して別の変更を加えるまでレプリケートされません。

データとスキーマ要素の対応付け

サイト調査で識別したデータを既存のディレクトリスキーマに割り当てる必要があります。この作業は、次の手順で行います。

- データが記述するオブジェクトのタイプを識別する。

  サイト調査に記載されているデータにもっとも適したオブジェクトを選択します。データで複数のオブジェクトを記述できる場合があります。必要に応じて別の要素をスキーマに入れるかどうかを決める必要があります。たとえば、電話番号に社員の電話番号と会議室の電話番号を記述できます。これらの電話番号データを、スキーマで異なるオブジェクトとみなすかどうかは設計者が決めます。

- デフォルトスキーマから類似のオブジェクトクラスを選択する。

  group、people、organization などの共通オブジェクトクラスを使用します。

- 対応するオブジェクトクラスから類似の属性を選択する。

  対応するオブジェクトクラスから、サイト調査で識別したデータにもっとも適した属性を選択します。

- サイト調査で収集したデータの中で対応しないデータを識別する。

  デフォルトのディレクトリスキーマで定義されているオブジェクトクラスと属性に対応しないデータがある場合は、スキーマをカスタマイズする必要があります。詳細は、48 ページの「スキーマのカスタマイズ」を参照してください。

次の表に、ディレクトリスキーマの要素をサイト調査で識別したデータに割り当てた例を示します。

表 3-1 デフォルトディレクトリスキーマに割り当てられたデータ

データ	所有者	オブジェクトクラス	属性
社員名	HR	person	cn(commonName)
ユーザーパスワード	IS	person	userPassword
自宅の電話番号	HR	inetOrgPerson	homePhone
社員の所属場所	IS	inetOrgPerson	localityName
会社の電話番号	設備	person	telephoneNumber

表 3-1 では、個人を社員名で表しています。デフォルトのディレクトリスキーマには、top オブジェクトクラスから継承する person オブジェクトクラスがあります。このオブジェクトクラスには複数の属性が許可されており、その中に個人の氏名を記述する cn (commonName) という属性があります。この属性は、社員名データを入れる目的にもっとも適しています。

ユーザーパスワードも person オブジェクトに関連付けることができます。person オブジェクトの許可された属性に userPassword が含まれています。

自宅の電話番号は、特定の個人のある側面を表すものです。しかし、person オブジェクトクラスに関連するリストには、該当する属性がありません。自宅の電話番号をより本質的に分析すると、これは、組織的な企業ネットワークにおいて、特定の個人を表すものだといえます。このためこのオブジェクトは、ディレクトリスキーマの inetOrgPerson オブジェクトクラスに対応します。inetOrgPerson オブジェクトクラスは organizationalPerson オブジェクトクラスから継承し、organizationalPerson オブジェクトクラスは person オブジェクトクラスから継承します。inetOrgPerson オブジェクトの許可された属性には、社員の自宅の電話番号を入れるのに適した homePhone 属性があります。

スキーマのカスタマイズ

標準スキーマの制限が多すぎて、ディレクトリの要件に対応できない場合は、標準スキーマを拡張できます。Directory Server コンソール は、スキーマ定義を管理する上で役立ちます。詳細は、『Directory Server 管理ガイド』の「ディレクトリスキーマの拡張」を参照してください。

スキーマをカスタマイズするときは、次の規則に留意してください。

- できるかぎり既存のスキーマ要素を再利用する。既存のすべてのスキーマ要素のリストは、『Directory Server Administration Reference』の「Object Class Reference」および「Attribute Reference」に記載されています。
- 各オブジェクトクラスに定義する必須の属性の数を最小限にする。
- 同じ目的で複数のオブジェクトクラスまたは属性を定義しない。
- できるかぎりスキーマを簡潔にする。

注 スキーマをカスタマイズする場合は、標準スキーマの属性またはオブジェクトクラスの既存の定義の変更、削除、および置換は行わないでください。標準スキーマを修正すると、ほかのディレクトリや LDAP クライアントアプリケーションとの互換性に問題が生じます。

注 Directory Server の内部オペレーショナル属性は変更しないでください。これらの属性は、Sun により今後のリリースで変更または書き替えられる可能性があります。ただし、外部アプリケーション用に独自のオペレーショナル変数を作成できます。

カスタムのオブジェクトクラスと属性は、次のファイル内に定義されます。

ServerRoot/slapd-*serverID*/config/schema/99user.ldif

次の項目ごとに、ディレクトリスキーマのカスタマイズについて詳しく説明します。

- スキーマの拡張が必要な場合
- オブジェクト識別子の取得と割り当て
- 属性とオブジェクトクラスの命名
- 新しいオブジェクトクラスの定義戦略
- 新しい属性の定義方法
- スキーマ要素の削除
- カスタムスキーマファイルの作成 - 最良の事例と落とし穴

スキーマの拡張が必要な場合

Directory Server が提供するオブジェクトクラスと属性は、ユーザーのほとんどの要件を満たしますが、既存のオブジェクトクラスによっては組織の特殊な情報を格納できない場合もあります。また、LDAP 対応アプリケーションの独自のデータ要件に適したオブジェクトクラスや属性をサポートできるように、スキーマを拡張しなければならない場合もあります。

オブジェクト識別子の取得と割り当て

各 LDAP オブジェクトクラスまたは属性には、一意の名前とオブジェクト識別子 (OID) が割り当てられている必要があります。スキーマを定義するときは、組織に固有の OID が必要です。1 つの OID ですべてのスキーマ要件に対応できます。別の階層レベルを追加するだけで、属性とオブジェクトクラスに新しい分岐を作成できます。OID の取得とスキーマでの割り当ては、次の手順で行います。

- IANA (Internet Assigned Numbers Authority) または国内の機関から組織の OID を取得する。

  国によっては、企業にすでに OID が割り当てられています。所属する組織がまだ OID を持っていない場合は、IANA から取得できます。詳細は、IANA の Web サイト `http://www.iana.org/cgi-bin/enterprise.pl` を参照してください。

- OID の割り当てを追跡できるように、OID レジストリを作成する。

  OID レジストリは、ディレクトリスキーマで使用する OID と OID の説明を提供するリストで、作成者が保持します。このレジストリにより、OID が複数の目的に使用されないようにすることができます。次に、スキーマで OID レジストリを公開する必要があります。

- スキーマ要素を入れるために、OID ツリーに分岐を作成する。

  OID 分岐またはディレクトリスキーマの下に少なくとも 2 つの分岐 (1 つは属性用の *OID.1*、もう 1 つはオブジェクトクラス用の *OID.2*) を作成します。独自のマッチングルールや制御を定義する場合は、必要に応じて *OID.3* などの新しい分岐を追加できます。

属性とオブジェクトクラスの命名

新しい属性やオブジェクトクラスに名前を付けるときは、できるかぎりわかりやすいものにします。わかりやすい名前にすると、Directory Server の管理者がスキーマを使いやすくなります。

作成する要素に固有の接頭辞を付けて、作成したスキーマ要素と既存のスキーマ要素間での名前の衝突を防ぎます。たとえば、Example.com 社では、各カスタムスキーマ要素の前に `Example` という接頭辞を追加しています。また、ディレクトリ内の Example.com 社員を識別するために `ExamplePerson` という特別なオブジェクトクラスを追加しています。

新しいオブジェクトクラスの定義戦略

ディレクトリのエントリに格納する必要がある情報の中に既存のオブジェクトクラスがサポートしていないものがある場合は、新しいオブジェクトクラスを追加します。新しいオブジェクトクラスを作成するには、次の 2 つの方法があります。

- 属性を追加するオブジェクトクラス構造ごとに 1 つずつ、多数の新しいオブジェクトクラスを作成する。
- ディレクトリ用に作成するすべての属性を含む 1 つのオブジェクトクラスを作成する。このオブジェクトクラスは AUXILIARY オブジェクトクラスとして定義して作成する。

2 つの方法を併用するのがもっとも簡単です。

サイトに `ExampleDepartmentNumber` と `ExampleEmergencyPhoneNumber` という属性を作成するとします。これらの属性にいくつかのサブセットを許可する複数のオブジェクトクラスを作成できます。`ExamplePerson` というオブジェクトクラスを作成し、そのオブジェクトクラスが `ExampleDepartmentNumber` と `ExampleEmergencyPhoneNumber` を許可するようにします。`ExamplePerson` の親は `inetOrgPerson` であるとします。`ExampleOrganization` というオブジェクトクラスを作成し、そのオブジェクトクラスが `ExampleDepartmentNumber` と `ExampleEmergencyPhoneNumber` を許可するようにします。`ExampleOrganization` の親は `organization` オブジェクトクラスであるとします。

新しいオブジェクトクラスは、LDAPv3 スキーマ形式では次のようになります。

```
objectclasses: ( 1.3.6.1.4.1.42.2.27.999.1.2.3 NAME 'ExamplePerson'
 DESC 'Example Person Object Class' SUP inetorgPerson STRUCTURAL MAY
 (ExampleDepartmentNumber $ ExampleEmergencyPhoneNumber) )
```

```
objectclasses: ( 1.3.6.1.4.1.42.2.27.999.1.2.4 NAME
 'ExampleOrganization' DESC 'Example Organization Object Class' SUP
 organization STRUCTURAL MAY (ExampleDepartmentNumber
 $ ExampleEmergencyPhoneNumber) )
```

このようにする代わりに、これらの属性のすべてを許可する 1 つのオブジェクトクラスを作成して、これらの属性を使用する任意のエントリでこのオブジェクトクラスを使用できるようにします。1 つのオブジェクトクラスは、次のようになります。

```
objectclasses: (1.3.6.1.4.1.42.2.27.999.1.2.5 NAME 'ExampleEntry'
 DESC  'Example Auxiliary Object Class' SUP top  AUXILIARY MAY
 (ExampleDepartmentNumber $ ExampleEmergencyPhoneNumber) )
```

新しい ExampleEntry オブジェクトクラスには、構造上のオブジェクトクラスに関係なく任意のエントリで使用できることを示す AUXILIARY が付いています。

注 この例の新しいオブジェクトクラスの OID は、Sun Java System OID 接頭辞に基づいており、配備した製品内では使用できません。独自の新しいオブジェクトクラスを作成するには、独自の OID を取得する必要があります。詳細は、49 ページの「オブジェクト識別子の取得と割り当て」を参照してください。

新しいオブジェクトクラスを実装する方法を決めるときは、次の点に留意します。

- 複数の STRUCTURAL オブジェクトクラスを作成すると、作成および管理するスキーマ要素の数も増える。

  一般に、要素の数が少なければ、管理の手間も少なくて済みます。しかし、スキーマに 2 ～ 3 つのオブジェクトクラスを追加する場合は、1 つのオブジェクトクラスを使用する方が簡単です。

- 複数の STRUCTURAL オブジェクトクラスを作成する場合は、より厳密かつ注意深いデータ設計が必要となる。

  データを厳密に設計するには、個々のデータを配置するオブジェクトクラス構造を考慮する必要があります。これは、有用であることもあれば、面倒なこともあります。

- 複数のタイプのオブジェクトクラス構造に入れたいデータがある場合は、1 つの AUXILIARY オブジェクトクラスを使用した方がデータ設計が簡単になる。

  たとえば、preferredOS 属性を人のエントリとグループエントリの両方に設定するとします。このような場合は、1 つのオブジェクトクラスを作成して、そのクラスでこの属性が許可されるようにします。

- 目的に合ったグループを構成する実際のオブジェクトとグループ要素に関連するオブジェクトクラスを設計する。

- 新しいオブジェクトクラスに必須の属性を設定しない。

  必須の属性を設定するとスキーマに柔軟性がなくなります。新しいオブジェクトクラスを作成する場合は、必須の属性より許可の属性にするようにします。

新しいオブジェクトクラスを定義したら、そのオブジェクトクラスの許可された属性と必須の属性、および継承するオブジェクトクラスを決める必要があります。

新しい属性の定義方法

ディレクトリのエントリに格納する必要がある情報の中に既存の属性がサポートしていないものがある場合は、新しい属性を追加します。できるかぎり、標準属性を使用するようにします。デフォルトのディレクトリスキーマにある属性を探し、それらを新しいオブジェクトクラスに関連付けて使用します。

たとえば、`person`、`organizationalPerson`、または `inetOrgPerson` の各オブジェクトクラスがサポートしている以外の情報を、個人のエントリに格納したい場合があります。ディレクトリに生年月日を格納する場合、Directory Server の標準スキーマには対応する属性がありません。したがって、`dateOfBirth` という新しい属性を作成し、この属性を許可する新しい補助クラスを定義して、個人を表すエントリでこの属性を使用できるようにします。

スキーマ要素の削除

Directory Server に含まれているスキーマ要素は削除しないでください。未使用のスキーマ要素は、Directory Server の動作や管理において、オーバーヘッドになることはありません。標準 LDAP スキーマの一部を削除すると、将来提供される新しいバージョンの Directory Server や LDAP 対応アプリケーションとの互換性に問題が生じる可能性があります。

拡張したスキーマの新しい要素を使用しないときは、その不要要素を削除できます。スキーマ要素を削除する前に、ディレクトリ内のどのエントリもそれを使用しないことを確認してください。このためのもっとも簡単な方法は、そのスキーマ要素を含むすべてのエントリを返す ldapsearch を実行することです。

たとえば、`myObjectClass` というオブジェクトクラスを削除する前に、次の ldapsearch コマンドを実行します。

```
ldapsearch -h host -p port -s base "objectclass=myObjectClass"
```

これに該当するエントリが見つかった場合は、それを削除するか、スキーマから消去される部分を削除することができます。あるスキーマ定義を使用するエントリを削除する前にそのスキーマ定義を削除すると、そのエントリを後から変更できなくなることがあります。不明な値をエントリから削除しない限り、変更されたエントリに関するスキーマ検査も失敗します。

カスタムスキーマファイルの作成 - 最良の事例と落とし穴

Directory Server に含まれている `99user.ldif` ファイルのほかに、独自のカスタムスキーマファイルを作成できます。ただし、カスタムスキーマファイルを作成するとき、特にレプリケーションを使用する場合は、次の点に注意する必要があります。

- 新たに追加したスキーマ要素をオブジェクトクラスで使用するためには、事前にすべての属性が定義されている必要があります。属性とオブジェクトクラスは同じスキーマファイル内で定義できます。
- 作成する各カスタム属性またはオブジェクトクラスは、1 つのスキーマファイル内でだけ定義されている必要があります。これにより、サーバーが最新のスキーマを読み込むときに、前の定義を上書きするのを防ぐことができます。サーバーは、最初に数字順、次にアルファベット順にスキーマを読み込みます。
- 新しいスキーマ定義を手動で作成するときは、その定義を `99user.ldif` ファイルに追加する方法が最も適しています。

  LDAP を使用するスキーマ要素を更新すると、新しい要素が自動的に `99user.ldif` ファイルに書き込まれます。このため、カスタムスキーマファイルに加えたそれ以外のスキーマ定義の変更が、上書きされる可能性があります。`99user.ldif` ファイルのみを使用すると、スキーマ要素の重複と、スキーマの変更の上書きを防ぐことができます。
- Directory Server はスキーマファイルを英数字順に読み込む場合、つまり、数字が小さいものから先に読み込む場合、カスタムスキーマファイルの名前を次のように指定する必要があります。

  `[00-99]`*filename*`.ldif`

  この数字は、すでに定義されているどのディレクトリ標準スキーマよりも大きな値にする必要があります。

  スキーマファイルの名前を標準のスキーマファイルより小さい数字で指定すると、そのスキーマを読み込むときにサーバーにエラーが発生することがあります。また、標準の属性およびオブジェクトクラスはすべて、カスタムスキーマ要素が読み込まれた後にしか読み込まれません。
- 作成するカスタムスキーマのファイル名は、数値やアルファベットの順序で `99user.ldif` を超えないよう、注意してください。なぜなら Directory Server は、内部スキーマの管理のために、(数値順、次にアルファベット順で) もっとも大きい名前を持つファイルを使用するからです。

作成したスキーマファイルに 99zzz.ldif という名前を付けると、次に LDAP または Directory Server コンソールを使用してスキーマを更新したときに、'user defined' (通常は 99user.ldif ファイルに格納される) の値として X-ORIGIN を持つすべての属性が 99zzz.ldif に書き込まれます。その結果、重複した情報を持つ 2 つの LDIF ファイルが存在し、99zzz.ldif ファイル内のいくつかの情報が削除される可能性があります。

- 原則として、追加するカスタムスキーマ要素の識別には、次の 2 つの項目を使用します。
  - カスタムスキーマファイルの X-ORIGIN フィールドに指定されている 'user defined'
  - ほかの管理者カスタムスキーマ要素を理解しやすくするため、X-ORIGIN フィールドの 'Example.com Corporation defined' のような、よりわかりやすいラベル。たとえば、X-ORIGIN ('user defined' 'Example.com Corporation defined') など

  スキーマ要素を手動で追加し、X-ORIGIN フィールドの 'user defined' を使用しない場合、これらのスキーマ要素は Directory Server コンソールの読み取り専用セクションに表示されるため、コンソールを使用して編集することができません。

  'user defined' という値は、LDAP または Directory Server コンソールを使用してカスタムスキーマ定義を追加する場合は、サーバーによって自動的に追加されます。ただし、X-ORIGIN フィールドによりわかりやすい値を追加しないと、どのスキーマが関連しているかを後で理解することが難しくなります。

- 変更は自動的にはレプリケートされないため、カスタムスキーマファイルはすべてのサーバーに手動で伝達します。

  ディレクトリスキーマを変更すると、サーバーはスキーマがいつ変更されたのかを示すタイムスタンプを記録します。各レプリケーションセッションの最初に、サーバーはコンシューマのタイムスタンプとこのタイムスタンプを比較し、必要であればスキーマの変更をプッシュします。カスタムスキーマファイルについては、サーバーは 99user.ldif ファイルに関連付けられている 1 つのタイムスタンプだけを維持します。つまり、カスタムスキーマファイルに加えた変更、または 99user.ldif 以外のファイルに対する変更は、レプリケートされません。このため、トポロジ全体にすべてのスキーマ情報が行き渡るように、カスタムスキーマファイルをほかのすべてのサーバーに伝達する必要があります。

  カスタムスキーマの変更は、次のいずれかの方法で伝達できます。

  - schema_push.pl スクリプトを実行して変更をレプリケートする。
  - 作成したカスタムスキーマファイルをすべてのサーバーに手動でコピーする。

どちらの方法でも、それぞれのサーバーを再起動する必要があります。schema_push.pl スクリプトを使用してカスタムスキーマ定義をレプリケートする場合は、1 つのマスターだけでスキーマを維持する必要があります。スキーマ定義が存在しないコンシューマにスキーマ定義がレプリケートされると、それを定義したカスタムスキーマファイルではなく、99user.ldif ファイルに定義が格納されます。スキーマ要素をコンシューマの 99user.ldif ファイルに格納しても、1 つのマスターサーバーだけでスキーマを管理している限り問題はありません。

スキーマファイルを手動でコピーする方法を選択した場合は、変更のたびにファイルをコピーする必要があります。変更のたびにコピーしないと、変更がレプリケートされ、コンシューマの 99user.ldif ファイルに格納されることがあります。変更が 99user.ldif ファイル内に格納されるとスキーマの管理が難しくなります。これは、一部の属性が、コンシューマ上の 2 つのスキーマファイル (サプライヤからコピーした元のカスタムスキーマファイルとレプリケート後の 99user.ldif ファイル) の両方に現れるためです。

- カスタムスキーマ要素がレプリケーショントポロジ内の別のサーバーにレプリケートされないようにするには、次のようにします。
  - レプリケートしたくないスキーマ要素を別のファイルに定義する。
  - それらの要素を X-ORIGIN フィールドの 'user defined' として識別しない。
  - X-ORIGIN フィールドで 'user defined' というラベルを持つスキーマだけがレプリケートされるように、nsslapd-schema-repl-useronly 属性を on に設定する。

注 また、バージョン 5.0 または 5.1 の Directory Server にレプリケートするときは、この nsslapd-schema-repl-useronly 属性を on に設定する必要があります。

スキーマのレプリケートについては、156 ページの「スキーマのレプリケーション」を参照してください。

データの整合性の維持

Directory Server 内のスキーマの整合性を維持すると、LDAP クライアントアプリケーションがディレクトリのエントリを検出しやすくなります。ディレクトリに格納している情報のタイプごとに、その情報をサポートするために必要なオブジェクトクラスと属性を選択し、常に同じものを使用します。整合性のないスキーマオブジェクトを使用すると、情報を効率よく検出できなくなります。

次のようにすると、整合性のあるスキーマを維持できます。

- スキーマ検査を使用して、属性とオブジェクトクラスが必ずスキーマ規則にマッチしていることを確認する。
- 整合性のあるデータ形式を選択して適用する。

次に、スキーマの整合性を維持する方法について詳しく説明します。

スキーマ検査

スキーマ検査は、すべての新しいまたは変更したディレクトリエントリが、スキーマ規則にマッチしているかどうかを検査します。規則にマッチしていない場合、ディレクトリは変更要求を拒否します。

注 スキーマ検査では、適切な属性があるかどうかだけを検査します。属性値が当該属性について正しい構文で記述されているかどうかは検査しません。Directory Server には `nsslapd-valuecheck` という属性があり、これを使用することで値に DN 構文が含まれている属性を検索することができます。ただし、この属性はデフォルトではオフに設定されているため、属性値の検査は行われません。

デフォルトでは、スキーマ検査はデフォルトで有効になっています。クライアントの更新を受け付けるサーバーでは、スキーマ検査はオフにしないでください。スキーマ検査をオン、オフする方法については、『Directory Server 管理ガイド』の「スキーマ検査」を参照してください。

スキーマ検査を有効にする場合は、オブジェクトクラスで定義されている必須の属性と許可された属性に注意する必要があります。通常、オブジェクトクラス定義には少なくとも 1 つの必須の属性と 1 つ以上の省略可能な属性が含まれています。省略可能な属性とは、ディレクトリのエントリに追加できるが必須ではない属性のことです。エントリのオブジェクトクラス定義で必須でなく許可されてもいない属性をエントリに追加しようとすると、Directory Server はオブジェクトクラス違反メッセージを返します。

たとえば、エントリで organizationalPerson オブジェクトクラスを使用するように定義した場合は、commonName（cn）および surname（cn）属性がそのエントリの必須の属性になります。これらの属性にはエントリの作成時に値を指定する必要があります。さらに、オプションとしてエントリで使用できる属性の長いリストがあります。このリストには、telephoneNumber、Uid、streetAddress、userPassword などの説明属性が含まれています。

スキーマ検査を設定するときは、次の点に留意してください。

- 一般には、スキーマ違反を回避するために、スキーマに定義されている各エントリのすべての必須属性をレプリケートします。部分レプリケーションを使用して一部の必須属性をフィルタリングする場合は、スキーマ検査を無効にする必要があります。
- スキーマ検査で部分レプリケーションが有効になっていると、サーバーを ldif ファイルからオフラインで初期化できなくなる可能性があります。これは、必須属性をフィルタリングすると、サーバーが ldif ファイルを読み込めなくなるためです。
- スキーマ検査を無効にすると、パフォーマンスが向上する場合があります。
- 部分コンシューマレプリカでスキーマ検査を無効にした場合、その部分コンシューマレプリカが存在するサーバーインスタンス全体にスキーマが適用されなくなります。このため、同じサーバーインスタンスではサプライヤ (読み取り、書き込み) レプリカを設定しないようにする必要があります。
- 部分レプリケーションの設定ではサプライヤがスキーマをプッシュするため、部分コンシューマレプリカのスキーマは、マスターレプリカのスキーマのコピーとなります。このため、適用される部分レプリケーション設定には対応しません。

整合性のあるデータ形式の選択

LDAP スキーマを使用して、必要なデータを任意の属性値に格納できます。ただし、LDAP クライアントアプリケーションとディレクトリユーザーに適切な形式を選択して、ディレクトリツリー内で整合性を維持するようにデータを格納することが重要です。

LDAP プロトコルと Directory Server を使用する場合は、RFC 2252 で規定されているデータ形式でデータを表す必要があります。また、電話番号の正しい LDAP 形式は、次の ITU-T 勧告ドキュメントで定義されています。

- ITU-T 勧告 E.123。

  国内および国際電話番号に関する注記。

- ITU-T 勧告 E.163。

  国際電話サービスの番号計画。

たとえば、米国の電話番号は次のような形式になります。

```
+1 555 222 1717
```

postalAddress 属性には、区切り文字としてドル記号 ($) を使用する複数行文字列形式の属性値が必要です。適切に形式化されたディレクトリエントリは次のようになります。

```
postalAddress: 1206 Directory Drive$Pleasant View, MN$34200
```

レプリケートされたスキーマでの整合性の維持

レプリケート環境で整合性のあるスキーマを維持するには、次の点に留意します。

- コンシューマサーバーのスキーマを変更しない。

  コンシューマサーバーのスキーマを変更すると、マスターサーバーのスキーマよりも新しいスキーマとなります。マスターがレプリケーションで更新をコンシューマに送信すると、コンシューマのスキーマが新しいデータをサポートできないため、多数のレプリケーションエラーが発生します。

- マルチマスターレプリケーション環境では、1 つのマスターサーバーだけでスキーマを変更する。

  2 つのマスターサーバーのスキーマを変更すると、最後に更新されたマスターのスキーマがコンシューマに伝達されます。これは、コンシューマのスキーマと他方のマスターのスキーマとの間に整合性がないことを意味します。

注

Directory Server 5.2 は 11rfc2307.ldif スキーマファイルを使用します。このスキーマファイルは、rfc2307 に準拠しています。

Directory Server 5.2 より前のバージョンの Directory Server では、10rfc2307.ldif スキーマファイルを使用します。

異なるスキーマファイルを実行しているサーバー間でレプリケーションが有効な場合、レプリケーションは失敗します。

10rfc2307.ldif ファイルを使用している Directory Server インスタンスに、11rfc2307.ldif ファイルをコピーし、10rfc2307.ldif ファイルを削除します。

スキーマレプリケーションについては、156 ページの「スキーマのレプリケーション」を参照してください。

その他のスキーマ関連資料

標準 LDAPv3 スキーマについては、次のドキュメントを参照してください。

- Internet Engineering Task Force (IETF)
  http://www.ietf.org
- 『Understanding and Deploying LDAP Directory Services』
  (T. Howes、M. Smith、G. Good 著、Macmillan Technical Publishing 発行、1999 年)
- RFC 2252: LDAPv3 Attribute Syntax Definitions
  http://www.ietf.org/rfc/rfc2252.txt
- RFC 2256: Summary of the X.500 User Schema for Use with LDAPv3
  http://www.ietf.org/rfc/rfc2256.txt
- RFC 2251: Lightweight Directory Access Protocol (v3)
  http://www.ietf.org/rfc/rfc2251.txt

第 4 章

ディレクトリ情報ツリー

ディレクトリ情報ツリー (DIT) を使用することで、ディレクトリに格納されているデータを参照することができます。格納した情報のタイプ、企業の地理的な特性、ディレクトリで使用するアプリケーション、および使用するレプリケーションのタイプによって、ディレクトリツリーの設計方法が決まります。この章では、ディレクトリツリーの設計手順について概要を説明します。説明する内容は次のとおりです。

- ディレクトリツリーの概要
- ディレクトリツリーの設計
- ディレクトリエントリのグループ化と属性の管理
- その他のディレクトリツリー関連資料

ディレクトリツリーの概要

ディレクトリツリーを使用すると、クライアントアプリケーションからディレクトリデータに名前を付けたり、参照したりできるようになります。ディレクトリツリーは、ディレクトリデータの分散、レプリケート、アクセス制御の方法など、その他の設計判断と緊密に連携します。

適切に設計されたディレクトリツリーでは、次のことが可能になります。

- ディレクトリデータの管理を簡単にする
- レプリケーションポリシーとアクセス制御の作成における柔軟性
- ディレクトリを使用するアプリケーションをサポートする
- ユーザーが簡単にディレクトリを操作する

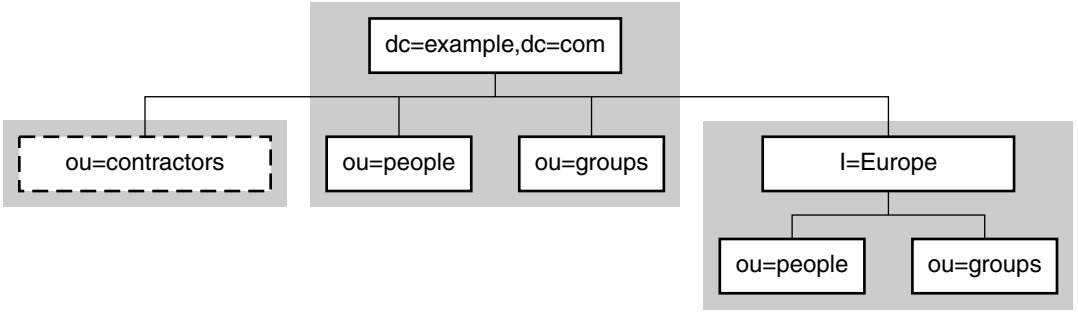
ディレクトリツリーの構造は、階層型の LDAP モデルに従います。ディレクトリツリーでは、たとえば、グループ、ユーザー、あるいは場所ごとにデータを編成できます。また、ディレクトリツリーによって複数のサーバー間でどのようにデータを分散して配置するかが決まります。データはサフィックスレベルのみで分割できるため、複数のサーバー間でデータを分散するには、適切なディレクトリツリー構造が必要です。

ディレクトリツリー設計は、レプリケーション設定にも影響します。ディレクトリツリーの一部だけをレプリケートしたい場合は、ディレクトリツリーの設計時にこれを考慮する必要があります。分岐点にアクセス制御を適用する場合も、設計時にそれを考慮する必要があります。

管理上、ディレクトリツリーは、サフィックス、サブサフィックス、および連鎖サフィックスによって定義されています。サフィックスとは分岐またはサブツリーであり、サフィックスの内容全体が管理タスクの単位として扱われます。たとえば、サフィックス全体に対してインデックスが定義され、サフィックス全体を 1 回の操作で初期化でき、また、サフィックスはレプリケーションの単位となります。同じ方法でアクセスおよび管理したいデータは、同じサフィックスに格納する必要があります。サフィックスはディレクトリツリーのルートに配置することもでき、その場合はルートサフィックスと呼ばれることがあります。

次の図に、2 種類のルートサフィックスを配置したディレクトリを示します。それぞれ、個別の企業エンティティに対応しています。

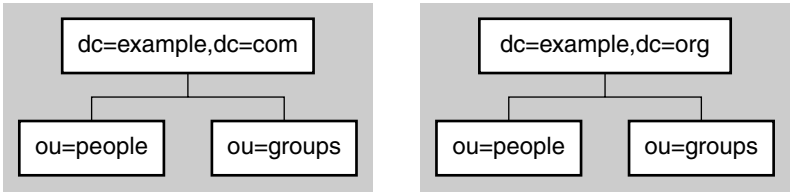
図 4-1 単一 Directory Server 内の 2 種類のルートサフィックス



サフィックスは、他のサフィックスの分岐となることもできます。その場合はサブサフィックスと呼ばれます。親サフィックスを管理する場合、サブサフィックスの内容は操作対象に含まれません。つまり、サブサフィックスは、親から独立して管理されます。ただし、LDAP 操作の結果には、サフィックスに関する情報は含まれません。また、ディレクトリクライアントは、エントリがルートサフィックスの一部であるか、サブサフィックスの一部であるかを認識できません。

次の図に、大規模な企業エンティティに対して、単一のルートサフィックスと複数のサブサフィックスを配置したディレクトリを示します。

図 4-2　複数のサブサフィックスを持つ単一のルートサフィックス



サフィックスは、サーバー内の個々のデータベースに対応します。ただし、データベースおよびそのファイルは、サーバーが内部的に管理するようになっています。そのため、Directory Server 5.2 ではデータベース用語は使用されていません。

連鎖サフィックスは、仮想ディレクトリツリーを作成して、別のサーバー上にあるサフィックスを参照します。連鎖サフィックスを使用すると、Directory Server は、リモートサフィックスに対する操作でも、ローカルで実行したかのように結果を返します。サフィックスが連鎖されていて、データがリモートサーバーから取得されることは、クライアントは認識できないため、データの位置が透過的になります。サーバー上のルートサフィックスが、別のサーバーに連鎖されているサブサフィックスを持つ場合もあります。その場合、クライアントからは、単独のツリー構造が作られているように見えます。

カスケード型連鎖の場合、連鎖サフィックスはリモートサーバー上などの別の連鎖サフィックスを参照することがあります。各サーバーは操作を転送し、最終的にクライアントの要求を処理するサーバーへ結果を返します。

連鎖の一般的な情報については、第 5 章「分散、連鎖、およびリフェラル」を参照してください。

ディレクトリツリーの設計

ディレクトリツリーの設計では、データを格納するサフィックスの選択、データエントリ間の階層関係の決定、ディレクトリツリー階層内のエントリのネーミングを行います。次に、設計の各段階について詳しく説明します。

- サフィックスの選択
- ディレクトリツリー構造の作成
- 識別名、属性、および構文
- エントリのネーミング

サフィックスの選択

サフィックスは、ディレクトリツリーのルートにあるエントリの名前です。サフィックスの下にディレクトリデータが格納されます。一般的なルートポイントを持たないディレクトリツリーが複数ある場合、複数のサフィックスを使用することもできます。

デフォルトの Directory Server のインストールでは、データの格納用に 1 つのサフィックスが、設定情報やディレクトリのスキーマなど、ディレクトリの内部処理に必要なデータ用に複数のサフィックスが使用されます。デフォルトのディレクトリサフィックスについては、『Directory Server 管理ガイド』の「ディレクトリツリーの作成」を参照してください。

サフィックスの命名規則

ディレクトリ内のすべてのエントリは、共通のベースエントリ (サフィックス) の下に格納する必要があります。それぞれのサフィックス名は、次のように指定する必要があります。

- グローバルに一意の名前にする
- 変更しない、あるいはまれにしか変更しないようにする
- そのサフィックスの下にあるエントリが画面上で読みやすいように短い名前にする
- ユーザーが容易に入力および記憶できるものにする

1 つの企業環境では、DNS 名またはインターネットドメイン名に合わせてサフィックスを選択します。たとえば、企業が `Example.com` のようなドメイン名を所有している場合、サフィックスは次のようになります。

```
dc=example,dc=com
```

`dc` (`domainComponent`) 属性は、ドメイン名をコンポーネントに分割してサフィックスを表します。

サフィックスの名前を付けるときには、任意の属性を使用できます。ただし、ホスト環境では、サフィックスに使用する属性は次のものに限定する必要があります。

- `c (countryName)`

  ISO の定義に準拠した、国名を表す 2 桁のコードを含めます。

- `l (localityName)`

  エントリが存在する、あるいはエントリに関連付けられた国や都市などの地域を示します。

- `st (stateOrProvinceName)`

  エントリがある州または県を示します。

- `o (organizationName)`

  エントリが属する組織の名前を示します。

これらの属性がサフィックスに含まれていると、加入者のアプリケーションとの相互運用性が高まります。たとえば、ホスティングサービス事業者がこれらの属性を使用して、クライアントの Example.com に、次のようなルートサフィックスを作成する場合を考えてみます。

```
o=Example.com,st=Washington,c=US
```

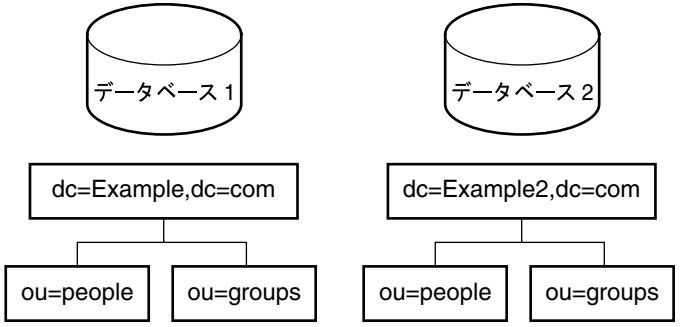
これらの属性については、68 ページの表 4-1 を参照してください。

組織名のあとに国名コードを使用する指定方法は、X.500 のサフィックスの命名規則に従っています。

複数のサフィックスの使用

定義する各サフィックスは、それぞれ固有のディレクトリツリーを構成します。複数のディレクトリツリーを作成し、別々のデータベースに格納することができます。たとえば、図 4-3 に示すように Example.com と Example2.com 用に別々のサフィックスを作成し、それぞれを異なるデータベースに格納できます。

図 4-3　2 つの異なるデータベースに格納される 2 つのサフィックス



データベースは、リソースの制限に応じて 1 つのサーバーまたは複数のサーバーに格納できます。

ディレクトリツリー構造の作成

ディレクトリツリーの構造は、フラットの場合も階層型の場合もあります。一般には、ディレクトリツリーはできるだけフラットにします。ただし、複数のデータベース間にデータを分散したり、レプリケートできるようにしたり、アクセス制御を設定したりする場合は、ある程度階層化することも必要です。

ディレクトリツリー構造の設計に関しては、次の項目に記載しています。

- ディレクトリの分岐点の作成
- 分岐点属性の特定
- レプリケーションに関する検討事項
- アクセス制御に関する検討事項

ディレクトリの分岐点の作成

分岐点は、ディレクトリツリー内の新しい分割を定義する場所です。分岐点を決定するときは、問題が生じる可能性のある名前の変更は避けてください。名前を変更する確率は、名前が変更される可能性があるコンポーネントが、ネームスペース内に多く含まれているほど高くなります。ディレクトリツリーの階層が深いほどネームスペース内のコンポーネントは多くなり、名前を変更する確率が高くなります。

分岐点を定義する際には、次のガイドラインが役に立ちます。

- 企業組織内でもっとも大きい部門区分のみを表すようにツリーを分岐させます。このような分岐点は、部門 (企業情報サービス、カスタマサポート、販売サービス、専門サービスなど) だけを表します。分岐させる部門は安定したものにします。組織変更が頻繁に行われる場合は、この種の分岐は使用しないようにします。

- 組織の実際の名前ではなく、機能を表す名前または一般的な名前を使用します。組織名は変更されることが考えられます。会社が部門の名前を変更するたびにディレクトリツリーを変更する必要に迫られるのでは困ります。代わりに、組織の機能を表す一般的な名前を使用します。たとえば、「Widget 研究開発」ではなく「エンジニアリング」を使用します。
- 似たような機能を持つ組織が複数ある場合は、部門の構成に基づいて分岐点を作成するのではなく、その機能を表す分岐点を 1 つ作成します。たとえば、特定の製品ラインを担当する複数のマーケティング部門がある場合でも、1 つのマーケティング サブツリーのみを作成します。すべてのマーケティングエントリは、そのツリーに所属させます。

次に、企業とホスト環境のディレクトリツリー構造の例を示します。

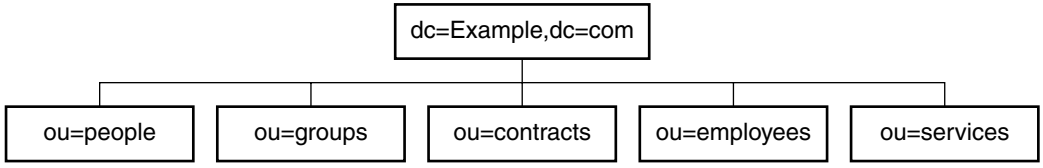
企業環境での分岐点の作成

変更される可能性の低い情報に基づいてディレクトリ構造を決定すれば、名前の変更を避けられます。たとえば、組織ではなくツリー内のオブジェクトのタイプに基づいて構造を定義します。次に、ツリー構造の定義に使用するオブジェクトの例を示します。

- `ou=people`
- `ou=groups`
- `ou=contracts`
- `ou=employees`
- `ou=services`

図 4-4 は、Example.com という企業を例として、これらのオブジェクトを使用したディレクトリツリーの構造を示しています。

図 4-4　5 つの分岐点を持つディレクトリ情報ツリーの例



よく使用されている従来型の分岐点属性だけを使用してください (表 4-1 を参照)。従来からある属性を使用すると、サードパーティの LDAP クライアントアプリケーションとの互換性が保たれる可能性が高くなります。また、従来からある属性は、デフォルトのディレクトリスキーマで認識可能なので、分岐 DN のエントリを作成しやすくなります。

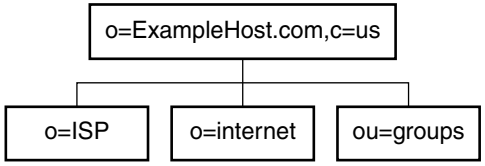
表 4-1 従来からある DN 分岐点の属性

属性名	定義
c	国名。
o	組織名。通常、この属性は、企業の部門、教育機関での学部 (人文学部、理学部など)、子会社、企業内の主要部門など、大きな部門を表すために使用します。また、64 ページの「サフィックスの命名規則」で説明したように、ドメイン名を表す場合もこの属性を使用する必要があります。
ou	組織の構成単位。通常この属性は、組織よりも小さな組織内の部門を表すために使用します。組織単位は、一般にすぐ上の組織に属します。
st	州または県の名前。
l	地域 (都市、地方、オフィス、施設名など)。
dc	64 ページの「サフィックスの命名規則」で説明されているドメインのコンポーネント。

ホスト環境での分岐点の作成

ホスト環境では、organization(o) オブジェクトクラスの 2 つのエントリと organizationalUnit(ou) オブジェクトクラスの 1 つのエントリをルートサフィックスの下に含むツリーを作成します。インターネットホスト ExampleHost.com のディレクトリツリーを、例として図 4-5 に示します。

図 4-5 ISP ExampleHost.com のディレクトリ情報ツリー



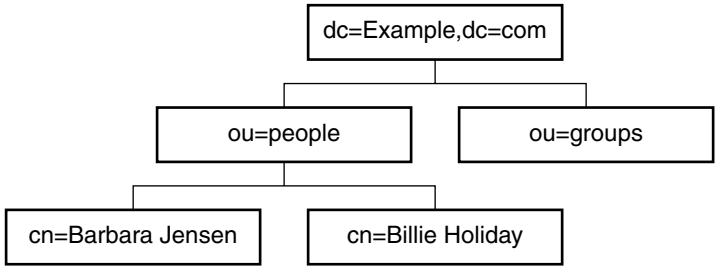
分岐点属性の特定

ディレクトリツリーを設計するときは、分岐点の特定に使用する属性を決定する必要があります。DN は、属性とデータのペアで構成される一意の文字列です。たとえば、Example.com 社の社員 Barbara Jensen 用のエントリの DN は次のようになります。

```
cn=Barbara Jensen,ou=people,dc=example,dc=com
```

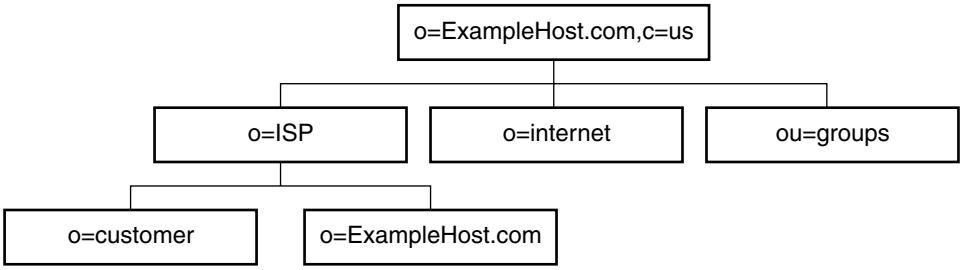
各属性とデータのペアは、ディレクトリツリーの分岐点を表します。Example.com 社のディレクトリツリーを、図 4-6 に示します。

図 4-6 Example.com 社のディレクトリ情報ツリー



ExampleHost.com のディレクトリツリーを、図 4-7 に示します。

図 4-7 ExampleHost.com 社のインターネットホストディレクトリ情報ツリー



ルートサフィックスのエントリ `o=ExampleHost.com,c=us` の下で、ツリーは 3 つに分岐しています。ISP の分岐には、顧客データと ExampleHost.com の社内情報が含まれています。internet の分岐は、ドメインのツリーです。groups の分岐には、管理グループに関する情報が含まれています。

分岐点の属性を選択するときは、一貫性を持たせることが重要です。ディレクトリツリー全体で識別名 (DN) の形式が統一されていないと、一部の LDAP アプリケーションで混乱が生じる可能性があります。たとえば、ディレクトリツリーのある部分で `l` (`localityName`) が `o` (`organizationName`) の下位にある場合、ディレクトリのほかの部分でも `l` が `o` の下位にあることを確認する必要があります。

よくある間違いに、識別名で使用されている属性に基づいてディレクトリを検索してしまうことがあります。識別名はディレクトリエントリをほかと識別するだけのもので、これを検索対象にすることはできません。ただし、エントリ自体に格納された属性とデータのペアに基づいてエントリを検索することは可能です。

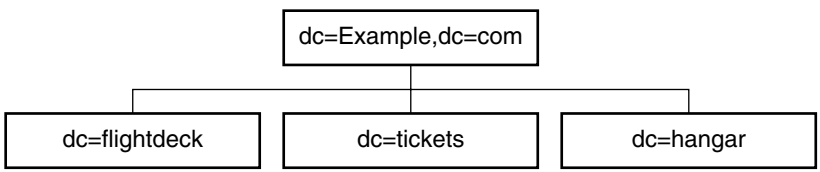
レプリケーションに関する検討事項

ディレクトリツリーを設計するときは、レプリケートするエントリについて検討してください。エントリセットをレプリケートする場合は、サブツリーの頂点で識別名 (DN) を指定し、その下にあるエントリをすべてレプリケートするのが自然な方法です。また、このサブツリーは、ディレクトリデータの一部を含むディレクトリパーティションである、データベースに対応します。

企業環境では自社内のネットワーク名に対応させてディレクトリツリーを編成できます。ネットワーク名が変更されることはほとんどないので、ディレクトリツリー構造は安定したものになります。また、レプリケーションを使用して別の Directory Server を連動させる場合は、ネットワーク名を使用してディレクトリツリーの最上位の分岐点を作成する方法が有効です。

たとえば、Example.com 社に `flightdeck.Example.com`、`tickets.Example.com`、`hanger.Example.com` という 3 つのプライマリネットワークがあるとします。図 4-8 は、このディレクトリツリーの最初の分岐を示しています。

図 4-8 Example.com 社 DIT の 3 つの主要ネットワーク



ツリーの最初の構造を作成したあと、ネットワークをさらに図 4-9 のように分岐させています。

図 4-9　Example.com 社 DIT の 3 つの主要ネットワークの詳細

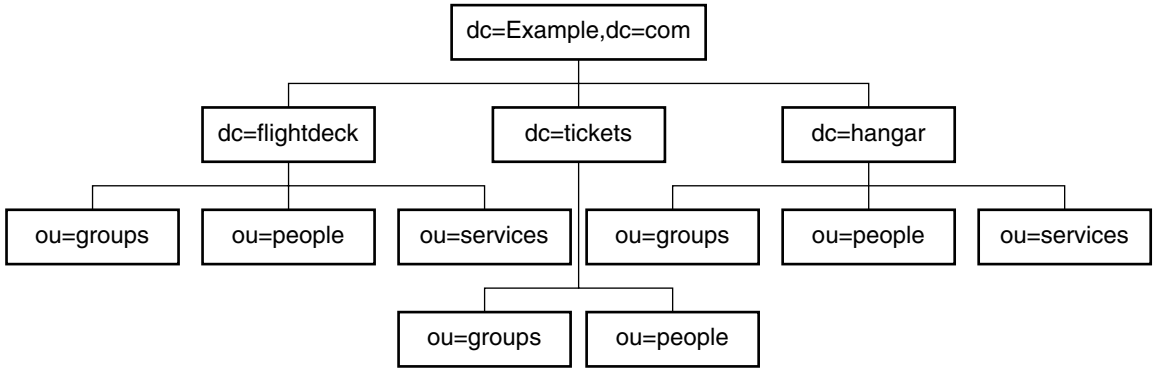
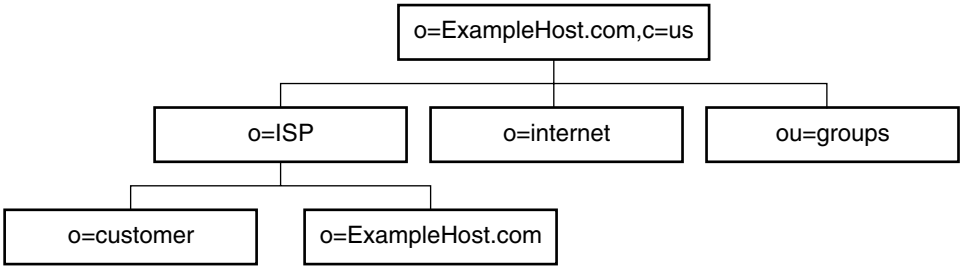


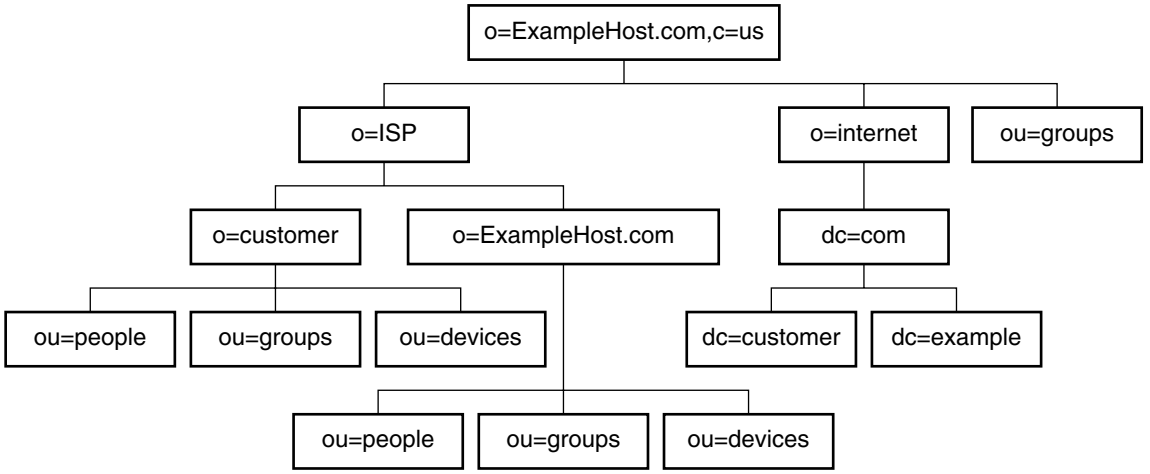
図 4-10 は、インターネットホスティング会社である ExampleHost.com のディレクトリ分岐の例を示しています。

図 4-10　ExampleHost.com のディレクトリ情報ツリー



ディレクトリツリーの最初の構造を作成したあと、ネットワークをさらに図 4-11 のように分岐させています。

図 4-11　ExampleHost.com 社の詳細な DIT



どちらの企業も、あまり変更されることのない情報に基づいてデータ階層を設計しています。

アクセス制御に関する検討事項

ディレクトリツリーを階層化すると、特定のタイプのアクセス制御を使用できるようになります。レプリケーションの場合と同様に、類似したエントリをグループ化すると、それらのエントリを 1 つの分岐点から簡単に管理できます。

また、ディレクトリツリー階層で管理を分散させることもできます。たとえば、営業部の管理者に営業のエントリへのアクセス権を与え、マーケティング部の管理者にマーケティングのエントリへのアクセス権を与える場合は、ディレクトリツリーの設計を通じてアクセス権を付与することができます。

ディレクトリツリーではなくディレクトリの内容に基づいてアクセス制御を設定することもできます。ACI フィルタを適用するターゲットメカニズムを使用すると、ディレクトリエントリが特定の属性値を含むすべてのエントリへのアクセスを持つように規定した、1 つのアクセス制御規則を定義できます。たとえば、`ou=Sales` という属性を含むすべてのエントリへのアクセス権を営業部の管理者に与える ACI フィルタを設定できます。

ただし、ACI フィルタは管理が簡単ではありません。ディレクトリツリー階層で組織に対応した分岐を作成する、ACI フィルタを適用する、あるいは両者を組み合わせるなどして、ディレクトリにもっとも適したアクセス制御方法を決める必要があります。ACI をより簡単に管理するため、Directory Server 5.2 では、エントリの実効権限を取

得できるようにします。これは、ディレクトリのエントリと属性に対するユーザーのアクセス制御権です。実効権限機能により、ユーザー管理、アクセス制御ポリシーの検証、およびデバッグが簡単になります。詳細は、184 ページの「実効権限に関する情報の取得」を参照してください。

識別名、属性、および構文

ここでは、識別名、属性、および構文の情報を簡単に説明します。

識別名

識別名 (DN) は、LDAP ディレクトリにあるエントリの名前と場所を表す文字列です。DN はディレクトリのエントリへのパスを記述します。社内の各ユーザーとグループは、Directory Server では DN で表されています。ディレクトリにあるユーザーとグループの情報を変更するときは、常に識別名を使用します。

DN は、相対識別名 (RDN) と呼ばれる多数のコンポーネントで構成されます。それぞれの RDN は、ディレクトリ内の特定のエントリを識別します。各ディレクトリエントリを確実に一意なものにするには、LDAP により、1 つの親エントリはその下に 2 つの同じ RDN を持つことはできないと指示します。

通常、ユーザーまたはグループの DN には、少なくとも 3 つのタイプの RDN が含まれています。

- ユーザー名、ユーザー ID、またはグループ名 (cn または uid キーワードで識別される)
- 組織名 (o キーワードで識別される)
- 1 つまたは複数のドメイン名のコンポーネント (dc キーワードで識別される)。たとえば、example.com には、example と com という 2 つのドメイン名のコンポーネントが含まれる。

その他の一般的な RDN は、組織の構成単位 (ou)、州 (st)、国 (c) などです。

DN の構成は、ディレクトリの構造によって異なります。ほとんどのディレクトリは、国名コードと組織名以外のほかのカテゴリも使用して構成されています。このため、エントリの識別に使われる DN は長くなり、より多くの特定の属性とデータのペアが格納されます。たとえば、同じ会社にいる 3 人の従業員またはユーザーの DN は、次のようになります。

```
cn=Ben Hurst, ou=Operations, o=Example Corp, st=CA, c=US

cn=Jeff Lee, ou=Marketing, o=Example Corp, st=CA, c=US

cn=Mary Smith, ou=Sales, o=Example Corp, st=MN, c=US
```

この例では、3 人のユーザーすべてが別々の部門または組織の構成単位 (ou) で、同じ会社または組織 (o) である Example Corp に勤務しています。3 番目のユーザーは、最初の 2 人のユーザーとは別の州 (st) で勤務しています。

LDAP により、組織と組織の構成単位がその他の組織や組織の構成単位を格納でき、複雑な企業を表すことができます。たとえば、大きな会社の中のグループの DN は次のようになります。

```
cn=Technical Publications, ou=Super Server Group, ou=Server
Division, o=Example Corporation, o=MegaCorp, dc=megacorp, dc=com
```

表 4-2 に、一般的な RDN キーワードのリストを示します。

表 4-2 DN で使用される一般的な RDN キーワード

RDN キーワード	DN での意味	内容
c	国	ユーザーまたはグループが在住する国です。 例: c=US c=GB
cn	共通名またはフルネーム	エントリによって定義される人またはオブジェクトのフルネームです。 例: cn=Wally Henderson cn=Database Administrators cn=printer 3b
dc	ドメインのコンポーネント	DNS ドメインの一部。このキーワードは、通常はディレクトリツリーの最上位で使用されます。 たとえば、example.com ドメインのユーザーは、次のような DN を持つことがあります。 cn=Barbara Jones,ou=Engineering, dc=example, dc=com

表 4-2　DN で使用される一般的な RDN キーワード (続き)

RDN キーワード	DN での意味	内容
l	地域	ユーザーまたはグループが在住する地域です。都市、地方、町、またはその他の地理的な地域を表します。 例 : l=Tucson l=Pacific Northwest l=Anoka County
o	組織	ユーザーまたはグループが所属する組織です。 例 : o=Sun Java System Software o=Public Power & Gas
ou	組織単位	組織内の単位です。 例 : ou=Sales ou=Manufacturing
sn	surname	ユーザーの姓。 例 : sn=Henderson
st	州または県	ユーザーまたはグループが在住する州または県を表します。 例 : st=Iowa st=British Columbia

属性

ディレクトリ属性は、エントリに関する記述的な情報を保持します。たとえば、ユーザーエントリは、ユーザー ID、電子メールアドレス、ファーストネーム、およびパスワードの属性を保持することがあります。

表 4-3 に、一般的なユーザーおよびグループのディレクトリ属性のリストを示します。

表 4-3　一般的なユーザーおよびグループのディレクトリ属性

属性キーワード	属性名	内容
`givenName`	given name	ユーザーの名 (ファーストネーム) です。
`mail`	email address	ユーザーまたはグループの電子メールアドレスです。
`streetAddress`	street	エントリで定義されているユーザーまたはグループの住所です。例 : `street=12 Main Street`
`telephoneNumber`	telephone	ユーザーまたはグループの電話番号です。例 : `(800) 555-9SUN`

`title`	title	ユーザーの役職です。例 :
		`title=writer`
		`title=manager`

`uid`	user ID	エントリによって定義された人またはオブジェクトを一意に識別する名前です。
`userPassword`	パスワード	ユーザーのパスワードです。

ユーザーエントリには、上記のリストのほかにも多数の属性を含めることができます。また、自社のニーズに合うように新しい属性を作成することもできます。属性については、第 3 章「Directory Server のスキーマ」を参照してください。

DN および属性のガイドラインと構文

ディレクトリのエントリを作成、選択、および使用するときは、次のガイドラインに従ってください。

RDN 値ではコンマをエスケープするか、引用符で囲みます。RDN 値にコンマが含まれる場合は、コンマを使用する名前の一部を二重引用符で囲むか、円記号を付けてエスケープします。たとえば、DN に文字列 `Ace Industry, Corp` を含めるには、次の形式を使用します。

```
o="Ace Industry, Corp", c=US
```

次の形式でも、同じ結果になります。

```
o=Ace Industry¥, Corp, c=US
```

スキーマ検査が有効になっている場合は、属性がディレクトリスキーマと一致する必要があります。スキーマ検査が有効になっている場合は、Directory Server で認識でき、エントリのオブジェクトクラスに許可されている RDN キーワードと属性を使用します。スキーマ検査が無効になっている場合は、エントリのオブジェクトクラスとは無関係に、すべての属性を使用できます。スキーマ検査については、56 ページの「スキーマ検査」を参照してください。

RDN を同じシーケンスまたはパスに指定します。DN は、ディレクトリツリーからのパスを表します。RDN キーワードが適切な順序で指定されていないと、Directory Server でエントリを検出できない場合があります。次に例を示します。

```
cn=Ralph Swenson, ou=Accounting, o=Example Corp, c=US
```

これは、次のものとは異なります。

```
cn=Ralph Swenson, o=Example Corp, ou=Accounting, c=US
```

これは、組織の構成単位 (ou) と組織 (o) のキーワードが、同じ順序で指定されていないためです。

ユーザー ID は一意にする必要があります。`ldapmodify` コマンドを使用してユーザーを作成するときは十分に注意してください。ディレクトリで属性の一意性プラグインがユーザー ID 属性に対して有効になっていない限り、このユーティリティは重複したユーザー ID をチェックしないためです。詳細は、『Directory Server 管理ガイド』の「属性値の一意性の適用」を参照してください。

エントリのネーミング

ディレクトリツリー構造を設計したあとは、ツリー構造内のエントリに名前を付けるときに使用する属性を決定する必要があります。一般に、名前は 1 つ以上の属性値を選び、相対識別名 (RDN) を形成して作成します。名前を付けるエントリのタイプによって使用する属性が変わります。

エントリの名前は、次の規則に従って付ける必要があります。

- 変更される可能性の低い属性を選んで名前を付ける。
- 名前はディレクトリ全体で一意でなければならない。名前を一意にすると、DN によってディレクトリ内の複数のエントリが参照されることがなくなる。

エントリを作成するときは、エントリ内で RDN を定義します。エントリ内で少なくとも RDN を定義しておけば、簡単にエントリを検出できます。これは、検索が実際の DN を対象にして実行されるのではなく、エントリ自体に格納されている属性値を対象に実行されるからです。

属性名には意味があるので、属性が表すエントリのタイプに合った属性名を使用するようにします。たとえば、組織を表すために l (locality) を使用したり、組織の構成単位を表すために c (country) を使用したりしないでください。

エントリに名前を付けるときの手法について、次の項目ごとに説明します。

- 人のエントリのネーミング
- 組織エントリのネーミング
- その他のエントリのネーミング

人のエントリのネーミング

人のエントリの名前 (DN) は一意である必要があります。従来、識別名ではその人のエントリに名前を付けるときに、commonName または cn 属性を使用しています。つまり、Babs Jensen という名前のユーザーのエントリには、次のような識別名が付けられます。

```
cn=Babs Jensen,dc=example,dc=com
```

このネーミング方法でエントリと関連付けられているユーザーを識別するのは簡単ですが、そのエントリは同じ名前のユーザーが組織に 2 人いる場合は一意にならない場合があります。この場合、DN の名前の衝突として知られている、複数のエントリが同じ識別名を持つ問題が生じます。

共通名の衝突は、一意の識別子を共通名に追加することで避けられます。たとえば、次のようにします。

```
cn=Babs Jensen+employeeNumber=23,dc=example,dc=com
```

ただし、このネーミング方法では、管理が難しくなり、大きなディレクトリの場合は扱いにくい共通名となります。

より良い方法は、cn 以外の属性で人のエントリを特定することです。次のいずれかの属性を使用することを検討してください。

- uid

  uid (userID) 属性を使用して、個人に固有な値を指定します。たとえば、ユーザーのログイン ID や社員番号などが使用できます。ホスト環境の加入者は、uid 属性で識別する必要があります。

- mail

  mail 属性を使用して、個人の電子メールアドレスの値を追加します。この方法でも、重複した属性値を含む扱いにくい DN になる場合があるので (たとえば mail=bjensen@Example.com, dc=example,dc=com)、uid 属性で使用可能な一意の値がなかった場合にのみ、この方法を使用します。たとえば、会社が社員番号やユーザー ID を臨時社員や契約社員に割り当てない場合は、uid 属性の代わりに mail 属性を使用します。

- employeeNumber

  inetOrgPerson オブジェクトクラスの社員には、employeeNumber などの会社側が割り当てた属性値を使用することを検討します。

人のエントリの RDN の属性とデータのペアにどのような属性値を使用する場合でも、一意で永続的な値を使用する必要があります。

ユーザーがサービスの加入者の場合、そのエントリは inetUser オブジェクトクラスにし、uid 属性を含める必要があります。属性は顧客のサブツリーで一意である必要があります。ユーザーがホスティングサービス事業者に属している場合は、nsManagedPerson オブジェクトクラスの inetOrgPerson として表します。

ディレクトリツリーに人のエントリを配置するときは、次の点を考慮してください。

- 企業内のユーザーのエントリは、組織のエントリの下にあるディレクトリツリーに配置する必要がある。
- ホスティングサービス事業者の加入者は、ホストされる組織の ou=people 分岐の下に配置する必要がある。

組織エントリのネーミング

組織エントリ名は、ほかのエントリと同様に一意である必要があります。ほかの属性値と組織の法的な名前を組み合わせて使用すると、名前は確実に一意になります。たとえば、次のように組織エントリに名前を付けます。

```
o=Example.com+st=Washington,o=ISP,c=US
```

登録商標を使用することもできますが、一意である保証はありません。

ホスト環境では、組織エントリに次の属性を組み込みます。

- o (organizationName)
- top、organization、および nsManagedDomain の値を持つ objectClass

その他のエントリのネーミング

地域、州、国、デバイス、サーバー、ネットワーク情報、その他のタイプのデータなど、ディレクトリには多くのものを表すエントリが含まれています。

これらのタイプのエントリには、可能な場合は RDN 内で commonName (cn) 属性を使用してください。たとえば、グループエントリに名前を付ける場合は、次のように名前を付けます。

```
cn=allAdministrators,dc=example,dc=com
```

commonName 属性がサポートされていないオブジェクトクラスを持つエントリに名前を付けなければならないこともあります。この場合は、エントリのオブジェクトクラスでサポートされている属性を使用します。

エントリの DN で使用される属性とエントリ内で実際に使用されている属性が対応している必要はありません。ただし、指定する属性を DN で見えるようにすると、ディレクトリツリーを簡単に管理できます。

ディレクトリエントリのグループ化と属性の管理

ディレクトリツリーは、エントリの情報を階層構造で構成します。階層もグループ化メカニズムの 1 つですが、分散しているエントリの関連付け、頻繁に変更される組織、または多数のエントリで繰り返されるデータには適していません。Directory Server は、追加のグループ化メカニズムとしてグループとロールの 2 つを提供しています。こちらのほうがエントリ間の関係を柔軟に定義できます。

これらのグループ化メカニズムのほかに、Directory Server ではサービスクラス (CoS) というメカニズムを利用できます。これは、アプリケーションの介入なしでエントリ間で属性を共有できるように、属性を管理するメカニズムです。ロールメカニズムと同様に、エントリが検出されると、CoS がそのエントリに対して仮想属性を生成します。ただし、CoS はメンバーを定義するのではなく、一貫性の保持と容量の節約のために、関連するエントリがデータを共有できるようにします。

次に、エントリのグループ化と属性管理のメカニズム、およびそれぞれの利点と制限について説明します。

- スタティックグループとダイナミックグループ
- 管理されているロール、フィルタを適用したロール、入れ子のロール
- ロールの列挙とロールメンバーシップの列挙
- ロールの範囲
- ロールの制限事項
- グループとロールのどちらを使用するかの決定
- サービスクラス (CoS) による属性の管理
- CoS について
- CoS 定義エントリと CoS テンプレートエントリ
- CoS の優先順位
- ポインタ CoS、間接 CoS、クラシック CoS
- CoS の制限事項

スタティックグループとダイナミックグループ

グループとは、そのメンバーであるほかのエントリを特定するエントリです。グループに含めることが可能なメンバーの範囲は、グループ定義エントリの位置に関係なく、ディレクトリ全体となります。グループ名がわかっている場合は、そのすべてのメンバーエントリを簡単に検索できます。次に、スタティックグループとダイナミックグループの特徴と、それぞれのグループをいつ使用するべきかを説明します。

- スタティックグループは、そのメンバーエントリの名前を明示的に指定します。スタティックグループを定義するエントリは、`groupOfNames` または `groupOfUniqueNames` オブジェクトクラスを使用し、各メンバーの DN をそれぞれ `member` または `uniqueMember` の属性値として含みます。`member` 属性には、サーバーがグループのメンバーシップを確立するときにチェックする DN が含まれます。`uniqueMember` 属性には、オプションとしてハッシュ (#) と一意の識別子のラベルが続く DN が含まれます。メンバーシップのチェックは、この DN に対して行われます。
- スタティックグループは、ディレクトリ管理者のグループのようにメンバーの少ないグループに適しており、非常に大きいグループには適していません。パフォーマンスが著しく低下するため、メンバー数が 20,000 を超えるスタティックグループを作成することは避けてください。これ以上のサイズのグループには、ダイナミックグループまたはロールを使用してください。20,000 を超えるメンバーに対してスタティックグループを使用する必要がある場合は、1 つの大きなスタティックグループを使用するのではなく、グループのグループ化を使います。
- ダイナミックグループはフィルタを指定し、そのフィルタと一致するすべてのエントリがグループのメンバーとなります。このようなグループは、フィルタが評価されるたびにメンバーが定義されるため、ダイナミックグループと呼ばれます。ダイナミックグループの定義エントリは、`groupOfUniqueNames` および `groupOfURLs` オブジェクトクラスに属します。グループメンバーシップは、`memberURL` 属性の LDAP URL 値として表される 1 つまたは複数のフィルタ、または `uniqueMember` 属性の値として表される 1 つまたは複数の DN としてリストされます。

注 ダイナミックグループの `uniqueMember` 属性として別のグループの DN を使用すると、グループをほかのグループ内に配置することができます。これらのグループを、入れ子のグループと呼びます。

どちらのグループも、ディレクトリ内のどこにいるメンバーでも識別できますが、グループ定義は、`ou=Groups` などの適切な名前のノードに格納する必要があります。たとえば、このように格納することで、バインドの証明情報がグループのメンバーである場合に、アクセスを許可または制限するアクセス制御命令 (ACI) を定義するときに、検索が簡単になります。

管理されているロール、フィルタを適用したロール、入れ子のロール

ロールはエントリのグループ化メカニズムです。エントリがディレクトリから取得されるとすぐに、ロールのメンバーシップを決定できます。これにより、グループ化メカニズムの主な欠点が解消されます。各ロールはメンバー (そのロールを所有するエントリ) を持ちます。ロールに属しているすべてのエントリに nsRole 仮想属性が指定されます。この属性の値は、そのエントリがメンバーになっているすべてのロールの DN です。グループと同じようにロールのメンバーを明示的またはダイナミックに指定できます。

ディレクトリがロールのメンバーを自動的に算出するため、ロールメカニズムはクライアントから簡単に使用できます。nsRole は、実行中にサーバーによって生成され、実際にはディレクトリに格納されないため、仮想属性と呼ばれています。つまり、ロールを使って処理を実行すると、グループを使う場合よりもサーバー側でより多くのリソースが消費されます。これは、クライアントアプリケーションのためにサーバーがその処理を実行するためです。ただし、ロールのメンバーの検査方法は一貫しており、サーバー側で透過的に実行されます。

Directory Server は、次の 3 種類のロールをサポートしています。

- 管理されているロール : 明示的にメンバーエントリにロールを割り当てる。
- フィルタを適用したロール : 指定した LDAP フィルタと一致するエントリを割り当てる。このため、ロールは各エントリに含まれている属性によって異なる。
- 入れ子のロール : ほかのロールを含むロールを作成できる。

管理されているロール

管理されているロールは、メンバーがロール定義エントリではなく各メンバーエントリ内で定義されていることを除いて、スタティックグループと似ています。管理されているロールを使用すると、管理者は、対象となるエントリに nsRoleDN 属性を追加することにより、ロールを割り当てることができます。この属性の値は、ロール定義エントリの DN です。スタティックロールの定義エントリは、対象の範囲だけを定義します。そのロールのメンバーは指定範囲内で、nsRoleDN 属性がロール定義エントリの DN を指定しているエントリです。

フィルタを適用したロール

フィルタを適用したロールはダイナミックグループと似ており、ロールのメンバーを決定するフィルタを定義します。nsRoleFilter 属性の値は、フィルタを適用したロールを定義します。サーバーが、フィルタ文字列と一致する、フィルタを適用したロールの適用範囲内のエントリを返す場合、そのエントリにはロールを識別する nsRole 属性が常に含まれています。

入れ子のロール

入れ子のロールを使用して、別のロールを含むロールを作成できます。入れ子のロールはほかのロールの定義エントリをリストし、それらのロールのすべてのメンバーを組み合わせます。あるエントリが、入れ子のロールのリストに含まれているロールのメンバーである場合、そのエントリは入れ子のロールのメンバーでもあることになります。

ロールの列挙とロールメンバーシップの列挙

ロールの列挙

nsRole 属性はほかの属性と同様に読み取られます。クライアントはこの属性を使用して、任意のエントリが属しているすべてのロールを列挙することができます。ロールメカニズムで使用されるのは、nsRole 属性だけで、この属性はすべての変更操作から保護されています。ただし、読み取りは可能なので、ロールメンバーシップを公開したくない場合は、アクセス制御を定義して読み取りから保護する必要があります。

ロールメンバーシップの列挙

仮想属性に対する検索が可能なため、nsRole 属性を検索し、ロールのメンバーを列挙することができます。ただし、検索操作でインデックスが付けられていない属性は、パフォーマンスに大きな影響を与える可能性があるため注意が必要です。

等価フィルタに基づく検索の多くにはインデックスが付けられるため効率的ですが、否定検索にはインデックスが付けられず、パフォーマンスの低下を招くことになります。nsRoleDN 属性にはデフォルトでインデックスが付けられるので、管理されているロールに対する検索は比較的効率的です。フィルタを適用したロールと入れ子のロールでは、インデックスが付けられた属性と付けられていない属性の両方がフィルタに含まれている可能性があるため、インデックスが付けられていない検索を行わないように、少なくとも 1 つのインデックス付き属性がフィルタに含まれていることを確認する必要があります。

ロールの範囲

Directory Server では、ロールの範囲をロール定義エントリのサブツリーを超えて拡張するための属性を使用できます。これは、nsRoleScopeDN という 1 つの値からなる属性で、既存のロールに追加する範囲の DN を含みます。nsRoleScopeDN 属性を追加できるのは、入れ子のロールだけです。

nsRoleScopeDN 属性により、あるサブツリーのロールの範囲を拡張して、別のサブツリーにエントリを含めることができます。たとえば、Example.com のディレクトリに、o=eng,dc=example,dc=com (エンジニアリングサブツリー) と o=sales,dc=example,dc=com (販売サブツリー) という 2 つのサブツリーがあると仮定します。エンジニアリングサブツリーのユーザーには、販売サブツリーのロールで管理される販売アプリケーション (SalesAppManagedRole) に対するアクセス権が必要です。ロールの範囲を拡張するには、次のようにします。

1. エンジニアリングサブツリーに、たとえば EngineerManagedRole のようにユーザーのロールを作成します。この例では、管理されているロールを使用していますが、フィルタを適用したロールでも、入れ子のロールでもかまいません。
2. 販売サブツリーに、たとえば SalesAppPlusEngNestedRole のような入れ子のロールを作成し、新たに作成した EngineerManagedRole と、当初からの SalesAppManagedRole を格納します。
3. SalesAppPlusEngNestedRole に nsRoleScopeDN 属性を追加します。属性値には、追加するエンジニアリングサブツリーの範囲の DN を指定します。この例では、o=eng,dc=example,dc=com を指定します。

エンジニアリングユーザーには、SalesAppPlusEngNestedRole ロールにアクセスして販売アプリケーションにアクセスできるよう、適切なアクセス権を与える必要があります。さらに、ロールの範囲全体をレプリケートする必要があります。

注 範囲の拡張は入れ子のロールに限定されているため、これまで 1 つのドメインのロールを管理していた管理者は、その他のドメインにすでに存在するロールを使用する権限だけを持つことになり、その他のドメインで任意のロールを作成することはできなくなります。

ロールの制限事項

ディレクトリサービスをサポートするロールを作成する場合は、次の制限事項を考慮する必要があります。

- ロールと連鎖

連鎖機能を使用してディレクトリツリーを複数のサーバーに分散している場合は、ロールを定義するエントリをそれらのロールを所有するエントリと同じサーバーに配置する必要があります。連鎖を介して、サーバー A が別のサーバー B からエントリを受け取る場合は、それらのエントリにはサーバー B で定義されたロールが含まれますが、サーバー A で定義されたロールは割り当てられません。

- フィルタが適用されたロールは CoS によって生成される属性を使用できない

フィルタを適用したロールでは、CoS 仮想属性の値に基づくフィルタ文字列を使用できません。詳細は、88 ページの「CoS について」を参照してください。ただし、CoS 定義の指示子属性は、ロール定義によって生成された `nsRole` 属性を参照できます。ロールベースの属性の作成については、『Directory Server 管理ガイド』の「ロールに基づく属性の作成」を参照してください。

- ロールの範囲拡張

ロールの範囲は異なるサブツリーに拡張できますが、それらは同一サーバーインスタンス上になければなりません。ロールの範囲を別のサーバーにまで拡張することはサポートされていません。

グループとロールのどちらを使用するかの決定

グループとロールのメカニズムは一部の機能が重複していますが、どちらにも利点と欠点があります。一般に、より新しく設計されたロールメカニズムのほうが、頻繁に必要となる機能を効率的に提供できるように設計されています。グループ化メカニズムは、サーバーの複雑さに影響を及ぼし、クライアントによるメンバー情報の処理方法を決めるため、グループ化メカニズムは慎重に計画する必要があります。どのメカニズムが適しているかを判断するには、実行するメンバーシップクエリと管理操作を理解する必要があります。

グループメカニズムの利点

- メンバーシップが 20,000 以下の場合にメンバーを列挙するには、スタティックグループが適している。

特定セットのメンバーを列挙するだけであれば、スタティックグループを使用するほうが負担が少なくなります。ただし、メンバー数が 20,000 以下であることが前提となります。20,000 を超えるメンバーを含むスタティックグループはパフォーマンスを低下させます。member 属性を取得してスタティックグループのメンバーを列挙することは、同じロールを共有するすべてのエントリを調べるより簡単です。

- メンバーの割り当てと削除などの管理操作には、スタティックグループが適している。

  グループにユーザーを追加する上で特別なアクセス権が必要ないため、スタティックグループはユーザーの割り当てや削除を行う場合の最適なグループ化メカニズムです。

  グループエントリを作成する権限を持つということは、グループにメンバーを割り当てる権限を持つということになります。これに対して、管理されているロールとフィルタが適用されたロールでは、管理者はユーザーエントリに nsroledn 属性を書き込む権限も持つ必要があります。このアクセス権の制限は、入れ子のロールにも間接的に適用されます。これは、入れ子のロールを作成するということは、すでに定義されているその他のロールをまとめる権限が必要であることを意味するためです。

- フィルタベースの ACI で使用するには、ダイナミックグループが適している。

  ACI でのバインドルールの指定など、フィルタに基づいてすべてのメンバーを検索するだけの場合は、ダイナミックグループを使用します。フィルタを適用したロールはダイナミックグループと似ていますが、ロールメカニズムを探して確認し、nsRole 仮想属性を生成します。クライアントが nsRole 値を必要としない場合は、ダイナミックグループを選択することで、この計算のオーバーヘッドを回避できます。

- 既存のセットに対してセットの追加や削除を行うときは、グループが適している。

  既存のセットに対してセットの追加や削除を行う場合は、入れ子の制限がないため、グループメカニズムがもっとも適しています。ロールメカニズムでは、他のロールを受け入れられるのは入れ子のロールに限られています。

- エントリのグループ化範囲の柔軟性が重要になる場合は、グループが適している。

  グループのメンバー範囲は、グループ定義エントリの場所に関係なくディレクトリ全体であるため、範囲の面ではグループには柔軟性があります。ロールでも、特定のサブツリーを超えて範囲を拡張することができますが、入れ子のロールに範囲拡張属性 nsRoleScopeDN を追加する必要があるため、範囲拡張には制限があります。

ロールメカニズムの利点

- セットのメンバーを列挙し、エントリのすべてのメンバーを検索する場合は、ロールが適している。

ロールはメンバーシップ情報をユーザーエントリにプッシュします。そこでは情報をキャッシュできるので、以後のメンバーシップのテストをより効率的に行えます。サーバーがすべての計算を実行し、クライアントは `nsRole` 属性の値を読み取るだけです。さらに、この属性にすべてのタイプのロールが含まれ、クライアントはすべてのロールを均等に処理できます。グループよりもロールの方が、両方の処理をより効率よく実行でき、クライアント側での処理が簡単になります。

- CoS、パスワードポリシー、アカウントの無効化、ACI など、Directory Server の既存の機能とグループ化メカニズムを統合する場合は、ロールが適している。

  サーバー上のセットのメンバーシップを「自然に」使用する、つまり、メンバーシップに関する計算をサーバーが自動的に行う利点を活用する場合、ロールメカニズムが最適です。ロールは、リソース指向の ACI で、CoS のベースとして、より複雑な検索フィルタ、パスワードポリシー、アカウントの無効化などの一部として使用することができます。グループを使用してこのような統合を行うことはできません。

サービスクラス (CoS) による属性の管理

CoS メカニズムを使用すると、アプリケーションによる介入なしでエントリ間で属性を共有することができます。CoS は、ロールメカニズムと同じように、エントリの取得時にエントリの仮想属性を生成します。CoS は、メンバーシップを定義しません。ロールメカニズムのようにエントリをグループ化するのではなく、一貫性の保持と容量の節約のために、関連するエントリがデータを共有できるようにします。ここでは、CoS メカニズムについて詳しく説明します。説明する内容は次のとおりです。

- CoS について
- CoS 定義エントリと CoS テンプレートエントリ
- CoS の優先順位
- ポインタ CoS、間接 CoS、クラシック CoS
- CoS の制限事項

CoS について

facsimileTelephoneNumber 属性の値が等しい何千ものエントリがディレクトリに格納されているとします。従来のやり方で FAX 番号を変更するには、各エントリを個別に更新する必要があり、管理者にとっては時間のかかる作業でした。CoS を使用すると、FAX 番号は 1 か所に保存され、Directory Server は、エントリが返されるときに、関係のあるすべてのエントリに対して facsimileTelephoneNumber 属性を自動的に生成します。

クライアントアプリケーションでは、生成された CoS 属性はほかの属性と同じように検出されます。ただし、ディレクトリ管理者が管理するのは 1 つの FAX 番号値だけです。また、ディレクトリに格納される値が少ないため、データベースが使用するデータ領域が少なくて済みます。CoS メカニズムを使用すると、生成された値をエントリで上書きすることも、同じ属性に対して複数の値を生成することも可能です。

注 CoS 仮想属性にはインデックスが付けられないため、LDAP 検索フィルタで参照した場合、パフォーマンスに影響が生じる可能性があります。

生成される CoS 属性には、複数の値が設定されている場合があります。指示子が複数のテンプレートエントリを指定する場合もあれば、同じ属性に複数の CoS 定義がある場合もあります。また、すべてのテンプレートから 1 つの値だけが生成されるように、テンプレートの優先順位を指定することもできます。詳細は、『Directory Server 管理ガイド』の「サービスクラス (CoS) の定義」を参照してください。ロールとクラシック CoS を組み合わせて使用すると、ロールに基づく属性を指定できます。エントリは関連付けられた CoS テンプレートを持つ特定のロールを所有しているため、これらの属性がエントリ上に現れます。たとえば、ロールに基づいた属性を使用して、ロール単位で検索制限を設定できます。

CoS の機能は再帰的に使用できます。つまり、Cos で生成されたほかの属性によって指定される CoS から属性を生成できます。CoS スキーマを複雑にすると、クライアントから情報へのアクセスが単純化され、繰り返し使用する属性を簡単に管理できるようになりますが、同時に管理が複雑になり、サーバーのパフォーマンスが低下します。極端に複雑な CoS スキーマは避けてください。たとえば、多くの間接 CoS スキーマは、クラシック CoS またはポインタ CoS として再定義することができます。

また、CoS 定義を必要以上に変更しないでください。サーバーは CoS 情報をキャッシュに保存するため、CoS 定義への変更がすぐには反映されません。キャッシュを利用することで、生成された属性に高速にアクセスできるようになりますが、CoS 情報が変更されると、サーバーはキャッシュを再構築しなくてはなりません。これは、多少時間のかかるタスク (通常は数秒) です。キャッシュの再構築中は、読み取り操作は新たに変更された情報ではなく、古いキャッシュに残されされている情報に対して行われます。このため、CoS 定義を頻繁に変更した場合、古くなったデータにアクセスする可能性が高くなります。

CoS 定義エントリと CoS テンプレートエントリ

CoS メカニズムは、CoS 定義エントリと CoS テンプレートエントリという 2 種類のエントリに依存します。

CoS 定義エントリ

CoS 定義エントリは、CoS のタイプおよび生成される CoS 属性の名前を特定します。このエントリは、ロール定義エントリと同様に、`LDAPsubentry` オブジェクトクラスから継承されます。CoS の範囲は、定義エントリの格納場所によって決定されます。この範囲は、CoS 定義エントリの親の下にあるサブツリー全体となります。定義エントリの親の分岐内のすべてのエントリを CoS 定義のターゲットエントリと呼びます。同じ CoS 属性に複数の定義が存在することもあります。この場合は、複数の値が含まれます。

CoS 定義エントリは、`cosSuperDefinition` オブジェクトクラスのインスタンスです。CoS 定義エントリは、CoS のタイプを指定する、次のオブジェクトクラスのいずれかから継承されます。

- `cosPointerDefinition`
- `cosIndirectDefinition`
- `cosClassicDefinition`

CoS 定義エントリには、仮想 CoS 属性、テンプレート DN、およびターゲットエントリの指示子属性を指定できるように、CoS のそれぞれのタイプに固有の属性が含まれています。デフォルトでは、CoS メカニズムは、CoS 属性と同じ名前を持つ既存の属性の値を上書きしません。ただし、CoS 定義エントリの構文を使用することで、この動作を制御できます。

注 スキーマ検査が有効になっている場合は、その Cos 属性を設定できるすべてのターゲットエントリにこの属性が生成されます。スキーマ検査が無効になっている場合は、すべてのターゲットエントリに CoS 属性が生成されます。

CoS テンプレートエントリ

CoS テンプレートエントリには、CoS 属性に生成される値が含まれます。CoS の適用範囲内のすべてのエントリで、ここに定義された値が使用されます。それぞれが異なる値を持つ複数のテンプレートが存在することもあります。この場合、生成される属性は複数の値を持ちます。CoS メカニズムは、定義エントリとターゲットエントリの内容に基づいていずれかの値を選択します。

CoS テンプレートエントリは、cosTemplate オブジェクトクラスのインスタンスです。CoS テンプレートエントリには、CoS メカニズムによって生成された 1 つ以上の値が含まれます。特定の CoS 用のテンプレートエントリは、その CoS 定義と同じレベルのディレクトリツリー内に格納されます。

注 管理を容易にするため、定義エントリとテンプレートエントリはできるだけ同じ場所に格納してください。また、それらが提供する機能を説明するような名前を付けてください。たとえば、定義エントリ DN に "cn=classicCosGenEmployeeType,ou=People,dc=example,dc=com" などの名前を付けると、"cn=ClassicCos1,ou=People,dc=example,dc=com" よりもわかりやすくなります。CoS の各タイプに関連するオブジェクトクラスと属性については、『Directory Server 管理ガイド』の「サービスクラス (CoS) の定義」を参照してください。

CoS の優先順位

属性値を得る上で互いに競合する CoS スキームが作成される場合があります。たとえば、CoS 定義エントリに複数の値を持つ cosSpecifier があると仮定します。このような場合、どのテンプレートが属性値を提供するかを決定するために、各テンプレートエントリにテンプレート優先順位を指定することができます。テンプレート優先順位の設定には、cosPriority 属性を使用します。この属性は、特定のテンプレートのグローバルな優先順位を数字で表します。優先順位 0 は、優先順位がもっとも高いことを示します。

cosPriority 属性を持たないテンプレートは、優先順位がもっとも低いとみなされます。2 つ以上のテンプレートが属性値を提供する状況で、その優先順位が同じまたは設定されていない場合は、任意の値が選択されます。

ポインタ CoS、間接 CoS、クラシック CoS

テンプレートの選択方法が異なり、それによって生成される値が異なる 3 つのタイプの CoS があります。次に、これらの 3 種類の CoS について詳しく説明します。

- ポインタ CoS
- 間接 CoS
- クラシック CoS

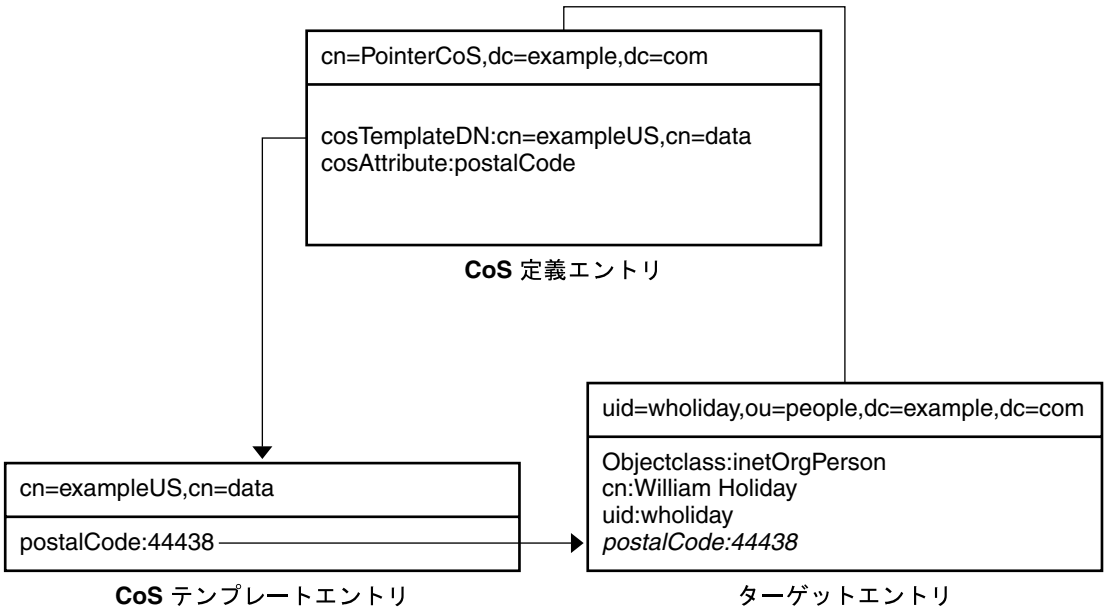
ポインタ CoS

ポインタ CoS はもっとも単純なタイプの CoS です。ポインタ CoS 定義エントリによって、cosTemplate オブジェクトクラスの特定のテンプレートエントリの DN が決定されます。すべてのターゲットエントリに、このテンプレートで定義されているものと同じ CoS 属性値が設定されます。

ポインタ CoS の例

図 4-12 に、dc=example,dc=com の下に格納されるすべてのエントリに共通の郵便番号を定義するポインタ CoS を示します。ここでは、Cos 定義エントリ、Cos テンプレートエントリ、およびターゲットエントリを示しています。

図 4-12 ポインタ CoS の定義とテンプレートの例



テンプレートエントリは、CoS 定義エントリ内の DN (cn=exampleUS,cn=data) によって特定されます。エントリ dc=example,dc=com で postalCode 属性が照会されるたびに、Directory Server は、テンプレートエントリ cn=exampleUS,cn=data 内の使用可能な値を返します。したがって、郵便コードは、エントリ uid=wholiday,ou=people,dc=example,dc=com と一緒に表示されますが、このエントリには格納されません。

エントリ内に実際に存在する属性の代わりに、CoS によって生成されるいくつかの属性が数千または数百万のエントリで共有される例を考えると、CoS を利用することで節約できる格納のための容量とパフォーマンスの向上を理解することができます。

間接 CoS

間接 CoS を使用すると、ディレクトリ内の任意のエントリをテンプレートにして、CoS 値を指定することができます。間接 CoS 定義エントリは、間接指示子と呼ばれる属性を識別します。ターゲットエントリに含まれるこの属性の値によって、そのエントリで使用されるテンプレートが決定されます。ターゲットエントリ内の間接指示子属性には、DN が含まれています。間接 CoS を使用することで、各ターゲットエントリで異なるテンプレートを使用できるため、CoS 属性に異なる値を指定することができます。

たとえば、departmentNumber 属性を生成する間接 CoS では、指示子として社員の上司を使用できます。ターゲットエントリを検索する場合、CoS メカニズムはテンプレートとして manager 属性の DN 値を使用します。そして、上司の部門番号と同じ値を使用して、社員の departmentNumber 属性を生成します。

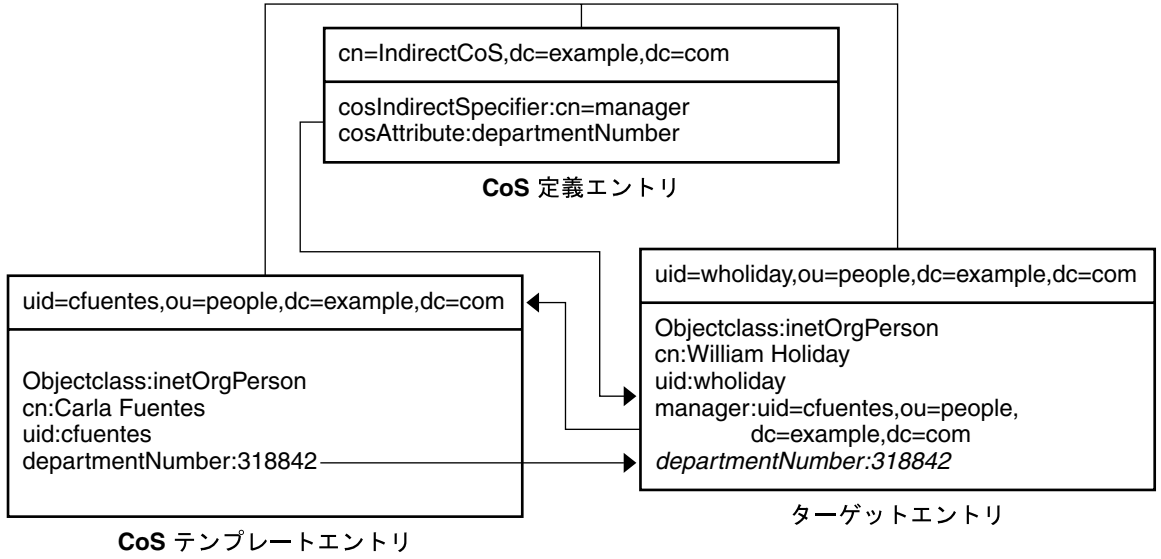
警告 テンプレートは、ディレクトリツリー内の任意の場所にある任意のエントリである可能性があるため、間接 CoS へのアクセス制御の実装は、非常に複雑になります。間接 CoS はリソースを多用するため、パフォーマンスが重要な配備でも使いすぎないようにしてください。

多くの場合、クラシック CoS を使用してターゲットエントリの場所を限定するか、比較的柔軟性の低いポインタ CoS メカニズムを使用して、間接 CoS を使用した場合と同様の結果を得ることができます。

間接 CoS の例

93 ページの図 4-13 では、ターゲットエントリの manager 属性を使用してテンプレートエントリを特定する間接 CoS を示しています。CoS メカニズムでは、この方法で、すべての従業員に対して上司と同じ departmentNumber 属性を生成することにより、常に最新の状態を維持できます。

図 4-13　間接 CoS の定義とテンプレートの例



間接 CoS の定義エントリは、指示子属性の名前を指定します。この例では、manager 属性です。William Holiday のエントリは、この CoS のターゲット エントリの 1 つであり、その manager 属性には、uid=cfuentes,ou=people,dc=example,dc=com の DN が含まれます。したがって、Carla Fuentes のエントリは、departmentNumber 属性値 318842 を提供するテンプレートです。

クラシック CoS

クラシック Co s は、ポインタ CoS と間接 Co s の動作を組み合わせたものです。クラシック CoS の定義エントリは、テンプレートのベース DN と指示子属性を特定します。次のように、ターゲットエントリの指示子属性の値は、テンプレートエントリの DN の構築に使用されます。

cn=*specifierValue* , *baseDN*

CoS 値を含むテンプレートは、ターゲットエントリの指示子属性の RDN (相対識別名) 値とテンプレートのベース DN を組み合わせることで決定されます。

クラシック CoS テンプレートは、任意の間接 CoS テンプレートに関連するパフォーマンスの問題を回避するための、cosTemplate オブジェクトクラスのエントリです。

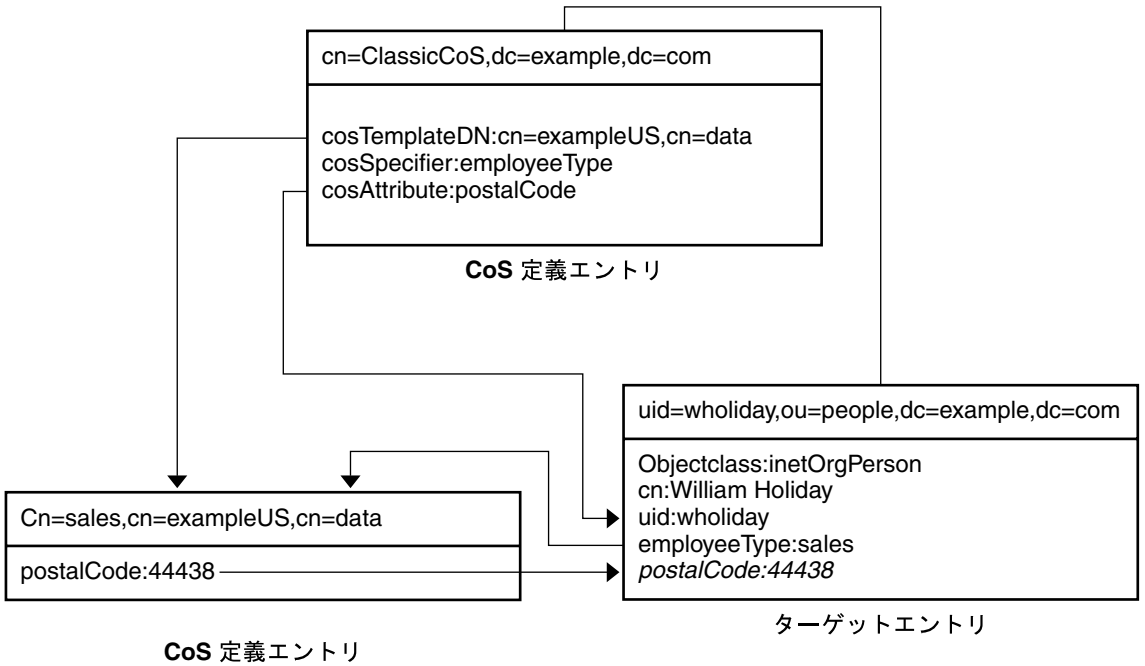
クラシック CoS の例

クラシック CoS メカニズムでは、定義エントリで指定されたベース DN とターゲットエントリの指示子属性からテンプレートの DN が決定されます。指示子属性の値は、テンプレート DN の cn 値として使用されます。したがって、クラシック CoS のテンプレート DN は、次のような構造になります。

cn=*specifierValue*,*baseDN*

図 4-14 は、郵便番号属性の値を生成するクラシック CoS の定義を示しています。

図 4-14 クラシック CoS の定義とテンプレートの例



この例では、CoS 定義エントリの cosSpecifier 属性が、employeeType 属性を指定します。この属性とテンプレート DN を組み合わせると、テンプレートエントリを cn=sales,cn=exampleUS,cn=data として識別できます。このテンプレートエントリは、postalCode 属性の値をターゲットエントリに与えます。

CoS の制限事項

CoS 機能は、複雑なメカニズムであり、パフォーマンスおよびセキュリティ上の理由から次の制限事項が適用されます。

- サブツリーの制限

  `cn=config` サブツリーと `cn=schema` サブツリーには、CoS 定義を作成できません。

- インデックスのない検索

  属性が CoS 生成属性として宣言されているサフィックスの検索は、インデックスが付けられていない検索となります。これは、パフォーマンスに重大な影響を生じる可能性があります。同じ属性が CoS 属性として宣言されていないサフィックスでは、検索にインデックスが付けられます。

- 属性タイプの制限

  次の属性は、同じ名前の実際の属性と動作が異なるため、CoS では生成できません。

  - `userPassword`: CoS で生成されたパスワード値は、Directory Server へのバインドに使用できない。
  - `aci`: Directory Server では、CoS によって定義された仮想 ACI 値の内容に基づいてアクセス制御を適用しない。
  - `objectclass`: Directory Server では、CoS によって定義された仮想オブジェクトクラスの値を検査するスキーマが実行されない。
  - `nsRoleDN`: CoS によって生成された `nsRoleDN` 値は、サーバーによるロールの生成に使用されない。

- すべてのテンプレートをローカルに配置する必要がある

  CoS 定義またはターゲットエントリの指示子に指定されているテンプレートエントリの DN は、ディレクトリ内のローカルエントリを参照する必要があります。テンプレートとそこに含まれる値は、ディレクトリ連鎖またはリフェラルからは取得できません。

- CoS 仮想値と実際の値と組み合わせることはできない

  CoS 属性の値では、エントリの実際の値とテンプレートの仮想値を組み合わせることはできません。CoS により実際の属性値が上書きされると、実際の値はすべてテンプレートの値に置き換えられます。ただし、CoS メカニズムでは、複数の CoS 定義エントリから仮想値を組み合わせることができます。詳細は、『Directory Server 管理ガイド』の「CoS の制限事項」を参照してください。

- フィルタが適用されたロールは CoS によって生成される属性を使用できない

  フィルタを適用したロールでは、CoS 仮想属性の値に基づくフィルタ文字列を使用できません。ただし、CoS 定義の指示子属性は、ロール定義によって生成された `nsRole` 属性を参照できます。詳細は、『Directory Server 管理ガイド』の「ロールに基づく属性の作成」を参照してください。

- アクセス制御命令 (ACI)

  格納されている通常の属性へのアクセスと同様に、CoS によって生成された属性へのアクセスが制御されます。ただし、CoS によって生成された属性値に依存するアクセス制御規則は、95 ページの「属性タイプの制限」で説明されている条件に従います。

- CoS キャッシュの応答時間

  CoS キャッシュは、パフォーマンスを向上させるためにすべての CoS データをメモリに保持する Directory Server の内部構造です。このキャッシュは、仮想属性の算出時に使用される CoS データの取得用に最適化されており、CoS 定義エントリおよびテンプレートエントリの更新中でも使用できます。したがって、定義エントリおよびテンプレートエントリを追加または変更すると、変更内容が反映されるまでわずかに時間がかかる場合があります。この遅延時間は、CoS 定義の数と複雑さ、および現在のサーバーの負荷によって異なりますが、通常は、数秒間かかります。複雑な CoS 設定を行うときは、この遅延時間を考慮してください。

その他のディレクトリツリー関連資料

ディレクトリツリーの設計についての関連資料は、次のリンクにあります。

- RFC 2247: Using Domains in LDAP/X.500 Distinguished Names (LDAP/X500 識別名でドメインを使用する方法)
  http://www.ietf.org/rfc/rfc2247.txt
- RFC 2253: LDAPv3, UTF-8 String Representation of Distinguished Names (LDAPv3、識別名を UTF-8 文字列で表す方法)
  http://www.ietf.org/rfc/rfc2253.txt

第 5 章

分散、連鎖、およびリフェラル

第 4 章「ディレクトリ情報ツリー」では、Directory Server がエントリを格納する方法について説明します。Directory Server は膨大な数のエントリを格納できるので、エントリを複数のサーバーに分散して配置しなければならない場合があります。ディレクトリのトポロジは、ディレクトリツリーをどのように複数の物理的な Directory Server に分割するか、およびこれらのサーバーをどのように相互にリンクするかを示します。

この章では、データの分散、連鎖、およびリフェラルを使用してディレクトリデータをより効率的に管理する方法を説明します。この章は、次の節から構成されています。

- トポロジの概要
- データの分散
- リフェラルと連鎖

トポロジの概要

分散ディレクトリは、複数の物理的な Directory Server に分散されているディレクトリツリーにあるディレクトリです。ディレクトリをこのように分割することで、次のことが可能になります。

- ディレクトリを使用するアプリケーションに適切なパフォーマンスを提供する
- ディレクトリの可用性を高める
- ディレクトリの管理を向上させる

ディレクトリが複数のサーバーに分割されている場合、各サーバーはディレクトリツリーの一部分だけを処理します。分散されたディレクトリの動作は、DNS ネームスペースの各部分を特定の DNS サーバーに割り当てるドメインネームサービス (DNS) に似ています。同様に、ディレクトリのネームスペースを複数のサーバーに分散し、クライアント側からは単一のディレクトリツリーとして管理させることができます。

Directory Server には、異なるデータベースに格納されているディレクトリデータをリンクするために、リフェラルと連鎖のメカニズムも用意されています。サフィックスは、レプリケーション、バックアップ、データの復元などの作業の基本単位です。この章では、サフィックス、リフェラル、連鎖について、およびディレクトリのパフォーマンスの向上のためにインデックスを設計する方法について説明します。

データの分散

データを分散すると、すべてのディレクトリのエントリを社内の各サーバー上に格納することなく、ディレクトリを複数のサーバーインスタンスにわたって拡張できます。サーバーインスタンスは、パフォーマンス要件を満たすために複数のマシンに格納することができます。そのため、分散型ディレクトリは、1 つのサーバーで保持できる数よりはるかに多くのエントリを保持できます。

また、分散されていることがユーザーやクライアントアプリケーションからは見えないようにディレクトリを設定することもできます。ディレクトリクライアントに関する限り、単一のディレクトリがそのディレクトリへのクエリに応答します。

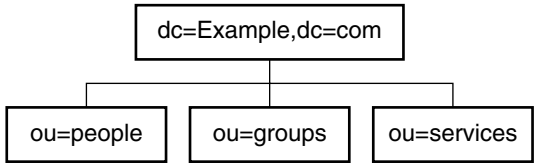
次に、データ分散のしくみについて詳しく説明します。

- 複数のデータベースの使用
- サフィックスについて

複数のデータベースの使用

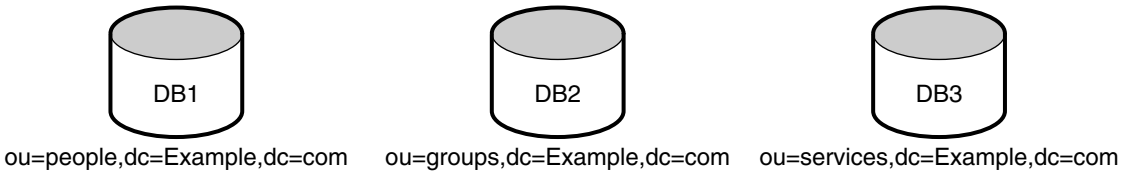
Directory Server はデータを LDBM データベースに格納します。LDBM データベースはディスクベースの高パフォーマンスデータベースです。各データベースは、割り当てられたすべてのデータを含む大きなファイルのセットから構成されます。ディレクトリツリーの異なる部分を別々のデータベースに格納できます。たとえば、図 5-1 のような 3 つのサブサフィックスを含むディレクトリツリーがあるとします。

図 5-1　3 つのサブサフィックスを持つディレクトリツリー



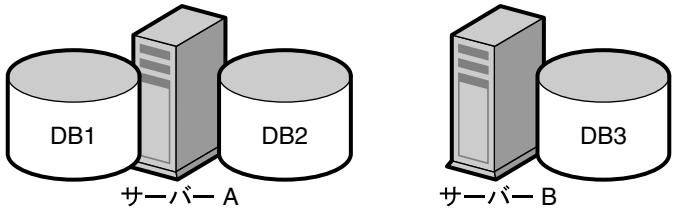
ツリーにある 3 つのサブサフィックスのデータを、図 5-2 のように 3 つの異なるデータベースに格納できます。

図 5-2　3 つの異なるデータベースに格納された 3 つのサブサフィックス



ディレクトリツリーを多数のデータベースに分割すると、それらのデータベースを複数のサーバーに分散できます。通常は、パフォーマンスの向上のために複数のマシンにインストールされた複数のデータベースに分散します。上の図で示した 3 つのデータベースは、図 5-3 に示すように 2 つのサーバー上に格納できます。

図 5-3　2 つのサーバー A、B に格納された Example.com 社の 3 つのデータベース



データベースを複数のサーバーに分散すると、各サーバーが処理しなければならない作業量を削減できます。このようにして、1 つのサーバーで保持できる数よりもはるかに多いエントリに対応するようにディレクトリを拡張できます。また、Directory Server ではデータベースをダイナミックに追加できるので、ディレクトリ全体をオフラインにしなくても、必要に応じて新しいデータベースを追加できます。

サフィックスについて

各データベースには Directory Server のサフィックスにあるデータが格納されます。サフィックスとサブサフィックスの両方を作成して、ディレクトリツリーの内容を編成できます。サフィックスはツリーのルートにあるエントリです。これは、ディレクトリツリー全体のルートか、または大きなツリーの一部である可能性があります。

サブサフィックスは、サフィックスの下にある分岐です。サブサフィックスは、ディレクトリデータの分散を表します。

たとえば、Example.com 社のディレクトリツリーは図 5-4 のようになります。

図 5-4　Example.com 社のディレクトリツリー

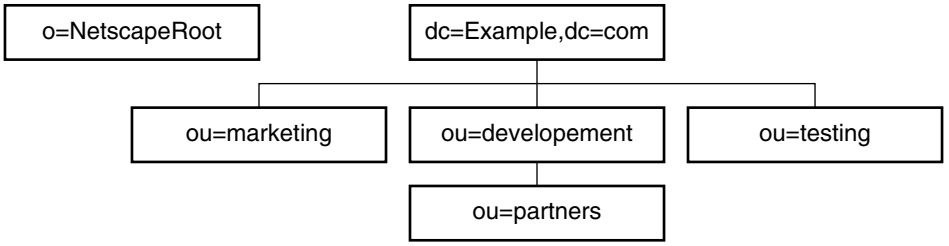
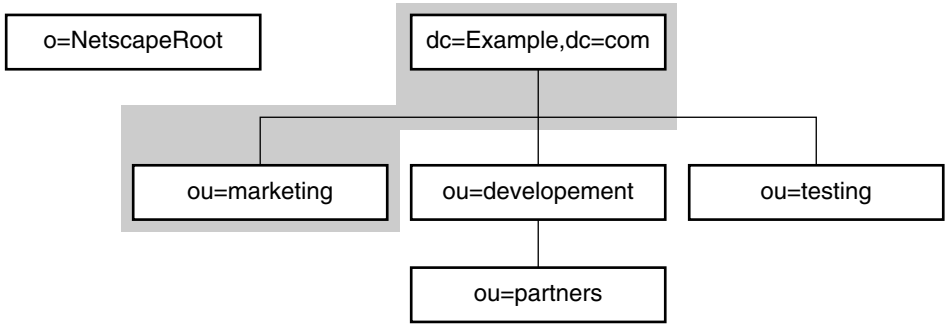


図 5-5 に示すように、Example.com 社がディレクトリツリーを 5 つの異なるデータベースに分割するとします。

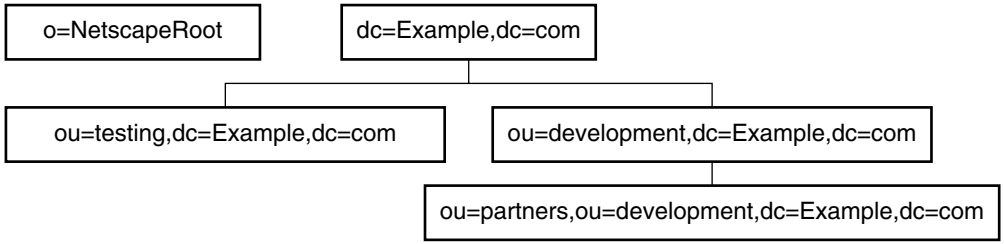
図 5-5　5 つのデータベースに分割された Example.com 社のディレクトリツリー



`o=NetscapeRoot` と `dc=Example,dc=com` はどちらもサフィックスです。それ以外の `ou=testing,dc=Example,dc=com`、`ou=development,dc=Example,dc=com`、`ou=partners,ou=development,dc=Example,dc=com` は、`dc=Example,dc=com` サフィックスのサブサフィックスです。サフィックス `dc=Example,dc=com` には、元のディレクトリツリーの分岐である `ou=marketing` のデータが含まれます。

この分割により、サフィックスとサブサフィックスは図 5-6 のようにエントリを格納します。

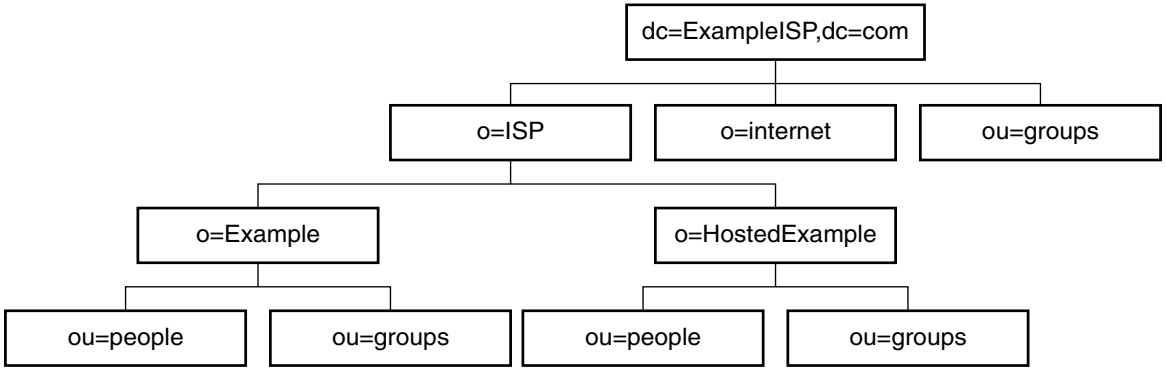
図 5-6　Example.com 社のサフィックスと関連エントリ



ディレクトリには複数のサフィックスを持たせることができます。たとえば、ExampleISP.com という ISP が複数の Web サイトをホスティングし、1 つが独自の Web サイトである ExampleISP.com、もう 1 つが HostedExample.com という別の Web サイトであると仮定します。ISP は、すべてを格納する 1 つのサフィックスを作成するか、同社がホスティングする部分と ExampleISP.com 社の社内データのために 2 つの異なるサフィックスを作成するかを選択できます。

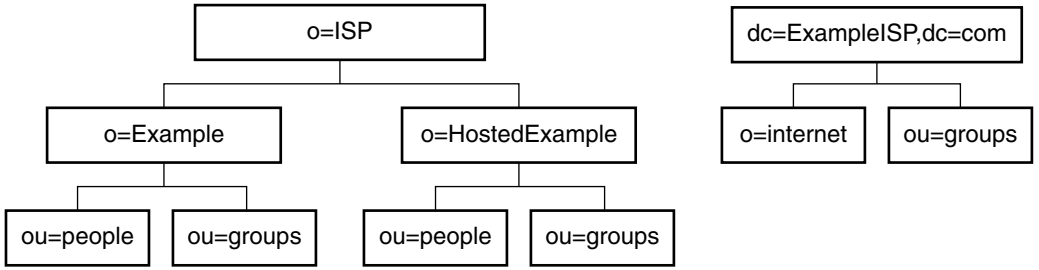
すべてのデータを 1 つのサフィックスに格納する最初のソリューションでは、ディレクトリ情報ツリーは 101 ページの図 5-7 のようになります。

図 5-7　1 つのサフィックスを持つ ExampleISP.com 社のディレクトリツリー



ISP が、1 つはその独自の命名コンテキストに対応し、もう 1 つはそれがホスティングする組織に対応する 2 つのサフィックスを作成した場合、ディレクトリ情報ツリーは次のようになります。

図 5-8　2 つのサフィックスを持つ ExampleISP.com 社のディレクトリツリー



ホスティングされる各組織のエントリ (o=Example と o=HostedExample) は o=ISP サフィックスのサブサフィックスで、ou=people と ou=groups はホスティングされる各組織のサブサフィックスとして分岐します。

リフェラルと連鎖

データが複数のサフィックスに分散されている場合は、分散したデータ間の関係を定義する必要があります。これは、別々のサフィックスに保持されているディレクトリ情報へのポインタを使用して行います。Directory Server には、分散されたデータを単一のディレクトリツリーとしてリンクするために、リフェラルと連鎖というメカニズムが用意されています。

- リフェラル

  サーバーは、要求を完了するために別のサーバーに接続する必要があることを示す情報をクライアントアプリケーションに返します。

- 連鎖

  サーバーはクライアントアプリケーションの代わりに別のサーバーに接続し、操作が終了すると、結合した結果をクライアントアプリケーションに返します。

次に、2 つのメカニズムをより詳細に比較します。

リフェラルの使用方法

リフェラルはサーバーが返す情報であり、操作要求を完了するために接続する必要があるサーバーをクライアントアプリケーションに指示します。Directory Server は、次の 3 種類のリフェラルをサポートしています。

- デフォルトリフェラル

  クライアントアプリケーションが提示した DN に対応するサフィックスがサーバーにない場合、ディレクトリはデフォルトリフェラルを返します。デフォルトリフェラルは、`nsslapd-referral` 属性を使用してサーバーレベルで設定します。

- サフィックスリフェラル

  サフィックス全体がメンテナンスまたはセキュリティ上の理由でオフラインになった場合、サーバーはサフィックスに定義されているリフェラルを返します。クライアントが書き込み処理を要求する場合、サフィックスの読み取り専用レプリカも、マスターサーバーにリフェラルを返します。

- スマートリフェラル

  スマートリフェラルは、ディレクトリ自体のエントリに格納されています。このリフェラルは、そのスマートリフェラルが格納されているエントリの DN と一致する DN を持つサブツリーに関する情報を保有している Directory Server を指します。

すべてのリフェラルは、LDAP URL (Uniform Resource Locator) の形式で返されます。次に、LDAP リフェラルの構造と、Directory Server がサポートする 3 つのタイプのリフェラルについて説明します。

LDAP リフェラルの構造

LDAP リフェラルには LDAP URL 形式の情報が含まれます。LDAP URL には、次の情報が含まれます。

- 接続先サーバーのホスト名。
- サーバーのポート番号。
- 検索操作の場合は ベース DN、追加、削除、および変更操作の場合はターゲット DN。

たとえば、クライアントアプリケーションが `Jensen` という姓を持つエントリを `dc=Example,dc=com` 内で検索するとします。リフェラルは、次の LDAP URL をクライアントアプリケーションに返します。

```
ldap://europe.Example.com:389/ou=people,l=europe,dc=Example,dc=com
```

リフェラルは、LDAP ポート 389 のホスト europe.Example.com に接続し、ou=people,l=europe,dc=Example,dc=com にルート設定された検索要求を送信するように、クライアントアプリケーションに指示します。

リフェラルがどのように処理されるかは、LDAP クライアントアプリケーションによって決まります。一部のクライアントアプリケーションは、参照先サーバーに対して自動的に操作を再実行します。別のクライアントアプリケーションは、ユーザーにリフェラル情報を返します。コマンド行ユーティリティなど、Directory Server が提供するほとんどの LDAP クライアントアプリケーションは、自動的にリフェラルを実行します。参照先サーバーへのアクセスには、最初のサーバー要求で指定したバインド証明情報が使用されます。

ほとんどのクライアントアプリケーションは、リフェラルの制限数、あるいはホップ数だけリフェラルを実行します。実行するリフェラル数を制限すると、クライアントアプリケーションがディレクトリ検索要求を完了しようとして費やす時間を短縮でき、また循環リフェラルパターンが原因で発生する処理停止を防ぐのにも役に立ちます。

デフォルトリフェラル

Directory Server は、要求されたディレクトリオブジェクトの DN とローカルサーバーでサポートされているディレクトリサフィックスを比較して、デフォルトリフェラルを返すかどうかを決めます。DN とサポートされているサフィックスが一致しない場合、Directory Server はデフォルトリフェラルを返します。

たとえば、ディレクトリクライアントが次のディレクトリエントリを要求したとします。

```
uid=bjensen,ou=people,dc=Example,dc=com
```

ところが、サーバーは dc=europe,dc=Example,dc=com サフィックスの下に格納されているエントリしか管理していません。このような場合、ディレクトリは、dc=Example,dc=com サフィックスに格納されているエントリを取得するために接続する必要があるサーバーを示すリフェラルをクライアントに返します。デフォルトリフェラルを受け取ったクライアントは適切なサーバーに接続して、元の要求を再送信します。

デフォルトリフェラルは、ディレクトリの分散に関する情報をより多く保持している Directory Server を指すように設定します。サーバーのデフォルトリフェラルは、dse.ldif 設定ファイルに格納されている nsslapd-referral 属性で設定します。

デフォルトリフェラルの設定については、『Directory Server 管理ガイド』の「デフォルトリフェラルの設定」を参照してください。

サフィックスリフェラル

サフィックスを完全に無効にすることなくサフィックスへのアクセスを制限するには、アクセス権を変更して、読み取り専用アクセスを許可することもできます。この場合、書き込み操作に対しては、別のサーバーへのサフィックスリフェラルを定義する必要があります。また、読み取りアクセスと書き込みアクセスの両方を拒否し、サフィックスへのすべての操作に対するリフェラルを定義することもできます。さらに、サフィックスリフェラルを使用して、クライアントアプリケーションが一時的に別のサーバーを使用するように設定することもできます。たとえば、サフィックスにリフェラルを追加しておくことにより、サフィックスの内容のバックアップ中に、クライアントが別のサーバーを使用するように設定できます。

米国内に 2 つの主要サイトがあり、1 つはニューヨーク、もう 1 つはロサンゼルスに基盤を置いていると仮定します。クライアントアプリケーションは、ニューヨークのサイトを次のように照会します。

```
uid=bjensen,ou=people,dc=US,dc=Example,dc=com
```

サフィックスリフェラルを `dc=NewYork,dc=US,dc=Example,dc=com` と設定することで、`dc=NewYork` サブツリーを含むサフィックスによって要求が処理されるようになります。

サフィックスリフェラルは、そのサフィックスのマッピングツリーエントリに含まれる `nsslapd-state` および `nsslapd-referral` 属性によって設定されます。`nsslapd-referral` 属性は、サフィックスが返す LDAP URL を指定します。`nsslapd-state` 属性は、次の 4 つの値のいずれかをとることができます。

- `nsslapd-state: backend`: すべての操作はこのサフィックスによって処理される。
- `nsslapd-state: disabled`: このサフィックスは処理には利用できず、クライアントアプリケーションからの要求に応答してエラーを返す。
- `nsslapd-state: referral`: このサフィックスに対するすべての要求に対してリフェラルが返される。
- `nsslapd-state: referral on update`: すべての操作に対してこのサフィックスが使用され、更新要求に対してはリフェラルが返される。`referral on update` 状態は、レプリケーションが設定された場合に、コンシューマが更新要求を処理しないようにサーバーで内部的に使用される。ただし、ロードバランスとパフォーマンスのために、特定のサフィックスでの読み取り操作へのアクセスの制限にも、この状態を使用できる。

サフィックスリフェラルの設定については、『Directory Server 管理ガイド』の「アクセス権とリフェラルの設定」を参照してください。

スマートリフェラル

また、Directory Server は、ディレクトリのエントリまたはディレクトリツリーを特定の LDAP URL に関連付けることができる、「スマートリフェラル」もサポートしています。ディレクトリのエントリを特定の LDAP URL に関連付けると、次のいずれかに要求を転送できます。

- 異なるサーバー上の同じネームスペース
- ローカルサーバー上の異なるネームスペース
- 同じサーバー上の異なるネームスペース

デフォルトリフェラルとは異なり、スマートリフェラルはディレクトリ自体に格納されます。

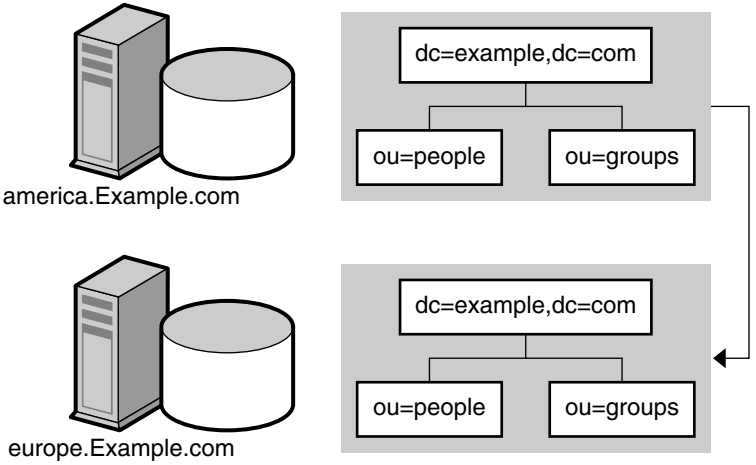
たとえば、Example.com の米国支社のディレクトリに、`ou=people,dc=Example,dc=com` というディレクトリ分岐点があるとします。

`ou=people` エントリ自体にスマートリフェラルを指定することで、この分岐への要求をすべて Example.com 社のヨーロッパ支社の `ou=people` 分岐に転送できます。このスマートリフェラルは、次のようになります。

```
ldap://europe.Example.com:389/ou=people,dc=Example,dc=com
```

米国支店のディレクトリにある people 分岐への要求はすべて、ヨーロッパのディレクトリに転送されます。106 ページの図 5-9 は、このようなスマートリフェラルを示しています。

図 5-9 米国のディレクトリからヨーロッパのディレクトリへのスマートリフェラル



同じメカニズムを使用して、異なるネームスペースを使っている別のサーバーにクエリを転送できます。たとえば、Example.com 社のイタリア支社の従業員がヨーロッパのディレクトリに米国の Example.com 社員の電話番号を要求したとします。ディレクトリは次のリフェラルを返します。

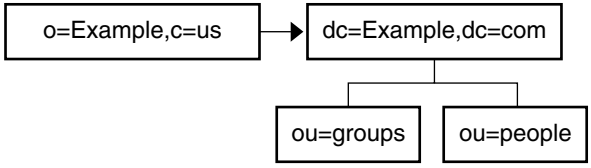
```
ldap://europe.Example.com:389/ou=US employees,dc=Example,dc=com
```

同じサーバー上に複数のサフィックスを設定している場合は、あるネームスペースから同じマシン上の別のネームスペースにクエリを転送できます。ローカルマシン上の `o=Example,c=us` に対するすべてのクエリを `dc=Example,dc=com` に転送する場合は、`o=Example,c=us` エントリに次のスマートリフェラルを設定します。

```
ldap:///dc=Example,dc=com
```

図 5-10 を参照してください。

図 5-10　スマートリフェラルのトラフィック



1 つのネームスペースからのクエリを同一マシン上の別のネームスペースに転送するため、通常は URL に指定される `host:port` の情報ペアを指定する必要はありません。URL にこの情報ペアが指定されていないため、同じ Directory Server を指す URL には 3 つのスラッシュが含まれます。

注	リフェラルを最大限に活用できるように、リフェラルが設定されている場所の下を検索のベースにしないでください。

LDAP URL の詳細については、『Directory Server Administration Reference』の「LDAP URL Reference」を参照してください。Directory Server のエントリにスマート URL を含める方法については、『Directory Server 管理ガイド』の「スマートリフェラルの作成」を参照してください。

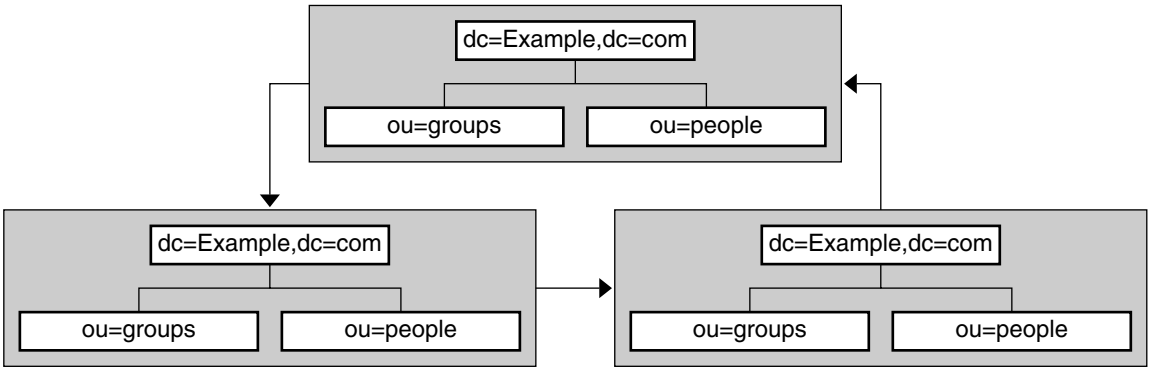
スマートリフェラルを設計する際のヒント

スマートリフェラルを使用するときは、次の点を考慮してください。

- 設計を単純にする。

複雑に交錯したリフェラルは、ディレクトリの管理を難しくします。また、スマートリフェラルを使いすぎると、循環リフェラルパターンを引き起こしてしまう場合もあります。この場合、リフェラルがある LDAP URL を指し、その LDAP URL が別の LDAP URL を指すといったことが、連鎖のどこかで最後に元のサーバーに戻ってしまうまで続くという状況になります。108 ページの図 5-11 は、循環リフェラルのパターンを示しています。

図 5-11 スマートリフェラルの使いすぎによる循環リフェラルのパターン



- 主要な分岐点でリダイレクトする。

  サフィックスレベルで転送を処理するように、リフェラルを制限します。スマートリフェラルを使用すると、最下位のエントリ (分岐ではない) への検索要求を別のサーバーや DN に転送できます。そのため、スマートリフェラルをエイリアスメカニズムとして使用することがよくあり、その結果、ディレクトリ構造の安全保護を複雑で困難なものにしています。リフェラルの使用をディレクトリツリーのサフィックスまたは主要な分岐点に限定することで、管理するリフェラル数が減少し、管理に伴うオーバーヘッドを軽減できます。

- セキュリティへの影響を考慮する。

  アクセス制御はリフェラルの境界を越えると機能しません。要求を発信したサーバーがエントリへのアクセスを許可している場合でも、スマートリフェラルがクライアントの要求を別のサーバーに転送したときは、クライアントアプリケーションがアクセスを拒否されることもあります。

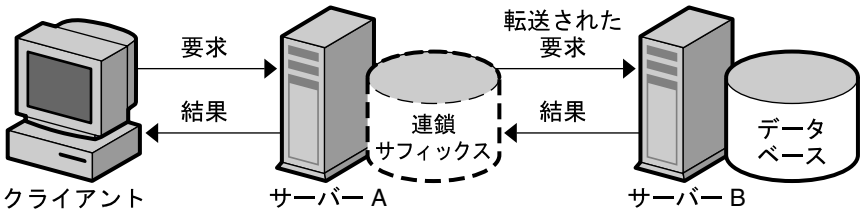
  また、クライアントが認証されるためには、クライアントは転送先のサーバー上でクライアント証明情報を使用できなければなりません。

連鎖の使用方法

連鎖は要求を別のサーバーに中継する手法の 1 つです。この手法は連鎖サフィックスを介して実行されます。98 ページの「データの分散」で説明しているように、連鎖サフィックスにはデータは含まれません。連鎖サフィックスは、クライアントアプリケーションの要求をデータがあるリモートサーバーに転送します。

連鎖時に、サーバーは格納されていないデータに対する要求をクライアントアプリケーションから受け取ります。サーバーは連鎖サフィックスを使用して、クライアントアプリケーションの代わりに他のサーバーにアクセスし、結果をクライアントアプリケーションに返します。図 5-12 は、この処理を示しています。

図 5-12　連鎖による処理



各連鎖サフィックスは、データを保持しているリモートサーバーに関連付けられています。障害が発生したときに連鎖サフィックスが使用する、データのレプリカが入った代替リモートサーバーを設定することもできます。連鎖サフィックスの設定については、『Directory Server 管理ガイド』の「連鎖サフィックスの作成」を参照してください。

連鎖サフィックスには次の機能があります。

- リモートデータへの透過的なアクセス。

  連鎖サフィックスがクライアントからの要求を処理するので、データの分散はクライアントからは見えません。

- 動的な管理。

  システム全体をクライアントアプリケーションが使用できる状態にしたままで、ディレクトリを部分的にシステムに追加したり、システムから削除したりできます。連鎖サフィックスは、エントリがディレクトリに再分散されるまで、リフェラルを一時的にアプリケーションに返すことができます。サフィックスを使用してこの機能を実装することもできます。サフィックスはクライアントアプリケーションをデータベースに転送するのではなく、リフェラルを返します。

- アクセス制御。

  連鎖サフィックスは、クライアントアプリケーションに代わって該当する認証情報をリモートサーバーに提示します。アクセス制御を評価する必要がない場合は、リモートサーバーに対するユーザー代行機能を無効にすることができます。アクセス制御と連鎖サフィックスの関係については、『Directory Server 管理ガイド』の「連鎖サフィックスのアクセス制御」を参照してください。

リフェラルと連鎖の選択

分割されたディレクトリ部分をリンクする場合、どちらの手法にも利点と欠点があります。ディレクトリの要件に応じて、どちらか一方、あるいは両方を組み合わせて使用します。

リフェラルと連鎖の大きな違いは、分散された情報の検索方法を認識するための情報が置かれている場所です。連鎖システムでは、この情報がサーバー内に実装されます。リフェラルを使用するシステムでは、この情報はクライアントアプリケーション内に実装されます。

連鎖では、クライアント側の処理は簡単になりますが、その分サーバー側の処理が複雑になります。連鎖対象のサーバーは、リモートサーバーと協調して結果をディレクトリクライアントに送信する必要があります。

リフェラルでは、クライアントがリフェラルを検索して検索結果を照合する必要があります。ただし、連鎖よりもリフェラルを使用した方がクライアントアプリケーションを柔軟に作成でき、開発者は分散型ディレクトリの進行状況をより適切な形でユーザーにフィードバックできます。

次に、リフェラルと連鎖の違いについて詳しく説明します。

使用方法の違い

リフェラルをサポートしないクライアントアプリケーションもあります。連鎖を使用すると、クライアントアプリケーションは 1 つのサーバーと通信するだけで、多数のサーバーに格納されているデータにアクセスすることができます。企業のネットワークがプロキシを使用している場合は、リフェラルが機能しないことがあります。たとえば、クライアントアプリケーションがファイアウォール内の 1 つのサーバーとだけ通信する権限を持っているとします。クライアントアプリケーションが別のサーバーに転送された場合、クライアントはそのサーバーに正常に接続できません。

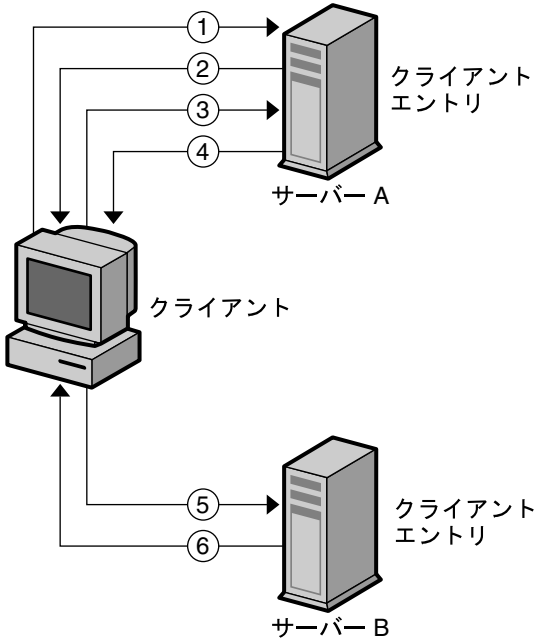
また、リフェラルでは、クライアントを認証する必要があります。つまり、クライアントの転送先のサーバーにクライアントの証明情報が格納されている必要があります。連鎖では、クライアント認証は 1 回だけで済みます。要求の連鎖先のサーバーでクライアントを再び認証する必要はありません。

アクセス制御の評価

連鎖は、リフェラルとは異なる方法でアクセス制御を評価します。リフェラルでは、バインド DN エントリがすべての転送先サーバー上に存在しなければなりません。連鎖では、クライアントエントリがすべての転送先サーバー上に存在している必要はありません。

たとえば、クライアントが検索要求をサーバー A に送信する場合を考えてみます。111 ページの図 5-13 は、リフェラルを使用した場合の動作を示しています。

図 5-13　リフェラルによって転送されるクライアントアプリケーションの検索要求



この図では、クライアントアプリケーションは次の手順を実行します。

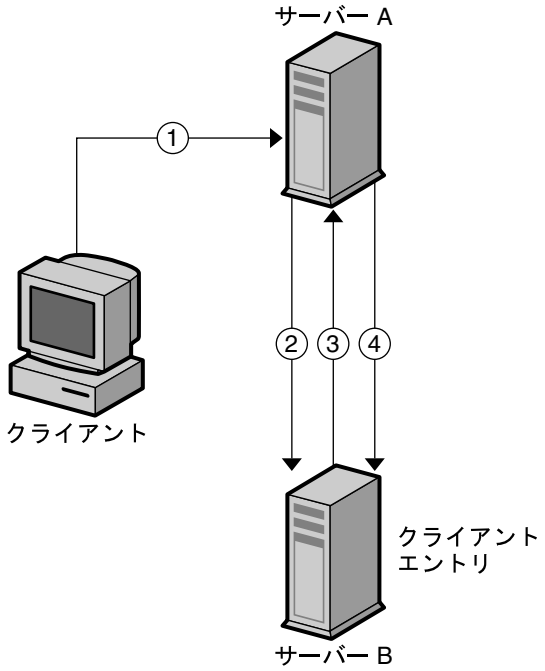
1. クライアントアプリケーションは、まずサーバー A にバインドします。
2. サーバー A は、ユーザー名とパスワードを提供するクライアントのエントリを保持しているので、バインド受け入れメッセージを返します。リフェラルが機能するためには、サーバー A にクライアントエントリが存在する必要があります。
3. クライアントアプリケーションがサーバー A に操作要求を送信します。
4. しかし、サーバー A には要求された情報が格納されていません。サーバー A は、代わりにリフェラルをクライアントアプリケーションに返し、サーバー B へのアクセスを指示します。

5. クライアントアプリケーションが、バインド要求をサーバー B に送信します。サーバーへのバインドに成功するには、サーバー B にクライアントアプリケーションのエントリが存在する必要があります。
6. バインドに成功したので、クライアントアプリケーションは再び検索操作をサーバー B に送信できます。

この方法では、サーバー A からレプリケートされたクライアントエントリのコピーがサーバー B に必要です。

連鎖では、この問題は発生しません。図 5-14 は、連鎖を使用した場合の検索要求の動作を示しています。

図 5-14 連鎖を使用した検索要求



この図では、次の手順が実行されます。

1. クライアントアプリケーションがサーバー A にバインドし、サーバー A はユーザー名とパスワードが正しいことを確認しようとします。
2. サーバー A にはクライアントアプリケーションに対応するエントリが存在しません。その代わりに、サーバー A にはクライアントの実際のエントリがあるサーバー B への連鎖サフィックスがあります。サーバー A はバインド要求をサーバー B に送信します。

3. サーバー B はサーバー A にバインド受け入れメッセージを返信します。
4. サーバー A は連鎖サフィックスを使用してクライアントアプリケーションからの要求を処理します。連鎖サフィックスはサーバー B にあるリモートデータストアに接続して検索操作を処理します。

連鎖システムでは、クライアントアプリケーションに対応するエントリが、クライアントが要求するデータと同じサーバー上にある必要はありません。図 5-15 は、クライアントからの検索要求を処理するために、2 つの連鎖サフィックスを使用する方法を示しています。

図 5-15 クライアントからの検索要求を処理するために 2 つの連鎖サフィックスを使用する連鎖

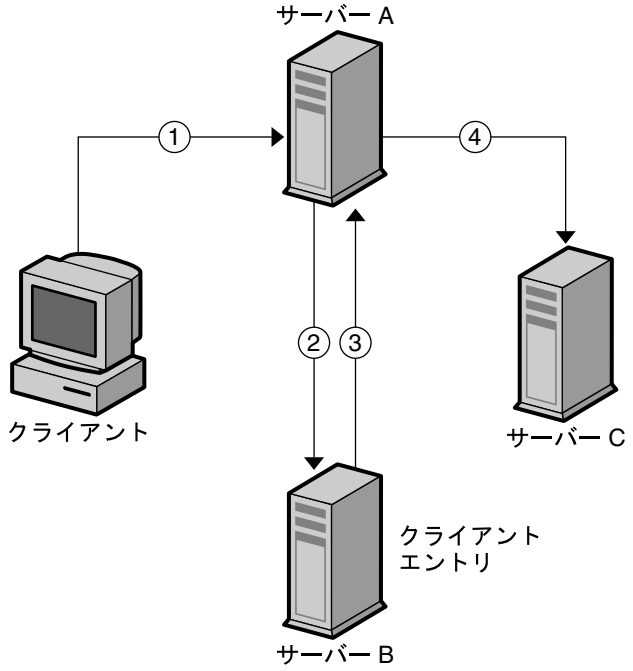


図 5-15 では、次の手順が実行されます。

1. クライアントアプリケーションがサーバー A にバインドし、サーバー A はユーザー名とパスワードが正しいことを確認しようとします。
2. サーバー A にはクライアントアプリケーションに対応するエントリが存在しません。その代わりに、サーバー A にはクライアントの実際のエントリがあるサーバー B への連鎖サフィックスがあります。サーバー A はバインド要求をサーバー B に送信します。
3. サーバー B はサーバー A にバインド受け入れメッセージを返信します。

4. サーバー A は別の連鎖サフィックスを使用してクライアントアプリケーションからの要求を処理します。連鎖サフィックスはサーバー C にあるリモートデータストアに接続して検索操作を処理します。

ただし、連鎖サフィックスは、次のアクセス制御をサポートしません。

- ユーザーエントリが別のサーバー上にある場合、そのユーザーエントリの内容にアクセスする必要がある制御はサポートされません。これには、グループ、フィルタ、およびロールに基づくアクセス制御が含まれます。
- クライントの IP アドレスまたは DNS ドメインに基づく制御は拒否される場合があります。これは、連鎖サフィックスがクライアントとしてリモートサーバーに接続するためです。リモートデータベースに IP に基づくアクセス制御が含まれている場合、リモートデータベースは、元のクライアントのドメインではなく、連鎖サフィックスのドメインを使用してアクセス制御を評価します。

第 6 章

レプリケーションについての理解

ディレクトリの内容をレプリケートすると、ディレクトリの可用性が向上します。また、ロードバランスのような追加の指標が実装されている場合は、グローバルな検索パフォーマンスを向上させる上で役立ちます。レプリケーションでは書き込みの可用性を高めることはできますが、書き込みまたは更新のパフォーマンスは向上しません。

第 4 章と第 5 章では、ディレクトリツリーとディレクトリトポロジの設計について検討しました。この章では、データの物理的および地理的な場所に焦点を当て、特に、レプリケーションを使用していつでもどこでも必要なときにデータを使用できるようにする方法について考察します。

この章では、配備でのレプリケーションの使用について説明します。説明する内容は次のとおりです。

レプリケーションについて

レプリケーションとは、1 つの Directory Server から別の Directory Server にディレクトリのデータを自動的にコピーするメカニズムのことです。レプリケーションを行うと、それ自身のサフィックスに格納しているディレクトリツリーやサブツリーをサーバー間でコピーできます。ただし、設定や監視の情報サブツリーはコピーされません。

レプリケーションを行うと、可用性の高いディレクトリサービスを提供でき、データを地理的に分散することができます。具体的には、レプリケーションには次のような利点があります。

- 耐障害性とフェイルオーバー

  ディレクトリツリーを複数のサーバーにレプリケーションすると、ハードウェア、ソフトウェア、またはネットワークに障害が発生し、ディレクトリクライアントアプリケーションが特定の Directory Server にアクセスできない場合でも、ディレクトリを利用可能にできます。読み取りや書き込み操作を実行するために、クライアントを別のディレクトリに転送することができます。書き込みフェイルオーバーに対応するには、レプリケーション環境にデータのマスターコピーが複数必要であることに注意してください。

- ロードバランスによる応答時間の短縮

  ディレクトリツリーを複数のサーバーにレプリケートすることで、各マシン上のアクセス負荷を軽減し、サーバーの応答時間を短縮できます。レプリケーションは、書き込みのスケーラビリティのためのソリューションではありません。配備内で書き込みのスケーラビリティを改善するには、データの分散を考慮する必要があります。

- データのローカライズによる応答時間の短縮

  ディレクトリのエントリをユーザーに近い場所にレプリケートすることで、ディレクトリの応答時間を短縮できます。

- ローカルでのデータの管理

  レプリケーションを行うと、ローカルにデータを所有して管理でき、同時に全社レベルでそのデータをほかの Directory Server と共有できます。

レプリケーション戦略を決定する前に、レプリケーションの動作を理解しておく必要があります。ここでは、次の点について概要を説明します。

- レプリケーションの概念
- データの整合性

レプリケーションの概念

レプリケーションの実装について考慮するときは、次のような基本的な質問に回答することから始めてください。

- レプリケートする情報
- 情報のマスターコピーを保持するサーバー (複数も可)
- 情報の読み取り専用コピーを保持するサーバー (複数も可)
- 読み取り専用サーバーがクライアントアプリケーションから変更要求を受け取ったときに、要求を転送する転送先サーバー

これらの事項は、Directory Server がレプリケーションをどのように実装するかを理解していないと効率的に決定できません。たとえば、レプリケートする情報を決める場合は、Directory Server が処理できる最小のレプリケーション単位を知っておく必要があります。次に、Directory Server に実装されるレプリケーションの概念について説明します。

レプリカ

レプリケーションに関与するデータベースを、レプリカと呼びます。レプリカには、次の 3 つの種類があります。

- マスターレプリカ (読み書き可能レプリカ): ディレクトリデータのマスターコピーを格納する読み書き可能データベース。マスターレプリカは、ディレクトリクライアントからの更新要求を処理できる。
- コンシューマレプリカ : マスターレプリカに保持される情報のコピーを格納する読み取り専用データベース。コンシューマレプリカは、ディレクトリクライアントからの検索要求を処理できるが、更新要求はマスターレプリカを照会する。
- ハブレプリカ : コンシューマレプリカと同じく、読み取り専用データベース。ただし、1 つまたは複数のコンシューマレプリカを提供する Directory Server 上に格納される。

Directory Server は、複数のレプリカを管理するように設定できます。各レプリカには、レプリケーションにおいて異なる役割を持たせることができます。

レプリケーションの単位

レプリケーションの最小単位はサフィックスです。レプリケーションメカニズムでは、サフィックスとデータベースが 1 対 1 で対応している必要があります。つまり、カスタム分散ロジックを使用している 2 つ以上のデータベースにまたがって分散されているサフィックス (またはネームスペース) はレプリケートできません。レプリケーション単位は、コンシューマとサプライヤの両方に適用されます。つまり、1 つのサフィックスだけを保持するコンシューマに 2 つのサフィックスをレプリケートすることはできません。また、その反対も同様です。

レプリカ ID

マスターレプリカは一意のレプリカ識別子 (ID) を必要とし、コンシューマレプリカはすべて同じレプリカ ID を持ちます。マスターのレプリカ ID は 1 ～ 65534 の間の任意の 16 ビット整数で、コンシューマレプリカのレプリカ ID はすべて 65535 です。レプリカ ID は、どのレプリカで変更が発生したかを識別し、これによってレプリケーションが正しく行われます。

サーバーが複数のレプリカ (またはサフィックス) をホスティングする場合、1 つのレプリケートされたサフィックスを持つ複数のマスターの間でレプリカ ID が一意であれば、それぞれのレプリカがすべて同じレプリカ ID を持つことができます。マスター上のサフィックスすべてに同じレプリカ ID を使用することで、サフィックスとは無関係に、1 つのマスターを 1 つのレプリカ ID のみに関連付けることができます。

サプライヤとコンシューマ

別のサーバーにレプリケートされる Directory Server をサプライヤと呼びます。別のサーバーによって更新される Directory Server をコンシューマと呼びます。サプライヤは、特別に設計された LDAP v3 拡張処理により、コンシューマ上のすべての更新を再現します。このためサプライヤは、パフォーマンスの面ではコンシューマに対する要件の多いクライアントアプリケーションに似ています。

サーバーは、場合によってはサプライヤとコンシューマの両方の機能を持ちます。これは、次の場合に当てはまります。

- サーバーがハブレプリカを含む場合。つまり、サプライヤから更新を受け取り、変更内容をコンシューマにレプリケートする場合。詳細は、132 ページの「カスケード型レプリケーション」を参照してください。
- マルチマスターレプリケーションで、一方のサーバーが他方のサーバーのサプライヤおよびコンシューマとして動作する 2 つの Directory Server がマスターレプリカを保持する場合。詳細は、127 ページの「マルチマスターレプリケーション」を参照してください。
- サーバーがマスターレプリカとコンシューマレプリカの組み合わせを管理する場合

コンシューマとしてだけ機能するサーバー (コンシューマレプリカだけを保持するサーバー) を、専用コンシューマと呼びます。

Directory Server では、レプリケーションは常にサプライヤサーバーから開始されます。コンシューマサーバーから開始されることはありません。サプライヤがコンシューマにデータをプッシュすることから、これをサプライヤ主導レプリケーションと呼びます。Directory Server の初期のバージョンでは、サプライヤからのデータの送信に関して、コンシューマ側に設定するコンシューマ主導のレプリケーションも許されていました。Directory Server 5.0 以降では、コンシューマが更新の送信をサプライヤに要求する手順に切り替えられました。

マスターレプリカでは、サーバーは次のように機能する必要があります。

- ディレクトリクライアントからの更新要求に応答する。
- レプリカの履歴情報と更新履歴ログを保持する。
- コンシューマへのレプリケーションを開始する。

マスターレプリカを保持するサーバーは、管理するマスターレプリカに対して行われる変更を記録する必要があります。これにより、すべての変更がコンシューマにレプリケートされます。

ハブレプリカでは、サーバーは次のように機能する必要があります。

- 読み取り要求に応答する。
- マスターレプリカを保持するサーバーに更新要求を送信する。
- レプリカの履歴情報と更新履歴ログを保持する。
- コンシューマへのレプリケーションを開始する。

コンシューマレプリカでは、サーバーは次のように機能する必要があります。

- 読み取り要求に応答する。
- レプリカの履歴情報を保持する。
- マスターレプリカを保持するサーバーに更新要求を送信する。

  コンシューマは、エントリの追加、削除、変更の要求を受け取ると、その要求はマスターレプリカを保持するサーバーにクライアント経由で送信されます。つまり、レプリケーションの流れの中で、サーバーがサプライヤとして機能します。サプライヤは要求を実行し、変更をレプリケートします。

  セキュリティまたはパフォーマンス上の理由から、リフェラルではなく、エラーを返すようにコンシューマレプリカまたはハブレプリカを設定することもできます。詳細は、120 ページの「リフェラル」を参照してください。

レプリカのオンライン昇格とオンライン降格

レプリカを、オンライン昇格したりオンライン降格したりできます。レプリカの昇格と降格は、レプリケーショントポロジで、レプリカの役割を変更することを意味します。専用コンシューマをハブに変更したり、ハブをマスターに変更したりできます。マスターをハブに変更したり、ハブを専用コンシューマに変更したりすることもできます。コンシューマレプリカをマスターレプリカに昇格させるには、まず、ハブレプリカに昇格させ、それからマスターレプリカに昇格させます。オンライン降格にもこれと同じ段階的なアプローチが適用されます。詳細は、『Directory Server 管理ガイド』の「レプリカの昇格と降格」を参照してください。

オンライン昇格および降格によって柔軟性が向上するだけでなく、フェイルオーバー機能も向上します。たとえば、ロードバランスとフェイルオーバーのために 2 つのハブが設定された双方向のマルチマスターのレプリケーション環境を例に考えてみます。いずれかのマスターがオフラインになった場合は、いずれかのハブを昇格させるだけで読み取り、書き込みの最適な可用性を維持できます。マスターレプリカがオンラインに復帰したときは、ハブレプリカに降格させるだけで元のトポロジを回復できます。

注 ハブがコンシューマに降格されると、レプリカは変更を伝達できなくなり、コンシューマとして更新履歴ログを持たなくなります。このため、ハブをコンシューマに降格する前に、そのハブが他のサーバーと同期していることを確認する必要があります。これを確認するには、レプリケーション監視ツール insync を使用します。このツールについては、157 ページの「レプリケーションの監視」を参照してください。

リフェラル

コンシューマは、変更要求を受け取っても、マスターレプリカを保持するサーバーにその変更要求を転送しません。代わりに、クライアントからの変更要求を正しく実行できる可能性のあるマスターの URL リストを返します。このような URL がリフェラルです。

レプリケーションのメカニズムでは、レプリケーショントポロジに含まれるすべての既知のマスターのリフェラルを返すようにコンシューマを自動的に設定します。ただし、自分のリフェラルを追加し、サーバーが自動的に設定するリフェラルを上書きすることもできます。リフェラルを制御する機能は、セキュリティとパフォーマンスを次のように最適化する際に役立ちます。

- リフェラルがセキュリティ保護されたポートだけを指すようにする。
- ロードバランスのために Directory Proxy Server を指すようにする。
- WAN によってサーバーが分割されている場合にのみ、ローカルサーバーにリダイレクトするようにする。
- 4 方向のマルチマスタートポロジにあるマスターのサブセットだけにリフェラルを限定する。

  デフォルトリフェラルの設定については、『Directory Server 管理ガイド』の「リフェラルの設定」を参照してください。

更新履歴ログ

サプライヤとして機能するすべてのサーバー (マスターレプリカ、ハブレプリカ) は、更新履歴ログを維持します。更新履歴ログは、マスターレプリカに対して行われた変更を記述した記録です。サプライヤは、この変更をコンシューマ上で再現します。

エントリの変更、名称変更、追加、または削除が行われると、実行された LDAP 操作を記述する変更レコードが更新履歴ログに記録されます。

Directory Server の従来のバージョンでは、LDAP から更新履歴ログにアクセスできました。しかし現在では、更新履歴ログはサーバーが内部的に使用するだけで、専用データベースに格納され、LDAP 経由でアクセスできなくなりました。使用しているアプリケーションで更新履歴ログを読み取る必要がある場合は、旧バージョン対応更新履歴ログのプラグインを使用して、下位互換性を保つことができます。旧バージョン対応更新履歴ログプラグインについては、150 ページの「レプリケーションと旧バージョン対応更新履歴ログプラグイン」を参照してください。

注 エントリが更新履歴ログから削除されると、それらのエントリはレプリケートできなくなります。このため、予定される変更の数とサイズを考慮して、更新履歴ログに十分なディスク容量を確保しておく必要があります。詳細は、『Directory Server Performance Tuning Guide』の「Multi-Master Replication Change Logging」を参照してください。

レプリケーションの認証

サプライヤがコンシューマにバインドしてレプリケーション更新を送信するとき、コンシューマサーバーはサプライヤサーバーを認証します。この認証処理を実行するには、サプライヤがコンシューマにバインドする際に使用するエントリがコンシューマサーバーに格納されている必要があります。このエントリをレプリケーションマネージャエントリと呼びます。レプリケーション時に Directory Server コンソールが DN またはバインド DN を参照する場合、レプリケーションマネージャエントリの DN が参照されます。

レプリケーションマネージャあるいはその役割を果たすために作成したエントリは、次のような条件を満たしている必要があります。

- 各コンシューマサーバー上に少なくとも 1 つ存在する (マルチマスター環境における専用コンシューマ、ハブ、マスターもこれに相当する)。
- 初期化およびセキュリティ上の理由から、このエントリをレプリケートされたデータの一部にしない。

レプリケーションマネージャエントリは、コンシューマサーバーに定義されたすべてのアクセス制御規則を迂回する、特別なユーザープロファイルとなります。この特別ユーザープロファイルはレプリケーションだけで有効です。

2 つのサーバー間でレプリケーションを設定する場合は、両方のサーバーでレプリケーションマネージャエントリを識別する必要があります。

- コンシューマサーバーでは、コンシューマレプリカ、ハブレプリカ、またはマスターレプリカ (マルチマスターレプリケーションの場合) を設定するときに、レプリケーション更新の実行が承認されたエントリとしてこれを指定する必要があります。コンソールを使用する場合、デフォルトではレプリケーションマネージャのエントリが使用されます。

- サプライヤサーバー、つまりすべてのマスターレプリカとハブレプリカでは、レプリケーションアグリーメントを設定するときに、このエントリのバインド DN を指定する必要があります。

レプリケーションマネージャのエントリは、デフォルトでは Directory Server コンソール を使ってレプリケーションを設定するときに作成されます。自分のレプリケーションマネージャのエントリを作成することもできます。

レプリケーションと一緒に SSL を使用している場合、認証を行うには次の 2 つの方法があります。

- SSL サーバー認証を使用する場合は、認証するサーバーのレプリケーションマネージャエントリと、認証のためのパスワードが必要です。
- SSL クライアント認証を使用する場合は、認証するサーバーに、証明書を保持するエントリが含まれている必要があります。このエントリは、レプリケーションマネージャエントリとマッピングさせる、またはマッピングさせないことができます。

レプリケーションアグリーメント

Directory Server は、2 つのサーバー間でレプリケーションがどのように行われるかを、レプリケーションアグリーメントを使用して定義します。レプリケーションアグリーメントは、1 つのサプライヤと 1 つのコンシューマの間のレプリケーションを定義します。レプリケーションアグリーメントははサプライヤ上に設定され、レプリケーションが正しく機能するようにする必要があります。既存のレプリケーションアグリーメントを有効にしたり、無効にしたりできます。これは、現在のところレプリケーションアグリーメントの必要はなくても、将来の必要性を考えて保持しておきたい場合に便利です。

レプリケーションアグリーメントは次のものを識別します。

- レプリケートするサフィックス。
- データがレプリケートされるコンシューマサーバー。
- レプリケーションを実行できる時間帯。
- サプライヤがコンシューマへのバインドに使用する必要があるバインド DN と資格 (121 ページの「レプリケーションの認証」を参照)。
- 接続をセキュリティ保護する方法 (SSL、クライアント認証)。
- 部分レプリケーションが設定されている場合は、除外するか含める属性セットへのポインタ (137 ページの「部分レプリケーション」を参照)。
- 1 つの要求にグループ化できる変更の数と、コンシューマからの受信通知が必要となるまでに送信できる要求の数を設定するグループサイズとウィンドウサイズ。
- この特定のアグリーメントのレプリケーションの状態に関する情報。

- Solaris および Linux システムでは、レプリケーション時に使用する圧縮のレベル。

コンシューマの初期化

コンシューマの初期化は、すべてのデータをサプライヤからコンシューマに物理的にコピーする完全更新プロセスです。レプリケーションアグリーメントを作成したら、アグリーメントに定義されているコンシューマを初期化する必要があります。コンシューマが初期化済みの場合、サプライヤは、更新操作をコンシューマに再現、つまりレプリケートすることができます。通常の環境では、それ以後のコンシューマの初期化は必要ありません。ただし、サプライヤ上のデータをバックアップから復元した場合は、そのサプライヤに合わせてコンシューマを初期化し直さなければならない可能性があります。たとえば、復元されたサプライヤがトポロジ内でコンシューマの唯一のサプライヤの場合、コンシューマの再初期化が必要になることがあります。

コンシューマは、オンラインでもオフラインでも初期化できます。コンシューマの初期化プロセスについては、『Directory Server 管理ガイド』の「レプリカの初期化」を参照してください。

マルチマスターレプリケーショントポロジでは、バックアップファイルまたは LDIF ファイルから再初期化された読み書き可能レプリカは、デフォルトではクライアントからの更新要求を拒否します。レプリカはデフォルトで永続的に読み取り専用モードになり、更新操作の要求についてはトポロジ内の他のサプライヤを参照します。この場合は、更新の受け入れをレプリカに設定する必要があります。『Directory Server 管理ガイド』の「マルチマスター初期化後のマスター間の一致」を参照してください。

Directory Server には、拡張されたバイナリコピー機能が用意されています。この機能を利用することで、1 つのサーバーのバイナリバックアップファイルを使って、別のサーバー上の同じディレクトリの内容を復元することで、マスターレプリカまたはコンシューマレプリカのクローンを作成できます。特定の制限が適用されるため、この機能が実用的で時間の節約となるのは、大規模なデータベースファイルのレプリカを対象とする場合だけです。バイナリコピーの手順と機能の制限リストについては、『Directory Server 管理ガイド』の「バイナリコピーによるレプリカの初期化」を参照してください。

差分更新

コンシューマが初期化済みの場合、サプライヤ上で変更が行われたときに、コンシューマにレプリケーション更新が送信されます。このような更新を、差分更新と呼びます。コンシューマは、更新が異なるレプリカ ID に由来する場合は、複数のサプライヤで同時に差分更新することができます。

データの整合性

整合性とは、レプリケートされたデータベースの内容が、任意の時点でどの程度一致しているかを示します。2 つのサーバー間でレプリケーションを設定する場合、設定の一部として更新のスケジュールが含まれます。サプライヤは、いつコンシューマを更新すべきかを判断し、レプリケーションを開始します。レプリケーションを開始できるのは、コンシューマの初期化が完了してからだけです。

Directory Server には、レプリカの同期を常に維持するオプションと、特定の時間または曜日に更新をスケジュールするオプションが用意されています。レプリカの同期を常に維持する利点は、トポロジ間でデータの整合性が保たれるという点です。ただしその場合は、頻繁な更新操作によってネットワークトラフィックが増大します。このソリューションは、次のようなときに最適です。

- サーバー間で信頼性が高く高速な接続を利用できる場合。
- ディレクトリのサービスを受けるクライアント要求が主に検索、読み取り、および比較の操作であり、追加と変更の操作は比較的少ない場合。

データの整合性が低くてもかまわない場合は、更新の頻度を選択して、ネットワークトラフィックへの影響を軽減することができます。このソリューションは、次のようなときに最適です。

- ネットワーク接続の信頼性が低いか、あるいは断続的なネットワーク接続を使用している場合 (ダイヤルアップ接続を使用してレプリカの同期をとっている場合など)。
- ディレクトリのサービスを受けるクライアント要求が主に追加と変更の操作である場合。
- 通信コストを削減する必要がある場合。

マルチマスターレプリケーションの場合は、一般に、各マスターに格納されているデータ間に違いがある可能性があるため、各マスター上のレプリカは緩やかな整合性を保っている状態といえます。これは、レプリカの同期を維持するように選択している場合にも当てはまります。理由は次のとおりです。

- マスター間のレプリケーションの更新操作の伝達に遅延があるため。
- 追加または変更の操作を実行したマスターは、2 番目のマスターがその更新操作を検証するのを待たずに「操作は正常に完了しました」というメッセージをクライアントに返すため。

一般的なレプリケーション設定

レプリケーショントポロジは、レプリケーションの更新情報をサーバー間でやり取りする方法と、更新情報を伝達するときのサーバー間の相互作用の方法を決定します。基本的なレプリケーション設定は 5 つあり、配備に合うように組み合わせることができます。

- シングルマスターレプリケーション
- マルチマスターレプリケーション
- カスケード型レプリケーション
- 混合環境
- 部分レプリケーション

次に、上記の設定ついて説明し、配備にもっとも適した方法を決定するための指針を示します。

注 どのようなレプリケーション設定を実装する場合でも、スキーマのレプリケーションを考慮する必要があります。詳細は、156 ページの「スキーマのレプリケーション」を参照してください。

シングルマスターレプリケーション

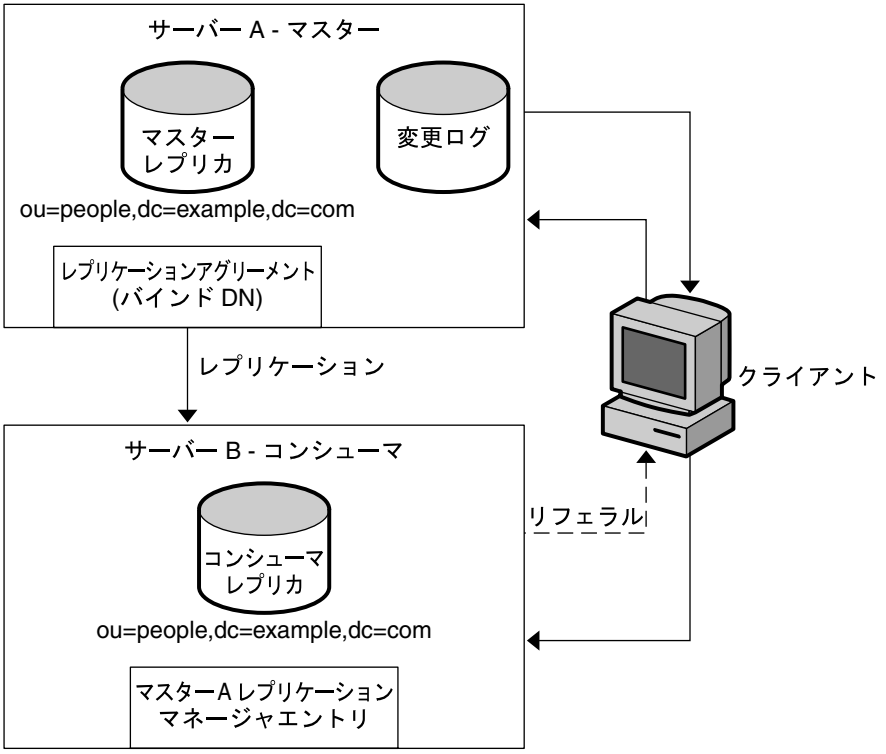
レプリケーションのもっとも基本的な設定では、サプライヤが 1 つまたは複数のコンシューマにマスターレプリカを直接コピーします。この設定では、すべてのディレクトリ変更はマスターレプリカに対して行われ、コンシューマには読み取り専用のデータコピーが維持されます。

サプライヤは、レプリカに対するすべての更新を記録した更新履歴ログを管理します。サプライヤは、レプリケーションアグリーメントも定義します。

サプライヤがレプリケーションの更新を送信するためにコンシューマにバインドしたときに、コンシューマがサプライヤを認証できるように、コンシューマにはレプリケーションマネージャエントリに対応したエントリが格納されます。

サプライヤは、レプリケーションアグリーメントに従って、すべての変更をコンシューマレプリカに伝達します。次の図は、この基本例を示しています。

図 6-1　シングルマスターレプリケーション



この例では、ou=people,dc=example,dc=com サフィックスが、クライアントから多数の検索要求と更新要求を受け取ります。負荷を分散するために、サーバー A 上にマスターのあるこのサフィックスは、サーバー B 上にあるコンシューマレプリカにレプリケートされます。

サーバー B は、クライアントからの検索要求を処理できますが、ディレクトリエントリの変更要求は処理できません。サーバー B は、クライアントにサーバー A に対するリフェラルを返すことによって、クライアントから受け取った変更要求を処理します。コンシューマにサプライヤのリフェラル情報が格納されますが、クライアントからの変更要求は、サプライヤに転送されません。代わりに、クライアントはコンシューマによって返されたリフェラルを実行します。

この例では、1 つのサーバーだけがコンシューマとして機能していますが、サプライヤは複数のコンシューマにレプリケートすることができます。1 つのサプライヤが管理できるコンシューマの総数は、ネットワークの速度と 1 日当たりのエントリの変更総数によって異なります。

マルチマスターレプリケーション

マルチマスターレプリケーション設定では、同じデータを持つマスターレプリカが複数のサーバー上に存在します。この節は、次の節で構成されています。

- マルチマスターレプリケーションの基本概念
- マルチマスターレプリケーションの機能
- 広域ネットワークを介したマルチマスターレプリケーション
- 完全にメッシュ化されたマルチマスタートポロジ

マルチマスターレプリケーションの基本概念

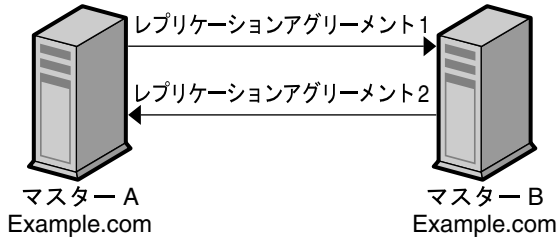
マルチマスター設定では、データは異なる場所で同時に更新することができます。各マスターはそのレプリカの更新履歴ログを格納し、各マスターで行われた更新は、他のサーバーにレプリケートされます。つまり、各マスターはサプライヤとコンシューマの役割を果たします。マルチマスター設定には、次の利点があります。

- 1 つのサプライヤにアクセスできなくなった場合でも、自動的に書き込み処理のフェイルオーバーが実行される。
- 地域分散型環境のローカルサプライヤで更新処理を実行できる。

2 つのサーバー間で更新が送信された場合、更新の競合を解決する必要があります。ほとんどの場合、各変更に関連するタイムスタンプに基づいて競合は自動的に解決されます。もっとも新しい変更が優先されます。ただし、更新の競合を解決するためにユーザーの介入が必要となる場合もあります。詳細は、『Directory Server 管理ガイド』の「よく発生するレプリケーションの競合の解決」を参照してください。

独立した 2 つのサーバーが同じデータのマスターコピーを保持することはできますが、1 つのレプリケーションアグリーメントにおいては、1 つのサプライヤと 1 つのコンシューマだけしか存在できません。このため、同じデータの管理責任を共有する 2 つのサプライヤ間にマルチマスター環境を構築するには、2 つのレプリケーションアグリーメント (各サプライヤで 1 つ) を作成する必要があります。128 ページの図 6-2 は、この設定を示しています。

図 6-2　マルチマスターレプリケーションの設定 (2 つのマスター)



この図では、マスター A とマスター B は、それぞれが同じデータを持つマスターレプリカを保持し、レプリケーションの流れを制御する 2 つのレプリケーションアグリーメントが存在します。マスター A は、レプリケーションアグリーメント 1 の範囲ではマスターとして機能し、レプリケーションアグリーメント 2 の範囲ではコンシューマとして機能します。

マルチマスターレプリケーショントポロジでは、最大 4 つのマスターがサポートされます。コンシューマレプリカとハブの数は、理論的には無制限です。ただし、1 つのサプライヤがレプリケートできるコンシューマの数は、サプライヤサーバーの性能によって異なります。

マルチマスターレプリケーションの機能

レプリケーションプロトコルにより、次のことが可能です。

- レプリカ ID に基づいて更新をレプリケートします。レプリカ ID ベースの更新により、複数のサプライヤがコンシューマを同時に更新できるようになるため、パフォーマンスが向上します (それぞれの更新が異なるレプリカ ID に由来する必要がある)。
- レプリケーションアグリーメントを有効化または無効化して、レプリケーション設定の柔軟性を向上させます。レプリケーションアグリーメントは設定できますが、無効化したままの場合、必要になったときすぐに有効化する必要があります。

広域ネットワークを介したマルチマスターレプリケーション

広域ネットワークを介したマルチマスターレプリケーションは、Directory Server 5.2 より前のバージョンの Directory Server では使用できません。Directory Server 5.2 より前のバージョンの Directory Server では、マルチマスターを高速で待ち時間の短いネットワークを介して接続する必要があります。完全にサポートするためには、100M バイト / 秒以上の転送スピードのネットワークが必要であり、WAN を介したマルチマスターレプリケーションの可能性は問題外です。

Directory Server 5.2 では、WAN を介したマルチマスターレプリケーションをサポートしています。この機能により、マルチマスターレプリケーション設定を地理的に離れた場所にある複数のデータセンターに国際的に配備できます。

レプリケーションプロトコルが、Solaris および Linux システム上での完全な非同期のサポート、ウィンドウとグループ化メカニズム、および圧縮のサポートを行います。これらの機能は、WAN を介したマルチマスターレプリケーションの配備を実現します。WAN を介したマルチマスターレプリケーションはプロトコルの改善によって実現しましたが、この改善は、ローカルエリアネットワーク (LAN) を介した配備にも同様に有効です。

WAN を介したマルチマスターアプリケーションの設定では、WAN によって分割されるすべての Directory Server インスタンスは、5.2 インスタンスである必要があります。

グループとウィンドウのメカニズム

レプリケーションの流れを最適化するために、Directory Server では変更を個別に送信する代わりにグループ化して送信することができます。また、サプライヤが処理継続のためのコンシューマからの受信通知を待たずにコンシューマに送信できる要求の数を指定することもできます。

グループとウィンドウのメカニズムはエントリのサイズに基づくため、エントリサイズが大きく異なる場合、これらのメカニズムを使ってレプリケーションのパフォーマンスを最適化することは実用的でない場合があります。エントリのサイズが比較的均一である場合は、グループとウィンドウのメカニズムを使用して、差分更新と完全更新を最適化することができます。WAN を介したマルチマスターレプリケーションのパフォーマンスは、使用する WAN の待ち時間と帯域幅によって異なることに注意してください。

ウィンドウとグループのサイズ調整については、『Directory Server 管理ガイド』の「ネットワークパラメータの設定」を参照してください。

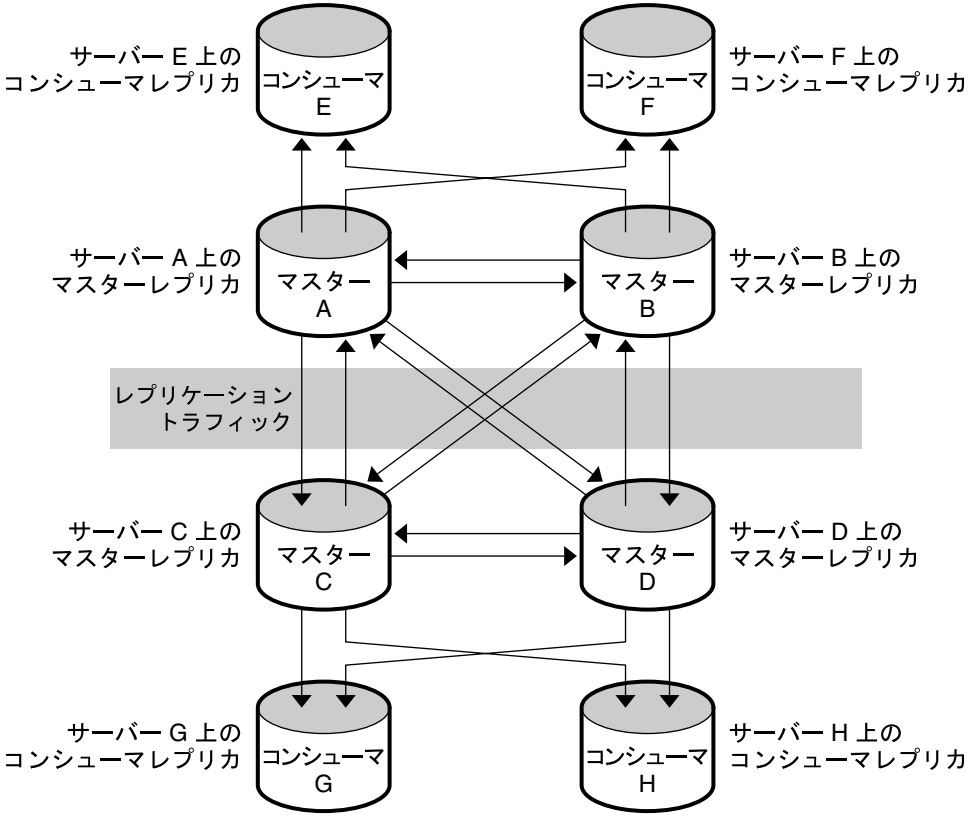
レプリケーションの圧縮

Directory Server は、グループとウィンドウのメカニズムのほかにも、Solaris および Linux システムの圧縮メカニズムを用意しています。Directory Server 5.2 より前のバージョンの Directory Server では、帯域幅の制限が WAN を介したレプリケーションでボトルネックとなることがよくありました。レプリケーションの圧縮は、レプリケーションの流れを効率的にして、このボトルネックを回避するのに役立ちます。コマンド行を使ったレプリケーションの圧縮方法については、『Directory Server Administration Reference』を参照してください。

完全にメッシュ化されたマルチマスタートポロジ

完全にメッシュ化されたトポロジとは、トポロジ内のマスターのそれぞれが、他のマスターのそれぞれに接続されていることを意味します。このようなトポロジは、高可用性と保証されたデータの整合性を提供します。130 ページの図 6-3 は、完全にメッシュ化された 4 方向のマスタートポロジを示しています。

図 6-3　完全にメッシュ化された 4 方向のマルチマスターレプリケーション構成



この例では、4 つのマスターで ou=people,dc=example,dc=com サフィックスが維持され、変更要求に常に対応できるようにしています。各マスターは、自身の更新履歴ログを管理します。いずれかのマスターがクライアントからの変更要求を処理すると、そのマスターは操作を更新履歴ログに記録します。次に、その他のマスターにレプリケーション更新を送信し、そこからその他のコンシューマに伝達されます。このためマスターは、マスター間のレプリケーションアグリーメントだけでなく、コンシューマとのレプリケーションアグリーメントも保持する必要があります。また各マスターには、レプリケーション更新を送信するために他のマスターにバインドできるように、他のマスターの認証に使用するレプリケーションマネージャエントリも格納されます。

各コンシューマはレプリケーションマネージャエントリに対応する 1 つまたは複数エントリを保持しています。これにより、コンシューマはマスターがレプリケーション更新を送信する際にバインドするときに、マスターを認証できます。各コンシューマが 1 つのレプリケーションマネージャエントリだけを保持し、すべてのマスターが同じレプリケーションエントリを認証に使用するように設定することもできます。デ

フォルトでは、コンシューマはトポロジ内のすべてのマスターに対するリフェラルを保持します。クライアントから変更要求を受け取ると、コンシューマはマスターへのリフェラルをクライアントに返します。リフェラルについては、120 ページの「リフェラル」を参照してください。

このトポロジは、読み書き対応フェイルオーバー機能の面ではもっとも安全ですが、この機能を使用するとパフォーマンスに影響する場合があります。完全にメッシュ化されたトポロジは、配備において高可用性がきわめて重大な場合に最適です。高可用性がそれほど重要でない場合、またはパフォーマンス上の理由からレプリケーショントラフィックを削減する場合は、読み書き可能フェイルオーバーの面でより「軽量」の配備が適しています。

レプリケーションの要素を理解しやすくするために、完全にメッシュ化された 4 方向のマルチマスターレプリケーションの設定について説明します。図 6-4 は、マスター A で設定する必要があるレプリケーションアグリーメント、更新履歴ログ、レプリケーションマネージャエントリを示しています。図 6-5 は、コンシューマ E で設定する必要がある同じ要素を示しています。

図 6-4　マスター A のレプリケーション設定 (完全にメッシュ化されたトポロジ)

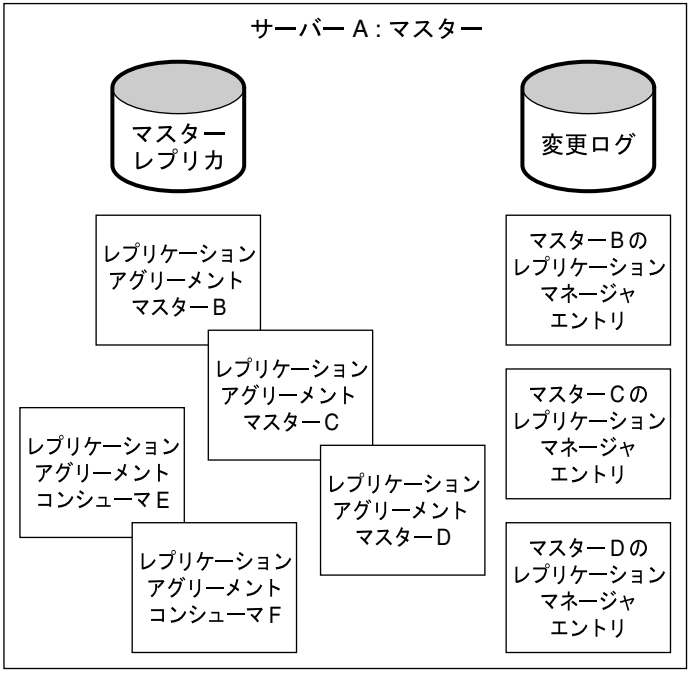


図 6-4 に示されるように、マスター A は、マスターレプリカ、更新履歴ログ、マスター B、C、D 用のレプリケーションマネージャエントリ (4 つすべてのマスターでは、必ずしも同じレプリケーションマネージャエントリを使用しない場合) を必要とします。また、マスター A は、マスター B、C、D 用のレプリケーションアグリーメントとコンシューマ E、F 用のレプリケーションアグリーメントも必要とします。

図 6-5 コンシューマサーバー E のレプリケーション設定 (完全にメッシュ化されたトポロジ)

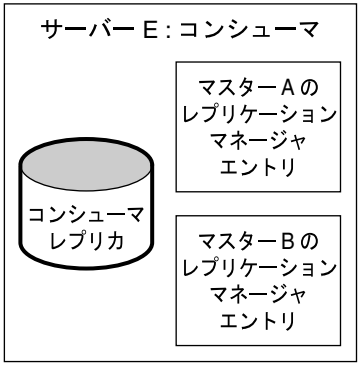


図 6-5 は、コンシューマ E のレプリケーション設定の詳細を示しています。コンシューマ E は、コンシューマレプリカ、およびレプリケーション更新の送信時にマスター A、B とのバインドで認証に使用されるレプリケーションマネージャエントリを必要とします。

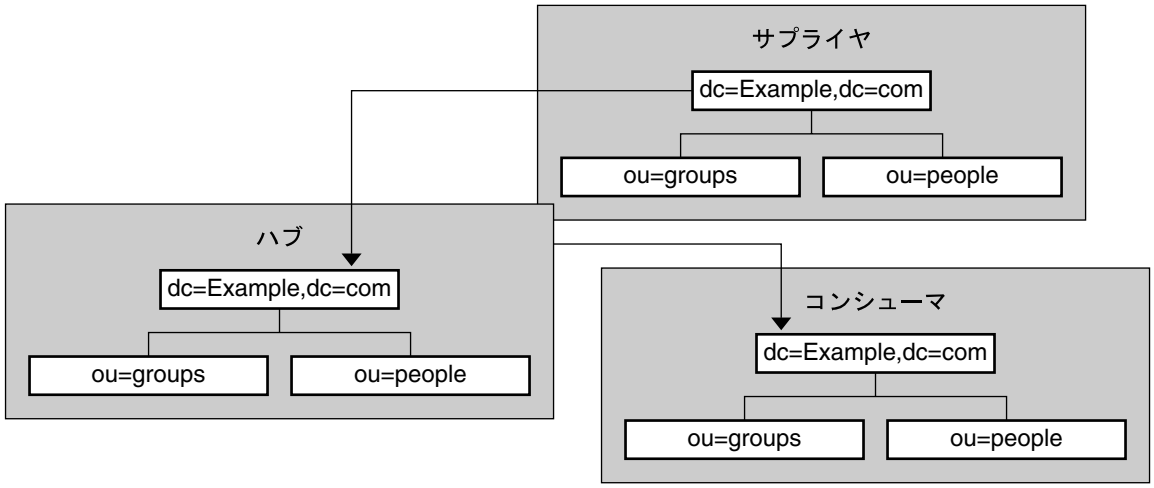
カスケード型レプリケーション

カスケード型レプリケーション設定では、ハブとして機能するサーバーがサプライヤとして機能するサーバーから更新を受け取り、その更新をコンシューマ上で再現します。ハブはコンシューマとサプライヤの、両者の機能を合わせ持っています。つまり、通常のコンシューマと同様にデータの読み取り専用コピーを保持すると同時に、通常のサプライヤと同様に更新履歴ログも保持しています。

ハブは、元のマスターから受け取ったマスターデータのコピーを渡し、ディレクトリクライアントからの更新要求についてマスターを参照します。

図 6-6 は、カスケード型レプリケーション設定を示しています。

図 6-6　カスケード型レプリケーション設定

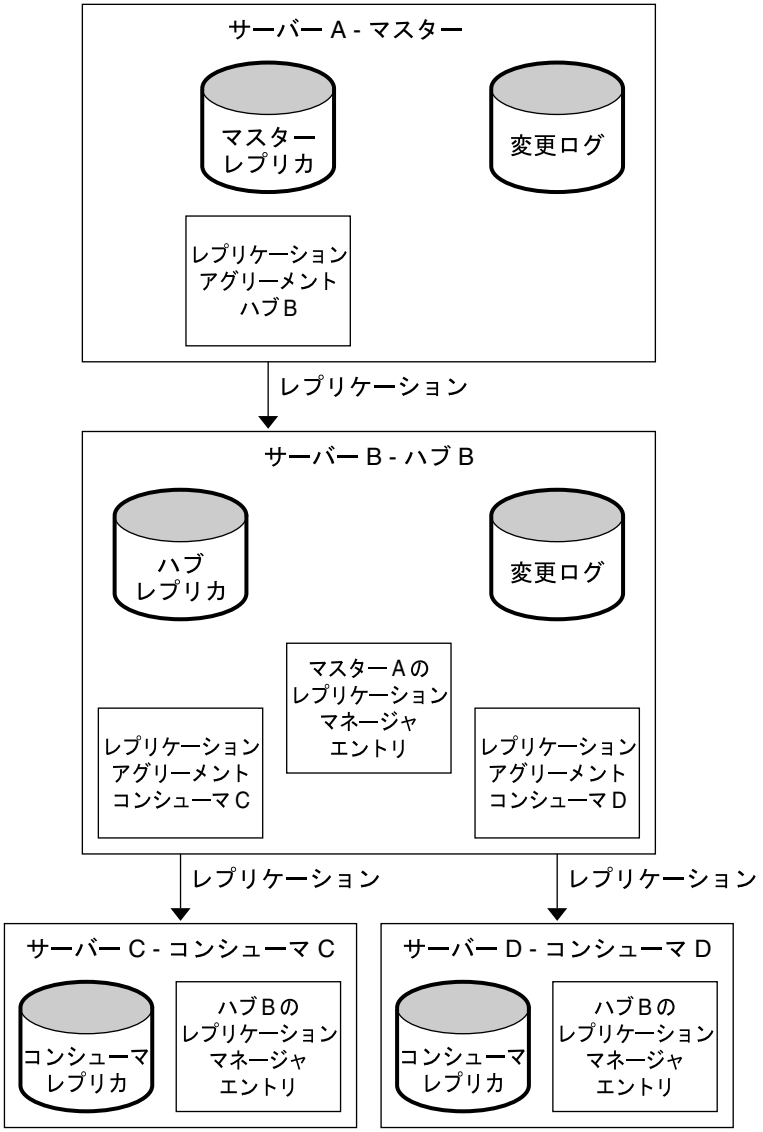


カスケード型レプリケーションは、次の場合に特に便利です。

- 非常に大きなトラフィック負荷のバランスをとる必要がある場合。レプリケーショントポロジのマスターはすべての更新トラフィックを処理するため、コンシューマへのレプリケーショントラフィックをサポートするには、大きな負荷がかかります。多数のコンシューマに対するレプリケーション更新を実行できるハブにレプリケーショントラフィックを任せることで、負荷を軽減できます。
- 地域分散型環境でローカルハブサプライヤを使用することで、接続費用を削減したい場合。
- ディレクトリサービスのパフォーマンスを向上させる場合 : 読み取り操作を行うすべてのクライアントアプリケーションをコンシューマへ、更新操作を行うすべてのクライアントアプリケーションをマスターへ導くことができる場合は、ハブのすべてのインデックス (システムインデックスを除く) を削除できます。これにより、マスターとハブの間のレプリケーション速度が向上します。

図 6-7 は、前の例で説明したサーバーがレプリケーションアグリーメント、更新履歴ログ、デフォルトリテラルの面からどのように設定されるかを示しています。

図 6-7　カスケード型レプリケーションのサーバー設定



この例では、ハブ B はコンシューマ C、D にレプリケーション更新をリレーするために使用され、マスター A は多くのリソースを含んだままディレクトリの更新処理を続けます。マスターとハブは、どちらも更新履歴ログを維持します。ただし、クライアントからの変更要求を直接処理できるのは、マスターだけです。ハブにはマスター A

のレプリケーションマネージャエントリが含まれるので、マスター A はハブにバインドし、レプリケーション更新を送信できます。コンシューマ C、D にはハブ B のレプリケーションマネージャエントリが含まれ、コンシューマへの更新の送信時の認証にはこれらのエントリが使用されます。

コンシューマとハブは、クライアントからの検索要求を処理できますが、変更要求の場合は、マスターへのリフェラルをクライアントに送信します。図 6-7 は、コンシューマ C、D がマスター A へのリフェラルを保持していることを示しています。これは、ハブとコンシューマの間のレプリケーションアグリーメントを作成するときに自動的に作成されるリフェラルです。ただし、パフォーマンスまたはセキュリティ上の理由から、これらのリフェラルを書き換えることもできます。詳細は、120 ページの「リフェラル」を参照してください。

混合環境

上で説明したレプリケーション設定を、配備に合うように自由に組み合わせることができます。たとえば、マルチマスター設定とカスケード型設定を組み合わせて、図 6-8 に示すようなトポロジを構築できます。

図 6-8　マルチマスターレプリケーションとカスケード型レプリケーションの組み合わせ

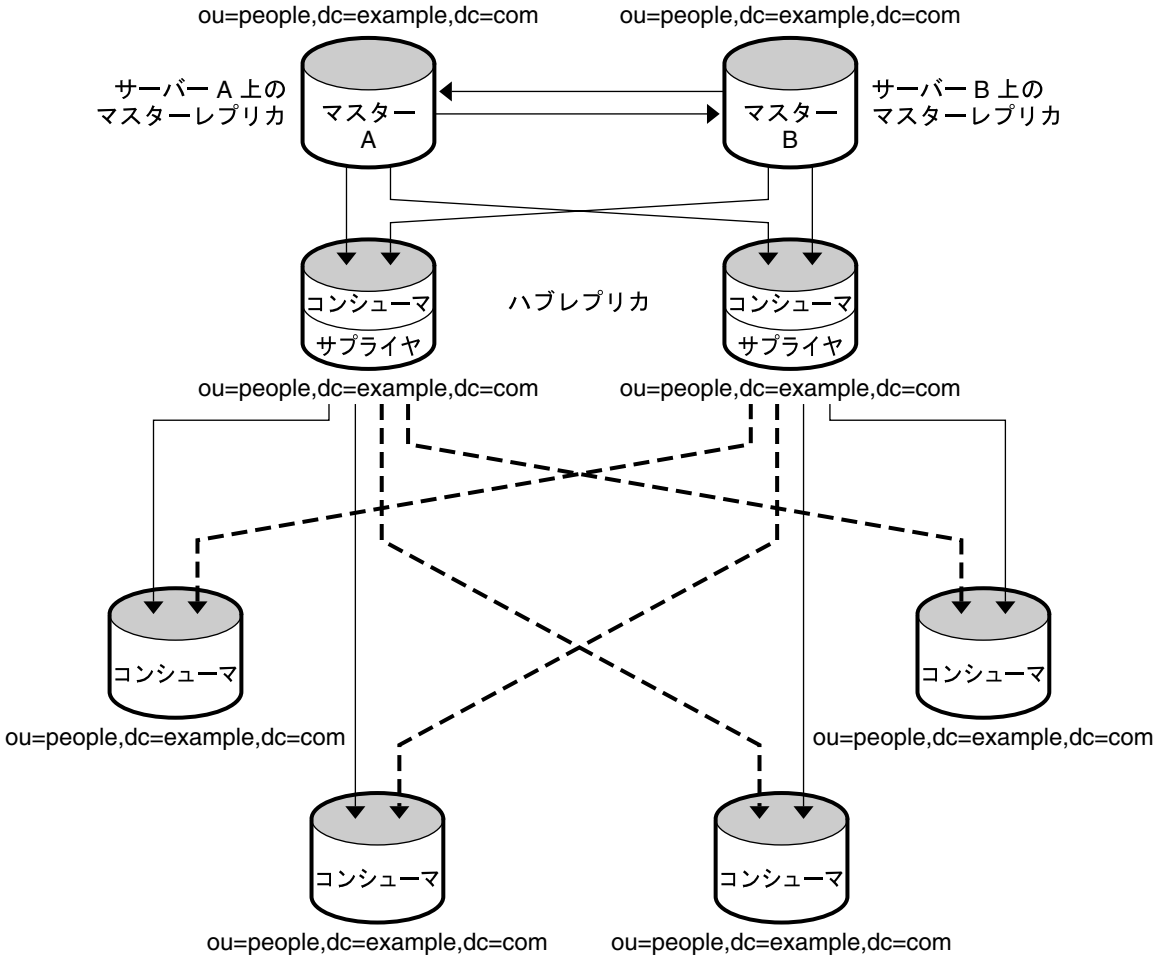


図 6-8 では、2 つのマスターと 2 つのハブが 4 つのコンシューマにデータをレプリケートしています。前の例のように、マスターとハブの間でハブを共有することで、レプリケーション更新の負荷のバランスをとります。

この例に示される点線は、無効化されたレプリケーションアグリーメントを示しています。これらのレプリケーションアグリーメントを有効化しない場合、いずれかのハブがオフラインになった場合にこのトポロジにはシングルポイント障害が発生します。このテクノロジを配備する際は、パフォーマンスと高可用性の要件を比較考慮して、すべてのレプリケーションアグリーメントを有効化して完全な読み書き対応フェイルオーバーを提供するかどうかを決定する必要があります。

部分レプリケーション

レプリケーションの単位はサフィックスまたはサブサフィックスですが、部分レプリケーション機能はレプリケーションを大幅に細分化します。部分レプリケーションでは、サフィックスまたはサブサフィックスのすべてのエントリの属性のサブセットをレプリケートできます。

部分レプリケーションの利点

部分レプリケーションは、さまざまな状況で使用できます。

イントラネットサーバーとエクストラネットサーバーの間で同期が必要で、セキュリティ上の理由から一部のコンテンツをフィルタリングしなければならない場合に、部分レプリケーションがフィルタリング機能を提供します。

部分レプリケーションではレプリケートするものを選択できるため、レプリケーションの費用を削減することができます。すべての場所で特定の属性を利用できるようにすることだけが必要な配備では、すべての属性をレプリケートする代わりに、部分レプリケーション機能を使用して必要な属性だけをレプリケートできます。

たとえば、ほかの属性が頻繁に変更され、ネットワークトラフィックの負荷が非常に大きくなる場合は、ユーザーエントリのすべての属性をレプリケートするのではなく、電子メールアドレスと電話番号の属性だけをレプリケートすることもできます。部分レプリケーションを使用することで、必要な属性をフィルタリングし、トラフィックを最小限に抑えることができます。このフィルタリング機能は、WAN を介したレプリケーションでは特に有効です。

部分レプリケーションは、Directory Server 5.2 より前のバージョンの Directory Server との下位互換性はありません。部分レプリケーションを使用する場合は、Directory Server のほかのインスタンスのすべてが Directory Server 5.2 インスタンスである必要があります。

部分レプリケーションの設定

部分レプリケーションは、Directory Server コンソールから簡単に設定することができます。部分レプリケーションは、次のいずれかの方法で設定します。

- レプリケーションに含められる属性のリストを指定する。

- すべての属性を選択し、そのうちの除外されるものを指定する。

ほとんどの場合、推奨される方法は排他的な設定アプローチです。ACI、CoS、ロールなど、一部の機能の複雑さ、および特定の属性に対するこれらの機能の依存性を考えると、対象から除外する属性のリストを管理するほうが、対象に含める属性のリストを管理するより安全であり、人的エラーの可能性が少なくなります。

部分レプリケーションを設定するときは、レプリケートされるサーバーが読み取り専用レプリカである必要があります。

一般には、スキーマ違反を回避するために、スキーマに定義されている各エントリのすべての必須属性をレプリケートします。部分レプリケーションを使用して一部の必須属性をフィルタリングする場合は、スキーマ検査を無効にする必要があります。スキーマ検査で部分レプリケーションが有効になっていると、サーバーを LDIF ファイルからオフラインで初期化できなくなる可能性があります。これは、必須属性をフィルタリングすると、サーバーが LDIF ファイルを読み込めなくなるためです。部分コンシューマレプリカでスキーマ検査を無効にした場合、その部分コンシューマレプリカが存在するサーバーインスタンス全体にスキーマが適用されなくなります。部分レプリケーションの設定ではサプライヤがスキーマをプッシュするため、部分コンシューマレプリカのスキーマは、マスターレプリカのスキーマのコピーとなります。このため、適用される部分レプリケーション設定には対応しません。

部分レプリケーション設定を変更する前に、それによって影響を受けるレプリケーションアグリーメントを無効にする必要があります。設定の変更が完了したら、レプリケーションアグリーメントを再度有効化し、新しい設定が適用されるようにコンシューマを再初期化します。

詳細は、『Directory Server 管理ガイド』の「部分レプリケーションの設定」を参照してください。

レプリケーション戦略の定義

使用するレプリケーション手法は、提供するサービスによって異なります。この節では、次の内容を詳しく示すレプリケーショントポロジの例を示します。

- 高可用性を実現するためのレプリケーションの使用
- ローカルでのデータの可用性を高めるためのレプリケーションの使用
- ロードバランスのためのレプリケーションの使用

配備においてこれらの側面のそれぞれがどの程度重要かを考えるには、ネットワーク、ユーザー、クライアントアプリケーション、およびそれらでディレクトリサービスを使用する方法の調査を行うことから始めてください。この調査のガイドラインについては、139 ページの「レプリケーション調査の実施」を参照してください。

レプリケーション手法を決定したら、Directory Server の配備を開始できます。ディレクトリを実際の環境に配備する段階に入ると、ディレクトリを全体的に配備した際の負荷をより正確に把握できるようになります。実際に運用されているディレクトリに基づいた負荷分析を行うことができない場合は、ディレクトリの使用状況をよく把握して、ディレクトリを修正するために準備してください。

次に、レプリケーション手法の決定に影響する要因について説明します。

- レプリケーション調査の実施
- レプリケーションリソースの要件
- レプリケーションの下位互換性
- 高可用性を実現するためのレプリケーションの使用
- ローカルでのデータの可用性を高めるためのレプリケーションの使用
- ロードバランスのためのレプリケーションの使用
- 小規模サイト向けのレプリケーション手法の例
- 大規模サイト向けのレプリケーション手法の例

レプリケーション調査の実施

レプリケーション調査を実施するときは、次の情報を収集することに専念してください。

- 異なる建物間やリモートサイト間を接続するネットワークの品質と使用可能な帯域幅。
- ユーザーの物理的な位置、各サイトのユーザー数、ディレクトリ上の起こりうるユーザーのアクティビティ。

  たとえば、人事データベースや財務情報を管理するサイトは、ディレクトリを単に電話帳として使用する技術スタッフを含むサイトより、ディレクトリに大きな負荷をかけます。
- ディレクトリにアクセスするアプリケーションの数と、書き込み操作に対する読み取り、検索、および比較の操作の比率。

  たとえば、メッセージングサーバーがディレクトリを使用する場合は、処理するメールメッセージごとにディレクトリが実行する操作数を調べる必要があります。ディレクトリを使用するその他の認証アプリケーションには、META Directory アプリケーションなどがあります。アプリケーションごとに、ディレクトリで実行される操作のタイプと頻度を調べる必要があります。
- ディレクトリに格納するエントリのおおよその数とサイズ。

レプリケーションリソースの要件

レプリケーション機能は、システムリソースを必要とします。レプリケーション手法を決定するときは、次のリソース要件を検討してください。

- ディスク使用量

  サプライヤでは、各更新操作後に更新履歴ログが書き込まれます。サプライヤが複数のレプリケートされたサフィックスを保持する場合、更新履歴ログはより頻繁に更新されるため、ディスク使用量はさらに増えます。

  ボトルネックを回避するため、コンシューマのマシンの規模は、少なくともサプライヤと同等である必要があります。

- サーバースレッド

  各レプリケーションアグリーメントは、2 つの追加スレッドを作成します。レプリケーションアグリーメントのスレッドは、操作スレッドとは分けられます。このため、いくつかのレプリケーションアグリーメントが存在する場合、クライアントアプリケーションで使用できるスレッドの数が少なくなり、クライアントアプリケーションに対するサーバーのパフォーマンスに影響が生じる可能性があります。

- ファイルディスクリプタ

  サーバーで使用できるファイルディスクリプタの数が、更新履歴ログ (1 ファイルディスクリプタが必要) と各レプリケーションアグリーメント (アグリーメントごとに 1 ファイルディスクリプタが必要) によって減少します。

レプリケーションの下位互換性

レプリケーショントポロジで複数のバージョンの Directory Server を使用する場合は、表 6-1 の下位互換性に関する情報を考慮する必要があります。この表は、ざまざまなバージョンの Directory Server 間で考えられるサプライヤとコンシューマの組み合わせを示しています。

表 6-1　さまざまなバージョンの Directory Server でのレプリケーションの下位互換性

	4.x コンシューマ	5.0/5.1 コンシューマ	5.0/5.1 マスター	5.2 コンシューマ	5.2 マスター	5.x ハブサプライヤ
4.x マスター	はい	はい	はい	はい	はい	いいえ
5.0/5.1 マスター	いいえ	はい	はい	はい	はい	はい
5.2 マスター	いいえ	はい	はい	はい	はい	はい

さまざまなバージョンの Directory Server でレプリケーションを使用する場合は、次の点に注意してください。

- 4.x マスターから 5.x マスターにレプリケートする設定で、5.x マスター上で旧バージョンのレプリケーションを有効にする場合、5.x マスターは、トポロジ内のその他の 5.x マスターからクライアント更新とレプリケーション更新を受信できなくなります。この場合、4.x マスターからのレプリケーション更新だけが受信可能です。ただし、旧バージョンのレプリケーションを無効にすると、5.x マスターのマスターレプリケーションとしての動作は完全に復元されます。
- Directory Server 5.2 を実行しているサーバーから Directory Server 5.0/5.1 を実行しているサーバーにレプリケートするときは、Directory Server 5.2 の新しい機能や拡張機能は使用しないでください。これらの機能には、部分レプリケーション、複数パスワードポリシー、WAN を解したマルチマスターレプリケーション、オンライン昇格および降格があります。
- Directory Server 5.2 スキーマ拡張によって 5.1 サーバーが混乱することがないように、`nsslapd-schema-replicate-useronly` 属性は `on` に設定する必要があります。

高可用性を実現するためのレプリケーションの使用

レプリケーションを使用して、単一のサーバーの障害によってディレクトリが使用できなくなることを防止することができます。最低限、ローカルのディレクトリツリーを少なくとも 1 つのバックアップサーバーにレプリケートしておきます。

ディレクトリの設計者の中には、データの信頼性を最大限保証するために物理的な場所ごとに 3 回はレプリケートしておく必要があると主張する人もいます。しかし、耐障害性を目的としたレプリケーションの使用頻度はユーザーの決定事項ですが、その決定はディレクトリで使用するハードウェアとネットワークの品質を考慮したものでなければなりません。信頼性の低いハードウェアでは、より多くのバックアップサーバーが必要になります。

注	通常のデータバックアップポリシー代わりにレプリケーションを使用することはできません。ディレクトリデータのバックアップについては、215 ページの「バックアップ方法の選択」および『Directory Server 管理ガイド』の「データのバックアップ」を参照してください。

ディレクトリクライアントに書き込みフェイルオーバーを保証するには、マルチマスターレプリケーションのトポロジを使用する必要があります。読み取りフェイルオーバーが十分で、ディレクトリが地理的に分散されていない場合は、シングルマスターレプリケーションを使用できます。

LDAP クライアントアプリケーションは通常、1 つの LDAP サーバーだけを検索するように設定します。つまり、カスタムクライアントアプリケーションを異なる DNS ホスト名にある LDAP サーバーを循環するように作成していない限り、LDAP クライアントアプリケーションが Directory Server の単一の DNS ホスト名を検索するように設定するだけで済みます。したがって、バックアップ用の Directory Server にフェイルオーバーを提供するには、DNS ラウンドロビンまたはネットワークソートのどちらかを使用する必要があります。DNS ラウンドロビンとネットワークソートの設定方法と使用方法については、DNS のマニュアルを参照してください。

地理的に離れた 2 つの場所にまたがって書き込みフェイルオーバーを維持するには、WAN を介した 4 方向のマルチマスターレプリケーションを使用します。この場合、1 つの場所に 2 つのマスターサーバーを設定し、もう 1 つの場所にも 2 つのマスターサーバーを設定して、それらが WAN を介して完全にメッシュ化されるように設定します。これが、1 つのマスターがオフラインになった場合の安全対策になります。

Sun Java System Directory Proxy Server 製品を代わりに使用することもできます。Directory Proxy Server については、http://www.sun.com/software/products/directory_proxy/home_dir_proxy.html を参照してください。

ローカルでのデータの可用性を高めるためのレプリケーションの使用

次の場合、ローカルでもデータを利用するためにレプリケーションを使用することができます。

- データのローカルマスターコピーが必要である。

  これは、特定の地理的な場所のユーザーだけにとって重要なディレクトリ情報を管理する必要がある、大規模な国際企業にとって重要です。また、データのローカルマスターコピーを保有することは、データを部門レベルまたは組織レベルで管理するようにしている企業にとっても便利です。

- 信頼性が低いネットワーク接続、または断続的なネットワーク接続を使用している。

  データのコピーをローカルで保有することは、次のようなネットワークの問題が発生しても、ディレクトリをそのまま使用できることを意味します。

- ネットワークに過度な負担が定期的にかかり、ディレクトリのパフォーマンスが低下する。

  旧式のネットワークを使用している企業では、通常の営業時間帯にこのような状態が発生します。

- マスターレプリカでのネットワーク負荷と作業負荷を両方とも軽減する。

  ネットワークの信頼性と可用性に問題がない場合でも、ネットワーク費用の削減が求められることがあります。

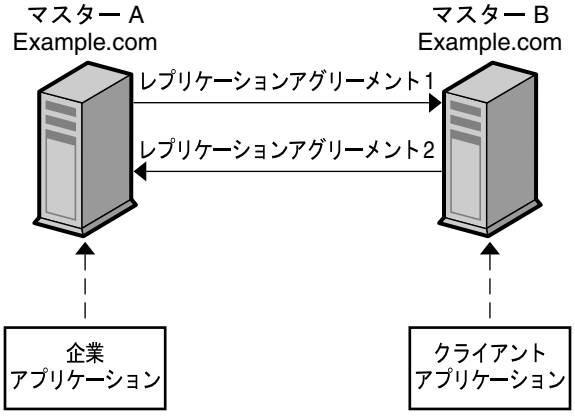
ロードバランスのためのレプリケーションの使用

レプリケーションを使用すると、次のような方法で Directory Server にかかる負荷のバランスをとることができます。

- ユーザーの検索アクティビティを複数のサーバーに分散する。
- 書き込みはマスターレプリカを保持するサーバーだけに限定し、その他のサーバーは読み取り専用に設定する。
- メールサーバーのように、特定のタスクに専用サーバーを割り当てる。

図 6-9 は、レプリケーションを使用してさまざまなアプリケーション間でディレクトリのアクティビティをどのように分割し、それによって各サプライヤサーバーにかかる負荷を削減できるかを示しています。

図 6-9 ロードバランスのためのマルチマスターレプリケーションの使用



ディレクトリデータをレプリケートすることで、ネットワークにかかる負荷も調整されます。可能であれば、高速で信頼性の高いネットワーク接続を介してアクセス可能なサーバーにデータを移動します。

通常、ディレクトリエントリの平均サイズは約 1K バイトです。したがって、ディレクトリの検索ごとにネットワークの負荷が約 1K バイト増加します。ディレクトリユーザーが 1 日当たり約 10 回検索を実行すると、ネットワークの負荷はユーザー 1 人につき 1 日当たり約 10,000 バイト増加します。低速な、負荷が大きい、あるいは信頼性が低い WAN を使用している場合は、ディレクトリツリーをローカルサーバーにレプリケートする必要がある場合もあります。

ローカルで使用できるデータの利点は、レプリケーションによって増加するネットワークトラフィックの費用と比較して考慮する必要があることに注意してください。たとえば、ディレクトリツリー全体をリモートサイトにレプリケーションする場合は、ユーザーのディレクトリの検索によるトラフィックに比べ、はるかに大きな負荷をネットワークに課すことになります。このことは、ディレクトリツリーが頻繁に変更されるのに対して、リモートサイトの少数のユーザーが、1 日当たり数回のディレクトリ検索しか実行しない場合は、特に当てはまります。

ディレクトリツリーに平均して 1,000,000 件を超えるエントリがあり、毎日 10 % 前後が変更される場合を考えてみます。ディレクトリエントリのサイズが平均して 1K バイトとすると、ネットワークの負荷が 1 日当たり 100M バイト増えることになります。しかし、リモートサイトの従業員がたった 100 人で、1 日当たり平均して 10 回のディレクトリ検索を実行している場合は、ディレクトリアクセスによるネットワークの負荷は 1 日当たり 1M バイトにすぎません。

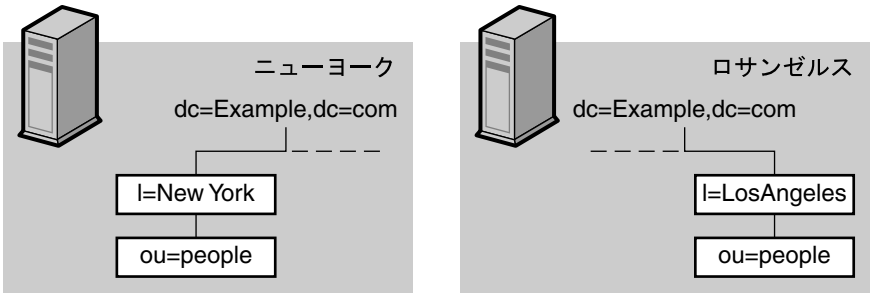
レプリケーションによるネットワークの負荷と通常のディレクトリの使用による負荷の違いから、ネットワークのロードバランスを目的とするレプリケーションは実現可能ではないという結論に達する場合もあります。反対に、ネットワークにかかる負荷を考慮しても、ディレクトリデータをローカルで利用できる利点の方が勝っていると判断する場合もあります。

ネットワークに過度な負荷をかけずにデータをローカルサイトで使用できるようにするには、スケジュールされたレプリケーションを使用します。データの整合性とレプリケーションスケジュールについては、124 ページの「データの整合性」を参照してください。

ネットワークのロードバランスの例

2 つの都市に事務所を持つ企業を考えてみます。図 6-10 に示すように、それぞれの事務所は別々のサブツリーを管理します。

図 6-10　ニューヨークとロサンゼルスでそれぞれにあるサブツリー ニューヨークとロサンゼルスの 2 つのデータセンター



各事務所には高速の LAN がありますが、2 都市間の通信にはダイヤルアップ接続を使用しています。このような場合は、以下のようにネットワークの負荷のバランスをとります。

- 事務所ごとに、ローカルで管理するデータのマスターサーバーとなるサーバーを 1 つ選択する。

  ローカルで管理するデータを、選択したサーバーからリモートオフィスのマスターにレプリケートします。それぞれの地域でマスターコピーを保持することで、ユーザーはダイヤルアップ接続経由で更新と検索を行う必要がなくなり、パフォーマンスが最適化されます。

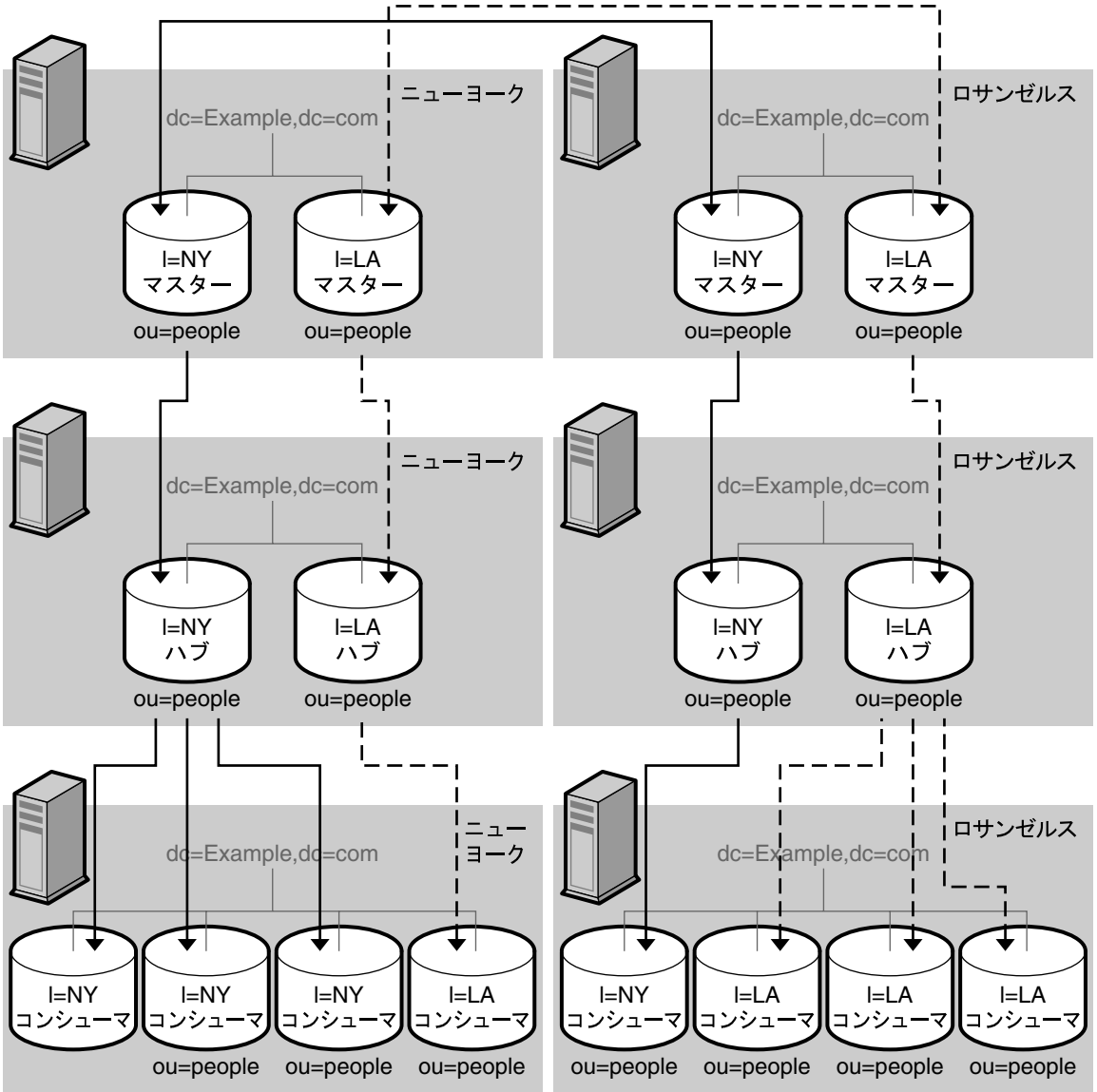
- ディレクトリデータのローカルでの可用性を保証するために、リモートオフィスからのデータも含め、各マスター上のディレクトリツリーを少なくとも 1 つのローカル Directory Server にレプリケートする。

- それぞれの地域でカスケード型レプリケーションを設定し、ローカルデータに対する検索に特化したコンシューマの数を増やすことで、ロードバランスをさらに進める。

  ニューヨーク事務所では、ロサンゼルスに関する検索よりもニューヨークに関する検索のほうが多く行われるため、この例では、ニューヨーク事務所に 3 つのニューヨークデータコンシューマと、1 つのロサンゼルスデータコンシューマが設定されています。同様に、ロサンゼルス事務所には 3 つのロサンゼルスデータコンシューマと 1 つのニューヨークデータコンシューマがあります。

図 6-11 は、このネットワークロードバランスの設定を示しています。

図 6-11　マルチマスターレプリケーションとカスケード型レプリケーションによるロードバランス



——— ニューヨークのデータレプリケーションの流れ

– – – ロサンゼルスのデータレプリケーションの流れ

パフォーマンス向上のためのロードバランスの例

この例では、ディレクトリには 15,000,000 のエントリが格納されて 10,000,000 のユーザーがアクセスし、各ユーザーが 1 日当たり 10 件のディレクトリ検索を実行します。メッセージングサーバーは 1 日当たり 250,000,000 通のメールメッセージを処理し、メールメッセージごとに 5 件のディレクトリ検索を実行します。メール処理に 1 日当たり 1,250,000,000 件のディレクトリ検索が実行されます。ディレクトリとメッセージングシステムの合計トラフィックでは 1 日当たり 1,350,000,000 件のディレクトリ検索が実行されることになります。

1 日の就業時間は 8 時間、ディレクトリユーザーが 4 つの時間帯に分かれているとすると、4 つの時間帯にわたる就業時間 (または、ピーク時の利用率) は 12 時間となります。したがって、ディレクトリは、1 日 12 時間で 1,350,000,000 件の検索をサポートする必要があります。これは毎秒 31,250 (1,350,000,000/(60 × 60 × 12)) 件の検索をサポートするのと同じです。つまり、次のようになります。

10,000,000 人のユーザー	1 ユーザーにつき 10 件の検索 =	100,000,000 件の読み取り / 日
250,000,000 通のメッセージ	1 メッセージにつき 5 件の検索 =	1,250,000,000 件の読み取り / 日
	合計読み取り / 日 =	1,350,000,000
12 時間は 43,200 秒	合計読み取り / 秒 =	31,250

ディレクトリで毎秒 5,000 件の読み取りをサポートできる CPU と RAM の組み合わせを考えてみます。簡単な割り算で、この負荷をサポートするには 6 ～ 7 つの Directory Server が必要であることがわかります。10,000,000 人のディレクトリユーザーを有する企業の場合は、ローカルでデータを利用するという目的も考慮するとさらに Directory Server を追加する必要があります。

Directory Server 5.2 では、適切なハードウェアと設定により、1 台のサーバーで、1 秒あたり 5,000 回を超える読み取り処理の継続が可能になります。

この場合、次のようなレプリケートを行います。

- 1 つの都市に 2 つの Directory Server をマルチマスター設定で配置し、すべての書き込みトラフィックを処理する。

  この設定は、すべてのディレクトリデータを 1 か所で管理することを想定しています。

- 上記のマスターを使用して 1 つまたは複数のハブにレプリケートする。

  ディレクトリがサービスする読み取り、検索、および比較の要求はコンシューマで処理されるので、マスターは書き込み要求の処理に専念できます。詳細は、132 ページの「カスケード型レプリケーション」を参照してください。

- ハブを使用して、会社全体のローカルサイトにレプリケートする。

  ローカルサイトにレプリケートすることにより、サーバーや WAN の負荷のバランスをとり、ディレクトリデータの可用性を高めることができます。

- 各サイトで、少なくとも 1 回はレプリケートして可用性を高める。最低でも読み取り操作を実行する。

  DNS ソートを使用して、ユーザーがディレクトリ検索に使用できるローカルの Directory Server を必ず見つけられるようにします。

小規模サイト向けのレプリケーション手法の例

会社全体が 1 つの建物内にある場合を考えてみます。この建物には、毎秒 100M バイトの高速で使いやすいネットワークが装備されています。ネットワークは安定しており、サーバーのハードウェアと OS プラットフォームの信頼性も十分に高いものとします。また、1 つのサーバーの処理能力でサイトの負荷を容易に処理できるものとします。

このような場合、保守やハードウェアのアップグレードのためにプライマリサーバーが停止されたときでも、ディレクトリデータが利用できるようにしておくために、少なくとも 1 回はレプリケートしておきます。また、Directory Server の 1 つが使用できなくなったときに LDAP 接続のパフォーマンスを上げるために、DNS ラウンドロビンを設定します。代替方法として、Sun Java System Directory Proxy Server などの LDAP プロキシを使用することもできます。Directory Proxy Server については、http://www.sun.com/software/products/directory_proxy/home_dir_proxy.html を参照してください。

大規模サイト向けのレプリケーション手法の例

会社が 2 つの建物に分かれている場合を考えてみます。それぞれの建物には、毎秒 100M バイトの高速で使いやすいネットワークが装備されています。ネットワークは安定しており、サーバーのハードウェアと OS プラットフォームの信頼性も十分に高いものとします。また、1 つのサーバーの処理能力で、各建物内のサーバーにかかる負荷を容易に処理できるものとします。

建物間は低速接続 (ISDN) で、通常の営業時間中に大きな負荷がかかります。

この例での典型的なレプリケーション手法は、次のとおりです。

- いずれかの建物で、ディレクトリデータのマスターコピーを格納する 1 つのサーバーを選択する。

  このサーバーは、データのマスターコピーの管理に責任を持つ人がもっとも多くいる建物内に設置するようにします。これを建物 A とします。

- データが常に利用できるようにするために、建物 A で少なくとも 1 回はレプリケートする。

  書き込みフェイルオーバーを保証するには、マルチマスターレプリケーションを使用します。

- もう一方の建物 (建物 B) 内に 2 つのレプリカを作成する。
- データのマスターコピーとレプリケートされたコピーの間で厳密な整合性が必要ない場合は、ピーク時間外にレプリケーションが行われるようにスケジュールする。

大規模な国際企業のレプリケーション戦略

企業の 2 つのデータセンターがフランスと米国にあり、WAN によって分割されていると仮定します。WAN 経由によるレプリケーションが必要なだけでなく、パートナー企業にすべてのデータを開示しないために、一部のデータをフィルタリングしなければなりません。通常業務時間内は、使用するネットワークはとても混み合っています。

この例での典型的なレプリケーション手法は、次のとおりです。

- ディレクトリデータのマスターコピーを、両方のデータセンターのそれぞれのサーバーで保持する。
- フランスサイト内と米国サイト内の書き込みフェイルオーバーのために、各データセンターの第 2 のマスターにデータをレプリケートする。
- フランスと米国の間に完全にメッシュ化された 4 方向のマルチマスターレプリケーショントポロジを配備し、配備全体で高可用性と書き込みフェイルオーバーを実現する。
- 各データセンターで必要なだけのコンシューマを配備し、ディレクトリ検索の面でマスターへの負荷を軽減する。
- 両地域のマスターとコンシューマの間に部分レプリケーションアグリーメントを設定し、パートナー企業にアクセスさせたくないデータをフィルタリングする。
- 帯域幅を最適化できるように、混雑していない時間帯に実行されるようにレプリケーションをスケジュールする。

レプリケーションとほかのディレクトリ機能との併用

レプリケーションは Directory Server のほかの機能と連携して、高度なレプリケーション機能を提供します。次に、最適なレプリケーションの設計に役立つ、機能の併用について説明します。

レプリケーションとアクセス制御

ディレクトリはエントリの属性として ACI を格納しています。つまり、ACI はほかのディレクトリ内容と一緒にレプリケートされます。Directory Server は ACI をローカルに評価するため、これは重要です。

ディレクトリのアクセス制御の設計方法については、第 7 章「アクセス制御、認証、および暗号化」を参照してください。

レプリケーションと旧バージョン対応更新履歴ログプラグイン

旧バージョン対応更新履歴ログは、アプリケーションと Directory Server 4.x バージョンとの互換性を保つために LDAP クライアントが使用するプラグインです。旧バージョン形式の履歴更新ログは、Directory Server 更新ログとは別のデータベースの、サフィックス `cn=changelog` の下に格納されます。

旧バージョン対応更新履歴ログは、スタンドアロンのサーバーまたはレプリケーショントポロジの各サーバーで有効化できます。サーバーで旧バージョン対応更新履歴ログが有効になると、そのサーバーのすべてのサフィックスに対する更新がデフォルトでログに記録されます。

Directory Server 5.2 2005Q1 より前のバージョンの Directory Server では、旧バージョン対応更新履歴ログがマルチマスタートポロジの各レプリカへの更新順序を認識しませんでした。そのため、旧バージョン対応更新履歴ログは、マルチマスターレプリケーション環境では使用できませんでした。

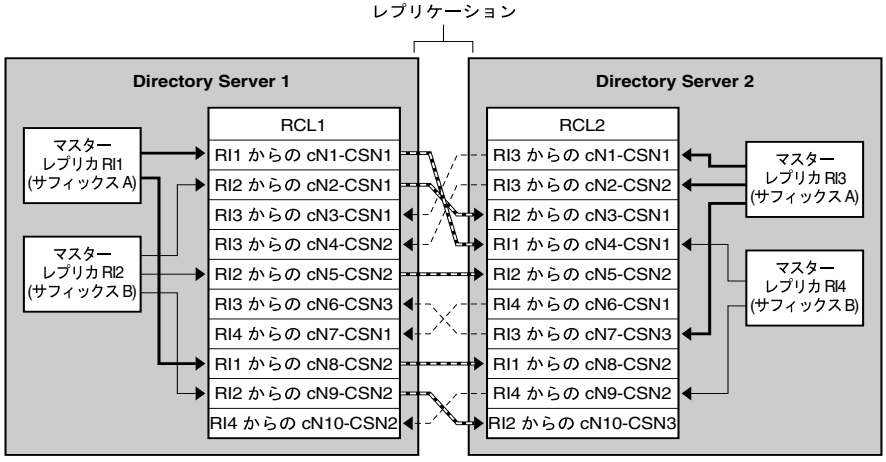
Directory Server 5.2 2005Q1 では、旧バージョン対応更新履歴ログが各レプリカ識別子に対して行われた更新の順序を識別します。現在では、旧バージョン対応更新履歴ログをマルチマスターレプリケーション環境で使用できるようになっています。旧バージョン対応更新履歴ログの使用に関する制限事項については、154 ページの「旧バージョン対応更新履歴ログの制限事項」を参照してください。

旧バージョン対応更新履歴ログの使用方法については、『Directory Server 管理ガイド』の「旧バージョン対応更新履歴ログプラグインの使用」を参照してください。旧バージョン対応更新履歴ログプラグインで使用される属性については、『Directory Server Administration Reference』の「Server Configuration Reference」を参照してください。

旧バージョン対応更新履歴ログおよびマルチマスターレプリケーション

旧バージョン対応更新履歴ログがレプリケーションで有効になると、旧バージョン対応更新履歴ログはトポロジのすべてのマスターレプリカからの更新を受け取ります。各レプリカからの更新は旧バージョン対応更新履歴ログで結合されます。次の図に、マルチマスタートポロジでの 2 つのサーバー間の旧バージョン対応更新履歴ログを示します。

図 6-12 旧バージョン対応更新履歴ログおよびマルチマスターレプリケーション



旧バージョン対応更新履歴ログでは、レプリケーション中に次の属性が使用されます。

- replicaIdentifier (RI) - 旧バージョン対応更新履歴ログを更新しているレプリカを識別する
- changeNumber (cN) - 旧バージョン対応更新履歴ログに記録される更新の順序を識別する
- replicationCSN (CSN) - 特定のレプリカに更新が行われる時刻を識別する

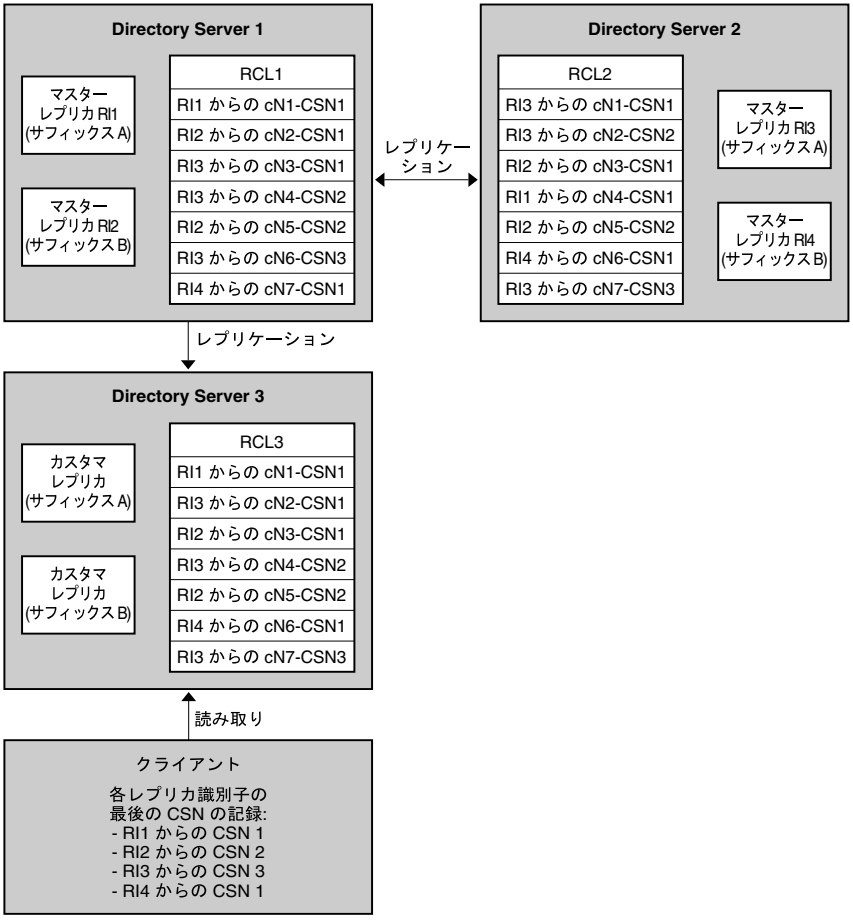
旧バージョン対応更新履歴ログのほかの属性および replicationCSN 属性については、『Directory Server Administration Reference』を参照してください。

図 6-12 は、旧バージョン対応更新履歴ログ、RCL1、および RCL2 には同じ更新リストが含まれていますが、更新が同じ順序で行われないことを示しています。ただし、特定の replicaIdentifier については、更新が同じ順序で各旧バージョン対応更新履歴ログに記録されます。更新が旧バージョン対応更新履歴ログに記録される順序は、changeNumber 属性 (cN) によって指定されます。

旧バージョン対応更新履歴ログのフェイルオーバー

図 6-13 は、クライアントがコンシューマサーバーの旧バージョン対応更新履歴ログを読み取る簡単なレプリケーショントポロジを示しています。

図 6-13 旧バージョン対応更新履歴ログのレプリケーションの簡単なトポロジ

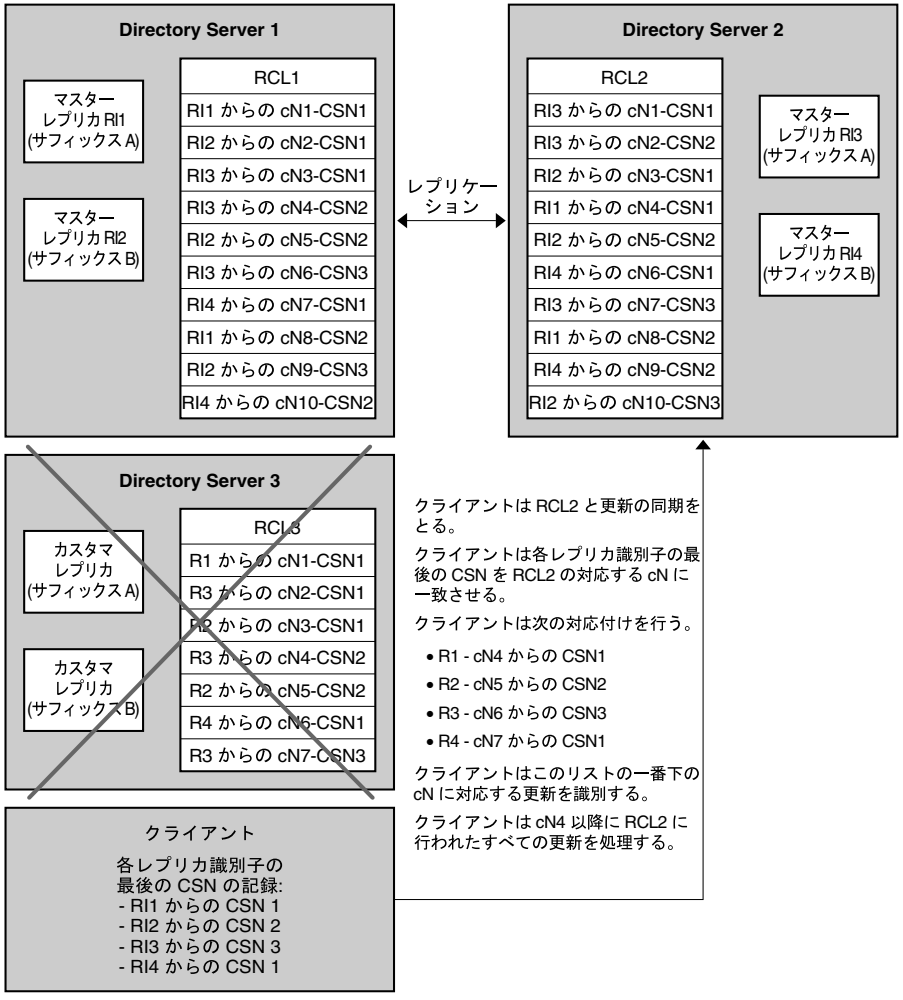


トポロジ内の各マスターレプリカに対して行われるすべての更新は、トポロジ内の各旧バージョン対応更新履歴ログに記録されます。

クライアントアプリケーションは、Directory Server 3 の旧バージョン対応更新履歴ログを読み取り、各レプリカ識別子の最後の CSN を格納します。各レプリカ識別子の最後の CSN は、`replicationCSN` 属性によって指定されます。

次の図は、Directory Server 3 に障害が発生したあと、クライアントが読み取りを Directory Server 2 に転送している状態を示しています。

図 6-14　旧バージョン対応更新履歴ログのフェイルオーバー



フェイルオーバー後は、クライアントアプリケーションは Directory Server 2 の旧バージョン対応更新履歴ログ (RCL2) を使用して更新を管理する必要があります。RCL2 の更新順序は RCL3 での順序と同じではないため、クライアントは RCL2 と更新の同期をとる必要があります。

クライントは RCL2 を検査して、各レプリカ識別子の最後の CSN の記録に対応する cN を識別します。図 6-14 の例では、最後の CSN と cN が次のように対応していることをクライアントが識別しています。

- R1 からの CSN 1 が RCL2 の cN4 に対応する
- R2 からの CSN 2 が RCL2 の cN5 に対応する
- R3 からの CSN 3 が RCL2 の cN6 に対応する
- R4 からの CSN 1 が RCL2 の cN7 に対応する

クライアントはこのリストの一番下の cN に対応する更新を識別します。図 6-14 の例では、リストの一番下の cN は cN4 です。クライアントが確実にすべての更新を処理するためには、cN4 以降に RCL2 にログを記録されたすべての更新を処理する必要があります。クライアントは cN4 より以前に RCL2 にログを記録された更新は処理しません。また cN4 に対応する更新も処理しません。

旧バージョン対応更新履歴ログの制限事項

旧バージョン対応更新履歴ログを使用する場合は、次の制限事項に従ってください。

- Directory Server 5.2 を実行しているマスターレプリカを、Directory Server 4.x を実行しているコンシューマレプリカのサプライヤにすることはできません。ただし、Directory Server 4.x を実行しているマスターレプリカを、Directory Server 5.2 を実行しているコンシューマレプリカのサプライヤにすることはできます。
- レプリケーショントポロジでは、更新の競合がある場合、旧バージョン対応更新履歴ログ間の切り換えは機能しません。競合は、特定のエントリを 1 つのマスターだけで変更することにより防止できます。
- 2 つのサーバー間で更新が送信された場合、通常、各変更に関連するタイムスタンプを使用することにより更新の競合は解決できます。旧バージョン対応更新履歴ログで複数のエントリが同じ CSN を使用する場合、変更が競合する可能性があります。
- レプリケーショントポロジでは、レプリケートされるサーバー間の旧バージョン対応更新履歴ログは最新の状態にする必要があります。こうすることにより、旧バージョン対応更新履歴ログの切り換えが可能になります。図 6-14 の例では、RCL3 の各レプリカ ID の最後の CSN が RCL2 に存在する必要があります。

レプリケーションと参照整合性プラグイン

参照整合性プラグインがすべてのマスターレプリカで有効になっている場合は、プラグインとマルチマスターレプリケーションを一緒に使用できます。参照整合性プラグインは、デフォルトでは無効になっているため、Directory Server コンソールまたはコマンド行を使用して有効化する必要があります。

整合性チェックにはメモリと CPU が多大に消費されるので、変更要求を送信するサーバーで参照整合性プラグインを有効化する前に、パフォーマンスリソース、時間、整合性の必要性を分析してください。

詳細は、『Directory Server 管理ガイド』の「レプリケーションにおける参照整合性の使用」を参照してください。

レプリケーションと操作前、操作後プラグイン

操作前プラグインと操作後プラグインを作成するときは、これらのプラグインがレプリケーション操作を無視するように指定することができます。ほとんどの場合は、これがプラグインの目的の動作である可能性があります。レプリケーション操作を変更すると予期しない動作が起きる可能性があることに注意してください。

詳細は、『Directory Server Plug-In Developer's Guide』の「Pre-Operation and Post-Operation Plug-Ins」を参照してください。

レプリケーションと連鎖サフィックス

連鎖を使用してエントリを分散する場合は、連鎖サフィックスを保持するサーバーが、実際のデータを含むリモートサーバーを指します。このリモートサーバーは、ファームサーバーとも呼ばれています。このような環境では、連鎖サフィックス自体をレプリケートすることはできません。ただし、実際のデータを格納しているサフィックスのレプリケートは可能です。連鎖サフィックスを含むサーバー上ではなく、リモートサーバー上のレプリケーションアグリーメントを設定する必要があります。

レプリケーションを連鎖サフィックスのバックアップとしては使用しないでください。連鎖サフィックスは手動でバックアップする必要があります。連鎖とエントリの分散については、102 ページの「リフェラルと連鎖」を参照してください。

スキーマのレプリケーション

レプリケーション環境で Directory Server を使用する場合は、レプリケーションに含まれるすべてのサーバーについてスキーマの一貫性を維持する必要があります。サーバー間でスキーマの一貫性が維持されない場合は、レプリケーションでエラーが発生する可能性があります。

スキーマの一貫性を確保するために、マルチマスターレプリケーショントポロジの場合でも、1 つのマスターでスキーマの変更を行うことをお勧めします。スキーマの変更についての競合解決はありません。したがって、マルチマスタートポロジの 2 つのマスターサーバーでスキーマに変更を加えた場合、最後に変更されたマスターがそのスキーマをコンシューマに伝達します。つまり、あとから加えた変更が、もう一方のマスターに加えた変更と異なる場合は、その内容が失われる危険があります。

コンシューマ上のスキーマは絶対に更新しないでください。コンシューマ上のスキーマを更新し、その結果としてサプライヤ上のスキーマのバージョンがコンシューマ上のスキーマのバージョンより古くなった場合は、コンシューマを検索したりサプライヤを更新しようとするとエラーが発生します。

スキーマのレプリケーションは自動的に行われます。サプライヤとコンシューマ間でレプリケーションが設定されている場合は、デフォルトでスキーマがレプリケートされます。

Directory Server でスキーマのレプリケーションに使用されるロジックは、次のように説明できます。

1. データをコンシューマにプッシュする前に、サプライヤは自身のスキーマが、コンシューマ上で保持されているバージョンのスキーマと同期しているかどうかを検査します。
2. サプライヤとコンシューマ両方のスキーマエントリが同じであれば、レプリケーションが実行されます。
3. サプライヤ上のスキーマがコンシューマに格納されているものよりも新しい場合、サプライヤはデータのレプリケーションを実行する前に自身のスキーマをコンシューマにレプリケートします。

注　スキーマに含まれる ACI は、レプリケートされます。

カスタムスキーマファイルへの変更は、スキーマが LDAP または Directory Server コンソールを使用して更新されている場合にのみレプリケートされます。カスタムスキーマファイルは、すべてのサーバー上で情報を同じスキーマファイルに保持するために、各サーバーにコピーする必要があります。詳細は、『Directory Server 管理ガイド』の「スキーマ定義のレプリケーション」を参照してください。

ユーザー定義のスキーマのみをレプリケートすることで、転送されるデータの量が削減され、スキーマのレプリケーションを高速にすることができます。詳細は、『Directory Server 管理ガイド』の「スキーマレプリケーションの制限」を参照してください。

レプリケーションと複数パスワードポリシー

複数パスワードポリシーを使用するときは、レプリケートされたエントリに適用するポリシーの定義を含む LDAP サブエントリをレプリケートする必要があります。これを行わなければ、デフォルトのパスワードポリシーが適用されます。このポリシーは、デフォルト以外のパスワードポリシーを使用するように設定されているエントリに対しては機能しません。

これらのエントリを Directory Server 5.0/5.1 サーバーにレプリケートすると、レプリケーションは正しく機能しますが、パスワードポリシーは Directory Server 5.0/5.1 サーバー上で実施されません。複数パスワードポリシー機能は Directory Server 5.2 でサポートされています。

レプリケーションの監視

コマンド行ツールによりサーバー間のレプリケーションを監視できます。レプリケーションアクティビティを監視する機能は、レプリケーションに関する問題を特定する上で役立ちます。すべての監視ツールは、セキュリティ保護された接続を介して使用できます。

レプリケーション監視ツールは LDAP クライアントを構成するため、サーバーへの認証と、`cn=config` に対する読み取りアクセス権を持つバインド DN が必要となります。

次のレプリケーション監視ツールがあります。

- `insync` - マスターレプリカと 1 つまたは複数のコンシューマレプリカとの間の同期状態を示す。
- `entrycmp` - 複数のサーバー上の同一エントリを比較することができる。
- `repldisc` - レプリケーショントポロジを推測することができる。トポロジの推測は、1 つのサーバーから始まり、トポロジ内のすべての既知のサーバーをグラフ化することで行われます。このレプリケーショントポロジ推測ツールは、配備したグローバルトポロジを思い出すことが難しい、大規模で複雑な配備で便利です。

レプリケーション監視ツールについては、『Directory Server Administration Reference』の「Replication Monitoring Tools」を参照してください。特定のレプリケーション属性で使用できる監視機能については、『Directory Server Administration Reference』の「Core Server Configuration Attributes」のレプリケーション属性の節を参照してください。

第 7 章

アクセス制御、認証、および暗号化

Directory Server のデータを保護する方法は、ほかのすべての設計領域に影響します。この章では、セキュリティ要件の分析方法と、その要件を満たすディレクトリの設計方法について説明します。この章は、次の節で構成されています。

セキュリティに対する脅威

ディレクトリのセキュリティに対する脅威となるものは数多く存在します。一般的な脅威についての理解を深めておくと、全体的なセキュリティ設計を行うときに役立ちます。ディレクトリのセキュリティに対する代表的な脅威は、次のカテゴリに分類できます。

- 不正なアクセス
- 不正な改ざん
- サービス拒否

不正なアクセス

不正なアクセスからディレクトリを保護することは簡単なようにみえますが、この問題はかなり複雑です。ディレクトリ情報の配信経路には、権限のないクライアントがデータにアクセスできる箇所がいくつもあります。不正なアクセスには、次のようなものがあります。

- データ取り込み操作によるデータへの不正なアクセス
- 他のユーザーのアクセスを監視することによる、再利用可能なクライアント認証情報への不正なアクセス
- 他のユーザーのアクセスを監視することによる、データへの不正なアクセス

たとえば、権限のないクライアントが別のクライアントの証明情報を利用してデータにアクセスする場合や、権限のないクライアントが、正当なクライアントと Directory Server の間でやり取りされる通信情報を傍受する場合があります。

権限のないアクセスは社内から発生することも、また、会社がエクストラネットやインターネットに接続している場合は、外部から発生することもあります。

Directory Server に備わっている認証方法、パスワードポリシー、およびアクセス制御のメカニズムは、不正アクセスの防止に効果があります。詳細は、165 ページの「適切な認証方法の選択」、171 ページの「パスワードポリシーの設計」、および 180 ページの「アクセス制御の設計」を参照してください。

不正な改ざん

侵入者がディレクトリへアクセスしたり、Directory Server とクライアントアプリケーションの間の通信を傍受した場合は、ディレクトリデータが変更 (あるいは改ざん) される潜在的な危険があります。このような不正な改ざんには、次のようなものがあります。

- データの不正な改ざん
- 設定情報の不正な改ざん

クライアントがデータを信頼できなかったり、ディレクトリ自体がクライアントから受信する変更や照会を信頼できない場合、ディレクトリは役に立たなくなります。

ディレクトリが改ざんを検出できない場合、侵入者がクライアントからサーバーへの要求を変更したり、要求を取り消したり、サーバーからクライアントへの応答を変更することができます。SSL (Secure Socket Layer) プロトコルや、それと同様の技術を利用して、接続の両端で情報に署名することで、この問題は解決できます。Directory Server での SSL の使用については、188 ページの「SSL による接続のセキュリティ保護」を参照してください。

サービス拒否

サービス拒否攻撃とは、侵入者がディレクトリによるクライアントへのサービスの提供を妨害することです。たとえば、侵入者はひたすらシステムのリソースを消費して、ほかのユーザーがリソースを使用するのを妨害します。

Directory Server では、特定のバインド DN に割り当てるリソースに制限を設定することで、サービスの拒否攻撃を防ぎます。詳細は、『Administration Server Administration Guide』の「Setting Individual Resource Limits」を参照してください。

セキュリティ手法の概要

セキュリティポリシーは、権限のないユーザーが機密情報を変更したり取り出したりできないような強固なものであると同時に、簡単に情報の管理ができるものでなければなりません。複雑なセキュリティポリシーを作成すると、許可したユーザーが情報にアクセスできなかったり、あるいはアクセスを許可していないユーザーがディレクトリ情報を変更したり取り出したりする問題につながります。

Directory Server では、次のセキュリティ手法を使用できます。

- 認証

  一方が他方の識別情報を検証する方法です。たとえば、LDAP のバインド操作時に、クライアントは Directory Server にパスワードを呈示します。

- パスワードポリシー

  たとえば、有効期限、長さ、構文など、パスワードの有効性を証明するための条件を定義します。

- 暗号化

  情報の機密性を保護します。データを暗号化すると、データは受信者だけが理解できるような方法で符号化されます。

- アクセス制御

  さまざまなディレクトリユーザーに与えるアクセス権限を調整し、必要な証明情報またはバインド属性を指定する方法を提供します。

- アカウントの無効化

  ユーザーアカウント、アカウントのグループ、またはドメイン全体を無効にして、すべての認証の試行に対して、自動的に拒否するようにします。

- SSL (Secure Sockets Layer)

  情報の完全性を保持します。送信する情報に暗号化とメッセージダイジェストを適用した場合、受信者は、その情報が転送中に改ざんされていないことを確認できます。

- 監査

  ディレクトリのセキュリティが危険にさらされていないかを確認できます。たとえば、ディレクトリで保持されるログファイルを監査できます。

セキュリティを管理するこれらのツールは、セキュリティの設計と組み合わせて使用できます。セキュリティの設計をサポートするために、レプリケーションやデータの分散など、ほかのディレクトリの機能を使用することもできます。

セキュリティ要件の分析

第 2 章「ディレクトリデータの計画とアクセス」でサイト調査を実施した際に、ディレクトリ内の各データについてどのユーザーが読み取りまたは書き込みができるかの決定は終わっているはずです。この情報が、ここではセキュリティの設計の基盤となります。

また、セキュリティの実装方法は、ディレクトリをどのように使用して業務をサポートするかによって異なります。イントラネットを供給するディレクトリでは、エクストラネットをサポートするディレクトリやインターネット上に公開されている電子商取引アプリケーションと同等のセキュリティレベルが要求されることはありません。

ディレクトリがイントラネットだけにサービスを提供する場合は、次の事項を考慮する必要があります。

- 業務の遂行に必要な情報へのアクセスをユーザーとアプリケーションに与える。
- 社員や会社の機密データを通常のアクセスから保護する。
- 情報の完全性を保証する。

ディレクトリがエクストラネットにサービスを提供したり、インターネットを介して電子商取引アプリケーションをサポートしたりする場合は、顧客とパートナー企業に機密性を保証する必要もあります。

次に、セキュリティ要件の分析について、以下の項目ごとに説明します。

- アクセス権限の決定
- データの機密性と完全性の保証
- セキュリティ監査の実行

アクセス権限の決定

データ解析を実行するときは、ユーザー、グループ、取引先、顧客、およびアプリケーションがアクセスする必要のあるデータを特定します。

次の 2 通りの方法でアクセス権を与えます。

- すべてのカテゴリのユーザーに、自己管理を行うまたは管理を委託する権限を与え、同時に機密性の高いデータを保護する。

  この開かれた方法を選ぶ場合は、どのデータが業務上の機密事項あるいは重要事項に該当するのかを十分検討する必要があります。

- 各カテゴリのユーザーに業務に必要な最小限のアクセスだけを与える。

  この方法は制限が多いので、この方法を選ぶ場合は、各カテゴリの内部ユーザーが必要とする情報、また場合によっては外部のユーザーが必要とする情報について、ある程度の時間をかけて検討する必要があります。

どちらの方法でアクセス権限を与える場合でも、組織のユーザーが属するカテゴリとそのカテゴリに与えるアクセス権限を一覧にした、簡単な表を作成してください。また、ディレクトリに保持する機密データ、および各データを保護するために使用した手法を一覧にした表も必要に応じて作成してください。

ユーザーの識別方法については、165 ページの「適切な認証方法の選択」を参照してください。ディレクトリ情報へのアクセスを制限する方法については、180 ページの「アクセス制御の設計」を参照してください。

データの機密性と完全性の保証

エクストラネットを介して取引先との情報交換を行う場合、あるいはインターネット上で顧客が使用する電子商取引アプリケーションをサポートするためにディレクトリを使用する場合は、交換するデータの機密性と完全性を保証する必要があります。

これには、次のような方法があります。

- データを暗号化する
- 証明書を使用してデータに署名する
- データ転送を暗号化する

Directory Server で使用可能な暗号化方法については、174 ページの「パスワード保存スキーマ」および 190 ページの「属性の暗号化」を参照してください。データの署名については、188 ページの「SSL による接続のセキュリティ保護」を参照してください。

セキュリティ監査の実行

セキュリティ監査により、セキュリティポリシーの実装が機能していることが保証されます。セキュリティ監査にはいくつかの側面があります。基本的な監査手順は、ネットワーク上のぜい弱性を識別するネットワークベースのセキュリティスキャナを使用したテストを自動化することです。また、ログファイルと SNMP エージェントによって記録された情報を検査することで、監査を実行することもできます。ディレクトリの監視については、第 8 章「Directory Server の監視」を参照してください。

適切な認証方法の選択

セキュリティポリシーに関して決定が必要な項目に、Directory Server に対するユーザーのアクセス方法があります。匿名アクセスを許可するかどうか、または Directory Server を使用するすべてのユーザーにディレクトリへのバインドを要求するかどうかを決定します。

Directory Server は、次の認証方法をサポートしています。

- 匿名アクセス
- 簡易パスワード
- プロキシ承認
- セキュリティ保護された接続での簡易パスワード
- 証明書に基づくクライアント認証
- SASL ベースのクライアント認証

相手がユーザーか LDAP 対応アプリケーションかにかかわらず、すべてのユーザーに対して同じ認証メカニズムが適用されます。

クライアント単位またはクライアントのグループ単位での認証の防止方法については、170 ページの「アカウントの無効化による認証の防止」を参照してください。

匿名アクセス

匿名アクセスは、ディレクトリにアクセスするもっとも簡単な形式です。匿名アクセスを使用すると、認証とは関係なくだれでもディレクトリデータを利用できます。

しかし、匿名アクセスでは、だれがどのような検索を実行しているのかを追跡することはできず、検索を実行しているユーザーがいるということしかわかりません。匿名アクセスを許可すると、ディレクトリに接続するユーザーはだれでもデータにアクセスできます。

したがって、特定のユーザーまたはユーザーのグループによるディレクトリデータへの読み取りを禁止しようとしても、そのデータへの匿名アクセスを許可していれば、ユーザーは匿名でディレクトリにバインドすることでデータへのアクセスが可能になります。

匿名アクセスの特権は制限できます。通常、ディレクトリ管理者は、匿名アクセスに対して読み取り、検索、および比較の特権だけを許可します。また、アクセスは、ユーザー名、電話番号、電子メールアドレスなど、一般的な情報を含む属性のサブセットに限定されるのが普通です。匿名アクセスは、政府指定の ID (米国の社会保障番号など)、自宅の電話番号と住所、給与情報など機密データには絶対に許可しないでください。

ユーザーパスワード属性が含まれていないエントリでユーザーがバインドを試行した場合、Directory Server は次のどちらかを実行します。

- ユーザーがパスワードの入力を試みない場合、匿名アクセスを許可する。
- ユーザーがパスワードに、何らかの文字列の入力を試みた場合、アクセスを拒否する。

たとえば、次の ldapsearch コマンドを考えてみます。

```
% ldapsearch -h ds.example.com -D "cn=joe,dc=Example,dc=com"
  -w secretpwd -b "cn=joe,dc=Example,dc=com" "objectclass=*"

ldap_simple_bind_s: Invalid credentials
```

この場合、Directory Server によって読み取りについての匿名アクセスが許可されますが、`ldapsearch` コマンドで与えたパスワードと一致するパスワードが自分のエントリに含まれていないため、Joe は自分のエントリにアクセスできません。

簡易パスワード

匿名アクセスを設定しない場合、ディレクトリの内容にアクセスするにはクライアントで Directory Server への認証を行う必要があります。簡易パスワード認証では、クライアントは再使用可能な簡単なパスワードを送信して、サーバーへの認証を行います。

たとえば、識別名と証明情報を送信するバインド操作によって、クライアントは Directory Server への認証を行います。サーバーはクライアント DN に対応したディレクトリ内のエントリを検出し、クライアントから送信されたパスワードとエントリ内に格納されている値が一致するかどうかを確認します。一致した場合、サーバーはクライアントを認証します。一致しない場合、認証操作は失敗し、クライアントにエラーメッセージが返されます。

多くの場合、バインド DN はユーザーのエントリに対応しています。ただし、ユーザーのエントリとしてではなく管理者のエントリとしてバインドする方が便利だと考えるディレクトリ管理者もいます。Directory Server では、バインドに使用するエントリは、`userPassword` 属性を使用できるオブジェクトクラスのエントリである必要があります。これにより、ディレクトリがバインド DN とパスワードを認識することができます。

ユーザーが DN の長い文字列を記憶できない場合もあるため、多くの LDAP クライアントはバインド DN をユーザーに表示しません。クライアントがバインド DN をユーザーに表示しない場合、クライアントは次のようなバインドアルゴリズムを使用します。

1. ユーザーはユーザー ID などの一意の識別子を入力する (たとえば、`bjensen`)。

2. LDAP クライアントアプリケーションはこの識別子でディレクトリを検索し、関連付けられている識別名を返す (`uid=bjensen,ou=people,dc=Example,dc=com` など)。
3. LDAP クライアントアプリケーションは、検出した識別名とユーザーが入力したパスワードを使用してディレクトリにバインドする。

注 簡易パスワード認証の欠点は、パスワードがクリアテキストで送信されることです。悪意を持ったユーザーが盗聴している場合、承認されたユーザーになりすます可能性があり、Directory Server のセキュリティを危険にさらすことになります。

簡易パスワード認証ではユーザーを簡単に認証できますが、組織のイントラネットだけに使用を限定した方がよいでしょう。この認証では、エクストラネットを介した取引先との転送やインターネット上での顧客との転送に求められるレベルのセキュリティは提供されません。

プロキシ承認

プロキシ承認は特殊な形式のアクセス制御です。ユーザーは自分の ID を使用して Directory Server にバインドしますが、プロキシ承認によって別のユーザーの権限が与えられます。

プロキシ承認を使用すると、ディレクトリ管理者は一般ユーザーとして Directory Server へのアクセスを要求できます。ディレクトリ管理者は自分の証明情報を使用してディレクトリにバインドしますが、アクセス制御の評価のために、一般ユーザーの権限が与えられます。このような仮のユーザーはプロキシユーザーと呼ばれ、そのユーザーの DN は プロキシ DN と呼ばれます。

プロキシ要求を実行できる Directory Server を設定するには、次の手順を実行します。

- 管理者には、ほかのユーザーとしてのプロキシ権限を与えます
- 一般ユーザーには、アクセス制御ポリシーで定義されている通常のアクセス権限を与えます。

注 Directory Manager 以外のディレクトリのすべてのユーザーにプロキシ権限を与えることができます。プロキシ権限により、すべての DN (Directory Manager DN を除く) をプロキシ DN として指定する権限が与えられるので、プロキシ権限を与える場合には十分な注意が必要です。

プロキシの主な利点の 1 つとして、Directory Server に要求を送信している複数のユーザーにサービスを提供するために、LDAP アプリケーションが 1 つのバインドで 1 つのスレッドを使用できるようにする点が挙げられます。ユーザーごとにバインドして認証する代わりに、クライアントアプリケーションはプロキシ DN を使用して Directory Server にバインドします。

プロキシ承認については、『Directory Server 管理ガイド』の「アクセス制御の管理」を参照してください。

セキュリティ保護された接続での簡易パスワード

セキュリティ保護された接続では、暗号化によって第三者がデータを読めないようにした上で、Directory Server とクライアントの間でデータを送信します。クライアントは、次のいずれかの方法でセキュリティ保護された接続を確立できます。

- SSL (Secure Socket Layer) を使用してセキュリティ保護されたポートにバインドする。
- 匿名アクセスでセキュリティ保護されていないポートにバインドし、Start TLS 制御を送信して TLS (Transport Layer Security) を使用する。

どちらの場合も、サーバーにはセキュリティ証明書が必要で、この証明書を信頼するようにクライアントを設定する必要があります。サーバーは、証明書をクライアントに送信することで、公開鍵暗号化方式を使用してサーバー認証を行います。その結果、クライアントは目的のサーバーに接続していること、およびサーバーが改ざんされていないことを認識します。

これ以後、クライアントとサーバーは、この接続を通じて伝送されるすべてのデータを機密保護のために暗号化します。クライアントは、暗号化された接続でバインド DN とパスワードを送信してユーザー認証を受けます。それ以後のすべての操作は、そのユーザーの ID、またはバインド DN に別のユーザー ID へのプロキシ権限が含まれる場合はプロキシ ID による操作として実行されます。操作の結果がクライアントに返されるときは、すべてが暗号化されます。

SSL については、188 ページの「SSL による接続のセキュリティ保護」を参照してください。証明書の設定と SSL の有効化については、『Directory Server 管理ガイド』の「認証と暗号化の管理」を参照してください。

証明書に基づくクライアント認証

SSL または TLS を使用する暗号化された接続を確立するときに、サーバーがクライアント認証を要求するように設定することもできます。クライアントは、ユーザー ID の確認のためにサーバーに証明書を送信する必要があります。バインド DN の決定には、ユーザーの DN ではなく、証明情報が使用されます。クライアント認証は、ユーザーの偽装を防ぐ保護で、もっとも安全な接続です。

クライアントが送信できる証明情報の 1 つにユーザー証明書があります。証明書ベースの認証を行うには、証明書のマッピングを行うようにディレクトリを設定し、すべてのユーザーは各自の証明書をそれぞれのエントリに格納しておく必要があります。クライアントからユーザー証明書を受け取ると、サーバーは証明書の内容に基づいてマッピングを行い、ディレクトリからユーザーエントリを検索します。このエントリには、そのユーザーの証明書とまったく同じコピーが含まれている必要があり、これによってユーザー ID の有効性が識別されます。すべての操作はこのエントリの DN をバインド DN として行われ、すべての結果は SSL または TLS 接続によって暗号化されます。

証明書のマッピングについては、『Administration Server Administration Guide』の「Using Client Authentication」および『Directory Server 管理ガイド』の「クライアントでの証明書ベースの認証の設定」を参照してください。

Directory Server は、従来可能だったものよりも大きい証明書に対応できる証明書ベースを含む、NSS (Network Security Services) コンポーネントのバージョンをサポートするようになりました。また、14K バイトより大きいオブジェクトがインポートされると、証明書データベースの次にディレクトリが自動的に作成されます。

SASL ベースのクライアント認証

SSL または TLS 接続でクライアントを認証する方法としては、SASL (Simple Authentication and Security Layer) によるクライアント ID の確認もあります。Directory Server は、SASL の汎用セキュリティインタフェースを通じて次のメカニズムをサポートします。

- DIGEST-MD5: このメカニズムは、クライアントから送信されたハッシュ値とユーザーパスワードのハッシュを比較することでクライアントを認証します。ただし、このメカニズムはユーザーパスワードを読み取る必要があり、DIGEST-MD5 による認証を希望するすべてのユーザーは、ディレクトリ内に {CLEAR} パスワード (クリアテキスト形式のパスワード) を持つ必要があります。

- GSSAPI: Solaris オペレーティングシステムで利用できる GSSAPI (General Security Services API) では、Directory Server は Kerberos V5 セキュリティシステムとの対話によってユーザーを識別します。クライアントアプリケーションは Kerberos システムに証明情報を提示し、このシステムがユーザーの ID を Directory Server に返します。

どちらの SASL メカニズムを使用する場合も、ID のマッピングを行うようにサーバーを設定する必要があります。SASL 証明情報は主体と呼ばれ、各メカニズムは特定のマッピングを使用して主体の内容からバインド DN を決定します。主体が 1 つのユーザーエントリにマッピングされ、SASL メカニズムがそのユーザーの ID を検証すると、ユーザーの DN がその接続のバインド DN となります。

詳細は、『Directory Server 管理ガイド』の「クライアントでの SASL DIGEST-MD5 の使用」および「クライアントでの Kerberos SASL GSSAPI の使用」を参照してください。

アカウントの無効化による認証の防止

ユーザーアカウントまたはアカウントのセットを一時的に無効にできます。アカウントが無効になると、ユーザーは Directory Server にバインドできないため、このユーザーの認証操作は失敗します。

アカウントの無効化は、`nsAccountLock` オペレーショナル属性を使用して実装されます。エントリに `true` の値を持つ `nsAccountLock` 属性が含まれている場合、サーバーはバインドを拒否します。`nsAccountLock` 属性は、手動ではなくコマンド行ユーティリティを使って変更してください。

ユーザーとロールの無効化にも、同じ手法を使用します。ただし、ロールの無効化は、そのロールのメンバー全員を無効にしますが、ロールのエントリ自体は無効にはしません。ロールについては、82 ページの「管理されているロール、フィルタを適用したロール、入れ子のロール」を参照してください。

パスワードポリシーの設計

パスワードポリシーは、システム内でパスワードがどのように管理されるかを規定した規則の集合です。

パスワードポリシーにより、ディレクトリ全体に対して 1 つのグローバルポリシーではなく、複数のパスワードポリシーを設定できます。また、Cos およびロール機能を使用して、パスワードポリシーを特定のユーザーまたはユーザーセットに適用することができます。特定のユーザーまたはロールに合わせてパスワードポリシーを設定できるので、パスワードポリシーによるセキュリティ対策の実装時にも適用範囲が大幅に広がります。

パスワードポリシーは、スタティックグループには適用できません。

パスワードポリシーの作成に使用できる属性については、『Directory Server Administration Reference』の「Password Policy Attributes」および「Account Lockout Attributes」を参照してください。パスワードポリシーの設定と管理については、『Directory Server 管理ガイド』の「ユーザーアカウントとパスワードの管理」を参照してください。この節は、次の項目で構成されています。

パスワードポリシーの機能

ここでは、パスワードポリシーの主な機能について説明します。説明する内容は次のとおりです。

ユーザー定義のパスワード

パスワードポリシーを設定して、ユーザーが自分のパスワードを変更することを許可または禁止することができます。適切なパスワードを用いることは、パスワードポリシーを強固なものにする上で重要です。適切なパスワードとは、安易な単語、つまり辞書に載っているような単語、ペットや子供の名前、誕生日、ユーザー ID、あるいは、簡単に見破られる可能性があるユーザーに関するその他の情報 (ディレクトリ自体に格納されている情報も含む) を使用していないものです。

また、パスワードには、文字、数字、記号などの組み合わせを含めるようにしてください。しかし、ユーザーは単に覚えやすいパスワードを使用する傾向があります。そのため、「適切な」パスワードの条件を満たすパスワードを事前に設定し、ユーザーによるパスワードの変更を許可しない企業もあります。

ただし、ユーザーにパスワードを割り当てる作業は、管理者にとってかなりの時間がかかる作業です。また、自分にとって意味があり覚えやすいパスワードをユーザー自身が選択するのではなく、管理者がパスワードを提供すると、ユーザーはそのパスワードをどこかに書き留めてしまい、だれかが見つけてしまう危険があります。デフォルトでは、ユーザー定義のパスワードは許可されています。

最初のログインまたはリセット時のパスワードの変更

Directory Server のパスワードポリシーでは、初回のログイン時または管理者がパスワードをリセットしたあとに、ユーザーがパスワードを変更する必要があるかどうかを決めることができます。

多くの場合、管理者が設定した初回のパスワードは、ユーザーのイニシャル、ユーザー ID、会社名など、ある種の表記規則に従って設定されます。一度表記規則が発見されると、通常ハッカーがシステムに侵入するために最初に入力を試みる候補となります。そのため、初回のログイン時または管理者によるリセット後に、パスワードの変更をユーザーに義務付けるとよいでしょう。このオプションをパスワードポリシーに設定すると、ユーザーが定義したパスワードが無効になっている場合でも、ユーザーはパスワードを変更するように要求されます。前の「ユーザー定義のパスワード」の項を参照してください。

デフォルトでは、ログイン時またはリセット後にユーザーがパスワードを変更する必要はありません。

パスワードの有効期限

パスワードポリシーは、ユーザーが同じパスワードを無期限に使用できるように、または一定期間が過ぎるとパスワードが期限切れになるように設定することができます。一般に、パスワードの有効期限が長いほど見破られやすくなります。ただし、パスワードが頻繁に期限切れになると、ユーザーはパスワードを憶えることが困難になり、パスワードを書き留めるようになってしまいます。一般的なポリシーでは、パスワードを 30 から 90 日で期限切れにします。

Directory Server では、パスワードの有効期限を無効にしても、パスワードの有効期限の設定は残されます。つまり、パスワードの有効期限のオプションを有効に戻した場合、パスワードの有効期限は、最後にこの機能を無効にしたときに設定していた期間になります。たとえば、パスワードが 90 日で期限切れになるように設定して、次にパスワードの有効期限を無効にするように設定したとします。パスワード期限をもう一度有効にするように設定を戻すと、この機能を無効にする前には有効期限を 90 日に設定していたので、デフォルトのパスワードの有効期限は 90 日になります。

デフォルトでは、ユーザーパスワードは期限切れになりません。

新しいグローバル設定属性 `usePwdChangedTime` により、パスワードが変更されたあと、たとえば管理者がパスワードをリセットしたあとで、ユーザーがログインできる時間を制限することができます。

期限切れの警告

一定の期間が過ぎると、ユーザーパスワードが期限切れになるようにパスワードポリシーを設定した場合は、パスワードが期限切れになる前にユーザーに警告を送信します。パスワードが期限切れになる 1 ～ 24,855 日前に、ユーザーに警告が送信されるようにポリシーを設定できます。ユーザーがサーバーにバインドすると、Directory Server によって警告が表示されます。デフォルトでは、パスワードの有効期限をオンに設定した場合、ユーザーのクライアントアプリケーションがこの機能をサポートしていれば、ユーザーパスワードが期限切れになる 1 日前に (LDAP メッセージを介して) 警告がユーザーに送信されます。

パスワードの構文検査

パスワードポリシーには、パスワード文字列の構文ガイドラインがあります。パスワードの構文検査メカニズムによって、パスワード文字列がこれらのガイドラインに従っているかどうかが検査されます。デフォルトでは、パスワードの構文検査はオフに設定されています。パスワード長の検査は、パスワードの構文検査が有効な場合にだけ行われます。

パスワード長

Directory Server では、ユーザーパスワードに必要な最少文字数を指定できます。一般に、パスワードが短いほど、不正な手段による解読が簡単になります。2 ～ 512 文字のパスワードを要求できます。パスワードに適した長さは、8 文字です。これは不正な手段で解読することが難しく、またユーザーが記録しておかなくても覚えられる長さです。

デフォルトの最小パスワード長は 6 文字です。最小パスワード長の検査は、パスワードの構文検査が有効な場合にだけ行われます。

パスワードの最短有効日数

パスワードポリシーは、ユーザーが指定された期間の間パスワードを変更できないように設定できます。この機能とパスワード履歴機能を組み合わせると、ユーザーが古いパスワードを再使用するのを防止できます。

たとえば、パスワードの最短有効日数を 2 日に設定すると、1 つのセッションの間にパスワードを繰り返し変更して古いパスワードを履歴からいったんなくし、その後古いパスワードを再使用するという方法を防止できます。0 ～ 24,855 日の間の任意の数字を指定できます。ゼロ (0) の値は、ユーザーがすぐにパスワードを変更できることを示します。

パスワードの履歴

Directory Server は、これまでに使用されたパスワードを最大 24 個格納するように設定できます。

パスワード履歴が有効な場合、ユーザーが Directory Server に格納されているパスワードを再使用しようとすると、そのパスワードが拒否されます。この機能を使用した場合、覚えやすい数個のパスワードをユーザーが再使用することはできなくなります。

パスワード履歴機能が無効の場合は、過去に使用されたパスワードは格納されず、ユーザーはパスワードを再使用できます。デフォルトでは、Directory Server はパスワード履歴を保持します。

パスワード保存スキーマ

パスワード保存スキーマでは、ディレクトリ内にパスワードを格納するときに使用する暗号化のタイプを指定します。次のタイプを指定できます。

- クリアテキスト (暗号化なし)。
- SHA (Secure Hash Algorithm)。
- SSHA (Salted SHA)。これはデフォルトの暗号化方式です。
- UNIX CRYPT アルゴリズム。

ディレクトリに格納されているパスワードは ACI (アクセス制御情報) 命令を使用して保護できますが、ディレクトリにクリアテキストでパスワードを格納することはお勧めできません。CRYPT アルゴリズムは、UNIX のパスワードと互換性があります。SSHA はもっとも安全な方式で、Directory Server のデフォルトのハッシュアルゴリズムです。

パスワードポリシーの設定

次のようなパスワードポリシーのオプションがあります。

- グローバルパスワードポリシー。cn=Password Policy,cn=config の下に格納され、デフォルトで適用されます。このグローバルポリシーのパラメータは、変更可能です。
- Directory Server 5.2 より前の Directory Server のバージョンでは、グローバルパスワードポリシーが唯一のパスワードポリシーでした。グローバルパスワードポリシーのバックアップとして、ハードコーディングされたパスワードポリシーが用意されています。このポリシーは、グローバルパスワードポリシーが見つからない場合や、変更によって有効でなくなった場合に適用されます。ハードコーディングされたパスワードポリシーの属性値は、デフォルトのグローバルパスワードポリシーの値と同じです。
- パスワードポリシーを定義して、それを特定のユーザーに適用できます。
- パスワードポリシーを定義し、CoS 機能やロール機能を使用してそれをユーザーセットに適用することができます。

注 : パスワードポリシーは、スタティックグループには適用できません。

ここでは、これらのオプションの詳細と、特定のユーザーエントリに複数のパスワードポリシーが存在する場合にパスワードポリシーの適用を制御する優先順位について説明します。この節は、次の項目で構成されています。

- グローバルパスワードポリシー
- ユーザーまたはユーザーセットに適用するパスワードポリシーの定義
- 複数のパスワードポリシーとその優先順位

グローバルパスワードポリシー

グローバルパスワードポリシーは、cn=Password Policy,cn=config の下に格納され、デフォルトで適用されます。このポリシーは、セキュリティのニーズに合うように変更できます。デフォルトのグローバルポリシーには、次の項目が適用されます。

- SSHA 保管スキーマ。
- ユーザーはパスワードを変更できる。
- 最初のログイン時、または管理者がパスワードをリセットしたあとに、ユーザーはパスワードを変更する必要はない。
- パスワードの構文検査 (最小文字数の遵守) を行わない。
- パスワードには有効期限は設定されない。
- パスワードの有効期限を有効化した場合、パスワードの最大有効期間は 100 日である。

- パスワードの変更が可能になるまでの期間は設定されない。
- パスワードの有効期限を有効化した場合は、パスワードの有効期限が切れる 1 日前の最初のバインド試行時にパスワードの期限切れ警告が送信される。
- 使用したパスワードは記録されない。
- ユーザーはアカウントをロックアウトされない。
- アカウントロックアウトを有効にした場合は、3 回バインド試行に失敗すると、ユーザーはロックされ、ロックアウトは 1 時間継続する。
- パスワード失敗の記録は、600 秒後に失敗カウンタから削除される。

パスワードに有効期限が設定されず、構文検査は行われず、アカウントロックアウトメカニズムも有効化されていないデフォルトポリシーは、管理のオーバーヘッドは低くても、もっとも安全なものではありません。セキュリティ要件と、厳しいパスワードポリシーによる管理のオーバーヘッドの間でバランスを取る必要があります。

注 従来バージョンの Directory Server では、グローバルパスワードポリシー属性は `cn=config` の直下に格納されていました。現在は、`cn=Password Policy,cn=config` の下に格納されます。このエントリが存在しない場合は、Directory Server に用意されているハードコーディングされたパスワードポリシーが適用されます。

ユーザーまたはユーザーセットに適用するパスワードポリシーの定義

特定のユーザーエントリまたはユーザーセットに適用するパスワードポリシーは、`passwordPolicySubentry` という属性を使用して定義します。この属性の値は、ユーザーのエントリに直接適用するパスワードポリシー属性が含まれる LDAP サブエントリの DN です。この属性は、実際の属性にすることも、CoS 定義によって生成した仮想属性にすることもできます。

CoS 定義を設定し、ユーザーエントリが持つロールの機能として、ユーザーエントリ内の `passwordPolicySubentry` 属性に値を指定することで、ユーザーセットに適用するパスワードポリシーを定義できます。ユーザーセットに適用するパスワードポリシーを定義する方法はほかにもありますが、Directory Server コンソール ではこの方法を使用します。パスワードポリシーは、スタティックグループには適用できません。

パスワードポリシーの管理を容易にするために、パスワードポリシーの適用対象となるユーザーまたはユーザーセットと、パスワードポリシー自体を同じ場所で保管してください。これにより、パスワードポリシーの LDAP サブエントリがパスワードポリシーと一緒にレプリケートされます。

パスワードポリシーは、ダイナミックグループに適用できますが、スタティックグループには適用できません。

個々のパスワードポリシーの定義手順については、『Directory Server 管理ガイド』の「個別パスワードポリシーの管理」を参照してください。

複数のパスワードポリシーとその優先順位

ユーザーエントリに複数のパスワードポリシーが割り当てられている場合にパスワードポリシーの適用を制御する優先順位には、主に 3 つの規則が適用されます。適用する規則は次のとおりです。

1. CoS 定義によって生成されるパスワードポリシーは、ユーザーエントリに直接割り当てられているパスワードポリシーに優先して適用されます。これは、CoS 定義エントリに定義されている `cosAttribute` 値には `operational` 修飾子が必ず含まれているためで、これにより、CoS 生成によるパスワードポリシーは、ユーザーに直接割り当てられたあらゆる実属性に優先して適用されます。ロールと CoS メカニズムについては、第 4 章「ディレクトリ情報ツリー」を参照してください。
2. ユーザーエントリに直接割り当てられているパスワードポリシーは、グローバルパスワードポリシーに優先して適用されます。
3. `cn=Password Policy,cn=config` の下に格納されているグローバルパスワードポリシーは、Directory Server に用意されているハードコーディングされたパスワードポリシーの値に優先して適用されます。

警告 CoS を使用してパスワードポリシーを設定するときは、CoS によって生成された複数のパスワードポリシーがユーザーに適用された場合の優先順位を指定する必要があります。優先順位を指定するには、CoS テンポラリエントリを作成するときに `cosPriority` 属性に適切な値を入力します。最優先を指定するには、値に 0 を割り当てます。`cosPriority` 属性を持たない CoS テンプレートの優先順位は最低と見なされ、複数のテンプレートの `cosPriority` 属性の値が同じ (または指定されていない) 場合は、任意の優先順位が適用されます。ロールと CoS については、第 4 章「ディレクトリ情報ツリー」を参照してください。

辞書攻撃の防止

辞書攻撃では、侵入者は認証が得られるまで、繰り返し一般用語のリストからパスワードを推測して解読しようとします。このような攻撃に対応するために、サーバーには 3 つのツールが用意されています。

- パスワード構文検査では、ユーザーエントリの `uid`、`cn`、`sn`、`givenName`、`ou`、または `mail` 属性の一致を検証します。いずれかの値と一致する場合、サーバーはユーザーによるパスワードの設定を拒否します。ただしパスワード構文検査では、`/usr/dict/words` の単語をすべて試すような実際の辞書攻撃は阻止されません。
- パスワードに最小限必要な文字数を設定すると、ユーザーは短いパスワードを設定できません。パスワードが長くなるほど、すべての値を想像したり、組み合わせることが指数関数的に困難になります。Directory Server では、パスワード構文検査と最小限必要な文字数の設定の両方を有効にする必要があります。
- アカウントロックアウトメカニズムは、認証の試みが数回失敗した後にバインドを拒否します。パスワードポリシーの厳密度に応じて、一時的なロックアウトと永続的なロックアウトのいずれかが適用されます。

  いずれも、自動的なパスワードの推測を効果的に防止します。たとえば、5 回までの試行を許可してその後にユーザーアカウントを 5 分間ロックアウトした場合、平均すると侵入者は 1 分間に 1 回の推測しか行えず、正規のユーザーが入力ミスなどでロックアウトされた場合も短時間で再試行できます。永続的なロックアウトの場合は、Directory Manager から手動でパスワードをリセットする必要があります。

  アカウントロックアウトのカウンタは Directory Server 固有です。この機能は、ディレクトリサービスからグローバルにロックアウトするようには設定されていません。つまり、レプリケーション環境でもアカウントロックアウトのカウンタはレプリケートされません。

レプリケーション環境でのパスワードポリシー

グローバルパスワードポリシーの設定情報は `cn=config` の下にあるエントリのため、レプリケートされません。このため、グローバルパスワードポリシーを変更する場合は、トポロジ内の各サーバーで、同じ変更を手動で加える必要があります。レプリケートされるグローバルパスワードポリシーが必要な場合は、レプリケートされるディレクトリツリーの一部の下に、そのようなポリシーを定義する必要があります。

ユーザーエントリに格納されるパスワード情報 (現在のパスワード、パスワード履歴、パスワードの有効期限など) は、すべてレプリケートされます。アカウントロックアウトのカウンタはローカルサーバーレベルで格納され、レプリケートはされません。

レプリケートされた環境では、パスワードポリシーによる次の影響を考慮する必要があります。

- パスワードの期限切れが近づいたユーザーは、パスワードを変更するまで、バインドするすべてのレプリカから警告を受信します。
- ユーザーがパスワードを変更すると、すべてのレプリカでパスワード変更の情報が更新されるまでに時間がかかります。ユーザーがパスワードを変更し、すぐに新しいパスワードでコンシューマレプリカのどれかに再度バインドしようとすると、レプリカが更新されたパスワードを受信するまでは、バインドに失敗します。
- 各レプリカには、レプリケートされない個別のアカウントロックアウトカウンタが保持されています。その結果、ロックアウトポリシーはどれか 1 つのレプリカで適用されますが、ユーザーが複数のレプリカでバインドを試みるとポリシーが適用されないことがあります。たとえば、レプリケーショントポロジに 10 のサーバーがあり、3 回の試行後にロックアウトが有効になる場合、侵入者はパスワードの推測を 30 回行える計算になります。

  レプリケーションによって侵入者が推測できるパスワードの数は増えますが、推測される無数の組み合わせと比較すれば、影響はほとんどありません。それ以上に、パスワード検査を有効にし、6 文字以上のパスワードを設定して、強度の高いパスワードをユーザーに強制することのほうが重要です。また、辞書に登場するような一般的な単語ではないパスワードを選び、記憶する方法について、ユーザーにガイドラインを示すことも必要です。最後に、すべてのディレクトリ管理者ユーザーが強力なパスワードを持たなければならないことは言うまでもありません。
- 複数パスワードポリシーを使用する環境では、レプリケートされたエントリに適用するポリシーの定義を含む LDAP サブエントリをレプリケートする必要があります。これを行わない場合、ポリシー定義を含む LDAP サブエントリが存在しないため、デフォルトのパスワードポリシーが適用されます。
- レプリケーションのために作成したエントリ (サーバーの識別するレプリカマネージャエントリなど) には、無期限のパスワードを設定する必要があります。これらの特別なユーザーに確実に無期限のパスワードを使用させるには、`passwordExpirationTime` 属性をエントリに追加し、この属性に `20380119031407Z` (有効範囲の最大値) を指定します。

アクセス制御の設計

アクセス制御では、特定の情報へのアクセス権を一部のクライアントに与え、その他のクライアントには与えないように指定することができます。アクセス制御は、1 つまたは複数のアクセス制御リスト (ACL) を使用して実装します。ACL は、指定したエントリやその属性についてアクセス権限 (読み取り、書き込み、検索、プロキシ承認、追加、削除、比較など) を許可または拒否する一連のアクセス制御命令 (ACI) から構成されます。

ACL を使用すると、次の項目に対する権限を設定できます。

- ディレクトリ全体
- ディレクトリの特定のサブツリー
- ディレクトリ内の特定のエントリ
- 特定のエントリの属性セット
- 特定の LDAP 検索フィルタにマッチするすべてのエントリ

また、特定のユーザー、グループに属するすべてのユーザー、およびディレクトリのすべてのユーザーに対する権限を設定できます。さらに、IP アドレスや DNS 名など、ネットワークの場所に対してアクセス権を定義することもできます。

ここでは、Directory Server のアクセス制御メカニズムについて説明します。説明する内容は次のとおりです。

ACI 形式

ACI は、エントリの属性としてディレクトリ内に格納されます。`aci` 属性はオペレーショナル属性です。この属性は、そのエントリのオブジェクトクラス用に定義されたものであるかどうかに関わらず、ディレクトリ内のすべてのエントリで使用できます。aci 属性は、Directory Server がクライアントから LDAP 要求を受け取るときに、どの

アクセス権が与えられ、どのアクセス権が拒否されるかを判定するために使用されます。aci 属性が ldapsearch 処理で返されるように指定することができます。ACI の形式については、『Directory Server 管理ガイド』の「ACI の構文」を参照してください。

デフォルト ACI

Directory Server をインストールしたときや新しいサフィックスを追加したときは、多数のデフォルト ACI が定義されます。この ACI は、セキュリティ要件に合うように変更できます。デフォルト ACI とそれを変更する方法については、『Directory Server 管理ガイド』の「デフォルト ACI」を参照してください。

権限の設定

ディレクトリに ACI が存在しない場合のデフォルトポリシーは、ユーザーにいかなるアクセス権も与えません。ただし、Directory Manager は例外です。このため、ユーザーがディレクトリにアクセスできるように、いくつかの ACI を設定する必要があります。次に、Directory Server に用意されているアクセス制御のメカニズムについて説明します。ACI の設定については、『Directory Server 管理ガイド』の「コマンド行からの ACI の作成」および「コンソールを使用した ACI の作成」を参照してください。

優先規則

ユーザーがディレクトリエントリに対してどのようなタイプのアクセスを試みた場合でも、Directory Server はディレクトリ内のアクセス制御セットを検査します。アクセスを確定するために、Directory Server は優先規則を適用します。この規則は、2 つの競合する権限が存在する場合、アクセス拒否の権限がアクセス許可の権限よりも常に優先されることを規定しています。

たとえば、ディレクトリのルートレベルで書き込み権限を拒否して、Directory Server にアクセスするすべてのユーザーにこの権限を適用した場合、ユーザーに許可されているほかの権限に関係なく、だれもディレクトリに書き込むことはできません。特定のユーザーに Directory Server への書き込み権限を許可するには、元の書き込み拒否の適用範囲を限定してそのユーザーを除外しておく必要があります。また、ユーザーに対して書き込みを許可する権限を作成し追加しておく必要があります。

アクセスの許可または拒否

ディレクトリツリーへのアクセスは、明示的に許可または拒否できます。Directory Server へのアクセスを明示的に拒否する場合は、慎重に行なってください。優先規則が適用されるため、明示的にアクセスを禁止する規則がディレクトリにある場合、アクセスを許可する権限の有無にかかわらず、アクセスは拒否されることになります。

アクセスを許可する規則の適用範囲を限定して、最低限のユーザーまたはクライアントアプリケーションだけが含まれるようにしてください。たとえば、ユーザーのディレクトリエントリにあるすべての属性への書き込み権限を許可し、ディレクトリ管理者を除くすべてのユーザーに `uid` 属性への書き込み権限を拒否するように権限を設定できます。あるいは、次の方法で書き込み権限を許可する 2 つのアクセス規則を作成するという方法もあります。

- `uid` 属性を除くすべての属性への書き込み特権を許可する規則を 1 つ作成する。この規則はすべてのユーザーに適用する必要がある。
- `uid` 属性への書き込み特権を許可する規則を 1 つ作成する。この規則は、ディレクトリ管理者グループのメンバーだけに適用する必要がある。

アクセス権だけを作成することにより、明示的な拒否アクセス権を設定する必要はなくなります。

アクセスを拒否する場合

明示的な拒否を設定する必要は、ほとんどありません。ただし、次のような状況では明示的な拒否が有効である場合もあります。

- 複雑な ACL が全体的に適用されている大きなディレクトリツリーがある場合。

  セキュリティ上の理由から、特定ユーザー、特定グループ、または物理的な場所へのアクセスを拒否する必要性が突然生じる場合があります。許可の権限の適切な制限方法を把握するために既存の ACL を詳しく調べるよりも、検討の余裕ができるまでの間、明示拒否を一時的に設定することができます。ACL がこのように複雑になってしまった場合、拒否の ACI 設定は、長期的には管理の負担を増大させるだけです。できるだけ早期に ACI を修正して、明示的な拒否を取り除きアクセス制御スキーマ全体を簡略化します。

- 曜日または時間帯に基づいてアクセス制御を限定する場合。

  たとえば、日曜日の午後 11:00 (2300) から月曜日の午前 1:00 (0100) まで、すべての書き込み操作を禁止します。管理上の観点からは、ディレクトリを検索してすべての ACI 書き込み権限を特定し、この時間帯における許可範囲を制限するよりも、この種の時間に基づいたアクセスを明示的に制限する ACI を管理した方が簡単な場合があります。

- ディレクトリの管理権限を複数の管理者に与えるときに、特権を制限する場合。

  個人またはグループのメンバーにディレクトリツリーの管理を部分的に許可し、その上でそのツリーの一部を変更しないようにする場合は、明示的な拒否を使用します。たとえば、メール管理者に共通名属性への書き込み権限を許可しないようにする場合、共通名属性への書き込み権限を明示的に拒否する ACI を設定します。

アクセス制御規則の格納場所

アクセス制御規則は、ディレクトリの任意のエントリに置くことができます。多くの場合、管理者は `country`、`organization`、`organizationalUnit`、`inetOrgPerson`、または `group` のタイプのエントリにアクセス制御規則を置きます。

ACL の管理を簡単にするために、この規則はできるだけグループ化します。通常、規則はターゲットエントリとそのエントリのすべての子に適用されるため、最下位にある個々のエントリ (ユーザーなど) にアクセス制御規則を分散させるよりも、ディレクトリのルートポイントまたは分岐点にアクセス制御規則を配置すると良いでしょう。

フィルタが適用されたアクセス制御規則の使用

Directory Server の ACI モデルの強力な機能の 1 つに、LDAP 検索フィルタを使用してアクセス制御を設定できるという機能があります。LDAP 検索フィルタでは、定義した条件に一致するすべてのディレクトリエントリへのアクセス権限を設定できます。

たとえば、マーケティングに設定されている `organizationalUnit` 属性を含むすべてのエントリへの読み取り権限を許可できます。

フィルタが適用されたアクセス制御規則では、事前にアクセスレベルを定義することもできます。たとえば、ディレクトリに自宅の住所と電話番号に関する情報が含まれているとします。これを公開したいと考える人と、リストからの情報の削除を求める人がいます。この情報は、次のように解決できます。

- 各ユーザーのディレクトリエントリに `publishHomeContactInfo` という属性を作成します。
- `publishHomeContactInfo` 属性が TRUE (有効を意味する) に設定されているエントリだけに、`homePhone` と `homePostalAddress` 属性への読み取り権限を許可するアクセス制御規則を設定します。LDAP 検索フィルタを使用して、この規則のターゲットを示します。
- ディレクトリユーザーが自分の `publishHomeContactInfo` 属性の値を TRUE または FALSE に変更できるようにします。このようにすると、この情報を公開するかどうかをディレクトリユーザーが決定できます。

ACI との LDAP 検索フィルタの使用については、『Directory Server 管理ガイド』の「LDAP フィルタを使用したエントリまたは属性のターゲット指定」を参照してください。

マクロ ACI の使用

マクロ ACI は、ACI の中で DN、または DN の一部を表すために使用される可変部分です。マクロを使用すると、ACI のターゲット部分またはバインド規則部分、あるいはその両方の DN を表すことができます。Directory Server が LDAP 要求を受け取ると、マクロ ACI が LDAP 操作のターゲットとなるリソースと照合されます。一致した場合、マクロは対象となるリソースの DN の値に置き換えられます。次に Directory Server は ACI を通常どおりに評価します。

サーバーを起動したとき、すべての ACI は、キャッシュの制限を受けることなくメモリに読み込まれます。これは、メモリの消費に大きな影響を与える可能性があります。このため、繰り返しのディレクトリツリー構造を使用する場合は、可能であればマクロ ACI を使用して、Directory Server で使用される ACI の数を最適化してください。

マクロ ACI については、『Directory Server 管理ガイド』の「高度なアクセス制御 : マクロ ACI の使用」を参照してください。

実効権限に関する情報の取得

Directory Server が提供するアクセス制御モデルは強力で、さまざまなメカニズムを使用してユーザーにアクセス権を与えることができます。ただし、このような柔軟性が、セキュリティポリシーをかなり複雑にしてしまう要因になることがあります。たとえば、IP アドレスとマシン名、時刻、認証方法など、ユーザーのセキュリティはいくつかのパラメータを使って定義することができます。このため、ディレクトリのエントリと属性に対する特定のユーザーの権限をリスト化しておくと便利です。

Directory Server では、ユーザーが指定したディレクトリのエントリと属性に対して持つ実効権限を要求できます。取得される実効権限情報は、次のものに相当します。

- 照会時に有効な ACI
- 適用される認証方法
- 照会元ホストマシンの名前とアドレス

ユーザーが特定のデータに対するアクセス権を持つ、または持たない理由を特定するときは、実効権限の検索を行うときに、ユーザーのパラメータすべてを反映する必要があります。

ここでは、実効権限機能の詳細について説明します。説明する内容は次のとおりです。

- 実効権限機能について
- 実効権限機能に対するアクセス制御
- 実効権限要求の結果

実効権限機能について

実効権限機能は、DSRK (Directory Server リソースキット) に含まれる `ldapsearch` ユーティリティを使用します。照会する権限情報の指定には、特別なオプションを使用します。権限に関する情報は、`ldapsearch` の結果として返されます。実効権限機能の使用方法については、『Directory Server 管理ガイド』の「実効権限の表示」を参照してください。

実効権限機能に対するアクセス制御

実効権限情報を取得するには、実効権限制御を使用するためのアクセス制御権と、`aclRights` 属性に対する読み取りアクセス権が必要です。実効権限機能に対する基本的なセキュリティは、この二重のアクセス制御によって得られ、これを必要に応じてさらに調整することができます。プロキシ承認のように、`aclRights` に対する読み取りアクセス権があれば、エントリと属性に対するどのユーザーの権限に関する情報でも要求することができます。つまり、リソースを管理するユーザーは、だれがそのリソースに対する権限を持つかを決定できます。これは、そのユーザーが実際にはその権限を使って管理していない場合も同様です。

権限情報を照会するユーザーが実効権限制御を使用する権限を持たない場合、操作は失敗し、エラーメッセージが返されます。一方、権限情報を照会するユーザーがこの制御を使用する権限は持つが、`aclRights` 属性に対する読み取りアクセス権を持たない場合は、返されるエントリから `aclRights` 属性が省略されるだけです。この動作は、Directory Server の通常の検索動作を反映しています。

実効権限制御に基づく検索操作にプロキシ制御が関わっている場合、実効権限の照会操作はプロキシユーザーとして承認されます。つまり、権限を照会して実効権限制御を使用するのはプロキシユーザーであり、返されるエントリは、そのプロキシユーザーが検索と表示の権限を持つエントリとなります。

実効権限要求の結果

実効権限要求の結果として返される情報には、次のものがあります。

- 権限情報 (エントリレベルの権限、属性レベルの権限、ログなど)。
- 書き込み、本人による書き込みの追加、および本人による書き込みの削除のアクセス権。
- ログ情報 (アクセス制御に関する問題をデバッグできる)。

これらの側面のそれぞれについては、『Directory Server 管理ガイド』の「実効権限検索結果の内容」を参照してください。

ACI の使用に関するヒント

次に示すヒントは、ディレクトリのセキュリティモデルを管理する際の負担を軽減し、ディレクトリのパフォーマンス向上にも役立ちます。一部のヒントについてはこの章ですでに説明しましたが、ここでは完全なリストを示します。

- ディレクトリ内の ACI の数を最小化し、可能であればマクロ ACI を使用する。

  Directory Server は 50,000 を越える ACI を評価できますが、多数の ACI 文を管理することは難しく、ACI が多すぎるとメモリ消費に悪影響を及ぼす可能性があります。

- 許可権限と拒否権限のバランスを図る。

  デフォルトの規則ではアクセスを与えられていないすべてのユーザーに対してアクセスを拒否しますが、アクセスを許可する 1 つの ACI をツリーのルートの近くで使用し、アクセスを拒否する少数の ACI を最下位のエントリの近くで使用することで、ACI の数を減らすことができます。このようにすると、最下位のエントリの近くでアクセスを許可する ACI が必要以上に多くなることがなくなります。

- ACI では最小の属性セットを指定する。

  オブジェクトの属性の一部でアクセスを許可または制限している場合、その最小の属性リストが、許可される属性セットであるか、拒否される属性セットであるかを判定することを意味します。次に、最小の属性リストを管理するように ACI を表します。

  たとえば、ユーザーオブジェクトクラスに多くの属性が含まれている場合を考えます。これらの属性のうち、1 つか 2 つだけをユーザーが更新できるようにする場合、その少数について書き込み権限を許可する ACI を作成します。逆に、1 つか 2 つの属性以外のすべてをユーザーが更新できるようにする場合は、それらの 1 つか 2 つの属性について書き込み権限を拒否する ACI を作成します。

- LDAP 検索フィルタは慎重に使用する。

  検索フィルタではアクセスを管理するオブジェクト名が直接指定されないため、特にディレクトリが複雑になる場合などは、検索フィルタを使用すると、予期しない状況が発生することがあります。ACI の中で検索フィルタを使用する場合、同じフィルタを使用して `ldapsearch` 操作を実行し、変更の結果がディレクトリに及ぼす影響を確認します。

- ディレクトリツリーの別の部分で ACI を重複させない。

  ACI が重複しないよう注意してください。たとえば、あるグループに `commonName` および `givenName` 属性への書き込み権限を許可する ACI がディレクトリのルートポイントにあり、同じグループに `commonName` 属性だけへの書き込み権限を許可する別の ACI がある場合は、ACI を作り直して 1 つの制御でそのグループに書き込み権限を許可するようにしてください。

ディレクトリが複雑になるにつれ、偶発的に ACI が重複する事態が発生しやすくなります。ACI の重複を避けることにより、セキュリティ管理が容易になる上、ディレクトリに含まれる ACI の総数も減少します。

- ACI に名前を付ける。

  ACI に名前を付けるかどうかは任意ですが、各 ACI にわかりやすい短い名前を付けておくと、特に Directory Server コンソールから ACI を検査するときなど、セキュリティ モデルの管理に役立ちます。

- ユーザーエントリに標準属性を使用して、アクセス権限を決める。

  可能な限り、すでに標準ユーザーエントリの一部となっている情報を使用して、アクセス権限を決めてください。特別な属性を作成する必要がある場合は、ロールまたはサービスクラス (CoS) の定義の一部として作成することを検討します。ロールと CoS については、80 ページの「ディレクトリエントリのグループ化と属性の管理」を参照してください。

- ディレクトリ内のできるだけ近い場所に ACI をグループ化する。

  ACI の配置をディレクトリのルートポイントとディレクトリの主な分岐点に限定します。ACI をグループ化すると、ACI を総合したリストの管理に役立つだけでなく、ディレクトリ内の ACI の総数を最小限に留めるのにも役立ちます。

- 二重否定は使用しないようにする (バインド DN が `cn=Joe` と等しくない場合の書き込みの拒否など)。

  サーバーはこの構文を完全に理解できますが、管理者にとっては混乱の元となります。

ACI の制限事項

アクセス制御ポリシーを決定するときは、次の制限事項に注意してください。

- ディレクトリツリーが連鎖機能によって複数のサーバー上に分散されている場合は、アクセス制御文で使用できるキーワードにいくつかの制約が適用されます。
  - グループエントリ (`groupdn` キーワード) に依存する ACI は、グループエントリと同じサーバー上に置く必要があります。ダイナミックグループである場合は、そのグループのすべてのメンバーも同じサーバー上にエントリを持つ必要があります。スタティックグループである場合は、リモートサーバー上にメンバーのエントリを置くことができます。
  - ロール定義 (`roledn` キーワード) に依存する ACI は、ロール定義エントリと同じサーバー上に置く必要があります。ロールを持たせる予定のエントリも、すべて同じサーバー上に置く必要があります。

ただし、ターゲットエントリに格納された値と、バインドユーザーのエントリに格納された値のマッチングは可能です (`userattr` キーワードなどを使用)。ACI を持つサーバー上にバインドユーザーがエントリを持っていない場合も、通常どおりにアクセスに対する評価が行われます。

アクセス制御の評価の連鎖方法については、『Directory Server 管理ガイド』の「連鎖サフィックスのアクセス制御」を参照してください。

- CoS によって作成された属性を、すべての ACI キーワードで使用できるとは限りません。特に、`userattr` キーワードによって CoS で作成した属性を使用しないでください。アクセス制御規則が機能しなくなります。
- アクセス制御規則の評価は、常にローカルサーバー上で行われます。したがって、ACI キーワードで使用される LDAP URL で、サーバーのホスト名やポート番号を指定する必要はありません。指定しても、LDAP URL は無視されます。LDAP URL の詳細については、『Directory Server Administration Reference』の「LDAP URL Reference」を参照してください。
- サーバーが使用するメモリが利用可能な物理メモリに収めるためのキャッシュ設定は ACI キャッシュには適用されません。つまり、ACI の数が多すぎると、利用可能メモリが不足する可能性があります。
- プロキシ権限を与える場合、ユーザーに Directory Manager となるプロキシ権限を与えたり、Directory Manager にプロキシ権限を与えたりすることはできません。

SSL による接続のセキュリティ保護

ユーザーを識別するための認証スキーマと、情報を保護するためのアクセス制御スキーマを設計することに加え、サーバーとクライアントアプリケーションの間のネットワーク上でやり取りされる情報の完全性を保護する必要があります。

ネットワーク上で安全な通信を可能にするために、SSL (Secure Sockets Layer) 上で LDAP プロトコルと、DSML over HTTP の両方を使用することができます。SSL を設定して有効化すると、クライアントはセキュリティ保護された専用ポートに接続します。このポートとの SSL 接続が確立されると、すべての通信内容が暗号化されます。Directory Server は、Start TLS (Start Transport Layer Security) 制御もサポートしています。この制御を使用することで、クライアントは標準の LDAP ポート上での暗号化通信を開始できます。SSL と TLS の概要については、『Administration Server Administration Guide』の「Using SSL and TLS with Sun Java System Servers」を参照してください。

Directory Server は、SSL によるセキュリティ保護された接続と、SSL が適用されていない接続を同時にサポートします。

SSL は暗号化によって機密情報を保護し、チェックサムのハッシュによってデータの完全性を確保します。SSL 接続を確立すると、クライアントアプリケーションと Directory Server は、両者が設定に対して共通して持つ強力な暗号化アルゴリズム (暗号化方式) を選択します。Directory Server で使用できる暗号化方式は、次のとおりです。

- DES: 56 ビットのブロック暗号化方式
- 3DES (「トリプル DES」): 156 ビットのブロック暗号化方式
- RC2: 128 ビットのブロック暗号化方式 (または 40 ビットのエクスポート暗号化方式)
- RC4: 128 ビットのストリーム暗号化方式 (または 40 ビットのエクスポート暗号化方式)

暗号化方式は、次のいずれかのハッシュアルゴリズムと組み合わせて使用できます。

- MD5
- SHA-1

セキュリティ保護された接続が確立されると、SSL プロトコルによってサーバーはクライアントに証明書を送信します。公開鍵暗号化方式を使用して、クライアントは、その証明書がクライアントが信頼する認証局によって発行されたものであるかどうかを検証し、証明書の確実性を判断することができます。証明書の検証によって、クライアントは、第三者への成りすましによる介入者攻撃を防止することができます。

また、クライアントからの認証を要求するようにサーバーを設定することもできます。Directory Server は、証明書ベースと SASL ベースのクライアント認証をサポートしています。これらのメカニズムについては、165 ページの「適切な認証方法の選択」を参照してください。サーバーへのクライアント認証により、クライアントとサーバーとの間の通信が第三者によって傍受または妨害される可能性がなくなり、最大レベルのセキュリティが提供されます。

Directory Server 5.2 は Sun Crypto Accelerator ボードをサポートしています。この機能により、証明書ベースの認証を使用する SSL プロトコル通信のパフォーマンスが向上します。このボードは、SSL キー関連の計算を高速化します。クライアントアプリケーションが SSL 経由のバインド、検索、バインド解除を何度も繰り返す配備で有効な場合があります。キー関連の計算がパフォーマンスのボトルネックとならない場合、SSL アクセラレータボードを使用しても Directory Server のパフォーマンスが向上しない可能性があります。SSL アクセラレータボードがもっとも効果的なのは、クライアントが別のマシンから通信を確立する場合です。システムが SSL ベースの複数の接続を確立する場合、SSL キャッシングセッションによって RSA 操作の数が限定される可能性が高くなります。この場合、アクセラレータボードを利用するメリットも限定されてしまいます。Sun Crypto Accelerator ボードのインストール方法と設定方法については、『Directory Server 管理ガイド』の「Sun Crypto Accelerator ボードの使用」を参照してください。

Directory Server およびクライアントでの SSL の設定と有効化については、『Directory Server 管理ガイド』の「認証と暗号化の管理」を参照してください。

属性の暗号化

ここでは、属性暗号化機能について説明します。説明する内容は次のとおりです。

- 属性暗号化とは
- 属性暗号化の実装
- 属性の暗号化とパフォーマンス
- 属性暗号化の使用に関する注意点

属性暗号化とは

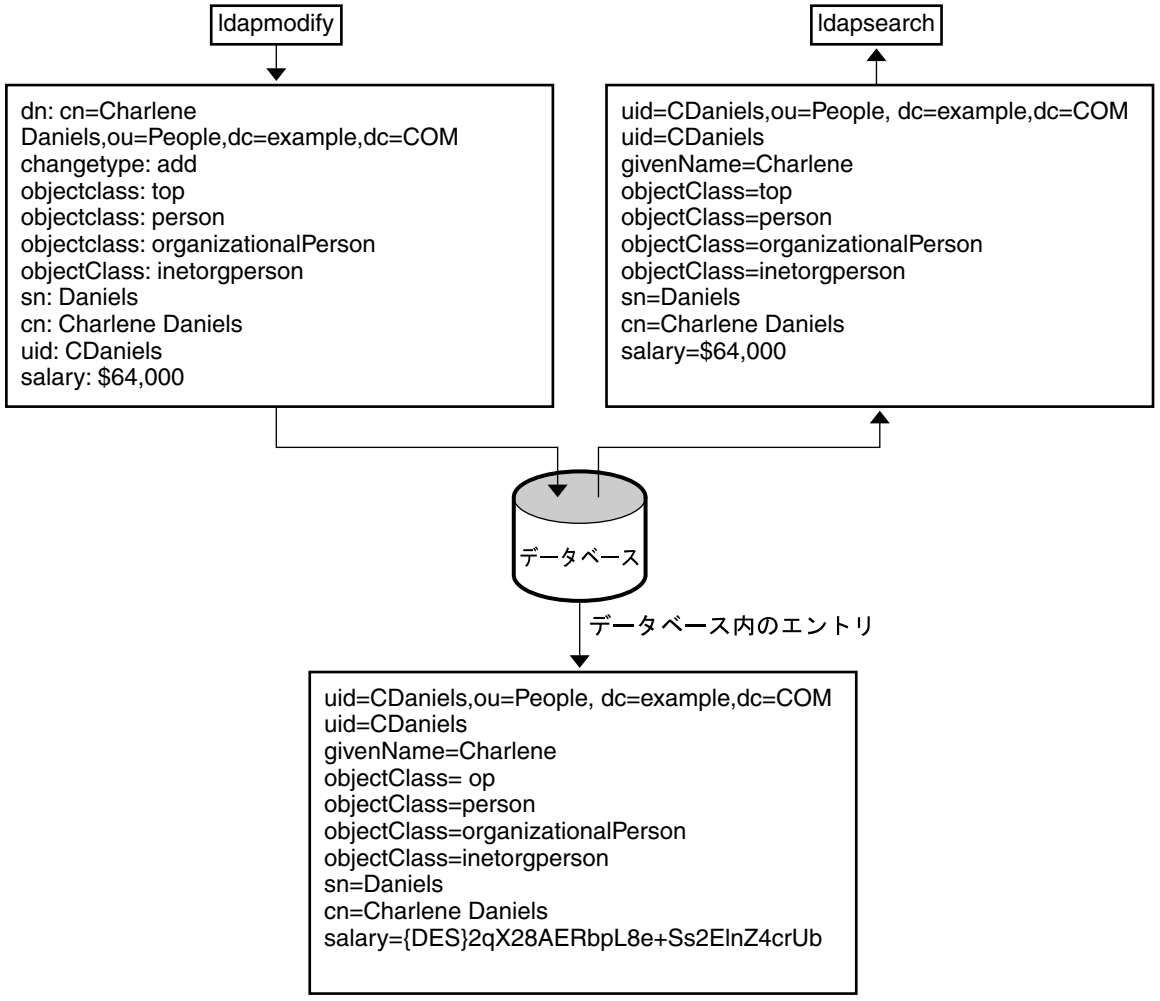
Directory Server では、簡易パスワード認証、証明書ベースの認証、SSL、プロキシ承認など、アクセスレベル (ディレクトリへの読み取りおよび書き込みアクセス時) でデータを保護するためのさまざまな機能が用意されています。しかし、データベースファイル、バックアップファイル、LDIF ファイルに記録されているデータの保護が必要になることも少なくありません。ディレクトリ内に 4 桁の暗証番号を記録している銀行を考えてみてください。保護されていないデータベースファイルにダンプが行われる場合、未認証のユーザーがこの機密情報にアクセスすることも考えられます。属性を暗号化する機能を使用することで、格納されている機密情報にユーザーがアクセスできないように保護できます。

属性暗号化では、どの属性を暗号化形式で格納するかを指定できます。これは、データベースレベルで設定されます。つまり、属性の暗号化を決定すると、データベース内のすべてのエントリでその属性は暗号化されます。属性の暗号化はエントリレベルではなく属性レベルで行われるため、エントリ全体を暗号化するには、すべての属性を暗号化する必要があります。

属性暗号化機能を使用することで、保管中のデータを保護できるだけでなく、データを別のデータベースに暗号化された状態でエクスポートすることができます。ただし、属性を暗号化する目的は、保管中またはエクスポート中の機密データを保護することであり、暗号化は常に可逆的です。このため、検索要求の結果として返された場合は、暗号化された属性は復号化されます。

図 7-1 は、データベースに追加されるユーザーエントリを示しています。ここで設定されている属性暗号化は、`salary` 属性を暗号化します。

図 7-1　属性暗号化のロジック



属性暗号化の実装

属性暗号化機能は、さまざまな暗号化アルゴリズムをサポートしているため、異なるプラットフォーム間での可搬性が確保されます。追加のセキュリティ対策として、属性暗号化はサーバーの SSL 証明書の非公開鍵を使用して専用の鍵を作成します。この鍵は、暗号化と復号化を行うときに使用されます。つまり、属性を暗号化するには、サーバーが SSL 経由で実行されている必要があります。SSL 証明書とその非公開鍵はデータベース内にセキュリティ保護された状態で格納されており、パスワードによって保護されています。サーバーへの認証では、この鍵データベースのパスワードが必要となります。つまり、この鍵データベースのパスワードにアクセスできるユーザーであれば、だれもが暗号化されたデータのエクスポートを承認されます。

データを暗号化するためにオンラインでインポートする場合、サーバーへの認証に使用される鍵データベースのパスワードはすでに指定されているため、2 回目には要求されなくなります。データをオフラインでインポートする場合、インポートするデータを暗号化するたびに Directory Server はパスワードを要求します。より機密性の高い操作であるデータの復号化では、エクスポートをオンライン、オフラインのどちらで行うかに関係なく、Directory Server は常に鍵データベースのパスワードを要求します。これにより、セキュリティはさらに高まります。

注 証明書または非公開鍵が変更されない限り、サーバーは同じ鍵を生成し続けます。このため、1 つのサーバーインスタンスから別のサーバーインスタンス (両者が同じ証明書を使用する場合) データを転送 (エクスポート後にインポート) することができます。

属性の暗号化とパフォーマンス

属性を暗号化することでデータのセキュリティは向上しますが、パフォーマンスに特定の影響が生じます。どの属性を暗号化するかを決定するときは、これを念頭に慎重に行い、特に機密性が高いと考えられる属性だけを暗号化してください。

インデックスファイルから機密性の高いデータに直接アクセスすることができるので、属性を完全に保護するために、暗号化された属性に対応するインデックスキーを暗号化する必要があります。インデックス付け自体がすでに Directory Server のパフォーマンスに影響しているため (インデックスキーの暗号化による負荷は含まれない)、データをインポートする、またはデータベースに初めて追加する前に属性暗号化を設定してください。こうすることで、暗号化された属性には最初からインデックスが付けられます。

属性暗号化の使用に関する注意点

属性暗号化機能を実装するときは、次の点を考慮してください。

- 一般に、属性暗号化の設定を変更するときは、データをエクスポートし、変更を加えた上で、新たに設定されたデータをインポートする必要があります。

  これにより、機能が喪失することなく、すべての設定変更が全体として適用されます。このような方法で行わなかった場合、一部の機能が喪失し、データのセキュリティが危険にさらされる可能性があります。

- 既存のデータベースに対する属性暗号化の設定を変更すると、パフォーマンスに大きく影響することがあります。

  たとえば、既存のデータが格納されたデータベースインスタンスについて考えてみましょう。このデータベースには、機密性の高い `mySensitiveAttribute` という属性を持つエントリが格納されています。この属性の値は、クリアテキストとしてデータベースとインデックスファイルに格納されています。`mySensitiveAttribute` 属性の暗号化を決定した場合、属性暗号化の設定を適用するためにサーバーがデータベースとインデックスファイルを更新する必要があり、データベースインスタンス内のすべてのデータはエクスポートされ、データベースに再びインポートされます。これにより、パフォーマンスに大きな影響が生じます。これは、この属性を最初から暗号化していた場合には回避できる影響です。

- 復号化された形式でデータをエクスポートするときに、誤ったパスワードを指定するとエクスポートが拒否されます。

  セキュリティ対策として、復号化された形式でデータをエクスポートするユーザーに対してサーバーはパスワードを要求します。ユーザーが誤ったパスワードを指定した場合、復号化されたデータのエクスポートは拒否されます。パスワードは、直接入力するだけでなく、SSL パスワードファイルと同じ構文を持つパスワードが記録されたファイルへのパスを指定できます。

- アルゴリズムの変更はサポートされていますが、正しく作成されていない場合、インデックス付け機能は失われる可能性があります。

  データの暗号化に使用するアルゴリズムを変更するときに、インデックス付けに関連する機能が喪失しないようにするには、データをエクスポートし、属性暗号化の設定を変更してから、データをインポートし直します。この手順どおりに行わない場合、内部暗号化アルゴリズムに基づいて作成されたインデックスは機能しなくなります。

  暗号化された属性の前には、適用された暗号化アルゴリズムを指定する暗号化方式が付けられ、データのインポートはサーバーの内部処理として行われます。このため、Directory Server でアルゴリズムを変更する前にデータを暗号化された形式でエクスポートできるようにすることで、セキュリティがさらに提供されます。

- サーバーの SSL 証明書を変更すると、暗号化されたデータを復号化できなくなります。

  属性暗号化機能は、専用鍵の生成にサーバーの SSL 証明書を使用し、暗号化と復号化の実行にはこの専用鍵を使用します。このため、暗号化されたデータを復号するには、この証明書が必要となります。事前にデータを復号化せずにサーバーの SSL 証明書を変更すると、そのデータを復号化できなくなります。これを回避するには、復号化された形式でデータをエクスポートし、証明書を変更してからデータをインポートし直す必要があります。

- 暗号化されたデータを伝送する、つまり、サーバーインスタンス間でエクスポートとインポートを行うには、両方のサーバーインスタンスが同じ証明書を使用する必要があります。

属性暗号化機能の手順については、『Directory Server 管理ガイド』の「属性値の暗号化」を参照してください。

エントリの安全なグループ化

ここでは、エントリのグループ化に関するセキュリティ上の問題について説明します。説明する内容は次のとおりです。

- ロールの安全な使い方
- CoS の安全な使い方

ロールの安全な使い方

セキュリティの状況によっては、ロールの使用が適していない場合があります。新しいロールを作成するときは、エントリへのロールの割り当てやエントリからのロールの削除がどの程度簡単にできるかを考慮します。ロールへのユーザーの追加やロールからの削除をユーザー自身が実行できることが望ましい場合もあります。たとえば、Mountain Biking という名前の同好会のロールがある場合は、興味のあるユーザーが自身を簡単に追加または削除できるようにする必要があります。

ただし、セキュリティの状況によっては、このような開放的なロールが適していない場合があります。たとえば、アカウントの無効化に関するロールがあるとします。デフォルトでは、アカウントの無効化に関するロールには、そのサフィックスに対して定義された ACI が含まれています。アカウントの無効化については、『Directory Server 管理ガイド』の「ユーザーとロールの無効化と有効化」を参照してください。サーバー管理者は、ロールを作成するときに、ロールへのユーザーの追加やロールからの削除をユーザー自身が実行できるようにするかどうかを決定します。

たとえば、ユーザー A が、管理されているロール MR を持っているとします。さらに、MR ロールが、コマンド行からアカウントの無効化によってロックされたとします。つまり、ユーザー A の nsAccountLock 属性は「true」として計算されるので、ユーザー A はサーバーにバインドできません。ただし、ユーザーがバインド済みで、MR ロールに関して現在ロックされているという通知を受けたとします。ユーザーの行為を禁止する ACI がない場合は、ユーザーは、自分のエントリから nsRoleDN 属性を削除し、自分でロックを解除できます。

ユーザーが nsRoleDN 属性を削除できないようにするには、ACI を適用する必要があります。フィルタを適用したロールを使用する場合、ユーザーが属性を変更してフィルタが適用されたロールを放棄することを防ぐために、フィルタの一部を保護する必要があります。フィルタを適用したロールで使用される属性の追加、削除、変更をユーザーが実行できないようにし、フィルタの属性値が計算によって決定される場合は、フィルタの属性値に影響する可能性のあるすべての属性も保護する必要があります。入れ子のロールは、フィルタを適用したロールと管理されているロールから構成されることがあるため、入れ子のロールを構成する各ロールについても上記注意点を考慮する必要があります。

CoS の安全な使い方

読み取り用のアクセス制御は、エントリの実際の属性と仮想属性の両方に適用されます。サービスクラスメカニズムによって生成された仮想属性は、通常の属性として読み取ることができるので、読み取り保護も同様の方法で指定します。

ただし、CoS 値をセキュリティで保護するには、定義エントリ、テンプレートエントリ、ターゲットエントリなど、使用するすべての情報のソースを保護する必要があります。これは更新処理でも同様です。情報のソースから生成された値を保護するために、各情報のソースに対する書き込みを制御する必要があります。

次に、各 CoS エントリのデータに読み取りおよび書き込み保護を設定する際の一般的な原則について説明します。個々の ACI (アクセス制御命令) の定義手順については、『Directory Server 管理ガイド』の「アクセス制御の管理」を参照してください。

CoS 定義のエントリの保護

CoS 定義のエントリには、生成された属性の値は含まれません。このエントリは、値を検索するための情報を提供します。CoS 定義のエントリを読み取ると、値を含むテンプレートエントリの検索方法がわかり、このエントリへの書き込みによって、仮想属性の生成方法を変更できます。

したがって、CoS 定義のエントリに読み取りと書き込みの両方のアクセス制御を定義する必要があります。

CoS テンプレートエントリの保護

CoS テンプレートエントリには、生成された CoS 属性の値が含まれます。したがって、少なくともテンプレートの CoS 属性の読み取りと更新を保護する必要があります。

ポインタ CoS の場合は、名前の変更が禁止されているテンプレートエントリが 1 つ存在します。通常、テンプレートエントリ全体を保護するのがもっとも簡単な方法です。

クラシック CoS では、すべてのテンプレートエントリは、定義エントリで指定された共通の親を持ちます。この親エントリにテンプレートを格納するだけで、親エントリに対するアクセス制御によってテンプレートが保護されます。ただし、親の下のほかのエントリにアクセスする場合は、テンプレートエントリを個別に保護する必要があります。

間接 CoS の場合は、アクセスする必要があるユーザーエントリを含む、ディレクトリ内の任意のエントリにテンプレートを指定できます。必要に応じて、ディレクトリ全体の CoS 属性に対するアクセスを制御するか、またはテンプレートとして使用される各エントリの CoS 属性のセキュリティを保護するかを選択できます。

CoS のターゲットエントリの保護

仮想 CoS 属性が生成される、CoS 定義の適用範囲内のすべてのエントリも値の算出に役立ちます。

CoS 属性がターゲットエントリにすでに存在する場合は、デフォルトでは、CoS メカニズムはこの値を上書きしません。この動作を変更する場合は、ターゲットエントリを上書きするように CoS を定義するか、すべてのターゲットエントリで CoS 属性を保護します。手順については、『Directory Server 管理ガイド』の「ID とロールの管理」を参照してください。

間接 CoS とクラシック CoS は、ターゲットエントリの指示子属性に依存します。この属性は、使用するテンプレートエントリの DN または RDN を指定します。この属性を保護する場合は、CoS の適用範囲全体でグローバルに保護するか、または各ターゲットエントリで必要に応じて個別に保護する必要があります。

その他の従属関係の保護

生成されたその他の CoS 属性およびロールに関して仮想 CoS 属性を定義することができます。仮想 CoS 属性を確実に保護するために、これらの従属関係を理解し保護する必要があります。

たとえば、ターゲットエントリの CoS 指示子属性には `nsRole` を指定できます。したがってロール定義も保護する必要があります。詳細は、194 ページの「エントリの安全なグループ化」を参照してください。

一般に、仮想属性値の算出に関係する属性またはエントリには、読み取りおよび書き込みアクセス制御を設定します。このため、複雑な従属関係は、十分に計画してから設定するか、以後のアクセス制御の実装の複雑さを軽減できるように簡素化する必要があります。その他の仮想属性との従属関係を最小限に抑えると、ディレクトリのパフォーマンスを向上させ、管理作業を削減することができます。

設定情報のセキュリティ保護

ほとんどの配備では、ルート DSE エントリ (長さがゼロの DN が指定されたベースオブジェクト検索で返されるエントリ)、または cn=config、cn=monitor、または cn=schema の下のサブツリーでは、追加のアクセス制御は必要ありません。ルート DSE エントリとこれらのサブツリーには、Directory Server が自動的に生成する属性が含まれ、LDAP クライアントは Directory Server の機能と設定を判断するときにこの属性を使用します。

しかし、namingContexts というルート DSE エントリ属性には、各 Directory Server データベースのベース DN のリストが含まれます。このリストに加え、これらの DN は cn=config および cn=monitor の下のマッピングツリーエントリにも格納されます。セキュリティ上の理由から 1 つまたは複数のサブツリーを非表示に設定して設定情報を保護するには、次のような配置が必要です。

- ACI 属性を、非表示にするサブツリーのベースにあるエントリに配置する。
- ルート DSE エントリに含まれる ACI を namingContexts 属性に配置する。
- ACI を cn=config サブツリーと cn=monitor サブツリーに配置する。

その他のセキュリティ関連資料

セキュリティ保護されたディレクトリの設計については、以下の資料を参照してください。

- Sun 開発者向けのセキュリティ関連資料
  http://developers.sun.com/techtopics/security/index.html
- 『Understanding and Deploying LDAP Directory Services』
  (T. Howes、M. Smith、G. Good 著、Macmillan Technical Publishing 発行、1999 年)
- SecurityFocus.com
  http://www.securityfocus.com/
- コンピューター緊急事態対策チーム (CERT) 管理センター
  http://www.cert.org/

- CERT セキュリティ向上モジュール
  http://www.cert.org/security-improvement/

第 8 章

Directory Server の監視

Directory Server の配備を成功させるには、効果的な監視とイベント管理の戦略が必要です。この戦略は、監視対象となるイベント、使用するツール、そのイベントが発生した場合に実行するアクションを定義します。発生しがちなイベントについて対策を用意しておくことで、サービスの停止やレベル低下を防ぎ、可用性とサービスの品質を向上することができます。

監視とイベント管理の戦略には、レプリケーション設定などの具体的なアーキテクチャコンポーネントを含める必要があり、また、システムとネットワークの監視も含める必要があります。この章では、効果的な監視戦略に何を含めるかについて説明し、Directory Server の監視機能について解説します。

注　システムとネットワークの監視は Directory Server に固有の話題ではないので、この章では詳しく触れません。

この章は、次の節から構成されています。

監視およびイベント管理戦略の定義

ここでは、監視とイベント管理の戦略を定義するための手順の概要について説明します。このプロセスは次の段階に分けることができます。

1. 適切な監視ツールを選択します。オペレーティングシステムのツール、Directory Server の監視ツール、サードパーティ製の監視ツールを使用できます。
2. 重要な監視対象をディレクトリアーキテクチャから選択します (多くの場合、これはサイズ設定属性と調整属性)。
3. パフォーマンス指標を監視する場合に、イベントまたはアラーム状態を始動する条件を定義します。つまり、パフォーマンスの受容可能レベル、または各パフォーマンス指標に対する処理を定義します。
4. アラーム状態が発生したときに実行するアクションを決定します。

Directory Server の監視ツール

ここでは、Directory Server で使用できる監視ツールと、Directory Server のアクティビティの監視に利用できるその他のツールについて、概要を説明します。次節で説明する主要パフォーマンス指標は、これらのツールを使用して、またはツールを組み合わせて、すべて監視できます。

- コマンド行ツール

  コマンド行監視ツールには、ディスク使用量などのパフォーマンスを監視するオペレーティングシステムに固有のツール、ディレクトリに格納されているサーバー統計を収集する `ldapsearch` などの LDAP ツール、サードパーティ製ツール、カスタムシェル、Perl スクリプトが含まれます。

- Directory Server ログ

  Directory Server で使用できるアクセスログ、監査ログ、エラーログには、監視に利用できる情報が豊富に含まれます。これらのログを手動で監視したり、カスタムスクリプトを使用して解析することで、配備に関連する監視情報を抽出できます。Directory Server Resource Kit には、ログ分析ツール、`logconv.pl` が用意されています。これを使って、Directory Server アクセスログを分析することができます。このログ分析ツールは、利用率統計を抽出し、重要なイベントの発生をカウントします。このツールの詳細については、『Directory Server Resource Kit Tools Reference』の「The Log Analyzer Tool」を参照してください。ログファイルの表示と設定については、『Directory Server 管理ガイド』の「ログファイルを使用した Directory Server の監視」を参照してください。

- Directory Server コンソール

  Directory Server コンソール では、ディレクトリの動作をリアルタイムにグラフィカルユーザーインタフェースで監視できます。このコンソールは、リソースサマリー、現在のリソース使用状況、接続状態、グローバルデータベースキャッシュ情報など、一般的なサーバー情報を提供します。また、データベースのタイプと状態、エントリキャッシュ統計、キャッシュ情報、データベース内の各インデックスに関連する情報など、一般的なデータベース情報も提供します。さらに、コンソールには接続と各連鎖サフィックスで実行される操作に関する情報も表示されます。

- レプリケーション監視ツール

  Directory Server のレプリケーション監視ツールを使用すると、次の操作を実行できます。

  - マスターレプリカと 1 つまたは複数のコンシューマレプリカとの間の同期状態を監視する
  - 複数の異なるレプリカの間で同じエントリを比較することで、レプリケーションの状態を確認する
  - 完全なレプリケーショントポロジを描写する (これは複雑なディレクトリ配備で特に有用となる)

- SNMP (Simple Network Management Protocol)

  Directory Server は、SNMP (Simple Network Management Protocol) による監視をサポートしています。SNMP は、グローバルネットワーク制御と監視のための標準メカニズムで、ネットワーク管理者はネットワーク管理作業を一元的に行えます。

  SNMP と Directory Server がサポートする SNMP 管理対象オブジェクトについては、208 ページの「SNMP による監視」を参照してください。SNMP の設定方法については、『Directory Server 管理ガイド』の「SNMP を使用した Directory Server の監視」を参照してください。

Directory Server の監視

監視とイベント管理の戦略を定義する上でもっとも重要な手順は、ディレクトリアーキテクチャ上の 1 つまたは複数のコンポーネントで重要な監視対象を決定することです。何を監視対象とし、それをどの程度監視するかは、配備によって大きく異なります。

ここでは、監視対象となるパフォーマンス指標について説明します。説明する内容は次のとおりです。

- Directory Server アクティビティの監視
- データベースアクティビティの監視
- ディスクの状態の監視
- レプリケーションアクティビティの監視
- インデックス付けの効率の監視
- セキュリティの監視

注

- ディレクトリの `cn=monitor` 分岐内の監視情報に対して `ldapsearch` コマンドを実行するときは、ユーザーは認証を受け、情報にアクセスするための適切な権限を持っている必要があります。監視戦略を定義するには、このようなアクセス権限の取得が前提条件となります。
- システムが効率的に稼動し、サーバーに悪影響を生じていないことを確認する上で、Directory Server が稼動するオペレーティングシステム環境を監視することが重要です。ただし、これは Directory Server に固有の話題ではないので、この章では詳しく触れません。詳細は、オペレーティングシステムのマニュアルを参照してください。

Directory Server アクティビティの監視

Directory Server では、さまざまな方法でサーバーの状態を監視できます。たとえば、次のような方法があります。

- Sun Java System サーバーコンソール の「サーバーおよびアプリケーション」タブには、インストール日、バージョン、サーバーの状態 (起動されているかどうか)、ポート番号など、サーバーに関する一般的な情報が表示されます。
- Directory Server コンソール では、これ以外の情報も確認することができます。このコンソールの「状態」タブには、次の情報が表示されます。
  - 起動時刻とサーバーの現在の時刻。
  - 接続、開始および終了された操作、クライアントに送信されたエントリとバイト数の詳細を示すリソースサマリー。

- アクティブスレッド、開いている接続と利用できる接続、クライアントからの読み取りを待つスレッドの数、使用中のデータベースの数など、現在のリソース使用状況に関する情報。
- 接続が開かれた日時、開始および終了された接続の数、クライアントがサーバーへのバインドに使用する識別名、接続の状態 (ブロック、または非ブロック)、接続の種類 (LDAP または DSML) など、開いているすべての接続に関する情報。

Directory Server コンソール で利用できるパフォーマンスカウンタについては、『Directory Server 管理ガイド』の「コンソールを使用したサーバの監視」を参照してください。

- `cn=monitor` エントリに対して `ldapsearch` コマンドを実行すると、Directory Server コンソールの「状態」タブに表示される情報と同じ情報を取得できます。一部の情報は、`ldapsearch` コマンドを実行するユーザーがディレクトリ管理者としてバインドしている場合にだけ表示されます。このアクセス制限を解除するには、この情報に関連付けられているアクセス制御を設定し直す必要があります。`cn=monitor` の下に格納されているパフォーマンスカウンタについては、『Directory Server 管理ガイド』の「コマンド行からのサーバーの監視」を参照してください。
- `ps` コマンドを実行すると、現在稼動しているプロセスが表示されます。これにより、Directory Server `slapd` デーモンの稼動状態を確認することができます。詳細は、`ps(1)` マンページを参照してください。
- `ldapsearch` コマンド行ユーティリティを使用して、Directory Server が要求に応答しているかどうかを確認することができます。時間のかかる、インデックス付けされていない検索を避け、ベースレベルの検索を行います。複数のデータベースがあるときは、各データベースサフィックスへの LDAP クエリを作成し、データベースがオンライン状態にあり、応答しているかどうかを確認することができます。
- Directory Server のアクセスログを使用して、サーバーの動作を監視し、サーバーが稼動しているかどうかを確認することができます。アクセスログと接続コードについては、『Directory Server Administration Reference』の「Access Log Content」および「Common Connection Codes」を参照してください。
- Directory Server のエラーログには、サーバーの起動と停止の状態が記録されます。この情報に基づいて、サーバーが稼動しているかどうかを確認することができます。ログファイルの表示と設定については、『Directory Server 管理ガイド』の「ログファイルを使用した Directory Server の監視」を参照してください。

データベースアクティビティの監視

データベースのアクティビティを監視することは、データベースがオンラインの状態にあり、必要になったときに確実にアクセスできることを確認する上で有効です。データベースの監視情報にアクセスするには、cn=config 分岐の特定の領域に対して ldapsearch コマンドを実行します。表 8-1 は、提供される監視情報の種類と、cn=config 分岐の対応領域を示しています。

表 8-1　cn=config 内のデータベース監視情報ソース

情報の領域	対応する cn=config の分岐
データベースの一般情報	cn=database,cn=monitor,cn=ldbm database, cn=plugins,cn=config
データベースのキャッシュ情報	cn=monitor,cn=ldbm database,cn=plugins,cn=config
特定のデータベースインスタンスの情報	cn=monitor,cn=*suffixName*,cn=ldbm database, cn=plugins,cn=config
連鎖サフィックスの情報	cn=monitor,cn=*suffixName*,cn=chaining database, cn=plugins,cn=config

次に、データベース監視情報の領域について、さらに詳しく説明します。

- cn=database,cn=monitor,cn=ldbm database,dn=plugins,cn=config 分岐は、キャッシュへのアクセス、トランザクション、ロック、ログの情報を提供します。Directory Server の設定属性の完全なリストについては、『Directory Server Administration Reference』の第 2 章「Server Configuration Reference」を参照してください。

  監視する一般データベース情報は、配備するディレクトリに固有の要件によって異なります。たとえば、Directory Server が複数のトランザクションを同時に処理することが多い場合は、一度に並行して処理できるトランザクションの最大数の監視が必要になるかもしれません。この値 (nsslapd-db-max-txns 属性によって定義される) が、許容されている最大トランザクション数 (nsslapd-db-configured-txns 属性によって定義される) に達した場合に操作の失敗を防ぐには、許容される最大トランザクション数を増やす必要があります。

- データベースのキャッシュとインデックス付けのパフォーマンスを監視するときは、Directory Server コンソールの「状態」タブを使用するか、次の分岐で ldapsearch コマンドを実行します。

  cn=monitor,cn=ldbm database,cn=plugins,cn=config および cn=monitor,cn=*suffixName*,cn=ldbm database,cn=plugins,cn=config

  関連する設定属性の完全なリストについては、『Directory Server Administration Reference』の「Server Configuration Reference」を参照してください。

- `cn=monitor,cn=`*suffixName*`,cn=chaining database,cn=plugins,cn=config` 分岐は、接続、LDAP、および実行中のバインド操作、バインド解除操作に関する情報を提供します。この情報は、Directory Server コンソールの「状態」タブにも表示されます。

ディスクの状態の監視

ディスク容量を効果的に監視することで、ディスクリソースの不足に関連する問題を防止できます。`cn=disk,cn=monitor` エントリにより、次の監視情報が提供されます。

- データベースインスタンスへのパス。同一ディスクに複数のデータベースインスタンスが存在する場合、またはインスタンスが同一ディスク上の複数のディレクトリを参照する場合は、短いパス名が表示される。
- サーバーが利用できるディスク容量 (M バイト単位)。
- ディスクの状態。正常 (normal)、残量低下 (low)、残量なし (full) で表される。この状態は、利用可能容量、および「low」、「full」の警告を始動するためのしきい値によって決定される。この制限に達するとディレクトリは更新を受け付けなくなるため、ディスク残量なしのしきい値を監視することは特に重要である。

`cn=disk,cn=monitor` 属性、および設定可能なディスク容量の 2 つのしきい値については、『Directory Server Administration Reference』の「Server Configuration Reference」を参照してください。

レプリケーションアクティビティの監視

レプリケーション状態の監視は、グローバルな監視戦略では重要な要素です。レプリケーションの問題の可能性を早く認識できるほど、問題を迅速に解決し、正しいレプリケーション動作を再開できます。

レプリケーション機能のさまざまな側面を監視する 3 つのレプリケーション監視ツールが用意されています。レプリケーション監視ツールは LDAP クライアントとして機能し、標準接続またはセキュリティ保護された接続 (LDAPS) を介して使用できます。次のレプリケーション監視ツールがあります。

- insync
- entrycmp
- repldisc

insync

insync ツールは、マスターレプリカと 1 つまたは複数のコンシューマレプリカとの間の同期状態、またはレプリケーションの遅延を示します。このレプリケーションの遅延は、マスター上のデータと比較して、コンシューマ上のデータがどの程度正確かを示します。

entrycmp

entrycmp ツールを使用することで、複数の異なるサーバー上の同一エントリを比較することができます。マスターレプリカからエントリが取得され、エントリの nsuniqueid を使用して指定コンシューマから同じエントリを取得します。エントリの属性と値が比較され、どちらも同じであれば、これらのエントリは同一であると見なされます。

注 insync ツールと entrycmp ツールを実行するマシンは、指定されたすべてのホストにアクセスできる必要があります。ファイアウォール、VPN、またはネットワーク設定上のその他の理由からホストにアクセスできない場合、これらのツールを使用することは困難になります。同じ理由で、レプリケーション監視ツールを実行する前に、すべてのサーバーが起動され、稼動していることを確認してください。

repldisc

repldisc ツールを使用することで、レプリケーショントポロジを推測することができます。トポロジの推測は、1 つのサーバーから始まり、トポロジ内のすべての既知のサーバーをグラフ化することで行われます。次に、repldisc ツールはトポロジを示す隣接マトリクスを出力します。このレプリケーショントポロジ推測ツールは、配備したグローバルトポロジを思い出すことが難しい、大規模で複雑な配備で便利です。

注

- レプリケーション監視ツールを使用するときは、ホストをすべてを記号名で指定するか、またはすべてを IP アドレスで指定します。2 つを組み合わせた場合、問題が生じる可能性が高くなります。
- SSL が有効な状態でレプリケーション監視ツールを実行するときは、ツールの実行対象サーバーには、トポロジ内の他のサーバーが使用するすべての証明書のコピーが保持されている必要があります。
- これらのツールは LDAP クライアントベースであるため、サーバーへの認証と、cn=config に対する読み取りアクセス権を持つバインド DN が必要となります。これらのツールの詳細な設定方法と、SSL が有効な状態での使用については、『Directory Server 管理ガイド』の「レプリケーション状態の監視」を参照してください。

レプリケーション監視ツールについては、『Directory Server Man Page Reference』を参照してください。

インデックス付けの効率の監視

インデックス付けは、読み取りのパフォーマンスには肯定的な影響を持ち、書き込みのパフォーマンスには否定的な影響を持ちます。このため、読み取りと書き込みのパフォーマンスのバランスを適切に保つには、インデックス付けの効率を監視することが重要です。効果的なインデックス付け戦略により、不要なインデックスをなくし、クライアントアプリケーションで必要なインデックスだけを維持することができます。

インデックス付けの効率は、次の方法で監視できます。

- アクセスログを調べ、インデックスのない検索が完了するまでの時間を監視することで、不釣合いな時間を消費した、インデックスのない検索を識別することができます。インデックスのない検索はログファイル内では `notes=U` によって識別され、長時間の検索では `etime` の値が高くなります。

  アクセスログには、検索とそのフィルタに関する情報も記録されています。これに基づいて、インデックスを作成することでパフォーマンスを向上できるかどうかを判断できます。Directory Server Resource Kit には、ログ分析ツール、`logconv.pl` が用意されています。これを使って、Directory Server アクセスログを分析することができます。このツールの詳細については、『Directory Server Resource Kit Tools Reference』の「The Log Analyzer Tool」を参照してください。

- Directory Server コンソールの「状態」タブでは、サフィックスまたは連鎖サフィックスごとにもっとも頻繁に使用されたインデックスを監視できます。これは、インデックスの使用が何回試みられ、何回成功したかを示します。`cn=monitor,cn=`*suffixName*`,cn=ldbm database,cn=plugins,cn=config` 分岐に対して、`ldapsearch` コマンドを実行しても同じ監視情報が得られます。

  設定されているインデックスのリストは、Directory Server コンソールの「設定」タブに表示されます (「データ」> 「*suffixName*」ノードの下)。前述の頻繁に使用されるインデックスと、設定されているインデックスのリストを比較することで、リソースを不必要に消費しているインデックスを特定し、それを削除するかどうかを決定できます。エントリにインデックス付けされた属性が含まれ、インデックスが使用されていない場合、そのインデックスを削除するとパフォーマンスが向上します。

アクセスログの内容と接続コードについては、『Directory Server Administration Reference』の「Access Log Content」および「Common Connection Codes」を参照してください。Directory Server の設定属性の完全なリストについては、『Directory Server Administration Reference』の「Server Configuration Reference」を参照してください。

セキュリティの監視

配備のセキュリティの監視は、安全にアクセスできるディレクトリにとって不可欠です。受容可能なセキュリティレベルを維持する上で、Directory Server を次のように監視することをお勧めします。

- バインド試行の失敗数の監視は、ディレクトリへの侵入試行に関する警告となります。SNMP エージェントが稼動している場合は、`cn=snmp,cn=config` の下にある SNMP 管理対象オブジェクトカウンタ `dsBindSecurityErrors` に対して `ldapsearch` を実行することで、バインド試行の失敗を監視できます。
- アクティビティが行われていない、開いている接続の数の監視は、サービス拒否攻撃の可能性に関する警告となります。現在の接続数と完了した操作の数は、Directory Server コンソールの「状態」タブ、または `cn=monitor` の下にある属性を検索することで確認できます。
- 実効権限機能を使用することで、クライアントはディレクトリのエントリと属性に対して持つアクセス制御権を照会できます。ユーザーがアクセス権を照会できるようになったことで、ユーザーの管理、アクセス制御ポリシーの検証、設定内容の決定が容易になりました。

  実効権限機能は、日々の操作としてではなく、定期的に使用されることが多いと考えられます。実効権限については、184 ページの「実効権限に関する情報の取得」を参照してください。

SNMP による監視

SNMP はグローバルなネットワーク制御と監視のための標準メカニズムです。SNMP を使用することで、ネットワーク管理者はネットワーク管理作業を一元的に行い、さまざまなデバイスをリアルタイムで監視することができます。ここでは、SNMP を使用して Directory Server の操作を監視する方法について説明します。説明する内容は次のとおりです。

- SNMP について
- Directory Server での SNMP 監視

SNMP について

SNMP は、ネットワークアクティビティに関するデータを交換するために使用されるプロトコルです。SNMP を使用すると、管理対象デバイスと、ユーザーがネットワークをリモート管理する NMS (ネットワーク管理ステーション) の間で、データが移動します。管理対象デバイスは、ホスト、ルータ、Directory Server など、SNMP が稼動するすべてのデバイスです。NMS とは通常、1 つ以上のネットワーク管理アプリケーションが稼動する強力なワークステーションを指します。ネットワーク管理アプリケーションは、管理対象デバイスに関する情報 (どのデバイスが有効または無効か、どのエラーメッセージをいくつ受け取ったか、など) を、多くの場合はグラフィカルに表示します。

情報は、サブエージェントとマスターエージェントの 2 種類のエージェントを使用して、NMS と管理対象デバイスの間で転送されます。サブエージェントは、管理対象デバイスに関する情報を収集し、その情報をマスターエージェントに渡します。Directory Server にはサブエージェントがあります。マスターエージェントは、各種サブエージェントと NMS の間で情報を交換します。マスターエージェントは、通信先のサブエージェントと同じホストマシン上で動作します。

ホストマシンには、複数のサブエージェントをインストールできます。たとえば、Directory Server、Application Server、および Messaging Server をすべて同じホスト上にインストールした場合、これらの各サーバーのサブエージェントは、同一のマスターエージェントと通信します。マスターエージェントは、Administration Server と一緒にインストールされます。

照会可能な SNMP 属性の値は、管理対象デバイス上に保持され、必要に応じて NMS に報告されます。各属性または変数は管理対象オブジェクトと呼ばれ、エージェントはこのオブジェクトにアクセスし、これを NMS に送ることができます。管理対象オブジェクトはすべて、ツリーのような階層を持つデータベースである MIB (管理情報ベース) に定義されています。この階層の最上位には、ネットワークに関する一般的な情報が含まれています。下位の各階層には、個々のネットワーク領域に関するより詳細な情報が含まれています。

SNMP は、PDU (プロトコルデータ単位) の形式でネットワーク情報を交換します。PDU には、管理対象デバイス上に格納された変数に関する情報が含まれています。これらの変数は管理対象オブジェクトとも呼ばれ、必要に応じて NMS に報告される値と名前を保持しています。NMS と管理対象デバイスとの間の通信は、次の 2 つのどちらかの方法で行われます。

- NMS 主導の通信
- 管理対象デバイス主導の通信

次に、Directory Server がサポートしている NMS 主導の通信について説明します。

NMS 主導の通信

これは、NMS と管理対象デバイス間の通信で、もっとも一般的なタイプの通信です。このタイプの通信では、NMS は管理対象デバイスから情報を要求するか、または管理対象デバイス上に格納された変数の値を変更します。

NMS によって開始される SNMP セッションは、次のように実行されます。

1. NMS が、どの管理対象デバイスおよびオブジェクトに監視が必要であるかを判断します。
2. NMS が、マスターエージェントを介して、PDU を管理対象デバイスのサブエージェントに送ります。この PDU は、管理対象デバイスから情報を要求するか、または管理対象デバイス上に格納された変数の値を変更するようサブエージェントに指示します。
3. 管理対象デバイスのサブエージェントが、マスターエージェントから PDU を受け取ります。
4. NMS から送られた PDU が変数に関する情報を要求するものである場合、サブエージェントはマスターエージェントにその情報を送り、マスターエージェントは別の PDU として NMS に情報を送り返します。次に、NMS が、この情報を文字またはグラフィック形式で表示します。

   NMS から送られた PDU が、変数に値を設定するようサブエージェントに要求するものである場合、サブエージェントは変数値を要求のとおりに設定します。

Directory Server での SNMP 監視

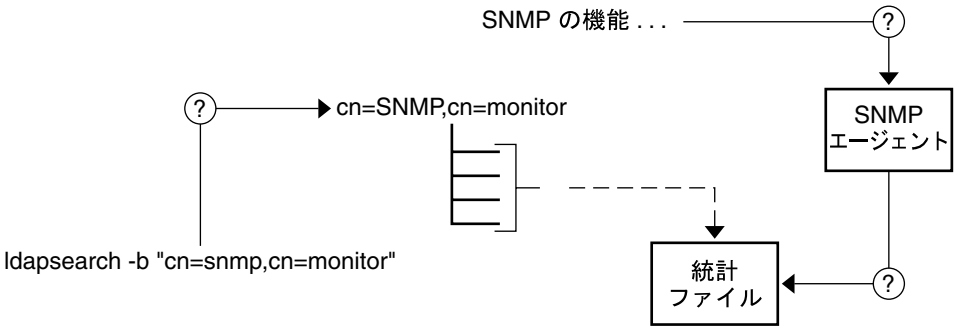
Directory Server は、次の 2 つの方法による SNMP 監視をサポートしています。

- SNMP エージェント経由の監視。SNMP 属性は統計ファイルにマッピングされ、SNMP エージェントが照会されるたびにこのファイルが読み取られます。Directory Server が稼動していない場合は、この統計ファイルは存在しません。
- `ldapsearch` コマンド行ユーティリティによる監視。SNMP 属性は `cn=snmp,cn=monitor` エントリの下に格納されます。次の `ldapsearch` コマンドを実行すると、Directory Server のすべての SNMP 属性がリスト表示されます。

  ```
  ldapsearch -h host -p port -s base -b "cn=snmp,cn=monitor"
   "objectclass=*"
  ```

図 8-1 は、Directory Server から SNMP 監視情報を取得する 2 つの方法を示しています。

図 8-1　Directory Server での SNMP 監視



MIB の定義場所および SNMP の使用方法については、『Directory Server 管理ガイド』の「SNMP を使用した Directory Server の監視」を参照してください。

Directory Server がサポートする SNMP 管理対象オブジェクトは、Directory Server Monitoring MIB RFC 2605 の初期草案に基づいています。SNMP エージェントから返される SNMP 操作管理オブジェクトは、`ldapsearch` コマンドによって返される SNMP 監視属性と同じです。これらの属性については、『Directory Server Administration Reference』の「Monitoring Attributes」を参照してください。SNMP エージェントから返される属性の名前には、`ds` という接頭辞が付けられます。

操作管理オブジェクトに加え、Directory Server は、監視対象のサーバーとそのピアサーバーの間の対話に関連する管理対象オブジェクトと、サーバーの現在のインストールに関する情報を含むエンティティ関連の管理対象オブジェクトをサポートしています。これらのオブジェクトについては、『Directory Server Administration Reference』の「Interactions Table of Supported SNMP Managed Objects」および「Entity Table of SNMP Supported Managed Objects」を参照してください。

第 9 章

参考のアーキテクチャとトポロジ

ディレクトリの配備を計画するときは、いくつかの事項を考慮する必要があります。もっとも重要な事項には、データの物理的な配置場所、このデータをどこに、また、どのようにレプリケートするか、障害を最小化するにはどうすればよいか、障害が発生した場合にどのように対応するか、などが含まれます。この章で説明するアーキテクチャ戦略は、これらの事項について考える上でのガイドラインとなります。

この章は、次の節から構成されています。

障害と復元について

障害発生時にサービスの中断を最小限にとどめるための戦略を準備しておくことが重要です。ここでいう障害とは、必要とされる最小限のサービスを Directory Server が提供できなくなる原因として定義されます。ここでは、配備で発生する障害の原因を特定し、障害に迅速に対応できるように、障害のさまざまな発生原因について説明します。

障害は、主に次の 2 つに分けられます。

- システムが使用できなくなる
- システムが信頼できなくなる

システムが使用できなくなることには、次のような原因があります。

- ネットワークの問題 : ネットワークが停止している、速度が低下している、または断続的になる。
- プロセス (`slapd`) の問題 : プロセスが停止している、ビジーである、再起動している、または動作不良を起こす。

- ハードウェアの問題 : ハードウェアが稼動していない、障害が発生した、または再起動している。

システムが信頼できなくなることには、次のような原因があります。

- レプリケーションに失敗または遅延が生じ、データが古くなったり、同期がとれなくなる。
- システムが過度のビジー状態になる : 読み取りまたは書き込み操作が過剰に行われ、データの信頼性が失われる。

書き込み可能なサーバーが利用できなくなった場合に、ディレクトリ内のデータを追加、変更する機能を維持するには、書き込み操作が代替サーバーを経由する必要があります。書き込み操作のルーティングにはさまざまな方法があり、Sun Java System Directory Proxy Server の使用もそれに含まれます。

ディレクトリ内のデータを読み取る機能を維持するには、適切なロードバランス戦略を配備する必要があります。読み取りの負荷を複数のコンシューマレプリカに分散するには、ソフトウェアとハードウェアの両方のロードバランスソリューションを利用できます。それぞれのソリューションには、各レプリカの状態を特定し、ロードバランストポロジの中でどのような役割を果たすべきかを管理する機能 (完全性と精度はそれぞれ異なる) があります。

ディレクトリの内容をレプリケートすると、Directory Server の可用性が向上します。信頼できるレプリケーショントポロジを構築することで、障害が発生した場合でも、データセンターにアクセスするすべてのクライアントは最新のデータに確実にアクセスできます。

次に、読み取り操作と書き込み操作のための障害戦略について説明します。これは、レプリケーショントポロジにも関連します。

バックアップ戦略の策定

データの破損や喪失をともなう障害では、データの最新のバックアップが不可欠になります。最新のバックアップが得られない場合、障害が発生したマスターを別のマスターから初期化し直す必要があります。データのバックアップについては、『Directory Server 管理ガイド』の「データのバックアップ」を参照してください。

ここでは、バックアップと復元の戦略を計画する上で考慮すべき事項について簡単に説明します。

バックアップ方法の選択

Directory Server では、バイナリバックアップ (db2bak) と、LDIF ファイルへのバックアップ (db2ldif) という 2 つの方法でデータをバックアップできます。これらのコマンドは、両方とも directoryserver コマンドのサブコマンドです。詳細は、『Directory Server Man Page Reference』を参照してください。どちらの方法にも利点と制限があるため、効果的なバックアップ戦略を計画するには、それぞれの方法について理解することが役立ちます。

バイナリバックアップ (db2bak)

バイナリバックアップは、ファイルシステムレベルで行われます。バイナリバックアップの出力は、すべてのエントリ、インデックス、更新履歴ログ、トランザクションログを含むバイナリファイルのセットです。

注 バイナリバックアップでは、dse.ldif 設定ファイルはバックアップされません。失われた設定情報を復元するには、このファイルを手動でバックアップする必要があります。

バイナリバックアップには、次のような利点があります。

- すべてのサフィックスを一度にバックアップできる。
- LDIF へのバックアップと比較して、バイナリバックアップは格段に高速である。

バイナリバックアップには、次のような制約があります。

- バイナリバックアップからの復元は、同じ設定のサーバーだけで実行できる。つまり、次の制約が適用される。
  - 両方のマシンが同じハードウェア、同じオペレーティングシステム (サービスパックやパッチも含まれる) を使用している必要がある。
  - 両方のマシンに同じバージョンの Directory Server (バイナリ形式 (32 ビットまたは 64 ビット)、サービスパック、パッチも含まれる) がインストールされている必要がある。
  - 両方のサーバーは、同じサフィックスに分岐する同じディレクトリツリーを持つ必要がある。すべてのサフィックスのデータベースファイルをまとめてコピーする必要があり、サフィックスを個別にコピーすることはできない。
  - 両方のサーバーの各サフィックスには同じインデックス (VLV (仮想リスト表示) インデックスも含まれる) が設定されている必要がある。サフィックスのデータベースの名前は同じである必要がある。
  - コピーする Directory Server には o=NetscapeRoot サフィックスを含めることはできない。つまり、Directory Server を Administration Server の設定ディレクトリにすることはできない。

- 各サーバーでは、同じサフィックスがレプリカとして設定されている必要があり、両方のサーバーでレプリカに同じ役割 (マスター、ハブ、コンシューマ) が設定されている必要がある。部分レプリケーションが設定されている場合は、すべてのマスターサーバーが同じように設定されている必要がある。
- どちらのサーバーでも、属性の暗号化を使用していてはならない。

バイナリ復元機能によるデータの復元については、『Directory Server 管理ガイド』の「バイナリコピーによるレプリカの初期化」を参照してください。

互いに連携するマシンの各セット (上で定義した同じ設定を持つマシン) で、少なくとも定期的にバイナリバックアップを行う必要があります。

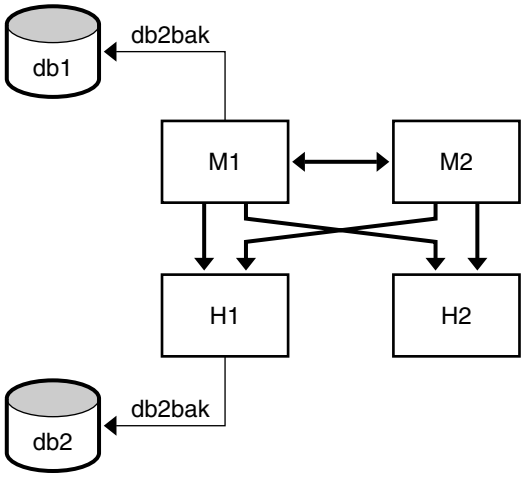
注 ローカルバックアップからの復元のほうが簡単なので、各サーバーでバイナリバックアップを行なってください。

この章に示される図で使用される略語の意味は次のとおりです。

- M = マスター
- H = ハブ
- C = コンシューマ
- RA = レプリケーションアグリーメント

図 9-1 は、M1 と M2 が同じ設定を持ち、H1 と H2 が同じ設定を持つことを前提としています。この例では、M1 と H1 でバイナリバックアップが行われます。障害が発生した場合、いずれかのマスターを M1 (db1) のバイナリバックアップから復元し、いずれかのハブを H1 (db2) のバイナリバックアップから復元することができます。H1 のバイナリバックアップからマスターを復元したり、M1 のバイナリバックアップからハブを復元することはできません。

図 9-1　バイナリバックアップ



LDIF (db2ldif) へのバックアップ

LDIF へのバックアップはサフィックスレベルで行われます。`db2ldif` の出力は、フォーマットされた LDIF ファイルです。このため、このプロセスはバイナリバックアップと比較して時間がかかります。

注　`db2ldif` の実行時に `-r` オプションを指定しない限り、レプリケーション情報はバックアップされません。

LDIF へのバックアップでは、`dse.ldif` 設定ファイルはバックアップされません。失われた設定情報を復元するには、このファイルを手動でバックアップする必要があります。

LDIF へのバックアップには、次のような利点があります。

- LDIF へのバックアップは、設定に関係なくどのサーバーからも実行できる。
- LDIF バックアップからの復元は、設定に関係なくどのサーバーからも実行できる (`-r` オプションを指定してレプリケーション情報がエクスポートされている場合)。

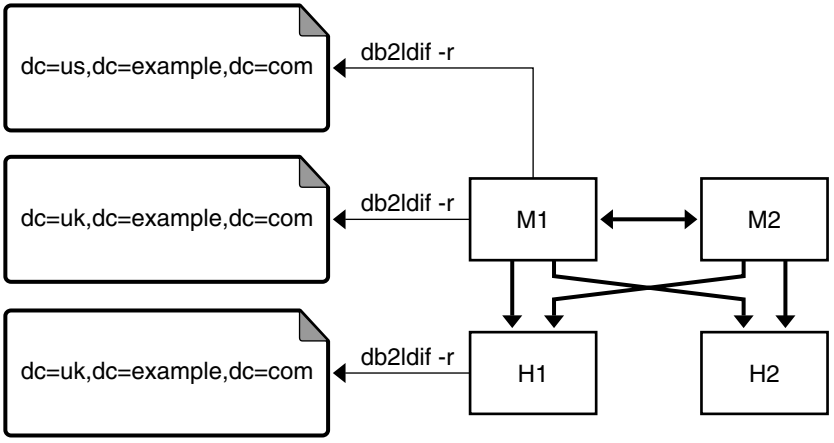
LDIF へのバックアップには、次のような制約があります。

- 迅速なバックアップと復元が必要な状況では、LDIF へのバックアップでは時間がかかり過ぎる可能性がある。

トポロジの単一マスターで、レプリケートされた各サフィックスを LDIF に定期的にバックアップしてください。

次の図では、M1 だけ、または M1 と H1 のレプリケートされた各サフィックスに対して db2ldif -r を実行しています。

図 9-2　db2ldif -r によるバックアップ



警告　パージ遅延より頻繁にバックアップを行うことが重要です。nsDS5ReplicaPurgeDelay 属性によって指定されるパージ遅延は、更新履歴ログに対して内部パージ操作を開始するまでの期間 (秒単位) です。デフォルトのパージ遅延は 604800 秒 (1 週間) です。更新履歴ログは、レプリケートが完了している、またはレプリケートが完了していない更新の記録を保持しています。

更新の頻度がパージ遅延より低い場合、バックアップを行う前に更新履歴ログの内容がクリアされてしまう可能性があります。この場合、バックアップからデータを復元しようとしても、変更は失われています。

復元方法の選択

Directory Server では、バイナリ復元 (bak2db) と、LDIF ファイルからの復元 (ldif2db) という 2 つの方法でデータを復元できます。すでに説明したバックアップ方法と同様に、どちらの方法にも利点と制限があります。

バイナリ復元 (bak2db)

バイナリ復元では、データベースレベルでデータがコピーされます。このため、バイナリ復元によるデータの復元には、次のような利点があります。

- すべてのサフィックスを一度に復元できる。

- LDIF ファイルからの復元と比較して、バイナリ復元は格段に高速である。

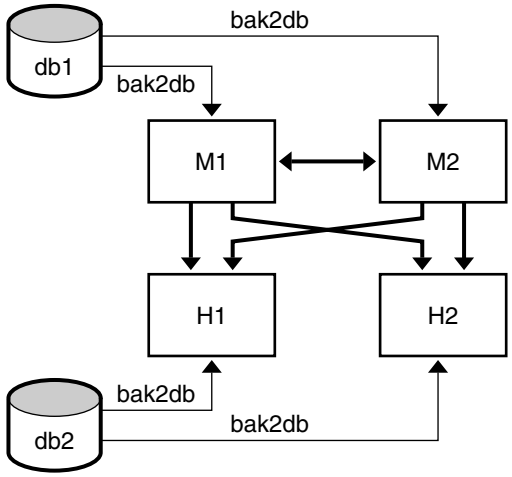
バイナリ復元によるデータの復元には、次のような制約があります。

- 復元は、同じ設定を持つサーバー (215 ページの「バイナリバックアップ (db2bak)」を参照) だけで実行できる。バイナリ復元機能によるデータの復元については、『Directory Server 管理ガイド』の「バイナリコピーによるレプリカの初期化」を参照してください。
- バイナリバックアップではデータベースの同一コピーが作成されるため、データベースが破損していることに気付かずにバイナリバックアップを行った場合、破損したデータベースが復元される危険性がある。

マシンの設定が同一であり、実行時間に特に考慮が必要な場合、推奨される復元方法はバイナリ復元になります。

図 9-3 は、M1 と M2 が同じ設定を持ち、H1 と H2 が同じ設定を持つことを前提としています。この例では、いずれかのマスターを M1 (db1) のバイナリバックアップから復元し、いずれかのハブを H1 (db2) のバイナリバックアップから復元することができます。

図 9-3 バイナリ復元



LDIF からの復元 (ldif2db)

LDIF ファイルからの復元は、サフィックスレベルで行われます。このため、このプロセスはバイナリ復元と比較して時間がかかります。LDIF ファイルからの復元には、次のような利点があります。

- LDIF からの復元は、設定に関係なくどのサーバーからも実行できる。

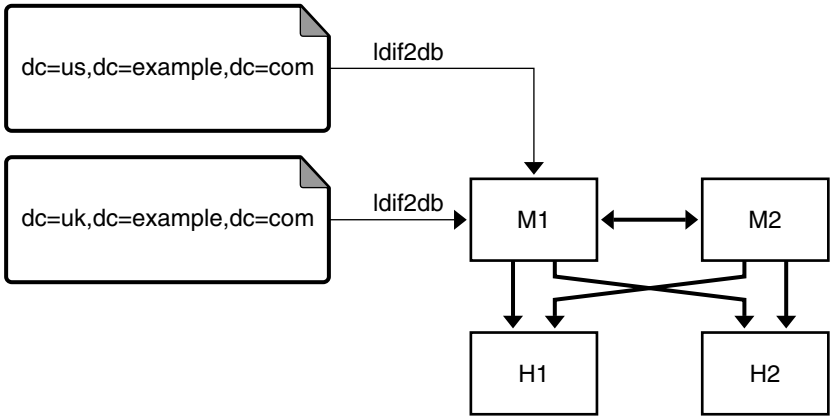
- レプリケーショントポロジに関係なく、ディレクトリサービス全体の配備に単一の LDIF ファイルを使用できる。予定されているビジネスニーズに合わせてディレクトリサービスをダイナミックに拡張または縮小する場合に、これは特に便利である。

LDIF ファイルからの復元には、次のような制限があります。

- 迅速な復元が必要な状況では、`ldif2db` の実行は時間がかかり過ぎる場合がある。

次の図では、M1 だけ、または M1 と H1 のレプリケートされた各サフィックスに対して `ldif2db` を実行しています。

図 9-4 ldif2db による復元



レプリケーショントポロジの例

レプリケーショントポロジは、企業の規模と、データセンターの物理的な場所によって決定されます。このため、ここで紹介するレプリケーショントポロジの例は、企業がディレクトリを維持するデータセンター (サイト) の数によって分けられています。

ディレクトリを最初に配備するときは、エントリの現在の数と、ディレクトリへの読み取りおよび書き込み操作の現在のボリュームに基づいて配備を行います。エントリ数が増えると、読み取りパフォーマンス向上のためにディレクトリのスケーリングが必要になる場合があります。それぞれの企業に合わせたスケーラビリティが提案されます。

これらのトポロジは、1 つのコンポーネントで障害が発生した場合にも、迅速な人的対応なしでサービスを提供し続けることを目的としています。1 つまたは 2 つのデータセンターで、読み取りと書き込みのフェイルオーバーもローカルに行われます。

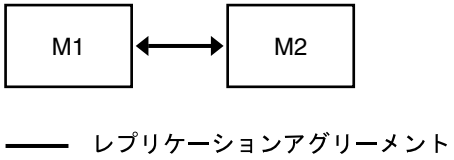
1 つのデータセンター

1 つのデータセンターのトポロジには、ディレクトリの想定パフォーマンス要件が大きく影響します。提案される基本的なトポロジでは、読み取りと書き込みの操作を処理するために、少なくとも 2 つのサーバーの動作が保証される配備が想定されています。2 つのマスターによって、高可用性ソリューションも提供されます。

1 つのデータセンターの基本トポロジ

図 9-5 に示されるトポロジは、読み取りと書き込みのパフォーマンスのために 2 つのマスターを持つ 1 つのデータセンターを示しています。この基本例では、クライアントはいずれかのマスターに書き込みを行い、いずれかのマスターから読み取りを行います。

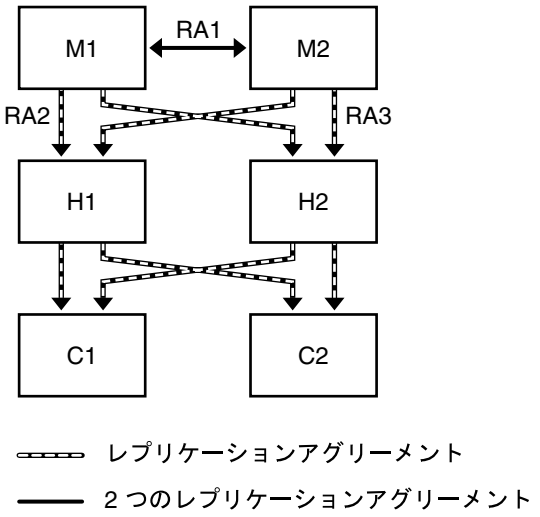
図 9-5　1 つのデータセンター : 基本トポロジ



読み取りパフォーマンスのための 1 つのデータセンターのスケーリング

図 9-6 に示されるように、ハブとコンシューマを追加することで、読み取りパフォーマンスが向上します。第 3 レベルのコンシューマを簡単に追加できるように、まず、マスターの下にハブを追加します。第 2 レベルのサーバーをハブとして設定することで、マシンを再設定する必要なく、その下にコンシューマを追加できます。

図 9-6　読み取りパフォーマンスのための 1 つのデータセンターのスケーリング



1 つのデータセンターの障害マトリックス

図 9-6 の例では、213 ページの「障害と復元について」に示されるいずれかの原因により、さまざまなコンポーネントが使用不可能になる可能性があります。表 9-1 は、これらの障害と、それに対応する復元処理を示しています。

表 9-1　1 つのデータセンター : 障害マトリックス

障害が発生したコンポーネント	対策
M1	Directory Proxy Server、クライアントサーバーリスト、ハードウェアまたはソフトウェアのロードバランサにより、ローカル書き込みは M2 にルーティングされます。M2 は、H1、H2 へのレプリケーションを継続します。
M2	Directory Proxy Server、クライアントサーバーリスト、ハードウェアまたはソフトウェアのロードバランサにより、ローカル書き込みは M1 にルーティングされます。M1 は、H1、H2 へのレプリケーションを継続します。
RA1 をサポートする LAN リンク	どちらのマスターのローカル書き込みも受け付けます。コンシューマに同じデータが含まれるように、競合の解決はハブレベルで行われます。

表 9-1　1 つのデータセンター : 障害マトリックス (続き)

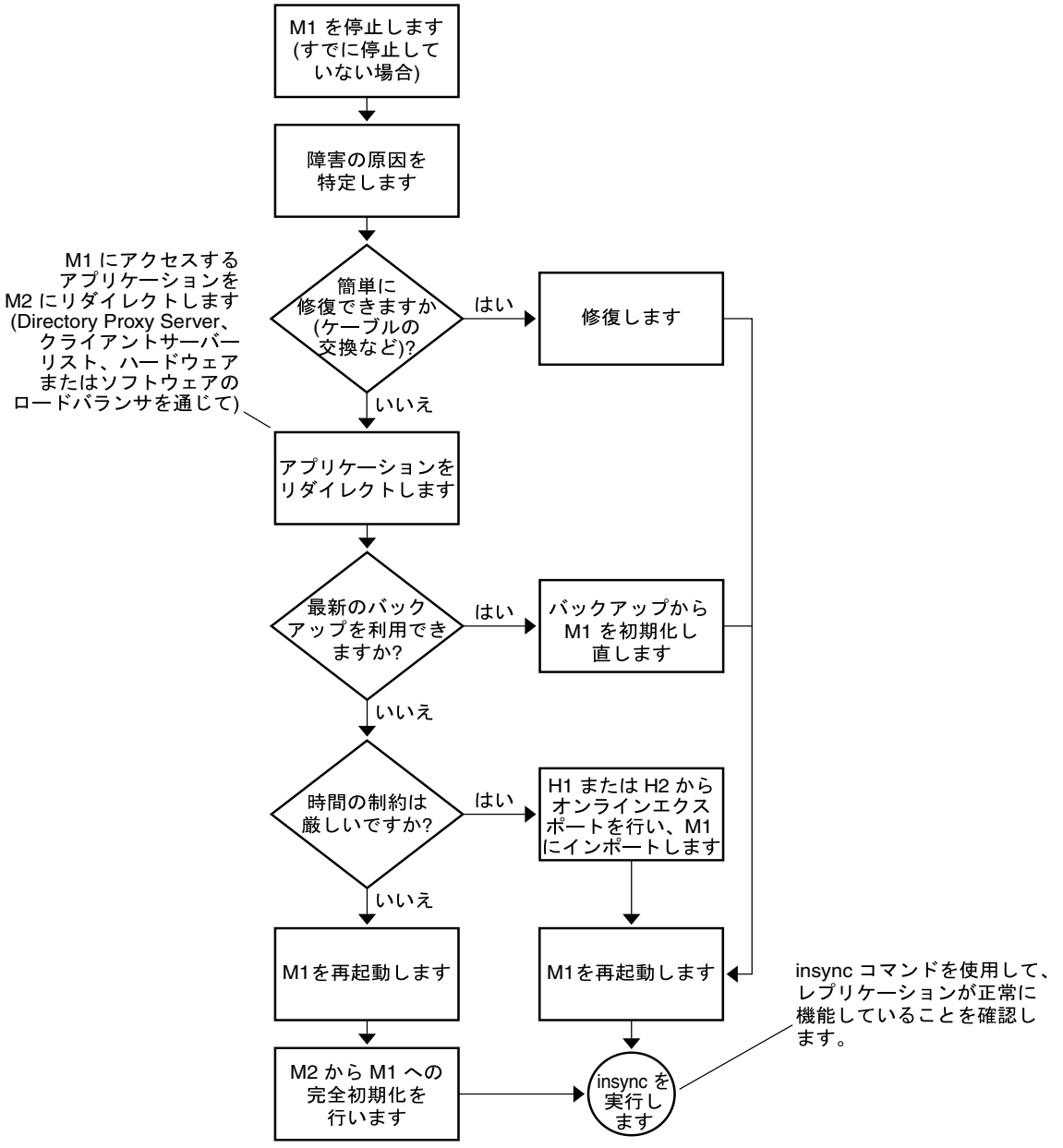
障害が発生したコンポーネント	対策
H1 または H2	どちらのマスターのローカル書き込みも受け付けます。コンシューマに同じデータが含まれるように、競合はマスターレベルで解決され、正常に稼動するほうのハブを通じてすべてのコンシューマにレプリケートされます。
RA2 をサポートする LAN リンク	どちらのマスターのローカル書き込みも受け付けます。H1 へのレプリケーションは M2 から行われ、ハブからコンシューマへのレプリケーショントラフィックは正常に機能し続けます。

1 つのデータセンターの復元手順 (1 つのコンポーネント)

2 つのマスターを持つ 1 つのデータセンターでは、1 つのマスターで障害が発生しても読み取りと書き込みの機能は維持されます。ここでは、障害が発生したコンポーネントの復元に適用できる復元戦略の例について説明します。

図 9-7 のフローチャートとその後の手順は、1 つのマスター (M1) で障害が発生したことを前提としています。

図 9-7　1 つのデータセンターの復元手順を示すフローチャート (1 つのコンポーネント)



1. M1 を停止します (すでに停止していない場合)。
2. 障害の原因を特定します。たとえばネットワークケーブルの交換などによって簡単に修復できる場合は、修復します。
3. 問題がより重大で修復に時間がかかる場合は、M1 にアクセスするアプリケーションが、Sun Java System Directory Proxy Server、クライアントサービスリスト、ハードウェアまたはソフトウェアのロードバランサを通じて M2 にリダイレクトされることを確認します。
4. 最新のバックアップがあれば、バックアップから M1 を初期化し直します。
5. 最新のバックアップを利用できない場合は、M1 を再起動し、M2 から M1 への完全初期化を行います。詳細な手順については、『Directory Server 管理ガイド』の「オンラインでのレプリカ初期化の実行」を参照してください。
6. 最新のバックアップを利用できず、完全初期化を行う時間もない場合は、H1 または H2 からオンラインエクスポートを行い、M1 に `ldif2db` でインポートします。
7. M1 を起動します (すでに起動していない場合)。
8. M1 のモードが読み取り専用モードであれば、読み書き有効モードに設定します。
9. `insync` コマンドを使用して、レプリケーションが正常に機能していることを確認します。詳細は、『Directory Server Man Page Reference』を参照してください。

注 オンラインエクスポートの実行はサーバーのパフォーマンスに影響します。このため、エクスポートにはマスター (M2) ではなく、その時点で書き込み操作を実行できる唯一のサーバーであるハブを使用してください。

1 つのデータセンターの復元手順 (2 つのコンポーネント)

この例で 2 つのマスターに障害が発生した場合、書き込み機能は失われます。障害が重大で修復に時間がかかる場合は、できるだけ早急に書き込み機能を提供するための戦略を実装する必要があります。

次の手順は、M1 と M2 の両方に障害が発生し、すぐには復元できない状況を前提としています。もっとも迅速で、できるだけ簡単な復元方法を考える必要があります。この手順では、もっとも簡単な方法の例として、サーバーの昇格を紹介します。

1. H1 を書き込み可能なマスターに昇格させます。具体的な手順については、『Directory Server 管理ガイド』の「レプリカの昇格と降格」を参照してください。
2. M1 または M2 にアクセスしていたアプリケーションが新しいマスターにリダイレクトされることを確認します。
3. 変更がコンシューマにレプリケートされ続けるように、新しいマスターと H2 の間に新しいレプリケーションアグリーメントを追加します。

2 つのデータセンター

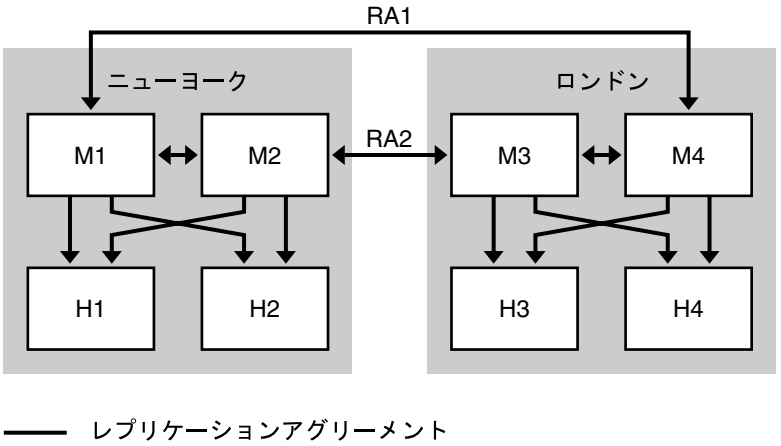
複数のサイトでデータが共有される場合は、パフォーマンスとフェイルオーバーの両方の面で効果的なレプリケーショントポロジが欠かせません。

2 つのデータセンターの基本トポロジ

図 9-8 に示されるトポロジは、最適な読み取り、書き込みパフォーマンスのために、各データセンターが 2 つのマスターと 2 つのハブを持つことを前提としています。第 2 レベルのサーバーをハブとして設定することで、マシンを再設定する必要なく、その下にコンシューマを追加できます。

この例では、レプリケーションアグリーメント RA1 と RA2 を異なるネットワーク経由で設定しています。この設定であれば、いずれかのネットワークリンクが使用不可能になったり、信頼できなくなった場合でも、データセンター間のレプリケーションを行えます。

図 9-8 2 つのデータセンターの基本トポロジ

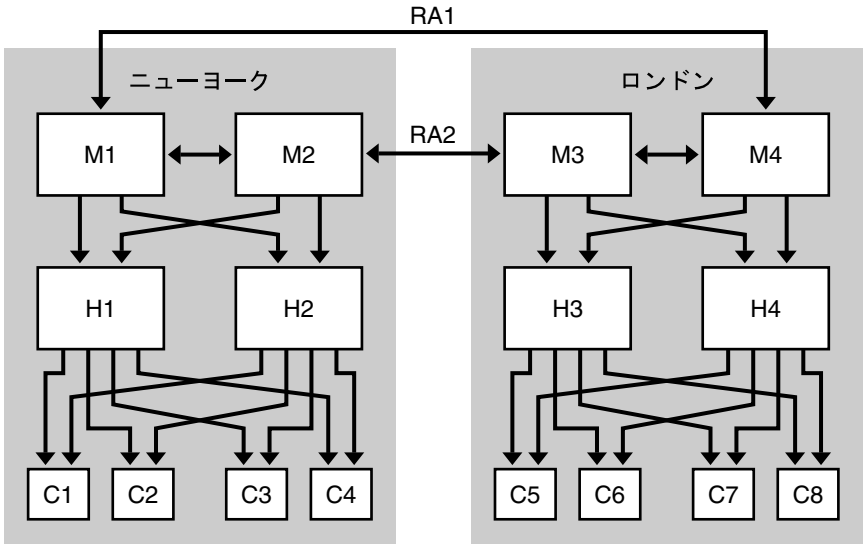


読み取りパフォーマンスのための 2 つのデータセンターのスケーリング

図 9-6 に示される 1 つのデータセンターの例のように、ハブとコンシューマを追加することで読み取りパフォーマンスを向上させることができます。

この例では、レプリケーションアグリーメント RA1 と RA2 を異なるネットワーク経由で設定しています。この設定であれば、いずれかのネットワークリンクが使用不可能になったり、信頼できなくなった場合でも、データセンター間のレプリケーションを行えます。

図 9-9　読み取りパフォーマンスのための 2 つのデータセンターのスケーリング



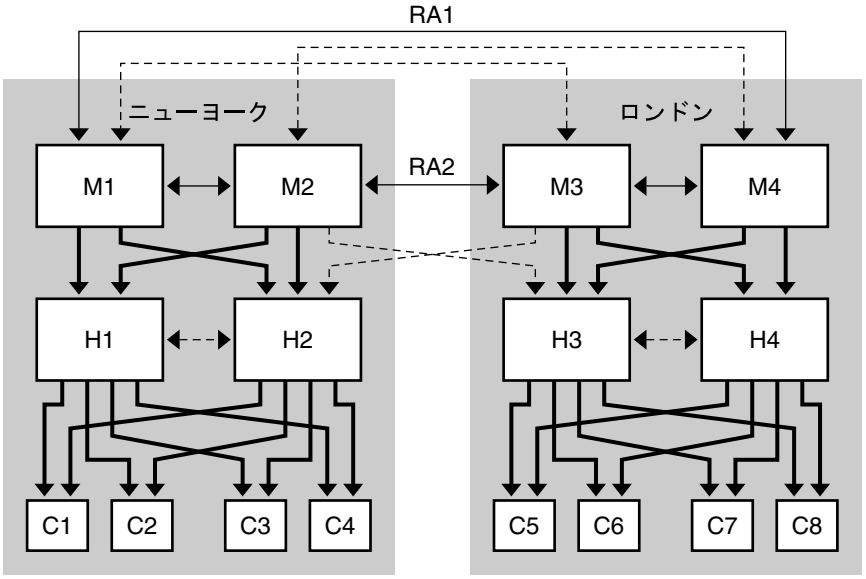
2 つのデータセンターの復元例

図 9-9 に示される配備では、1 つのマスターで障害が発生した場合に、1 つのデータセンターに適用した復元戦略をそのまま適用できます。いずれかのデータセンターの 1 つのデータセンターが使用不可能な状態になった場合でも、M1 と M4 の間、および M2 と M3 の間のレプリケーションアグリーメントにより、どちらのデータセンターもレプリケートされた更新を受け取り続けることができます。

しかし、複数のマスターで障害が発生した場合は、高度な復元戦略が必要となります。これには、復元アプリケーションアグリーメントの作成が関連します。このアグリーメントは、デフォルトでは無効化されていますが、障害が発生した場合には迅速に有効化できます。

図 9-10 は、この復元戦略を示しています。

図 9-10 2 つのデータセンターの復元レプリケーションアグリーメント



デフォルトレプリケーションアグリーメント

復元レプリケーションアグリーメント

適用される復元戦略は、どのような組み合わせでコンポーネントに障害が発生するかによって異なります。しかし、複数の障害に対する基本的な戦略を準備しておけば、その他のコンポーネントに障害が発生した場合にもそれを適用できます。

図 9-10 のトポロジ例は、ニューヨークデータセンターの両方のマスターに障害が発生したことを前提としています。この場合の復元戦略は、次のようになります。

1. M3 と H2 の間の復元レプリケーションアグリーメントを有効化します。

   これにより、ロンドンサイトへのリモート書き込みが、引き続きニューヨークサイトにレプリケートされます。

2. H2 を書き込み可能なマスターに昇格させます。具体的な手順については、『Directory Server 管理ガイド』の「レプリカの昇格と降格」を参照してください。

   これにより、ニューヨークサイトでの書き込み機能が維持されます。

3. 新たに昇格したマスター (それまでの H2) と M3 の間にレプリケーションアグリーメントを作成します。

これにより、ニューヨークサイトへのリモート書き込みが、引き続きロンドンサイトにレプリケートされます。

4. H2 と H1 の間の復元レプリケーションアグリーメントを有効化します (1 方向のみ)。

   これにより、H1 は、引き続きレプリケーショントポロジ全体からの更新を受け取ることができます。

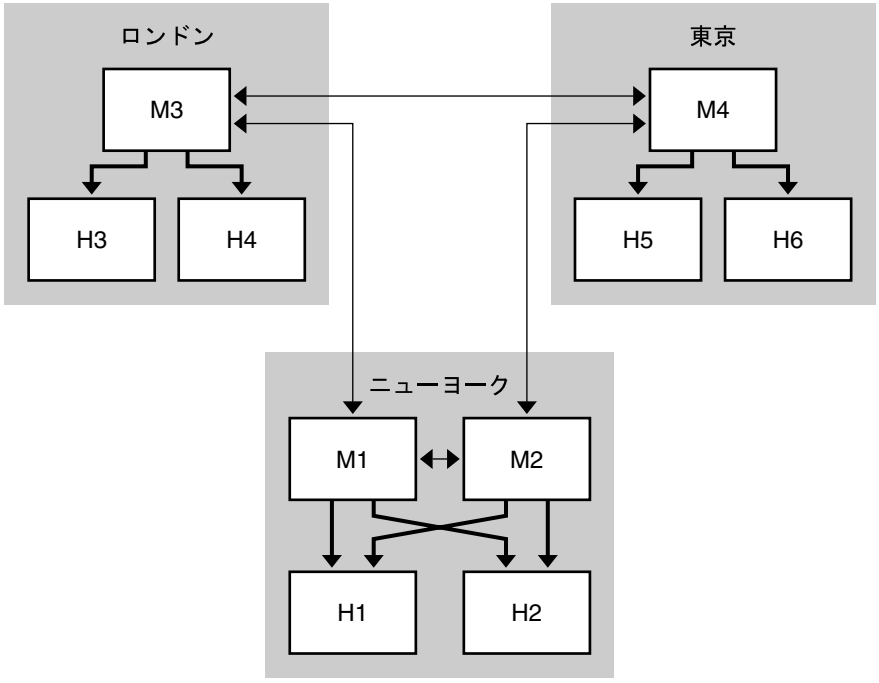
3 つのデータセンター

Directory Server 5.2 は、4 方向のマルチマスターレプリケーションをサポートしています。地理的に離れた 3 つの拠点にまたがる企業では、それぞれの地域で 2 つのマスターを持つ 1 つのデータセンターが必要になる可能性があります。このディレクトリの機能をどのように分割するかは、各データセンターで行われる読み取りおよび書き込み操作の相対的なトラフィックボリューム (など) によって異なります。

3 つのデータセンターの基本トポロジ

図 9-11 に示されるトポロジは、ニューヨークのデータセンターに対する読み取り、書き込み要求の数が最大で、3 つのデータセンターのそれぞれでローカルな読み取り、書き込み要求が可能であることを前提としています。

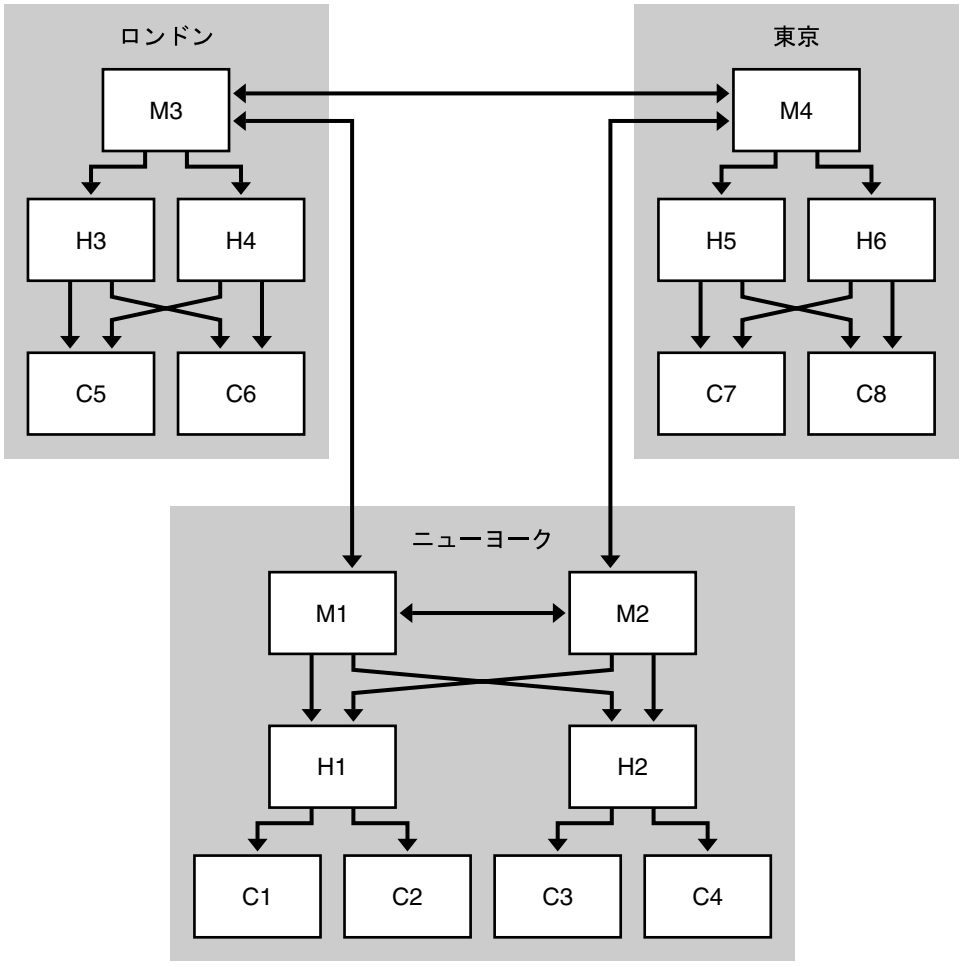
図 9-11　3 つのデータセンターの基本トポロジ



読み取りパフォーマンスのための 3 つのデータセンターのスケーリング

これまでの例のように、ハブとコンシューマを追加することで読み取りパフォーマンスを向上させることができます。ただし、それぞれのデータセンターでの想定パフォーマンス要件を考慮する必要があります。図 9-12 は、このトポロジを示しています。

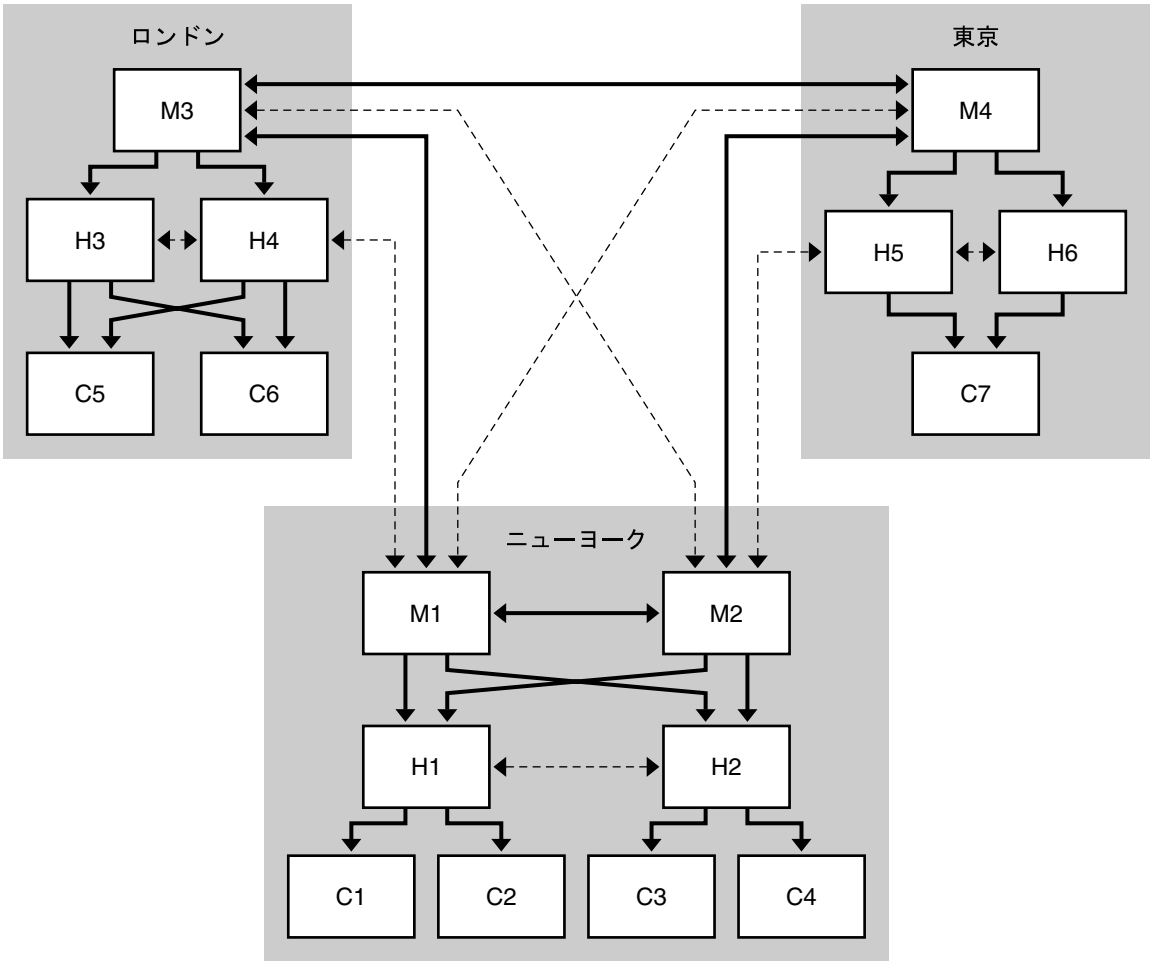
図 9-12　読み取りパフォーマンスのための 3 つのデータセンターのスケーリング



3 つのデータセンターの復元例

2 つのデータセンターの場合と同様に、複数のマスターに障害が発生した場合の復元戦略には、復元レプリケーションアグリーメントの作成が必要となります。図 9-13 に示されるように、これらのアグリーメントは、デフォルトでは無効化されていますが、障害が発生した場合には迅速に有効化できます。

図 9-13　3 つのデータセンターの復元レプリケーションアグリーメント



3 つのデータセンターの復元手順 (1 つのコンポーネント)

図 9-13 の例で、ロンドンまたは東京のマスターに障害が発生し、ローカル書き込み機能が失われたと仮定します。次の手順は、M3 (ロンドン) で障害が発生したことを前提としています。

1. H4 を書き込み可能なマスターに昇格させます。具体的な手順については、『Directory Server 管理ガイド』の「レプリカの昇格と降格」を参照してください。

2. H4 から H3 への復元レプリケーションアグリーメントを有効化し、ローカル書き込みがすべてのコンシューマにレプリケートされるようにします。
3. M1 と H4 の間の復元レプリケーションアグリーメントを有効化し、ローカル書き込みがリモートデータセンターにレプリケートされ、リモート書き込みがローカルコンシューマにレプリケートされるようにします。
4. M3 にアクセスしていたアプリケーションが新しいマスターにリダイレクトされることを確認します。

注 この手順は、M3 の修復中に、直接的なローカルの読み取りおよび書き込み機能を提供するための中間ソリューションです。

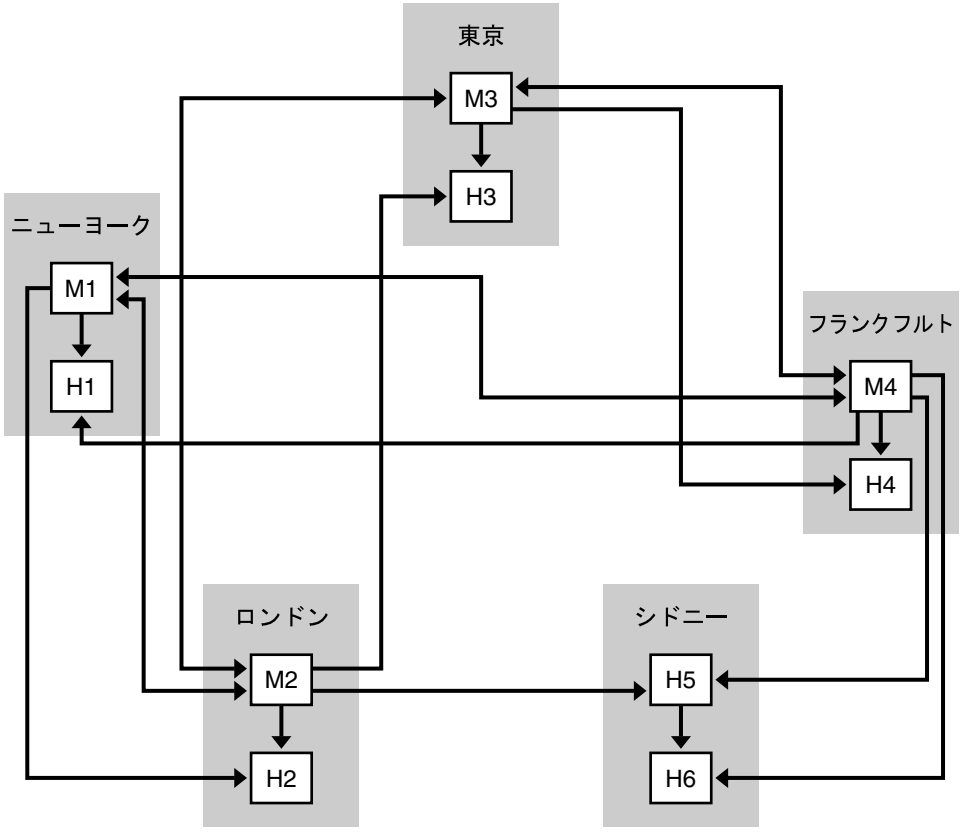
5 つのデータセンター

Directory Server 5.2 は、4 方向のマルチマスターレプリケーションをサポートしています。地理的に離れた 5 つの拠点にまたがる企業では、どの地域のローカル更新パフォーマンス要件が最小であるかを考慮する必要があります。この地域にはマスターサーバーを置かず、書き込みをいずれかの地域のマスターにリダイレクトします。

5 つのデータセンターの基本トポロジ

図 9-14 に示されるトポロジは、シドニーデータセンターがもっとも少ない書き込み要求を受け取ることを前提としています。ローカル読み取り要求は、5 つのそれぞれのデータセンターで可能です。

図 9-14　5 つのデータセンターの基本トポロジ



読み取りパフォーマンスのための 5 つのデータセンターのスケーリング

これまでの例のように、ハブとコンシューマを追加することで読み取りパフォーマンスを向上させることができます。ただし、それぞれのデータセンターでの想定パフォーマンス要件を考慮する必要があります。

5 つのデータセンターの復元例

2 つのデータセンターの場合と同様に、複数のマスターに障害が発生した場合の復元戦略には、復元レプリケーションアグリーメントの作成が必要となります。236 ページの図 9-15 に示されるように、これらのアグリーメントは、デフォルトでは無効化されていますが、障害が発生した場合には迅速に有効化できます。

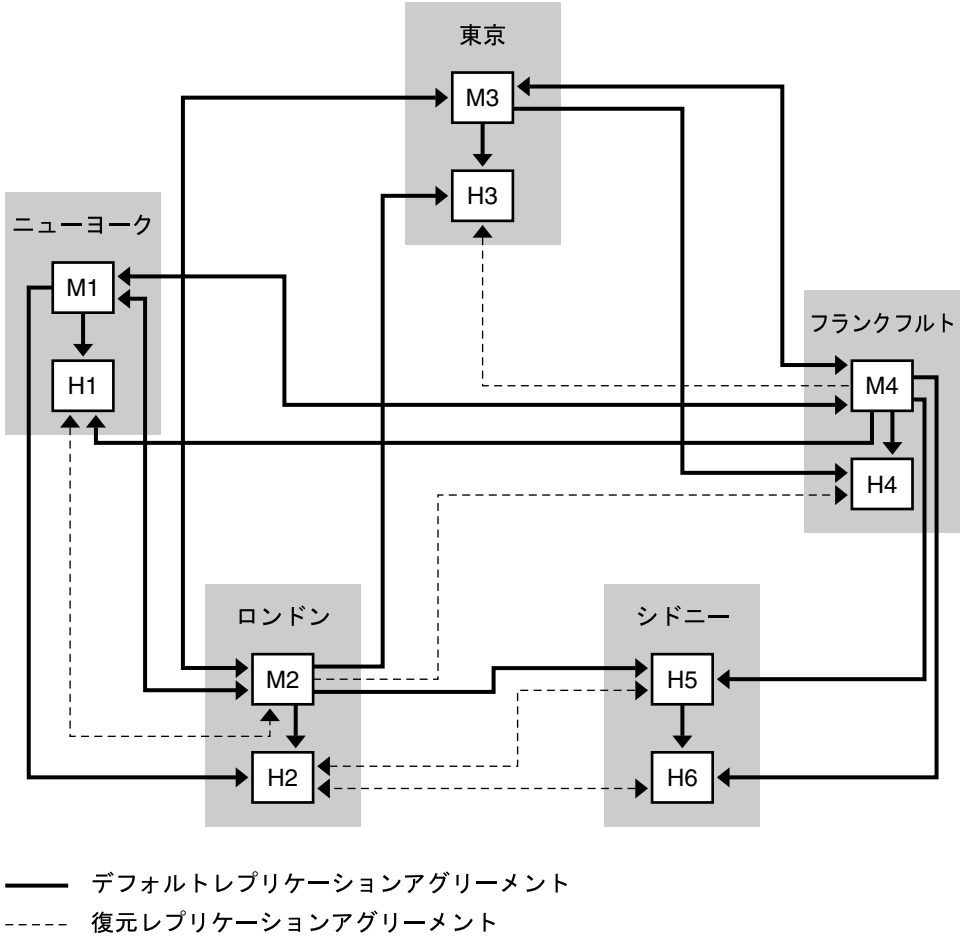
5 つのデータセンターの復元手順 (1 つのコンポーネント)

図 9-15 の例で、いずれかのマスターに障害が発生し、ローカル書き込み機能が失われたと仮定します。次の手順は、M1 (ニューヨーク) で障害が発生したことを前提としています。

1. H1 を書き込み可能なマスターに昇格させます。具体的な手順については、『Directory Server 管理ガイド』の「レプリカの昇格と降格」を参照してください。
2. M2 から H1 への復元レプリケーションアグリーメントを有効化し、ローカル書き込みがリモートデータセンターにレプリケートされ、リモート書き込みがローカルコンシューマにレプリケートされるようにします。
3. M1 にアクセスしていたアプリケーションが新しいマスターにリダイレクトされることを確認します。

注 この手順は、M1 の修復中に、直接的なローカルの読み取りおよび書き込み機能を提供するための中間ソリューションです。

図 9-15　5 つのデータセンターの復元レプリケーションアグリーメント



旧バージョン対応更新履歴ログプラグインを使用する 1 つのデータセンター

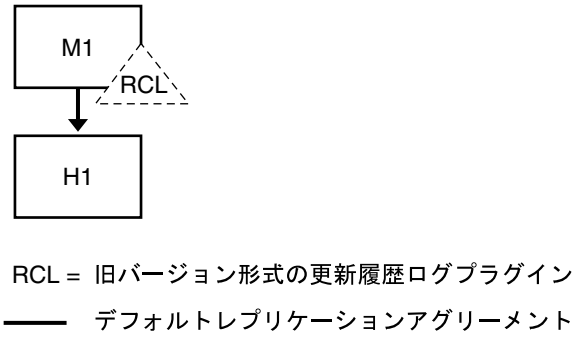
すでに説明した 1 つのデータセンターのトポロジは、旧バージョン形式の更新履歴プラグインの使用を考慮していませんでした。旧バージョン形式の更新履歴に依存するほとんどのアプリケーションは、更新が順序よくリストされていることを前提としており、マルチマスターレプリケーションモデルの旧バージョン形式の更新履歴で一貫性が失われることによって失敗してしまいます。一般に、配備するアプリケーションが旧バージョン形式の更新履歴を必要とする場合は、その配備ではマルチマスターレ

プリケーショントポロジを採用するべきではありません。詳細は、150 ページの「レプリケーションと旧バージョン対応更新履歴ログプラグイン」および『Directory Server 管理ガイド』の「旧バージョン形式の変更履歴ログプラグインの使用」を参照してください。

旧バージョン対応更新履歴ログプラグインの基本トポロジ

マルチマスターレプリケーションを配備できない場合は、図 9-16 に示される基本トポロジが推奨されます。

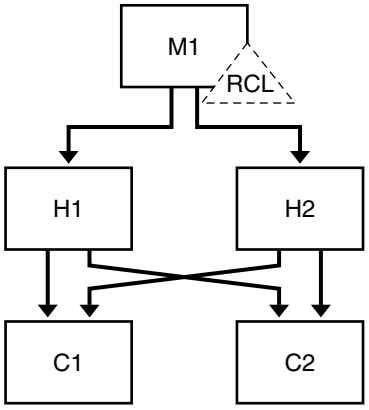
図 9-16　旧バージョン対応更新履歴ログプラグインを使用する 1 つのデータセンター



読み取りパフォーマンスのための旧バージョン対応更新履歴ログプラグインのスケーリング

1 つのデータセンターのトポロジと同様に、図 9-17 に示されるように、ハブとコンシューマを追加することで読み取りパフォーマンスを向上させることができます。

図 9-17　旧バージョン対応更新履歴ログプラグインを使用する 1 つのデータセンター (スケーリング)



RCL = 旧バージョン形式の更新履歴ログプラグイン

—— デフォルトレプリケーションアグリーメント

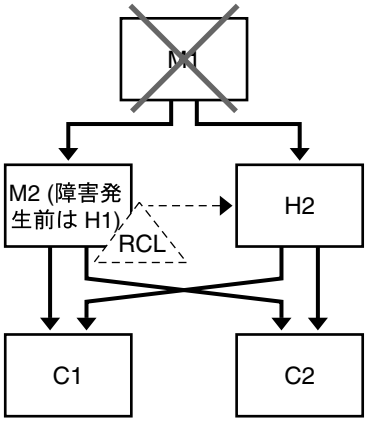
旧バージョン対応更新履歴ログプラグインの復元手順

図 9-17 に示される配備では、マスターサーバーに障害が発生した場合に次の戦略を適用できます。

1. M1 を停止します (すでに停止していない場合)。
2. H1 または H2 をマスターサーバーに昇格させます。具体的な手順については、『Directory Server 管理ガイド』の「レプリカの昇格と降格」を参照してください。
3. 新しいマスター (M2) で旧バージョン対応更新履歴ログプラグインを有効化します。
4. 旧バージョン対応更新履歴ログのバックアップを M2 に復元します。
5. サーバーを再起動します。
6. 変更がハブにレプリケートされ続けるように、M2 と残りのハブの間に新しいレプリケーションアグリーメントを追加します。

   図 9-18 は、この復元戦略を示しています。

図 9-18　旧バージョン対応更新履歴ログプラグインを使用する 1 つのデータセンター (復元)



RCL = 旧バージョン形式の更新履歴ログプラグイン
——— デフォルトレプリケーションアグリーメント
- - - - - 復元レプリケーションアグリーメント

第 10 章

システムのサイジング

適切にハードウェアをサイジングすることは、ディレクトリサービスの計画と配備において必須事項です。ハードウェアのサイジングには、利用可能なメモリ容量と利用可能なローカルディスクの空き容量がきわめて重要になります。

注 最良の結果を得るために、実際の運用で使用されるエントリに相当するエントリのサブセットを用いて、テストシステムをインストールし、設定します。次に、テストシステムを使用して、実際の運用サーバーに近い動作を行います。

特定のシステムに対して最適化する場合は、Directory Server をサポートするためのチューニングの際に I/O サブシステムの機能を利用できるように、システムバス、周辺機器バス、I/O デバイス、およびサポートされるファイルシステムがどのように動作するかを確認します。

この章では、Directory Server インスタンスのディスクとメモリの要件を見積もる方法を紹介します。また、ネットワーク要件や SSL アクセラレータハードウェアの要件についても説明します。

推奨される最小要件

表 10-1 に、実際の運用環境でソフトウェアをインストールして使用するためのメモリとディスクの最小要件を示します。

指定したエントリ数における最小要件は、実際には表 10-1 で示す要件と異なることがあります。ここでのサイズは、比較的小さなエントリで、デフォルトの設定によるインデックスと最小限チューニングされたキャッシュを用いる場合です。デジタル写真などの大きなバイナリ属性値がエントリに含まれる場合や、インデックスまたはキャッシュに異なる設定を用いている場合は、それに応じてディスク容量とメモリの最小見積もりをそれぞれ上方修正します。

表 10-1　ディスク容量とメモリの最小要件

ケース	ローカルディスクの空き容量	空き RAM
製品の解凍	最低 125M バイト	-
製品のインストール	最低 200M バイト	最低 256M バイト
10,000 ～ 250,000 エントリ	最低 3G バイト追加	最低 256M バイト追加
250,000 ～ 1,000,000 エントリ	最低 5G バイト追加	最低 512M バイト追加
1,000,000 エントリ以上	8G バイト以上追加	1G バイト以上追加

ディスク容量の最小要件には、アクセスログ専用の 1G バイトが含まれています。デフォルトでは、Directory Server は 10 個 (`cn=config` における `nsslapd-accesslog-maxlogsperdir`) のアクセスログファイルをローテーションするように設定されており、それぞれのログファイルは最大 100M バイト (`cn=config` における `nsslapd-accesslog-maxlogsize`) のメッセージを保持します。エラーログと監査ログの大きさは、Directory Server の設定状況によって変わります。ログ設定の詳細は、『Directory Server 管理ガイド』の「ログファイルを使用した Directory Server の監視」を参照してください。

最小の利用可能なメモリ

メモリの最小見積もりは、通常の配備における Directory Server インスタンスで使用されるメモリを反映しています。見積もりには、システムやほかのアプリケーションで使用されるメモリは含まれていません。より正確な見積もりを得るには、メモリの使用量を経験的に測定する必要があります。詳細は、244 ページの「物理メモリのサイジング」を参照してください。

原則として、利用可能なメモリ量が多ければ多いほど良い状況であるといえます。

最小のローカルディスクの空き容量

ローカルディスクの空き容量の最小見積もりは、典型的な配備における Directory Server インスタンスで必要となる空き容量を反映しています。経験上、ディレクトリエントリが大きい場合、必要な空き容量は、ディスク上にある同等の LDIF のサイズを少なくとも 4 倍したサイズになります。詳細は、247 ページの「ディスクサブシステムのサイジング」を参照してください。

ネットワークディスクにアクセスするサーバーやデータをインストールしないでください。Directory Server では、NFS、AFS、または SMB を介してネットワークに接続されたストレージの使用をサポートしていません。設定、ログ、データベース、およびインデックスファイルはすべて、インストールを終えてからも、常にローカルストレージ上に配置する必要があります。

最小プロセッサ能力

大ボリュームシステムでは、通常、複数の高速プロセッサを搭載することで、複数の同時検索、拡張インデックス、レプリケーション、およびその他の機能に適切な処理能力を実現しています。詳細は、257 ページの「マルチプロセッサシステムのサイジング」を参照してください。

最小ネットワークキャパシティ

テストによると、想定される最大スループットによっては、サービスプロバイダレベルのパフォーマンスでも 100M ビット Ethernet で十分であることが示されています。論理的な最大スループットは、次のようにして見積もることができます。

最大スループット = 返される最大エントリ数 / 秒×平均エントリサイズ

たとえば、Directory Server がピーク時には 1 秒あたり 5000 個の検索に対応しなければならず、平均サイズが 2000 バイトのエントリを返すとします。このとき論理的な最大スループットは 10M バイト、つまり 80M ビットとなります。しかし 80M ビットの方が、100M ビット Ethernet アダプタ 1 つで実現するスループットよりも大きくなる傾向があります。実際に測定されるパフォーマンスは異なることがあります。

WAN (広域ネットワーク) でマルチマスターレプリケーションを実行したい場合は、その接続で、待ち時間が最小でパケット損失がほとんどない十分なスループットが得られることを確認します。

詳細は、257 ページの「ネットワークキャパシティのサイジング」を参照してください。

物理メモリのサイジング

Directory Server では、データベーステクノロジを使用して情報を格納します。データベーステクノロジに依存するあらゆるアプリケーションでは、高速なメモリが十分あることが Directory Server のパフォーマンスを最適化する上で重要です。原則として、利用可能なメモリが多ければ多いほど、すばやくアクセスするためにキャッシュされるディレクトリ情報も多くなります。ディレクトリ内容全体を常にキャッシュするのに十分なメモリが各サーバーにあるのが理想的です。Directory Server5.2 では 64 ビットメモリアドレッシングをサポートしているため、現在は 64 ビットのプロセッサが対応できるだけの合計キャッシュサイズを処理できます。

注 Directory Server を実際の運用環境で配備するときは、キャッシュサイズを論理的な処理上限よりも小さく設定し、通常のシステム操作で使用できる適切なリソースを確保します。

Directory Server を実行するのに必要なメモリサイズを見積もるには、特定の Directory Server 設定と、Directory Server が実行されるシステムの両方とで必要なメモリを見積もります。

Directory Server 用メモリのサイジング

特定の配備に対して見積もった設定値が決まると、Directory Server インスタンスに必要な物理メモリを見積もることができます。表 10-2 に、この節で説明する計算で使用した値をまとめて表示します。

表 10-2 Directory Server 用メモリのサイジングの値

値	内容[1]
`nsslapd-cachememsize`	サフィックスのエントリキャッシュサイズ エントリキャッシュには形式化されたエントリが含まれており、クライアント要求に応答して送信されることになります。1 インスタンスで複数のエントリキャッシュを処理できます。
`nsslapd-dbcachesize`	データベースキャッシュサイズ データベースキャッシュは、サーバーで使用されるデータベースとインデックスからの要素を保持します。

表 10-2　Directory Server 用メモリのサイジングの値 (続き)

値	内容[1]
nsslapd-import-cachesize	一括インポート用のデータベースキャッシュサイズ インポートキャッシュは、エントリのインポート時にのみ使用される。インポートキャッシュ用に余分なメモリを確保する必要はないが、オフラインインポートだけを実行する場合は、エントリキャッシュやデータベースキャッシュ用に確保したメモリを再利用します。
nsslapd-maxconnections	管理される接続の最大数。
nsslapd-threadnumber	サーバーの起動時に作成される操作スレッド数

1. 完全な説明は、『Directory Server Administration Reference』を参照してください。

おおよそのメモリサイズを見積もるには、次の手順を実行します。

1. サーバープロセスの基本サイズ slapdBase を見積もります。

```
slapdBase = 75MB +(nsslapd-threadnumber x 0.5MB) +(nsslapd-maxconnections x
0.5 KB)
```

2. エントリキャッシュサイズの合計 entryCacheSum を決定します。

```
entryCacheSum = Sum 全エントリキャッシュ (nsslapd-cachememsize)
```

エントリキャッシュには、割り当てのオーバーヘッドが含まれていることに注意してください。言い換えれば、キャッシュは、nsslapd-cachememsize パラメータに指定するメモリより多くのメモリを消費します。これはメモリリークと表示されることもありますが、そうではありません。メモリ割り当てライブラリが要求を処理する方法に応じて、使用される実際のメモリが指定されたメモリよりかなり多くなることがあります。詳細は、『Directory Server Performance Tuning Guide』の「Entry Cache」を参照してください。

3. 全キャッシュの合計サイズ cacheSum を決定します。

```
cacheSum = entryCacheSum + nsslapd-dbcachesize + nsslapd-import-cachesize
```

4. Directory Server プロセスの合計サイズ slapedSize を決定します。

```
slapdSize = slapdBase + cacheSum
```

Directory Server で使用される物理メモリを測定するのに、Solaris システムの pmap(1) などのユーティリティまたは Windows のタスクマネージャを使用することができます。

5. 着信クライアント要求を処理するのに必要なメモリ slapdGrowth を見積もります。

```
slapdGrowth   = 20% x slapdSize
```

最初の見積もりなので、クライアント要求を処理するためのオーバーヘッドとして 20% を想定します。実際の割合は、その配備の特徴によって異なる可能性があります。この割合は Directory Server を運用環境に置く前に、実際に検証します。

6. Directory Server の合計メモリサイズ `slapdTotal` を決定します。

```
slapdTotal    = slapdSize + slapdGrowth
```

32 ビットサーバーを含む大規模配備では、`slapdTotal` の値が実際の上限である約 3.4G バイト (Linux システムの場合は 2.5G バイト) を超えたり、論理的な処理上限である約 3.7G バイトを超えることもあります。この場合、キャッシュをチューニングしてシステムの制限内で動作させるか、あるいは製品の 64 ビットバージョンを使用することができます。詳細は、『Directory Server Performance Tuning Guide』の「Tuning Cache Sizes」を参照してください。

オペレーティングシステム用メモリのサイジング

オペレーティングシステムを実行するのに必要なメモリを見積もるには、経験的に行うことが必要となり、オペレーティングシステムのメモリ要件は、固有のシステム設定に基づいて大きく異なります。そのため、オペレーティングシステムで必要なメモリ量を見積もろうとする前に、配備に対して典型的なシステムでチューニングすることを検討します。詳細は、『Directory Server Performance Tuning Guide』の「Tuning the Operating System」を参照してください。システムをチューニングしたら、最初の見積もりの `systemBase` に到達するまで、メモリの使用状況を監視します。メモリの使用状況を監視するには、Solaris の `sar` (1M) などのユーティリティまたは Windows のタスクマネージャを使用することもできます。

注　最高のパフォーマンスを得るために、Directory Server を実行するシステムはこのサービス専用で使用します。

その他のアプリケーションやサービスを実行する必要がある場合は、必要な合計メモリをサイジングするときに、使用するメモリも監視します。

さらに、一般的なシステムオーバーヘッドと通常の管理目的とにメモリを割り当てます。`systemOverhead` の最初の見積もりは、大きさに関係なく、少なくとも数百 M バイト、または合計物理メモリの 10% のいずれかになります。システムが実際の運用環境でメモリのページインやページアウトを行わないよう、`systemOverhead` に十分な容量を割り当てることが目的です。

オペレーティングシステムで必要な合計メモリ `systemTotal` は、次のようにして見積もることができます。

```
systemTotal = systemBase + systemOverhead
```

合計メモリのサイジング

前述の節で見積もった slapdTotal および systemTotal を使用して、必要な合計メモリ totalRAM を見積もります。

```
totalRAM = slapdTotal + systemTotal
```

totalRAM は必要な合計メモリの見積もりであり、システムが Directory Server プロセス専用であることが前提です。また、この値には、システム上で実行する可能性のあるすべてのアプリケーションとサービスに対するメモリ使用量の見積もりが含まれています。

メモリ不足での処理

多くの場合、Directory Server で使用される全データをキャッシュするのに十分なメモリを用意するのは、費用効率は良くありません。

少なくとも Directory Server を実行するのに十分なメモリをサーバーに搭載すれば、継続的なページスワップは発生しません。継続的にページスワップすることは、パフォーマンスに強い悪影響があります。Directory Server を起動してエントリキャッシュをプライムする前後でメモリ統計を表示するには、Solaris やその他のシステムの vmstat (1M) などのユーティリティを使用することができます。Solaris システムの MemTool など、個別には利用可能だがサポートされていないユーティリティも、アプリケーションがテストシステムで実行されているときに、メモリの使用状況や割り当て状況を監視するのに役立ちます。

システムに追加のメモリを搭載できない場合は、これまでどおり継続的なページスワップを監視し続け、データベースキャッシュやエントリキャッシュのサイズを減らします。スワップスペースが不足すると、Directory Server がそれ自体を停止します。

全ディレクトリデータをキャッシュするのに十分な物理メモリを確保できないときの可能な代替策は、『Directory Server Performance Tuning Guide』の「Tuning Cache Sizes」を参照してください。

ディスクサブシステムのサイジング

ディスクの使用と I/O 能力は、パフォーマンスに強く影響します。特に大量の変更をサポートする配備では、ディスクサブシステムは I/O のボトルネックとなる可能性があります。この節では、Directory Server インスタンスで全体のディスク容量を見積もったり、ディスク I/O のボトルネックを軽減したりするための推奨について説明します。

ディスク I/O のボトルネックの軽減については、『Directory Server Performance Tuning Guide』の「Tuning Logging」を参照してください。

ディレクトリサフィックスのサイジング

サフィックスに対するディスク容量の要件は、ディレクトリ内のエントリのサイズや個数によって変化するだけでなく、ディレクトリ設定や、特にサフィックスのインデックス方法によっても変化します。大規模配備で必要なディスク容量を測定するには、次の手順を実行します。

1. 配備で想定される 3 つの代表的なエントリのセットである 10,000 エントリ、100,000 エントリ、1,000,000 エントリの各セットに対して LDIF を生成します。

   生成したエントリは想定されるエントリタイプ (ユーザー、グループ、ロール、拡張スキーマのエントリ) の組み合わせを反映するとともに、特に `userCertificate` や `jpegPhoto` のように単一の大きな属性値が想定される場合の個々の属性値の平均サイズも反映する必要があります。

2. Directory Server のインスタンスを配備で想定されるように設定します。

   特に、実際の運用ディレクトリに対して実行するようにデータベースをインデックスします。あとでインデックスを追加する可能性がある場合は、追加するインデックスのスペースも追加する必要があります。

3. 各エントリセットを読み込み、各セットで使用されるディスク容量を記録します。
4. 結果をグラフにし、配備におけるサフィックスサイズの見積もりを推定します。
5. エラーや変更のために使用されるディスク容量を追加します。

レプリケーションを使用する場合は、エントリの状態情報 (古い値のリスト) はそのエントリとともに保管され、競合解決の際に使用されることに注意してください。状態情報はエントリのサイズを大幅に増加させることがあるため、サフィックスをサイジングするときに考慮する必要があります。

サフィックスのディスク容量は、全体の一部にすぎません。同様に、Directory Server のディスク使用状況についても検討する必要があります。

Directory Server の使用するディスク

ディレクトリのサフィックスは、Directory Server によってディスクに格納されるものの一部です。ディスク使用に影響するその他の多くの要因は、配備後の Directory Server の使用状況によっても大きく異なります。そのため、ここでは一般的な状況について説明します。ここで説明する項目を設定する手順は、『Directory Server 管理ガイド』を参照してください。

Directory Server バイナリ

このバージョンの Directory Server をインストールするには、約 200M バイトのディスク容量が必要です。ただし、この見積もりにはデータやログに対する容量は含まれていません。含まれているのは製品のバイナリに対する容量だけです。

イベントログ

ログファイルのディスク使用の見積もりは、Directory Server 動作の割合、ログのタイプとレベル、およびログローテーションの方法によって変わります。

多くのログ要件は予測でき、事前に計画をたてることができます。Directory Server でログや特に監査ログに書き込みが行われる場合、負荷レベルとともにディスクの使用状況が増大します。高負荷の配備で大規模なログを必要とするときは、高負荷に適応するように、ディスク容量を追加することを計画します。高負荷のログで配備に必要なディスク容量を減らすには、インテリジェントログローテーションおよび保管システムを構築し、頻繁にログをローテーションします。そして古いファイルはテープや、より安価なディスククラスタなど費用がかからず大容量のストレージメディアに自動的に移行させます。

ログ要件は簡単に予測できないこともあります。たとえばログをデバッグすると、一時的ですが急激に `errors` ログが増大します。大規模で高負荷の配備では、一時的で大量のデバッグログ専用に、数 G バイトのディスク容量を別に設定することを検討します。詳細は、『Directory Server Performance Tuning Guide』の「Tuning Logging」を参照してください。

トランザクションログ

トランザクションログの量は、ピーク時の書き込み負荷によって変わります。書き込みが急激に起こる場合は、書き込み負荷が一定である場合と比べて、トランザクションログで使用する容量が多くなります。Directory Server では、トランザクションログを定期的に削除しています。そのため、トランザクションログはチェックなしで増大し続けることはありません。

コピー中は、データベースファイルは変更できないため、オンラインバックアップ中は、トランザクションログはフラッシュされません。これにより、データイメージの整合性がなくなることがあります。トランザクションログは、バックアップの最後の手順としてバックアップ位置にコピーされます。

Directory Server では、一般的に永続トランザクションが有効な状態で実行されています。永続トランザクション機能が有効であると、Directory Server では、変更操作 (`add`、`delete`、`modify`、および `modrdn`) ごとにトランザクションログへ同期書き込みを行います。この場合、ディスクがビジーの場合は操作がブロックされ、潜在的な I/O ボトルネックとなります。

更新パフォーマンスが重要な場合は、高速な書き込みキャッシュを持つディスクサブシステムをトランザクションログ用に使用することを計画します。詳細は、『Directory Server Performance Tuning Guide』の「Tuning Logging」を参照してください。

レプリケーションの更新履歴ログデータベース

Directory Server 配備時にレプリケーションを行う場合は、更新履歴ログを記録します。更新履歴ログのサイズは、変更される量と、使用された更新履歴ログのタイプによって異なります。変更履歴ログの削除方法に応じて、容量を計画します。大規模で高負荷の配備では、変更率が異常に高いときの変更履歴ログの増加を処理するために、数 G バイトのディスク容量を別に設定することを検討します。詳細は、『Directory Server Performance Tuning Guide』の「Tuning Logging」を参照してください。

サフィックスの初期化と LDIF ファイル

サフィックスの初期化は一括ロードまたは一括インポートとも呼ばれます。この最中、Directory Server で必要となるディスク容量は、サフィックスデータベースファイルとサフィックスの初期化に使用される LDIF ファイルの分だけではなく、初期化プロセス中に使用される中間ファイルの分も必要となります。サフィックスの初期化中に使用される LDIF ファイルと中間ファイル用に、追加の容量をデータベースファイルと同じディレクトリに一時的に確保することを計画します。これは、作成済みのインデックスに応じて、最大のインデックスのサイズの 2 倍になることがあります。

バックアップと LDIF ファイル

バックアップでディスク容量を大量に消費することは珍しくありません。バックアップのサイズは、関連するデータベースファイルのサイズ、およびトランザクションログと等しくなります。データベースファイルの量を数倍した容量を割り当てることで、複数のバックアップに対応します。これにより、データベースとそれに対応するバックアップが別々のディスクで維持されます。時間が経過するにつれてバックアップを安価なストレージメディアに移行させるには、優れた方法を使用します。

配備時にレプリケーションを行う場合、初期化用の LDIF ファイルを保持するためのディスク容量を追加することを計画します。これは、初期化用の LDIF ファイルはバックアップ用の LDIF ファイルと異なるためです。

ディスクベースでなくメモリベースのファイルシステム

一部のシステムでは、メモリベースの `tmpfs` ファイルシステムをサポートしています。たとえば Solaris 上では、`/tmp` は、パフォーマンスを向上させるために、メモリベースのファイルシステムとしてマウントされます。

メモリベースのファイルシステムには、データベースキャッシュファイルのみを置く必要があります。詳細は、『Directory Server Administration Reference』の「nsslapd-db-home-directory」を参照してください。メモリベースのファイルシステムには、データベースまたはトランザクションログのバイナリや設定ファイルを置かないでください。

システムのほかのアプリケーションと共用されている /tmp にキャッシュファイルが置かれている場合は、/tmp 下で容量不足が起こらないように注意する必要があります。このようにしないと、メモリ量が低下し、メモリベースのファイルシステム内の Directory Server ファイルが、スワップパーティション専用のディスク容量にページングされる可能性があります。

一部のシステムは、RAM ディスクやその他のメモリベースファイルシステムをサポートしています。メモリベースのファイルシステムを作成および管理する手順については、オペレーティングシステムのマニュアルを参照してください。そのようなファイルシステム内のデータはすべて揮発性であり、システムリブート後にそれらのデータをメモリ内に再読み込みする必要があることに注意してください。この最初期化は、プロセッサの速度、メモリの速度、およびメモリのサイズのような要因に応じて、時間がかかることがあります。

core ファイル

少なくとも core ファイル 1 ～ 2 個分の空きスペースを残しておきます。Directory Server ではコアダンプするべきではありませんが、クラッシュ中に生成された core ファイルを調査に使用できる場合は、クラッシュ後の回復とトラブルシューティングがきわめて簡単になる場合があります。core ファイルは生成されると、cn=config の場合に nsslapd-errorlog で指定したファイルと同じディレクトリに格納されます。起動中にクラッシュした場合は *ServerRoot*/bin/slapd/server/ に格納されます。

管理用のディスク容量

システムと Directory Server の管理など、予期されるシステムの使用に対して空き容量を確保します。基本となる Directory Server インストール、ローカルインスタンス上に置かれたときの設定サフィックス、設定ファイルなどに十分なディスク容量を確実に割り当てます。

ディスク間のファイルの分散

当然のように更新される Directory Server データベースやログファイルを別のディスクサブシステムに配置することで、複数のディスクスピンドルやコントローラ間に I/O トラフィックの範囲を広げることができ、I/O ボトルネックを回避できます。次の各項目に対して、専用のディスクサブシステムを用意することを検討します。

トランザクションログ

永続トランザクション機能が有効であると、Directory Server では、変更操作ごとにトランザクションログへ同期書き込みを行います。そのため、ディスクがビジーであると、操作がブロックされてしまいます。トランザクションログを専用ディスクに配置することで書き込みパフォーマンスが向上し、Directory Server で処理可能な変更率が増加します。

『Directory Server Performance Tuning Guide』の「Transaction Logging」を参照してください。

データベース

複数のデータベースをサポートすることで、各データベースをそれ専用の物理ディスクに置くことが可能になります。そのため、個々のデータベースが専用のディスクサブシステム上にある場合、Directory Server の負荷を複数のデータベース間に分散させることができます。データベース操作の I/O 競合を防ぐには、データベースファイルの各セットを個別のディスクサブシステムに配置することを検討します。

最高のパフォーマンスを得るには、データベースファイルを大きい I/O バッファを持つ専用高速ディスクサブシステムに配置します。キャッシュに候補エントリが見つからない場合、Directory Server はディスクからデータを読み込みます。そして、定期的に書き込みをフラッシュします。これらの操作で専用の高速ディスクサブシステムを使用すると、潜在的な I/O ボトルネックが軽減されます。

`cn=config,cn=ldbm database,cn=plugins,cn=config` における `nsslapd-directory` 属性では、Directory Server がインデックスファイルを含むデータベースファイルを格納するディスク位置を指定します。これらのファイルはデフォルトで、*ServerRoot*`/slapd-`*ServerID*`/db/` にあります。

当然のことながら、データベース位置の変更では、Directory Server の再起動だけでなく、データベースの完全な再構築も必要となります。実際の運用サーバーでデータベース位置を変更することは、大掛かりな作業になる場合があるため、もっとも重要なデータベースを特定し、それを別のディスク上に配置したあとで、サーバーの運用を開始してください。

ログファイル

Directory Server には、アクセス、エラー、および監査の各ログが用意されており、バッファのあるログ能力を備えています。バッファがあるにもかかわらず、これらのログファイルに書き込みをするには、ほかの I/O 操作と競合する可能性のあるディスクアクセスが必要です。パフォーマンス、容量、および管理を改善するには、ログファイルを別々のディスクに配置することを検討します。

詳細は、『Directory Server Performance Tuning Guide』の「Tuning Logging」を参照してください。

メモリベースファイルシステム上のキャッシュファイル

tmpfs ファイルシステムでは、たとえば、物理メモリが使い果たされたときだけファイルがディスクにスワップされます。すべてのキャッシュファイルを物理メモリに保持するのに十分なメモリがあれば、tmpfs ファイルシステム (Solaris プラットフォーム)、または RAM ディスクなどのその他のメモリベースファイルシステム (その他のプラットフォーム) に同じサイズのディスク容量を割り当て、Directory Server がそのファイルシステムにキャッシュファイルを格納するように nsslapd-db-home-directory の値を設定することで、パフォーマンスの向上が得られます。これにより、システムでは、キャッシュファイルにマッピングされたメモリが、不必要にディスクへフラッシュしなくなります。

ディスクサブシステムの選択

「*Fast, cheap, safe: pick any two.* (速度、低コスト、安全性 : いずれか 2 つを選択)」 ― 『Sun Performance and Tuning』、Cockroft と Pettit 著

速度と安全性

パフォーマンスと稼動時間の両方が重要な配備を実装するときは、大規模のディスクアレイ間に分散した I/O を高速でバッファできるように、不揮発性のメモリキャッシュを備えているハードウェアベースの RAID コントローラを検討します。負荷を多くのディスクで分散させ、高速接続でアクセスをバッファすることで、I/O は最適化され、高パフォーマンスの RAID ストライピングやパリティブロックによってきわめて安定します。

Sun StorEdge™ 製品で提供されているような大容量で不揮発性 I/O バッファや高パフォーマンスのディスクサブシステムによって、Directory Server のパフォーマンスや稼動時間は大きく向上します。

高速な書き込みキャッシュカードでは、特にデータベースやトランザクションログ専用としたときに、書き込みパフォーマンスが向上する可能性があります。高速な書き込みキャッシュカードには、ディスクコントローラから独立した不揮発性のメモリキャッシュが備わっています。

速度と低コスト

高速で低コストのパフォーマンスを得るには、十分なディスク容量が複数ディスク間に分散されていることを確認します。回転速度が速く、シークタイムが短いディスクの使用を検討します。最高の結果を得るには、分散されたコンポーネントそれぞれに対して 1 つのディスクを専用にします。シングルポイントのエラーを避けるには、マルチマスターレプリケーションを使用することを検討します。

低コストと安全性

安価で安全な設定には、Solaris Volume Manager のように低コストでソフトウェアベースの RAID コントローラの使用を検討します。

RAID の選択

RAID とは Redundant Array of Inexpensive Disks (安価なディスクを複数使用した冗長アレイ) のことです。名前が示すとおり、RAID の第 1 の目的は回復力を備えることです。アレイ内のディスクで障害が発生しても、そのディスク上のデータは失われずに、アレイ内のほかのディスクで依然として利用できます。回復力を実装するために、RAID では抽象概念を持ち込み、通常は単一のボリュームとして参照される大容量の仮想ディスクとして、複数のディスクドライブを設定することができます。これは、物理ディスクを連結、ミラーリング、またはストライピングすることで実現されます。連結の実装では、あるディスクのブロックの後に別のディスクのブロックを論理的に続けます。たとえば、ディスク 1 にはブロック 0 ～ 99、ディスク 2 にはブロック 100 ～ 199、というようになります。ミラーリングの実装では、あるディスクのブロックを別のディスクにコピーし、その後は同期し続けます。ストライピングでは、複数の物理ディスクに仮想ディスクのブロックを分散させるアルゴリズムを使用します。

ストライピングの目的はパフォーマンスを得ることです。ランダム書き込みでは、書き込まれるデータがストライピングボリュームの複数のディスクに送信される可能性があるため、非常に迅速に処理されます。したがって、ディスクは並列で動作することが可能です。ランダム読み込みにも同じことが言えます。大量のシーケンシャルな読み込みと書き込みの場合は、それほど単純ではありません。しかし、シーケンシャルな I/O パフォーマンスも向上することが確認されています。たとえば、多くの I/O 要求を生成するアプリケーションでは、単一のディスクコントローラに集中することがあります。ストライピングボリュームのディスクすべてで専用のコントローラを使用している場合、このような集中は起こりにくく、パフォーマンスが向上します。

ソフトウェアとハードウェアのどちらの RAID 管理デバイスでも、RAID を実装することができます。それぞれの方法には利点と欠点があります。

- 一般にハードウェア RAID では、ハードウェアに実装されているため、高パフォーマンスが得られます。それにより、ソフトウェア RAID に比べて処理のオーバーヘッドが少なくて済みます。さらに、ハードウェア RAID はホストシステムから分離されているため、アプリケーションを実行するためのホストリソースを消費しないで済みます。
- 一般にハードウェア RAID は、ソフトウェア RAID に比べて高価です。
- ソフトウェア RAID は、ハードウェア RAID に比べて柔軟性が高くなっています。たとえば、通常、ハードウェア RAID マネージャは単一のディスクアレイや指定されたディスクアレイのセットと関連付けられています。一方、ソフトウェア RAID では、任意の個数のディスクアレイや、必要ならばアレイ内の特定のディスクだけをカプセル化できます。

次の節では、レベルとして知られている RAID 設定について説明します。もっともよく使用される RAID レベルは 0、1、1+0、および 5 であり、これらについて詳細を説明しますが、あまり使用されないレベルについては比較と対照だけの対象とします。

RAID 0、ストライピングボリューム

ストライピングでは、データを複数の物理ディスクに分散します。論理ディスクや論理ボリュームは、チャンク (まとまり) やストライプ (しま状) に分割され、ラウンドロビン方式で物理ディスク上に分散されます。ストライプは常に 1 つ以上のディスクブロックから成り、すべてのストライプは同じサイズになります。

RAID 0 という名前は、冗長性がないことと矛盾しています。RAID 0 のストライプでディスク障害が発生すると、論理ボリューム全体が失われてしまいます。ただし、RAID 0 はすべての RAID レベルの中でもっとも安価であり、すべてのディスクをデータ専用にします。

RAID 1、ミラーボリューム

ミラーの目的は冗長性を実現することです。ミラー内のディスクの 1 つがエラーになっても、データは利用可能なままであり、処理を継続することができます。各物理ディスクがミラーされるため、物理ディスク容量の半分がミラー用に当てられてしまうのが欠点です。

RAID 1+0

RAID 10 とも呼ばれます。RAID 1+0 では、高レベルのパフォーマンスと回復力を備えています。その結果、RAID レベルの中では実装にもっとも費用がかかります。障害を起こしたすべてのディスクで別のミラーを作っている限り、最高で 3 ディスクが障害を起こしてもデータは利用可能なままです。RAID 1+0 はセグメントが RAID 1 であるストライピングアレイとして実装されます。

RAID 0+1

RAID 0+1 は RAID 1+0 に比べると回復力の点で劣っています。ストライプは作成されるとミラーされます。ミラーの同じ側にある 1 つ以上のディスクで障害が起きても、データは利用可能なままです。ただし、その後でミラーの別の側にあるディスクが障害を起こすと、論理ボリュームは失われます。RAID 1+0 とのわずかな違いは、RAID 1+0 では、どちらの側のディスクが同時に障害を起こしても、データが利用可能なままであるという点です。RAID 0+1 はセグメントが RAID 0 であるミラーアレイとして実装されます。

RAID 5

RAID 5 にはミラーリングのような回復力はありませんが、それにもかかわらず、単一ディスクが障害を起こしてもデータは利用可能なままであるという、冗長性を備えています。RAID 5 では冗長性を実現するために、論理エクスクルーシブを実行して作成されたパリティストライプや、ほかのディスク上で対応するストライプのバイトにあ

るパリティストライプを使用します。1 つのディスクが障害を起こすと、そのディスクのデータは、残りのディスクにある対応するストライプのデータとパリティを使用して、もう一度計算されます。ただし、このような補正計算を実行する必要がある場合は、パフォーマンスに影響します。

通常の運用では、RAID 5 は RAID 0、1+0、および 0+1 に比べてパフォーマンスが低いのが普通です。RAID 5 ボリュームでは論理書き込みのたびに 4 回の物理 I/O 操作を実行する必要があります。古いデータとパリティが読み込まれ、エクスクルーシブまたは操作が 2 回実行され、そして新しいデータとパリティが書き込まれます。読み込み操作では、同じ負荷はかからないため、同数のディスクを使用した標準的なストライプの場合と比べてもパフォーマンスがわずかに低下するだけです。つまり RAID 5 ボリュームでは、パリティ専用のディスク容量があるため、ストライプ内には事実上 1 つ少ないディスクがあることになります。RAID 5 では利用可能なディスク容量以上をデータに使用するため、RAID 5 ボリュームは RAID 1+0 や 0+1 と比べて一般的に安価であるということです。

パフォーマンスの問題から、データが読み取り専用でない限り、またはボリュームへの書き込みが非常に少ない場合を除いて、通常は RAID 5 をお勧めしません。ただし、書き込みキャッシュと高速なエクスクルーシブまたは論理エンジンを備えたディスクアレイでは、パフォーマンスの問題を緩和することができ、配備によっては RAID 5 が安価な、ミラーに代わる実現可能な代替策となります。

RAID レベル 2、3、および 4

RAID レベル 2 および 3 は、ビデオストリーミングのように、大量のデータがシーケンシャルに転送される場合に適しています。どちらのレベルでも、一度に I/O 操作は 1 回だけ処理可能で、ランダムアクセスを要求するアプリケーションには向いていません。RAID 2 はハミングエラー訂正コーディング (ECC) を使用して実装されます。3 つの物理ディスクドライブに ECC データを格納する必要があり、RAID 5 よりも高価ですが、ストライプが 3 つより多くのディスクで構成されている場合は RAID 1+0 より安価になります。RAID 3 ではビットワイズパリティ方法を使用して冗長性を実現しています。パリティは、RAID 5 のようには分散されませんが、単一の専用ディスクに書き込まれます。

RAID 2 や RAID 3 とは異なり、RAID 4 では複数のディスクドライブに同時にアクセス可能な独立アクセス技術を使用しています。RAID 5 に似た方法でパリティを使用しますが、パリティが単一ディスクに書き込まれる点が異なります。そのため、書き込みごとにパリティディスクにアクセスされるため、パリティディスクがボトルネックとなり、事実上、複数の書き込みが直列処理されています。

ソフトウェアのボリュームマネージャ

Solaris™ Volume Manager のようなボリュームマネージャは、Directory Server のディスク管理でも使用できます。Solaris Volume Manager は、実際の運用環境への配備において、その他のソフトウェアのボリュームマネージャと好意的に比較されています。

I/O およびディスク使用の監視

通常の運用環境で、ディスクをいっぱいにしないでください。潜在的な I/O ボトルネックを分離するには、Solaris システムやその他のシステムにおける `iostat` (1M) などのユーティリティを使用することもできます。Windows システムの I/O ボトルネックへの対応方法の詳細は、Windows ヘルプを参照してください。

マルチプロセッサシステムのサイジング

Directory Server ソフトウェアは、複数のプロセッサ間でスケールされるように最適化されています。一般に、プロセッサを追加すると、全体的な検索、インデックスの保守、およびレプリケーションパフォーマンスが向上します。

しかし、特定のディレクトリの配備では、プロセッサを追加してもパフォーマンスが大幅に改善しない効果がなくなる点に達していることがあります。検索、インデックス、およびレプリケーションで多大に要求されるパフォーマンス要件を扱うときは、解決策の一部として、ロードバランスやディレクトリプロキシなどの技術を検討します。

ネットワークキャパシティのサイジング

Directory Server はネットワークを集中利用するアプリケーションです。Directory Server インスタンスでネットワークの可用性を向上させるには、システムにネットワークインタフェースを 2 つ以上搭載します。Directory Server ではそのようなハードウェア設定をサポートしており、同一プロセス内の複数のネットワークインタフェースで待機します。

ロードバランスのために、同じネットワーク上で Directory Server をクラスタにする場合は、新たに生じる負荷がネットワークインフラストラクチャで対応できることを確認します。更新率の高いレプリケーションを WAN 環境でサポートする場合は、経験テストを通じて、ネットワーク品質と帯域幅がレプリケーションのスループット要件を満たすことを確認します。

SSL 用のサイジング

ソフトウェアには、Secure Sockets Layer (SSL) プロトコルがデフォルトで実装されています。ソフトウェアベースの SSL 実装を使用すると、Directory Server パフォーマンスに重大な悪影響を及ぼすことがあります。SSL モードでディレクトリを実行するには、全体のパフォーマンス要件を満たすために、複数のディレクトリのレプリカの配備が必要な場合があります。

ハードウェアアクセラレータカードでは SSL を使用する影響を取り除くことはできませんが、ソフトウェアベースの実装の場合と比べてパフォーマンスが大幅に改善されます。Directory Server 5.2 では、サポートされている Sun Crypto Accelerator ハードウェアのような SSL ハードウェアアクセラレータを使用できます。

Sun Crypto Accelerator ボードの使用は、SSL 鍵計算がボトルネックになっている場合に有効です。ただし、SSL 鍵計算がボトルネックでない場合には、そのようなハードウェアを使用しても、パフォーマンスの改善は望めません。ハードウェアが実際に加速するのは、接続確立時の SSL ハンドシェイク中に行われる鍵計算の処理であり、接続確立後のメッセージの暗号化や復号化の処理ではないからです。このようなハードウェアを Directory Server インスタンスで使用する手順については、『Directory Server 管理ガイド』の「Sun Crypto Accelerator ボードの使用」を参照してください。

用語集

このマニュアルで使用される用語の完全なリストについては、『Java Enterprise System 用語集』(http://docs.sun.com/doc/819-1933?l=ja) を参照してください。

索引

え

お

か

く

け

こ

さ

し

す

せ

そ

た

て

と

に

ね

は

ひ

ふ

ほ

ま

ゆ

り

る

れ

ろ